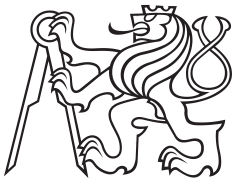
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra softwarového inženýrství

Návrh a implementace sportovního deníku uzpůsobeného pro lukostřelecké atlety

Václav Lokaj

Vedoucí: Ing. Tomáš Beneš
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Lokaj** Jméno: **Václav** Osobní číslo: **485254**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Studijní obor: **bez oborů**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh a implementace sportovního deníku uzpůsobeného pro lukostřelecké atlety

Název bakalářské práce anglicky:

Design and implementation of a training log application customized for archery

Pokyny pro vypracování:

Tréninkové deníky pro atlety jsou důležitou součástí přípravy elitních střelců. Obecné implementace treninkových deníku pro atlety jsou nedostatečné pro specifické potřeby lukostřelců. Student provede rešerši existujících řešení jak pro lukostřelbu tak i pro obecné atlety. Na základě rešerše sestaví seznam základních požadavků, které by aplikace měla splňovat. Seznam by měl obsahovat i cíle, které bude projekt v budoucnu implementovat i mimo rozsah bakalářské práce. Na základě požadavků student vytvoří návrh uživatelského rozhraní v příslušných nástrojích pro UX desing. Student na základě požadavků provede rešerši v dostupné literatuře technologií dostupných pro vytváření fronteend a backend component. Pomocí vybraných technologii poté vytvoří open-source repozitář vyžívající průbežné integrace pro testování, sestavení a vydávání applikace. Cílem práce je vytvořit aplikaci, která bude splňovat základní praktiky moderního programátora a splňuje základní potřeby střelce jako zaznamenávání tréninků, bodovaní jednotlivých výstřelů na terči v rámci daného tréninku a poskytovat přehledné statistiky o průbežné výkonu střelce.

Seznam doporučené literatury:

LEHNERT, Michal; MICHAL, Boteket; MARTIN, Sigmund; DAVID, Smékal et al., 2014. Kondiční trénink [online]. Univerzita Palackého v Olomouci [cit. 2024-01-12]. isbn 978-80-244-4369-0
VYCUDILÍKOVÁ, Pavla, 2008. Výživa sportovců [online]. [cit. 2024-01-12].
HORÁČKOVÁ, Marie, 2022 Model celoročního tréninkového cyklu vrcholového lukostřelce

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Tomáš Beneš katedra číslicového návrhu FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **19.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **15.02.2026**

Ing. Tomáš Beneš podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Děkuji ČVUT, že mi je tak dobrou *alma mater*. Chtěl bych také poděkovat vedoucímu práce Ing. Tomáši Benešovi za jeho rady a pomoc při psaní závěrečné práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 24. května 2024

Abstrakt

Tato bakalářská práce se zaměřuje na návrh sportovního deníku přizpůsobeného pro potřeby lukostřeleckých atletů. Tato aplikace umožní uživatelům zaznamenávat údaje o své střelbě a poskytne jim efektivní nástroj pro sledování tréninkových výsledků.

V průběhu práce byla provedena důkladná analýza požadavků, analýza existujících řešení a jejich nedostatků, následně návrh UI, návrh aplikace, implementace a testování aplikace. Dále práce popisuje moderní technologie a způsob jejich použití při implementaci prototypu aplikace, včetně testování samotného prototypu.

Klíčová slova: lukostřelecký deník, návrh softwaru, implementace softwaru, Spring Boot, TypeScript, React Native, Android aplikace, backendový vývoj, frontendový vývoj, REST API, Git

Vedoucí: Ing. Tomáš Beneš

Abstract

This bachelor's thesis focuses on the design of a sports diary tailored to the needs of archery athletes. This application will enable users to record their shooting data and provide them with an effective tool for tracking training results.

Throughout the project, a thorough analysis of requirements was conducted, along with an analysis of existing solutions and their shortcomings, followed by UI design, application design, implementation, and testing of the application. Furthermore, the thesis describes modern technologies and their use in implementing the application prototype, including testing of the prototype itself.

Keywords: archery diary, software design, software implementation, Spring Boot, TypeScript, React Native, Android application, backend development, frontend development, REST API, Git

Title translation: Design and implementation of a training log application customized for archery

Obsah

Obrázky

Tabulky

Úvod

Předkládaná bakalářská práce si klade za cíl vytvořit aplikaci, která bude plnit potřeby tréninkového deníku uzpůsobeného pro pokročilé i amatérské střelce. Atleti potřebují zaznamenávat svůj trénink, aby sledovali, jak se jejich výkonnost v čase mění. Tato data pak mohou buď sami, nebo za pomocí trenéra vyhodnocovat a upravovat podle toho svůj tréninkový plán. Tento proces vytváří poptávku po platformě, na které by atleti mohli vhodně sdílet data o své aktivitě s trenéry.

Na druhé straně trenéři chtějí mít přístup k záznamům o svých svěřencích, jejich současné výkonnosti a potřebují nástroj, který by jim umožňoval měnit tréninkové aktivity v čase. Aplikace, které jsou v současné době na trhu dostupné, nedostatečně naplňují potřeby atletů a trenérů zaznamenávat tréninkové aktivity. Existující aplikace neřeší problémy plánování, často jsou buď již zastaralé, anebo proprietární bez vhodného UI. Vznikla tak potřeba vytvořit vlastní implementaci, která splňuje základní potřeby střelce jako zaznamenávání tréninků, bodování jednotlivých výstřelů na terči v rámci daného tréninku a poskytování přehledné statistiky o průběžném výkonu střelce.

Atlet se také musí udržovat v dobré fyzické kondici, proto je vhodné integrovat rozhraní s již existující platformou, která zaznamenává informace o životním stylu, jako jsou fyzická aktivita, jídelníček nebo délka spánku.

Tato práce v první kapitole seznamuje čtenáře s potřebou zaznamenávat data o aktivitě atleta a popisuje nezbytnost role trenéra a důležitost jeho komunikace s atletem. Na konci kapitoly je probrána specifikace v tréninku

terčové lukostřelby a identifikace informací, které je potřeba mezi atletem a trenérem sdílet. V druhé kapitole jsou porovnány existující implementace jak pro lukostřelecké deníky, tak pro obecný atletický trénink, a jsou zde podrobně rozebrány jejich silné a slabé stránky. Třetí kapitola se zabývá požadavky na aplikaci a metodou použitou pro sběr požadavků. Čtvrtá kapitola se zabývá návrhem aplikace. Je zde popsané a odůvodněné architektonické schéma použité při implementaci a jsou zde detailně popsány jednotlivé technologie, které jsou poté využity při implementaci. Pátá kapitola se věnuje implementaci všech částí aplikace, je v ní popsaná adresářová struktura a podrobně se zde probírají konkrétní technická řešení. Šestá kapitola se věnuje testování aplikace. Poslední kapitola obsahuje závěr práce.

Kapitola 1

Úvod do problematiky sportovních deníků uzpůsobených pro lukostřelecké atlety

Tato kapitola se zaobírá obecným popisem sportovní lukostřeby a sportovních deníků. Jsou zde popsány základní pravidla a komponenty potřebné k lukostřelbě a uvedeny důvody, proč je pro atleta důležité vést si sportovní deník.

1.1 Sportovní lukostřelba

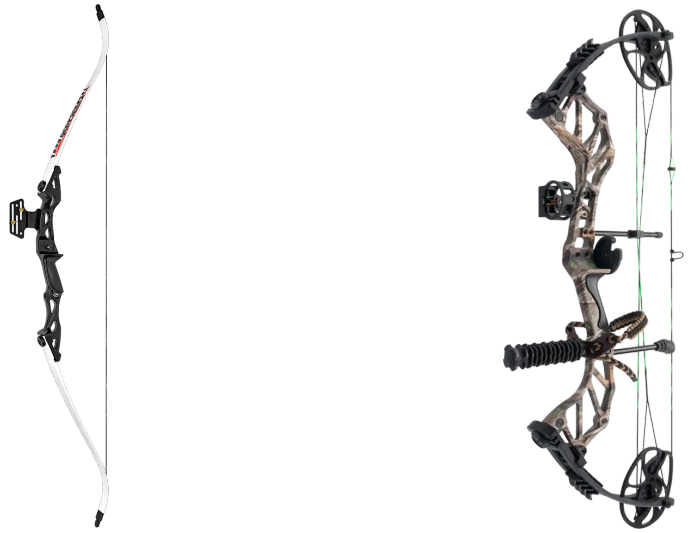
První zmínky o používání luku a šípů pro lov jsou nejméně 50000 let staré. Luk se používal nejprve pro lov a osobní ochranu. Postupem času se z něj stal i symbol společenského postavení a byl důležitým nástrojem hrdinů v příbězích, jako je například příběh o Robinu Hoodovi. [42] Do dnešní doby se lukostřelba transformovala do sportovní disciplíny. Sportovní lukostřelba se může provozovat uvnitř anebo venku. Podle toho, jestli je lukostřelba provozována uvnitř nebo venku, hovoříme o „indoor“ nebo „outdoor“ lukostřelbě. Analogicky je pak rozlišováno mezi „indoor archery“ a „outdoor archery“, které tvoří dvě samostatné disciplíny v rámci tohoto sportu. Atelti střílí na terč až na 90metrovou vzdálenost a snaží se zasáhnout střed terče. Terč má tvar kruhu a skládá se obvykle z 10 do sebe vepsaných kruhů, které jsou barevně označeny. Používají se barvy zlatá, červená, černá, modrá a bílá. Sport je kombinací atletovy síly, zručnosti a preciznosti při střelbě šípy na terč. Atleti mezi sebou soutěží v průběhu sezóny na domácích nebo mezinárodních turnajích. Mistrovství světa v lukostřelbě se koná každé 2 roky. Lukostřelba

je také zastoupena na letních olympijských hrách. Zde se soutěží v několika kategoriích, konkrétně: muži, ženy, tým muži, tým ženy a smíšený tým. Střílí se na terč ze vzdálenosti 70 metrů a ke střelbě je používán reflexní luk. Utkání mezi lukostřelci se dělí na sady. Například, pokud soutěží lukostřelci v soutěži, ve které mají na terč postupně vystřílet 15 šípů během 5 sad o 3 šípech, pak probíhá utkání tak, že první z atletů vystřelí na terč 3 šípy a počká, než poté druhý atlet vystřelí své 3 šípy. Takto se střídají, než oba dva odstřílejí svých 15 šípů; vítězí atlet s vyšším bodovým scóre. Bodové scóre se tvoří součtem všech bodů, které byly atletovi přiděleny na základě míst v terči, které zasáhl. Pokud je zápas nerozhodný, pak se o výsledku rozhodne tak, že oba atleti vystřelí jeden šíp, a ten, jehož šíp dopadl blíže středu terče, vyhrává. [67]

K lukostřelbě lze používat několik typů luků. Jednotlivé typy luků se od sebe liší jak svojí konstrukcí, tak náročností, která je potřeba k jejich obsluze. Základní typ luku používaný ve sportovní lukostřelbě se nazývá reflexní luk. Reflexní luk (někdy také nazývaný zvratný luk) se skládá z držadla osazeného rameny, mezi kterými je napnutá tětiva, a případně dalších prvků, jako jsou klapačka, mířidla, stabilizátor anebo jiné bonusové prvky. Reflexní luky se vyznačují prohnutím konců ramen dopředu směrem k terči, jak je vidět na obrázku 1.1a. Je pro ně typické, že během napínání tětivy je nutné zvrátit ramena luku proti jejich přirozenému ohybu. Tato konstrukce přináší několik výhod. Reflexní luky mají obvykle vyšší rychlost výstřelu a zvýšenou energii přenášenou na šíp. Toto zvýšení výkonu může být zvláště výhodné pro lukostřelce, kteří chtějí dosáhnout vyšší účinnosti a dostřelu. Luk se ale v době výstřelu chová neklidně a jeho rázy se přenášejí do lukostřelcovy paže. Rozkmit ramen pak vede k nepřesné střelbě. Síla, kterou musí lukostřelec vyvinout při nátahu luku po založení šípu, stoupá u reflexního luku spolu s jeho napnutím luku, a v poslední fázi nátahu musí střelec vyvinout značně větší sílu, aby luk plně napnul, než na začátku napínání. Právě tato vlastnost je hlavním důvodem, proč existují rozdílné disciplíny pro střelbu s reflexními a kladkovými luky.[10]

Druhý typ luku se nazývá holý luk. Vizuálně i konstrukčně je velmi podobný reflexnímu luku; jedná se prakticky o jeho jednodušší variantu, která ale na rozdíl od reflexního luku není doplněna o klapačku, mířidla, stabilizátor a jiné bonusové prvky.

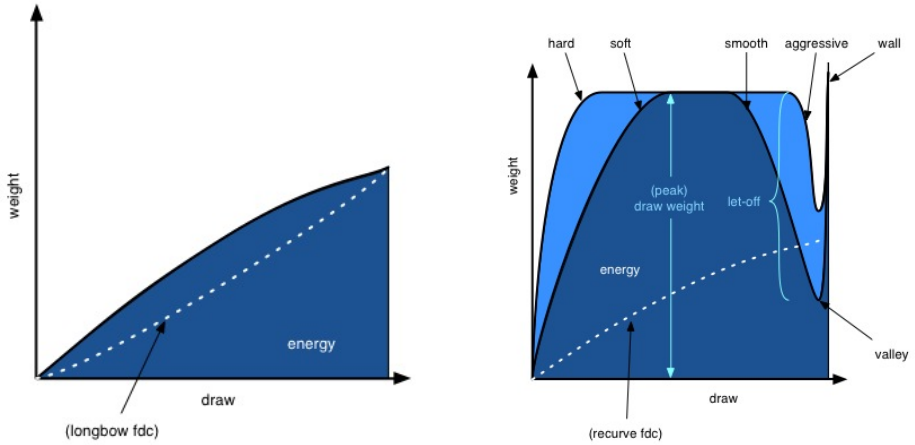
Třetím typem luku, který se používá při střelbě, je kladkový luk. Na první pohled je konstrukce luku odlišná od reflexního luku. V porovnaní s reflexními luky bývají větší. Na držadle jsou připevněna ramena, na vrcholku těchto ramen jsou pak připevněné kladky, mezi kterými je napnuta tětiva. Držadlo kladkového luku bývá stějně jako držadlo reflexního luku doplněno o další mířící prvky, které pomáhají lukostřelci ve střelbě. Typický kladkový luk je vidět na obrázku 1.1b. Hlavním rozdílem mezi oběma typy luků je odlišný



(a) : Reflexní luk [6] **(b) :** Kladkový luk [5]

Obrázek 1.1: Fotografie zachycující odlišné typy luků

průběh nátahu, a to konkrétně síla, kterou musí lukostřelec vyvinout, aby luk napnul a poté zafixoval tětivu v napnuté poloze před tím, než zamíří a vypustí šíp. Zatímco u reflexního luku síla roste přímo úměrně natahování, u kladkového luku je maximální síla přibližně v 80% natažení (obrázek 1.2b). V poslední části nátahu tato síla naopak klesá, takže je pro lukostřelce mnohem snadnější udržet luk napnutý a zamířit na terč těsně před tím, než dojde k vypuštění šípu. 1.2a [12] Proto je střelba z kladkového luku obecně považována za jednodušší oproti střelbě z reflexního luku.



(a) : Průběh nátahu reflexního luku [47] **(b) :** Průběh nátahu kladkového luku [13]

Obrázek 1.2: Grafy síly potřebné pro nátah odlišných typů luků

Sportoviště, na kterém lukostřelci střílí z luku na terč, se nazývá střelnice. Tyto střelnice se mohou nacházet buď uvnitř, anebo venku. Střelba v nezakrytých prostorech je zpravidla kvůli různým povětrnostním a klimatickým

podmínkám náročnější než střelba v hale. Střelnice obvykle vypadá tak, že na jedné straně jsou vedle sebe vyskládány terče a na opačné straně stojí v řadě lukostřelci, kteří na terč střílí. Platí přitom, že každý lukostřelec střílí na svůj vlastní terč, který ale není současně využíván jiným střelcem. Terče jsou umístěny na terčovnici, aby se snížila pravděpodobnost poničení šípu poté, co zasáhne cíl. Za terči bývá umístěna záchytná stěna, volně visící záchytná síť anebo dostatečně velký prostor. Důvodem tohoto opatření je zachytit šípy, které minou terčovnici. Střelec při střelbě používá pomocné ochranné prvky, jako jsou chrániče prstů a předloktí, a je oděn do přiléhavého oblečení.

Střelba se dělí na řízenou a neřízenou, přičemž platí, že při řízené střelbě je ohlašován začátek, konec a případně přerušení střelby zvukovým signálem, a během střelby je zakázáno, aby se střelci nebo jakékoliv jiné osoby nacházeli v blízkosti terčů. Platí, že střelci se mohou vydat k terči vytahovat šípy pouze společně po zaznění zvukového signálu a střelba může být také zahájena pouze po příslušném zvukovém signálu. V případě neřízené střelby není střelba aktivně nikým řízena, ale i tak je stále nutné zachovávat požadavky na bezpečnost.

1.2 Sportovní příprava lukostřeleckého atleta

Sportovní příprava, nebo také sportovní trénink, lze chápat jako sérii aktivit, které mají za cíl dovést sportovce k vytouženému výkonu v jeho sportovní disciplíně. Jedná se o sérii fyzických a psychických aktivit, jejímž pravidelným opakováním se optimalizuje výkon, kterého je poté sportovec schopný na soutěži dosáhnout. Sportovní trénink je komplexní proces zahrnující serii aktivit, jako jsou například: posílení psychické stránky sportovce nebo příprava na sportovní zatížení. Nelze jej tedy chápat jen jako samotnou pohybovou činnost. Obsahem sportovního tréninku je příprava sportovce v dané sportovní disciplíně. Tréninkový proces je nutné vykonávat delší časový úsek, než je dosaženo zlepšení ve sportovních výkonech atleta. Lukostřelba je sportem, v němž mají atleti různorodý kariérní růst, není proto pevně specifikován věk, ve kterém by měl sportovec dosahovat svých nejlepších výsledků. [27]

1.2.1 Tréninkový plán

K tomu, aby mohl sportovec dosáhnout co nejlepšího výkonu, je zapotřebí mít sestavený tréninkový plán. Tréninkový plán sestavuje trenér a jedná se

Období	Hlavní úkol období
Přípravné	Rozvoj kondice, trénovanosti
Předsoutěžní	Zvyšování výkonnosti (tapering)
Soutěžní	Udržení vysoké úrovně výkonu
Přechodné	Fyzická a psychická regenerace

Obrázek 1.3: Tabulka významu jednotlivých tréninkových období [70]

o základní jednotku tréninkového procesu, skládá se z jednotlivých tréninkových cyklů. Tréninkové cykly se dále dělí na makrocyklus, mezocyklus a mikrocyklus. Makrocyklus je období, které se upíná na hlavní vytyčený cíl, např. mistrovství světa v lukostřelbě. Makrocyklus se dělí na mezocykly. Každý mezocyklus má svůj úkol, například: předsoutěžní mezocyklus, závodní mezocyklus, regenerační mezocyklus a podobně. Mezocyklus se dělí na mikrocykly. Mikrocyklus je tvořen na základě aktuální fáze mezocyklu a jsou v něm již obsažené konkrétní úkony, které se budou v dané tréninkové jednotce vykonávat. Obvyklá doba trvání mikrocyklu je jeden týden.

Makrocyklus se také dělí na jednotlivá tréninková období. Tato období se nazývají přípravné období, předsoutěžní období, závodní období a přechodné období. Každé období se vztahuje k časovému úseku v rámci jednoho tréninkového cyklu a jejich význam je popsán v tabulce 1.3. Po dokončení sezóny se data v plánu zhodnotí a sestaví se nový plán na další sezónu. Právě kvůli zpětné analýze je potřeba data zaznamenávat do tréninkových deníků. [27]

1.2.2 Životospráva

Volba vhodné výživy je jednou z nejdůležitějších součástí přípravy sportovce. Přináší možnost zvýšení sportovní výkonnosti, a to jak svým dlouhodobým účinkem, tak i krátkodobým vlivem specifických dietních manipulací. Kvalita výživy výrazně mění schopnost absolvovat fyzickou zátěž. Jedině dlouhodobá vysoce kvalitní výživa vede k dokonalému zdraví a umožňuje kvalitní trénink. Nemůže však odstranit některé dědičné nedostatky, i když je v řadě případů může výrazně zredukovat. [62] Správný poměr výživných látek je nezbytný k dosažení ideální kondice pro konkrétní sportovní disciplínu. Jinak složený jídelníček bude mít atlet, který provozuje silový trénink (kulturista), a atlet, který provozuje sport vytrvalostního charakteru (maratonský běžec). Dalším důležitým parametrem, který je během tréninku u atleta sledován, je doba a kvalita spánku. Pro podrobný monitoring a snímání fyziologických údajů

se využívají majoritně chytré hodinky pro svou finanční dostupnost, rychlost zaznamenávání dat a jednoduchou obsluhu.

1.2.3 Složky sportovního tréninku

Technická příprava

Technická příprava v lukostřelbě se zaměřuje na rozvoj lukostřeleckých schopností a sladění lukostřeleckého vybavení (šípu a luku). Cílem technického tréninku je správné provedení pohybového úkonu. Aby si lukostřelec osvojil pohybový stereotyp, je nutné se nejprve naučit jednotlivé fáze pohybu. Poté co si je atlet zafixuje, začne tyto fáze spojovat, z čehož vzniká konečný pohybový stereotip nejblíže podobný ideálu provedení. [27] Technický trénink lze definovat také jako oblast tréninku zaměřenou na osvojování a zdokonalování pohybových dovedností a sportovní techniky, která je základem pro výkonnostní potenciál v soutěžních podmínkách. [35]

Kondiční příprava

Kondiční příprava se charakterizuje jako součást tréninkového procesu zaměřená zejména na rozvoj bioenergetického, funkčního a pohybového potenciálu sportovce vzhledem k požadavkům na sportovní výkon a přípravě na jeho podávání. Kondiční trénink se dělí na nespecifický a specifický. Nespecifický kondiční trénink je uplatňován především u mládeže a v menší míře u vyspělých sportovců. Jeho obsahem je široké spektrum nespecifických kondičních cvičení zaměřených na ovlivňování kondičních motorických schopností, které jsou základem pro všechny sporty nebo určitý okruh disciplín. Specifický kondiční trénink je především zaměřen na ovlivňování specifických kondičních motorických schopností. Výběr cvičení vychází z přehledu o nejintenzivnějších pohybech obsažených ve sportovním výkonu. [35] Mezi sporty, jejichž typickým rysem je přetěžování určitých svalových skupin, patří právě lukostřelba.

Taktická příprava

Cílem taktické přípravy je připravit sportovce na řešení situací, které mohou nastat během zápasu nebo soutěže. Jedná se především o schopnost řešit různé herní situace. Taktika hraje svoji důležitou roli v kolektivních sportech. V lukostřelbě může střelec například použít taktiku k tomu, aby si dopředu naplánoval rozložení síly v průběhu soutěže. [27]

Psychologická příprava

Psychologie sportovce je značně individuální a odráží se v ní sportovcova osobnost, disciplína, odhodnání a sebevědomí. Vzhledem k tomu, že na vrcholové úrovní má většina sportovců obdobné kondiční, technické i materiální podmínky, je to právě psychologie, která rozděluje výkony a určuje vítěze. Na soutěži je na lukostřelce kladen tlak, ať už kvůli přítomnosti diváků, nebo z touhy dosáhnout nejlepšího možného výkonu. Právě zvládání emocí vyvolaných tímto tlakem je obvykle faktorem, který rozhoduje o vítězi a poraženém. [27]

1.2.4 Post mortem analýza během tréninku lukostřelby

Jak již bylo probráno v sekci „Tréninkový plán“, trenér sestavuje atletovi tréninkový plán na základě dat, která si atlet zaznamenával v průběhu tréninkového cyklu do tréninkového deníku. Existuje ale i varianta tréninku, během které naopak trenér se závodníkem procházejí data z již uplynulého tréninku a snaží se na základě nich identifikovat, co se během střelby dařilo a co nikoliv. Tento přístup k analýze se obecně nazývá post mortem analýza a je hojně využíván například i v IT projektech. Předpokladem pro úspěšnou aplikaci jsou právě detailní záznamy o střelbě, na jejich základě je poté trenér schopen určit například vadný šíp či jiný defekt, který ovlivnil lukostřelcovu střelbu.

1.2.5 Struktura tréninku

Trénink má obvykle cíl, který je předem stanoven trenérem nebo střelcem samotným. U profesionálních střelců se využívá periodizace tréninku, která

zahrnuje různe fáze tréninku s ruznými cíly a prioritami. Obvykle jsou součástí tréninku různé typy cvičení, jako je střelba na odlišné vzdálenosti, simulace soutěže, technické cvičení a kondiční trénink. Je důležité zaznamenávat, jak se střelci daří při plnění těchto cílů a jak se mění jeho výkony v průběhu času. Trénink se dá rozložit na několik částí.

Rozehřátí, lidověji známější jako „rozcvička“ zabíra obvykle 10 až 20 minut na začátku tréninku. Jeho náplní je lehké protahování a tahy s lehkým lukem nebo cvičnou gumou, která simuluje luk. Po dokončení rozehřátí následuje zhruba 20 až 30 minut trvající rozehřátí s lukem. Cílem je rozehřát se s závodním lukem tak, aby atlet předešel úrazům, které jsou často spojeny s repetetivní podstatou střelby s těžkým lukem. Střílí se obvykle na vzdálenost 5 až 10 metrů bez terče. Tato část tréninku se označuje v češtině výrazem „školka“ a v angličtině jako „blank bale“. Po rozehřátí s lukem následuje hlavní podstata střeleckého tréninku. Zde se může jednat o střílení různého počtu šípů na různé vzdálenosti a na různé typy terčů včetně střelby bez terče. Pokud se střelec právě nachazí v období, ve kterém je jeho cíl zvýšit sílu, tak se trénink skládá ze střelby většího množství šípů a neklade se důraz na přesnost střelby. Naopak, pokud se střelec aktuálně nachází v technické části tréninkového cyklu, tak se zaznamenávají zásahy a jejich přesnost tak, aby byl střelec nebo jeho trenér zpětně schopný analyzovat změny v střelcově výkonnosti. V předsoutěžním období střelec střílí simulované soutěže s pevně danými objemy a používá časomíru, aby se co nejvíce přiblížil podmínkám na soutěži. Na konci tréninku střelci opakují střelbu na krátkou vzdálenost a následně si protáhují namožené svaly. Pro atleta je důležité zaznamenat si, jaký typ tréninku zrovna podstoupil, ať už zadáním od trenéra nebo zpětně od střelce.

1.2.6 Informace, které by si mel střelec zaznamenávat v tréninkovém deníku

Vzdálenost a terč: Každý výstřel by měl být zaznamenán s uvedením vzdálenosti od terče. Je důležité sledovat, jak se mění přesnost a úspěšnost na různých vzdálenostech.

Počet výstřelů: Zaznamenání počtu šípů, které vystřelil altet během tréninku, pomůže sledovat celkový objem práce a pokrok v tréninkovém plánu.

Úspěšnost zásahu: Pro každý výstřel je užitečné zaznamenat, zda se zásah dostal do terče, a pokud ano, jak blízko k centru. To pomáhá sledovat celkovou přesnost a konzistenci.

Poznámky o technice: Lukostřelec by měl zaznamenávat jakékoli poznámky o své technice během tréninku. To může zahrnovat postavení těla, úhel

tahu, zaměření a další aspekty techniky.

Pocit po výstřelu: Lukostřelec by měl také zaznamenávat své pocity a dojmy po každém výstřelu. To může zahrnovat hodnocení stability, koncentrace a celkového pocitu spokojenosti s výkonem.

Podmínky: Zaznamenání meteorologických podmínek (např. vítr, déšť, teplota) může pomoci pochopit, jak tyto faktory ovlivňují přesnost a výkon střelce.

Typ lukostřeleckého luku: Zaznamenání, jaký typ luku byl použit při každém tréninku (např. olympijský luk, lovecký luk, olympijský luk bez stabilizátorů atd.), může pomoci při hodnocení, jaký vliv má daný luk na výkon.

Kapitola 2

Rešerše existujících řešení pro lukostřelecké deníky

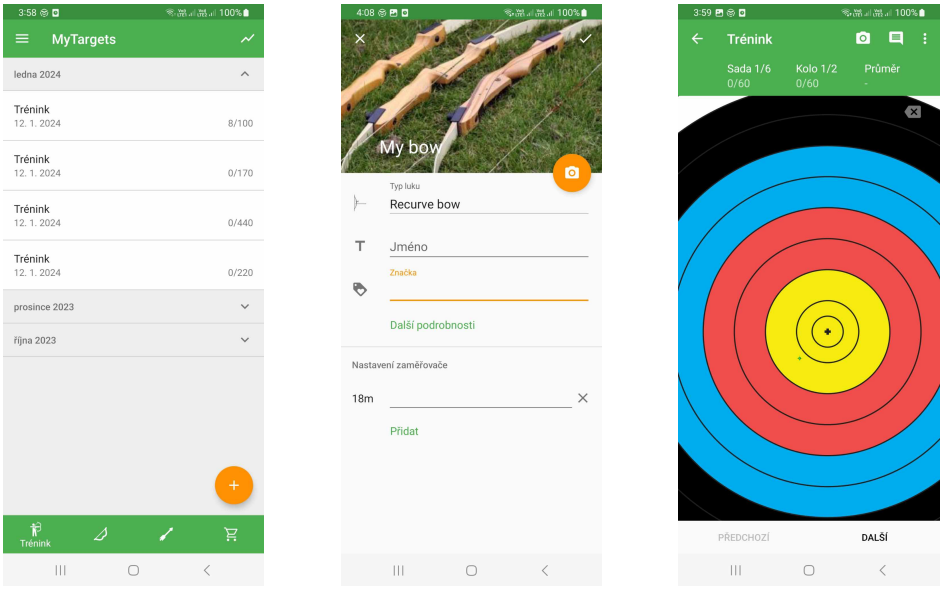
Tato kapitola se zaobírá existujícími řešeními sportovních deníků pro lukostřelecké atlety. Probral jsem zde uživastelské rozhraní, funkcionalitu, klady a zápory vybraných existujících aplikací.

2.1 My Targets

My Targets [40] je produkt od americké společnosti Mantis, která má sídlo ve městě Oswego ve státě Illinois. Hlavním zaměřením společnosti je výzkum a vývoj nových technologií týkajících se střelby z různých zbraní a jejích propojení s chytrými telefony. My Targets podporuje pouze telefony s operačním systémem Android a je zdarma ke stažení v obchodě Google Play. Od 24. října 2014, kdy byla aplikace poprvé vydána, má více než 100 000 stažení (informace z května 2024). Aplikace je určena pro zaznamenávání dat o tréninku lukostřelce a pozdější nahlížení do těchto záznamů. Aplikace je volně k použití.

2.1.1 Uživatelské rozhraní

Aplikace My Targets má uživatelsky příjemné a jednoduché rozhraní (obrázek 2.1a). Podle popisu autora byla k tvorbě aplikace použita knihovna Material Design od společnosti Google [36]. V aplikaci se uživatel pohybuje pomocí ikonek mezi 4 záložkami („Trénink“,„Luk“, „Šípy“ a „Obchod“) a bočního menu. Uživatel se na jednotlivých záložkách pohybuje mezi okny klikáním na ikony a na tlačítka. Pohyb pomocí swapu mezi jednotlivými záložkami neni integrován. Okna mají zpravidla společný prvek a tím je „aplikační lišta“. Aplikační lišta je prvek, který umožňuje uživateli vykonávat dodatečnou funkcionalitu v rámci okna, na kterém se právě nachází. Barevné schéma je velmi minimalistické a kombinuje zelenou a bílou barvu v hlavičce stránky. Text je zobrazen černě na bílém nebo šedém pozadí. 2.1



(a) : Seznam tréninků **(b) :** Přidání luku **(c) :** Zanesení výstřelu

Obrázek 2.1: Ukázkové obrazovky z aplikace My Targets

2.1.2 Funkce aplikace

Aplikace umožňuje uživateli přidat luk, který používá při tréninku (obrázek 2.1b). O luku si při přidání do aplikace uživatel zaznamená technické parametry a poté ho může přidat k libovonému tréninku. Další funkcí je uložení si sady šípů, které altet používá při střelbě. Mezi šípy aplikace rozlišuje pomocí pořadového čísla, které určuje, kolikátý v pořadí byl šíp při tréninku vystřelen. Hlavní funkcí aplikace je možnost zaznamenat střelbu na terč.

Na výběr má uživatel z 2 tipů tréninku. První typ je „Volný trénink“, při založení tohoto typu tréninku uživatel prvně zadá vzdálenost, následně si zvolí typ terče, na který bude strílet, poté luk a šípy, které bude používat, a nakonec může zadat další nepovinné informace, jako je počasí apod. Trénink se skládá z po sobě opakujících se kol. V každém kole si uživatel může zvolit jiný typ terče, vzdálenost a počet šípů. Počet kol je maximálně omezen na 100. Kola se pak skládají ze sad. Každá sada obsahuje stejný počet šípů jako uživatel zadal v parametrech při vytvoření tréninku. Uživatel je okamžitě po vytvoření tréninku přesměrován do první sady. Nejprve musí uživatel zahájit odpočítávání (20 sekund), které má na to, aby se připravil ke střelbě, a poté za 150 sekund odstřílet šípy do terče bez ohledu na to, kolik šípů použil. 30 sekund před koncem odpočítávání je upozorněn zvukovým signálem na to, že se jeho čas k odstřílení šípů blíží ke konci. Po konci střelby zanáší uživatel informace o zasažené části terče do chytrého telefonu pomocí kurzoru, kterým pohybuje pomocí gest (obrázek 2.1c). Poté co uživatel zanese alespon 3 šípy, začne apikace zobrazovat graf, který ohraničuje prostor, kde byly šípy zaneseny, a zobrazuje střed tohoto grafu. Informace o střelbě se postupně zaznamenávají v horní části obrazovky. Uživatel také může k sadě přidat fotografii a poznámky o střelbě. Po zanesení sady může uživatel buď zahájit další sadu a znovu střílet na terč, nebo ukončit trénink. Počet sad ve volném tréninku je omezený na maximálně 50. Časomíru je možné kdykoliv během tréninku vypnout nebo zapnout.

Druhý typ tréninku se nazývá „Trénink standartní kolo“. Struktura tréninku je stejná jako v případě volného tréninku, ale atlet si na začátku pevně určí průběh tréninku. To znamená, že již při vytvoření má pevně daný počet kol, sad v každém kole a počet šípů v každé sadě. Poté co atlet dokončí trénink, má v aplikační liště možnost přidat poznámku o tréninku. V aplikační liště je také umístěna ikona, která vytvoří atletovi na základě tréninku graf úspěšnosti jeho střelby. Poslední funkce, kterou aplikační lišta nabízí, je vytvoření „Scoreboard“, tedy tabulky, kde jsou vidět informace o tréninku a průběh střelby po jednotlivých kolech. Tento dokument je pak uživatel schopný vyexportovat ve formátu pdf nebo sdílet ve stejném formátu se svými kontakty.

Záložka „Obchod“ přesměruje uživatele na internetový obchod Mantis nabízející střelecké vybavení.

Boční menu uživateli nabízí kategorie „Časomíra“, „Nastavení“ a „Nápověda“. V „Nastavení“ je uživatel schopný si vytvořit profil a nastavit si formát tréninku, zadavání dat a generování výsledných statistik a grafů. Také zde může importovat zálohu dat (bude detailně popsáno v sekci Ukládání a sdílení dat). Poslední fukncí, kterou nastavení nabízí, je vytvoření zálohy dat v aplikaci a její uložení do paměti zařízení.

2.1.3 Ukládání a sdílení dat

V nastavení je možné vytvořit si zálohu dat uložených v aplikaci. Uživatel má poté na výběr, zda chce zálohu umístit na fyzické zařízení, na kterém je My Targets spuštěna, anebo na Google Drive. V nastavení aplikace je také experimentální funkce, po jejímž zapnutí bude My Targets ve vybraném časovém intervalu automaticky tvořit zálohu dat. Obdobně jako je aplikace schopná data zálohovat, je shopná tuto zálohu poté načíst. Pokud se tak stane, jsou všechna data na zařízení smazána a nahrazena daty ze souboru se zálohou.

2.1.4 Výhody aplikace

Aplikace má jednoduché a intuitivní uživatelské rozhraní. Dále má sjednocené funkce napříč obrazovkami, například tlačítko pro pohyb zpět je vždy v levém horním rohu, tlačítko pro pořízení fotografie nebo přidání komentáře v pravém horním rohu apod. Dobře zvolené barvy zajišťují dobrou čitelnost i za horších podmínek, např. během slunného dne. Další výhodou je přehledná navigace v aplikaci, pro kterou slouží aplikační lišta spolu se záložkami. Díky uživatelsky přívětivým názvům jednotlivých komponent a formulářových polí je i pro nového uživatele snadné se zorientovat, založit nový trénink a začít zaznamenávat šípy.

2.1.5 Nevýhody aplikace

Zásadní nevýhodou aplikace je, že neposkytuje možnost propojit aplikaci s externím účtem, takže pokud atlet příjde o svůj mobilní telefon, tak automaticky příjde i o všechna data, která zaznamenal do My Targets. Jedinou možností, jak se tomuto scénáři bránit, je pravidelně si manuálně vytvářet zálohu a tu poté přemístit mimo zařízení, což ale není příliš uživatelsky přívětivé. Další nevýhoda je, že aplikace neposkytuje žádný způsob sdílení dat mezi zařízeními navzájem, takže pokud například bude chtít závodník sdílet s trenérem data ze svého tréninku, tak musí vygenerovat report a ten poté trenérovi poslat. Není ale možné exportovat data z tréninku, která by mohl trenér importovat do My Targets na svém zařízení a procházet. Nevýhoda, která přímo souvisí s předchozím bodem, je, že aplikace neintegruje rozhraní pro trenéry, tudíž je atlet nucený si vše nastavovat sám a nemůže importovat předpřipravený tréninkový plán. Kvůli chybějícímu rozhraní pro trenéry také není žádný

způsob, jak by mohl trenér prohlížet data svých svěřenců a komunikovat s nimi.

Během samotné střelby se pak uživatel může setkat s méně závažnými nepříjemnostmi, jako je omezený výber terčů a chyvbějící rozhraní pro přidání nového terče. Během zaznamenávání šípu na terč také nelze přepínat mezi šípy, ale pohyb je možný pouze o krok zpět, takže pokud například při zadávání posledního šípu uživatel zjistí, že udělal chybu během pozicování prvního šípu, tak musí odebrat záznamy o všech šípech, než aplikace dovolí znovu napozicovat první šíp. Během kola také nelze změnit počet šípů, kterými atlet střílí, takže pokud například příjde o šíp, protože se poškodí, tak aplikace nedovolí ho v další střelbě vyřadit. Není možné také upravit počet kol a počet šípů, které atlet bude střílet v jednotlivých kolech.

Při pohybu mezi záložkami nejsou implementována gesta jako swap vlevo nebo swap vpravo. Chybí také barevné zvýraznění záložky, na které se právě uživatel nachází.

Velkým limitem je také chybějící verze aplikace pro operační systém iOS používaný na chytrých telefonech od firmy Apple. Pro aplikaci neexistuje zpracovaná dokumentace.

2.1.6 Shrnutí

My Targets je ideální volbou pro středně pokročilé a mírně pokročilé lukostřelce nebo sportovce, kteří se chtějí lukostřelbě věnovat pouze rekreačně. Díky velmi jednoduchému uživatelskému rozhraní není těžké se v aplikaci orientovat a pohodlně do ní zaznamenávat svůj trénink. Chybějící rozhraní pro komunikaci s trenérem ji činí nevhodnou pro použití atlety, kteří chtějí koordinovat a sdílet svůj trénink s trenérem.

2.2 ArtemisLite

ArtemisLite (nebo také Artemis) [7] je aplikace určená pro (polo)profesionální lukostřelce a pro jejich trenéry. Stejně jako My Targets, kterou jsem analyzoval v sekci 2.1, umožňuje uživateli počítat skóre při střelbě a zaznamenávat polohu šípů. Podle popisu autora ale také disponuje důmyslnou statistickou analýzou.

Hlavním rozdílem oproti konkurenční My Targets je rozhraní a funkce pro komunikaci a plánování tréninku mezi střelcem a trenérem. ArtemisLite byl vyvinut Marcelem van Apeldoornem, bývalým mezinárodním lukostřelcem, leteckým a softwarovým inženýrem. [29] První verze aplikace byla umístěna na Google Store v říjnu 2011. Aplikace je částečně volně k použití, obsahuje ale také placenou část pro lukostřelce a placenou část pro trenéry.

2.2.1 Uživatelské rozhraní

Artemis se skládá ze 4 záložek: „Archer“, „Setup“, „Records“ a „Analyze“. Záložka „Archer“ se v novější verzi aplikace rozdělila mezi dvě záložky, které se ale stále obě v dokumentaci nazývají „Archer“. Mezi záložkami se uživatel pohybuje pomocí gest doleva nebo doprava. Na každé záložce se nacházejí ikony, které odkazují na jednotlivé funkce aplikace. Každá ikona po kliknutí přesměruje uživatele na novou obrazovku s funkcionalitou popsanou v další sekci. V pravém horním rohu se nachází ikona pro přesměrování uživatele do nastavení aplikace. Barevný vzhled aplikace je uživateli umožněno měnit.

2.2.2 Funkce aplikace

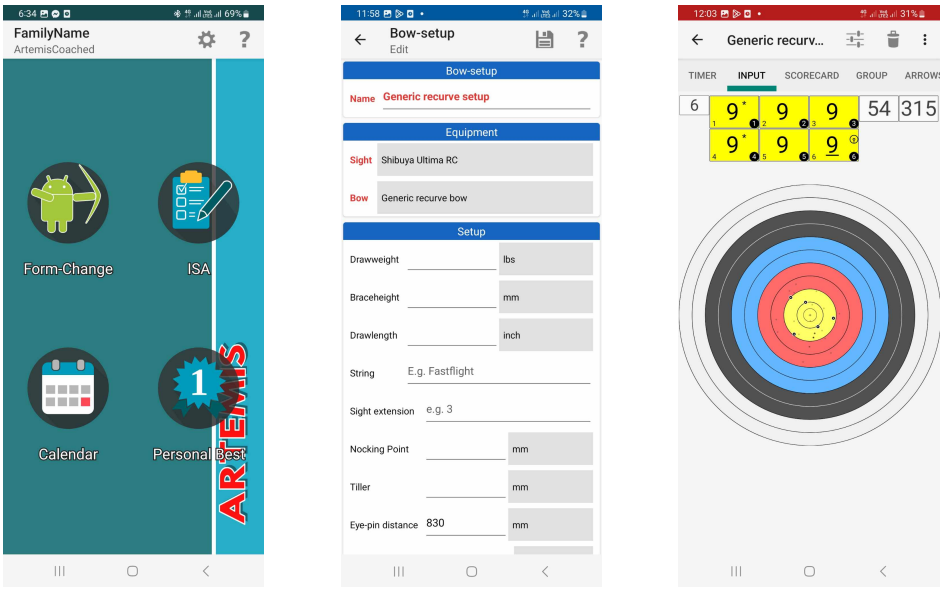
Uživatel se v aplikaci pohybuje tak, že se přesouvá mezi záložkami. Každá záložka obsahuje ikony, které poté co je uživatel vybere, otevřou obrazovku s funkcionalitou, na kterou odkazují. První záložka se nazývá „Archer“ (lukostřelec) 2.2a. Zde se zobrazují informace o konkrétním lukostřelci vhodné pro následnou analýzu za účelem zlepšení jeho výkonu. Funkce „Form Change“ (změna formy) umožňuje, aby si atlet zapisoval do formuláře poznámky o svojí aktuální formě a později si je mohl zpětně zobrazit. V ISA (Instantaneous Self-Assessment, okamžité sebehodnocení) dotazníku vybírá atlet z předpřipravených odpovědí na otázky. Dotazník si buď může sestavit sám, anebo ho dostane zadaný od trenéra. „Calendar“ (kalendář) implementuje kalendář, ve kterém jsou vyznačené tréninky a soutěže. „Personal Best“ (Osobní rekord) obsahuje informace o osobních rekordech dosažených během tréninku nebo na soutěži. [4]

Druhá záložka se nazývá „Setup“ (uspořádání). Funkce „Bow“ (luk) umožňuje uživateli vybrat si ze seznamu dostupných luků. Luky mohou být typu reflexní, skládaný anebo kuše. Při vytvoření luku se zadávají parametry, jako jsou typ rukojeti nebo typ ramen. Údaje o luku, jako jsou vzdálenost tětivy od břicha luku, místo na tětivě k založení šípu atd., jsou uloženy v „Bow-Setup“

(nastavení luku) 2.2b. Funkce „Sight“ (mířidla) slouží k přidání nebo editaci mířícího zařízení, se kterým aplikace pracuje. Mířidla hrají důležitou roli v aplikaci. Na základě informací uvedených v mířidlech je aplikace schopná spočítat velmi přesné úpravy míření nebo poskytnout rady, jak přenastavit míření během lukostřelecké soutěže. Po každém zaznamenaném výstřelu provede Atremis komplexní analýzu pohybových trendů několika posledních výstřelů a porovná je se schopnostmi střelce. Pokud na základě výpočtů aplikace usoudí, že by bylo výhodné upravit nastavení míření, upozorní uživatele notifikací včetně informací, jaké úpravy míření je třeba udělat (v obou směrech). Další funkcí je „Bow-Setup“ (nastavení luku). Zde jsou uchovávány údaje o luku, který střelec používá. Nastavení luku slouží k uchovávání parametrů, které většina lukostřelců používajících reflexní a skládané luky využívá k vylepšení svých luků. Poslední funkcí na této obrazovce je „Quiver/Arrow-set“ (toulec, nebo množina šípů). Toulec, nebo množina šípů obsahuje všechny parametry jednotlivých šípů. Uživatel také může jednotlivé šípy přidávat, mazat a editovat. Důležitou součástí Quiver/Arrow-set je také možnost ohodnotit si každou sadu šípů zvlášť. [4]

Třetí záložka se nazývá „Records“ (záznamy). Záložka „záznamy“ slouží ke schromáždění maximálního počtu informací o atletově střelbě na terč, ať už během tréninku nebo lukostřelecké soutěže. Záložka obsahuje také nastavení mířidel a časomíru pro střelbu. Pro pochopení významu následujících fukncí je potřeba definovat, co se v Artemis rozumí pod pojmy „Round“ (kolo) a „Match“ (utkání) a jaký je mezi nimi rozdíl. Utkání je z pohledu Artemis jedna entita libovolné lukostřelecké aktivity, například 30 šípů vystřílených na 18 metrů, nebo jeden eliminační zápas atd. Kolem poté nazýváme množinu více utkání, tj. například trénink skládající se z 30 šípů na rozstřílení, 36 šípů vystřílených na 50 m do terče s průměrem 80 cm a 4 eliminačních zápasů po 15 šípech bude v Artemis zaznamenán jako 6 individuálních záznamů utkání. Kolo je kolekce utkání, kde celkové skóre kola je součet všech skóre z jednotlivých utkání. Například „World Archery 1440“ mužská soutěž je kolekce 4 utkání, každé utkání se skládá z 36 šípů na 90 m a 70 m na terč o průměru 122 cm a 36 šípů na 50 m a 30 m na terč o průměru 80 cm. Celkové skóre dosažené v soutěži je poté součet skóre dosaženého v jednotlivých utkáních. [68] Funkce „Round“ (kolo) zobrazí seznam všech kol, které lukostřelec absolvoval včetně skóre, počtu vystřelených šípů atd. Umožňuje také přidat nové kolo; během vytvoření nového kola uživatel musí vybrat z nabídky předpřipravených kol, např. „World Archery 1440“, nastavení luku, množiny šípů, vzdálenosti a dalších parametrů. Každé kolo v seznamu je možné editovat, sdílet jako dokument ve formátu pdf, sdílet pomocí QR kódu anebo smazat. Ve funkci „Match“ (utkání) je na výběr ze seznamu utkání, která sem uživatel dříve přidal. Při vytvoření nového utkání si uživatel zvolí luk, množinu šípů atd. stejně jako při vytváření kola, navíc ale zde musí vybrat i terč, na který bude střílet, a zvolí si formát střelby, například 3x10 šípů. Po vytvoření nového utkání se uživateli zobrazí obrázek terče, kam postupně výstřely zanáší, a má možnost zvolit z množiny šípů šíp, který byl pro konkrétní výstřel použit. Součástí

obrazovky pro zanesení šípu je tabulka, kde se zobrazuje číslo aktuálního kola, bodové ohodnocení dosažené konkrétním výstřelem, skóre dosažené během kola a skóre dosažené celkově (obrázek 2.2c). V rámci obrazovky pro zanášení šípů během střelby je na výběr mezi několika záložkami. První záložka „Timer“ (časomíra) slouží pro odpočet času na střelbu. Druhá záložka „Input“ (vstup) se používá pro zanášení pozice šípu na terč, jak je popsáno výše. Třetí záložka „Scorecard“ (karta se skóre) slouží pro znovuzobrazení skóre z předchozích sad. Na záložce „Group“ (skupina) jsou graficky znázorněné trendy ve střelbě, to jest část terče, která byla zaznamenanými výstřely nejčastěji zasáhnuta. Záložka „Arrows“ (šípy) zobrazí terč a čísla jednotlivých šípů vedle pozice, na které jsou na terči zaneseny. Na poslední záložce „Shots“ (Výstřely) jsou zaneseny jednotlivé výstřely vždy na individuálním obrázku terče. Další dvě fuknce na záložce „Záznamy“ jsou Sight-Settings (nastavení mířidel) a Timer (časomíra), které již byly popsány v textu dříve. [4]



(a) : Lukostřelec **(b)** : Nastavení **(c)** : Utkání

Obrázek 2.2: Ukázkové obrazovky z aplikace Artemis

Poslední záložka v aplikaci se nazývá „Analyze“ (analýza). První dvě funkce se jmenují „Multi-View“ (multi-pohled) a „GroupView“ (skupinový pohled). V dokumentaci je v době psaní této práce pod těmito dvěma kategoriemi uvedena pouze zkratka „TBD“, neboli „to be done“, a proto zde nelze podrobněji popsat dané funkce. Třetí funkce se nazývá „Graphs“ (Grafy) a obsahuje funkcionality, které zobrazí uživateli v grafu, kolik šípů vystřílel za určité časové období nebo jaké bylo jeho dosažené skóre. V grafu je také možné si zopbrazit „Accuracy“ (přesnost), „Skill Level“ (Úroveň dovednosti) nebo „Timing“ (Časování). Ani o jedné z těchto funkcí není v dokumentaci popsáno, jak je Artemis určuje. Poslední fuknce se nazývá „Sjef's Arrow Selector“ (Sjefův selektor šípů) Tato funkce vyhodnocuje přesnost jednotlivých šípů. Přesnost je určena vzdáleností šípu od středu terče. Výsledkem této analýzy

je seznam šípů podle jejich přesnosti a počtu výstřelů, ke kterým byl šíp použit. Tato funkce slouží k výběru toho nejvhodnějšího šípu pro rozstřel v okamžiku, kdy mají oba závodnící stejné skóre. Tudíž si oba lukostřelci vyberou jeden šíp a ten z nich, který se tímto posledním výstřelem přiblíží více středu terče, se stává vítězem. [4]

Nastavení se uživateli zobrazí po kliknutí na tlačítko v pravém horním rohu aplikace. V nastavení má uživatel přístup k obecným nastavením, jako jsou jazyk, barevné schéma apod. Může si zde změnit licenci, pod kterou aplikaci používá, upravit informace o atletovi a poté specifické nastavení jednotlivých funkcí aplikace, např. v jaké tabulce se budou zobrazovat výsledky střelby, v jaký den začíná týden apod. V nastavení se také dají tvořit zálohy dat (bude detailně popsáno v sekci Ukládání a sdílení dat). Nakonec skrze nastavení jde také propojit aplikaci s dalšími přístroji pro střelce, jako je senzor srdečního tepu, BOWdometer (aplikace pro sbírání dat o střelbě) nebo RyngDyng system (aplikace pro určování pozici šípu v terči).

2.2.3 Ukládání a sdílení dat

Ukládání dat o střelbě je v Artemis řešeno pomocí takzvané „aktivní databáze“. Všechny informace jsou ukládány na zařízení, na kterém je Artemis provozována, a nejsou sdílena s žádnou třetí stranou. Pokud je tedy zařízení ztraceno nebo jsou data poškozena, nejdou data obnovit. Artemis umožňuje si vytvořit databázové snapshoty přímo v aplikaci a tyto snapshoty jsou poté použitelné pro obnovení ztracených dat. Artemis nabízí funkci pro sdílení dat mezi závodníky a trenéry. Tato funkce umožňuje zapnout na obou zařízeních databázovou synchronizaci za použití Google Drive. Synchronizace závodníka s trenérem se poté skládá ze dvou kroků. V prvním kroku závodník přenese snapshot své databáze na Google Drive. Ve druhém kroku si ho poté trenér stáhne na své zařízení a importuje ho do Artemis.

2.2.4 Výhody aplikace

Artemis integruje rozhraní pro komunikaci a sdílení informací mezi trenérem a závodníkem. Protože tvůrce byl členem Nizozemské lukostřelecké reprezentace, odpovídají funkce aplikace interakci mezi trenérem a závodníkem během tréninkového cyklu. Příkladem takové interakce je potřeba zapisovat si data o duševní a fyzické formě. Střelec je si také schopen uložit velmi detailní data o luku, šípech a mířících zařízeních, které při tréninku využívá. Během

vytváření tréninku je uživateli umožněno velmi detailně si zaznamenat data o tréninku. Po zanešení prvních tří šípů na terč je již aplikace schopná tvořit grafy, které zobrazují průběh atletovy střelby a také trendy v jeho střelbě. Celý průběh tréninku je poté Artemis schopná konvertovat do dokumentu ve formátu pdf a sdílet. Atremis má také poměrně rozsáhlou dokumentaci, ve které je detailně popsaná většina funkcionalit, které je shopná uživateli poskytnout. Na poslední záložce má poté uživatel možnost tvorby mnoha typů grafů zobrazujících data o jeho střelbě.

2.2.5 Nevýhody aplikace

Vzhledem ke stáří aplikace a vlastnímu UI desingu, který byl použit při jejím vývoji, je pro nového uživatele značně obtížné se v Artemis zorientovat. Jednotlivé UI komponenty byly navrhnuté a nadesingované samotným autorem v době, kdy aplikace na chytrých telefonech vypadaly odlišně, a od té doby se příliš nezměnily. Problémem je i špatná viditelnost ikon jednotlivých funkcí na barevném pozadí. Artemis nejde propojit s Google účtem ani jinou aplikací. Další problém je v implementaci kalendáře, který není možné propojit s Google kalendářem nebo jiným kalendářem již používaným na zařízení; uživatel si tedy může naplánovat události, ty se mu ale zobrazí pouze po otevření Artemis a není možné je synchronizovat s událostmi v jiném kalendáři. Při vytváření tréninku má pak střelec na výběr jen mezi čtyřmi typy terčů. Po vytvoření tréninku chybí tlačítko, kterým by se atlet přesunul do režimu střelby, takže pokud chce atlet dokončit proces vytváření tréninku a začít střelbu, musí se po vytvoření tréninku vrátit do seznamu tréninků a tam zvolit pro střelbu nově vytvořený trénink. Aplikace neumožňuje uživateli editovat parametry tréninku vytvořeného v minulosti. Značně uživatelsky nepřívětivé je také potvrzení umístění šípu na terči, které se aplikuje dvojklikem na obrazovku. Limitující je také chybějící verze aplikace pro operační systém iOS. Bohužel dokumentace, která je na stránkách výrobce a autora, neobsahuje kapitoly, kde by byl popsán postup, podle kterého jsou vytvářeny grafy na záložce „Analýza“; je proto obtížné interpretovat funkci některých grafů, které je aplikace schopná vytvořit.

2.2.6 Shrnutí

Artemis je vhodnou volbou pro profesionální nebo poloprofesionální lukostřelce, kteří chtějí dlouhodobě zaznamenávat svůj sportovní výkon. Je také vhodnou volbou pro atlety, kteří koordinují svůj trénink s trenérem a je pro ně důležité vzájemné sdílení a vyhodnocování dat o střelbě. Je ale také

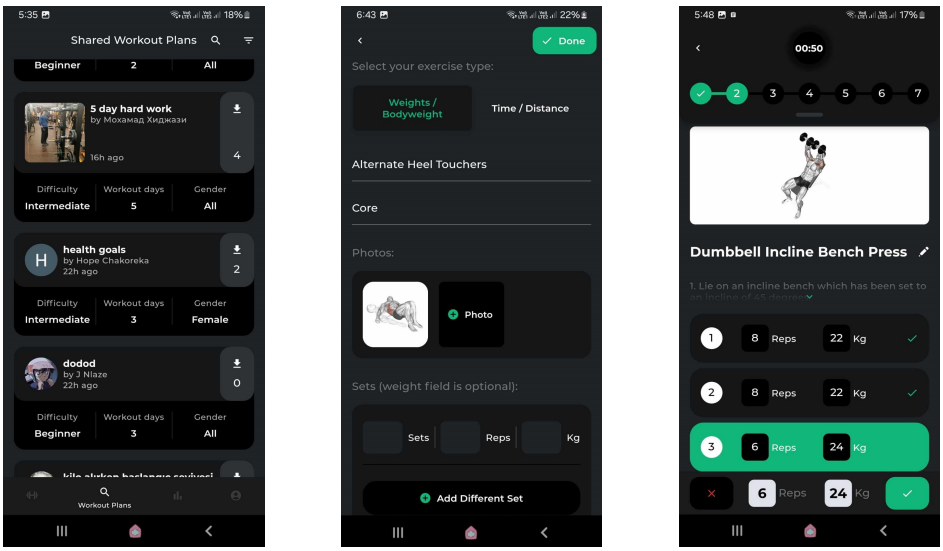
poměrně uživatelsky nepřívětivá a pro práci s ní je nutné detailně se seznámit s dokumentací.

2.3 My Workout Plan - Gym Tracker

Aplikace „My Workout Plan - Gym Tracker“ byla vytvořena studiem SosisApps sídlícím v Israeli. Jedná se o užitečný nástroj pro ty, kteří se věnují cvičení v posilovně. Pomáhá uživatelům sledovat jejich tréninky, stanovovat si cíle a sledovat vývoj své formy v čase. Aplikace v současné době podporuje jak chytré telefony s operačním systémem Android, tak chytré telefony s operačním systémem iOS. Aplikace je dostupná ve verzi zdarma, nebo PRO. [39]

2.3.1 Uživatelské rozhraní

Artemis se skládá ze 4 záložek „Profile“ (profil), „Statistic“ (statistika), „Workout Plans“ (tréninkové plány), „My Workout“ (můj trénink) – obrázek 2.3a. Každá záložka má v sobě implementovanou 2. vrstvu záložek, mezi kterými se uživatel může pohybovat pomocí swapu doleva a doprava. Aplikace kombinuje tmavé pozadí s bílým textem a zelenými tlačítky.



(a) : Seznam tréninků **(b) :** Editace cviku **(c) :** Probíhající trénink

Obrázek 2.3: Ukázkové obrazovky z aplikace My Workout Plan - Gym Tracker

2.3.2 Funkce aplikace

V záložce „Profil“ je uživatel schopný si propojit aplikaci s účtem Google Play nebo Apple Play. Uživatel také může v parametrech svého profilu uvést, že je certifikovaný trenér, a vyplnit informace o sobě včetně profilové fotografie. Na záložce „History“ (Historie) jsou dvě záložky „Calendar“ (Kalendář) a „Statisic“ (Statistiky). Na záložce „Kalendář“ se zobrazuje kalendář s historií absolvovaných tréninků uživatele. Na záložce „Statistika“ jsou pak vidět statistiky utvořené na základě uživatelova tréninku. Třetí záložka se nazývá „Tréninkové plány“ a umožňuje uživateli vybrat si ze seznamu tréninkových plánů připravených a sdílených jinými uživateli, anebo přes formulářové okno přidat do tohoto seznamu nový tréninkový plán. Na záložce „Můj trénink“ si uživatel může vybrat ze svého seznamu tréninků anebo vytvořit si nový trénink. Pokud se rozhodne vytvořit nový trénink, tak postupně vybere cviky z předpřipraveného seznamu cviků, vyplní počet opakování každého cviku a počet sérií, které chce při tréninku provádět, a případně i váhu závaží, se kterým cvik provádí (2.3b). Může si také nastavit, jak dlouhé přestávky chce mezi jednotlivými cviky dělat. Uživatel může do seznamu přidávat i nové cviky. V okamžik, kdy je trénink připravený, ho uživatel může zahájit. Po zahájení se vygeneruje seznam všech sérií a opakování cviků, které postupně uživatel plní, a aplikace ho takto provází celým tréninkovým procesem (2.3c). Na konci tréninku vytvoří aplikace graf, kde je zobrazeno, jaké části těla a v jakém poměru byly během tréninku zatíženy.

2.3.3 Ukládání a sdílení dat

Aplikace je napojená na Google Play nebo Apple Play účet a automaticky se synchronizuje, pokud je uživatel přihlášený na více zařízeních najednou nebo pokud se rozhodne zařízení vyměnit. Druhou variantou sdílení dat je, že uživatelé mohou svoje tréninkové plány přidávat do společného seznamu, ze kterého si ho poté může kdokoliv stáhnout do svého zařízení a spustit.

2.3.4 Výhody aplikace

Aplikace má skromné a intuitivní uživatelské rozhraní. Díky propojení s účtem na Google Play nebo Apple Play se dá paralelně používat na více zařízeních. Velmi užitečná funkce je výběr z již existujících tréninkových plánů, které sdíleli ostatní uživatelé. Nový uživatel se tak může při svém tréninku inspirovat při sestavování tréninku anebo si vybrat ze seznamu už připravený trénink.

U každého cviku je animace, na které je zobrazeno, jak se má cvik správně provádět, a v detailu je i textový popis daného cviku.

2.3.5 Nevýhody aplikace

Během provádění cviků musí uživatel po každé sérii kliknout na malé tlačítko ve spodní části obrazovky, aby odstartoval novou sérii, což není v posilovně příliš praktické. Existuje pouze filtr, který ukazuje počet stažení tréninkového plánu, chybí tedy jakékoliv jiné hodnocení, jak s tréninkem byli nebo nebyli uživatelé spokojeni. Při vybírání tréninkového plánu ze seznamu není nijak poznat, jestli je uživatel, který tam trénink přidal, trenér, nebo nikoliv. V aplikaci není rozhraní, které by dovolovalo vytvořit soukromý trénink pro jiného uživatele, aniž by se nezobrazoval ve všeobecném seznamu tréninků – není to tedy příliš vhodné pro soukromé sdílení tréninků mezi trenérem a závodníkem. Pro aplikaci neexistuje zpracovaná dokumentace.

2.3.6 Shrnutí

Rozhraní je v porovnání s My Targets nebo Artemis minimalistické a přehledné již při prvním použití aplikace. Při provádění cviku vidí uživatel animaci, která mu zobrazuje, jak má být cvik prováděn, takže do jisté míry simuluje roli trenéra v tréninku. Možnost sdílet nebo si vybrat ze seznamu sdílených tréninků a stáhnout si je do svého zařízení činí aplikaci velmi uživatelsky přívětivou i pro fitness začátečníky. Chybí zde ale rozhraní pro personalizovaný trénink, což činí aplikaci méně vhodnou pro vrcholové sportovce a jejich trenéry, kteří chtějí uchovávat svoje tréninkové plány v soukromí.

Kapitola 3

Požadavky na aplikaci

Se sběrem požadavků se setkáme u návrhu každé aplikace. Jeho cílem je definovat si požadavky, které mají být v rámci aplikace implementovány tak, aby výsledný produkt odpovídal potřebám zákazníka. Specifikace vývojářského teamu jsou důležité pro porozumění požadavků zákazníka. Pokud jsou požadavky správně zaznamenány, má to přímý vliv na plnění časového harmonogramu projektu a nasmlouvané ceny. Pokud jsou požadavky zaznamenány nekvalitně nebo v nedostatečném rozsahu, vznikají nedorozumnění a projekt se prodlužuje i prodražuje. [38] Ačkoliv existuje několik typů požadavků (např. systémové, uživatelské, funkční), v rámci této kapitoly budou zanalyzovány pouze funkční požadavky na aplikaci, a to protože jsou z pohledu programátora základem pro návrh aplikace.

3.1 Vlastnosti požadavků

Požadavky by měly být správně zpracovány. Protože je běžné, že se požadavky navzájem ovlivňují nebo si dokonce protiřečí, je k vytvoření kvalitních požadavků vhodné je pečlivě zaznamenávat. Jako příklad uvádím výčet vlastností požadavků podle Karla E. Wiegerse [66]. Každý z požadavků by podle něj měl obsahovat následující vlastnosti:

Úplnost: Všechny požadavky by funkcionalitu měly popisovat úplně. To znamená, že by měly obsahovat všechny informace potřebné pro návrh a implementaci.

Správnost: Všechny požadavky by měly být popsány přesně. To znamená, že by měly odpovídat požadavkům uživatele a korespondovat s nadřazenými systémovými požadavky.

Proveditelnost: Už při analýze by se měly zavrhnout neproveditelné požadavky. To se dá zajistit například tím, že je k marketingovému týmu či analytikovi přiřazen vývojář, který je schopen rychle určit, co je proveditelné a co ne.

Nepostradatelnost: Každý požadavek by měl být nepostradatelný. To znamená, že by měl představovat funkci, kterou zákazník vyžaduje nebo která je vynucená vnějším systémem či standardem.

Priorita: Každému požadavku by měla být přidělena priorita, která určuje, jak moc je daná věc důležitá pro vydání celého systému. Priorita je důležitá pro tvorbu termínů, rozpočtu, kvalitativních cílů a slazení s dostupným personálem. V případě, že je v průběhu vývoje přidán další požadavek, lze vypustit nebo odsunout méně důležitý požadavek na další verzi systému.

Jednoznačnost: Požadavek by měl být jednoznačný, aby každý čtenář mohl dojít pouze k jednomu výkladu. Aby se tohoto dosáhlo, je potřeba požadavek popisovat jednoduchým, stručným, srozumitelným a přímočarým jazykem.

Ověřitelnost: Správnost implementace požadavku by měla jít ověřit testem.

3.2 Metoda sběru požadavků

Ačkoliv existuje velké množství metod, kterými lze sbírat požadavky, rozhodl jsem se v této kapitole popsat pouze ty metody, které jsem použil při sběru požadavků v rámci tvorby návrhu a implementace sportovního deníku. Metody, které zde budu popisovat, jsou blíže popsány v knize [66]. První metoda je formulována jako „Recyklujte požadavky z předchozích projektů“. Konkrétně jde o medotu, která je založená na již existujícím požadavku implementovaném v rámci nějakého hotového projektu. V rámci této práce se jedná například o požadavek UCHP11, který je založen na požadavku pro zaznamenání pozice šípu na terči, které je vysvětleno a popsáno v rámci funkcí apikace My Targets/Artemis. Další metoda, kterou jsem použil, je popsána jako „Vyberte skupinu typických uživatelů“. Principem této metody je výběr uživatele nebo skupiny uživatelů, kteří používali nějaký podobný výrobek (v tomto případě sportovní deník určený pro lukostřelce) a zjištění jejich názoru na funkce a kvalitu tohoto výrobku. Poslední metoda zní jako „Sledujte uživatele při práci“. V rámci této metody jsem se zúčastnil lukostřeleckého tréninku

ID	Název aktéra	Popis aktéra
NA	Návštěvník	Osoba přistupující k systému. Má pouze možnost přihlášení do systému a získání vyššího oprávnění.
AT	Atlet	Přihlášená osoba, která má základní oprávnění pro používání aplikace. Systém používá za cílem zaznamenání svých tréninkových dat.
TR	Trenér	Přihlášená osoba, která má vyšší oprávnění pro používání aplikace. Systém používá za cílem čtení dat uživatelů s rolí Atlet a vytváření nových tréninků pro tyto uživatele.

Tabulka 3.1: Tabulka aktérů

ID	Název priority	Popis priority
1	Vysoká priorita	Požadavky, které je nutné implementovat do aplikace již v rámci této práce. V realitě by se poté jednalo o požadavky, které mají být implementovány např. v rámci tvorby prototypu aplikace pro zákazníka.
2	Střední priorita	Požadavky, které je potřeba implementovat, aby byla aplikace v praxi pro uživatele použitelná.
3	Nízká priorita	Požadavky, které nejsou zásadní pro chod aplikace, ale poskytují uživateli nadstandartní funkcionalitu.

Tabulka 3.2: Tabulka priorit požadavků

a sledoval, jakým způsobem probíhají všechny činnosti spojené se střelbou na terč a následným zaznamenáváním tréninku do deníku.

3.3 Seznam požadavků

Na základě požadavků, které jsem sesbíral podle metodik popsaných v předchozím odstavci, jsem sestavil seznam aktérů Table 3.1 a popsal jejich role při interakci s aplikací. V rámci sbírání požadavků jsem určil u každého požadavku jeho prioritu při implementaci. Popis priorit jednotlivých požadavků je popsán v tabulce Table 3.2. Na základě předchozích dvou seznamů a sesbíraných požadavků jsem vytvořil seznamy požadavků s vysokou, střední a nízkou priritou. V rámci této kapitoly jsem implementoval tabulku s vybranými uživatelskými požadavky Table 3.3. Kompletní uživatelské požadavky jsou dostupné v příloze Appendix B.

ID	Název požadavku	Popis požadavku	Aktéři
UCHP11	Zaznamenání střelby na terči	Po zahájení střelby se uživateli zobrazí terč, na kterém může pohybovat kurzorem umístěným uprostřed obrazovky a zaznamenat tak pozici šípu. V horní části obrazovky je tabulka, kde se uživateli zobrazuje skóre (UCHP12). Ve spodní části obrazovky jsou umístěna tlačítka „předchozí“ a „použít“. Po kliknutí na tlačítko „předchozí“ je poslední uživatelem zaznamenaný šíp smazán. Tlačítkem „použít“ uživatel zaznamená aktuální pozici kurzoru na terči. Po zanesení posledního šípů je uživatel přesměrován zpět na záložku se sadami šípů.	AT
UCMP4	Časomíra	Po zahájení tréninku se spustí časomíra, která uživateli určí dobu, po kterou se nejprve připraví na střelbu (20 s), poté dobu, po kterou může střílet (140 s), a upozorní ho 30 s před koncem časového limitu.	AT
UCLP4	Alternativní zanášení pozice šípu na terči	Implementace zanášení pozice šípu na terči pomocí kurzoru, který bude uživatel schopný přesouvat po terči a tím zaznamenat pozici šípu. Jde tedy o opačný postup k základnímu zanášení šípu popsanému v (UCHP11).	AT

Tabulka 3.3: Tabulka s vybranými uživatelskými požadavky

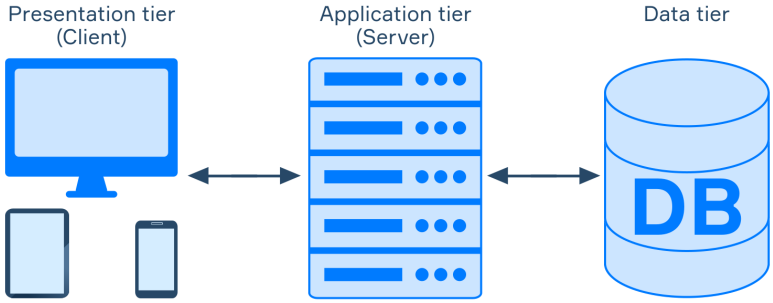
Kapitola 4

Návrh aplikace

Cílem této kapitoly je poskytnout čtenáři popis, jakým způsobem bude postupováno během implementace. Podrobně zde budou rozebrány jednotlivé technologie jak pro frontend, tak pro backend a bude vysvětleno, proč byly při implementaci použity. Dále pak budou popsány architektura aplikace a databázové schéma.

4.1 Architektura aplikace

Při zkoumání, jak navrhnout architekturu aplikace, jsem analyzoval dvě varianty implementace: monolitickou a třívrstvou architekturu. V případě monolitické architektury by byla vytvořena pouze mobilní aplikace, ve které bude implementována veškerá business logika, a databáze bude uložená v souboru přímo na disku v telefonu. Zapisování a čtení z databáze by obsluhovala pouze mobilní aplikace. Tento implementační přístup byl použitý jak v případě Artemis, tak v případě My Targets. I když provoz obou aplikací dokazuje, že je tento přístup funkční a disponuje výhodou oproti implementaci za pomoci API, a to konkrétně, že pro provoz uživatel nepotřebuje přístup k internetu, v rámci mé implementace jsem ho zavrhnul, protože důležitým požadavkem pro aplikaci je sdílení dat mezi trenérem a závodníkem. Pokud bych se rozhodl používat přístup založený na ukládání dat do lokálního souboru, musel by atlet po každém novém záznamu přenášet celou databázi ze svého zařízení do zařízení trenéra a ten by ho musel to své aplikace importovat. Zároveň, pokud by závodník měl například zadaný trénink od trenéra, musel by soubor přenášet v opačném směru a poté ho do aplikace



Obrázek 4.1: Schéma třívrstvé architektury [57]

na závodníkově zařízení importovat. Sdílení dat tímto způsobem je časově náročné a komplikované pro uživatele.

Při implementaci pomocí třívrstvé architektury (někdy také označované jako třístupňová architektura) se aplikace rozdělí na několik částí, které spolu budou vzájemně komunikovat (obrázek 4.1). Jednotlivé vrstvy se nazývají prezentační vrstva, business vrstva, datová vrstva. Vrstvy jsou takto rozděleny za účelem rozdělení kompetencí v rámci aplikace. Prezentační vrstva je jediná vrstva, kterou uživatel vidí. Slouží k prezentaci vstupních dat uživateli a také jako vrstva, se kterou uživatel interaguje, takže zároveň je také schopná přijímat vstupní informace od uživatele. Aplikační vrstva obsahuje buisiness logiku aplikace, jedná se o prostřední vrstvu v modelu a účelem této vrstvy je provádět výpočty, transformace a operace s daty z ostatních vrstev. Aplikační vrstva má také na starost komunikaci mezi prezentační a datovou vrstvou. Nejnižší vrstva se nazývá datová. Zajišťuje operace s daty, uložení dat, zajištění integrity atd. Výhodou třívrstvé architektury je, že jednotlivé vrstvy mohou být vyvíjeny a implementovány odděleně, pokud mají pevně specifikované rozhraní pro komunikaci mezi sebou, a také to, že jsou na sobě nezávislé. [71]

4.2 Frontend

Jako operační systém, pro který budu aplikaci vyvíjet, jsem si určil Android. Důvodem pro to bylo, že veškeré aplikace, jejichž funkcionalitu jsem mapoval v rámci rešerše, jsou určené pro chytré telefony s operačním systémem Android. Dalším důvodem je, že mnou vlastněný a používaný telefon má stejný operační systém a mohu tak aplikaci také testovat na skutečném zařízení a nikoliv pouze v emulátoru. Navíc podle webu Statcounter GlobalStats je Android nejpoužívanějším operačním systémem (71%) [56]. Pro implementaci klienta (prezentační vrstvy) jsem zvolil programovací jazyk TypeScript [58] a framework React Native [46]. V rámci výběru technologie, kterou budu

používat, jsem zvažoval také programovací jazyk Kotlin, který je určený přímo pro vývoj aplikací pro operační systém Android. Důvod, proč jsem se rozhodl pro technologii React Native, je, že je vývoj v ní mnohem podobnější vývoji webových aplikací, se kterým mám již zkušenosti, a zároveň je multiplatformní. Pokud bych tedy v budoucnu chtěl vytvořit verzi aplikace i pro operační systém iOS, bude to možné. V rámci jazyku Kotlin sice také existují frameworky pro multiplatformní aplikace jako Kotlin Multiplatform a Kotlin Native, ale v porovnání s React Native jsou výrazně mladší a mají menší uživatelskou základnu. V rámci React Native se pro psaní kódu používá programovací jazyk JavaScript, jenž je objektově orientovaný, avšak není silně typovaný. Proto, abych si v rámci projektu zajistil silné typování, jsem přidal nadstavbu jazyka JavaScript, a to TypeScript. TypeScript zakazuje některé způsoby zápisu kódu, které jsou v rámci jazyka JavaScript možné, jako deklarace proměnných pomocí slova „var“ nebo uložení textu (datový typ string) do proměnné, která byla deklarována jako proměnná datového typu number (typ number slouží pro uchovávání číselné hodnoty). Silné typování je v TypeScriptu povinné u všech proměnných a funkcí, i když programátorovi teoreticky umožňuje vyhnout se tomu pomocí klíčového slova „any“. Při správném použití těchto programovacích technik je pak program schopný odhalit chybu způsobenou špatným typem objektu již při kompilaci a ne až za běhu, jak je tomu v případě JavaScriptu. Veškerý kód, který je napsaný v TypeScriptu, se transpiluje do JavaScriptu, který se poté vykonává v prohlížeči, takže i kód, který není napsaný v souladu s pravidly TypeScriptu, může být v prohlížeči spustitelný, jak je popsáno v dokumentaci [34]. Pro řízení závislostí v rámci klientské aplikace jsem zvolil nástroj npm (Node Package Manager) [41], alternativně jsem mohl použít i balíčkovací systém Yarn [69], ale s balíčkovým systémem npm mám již zkušenost z předchozího studia, proto jsem se rozhodl pro něj.

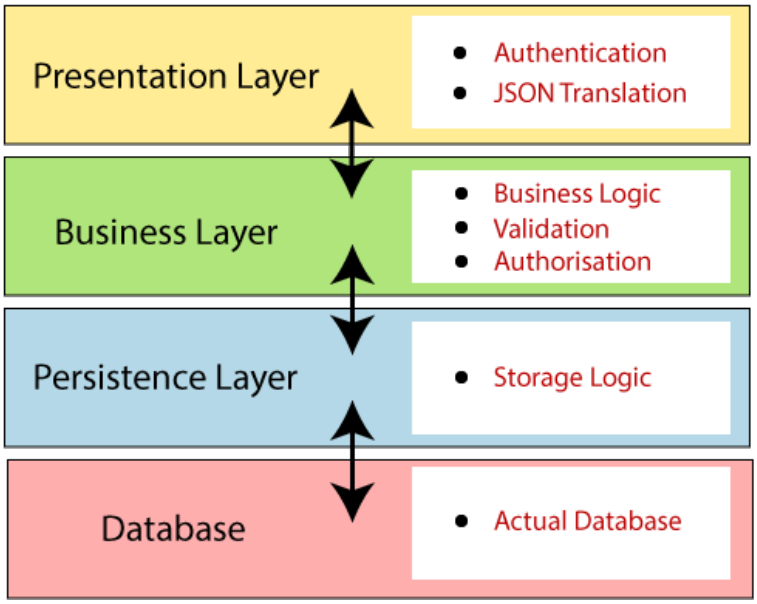
4.3 Backend

Pro implementaci serverové (business) vrstvy aplikace jsem zvolil programovací jazyk Java. Důvodů, proč jsem zvolil tento programovací jazyk, je několik. Za prvé, jedná se o objektově orientovaný programovací jazyk. Dalším důvodem bylo, že Java je silně typovaný programovací jazyk. Na rozdíl od JavaScriptu nebo Pythonu je už během kompilace možné odhalit množství chyb, které můžou nastat za běhu programu. V poslední fázi jsem se rozhodoval, zda implementovat serverovou vrstvu v objektově orientováném jazyku z rodiny .NET a nebo Java. Pro Javu jsem se nakonec rozhodl, protože s programováním v Javě mám zkušenosti ze studia na ČVUT Fakultě Elektrotechnické, zatímco s konkurenčními objektově orientovanými jazyky z rodiny .NET žádné zkušenosti nemám. Pro implementaci jsem se rohodl použít framework Spring Boot [50]. Jako alternativu jsem zvažoval, zda nepoužít Spring framework

(Spring Boot je nadstavba Spring frameworku), ale Spring Boot na rozdíl od „čistého“ Spring frameworku již obsahuje předkonfigurovaný servlet kontejner (Tomcat). Pro automatické sestavování aplikace jsem použil nástroj Gradle [23], alternativně jsem mohl použít nástroj Maven. Důvod, proč jsem si zvolil Gradle, je, že na rozdíl od Mavenu se dá více přizpůsobit individuálním potřebám projektu a také implemtuje programovací jazyk Groovy [3] pro práci s ním, zatímco Maven používá pro svoji konfiguraci jazyk XML. [24]

Spring framework je jeden z nejpopulárnějších frameworků pro vývoj webových aplikací. Ačkoliv se stal velmi populárním stejně jako například framework Jakarta Enterprise Edition [31], potýkal se s obtížnou konfigurací servlet kontejneru, na kterém aplikace běží. Tento problém vyřešil až Spring Boot, který obsahuje již předkonfigurovaný servlet kontejner (Tomcat). Spring a Spring Boot jsou charakteristické tím, že mají ve svém jádru implementovaný návrhový vzor IoC (Inversion of Control). IoC uvolňuje pevné vazby mezi objekty a v rámci Spring Boot aplikací přesouvá odpovědnost za vznik vazeb z programátora na framework. S IoC přímo souvisí také návrhový vzor Dependency Injection. Tento mechanizmus má na starosti spravování závislostí mezi jednotlivými třídami. Například, pokud se vytvoří instance třídy A, která je závislá na třídě B, je instance třídy B vložena (injektována) do třídy A. O tuto injekci se stará framework. Objekty, jejichž životní cyklus má na starost IoC container manager, se označují ve Spring Bootu jako „beany“. „Beana“ (anglicky „bean“) je objekt, který vykonává nějakou funkci, například zajišťuje připojení do databáze. Beany se nachází v kontejneru po celou dobu běhu aplikace. IoC container manager je vždy v aplikaci pouze jeden a je součástí jádra aplikace. [50]

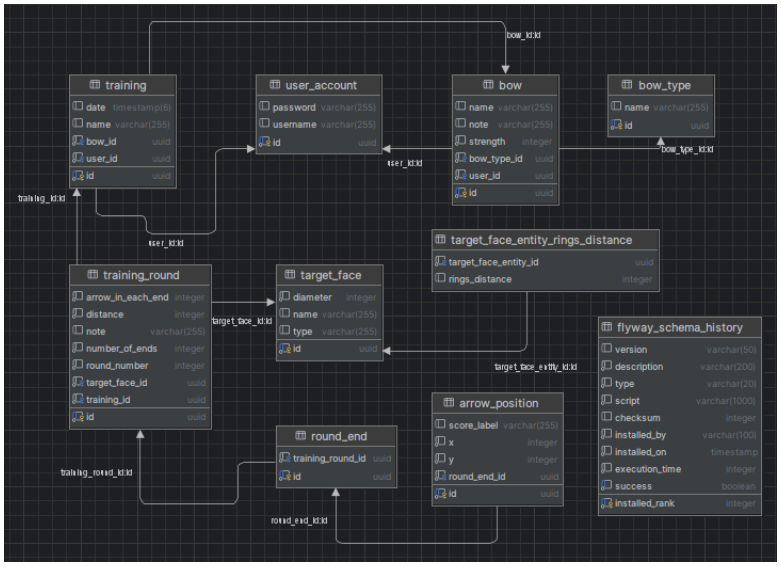
Architektura Spring Boot frameworku se skládá ze 4 vrstev: prezentační vrsty, business vrstvy, perzistenční vrstvy a databázové vrstvy. Prezentační vrstva je první vrstvou, která je zodpovědná za zpracování HTTP requestů, mapování dat ve formátu JSON na objekty, autentifikaci a transformaci dat, která poté předá dále business vrstvě. Business vrstva je zodpovědná za business logiku aplikace jako výpočty, transformace a operace s daty. Perzistenční vrstva slouží pro manipulaci s daty a jejich převodem mezi business vrstvou a záznamem v databázi. Databázová vrstva poté slouží k provádění databázových operací CRUD (create, retrieve, update, delete). Důležitou vlastností této architektury je, že spolu mohou komunikovat pouze sousední vrstvy, a proto má každá z těchto vrstev pevně danou kompetenci v rámci aplikace. [52] Rozdělení vrstev v rámci Architektury Spring Boot aplikací zachyceno na obrázku 4.2



Obrázek 4.2: Schéma architektury Spring Boot frameworku. [53]

4.4 Popis databáze

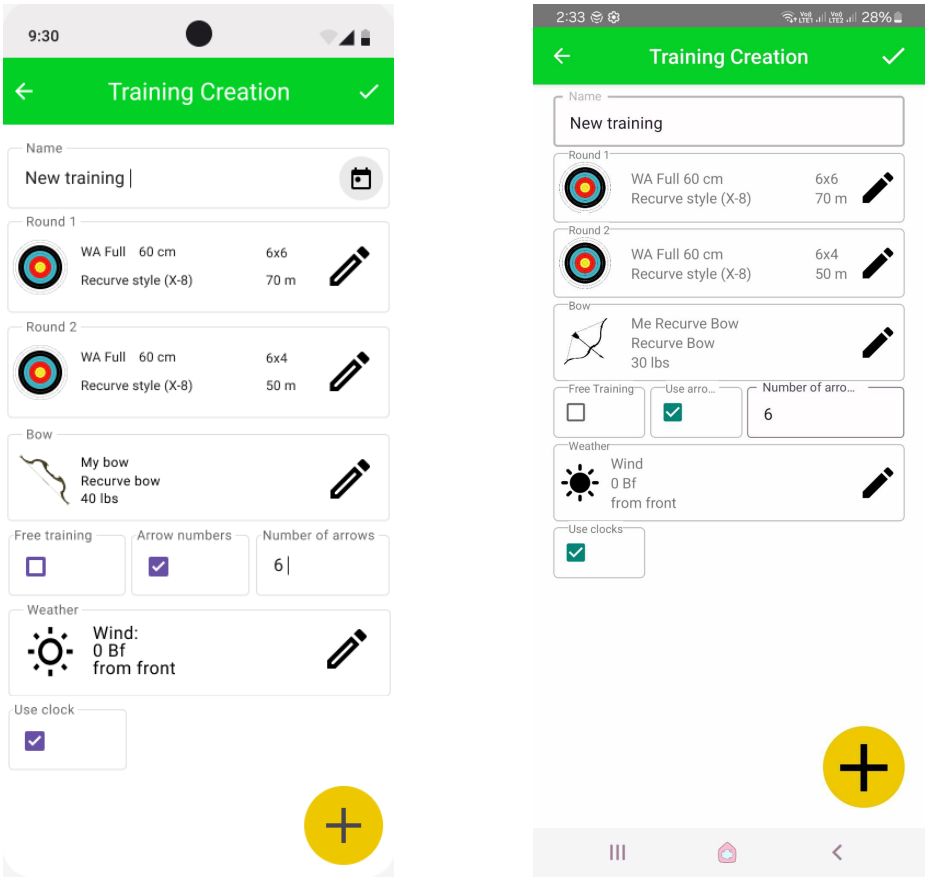
V rámci popisu serverové části aplikace jsem krátce popsal smysl i funkci perzistenční vrstvy. Blíže se této problematice budu věnovat zde v rámci popisu standardu programovacího jazyka Java nazývaného JPA (Java Persistence API). JPA slouží pro objektově relační mapování, to zjednodušuje ukládání objektů do databáze a jejich načítání z ní. Je-li vytvořena entitní třída, v databázi je pak tato třída reprezentována tabulkou, a instance této třídy budou v tabulce reprezentovány jednotlivými řádky. Jednotlivé atributy entity se ukládají do příslušného sloupce v rámci databáze. Deklarovným atributům v rámci entity lze pomocí anotace přiřadit speciální roli, například primárního klíče nebo cizího klíče (podrobnější popis je k nalezení v kapitole Implementace). Výhodou použití této technologie je, že programátor nemusí zvlášť vytvářet databázi a zvlášť serverovou část aplikace, databáze je generovaná na zákaldě entitních tříd [33]. Schéma takto vygenerované databáze je zachyceno na obrázku 4.3. Pro implementaci jsem zvolil databázový systém PostgreSQL [43]. Pro tento projekt je vhodný, protože se jedná o objektově relační databázový systém, který je velmi populární mezi komunitou vývojářů, takže existuje velké množství článků a návodů na to, jak ho integrovat a používat spolu s frameworkem Spring Boot. Další argument, proč ho použít, je, že jsem se s ním již během studia pracoval.



Obrázek 4.3: Databázové schéma aplikace

4.5 Návrh uživatelského rozhraní

Pro návrh uživatelského rozhraní jsem použil software pro tvorbu návrhu UI a prototypů aplikací zvaný Figma [20]. Důvodem, proč jsem zvolil právě Figmu, je možnost editovat a nahlížet projekt ve webovém prohlížeči a sdílet ho mezi více uživateli, a také jednoduché uživatelské rozhraní, jak je popsáno v článku [65]. Základem pro uživatelské rozhraní jsou komponenty převzaté z knihovny Material 3 Design Kit (Community) [37] a ikony převzaté z knihovny Material 3 Design Icons (Community) [28]. Při návrhu uživatelského rozhraní jsem vycházel z analýzy existujících aplikací Artemis a My Targets. Cílem bylo vytvořit dostatečně jednoduché uživatelské rozharní, aby se v něm zorientoval i nový uživatel. Zároveň je ale dost rozsáhlé na to, aby v něm byla zaznamenávána všechna důležitá data o tréninku. Příklad návrhu obrazovky a její následné implementaci v FE aplikaci je vidět na obrázku 4.4. Adresa pro zobrazení celého návrhu vytvořeného v Figmě je dostupná na odkazu uvedeném v příloze Návrh uživatelského rozhraní.



(a) : Figma UI **(b) :** Implementované UI

Obrázek 4.4: Porovnání návrhu UI ve Figmě a implementovaného UI

Kapitola 5

Implementace

V této kapitole je popsán postup při zakládání, konfiguraci a vytvoření repozitáře pro open-source projekt. Bude zde sepsán seznam komponent, patternů a programátorských postupů použitých při implementaci backend a frontend části aplikace. Verze jednotlivých knihoven, verze aplikací použitých při vývoji a zdrojové kódy se nacházejí v příloze.

5.1 Frontend

Tato sekce se věnuje technickému popisu frontendu, neboli klientské části aplikace (zkráceně FE). V této sekci je popsán postup implementace a jsou zde rozebránzy důležité částí této části aplikace. Protože se jedná o téma, které lze popsat skutečně velmi do detailu, tak se zde nachází pouze popis nejdůležitějších částí aplikace.

5.1.1 Založení a konfigurace projektu

Pro založení projektu byl použitý nástroj pro vývoj React Native aplikací CLI (React Native Command Line Interface). Pomocí tohoto balíčku byla vygenerována základní kostra frontend aplikace [49]. Do aplikace byl poté přidán TypeScript. Konfigurace knihoven potřebných při spuštění aplikace

se nachází v souboru package.json. V souboru .env jsou uvedeny proměnné pro připojení k backendové části aplikace. Projekt obsahuje 2 adresáře */ios* a */android*. Tyto adresáře obsahují soubory se zdrojovými kódy aplikace pro oba dva operační systémy, jak je popsáno v článku [59]. Soubor, ve kterém se nachází základní aplikační komponenta, se nazývá App.tsx.

5.1.2 Adresářová struktura

Pro adresářovou strukturu frontendové části aplikace byl zvolen přístup zvaný separate by layer (dělení podle vrstvy). Jedná se o přístup doporučený při tvorbě frontend aplikací ve frameworku React Native. [59]. Jeho principem je třídit soubory do adresářů podle vrstvy, na které se v rámci aplikace nacházejí. Takže například všechny obrazovky budou uložené v jednom adresáři pohromadě, i když se každá z těchto obrazovek vztahuje k jiné funkcionalitě. Stejně tomu bude u services („služeb“), components („komponent“) atd. v rámci celé aplikace. Popis adresářové struktury je převzaný z článku [59].

V projektu je tato metodika použitá konkrétně v adresáři */app/src*. Ukázka dělení adresářů podle vrstvy je následující:

/app/src
- `assets` ..... Obrázky, soubory obsahující komponenty se styly atd.
- `components` ..... Komponenty, které jsou používány napříč aplikací.
- `models` . Třídy a rozhraní pro předávání dat v rámci aplikace anebo REST API.
- `screens` Individuální obrazovky/stránky, které se zobrazují uživateli.
- `services` Moduly pro volání REST API a specifickou logiku aplikace, jako je ukládání a přístup k sessions, formátování dat atd.

5.1.3 Zabezpečení aplikace

Po spouštění aplikace je na úvodní obrazovce zobrazen přihlašovací formulář, do kterého uživatel musí zadat jméno a heslo, a následně je aplikací autentifikovován. Autentifikace probíhá tak, že FE provede volání BE za cílem autentifikovat uživatele pomocí jména a hesla. Pokud se autentifikace povede, FE uloží do AsyncStorage token, který byl uživateli vygenerován pro přístup do aplikace. AsynsStorage je asynchronní systém pro ukládání dat, jako jsou tokeny nebo stavy aplikace. Data jsou zde ukládána podobně jako v případě SessionStorage za pomocí klíče a hodnoty [8]. V service AbstractApiService

je poté předpřipravená funkcionalita k obohacení REST API volání o token, který aplikace získala během přihlášení uživatele. Zároveň je zde i aplikační logika, která zajistí, že uživatel bude přesměrován na úvodní přihlašovací obrazovku v okamžiku, kdy mu vyprší platnost tokenu.

5.1.4 Komponenty

Komponenty jsou bloky kódu, které se obvykle opakovaně používají v různých částech aplikace. Příkladem komponenty je komponenta pojmenovaná TextInput (textový vstup). Tato komponenta se používá pro získání vstupních dat, které uživatel zadává obvykle pomocí klávesnice do formulářového pole, například jméno a heslo během přihlášení. V Reactu existují dvě varianty, jak psát komponenty: jako funkcionální komponenty a jako třídní komponenty.

Funkcionální komponenty jsou „pouze“ stateless JavaScript funkce. Skládají se z názvu funkce, vstupních parametrů a těla. Tělo této funkce musí na svém konci vždy vracet JSX element, který reprezentuje UI komponenty. Jejich výhoda je, že při jejich volání pro ně není potřeba vytvářet novou instanci třídy. Navíc fukncionální komponenty jsou tzv. „pure functional“ (čistě funkcionální), takže pro stejný vstup bude vždy stejný výstup. Pokud programátor potřebuje držet v této komponentě data, používá k tomu tzv. „states“. States jsou objekty, které umožňují uchovávat data v rámci komponenty. Pro manipulaci s životním cyklem komponenty se používají tzv. „hooks“. Hooks jsou speciální funkce, které umožňují programu reagovat na konkrétní událost během životního cyklu komponenty; například useEffect hook je spuštěn, pokud životní cyklus komponenty dosáhne stavu componentDidMount. UseEffect hook se používá obvykle pro provedení nějaké dodatečné logiky, například stažení dat ze serveru. ComponentDidMount je funkce životního cyklu komponenty, která je zavolaná poté, co je komponenta načtena do DOM (Document Object Model) stránky. States a hooks bylo původně možné používat pouze v třídních komponentách, ale od verze React 16.8. jsou implementovány i do funkcionálních komponent. [21]

Třídní komponenty jsou založené na React třídě Component nebo PureComponent a musí od ní při své deklaraci dědit. Oproti funkčním komponentám nelze specifikovat v hlavičce vstupní argumenty, ale data (ve skutečnosti je možné jako argument vložit i funkci) jsou do nich vkládána jako argument pomocí speciální fuknce zvané konstruktor, která je zodpovědná za vytvoření instance třídy. V konstruktoru se také specifikují states. Komponenta může obsahovat tzv. lifecycle metody, pomocí kterých reaguje na události v rámci svého životního cyklu. Povinnou součástí třídní komponenty je funkce return() která vrací JSX element. [21]

5.1.5 Modely

Models (modely) jsou soubory obsahující interfaces (rozhraní) a classes (třídy), které jsou v aplikaci použity pro zachování silného datového typování. Rozdílů mezi rozhraními a třídami je několik. Nelze deklarovat instanci rozhraní, instanci třídy naopak deklaroval lze. Rozhraní slouží pouze pro zachování typování oběktu. Třída navíc obvykle obsahuje metody a jejím cílem je implementovat i logiku pro práci s objektem. Rozhraní jsou pouze virtuální struktura na úrovni TypeScriptu, zatímco třídy se transpilují do JavaScriptu. [14]

5.1.6 Obrazovky

Screens (Obrazovky) slouží pro organizování aplikace do sekcí. Jsou obvykle pevně spjaté s routováním aplikace. Routování je funkcionalita, která slouží pro navigaci a pohyb uživatele mezi jednotlivými obrazovkami. Obrazovky se skládají z komponent a obsahují implementaci jedné logické části aplikace, např. příhlášení, seznam tréninků atd.

5.1.7 Servicy

Services (Servicy) jsou zodpovědné za komunikaci s REST API. Obsahují logiku, která tuto komunikaci obsluhuje, a obvykle zajišťují asynchronní komunikaci s BE pomocí HTTP protokolu. Asynchronní komunikace se v JavaScriptu a i TypeScriptu obvykle dosahuje pomocí tzv. promises (slibů). Třída Promise slouží k práci s asynchronními operacemi. Volání BE pomocí HTTP protokolu z FE je typická asynchronní operace, protože nelze předem určit, za jak dlouho na volání odpoví, a FE aplikace musí stále s uživatelem interagovat i během čekání na odpověď od BE [45]. Pro implementaci REST API metod v rámci FE byla použita knihovna Axios [9]. Axios je klintská knihovna implementující volání API pomocí výše popsaných promises.

5.2 Backend

Tato sekce se věnuje technickému popisu backendu, neboli serverové části aplikace (BE). V této sekci je vysvětlen postup implementace a jsou rozebrány klíčové komponenty této části aplikace. Protože se jedná o téma, které lze velmi podrobně popsat, je zde uveden pouze popis nejdůležitějších částí aplikace.

5.2.1 Založení a konfigurace projektu

Pro založení projektu byl použit Spring Initializr. Spring Initializr je API, které slouží pro generování projektů běžících pod JVM (Java Virtual Machine) projektů [55]. Konkrétní technologie, které byly zvoleny pro implementaci, jsou popsané v sekci 4.3. Pro vývoj byl vybrán programovací jazyk Java. Pro dosažení co nejjednodušší konfigurace projektu byl kladen důraz na zachování stejné verzi Javy pro vývoj FE i BE. Třída, která obsahuje metodu main, tedy vstupní bod aplikace, který je zodpovědný za spuštění a běh aplikace, se nazývá ApiApplication a nachází se v balíčku *api/src/main/java/com/cvut*. Konfigurace aplikace se nachází v souboru application.yaml a application-local.yaml v adresáři *api/src/main/resources*.

5.2.2 Adresářová struktura

Pro adresářovou strukturu byl zvolen přístup zvaný separate by feature (dělení podle funkce). V této adresářové struktuře jsou jednotlivé soubory tříděny podle funkce v aplikaci, například soubory sloužící k zabezpečení aplikace budou v samostatném adresáři, soubory sloužící pro práci s entitou Trénink budou v samostatném adresáři atd. Výhodou tohoto přístupu je snažší orientace v adresáři, protože se všechny soubory vztahující se k dané funkcionalitě nacházejí v jednom adresáři [51]. Alternativně bylo možné použít separate by layer přístup, který je popsán v sekci 5.1.

V následujícím seznamu je ukázána obecná struktura tohoto přístupu k řazení adresářů. V projektu je tato metodika použitá konkrétně v balíčku */api/src/main/java/com/cvut*. Ukázka toho to balíčku a dělení adresářů podle funkce je následující:

/api/src/main/java/com/cvut
- `domain` ..... Adresáře se soubory obsahujícími perzistenční vrstvu.
  - `model` ..... Databázové entity, podle kterých se vytváří tabulky.
  - `repository` ..... Repozitáře pro operace nad entitami.
- `service` ..... Servisy obsahující business vrstvu aplikace.
- `web` ..... Adresáře se soubory obsahujícími prezentační vrstvu.
  - `controller` .Třídy vystavující endpointy, které může klient volat a tím interagovat s aplikací.
  - `model` .... Modely určující datové rozhraní, ve kterém je aplikace schopná příjmat dotazy.

5.2.3 Zabezpečení aplikace

Aplikace používá zabezpečení implementované v modulu Spring Security. Konfigurace zabezpečení aplikace se nachází ve třídě SecurityConfig. Pomocí anotace *@Configuration* Spring Framework dostane informaci, že zde bude deklarovaná Beana. Jedná se o použití návrhového vzoru IoC popsaného dříve v této kapitole. Nacházejí se zde 2 beany. První beana passwordEncoder() je zodpovědná za vytvoření instance třídy BCryptPasswordEncoder sloužící pro zahashování uživatelského hesla. Používá k tomu stejnojmenný hashovací algoritmus BCrypt [64]. Druhá beana pojmenovaná securityFilterChain je implmenetací patternu Chain of Responsibility (řetězec zodpovědnosti). Nejdůležitější článek tohoto řetězce je jwtAuthorizationFilter, což je třída, ve které je implementována validace JWT (JSON Web Token, dále jen token), kterým se klient musí prokázat, pokud se chce autorizovat. Aplikace je navržena tak, aby byly bez použití tokenu dostupné pouze endpointy, které slouží pro registraci, přihlášení uživatele nebo kontrolu stavu (healthcheck). Při posílání dotazu na jakýkoliv jiný endpoint aplikace vyžaduje validní token a pokud není v requestu přítomen, tak je tento request zamítnut filtrem již během procesu autorizace. JWT token klient získá tak, že pošle request na endpoint /login; pokud uživatelské jméno a heslo souhlasí se záznamy uloženými v databázi (v případě hesla zašifrovanými), tak odpověd bude obsahovat token, pomocí kterého se bude moci uživatel po omezenou dobu platnosti tokenu prokázat při svých dalších dotazech do aplikace.

5.2.4 REST Kontrolery

REST Kontrolery, neboli třídy označené anotací *@RestController*, se nachází v adresáři controllers. *@RestController* je kombinací anotací *@Controller* a *@ResponseBody*. Je vytvořená pro budování RESTful webových služeb. Její účel je obstarat zpracování požadavku (request) směrem ke kontroleru a serializovat odpověď (response), kterou kontroler klientovi vrací. V jazyku Java je implementována pro zjednodušení vývoje REST API. *@RequestMapping* určuje prefixy url, na které se mají vystavené endpointy v rámci kontroleru mapovat [11]. Jednotlivé endpointy jsou poté mapované na metody v závislosti na struktuře url, proměnných a HTTP metodách, ke kterým se vážou. Typické anotace použité pro označení endpointů v rámci kontroleru se nazývají *@GetMapping*, *@PostMapping* atd. Namapování proměnné z HTTP requestu na proměnnou v jazyce Java, kterou poté lze používat při volání enpointu, bude dosaženo použitím anotace *@PathVariable*. Alternativně, je-li nutné docílit namapování těla HTTP requestu na objekt v Javě, použije se pro to anotace *@RequestBody*. K vytvoření response je vhodné používat objekt třídy ResponseEntity, který je svou strukturou i daty připraven k reprezentaci response v rámci HTTP protokolu. Kontroler třídy patří do prezentační vrstvy Spring Boot aplikace.

5.2.5 Servicy

Servicy (Service) patří do business vrstvy v rámci Spring Boot aplikací. Je v nich uložena logika pro transformaci dat a operace s daty. V Spring Bootu je nezbytné Service třídy anotovat jako *@Service*. Tato anotace ale nerozšiřuje třídu o žádnou specifickou funkci. Slouží tedy pouze pro označení třídy v rámci projektu [1].

5.2.6 Repozitáře

Rozhraní anotované pomocí *@Repository* se označuje jako „repozitáře“. Tato rozhraní jsou součástí perzistenční vrstvy aplikace a slouží pro přístup k entitám uloženým v databázi a k provádění operací s těmito entitami. *@Repository* anotace je stejně jako v případě Service třídy nezbytná, protože díky jejímu použití Spring Boot dostane informaci, že se jedná o repozitář pro přístup do databázové vrstvy. Obvykle se repozitář dědí od jiného existujícího generického rozhraní, které je v rámci Spring Bootu předpřipraveno plnit

tuto funkci v aplikaci. V této aplikaci bylo za tímto účelem použito rozhraní JpaRepository. JpaRepository je součástí modulu spring-boot-starter-data-jpa a obsahuje všechny CRUD (Create, Read, Update, Delete) operace pro práci s databázovou entitou. Zároveň do repozitáře lze přidávat pomocí anotace @*Query* i vlastní funkce, které mohou provádět specifické operace nad daty uloženými v databázi, které nejsou implementovány v rámci JpaRepository. [54]

5.2.7 Entity

Třídy anotované anotací @*Entity* jsou součástí perzistenční vrstvy a slouží pro vytváření databázových tabulek, vazeb mezi nimi, sekvencí a dalších funkcionalit používaných při tvorbě databáze. Pro zobecnění struktury těchto entit je v aplikaci vytvořena abstraktní třída AbstractEntity, která je použita jako základ všech ostatních entit. AbstractEntity deklaruje primární klíč ve formátu UUID v4, což je formát, který vznikl pro identifikaci objektu v rámci internetu [60]. Zajímavé jsou anotace @*Entity* a @*Table*. Anotace @*Entity* slouží pro identifikaci třídy v rámci psaní databázových dotazů (@*Query*). Anotace @*Table* určuje jméno tabulky v databázi. Anotace @*Id* označuje primární klíč a anotací @*Generator* je definováno, jak se tento klíč má vytvářet. Nakonec vazby mezi tabulkami jsou definovány pomocí anotací @*ManyToOne*, @*OneToMany*, @*OneToOne*, které odpovídají stejnojmenným vazbám standartně používaným v objektově relačních databázích. [48]

5.2.8 Záznamy

Records (Záznamy) se v Javě označují klíčovým slovem „record“. Jedná se o reprezentaci imutabilní třídy, která obsahuje pouze metody equals(), hashCode() a toString(), a nelze v nich deklarovat žádné další metody. Motivací pro zavedení tohoto datového typu je fakt, že tradiční třídy deklarované za pomocí klíčového slova „class“ obsahují velké množství tzv. „boilerplate code“ (kódu, který se opakuje s žádnou nebo malou variací). Je vhodné je použít při reprezentaci jednoduchých datových struktur, jako jsou například tělo požadavku v rámci REST API, které nepotřebují implementovat specifickou logiku, ale slouží pouze pro transfer dat. Tento koncept byl přidán až v Javě 14, dříve se pro stejný účel používaly tradiční třídy deklarované klíčovým slovem „class“. [32]

5.2.9 Validátory

Součástí Spring Frameworku je i modul sloužící pro validaci: spring-boot-starter-validation. Pokud bude tento modul přidán do aplikace, lze v aplikaci poté začít používat tzv. validátory. Validátory jsou komponenty zodpovědné za validování objektů, se kterými jsou spojeny pomocí anotací. Příklad validátoru je @*NotNull*, který kontroluje, zda hodnota proměnné není null. Validátory ale nefungují automaticky, je nutné poskytnout Spring Bootu informaci o tom, že má validátor použít. K tomu slouží anotace @*Validated*, nebo @*Valid* [61]. @*Validated* se používá při validaci vstupních proměnných metod, zatímco @*Valid* je určeno pro použití u vstupních proměných u bean.

5.2.10 Zpracování výjimek

Exception handling (Zpracování výjimek) je proces, jehož cílem je zachytit a zpracovat výjimku, která nastala za běhu aplikace. Typickým případem z uživatelské praxe je dotaz se špatně vyplněným uživatelským jménem a heslem. Aplikace poté ukončí zpracovávání HTTP requestu s výjimkou InvalidPasswordOrUserException a potřebuje tuto informaci předat klientovi. K tomu slouží třída GlobalExceptionHandler. Anotace @*ControllerAdvice* určí, že třída bude sloužit k zachytávání výjmek, které mohou nastat za běhu aplikace. Pro každý typ výjimky, který může nastat, je potřeba zadefinovat si metodu s anotací @*ExceptionHandler* a specifikací typu výjimky, který má daná metoda zpracovávat. V těle metody je výjimka obvykle přeformulována do uživatelsky přívětivého formátu pomocí třídy ResponseEntity a i se správným návratovým kódem vrácena klientovi. [18]

5.3 Databáze a migrování databáze

Pro ukládání dat jsem si zvolil databázi PostgreSQL. Důvody, proč jsem si vybral právě PostgreSQL, jsou podrobně popsány v sekci 4.4. Pro migraci a verzování databáze jsem zvolil migrační nástroj Flyway [26]. Flyway je nástroj pro správu verzí databází, který umožňuje jednoduše provádět změny struktury databáze prostřednictvím tzv. migračních skriptů. Tyto skripty jsou uloženy v adresáři */resources/db/migration* a Flyway je automaticky aplikuje na databázi při spuštění aplikace nebo při nasazení nové verze. Díky použití této technologie je programátor při vývoji schopný aktualizovat strukturu již existující databáze.

5.4 Prostředí pro vývoj aplikace

Pro lokalní vývoj BE a editování Java kódu jsem použil vývojové prostředí Intellij IDEA [30] od firmy JetBrains. V tomto prostředí je již integrován plugin, který slouží k obsluze nástroje Gradle. Další plugin, který prostředí poskytuje, se jmenuje „Database tool“ (databázový nástroj). Tento nástroj slouží pro připojení k databázi a vykonávání SQL příkazů. Dále jsem při vývoji FE i BE používal nástroj GitHub Copilot [22], se kterým jsem se radil při implementaci kódu a používal ho pro generování dokumentace k aplikaci. Pro vytvoření Javadoc dokumentace ze zdrojových souborů jsem nakonec použil stejnojmenný nástroj integrovaný ve vývojovém prostředí Intellij IDEA. Dokumentace je v repozitáři umístěna v adresáři */app/javadoc*.

K vývoji FE části aplikace bylo použité vývojové prostředí WebStorm [63]. Pro práci s npm jsem používal Windows příkazovou řádku integrovanou v tomto prostředí. Pro spuštění Android aplikace v počítači byl použit Android emulátor integrovaný ve vývojovém prostředí Android Studio [2].

Pro lokální vývoj a testování databáze byla používána kontejnerizovaná verze databáze, což zajišťuje aplikace Docker [16] a nástroj Docker Compose [17]. Docker je platforma, která umožňuje vytváření, nasazení a spouštění aplikací pomocí kontejnerů. Docker Compose je nástroj, který na základě konfiguračního souboru v jazyce YAML umožňuje konfigurovat a spouštět vícenásobné kontejnery. Tento přístup zajišťuje, že všechny části aplikace, včetně databáze, jsou snadno přenositelné a konzistentní mezi různými prostředími.

5.5 Nastavení repozitáře a verzování aplikace

Pro vývoj aplikace byl založen open-source projekt v Gitlab repozitáři fakulty jehož adresa je uvedena v příloze Git repozitář. Repozitáž je veřejně přístupný a nakonfigurovaný tak, aby se potenciálně i ostatní uživatelé Gitlabu mohli podílet na jeho vývoji. V souboru READ.ME, který je v projektu umístěn na nejvyšší úrovni, se nachází popis projektu, použitých technologií a doporučených prostředí pro vývoj a spuštění aplikace. Soubor gitlab-ci.yml obsahuje konfiguraci pipeline, která slouží pro automatizovaný deploy BE části apliakce na cloud prostředí DigitalOcean [15]. Adresa pro přístup k aplikaci je uvedena v příloze Serverová část a kolekce s připravenými dotazy je umístěna v git repozitáři v adresáři */api/postman*. Pokud se běh pipeline úspěšně nedokončí, tak se deploy nové verze aplikace z gitu do DigitalOcean

neprovede. Konfigurace automatického deploye v pipeline se dělí na 3 stage:

build stage: stáhne image, který obsahuje předkonfigurované javu a gradle, a zkompiluje zdrojový kód k BE.

test stage: stáhne image, který obsahuje předkonfigurované javu a gradle, a spustí unit (jednotkové) testy, které otestují BE.

deploy stage: stáhne image s předkonfigurovaným dockerem a vytvoří z BE docker image, který poté pushne do cloudu.

Verzování aplikace je rozděleno na verzování FE a verzování BE. Informace o verzi FE se nacházejí v souboru */app/package.json*. Informace o verzi BE se nacházejí v souboru */api/src/main/resources/application.yaml*.

Kapitola 6

Testování

Testování softwaru je proces ověřování, zda software splňuje specifikované požadavky a zda funguje správně. Cílem testování je odhalit chyby v softwaru, které vznikly během implementace či nedokonalým návrhem aplikace. Tato kapitola popisuje, jak byla aplikace testována po dokončení implementace.

6.1 Jednotkové testy

Unit (jednotkové) testy slouží pro testování tříd a metod. Obvykle jsou tyto testy implementovány již programátory a jsou zapisovány ve formě programového kódu. Tyto testy jsou obvykle vytvářeny za využití nějakého frameworku [19]. V případě aplikace, kterou jsem implementoval, je těmito testy pokryta funkčnost kontroleru a servisy v rámci balíčku „trénink“. Ostatní třídy v balíčku nebylo potřeba pokrývat testy, protože ve svém těle neimplementují žádné metody, které by přidávaly funkcionalitu, kterou by bylo třeba testovat. Konkrétně se jedná tedy o třídy a rozharní, které zastupují roli data modelu nebo repozitáře. Tyto testy jsou vytvořené s cílem pokrýt validní i nevalidní vstupy, se kterými může být třída provolána. Testy k balíčku „trénink“ jsou umístěny v BE aplikaci, konkrétně v adresáři *api/src/test/java/com/cvut/api/training*. Pro ostatní moduly by bylo vhodné implementovat obdobné testy při budoucím rozšiřování projektu.

Automatické unit testy nad BE jsou spouštěné při každém přenasazení aplikace, jak je uvedeno v sekci 5.5. Funkce v balíčku training jsou podle

vývojového prostředí Intelij IDEA aktuálně ze 86% pokryté test coverage (testovacím pokrytím). Celý projekt je ale pokrytý testy pouze z 22% protože kromě balíčku training, nebyly v rámci BE implemntovány další jednotkové testy.

Jednotkové testy by bylo možné implementovat také v rámci FE. Při práci na projektu byla ale překážkou časová náročnost implementace těchto testů v rámci složitější aplikační logiky vyplývající z interakce uživatele s UI.

6.2 Integrační testy

Cílem integračních testů je ověřit komunikaci mezi jednotlivými komponentami v rámci aplikace. Integrační testy obvykle nepřipravuje programátor, ale testovací team. Při implementaci integračních testů obvykle začíná tester s testováním komunikace mezi dvěma komponentami a postupně přidává další pro složitější testovací scénáře. Integrační testy nemusí sloužit pouze pro ověření komunikace mezi komponentami uvnitř systému, ale například také k ověření komunikace mezi komponentou a operačním systémem, hardwarem, databází atd. Integrační testování může být buď manuální, nebo automatické, jak je popsáno v sekci „Integrační testování“ v článku „Fáze a úrovně provádění testů“ [19].

V rámci testování implementace 5 jsem prováděl manuální integrační testování BE části aplikace a DB pomocí nástroje Postman [44]. Pro testování komunikace BE a FE byl použit webový prohlížeč Google Chrome a jeho vývojářské nástroje v kombinaci s Hermes [25], což je nástroj, který umožňuje vypisovat do konzole, debugovat, sledovat síťový provoz atd. u React Native aplikací spuštěných v emulátoru, stejně jako by se to dělalo u aplikace běžící ve webovém prohlížeči. Pomocí integračních testů bylo detekováno množství chyb, které byly následně opraveny.

6.3 Průběh testování

Jako autor mobilní aplikace jsem provedl řadu testů, abych zajistil, že aplikace splňuje stanovené požadavky. Během testování jsem se zaměřil na různé vlastnosti aplikace včetně funkčnosti, praktické použitelnosti a vzhledu. Kromě

toho jsem dal aplikaci testovat konkrétnímu uživateli, abych získal jeho reálnou zpětnou vazbu a názory.

Výsledky testování naznačují, že z hlediska použitelnosti a uživatelského rozhraní jsou přítomny nedostatky, které komplikují uživateli používání aplikace. V prvním prototypu bylo identifikováno několik oblastí, ve kterých lze UI vylepšit (vytvoření nového tréninku, zaznamenání střelby na terči), aby byl pro uživatele přehlednější. Je však důležité zdůraznit, že vzhledem k omezenému času nebylo možné všechny prvky UI dokonale ladit a optimalizovat, jelikož ladění uživatelského rozhraní je časově náročný proces, který vyžaduje pečlivou analýzu a iterativní úpravy. Cílem této práce bylo především dokončení celkového konceptu a funkčnosti aplikace. Uživatelské rozhraní je samostatnou kapitolou, která bude podrobněji analyzována a vylepšena v další fázi vývoje.

Z funkčního hlediska aplikace splňovala požadavky na zavedení tréninku a realizace aplikace se potkala se sportovcovými očekáváními. Je důležité mít na paměti, že aplikace byla navržena pouze s minimální sadou funkcí. Z tohoto důvodu lze usoudit, že konkurenční aplikace budou stále uživateli více preferované, protože na jejich vývoji bylo stráveno mnohem více času.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat sportovní deník uzpůsobený pro lukostřelecké atlety. V rámci této práce proběhla podrobná analýza lukostřelby, která zahrnovala různé typy luků a příslušenství používaného během střelby.

V další kapitole byla analyzována existující řešení, včetně jejich předností a nedostatků. Na základě této analýzy jsem sestavil seznam požadavků. Požadavky byly rozděleny do několika skupin a byla vybrána minimální množina funkcí pro vytvoření prototypu aplikace. Tato množina požadavků se musela v průběhu práce měnit kvůli podcenění rozsahu celkového úkolu.

V následující kapitole byly vybrány moderní technologie a popsán jejich účel a použití při vytváření prototypu aplikace.

Během implementace jsem se setkal s několika problémy, které byly způsobeny nepřesně definovanými uživatelskými požadavky nebo podceněním náročnosti implementace. Nakonec jsem byl schopen dokončit prototyp, který splnil všechny případy užití, jež jsem si stanovil jako nezbytné pro vytvoření aplikace. Serverová část aplikace, která byla v rámci prototypu vyvinuta, byla úspěšně nasazena na cloud a její klientská část byla nainstalována na chytrý mobilní telefon.

V poslední kapitole bylo rozebráno testování prototypu. Zaznamenal jsem zpětnou uživatelskou vazbu a označil oblasti, ve kterých by bylo vhodné aplikaci během budoucího vývoje zlepšit. I když byly zaznamenány výhrady

uživatelů směrem k uživatelskému rozhraní, aplikace pokryla požadovanou funkcionalitu.

Do budoucna je možné aplikaci dále rozšiřovat. V úvahu přichází sekce pro komunikaci trenéra a atleta, časomíra, číslování šípů a mnoho dalších funkcí uvedených v příloze. Pro budoucí rozvoj aplikace jsem v rámci této práce stanovil požadavky a zveřejnil zdrojové kódy ve formě open source na GitLab repozitáři fakulty, aby bylo možné na práci dále navázat.

Literatura

[1] *@Component vs @Repository and @Service in Spring*. URL: https://www.baeldung.com/spring-component-repository-service (cit. 10. 05. 2024).

[2] *Android Studio*. URL: https://developer.android.com/studio (cit. 19. 05. 2024).

[3] *Apache Groovy*. URL: https://groovy-lang.org/ (cit. 18. 05. 2024).

[4] Marcel van Apeldoorn. *ARTEMIS User Manual*. vApeldoorn, 2017. URL: http://artemis.vapeldoorn.net/downloads/ArtemisUserManual.pdf (cit. 03. 04. 2024).

[7] *ArtemisLite*. URL: https://play.google.com/store/apps/details?id=com.vapeldoorn.artemislite&hl=en_US (cit. 23. 05. 2024).

[8] *AsyncStorage*. URL: https://reactnative.dev/docs/asyncstorage (cit. 11. 05. 2024).

[9] *Axios*. URL: https://www.npmjs.com/package/axios (cit. 12. 05. 2024).

[10] Lucie Boušová. "Vznik, vývoj a současná podoba lukostřelby". Bakalářská práce. Západočeská univerzita v Plzni, Pedagogická fakulta, 2015. URL: https://dspace5.zcu.cz/handle/11025/19975 (cit. 28. 02. 2024).

[11] *Building a RESTful Web Service*. URL: https://spring.io/guides/gs/rest-service (cit. 10. 05. 2024).

[12] *Compound Bows: Term Definition Essay (Critical Writing)*. URL: https://ivypanda.com/essays/compound-bows-term-definition/ (cit. 12. 03. 2024).

[14] *Difference between interfaces and classes in TypeScript*. URL: https://www.geeksforgeeks.org/difference-between-interfaces-and-classes-in-typescript/ (cit. 12. 05. 2024).

[15] *DigitalOcean*. URL: https://www.digitalocean.com/ (cit. 17. 05. 2024).

[16] *Docker*. URL: https://www.docker.com/ (cit. 18. 05. 2024).

[17] *Docker Compose*. URL: https://docs.docker.com/compose/ (cit. 18. 05. 2024).

[18] *Exception Handling — Spring Boot REST API*. URL: https://medium.com/thefreshwrites/exception-handling-spring-boot-rest-api-c2656b575fee (cit. 10. 05. 2024).

[19] *Fáze a úrovně provádění testů*. URL: http://testovanisoftwaru.cz/tag/integracni-testovani/ (cit. 21. 05. 2024).

[20] *Figma*. URL: https://www.figma.com/ (cit. 18. 05. 2024).

[21] *Function Components vs Class Components in React – With Examples*. URL: https://www.freecodecamp.org/news/function-component-vs-class-component-in-react/ (cit. 11. 05. 2024).

[22] *GitHub Copilot*. URL: https://github.com/features/copilot (cit. 18. 05. 2024).

[23] *Gradle*. URL: https://gradle.org/ (cit. 18. 05. 2024).

[24] *Gradle vs. Maven: Performance, Compatibility, Builds and More*. URL: https://stackify.com/gradle-vs-maven/ (cit. 17. 04. 2024).

[25] *Hermes*. URL: https://github.com/facebook/hermes/blob/main/README.md (cit. 21. 05. 2024).

[26] *Homepage - Flyway*. URL: https://flywaydb.org/ (cit. 10. 05. 2024).

[27] Marie Horáčková. "Model celoročního tréninkového cyklu vrcholového lukostřelce". Bakalářská práce. Západočeská univerzita v Plzni, Pedagogická fakulta, 2022. URL: https://dspace5.zcu.cz/bitstream/11025/50102/1/Bakalarska%20prace_finalni%20verze.pdf (cit. 13. 03. 2024).

[28] *Icons - Material Design 3*. URL: https://m3.material.io/styles/icons/overview (cit. 18. 05. 2024).

[29] *Information about Artemis(Lite)*. URL: https://www.facebook.com/ArtemisLite/about_details (cit. 03. 04. 2024).

[30] *IntelliJ IDEA*. URL: https://www.jetbrains.com/idea/ (cit. 18. 05. 2024).

[31] *Jakarta EE*. URL: https://jakarta.ee/ (cit. 18. 05. 2024).

[32] *Java 14 Record Keyword*. URL: https://www.baeldung.com/java-record-keyword (cit. 10. 05. 2024).

[33] *JPA*. URL: https://docs.spring.io/spring-framework/reference/data-access/orm/jpa.html (cit. 19. 04. 2024).

[34] *JS Projects Utilizing TypeScript*. URL: https://www.typescriptlang.org/docs/handbook/intro-to-js-ts.html (cit. 17. 04. 2024).

[35] Michal LEHNERT et al. *Kondiční trénink*. Univerzita Palackého v Olomouci, 2014. ISBN: 978-80-244-4369-0. URL: https://publi.cz/books/149/Cover.html (cit. 12. 01. 2024).

[36] *Material Design*. URL: https://m3.material.io/ (cit. 08. 05. 2024).

[37] *Material Design 3*. URL: https://m3.material.io/ (cit. 18. 05. 2024).

[38] Radek Micka. "Metody sběru požadavků na informační systém". Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, 2012. URL: https://insis.vse.cz/zp/31654 (cit. 10. 04. 2024).

[39] *My Workout Plan - Gym Tracker*. URL: https://play.google.com/store/apps/details?id=com.myworkoutplan.myworkoutplan&hl=en_US (cit. 02. 04. 2024).

[40] *MyTargets Archery*. URL: https://play.google.com/store/apps/details?id=de.dreier.mytargets&hl=en_US (cit. 21. 05. 2024).

[41] *Node Package Manager*. URL: https://www.npmjs.com/ (cit. 18. 05. 2024).

[43] *PostgreSQL*. URL: https://www.postgresql.org/ (cit. 18. 05. 2024).

[44] *Postman*. URL: https://www.postman.com/ (cit. 21. 05. 2024).

[45] *Promises in JavaScript and Typescript*. URL: https://dev.to/bcostaaa01/promises-in-javascript-and-typescript-5eh9 (cit. 12. 05. 2024).

[46] *React Native*. URL: https://reactnative.dev/ (cit. 17. 05. 2024).

[48] *Relationships in SQL*. URL: https://www.geeksforgeeks.org/relationships-in-sql-one-to-one-one-to-many-many-to-many/ (cit. 10. 05. 2024).

[49] *Setting up the development environment*. URL: https://reactnative.dev/docs/environment-setup (cit. 11. 05. 2024).

[50] *Spring Boot*. URL: https://spring.io/projects/spring-boot (cit. 17. 04. 2024).

[51] *Spring Boot – Code Structure*. URL: https://www.geeksforgeeks.org/spring-boot-code-structure/ (cit. 30. 04. 2024).

[52] *Spring Boot Architecture*. URL: https://www.javatpoint.com/spring-boot-architecture (cit. 17. 04. 2024).

[54] *Spring Boot JpaRepository with Example*. URL: https://www.geeksforgeeks.org/spring-boot-jparepository-with-example (cit. 10. 05. 2024).

[55] *Spring Initializr*. URL: https://start.spring.io/ (cit. 30. 04. 2024).

[56] *Statcounter GlobalStats*. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (cit. 17. 05. 2024).

[58] *Typescript*. URL: https://www.typescriptlang.org/ (cit. 17. 05. 2024).

[59] *Understanding the React Native Project Structure: Best Practices and Organization Tips*. URL: https://medium.com/@swift3.0devlopment/understanding-the-react-native-project-structure-best-practices-and-organization-ips-ed335fded597-ed335fded597 (cit. 11. 05. 2024).

[60] *UUID (Universal Unique Identifier)*. URL: https://www.techtarget.com/searchapparchitecture/definition/UUID-Universal-Unique-Identifier (cit. 10. 05. 2024).

[61] *Validation in Spring Boot*. URL: https://www.baeldung.com/spring-boot-bean-validation (cit. 10. 05. 2024).

[62] Pavla Vycudilíková. "Výživa sportovců". Diplomová práce. Masarykova univerzita, Přírodovědecká fakulta, 2008. URL: https://is.muni.cz/th/106308/prif_m/?zoomy_is=1#panelbiblatex (cit. 12. 01. 2024).

[63] *Webstorm*. URL: https://www.jetbrains.com/webstorm/ (cit. 18. 05. 2024).

[64] *What is bcrypt and how does it work?* URL: https://nordvpn.com/blog/what-is-bcrypt/ (cit. 10. 05. 2024).

[65] *What is Figma?* URL: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma (cit. 02. 04. 2024).

[66] Karl Eugene WIEGERS. *Požadavky na software*. Vyd. 1. Brno: Computer Press, 2008. ISBN: 978-80-251-1877-1. (Cit. 10. 04. 2024).

[67] *World Archery Federation*. URL: https://web.archive.org/web/20151006005059/http://archery.org/Disciplines (cit. 07. 03. 2024).

[68] *WorldArchery rule book*. URL: https://www.worldarchery.sport/rulebook/article/793 (cit. 02. 04. 2024).

[69] *Yarn*. URL: https://yarnpkg.com/ (cit. 18. 05. 2024).

[70] Pavel ZAHRADNÍK David a KORVAS. *Základy sportovního tréninku*. Brno: Masarykova Univerzita, 2017. ISBN: 978-80-210-5889-7. URL: https://publi.cz/books/51/index.html?secured=false#cover (cit. 27. 03. 2024).

[71] Tomáš Záruba. "Situační analýza výroby součástek pro Průmysl 4.0 a Smart Cities". Diplomová práce. Fakulta elektrotechnická katedra ekonomie, manažerství a humanitních věd, 2018. URL: http://hdl.handle.net/10467/73994 (cit. 16. 04. 2024).

Zdroje k obrázkům

[5] *APOLLO kladkový luk*. URL: https://www.kuse.cz/kuse/eshop/2-1-Luky/0/5/581-LUK-REFLEXNI-APOLLO (cit. 21. 05. 2024).

[6] *APOLLO reflexní luk*. URL: https://www.kuse.cz/kuse/eshop/2-1-Luky/0/5/581-LUK-REFLEXNI-APOLLO (cit. 21. 05. 2024).

[13] *Compound force-draw curve*. URL: https://www.roystonarchery.org/beginners-2/equipment/bow-physics/ (cit. 21. 05. 2024).

[47] *Recurve Force-Draw Curve*. URL: https://www.roystonarchery.org/beginners-2/equipment/bow-physics/ (cit. 21. 05. 2024).

[53] *Spring Boot Architecture*. URL: https://www.javatpoint.com/spring-boot-architecture (cit. 21. 05. 2024).

[57] *Three-tier architecture*. URL: https://hyperskill.org/learn/step/25083 (cit. 21. 05. 2024).

[70] Pavel ZAHRADNÍK David a KORVAS. *Základy sportovního tréninku*. Brno: Masarykova Univerzita, 2017. ISBN: 978-80-210-5889-7. URL: https://publi.cz/books/51/index.html?secured=false#cover (cit. 27. 03. 2024).

Příloha A

Seznam odkazů

- **Návrh uživatelského rozhraní:** `https://www.figma.com/design/2XtWGLg0692nZxaP6ahH0i/Mobile-desing?node-id=0-1`
- **Git repozitář:** `https://gitlab.fel.cvut.cz/lokajva1/bow-diary`
- **Adresy nasazené aplikace:**
  - Serverová část: `https://orca-app-jx2d2.ondigitalocean.app/api`

Příloha B

Uživatelské požadavky

ID	Název požadavku	Popis požadavku	Aktéři
UCHP1	Přihlášení uživatele	Návštěvníkovi se zobrazí okno s formulářem pro přihlášení do aplikace. Aplikace je po vyplnění údajů schopná ověřit, zda je daný uživatel registrovaný a zda jméno a heslo odpovídají jeho záznamu uloženemému v databázi.	NA
UCHP2	Zobrazení seznamu tréninků	Uživateli se zobrazí seznam lukostřeleckých tréninků spojených s jeho účtem. U každého tréninku se mu zobrazí ikona, která identifikuje trénink jako lukostřelecký, název tréninku, datum, počet započatých kol a skóre, kterého bylo během tréninku dosaženo.	AT
UCHP3	Zobrazení seznamu luků	Uživateli se zobrazí seznam luků, které má v aplikaci spojené s účtem. U každého luku je fotografie luku (nebo ikona, pokud luk fotografii nemá), název luku, typ luku, síla nátahu a ikona pro editaci luku.	AT
UCHP4	Vytvoření nového tréninku	Uživatel po zobrazení obrazovky „Vytvoření nového tréninku“ musí, vyplnit pole název, zvolit datum (UCMP5), přiřadit luk k tréninku (UCMP8), volný trénink (UCMP6), číslování šípů (UCMP7), počet šípů, počasí (UCMP9) a použít časomíry (UCMP4). Poté co budou všechna povinná pole vyplněna, tak uživatel ještě musí přidat tréninkové kolo (UCHP7) a poté kliknutím na tlačítko „done“ vytvoří trénink.	AT
UCHP6	Přiřazení luku k tréninku	Poté co uživatel během přidání nového tréninku klikne na pole luk, zobrazí mu aplikace seznam luků (UCHP3). Po kliknutí na luk v seznamu bude přesměrován zpět a bude mít přiřazený luk k tréninku.	AT
UCHP7	Přidání tréninkového kola	Uživatel zvolí terč, na který bude střílet, počet sad, počet šípů v každé sadě a poznámku. Poté kliknutím na tlačítko v pravém horním rohu obrazovky potvrdí svůj výběr.	AT
UCHP8	Přiřazení tréninkového kola k tréninku	Poté co uživatel během přidání nového tréninku (UCHP4) klikne na tlačítko „+“ v pravém dolním rohu obrazovky, zobrazí se mu obrazovka pro přidání nového tréninkového kola. Po přidání nového tréninkového kola (UCHP7) je přesměrován zpět na obrazovku přidání nového tréninku (UCHP4).	AT

Tabulka B.1: Tabulka požadavků s vysokou prioritou 1/2

UCHP9	Zobrazení seznamu tréninkových kol	Po vytvoření nového tréninku (UCHP4) nebo kliknutí na trénink na obrazovce Seznam tréninků (UCHP2) je uživatel přesměrován na seznam tréninkových kol.	AT
UCHP10	Zobrazení seznamu sad v rámci tréninkového kola	Po výběru ze seznamu kol (UCHP9) se uživateli zobrazí seznam sad. Po výběru v seznamu sad je uživatel přesměrován na obrazovku pro zaznamenání střelby na terč (UCHP11).	AT
UCHP11	Zaznamenání střelby na terči	Po zahájení střelby se uživateli zobrazí terč, na kterém může pohybovat kurzorem umístěným uprostřed obrazovky a zaznamenat tak pozici šípu. V horní části obrazovky je tabulka, kde se uživateli zobrazuje skóre (UCHP12). Ve spodní části obrazovky jsou umístěna tlačítka „přechozí“ a „použít“. Po kliknutí na tlačítko „předchozí“ je poslední uživatelem zaznamenaný šíp smazán. Tlačítkem „použít“ uživatel zaznamená aktuální pozici kurzoru na terči. Po zanesení posledního šípů je uživatel přesměrován zpět na záložku se sadami šípů.	AT
UCHP12	Zobrazení ukazatele skóre	V horní části obrazovky se uživateli bude zobrazovat komponenta s aktuálním skóre. Součástí komponenty je číslo kola, ve kterém se uživatel nachází, jednotlivé bodové ohodnocení, kterého svými šípy v tomto kole dosáhl, skóre dosažené v kole a skóre dosažené celkově.	AT
UCHP14	Zobrazení statistik střelby	Na záložce se seznamem tréninkových kol (UCHP9) se na aplikační liště nachází ikona pro zobrazení statistik střelby. Po kliknutí na tuto ikonu je uživatel přesměrován na samostatnou obrazovku, kde se mu zobrazí graf distibuce šípů na terči (UCHP15) a graf dosaženého skóre v průběhu střelby (UCHP16).	AT
UCHP15	Graf distibuce šípů na terči	Graf s 11 sloupci 0 – 10, ve kterém každý sloupec odpovídá právě jedné zásahové ploše v rámci terče. Graf obsahuje čísla odpovídající zásahům jednotlivých ploch. Čísla jsou zobrazena v sloupcích, které svojí výškou odpovídají počtu zásahů dané plochy terče.	AT
UCHP16	Graf dosaženého skóre v průběhu střelby	Graf, kde je na ose Y skóre dosažené v rámci sady a na ose X jsou čísla jednotlivých kol a sad určující pořadí sady.	AT

Tabulka B.2: Tabulka požadavků s vysokou prioritou 2/2

ID	Název požadavku	Popis požadavku	Aktéři
UCMP1	Registrace uživatelete	Návštěvníkovi se zobrazí na úvodním okně pro přihlášení tlačítko, které ho přesměruje na okno s formulářem pro registraci. Po vyplnění formuláře aplikace ověří, zda je možné uživatele zaregistrovat. V případě, že to možné není, zobrazí aplikace uživateli zprávu s chybou. Pokud je registrace úspěšná, do databáze se uloží uživatelovo přihlašovací jméno a hash jeho hesla.	NA
UCMP2	Přístup k seznamu obecných tréninků	Uživateli se zobrazí seznam obecných tréninků, které jsou propojené s jeho účtem.	AT
UCMP3	Zobrazení seznamu terčů	Uživatel si je schopný zobrazit seznam terčů.	AT
UCMP4	Časomíra	Po zahájení tréninku se spustí časomíra, která uživateli určí dobu, po kterou se nejprve připraví na střelbu (20 s), poté dobu, po kterou může střílet (140 s), a upozorní ho 30 s před koncem časového limitu.	AT
UCMP5	Implementace kalendáře	Po kliknutí na ikonku kalendáře se uživateli zobrazí kalendář, ve kterém bude moci zvolit datum a poté i čas. Tyto parametry se následně uloží ve formuláři, ve kterém je pole s kalendářem implementováno.	AT
UCMP6	Režim „volný trénink“	Režim „volný trénink“ se zapíná kliknutím na checkbox během vytváření nového tréninku. Pokud je „volný trénink“ aktivní, uživateli se nepočítají kola ani sady v rámci jednotlivých kol. Celý trénink se tedy zaznamená do jednoho kola a jedné sady.	AT
UCMP7	Číslování šípů	„Číslování šípů“ se zapíná kliknutím na checkbox během vytváření nového tréninku. Pokud je tato funkce zapnutá, uživateli se po zanešení pozice šípu na terči zobrazí dialog, ve kterém má na výběr z čísel jednotlivých šípů (1, 2..n). Počet šípů, které jsou při střelbě použity, nastavuje uživatel během vytváření trénininku v poli „Number of arrows“ (počet šípů).	AT
UCMP8	Přidání nového luku	Uživatel je schopný vytvořit entitu Luk, vyplnit název luku, načíst z telefonu fotografii luku a přiřadit ji k luku, zvolit typ luku, vyplnit sílu v librách a poznámku.	AT

Tabulka B.3: Tabulka požadavků se střední prioritou 1/2

ID	Název požadavku	Popis požadavku	Aktéři
UCMP9	Evidence počasí v rámci entity Trénink	Uživatel je schopný během přidání nového tréninku (UCHP4) editovat údaje o počasí. Počasí je reprezentováno jako množina atributů (slunce, teplota, rychlost větru a směr větru) v rámci entity Trénink. Po kliknutí na komponentu s počasím na obrazovce „Trénink“ je uživatel přesměrován na obrazovku „Počasí“, kde je mu umožněno editovat dříve zmíněné atributy. Po kliknutí na ikonku „done“ (hotovo) v pravém horním rohu se ukončí editace počasí a změny se zobrazí na obrazovce přidání nového tréninku	AT
UCMP10	Editace a smazání entity Trénink	V rámci seznamu tréninků po kliknutí na entitu Trénink může uživatel tuto entitu editovat nebo odstranit ze seznamu.	AT
UCMP11	Editace a smazání entity Tréninkové kolo	V rámci tréninku uživatel po kliknutí na entitu Tréninkové kolo může tuto entitu editovat nebo odstranit ze seznamu.	AT
UCMP12	Editace a smazání entity Sada	V rámci tréninkového kola může uživatel editovat nebo smazat dříve vytvořenou sadu.	AT
UCMP13	Smazání a editace enity Luk	V rámci zobrazení seznamu luků může uživatel editovat nebo smazat dříve vytvořený luk.	AT
UCMP14	Propojení aplikace s uživatelem s rolí „trenér“	Uživatel je schopný propojit svůj účet s uživatelem s jiným uživatelským účtem. Tento účet, se kterým je uživatelův účet propojen, poté vůči němu bude vystupovat v roli „trenér“. Uživatel může být propojen s více trenéry a trenér může být propojen s více uživateli.	AT, TR
UCMP15	Přístup uživatele s rolí „trenér“ k propojeným účtům	Uživatel s rolí „trenér“ je schopný zobrazit si záznamy o trénincích atletů propojených s jeho účtem. Trenér je také schopen plánovat propojeným atletům tréninky skrze standartní rozhraní vytváření tréninku.	TR

Tabulka B.4: Tabulka požadavků se střední prioritou 2/2

ID	Název požadavku	Popis požadavku	Aktéři
UCLP1	Filtrace tréninků	Uživatel je schopný filtrovat tréninky, které se mu zpřístupní podle jeho role. Atlet (typ tréninku, datum), trenér (typ tréninku, datum, jméno atleta).	AT, TR
UCLP2	Přidání, editace, smazání uživatelského terče	Uživatel je schopný přidat si do seznamu terčů nový terč a nastavit pro něj bodování. Uživateli je také umožněno editovat terče, které přidal do aplikace dříve, a případně je smazat.	AT
UCLP3	Přidání fotografie k tréninkové sadě	V rámci sady je uživatel schopen vytvořit novou fotografii anebo vybrat fotografii ze zařízení. Tato fotografie je poté přiřazena k tréninkové sadě.	AT
UCLP4	Alternativní zanášení pozice šípu na terči	Implementace zanášení pozice šípu na terči pomocí kurzoru, který bude uživatel schopný přesouvat po terči a tím zaznamenat pozici šípu. Jde tedy o opačný postup k základnímu zanášení šípu popsanému v (UCHP11).	AT
UCLP5	Přidání poznámky k sadě	Během zanášení pozice šípů v rámci tréninkové sady je uživatel po kliknutí na ikonku „poznámka“ v aplikacnčím menu schopný zanést poznámku o sadě, kterou střílel.	AT
UCLP6	Sdílení tréninku	V rámci obrazovky pro zobrazení tréninkových kol se v aplikačním menu nachází ikonka na sdílení tréninku. Po kliknutí na tuto ikonku bude uživateli zobrazen seznam aplikací kde může sdílet výsledek střelby v textovém formátu.	AT
UCLP7	Export tréninku do PDF	V rámci obrazovky pro zobrazení tréninkových kol, se v aplikačním menu nachází ikonka na sdílení tréninku (UCLP6). Po kliknutí na ikonku zde bude i možnost „exportovat do PDF“, po jejímž zvolení se uživateli data o střelbě přeformátují do PDF, které bude moci uložit v zařízení nebo sdílet.	AT

Tabulka B.5: Tabulka požadavků s nízkou prioritou