VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SJEDNOCENÍ A ZPŘÍSTUPNĚNÍ INFORMACÍ A OVLÁDÁNÍ ARÉN LASERGAME

UNIFIED AND AVAILABLE INFORMATION AND CONTROLS OF LASERGAME ARENAS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ VOJÍK**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Vojík Tomáš**
Program: Informační technologie
Název: **Sjednocení a zpřístupnění informací a ovládání arén LaserGame**
Unified and Available Information and Controls of LaserGame Arenas
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s problematikou návrhu a vývoje webových aplikací.
2. Analyzujte a popište systémy pro ovládání LaserGame a zobrazování výsledků v těchto hrách.
3. Experimentujte s postupy a algoritmy, které umožní napojení vlastní aplikace na jednotlivé systémy ovládání LaserGame.
4. Navrhněte aplikaci, která umožní zobrazovat výsledky a ovládat vybrané aspekty LaserGame - popište případy užití, způsoby práce s daty, prvky uživatelského rozhraní a uživatelské zkušenosti, atp.
5. Implementujte navrženou aplikaci.
6. Testujte propojení vytvořené aplikace s relevantními systémy pro ovládání LaserGame a iterativně je vylepšujte.
7. Testujte vytvořenou aplikaci s uživateli a iterativně ji vylepšujte.
8. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování výsledků projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodů 4. a 5.

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

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

V práci se zabývám tvorbou systému pro hráče a provozovatele rozrůstajícího se sportu LaserGame. Hlavní motivací pro tvorbu takového systému je vytvoření společné komunity napříč různými arénami, nehledě na výrobce jejich LaserGame vybavení. Hotová aplikace bude pracovat s výsledky různých systémů, poskytne detailnější statistiky a nástroje pro arény, které doplní možnosti systému výrobce.

Jako řešení byla vytvořena webová aplikace, která je distribuovaná v Docker kontejnerech v jednotlivých arénách. Jednotlivé aplikace navíc komunikují pomocí REST API s jedním centrálním serverem, který výsledky shromažďuje.

Vytvořený systém již splňuje všechny základní požadavky pro systém Evo5 výrobce LaserMaxx. Zpracovává výsledky, prezentuje je v jednoduché, vizuálně atraktivní formě hráčům a synchronizuje výsledky na veřejný portál. Ze strany hráčů jsou na změny velmi pozitivní ohlasy.

Abstract

In this work, I deal with creating a system for players and operators of the growing sport LaserGame. The primary motivation for creating such a system is creating a shared community across different arenas, regardless of the manufacturer of their LaserGame equipment. The finished application will work with the scores of various systems and provide more detailed statistics and tools for arenas that extend the capabilities of the manufacturer's design.

A web application was created and distributed in Docker containers in individual arenas as a solution. In addition, individual applications communicate using a REST API with one central server that collects the results.

The created system already meets all the basic requirements for the Evo5 system manufactured by LaserMaxx. It processes the results, presents them in a simple, visually attractive form to the players and synchronises the results on a public portal. There is a very positive response from the players to the changes.

Klíčová slova

webová aplikace, LaserGame, uživatelská rozhraní, sportovní informační systém, prezentace výsledků, responsivní design, Docker

Keywords

web application, LaserGame, user interfaces, sports information system, presentation of scores, responsive design, Docker

Citace

VOJÍK, Tomáš. *Sjednocení a zpřístupnění informací a ovládání arén LaserGame*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Sjednocení a zpřístupnění informací a ovládání arén LaserGame

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta Ph.D. Další informace mi poskytli: Laser aréna Písek a Lasergame Říčany. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Tomáš Vojík
9. května 2022

Poděkování

Rád bych poděkoval všem, kdo mi poskytli pomoc při tvorbě této práce. Převážně svojí ženě Sofiji, která byla mým hlavním testerem a tolerovala mou neustálou potřebu povídat si o tomto projektu. Jako další bych rád poděkoval mému vedoucímu prof. Ing. Adamu Heroutovi Ph.D. za jeho konzultace. V poslední řadě děkuji všem hráčům české LaserGame komunity, kteří se mnou konzultovali mé myšlenky, Laser aréně Písek za poskytnutí prostředků pro vývoj aplikace a Lasergame Říčany za poskytnutí informací o systému Laserforce.

Obsah

Kapitola 1

Úvod

Tato práce je jedním z prvních kroků mé dlouholeté vize. Vytvoření vlastního dokonalého systému, který doplní a vylepší funkce nedokonalých systémů výrobců vybavení LaserGame. V České a Slovenské republice je v době psaní této práce 79 LaserGame arén[1], kdy část z nich řeší různé problémy a snaží se všemožnými způsoby obejít, či nějak nahradit funkce, které jim nevyhovují, nebo chybí[2].Vzniká tím spleť různých dalších nástaveb. Většina těchto systémů pracuje hlavně s výsledky ze hry, které se snaží nějak vizuálně vylepšovat, případně doplňovat o nové detaily. Žádný z těchto systémů nespolupracuje napříč arénami, ale fungují jen v jedné, ačkoliv třeba jednu nástavbu využívá arén několik.

Mým dlouholetým snem je vznik celorepublikové komunity praktikující tento novodobý sport. Pořádají se celorepublikové soutěže a využívá se jedné univerzální aplikace. Vize je to ambiciózní a její naplnění bude trvat ještě možná několik let.

V první řadě se musím hlavně zaměřit na fungování systému, který chci využít. Mou největší výhodou je přímý přístup k laser aréně v Písku[3], ale i aktivní kontakt s dalšími majiteli a hráči, díky čemuž mohu vyvíjet a testovat na reálných datech a systémech. Právě i díky tomu jsem se rozhodl celou aplikaci nejprve postavit a otestovat na LaserGame vybavení Evo5 výrobce LaserMaxx[4].

V době psaní této práce je aplikace aktivně využívaná ve dvou arénách s dalšími dvěma potenciálními zájemci.

Jako důležité také považuji zmínit, že výsledná aplikace nemá sloužit ke kompletnímu nahrazení software, který poskytuje výrobce systému. Nabízí pouze dodatečné nástroje, které nejsou nutné pro fungování LaserGame arén, ale usnadňují některé procesy, které budou popsané v kapitole 3 – sekce 3.4. Velký důraz je však kladen právě na vytváření statistik a detailních výsledků pro hráče samotné.

Jako první si v kapitole 2 srovnáme software nejrozšířenějších výrobců. Shrneme jejich výhody, nevýhody a funkce, které by bylo dobré zachovat i v hotové aplikaci. Na základě těchto získaných informací vytvoříme v kapitole 3 detailní návrh našeho řešení, které budeme implementovat. Implementační detaily a problémy probereme v kapitole 4. Nakonec v kapitole 5 analyzujeme výsledky testování hotové aplikace na reálných uživatelích.

[1] Komunitou vytvořená mapa arén: https://www.google.com/maps/d/u/0/embed?mid=1DWy-GC1AZv7SJNdbJvNQTlSwzxZ02WfX&ll=48.997718900693734%2C16.028413804923414&z=7

[2] Příkladem takového rozsáhlého počinu je projekt http://lasergameareny.cz, kteří prodávají hotové řešení pro nově vznikající arény.

[3] https://laserarenapisek.cz/

[4] https://www.lasermaxx.com/en/products/evo5

Kapitola 2

Analýza aktuálních systémů ovládání LaserGame

Tato kapitola se zabývá výpisem, zhodnocení a možnostmi jednotlivých systémů různých výrobců vybavení LaserGame. Zaměřil jsem se převážně na dva v Česku nejrozšířenější: LaserMaxx a Laserforce. Software ostatních výrobců většinou využívá podobné prvky. Pro účely práce se nebudu zabývat popisem konkrétních hardware systémů, jelikož nijak neovlivňuje výslednou aplikaci.

2.1 Nejrozšířenější systémy pro ovládání LaserGame v České republice

Typický software výrobce LaserGame vybavení slouží primárně ke dvěma základním účelům:

1. Vytvoření a ovládání hry
2. Výpis výsledků pro hráče

Některé moderní systémy umožňují i sdílení výsledků veřejně na webu. Tato možnost bývá často omezena pouze na základní statistiky.

Téměř všechny systémy navíc pracují na speciálním hardware (například jsou distribuované na předinstalovaném počítači) a neumožňují přenositelnost a ovládání z jiných zařízení.

2.1.1 LaserMaxx

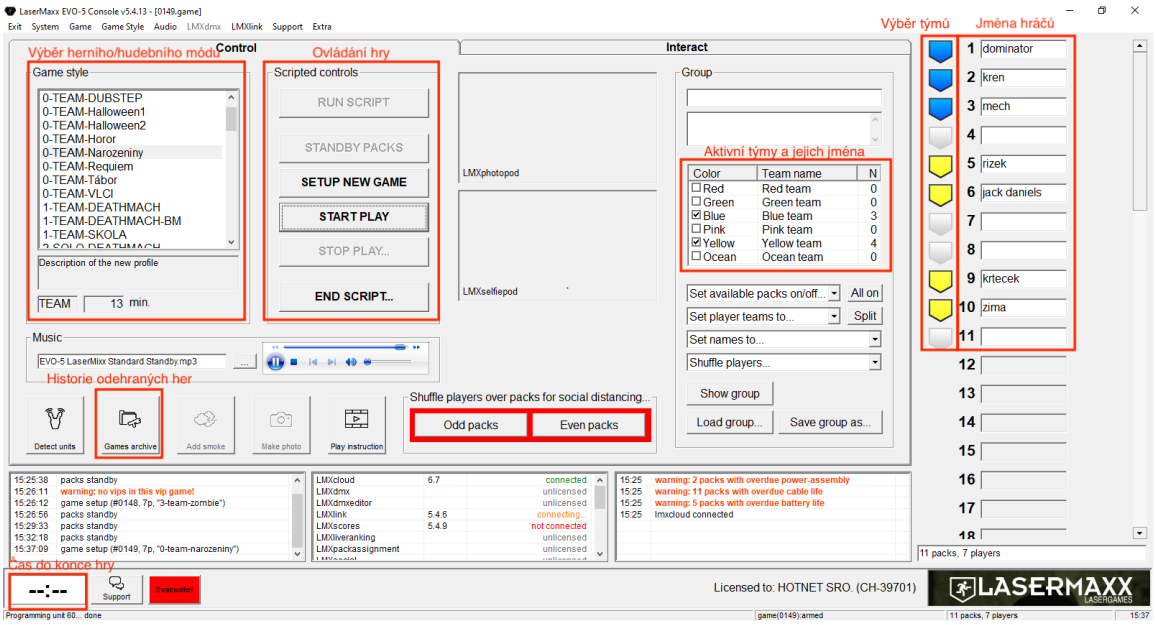
Aktuálně nejrozšířenější vybavení v České republice od nizozemského výrobce. V porovnání s ostatními systémy je poměrně jednoduchý a omezený, ale zároveň robustní.

Software systému komunikuje s vestami přes sériový port pomocí rádiové karty v počítači. Předává jen zprávy o nastavení hry, startu hry, předčasnému ukončení a žádost o stažení výsledků. Po stažení výsledků z vest je ukládá do textového souboru na disku, se kterým se dá dále pracovat.

Rozhraní aplikace

Samotná aplikace je poměrně jednoduchá, ale nenabízí vůbec českou lokalizaci. Operátor arény primárně manipuluje se třemi částmi rozhraní – viz. obrázek 2.1. V levé části se

nachází výběr herních módů, který ovlivňuje délku hry, hudební mód a další předem připravené nastavení. V pravé části píše operátor jména hráčů na konkrétní vesty a klikáním na barevné odznaky mění jednotlivým hráčům přiřazený tým. Přibližně uprostřed rozhraní se nachází tlačítka s ovládáním hry.



Obrázek 2.1: Rozhraní aplikace systému LaserMaxx

Výsledky ze hry

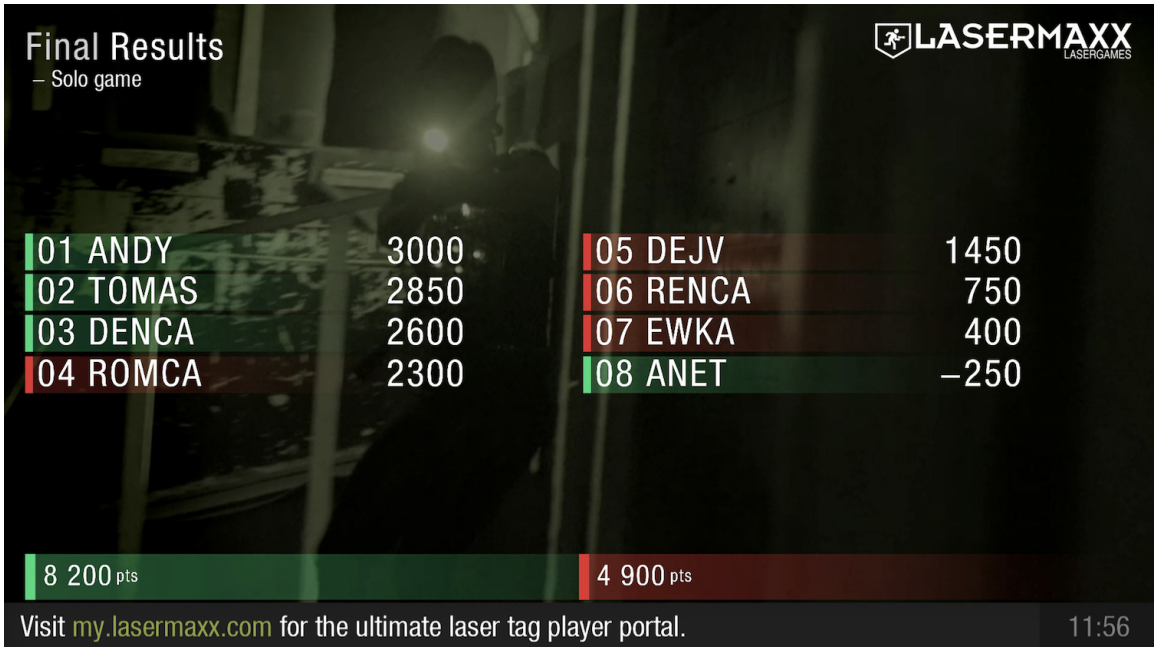
Jako první mají hráči přístup k výsledkové tabuli, která je většinou blízko vchodu do arény. Po hře se na tabuli zobrazují hráči v pořadí ve kterém se umístili, jejich skóre a ve spodní části skóre týmů – viz. obrázek 2.2. V čase mezi hrami se na tabuli zobrazují nejlepší hráči dne – viz. obrázek 2.3.

Výsledky ze hry se tisknou pro každého hráče zvlášť na papír velikosti A5 – viz. obrázek 2.4. Aréna má předtištěné karty s pozadím, tudíž se tisknou pouze černé texty. Šetří se tak inkoust, ale na druhou stranu výsledky nemohou vypadat rozdílně a obsahovat jiné statistiky pro různé herní módy. V horní a pravé části vidí hráč své osobní statistiky. V prostřední části se nachází výpis všech hráčů s jejich skóre, počtem zásahů a smrtí.

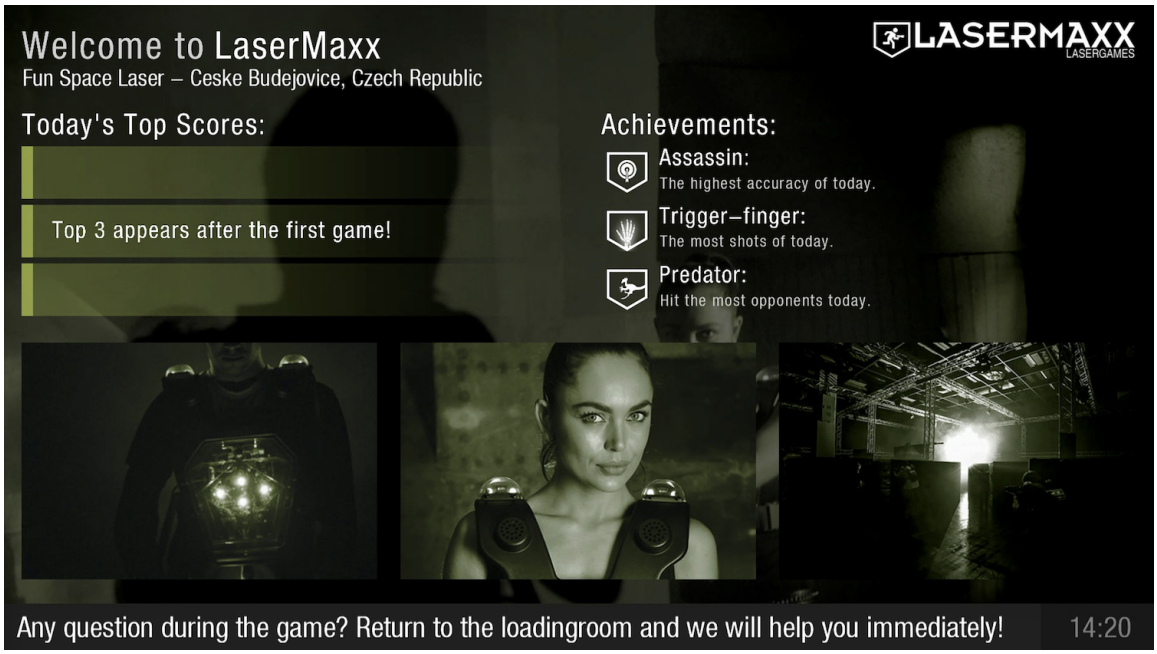
LaserMaxx nabízí i službu on-line výsledků, na které se hráč dostane skrze QR kód v pravém dolním rohu. Toto zobrazení nenabízí žádné další užitečné statistiky, ale zobrazí pouze vizuálně zajímavější přepis z papíru – viz. obrázek 2.5. Jak je zřejmé z obrázku 2.6, na mobilních zařízeních se tabulka hráčů nevejde na šířku obrazovky a působí nepřehledně. Po založení účtu a přihlášení nabízí systém přiřazení výsledků ke svému účtu a následné zobrazení souhrnných statistik jako: počet her, sumu nastříleného skóre a podobné.

2.1.2 Laserforce

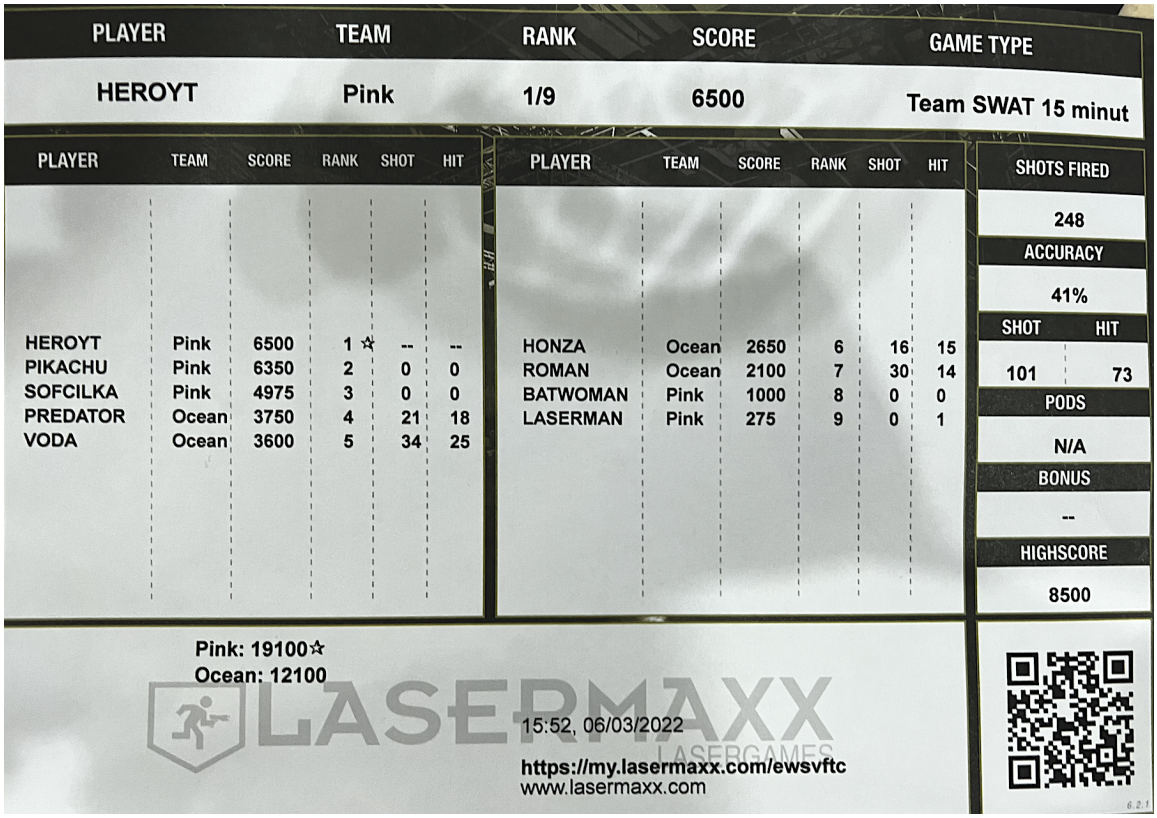
Vybavení novozélandského výrobce s rozsáhlou českou komunitou pravidelných hráčů. Systém nabízí velké množství herních nastavení, což oceňují pravidelní hráči, ale zároveň zvyšuje komplexitu celé aplikace.



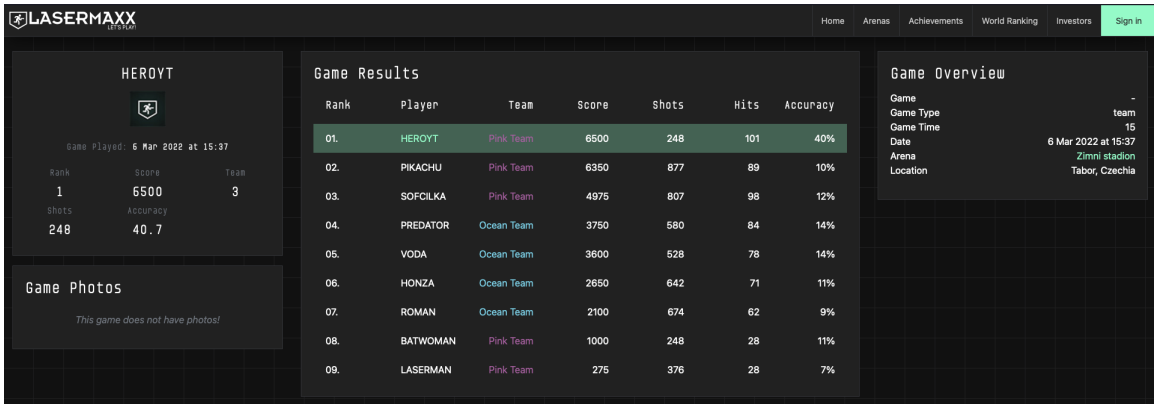
Obrázek 2.2: Výsledková tabule po hře LaserMaxx



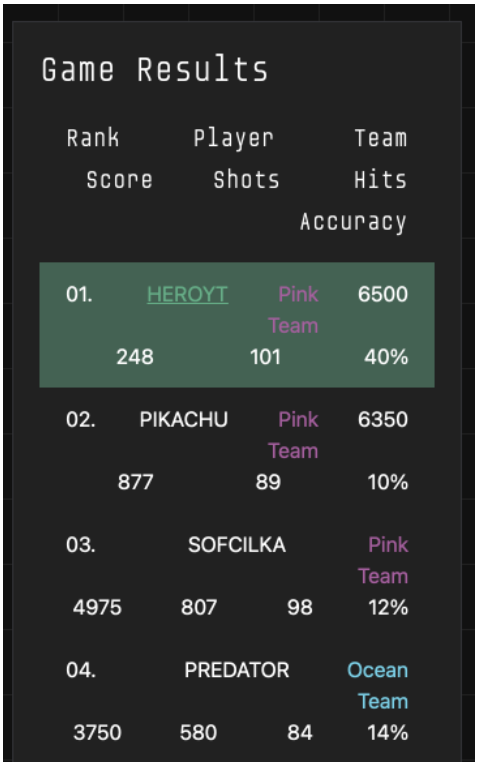
Obrázek 2.3: Výsledková tabule LaserMaxx – zobrazení nejlepších hráčů mezi hrami



Obrázek 2.4: Tisknutá výsledková karta pro systém LaserMaxx



Obrázek 2.5: On-line zobrazení výsledků pro systém LaserMaxx



Obrázek 2.6: On-line zobrazení výsledků pro systém LaserMaxx – na mobilním zařízení

Vesty a software komunikují po Wi-Fi. Celý systém je postavený na komunikaci v reálném čase a vesty nemůžou fungovat nezávisle bez řídícího počítače. Výsledky se ukládají do textových souborů jako přepis událostí, které nastali v čase a zároveň se ukládají celkové informace do interní `Microsoft SQL Server` databáze.

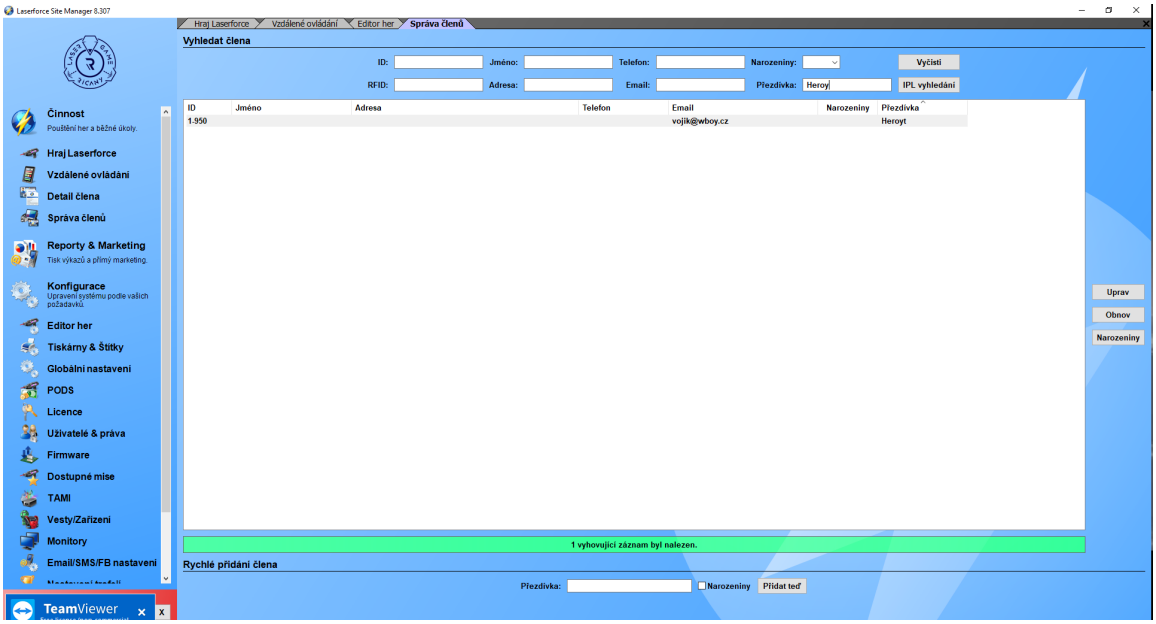
Rozhraní aplikace

Jak jsem již zmínil, aplikace je daleko komplexnější, nabízí velké množství detailních nastavení, nicméně operátor typicky pracuje pouze se třemi základními obrazovkami. Výhodou je také kompletní překlad do češtiny.

Jako první operátor vybírá z nabídky herních módů (obrázek 2.7). Následně se hráči přihlašují na vesty. Registrovaní členi mají vlastní `RFID` kartu, pomocí které vestu aktivují se svou herní přezdívkou. Neregistrovaní hráči si v některých arénách mohou vypůjčit náramkový čip, nebo operátor aktivuje jejich vestu v systému a hráč poté hraje s přednastaveným jménem. Rozdělení do týmů (obrázek 2.8) probíhá formou `drag and drop`. Z této samé obrazovky může operátor hru ovládat tlačítky v horní části. Třetí obrazovka, kterou může operátor využít je vyhledávání hráčů, kteří si svoji `RFID` kartu zapomněli (obrázek 2.9). Hráče může vyhledávat podle jeho unikátního čísla nebo pokud již v dané aréně někdy hrál podle přezdívky.

Obrázek 2.7: Výběr herního módu v systému Laserforce

Obrázek 2.8: Rozdělení hráčů do týmů v systému Laserforce



Obrázek 2.9: Vyhledávání registrovaných členů v systému Laserforce

Výsledky ze hry

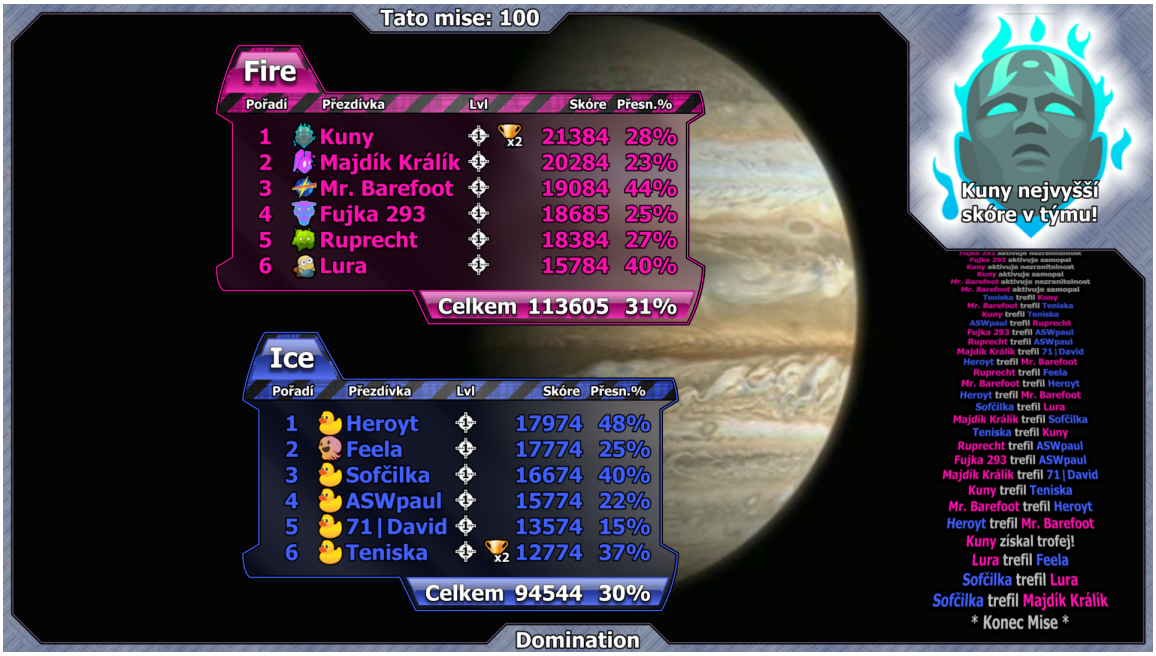
Výsledková tabule je jedna z hlavních zobrazení u systému Laserforce (obrázek 2.10). Výsledky se aktualizují v reálném čase. Na tabuli se nezobrazuje nic jiného, než aktuální/právě dokončená hra.

Každý hráč má své papírové výsledky ve formátu A5 (obrázek 2.11). Obsahují pouze základní informace: pořadí, skóre, zásahy, smrti...

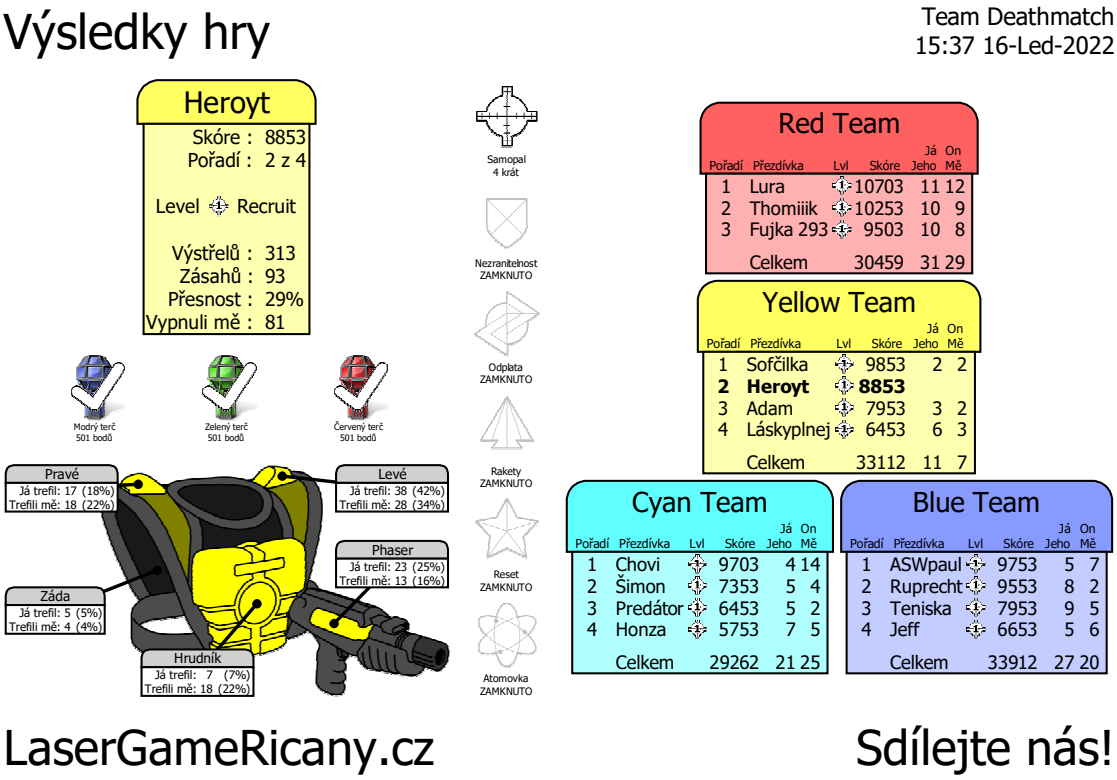
Laserforce také umožňuje zobrazovat výsledky on-line (obrázek 2.12). Tato možnost je ale dostupná jen pro registrované hráče a nabízí pouze výpis odehraných her, nikoliv detailní statistiky. Oblíbenou funkcí mezi hráči je získávání trofejí. Všechny Laserforce arény zároveň obsahují kiosek s podobným zobrazením a přístupem přes `RFID` kartu.

2.2 Porovnání software systémů LaserGame

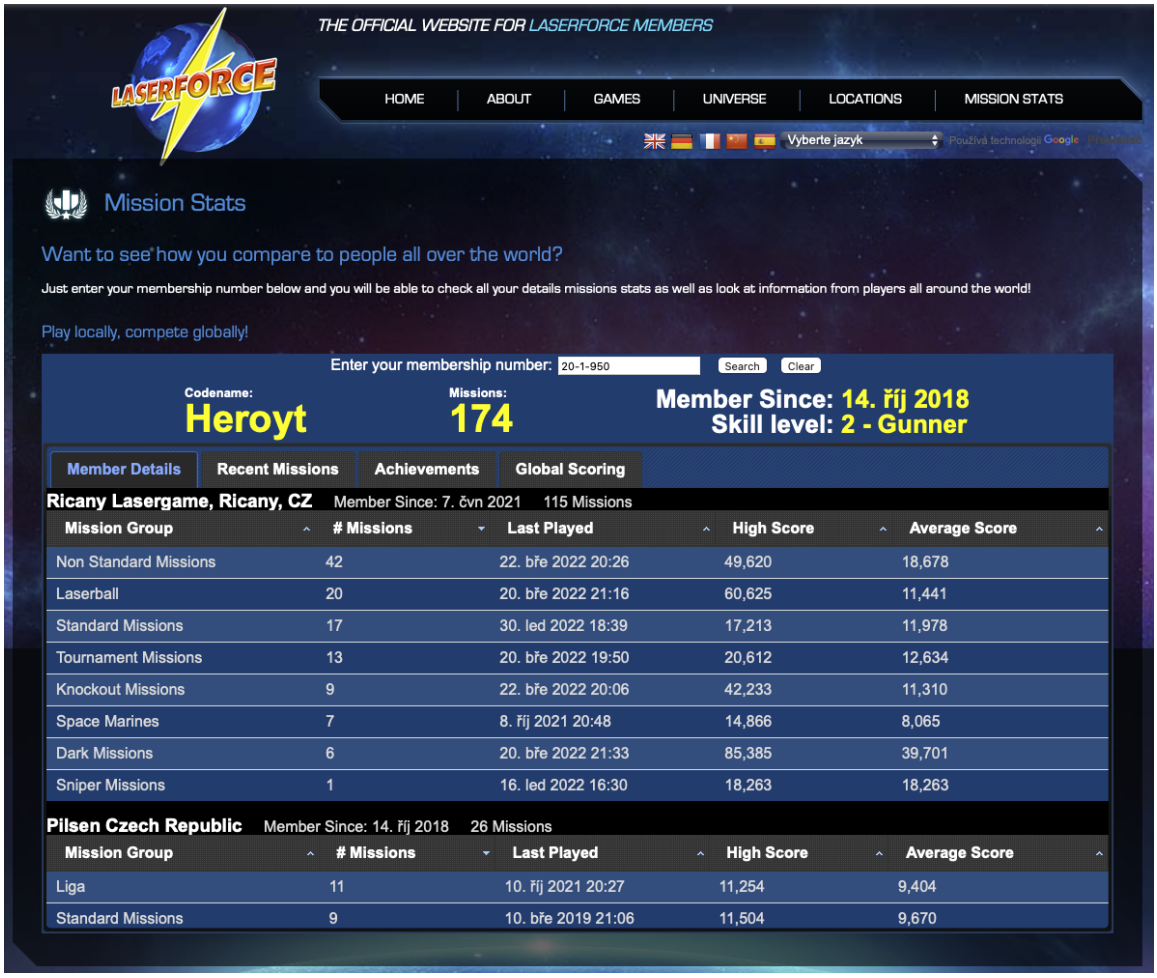
Tato část se zabývá konkrétními rozdíly mezi jednotlivými systémy a jejich srovnáním. Hodnocení vychází z předchozích sekcí 2.1.1 a 2.1.2. Shrnutí všech podstatných funkcí systému se nachází v tabulce 2.1.



Obrázek 2.10: Výsledková tabule pro systém Laserforce



Obrázek 2.11: Tisknutá výsledková karta pro systém Laserforce



Obrázek 2.12: On-line portál pro členy Laserforce

Funkce	LaserMaxx	Laserforce
Zadávání hráčů	Jména na konkrétní vesty	Přes `RFID` kartu, nebo bez možnosti vlastního jména
Výběr týmů	Výběr u každé vesty zvlášť	`Drag and drop`
Rozhraní aplikace	Jedna obrazovka s důležitým nastavením pro hru	Více obrazovek s postupným nastavováním hry
Tisknuté výsledky	A5, Pro každého hráče, pouze základní informace	A5, Pro každého hráče, pouze základní informace
Výsledková tabule	Pořadí a skóre	Pořadí, skóre, přesnost, případně další informace podle herního módu
On-line výsledky	Přepis papírových výsledků	Seznam odehraných her bez statistik

Tabulka 2.1: Shrnutí funkcí systémů LaserMaxx a LaserForce

Oba systémy jsou postaveny jiným způsobem. Každý systém má své výhody i nevýhody. Jako největší výhodu systému LaserMaxx považuji jeho přímočarost a jednoduchost celého programu. Jelikož nabízí jen omezené množství funkcí, obsluze arény stačí chápat, kde napíše jména hráčů a jak spustí hru. Naopak výhodou Laserforce jsou jeho pokročilé možnosti. Ačkoliv operátor musí znát daleko více informací o konkrétních herních módech a nastaveních, celý proces usnadňuje fakt, že celá aplikace je v češtině.

Ve které oblasti naopak dle mého názoru strádají oba systémy, jsou detailní výsledky z her. Všechny tři formy zobrazení výsledků nabízí pouze základní informace. Tištěné výsledky umožňují poskytnout hráči jen jeho osobní statistiky. Pokud by hráč chtěl získat přehled i o svých spoluhráčích, musí zkoumat další karty. Výsledky Laserforce jsou navíc zaměřené na týmy. Nepřehledné je i pořadí hráčů ve hře. Chce-li hráč vědět jeho individuální umístění, musí si pořadí sám spočítat. On-line výsledky neposkytují žádné větší detaily, nejsou interaktivní.

Kapitola 3

Návrh nového řešení

Tato kapitola se bude zaměřovat pouze na návrh uživatelského rozhraní a postupů na základě předem získaných informací. Konkrétní implementační detaily a technologie jsou popsány v kapitole 4.

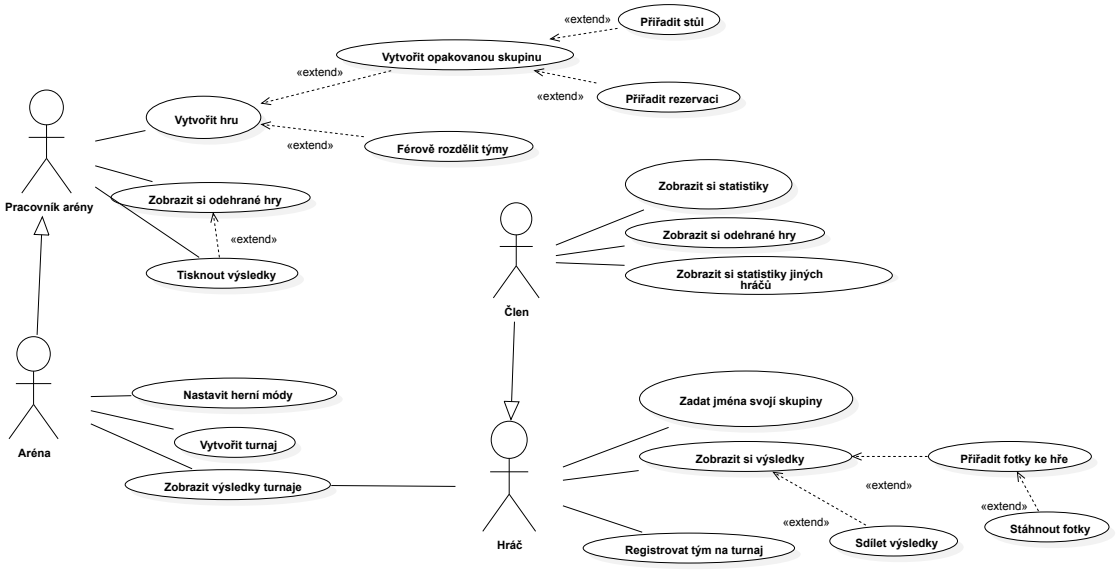
3.1 Specifikace požadavků a výběr vhodné formy aplikace

Na úplném začátku je potřeba rozhodnout formu celého systému. Nejprve musím jasně definovat veškeré požadavky a specifikace výsledného produktu (případy užití na obrázku 3.1). Z mých vlastních zkušeností i konzultací s majiteli arén a hráči jsem sestavil následující seznam požadavků:

- **Jednoduchost** – v IT světě celkem zavádějící výraz. Pod tímto pojmem si můžeme představit cokoliv. V případě našeho systému se ale jedná převážně o jednoduchost ovládání. Nejpoužívanější funkce systému musí být takzvaně „na dvě kliknutí“.
- **Použitelnost na různých zařízeních** – ne každá aréna má k dispozici stejný hardware. Aplikace by měla dobře fungovat na co možná nejvíce typech zařízení. U hráčů se dá předpokládat, že on-line funkce systému budou využívat primárně ze svých chytrých telefonů. Nevylučuje se ale ani možnost, že operátor arény nebude chtít pro ovládání celého systému využívat jiné zařízení než stolní počítač.
- **Distribuce** – arény by neměly být nuceny instalovat speciální hardware pro funkčnost celého systému. Ten by měl v ideálním případě umožnit i vzdálené monitorování a aktualizace. Není ale nutná instalace typu *Plug and Play* – prvotní instalace se provede pouze jednou a může vyžadovat odbornější přístup.
- **Spolehlivost** – systém by měl zvládat pracovat nezávisle na externích podmínkách. Například: pokud v aréně nebude fungovat internet, musí umožnit všechny nutné funkce jako stažení a tisk výsledků.

Vzhledem k těmto požadavkům jsem jako nejlepší variantu zvolil návrh systému jako webovou aplikaci. Toto rozhodnutí ze své podstaty splňuje požadavky na **použitelnost na různých zařízení**, ale musí být kladen důraz na responsivitu webu. Požadavku **spolehlivosti** dosáhneme instalací webové aplikace v lokální síti arény. Takto bude aplikace přístupná po LAN síti, nezávisle na připojení k internetu.

Distribuce je z mého pohledu nejnáročnější požadavek k řešení. Webová aplikace většinou vyžaduje existenci nějakého webového serveru. V tomto ohledu jsem se rozhodl jít cestou Docker kontejnerů. Hlavní výhodou je totiž možnost distribuce nezávisle na operačním systému, ale i úplná kontrola nad prostředím výsledné aplikace. Více se touto tématikou zabývám v kapitole 4.2.



Obrázek 3.1: Diagram případů užití pro jednotlivé uživatele výsledného systému.

3.2 Zpracování a ukládání dat z her

Tato část je nejvíce závislá na zvoleném vybavení LaserGame. Pro začátek jsem zvolil pouze vybavení LaserMaxx EVO5, ale také se mi podařilo získat informace o možnosti napojení na systém LaserForce.

3.2.1 LaserMaxx EVO5

Systém LaserMaxx ukládá všechny výsledky o hře do textových souborů s příponou `.game`. Tyto soubory nejsou strukturované v žádném typickém formátu jako `XML` nebo `JSON`, ale využívají vlastní formát, který není na první pohled úplně čitelný a vyžaduje lehké experimentování k pochopení všech uložených hodnot. Příklad a konkrétní popis souboru se nachází v příloze A. Pro zpracování výsledků stačí vytvořit jednoduchý parser, který data z textu zpracuje a uloží do relační `SQL` databáze.

Formát souboru s výsledky

Soubor obsahuje několik řádků ve formátu:

```
1 KATEGORIE{hodnoty,oddělené,čárkami}#
```

Výpis 3.1: Příklad jedné kategorie v souboru výsledků LaserMaxx

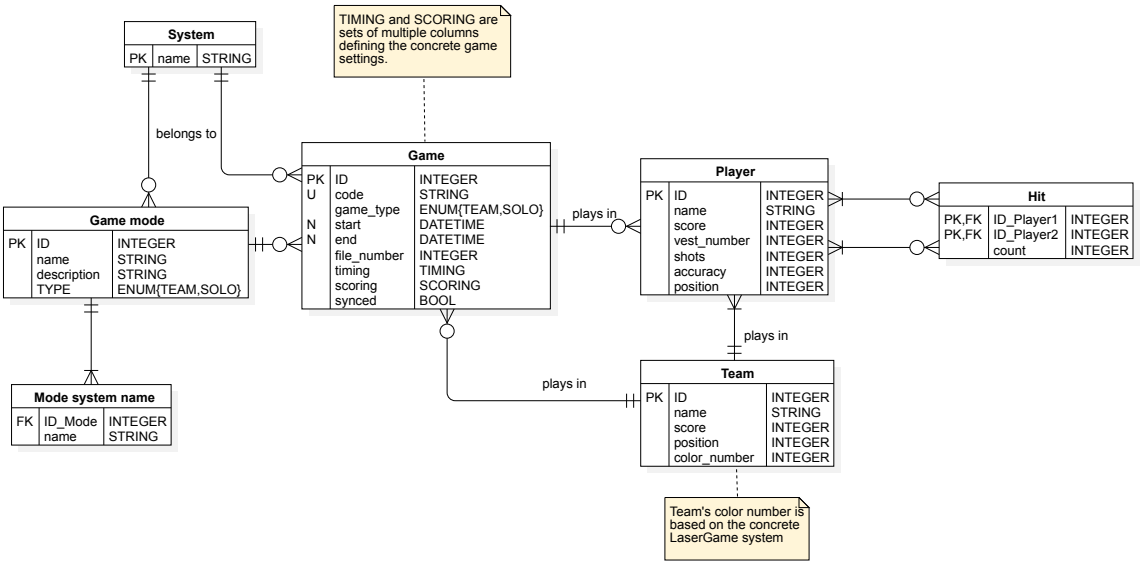
Kategorie jsou ve specifikovaném pořadí od nejzákladnějších informací o hře, po detaily o hráčích. Hodnoty kategorií jsou také neměnné a jejich počet a pořadí je jasně dané.

Konkrétní příklad a popis souboru se nachází v příloze A.

3.2.2 Návrh datové struktury

Z hlediska datových struktur lze rozlišit aplikaci na tři části: data her LaserGame, data pro framework (například uživatelé) a nastavení. Zároveň lze data rozdělit dle toho, zda se nachází v lokální aplikaci v arénách, nebo ve veřejné aplikaci pro hráče.

Jádrem jsou data LaserGame (obrázek 3.2). Jejich návrh je společný pro obě verze aplikace. Těžší bylo navrhnout takové rozhraní, které je co nejvíce univerzální pro různé systémy LaserGame. Zaměřil jsem se tedy převážně na vlastnosti, které mají všechny systémy společné.



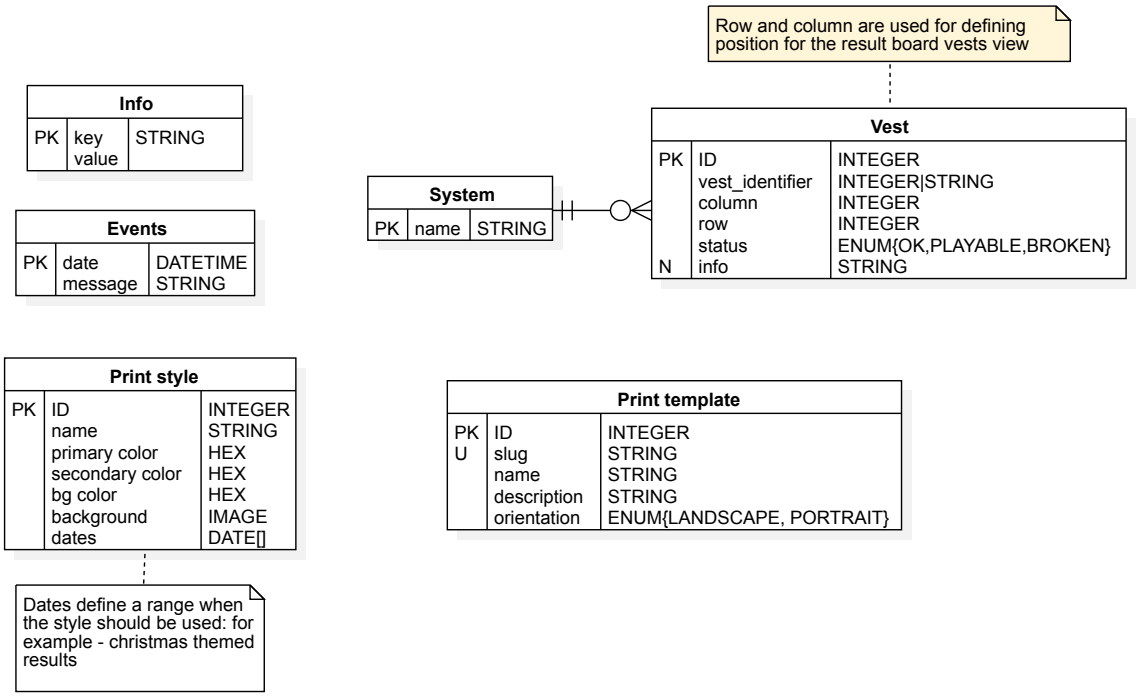
Obrázek 3.2: ER diagram dat pro ukládání základních informací o hrách LaserGame.

Framework lokální aplikace obsahuje jen pár pomocných entit (obrázek 3.3). Tyto entity slouží k ukládání samotného nastavení aplikace, případně systémových funkcí, jako je WebSocket event server (kapitola 4.5).

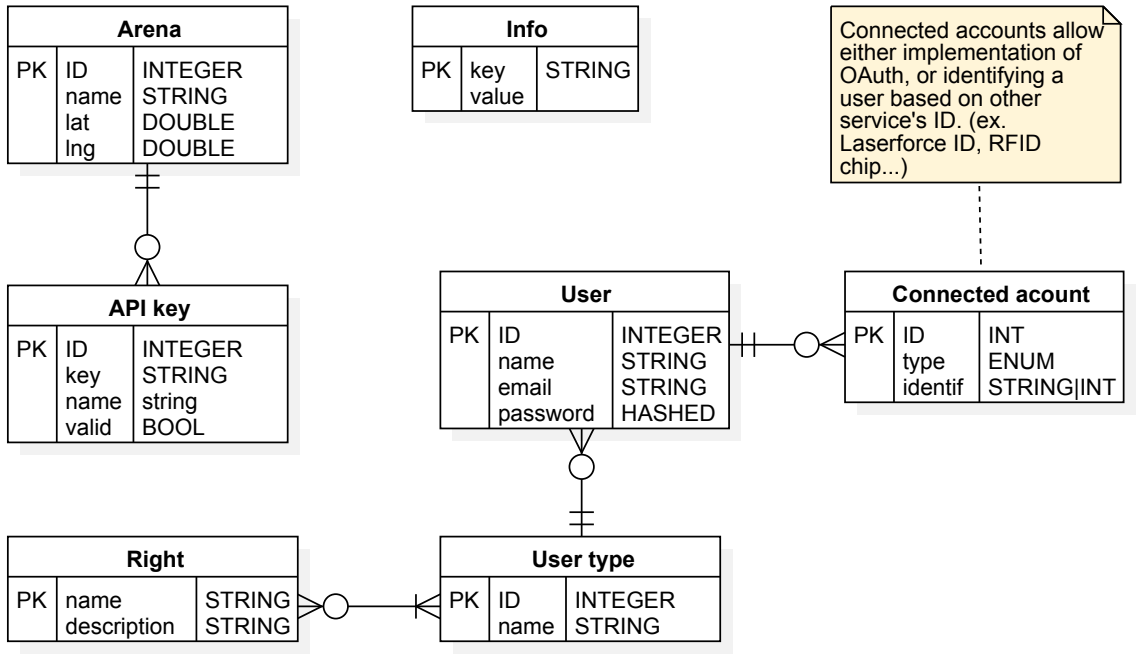
Pro veřejnou aplikaci se návrh lehce liší (obrázek 3.4). Obsahuje navíc informace o jednotlivých arénách a uživatelích. Aplikace umožňuje napojení uživatelů i na další systémy: například uložení Laserforce čísla hráče a RFID.

3.3 Zobrazování výsledků hráčům

Toto je jedna z nejpodstatnějších částí celého systému. Statistiky ze hry jsou první věc co hráče zajímají hned jak vyjdou z arény. Pokud výsledky nebudou pochopitelné pro 99% hráčů nebo nebudou obsahovat informace, které jsou pro hráče důležité, nebudou výsledky plnit svůj účel. Abych tyto požadavky dodržel, testoval jsem varianty přímo s hráči a vytvořil jsem dotazník, který pomohl v samotném návrhu – viz. příloha C.



Obrázek 3.3: ER diagram struktury nastavení a frameworku pro lokální aplikaci v aréně.



Obrázek 3.4: ER diagram struktury nastavení a frameworku pro lokální aplikaci v aréně.

3.3.1 Určení zobrazovaných statistik

Zde jsem nejvíce omezen statistikami, které mi poskytuje systém LaserGame. LaserMaxx neposkytuje žádné statistiky v čase, ale pouze základní informace, které vesty posílají na konci hry. To ale neznamená, že výsledky, které zobrazuje hráčům, musí obsahovat jen přepis informací z výsledného souboru, který software LaserMaxx generuje. Statistiky se dají zpracovávat, agregovat a doplňovat.

Ze základních statistik, které LaserMaxx poskytuje, je určitě nejdůležitější skóre (týmové i individuální). Dále získáme zásahy hráčů, procentuální přesnost střelby, počet výstřelů, získané bonusy při hře a zbývající životy i náboje.

Následně může systém dopočítat automaticky další zajímavé statistiky jako počet minutých výstřelů, vyhodnotit nejoblíbenější cíle hráčů (hráče, kterého jiný hráč zasáhl nejvíce) a spočítat poměr zásahů proti smrtím.

V poslední řadě jsou statistiky, které může systém vyhodnotit na základě nějaké logiky. Zde jsem vymyslel dvě různé možnosti. Trofeje, které hráč získá, když dosáhne nějakého milníku ve hře (např. hráč s největší přesností) a pořadí všech hráčů, kteří hráli v ten samý den. Obě tyto statistiky sklidily poměrně velký úspěch, jelikož je to jedna z prvních věcí, které hráči více rozebírají a probírají je se svými spoluhráči.

Jak vyplývá z výsledků dotazníků v příloze C, ne všechny statistiky jsou pro hráče stejně zajímavé. Proto je vhodné statistiky rozdělit, nezobrazovat všechny dostupné hned, ale na jednotlivých médiích je zobrazit postupně s různou mírou detailnosti. Rozdělení statistik zároveň pomáhá omezení vizuálního šumu a celkové čitelnosti. Různí uživatelé mají různou míru tolerance pro komplexní a rušivý obsah [12]. Jednotlivé formy zobrazení budou podrobně zhodnoceny v následujících sekcích.

3.3.2 Výsledky na výsledkové tabuli

Výsledková tabule je první věc, kterou hráč vidí po dokončení hry. Z výsledkové tabule se hráč ihned dozví jak dopadl, jestli zvítězil, nebo ne. Oproti ostatním médiím není výsledková tabule přenositelná a hráči jí vidí jen před dveřmi do arény. Hlavním cílem výsledkové tabule je tedy převážně rychle představit pořadí a skóre jednotlivých hráčů i týmů. Musí být okamžitě poznat, jak hra dopadla. Není žádoucí, aby výsledková tabule byla příliš detailní. Hráči by analyzovali svůj výkon příliš dlouho a blokovali prostor pro přípravu další hry.

Pokud hra ještě neskončila, na výsledkové tabuli není co zobrazit. LaserMaxx neumožňuje jednoduše získávat výsledky v reálném čase, proto není možné zobrazovat žádný reálný průběh. Rozhodl jsem se tedy pro vytvoření dalších dvou obrazovek: jednu na zobrazení statistik z dnešního dne a druhou zobrazující vesty se jmény v přípravné fázi před samotnou hrou. Na těchto obrazovkách je možné navíc zobrazit odpočet času do konce hry.

Výhodou návrhu výsledků pro výsledkovou tabuli je fakt, že bude zobrazovaná na obrazovkách, které mají stejné rozlišení, nebo alespoň poměr stran. Hráči si nebudou výsledkovou tabuli otevírat na svém telefonu. Kromě výsledkové tabule se na obrazovce nic jiného nezobrazuje. Scrollování není možné. Spíše než webovou stránku bude návrh připomínat plakát o poměru stran 16:9.

Obecný návrh

Návrh jako takový následuje principy **flat design**. Je použito jednoduché bezpatkové písmo. Neobsahuje žádné zbytečné ozdoby, pouze čisté barvy. Na všech prvcích, kromě loga, je slabý

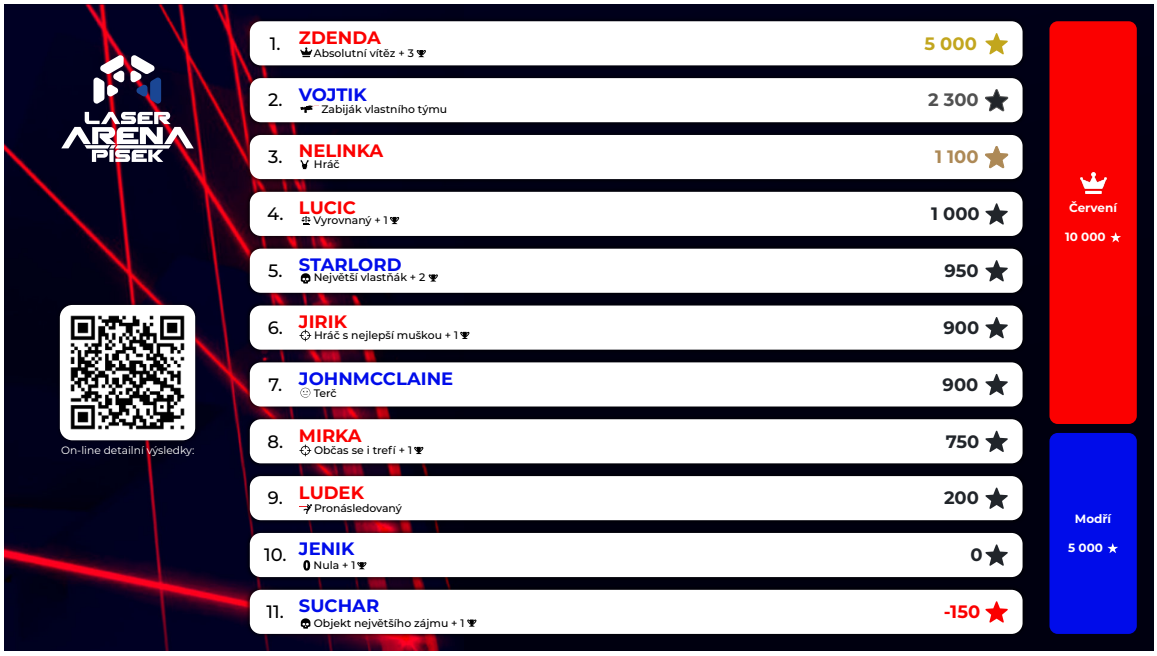
stín, který signalizuje zvýšenou pozici na ose Z a pomáhá rozlišení prvků v případě světlého pozadí. Inspiroval jsem se knihou **Flat design & Colors** [3].

Je vhodné, aby vzhled výsledků byl vždy přizpůsoben logotypu dané arény. V návrhu je zakomponovaný prostor pro logo arény a počítá se i s pozadím, které může mít každá aréna unikátní. Různé návrhy pozadí jsou vidět na obrázcích 3.5, 3.6 a 3.7.

Návrh výsledků hry

Jak jsem již zmínil, výsledky na výsledkové tabuli nesmí hráče přehltit. Zobrazují se jen ty nejpodstatnější informace, které hráče zajímají po ukončení hry. Jedná se jen o pořadí a skóre.

Obrazovka je jasně rozdělena na 2–3 sloupce. To pomáhá hráči v uvědomění si, jaké části spolu logicky souvisí a usnadňuje mu celkovou orientaci [12]. Celým výsledkům dominuje individuální pořadí a skóre, které se nachází ve středu a zabírá asi 70% celé plochy. V pravém sloupci je týmové skóre. Pořadí týmů je znázorněno pořadím boxů shora dolů. Výška boxů je zároveň závislá na poměru skóre jednotlivých týmů. Na první pohled je zřejmé, jak velký rozdíl mezi jednotlivými týmy je. V levé části jsou dodatečné prvky jako logo arény a QR kód s odkazem na on-line detailní výsledky.



Obrázek 3.5: Návrh výsledkové tabule – Výsledky po hře dvou týmů

V individuálních výsledcích hráčů se nachází jen pořadí, jméno hráče, skóre a jedna ze získaných trofejí. Jméno hráče je barevně rozlišeno podle týmu ve kterém hrál. Trofeje, ačkoliv ne úplně důležité, přidávají další, pro hráče poměrně zajímavou informaci[1]. Hráči se podívají a okamžitě vidí, kdo byl nejlepší střelec, kdo se nejméně trefoval,... Přidání trofejí je sice v rozporu s myšlenkou udržet výsledky co nejjednodušších a zobrazovat jen podstatné informace, ale na druhou stranu působí jako zajímavý bonus. Navíc hráče navnadí na další, detailnější výsledky.

[1] Trofejemi se více zabývám v sekci 3.3.4

Obrázek 3.6: Návrh výsledkové tabule – Výsledky po hře tří týmů

Pravý sloupec obsahuje všechny týmy, barevně rozlišené, které ve hře hráli[2]. Tento sloupec je vynechaný ve hře všichni proti všem (= bez týmů) – viz. obrázek 3.7. Box týmů obsahuje jméno týmu a jeho celkové skóre. Oproti individuálním výsledků se zde nenachází číslo pořadí. Při malém počtu týmů je zbytečné a pro hráče není těžké rychle odhadnout, jak se umístily. Hráče navíc většinou nezajímá přesné pořadí, ale jen první a poslední místo.

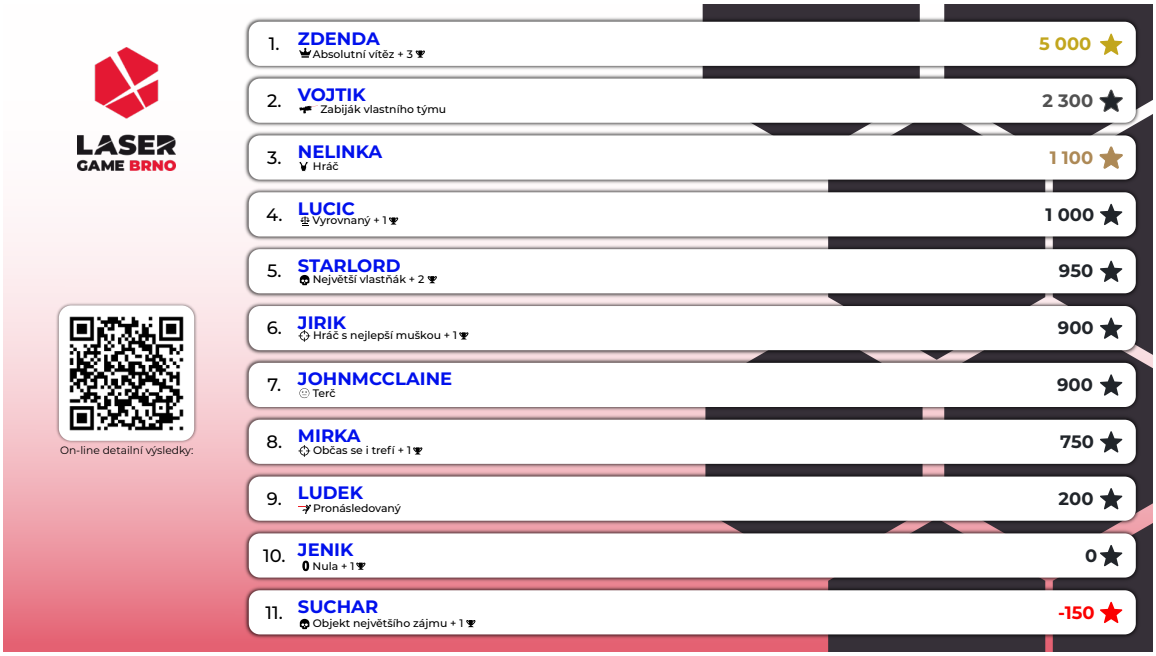
Levý sloupec je poměrně prázdný a nabízí tak prostor pro logo a další vlastní grafické prvky arény. QR kód umožňuje těm, kteří mají u sebe telefon, zobrazit detailní statistiky okamžitě, ještě než dostanou vytištěné papírové. Teoreticky může přítomnost QR kódu úplně zamezit nutnosti výsledky tisknout. V praxi by se tato možnost využívala pravděpodobně jen u pokročilých skupin.

Běžná akce hráče po hře vypadá následovně:

1. Hráč vyjde z arény. Na první pohled vidí barvy a tím ví, který tým vyhrál.
2. Ještě než vydechne, najde rychle v seznamu své jméno a zjistí svojí pozici.
3. Odloží vestu se zbraní, postaví se před výsledky ještě jednou, začne je zkoumat trochu podrobněji a diskutuje se svými spoluhráči.
4. Všimne si trofejí, a že je například „Největší mimoň“ (hráč, který se nejvícekrát netrefil).
5. V tu chvíli se se svými spoluhráči zasměje a jakmile si prošel vše zajímavé, odchází ke stolu.

Popsaná situace vychází z mé vlastní zkušenosti, ale i ze zkušenosti pracovníků Laser arény Písek.

[2]Pro LaserMaxx je maximální počet týmů 6



Obrázek 3.7: Návrh výsledkové tabule – Výsledky po hře všichni proti všem

Návrh statistik dnešního dne

Pro tuto obrazovku je potřeba určit, které statistiky může systém vůbec zobrazit. Jednou z možností je zobrazovat souhrnné, průměrné statistiky dne. Tato možnost se po krátkém průzkumu nejevila pro hráče příliš zajímavá. S větším úspěchem se setkala možnost zobrazení nejlepších hráčů dne.

Rozhodl jsem se tedy do návrhu zakomponovat tři nejlepší hráče dne podle skóre a jednoho nejlepšího hráče pro čtyři kategorie: počet zásahů, počet smrtí, přesnost střelby a počet výstřelů.

Vzhled je více tabulkový, ale použitím fontu, barev, tvarů a ikon souhlasí i s předchozím návrhem výsledků. Podbarvení boxů nadpisů a barva plovoucích textů může být změněna na základě vybraného pozadí a barev konkrétní arény.

Výsledný návrh je vidět na obrázku 3.8 a 3.9.

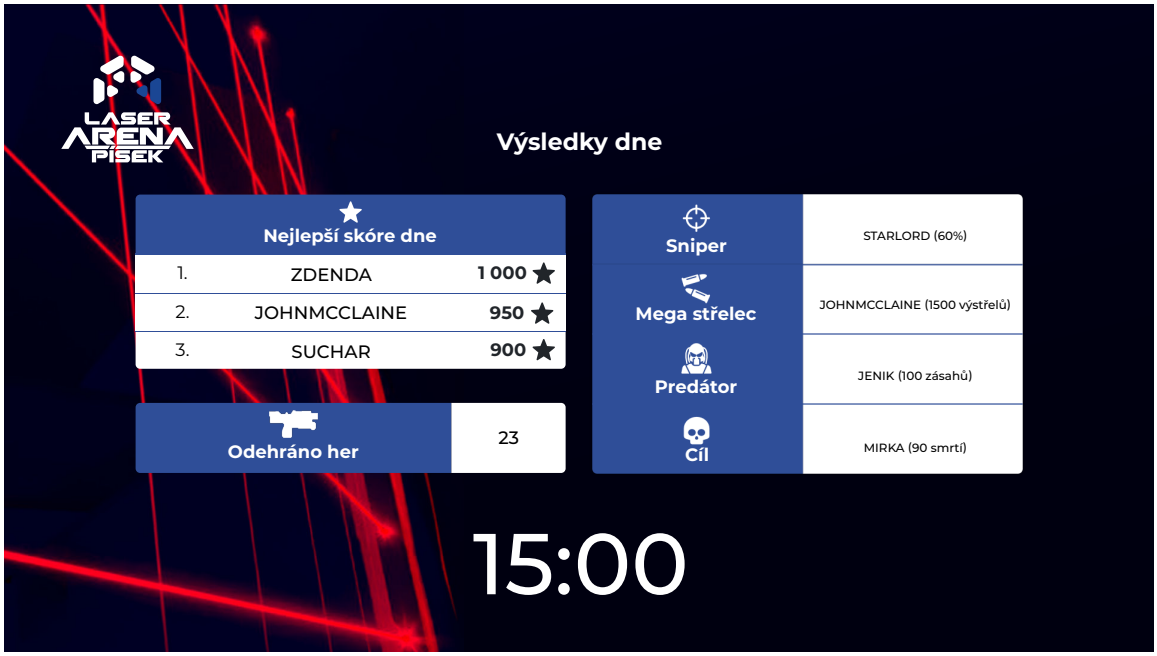
Návrh rozložení vest

Hlavním účelem této obrazovky je zobrazit vesty se jmény, pro rychlejší orientaci hráčů, která vesta je jejich. Každá vesta má vlastní číslo a display, na kterém se zobrazuje jméno hráče. Vesty jsou rozmístěné v přípravně na stojanech ve tvaru do U[3]. Na obrazovce tedy stačí jen zobrazit ikony vest s jejich číslem a jménem hráče, kterému vesta patří: obrázky 3.10 a 3.11.

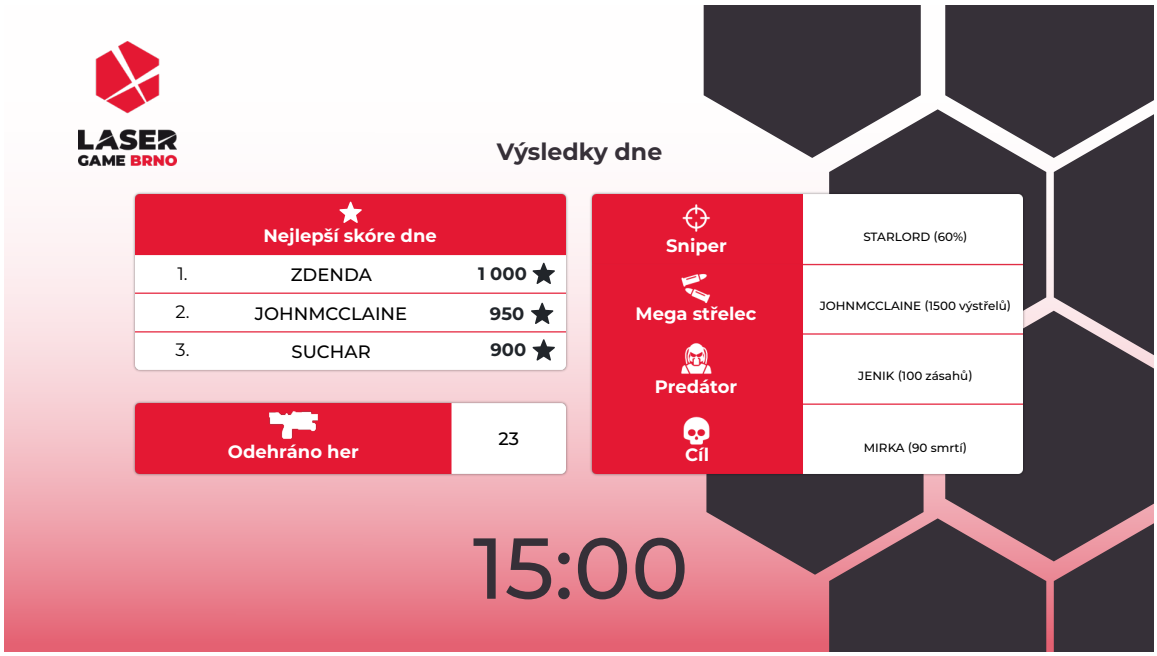
Návrh využívá stejných grafických prvků jako v obou předchozích případech. Boxy vest jsou barevně rozlišené podle barvy týmu hráče. Jméno hráče upravuje svojí velikost na základě délky jména, aby se do boxu vešlo[4].

[3]Rozložení vest po stěnách do U nebo L je konzistentní mezi většinou arén. Některé se liší jen tím, že vesty dávají do více řad nad sebe.

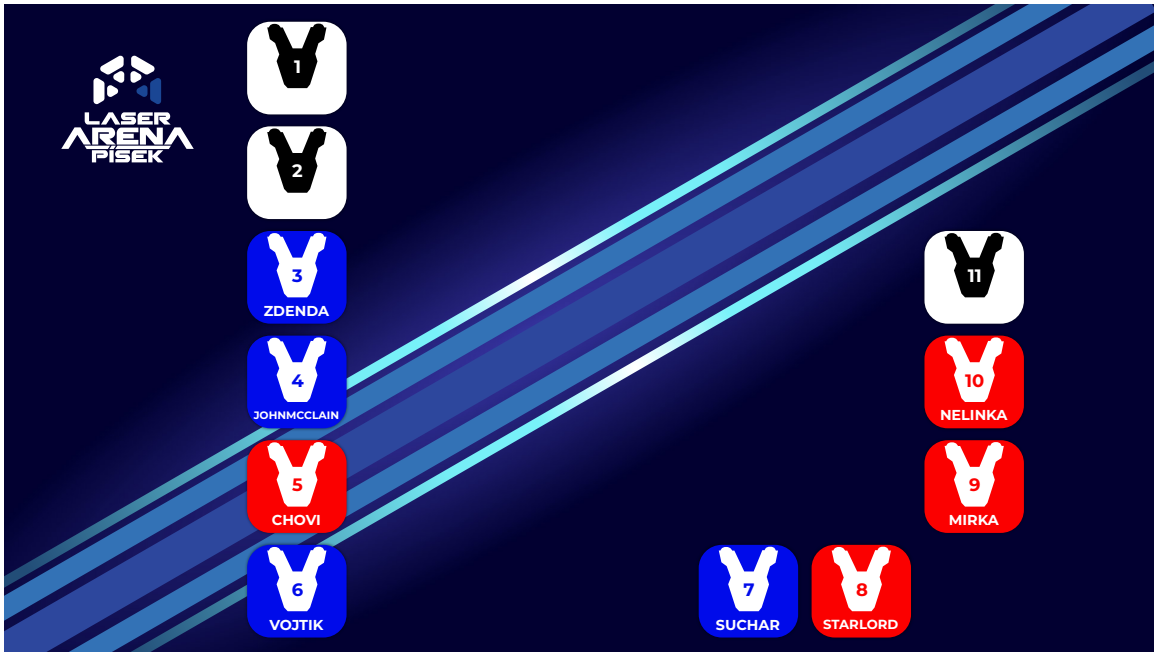
[4]Software LaserMaxx dovolí zadat jméno hráče maximálně 12 znaků dlouhé.



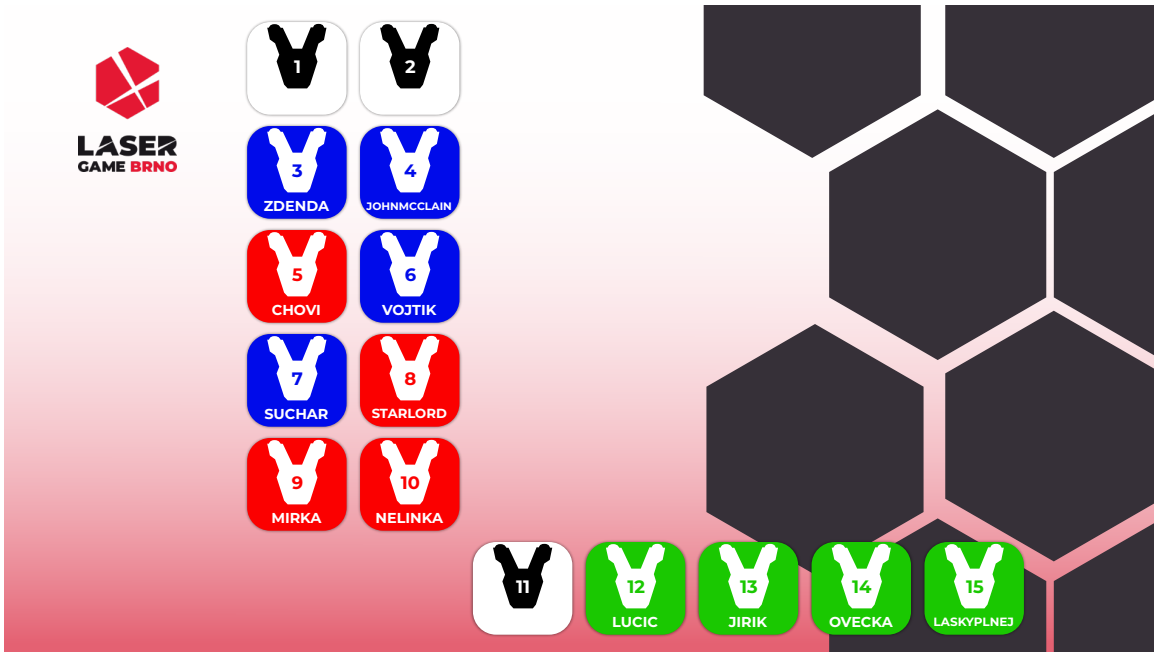
Obrázek 3.8: Návrh výsledkové tabule – Statistiky dnešního dne s odpočtem času do konce hry



Obrázek 3.9: Návrh výsledkové tabule – Statistiky dnešního dne s odpočtem času do konce hry na světlém pozadí



Obrázek 3.10: Návrh výsledkové tabule – Rozložení vest před hrou podle reálného umístění na stojanech



Obrázek 3.11: Návrh výsledkové tabule – Rozložení vest před hrou podle reálného umístění na stojanech na světlém pozadí

3.3.3 Tisknuté výsledky

Tisknuté výsledky se standardně rozdávají hráčům ve všech arénách a obsahují detaily ze hry. Hráči nad výsledky sedí a probírají je se svými spoluhráči. Musí tudíž obsahovat všechny důležité informace o jednotlivých hráčích. Stejně jako u výsledkové tabule i tisknuté výsledky pracují s jasně daným formátem – papír A4. Je samozřejmé, že výsledky nemohou být jakkoliv interaktivní, ani například animované.

Co se zobrazovaných statistik týče, není nutné se omezovat pouze na skóre jako u výsledkové tabule. Na druhou stranu výsledky pracují s omezeným prostorem a není jednoduché na papír zahrnout veškeré možné informace, které systém poskytuje.

Vytvořené řešení bude navíc umožňovat tisknout různé typy výsledků, které mohou vypadat pokaždé jinak. Toto rozhodnutí bylo podpořené hlavně faktem, že LaserGame hrají různí lidé, různých věkových a zkušenostních skupin. Tyto jednotlivé skupiny mají různé požadavky. Děti například ocení více grafickou podobu, která bude obsahovat jen základní a zajímavé informace. Dospělí, kteří hrají pravidelně naopak ocení detailnější statistiky a zorientují se snáz i ve složitějších tabulkách. Pro účely této práce byli navrženy dvě varianty, ale počítá se s možností více variant pro budoucí vývoj.

Tabulková varianta

Tato varianta je mířená převážně na hráče, kteří chtějí většinu detailních statistik rovnou na tištěných výsledcích, bez nutnosti pokračovat do on-line výsledků. Varianta vychází z návrhu, který byl již v Laser aréně Písek nějakou dobu používán. Hlavním cílem je zobrazit co největší množství informací přímo na papíře. Statistiky jsou proto pouze tabulkové, plné čísel.

Na obrázku 3.12 jsou výsledky hry dvou týmů. Pod hlavičkou se nachází sekce hráčů se všemi statistikami. Následuje tabulka umístění dne, kde je znázorněné celkové umístění hráčů podle různých statistik v daný den. Ve hře s bonusy by byla tabulka hráčů rozšířena o počty jednotlivých bonusů a tabulka umístění dne by se skryla. Ve spodní části se nachází tabulka zásahů mezi jednotlivými hráči, vpravo pak výsledky týmů a ocenění hráčů.

Grafická varianta

Grafická varianta cílí převážně na méně náročné hráče a děti. Poskytuje omezené množství detailů. Je navržena na papír A4 orientovaný na šířku. Cílí převážně na hráče samotné a jejich porovnání. Aktuální návrh počítá až s jedenácti hráči[5] a maximálně šesti týmy.

Při hře týmů (obrázek 3.13) se pod samotnou hlavičkou zobrazí řádek týmů, který podobným způsobem jako na výsledkové tabuli (kapitola 3.3.2) znázorňuje jejich pořadí zleva doprava a jejich skóre poměrem šířek boxů. Poté následují samotní hráči a jejich detailnější statistiky. Skóre je znázorněno podobně jako u týmů – barevným boxem, jehož výška odpovídá poměru skóre proti ostatním hráčům. Přesnost střelby je znázorněna koláčovým grafem. Počet výstřelů je uveden číslem nad kterým jsou nábojnice – počet naplněných se odvíjí od poměru vystřelených výstřelů proti ostatním hráčům. Zásahy a smrti hráče jsou opět znázorněny boxy, jejichž velikost vychází z poměru mezi vlastním počtem zásahů a smrtí hráče. Oba boxy mohou být navíc v pravé části rozšířené stejným způsobem o zásahy od/do vlastních hráčů. Jako poslední se zde nachází nejoblíbenější cíl hráče a největší zabiják hráče.

[5] Typický počet vest pro arény LaserMaxx

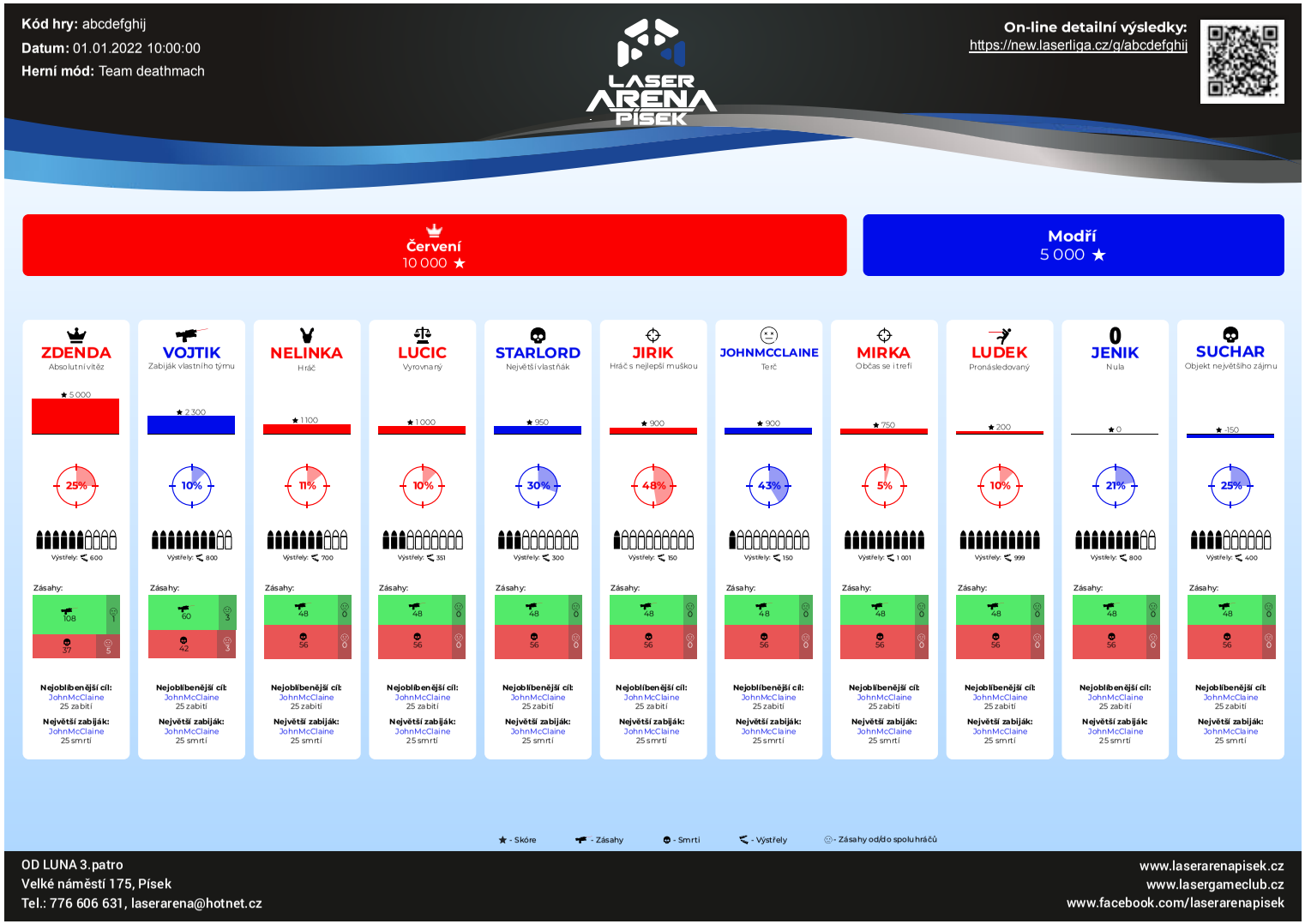
Jméno			Terka	Ritus	Anca	Sisa	Zuzka	Misa	David	Vena	Karolinka	Vaneska
Tým			●	●	●	●	●	●	●	●	●	●
Skóre			5 750	5 425	5 050	4 275	4 000	3 550	2 275	325	-250	-1 375
Počet výstřelů			316	591	329	383	441	312	378	542	379	227
Počet střel do prázdna			225	498	247	309	367	242	322	505	360	205
Úspěšnost			29%	16%	25%	19%	17%	22%	15%	7%	5%	10%
Hráči	Zásahy		91	93	82	74	74	70	56	37	19	22
	Smrti		69	61	60	52	68	65	65	60	46	72
Týmy	Soupeř	Zásahy	91	86	80	69	72	68	55	34	17	21
		Smrti	67	60	58	50	63	64	64	60	38	69
	Vlastní	Zásahy	0	7	2	5	2	2	1	3	2	1
		Smrti	2	1	2	2	5	1	1	0	8	3

Jméno		Terka	Ritus	Anca	Sisa	Zuzka	Misa	David	Vena	Karolinka	Vaneska
Denní pořadí podle	Skóre	114.	109.	103.	96.	92.	84.	67.	29.	16.	6.
	Přesnosti	126.	96.	119.	105.	101.	111.	90.	19.	10.	47.
	Zásahů	115.	116.	108.	96.	96.	91.	73.	42.	10.	15.
	Smrtí	103.	86.	80.	63.	100.	97.	97.	80.	49.	107.

Zasáhl si hráče	Terka	Ritus	Anca	Sisa	Zuzka	Misa	David	Vena	Karolinka	Vaneska
1. Terka	0	9	32	16	21	0	0	0	13	0
2. Ritus	8	0	1	2	0	27	17	15	4	19
3. Anca	24	0	0	0	2	12	16	13	0	15
4. Sisa	20	0	0	0	3	8	14	16	2	11
5. Zuzka	15	0	0	0	0	15	13	13	2	16
6. Misa	1	18	6	18	14	0	0	0	12	1
7. David	1	14	12	8	13	0	0	0	8	0
8. Vena	0	8	4	5	14	1	0	0	3	2
9. Karolinka	0	1	1	0	0	2	4	3	0	8
10. Vaneska	0	11	4	3	1	0	1	0	2	0

Tým	●	●	Zásahů	Skóre	Úspěšnost	Výstřelů
●	18	324	342	18 500	16,11%	2 123
●	269	7	276	10 525	15,55%	1 775

Obrázek 3.12: Návrh tisknutých výsledků – tabulková varianta, 10 hráčů, 2 týmy, bez bonusů



Obrázek 3.13: Návrh tisknutých výsledků – grafická varianta, 11 hráčů, 2 týmy, bez bonusů

Na obrázku 3.14 je vidět hra bez týmů (všichni proti všem). Výsledky neobsahují řádek týmů, ani zásahy spoluhráčů. Ve hře byli zapnuté bonusy, jejichž počty jsou vypsané v dolní části boxu u každého hráče. Jelikož hráčů bylo méně jak sedm, pro vyplnění prostoru zde přibyla i tabulka jednotlivých zásahů mezi hráči, která je běžně k vidění na on-line výsledcích (pod QR kódem).

3.3.4 On-line výsledky pro hráče

On-line výsledky mají ze všech tří zobrazení největší svobodu, jelikož nejsou omezené místem a hráči si je mohou zkoumat jak dlouho potřebují. Hlavním účelem této formy výsledků je poskytnout co nejdetailnější statistiky pro hráče a zároveň umožnit například i sdílení na sociální sítě.

Samotný návrh (obrázek 3.15) čerpá převážně z grafické varianty tištěných výsledků. Na úvodu stránky se nachází hlavička se všemi informacemi o hře a rozbalovací legendou. Následuje řádek týmů, který je totožný s tisknutými výsledky (obrázek 3.13). Každý hráč je ve svém rozbalovacím boxu, kde se nachází jeho detailní informace (obrázek 3.16). Pod nimi se nachází tabulka konkrétních zásahů mezi hráči. Tato tabulka je na mobilních zařízeních nahrazena přehledem zásahů v detailu hráče (obrázek 3.17).

Detail hráče (obrázek 3.16) opět používá téměř stejné prvky jako grafická varianta tisknutých výsledků s pár rozdíly. V horní části se nachází všechny trofeje, které ve hře hráč získal, s tlačítkem pro zobrazení přehledu všech dostupných trofejí. Ve spodní části se nachází denní umístění podle jednotlivých statistik. Po kliknutí na box se uživateli zobrazí tabulka s pořadím všech hráčů dle dané statistiky. Na mobilních zařízeních se všechny prvky přeskládají pod sebe.

Návrh implementace on-line výsledků

Veřejná část aplikace je kompletně oddělená od lokální aplikace v arénách. Její primární účel je sjednocování her ze všech připojených arén a zobrazování je hráčům.

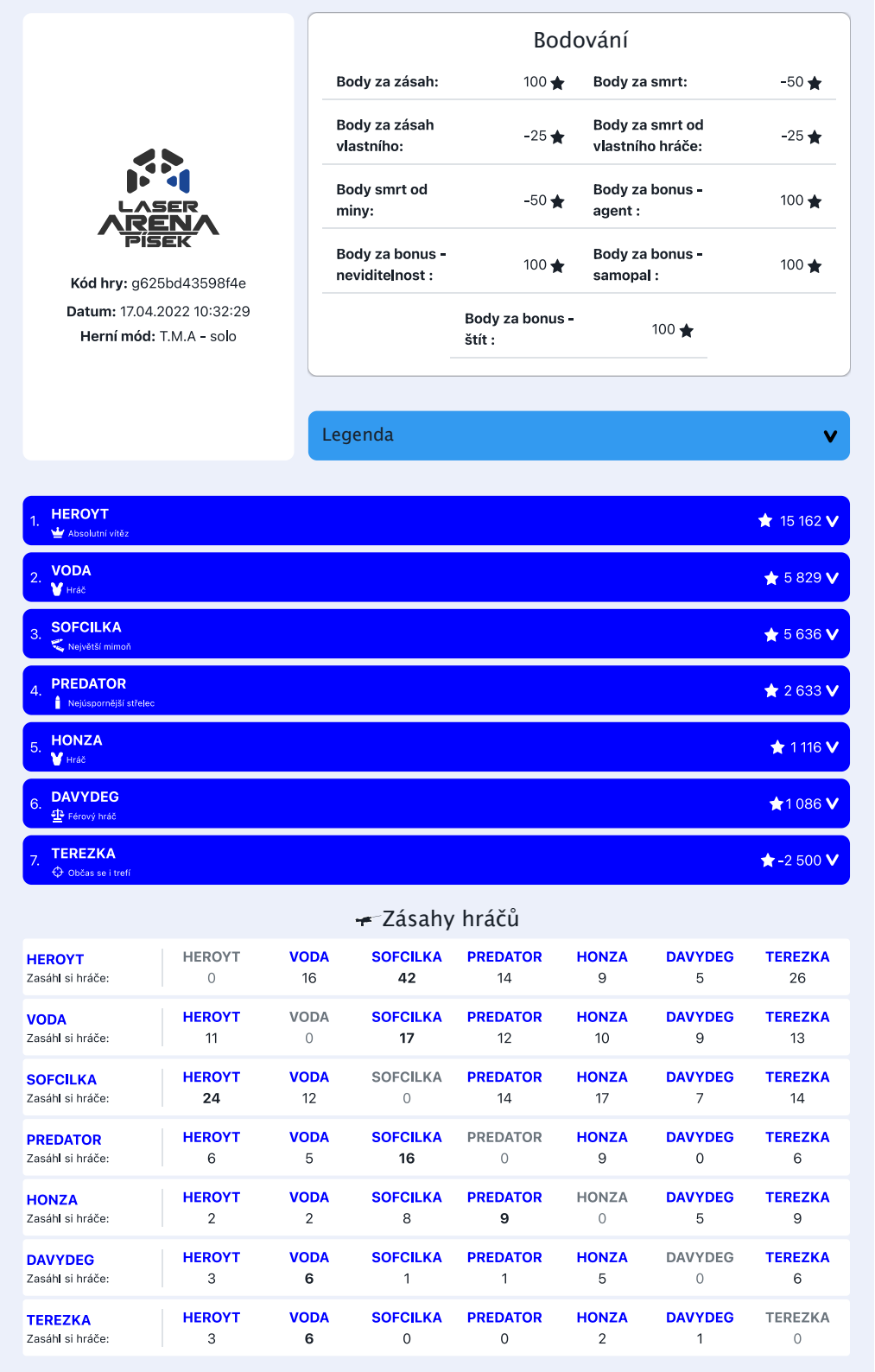
Aplikace běží na typickém webovém Apache serveru na doméně *laserliga.cz*. V počátku je dostatečné, pokud bude aplikace obsahovat pouze REST API pro synchronizaci dat a jedno view s výsledky hry. V budoucnosti je možné službu rozšířit o uživatelské účty hráčů, získávání průběžných statistik a žebříčků.

REST API musí umožňovat autentizaci arény. Toho lze dosáhnout pomocí unikátních tokenů, které bude lokální aplikace posílat v HTTP hlavičkách – `Authorization: Bearer <token>` [10]. Jedna aréna může mít přiřazených několik tokenů. Tokeny mají neurčitou platnost – je nutné je manuálně deaktivovat.

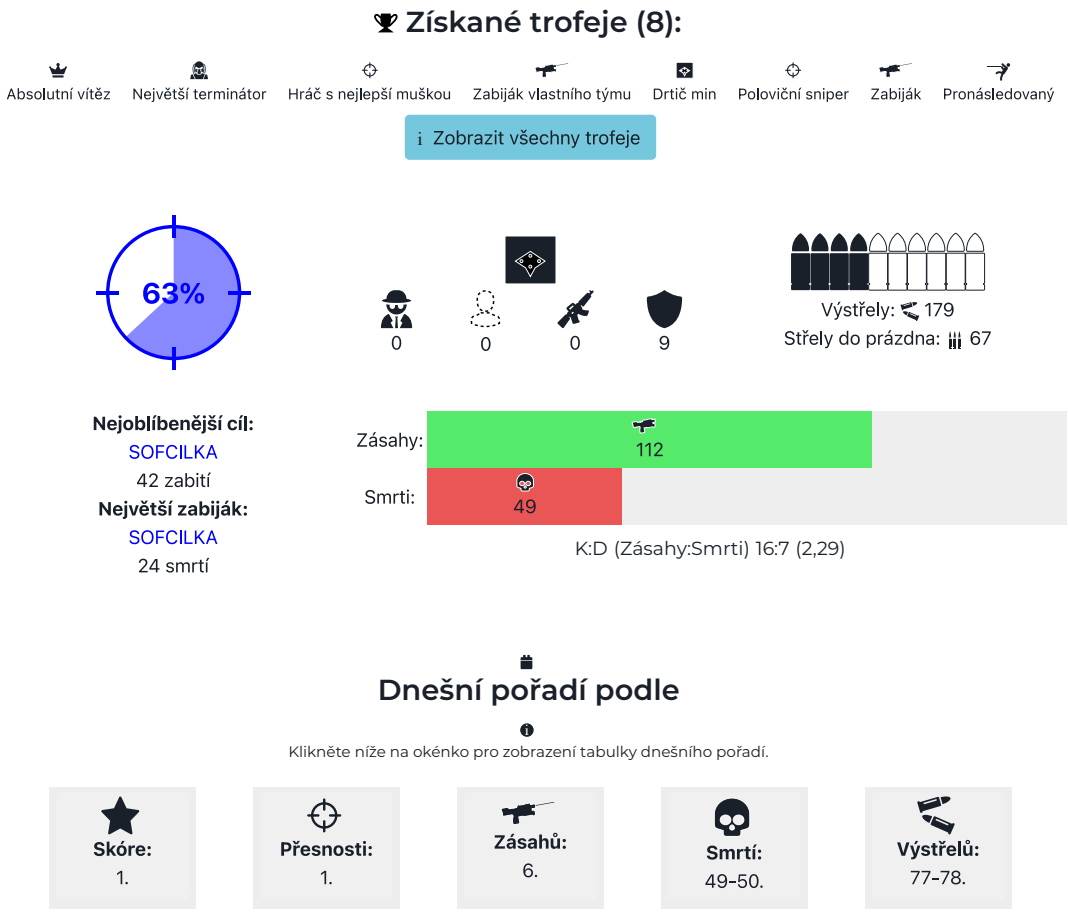
3.4 Funkce usnadňující práce pracovníka arény

Tato část není vyloženě nutná pro fungování celého systému, ale nabízí takové možnosti, které mohou v běžném provozu usnadnit operátorovi práci a čas. Tím „není nutná“ nemyslím ale, že je zbytečná. Při návrhu těchto funkcí jsem úzce spolupracoval s operátory arén a zjistil jsem, na čem ztrácejí nejvíce času, co by jejich práci usnadnilo. Zároveň jsem vytvořil prostor pro nové nápady, které aktuální systémy vůbec neumožňují. Mezi takové nápady patří například napojení na rezervační systém, vytváření skupin a organizace turnajů. Tyto nové funkce popisuji podrobněji v kapitolách níže.

Obrázek 3.14: Návrh tisknutých výsledků – grafická varianta, 6 hráčů, všichni proti všem, s bonusy



Obrázek 3.15: Návrh on-line výsledků. Celkový pohled na rozložení stránky. Ve hře bez týmů (všichni proti všem), 7 hráčů.



Obrázek 3.16: Návrh on-line výsledků. Detail statistik jednoho hráče.

Obrázek 3.17: Návrh on-line výsledků. Detail zásahů hráče pro mobilní zařízení.

3.4.1 Zadávání hráčů do hry

Funkce kterou operátor arény používá nejčastěji a stráví na ní nejvíce času. Celý systém zadávání většinou funguje tak, že na jednotlivé vesty operátor napíše jména hráčů, rozřadí je do týmů, vybere herní a hudební mód a nahrává hru. Tento postup je v každé aréně a na každém vybavení, až na drobné detaily, prakticky totožný. Samotné rozhraní zadávání můžeme rozdělit primárně do dvou kategorií.

1. **Zaměřené na vesty** (obrázek 3.18): operátor zadává jména na konkrétní vesty a týmy rozděluje následně

2. **Zaměřené na týmy** (obrázek 3.19): operátor zadává jména do týmů a vesty se přiřazují automaticky s možností manuální změny

Po diskusi mezi operátory nemohu s jistotou říct, který způsob zadávání je lepší. Jednodušší a rychlejší se zdá postup zaměřený na týmy. Na druhou stranu může někdy operátor chtít do hry pustit jen konkrétní vesty, případně nejprve zadat jména, a až poté týmy rozřadit.

Při návrhu jsem se inspiroval výstupem srovnání aktuálních systémů v kapitole 2.2. Rozhodl jsem se pro spojení všech důležitých funkcí do jedné jediné obrazovky, pro rychlou orientaci operátora.

Při zadávání hráčů na vesty jsem zvolil cestu výběru na základě preference operátora. Implementované bude zadávání zaměřené na vesty i na týmy. Obě řešení musí následně projít detailním uživatelským testováním.

Dalším zjednodušením, které by měla aplikace umožnit, je rozdělený výběr herních a hudebních módů. Toto je problém převážně u systému LaserMaxx, kde je herní a hudební mód spojený v jednom nastavení.

Velmi užitečnou a využívanou funkcí je náhodné rozřazení hráčů do týmů. Rozřazování může být i rozšířeno o váhy u jednotlivých hráčů, které indikují jejich výkonnostní úroveň ve hře. Algoritmus rozřazení se bude následně snažit vytvořit týmy co nejvíce vyrovnané. Tato varianta musí být co nejjednodušší, aby operátora co nejméně zatěžovala. Rozhodl jsem se zvolit variantu pouze tří výkonnostních úrovní, mezi kterými lze hráče přepínat: začátečník, hráč, pokročilý. V případě rozšíření skupin z následující sekce 3.4.2, lze váhy hráčů počítat i automaticky na základě jejich předchozích výkonů.

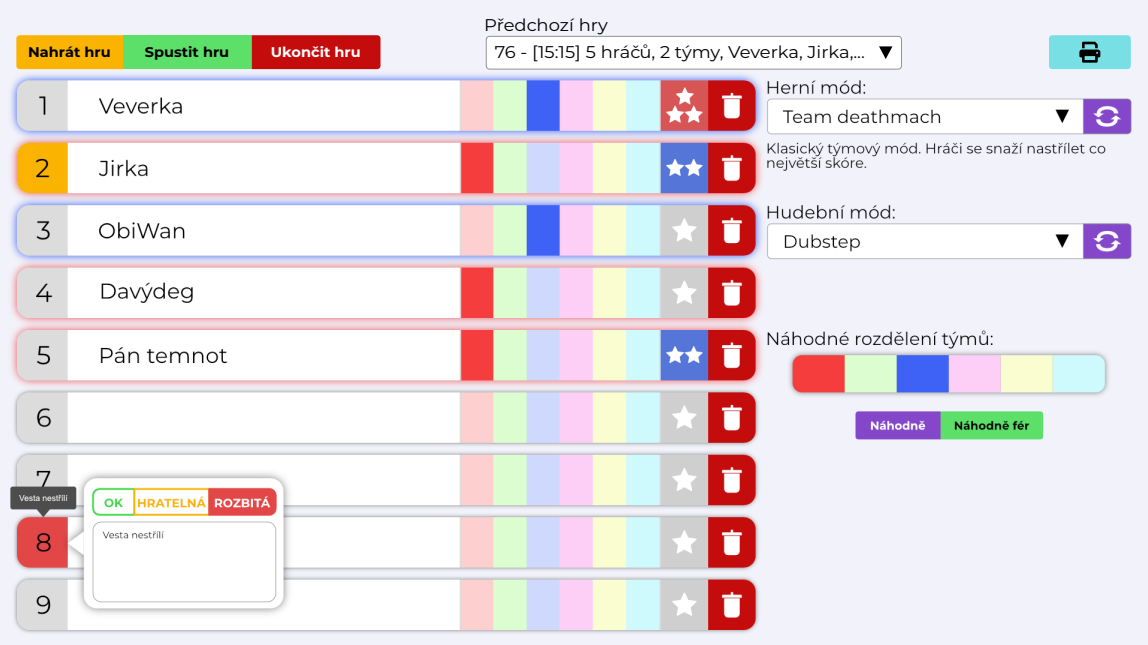
3.4.2 Vytváření skupin

V arénách může často nastat situace, kdy je třeba několikrát střídat ty samé skupiny hráčů. Jedna skupina si zahraje, druhá odpočívá a poté se prohodí. Operátoři nyní mají dvě možnosti na zadávání opakovaných skupin. Buďto manuálně zadají jména do systému znovu, nebo vyhledají předchozí hru a zkopírují jména hráčů[6]. Cílem funkce vytváření opakovaných skupin je tento proces ještě urychlit a nabídnout i nové zajímavé funkce:

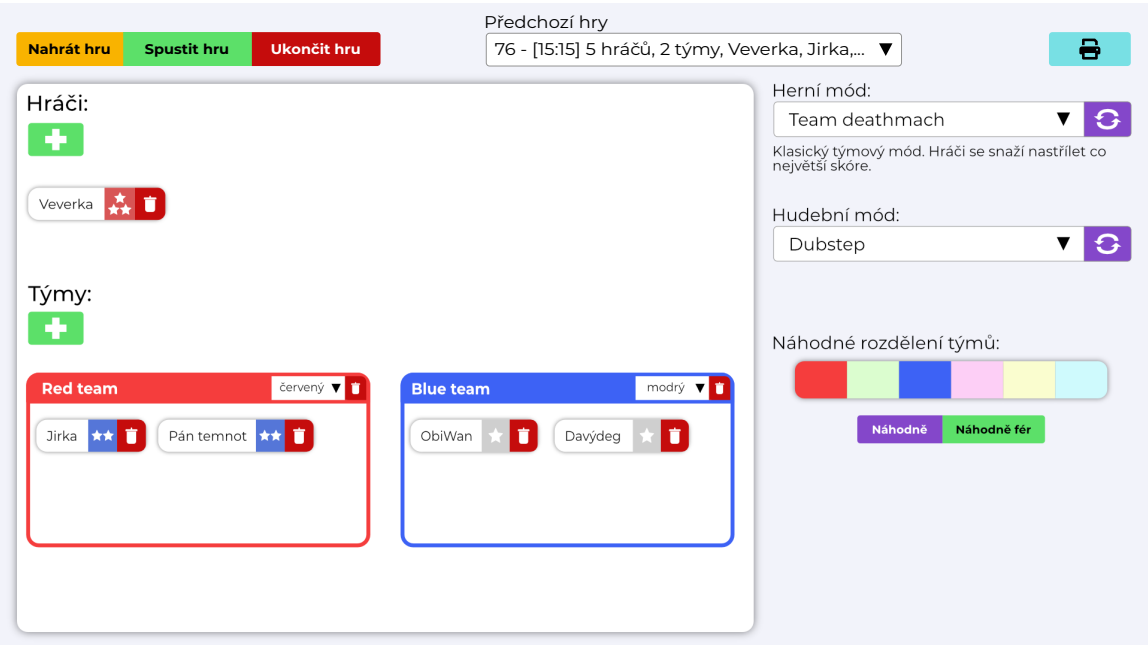
- Propojení výsledků z her v on-line rozhraní (pod QR kódem) pro snazší orientaci mezi hrami.

- Výpočet průběžných statistik hráčů ze všech odehraných her.

[6] V systému LaserMaxx lze z historie kopírovat všechny hráče pomocí kontextuálního menu po stisku pravého tlačítka myši.



Obrázek 3.18: Návrh rozhraní pro zadávání hráčů do hry zaměřené na vesty.



Obrázek 3.19: Návrh rozhraní pro zadávání hráčů do hry zaměřené na týmy.

- Možnost automatického férového rozřazení hráčů do týmů podle jejich předchozích výsledků.

Pokud konkrétní aréna využívá systém číslovaných stolů, například v kase (jako v kavárnách a restauracích), bylo by možné tyto skupiny přiřadit i na jednotlivé stoly a tím usnadnit komunikaci mezi několika operátory v jeden čas – „Můžeš mi připravit skupinu ze stolu 4?“

3.4.3 Napojení na rezervační systém

Rezervační systém je důležitou součástí všech arén. Tato funkce by měla za cíl umožnit zobrazení rezervací přímo v rozhraní aplikace pro zadávání her. Může též pomoci s identifikací skupin ze sekce 3.4.2. Arény nevyužívají jednotný systém rezervací, což bude komplikovat implementaci. Rezervační systém musí poskytovat nějaké API pro získávání záznamů. Řešením by samozřejmě bylo i vytvoření jednotného rezervačního systému v rámci on-line služeb aplikace, ale jelikož všechny arény už mají nějaký rezervační systém zažitý, přechod na úplně nový systém by mohl být poměrně komplikovaný.

3.4.4 Organizace turnajů

Turnaje se u nás v Česku, ale i v zahraničí, pořádají poměrně pravidelně. V době tvorby této práce neexistuje žádné ucelené řešení pro organizaci těchto turnajů. Cílem tohoto nástroje je propojit všechny části organizace turnaje od plánování, přes přihlášky týmů, po samotné hraní a výsledky. Funkce by měla umožnit velké množství variability nastavení turnajů, jelikož každý turnaj je unikátní a velmi záleží na zvoleném herním módu. Primárně jde tedy o automatizaci postupů, které aktuálně většina organizátorů manuálně zadává do excelové tabulky.

3.4.5 Nahlašování závad na vestách

Hardware vybavení není dokonalý a jeho závadovost je nevyhnutelná. Systém by měl umožnit operátorovi jednoduchým způsobem závadu přímo na vestu nahlásit. Aplikace poté se závadou na vestě počítá, zobrazí se varování všem uživatelům a při automatickém rozřazování vestu vynechá.

Každá vesta může být v jednom ze tří stavů: v pořádku, lehká závada (může hrát), těžká závada (nemůže do hry).

Kapitola 4

Implementace navrženého řešení

Tato kapitola pojednává o konkrétních problémech, rozhodnutích a celkové implementaci výsledné aplikace. Zaměří se převážně na řešené problémy a celý postup vývoje systému. Na tuto kapitolu volně navazuje kapitola 5, která se bude zabývat převážně uživatelským testováním částí popsaných v této kapitole.

4.1 Architektura systému

Vzhledem k výběru implementace jako webová aplikace jsem se rozhodl back-end aplikaci vytvořit v `PHP` (verze 8.1). Tento výběr je podpořen převážně mou vlastní zkušeností. Výhodou je jednoduchý vývoj, který umožňuje rychlé iterativní změny.

Pro aplikaci nebyl použit žádný hotový framework. Ačkoliv by použití nějakého frameworku výrazně ulehčilo začátek vývoje, většina moderních frameworků je velmi komplexní a tudíž by obsahovali funkce, které naše aplikace vůbec nepotřebuje. Navíc by určité specifické problémy mohli vyžadovat úpravu či obcházení některých funkcí frameworku, což by práci ještě více komplikovalo. Proto jsem vytvořil framework vlastní s využtím několika open-source knihoven (viz. 4.1.1).

Front-end část je tvořena co nejvíce nativními funkcemi s použitím minima knihoven. Jediná větší využitá knihovna je `Bootstrap`, která omezí nutnost tvořit vlastní `CSS`. Na kompilaci a sbalení všech prvků je použitý nástroj `Webpack`.

4.1.1 Seznam využitých technologií a knihoven

V této části popisuji nejzásadnější technologie a knihovny, které byli v projektu využity.

Back-end

- `Git`: verzování kódu a distribuce (viz. 4.2.3)
- `PHP 8.1`
- `Composer`[1]: Nástroj na správu PHP knihoven
- `Latte`[2]: šablonovací jazyk

[1] https://getcomposer.org
[2] https://latte.nette.org

- `Tracy`[3]: ladící nástroje
- `Dibi`[4]: databázová abstrakční knihovna
- `Nette utils`[5]: užitečné nástroje

Front-end

- `JavaScript`
- `npm`[6]: Nástroj na správu JavaScript knihoven
- `Bootstrap`[7]: Knihovna s připravenými prvky pro weby
- `SCSS`[8]: `CSS` preprocesor
- `Webpack`[9]: nástroj pro zabalení a kompilaci použitých prvků
- `Axios`[10]: JavaScript knihovna pro zjednodušení práce s `AJAX`
- `WebSocket`: Protokol pro komunikaci mezi serverem a klientem v reálném čase

4.2 Distribuce systému pomocí Docker kontejnerů

V této části se zaměřím na konkrétní řešení distribuce aplikace a konkrétní technologii – Docker.

Za starých časů potřebovala typicky každá aplikace jeden vlastní server. Dalším krokem byla virtualizace. V dnešní době využíváme kontejnery [19].

4.2.1 Princip kontejneru

Kontejner se v principu moc neliší od virtuálního stroje. Hlavním rozdílem mezi oběma technologiemi je hlavně to, že kontejner nepotřebuje svůj vlastní operační systém, ale využívá operačního systému svého hostitelského stroje. Právě i díky tomu se uvolní velké množství zdrojů, jako RAM, CPU a úložiště, které by potřeboval obyčejný virtuální stroj [19].

Virtualizace využívá rozdělování prostředků na úrovni hardware. Po bootování stroje se spustí `hypervizor`, který rozdělí fyzické zdroje (jádra procesoru, GB RAM,...) a spustí jednotlivé virtuální stroje, každý se svým operačním systémem a svými omezenými prostředky. Princip kontejnerů naopak rozděluje prostředky na úrovni hlavního operačního systému. Každý kontejner se sice tváří, že má vlastní operační systém, ale ve skutečnosti `engine` (nejčastěji Docker) jen rozděluje zdroje (souborový systém, procesy,...) operačního systému jednotlivým osamoceným kontejnerům. Tedy už jen z tohoto principu vyžaduje kontejner daleko méně fyzických zdrojů i přes jeho větší režii, protože si oproti virtualizaci nepřivlastňuje žádné zdroje, které žádný jiný virtuální stroj nevyužije.

[3] https://tracy.nette.org
[4] https://dibiphp.com
[5] https://doc.nette.org/cs/utils
[6] https://www.npmjs.com
[7] https://getbootstrap.com
[8] https://sass-lang.com
[9] https://webpack.js.org
[10] https://axios-http.com

Základ celého kontejneru je sestavený `image`. `Image` je ve svém principu takový balík, který zapouzdřuje vše potřebné pro danou aplikaci. Obsahuje kód aplikace, jednotlivé závislosti a nainstalované funkce operačního systému. Při vytváření vlastního `image` není nutné vše konfigurovat. Ve spoustě aplikacích lze využít již hotového řešení a případně si ho rozšířit o další potřebné funkce.

Kontejner je potom tedy jen spuštěná instance připraveného `image`. Je možné mít aktivní i více než jednu instanci stejného `image`.

4.2.2 Instalace a konfigurace

Nyní se zaměříme na samotné spuštění a instalaci aplikace v aréně. Zaměřím se převážně na konkrétní případ Laser arény Písek.

Prvním krokem celé instalace je nainstalovat Linux `backend` na kterém poběží celý Docker `engine`. Tento krok je nutný, jelikož hostitelský stroj, na který se aplikace bude instalovat má nainstalovaný operační systém Windows. Windows umožňuje instalaci technologie `WSL`, sloužící k integraci distribuce Linux do hlavního operačního systému. Samotná instalace je poměrně jednoduchá:

```
1    wsl --install
```

Stačí postupovat dle oficiální dokumentace Microsoft [14].

Dalším krokem je nainstalovat `Docker engine`. Zde se opět postupuje podle oficiální dokumentace [5].

Po kompletní instalaci všech potřebných nástrojů se zaměřuji přímo na konkrétní konfiguraci `image` výsledné aplikace. Projekt využívá celkem tři připravené obrazy:

1. `php:8.1.1-apache`[11]
2. `node:17.2.0-slim`[12]
3. `mariadb:latest`[13]

První dva se slučují do jednoho výsledného, kde běží samotná PHP aplikace. Důvod, proč je součastí kontejneru i `image node`, je potřeba sestavení zdrojů JavaScriptu i CSS pomocí Webpack. Jelikož zdrojový kód se distribuuje pomocí technologie Git, možnost sestavit nejnovější verzi přímo za běhu/při spuštění kontejneru je z hlediska praktičnosti vývoje ideální.

Třetí `image` běží paralelně a slouží pouze ke zprovoznění SQL databáze. Rozhodl jsem se pro distribuci MariaDB, protože už jí několik let využívám, je jedna z nejrozšířenějších volně dostupných řešení a nabízí další různé užitečné funkce, které obyčejná distribuce MySQL nenabízí (například `INTERSECT/EXCEPT`, `sharding`,...) [13].

Konkrétní konfigurace souborů pro sestavení Docker kontejneru je popsána v příloze B.

Obrazy používané kontejnery se sestavují v adresáři, kde se nachází soubory `Dockerfile` a `docker-compose.yml`, pomocí příkazu:

```
1    docker compose build
```

Kontejnery budou poté připravené ke spuštění v GUI Docker desktop aplikace, což je preferovaný způsob spouštění a monitorování běhu kontejnerů pro operační systém Windows.

[11] https://hub.docker.com/_/php?tab=description&page=1&name=8.1.1-apache
[12] https://hub.docker.com/_/node?tab=description&page=1&name=17.2.0-slim
[13] https://hub.docker.com/_/mariadb/

4.2.3 Aktualizace

V uzavřeném systému jako je Docker kontejner je jakákoliv změna zdrojového kódu poměrně náročná, jelikož vyžaduje opětovné sestavení `image`. Tento problém je ale možné lehce vyřešit s použitím verzovacího systémů `git` a sbalením sestavovacích scriptů přímo do samotného `image`. Při sestavení se celý zdrojový kód nekopíruje, ale stahuje se z veřejného repozitáře[14]. Při každém spuštění instance kontejneru se provede `pull` repozitáře a spustí se veškeré instalační scripty: `composer build`[15].

Alternativně framework umožňuje volání REST API, přes které je možné všechny nutné příkazy spouštět bez nutnosti restartovat celý kontejner.

4.2.4 Problémy instalace

Prvotní spuštění se samozřejmě neobešlo bez problémů a komplikací.

Jedním z velkých problémů bylo, že aplikace měla být spuštěna na jiném fyzickém zařízení, než systém výrobce LaserGame. Toto rozhodnutí bylo převážně způsobeno dvěma faktory. Vybrané zařízení má vyšší výpočetní výkon a nijak se neohrozí fungování hlavního počítače, bez kterého nelze v aréně hrát[16]. Díky tomu ale vznikl problém se sdílením souborů. Docker umožňuje sdílet adresáře z lokálního počítače do kontejneru. První pokus tedy byl, nasdílet adresář z počítače LaserGame na počítač aplikace protokolem `cifs` a následně napojit sdílený adresář z Windows do kontejneru. Toto řešení se ale ukázalo jako nefunkční. Bylo nutné napojit sdílený adresář pomocí protokolu `cifs` přímo do Linuxového kontejneru. Jenže toto řešení, ač funkční, se neobešlo bez svých vlastních komplikací. Počítače na sobě jsou nezávislé. Počítač aplikace byl spuštěný neustále, zato počítač LaserGame se vždy večer vypínal. Jakmile se vypl, sdílený adresář se od kontejneru odpojil a při opětovném zapnutí počítače se automaticky nepřipojil. Navíc, pokud by se kontejner restartoval v době, kdy je počítač LaserGame vypnutý, skončilo by jeho spuštění chybou. Tento problém jsem nakonec vyřešil implementací API přímo do aplikace. Na počítači LaserGame se po spuštění spouští jednoduchý PowerShell script, který posílá HTTP požadavek na REST API aplikace, která zavolá unixový příkaz `mount -a`, který opětovně připojí všechny sdílené disky. To samozřejmě neřeší problém restartu kontejneru při vypnutém počítači LaserGame. V průběhu tří měsíců, kdy aplikace funguje v Laser aréně Písek tento problém nastal jen jednou a to ve chvíli, kdy jsem já sám znovu sestavil `image` kontejneru před otevírací dobou arény. Tudíž, pro aktuální verzi systému, nepovažuji tento problém za příliš závažný.

Druhým problémem, který nastal, byla kontrola změn v souborech výsledků her. Chtěl jsem se vyhnout periodické kontrole souborů na úrovni aplikace a proto jsem hledal řešení, které může reagovat na notifikace operačního systému a volat definované scripty. Objevil jsem nástroj **Entr**[17], který sliboval odstranění přesně tohoto problému. Narazil jsem však na problém s funkčností u sdílených disků, kdy z nějakého důvodu notifikace operačního systému tato utilita nezachytávala. Rozhodl jsem se tedy ještě jednou jít cestou implementací přímo do API aplikace a jednoduchého PowerShell scriptu, který bude na počítači LaserGame sledovat změny přímo v systému Windows a posílat HTTP požadavky, jakmile detekuje jakoukoliv změnu. Toto řešení se ukázalo jako ideální, jelikož import výsledků pro-

[14] https://github.com/Heroyt/LaserArenaControl

[15] Zkratka, která volá další `composer` a `npm` scripty.

[16] Tento problém se týká konkrétně Laser arény Písek. V ostatních arénách se předpokládá využití jen jednoho počítače – LaserGame konzole.

[17] http://eradman.com/entrproject/

bíhá s minimální latencí[18] a zbytečně nevytěžuje žádné další zdroje, které by byli potřeba při periodické kontrole.

4.3 Principy vytvořeného PHP frameworku

Jak jsem již zmínil v kapitole 4.1, rozhodl jsem se nevyužít žádný hotový PHP framework, ale vytvořil jsem svůj vlastní na míru přizpůsobený všem požadavkům systému. V této kapitole se zaměřím na konkrétní prvky vytvořeného frameworku, jaké postupy využívá a co konkrétně jsem převzal z běžně využívaných řešení, se kterými jsem v minulosti pracoval.

4.3.1 Model MVC

`MVC` je jedním z nejpoužívanějších návrhových vzorů. Všechny velké webové frameworky[19] jsou postavené právě na tomto modelu, nebo na nějaké jeho variaci. Ve svém principu se týká pouze rozdělení částí aplikace na tři různé části [18]:

- `Model`: Všechno týkající se dat a práce s nimi.
- `View`: Prezentační vrstva, typicky HTML.
- `Controller`: Vrstva propojující obě předešlé. Stará se primárně o zpracování dat pro `view` a zpracování požadavků uživatele pro propis do `modelu`.

4.3.2 Architektura frameworku

Vytvořený framework se skládá z několika hlavních částí: jádro, controllery, šablony, služby a funkce pro práci se systémem LaserGame. O automatické načítání tříd se stará `composer autoloader`. Pro korektní funkčnost načítání tříd je třeba mít správně pojmenované adresáře i soubory tříd podle doporučení PSR-4 [11]. Konkrétně je nutné dodržovat následující pravidla:

- Každá třída je ve svém vlastním souboru
- Soubor je pojmenovaný stejně jako samotná třída
- Soubor třídy je v adresářové struktuře, která odpovídá `namespace` třídy. Základní `namespace App` je definovaný v souboru `composer.json` a odpovídá adresáři `src/`. Např. třída `\App\Core\Routing\Route` se nachází v adresáři `src/Core/Routing/`.

Adresářová struktura frameworku vypadá následovně:

```
assets/
    scss/
    js/
    images/
    icons/
config/    # konfigurační PHP soubory
include/   # ostatní nutné, pomocné scripty, které nejsou ve třídách
```

[18] Z pozorování import proběhne do tří sekund od uložení souboru, což je více než dostačující.
[19] Laravel, Symfony, Nette

```
languages/
private/   # privátní konfigurace (připojení na databázi,...)
routes/
src/       # veškeré zdrojové třídy
templates/ # latte šablony pro view
tests/
```

4.3.3 Třída App

V samém jádru celého frameworku leží třída `\App\Core\App`. Tato statická třída se stará o většinu funkcí nutných k fungování celé aplikace. Obsahuje všechny inicializační scripty pro načtení překladů, Latte enginu, cachovacího úložiště a načtení konfigurace ze souboru `private/config.ini`. Metoda `App::run()` je vstupní metodou celé aplikace.

Zároveň také obsahuje několik pomocných funkcí. Mezi nejvýznamnější funkce patří:

- `App::isProduction() : bool` – Kontroluje nastavení, zda aplikace běží v režimu `debug` nebo `production`.
- `App::redirect(`UrlRoute|array|string *url*, ?*Request*request = null) : bool| – Přesměrování na konkrétní URL, cestu, nebo route.
- `App::getLink(array $path) : string` – Vrací kompletní URL pro danou cestu.
- `App::getLogger() : Logger` – Vrací inicializovanou třídu pro logování.
- `App::getContainer() : Container` –Vrací dependency injection kontejner.
- `App::getService() : mixed` – Vrací objekt načtený pomocí dependency injection. Zkratka pro `App::getContainer()->getService($name)`.

4.3.4 Třída Info

Framework nabízí ukládání jednoduchých serializovatelných nastavení přímo do předem připravené databázové tabulky. Pro interakci s touto tabulkou slouží třída `\App\Core\Info`. Ta nabízí dvě statické metody přístupu: `get()` a `set()`. Tabulka v databázi obsahuje jen dva sloupce: primární klíč `VARCHAR ‘key‘` a `TEXT ‘value‘`. Hodnoty jsou serializované pomocí nativní PHP funkce `serialize()`. Lze tedy do databáze uložit jakoukoliv datovou PHP strukturu.

Třída info využívá statického asociativního pole pro zachování načtených hodnot v případě opakovaného čtení bez nutnosti opětovného dotazování databázi.

Kromě některých nastavení systému se pomocí této třídy ukládají i dočasné informace, jako například poslední načtená hra, nebo datum a čas posledního importu výsledků.

4.3.5 Třída DB

Pro komunikaci s databází jsem zvolil využití knihovny `dibi`[20]. Tuto knihovnu jsem vybral z důvodu její jednoduchosti při tvorbě dotazů pomocí `fluent rozhraní` [15], což je umožňuje velmi snadno vytvářet postupně za běhu. Další výhodou je využití rozsáhlých množství escapování výrazů a předcházení útoků typu `SQL Injection`.

[20] https://dibiphp.com

Třída DB slouží jako schránka nad knihovnou. Jedná se o čistě statický objekt obsahující aktivní instanci databázového připojení. Jelikož se v aplikaci nepočítá s možností připojení na několik databází, statický objekt umožňuje přístup z libovolného místa v kódu. Třída obsahuje několik základních metod pro všechny potřebné SQL dotazy (`SELECT`, `INSERT`, `UPDATE`, `REPLACE`, `DELETE`). Tyto metody buďto fungují sami o sobě, nebo vrací objekt `Dibi\Fluent`, se kterým se dá dále pracovat.

4.3.6 Dependency injection

Dependency injection je metoda automatického předávání informací pro funkce, která využívá dependency injection kenterneru. Kontejner se poté stará o samotnou inicializaci všech potřebných tříd a parametrů [16].

Zde framework využívá již pripravenou knihovnu z frameworku Nette: `nette/di`.

Konkrétních využití dependency injection je ve výsledné jen pár. V systému pro arény existuje jediná služba `LigaApi` pro komunikaci s veřejným API pro synchronizaci her.

4.3.7 Logování

Logování běhu je nedílnou součástí jakékoliv aplikace. Framework proto obsahuje třídu `\App\Logging\Logger`. Třída byla navržena, aby vyhovovala doporučením PSR-3 [2].

Instance třídy přijímá dva argumenty: cestu k adresáři, kam se mají logy ukládat a název logu (souboru). Pokud neexistuje vyžadovaný adresář, rekurzivně se vytvoří. Soubory logů se ukládají ve formátu `{název}-{rok}-{měsíc}-{den}.log`. V adresáři zůstávají vždy jen soubory za poslední dva dny, starší se ukládají do zip archivů rozdělených po týdnech.

Pro samotný zápis logů se využívá metod, které odpovídají jedné z osmi úrovní z RFC 5424: emergency, alert, critical, error, warning, notice, info, debug [7].

4.3.8 Routování

Základním údělem celého frameworku je routování. Routování je proces, jakým aplikace směruje cesty URL na funkce aplikace [18]. Bez použití nějakého routovacího scriptu by URL musela směrovat vždy na konkrétní PHP soubory, což v dnešní době již není velmi žádoucí.

Routování tohoto frameworku je postavené na dvou třídách: `\App\Core\Routing\Route` a `\App\Core\Request`[21].

Třída `\App\Core\Routing\Route` má na starosti uchovávání dostupných cest ve své statické struktuře `public static array $availableRoutes = [];`, kam se ukládají její instance. Tyto instance jsou vytvářené v souborech v adresáři `routes/`. `Route` obsahuje informaci o HTTP metodě[22], kterou využívá, URL cestě a metodě, kterou má volat. URL cesta může obsahovat i pojmenované parametry, které se volané metodě předávají.

`\App\Core\Request` se stará o zpracování požadavku, vyhledání správné routy a zavolání metody, která má požadavek zpracovat. Třída se předává jako volitelný argument volané metodě.

Definice `Route` může vypadat například takto:

```
1 <?php // routes/web.php
2
3 use \App\Core\Routing\Route;
```

[21] Framework využívá i některé odvozené třídy například pro zpracování CLI požadavků.
[22] Dostupné metody jsou: GET, POST, PUT, DELETE.

```php
use \App\Core\Request;
use App\Controllers\Results;

// Jednoduchá GET metoda
Route::get('/hello', function() {
    echo 'Hello, world!';
});

// POST metoda s jmenným parametrem {id}
Route::post('/form/{id}', function(Request $request) {
    // Musíme nastavit základní hodnotu v případě chybějícího parametru
    $id = (int) ($request->params['id'] ?? 0);

    // Další zpracovaní formuláře...
});

// Pojmenovaná GET route napojená na controller
Route::get('/resuts', [Results::class, 'show'])->name('results');
Route::get('/resuts/{game}', [Results::class, 'game'])->name('results-game');
```

Výpis 4.1: Příklady definice route.

4.3.9 Typy route

Pro účely této aplikace jsou ve frameworku definované tři různé typy `route` a `controllerů`.

Web route

Tento typ je obyčejná cesta k nějaké webové stránce, která se má zobrazit ve webovém prohlížeči uživateli. Primární vstup je tedy pomocí HTTP požadavků na jeden soubor `index.php`, který pomocí předem definovaných cest směruje požadavek na správný `controller` a jeho metodu.

REST API route

API route je typ, který uživatel nevyužívá přímo, ale komunikují s ním různé další části systému, případně i nabízí různé ladící nástroje, které umožňují správu celého systému vzdáleně.

Hlavní implementační rozdíl je ten, že `API controller` obsahuje rozhraní pro snazší návratové hodnoty, automatický převod odpovědi do JSON, automaticky přidává HTTP hlavičky a nastavuje daný HTTP kód odpovědi. Jinak, než tyto přidané funkce je typ totožný s obyčejnou route.

CLI route

Command line interface route – jak již z názvu vypovídá je typ cesty, která je přístupná jen pomocí terminálu. Hlavní výhodou je blízká implementace společně se všemi ostatními funkcemi frameworku. Přistupuje se přes `index.php` soubor a načítají se stejné inicializační scripty jako pro obyčejnou Web route, ale výstup a funkce jsou upravené pro rozhraní příkazové řádky.

Objekt `CliRoute` nabízí kromě základní definice cesty, názvu a volané metody i pomocné parametry jako popis funkce, argumenty, použití, případně i celý popis nápovědy. Díky tomu je možné na jednom místě popsat kompletní funkčnost k jednomu příkazu a v případě

chybně zadaných argumentů aplikace vrátí užitečné informace o konkrétním použití dané funkce.

`CliController` je rozšířen jen o metodu umožňující jednodušší výpis na standardní chybový výstup. Navíc je ve frameworku implementovaných několik pomocných tříd pro usnadnění práce s příkazovou řádkou, jako je například podpora barev.

4.3.10 Modely

Objekty modelů pracují s daty aplikace. Všechny modely dědí od jedné rodičovské třídy `AbstractModel`. Ta definuje rozhraní pomocí kterého mohou ostatní části kódu s třídami manipulovat.

Jeden typ modelu odpovídá jedné tabulce v databázi. Jeden řádek tedy odpovídá jedné instanci. Každá synovská třída musí ve své svých konstantách zadefinovat právě toto napojení na databázi.

- `TABLE`: název tabulky v databázi
- `PRIMARY_KEY`: název sloupce primárního klíče (nepodporuje složené primární klíče) – propíše se do proměnné `id`
- `DEFINITION`: pole obsahující všechny definované proměnné třídy a jejich nastavení – název klíče v poli odpovídá názvu sloupce, hodnota je pole
  - `validators`: pole názvu funkcí, které mají validovat hodnotu před uložením do databáze
  - `class`: pokud je hodnota nějaká třída, nebo `enum`, zde je uvedená

Modely mohou implementovat rozhraní `InsertExtendInterface`. To definuje dvě metody, které slouží pro práci ukládání podřazených tříd. Například třída `Player` obsahuje referenci na třídu `Game`. Tato reference je v databázi uložena jako sloupec `id_game`. Třída `Game` implementuje rozhraní `InsertExtendInterface`. Statická metoda `parseRow()` se pokusí z řádku z databáze vyhledat svůj primární klíč a v případě, že jej nalezne, vrací instanci třídy `Game`, kterou si třída `Player` ukládá. Po zavolání metody `save()` na `Player` se vyvolá metoda `addQueryData()` na instanci `Game`, která obráceně do řádku databáze svůj primární klíč vloží. Aby tento proces proběhl automaticky, musí třída `Player` ve své konstantě `DEFINITION` uvést parametr `class` (viz. výpis 4.2).

```
1  namespace App\GameModels\Game;
2
3  use App\Core\AbstractModel;
4
5  class Player extends AbstractModel {
6
7      public const TABLE = 'players';
8      public const PRIMARY_KEY = 'id_player';
9      public const DEFINITION = [
10         'game' => ['class' => Game::class],
11     ];
12
13     public Game $game;
14 }
15
16 class Game extends AbstractModel implements InsertExtendInterface {
```

```
17
18      public const TABLE = 'games';
19      public const PRIMARY_KEY = 'id_game';
20      public const DEFINITION = [
21          ...
22      ];
23
24      ...
25
26      public static function parseRow(Row $row) : ?InsertExtendInterface {
27          if (isset($row->{self::PRIMARY_KEY}) {
28              return self::get($row->{self::PRIMARY_KEY});
29          }
30          return null;
31      }
32
33    public function addQueryData(array &$data) : void {
34        $data[self::PRIMARY_KEY] = $this->id;
35    }
36 }
```

Výpis 4.2: Příklad minimální definice modelu Player a Game pro korektní funkčnost rozhraní InsertExtendInterface.

O načtení dat modelu se stará metoda `fetch()`, která se volá i v konstruktoru, pokud je mu předáno ID třídy. Metoda z databáze načte jeden řádek, který knihovna `dibi`[23] zpracuje a uloží do objektu `Row`. Přes všechny pole (sloupce tabulky) tohoto objektu se následně iteruje, kontroluje se existence vlastností v třídě a na konec se zavolá metoda `parseRow()` pro každou definovanou třídu implementující rozhraní `InsertExtendInterface`. V databázi jsou jednotlivé sloupce pojmenované ve `snake_case`. V PHP třídách mohou být také, ale preferuje se `camelCase` do kterého se v případě nutnosti názvy sloupců automaticky převádí.

Na konci listopadu 2021 vyšla verze PHP 8.1, která přinesla mimo jiné i podporu typů `enum` [17]. Projekt byl původně vyvíjen na verzi 8.0, ale rozhodl jsem se požadavky navýšit, právě z důvodu využití tohoto datového typu. `Enum` existuje i jako datový typ v MariaDB. Modely proto podporují automatický převod mezi hodnotami z databáze a definovaným PHP `enum`. Definice v modelu probíhá totožně jako u definice rozhraní `InsertExtendInterface` – stačí v konstantě `DEFINITION` uvést u dané vlastnost parametr `class` se jménem objektu `enum`. Objekt musí být definovaný jako takzvaný `Backed enum`[24]. To znamená, že ke každému klíči váže hodnotu, která odpovídá textu z databáze (viz. výpis 4.3).

```
1 enum Size : string {
2     case XL = 'extra-large';
3     case L = 'large';
4     case M = 'medium';
5     case S = 'small';
6     case XS = 'extra-small';
7 }
```

Výpis 4.3: Příklad jednoduchého typu enum

[23] https://dibiphp.com

[24] https://www.php.net/manual/en/language.enumerations.backed.php

Optimalizace instancinace tříd

Při běhu aplikace se může stát, že na jednu instanci bude potřebovat odkazovat více dalších objektů. Například jednoho hráče referencuje třída `Game` a `Team`. Tyto třídy znají ID hráče a dokáží podle něj vytvořit nový model. To znamená dva dotazy do databáze a dvojí zpracování toho samého řádku. Jelikož předpokládám, že tento případ bude nastávat poměrně často, obsahují všechny modely jedno statické asociativní pole, kde se instance ukládají. Pokud je třeba někde v kódu získat model podle ID, volá se místo konstruktoru statická metoda `get($id)`. Ta první zkusí vyhledat instanci v poli a pokud neexistuje, zavolá konstruktor a vytvoří novou.

Problém tohoto řešení může být lehké plýtvání paměti. Pokud by proces PHP byl dlouhý, instance třídy se pravděpodobně až do samotného konce drží v paměti, jelikož jsou stále uložené v poli a čekají na další vyvolání[25]. Vzhledem ke krátké životnosti procesů a relativně malému množství tříd se kterými pracuje toto řešení nevyvolává žádné větší komplikace.

4.3.11 Controllery

Controllery jsou v jádru celé aplikace. Dobrým pravidlem u každého controlleru je, aby se vždy věnoval pouze jedné problematice. Kód tak bude přehlednější a programátor vždy ví, kde nějakou funkci hledat.

Ve frameworku se všechny controllery nachází v `namespace \App\Controllers` a dědí od základní třídy `\App\Core\Controller`. Controller může vypadat například takto:

```
1  <?php // src/Controllers/MyController.php
2
3  namespace App\Controllers;
4
5  use App\Core\Controller;
6
7  class MyController extends Controller
8  {
9
10     // Titulek stránky
11   protected string $title = 'My page';
12   // Popisek stránky
13   protected string $description = 'This is a description';
14
15   public function show() : void {
16     echo 'Hello there!';
17   }
18
19 }
```

Výpis 4.4: Příklad jednoduchého controlleru.

Navíc může controller využívat metod z rodičovské třídy controller, které zjednoduší opakované procesy, jako je vykreslování Latte šablony.

```
1  class MyController extends Controller
2  {
3
4   public function show() : void {
5       // Pole $this->params se předává latte šabloně jako proměnné
6     $this->params['string'] = 'Hello!'; // V šabloně -> $string = 'Hello!'
```

[25] V posledních několika verzích PHP byl radikálně vylepšen optimalizátor a překlad do opcode. Je tedy možné, že se paměť korektně uvolní. Pravděpodobně však bude reference na třídy pořád existovat.

```
7
8     // Vykreslí šablonu /templates/pages/my/index.latte
9     $this->view('pages/my/index');
10  }
11
12  public function ajax() : never {
13      // Vrátí JSON data se správně nastavenými hlavičkami
14      $this->ajaxJson(['hello' => 'there!']);
15  }
16
17 }
```

Výpis 4.5: Příklad jednoduchého controlleru využívající metody rodičovské třídy pro jednodušší výpis.

4.3.12 View – Latte šablony

Údělem šablonovacích systémů je převod kousků HTML, PHP, případně dalších jazyků na výstup statického HTML [18]. Pro účely této práce jsem zvolil šablonovací systém Latte[26], jelikož mi ze všech šablonovacích systémů, které jsem za roky svojí práce vyzkoušel vyhovovalo úplně nejvíce.

Všechny šablony jsou uložené v souborech s příponou `.latte` v adresáři `templates/`. Pro obyčejné stránky je připravená šablony `@layout.latte`, která obsahuje všechno potřebné HTML a základní strukturu. Ostatní šablony definují jen konkrétní obsah, který se má zobrazit. Šablona stránky může vypadat například takto:

```
1 {* templates/pages/my/index.latte *}
2 {layout '../../@layout.latte'}
3
4 {block content}
5   <h2>Hello, there!</h2>
6 {/block}
```

Výpis 4.6: Příklad jednoduché šablony využívající šablony @layout.latte

Šablona `@layout.latte` nabízí mimo jiné i další bloky, které umožňují modifikace některých prvků šablony z podřazené šablony.

- `content`: Blok hlavního obsahu
- `errors`: Chybové hlášení těsně před obsahem
- `containerClasses`: HTML třídy (`class`), které se mají přidat na rodičovský HTML kontejner
- `addHead`: Blok, pro přidání obsahu do HTML hlavičky (`<head>`)

Další důležitou částí jsou nové značky, které přidávají funkčnost specifickou pro vytvořený framework. Tyto úpravy jsou automaticky generované ze souboru `config/latte.php`, kde jsou všechny značky zadefinované v jednom PHP poli.

- `{logo}`: Vykreslí logo arény jako `<img>`, nebo `<svg>`.

[26] https://latte.nette.org

- `{link [path..]}`: Volá metodu `\App\Core\App::getLink()`, která vrací úplný odkaz na danou cestu. Cesta se vkládá jako pole.
- `{getUrl}`: Vrací úplnou URL webu.
- `{csrf}`: Vygeneruje a vrátí CSRF token.
- `{csrfInput $name}`: Vygeneruje CSRF token a vrátí HTML `<input>`.
- `{lang ...}`: Volá funkci `lang()` pro překlad řetězců. Také dostupné jako filtr `|lang`.
- `{svgIcon $name, $width = '100%', $height = '100%'}`: Vrací `<svg>` ikony v adresáři `assets/icons/`.

4.4 Zpracování výsledků a napojení na systém LaserGame

Tato část pojednává o implementaci stěžejní části celé práce. Vytvořená aplikace obsahuje funkční napojení na výsledky ze systému LaserMaxx s přípravou univerzálních tříd na možné další rozšíření pro ostatní systémy LaserGame.

4.4.1 Architektura tříd

Všechny třídy související s propojením na systémem LaserGame se nachází v `namespace \App\GameModels`. Rozhodl jsem se tyto třídy vkládat do projektu jako git submodule [4, str. 303], jelikož toto řešení umožňuje pracovat na kódu nezávisle a řešení lze využít i v projektu pro veřejný web (kapitola 4.6).

Třídy jsou vytvořené jako obecné a případně abstraktní, ze kterých poté dědí třídy konkrétních systémů.

Modely výsledků

Nejzákladnější třídy jsou abstraktní modely obsahující data: `\App\GameModels\Game\Game`, `\App\GameModels\Game\Player` a `\App\GameModels\Game\Team`. Tyto třídy nabízí rozhraní přístupu k informacím z konkrétních výsledků. Jednotlivé implementace systémů LaserGame musí obsahovat synovské třídy k těmto třem abstraktním.

Dalším modelem výsledků jsou třídy herních módů. Tyto třídy dědí od rodičovské třídy `\App\GameModels\Game\GameModes\AbstractMode`. Modely herních módů mohou přepsat metodu `getWin()`, která definuje výherní podmínku pro tým/hráče. Pokud herní mód implementuje rozhraní `App\GameModels\Game\GameModes\CustomResultsMode`, může pomocí jeho metod: `getCustomResultsTemplate()` a `getCustomGateTemplate()` definovat i unikátní šablonu pro zobrazení výsledků na tisk a pro výsledkovou tabuli.

Kolekce modelů

Kolekce jsou speciální třídy seskupující několik instancí modelů. Jedná se primárně o třídy `PlayerCollection` a `TeamCollection`. Jejich rodičovskou třídou je `AbstractCollection`, která definuje rozhraní a implementuje základní operace jako přidávání objektů, iterátor nad objekty a podobné.

Velmi užitečné jsou také synovské třídy `AbstractCollectionQuery`. Tyto třídy poskytují rozhraní nad kolekcí pro filtrování, řazení a transformace objektů v kolekci. Využívá

takzvaného `fluent rozhraní` [15]. Metody se mohou větvit za zebe, jelikož každá metoda vrací svojí instanci (viz. výpis 4.4.1). Primárním účelem těchto tříd je zjednodušit práci s těmito datovými strukturami a poskytnutí rozhraní, které je podobné knihovně `dibi`[27] pro tvorbu SQL dotazů.

```
1  // Kolekce obsahující objekty Player
2  $collection = new PlayerCollection();
3  $collection->add(...);
4
5  $data = $collection
6            ->query() // Začátek query
7            ->filter('vest', 1, 2, 3) // Filtrovat pouze hráče s číslem vesty 1, 2, nebo 3
8            ->sortBy('score') // Řadit podle skóre
9            ->desc() // Sestupně
10           ->pluck('name') // Vrať pouze jména
11           ->get(); // Získej výsledek
12
13 // Proměnná $data obsahuje pole jmen hráčů (string)
```

Výpis 4.7: Příklad řetězení metod pro Query nad kolekcí.

Tovární třídy

„Polymorfismus a bázové třídy jsou skutečným základem objektově orientovaného programování (OOP). V určité fázi však musíte vytvořit konkrétní instanci potomka bázové třídy. Právě v této chvíli se vám náramně hodí vzor výrobní/tovární třídy.“ [8]

Továrny jsou speciálním typem tříd, jelikož sami o sobě slouží jen k vytváření jiných tříd. V případě tohoto projektu slouží tovární třídy ke korektnímu načítání tříd specifických pro daný systém LaserGame. Jelikož aplikace počítá s implementací více než jednoho systému (i na jednom serveru), je třeba vždy rozlišit například která třída `Game` je ta správná pro aktuální systém. Právě k tomu slouží tyto implementované objekty, které se nacházejí v namespace `\App\GameModels\Factory`.

Celkem se jedná o čtyři třídy, které se dají rozdělit do dvou kategorií: načítání objektů hry a načítání objektů herních módů.

Načítání objektů hry

V této kategorii se nachází továrny pro všechny tři objekty hry: `GameFactory`, `PlayerFactory`, `TeamFactory`. Nejpodstatnější je právě třída `GameFactory`, jelikož přes ni se primárně přistupuje i k ostatním objektům.

Třída implementuje několik statických metod pro vyhledávání objektů her:

- `getByCode($code)`: vrací konkrétní objekt hry podle jejího unikátního kódu
- `getById($id, $system)`: vrací konkrétní objekt hry podle jejího ID a systému LaserGame
- `getByDate($date)`: vrací objekty her, které se odehráli v daný den
- `queryGames()`: vrací objekt `Dibi\Fluent` s připraveným dotazem na databázi – spojení tabulek všech dostupných systémů LaserGame

[27] https://dibiphp.com

V principu načítání funguje poskládáním jednoho dotazu na databázi. V tomto dotazu jsou sloučené všechny tabulky jednotlivých systémů LaserGame.

Načítání objektů herních módů

Kategorie obsahuje jedinou továrnu: `GameModeFactory`. Ta slouží k získávání specifických objektů herních módů. Herní módy spadají vždy do jedné ze dvou kategorií: týmové a všichni proti všem. Tomu odpovídají vytvořené třídy: `Deathmach` a `TeamDeathmach`. V aplikaci se mohou vyskytnout i speciální módy s unikátními pravidly. Typicky tyto speciální módy dědí od jedné ze základních tříd. Všechny módy jsou uložené v databázi v tabulce `game_mode`. Jeden řádek nemusí nutně odpovídat jedné konkrétní třídě. Pokud třída neexistuje, využije se jedna ze základních.

Účelem továrny je tedy vyhledání správného herního módu podle systémového názvu ze souboru výsledků a v případě potřeby i vytvoření instance konkrétního objektu.

Implementuje několik statických metod:

- `getById($id)`: Vyhledání objektu herního módu podle ID
- `findByName($modeName, $modeType, $system)`: Vyhledání objektu herního módu podle názvu módu
- `find($modeName, $modeType, $system)`: Vyhledání objektu herního módu podle systémového názvu módu z výsledkového souboru

Vyhledávání tříd funguje na principu transformace názvu módu na název třídy. Z názvu se musí odstranit mezery a nahradit znaky, které nejsou ze sady `ASCII`. Pokud název módu začíná číslem (například mód „100 nábojů“), přidává se před název velké písmeno „M“. Poté se postupně kontroluje existence třídy a v případě, že správnou třídu nenalezne vrací `Deathmach`, nebo `TeamDeathmach`[28].

4.4.2 Parser dat systému LaserMaxx

Veškerá logika zpracování souboru s výsledky se nachází v parser třídě. Ta je zodpovědná za převod textových informací ze souboru do objektů, se kterými lze dále pracovat. Tyto parser třídy jsou specifické pro každý systém LaserGame. Všechny implementují rozhraní `ResultsParserInterface`, které definuje jednu metodu: `parse()`. Je možné také využít abstraktní třídy `AbstractResultsParser`, která implementuje metodu konstruktoru, načítající obsah daného souboru do `protected` vlastnosti.

Parser pro LaserMaxx zpracovává soubor po jednotlivých řádcích a využívá regulárních výrazů pro extrakci dat (výpis 4.8). Podle první zpracované skupiny (název kategorie) lze jedním dlouhým `switch` dekódovat každou kategorii separátně. Druhá skupina obsahuje všechny dostupné hodnoty oddělené čárkami. Ty se rozdělí do pole a očistí se o možné zbytečné mezery.

```
1 ([A-Z]+){([^{}]*)}#
```

Výpis 4.8: Regulární výraz pro zpracování jednoho řádku souboru s výsledky.

[28] Herní módy mají v databázi zadefinovaný svůj typ: SOLO/TEAM.

Samotné zpracování hodnot probíhá vytvářením objektů modelů her a přiřazování korektních hodnot. Využívá se jasně definovaného pořadí všech kategorií. Díky tomu by nemělo nastat, že se bude zpracovávat řádek `PACKX` před `PACK`. Pokud by se řádky prohodili, neexistovala by potřebná třída `Player` a parser by vyvolal vyjímku.

Po zpracování všech řádků souboru proběhne poslední proces: párování objektů hráčů a týmů. Během zpracování se do objektu hráčů ukládá pouze číslo týmu, dle kterého se poté vyhledává správný objekt.

4.4.3 Zobrazování výsledků

Tato část pojednává o konkrétní implementaci zobrazování a tisku výsledků, což je aktuálně nejpodstatnější část hotové aplikace.

Výsledková tabule

Výsledková tabule je to první, co hráč uvidí po konci hry. Zobrazuje se na celé obrazovce v samostatném okně prohlížeče. Chod by měl být co nejvíce automatický – okno prohlížeče se otevře společně se spuštěním počítače (ideálně rovnou na celou obrazovku) a pracovník arény by s ním neměl v průběhu celého dne jakkoliv manipulovat. První z požadavků musí být nastavený přímo v operačním systému Windows.

Tabule má celkem tři různé typy zobrazení (viz. kapitola 3.3.2). Mezi těmito zobrazeními se musí přepínat automaticky a s minimálním zpožděním. K tomu poslouží implementovaný event server (kapitola 4.5). Aplikace čeká na zprávy: `game-imported`, `game-started`, `game-loaded`, `gate-reload`. Pokud jednu ze zpráv přijme, načítá nový obsah pomocí AJAX požadavku. Ten směřuje na stejnou URL jako je tabule samotná[29]. Volá se tedy stejný controller a používá se stejná logika. Jediný rozdíl je v HTTP hlavičkách. Klient posílá hlavičku `X-Requested-With`, díky které server rozpozná, že se jedná o AJAX požadavek a vrací tedy jen HTML obsahu místo celé stránky. V odpovědi server přikládá vlastní hlavičku `X-Reload-Time`, která definuje čas v sekundách k opětovné aktualizaci obsahu. Toto umožňuje variabilně nastavovat, jakou dobu má být jaké zobrazení viditelné[30].

Jakou obrazovku zobrazit zjišťuje controller na základě informací ze systému. Ty jsou uložené pomocí nástroje `Info` (kapitola 4.3.4). Ukládá je služba importu, pokud detekuje hru, která je nahraná, nebo spuštěná, ale nedokončená. Spolu s třídou hry ukládá i časovou značku nahrání/spuštění hry.

Controller kontroluje obrazovky v následujícím pořadí:

1. Manuálně nastavené výsledky hry k zobrazení
2. Výsledky z poslední dokončené hry
3. Aktuálně spuštěnou hru (zobrazení průběžných výsledků s časovačem)
4. Aktuálně načtenou hru (zobrazení vest)
5. Jinak zobrazuje průběžné výsledky dne

[29] http://la.local:1234/gate

[30] Výsledky hry: 2 minuty, rozložení vest: maximálně 5 minut.

V kapitole 4.4.1 jsem zmínil, že zobrazení výsledků se může lišit i na základě herního módu pokud implementuje rozhraní `CustomResultsMode`. V takovém případě se volá metoda herního módu, který vrací název latte šablony. Pro ostatní módy se použije šablona `templates/pages/gate/results.latte`.

Tisknuté výsledky

Generování tisknutých výsledků probíhá na vyžádání (stisk tlačítka). Tato varianta umožňuje oproti jiným typům zobrazení navíc i některé vlastní nastavení. Krom různých šablon výsledků (viz. kapitola 3.3.3) nabízí rozhraní tisku i nastavení stylu, jazyka, počtu kopií a režimu černobílého tisku. Všechny tyto nastavení jsou předávány v URL.

Styl tisku ovlivňuje barvy a pozadí. Toto vzniklo jako požadavek od Laser arény Písek. Barvy se definují v nastavení aplikace. Styly mohou být například: vánoční, novoroční, čertovské, zimní... Každý jednotlivý styl má definovaná data, kdy se má automaticky vybírat.

V případě černobílého tisku jsou barevné indikátory, jako název týmů, na výsledcích nahrazené textem[31].

Výsledky negenerují PDF, ale HTML stránku. O převod do PDF pro tisk se stará prohlížeč. V aktuální verzi jsou tiskové styly optimalizované pro Chromium. V budoucích verzích je možné zakomponovat do aplikačního stacku i bezstavové API **Gotenberg**[32]. To by sloužilo k automatickému převodu výsledků do PDF pro maximální kompatibilitu mezi prohlížeči.

Latte šablony se nacházejí v adresáři `templates/results/templates`. Každá varianta (popsané v kapitole 3.3.3) používá právě jednu latte šablonu, která je pojmenovaná podle vlastnosti `slug`. Stejně jako u výsledkové tabule, i zde je možné definovat unikátní šablonu výsledků pro herní mód implementující rozhraní `CustomResultsMode` (viz. kapitola 4.4.1).

4.5 Aktualizace informací v reálném čase

Jednou, velmi užitečnou součástí byl i návrh systému, který umožňuje klientům aplikace (prohlížeč) okamžitě reagovat na jakékoliv změny, které nastanou (například import nových výsledků). Díky tomu bude celá aplikace působit daleko více responsivní a interaktivnější. Minimalizují se prodlevy mezi akcemi a následnými reakcemi. Na straně klienta jde toto řešit primárně třemi způsoby:

1. Nijak – Uživatel musí stránku vždy manuálně načíst
2. Periodickým dotazováním serveru – AJAX
3. Nějakým protokolem umožňující komunikaci v reálném čase ze serveru na klienta

Varianta 3 je rozhodně nejlepší řešení. Pro webové aplikace existuje protokol `WebSocket`. Tento protokol umožňuje obousměrnou komunikaci postavenou na protokolu `HTTP`, `TCP` mezi serverem a klientem. Samotnou komunikaci zahajuje `handshake` následovaný jednoduchými zprávami [6]. Protokol umožňuje přenos binárních dat (například obrázků) a textu. Pro účely této práce stačí předávat jednoduché textové zprávy bez nutnosti jakkoliv zprávy serializovat. Zpráva bude obsahovat pouze název události, která nastala a klient si sám rozhodne jaké akce budou dále následovat.

[31] V době psaní této práce se týká pouze tabulkové varianty.
[32] https://gotenberg.dev

4.5.1 Implementace WebSocket klienta

Implementace klienta je velmi jednoduchá a pomocí nativních tříd JavaScriptu se samotná komunikaci inicializuje jedním řádkem kódu a následná komunikace probíhá pomocí jedné `callback` funkce.

Pro účely jednoduché znovupoužití jsem vytvořil třídu `EventServer`, která obsahuje logiku veškeré komunikace. Třída spravuje připojení k serveru a ukládá si `callback` funkce, které se volají po přijetí jednotlivých zpráv. Tyto funkce jsou ukládané v objektu polí, kdy klíčem je právě název té události na kterou má funkce reagovat. Následné scripty se jen registrují pomocí metody `addEventListener()`. Je možné přiřadit jednu `callback` funkci i k více událostem.

```
1  EventServerInstance.addEventListener("game-imported", triggerNewGame);
2  EventServerInstance.addEventListener(
3      [
4          'game-started',
5          'game-imported',
6          'game-loaded'
7      ],
8      function() {
9          alert('Game changed!');
10     }
11 );
```

Výpis 4.9: Příklad registrace callback funkce pro třídu EventServer

Při implementaci bylo nutné počítat i s možným odpojením od serveru. Klient na to musí být připravený a v případě nutnosti se znovu připojit. Řešení je opět poměrně jednoduché. Stačí znovu vyvolat pokus o připojení na `WebSocket` objektu po událostech: `.onclose()` a `.onerror()`.

4.5.2 Implementace WebSocket serveru v PHP

Již z nadpisu této sekce si může neznalý programátor začít ťukat na čelo. Implementace WebSocket serveru v PHP sice není tak jednoduchá, ale díky svém základu v jazyce C celá implementace připomíná spíše typický TCP servery právě v C. Využívají se stejně pojmenované funkce knihovny BSD Sockets: `socket_select()`, `socket_bind()`, `socket_listen()`,... Pro získávání zpráv k rozesláním klientům server využívá jedné tabulky v databázi, na kterou se pravidelně dotazuje[33]. Tento postup má jednu velikou výhodu. Do tabulky v databázi může zprávy vkládat jakákoliv část aplikace. Například při importu výsledků script po dokončení vloží do databáze zprávu `game-imported` a na to WebSocket server reaguje a rozesílá zprávu všem klientům, kteří na ní reagují dle potřeby.

Princip celého serveru je celkem přímočarý:

1. Vytvoř BSD schránku – `socket_create()`
2. Začni poslouchat na daném portu – `socket_bind()` a `socket_listen()`
3. Čekej maximálně jednu sekundu na novou zprávu
4. Při připojení nového klienta odpověz na `handshake`, zahaj tak spojení a ulož schránku klienta do pole všech připojených klientů

[33] Database polling

5. Zkontroluj všechny připojené klienty, pokud neposlali nějakou zprávu
6. Pokud poslali, přepošli zprávu všem ostatním připojeným klientům (= broadcast)
7. Pokud klient ukončil připojení, zavři jeho schránku a odstraň jí z pole připojených klientů – `socket_close()`
8. Pomocí `SQL SELECT` příkazu, zkontroluj nové záznamy v databázi
9. Pro každý řádek z dotazu rozešli jeho zprávu všem připojeným klientům (= broadcast) a označ odeslané zprávy v SQL tabulce
10. Opakuj od bodu 3

V rámci celého PHP frameworku je tento server implementovaný jako `CLI Route` – viz. kapitola 4.3.9. V kontejneru se spouští paralelně s Apache serverem na portu `9999`.

V rámci vývoje testování výsledného řešení jsem ale narazil na jeden problém. Pokud WebSocket server běžel neustále už velmi dlouho (několik dní), mohlo se stát, že došlo k nějaké chybě a celý server se ukončil. Je téměř nemožné napsat dokonalou aplikaci [9]. Je tedy nezbytné, aby se server dokázal sám po ukončení znovu restartovat. V PHP je možné pomocí `register_shutdown_function()` registrovat funkci, která se provede těsně před ukončením programu. Navíc program může reagovat na signály (`SIGTERM`, `SIGHUP` `SIGINT`) operačního systému pomocí funkce `pcntl_signal()`. Na všechny tyto signály může reagovat jediná funkce, která korektně uzavře všechny otevřené schránky a pomocí `pcntl_exec()` spustí znovu celý script.

4.6 Veřejná část aplikace, on-line výsledky

V této sekci popisuji implementaci veřejného webu `laserliga.cz`, pomocí kterého mají hráči možnost zobrazovat si své výsledky on-line.

4.6.1 REST API

Primárním účelem API je synchronizace výsledků z lokálních aplikací v arénách na jeden centralizovaný server přístupný veřejně na webu. Pojmem REST se označuje taková služba, která splňuje architektonické omezení, která kladou důraz na rozšiřitelnost, obecná rozhraní, nezávislost distribuce jednotlivých komponent za účelem omezit latenci, zvýšit bezpečnost a zapouzdřit jednotlivé systémy [1].

Implementace na straně veřejného serveru

API se zásadně neliší od klasické implementace webu. Veškeré koncové body jsou definované jako route objekty, které jsou vázané na controllery. Tyto controllery vracejí typicky JSON objekt. Pro autentizaci požadavků se využívá tříd `Middleware`, které slouží ke kontrole požadavku před zavoláním controlleru. V případě, že použitá třída (konkrétně se jedná o třídu `ApiToken`) detekuje nějaký problém, korektním způsobem nastaví hlavičky, kód odpovědi a vrací hlášení o chybě.

Funkce v controllerech samotných nejsou příliš komplikované. V principu se jedná jen o zpracování příchozích JSON dat a převod na PHP objekty, nebo naopak jen získání informací z objektů a serializace do formátu JSON.

Rozhraní pro komunikaci s API v lokální aplikaci arény

Pro komunikaci s REST API je vytvořen jediný servis `\App\Services\LigaApi`. Tato třída využívá návrhového vzoru `singleton`. Tento vzor se liší od statických tříd především tím, že je instanciovaný, ale existuje jen jedna instance pro celou aplikaci [8]. Objekt lze získat pomocí jeho statické metody `getInstance()`, nebo pomocí dependency injection (viz. kapitola 4.3.6).

Třída má za úkol synchronizaci her a autentizaci. V budoucnu třída bude obsahovat navíc funkce pro stažení informací o registrovaných hráčích.

4.6.2 On-line výsledky

Zobrazení on-line výsledků se příliš neliší od generování klasických výsledků pro tisk. Hlavním rozdílem celého zobrazení je interaktivita a více detailních informací. Nedílnou součástí celého návrhu a implementace byla i nutná responsivita.

Rozložení využívá CSS funkce `grid`, které umožňuje jednoduchým způsobem definovat dvourozměrné[34] umístění veškerých synovských prvků. Díky tomu je možné mít na mobilním telefonu prvky v jiném pořadí než na tabletu a počítači s čtyřiceti palcovým monitorem.

Interaktivitu zajišťuje převážně knihovna **Bootstrap** a její nástroje `collapse` a `modal`.

[34] Řádky a sloupce.

Kapitola 5

Testování řešení

Kapitola pojednává o metodikách a výsledcích testování aplikace a změnách, které byly na základě zjištěných dat provedeny. Testování probíhalo převážně na reálných datech, hráčích a operátorech arén LaserGame. Kapitola je rozdělená na část testování funkčnosti aplikace jako takové z pohledu implementovaných algoritmů a dvě části uživatelského testování podle typu testovaného uživatele.

5.1 Testování funkčnosti systému

Z funkčního hlediska byli převážně testovány části frameworku a funkce pro práci s třídami LaserGame. Pro automatické PHP testy byla využita knihovna **PHPunit**[1].

Velký důraz byl kladen převážně na testování zpracování souborů výsledků a import/export objektů hry pro účely komunikace s REST API.

5.2 Testování operátory arén

Testování operátory arén probíhalo ve třech fázích.

1. Testování na grafických návrzích
2. Testování na prototypech
3. Testování v ostrém provozu

5.2.1 Testování na grafických návrzích

V této části jsem komunikoval s pracovníky tří různých arén[2]. Testování zúčastnilo 6 lidí, kteří měli zkušenost se systémem LaserGame používaný u nich v aréně[3].

První fáze

Jako první byli dotázaným položeny otázky týkající se jejich požadavků a požadavků, které pozorují běžně u hráčů. Tyto otázky a nejčastější odpovědi jsou uvedeny v tabulce 5.1.

[1] https://phpunit.readthedocs.io/en/stable/
[2] Laser aréna Písek, Lasergame Říčany, FunSpace ČB
[3] LaserMaxx a Laserforce

Otázka	Odpovědi
Jaké operace provádíte s aplikací systému LaserGame nejčastěji?	Načítání nových her, tisk výsledků
Jaké operace provádíte s aplikací systému LaserGame nejméně často?	Změna nastavení herních módů
Co Vás při práci s aktuálním systémem nejvíce zdržuje?	Tisknutí výsledků (přepínání mezi více okny), zdlouhavé vyhledávání předchozích her
Hrají v aréně spíše začátečníci, nebo pokročilí se hráči?	Pokročilých hráčů je mnohem méně než obyčejných začátečníků
Jak často hrajete Vy sám/sama?	Čtyři z dotázaných hrají pravidelně, ostatní jen občas
Čeho si hráči všímají na výsledcích nejvíce?	Většinou jen pořadí, skóre, případně přesnost
Jaká věková skupina hráčů je u Vás v aréně nejčastější?	Děti a studenti

Tabulka 5.1: Přehled odpovědí na otázky týkajících požadavků operátorů arén a hráčů

Druhá fáze

V další fázi byli dotázaným představeny grafické návrhy aplikace z kapitoly 3 a byli požádáni o co nejdetailnější popis.

V návrhu výsledkové tabule byli všichni schopni detailně popsat celý návrh všech tří možných obrazovek.

U návrhu tištěných výsledků se objevilo několik problémů, které byli postupně opraveny. Jednalo se o:

- V obou typech výsledků chyběl popis u QR kódu v pravém horním rohu.
- V grafické variantě nebyli jasné všechny ikony. Na základě toho byla do spodní části výsledků přidána legenda.
- Není úplně na první pohled jasné, co znamenají vybarvené a nevybarvené nábojnice.

Při popisování on-line výsledků se objevili následující problémy:

- Klikatelné prvky na detailu hráče (tlačítko pro zobrazení všech trofejí a sekce „Dnešní pořadí podle“) nebyli na první pohled zřejmé, že se dají otevřít. Na základě toho byli změněny popisky u tlačítek.
- Není úplně na první pohled jasné, co znamenají vybarvené a nevybarvené nábojnice.

V návrhu rozhraní pro zadávání hráčů byli vzneseny následující výhrady:

- V rozhraní zaměřeném na týmy chybí možnost manuálně vybrat konkrétní vestu hráče v případě nutnosti.
- V rozhraní zaměřeném na týmy by bylo dobré zakomponovat podobnou funkci, jako je v druhém rozhraní – nahlašování závad na vesty.

- Při tisku tlačítkem v pravém horním rohu by bylo dobré mít možnost navolit jazyk a počet tisknutých kopií.
- V rozhraní zaměřeném na vesty chybí možnost pojmenovat týmy.

5.2.2 Testování na prototypech

Po fázi testování na grafických návrzích byly vytvořeny klikatelné prototypy pomocí aplikace **Adobe XD** a k tomu již byli implementované základní funkce pro tisk. Toto bylo prezentováno třem různým pracovníkům, z Laser arény Písek[4]. Ti dostali předem připravený seznam úkolů, které postupně zkoušeli plnit a komentovat celý svůj proces. Úkoly a průběh testování je shrnut v tabulce 5.2.

5.2.3 Testování za provozu

Po spuštění v Laser aréně Písek byli operátoři požádáni o aktivní využívání aplikace a o reportování všech nedostatků a nových nápadů, které by jim práci usnadnili. Většina nedostatků byla rovnou opravena.

Nejzásadnější hlášené chyby a požadavky:

- Časovač hry na výsledkové tabuli ne vždy odpovídá reálnému času. Někdy při synchronizaci jde časovač napřed.
- Při stažení nových výsledků je nutné, aby se stránka s tiskem znovu načetla a zobrazila poslední výsledky.
- Občas se stává, že se výsledky nezobrazují na výsledkové tabuli. Tato chyba souvisí s problémem restartu event serveru, který je popsán v kapitole 4.5.2.
- Přidat nastavení minimálního času pro zobrazení odpočtu právě probíhající hry. Například při patnácti minutové hře zobrazit časovač až v posledních deseti minutách.

5.3 Testování hráči

Testování samotnými hráči se zaměřovalo pouze na podobu a detailnost výsledků. V této kapitole je popsáno osobní testování s konkrétními hráči. Pro účely práce byl vytvořen dotazník, který vyplnilo celkem 60 dobrovolníků a je popsán v příloze C.

Pro hráče byli připraveny různé typy tištěných výsledků[5]. Jejich úkolem bylo popsat jednotlivé výsledky, jakých prvků si všímají nejvíce, co se jim líbí a naopak nelíbí. Testování se zúčastnilo celkem jedenáct hráčů, z toho čtyři děti ve věku deset, dvanáct, třináct a patnáct let.

5.3.1 Výsledky testování

Největší kritiku sklidily výsledky LaserMaxx (obrázek 2.4). Všichni dotázaní hráči uvedli, že jsou nejméně detailní, vzhledově neatraktivní a nudné. Většina z hráčů si vůbec nevšimla

[4] Jeden z nich se účastnil předchozího testování na grafických návrzích.
[5] LaserMaxx, Laserforce, grafická varianta nových výsledků, tabulková varianta nových výsledků.

týmového skóre. Hráči kritizovali i neexistující překlad do češtiny. Naopak ocenili dobrou čitelnost a přehlednost hlavičky a pravého sloupce.

Výsledky Laserforce (obrázek 2.11) byli jako druhé nejvíce kritizované. Jejich barevnost sice oproti LaserMaxx pomáhá zaujmout, ale působí zmateně a neprofesionálně. Na první pohled zaujme obrázek vesty ve levém dolním rohu, ale dotázaní uvedli, že jim informace zásahů jednotlivých čidel přijde zbytečná. Hráči si navíc stěžovali na nesrozumitelné pořadí týmů a hráčů (výherní tým byl v pravém dolním rohu, druhý uprostřed, třetí nahoře a poslední v levém dolním rohu).

Tabulková varianta výsledků (obrázek 3.12) kritizovaná tolik nebyla. Převážně starší hráči ocenili detailnost a přehlednost tabulkového zobrazení. Tři dospělí hráči uvedli, že tuto variantu preferují. Mladším se v porovnáním s grafickou variantou tento typ moc nelíbil, ale nikdo neměl problém se zorientovat.

Grafická varianta (obrázky 3.14 a 3.13) byla všemi hráči označena za graficky nejatraktivnější. Ačkoliv obsahuje méně informací než tabulková, většina hráčů uvedla, že varianta působí detailně. Převážně hráči chválili znázornění poměrů – hlavně skóre a procentuální přesnost. Co naopak dva z hráčů zmínili je fakt, že si nejsou jistí významem vybarvených a nevybarvených nábojnic.

5.3.2 Vyhodnocení

Z odpovědí dotázaných hráčů lze zcela jasně vydedukovat, že nově navržené výsledky jsou výrazným vylepšením proti oběma původním variantám. Potvrdila se i původní předpověď, že méně náročí hráči a děti upřednostňují variantu grafickou. Pokročilí, dospělí hráči tabulkovou.

Na původních výsledcích není žádný prvek, který by na nových chyběl.

V budoucnu je žádoucí výsledky iterativně vylepšovat a testování opakovat, ideálně na větší skupině hráčů, ale i lidí, kteří do kontaktu s LaserGame ještě nikdy nepřišli. Například je vhodné zaměřit se na problémové nábojnice, navrhnout nové pozadí a implementovat proměnné styly u grafické varianty stejně jako u tabulkové.

Úkol	**Problémy**	**Komentář**
Připravte následující hry v rozhraní orientovaném na hráče. (dotázaným byl předán seznam tří různých her s jmény hráčů, vyžadovaným herním a hudebním módem)	Bez problémů	Žádný z dotázaných neměl problém zorientovat se v rozhraní a rovnou intuitivně zadávali všechny informace. Testovaní zároveň ocenili výběr týmu formou výběru z barev, místo neustálého klikání, na které jsou zvyklí ze systému LaserMaxx. Překvapilo mě, že nikdo z dotázaných neměl problém pochopit význam tlačítka pro výběr hráčské úrovně.
Zadejte stejné hry v rozhraní orientovaném na týmy.	Dotázaným chvíli trvalo, než přišli na to, že hráče do týmů přetahují	Úkol byl velmi podobný prvnímu.
Použijte předchozí odehranou hru jako šablonu pro novou.	Bez problémů	Dotázaným se líbila změna, že nahrání předchozí hry je přímo na obrazovce pro zadávání her.
Na libovolnou vestu zadejte závadu.	Dotázaní byli trochu zmatení, jelikož nikde neviděli žádné tlačítko, ale nakonec všichni úkol úspěšně splnili.	Dotázaní si novou funkci velice chválí. Původní zmatení je pochopitelné, jelikož funkce není nijak výrazně označena.
Vytiskněte aktuální výsledky hry.	Bez problémů	Dotázaní se rychle zorientovali a byli schopni popsat co jaká tlačítka dělají.
Vytiskněte libovolnou hru z minulého týdne.	Hráčům chvíli trvalo, než si uvědomili, že v menu se nachází položka „Seznam her“.	
Zobrazte výsledky libovolné hry ze seznamu na výsledkové tabuli.	Bez problémů	Po chvíli orientace všichni úspěšně nalezli příslušné tlačítko.

Tabulka 5.2: Úkoly a výstup z testování operátory arén na prototypech

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a implementovat webovou aplikaci pro sjednocení výsledků LaserGame. Jedná se o první krok k dosažení mé dlouholeté vize posunutí tohoto novodobého sportu na novou (celorepublikovou) úroveň.

Aplikace byla navrhována pro systém LaserMaxx a testována primárně v Laser aréně Písek. Od úplného počátku vývoje se používali reálná data her, řešili se opravdové problémy.

Podařilo se vytvořit takový projekt, který je v době psaní aktivně využíván ve dvou různých arénách LaserGame. Poskytuje nové možnosti zobrazení výsledků, které jsou detailnější, přehlednější a graficky atraktivnější, než jejich předchůdci. Zároveň systém umožňuje seskupení výsledků z různých arén na jednom centrálním serveru, kde jsou dostupné pro hráče. Podařilo se navrhnout a implementovat robustní framework a vytvořit všechny funkce potřebné k zobrazování statistik systému LaserMaxx.

Hotový systém splňuje všechny základní požadavky. Původní plán se sice nepodařilo naplnit na 100% v termínu, ale z obou arén, které aplikaci využívají, jsou na změny velmi pozitivní reakce od hráčů i od pracovníků arény.

V aktuální verzi má aplikace potenciální dosah přibližně 52% všech arén v České a Slovenské republice.

Díky práci jsem dokázal prohloubit své osobní znalosti v oblasti návrhu webových aplikací. Rozhodnutí vytvořit vlastní framework místo použití nějakého hotového řešení mě naučilo chápat komplexnější problémy. Běžný programátor tvořící webovou aplikaci se tímto nemusí vždy zabývat.

Aplikace nabízí stabilní základ pro možné budoucí rozšíření, jak z hlediska funkcí užitečných pro arénu, tak pro hráče jako takové. Mezi tyto rozšíření patří například nové rozhraní pro zadávání her a organizace turnajů, které plánuji implementovat v následujících pár měsících. Hráče naopak potěší rozšíření, které umožní registrace vlastních účtů, sbírání svých souhrnných statistik a účast v globálním žebříčku. Výhledově by aplikace mohla příští rok poskytovat funkční implementaci pro minimálně jeden další systém LaserGame.

Literatura

[1] Amundsen, M. *Restful Web API Patterns and Practices Cookbook*. 1. vyd. O'Reilly Media, Inc., 2022. 400 s. ISBN 978-1098106744.

[2] Boggiano, J. *PSR-3: Logger Interface* [online]. PHP Recomendation 4. PHP Framework Interoperability Group, March 2020. Dostupné z: https://www.php-fig.org/psr/psr-3/.

[3] Cao, J., Zieba, K., Stryjewski, K. a Ellis, M. *Elegant Web UI Design Techniques: Flat Design & Colors*. 1. vyd. UXPin Inc., 2015. https://www.uxpin.com/studio/ebooks/web-ui-design-techniques-colors-flat-design/(navštíveno 2022-04-19).

[4] Chacon, S. a Straub, B. *Pro Git: Everything you need to know about git*. 2. vyd. Apress Berkeley, CA, 2014. 419 s. ISBN 978-1-4842-0076-6.

[5] Docker. *Install Docker Desktop on Windows* [online]. 2022 [cit. 2022-04-28]. Dostupné z: https://docs.docker.com/desktop/windows/install/.

[6] Fette, I. a Melnikov, A. *The WebSocket Protocol* [Internet Requests for Comments]. RFC 6455. RFC Editor, December 2011. 1-71 s. Dostupné z: https://www.rfc-editor.org/rfc/rfc6455.txt.

[7] Gerhards, R. *The Syslog Protocol* [RFC 5424]. RFC 5424. Mar 2009. 38 s. Dostupné z: https://www.rfc-editor.org/info/rfc5424.

[8] Gutmans, A., Bakken, S. S. a Rethans, D. *Mistrovství v PHP 5*. 2. vyd. Computer press, a.s., 2008. 118-122 s. ISBN 978-80-251-1519-0.

[9] Howard, G. D. *Why Perfect Software Is Nearly Impossible* [online]. August 2019 [cit. 2022-05-02]. Dostupné z: https://gavinhoward.com/2019/08/why-perfect-software-is-nearly-impossible/.

[10] Jones, M. a Hardt, D. *The OAuth 2.0 Authorization Framework: Bearer Token Usage* [RFC 6750]. RFC 6750. Oct 2012. 18 s. Dostupné z: https://www.rfc-editor.org/info/rfc6750.

[11] Jones, P. M. *PSR-4: Autoloader* [online]. PHP Recomendation 4. PHP Framework Interoperability Group, December 2013. Dostupné z: https://www.php-fig.org/psr/psr-4/.

[12] Krug, S. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. 3. vyd. New Riders, 2014. ISBN 978-0-321-96551-6.

[13] MariaDB. *MariaDB vs. MySQL: Guide to Open Source Database Selection*. Září 2020 [cit. 2022-05-02]. 18 s. Dostupné z: https://go.mariadb.com/GLBL-WC2020MariaDBvs.MySQL_LP-Registration.html.

[14] Microsoft. *Windows Subsystem for Linux Documentation* [online]. 2022 [cit. 2022-04-28]. Dostupné z: https://docs.microsoft.com/en-us/windows/wsl/install.

[15] Naberezny, M. *Fluent Interfaces in PHP* [online]. 2005 [cit. 2022-05-08]. Dostupné z: http://mikenaberezny.com/2005/12/20/fluent-interfaces-in-php/.

[16] Nette. *Dependency Injection* [online]. 2008 [cit. 2022-05-02]. Dostupné z: https://doc.nette.org/cs/dependency-injection.

[17] PHP. *PHP: Enumerations* [online]. 2021 [cit. 2022-05-08]. Dostupné z: https://www.php.net/manual/en/language.enumerations.php.

[18] Pitt, C. *Pro PHP 8 MVC: Model View Controller Architecture-Driven Application Development*. 2. vyd. Apress Berkeley, CA, 2021. 367 s. ISBN 978-1-4842-6957-2.

[19] Poulton, N. *Docker Deep Dive: Zero to Docker in a single book*. 2020. vyd. Packt Publishing, 2020. 250 s. ISBN 978-1521822807.

Příloha A

Výsledkový soubor systému LaserMaxx

Tato příloha obsahuje příklad a popis souboru výsledků systému LaserMaxx, který výsledná aplikace zpracovává.

Soubor výsledků je rozdělený na tři hlavní části, které jsou oddělené jedním prázdným řádkem. První část obsahuje veškeré nastavení herního módu a časování. Druhá část uchovává informace o hráčích a týmech. Ve třetí části se nachází samotné výsledky hry pro jednotlivé hráče a týmy. LaserMaxx soubor výsledků aktualizuje třikrát v průběhu celého procesu hry. Poprvé při načtení hry na vesty – obsahuje pouze první dvě části bez výsledků. Podruhé po spuštění hry – aktualizuje časy v řádcích `TIMING` a `GAME`. Nakonec soubor aktualizuje po stažení výsledků z vest – přidá třetí část a znovu aktualizuje časy `TIMING` a `GAME`.

Hodnoty pro jednotlivé řádky bylo nutné vypozorovat, jelikož nejsou v souboru nijak pojmenované.

```
1  SITE{53874,0504011,EVO-5 MAXX}#
2  GAME{12,,20200526170920,20200526171000,7}#
3  TIMING{20, 15, 5, 20200526170940,20200526170954,20200526170959}#
4  STYLE{1-TEAM-DEATHMACH,Ordinary team game.,1,15, 0}#
5  STYLEX{5,9999,999,25359,5135}#
6  STYLELEDS{1,15,15,15,15,15,1,5,1,0,15}#
7  STYLEFLAGS{0,0,1,1,0,1,0,1,0,1,0,1,0,1,1,1,1,1,0,0,0,1,0,0,1,0,0}#
8  STYLESOUNDS{255}#
9  SCORING{-50,100,-50,-25,-50,0,100,100,100,100,21500,0,0,0,0,0}#
10 ENVIRONMENT{ , , , ,,C:\LaserMaxx\shared\music\evo5.mp3,,,,,}#
11 VIPSTYLE{0,5,20,1,12,0,70}#
12 VAMPIRESTYLE{0,0,0,6,9999,0}#
13 SWITCHSTYLE{0,0}#
14 ASSISTEDSTYLE{1,1,0,0,0,1,2,0}#
15 HITSTREAKSTYLE{0,5,11}#
16 SHOWDOWNSTYLE{0,3,0,3,0}#
17 ACTIVITYSTYLE{0,0,0}#
18 TERMINATESTYLE{0}#
19 MINESTYLE{0,0,0,0,Unnamed Unit}#
20 MINESTYLE{1,0,0,0,Unnamed Unit}#
21 MINESTYLE{2,0,0,0,Unnamed Unit}#
22 MINESTYLE{3,0,0,0,Unnamed Unit}#
23 MINESTYLE{4,0,0,0,Unnamed Unit}#
24 MINESTYLE{5,0,0,6,miniMina}#
25 MINESTYLE{6,0,2,6,MinaH}#
```

```
MINESTYLE{7,0,2,6,MinaD/BRANA}#

GROUP{,}#
PACK{1,Chmelda,0,0,0,0,0}#
PACK{2,Buzerant 2.0,0,0,0,0,0,0}#
PACK{3,Vojta,2,0,0,0,0}#
PACK{4,Dave,0,0,0,0,0}#
PACK{5,Rysa,0,0,0,0,0}#
PACK{6,Buzerant 3.0,2,0,0,0,0}#
PACK{8,Nufik,2,0,0,0,0}#
PACK{9,Davydeg,2,0,0,0,0}#
PACK{10,Buzerant,0,0,0,0,0}#
PACK{11,David,2,0,0,0,0}#
TEAM{0,Red team,4}#
TEAM{2,Blue team,3}#

PACKX{1,-100,11,0,2,7,,23}#
PACKX{2,-100,21,2,1,6,,51}#
PACKX{4,-100,11,0,2,4,,25}#
PACKX{5,-100,9,2,1,5,,33}#
PACKX{6,100,45,1,0,2,,18}#
PACKX{8,0,23,0,0,3,,25}#
PACKX{11,100,20,1,0,1,,20}#
PACKY{1,0,0,0,0,9988,0,0,0,0,0,0,0,0,0,2,997,886,-100,0,0,0,0}#
PACKY{2,0,0,0,0,9978,10,0,0,0,0,0,0,2,0,1,998,886,-100,0,0,0,0}#
PACKY{4,0,0,0,0,9988,0,0,0,0,0,0,0,0,2,0,996,886,-100,0,0,0,0}#
PACKY{5,0,0,0,0,9990,22,0,0,0,0,0,0,2,0,1,996,886,-100,0,0,0,0}#
PACKY{6,0,0,0,0,9954,2,0,0,0,0,0,1,0,0,0,999,886,100,0,0,0,0}#
PACKY{8,0,0,0,0,9976,0,0,0,0,0,0,0,0,0,0,999,886,0,0,0,0,0}#
PACKY{11,0,0,0,0,9979,5,0,0,0,0,0,1,0,0,0,999,886,100,0,0,0,0}#
TEAMX{0,-400,2}#
TEAMX{2,200,1}#
HITS{1,0,0,0,0,0,0,0}#
HITS{2,1,0,0,1,0,0,0}#
HITS{4,0,0,0,0,0,0,0}#
HITS{5,1,1,0,0,0,0,0}#
HITS{6,0,0,1,0,0,0,0}#
HITS{8,0,0,0,0,0,0,0}#
HITS{11,0,0,1,0,0,0,0}#

GAMECLONES{0}#
```

Výpis A.1: Příklad obsahu souboru výsledků systému LaserMaxx: `0012.game`

A.1 Nejzásadnější kategorie výsledkového souboru

Tato kapitola obsahuje výpis pár nejdůležitějších kategorií výsledků a popis jejich hodnot.

A.1.1 TIMING

Nastavení časování hry.

- čas v sekundách před začátkem hry
- délka hry v minutách
- čas konce hry v sekundách

- datum a čas začátku hry
- datum a čas konce hry
- datum a čas po stažení výsledků

A.1.2 PACK

Informace o nastavení konkrétní vesty.

- číslo vesty
- jméno hráče
- číslo týmu
- neznámý argument
- speciální označení vesty (VIP)
- neznámý argument
- neznámý argument

A.1.3 TEAM

Informace o nastavení týmu.

- číslo týmu (odpovídá barvě týmu)
- jméno týmu
- počet hráčů

A.1.4 PACKX

Základní výsledky hráče.

- číslo vesty
- skóre
- výstřely
- zásahy
- smrti
- pořadí
- neznámý argument
- neznámý argument

A.1.5 TEAMX

Souhrnné výsledky týmů.

- číslo týmu (odpovídá barvě týmu)
- skóre
- pořadí

A.1.6 PACKY

Detailní výsledky hráče.

- číslo vesty
- skóre za výstřely
- skóre za bonusy
- skóre za smrti od min
- zbývající náboje
- přesnost střelby v procentech
- zásahy do min
- počet bonusu – agent
- počet bonusu – neviditelnost
- počet bonusu – samopal
- počet bonusu – štít
- zásahy do protihráčů
- zásahy do spoluhráčů
- smrti od protihráčů
- smrti od spoluhráčů
- neznámý argument
- skóre
- čtyři neznámé argumenty

A.1.7 Hits

Počty zásahů mezi jednotlivými hráči.

- číslo vesty
- počet zásahů každého dalšího hráče v pořadí určené číslem vesty

Příloha B

Konfigurace Docker

Konfigurace Docker se nachází ve dvou souborech: `Dockerfile` a `docker-compose.yml`. Tato příloha se zaměřuje jen na některé části z těchto dvou souboru, které byli nezbytné nastavit.

Soubor `Dockerfile` obsahuje všechny kroky potřebné k sestavení celé aplikační části. Proces je rozdělen do několika podčástí pomocí příkazu `FROM <previous> AS <name>` což pomáhá v orientaci a Docker může jednotlivé stupně zachovat.

B.1 Instalace potřebných PHP knihoven

Po nakopírování rodičovských Docker obrazů a prvotní instalaci potřebných UNIX nástrojů je třeba nainstalovat dodatečné PHP knihovny, které aplikace potřebuje (výpis B.1). Rodičovský obraz `php:8.1.1-apache` v základu žádné dodatečné rozšíření neobsahuje.

```
1  # PHP extensions
2  RUN apt-get install -y libzip-dev unzip
3  RUN apt install -y curl libcurl4-openssl-dev
4  RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
5  RUN docker-php-ext-install curl && docker-php-ext-enable curl
6  RUN docker-php-ext-install gettext && docker-php-ext-enable gettext
7  RUN docker-php-ext-install sockets && docker-php-ext-enable sockets
8  RUN docker-php-ext-install pdo_mysql && docker-php-ext-enable pdo_mysql
9  RUN docker-php-ext-install zip && docker-php-ext-enable zip
10 RUN docker-php-ext-install pcntl && docker-php-ext-enable pcntl
11 RUN apt-get install -y libicu-dev
12 RUN docker-php-ext-install intl && docker-php-ext-enable intl
13 RUN apt-get update && apt-get upgrade -y
14 RUN apt-get install -y cifs-utils
```

Výpis B.1: Instalace všech potřebných PHP rozšíření v souboru Dockerfile.

B.2 Nastavení jazyků pro překlady

Jelikož celá aplikace podporuje překlad do několika jazyků, k čemuž využívá knihovny `gettext`[1], je nutné potřebné jazyky nainstalovat i v rámci operačního systému Unix (výpis B.2). Je třeba předem určit veškeré jazyky, které systém bude podporovat. Jazyky jsou nainstalovány s podporou kódování `UTF-8`.

[1] https://www.gnu.org/software/gettext/

```
# Setup gettext languages
RUN apt-get install -y locales \
  && sed -i -e 's/# cs_CZ.UTF-8 UTF-8/cs_CZ.UTF-8 UTF-8/' /etc/locale.gen \
  && sed -i -e 's/# en_US.UTF-8 UTF-8/en_US.UTF-8 UTF-8/' /etc/locale.gen \
  && sed -i -e 's/# de_DE.UTF-8 UTF-8/de_DE.UTF-8 UTF-8/' /etc/locale.gen \
  && sed -i -e 's/# fr_FR.UTF-8 UTF-8/fr_FR.UTF-8 UTF-8/' /etc/locale.gen \
  && sed -i -e 's/# sk_SK.UTF-8 UTF-8/sk_SK.UTF-8 UTF-8/' /etc/locale.gen \
  && sed -i -e 's/# ru_RU.UTF-8 UTF-8/ru_RU.UTF-8 UTF-8/' /etc/locale.gen \
    && dpkg-reconfigure --frontend=noninteractive locales
RUN locale-gen cs_CZ.UTF-8
RUN update-locale -y
```

Výpis B.2: Instalace jazyků pro knihovnu gettext v souboru Dockerfile.

B.3 Nastavení GIT a stažení aplikace

Jak jsem již zmínil v kapitole 4.2.3, aplikace se nebude kopírovat na Docker image při sestavení, ale bude využívat git repozitáře, který zajistí synchronizaci všech zdrojových kódů. Proto je nutné při sestavení obrazu repozitář inicializovat (výpis B.3).

```
# Move to project directory
WORKDIR /var/www/html/

# Initialize git and download project
RUN git init && git remote add origin https://github.com/Heroyt/LaserArenaControl.git &&
    git fetch && git checkout -t origin/master
RUN git config pull.ff only
RUN git pull --recurse-submodules
RUN git submodule init
RUN git submodule update
RUN git submodule update --init --recursive --remote
```

Výpis B.3: Inicializace git repozitáře v souboru Dockerfile.

B.4 Spuštění kontejneru

Při spuštění samotného kontejneru se musí provést několik úkonů: aktualizovat aktuální verzi aplikace z git repozitáře, sestavit všechny potřebné části aplikace a spustit servery (WebSocket event server a Apache)

```
# Start command
# Updates project, builds it and runs a start script which starts WS event server and
    Apache
CMD git pull --recurse-submodules && git submodule update --init --recursive --remote &&
    composer build && php install.php && sh ./start.sh
```

Výpis B.4: Příkaz po spuštění kontejneru v souboru Dockerfile.

B.5 Docker compose služby

Jelikož systém využívá dvou nezávislých kontejnerů, jsou oba definované v konfiguračním souboru `docker-compose.yml`. Tento soubor obsahuje informace, které služby se mají spustit a jejich parametry, jako otevřené porty a propojené adresáře.

```yaml
services:
    web:
        restart: always
        build: .
        ports:
            - "1234:80"
            - "9999:9999"
        links:
            - "db"
        depends_on:
            - db
        volumes:
            - lmx:/var/www/html/lmx
            - /mnt/d/Docker/private:/var/www/html/private
    db:
        restart: always
        image: "mariadb:latest"
        ports:
            - "3307:3306"
        environment:
            MARIADB_ROOT_PASSWORD: "super-secret-password"
            MARIADB_DATABASE: "lac"
        volumes:
            - /mnt/d/Docker/mysql:/var/lib/mysql
```

Výpis B.5: Konfigurace služeb v souboru docker-compose.yml.

B.6 Napojení sdíleného adresáře z počítače Lasergame

Jak již bylo popsáno v kapitole 4.2.4, v Laser aréně Písek běží Docker aplikace na jiném počítači, než systém LaserGame. Z tohoto důvodu je nutné sdílet adresář s výsledky přes síť přímo do kontejneru. Toho lze dosáhnout pomocí konfigurace CIFS napojení v souboru `docker-compose.yml`. Kontejner se pokouší sdílený adresář připojit při svém spuštění a ukončí se s chybou, pokud se připojení nezdařilo.

```yaml
volumes:
    lmx:
        driver: local
        driver_opts:
            type: cifs
            device: "//192.168.50.3/lasermaxx/shared"
            o: username=xxxx,password=xxxxx,rw,file_mode=0777,dir_mode=0777
```

Výpis B.6: Konfigurace napojení pomocí protokolu CIFS na adresář počítače LaserGame v souboru docker-compose.yml.

Příloha C

Dotazník na hráče

V této příloze je uveden popis průzkumu, který byl proveden na hráčích. Průzkum se skládal ze tří částí, každá s trochu jiným cílem. První část se snažila zjistit co hráči samotní od výsledků hry čekají a co za statistiky je zajímá. Druhá část se zaměřila na vytvořené řešení, její hodnocení přehlednosti a vzhledu. Třetí část měla za cíl stav budoucí, kterým směrem by se mohl systém pro hráče posouvat dál a jaké funkce by rádi viděli.

Průzkum byl prováděn na hráčích v Laser aréně Písek. Největší skupinu hráčů tvořili děti do 18ti let. Co se hráčské zkušenosti týče, dotázaní byli poměrně rovnoměrně rozložení od úplných nováčků po hráče, kteří hrají pravidelně každý týden.

Informace o dotázaných

Dotazník vyplnilo celkem 63 lidí. 51% byli děti do 18ti let, 29% byli hráči od 18ti do 30ti let. 37% dotázaných byli hráči, kteří hráli LaserGame pouze párkrát a 24% byli pravidelní hráči, kteří hrají alespoň jednou týdně, sedm (11%) dotázaných nehrálo nikdy. Přibližně polovina (46%) z dotázaných hrála LaserGame alespoň ve dvou různých arénách.

Seznam otázek

Dotázaným byly položeny následující otázky:

1. O Vás
   (a) Do jaké věkové skupiny patříte?
   (b) Jak často hrajete laser game?
   (c) Hrál/a jste někdy i v jiné aréně?

Průzkum požadavků hráčů

Tato část cílila primárně na jednotlivé požadavky hráčů a na jejich očekávání od výsledků. Cílem bylo vytvořit seznam jednotlivých statistik a jejich atraktivnosti pro hráče. Výsledky průzkumu pomohli určit hierarchii jednotlivých statistik – které statistiky mohou být více schované, a které by měli být vidět hned.

Seznam otázek

Dotázaným byly položeny následující otázky:

1. Hodnoťte 1-5, jak moc vás zajímají jednotlivé statistiky výsledků
   (a) Moje skóre a celkové pořadí (individuální)
   (b) Skóre a pořadí týmů
   (c) Moje přesnost střelby
   (d) Můj počet výstřelů
   (e) Můj počet střel do prázdna
   (f) Můj celkový počet zásahů a smrtí
   (g) Trofeje
   (h) Můj nejoblíbenější cíl a největší zabiják
   (i) Můj poměr zásahů a smrtí (K:D)
   (j) Moje dnešní umístění
   (k) Zásahy do jednotlivých hráčů

Výsledky

Z odpovědí je zřejmé, že nejvíce hráče zajímá skóre a pořadí. Hráče zajímá individuální skóre a pořadí více než týmové. 9% hráčů nezajímá týmové skóre vůbec. Nejméně zajímavou statistikou pro hráče je počet střel.

Zajímavé je, že pořadí statistik je stejné v rámci všech věkových skupin a není ani závislé na herních zkušenostech hráčů.

Seřazené statistiky

Pořadí statistik je určené na základě bodů z hlasů. Jelikož dotázaní hodnotili jednotlivé statistiky známkou jako ve škole (1-5), body se počítali přesně obráceně – hodnocení 1 = 4 body, hodnocení 5 = 0 bodů.

1. Moje skóre a celkové pořadí (individuální)
   133 bodů
2. Skóre a pořadí týmů
   128 bodů
3. Můj nejoblíbenější cíl a největší zabiják
   128 bodů
4. Zásahy do jednotlivých hráčů
   122 bodů
5. Můj celkový počet zásahů a smrtí
   118 bodů
6. Moje přesnost střelby
   112 bodů

7. Moje dnešní umístění
   111 bodů
8. Trofeje
   109 bodů
9. Můj poměr zásahů a smrtí (K:D)
   108 bodů
10. Můj počet výstřelů
    93 bodů
11. Můj počet střel do prázdna
    85 bodů

Hodnocení vytvořeného řešení

Cílem této části bylo získat zpětnou vazbu k vytvořenému řešení. Na základě odpovědí bylo řešení iterativně upraveno.

Seznam otázek

Dotázaným byly položeny následující otázky:

1. Papírové výsledky
   (a) Jak často si procházíte papírové výsledky
   (b) Ohodnoťte detailnost výsledků (dostatečně, málo detailní, moc detailní)
   (c) Ohodnoťte aktuální vzhled
2. On-line výsledky
   (a) Jak často si procházíte on-line výsledky
   (b) Ohodnoťte detailnost výsledků (dostatečně, málo detailní, moc detailní)
   (c) Ohodnoťte aktuální vzhled
3. Výsledková tabule
   (a) Ohodnoťte detailnost výsledků (dostatečně, málo detailní, moc detailní)
   (b) Ohodnoťte aktuální vzhled
4. Hodnoťte 1-5 srozumitelnost jednotlivých částí jednotlivých typů výsledků
5. Hodnoťte 1-5 vzhled jednotlivých částí jednotlivých typů výsledků

Výsledky

Z odpovědí bylo zjištěno, že 82% z dotázaných si rádo prochází tištěné výsledky, on-line výsledky si 53% z dotázaných otevírá jen, když je zajímají nějaký detail a 22% upřednostňuje on-line výsledky před tištěnými. Ne příliš překvapivý je fakt, že většina lidí upřednostňujících on-line výsledky je mladší třiceti let.

S detailností a aktuálním vzhledem výsledků neměl žádný z dotázaných problém.

Hodnocení budoucích funkcí

V této části byly položeny otázky týkající se možnými budoucími funkcemi systému pro hráče, které dotázaným byli stručně představeny a měli hodnotit 1-5 jak moc by je daná funkce zajímala. Výsledky těchto otázek pomohou určit priority pro budoucí vývoj.

Seznam otázek

Dotázaným byly položeny následující otázky:

1. Hodnoťte 1-5, jak moc vás daná funkce zajímá a jak moc byste chtěli funkci využívat
   (a) Vlastní profil hráče s dlouhodobými statistikami
   (b) Globální žebříček
   (c) Možnost přidávání přátel a vytváření týmů
   (d) Úroveň hráče
   (e) Nové trofeje
   (f) Organizace a registrace na turnaje

Výsledky

Dotázaní obecně projevili velký zájem o nové funkce, které by mohli být vytvořen. Nejvíce dotázaným by se líbila funkce vlastního profilu s úrovní hráče a organizace turnajů s registrací.

Seřazené statistiky

Pořadí statistik je určené na základě bodů z hlasů. Jelikož dotázaní hodnotili jednotlivé funkce známkou jako ve škole (1-5), body se počítali přesně obráceně – hodnocení 1 = 4 body, hodnocení 5 = 0 bodů.

1. Úroveň hráče
   140 bodů
2. Organizace a registrace na turnaje
   140 bodů
3. Vlastní profil hráče s dlouhodobými statistikami
   140 bodů
4. Možnost přidávání přátel a vytváření týmů
   132 bodů
5. Nové trofeje
   131 bodů
6. Globální žebříček
   124 bodů