UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

DIPLOMOVÁ PRÁCE

2023 Bc. František Šilar

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Webová aplikace pro zpracování sportovních výsledků

Diplomová práce

2023 Bc. František Šilar

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. František Šilar**
Osobní číslo:	**I20218**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Webová aplikace pro zpracování sportovních výsledků**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je vytvořit analýzu, návrh a implementaci webové aplikace pro zpracování sportovních výsledků vybraných soutěžních disciplín. Předpokladem je zaměření se na oblast časomíry, uchování dosažených výsledků a s tím související problematiku.

Důležitou částí diplomové práce je také návrh datové vrstvy aplikace kde je předpokladem použití relační databáze a ORM.

Pro back-end část aplikace bude použit framework Spring Boot a pro front-end část aplikace knihovna React.

Důležité části aplikace budou pokryty jednotkovými a integračními testy.

Rozsah pracovní zprávy: **50-60 normostran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

ARLOW J., NEUSTADT I. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2., vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

TURNQUIST L. G.Learning Spring Boot 2.0 – Second Edition: Simplify the development of lightning fast applications based on microservices and reactive programming. Packt Publishing, 2017, 370 pp. ISBN978-1786463784.

Vedoucí diplomové práce: **doc. Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **8. listopadu 2022**
Termín odevzdání diplomové práce: **19. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2022

Prohlašuji:

Práci s názvem Webová aplikace pro zpracování sportovních výsledků jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 24. 08. 2023

Bc. František Šilar v. r.

PODĚKOVÁNÍ

Rád bych poděkoval mému vedoucímu doc. Ing. Michaelu Bažantovi, Ph.D. za odborné vedení, cenné rady a pomoc při psaní této diplomové práce.

ANOTACE

Tato diplomová práce se zabývá problematikou zpracování a vyhodnocování sportovních výsledků disciplín požárního sportu. Popisuje analýzu, návrh a implementaci webové aplikace, která jejím uživatelům usnadňuje přístup k informacím o soutěžích v požárním sportu. Důležitou částí této aplikace je administrační část pro rozhodčí, časoměřiče, soutěžící a pořadatele závodu. Administrační část umožňuje soutěžícím podávat online rezervace startovních pořadí, pořadatelům spravovat ročníky závodů a pracovníky soutěže, časoměřičům v povolené době ukládat výsledky soutěže do systému a měnit je a také rozhodčím v povolené době penalizovat naměřené časy soutěžících. Systém naměřená data automaticky zpracovává a umožňuje jejich online zobrazení. Výsledkem praktické části této práce je webová aplikace, jejíž backendová část je napsaná v programovacím jazyku Java s využitím frameworku Spring Boot, pro frontend je pak použita knihovna React. Datová vrstva webové aplikace využívá databázi Microsoft SQL Server Express a objektově-relační mapování.

KLÍČOVÁ SLOVA

požární sport, zpracování sportovních výsledků, vyhodnocování sportovních výsledků, vývoj aplikací, webová aplikace, relační databáze, Java, Spring Boot, React, Microsoft SQL Server, ORM

TITLE

Web application for processing sports results

ANNOTATION

This diploma thesis deals with the issue of processing and evaluating the sports results of fire sports disciplines. Describes the analysis, design, and implementation of a web application that facilitates its users' access to information about fire sports competitions. An important part of this application is the administration part for judges, timekeepers, competitors and race organizers. The administrative part allows competitors to submit online reservations for starting positions, organizers to manage race years and competition staff, timekeepers to save competition results in the system and change them within the permitted time, and judges to penalize competitors' measured times within the permitted time. The system automatically processes the measured data and enables their online display. The result of the practical part of this work is a web application, the backend part of which is written in the Java programming language using the Spring Boot framework, and the React library is used for the frontend. The data layer of the web application uses a Microsoft SQL Server Express database and object-relational mapping.

KEYWORDS

fire sport, sports results processing, sports results evaluation, application development, web application, relational database, Java, Spring Boot, React, Microsoft SQL Server, ORM

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

IS	Informační systém
SW	Software
ERP	Podnikový informační systém
UP	Unified Process
PS	Požární sport
SH ČMS	Sdružení hasičů Čech, Moravy a Slezska
ISSV	Informační systém pro správu soutěžních výsledků
RZ	Ročník závodu
SK	Soutěžní kategorie
SP	Soutěžní pořadí
MSSQL	Microsoft SQL Server

ÚVOD

Stále větším trendem jsou internetové portály sloužící k pravidelnému informování uživatelů o průběhu a výsledcích nejrůznějších sportovních soutěží. Uživatel tak má možnost online sledovat zápasy ve fotbalu, tenise, hokeji, motosportu i atletice. Většina těchto webů se však zaměřuje především na disciplíny nejvyšších sportovních lig. Sledovat regionální zápasy je pak možné pouze v malé míře, anebo vůbec. Při výběru tématu této diplomové práce, která se bude zabývat zpracováním a reprezentací sportovních výsledků, proto bylo prioritou zaměřit se na disciplínu sportovního odvětví, pro niž doposud existují pouze omezené možnosti online sledování průběhu soutěží. Po průzkumu existujících webových portálů a zvážení mých znalostí o sportovních disciplínách bylo vybráno zaměření se na oblast požárního sportu.

Cílem této diplomové práce tak bude návrh, tvorba a implementace webového informačního systému, který usnadní uživatelům nejen přístup k informacím o výsledcích soutěží požárního sportu, ale bude užitečný i pro samotné soutěžící a pořadatele závodu. Uživatelé budou mít možnost podávání online rezervací na soutěže. Pořadatelům a rozhodčím závodu pak bude umožněn přístup k těmto rezervačním datům, jež jim usnadní tvorbu startovních a výsledkových listin. Modul pro rozhodčí bude dále obsahovat funkcionality sloužící k zadávání a hodnocení výsledných časů závodníků. Tento modul do budoucna umožní propojení s aplikací pro zpracování výsledků z časomíry. Informační systém pak automaticky data zpracuje – vytvoří tabulky výsledků jednotlivých soutěží, rekordů a statistik soutěžních disciplín požárního sportu v rámci všech závodů.

První část této práce seznámí čtenáře s pojmy data, informace a systém. Dále bude popsána obecná definice pojmu informační systém.

Druhá kapitola bude věnována analýze nejpoužívanějších metodik vývoje softwaru. V několika podkapitolách bude vždy v krátkosti shrnuta historie dané metodiky, v další části pak budou probrány výhody a nevýhody při jejím použití.

Třetí kapitola bude zaměřena na průzkum existujících informačních systémů věnujících se problematice zpracovávání výsledků soutěží požárního sportu. Vybrané webové portály budou analyzovány a bude též hodnocena kvalita poskytování informací i s ohledem na pravidla pro požární sport.

Čtvrtá kapitola bude věnována požadavkům na nový na nově vyvíjený IS pro sportovní výsledky. Praktickým výstupem této části budou diagramy funkčních a nefunkčních požadavků a tzv. Use Case diagram případů užití.

Pátá kapitola představí použité technologie. Bude zdůvodněn jejich výběr, následovat bude stručný popis přibližující každou z nich.

Šestá kapitola se bude týkat návrhu a implementace nového informačního systému. Nejprve bude popsán proces hledání vhodných analytických tříd. Další části budou věnovány implementaci back-endu a front-endu aplikace. V kapitole budou popsány i způsoby zabezpečení vyvinuté aplikace.

Závěrečná kapitola bude věnována testování nové aplikace. Budou představeny nejdůležitější jednotkové a integrační testy aplikace a bude vysvětlena jejich funkčnost.

Tato diplomová práce bude určena především pro vývojáře aplikací, neboť bude přibližovat obvyklé postupy při návrhu, analýze a implementaci programů. Výsledný softwarový produkt bude cílený především na příslušníky hasičských sborů, jimž by mohl výrazně zjednodušit pořádání soutěží. V neposlední řadě umožní aktivním účastníkům hasičských soutěží pohodlnější podávání rezervací a zjednoduší přístup k výsledkům požárního sportu nejen pro závodníky, ale i pro nadšence do tohoto sportovního odvětví.

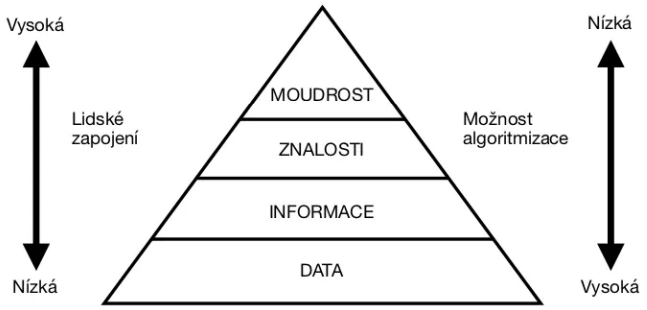
Disciplínám požárního sportu, jejich měření a postupu automatického vyhodnocování výsledků soutěže se věnuje již moje bakalářská práce s názvem „Software pro zpracování výsledků z časomíry pro požární sport“. Jejím výsledným produktem byla desktopová aplikace, která umožňuje načítání výsledků z časomíry pro požární sport do databáze dle předem připravené startovní listiny. Obsluha programu si pak může zobrazit aktuální výsledky, které je možné exportovat do tabulkového souborového formátu *.csv* a *.xlsx*.

1 INFORMACE A INFORMAČNÍ SYSTÉM

Informace provází člověka při aktivitách každodenního života. Lidé se již od útlého věku učí poznávat věci, procesy, osoby a vztahy mezi nimi. Od dávných dob je běžnou lidskou vlastností si důležité poznatky – informace zaznamenávat, ať už pro vlastní účely, např. abychom je nezapomněli, či pro předání dalším osobám. V této kapitole budou popsány základní definice pojmů informace, systém a informační systém.

1.1 Informace

Pro správnou informační gramotnost je nutné rozlišovat tyto čtyři spolu úzce související pojmy – data, informace, znalost a moudrost. Mohlo by se zdát, že znamenají totéž, avšak ve skutečnosti je mezi nimi zásadní rozdíl. Návaznost mezi jednotlivými pojmy odráží data-information-knowledge-wisdom (DIKW) pyramida. [1]



Obrázek 1 – DIKW hierarchická informační pyramida [37]

1.1.1 Data

Chod procesů z celého světa lze vyjádřit daty. Data v informační pyramidě zaujímají nejnižší stupeň. Mohou být v nejrůznějších podobách – čísla, písmena, znaky, obrázky a grafy, zvukové záznamy. Bez dalších vědomostí samotná data nejsou k užitku – poskytují informaci pouze tomu, kdo je dokáže pochopit.

Z pohledu práce s daty jsou rozlišována: [1]

- **strukturovaná data**, která explicitně zobrazují objekty a jejich vlastnosti;
- **nestrukturovaná data**, jež vyjadřují uspořádaný sled datových jednotek, které bez dalších znalostí nejsou pro člověka čitelné.

Typickým příkladem sloužícím k ukládání strukturovaných dat jsou objekty relační databáze. Nestrukturovaná data nelze uspořádat podle předem definovatelné struktury. Jsou to především audio a videosoubory, anebo textové soubory.

1.1.2 Informace

Informace je nehmotným základním stavebním kamenem pro znalosti. Lidská mysl se na základě svých dosavadních znalostí rozhoduje, zda jsou pro ni data nějakým způsobem důležitá. Informacemi se data stanou až v okamžiku, kdy jsou smysluplná, příjemce jim rozumí a jedná-li se z jeho pohledu o údaj nový nebo významný. Informace poskytuje člověku novou možnost ovlivnit jeho jednání v budoucnosti – snižuje míru nejistoty. Stejná data mohou různým osobám poskytovat naprosto odlišnou míru informace. Například zprávě: „Na semaforu svítí zelené světlo.“, budou největší důležitost přikládat řidiči, jenž jsou seznámeni s funkcí světelných signálů a zároveň projíždí místem, v němž se daný semafor nachází. [1]

V roce 1948 Claude Elwood Shannon publikoval svoji teorii informace, v níž se zabývá matematickou definicí množství informace přenášené v jedné datové zprávě. Jednotku množství informace nazval *binary digit*, později ustálenou pod zkráceným názvem *bit*. Shannon dokázal, že míru informace obsažené v datové zprávě oproštěné od její formy lze vyjádřit obyčejnou pravděpodobností. Jeden *bit* pak nesou data informující o jednom ze dvou jevů, jež mohou u pozorovaného objektu nastat, přičemž musí platit, že pravděpodobnost výskytu obou těchto jevů je shodná. V praxi se může jednat např. o odpověď ANO/NE na nějakou otázku. [2]

1.1.3 Znalost

Znalost vyplývá ze správného porozumění informacím a zahrnuje rovněž jejich zařazení do spojitostí se znalostmi týkajícími se stejného či příbuzného tématu. Tento pojem můžeme chápat jako informaci s dodanou hodnotou lidského zapojení. Znalost tedy dává člověku trvalou schopnost správně se rozhodovat na základě aktivního využití nabytých informací. [1]

1.1.4 Moudrost

Na špičce DIKW pyramidy se nalézá moudrost. Je to schopnost odvodit si souvislosti a vztahy mezi objekty na základě vlastního intelektu a zkušeností a použít je k provedení rozumných a smysluplných rozhodnutí. Moudrost nelze předat jiné osobě, ale moudrý člověk může být dobrým zdrojem cenných informací. Zatímco znalost si lze představit jako schopnost vyřešit daný problém, moudrost je snaha zabránit situaci vedoucí ke vzniku stejného problému. [1; 3]

1.2 Systém a informační systém

Systém je ucelená neprázdná množina skládající se z prvků a vazeb mezi těmito prvky, přičemž obě tyto podmnožiny společně definují atributy a chování celku. Matematická definice pojmu systém lze vyjádřit vzorcem $y = f(x)$, kde x je vstupní proces určitého typu a y je výstupní proces stejného typu, přičemž funkce $f(x)$ reprezentuje chování systému. Složité systémy lze dekomponovat na více jednodušších podsystémů a ty dále rozložit na jednotlivé prvky. [4]

Systémy primárně rozlišujeme podle jejich chování. Některé příklady systémů jsou uvedeny níže:

- **Integrovaný dopravní systém** – systém veřejné dopravy sjednocující přepravní a tarifní podmínky více dopravců za účelem většího uspokojení požadavků cestujících.
- **Operační systém** – komplexní systém abstrahující práci s hardwarem za účelem zvýšení bezpečnosti aplikací, snadnějšího vývoje softwaru pro programátory a jednoduššího ovládání zařízení.
- **Telekomunikační systém** – Systém pro přenos informací.
- **Informační systém** – Systém pro přenos, uchovávání a zpracování informací.

1.2.1 Informační systém

Informační systém je struktura založená na modelu reálného světa sloužící ke sběru, přenosu, k udržování, zpracování a poskytování informací. Cílem IS je odstranění překážek v přístupu k informacím. To je realizováno transformací vstupních dat do podoby využívající tato data efektivněji než v počátečním stavu. IS cíleně uspořádává vztahy mezi datovými zdroji informací, osobami používajících tento systém a transformačními procesy. [5]

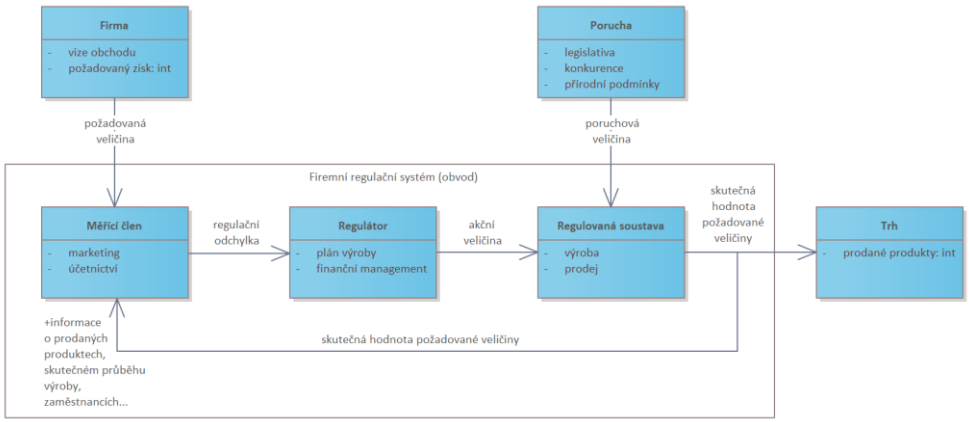
Informační systém si lze představit jako specifický typ regulačního systému. Jeho klíčovou vlastností je zpětná vazba pro korigování chování. Regulační systém lze dekompozicí rozložit na tři podsystémy: [5]

- **Řídící systém (regulátor)** – Slouží k ovládání řízeného systému za účelem dosažení a udržení jeho požadovaného stavu. Vstupní veličinou regulátoru je regulační odchylka, kterou lze vyjádřit jako rozdíl mezi požadovanou veličinou a skutečnou hodnotou regulované veličiny. Výstupní hodnotou regulátoru je akční veličina sloužící k ovládání regulované soustavy. Cílem regulátoru je nastavit takovou akční veličinu, aby se hodnota regulační odchylky blížila nule.

- **Řízený systém (regulovaná soustava)** – Zařízení, jehož chování je ovlivněno regulátorem. Vstupní hodnotou regulované soustavy je akční veličina z regulátoru doplněná o tzv. poruchovou veličinu, která vyjadřuje nepředvídanou změnu regulované veličiny. Výstupní hodnotou řízeného systému je skutečná hodnota požadované veličiny.
- **Měřící člen** – Zařízení sloužící k získávání regulační odchylky. Měří požadovanou veličinu i skutečnou hodnotu regulované veličiny a počítá jejich diferenci, kterou poskytuje regulátoru.

Lze-li chod systému definovat schématem regulačního obvodu, pak je rovněž možné pro tento systém vytvořit informační systém, jenž bude začleněn do stávajícího systému a podpoří jeho lepší chod. Schématem regulačního systému je rovněž možné popsat například většinu podnikových procesů. [5]

Příklad definice business procesů schématem regulačního obvodu je uveden i na následujícím obrázku:



Obrázek 2 – Analogie firemních procesů s regulačním obvodem [5; 38]

Z něho je patrné, že každá firma má svoji strategii vedoucí k naplnění požadovaných cílů a obchodních vizí. Na základě aktuálních finančních dispozic a poptávky trhu pak vedení firmy, regulátor, vybere nejvhodnější strategické kroky, od nichž se dále odvíjí např. plán výroby. Při ní však mohou nastat poruchy – zařízení se zničí, zaměstnanci onemocní, vichřice zničí střechu výrobní haly, zákazníci přejdou ke konkurenci… Na tyto situace musí umět vedení firmy pružně reagovat. Ve velkých podnicích však toto může být již velmi obtížné.

Stále více firem proto investuje do informačních systémů, které zmíněné podnikové procesy dokáží do jisté míry zautomatizovat. V roce 2022 využívalo podnikový informační systém (ERP) téměř 93 % českých firem zaměstnávajících více než 250 pracovníků. K největšímu nasazení systémů pro řízení podnikových procesů docházelo v IT a velkoobchodu. Více než třetina všech firem pak využívala nějaký informační systém. [6]

2 METODIKY VÝVOJE SOFTWARU

Metodikou pro vývoj softwaru se rozumí předem stanovené postupy, pomocí nichž vývojáři navrhují, implementují, testují a dokumentují nové programy. Cílem použití metodik při vývoji aplikací je koordinace činností a dostatečná informovanost všech členů vývojářského týmu vedoucí k maximalizaci efektivity práce. Metodika rozkládá náročný problém vývoje SW do několika menších a méně náročných etap, které jsou řešeny samostatně. Jednotlivé fáze jsou od sebe odděleny tzv. milníky. Touto separací je rovněž zajištěn pružnější přístup pro plánování. V této kapitole budou popsány nejpoužívanější metodiky při vývoji aplikací a budou prezentovány výhody i nevýhody jednotlivých přístupů.

2.1 Tradiční metodiky vývoje softwaru

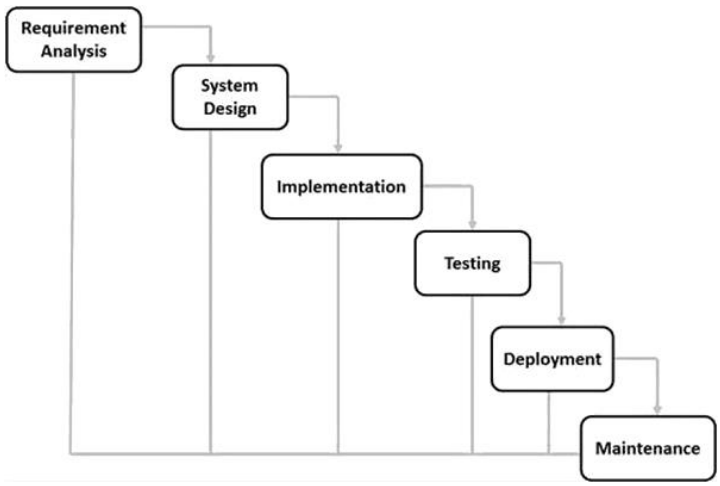
V roce 1948 napsal Tom Kilburn první počítačový program na světě, který obsahoval 17 instrukcí a sloužil ke hledání největšího společného dělitele mocnin čísla 2. Jednalo se o obrovský technický pokrok, který odstartoval postupné rozšíření aplikací pro výpočetní techniku. Metodiky pro vývoj softwaru v té době byly primitivní – vše záleželo na programátorech. Ti si shromažďovali své nápady, které uspořádali do návrhu a následně se tento návrh snažili implementovat v programu. Nebylo však blíže specifikováno, jakým způsobem mají své požadavky na systém dokumentovat. Masivnější rozvoj softwaru nastal až v 70. letech 20. století, kdy došlo k rozšíření počítačů ve firmách za účelem lepšího poskytování služeb svým zákazníkům. V té době začaly vznikat i první metodiky pro vývoj softwaru. Za významný milník, který odstartoval prudký rozvoj metodik při vývoji SW, je uváděn rok vynalezení vodopádové metodiky v roce 1970 Winstonem W. Roycem. [7]

2.1.1 Vodopádový (Waterfall) model

Vodopádový model byl rozšířenou metodikou při vývoji softwaru především v 70. a 80. letech 20. století. Inspiroval se tehdejšími vývojovými postupy v průmyslu. Tato metodika rozděluje proces vývoje SW do sedmi etap, přičemž každá fáze vývoje plynule přechází do další. Grafické znázornění této metodiky připomíná tekoucí vodopád (odtud pochází i název). Jednotlivé etapy metodiky jsou: [7; 8]

- **Požadavky** – Tato fáze se zabývá zachycením a dokumentací všech možných požadavků na nový systém.

- **Návrh –** V této etapě jsou zpracovávány zachycené požadavky, podle kterých je navrhována architektura nového systému. Dále jsou specifikovány hardwarové a softwarové požadavky.
- **Implementace** – V této fázi již vývojáři píší samotný kód aplikace. Veškerá implementace se řídi specifikacemi z etapy návrhu.
- **Integrace a testování** – V této fázi se nově vzniklý systém integruje s hardwarem a testuje. Etapa slouží ke kontrole, zda nová aplikace funguje bezchybně a splňuje-li všechny požadavky.
- **Instalace** – V této fázi je řádně otestovaný SW produkt hotový a dochází k jeho instalaci a spuštění u zákazníka.
- **Údržba** – Na základě softwarové smlouvy může vývojářský tým dále řešit jakékoli problémy odhalené při používání softwaru a vydávat aktualizace či opravy.



Obrázek 3 – Schéma vodopádového modelu [40]

Vodopádový model je snadno pochopitelný a lehce spravovatelný. Mezi jeho výhody patří možnost separace jednotlivých etap a také prostor pro kontrolu a řízení. Všechny fáze jsou jasně definovány, tudíž je možné vývoj softwaru snadno plánovat. Model vodopádu je vhodné použít, pokud jsou všem vývojářům požadavky velmi dobře známé, jsou jasně definované a neexistují mezi nimi nejednoznačnosti. Výborně se hodí pro menší projekty postavené na technologiích, s nimiž mají vývojáři bohaté zkušenosti a jsou pro ně srozumitelné. [7; 8]

Separace jednotlivých etap vývoje softwaru při použití metodik vodopádového modelu s sebou však přináší i nevýhody. Neumožňuje sledovat pracovní pokroky v jednotlivých etapách, při delších a složitějších projektech vnáší do vývoje velkou míru rizik a nejistoty a dovoluje pouze velmi malou interakci se zákazníkem. Pokud tedy dojde z jeho strany k rozsáhlejší změně požadavků, dochází k časovému zpoždění, neboť je nutné upravit všechny etapy. Při velmi rozsáhlé změně není výjimkou ani nutnost stávající vývoj projektu úplně

ukončit a začít znovu. Vodopádový model je nevhodný pro složité objektově orientované projekty nebo pro projekty s vysokou pravděpodobností rizika změny požadavků v pozdějších fázích vývoje. [7; 8]

2.1.2 Spirálový model

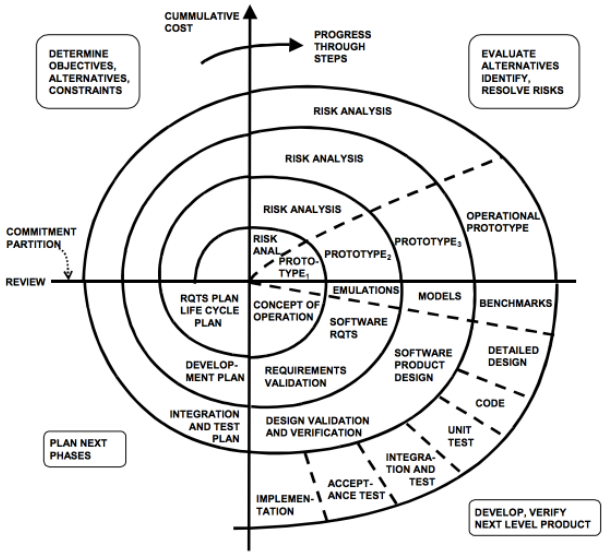
Spirálový model vytvořil Barry W. Boehm v roce 1988. Tento model částečně vychází z vodopádového modelu. Oproti němu klade důraz na předcházení rizik – jedná se o tzv. rizikově řízený přístup. Na rozdíl od Waterfall modelu nejsou jednotlivé etapy vykonávány sekvenčně, ale v několika iteracích. Klíčovým aspektem spirálového modelu je uspokojit požadavky zákazníka a všech koncových uživatelů tak, aby výsledný produkt splnil veškerá jejich očekávání. Výstupem každé iterace spirálového modelu je prototyp aplikace. Výsledný produkt je tedy zákazníkovi dodáván postupně a je v každé iteraci řádně otestován a vyhodnocen. Jedna iterace spirálového modelu je rozdělena do následujících 4 etap: [7; 8]

- **Plánování iterace** – V této etapě jsou specifikovány požadavky od zákazníka, které budou v dané iteraci řešeny. Dále jsou definovány cíle, omezení a identifikují se možná rizika dané iterace, která by při vývoji mohla nastat.
- **Analýza rizik** – V této fázi jsou analyzována specifikovaná rizika a stanovují se strategie vedoucí k jejich zmírnění. Tato rizika i strategie jsou podrobně dokumentovány.
- **Vývoj a testování** – V této fázi je vytvářen zdrojový kód aplikace a probíhá následné testování nově vzniklé části programu.
- **Vyhodnocení iterace** – Na konci každé iterace je provedeno její vyhodnocení. Nově vzniklý prototyp aplikace je předveden zákazníkovi, který vývojářskému týmu poskytne zpětnou vazbu. Hodnotí se především, zda prototyp již splňuje všechny požadavky na výsledný systém. Pokud jsou splněny, přechází se k instalaci softwaru a jeho průběžné údržbě dle metodik vodopádového modelu. V opačném případě dochází k zahájení plánování nové iterace založené na analýze nově vzniklého prototypu a celý proces se opakuje. [7; 8]

Spirálový model je výhodný především pro středně až vysoce rizikové projekty. Zmíněnými riziky může být například rozpočtové omezení vývoje aplikace, nebo náročné požadavky od zákazníka. Použití spirálového modelu lze uplatnit i v případě, kdy si zákazník není jistý všemi požadavky na nový software. Umožňuje totiž přidat nové specifikace bez konfliktu s předchozími požadavky a implementacemi. Zákazník svůj objednaný systém

dostává po menších částech a za kratší dobu, což je výhodné i pro vývojáře, neboť obdrží mnohem časnější zpětnou vazbu od všech uživatelů systému, než by tomu bylo v případě použití vodopádového modelu. Doba případných úprav daného prototypu se tak může výrazně zkrátit. [7; 8]

Použití spirálového modelu je nevýhodné pro malé nebo nízkorizikové projekty, neboť zbytečně komplikuje jejich jednoduchý vývoj. S tím narůstá i finanční nákladnost vývoje. Další nevýhodou je obtížné řízení takového projektu, které vyžaduje zkušené manažery. S narůstajícím počtem iterací se nadměrně zvyšuje i tvorba potřebné dokumentace pro takový projekt, který nabývá na složitosti. Znovupoužití zdrojového kódu v jiných projektech může být z důvodu velké míry přizpůsobení softwaru konkrétnímu zákazníkovi omezené. [7; 8]



Obrázek 4 – Schéma spirálového modelu [39]

2.2 Agilní metodiky vývoje softwaru

S tím, jak se během 80. let 20. století vývoj hardwarových komponent pro osobní počítače neustále zdokonaloval, rostla i poptávka po softwaru. Mnoho podniků si pořizovalo stále modernější výpočetní techniku a investovalo i do nových podnikových softwarů. Vývoj tehdejších aplikací však běžně trval i tři roky. Dlouhá čekací doba tak přestávala být kompatibilní se stále se zdokonalujícími firmami. Vývojáři naráželi na problémy, protože spousta aplikací během vývoje jednoduše zastarala, či pro ně zákazníci již nenašli uplatnění. Vše vyvrcholilo začátkem 90. let minulého století. Toto období je také nazýváno jako „krize vývoje aplikací“. Metodika vodopádového modelu přestávala být pro neustále rychleji se měnící požadavky na SW vhodná. Proto se více přistupovalo k vývoji s použitím iterativních metodik, vznikala různá vylepšení spirálového modelu. Vývojáři se snažili najít ještě

efektivnější metodiky, jež by umožňovaly větší flexibilitu. To podnítilo i setkání 17 špičkových programátorů v únoru 2001 v lyžařském středisku Snowbird v pohoří Wasatch v Utahu. Jedním z nich byl i Robert Cecil Martin, autor slavné knihy *Čistý kód*, v níž publikoval dnes všeobecně uznávané principy objektově orientovaného programování *SOLID.* Během tohoto uvolněného setkání vznikl *The Agile Manifesto*, manifest zdůrazňující potřeby alternativních metodik ke složitým iterativním metodikám, jež jsou náročné na dokumentaci. [9; 10]

Agilní metodiky staví na vysoké flexibilitě projektu, čímž šetrně zachází s časem i zdroji. Té je dosaženo rychlou zpětnou vazbou od zákazníka a ochotou vývojářského týmu přijímat změny v projektu. Díky přizpůsobivosti projektu je možné jej přirozeně upravovat dle aktuálních potřeb zákazníka. Agilní vývoj usiluje o krátkodobý vývojový cyklus. Vývojářský tým je menší, ale pevně provázaný. Agilní metodika se snaží rozdělit složitý projekt do několika menších, lépe zvladatelných etap, podobně jako tradiční metodiky. Na rozdíl od nich však nemá nastavená žádná striktní pravidla. Týmy by měly zůstat otevřené jakémukoliv vývoji, který povede k úspěšnému dosažení stanovených cílů. Mezi nejpoužívanější agilní metody plánování patří Scrum a Kanban. [11]

2.2.1 Scrum

Scrum je iterativní a inkrementální agilní metoda plánování. Model Scrum vymysleli Jeff Sutherland a Ken Schwaber, oba účastníci slavného setkání ve Snowbird, již počátkem 90. let. V té době ještě převládala tradiční vodopádová metodika pro vývoj SW. Termín „scrum“ pochází z rugby, kde označuje tým pracující na společném cíli. Scrum je založený na koncepci sestavení malých provázaných týmů. Scrum se oproti tradičním metodikám více zaměřuje na konkrétní cíle, kterých je nutné dosáhnout. Jak bude těchto cílů dosaženo, si určí tým sám. V každém týmu využívajícím Scrum jsou zastoupeny 3 role: [11]

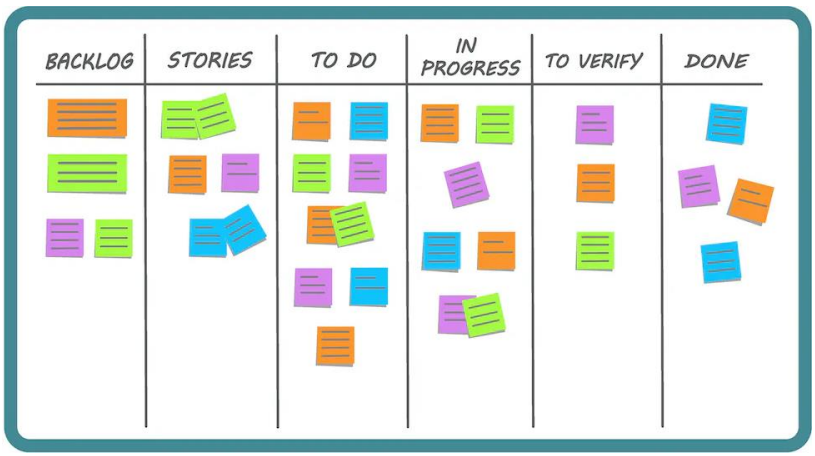
- **Product Owner** – jedná se o pověřenou osobu ze strany zákazníka, která má na starosti specifikaci požadavků a stanovení jejich priorit.
- **Tým vývojářů** – Vývojový tým je zodpovědný za dokončení úkolů v dané iteraci.
- **Scrum Master** – Tato osoba při sprintu zodpovídá za dodržování plánu práce a pomáhá týmu vývojářů řešit případné problémy.

Jedna iterace modelu Scrum je krátká a dělí se do dvou etap: [11]

Plánovací sezení – V této etapě se společně setkávají vývojáři s Product Ownerem, aby si stanovili cíle a priority dané iterace. Vývojáři vymýšlí efektivní způsoby, jak tyto cíle

dokončit a rozdělují si mezi sebe jednotlivé úkoly. Dále se snaží odhadnout vynaložené úsilí a sestavují tzv. Product Backlog. To je uspořádaný seznam všech pracovních činností, které bude potřeba vykonat pro úspěšné dokončení vývoje nové aplikace. Položka Product Backlogu vždy obsahuje časový odhad na vytvoření nové funkcionality a prioritu. Do sprintu jsou pak přednostně vybírány položky s vyšší prioritou.

Sprint – V období sprintu vývojáři pracují na dokončení cílů stanovených v sezení. Jeden sprint trvá obvykle dva týdny. Veškerá práce je během sprintu zaznamenávána na tabuli, která přehledně zobrazuje dokončené úkoly, úlohy, na kterých se aktuálně pracuje a úkoly, které teprve budou řešeny.



Obrázek 5 – Ukázka projektové tabule pro Scrum metodiku [41]

Mezi hlavní výhody Scrum patří velmi dobrá definice rolí a jejich odpovědností, což z velké části zamezuje nedorozuměním a konfliktům. Další předností je existence Product Backlogu, který všem pracovním činnostem stanovuje priority a očekávaný odhad pro dokončení, což umožňuje snadnější plánování i s ohledem na flexibilitu projektu. Je také snadné říct, v jakém stavu se projekt nachází, protože se všechny aktivity během sprintu zaznamenávají na tabuli. [11]

Největší nevýhodou Scrum metody je, že funguje pouze pro malé vývojové týmy, maximálně do 10 členů. Při větším počtu je již těžké v týmu udržet pevné vztahy. Řešením by bylo pro komplexní projekt složit více Scrum týmů s optimálním počtem členů, jenž budou pracovat na dílčích funkcionalitách, avšak koordinace takových týmů je značně složitá. [11]

2.2.2 Kanban

Kanban, podobně jako Scrum, klade důraz na neustálý vývoj a zlepšování procesu. Nemá však přesně stanovené časové rámce jednotlivých iteračních etap. Je více zaměřen na plnění úkolů na základě stanovených priorit a na vizualizaci práce. Slovo „kanban“ pochází

z japonštiny a označuje kartu obsahující veškeré nutné informace potřebné k dokončení produktu v každé fázi jeho řešení. Kanban, stejně jako Scrum, využívá tabuli pro přehlednou vizualizaci stavů všech úkolů vyvíjeného projektu. Kanbanové metody plánování však omezují počet karet umístěných na tabuli. Přidání nových úkolů je podmíněno dokončením úkolů předchozích. To má zabránit pocitu nátlaku na vývojáře a přetěžování členů týmu. V týmu Kanban nejsou povinné týmové role jako u Scrum metody. Vývoj při použití Kanban probíhá nepřetržitě a není rozdělen na etapy. [11]

Výhodou Kanban metodo plánování je snadná náročnost na pochopení oproti Scrum. Je vhodný i pro začínající vývojáře, kteří nemají tolik zkušeností. Ideální případ pro použití Kanban je při vývoji podobných projektů, které mají stabilní požadavky. [11]

Nevýhody Kanban plynou z nesprávného použití vizualizační tabule. Přestože metoda striktně žádné role nenastavuje, je vhodné pověřit osobu, která se bude starat o aktualizaci tabule. Nejčastější příčinou selhání Kanban metody je zastaralá deska, jež nezobrazuje aktuální stav projektu. V případě složitých projektů může být nevýhodou absence vývojových etap, která snadno vyústí v problémy se zpožděním při doručení softwaru. [11]

2.3 Metodika Unified Process

Unified Process je iterativní a přírůstková metodika zaměřující se na důležitost včasného plánování, předcházení rizikům a zvyšování produktivity práce formou sjednocení vývojových procesů. Klade důraz na znovupoužitelnost vytvářených SW komponent, čímž se výrazně krátí celková doba vývoje aplikace. Za zakladatele je považován Ivar Jacobson. Počátek vývoje Unified Process se datuje do roku 1967. Toho roku Jacobson pracoval ve firmě Ericsson, kde se podílel na zavádění nového vývojového modelu. Jeho princip spočíval v rozložení složitých systémů na menší vzájemně propojené komponenty, jejichž funkčnost je mnohem snadnější na pochopení, než systém jako celek. Jacobson roku 1987 založil vlastní softwarovou firmu Objectory AB, která vyvíjela a prodávala metodiku zaměřující se na tvorbu SW za pomoci množin diagramů. V roce 1995 byla firma Objectory AB prodána společnosti Rational, zaměřující se na vývoj pokročilých modelů pro vývoj softwaru. Jacobson se svým tehdejším týmem pracovali na sjednocení metodik obou společností. Výsledkem byla metodika ROP (Rational Objectory process). Z ní pochází i myšlenka zobrazení požadavků zákazníka v diagramu případů užití (use case), která je základním kamenem UP i modelovacího jazyka UML a snaží se zachytit všechny vztahy mezi aktéry (uživatele s konkrétními rolemi v systému) a dílčími funkcionalitami aplikace. V roce 1999 Ivar Jacobson publikuje svou knihu

Unified Software Development Proccess, ve které byla poprvé důkladně popsána UP metodika. Toto datum se často uvádí jako její vznik. Metodika UP však čerpá ze zmíněné starší metodiky ROP, Ericssonovy metody a několika dalších vývojových postupů. [12]

2.3.1 Struktura UP

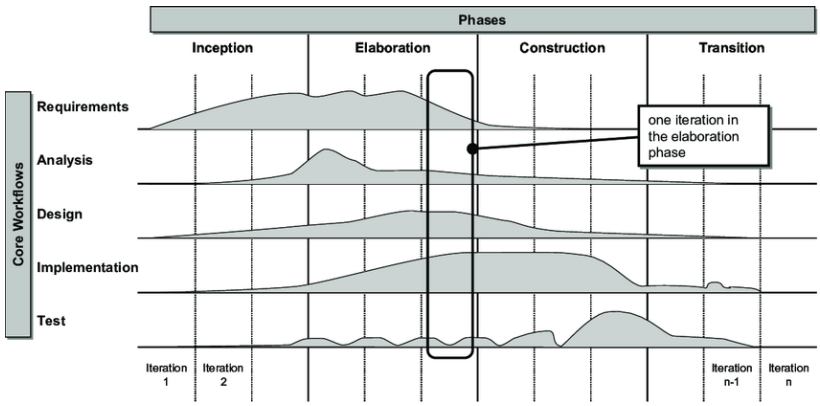
Životní cyklus projektu vyvíjeného metodikou UP je rozdělen do čtyř fází: [12]

- **Zahájení** – Tato fáze zahajuje vývoj projektu. Cílem fáze je sběr a dokumentace klíčových požadavků a identifikace rizik, která mohou při vývoji nastat. Požadavky se dělí na funkční a nefunkční. Funkční požadavky definují, co by měl systém dělat, nefunkční pak specifikují omezení systému (např. výkon a bezpečnost). V této fázi je vypracována i obchodní studie sloužící k obhajobě nově vznikajícího projektu a k odůvodnění potřeb jeho realizace. U složitých nebo vysoce rizikových projektů bývá vypracována i případová studie, jež má za cíl ověřit správnost logických rozhodnutí vyplývajících z požadavků za pomoci výpočetní techniky (např. tvorbou prototypu aplikace nebo vytvořením počítačové simulace).
- **Rozpracování** – Tato fáze je zaměřena především na tvorbu use case pro většinu funkčních požadavků definovaných při zahájení projektu. Jsou dokumentovány nové požadavky, jež byly opomenuty v prvotní etapě. Dále je implementován architektonický základ, což je částečně funkční verze systému. Tento základ je řádně otestován.
- **Konstrukce** – V této části probíhá většina implementačních kroků. Je dokončována tvorba analytických a návrhových modelů nového softwaru. V této etapě je také prostor pro odhalení drobných požadavků, jež mohly být do té doby opomenuty. Fáze zahrnuje i beta-testování aplikace, bývá v ní přistoupeno taktéž k pilotnímu testování u zákazníka.
- **Zavedení** – Tato fáze klade důraz na dokončení implementace a testování výsledného SW. Dochází k opravám nalezených chyb, které vyplynuly z výsledků testů z předchozích fází, a testují se dříve opomenuté části aplikace. Bylo-li při tvorbě projektu správně postupováno, netvoří se v této fázi téměř žádné nové požadavky.

Jednotlivé etapy životního cyklu Unified Process metodiky jsou od sebe odděleny milníky. Fáze životního cyklu je dále rozdělena do jedné, nebo více iterací dle složitosti vyvíjeného softwaru. Objem práce vykonaný v jednotlivých pracovních postupech dané iterace není stejný. Odvíjí se především od toho, v jaké fázi životního cyklu se projekt nachází. Každá

fáze na některé pracovní postupy iterace klade větší důraz. Jedna iterace se skládá z 5 pracovních postupů: [12]

- **Požadavky** – Specifikace nových funkcionalit softwaru. Největší část tvorby požadavků je prováděna ve fázích zahájení a rozpracování projektu.
- **Analýza** – Roztřídění a upřesnění požadavků. Většina analýz je vykonávána ve fázi rozpracování projektu.
- **Návrh** – Tvorba návrhové architektury pro požadavky. Většina práce je prováděna ve fázích rozpracování a konstrukce projektu.
- **Implementace** – Tvorba zdrojového kódu softwaru. Hlavní část probíhá ve fázi životního cyklu konstrukce.
- **Testování** – Ověření správnosti návrhu a implementace. Testování je prováděno průběžně ve všech etapách životního cyklu projektu s výjimkou fáze zahájení.



Obrázek 6 – Graf pracovního postupu při použití metodiky Unified Process [42]

Unified Process je týmová metodika. Nerozděluje tedy práci na jednotlivé úkoly pro konkrétní členy týmu, ale snaží se o vzájemnou spolupráci všech členů týmu ve všech fázích vývoje s cílem vytvořit ten nejlepší možný produkt kolektivně. UP klade velký důraz na dokumentaci a vývojové fáze. Unified Process předepisuje mnohem více pravidel než agilní metodiky. Tato pravidla mají za cíl předejít většině rizik a zpoždění při dodání softwaru, což pomáhá snížit náklady na vývoj. Vyvíjený produkt je rovněž řádně dokumentován. Použití UP metodiky nachází svoje uplatnění především u rizikových a rozsáhlých projektů, jejichž požadavky se mohou často měnit. UP je výhodný pro vývojářské týmy s větším počtem členů. Unified Process zároveň usiluje o vývoj znovupoužitelných komponent, čímž se může výrazně zkrátit celková doba vývoje softwaru. [12; 13]

Unified Process metodika vyžaduje určité zkušenosti a může být složitá na pochopení pro malé nebo začínající týmy. Nadměrná dokumentace však může vývoj jednoduchých a stabilních projektů silně prodloužit a prodražit. [12; 13]

Metodiky pro vývoj SW se neustále vyvíjí. Na trhu jsou i metodiky kombinující více přístupů, například Agile Unified Process. Neexistuje žádné obecné pravidlo, které by předepisovalo výběr metodiky pro konkrétní typy projektů. Je možné použít agilní vývoj i pro složité a náročné aplikace, stejně jako využít Unified Process metodiku pro malé projekty. Výběr metodiky nezáleží jenom na povaze projektu, ale i na složení týmu a osobních preferencích členů – někteří lidé mohou upřednostňovat více individualistický přístup zastoupený agilním vývojem, jiným členům je bližší vývoj týmový, do něhož spadá metodika UP.

3 ANALÝZA SOUČASNÉHO STAVU

Hasičský sport se v České republice těší velké oblibě. V Česku existuje přibližně 8 tisíc dobrovolných hasičských spolků. Především v malých obcích jsou hasičské sbory mnohdy jediným dobrovolnickým spolkem, který aktivně podporuje zájmovou činnost mládeže. Počet mladých hasičů mezi lety 2003-2018 vzrostl o dvě třetiny na 60 tisíc členů. Z toho se přibližně tři čtvrtiny mladých členů aktivně věnovalo hasičskému sportu. [14]

S rostoucím počtem mladých hasičů přibývá i počet soutěží. Oblíbený je především požární sport, konkrétně pak jeho „královská“ disciplína požární útok. V požárním útoku se pořádají různé regionální i celostátní ligy. Nejprestižnější tuzemskou soutěží je *Extraliga ČR v požárním útoku*, což je každoroční seriál přibližně 15 soutěží po celé České republice. Každý rok se zároveň konají tzv. „postupové soutěže“ v požárním sportu, jejichž organizátorem je *Sdružení hasičů Čech, Moravy a Slezska*. Soutěž má 4 kola – okrskové, okresní, krajské a republikové. V jednom kole se obvykle závodí ve více disciplínách – typicky v požárním útoku, štafetě 4 x 100 m s překážkami a běhu na 100 metrů s překážkami. Další disciplíny a pravidla jednotlivých disciplín vždy předepisují tzv. propozice, neboli dokument, jenž se řídí směrnicí pravidel požárního sportu a vydává jej pořadatel závodu. Postupové soutěže jsou pořádány odděleně pro dorostenecké týmy a pro dospělé, přičemž platí, že soutěž musí být dále oddělena zvlášť pro dobrovolnické spolky a zvlášť pro profesionální hasiče. [15]

3.1 Činnosti aktérů soutěže

Podle typu aktéra lze činnosti rozdělit do 3 částí. V této podkapitole budou vyjmenovány a popsány pouze činnosti, které mohou aktivně ovlivnit průběh soutěže či vyhodnocování výsledků. [15]

- **Činnosti pořadatele závodu**
    1. Pořádání ročníků závodu
    2. Tvorba rezervačních událostí pro soutěže
    3. Tvorba propozic pro soutěže
    4. Jmenování a odvolání rozhodčích a časoměřičů
    5. Zahajování a ukončování ročníků závodů
    6. Zveřejňování výsledků závodu na online portály

- **Činnosti rozhodčích**
    1. Zahajování a ukončování ročníků závodů
    2. Zahajování a ukončování disciplín ročníků závodů
    3. Zahajování a ukončování soutěžních kategorií disciplín ročníků závodu
    4. Penalizace nebo diskvalifikace soutěžících při porušení soutěžních pravidel
    5. Tvorba startovních a výsledkových listin
- **Činnosti časoměřičů**
    1. Měření soutěžních pokusů
    2. Přiřazování výsledků soutěžních pokusů závodníkům podle startovní listiny
    3. Penalizace nebo diskvalifikace soutěžících při porušení soutěžních pravidel
- **Činnosti soutěžících**
    1. Rezervace startovních pořadí před závodem
    2. Prezence nebo registrace před začátkem závodu
    3. Provedení soutěžního pokusu

3.1.1 Činnosti pořadatele závodu

Před závodem

Běžná činnost pořadatele začíná stanovením termínu soutěže. Nejpozději 14 dnů před termínem konání soutěže musí pořadatel zveřejnit její propozice. V případě soutěží, jež jsou zařazeny do soutěžních lig, které mají své vlastní předpisy opírající se o pravidla PS, může být podmínkou taktéž zajištění online rezervace startovních pořadí. Pořadatel musí před závodem určit sbor rozhodčích a organizační pracovníky soutěže v tzv. zvláštním předpise pořadatele. [15]

V průběhu závodu

Pořadatele v průběhu závodu zastupuje velitel soutěže uvedený ve zvláštním předpise pořadatele. Velitel soutěže je nadřazen všem rozhodčím, časoměřičům a organizačním pracovníkům soutěže. Velitel zabezpečuje zahájení a ukončení ročníku závodu, vyhotovuje startovní listinu pro účastníky soutěže, má právo přeřadit nebo odvolat podřízené rozhodčí a další pracovníky soutěže, pokud je to nutné. Má taktéž právo v případě potřeby přerušit soutěž. [15]

Po závodě

Po skončení závodu má pořadatel povinnost zveřejnit výsledky soutěže a zaslat zprávu o výsledcích soutěže SH ČMS. [15]

3.1.2 Činnosti rozhodčích

Role rozhodčích jsou hierarchicky uspořádané. Sbor rozhodčích je složen z hlavního rozhodčího, rozhodčích disciplíny, rozhodčích soutěžních kategorií disciplíny, časoměřičů disciplíny a časoměřičů soutěžní kategorie disciplíny. Nejvýše postavenou funkcí je hlavní rozhodčí, který je podřízen veliteli závodu a nadřízen všem rozhodčím disciplíny, rozhodčím soutěžních kategorií disciplíny a taktéž všem časoměřičům. Doba platností všech rolí disciplíny je omezena na určitý časový úsek v průběhu závodu. [15]

Činností hlavního rozhodčího

Hlavní rozhodčí navrhuje předpokládané začátky disciplín a soutěžních kategorií disciplín a má právo ovlivňovat jejich průběh. Je též oprávněn vyloučit družstvo ze soutěže při porušení pravidel. Je taktéž nařízen všem časoměřičům závodu, ale není oprávněn měnit naměřené hodnoty výsledných časů soutěžních pokusů. Při větších soutěžích může hlavní rozhodčí se souhlasem pořadatele závodu jmenovat svého zástupce. Role hlavního rozhodčího je platná v celém průběhu závodu. [15]

Činnost rozhodčího disciplíny

Rozhodčí disciplíny má právo na řízení průběhu dílčí disciplíny. Je podřízen hlavnímu rozhodčímu disciplíny a nadřízen všem rozhodčím soutěžních kategorií pro danou disciplínu. Má právo vyloučit družstva z provádění podřízené disciplíny při nerespektování pravidel. Je zodpovědný za pravdivost výsledků uvedených ve výsledkových listinách dané disciplíny. Je taktéž podřízen všem časoměřičům disciplíny, ale nemá oprávnění měnit naměřené hodnoty výsledných časů soutěžních pokusů. Při větších soutěžích může rozhodčí disciplíny se souhlasem pořadatele závodu jmenovat svého zástupce. Role rozhodčího disciplíny je platná po celou dobu průběhu disciplíny. [15]

Činnost rozhodčího soutěžní kategorie disciplíny

Rozhodčí soutěžní kategorie disciplíny řídí průběh soutěžní kategorie dané disciplíny. Je podřízen rozhodčímu disciplíny a nadřízen pomocným rozhodčím dané soutěžní kategorie disciplíny. Je nadřízen taktéž časoměřiči soutěžní kategorie disciplíny, avšak nemá právo měnit

naměřené výsledné časy soutěžních pokusů. Role rozhodčího soutěžní kategorie disciplíny je platná v celém průběhu této kategorie. [15]

Činnosti časoměřiče disciplíny

Hlavní rolí časoměřiče je přidělit naměřené časy soutěžícím uvedeným ve startovní listině pro danou disciplínu. Časoměřič disciplíny je podřízen rozhodčímu dané disciplíny a nadřízen všem časoměřičům soutěžních kategorií pro dílčí disciplínu. Časoměřič má zároveň oprávnění rozhodčího na udělení penalizace pro soutěžní pokusy, ve kterých závodníci porušili pravidla disciplíny. Role časoměřiče disciplíny je platná po celou dobu průběhu dané disciplíny. [15]

Činnosti časoměřiče soutěžní kategorie disciplíny

Časoměřič soutěžní kategorie disciplíny je nadřízen časoměřiči disciplíny. Má právo přidělovat výsledné časy pokusů soutěžícím ve startovní listině pro danou soutěžní kategorii disciplíny a je oprávněn tyto pokusy rovněž penalizovat. Role časoměřiče soutěžní kategorie disciplíny je platná po celou dobu průběhu dané kategorie. [15]

3.1.3 Činnosti soutěžících

Role soutěžícího nabývá platnosti provedením registrace při prezenci v závodě nebo provedením on-line rezervace. Role zaniká při ukončení soutěžní kategorie disciplíny, v níž byl tento závodník registrován. [15]

Před závodem

Soutěžící se registruje do závodu osobně v době prezence, nebo před závodem ve stanoveném čase dle pokynů uvedených v propozicích soutěže. Registrace a rezervace se podávají vždy do příslušné soutěžní kategorie disciplíny ročníku závodu. Podle pravidel PS nemůže soutěžící startovat na jednom závodě ve více disciplínách. [15]

Během závodu

Soutěžící musí úspěšně vykonat minimální počet soutěžních pokusů stanovených v propozicích soutěže, nejméně však 1. [15]

3.2 Existující pomocné prostředky pro pořadatele a rozhodčí

Všichni rozhodčí musí být patřičně proškoleni a potřebují mít platnou kvalifikaci. Úroveň kvalifikace se dle pravidel PS odvíjí od typu soutěže. Při krajských a republikových

postupových soutěžích musí mít rozhodčí vyšší úroveň kvalifikace než při regionálních soutěžích nebo u soutěžních lig, které se striktně nedrží všemi pravidly PS. Měření pokusů soutěžících může být prováděno stopkami, anebo elektronickou časomírou. Při krajských a republikových kolech postupových soutěží pravidla PS nařizují použití elektronické časomíry s přesností měření na dvě a více desetinných míst. [15].

Potřebný stupeň kvalifikace rozhodčího a použití elektronické časomíry při vyšších úrovních soutěží mají zajistit regulérnost soutěží. Tato opatření však nejsou odolná proti riziku lidského pochybení, ke kterému může dojít např. při přepisování výsledků z časomíry do pomocných programů sloužících k vyhodnocení výsledků. V této podkapitole budou představeny existující programy, které se při měření soutěží PS běžně využívají, bude analyzováno zabezpečení tohoto softwaru proti vzniku lidské chyby a bude kontrolován soulad tohoto software s aktuálními pravidly PS.

Programy pro zpracování výsledků POŽÁRNÍ SPORT

Jedná se o oficiální tabulky, které vydává a aktualizuje SH ČMS. Tabulky jsou volně k dispozici na webovém portálu mládeže SH ČMS a jsou implementovány v tabulkovém procesoru Microsoft Excel za podpory maker. V dokumentu MS Excelu je možné jednorázově zadat jména soutěžících v daných kategoriích a následně generovat startovní listiny jednotlivých disciplín a jejich soutěžních kategorií. Tento sešit MS Excelu umožňuje rovněž generování tiskových sestav všech soutěžních listin. [16]

Výhodou těchto tabulek je, že jsou normalizována všechna jména soutěžících v rámci sešitu. Také vyhodnocování časů je poloautomatizováno, to znamená, že po zadání výsledků soutěžících se automaticky vypočítají pořadí disciplín a pořadí závodu dle pravidel PS. Dalším kladem je fakt, že SH ČMS, jakožto vydavatel pravidel požárního sportu, tento program průběžně aktualizuje. Aktuální verze programu (k r. 2023) je ze 3. května.

Nevýhodou těchto tabulek je jejich vázanost na komerční aplikaci MS Excelu. Protože se nejedná o plnohodnotný program, není možné do budoucna očekávat např. podporu pro automatické načítání dat z časomír pro požární sport, jež umožňují komunikaci s PC. Offline tabulky jsou nevhodné pro jakékoliv online sdílení dat, které je nutné např. pro rezervace, vyhodnocování, anebo k informativním účelům pro soutěžící a veřejnost.

Webový portál firesport.eu

Firesport.eu je oblíbený webový portál, který slouží k vyhledávání, prohlížení a administraci hasičských soutěží. Obsahuje informace o téměř 12 000 ročnících závodů z ČR a SR

(k r. 2023). Web zobrazuje i podrobné statistiky výsledků soutěží, disciplín, týmů a soutěžních lig. Nejvyužívanější je tento web pro rezervace startovních pořadí, do systému bylo od jeho vzniku zadáno téměř 160 000 rezervací (k r. 2023). Rezervace jsou využívány především pro závody spadající do seriálů sportovních lig zejména v požárním útoku, ačkoli portál je určen pro všechny disciplíny PS. [17]

Nesporně užitečnou funkcí tohoto webu je podávání on-line rezervací pro dílčí soutěžní kategorie disciplíny ročníku závodu. Pro vkládání závodů, výsledků i rezervací na soutěž je nutná registrace na portálu, která však je bezplatná. Portál je především informativního charakteru. Poskytuje statistiky ze soutěží, které jsou atraktivní pro diváky a příznivce PS. Umožňuje soutěžícím vkládat odkazy na svá videa z pokusů, obsahuje tipovací sekci pro výsledky, umožňuje diskusi přihlášených uživatelů.

Portál není primárně určen ke zpracování výsledků. Postrádá sekci pro rozhodčí, kteří by mohli výsledky zadávat na web on-line již v době měření. Pokud chce pořadatel závodu zveřejnit své výsledky na portále, je nutné data importovat či přepsat z výsledkových listin soutěže. V tomto kroku může nastat problém a web tak nebude obsahovat platné údaje.

SW pro ukládání dat na webový portál pro časomíry LV

LV jsou časomíry vyráběné Ing. Liborem Valešem. Výrobce na svém webu casomiry.com nabízí měřící techniku nejen pro soutěže v požárním sportu. Na své výrobky poskytuje nadstandardní záruku 7 let. Tento výrobce hasičských časomír nabízí zdarma rovněž SW pro ukládání dat na webový portál casomiry.com. Tento webový portál umožňuje podávat online rezervace na soutěže, online zobrazování a export startovních a výsledkových listin. Výsledné časy se po jejich vložení a přiřazení soutěžícím ve startovních listinách automaticky vyhodnocují. Zdarma nabízený SW slouží pro přenos výsledných časů mezi časomírou a zmíněným portálem. Časy je možné do portálu vkládat i ručně. Verze webového portálu i SW pro přenos výsledků nejsou plnohodnotné (k r. 2023). Dle výrobce se jedná o intenzivně testovanou demoverzi. [18]

Tento portál, ač v demoverzi, nabízí zřejmě nejpropracovanější volně dostupné řešení ke zpracování výsledků z časomíry pro požární sport. Pro časomíry LV je dostupný software umožňující online ukládání naměřených výsledků na portál bez přepisování časů, které do procesu vnáší riziko lidského pochybení.

Webový portál je však určen především pořadatelům a časoměřičům disciplíny. Portál žádným způsobem neošetřuje oprávnění pro vkládání a úpravu naměřených časů. Pokud má

časoměřič přístupové údaje, může nahrávat časy, a to i tehdy, pokud soutěž již skončila, nebo aktuálně neprobíhá. Takové řešení není pro reálné použití dostatečné. Zároveň není zajištěna podpora pro ostatní rozhodčí, kteří jsou nadřazeni časoměřičům.

Komunikace s PC včetně SW k časomírám TRV elektronik

TRV elektronik je českým výrobcem sportovních časomír. Nabízí široké množství ukazatelů skóre i sportovních časomír včetně těch pro požární sport. Výrobce k hasičským časomírám nabízí rovněž obslužný SW, který slouží k ukládání naměřených časů a tvorbě startovních listin s možností jejich tisku. SW rovněž online zobrazuje aktuální časy z časomíry. Výrobce TRV jej na svém webu nabízí za cenu 2 196 Kč s DPH (k r. 2023). Podle informací z tohoto webu je možné další funkcionality aplikace na přání zákazníka doplnit. [19]

Aplikace Firesport Timekeeper

Firesport Timekeeper je formulářová aplikace Windows Forms vyvinutá autorem této diplomové práce. SW je podrobněji dokumentován v autorově bakalářské práci. Aplikace je rozdělena do 3 modulů, *Časomíra, Závody* a *Správa databáze*. Modul časomíry zajišťuje automatické ukládání naměřených časů pro jejich zálohu a další zpracování. V modulu je implementována vyhodnocovací logika, která automaticky detekuje neplatné pokusy na základě dat z přijatých časomíry. Tento modul je propojen s modulem *Závody*, který pro zpracování výsledků používá databázi a slouží ke tvorbě startovních listin jednotlivých závodů, přiřazování pokusů z modulu *Casomira* k soutěžícím ve startovních listinách a exportu těchto listin do tabulkového formátu *.csv* a *.xlsx*. Aplikace je navržena tak, aby většinu činností časoměřiče automatizovala a vyžadovala od něho pouze minimální kontrolu. Aplikace spolupracuje s časomírami LV vyráběnými Ing. Liborem Valešem. [20]

Aplikace *Firesport Timekeeper* je výhodná především k automatickému ukládání naměřených výsledků z časomíry do počítače k jejich dalšímu zpracování. Výhodou této aplikace oproti ostatním, jež byly v rámci této diplomové práce zkoumány, je zajištění automatické detekce neplatných pokusů na základě přijatých dat z časomíry. Ostatní testované programy ani časomíry tuto funkcionalitu neimplementují. Modul *Závody* je užitečný při vytváření startovních a výsledkových listin, umožňuje rovněž kopírování listin z již měřených závodů a další užitečné funkcionality při tvorbě startovních a výsledkových listin.

Aplikace je však, podobně jako *SW pro ukládání dat na webový portál pro časomíry LV*, určena převážně časoměřičům a neumožňuje kontrolu práce časoměřičů nadřízenými rozhodčími. Přestože software spolupracuje s databází, nebyla prozatím implementována

komponenta pokrývající možnost zadávání online rezervací. Aplikace neumožňuje ani spolupráci mezi pořadateli závodu, rozhodčími a soutěžícími.

3.3 Zhodnocení současného stavu

Protože obliba požárního sportu v České republice mezi mladými hasiči dlouhodobě roste, zvyšuje se i počet pořádaných soutěží v hasičském sportu. Navzdory tomuto trendu na současném trhu neexistuje komplexní aplikace, která by umožňovala online spolupráci mezi pořadateli závodu, rozhodčími i soutěžícími. Dostupné programy se zaměřují pouze na část operací spjatých s celkovou problematikou vyhodnocování výsledků. Tento software není vzájemně kompatibilní a nerespektuje hierarchii rolí pracovníků soutěže, především pak rozhodčích a časoměřičů, kterou stanovují pravidla pro požární sport.

4 POŽADAVKY NA NOVOU APLIKACI

Motivací pro návrh a implementaci nového systému určenému ke zpracování výsledků soutěží v požárním sportu je absence vhodného systému na aktuálním trhu. Současný stav, který nastiňuje pohled do problematiky vyhodnocování výsledků PS a analyzuje dostupné pomocné prostředky na trhu, je popsán v předchozí kapitole. V této kapitole bude podrobněji představen souhrn požadavků na aplikaci pro zpracování výsledků požárního sportu, která zajistí koordinaci mezi soutěžícími, rozhodčími a pořadateli závodu. Návrh těchto požadavků bude respektovat verzi pravidel pro požární sport z roku 2018. Cílem bude vyvinout bezpečný informační systém pro správu sportovních výsledků (ISSV), který bude dostupný pracovníkům soutěže, soutěžícím i příznivcům požárního sportu a přinese nejenom lepší informovanost soutěžících o výsledcích a statistikách ze soutěží, ale především ulehčí pořadatelům a rozhodčím náročnou organizátorskou práci a do určité míry zamezí chybným lidským rozhodnutím, která mohou nepříznivě ovlivnit výsledky soutěže.

4.1 Funkční požadavky

- FR1 ISSV bude zobrazovat výsledky aktuálně měřených soutěží.
- FR2 ISSV bude automaticky zpracovávat výsledky soutěží v požárním sportu.
- FR3 ISSV bude zobrazovat detaily o měřených soutěžích.
- FR4 ISSV bude zobrazovat startovní listiny závodů.
- FR5 ISSV bude zobrazovat výsledky závodů.
- FR6 ISSV bude zobrazovat rekordy sportovních výkonů vybraných soutěžních kategorií, disciplín a ve zvoleném časovém intervalu.
- FR7 ISSV bude zobrazovat rekordy sportovních výkonů vybraných soutěžních kategorií, disciplín, ve zvoleném časovém intervalu a ve vybraném závodě.
- FR8 ISSV bude zobrazovat osobní rekordy závodníků dle vybraných soutěžních kategorií, disciplín a ve zvoleném časovém intervalu.
- FR9 ISSV bude obsahovat administraci pro rozhodčí.
- FR10 ISSV bude obsahovat správu soutěžících pro oprávněné osoby s ohledem na omezenou dobu úprav dle průběhu závodu.
- FR11 ISSV bude obsahovat správu soutěžních pokusů pro oprávněné osoby s ohledem na omezenou dobu úprav dle průběhu závodu.
- FR12 ISSV bude obsahovat administraci pro časoměřiče.

- FR13 ISSV bude obsahovat správu sportovních soutěží pro oprávněné osoby.
- FR14 ISSV bude obsahovat online rezervaci na vybrané soutěže.
- FR15 ISSV bude umožňovat soutěžícím zaslat výsledky na e-mail v případě provedení online rezervace.
- FR16 ISSV bude umožňovat zasílání výsledků na e-mail pracovníkům soutěže.
- FR17 ISSV bude umožňovat registraci pro uživatele.
- FR18 ISSV bude umožňovat správu soutěžních pokusů oprávněným osobám s ohledem na omezenou dobu úprav u některých osob.
- FR19 ISSV bude umožňovat oprávněným osobám přidělení pokusů k soutěžícím dle startovní listiny s ohledem na omezenou dobu úprav u některých osob.
- FR20 ISSV bude umožňovat oprávněným osobám ukládat naměřené výsledky do systému s ohledem na omezenou dobu oprávnění u některých osob.
- FR21 ISSV bude umožňovat uživateli obnovit zapomenuté heslo.

4.2 Nefunkční požadavky

- NFR1 ISSV bude implementován jako webová aplikace s architekturou rozhraní REST.
- NFR2 ISSV bude naprogramován v jazyku Java s využitím frameworku Spring Boot.
- NFR3 Webové grafické uživatelské rozhraní ISSV bude vytvořeno pomocí knihovny React.
- NFR4 Jako databázový server bude použit Microsoft SQL Express.
- NFR5 ISSV bude umožňovat spolupráci se SW Firesport Timekeeper sloužícímu ke zpracování výsledků z časomíry pro požární sport.
- NFR6 ISSV bude pro přihlášené uživatele generovat JWT tokeny, aby došlo k šifrování přihlašovacích údajů a následnému zvýšení bezpečnosti systému.
- NFR7 ISSV bude implementovat několik úrovní oprávnění přístupu k uživatelským či administrátorským sekcím a umožní přístup do těchto sekcí pouze vybraným autoritám.
- NFR8 Design webového grafického uživatelského rozhraní ISSV bude responzivní.
- NFR9 ISSV bude ochraňovat veškerá uživatelská hesla patřičným šifrovacím algoritmem.
- NFR10 ISSV bude dostupný online prostřednictvím webového prohlížeče.
- NFR11 ISSV bude v případě potřeby umožňovat snadné rozšíření o nové funkcionality.
- NFR12 Rozhraní REST API bude podrobně dokumentováno pomocí specifikace OpenAPI za použití nástroje Swagger.

- NFR13 Databáze Microsoft SQL Server Express bude kontejnerizována pomocí nástroje Docker.
- NFR14 Doby platnosti uživatelských rolí souvisejících s administrací závodů se budou řídit pravidly požárního sportu z roku 2018. Tyto doby se odvíjí od aktuálního průběhu závodu. Tím bude zamezeno pozdější neoprávněné změně soutěžních výsledků.
- NFR15 ISSV bude pro zjednodušení práce se zeměpisnými souřadnicemi a údaji o adresách používat Mapy.cz REST API.

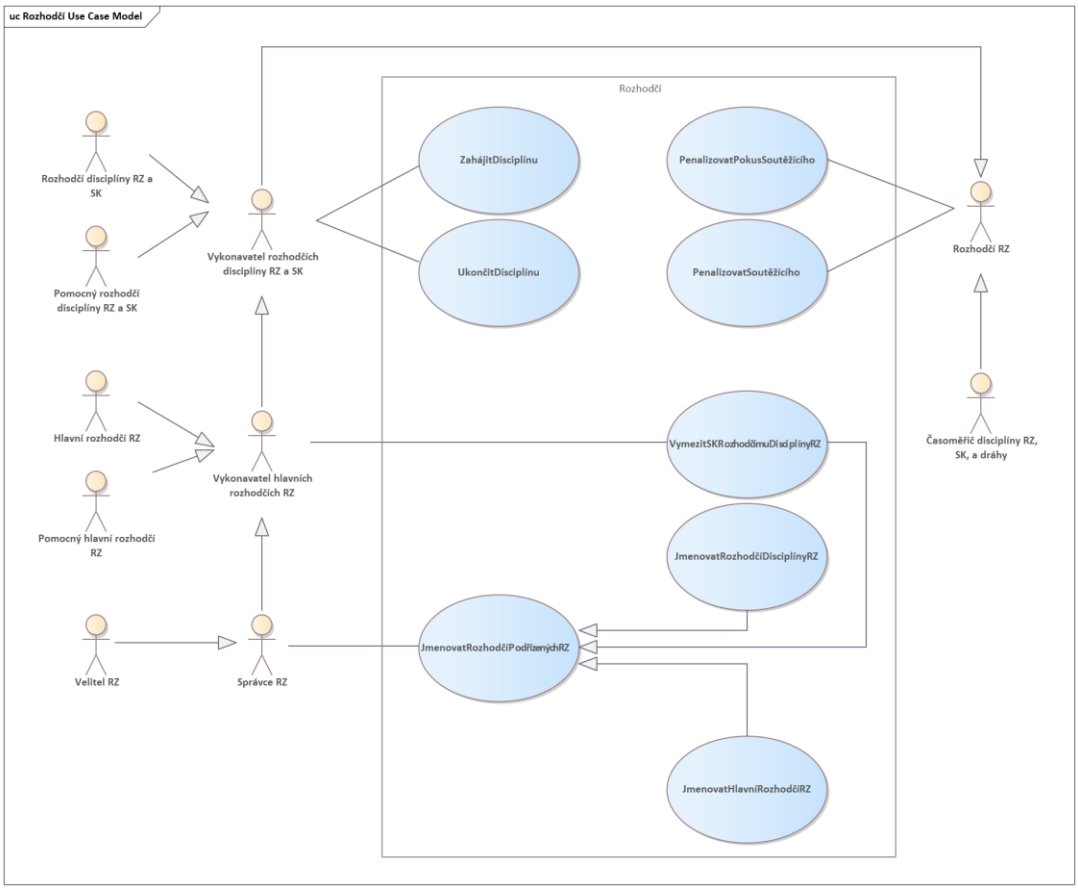
4.3 Diagramy případů užití

Při návrhu SW pro zpracování výsledků z časomíry pro požární sport bylo vytvořeno několik diagramů případů užití. Diagramy případů užití jsou doplňkovou dokumentací požadavků. Pomáhají určit hranice systému, specifikovat aktéry, kteří se systémem pracují a zobrazují vztahy mezi aktéry a případy užití. Všechny diagramy případů užití byly vytvářeny za pomoci grafického jazyka UML v souladu s pravidly metodiky Unified Process. K vytváření diagramů byl použit software Enterprise Architect. Model případů užití je poměrně rozsáhlý. V této podkapitole budou představeny pouze nejpodstatnější diagramy, jenž se týkají administrace závodů, administrace systému a autentizace či autorizace uživatelů. Ostatní diagramy včetně slovníčku pojmů a zkratek, které se v diagramech vyskytují, budou součástí přílohy této diplomové práce. U důležitých případů užití jsou rovněž definovány scénáře.

4.3.1 Velitel ročníku závodu a rozhodčí

Všichni rozhodčí jsou podřízeni veliteli ročníku závodu, který je může odvolat ze své funkce. Rozhodčí může v době platnosti role penalizovat pokusy soutěžících příslušné soutěžní kategorie disciplíny ročníku závodu. Může rovněž zahajovat a ukončovat příslušné disciplíny a soutěžní kategorie disciplíny, na něž má oprávnění. Hierarchie rolí je na obrázku vyznačena pomocí generalizace aktérů. Tato hierarchie vyplývá z hierarchicky uspořádaných rolí pracovníků soutěže, tak jak je definují pravidla požárního sportu. Dědičnost rolí v diagramu nelze zjednodušit, protože podle nefunkčního požadavku NFR14 musí systém respektovat pravidla požárního sportu.

Use case diagram funkcí rozhodčích je znázorněn v následujícím obrázku:

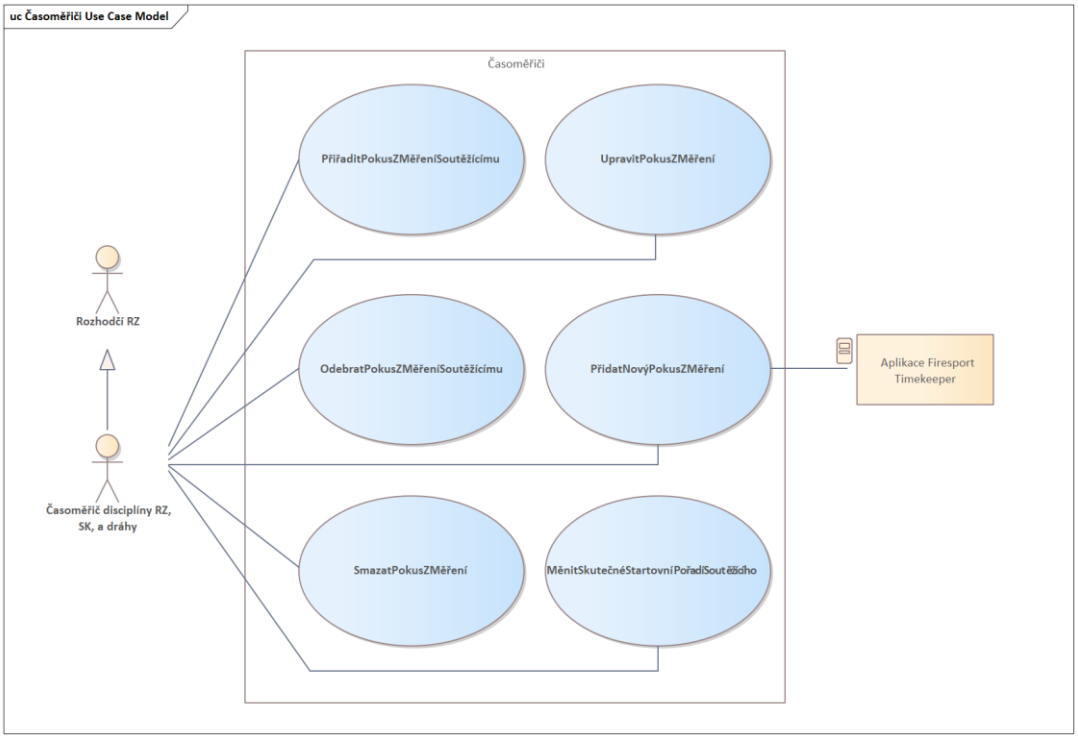


Obrázek 7 – Diagram případů užití – Rozhodčí

4.3.2 Časoměřiči

Časoměřič má na starosti úpravu veškerých naměřených sportovních výsledků. Tyto časy může v době platnosti své role zaznamenávat do systému a upravovat je. Časoměřič má zároveň částečnou pravomoc rozhodčího, protože může penalizovat soutěžící i soutěžní pokusy příslušných soutěžních kategorií či disciplín, pro něž má oprávnění. Tato skutečnost je v grafu znázorněna dědičností od rozhodčího ročníku závodu. Časoměřič je zároveň podřízen rozhodčímu příslušné soutěžní kategorie či disciplíny – ukončí-li nadřízený rozhodčí měření této kategorie či disciplíny, časoměřič již nemá oprávnění do naměřených dat jakkoli zasahovat (v tomto diagramu není vyznačeno). Další hlavní úlohou časoměřiče je v době platnosti role přiřazovat či odebírat pokusy soutěžícím dle startovních listin příslušné soutěžní kategorie. Do budoucna aplikace umožní spolupráci se SW Firesport Timekeeper.

Use case diagram pro časoměřiče vypadá následovně:



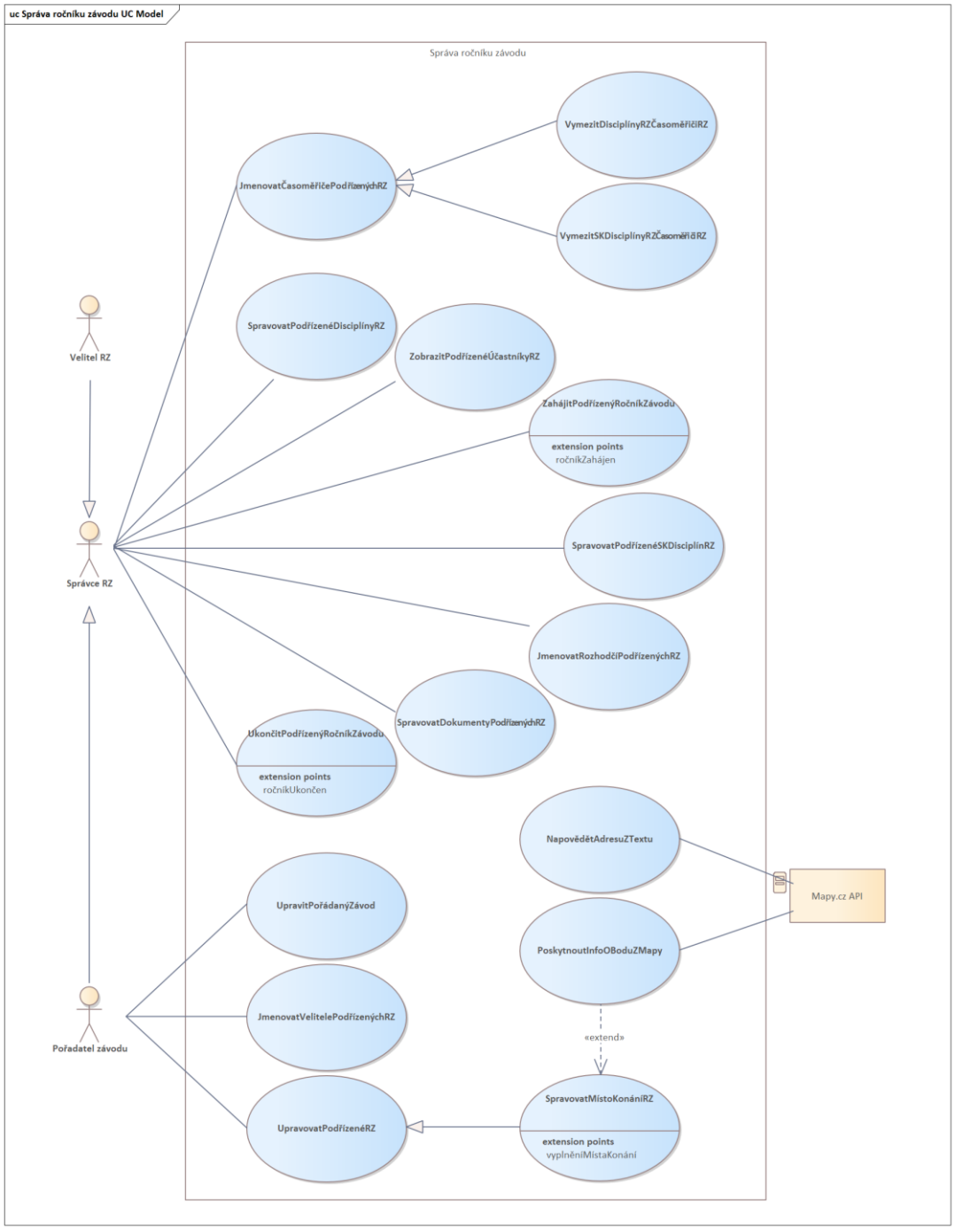
Obrázek 8 – Diagram případů užití – Časoměřiči

4.3.3 Správci závodu

Hlavním správcem závodu je pořadatel závodu. V průběhu závodu jej zastupuje velitel jmenovaný pro konkrétní ročník závodu. Velitel závodu zajišťuje hladký průběh soutěže. Velitele ročníku závodu jmenuje pořadatel závodu před jeho začátkem. Platnost role velitele ročníku závodu je omezena pouze na jeden konkrétní ročník soutěže. Velitel závodu má pravomoc všech časoměřičů i rozhodčích v daném ročníku závodu a tyto pracovníky může jmenovat a odvolávat ze své funkce, případně jim upravovat pravomoci pro disciplíny a soutěžní kategorie daného ročníku závodu.

Pořadatel soutěže má dále oprávnění týkající se úprav informací o příslušném závodě. Jsou to údaje o místě konání ročníku závodu v podobě GPS souřadnic, případně dle zadané adresy, a další informace. Pro zjednodušení práce se zeměpisnými souřadnicemi bude použito REST API Mapy.cz. Podrobněji je toto API popsáno v následující kapitole, která se věnuje použitým technologiím v praktické části této diplomové práce.

Nejdůležitější případy užití pro správce závodu shrnuje následující use case diagram:



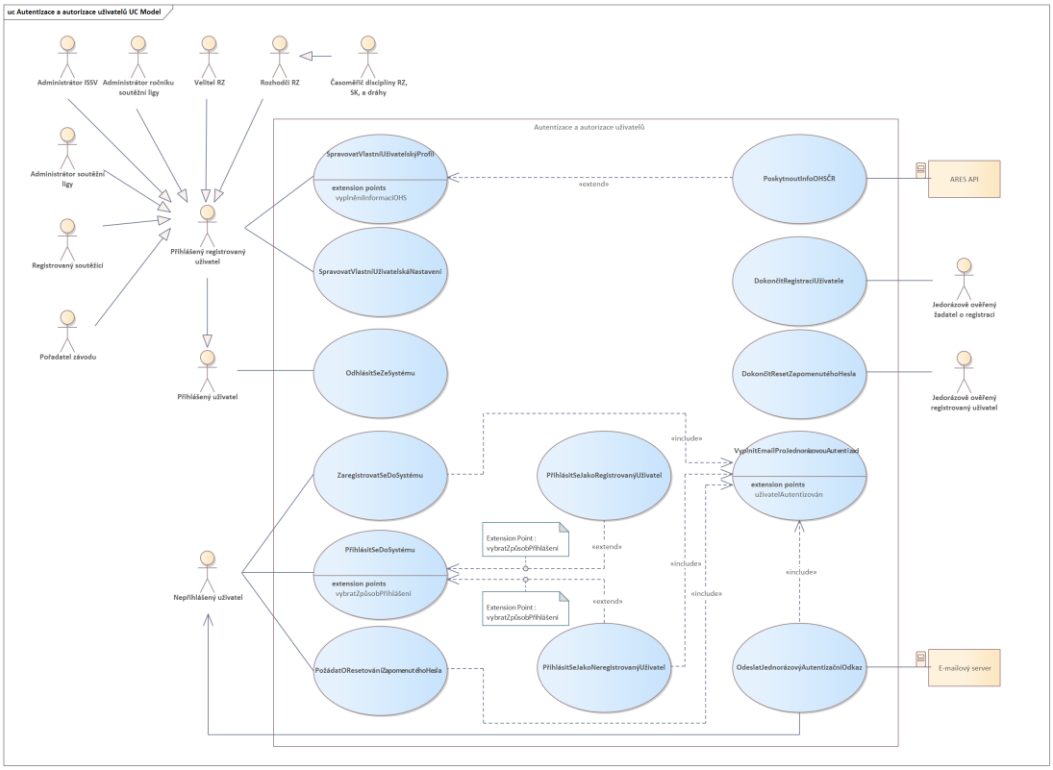
Obrázek 9 – Diagram případů užití – Správci závodů

4.3.4 Autentizace a autorizace uživatelů

ISSV bude umožňovat registraci pro uživatele. Registrovaní uživatelé získají výhodu v podobě možnosti podávání online rezervací do systému. Bude umožněna registrace jednak pro soutěžní týmy, jednak pro fyzické osoby (jednotlivce). Fyzickým uživatelským účtům bude moct správce systému přidělit další role týkající se správy a měření závodů. Navrhovaný systém bude

rozdělen do několika logických částí seskupujících funkcionality spojené s konkrétními rolemi. Uživatelské rozhraní bude jednotné pro všechny uživatelské role, včetně role správce ISSV. Na základě autorizace po přihlášení systém zpřístupní uživateli pouze logické části, na které má potřebná uživatelská oprávnění.

Případ užití pro autentizaci a autorizaci uživatelů se všemi výše popsanými funkcionalitami znázorňuje následující obrázek:



Obrázek 10 – Diagram případů užití – autentizace a autorizace

Z důvodu zvýšení bezpečnosti bude systém implementovat dvoufázovou registraci – po vyplnění registračního formuláře bude uživateli na e-mail odeslán odkaz s vygenerovaným jedinečným tokenem pro ověření uživatelského účtu. Pro úspěšné dokončení registrace bude muset uživatel v omezeném časovém intervalu tento odkaz otevřít, čímž dojde k aktivaci jeho účtu. V opačném případě bude jeho e-mailová adresa uvolněna pro další registrace.

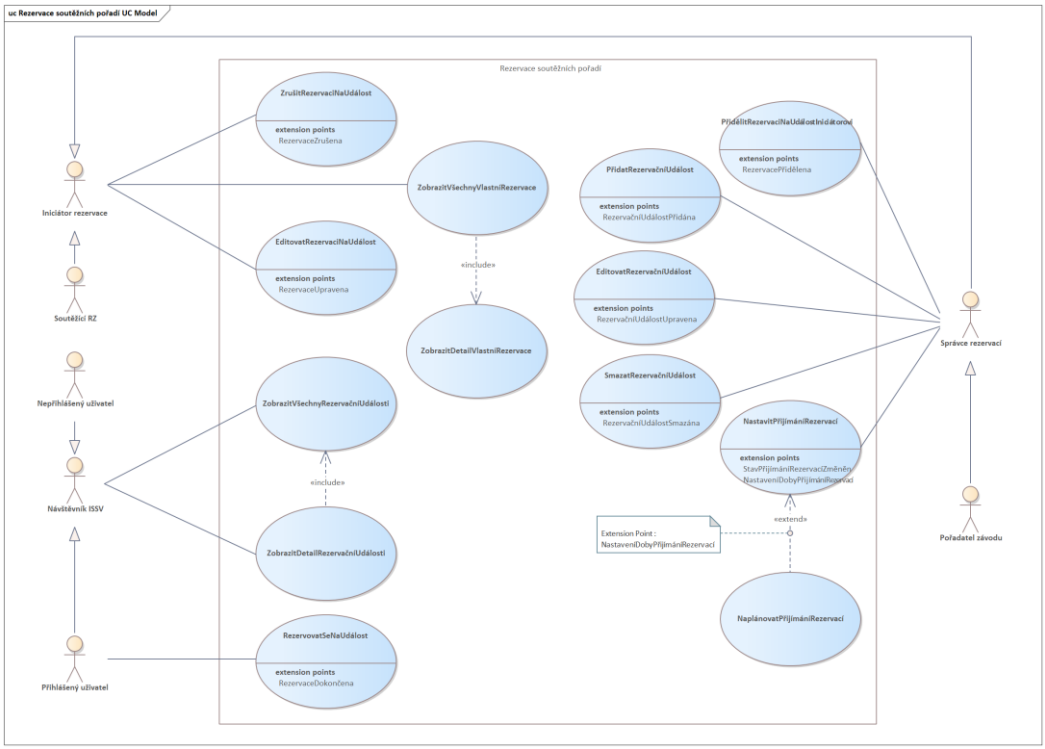
Pokud uživatel heslo zapomene, bude si moct své heslo resetovat. Tento reset hesla rovněž využije ověření identity pomocí e-mailu. Pokud uživatel správně zadá svůj e-mail, pomocí něhož se do systému registroval, bude mu opět vygenerován a odeslán jedinečný časově omezený odkaz s tokenem sloužícím pro změnu hesla.

Pro snazší pochopení jsou u některých use case týkající se autentizace a autorizace definovány scénáře, které podrobně popisují dílčí procesy z pohledu systému a aktéra, definují vstupní a výstupní podmínky pro úspěšné dokončení tohoto procesu a stručně specifikují možné

alternativní scénáře, jež mohou při vykonávání tohoto procesu nastat. Scénáře jsou součástí Use Case modelu vypracovaném v programu *Enterprise Architect* pro autentizaci a autorizaci uživatelů, který je zahrnut v příloze této diplomové práce.

4.3.5 Rezervace soutěžních pořadí

Případ užití znázorňující funkcionalitu rezervací SP je na následujícím obrázku:



Obrázek 11 – Diagram případů užití – rezervace soutěžních pořadí

Proces rezervace soutěžních pořadí je rozdělen do dvou úrovní oprávnění. Tou první je uživatelská část, která umožní uživateli systému ISSV registrovat se na závod. Administrátorská část slouží pro pořadatele závodu k vytváření rezervačních událostí a ke správě rezervací. Pokud je aktivní rezervační událost příslušné disciplíny ročníku závodu, mohou se uživatelé rezervovat na pozici v soutěžní pořadí, která doposud nejsou obsazena žádným jiným uživatelem. Po ukončení rezervace mohou soutěžní pořadí upravovat pouze pořadatelé závodu a dále také rozhodčí v průběhu závodu (tento případ užití není v následujícím Use Case diagramu zahrnut, ale je součástí přílohy diplomové práce v části věnované správě soutěžních pořadí). Nadřazenost pořadatele závodu, jakožto správce rezervací, je v grafu reprezentována pomocí generalizace aktérů. Use Case model rezervací je generalizován, protože je v projektu implementován rovněž případ užití pro rezervace časoměřiče na měření soutěží, který má obdobné chování. Tento krok bude realizován v další etapě vývoje ISSV a jeho Use Case diagram je rovněž součástí přílohy diplomové práce.

5 POUŽITÉ TECHNOLOGIE

Tato kapitola bude věnována technologiím, jež byly využity při tvorbě praktické části této diplomové práce. V úvodu kapitoly budou stručně zdůvodněny důvody pro výběr dílčích technologií, další podkapitoly již představí samotné technologie, jejich funkce a dostupné edice.

5.1 Výběr použitých technologií

Při výběru technologií byl brán zřetel na několik kritérií. Jednalo se o technologie předepsané v zadání této práce, autorovy předchozí zkušenosti s technologiemi a byl kladen důraz na moderní vývoj webových aplikací. Další podmínkou byla kompatibilita vyvíjeného systému s již existující aplikací *Firesport Timekeeper*, která by, po úpravě, měla být kompatibilní s nově vyvíjeným systémem pro zpracování soutěžních výsledků. Tuto podmínku definuje nefunkční požadavek NFR5.

Zadání diplomové práce předepisuje použití technologie Spring Boot pro back-end a knihovny React pro front-end. Podmínkou pro splnění zadání je i použití relační databáze a objektově-relačního mapování. Konkrétní databáze specifikována není. Software Firesport Timekepeer však pro zpracování výsledků z časomíry využívá databázi Microsoft SQL Server Express. Z důvodu zajištění kompatibility s tímto SW byla stejná databáze vybrána i pro nově vznikající aplikaci. Databáze však bude nově kontejnerizována pomocí nástroje Docker, což zajistí izolaci služeb a procesů databáze, přinese snadnější vývoj i zavádění aplikace a umožní snadnou virtualizaci databázového serveru.

5.2 Spring framework a Spring Boot

V této podkapitole budou popsány frameworky Spring a Spring Boot, a budou vysvětleny rozdíly mezi oběma pojmy, které jsou často nevhodně zaměňovány.

5.2.1 Spring framework

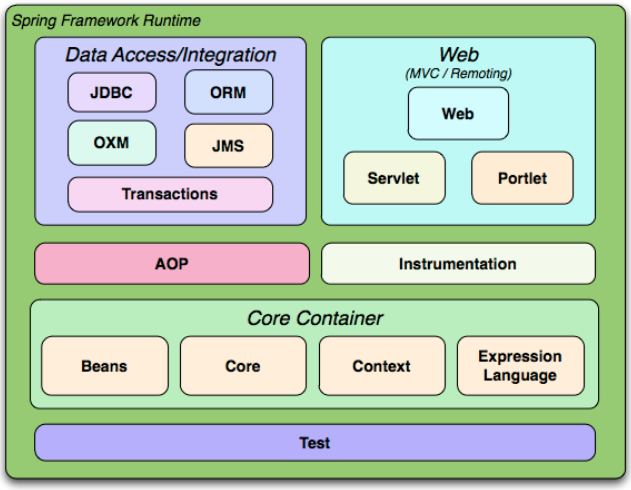
Spring je open-source framework vyvinutý pro programovací jazyk Java, který se používá pro usnadnění vytváření webových aplikací. První verzi napsal Rod Johnson v roce 2002, oficiální verze vyšla o rok později pod licencí Apache 2.0. Klíčovou vlastností Spring frameworku je automatická injektáž závislostí (Depenedency Injection), která snižuje složitost kódu.

Architektura Spring frameworku se skládá ze 6 vrstev. Tyto vrstvy podporují vývoj multiplatformních podnikových aplikací a informačních systémů se softwarovou architekturou MVC. [21; 22]

Jednotlivé vrstvy Spring frameworku jsou následující: [22]

- **Core Container** – Jedná se o základní modul Spring frameworku, jenž obsahuje jádro a dále moduly pro tvorbu Beans, kontextu a výrazové moduly. Beans přináší sofistikovanou implementaci návrhového vzoru Factory, čímž odpadá potřeba vytváření singletonů ve zdrojovém kódu. Namísto toho modul potřebné instance automaticky spravuje, čímž je oddělena konfigurace a deklarace závislostí od skutečné logiky programu. Modul kontextu svou funkčnost odvozuje z modulu Beans a poskytuje podporu pro načítání zdrojů, ověřování a šíření událostí nebo internacionalizaci a lokalizaci. Modul Expression Language je výkonný dotazovací jazyk, který slouží pro manipulaci s objekty za běhu aplikace.
- **Data Access/Integration** – Tato integrační vrstva poskytuje prostředky pro efektivní přístup k datům vyvíjené aplikace. Obsahuje modul ORM pro integraci s API sloužících k objektově-relačnímu mapování (např. Hibernate nebo JPA), modul JDBC generalizující prostředky pro připojení k relační databázi, transakční modul pro efektivní správu transakcí všech instancí datových entit a další moduly, např. pro mapování XML objektů na entitní třídy.
- **Web** – Tato vrstva se skládá z modulů Web-Servlet a Web-Portlet. Web-Servlet poskytuje implementaci architektury Spring MVC, která zajišťuje oddělení logiky aplikace od datového modelu a zpracování HTTP požadavků, přičemž neomezuje použití dalších funkcí Spring frameworku.
- **Test** – Tato vrstva implementuje modul pro testování Spring komponent pomocí testovacích frameworků určených pro programovací jazyk Java. Modul podporuje např. frameworky JUnit či TestNG. Testovací vrstva rovněž podporuje tzv. mock integrační testování, které zajišťuje izolaci objektů vytvořených pro testovací účely od produkčních dat.
- **AOP a Instrumentation** – Vrstva AOP zajišťuje podporu aspektově-orientovaného programování, které přináší lepší modularizaci součástí vyvíjené aplikace. Modul Instrumentation podporuje možnost přidání bajtového kódu do již zkompilovaných tříd projektu vyvíjeného ve Spring frameworku.

Architektura Spring frameworku je zobrazena na následujícím obrázku:



Obrázek 12 – Architektura Spring frameworku [43]

5.2.2 Spring Boot

Spring Boot je oblíbená nadstavba Spring Frameworku. Spring Boot rozšiřuje Spring o počáteční nastavení přinášející konkretizovaný přístup při vývoji aplikací, který je jednoduchý, osvědčený a vyhovuje většině požadavků na webové aplikace. Spring Boot ve výchozím nastavení obsahuje integrovaný Servlet server, který se automaticky konfiguruje a spouští při spuštění projektu, čímž zjednodušuje práci programátorovi. Spring Boot je ideálním řešením pro webové aplikace, jejichž vývoj nevyžaduje větší flexibilitu. Výsledný kód je jednodušší a snadnější na pochopení. [23]

5.3 Microsoft SQL Server Express 2022

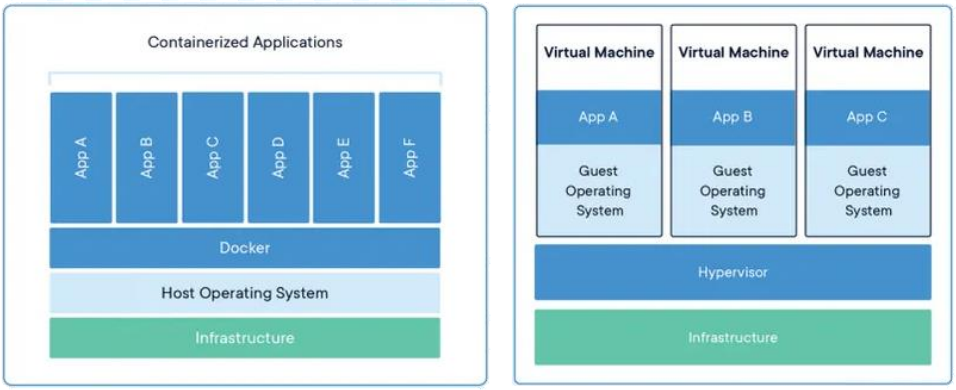
Microsoft SQL Server je relační a analytický databázový systém. MSSQL Server nabízí svou vlastní implementaci jazyka SQL s názvem T-SQL (Transact-SQL). Tento jazyk rozšiřuje SQL o možnost tvorby procedur a lokálních funkcí uložených na straně serveru. T-SQL dále poskytuje různé matematické funkce či metody sloužící pro práci s řetězci, nebo s geografickými či geometrickými daty. Společnost Microsoft nabízí širokou škálu edic tohoto systému, které cílí na konkrétní typy uživatelů – od velkých organizací, přes středně velké podniky až po malé firmy či jednotlivce. Edice SQL Server Express je dostupná zdarma, ostatní verze jsou zpoplatněny. Bezplatná verze však dokáže využít pouze jedno jádro procesoru a 1410 MB operační paměti pro buffer na každou instanci SQL serveru. Maximální velikost databáze u Express verze je 10 GB. Tato omezení však nejsou pro vývoj informačního systému pro správu soutěžních výsledků prozatím limitující. [24; 25]

5.4 Docker

Docker je open-source software, který umožňuje vývojářům nasazovat aplikace odděleně od ostatních v tzv. kontejneru. Tento kontejner je pak možné spouštět bez ohledu na hostitelský operační systém. Jedná se tedy vylepšenou formu virtualizace. První verze Dockeru vznikla v roce 2013. Docker je vyvinutý v programovacím jazyku GO. Tento programovací jazyk zkonstruovala společnost Google. Jazyk podporuje vývoj aplikací prostřednictvím různých programovacích paradigmat. Docker je podporován operačními systém Linux, Windows či macOS. [26; 27]

Klasická virtualizace spočívá plné virtualizaci stroje, který běží na virtuálním HW a je poháněný hostitelským OS. Výsledný virtuální stroj se chová jako plnohodnotný počítač. Velkou nevýhodou tohoto přístupu však jsou vysoké náklady na režii při virtualizaci hardwaru. Plně virtualizovaný stroj může spotřebovat až 20 % výkonu serveru. [27]

Kontejnerový přístup je odlišný. Není virtualizován celý stroj, avšak pouze jádro operačního systému. Tím je zajištěna izolace aplikací při virtualizaci za zlomek výpočetního výkonu a při menší velikosti souboru. Všechny kontejnery běží spolu na jednom OS a sdílejí veškeré HW prostředky, nebo jiné zdroje. Konfigurace aplikace pro běh v Dockeru začíná



Obrázek 13 – Porovnání plné a kontejnerové virtualizace [44]

vytvořením tzv. image. To je soubor, který obsahuje instalační obraz se základním nastavením aplikace pro běh v prostředí Dockeru. Je možné vytvářet si vlastní image, avšak Docker provozuje veřejný online registr Docker Hub, kam nahrávají oficiální obrazy aplikací i jejich vývojáři. Sestavená image Dockeru je určena pouze pro čtení, její konfiguraci je možné provádět před sestavením v souboru *.Dockerfile*. Jedna image může být společná pro několik kontejnerů. Pro konfiguraci vícekontejnerové aplikace slouží *docker–compose*, což je soubor YAML, který mimo konfigurace dílčích obrazů specifikuje i komunikaci mezi jednotlivými kontejnery a umožňuje izolovat data kontejnerů do dílčích svazků (volumes). [28]

5.5 OpenAPI a Swagger

OpenAPI je specifikace pro popis, testování a vizualizaci RESTful webových služeb. Specifikace není přímo závislá na programovacím jazyku. Výsledný dokumentační formát je jednotný a umožňuje strojové zpracování, přičemž zůstává jednoduše čitelný i pro programátora. API specifikace je důležitou součástí při tvorbě aplikační dokumentace. První verzi OpenAPI specifikace, tehdy ještě pod názvem Swagger API, začal vytvářet v roce 2009 Tony Tam. Ten navrhl jednoduché zobrazení služeb REST API pomocí formátu JSON. Ve Swagger Api 1.2, která byla vydána v roce 2014, došlo k oddělení specifikace od implementace. O rok později se původní Swagger API stalo součástí firmy SmartBear Software. V tom roce byla rovněž vydána verze 2.0, která byla rozdělena na standard OpenAPI a implementaci, sadu nástrojů Swagger. Pojmy Swagger a OpenAPI se však často nesprávně používají jako synonyma. Uživatelské rozhraní Swagger využívá html, css a Javascript pro automatické generování interaktivní dokumentace, která vývojářům umožní jednoduše vyzkoušet volání požadavků pro REST API ve webovém prohlížeči. [29; 30]

5.6 Mapy.cz REST API

Mapy.cz REST API je online služba, kterou provozuje společnost Seznam.cz. Rest API nabízí 3 skupiny funkcí pro práci se zeměpisnými souřadnicemi: [31]

- **Mapové dlaždice** – Mapové dlaždice poskytují funkce pro snadné vytvoření interaktivní mapy.
- **Geokódování** – Tato skupina funkcí nabízí prostředky pro dopředné i reverzní geokódování. Geokódování se používá při vyhledávání podrobných informací o adresách na základě textového dotazu. Geokódování je možné použít např. při našeptávání adres vyhledávací komponenty (pro našeptávání nabízí Mapy.cz API speciální funkci geokódování, která zohledňuje i neúplné textové dotazy). Reverzní geokódování se používá při potřebě určení adresy, případně textové specifikace územního celku na základě souřadnic GPS.
- **Plánování** – Tato část nabízí funkce určené pro plánování tras v mapě.

REST API Mapy.cz je podrobně dokumentováno, pro vývojáře nabízí i ukázky použití jednotlivých funkcí. Jednotlivé REST požadavky jsou rovněž podrobně dokumentovány pomocí specifikace OpenAPI. Použití API je zpoplatněno, avšak každý měsíc nabízí uživateli zdarma 250 000 kreditů. Pro volání jednotlivých funkcí jsou stanoveny ceníky v kreditech za

jedno volání. Pro využívání funkcí REST API je nutná registrace. Poté si musí vývojář nechat vygenerovat unikátní API klíč, jímž se bude při REST požadavcích identifikovat. [31]

V praktické části této diplomové práce se Mapy.cz REST API využívá pro potřeby geolokace. Při vkládání informací o adresách jsou s pomocí Mapy.cz REST API automaticky dohledány informace o GPS souřadnicích. Těchto dodatečných informací bude využito ve front-endové části k zobrazení mapy ročníků závodů.

5.7 React

React je open-source knihovna pro JavaScript, která se používá ke tvorbě moderních webových stránek. V architektuře MVC představuje React vrstvu View. Slouží tedy k transformaci dat do vhodné vizuální interaktivní formy, s níž pracuje samotný uživatel. React představila v roce 2013 společnost Facebook (Meta), která jej využívá pro vývoj stejnojmenné sociální sítě. React je velmi populární, v roce 2022 se stal v anketě *StackOverflow Developer Survey* 2. nejoblíbenější webovou technologií vývojářů těsně za *Node.js* [32; 33]

Základní myšlenkou Reactu je uspořádání fragmentů dynamického HTML kódu do znovupoužitelných komponent. Každá komponenta má vstupní vlastnosti, tzv. props, a zapouzdřuje vnitřní proměnné reprezentující stavy dílčích html prvků, které jsou součástí této komponenty. React je založen na deklarativním programování. Programátor tak nemusí řešit, jakým způsobem se budou komponenty překreslovat. Stačí pouze změnit její vnitřní stav a o překreslení se postará knihovna automaticky. Překreslení se provede pouze tehdy, změní-li se po nastavení vnitřního stavu jeho skutečná hodnota, což je velkou výhodou. Pro práci se stavy poskytuje React tzv. Hooks. Hooky jsou funkce nahrazující nutnost použití tříd, což ve vrstvě View architektury MVC přináší několik výhod. Takový kód nabízí větší flexibilitu a podporuje znovupoužitelnost v jiných komponentách. [32]

Knihovna React vytváří tzv. Single Page aplikace (SPA). Takový web se podobá desktopové aplikaci. Po přístupu na adresu URL je uživateli načtena celá aplikace, tudíž není potřeba opětovně načítat stránky ze serveru. Web je uživatelsky přívětivější a může po načtení fungovat i offline (pokud není potřeba přistupovat k REST-API). React v kombinaci s dalšími knihovnami podporuje i další způsoby vykreslování, např. server-side rendering, nebo rychlé vykreslování statických webových stránek. [32; 34]

React se používá v kombinaci s běhovým prostředím pro JavaScript Node.js. Toto běhové prostředí umožňuje spouštět kód JavaScript mimo webový prohlížeč. Node.js pohání V8 JS

engine, který se používá rovněž pro jádra webových prohlížečů Chromium či Google Chrome. Primární účel Node.js je pro tvorbu serverové části webové aplikace. Architektura Node.js umožňuje zpracovávat webové požadavky asynchronně, to je rozdíl např. oproti PHP. Asynchronní zpracování serverových požadavků přináší lepší škálovatelnost. Server tak zvládne obsloužit více klientů současně bez zvýšení latence. V Node.js lze vyvinout kompletní back-ednovou část webu. Mnohem častěji se však používá pouze pro jednodušší převod požadavků mezi front-endovou částí a REST API. To je rovněž případ, který je využit v praktické části této diplomové práce. Běhové prostředí Node.js podporuje rozšiřování funkcionalit o další moduly. K tomuto účelu je v běhovém prostředí integrován balíčkovací systém NPM, který je ovládán z příkazového řádku. Jednoduchým způsobem tak lze běhové prostředí rozšířit např. o knihovnu React či další komponenty. Balíčkovací systém podporuje instalaci balíčků z lokálního úložiště a z veřejného registru npmjs. [35]

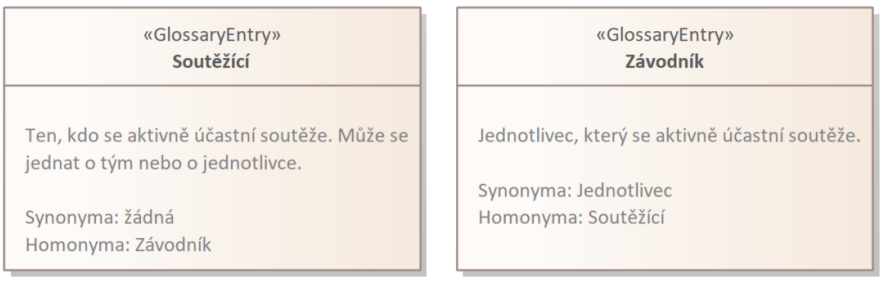
6 NÁVRH A IMPLEMENTACE ŘEŠENÍ

V této kapitole bude nejprve blíže specifikován návrh nově vyvíjené webové aplikace. Bude popsáno, jakým způsobem probíhalo hledání analytických tříd a budou představeny samotné modely těchto tříd. Další část kapitoly bude zasvěcena implementaci aplikace. Nejprve bude popsána struktura back-endové části SW. Budou představeny nejdůležitější třídy včetně vysvětlení jejich funkčnosti. Kapitola přiblíží zabezpečení aplikace proti zneužití neoprávněnými osobami. Závěr kapitoly se bude věnovat front-endové části. Bude popsán vývoj nejdůležitějších komponent a stručně bude představen i vzhled aplikace.

6.1 Návrh modelu tříd

Po dokončení specifikace požadavků a zhotovení diagramů případů užití bylo nutné nalézt vhodné třídy reprezentující datové entity. Metodika Unified Process navrhuje osvědčený postup v podobě analýzy podstatných jmen a sloves, který dokáže odhalit i skryté třídy, jež do té doby nemusely být odhaleny. Aby však byla tato technika účinná, je vhodné u složitějších návrhů zavést slovníček pojmů a ten při analýze respektovat. To zamezí vzniku nepravých a nežádoucích tříd. Pojem v takovém slovníku by měl být vždy přesně vysvětlen a měla by být uvedena i synonyma, zkratky a homonyma vztahující se k tomuto pojmu. [36]

Příklad správně dokumentovaného pojmu ve slovníčku uvádí následující obrázek:



Obrázek 14 – Správně uvedené výrazy ve slovníčku pojmů dle metodiky UP

Analýza podstatných jmen a sloves spočívá ve shromáždění co největšího počtu informací o systému. Zdrojem mohou být systémové požadavky, případy užití, slovníček pojmů, ale i další dokumenty, které s problematikou souvisejí. Následuje analýza získaných informací. Jsou vypsány následující slovní druhy: [36]

- Podstatná jména,
- spojení podstatných jmen,
- slovesa,
- slovesné fráze.

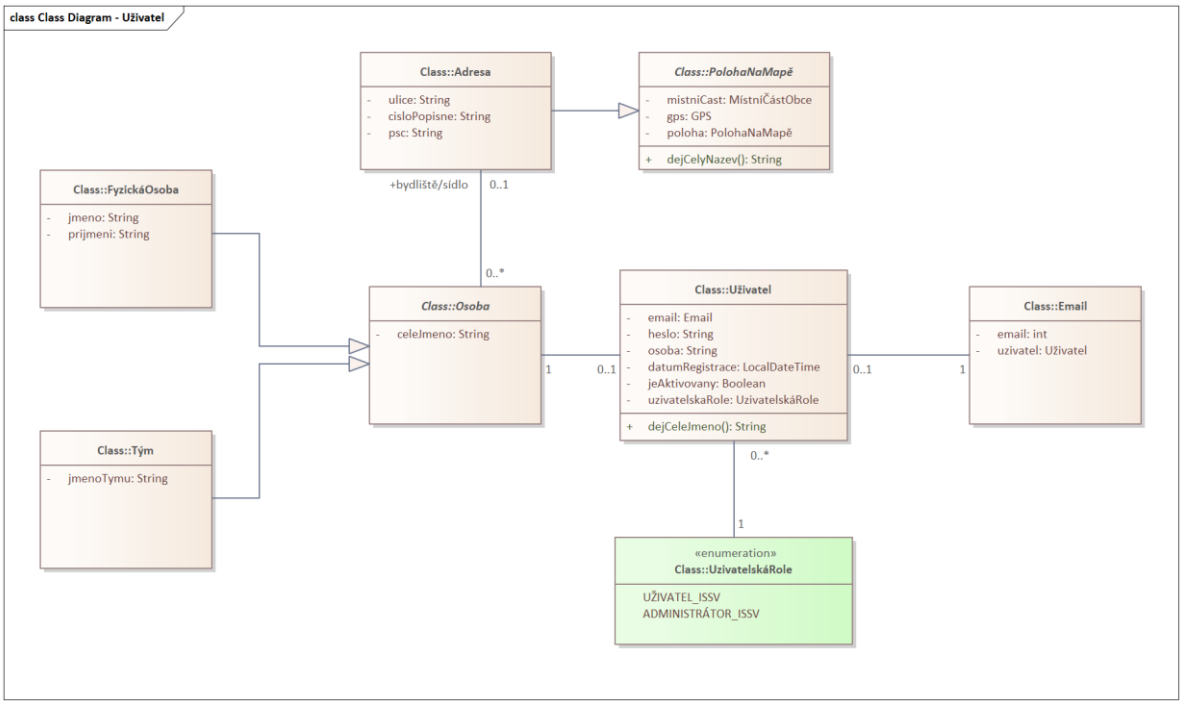
Z kategorizovaných slovních druhů jsou dále vybírány vhodné názvy pro třídy a jejich atributy či operace, přičemž platí jednoduché pravidlo: Podstatná jména a spojení podstatných jmen vyjadřují třídy a jejich atributy, zatímco slovesa a slovesné fráze jsou vyjádřením operací tříd a odpovědností. [36]

Při návrhu systému pro zpracování soutěžních výsledků bylo postupováno přesně podle výše popsané techniky. Slovníček pojmů byl vypracován v programu Enterprise Architect. Analýza podstatných jmen a sloves probíhala s využitím tabulkového procesoru MS Excel. Oba tyto výstupy jsou součástí přílohy diplomové práce.

Z vytvořené analýzy podstatných jmen a sloves bylo nakonec po několika úpravách navrženo 33 tříd a 6 výčtových typů reprezentujících datový model aplikace. Původní návrhy počítaly s ještě vyšším počtem tříd. Z časových důvodů však budou některé funkcionality realizovány až v dalších etapách vývoje. Zjednodušený model však byl navrhován s ohledem na budoucí rozšíření. Výsledný class model byl pro větší přehlednost rozdělen do 15 diagramů reprezentujících jednotlivé komponenty aplikace. Těm zásadním budou věnovány následující podkapitoly.

6.1.1 Uživatel

Tento diagram znázorňuje objekty pro uchovávání informací o uživatelích a osobách v systému. Z diagramu je patrné, že každá osoba může mít vytvořený maximálně jeden

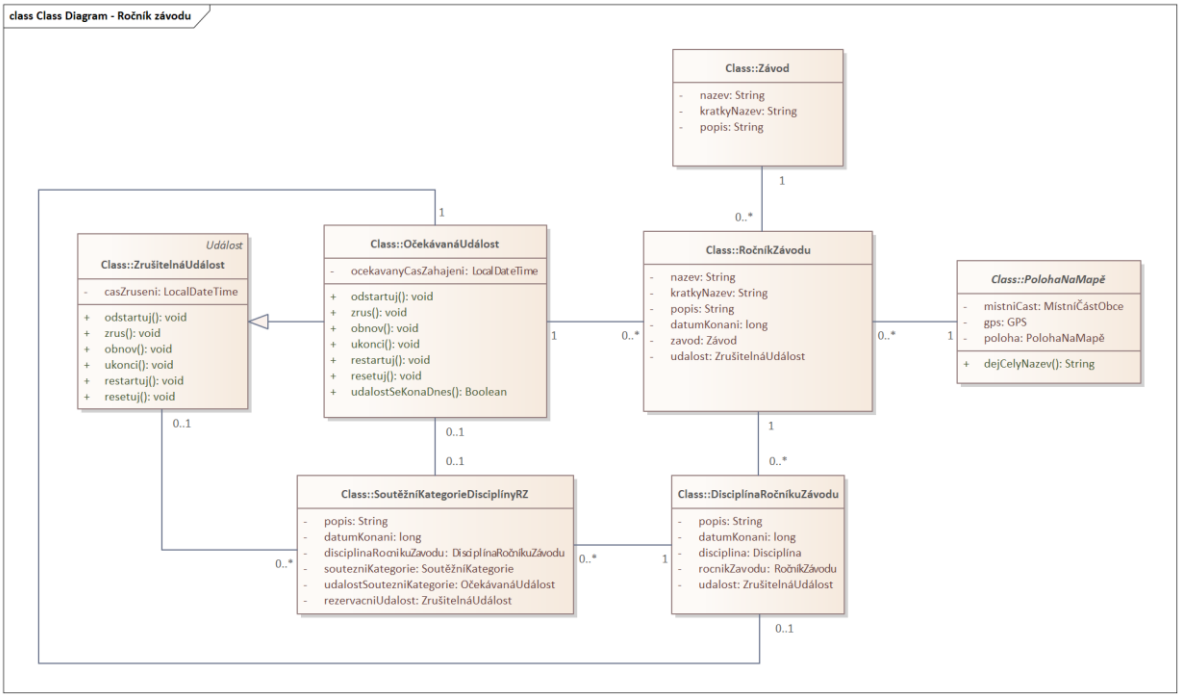


Obrázek 15 – Class diagram – Uživatelé

uživatelský účet vedený pod unikátním e-mailem. Každý uživatelský účet má přidělenou jednu uživatelskou roli v systému, buď roli běžného uživatele nebo administrátora systému. Osoba může mít vyplněné údaje o bydlišti resp. sídle. Mohlo by se zdát, že tabulka osob a uživatelů by se dala pro zjednodušení sloučit. K tomuto kroku nebylo přistoupeno, protože by v takovém případě každá osoba, která chce na závodě soutěžit, musela mít vytvořený uživatelský účet, což není žádoucí. Osoba se dle pravidel PS může registrovat i v den konání ročníku závodu při prezenci, v takovém případě zadává údaje do systému příslušný rozhodčí. Pro zjednodušení práce s osobami byla použita abstrakce fyzické osoby a týmu do abstraktní třídy *Osoba*.

6.1.2 Ročník závodu

V diagramu tříd s názvem *Ročníky závodů* jsou zobrazeny vztahy mezi závodem, ročníkem závodu, disciplínami a soutěžními kategoriemi:



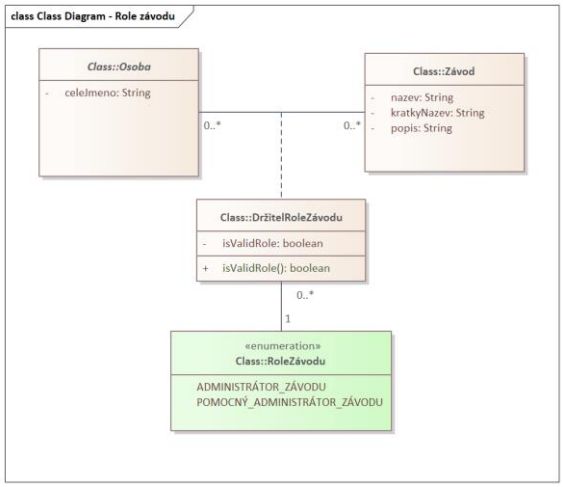
Obrázek 16 – Class diagram – Ročník závodu

Tento návrh opět vychází z pravidel požárního sportu. Jeden závod může mít několik ročníků závodu, přičemž jeden ročník závodu může mít více disciplín. Každá tato disciplína ročníku závodu má specifikovány soutěžní kategorie, v nichž se závodí. Kromě třídy Závod, která nespecifikuje konkrétní událost v čase a na konkrétním místě, mají ostatní třídy vazbu na událost. V diagramu se vyskytují dva typy událostí, *ZrušitelnáUdálost* a *OčekávanáUdálost*. Očekávaná událost má navíc atribut udávající očekávaný datum a čas začátku události. Tohoto atributu se využívá při platnosti rolí pracovníků ročníku závodu. Soutěžní kategorie má navíc ještě další událost typu *ZrušitelnáUdálost*. Ta reprezentuje rezervační událost pro konkrétní

soutěžní kategorii disciplíny ročníku závodu. Pokud je tato událost v průběhu, jsou povoleny online registrace do závodu. Třídy *Závod* a *RočníkZávodu*, které jsou v kompozičním vztahu, obsahují duplicitní atributy, což by se mohlo zdát jako chyba. Je to však záměrně. Často se stává, že větší soutěže mají své sponzory. Tito sponzoři se v průběhu let mění a s tím se i nepatrně odlišují názvy závodů aktuálních ročníků. Tyto názvy se používají např. při tisku soutěžních listin apod, kde musí být v názvech zahrnut i název sponzora. Zároveň má však každá soutěž oficiální název, bez sponzorů, který je v průběhu let neměnný. Vhodným řešením by se mohlo zdát třídu *Závod* úplně odstranit. K této třídě se však vztahují role administrátorů (pořadatelů) závodu, kteří jsou zpravidla stálí v průběhu let (s možností drobných změn oprávnění pro konkrétní osoby). Proto ani tento diagram nelze zjednodušit. Třídy rolí závodu popisuje následující podkapitola.

6.1.3 Role závodu

Kromě uživatelských rolí, které definují vztah mezi uživatelem a celým systémem ISSV, byly navrženy ještě role vztahující se ke třídám *Závod, RočníkZávodu, DisciplínaRočníkuZávodu,* a *SoutěžníKategorieDisciplínyRZ*. To jsou role administrátorů závodu, pracovníků ročníku závodu a soutěžících. Tyto role mohou mít omezenou platnost. Byla proto navržena asociační třída, která má funkci vracející logickou hodnotu, zda je role platná. Platnost role se může lišit v závislosti na typu role. Role závodu se nevztahují k uživateli, ale ke třídě *Osoba*. Je to opět z důvodu respektování pravidel PS. Mohl by nastat případ, kdy bude v závodě figurovat role, která nebude mít vytvořený uživatelský účet v systému. Taková osoba bude vykonávat svoji



Obrázek 17 – Class diagram – Role závodu

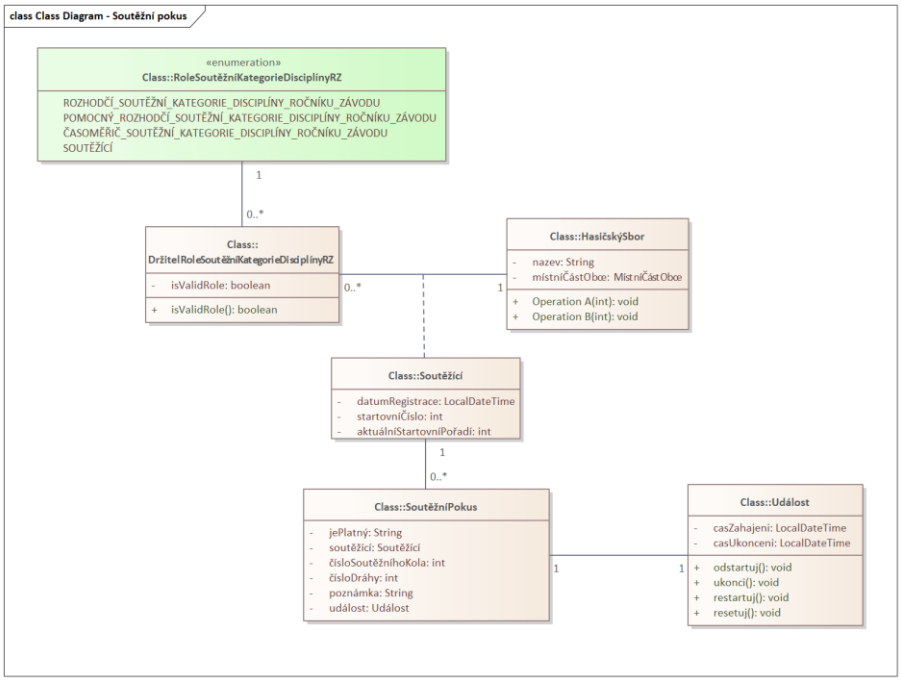
funkci během konání ročníku závodu, v systému ji však bude zastupovat nadřízená osoba. Pokud by přesto bylo nutné podřízenou osobu do systému z nějakého důvodu uvést, např. pro

výpis podrobných údajů o závodě pro potřeby pořadatele soutěže, nebylo by to možné. Podrobněji se hierarchii rolí bude věnovat jedna z podkapitol implementace řešení. Do sekce ISSV pro pracovníky závodů budou moci přistupovat osoby při splnění 3 podmínek – mají vytvořený a aktivní uživatelský účet v systému ISSV, mají přidělenou závodní roli pro daný ročník závodu a tato role je platná. Diagram uvádí model tříd pro role vztahující se k entitě *Závod*.

Stejným způsobem jsou navrženy i role pro ročník závodu, disciplíny a soutěžní kategorie disciplíny ročníku závodu. Diagramy těchto tříd jsou součástí přílohy této diplomové práce a nebudou zde z důvodu podobnosti a rozsáhlosti uváděny. Třídy se liší především v implementaci.

6.1.4 Soutěžní pokus

Tento diagram zobrazuje návrh tříd pro reprezentaci dat o soutěžících a jejich soutěžních výkonech. Soutěžící je konkrétním držitelem role soutěžní kategorie disciplíny ročníku závodu.



Obrázek 18 – Class diagram – Soutěžní pokus

Pro soutěžícího musí být dále uchovávány další atributy. Z tohoto důvodu byla tato entita navržena jako rozšiřující třída. Důležitým atributem je startovní číslo, což je unikátní číslo v příslušné soutěžní kategorii disciplíny ročníku závodu, kterým se soutěžící identifikuje. K soutěžícím se vztahují soutěžní pokusy – naměřené sportovní výkony. Jeden soutěžící může v příslušné soutěžní kategorii disciplíny ročníku závodu závodit vícekrát. Dle pravidel PS se

tak může stát ve dvou případech – první z nich je, že se soutěží ve více soutěžních kolech, druhý případ nastane, pokud je soutěžícímu uznán protest proti rozhodčím a on může svůj pokus opakovat.

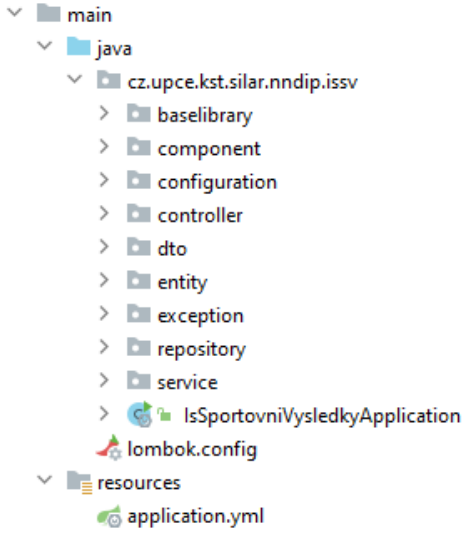
Další entity jsou součástí přílohy této diplomové práce. Jedná se především třídy pro ukládání číselníků.

6.2 Implementace back-endové části

Vývoj back-endové části probíhal v programovacím jazyku Java s využitím frameworku Spring Boot. Jako databázový server byla z důvodu zajištění kompatibility se SW Firesport Timekeeper vybrána databáze Microsoft SQL Server Express. Z důvodu umožnění nasazení aplikace na cloudové platformy byla databáze kontejnerizována. Veškerá konfigurace Dockeru je uložena v souboru *docker-compose.yaml*, jenž se nachází v kořenové složce adresáře projektu. Struktura projektu a důležité části zdrojového kódu budou bude popsána v následujících částech této podkapitoly.

6.2.1 Struktura projektu

Hlavní zdrojový kód back-endové části aplikace, nacházející se v package *main*, je rozčleněn do 9 hlavních balíčků s příslušnými třídami:



Obrázek 19 – Struktura projektu back-endové části aplikace

Názvy balíčků byly navrhovány s ohledem na dodržení zásad architektury MVC. Soubor *lombok.config* obsahuje konfiguraci knihovny Lombok. Ta byla použita pro usnadnění vývoje a urychlení rutinní práce při psaní getterů a setterů atributů třídy. Knihovna je dokáže na základě názvu atributu vygenerovat automaticky. Takový zdrojový kód je přehlednější a lépe vynikne

jeho skutečná logika, která není skryta za implementačními detaily. V souboru *application.yml* se nachází různé konfigurační detaily pro framework Spring Boot, především pro přístup k databázi, a pro samotnou aplikaci ISSV. Je možné v souboru konfigurovat doby expirací pro uživatelské a JWT tokeny (bude vysvětleno v další části kapitoly) a jsou zde uloženy přístupové údaje pro e-mailového klienta, jenž se v aplikaci používá k zasílání informačních zpráv uživatelům. Je zde uložen rovněž API klíč pro přístup k Mapy.cz REST API.

Jednotlivé balíčky hlavního zdrojového kódu mají následující význam:

- **baselibrary** – V tomto balíčku se nachází třída *StringFunctions* s metodami pro práci s textovými řetězci, které se v kódu opakovaly.
- **component** – Obsahuje třídy pro hromadné ošetřování výjimek aplikace, pro převod mezi SQL datovým typem a vlastní datovou třídou či komponenty pro filtrování HTTP požadavků na základě autentizace pomocí tokenu JWT.
- **configuration** – Balíček obsahuje konfigurační třídy, např. pro nastavení e-mailového připojení nebo konfiguraci zabezpečení.
- **controller** – Obsahuje třídy pro obsluhu mapování a konfiguraci zabezpečení REST požadavků.
- **dto** – Balíček zahrnuje třídy, použité pro převod dat mezi back-endem a front-endem aplikace.
- **entity** – Zahrnuje všechny datové třídy v back-endové části aplikace. Struktury těchto tříd jsou odvozeny z class modelu UML.
- **exception** – Obsahuje vlastní business výjimky.
- **repository** – Obsahuje rozhraní pro objekty DAO, které se používají pro mapování objektů z SQL databáze na entity objekty. Všechna tato rozhraní rozšiřují interface *JpaRepository*, které používá JPA standard pro objektově-relační mapování.
- **service** – Balíček obsahující rozhraní a jejich implementace sloužící pro práci s entitami. V servisních službách je implementována stěžejní logika aplikace. Hlavní třídy proto budou vysvětleny podrobněji v následující části této diplomové práce.

6.2.2 Hierarchie rolí

V aplikaci musela být z důvodu stromového uspořádání rolí pracovníků soutěže, která je definována pravidly PS, implementována vlastní hierarchie rolí. Spring Boot nabízí třídu

RoleHierarchy pro jednoduchou definici hierarchie uživatelských rolí. Veškerá tato hierarchie je založena na jednoduché definici v textovém řetězci pomocí názvů rolí a znamének „>“ a „<“. Dá se tak například jednoduše definovat hierarchie „ROLE_ADMIN > ROLE_USER“, která udává, že každý administrátor má rovněž roli uživatele. Při autorizaci pak stačí jednoduše kontrolovat nejnižší možnou roli, která je pro přístup k dané části aplikace vyžadována. K tomuto účelu slouží ve Spring Boot třída WebSecurity, která obsahuje metody *hasRole*, *hasAnyRole* a funkce pro povolení či zakázání přístupu k dané službě či controlleru. Podmínkou pro správné fungování autorizace využívající hierarchii rolí Spring Boot je implementace vlastní *AuthenticationProvider* Bean využívající *UserService*, která musí implementovat rozhraní *UserDetailsService*. To musí pracovat s entitou uživatele implementující interface *UserDetails*, které obsahuje metodu vracející všechny názvy rolí přidělených tomuto uživateli.

Hierarchie rolí ve Spring Boot je užitečná funkcionalita, pro potřeby této aplikace ale není dostatečná. Při uživatelských rolích vztahujících se k závodu či k jeho ročníkům, nestačí pouze kontrola, zda uživatel vlastní danou uživatelskou roli s konkrétním názvem. Je potřeba vzít v úvahu rovněž danou entitu, k níž se tato role vztahuje. Dále je nutné zkontrolovat, zda je tato role aktuálně platná. Platnost role se odvíjí od průběhu závodu a liší se na základě konkrétní role. Platnosti těchto rolí vychází z pravidel PS. Pro tyto účely bylo vytvořeno 5 výčtových typů s názvy uživatelských rolí:

- **RaceRole** – Uživatelské role vztahující se k závodu
- **RaceYearRole** – Uživatelské role vztahující se k ročníku závodu
- **RaceYearDisciplineRole** – Uživatelské role vztahující se k disciplíně ročníku závodu
- **RaceYearCategoryRole** – Uživatelské role vztahující se k soutěžní kategorii disciplíny ročníku závodu
- **UserRole** – Uživatelské role vztahující se k systému ISSV

V hierarchii rolí účastníků soutěže dle pravidel PS se vyskytuje vícenásobná dědičnost. Java však víceásobnou dědičnost nepodporuje. K tomuto účelu proto muselo být navrženo rozhraní s názvem *Role*, které všechny výše uvedené výčtové typy implementují. Toto rozhraní předepisuje jednoduchou funkci *includes* vracející logickou hodnotu, zda je daná role součástí role předané v parametru této funkce. Ve vlastních třídách výčtových typů jsou pak všechny implementace rozhraní *Role* provedeny obdobně – každá enum třída má privátní HashSet atribut, ve kterém jsou uloženy všechny zahrnuté role pro konkrétní hodnotu výčtu. V metodě *includes* pak probíhá kontrola, zda je hledaná role součástí tohoto hashSetu. Hierarchie rolí jsou

definovány v příslušných enum třídách pomocí inicializátoru *static*. Implementace bude lépe pochopitelná z ukázky kódu:

```
private final Set<Role> includedRoles = new HashSet<>();
@Getter
private final String name;


static {
    RACE_YEAR_CHIEF.includedRoles
            .addAll(List.of(
                    RaceYearRole.RACE_YEAR_REFEREE,
                    RaceYearDisciplineRole.DISCIPLINE_TIMEKEEPER));
    RACE_YEAR_REFEREE.includedRoles
            .add(RaceYearRole.RACE_YEAR_REFEREE_HELPER);
    RACE_YEAR_REFEREE_HELPER.includedRoles
            .add(RaceYearDisciplineRole.DISCIPLINE_REFEREE);
}

 st52165
@Override
public boolean includes(Role role) {
    return this.equals(role) || includedRoles.stream()
            .anyMatch(includedRole -> includedRole.includes(role));
}
```

Obrázek 20 – Ukázka implementace hierarchie rolí

Z ukázky je evidentní, že role *RACE_YEAR_CHIEF* má zároveň oprávnění *RACE_YEAR_REFEREE* a *DISCIPLINE_TIMEKEEPER*. Jinými slovy, velitel ročníku závodu je nadřazen hlavnímu rozhodčímu a časoměřičům disciplíny a přebírá od nich všechna oprávnění. Analogicky je tato statická definice provedena pro všechny výčty tříd enum, které jsou vyjmenované výše.

6.2.3 RoleServiceImpl

Tato služba implementuje metody pro kontrolu požadované role přihlášeného uživatele na základě id příslušné entity, ke které se oprávnění vztahují. Nejprve je provedena kontrola UserRole definující roli ve vztahu uživatele k systému. Tuto roli má každý uživatel přidělen v parametru *role* příslušné entity User. Role osoby vztahující se k závodům a jeho ročníkům, disciplínám a soutěžním kategoriím jsou uloženy ve 4 tabulkách, přičemž probíhá kontrola hierarchicky od nejnižší úrovně po nejvyšší. Nenajde-li se požadovaná úroveň oprávnění v hledané tabulce, přistupuje se ke hledání v tabulce s oprávněními vyšší úrovně. Zároveň se testuje shoda nejen na konkrétní roli výčtového typu, ale zároveň na všechny zahrnuté role definované ve statickém inicializátoru příslušné enum třídy pomocí metody *includes*. Pokud je příslušná role nalezena, je otestována její platnost, která se odvíjí od průběhu závodu pro

konkrétní roli. Pokud je i tato podmínka splněna, je vrácena logická hodnota *true*. Příklad funkce pro kontrolu požadované role v ročníku závodu uvádí následující obrázek:

```
1 usage   st52165
private boolean hasAnyRoleByRaceYearId(User user, long raceYearId, Role... roles) {
    final Set<PersonRaceYearRole> raceYearRoles = raceYearRoleRepository
            .getAllByPerson_User_IdAndRaceYear_Id(user.getId(), raceYearId);

    if (containsWantedValidRoles(user, raceYearRoles, roles)) {
        return true;
    }

    RaceYear foundRaceYear = raceYearService.getById(raceYearId);
    return hasAnyRoleByRaceId(foundRaceYear.getRace().getId(), roles);
}
```

Obrázek 21 – Ukázka metody pro kontrolu požadované role

Metoda *containsWantedValidRoles* provádí výše popsanou kontrolu. Pokud role není nalezena, přistupuje se ke hledání ve vyšší úrovni, v tomto případě ke hledání uživatelských rolí nadřazeného závodu.

6.2.4 PersonRaceYearRoleImpl

Do této entitní třídy jsou ukládána veškerá oprávnění vztahující se k roli ročníku závodu. Tato třída je deklarována jako abstraktní. Její potomci představují konkrétní role v konkrétním ročníku závodu pro konkrétní osoby a s definovanou platností. Tato třída má 3 potomky, *MainRaceYearReferee*, *MainRaceYearRefereeHelper* a *RaceYearChief*. Třída je anotována pomocí @Inheritance zajišťující mapování dědičnosti do databáze pomocí tzv. diskriminátoru. To je sloupec v tabulce, definující konkrétní třídu, jejíž instance se vytvoří při ORM. Diskriminátorem pro tabulku PersonRaceRoleImpl je název role závodu definovaný v enum RaceRole. Při mapování třídy *PersonRaceYearRoleImpl* do databáze se použije společná tabulka pro všechny potomky této třídy.

Entitní třídy, které slouží k uchovávání rolí pro závod, ročník závodu, disciplínu ročníku závodu či soutěžní kategorii disciplíny ročníku závodu, implementují rozhraní *PersonRaceRole*, které předepisuje metodu *isValidRole* společně s dalšími funkcemi, jež se používají ke kontrole oprávnění. Abstrakní metoda *isValidRole* defaultně vrací hodnotu true, pokud příslušný ročník závodu, k němuž se role vztahuje, právě probíhá. U některých potomků je tato metoda předefinována tak, aby platnost role byla v souladu s pravidly PS.

Definice tohoto rozhraní je uvedena na následujícím obrázku:

```
12 usages  20 implementations  st52165
public interface PersonRaceRole {
    4 implementations  st52165
    Person getPerson();

    4 implementations  st52165
    Race getRace();

    4 implementations  st52165
    Role getRole();

    3 usages  7 implementations  st52165
    boolean isValidRole();
}
```

Obrázek 22 – Ukázka rozhraní pro kontrolu platnosti rolí závodu

6.2.5 Bezpečnost aplikace

Registrace na server probíhá dvoufázově. Po vyplnění registračního formuláře musí uživatel registraci potvrdit pomocí odkazu s jedinečným tokenem, který se při registraci vygeneruje a uloží do databáze. Tento odkaz se v případě úspěchu při vyplnění formuláře odešle na uvedenou e-mailovou adresu. Pokud se uživatel tímto tokenem v omezeném čase prokáže, je jeho registrace potvrzena a může se přihlásit. Po uplynutí omezené doby, během níž nedojde k aktivaci účtu, je jeho e-mailová adresa uvolněna pro další registrace. Registračním tokenem je ověřena identita uživatele. Hesla do databáze se neukládají v raw podobě, ale jsou zašifrována pomocí funkce silného hashování BCrypt. K tomuto účelu je využita třída *BCryptPasswordEncoder*, která je součástí frameworku Spring Boot.

Pro zajištění bezpečnosti aplikace byla použita autentizace pomocí JWT tokenu. Tento token nahrazuje všechny informace o klientovi, který se snaží k serveru přistupovat. Pokud se uživatel úspěšně autentizuje, server mu vygeneruje token, který je digitálně podepsán za pomoci asymetrické kryptografie. Tímto tokenem se uživatel dále prokazuje při přístupu k zabezpečené části serveru. Token má pouze omezenou dobu platnosti, poté je nutné se znovu autentizovat. Generování JWT tokenů zajišťuje služba *JwtServiceImpl*.

Pro autorizaci k částem aplikace na základě role je použita Spring Boot anotace *@PreAuthorize* v kombinaci s třídou *RoleServiceImpl*, která implementuje rozhraní

RoleService. Tato anotace je uvedena nad potřebnými metodami controllerů mapujících požadavky HTTP.

Příklad zabezpečení metody controlleru uvádí následující obrázek:

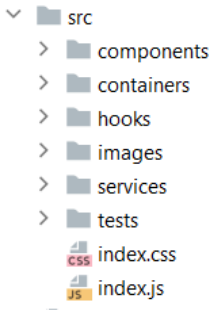
```
st52165
@DeleteMapping("/years/{id}")
@PreAuthorize("@RoleService.hasAnyRoleByRaceYearId(#id, @RaceRole.ADMIN)")
public ResponseEntity<String> delete(@PathVariable long id) {
    return ResponseEntity.ok(raceYearService.deleteByIdAndGetResponse(id));
}
```

Obrázek 23 – Příklad autorizace na základě HTTP požadavku

Pokud klient vykoná DELETE http požadavek s uvedeným mapováním, server před jeho provedením ověří logickou hodnotu, kterou vrátí výsledek metody *hasAnyRoleByRaceYearId*. Této metodě je předán parametr id z proměnné cesty http requestu spolu s rolí administrátora závodu. Příslušná metoda třídy *RoleService* musí vrátit true, jinak bude vrácena chyba http požadavku s kódem 401 značící neoprávněný přístup. V opačném případě se server pokusí vykonat metodu služby *RaceYearService*, která vymaže ročník závodu dle ID předaného z proměnné cesty. Jinými slovy, vymazat příslušný ročník závodu může pouze osoba, která má platnou roli administrátora závodu, pod nějž se tento ročník závodu vztahuje. Služba RoleService je blíže popsána v podkapitole 6.2.3.

6.3 Implementace front-endové části

Vývoj front-endové části probíhal v knihovně React s využitím javascriptového běhového prostředí Node.js. Struktura projektu je následující:



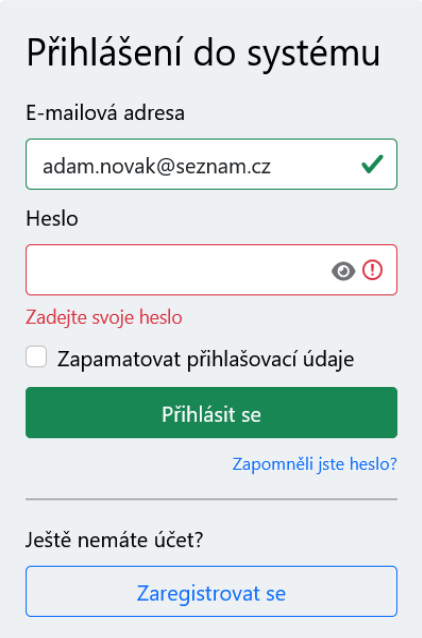
Obrázek 24 – Struktura front-endové části projektu

Ve složce components jsou uloženy veškeré funkční komponenty. Složka containers obsahuje hlavní definice veškerého směrování mezi vytvořenými komponentami. K tomuto účelu byl využit npm package *react-router-dom*. Složka obsahuje soubory s funkcemi, které zajišťují podmíněné vykreslování komponent na základě autentizace a autorizace uživatele dle příslušné HTTP cesty. Ve složce hooks jsou uloženy hooky, které se používají na více místech v kódu.

Ve složce images jsou uloženy statické obrázky pro html kód, složka *tests* je pro testy uživatelského rozhraní. Ve složce services jsou uloženy funkce, které se používají při komunikaci s back-endovou částí. Pro zjednodušení komunikace s REST API je využit npm package *axios*. Výhodou balíčku axios je, že automaticky transformuje data přijatá ze serveru do formátu JSON a umožňuje nastavit časový limit pro odezvu.

6.3.1 Design aplikace

Pro zjednodušení návrhu designu aplikace byla použita sada nástrojů Bootstrap. Bootstrap je .css a Javascript framework, který obsahuje předdefinované styly pro návrh běžně používaných komponent v moderních webech. Bootstrap styly podporují responzivní design. Framework má kvalitní dokumentaci, ve které nabízí i ukázky předpřipravených řešení.



Obrázek 25 – Ukázka designu přihlašovacího formuláře s využitím frameworku Bootstrap

6.3.2 Komponenty

Komponenty, které se ve front-endové části vyskytují vícekrát, jsou vyčleněny do samostatných funkcí. Jedná se především o formulářové prvky, kde se opakuje kód pro ošetření vstupů společně s popisky pro uživatele a css stylování. Komponenty byly navrhovány tak, aby do jisté míry některé atributy šly pozměnit, přičemž základní vzhled komponenty zůstane zachován. Aby upravitelné vstupní vlastnosti komponenty, tzv. *props*, měly svůj správný datový typ, byly v komponentách specifikovány tzv. PropTypes. Pro tento účel byl použit i stejnojmenný balíček

npm. Dokumentace atributů *props* zajistí, že v případě předání nesprávného parametru na vstupu běhové prostředí Node.js zahlásí chybu. Balíček *proptypes* podporuje rovněž specifikaci výchozích vlastností. Této možnosti je v kódu také využito. Při použití pak stačí definovat pouze vlastnosti, které je potřeba změnit, čímž se snižuje duplicita kódu. Příklad definice propTypes a defaultProps uvádí následující obrázek:

```
Email.propTypes = {
    name: PropTypes.string.isRequired,
    className: PropTypes.string,
    onChange: PropTypes.func
}

Email.defaultProps = {
    name: "email",
    className: "",
    onChange: null
}
```

Obrázek 26 – Ukázka specifikace vlastností React komponenty

7 TESTOVÁNÍ APLIKACE

Testování aplikace probíhalo ve 3 fázích. Testována byla převážně back-endová část aplikace, zejména části kódu týkající se autentizace a autorizace. Nejprve byly implemetovány jednotkové testy. K testování servisní vrstvy a vrstvy controlleru byly využity tzv. mocking testy. Pro účely mockování byl využit framework *Mockito*. Příslušné třídy nepracují s instancemi entit, ale pouze s jejím mockem, což je náhražka instance objektu jiným objektem. Mocking funguje na principu návrhového vzoru Zástupce (Proxy). Třída zastupujícího objektu Proxy implementuje stejné rozhraní jako testovaná třída. Po dokončení implementace komponenty byla nejprve její funkčnost ručně vyzkoušena za použití nástroje Swagger. Ve třetí části testovací fáze bylo přistoupeno k integračnímu testování důležitých částí aplikace.

Příklad jednoduchého jednotkového testu uvádí následující ukázka.

```
@Test
void shouldNotContainsAnyCycles() {
    var resultingStreamOfRoles : List<extends Role> = getStreamOfAllRoles()
            .toList();

    final var root : Role = Role.getRoot();

    assertDoesNotThrow(
            () -> {
                for (var roleToCheck : resultingStreamOfRoles) {
                    root.includes(roleToCheck);
                }
            }
    );
}
```

Obrázek 27 – Příklad jednotkového testu

Tato jednoduchá metoda testuje všechny role v systému procházením od kořenové role, v tomto případě od administrátora systému ISSV, až k listům a testuje, zda nenastane výjimka *StackOverflowException*, která by značila, že v hierarchii rolí existují cykly. Další jednotkové testy řeší např. aby každá role dědila uživatelskou roli USER_ISSV, nebo testuje, zda soutěžící nemá přiřazenu některou z rolí rozhodčího. Takové chování by bylo nežádoucí a tato chyba by mohla ohrozit bezpečnost aplikace.

Testy pracující výhradně s objekty DAO prostřednictvím příslušných *JpaRepository* rozhraní probíhají v transakcích. Na konci každého testu dojde k rollbacku (odvolání) příslušné transakce, tudíž veškeré změny, které vzniknou v databázi při běhu testu, nebudou uloženy. K tomuto účelu slouží ve frameworku Spring Boot anotace *@DataJpaTest* a *@AutoConfigureTestDatabase(replace = Replace.NONE)*. Tyto anotace se uvádějí nad příslušnou testovací třídou, případně testovací funkcí. I přesto je však pro testy využita

speciálně vyčleněná databáze a nepracuje se nad skutečnými daty. Schéma testovací databáze se vytváří nové na začátku každé session, tudíž obsahuje aktuální změny parametrů příslušných entit. Na konci každé session se schéma opět vymaže. Testovací databáze, stejně jako relační databáze pro vývoj, je kontejnerizována v Dockeru. Testovací databáze rovněž využívá Microsoft SQL Sever Express 2022.

Protože testy, které probíhají nad skutečnou databází, jsou časově náročné, je v některých případech jednotkového testování přistoupeno k mockování pomocí frameworku Mockito. Příklad takového testu je uveden na následujícím obrázku:

```
st52165 *
@Test
void RegisterCheckIsNotNull() {
    User user = UserDataFactory.buildUser();
    RegisterRequest<?> registerRequest = buildRegisterRequest();

    when(passwordEncoder.encode(Mockito.any(String.class)))
            .thenReturn(user.getPassword());
    when(personService.createNewFromRequestAndSave(Mockito.any(PersonRequest.class)))
            .thenReturn(user.getPerson());
    when(userEmailService.getOrCreateNewAndSave(Mockito.any(String.class)))
            .thenReturn(user.getUserEmail());
    when(userService.save(Mockito.any(User.class)))
            .thenReturn(user);

    RegistrationResponse registeredUser = authenticationService.register(registerRequest);

    verify(registrationTokenService, times( wantedNumberOfInvocations: 1))
            .generateToken(user);
    assertThat(registeredUser).isNotNull();
}
```

Obrázek 28 – Příklad testu s využitím mockování

Tento test provádí kontrolu, zda došlo k úspěšné registraci uživatele ze vstupního DTO. Registraci uživatele obstarává funkce *register* příslušné třídy *AuthenticationService*. Tato funkce interaguje s několika dalšími službami – *passwordEncoder*, *personService*, *userEmailService* a *userService*. Potřebné služby byly pro účely tohoto testu mockovány. K tomu slouží anotace *@Mock*, případně *@InjectMocks*, které se uvedou nad deklaracemi proměnných příslušných tříd. Anotace *@InjectMocks* po vytvoření instance příslušné mockované třídy automaticky vloží závislosti rovněž pro potřebné třídy označené anotací *@Mock*. Metoda *when* umožňuje jednoduše nakonfigurovat návratové chování pro potřebné instanční metody mockovaných tříd. K tomu slouží i funkce *thenReturn*, která po zavolání metody s příslušnými vstupními parametry navrátí „náhradní“ objekt uvedený ve vstupním parametru této funkce.

Cílem integračního testování je reálně vyzkoušet, zda jednotlivé komponenty mezi sebou komunikují dle očekávání. Na rozdíl od jednotkových testů, které testují funkčnost malých částí kódu, integrační testy ověřují funkčnost závislých komponent jako celku. Integrační testování proto probíhá nad reálnou testovací databází a se skutečnými komponenty. U integračního testování nelze použít mocking, který některé části systému nahrazuje. Výsledky takového testu

by neodpovídaly reálné funkčnosti komponent. V aplikaci byly implementovány integrační testy pro komponenty související s autentizací uživatele. Na následujícím obrázku je uveden příklad integračního testu ověřující přihlášení uživatele:

V ukázce kódu je nejprve prezentována část testovací metody pro přihlášení, ve které se

```
response = assertDoesNotThrow(this::postLoginRequest);
checkHttpResponseHasBodyAndIsOk(response);

final String token = response.getBody();
assertThat(StringFunctions.isNullOrWhiteSpace(token)).isFalse();
response = assertDoesNotThrow(() -> postHelloRequest(token));
assertThat(response.getStatusCode()).isEqualTo(HttpStatus.OK);


2 usages  st52165
private ResponseEntity<String> postLoginRequest() {
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);

    HttpEntity<?> entity = new HttpEntity<>(buildAuthenticationRequest(), headers);

    return restTemplate.exchange( url: baseUrl + "/auth/authenticate",
            HttpMethod.POST, entity, String.class);
}

1 usage  st52165
private ResponseEntity<String> postHelloRequest(String token) {
    HttpHeaders headers = new HttpHeaders();
    headers.setBearerAuth(token);

    HttpEntity<String> entity = new HttpEntity<>(headers);

    return restTemplate.exchange( url: baseUrl + "/hello",
            HttpMethod.GET, entity, String.class);
}
```

Obrázek 29 – Ukázka části kódu integračního testu pro přihlášení uživatele

testuje, zda byl úspěšně zpracován http požadavek s dto uživatelských údajů pro přihlášení. V další části metoda zpracovává JWT token, který metoda po přihlášení vrací. Je otestována přítomnost tohoto tokenu v těle http odpovědi a dále je otestován přístup k autorizované části systému pomocí GET http requestu „/hello“. I tento požadavek musí pro úspěšné vykonání testu vrátit http status „OK“.

K vytváření objektů pro testovací účely byly implementovány vlastní třídy s příponou *DataFactory*. Tyto třídy zjednodušují vytváření entitních objektů při implementaci nových testů a napomáhají větší čistotě kódu.

ZÁVĚR

Hlavním cílem diplomové práce bylo vytvořit analýzu, návrh a implementaci webové aplikace pro zpracování sportovních výsledků disciplín požárního sportu. Předpokladem bylo zaměřit se na oblast uchovávání sportovních výsledků disciplín požárního sportu a další s tím související problematiku. To zahrnuje především samotný proces vyhodnocování soutěžních výsledků z pohledu pořadatelů závodu, rozhodčích a soutěžících dle pravidel požárního sportu. Výsledná aplikace měla být navržena tak, aby byla přínosem pro všechny zmíněné osoby. Back-end část aplikace měl být vyvíjen ve frameworku Spring Boot, pro front-endovou část měla být použita knihovna React. Důležité části aplikace měly být pokryty jednotkovými a integračními testy.

Teoretická část diplomové práce se nejprve stručně zaměřuje na definice pojmu informace a informační systém, které s vývojem aplikací úzce souvisí. Následující kapitola představuje nejpoužívanější metodiky vývoje softwaru. V kapitole je krátce shrnuta historie metodik, následně jsou blíže představeny pracovní postupy a výhody či nevýhody použití jednotlivých metodik k vývoji aplikací.

Další kapitola se věnuje analýze současného stavu problematiky vyhodnocování výsledků disciplín požárního sportu. Nejprve jsou stručně popsány činnosti aktérů soutěže s ohledem na pravidla požárního sportu. Další část kapitoly analyzuje dostupné pomocné SW prostředky na současném trhu pro pořadatele a rozhodčí. Analýza se zaměřuje především na zabezpečení těchto aplikací proti lidskému pochybení, ke kterému může při zpracování soutěžních výsledků dojít, a možnost sdílení těchto dat mezi pracovníky soutěže a soutěžícími.

Důležitým poznatkem plynoucím z výsledků analýzy současného stavu byla absence komplexní aplikace pro vyhodnocování výsledků disciplín požárního sportu na současném trhu, která by umožňovala online spolupráci mezi pořadateli závodu, soutěžícími a rozhodčími. Na trhu existují systémy zaměřující se pouze na určitou část problematiky vyhodnocování soutěžních výsledků a není možná výměna informací mezi všemi aktéry soutěže.

V následujících kapitolách jsou blíže představeny všechny požadavky na novou aplikaci a důležité diagramy případů užití. Požadavky na tuto aplikaci byly navrhovány s ohledem na výsledky předchozí analýzy trhu tak, aby tyto nedostatky nová aplikace co nejvíce eliminovala. Při sběru požadavků na nový systém byla zjištěna rozsáhlost řešení dané problematiky. Z tohoto důvodu musely být stanoveny priority pro jednotlivé požadavky.

V další části této práce jsou stručně představeny technologie použité pro implementaci nové aplikace. Dále je věnována pozornost návrhu a implementaci řešení nové aplikace. Je představen a popsán model tříd pro důležité části aplikace a jsou blíže přiblíženy jednotlivé

implementační detaily back-endové a front-endové části včetně způsobu zabezpečení aplikace. Závěrečná část diplomové práce popisuje proces testování nové aplikace a představuje ukázky jednotkových a integračních testů.

Návrh a implementace musely být z důvodu komplexnosti řešení rozděleny do více etap, přičemž praktická část této diplomové práce je výsledkem první etapy. Při tvorbě praktické části diplomové práce bylo hlavní prioritou zaměřit se na vyhodnocování soutěžních výsledků z pohledu rozhodčích a pořadatelů závodu. Back-endová část aplikace obsahuje funkcionality pro zobrazení veškerých informací o závodech, jsou implementovány funkcionality pro rozhodčí, časoměřiče a pořadatele závodu respektující pravidla požárního sportu a umožňující vzájemné sdílení informací v průběhu procesu vyhodnocování soutěžních výsledků. Byla taktéž implementována funkcionalita zajišťující podávání on-line rezervací pro soutěžící. Tato rezervační data jsou za účelem snadného vytváření soutěžních listin sdílena mezi oprávněné rozhodčí a pořadatele závodu. Nejsložitějším úkolem při tvorbě back-endové části bylo vytvoření komponenty pro autorizaci k jednotlivým funkcionalitám souvisejících s problematikou vyhodnocování výsledků na základě hierarchie rolí definovaných pravidly požárního sportu. Tato část byla časově velmi náročná. Bylo nutné si podrobně nastudovat pravidla požárního sportu a prozkoumat možnosti při implementaci hierarchie rolí na základě vícenásobné dědičnosti, kterou Framework Spring ani Java nepodporují. Vše se nakonec podařilo vyřešit pomocí implementace specifických výčtových typů reprezentujících dílčí role závodu včetně hierarchie.

Následoval vývoj front-endové části a testování aplikace. Z důvodu problémů uvedených v závěru předchozího odstavce došlo k časovému zpoždění, které se na vývoji front-endu podepsalo. Front-endová část tak není dokončená v původně plánovaném rozsahu.

Veškeré implementované části programu byly řádně otestovány. Důležité back-endové části aplikace jsou pokryty jednotkovými a integračními testy, nad rámec zadání této práce byla vytvořena automatická dokumentace Swagger umožňující ručně testovat funkčnost REST API.

I přes všechny komplikace, které nastaly zejména během implementace, byly splněny všechny cíle definované v zadání této diplomové práce. Výsledné řešení není prozatím určené k reálnému nasazení do provozu. Je nutné odladit zejména front-endovou část aplikace včetně návrhu grafického designu. Vývoj nové aplikace závěrem této diplomové práce nekončí. Po vylepšení front-endové části je v plánu aplikaci integrovat s již existujícím programem Firesport Timekeeper, který je určen k automatickému ukládání naměřených dat z časomíry do počítače a umožňuje s pomocí relační databáze automaticky vytvářet startovní a výsledkové listiny měřených disciplín ročníku závodu.

CITOVANÁ LITERATURA

[1] Data, informace znalosti?. In: SKLENÁK, Vilém. *Data, informace, znalosti a Internet*. 1. Praha: C. H. Beck, 2001, s. 2-4. ISBN 9788071794097.

[2] SEKERKA, Michal. *Kolmogorovovská složitost a Shannonova informace*. Praha, 2019. Dostupné také z: https://dspace.cuni.cz/bitstream/handle/20.500.11956/118901/130285432.pdf?sequence=1&isAllowed=y. Bakalářská práce. Univerzita Karlova, Matematicko-fyzikální fakulta.

[3] GROŠUP, Tadeáš. *Reporting nad podnikovými daty jako nástroj pro manažerské rozhodování*. Liberec, 2020. Diplomová práce. Technická univerzita v Liberci, Ekonomická fakulta.

[4] Logistické systémy a jejich aplikace v dopravě. In: *Https://pernerscontacts.upce.cz/* [online]. Pardubice: Univerzita Pardubice, 2006 [cit. 2023-07-15]. Dostupné z: https://pernerscontacts.upce.cz/index.php/perner/article/view/1515/1288

[5] Podnik jako regulační obvod. In: VYMĚTAL, Dominik. *Informační systémy v podnicích: teorie a praxe projektování*. 1. Praha: Grada, 2009, s. 14. ISBN 9788024762807.

[6] *Digitalizace podniků 2022* [online]. 2023, [cit. 2023-07-15]. Dostupné z: https://www.assecosolutions.cz/files/2304-asseco-report-displej-v11.pdf

[7] Vybrané aspekty návrhu webových informačních systémů. In: *Vybrané aspekty návrhu webových informačních systémů*. Scientific Press Serie ; 2/2013. Vsetín: Scientific Press by Silhavy, 2013, s. 5-12. ISBN 9788090474130.

[8] In: *Software Engineering and Testing*. 1. Burlington, Massachusetts: Jones & Bartlett Learning, 2010, s. 37-46. ISBN 9780763783020.

[9] VARHOL, Peter. To agility and beyond: The history—and legacy—of agile development. In: *TechBeacon* [online]. Houston, Texas: TechBeacon, 2023 [cit. 2023-

08-07]. Dostupné z: https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development

[10] *Manifesto for Agile Software Development* [online]. Snowbird, Utah, US: Q7 Enterprises, Inc., 2001 [cit. 2023-08-07]. Dostupné z: https://agilemanifesto.org/

[11] In: *Agilní metody řízení projektů*. 2. vydání. Brno: Computer Press, 2019, s. 15-29. ISBN 9788025149614.

[12] In: *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007, s. 51-68. ISBN 978-80-251-1503-9.

[13] BARENSCHEER, Tim. What is The Difference Between Agile and Unified Process Methodology?. In: *Teamly* [online]. Las Vegas, Nevada, United States: Teamly, 2023 [cit. 2023-08-08]. Dostupné z: https://www.teamly.com/blog/difference-between-agile-and-unified-process-methodology/

[14] Hasičský sport získává na oblibě, zvláště dominuje na vesnicích. In: *TÝDEN.cz* [online]. Praha 4 - Braník, CZ: EMPRESA MEDIA, a.s., 2018 [cit. 2023-08-10]. Dostupné z: https://www.tyden.cz/rubriky/relax/ostatni/hasicsky-sport-ziskava-na-oblibe-zvlaste-dominuje-na-vesnicich_487169.html

[15] *Sbírka interních aktů řízení generálního ředitele HZS ČR - částka 10/2018*. In: . Praha: Hasičský záchranný sbor České republiky, 2018, ročník 10, 10/2018, číslo 2018. Dostupné také z: https://www.hzscr.cz/soubor/pravidla-ps-2018-pdf.aspx

[16] Zpracování výsledků. In: *SH ČMS: Úsek mládeže* [online]. Praha: Sdružení Hasičů Čech Moravy a Slezska, 2023 [cit. 2023-08-10]. Dostupné z: https://mladez.dh.cz/index.php/sportovni-cinnost/zpracovani-vysledku

[17] *Firesport.eu* [online]. Praha: INTERNET CZ, a.s., 2000 [cit. 2023-08-10]. Dostupné z: https://www.firesport.eu/

[18] *Časomíry Ing. Libor Valeš* [online]. Žákava, CZ: Ing. Libor Valeš, c2008-2021 [cit. 2023-08-10]. Dostupné z: https://casomiry.com/

[19] *TRV elektronik: Výroba sportovních časomír* [online]. Moravské Budějovice, CZ: Tomáš Kocáb, 2019 [cit. 2023-08-10]. Dostupné z: https://www.trv-kocab.cz/cs/

[20] *Software pro zpracování výsledků z časomíry pro požární sport: Měření disciplín požárního sportu*. Pardubice, 2020. Bakalářská práce. Univerzita Pardubice, Dopravní fakulta Jana Pernera.

[21] Introduction to Spring Framework. In: *GeeksForGeeks* [online]. Noida, Uttar Pradesh: GeeksForGeeks, 2023 [cit. 2023-08-22]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-spring-framework/

[22] Introduction to Spring Framework. In: *Spring: by VMware Tanzu* [online]. Palo Alto, CA, USA: VMware, Inc., c2004-2009 [cit. 2023-08-22]. Dostupné z: https://docs.spring.io/spring-framework/docs/3.0.0.M4/reference/html/ch01s02.html

[23] Co je Java Spring Boot?. In: *Microsoft* [online]. Redmond, WA, US: Microsoft Corporation, 2023 [cit. 2023-08-22]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-java-spring-boot

[24] T-SQL Tutorial. In: *JavaTpoint* [online]. Burlington, Massachusetts, US: Privacy Protect, LLC, c2011-2021 [cit. 2023-08-22]. Dostupné z: https://www.javatpoint.com/t-sql

[25] SQL Server Express je zdarma, ale…. In: *Michal Zobec: Blog* [online]. Praha: Michal Zobec, 2023 [cit. 2023-08-22]. Dostupné z: https://www.michalzobec.cz/sql-server-express-je-zdarma-ale-4121

[26] Use containers to Build, Share and Run your applications. In: *Https://www.docker.com/* [online]. California, US: Docker Inc., 2023 [cit. 2023-08-22]. Dostupné z: https://www.docker.com/resources/what-container/

[27] VONDRA, Marek. Lekce 1 - Docker - Teorie a instalace. *Itnetwork: Docker* [online]. 2022, **2022**(1), 1 [cit. 2023-08-22]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/site/docker/docker-teorie-a-instalace

[28] Docker for beginners: Introduction. In: *Docker for beginners* [online]. Noida, IN: Prakhar Srivastav, 2018 [cit. 2023-08-22]. Dostupné z: https://docker-curriculum.com/

[29] What Is the Difference Between Swagger and OpenAPI?. In: *Swagger: Supported by SMARTBEAR* [online]. Arizona, US: SmartBear Software, 2023 [cit. 2023-08-22]. Dostupné z: https://swagger.io/blog/api-strategy/difference-between-swagger-and-openapi/

[30] Swagger vs. OpenAPI: Understanding the Difference. In: *Medium* [online]. Seattle WA, US: Amazon Registrar, Inc., 2023 [cit. 2023-08-22]. Dostupné z: https://medium.com/cloud-native-daily/swagger-vs-openapi-understanding-the-difference-e4b735bc0076

[31] *Mapy.cz Developer: REST API Mapy.cz* [online]. Praha 5: Seznam.cz, a.s., c1996-2023 [cit. 2023-08-22]. Dostupné z: https://developer.mapy.cz/rest-api/

[32] MÁCA, Jindřich. Úvod do React. *Itnetwork: React* [online]. 2019, **2019**(1), 1 [cit. 2023-08-23]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/javascript/react/zaklady/uvod-do-react

[33] 2022 Developer Survey: Web frameworks and technologies. In: *Stackoverflow* [online]. New York, US: Stack Exchange, Inc., 2022 [cit. 2023-08-23]. Dostupné z: https://survey.stackoverflow.co/2022/#most-popular-technologies-webframe

[34] Vývoj webových aplikací: single-page vs. multi-page aplikace. In: *Rascasone* [online]. Praha: Rascasone s.r.o., 2021 [cit. 2023-08-23]. Dostupné z: https://www.rascasone.com/cs/blog/jednostrankove-vicestrankove-web-aplikace

[35] Proč k vývoji webových aplikací použít technologii NodeJS?. In: *Rascasone* [online]. Praha: Rascasone s.r.o., 2021 [cit. 2023-08-23]. Dostupné z: https://www.rascasone.com/cs/blog/node-js-architektura-moduly-npm#uacute-vod-do-v-yacute-voje-webov-yacute-ch-aplikac-iacute-s-node-js-npm-a-moduly

[36] Návrh. In: *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007, s. 327-380. ISBN 978-80-251-1503-9.

[37] ČERNÝ, Martin. Obrázek - DIKW hierarchická informační pyramida. In: *Medium.com* [online]. Hayes: Identity Protection Service, 2019 [cit. 2023-07-03]. Dostupné z:

https://miro.medium.com/v2/resize:fit:4800/format:webp/1*ivLeNNTBIKnIGj24pSS4CQ.png

[38] Obrázek - Blokové schéma jednoduchého regulačního obvodu. In: *Vysoká škola báňská - Technická univerzita Ostrava* [online]. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2006 [cit. 2023-07-04]. Dostupné z: http://books.fs.vsb.cz/SyntezaReg/images/obr/01.gif

[39] Obrázek - Spiral Model. In: *Airbrake* [online]. California, US: Airbrake Technologies, 2023 [cit. 2023-08-06]. Dostupné z: https://blog.airbrake.io/hubfs/Imported_Blog_Media/Screen-Shot-2016-09-29-at-7_56_49-AM.png

[40] Obrázek - Waterfall model. In: *Tutorialspoint* [online]. Arizona, US: Registration Private, 2023 [cit. 2023-08-06]. Dostupné z: https://www.tutorialspoint.com/sdlc/images/sdlc_waterfall_model.jpg

[41] Obrázek - Scrum Board. In: *Bluescape* [online]. Redwood City, CA: Amazon Registrar, Inc., 2023 [cit. 2023-08-07]. Dostupné z: https://www.bluescape.com/_next/image?url=https%3A%2F%2Fimages.ctfassets.net%2Ftapz5cpfdvpb%2F4Dgz0MrU7nwN026AS3NwVi%2F4aeda1e08259e8f883e60cbf04f84ec3%2FiStock-1282375527.jpg&w=1080&q=75

[42] Obrázek - The core workflows and phases of the Unified Process according to [JBR99]. In: *ResearchGate* [online]. Germany: MESH DIGITAL LIMITED, c2008-2023 [cit. 2023-08-08]. Dostupné z: https://www.researchgate.net/profile/Juergen-Muench/publication/259573166/figure/fig3/AS:669525993324566@1536638844794/The-core-workflows-and-phases-of-the-Unified-Process-according-to-JBR99.png

[43] Obrázek - Spring Framework Diagram. In: *Spring: by VMware Tanzu* [online]. Palo Alto, CA, USA: VMware, Inc., c2004-2009 [cit. 2023-08-22]. Dostupné z: https://docs.spring.io/spring-framework/docs/3.0.0.M4/reference/html/images/spring-overview.png

[44] Obrázek - Containers and Virtual Machines Together. In: *Docker* [online]. California, US: Docker Inc., 2023 [cit. 2023-08-22]. Dostupné z: https://www.docker.com/wp-content/uploads/2021/11/docker-containerized-and-vm-transparent-bg.png

PŘÍLOHY

PŘÍLOHA A – DIGITÁLNÍ PŘÍLOHA

PŘÍLOHA A – DIGITÁLNÍ PŘÍLOHA

Přiložené DVD obsahuje:

- Diplomovou práci ve formátu *.pdf*
- Aplikaci ISSV včetně zdrojových kódů
- Soubor Enterprise Architect *.ea* obsahující specifikací požadavků, UML Use Case diagramy, UML Class diagramy a slovníček pojmů.
- Vyexportované obrázky *.ea* diagramů ve formátu *.png*
- Vyhotovenou analýzu podstatných jmen a sloves ve formátu MS Excel *.xlsx*