FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Systém pro sportovní kluby
Student:	Lukáš Paukert
Vedoucí:	Ing. Filip Glazar
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat webovou aplikaci, která bude sloužit sportovním klubům, a to především těm, které se věnují orientačnímu běhu. V aplikaci bude možné spravovat jednotlivé uživatele, jakými jsou například členové klubu či trenéři. Systém zároveň musí umět evidovat jednotlivé závody a umožnit registraci ke konkrétním událostem. Systém bude integrovat data z informačního systému Českého svazu orientačních sportů ORIS.

Při realizaci postupujte dle následujících kroků:

1) Analyzujte existující řešení
2) Specifikujte funkční a nefunkční požadavky
3) Na základě předchozího bodu zvolte vhodné technologie
4) Navrhněte a implementujte prototyp aplikace
5) Prototyp podrobte uživatelskému testování
6) Na základě výsledku z testování navrhněte potřebné úpravy systému, pokud z testování nějaké vzešly
7) Aplikaci připravte pro nasazení a diskutujte další práci na projektu

---

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 8. prosince 2021 v Praze.

Bakalářská práce

SYSTÉM PRO SPORTOVNÍ KLUBY

Lukáš Paukert

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Filip Glazar
11. května 2022

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2022 Lukáš Paukert. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.

Odkaz na tuto práci: Paukert Lukáš. *Systém pro sportovní kluby.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Tímto bych chtěl poděkovat vedoucímu práce Ing. Filipu Glazarovi za cenné rady a přátelský přístup. Své poděkování bych rád také věnoval přítelkyni a celé mé rodině, kteří mi pomáhali s kontrolou práce a zároveň mě podporovali během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2022 .....

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací webové aplikace, která bude sloužit sportovním klubům, a to především těm, jež se věnují orientačnímu běhu. Prototyp aplikace je vytvořen v programovacím jazyku PHP s využitím Symfony frameworku. Aplikace umožňuje správu událostí a uživatelů, kteří se mohou na tyto události přihlašovat. Mezi její přednosti se řadí propojení s informačním systémem Českého svazu orientačních sportů, které minimalizuje počet manuálních úkonů nezbytných pro přidání nové události a odeslání evidovaných přihlášek do celorepublikového systému.

Samotné implementaci předchází analýza existujících řešení, specifikace funkčních a nefunkčních požadavků a výběr vhodných technologií. Výsledná aplikace byla podrobena uživatelskému testování a je připravena pro nasazení místo aktuálně využívaného systému.

Klíčová slova monolitická webová aplikace, informační systém, sportovní klub, orientační běh, přihlašování na události, ORIS API, PHP, Symfony

Abstract

This bachelor thesis deals with the design and implementation of a web application that will be used by sports clubs, especially those involved in orienteering. The application prototype is developed in the PHP programming language using the Symfony framework. The application allows the management of events and users who can log in to these events. One of its advantages is the connection with the information system of the Czech Orienteering Federation, which minimizes the number of manual actions needed to add a new event and send registered entries to the national system.

The implementation itself is preceded by an analysis of existing solutions, specification of functional and non-functional requirements and selection of suitable technologies. The developed application has been subjected to user testing and is ready for deployment in place of the currently used system.

Keywords monolithic web application, information system, sports club, orienteering, register for events, ORIS API, PHP, Symfony

Seznam zkratek

API	Application Programming Interface
CMS	Content Management System
CSS	Cascading Style Sheets
CSV	Comma-separated Values
DBAL	Database Abstraction Layer
DOM	Document Object Model
FTP(S)	File Transfer Protocol (Secure)
GDPR	General Data Protection Regulation
HTML	Hypertext Markup Language
HTTP(S)	Hypertext Transfer Protocol (Secure)
IS ORIS	Informační systém Českého svazu orientačních sportů
JSON	JavaScript Object Notation
MVC	Model–view–controller
ORM	Object–relational Mapping
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
SQL	Structured Query Language
SSH	Secure Shell
UI	User Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language

Kapitola 1
Úvod

1.1 Motivace

Již od mala jsem rád věnoval svůj volný čas různým sportovním aktivitám. Nejdříve jsem navštěvoval plavecký kroužek, poté jsem hrál fotbal, tenis a jednoho dne jsem se díky kamarádovi dostal i k orientačnímu běhu, u kterého jsem zůstal dodnes. Skoro každý víkend před začátkem pandemie covidu-19, která soutěže v orientačním běhu poprvé ovlivnila v roce 2020, se náš klub KOB Ústí nad Orlicí účastnil závodů ve východočeské oblasti nebo i závodů celorepublikové úrovně.

Na závody jsme se ze začátku přihlašovali vedoucímu našeho klubu pomocí e-mailu, zprávou na Facebooku nebo pomocí jakéhokoliv jiného komunikačního kanálu. S rozšiřující se členskou základnou klubu však tento způsob přihlašování přestával být udržitelný, a proto vzniklo klubové fórum, které mělo způsob přihlašování sjednotit. Kvůli komplexnosti nasazeného řešení však nebylo přihlašování na závody pro členy klubu tak jednoduché a přímočaré, jak bychom si přáli. Z tohoto důvodu jsem v druhé polovině roku 2016 vytvořil webovou aplikaci, která od roku 2017 nahradila používané diskuzní fórum.

Tato webová aplikace bude podrobněji představena v sekci 2.1.1, avšak jelikož se jednalo o jeden z mých prvních programátorských projektů, tak nyní i sám vidím, že aplikace nebyla navržena ani naprogramována nejlépe. To nyní způsobuje řadu problémů při jejím dalším rozvoji.

1.2 Cíle práce

V rámci své bakalářské práce vytvořím novou webovou aplikaci, která bude v porovnání se současným řešením uživatelsky přívětivější a která bude obsahovat i nové funkcionality, jež výrazně zjednoduší procesy spojené s přihlašováním na tréninky a závody. Aplikace musí umožňovat správu uživatelů a událostí. Na jednotlivé události se bude možné registrovat a přihlášky bude následně možné odeslat do informačního systému Českého svazu orientačních sportů ORIS.

Samotné implementaci prototypu bude předcházet analýza, která bude mimo jiné zahrnovat popis již existujících řešení a specifikaci funkčních a nefunkčních požadavků. Následovat bude návrh aplikace a výběr vhodných technologií. Po implementaci bude systém podroben uživatelskému testování a na základě výsledků budou navrženy potřebné úpravy. V neposlední řadě bude aplikace připravena pro nasazení místo aktuálně využívaného systému.

Kapitola 2
Analýza

Následující kapitola se zabývá analýzou existujících řešení, popisuje funkční a nefunkční požadavky na novou aplikaci a v neposlední řadě obsahuje jednotlivé případy užití, které představují detailnější specifikaci dříve uvedených funkčních požadavků.

2.1 Existující řešení

Následující podkapitola je věnována již existujícím informačním systémům, které by měly usnadnit sportovním klubům a skupinám administrativní činnost. V současné době jich na trhu existuje již poměrně velké množství, a proto se zaměřím pouze na ty, které mi přišly nejrozšířenější nebo nějakým způsobem zajímavé. V následujících sekcích budou představena dvě řešení z komerční sféry, jeden bezplatný systém a také bude podrobněji popsána aplikace, která je v současné době využívána klubem KOB Ústí nad Orlicí.

Několik systémů, které mají řešit podobné problémy, vzniklo v minulosti i v rámci závěrečných prací na různých českých univerzitách. Pro kompletnost mohu například zmínit práce [1, 2, 3], avšak jejich obsah zde nebudu detailněji rozebírat.

2.1.1 KUOris

Klub orientačního běhu Ústí nad Orlicí nyní využívá pro přihlašování na závody webovou aplikaci KUOris. Tento systém vznikl v roce 2016 jako nástupce řešení, které bylo založené na open source systému phpBB, jenž je nejčastěji využíván pro tvorbu diskuzních fór [4]. V průběhu let byl drobně upravován, avšak z důvodu špatného návrhu není možné do systému KUOris jednoduše přidávat nové funkcionality.

Systém v aktuálním stavu podporuje přidávání a následné spravování událostí. Při vytváření události se členům klubu, kteří si tuto funkcionalitu povolili, zasílá informační e-mail se základními informacemi o přidané akci. Na jednotlivé události se členové klubu mohou přihlašovat a mohou k nim psát komentáře.

Mezi největší nedostatky aplikace KUOris se řadí zastaralé uživatelské rozhraní a chybějící automatizace. V dnešní době, kdy na web přistupuje více lidí z mobilních zařízení než z počítačů [5], by mělo být uživatelské rozhraní jistě responzivní. Další již zmíněný nedostatek spatřuji u nevyužité možnosti automatizace. IS ORIS nabízí API, jež je detailněji popsané v sekci 4.3.1.3, pro získávání informací o závodech a i pro přihlašování na jednotlivé akce. V současném systému této možnosti není využíváno a všechny úkony tedy musí administrátor provádět ručně.

Nevýhody: složitá rozšiřitelnost, neresponzivní UI, absence propojení s IS ORIS

2.1.2 KIS – Klubový Informační Systém

KIS je komplexní klubový informační systém, který je díky svým pokročilým funkcionalitám využíván i největšími sportovními subjekty v České republice. Mezi uživatele systému se totiž například řadí Český svaz ledního hokeje, Fotbalová asociace České republiky, Český olympijský výbor a další velké fotbalové a hokejové kluby. [6, 7]

Mezi klíčové funkčnosti systému se řadí podpora pro pořádání klubových akcí, správa plateb, evidence dokumentů, zobrazování statistik, kalendář a mnoho dalších funkcí. K systému je možné přistupovat i přes mobilní aplikaci, která je dostupná pro zařízení s operačním systém iOS a Android. Jedná se o komerční systém, dle ceníku se cena skládá z jednorázového počátečního poplatku 9 900 Kč bez DPH a následného měsíčního poplatku 300 Kč bez DPH za technickou správu. [6]

Výhody: komplexní klubový systém s pokročilými funkcemi a mobilní aplikací

Nevýhody: nákladnost (jednorázový a následné měsíční poplatky), absence propojení s IS ORIS

2.1.3 Sportes – Český sportovní evidenční systém

Jedná se o další komplexní systém pro sportovní kluby, ke kterému je opět možné přistupovat jak z webového prohlížeče, tak i z mobilní aplikace. Obsahuje podobné funkcionality jako systém KIS, tj. správu událostí, evidenci členské základny, docházky a plateb. Pro menší sportovní kluby však pravděpodobně bude překážkou pro jeho využívání uvedená cena, neboť za variantu pro maximálně 100 členů je stanovena na 12 000 Kč bez DPH ročně. [8]

Výhody: systém pro kompletní správu sportovního klubu s vlastní mobilní aplikací

Nevýhody: nákladnost (12 000 Kč za rok bez DPH), absence propojení s IS ORIS

2.1.4 Týmuj

Týmuj je online služba, která byla do ostrého provozu spuštěna v květnu roku 2008. Od té doby v ní bylo vytvořeno přes 30 000 týmů a svůj účet si vytvořilo přes 250 000 sportovců. Služba Týmuj je od svého vzniku neustále vylepšována a v roce 2017 byly spuštěny i mobilní aplikace pro operační systémy iOS a Android.

Týmuj umožňuje vytvářet a spravovat události, evidovat docházku u jednotlivých událostí, posílat zprávy do týmového chatu a k jednotlivým akcím, evidenci plateb, sdílení fotografií a mnoho dalších věcí. V rámci týmu mají jednotliví členové jednu ze tří rolí (majitel, správce nebo hráč) a na základě přidělené role mají odpovídající práva.

Popularitu tohoto online nástroje dokazuje fakt, že jen za poslední rok bylo přes Týmuj zorganizováno přes 300 000 událostí. Takový počet akcí bude jistě z nemalé části způsoben tím, že využívání Týmuj není zpoplatněno a že taktéž neexistuje omezení na maximální počet členů týmu. [9]

Výhody: užívání není zpoplatněno, obsahuje mnoho funkcí, existuje webová i mobilní aplikace

Nevýhody: absence propojení s IS ORIS

2.2 Funkční požadavky

Funkční požadavky popisují požadované funkcionality systému. Mohou například specifikovat, jakým způsobem bude uživatel moci pracovat se systémem, jaké procesy by měl systém podporovat a jaké budou vstupy a výstupy těchto procesů. [10]

F1 evidence a správa uživatelů

Systém bude umožňovat registraci nových uživatelů, neregistrovaný uživatel nemá přístup do systému. Při registraci bude nutno zadat registrační číslo, e-mailovou adresu, heslo a jméno a příjmení člena klubu. Právě pomocí registračního čísla a zadaného hesla bude následně uživatel přistupovat do aplikace. Kromě těchto základních údajů bude mít administrátor možnost uživateli nastavovat a upravovat stav osobního konta, typ členství (aktivní nebo pasivní) a roli (viz sekce 2.2.1).

F2 evidence a správa závodů a tréninků

Dále systém bude umožňovat evidenci a správu tréninků a závodů (událostí). Konkrétně je potřeba mít možnost u každé události evidovat:

- název
- typ události (trénink nebo závod)
- datum a čas události (kdy událost začíná)
- místo konání
- pořadatele
- datum a čas uzávěrky přihlášek
- disciplínu
- vypsané kategorie
- další informace
- v případě závodu bude možné dále evidovat:
  - typ závodu
  - webovou stránku
- u tréninku bude možné oproti závodu naopak evidovat maximální kapacitu

Samotné zadání události do systému bude umožněno ručním vyplněním všech požadovaných údajů nebo automatickým importem z informačního systému Českého svazu orientačních sportů pomocí unikátního identifikátoru závodu.

Na hlavní stránce systému by se měly zobrazovat závody a tréninky s nejbližší uzávěrkou přihlášek. V zadaných trénincích a závodech bude možné vyhledávat a filtrovat a zadanou událost musí být možno zrušit bez odstraňování ze systému. Taková událost bude následně od nezrušených událostí vizuálně odlišena a nebude možné se na ni dále přihlašovat.

F3 přihlašování na závody a tréninky

Na událost evidovanou v systému se mohou registrovaní uživatelé přihlásit, pokud dosud neproběhla uzávěrka přihlášek. Při přihlášení si uživatelé musí navíc vybrat jednu z nabízených kategorií, do které se chtějí přihlásit, a mohou volitelně sdělit, zdali mají možnost jet vlastním autem a svést s sebou i další členy klubu. Svou účast mohou uživatelé do konce uzávěrky přihlášek také odřeknout.

F4 synchronizace přihlášek

Trenéři klubu budou mít možnost odeslat všechny přihlášky evidované u konkrétního závodu do IS ORIS přes jeho API. Tato možnost bude ze zřejmých důvodů dostupná pouze pro závody, které jsou v IS ORIS evidované.

F5 **oznámení**

Trenéři a administrátoři systému budou mít možnost vytvořit oznámení, která se budou zobrazovat na hlavní stránce systému.

F6 **komentáře**

Komentáře mohou psát registrovaní uživatelé ke všem tréninkům a závodům. Nejnovější komentáře by se měly zobrazovat (včetně informace kam byly napsány) na hlavní stránce systému.

2.2.1 Role

Každému uživateli bude náležet právě jedna z následujících uživatelských rolí:

- registrovaný uživatel
  - základní uživatelská role, která nemá žádná speciální oprávnění
  - výčet akcí, které může registrovaný uživatel vykonávat:
    - přihlašovat a odhlašovat se z tréninků a závodů
    - psát komentáře k jednotlivým tréninkům a závodům a následně je i upravovat
    - měnit svou e-mailovou adresu a heslo pro přihlášení do systému
- trenér
  - role, která náleží trenérům působícím v klubu
  - oproti roli registrovaný uživatel má právo:
    - přidávat a následně spravovat tréninky a závody
    - odesílat evidované přihlášky do IS ORIS
    - psát a upravovat svá oznámení
- administrátor
  - má kompletní přístup ke všem funkcionalitám systému
  - oproti uživatelům s rolí trenér může navíc:
    - upravovat a mazat veškeré komentáře a oznámení
    - spravovat uživatele
    - odhlašovat a přihlašovat uživatele na tréninky a závody (i po termínu přihlášek)
    - odebírat a přidávat informaci o možnosti vzít auto (i po termínu přihlášek)

2.3 Nefunkční požadavky

Nefunkční požadavky se na rozdíl od funkčních požadavků zabývají vlastnostmi a omezeními daného systému. Mohou se mezi nimi objevit požadavky, které budou podstatné jak pro samotné uživatele systému, jako například doba odezvy, zabezpečení systému, způsob zobrazování na mobilních zařízeních, tak zde mohou být uvedeny i požadavky techničtějšího charakteru, jako například použití konkrétních technologií nebo možnost snadné rozšiřitelnosti systému. [10]

N1 **responzivita**

Webová aplikace bude přizpůsobovat svůj vzhled na základě rozlišení klientského zařízení.

N2 **technologie**

S ohledem na aktuálně využívané hostingové služby musí být aplikace napsána v programovacím jazyku PHP a pro ukládání dat musí být využita relační databáze MariaDB.

N3 zabezpečení

Komunikace s aplikací bude probíhat pouze přes šifrovaný protokol HTTPS.

N4 rozšiřitelnost a modifikovatelnost

Aplikace by měla být snadno rozšiřitelná a modifikovatelná. Její využití by tedy nemělo být limitováno pouze na kluby orientačního běhu, ale své uplatnění by měla nalézt i u organizací, které se zaměřují na jiné sportovní aktivity.

N5 výkon a dostupnost

Aplikace bude schopná v jednu chvíli obsluhovat jednotky až nižší desítky uživatelů s odezvou v řádu sekund.

N6 právní požadavky

Registrovaného uživatele musí být možné anonymizovat v souladu s obecným nařízením o ochraně osobních údajů (GDPR).

2.4 Případy užití

Tato podkapitola se věnuje jednotlivým případům užití, které představují detailnější specifikaci funkčních požadavků. Případy užití poté mohou posloužit jako základ pro tvorbu uživatelské dokumentace a také mohou být podkladem pro tvorbu akceptačních testů. [11]

2.4.1 Aktéři

Aktérem nebo účastníkem nazýváme osobu (případně její roli), která bude pracovat se systémem. V naší aplikaci uvažujeme celkem čtyři různé aktéry – přihlášeného a nepřihlášeného uživatele, administrátora a trenéra. Vztahy mezi těmito aktéry jsou popsány v následujícím textu a jsou znázorněny na obrázku 2.1.

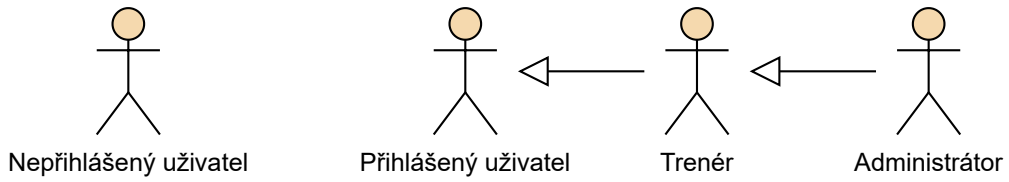
Nepřihlášený uživatel
Nepřihlášený uživatel má přístup pouze k přihlášení a registraci. Po provedení první uvedené akce se stane přihlášeným uživatelem.

Přihlášený uživatel
Přihlášený uživatel má práva, která byla již dříve popsána v sekci 2.2.1 u role registrovaný uživatel. Jelikož je uživatel již přihlášený do systému, tak nemá přístup k přihlašovací stránce a k registraci.

Trenér
Práva trenéra vycházejí z práv přihlášeného uživatele a navíc zahrnují práva na správu závodů a tréninků. Podrobný popis práv lze nalézt v sekci 2.2.1 u role trenér.

Administrátor
Administrátor má přístup ke všem funkcionalitám systému. Jeho práva jsou popsána v sekci 2.2.1 u role administrátor.

Obrázek 2.1 Aktéři



2.4.2 Případy užití

UC1 registrace

Před prvním přístupem do systému se musí člen klubu zaregistrovat.

1. Nepřihlášený uživatel klikne na položku *Registrace* v hlavní nabídce.
2. Pro úspěšné dokončení registrace musí vyplnit svou e-mailou adresu, registrační číslo, heslo, jméno a příjmení a kliknout na tlačítko *Registrovat*.
3. Registraci musí následně schválit některý z administrátorů a poté se již nepřihlášený uživatel může pomocí zadaného registračního čísla a hesla přihlásit do systému.

UC2 přihlášení a odhlášení z aplikace

Nepřihlášený uživatel bude při pokusu o přístup do systému vyzván k přihlášení.

1. Nepřihlášený uživatel zadá své registrační číslo a heslo a klikne na tlačítko *Přihlásit*.
2. Pokud přihlášení proběhlo úspěšně, bude uživatel přesměrován na hlavní stránku aplikace nebo na stránku, na kterou se snažil před přihlášením přistoupit.
3. Přihlášený uživatel má naopak možnost se ze systému odhlásit kliknutím na položku *Odhlásit* v hlavní nabídce.

UC3 zobrazení seznamu závodů nebo tréninků

Tento případ užití popisuje zobrazení seznamu závodů/tréninků evidovaných v systému.

1. Přihlášený uživatel klikne na položku *Závody* nebo *Tréninky* v hlavní nabídce.
2. Ve výchozím nastavení se zobrazují všechny nadcházející události, avšak upravením filtru v horní části stránky lze zobrazit i minulé události nebo provést filtraci na základě názvu události.
3. Na jedné stránce se vždy zobrazuje maximální 15 událostí, další události je možné zobrazit přejitím na následující (předchozí) stránku.

UC4 zobrazení detailních informací o události

V seznamu závodů/tréninků jsou zobrazovány pouze základní informace, pro zobrazení detailních informací o události je nezbytné přejít na samostatnou stránku.

1. Přihlášený uživatel se může prokliknout na stránku s informacemi o události z(e):
   a. seznamu závodů/tréninků (viz **UC3**).
   b. hlavní stránky, kde se zobrazuje 5 závodů a 3 tréninky, u kterých nejdříve končí uzávěrka přihlášek.
2. Přechod na stránku se všemi evidovanými informacemi o události proběhne po kliknutí na název události v jednom ze seznamů uvedených výše.

UC5 přihlášení na událost

Přihlášení na událost je možné provést pouze pokud událost nebyla zrušena, neproběhla uzávěrka přihlášek a pokud již není daný uživatel na danou událost přihlášený.

1. Přihlášený uživatel přejde na stránku s detailními informacemi o události (viz **UC4**).
2. Uživatel vybere jednu z nabízených kategorií a volitelně sdělí, zda má možnost jet vlastním autem a svézt s sebou i další členy klubu.
3. Účast na události potvrdí uživatel kliknutím na tlačítko *Přihlásit*.

UC6 odhlášení z události

Odhlášení z události je možné provést pouze pokud nebyla událost zrušena, neproběhla uzávěrka přihlášek a pokud je daný uživatel na danou událost přihlášený.

1. Přihlášený uživatel přejde na stránku s detailními informacemi o události (viz **UC4**).
2. Účast na události uživatel odřekne kliknutím na tlačítko *Odhlásit* a potvrzením následně zobrazeného dialogu.

UC7 přidání a úprava komentáře

Tento případ užití popisuje přidání a následnou úpravu komentáře u konkrétní události.

1. Přihlášený uživatel přejde na stránku s detailními informacemi o události (viz **UC4**).
2. Po napsání komentáře do textového pole klikne na tlačítko *Přidat komentář*.
3. Své komentáře má uživatel možnost upravit kliknutím na ikonku pro úpravu vedle konkrétního komentáře. To načte napsaný komentář do textového pole a po jeho úpravě dojde k uložení kliknutím na tlačítko *Upravit komentář*.

UC8 úprava osobních údajů

Přihlášený uživatel si může změnit svou e-mailovou adresu a heslo, pomocí něhož se přihlašuje do systému.

1. Přihlášený uživatel klikne na položku *Nastavení* v hlavní nabídce.
2. Po upravení požadovaných údajů klikne na tlačítko *Uložit změny*.

UC9 přidání události

Tento případ užití popisuje přidání nové události do systému.

1. Trenér klikne na položku *Administrace* v hlavní nabídce a následně vybere možnost *Přidat nový trénink* nebo *Přidat nový závod*.
2. Pokud je přidávaná událost již evidovaná v IS ORIS, může uživatel zadat ORIS ID události a většina požadovaných informací se automaticky předvyplní.
3. Po (do)vyplnění požadovaných informací se událost přidá do systému kliknutím na tlačítko *Přidat událost*.

UC10 úprava události

Tento případ užití popisuje proces upravení události již evidované v systému.

1. Trenér klikne na položku *Administrace* v hlavní nabídce a následně vybere ze seznamu událost, kterou si přeje upravit.
2. Po upravení požadovaných informací klikne na tlačítko *Upravit událost*.

UC11 synchronizace přihlášek

Role trenér v tomto systému sama o sobě nestačí pro odeslání přihlášek do IS ORIS. Je zapotřebí, aby uživatel měl také zřízený účet v IS ORIS a měl práva na přihlašování závodníků klubu.

1. Trenér klikne na položku *Administrace* v hlavní nabídce a vybere závod, jehož přihlášky si přeje odeslat.
2. Přihlášky evidované u dané události budou odeslány po vyplnění přístupových údajů do IS ORIS a kliknutí na tlačítko *Odeslat přihlášky*.

UC12 vytvoření a úprava oznámení

Tento případ užití popisuje vytvoření a úpravu oznámení.

1. Pro vytvoření nového oznámení klikne trenér na položku *Administrace* v hlavní nabídce a následně vybere možnost pro přidání nového oznámení.
2. Oznámení bude uloženo po vyplnění formuláře a kliknutí na tlačítko *Přidat oznámení*.
3. Úpravu existujícího oznámení je naopak možné provést po vybrání konkrétního oznámení v administraci a kliknutí na tlačítko *Upravit vybrané oznámení*.
4. Provedené změny budou uloženy po kliknutí na tlačítko *Upravit oznámení*.

UC13 úprava a anonymizace uživatele

Funkce anonymizace uživatele přenastaví členovo jméno na *Anonymní uživatel* a smaže všechny jeho údaje (registrační číslo, stav osobního konta, typ členství, e-mail a heslo). Tímto je uživateli znemožněno přihlášení do aplikace.

1. Administrátor klikne na položku *Administrace* v hlavní nabídce a následně vybere ze seznamu uživatele, kterého si přeje upravit nebo anonymizovat.
2. Anonymizaci člena lze provést kliknutím na tlačítko *Anonymizovat vybraného člena* a potvrzením následně zobrazeného dialogu.
3. Formulář pro úpravu uživatele se naopak zobrazí po kliknutí na tlačítko *Upravit vybraného člena*. Po upravení požadovaných informací se změny uloží kliknutím na tlačítko *Uložit změny*.

2.4.3 Pokrytí funkčních požadavků v případech užití

V tabulce 2.1 je zachyceno pokrytí funkčních požadavků v jednotlivých případech užití. Symbol ✓ říká, že příslušný funkční požadavek je součástí daného případu užití.

Tabulka 2.1 Mapování funkčních požadavků na případy užití

	F1	F2	F3	F4	F5	F6
UC1	✓					
UC2	✓					
UC3		✓				
UC4		✓				
UC5			✓			
UC6			✓			
UC7						✓
UC8	✓					
UC9		✓				
UC10		✓				
UC11				✓		
UC12					✓	
UC13	✓					

Kapitola 3
Návrh

Následující kapitola se zabývá návrhem nové webové aplikace. Nejprve bude představena její architektura a členění. Následně budou představeny technologie, které budou využity při vývoji aplikace, a též bude odůvodněno, proč byly jednotlivé technologie zvoleny.

3.1 Architektura

Jedno z prvních rozhodnutí, které bylo v souvislosti s návrhem aplikace potřeba udělat, se týkalo architektury aplikace. Dle [12] se v případě webových aplikací můžeme setkat s rozdělením na thin-client a thick-client. První zmíněná možnost znamená, že veškerá logika se nachází na straně serveru. Pokud se aplikace na straně serveru neskládá z více samostatně fungujících částí, můžeme se také setkat s označením monolitická aplikace. U možnosti thick-client je naopak aplikace na straně klienta a server poskytuje pouze API (viz sekce 4.3.1.1) pro účely komunikace.

Monolitické aplikace nejsou v dnešní době již tak populární, jako tomu bylo v minulosti, nicméně stále mají své uplatnění [13]. Typicky se monolitická architektura využívá u malých aplikací, u kterých se v budoucnu nepředpokládá větší rozšiřování. Jednou z výhod je jistě fakt, že monolitické aplikace se rychleji vyvíjejí a také se jednodušeji testují. Jejich nevýhodou může být naopak složitější údržba a problémové škálování. [12, 13] Jelikož je vyvíjená aplikace zamýšlená pouze pro jednotlivé sportovní kluby a nepředpokládá se potřeba přistupovat a upravovat data jiným způsobem než z nové webové aplikace, tak jsem se i s ohledem na rychlejší vývoj rozhodl využít monolitickou architekturu.

Výběr této architektury však neznamená, že by nemohla být aplikace vnitřně žádným způsobem členěna. Nějaká forma rozdělení je naopak i žádoucí, aby se oddělila odpovědnost a vznikla určitá forma abstrakce nad jednotlivými částmi systému. Díky tomu je například možné vyměnit datové úložiště bez dalších (nebo s minimálním počtem) změn v částech, které se zabývají uživatelským rozhraním nebo obsahují samotnou logiku aplikace. Jednou z možností takového strukturování je využití architektury MVC, která je podrobněji představena v sekci 3.1.1.

3.1.1 MVC architektura

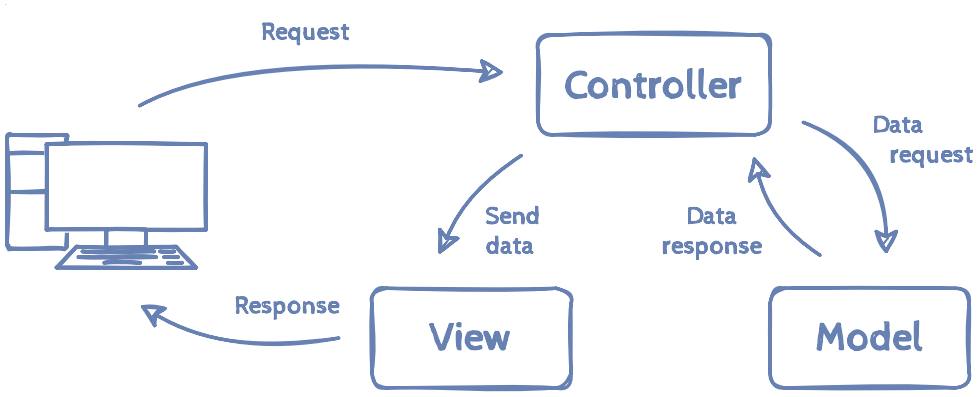
Následující část textu vychází ze zdrojů [14, 15]. Architektura model–view–controller odděluje datový model aplikace od řídící logiky a prezentační vrstvy s uživatelským rozhraním. Nejvíce se její použití rozšířilo u webových aplikací, poněvadž je součástí i mnoha webových frameworků. Na MVC architektuře jsou například založeny frameworky Laravel a Symfony, ASP.NET MVC framework a mnoho dalších.

MVC architektura se tedy skládá ze tří částí. Vrstva modelů definuje s jakými daty bude aplikace pracovat. Může také obsahovat validační pravidla pro tato data, případně i další funkce, které se váží k danému modelu. Takovým příkladem by mohla být funkce na výpočet věku z informace o datu narození. Při využití objektově relačního mapování (ORM) kopírují modely z velké části strukturu tabulek v databázi. Jeden model typicky představuje konkrétní tabulku a jeho atributy představují sloupce dané tabulky.

Druhou vrstvu tvoří pohledy, jež se starají o zobrazení dat uživateli. V případě webových aplikací bývá součástí pohledu šablona obsahující značkovací jazyk HTML, do které jsou následně vkládána data z modelů.

Poslední částí jsou controllery, jež se starají o načítání dat z modelů a jejich aktualizaci v závislosti na vstupu od uživatele. Propojují všechny ostatní vrstvy do funkčního celku.

Obrázek 3.1 Životní cyklus požadavku

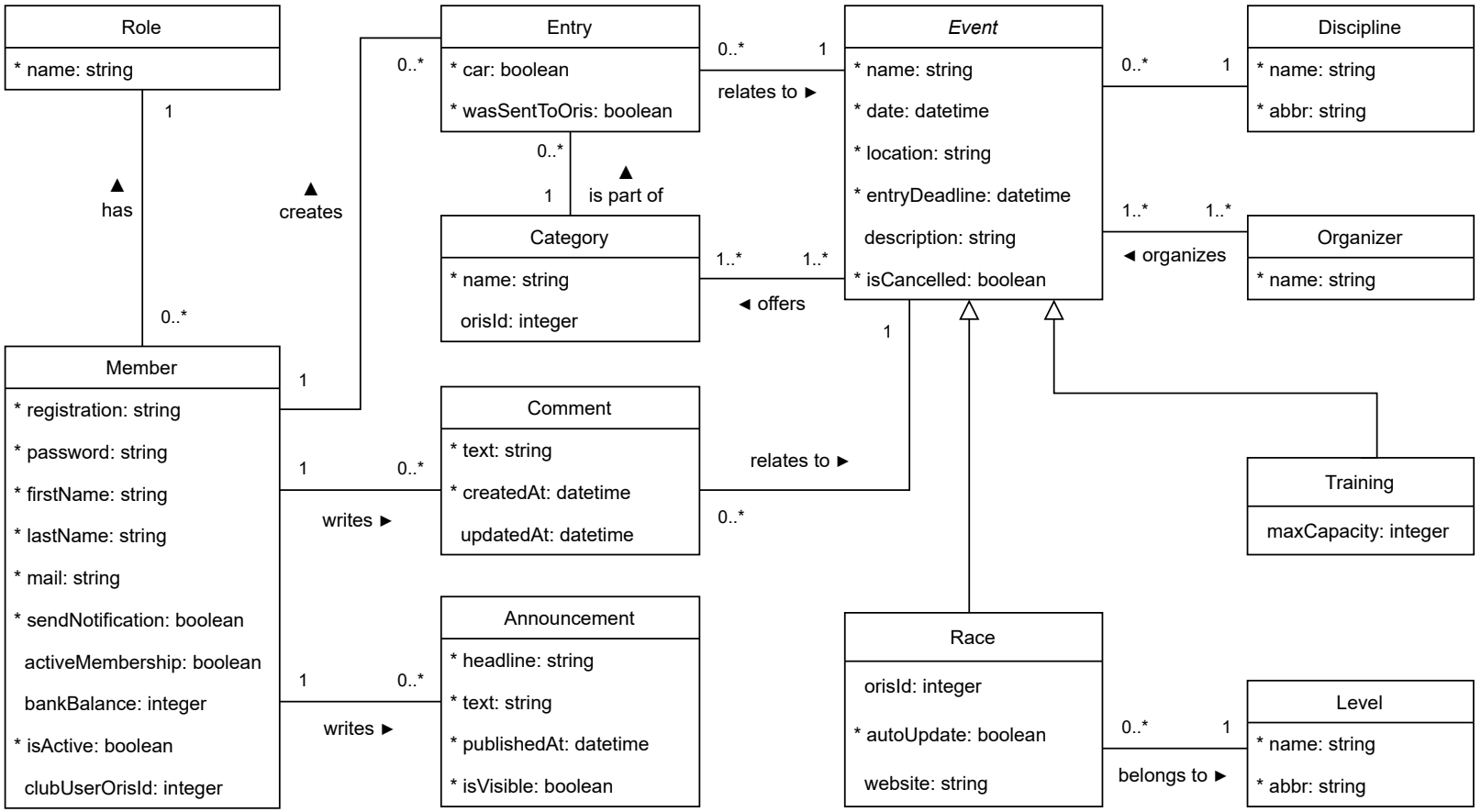


Obrázek 3.1 zobrazuje typický životní cyklus požadavku. Můžeme si například představit situaci, kdy si chce uživatel zobrazit nadcházející události evidované v aplikaci. Požadavek od uživatele nejdříve zachytí část systému nazývaná jako router, která na základě URL adresy rozpozná, jakému controlleru požadavek předat. Daný controller si následně od konkrétního modelu vyžádá uložená data o nadcházejících událostech. Poté, co je od něj získá, předá tato data pohledu, který je již pouze vloží do připravené šablony. Sestavená stránka je nakonec zaslána zpět uživateli.

3.2 Doménový model

V předchozí podkapitole byla představena MVC architektura a byly popsány její vrstvy. Tato se zaměřuje na vrstvu modelů a bude v ní podrobněji představeno s jakými daty bude nová aplikace pracovat. Ke znázornění modelů, jejich vlastností a vazeb mezi nimi je využit doménový model, který je na obrázku 3.2. Doménový model obsahuje u každé třídy (entity) všechny vlastnosti, které se u ní plánují evidovat. Povinné vlastnosti entit jsou označeny symbolem $\star$.

Obrázek 3.2 Doménový model



3.3 Technologie

Další důležité rozhodnutí se při návrhu webové aplikace týká zvoleného programovacího jazyka, neboť je možné vybírat z mnoha možností a každý jazyk má své specifické vlastnosti. Dle W3Techs převládá mezi stránkami, u nichž je znám programovací jazyk na straně serveru, již minimálně od roku 2011 programovací jazyk PHP. Z dostupných dat k 1. dubnu 2022 používá PHP více než 77 % webových stránek. Mezi další využívané jazyky se následně řadí ASP.NET (7,8 %), Ruby (6 %) a Java (3,9 %). [16] S ohledem na nefunkční požadavek **N2**, který vyžaduje vytvoření aplikace v programovacím jazyku PHP, bude pro vývoj využit právě tento jazyk.

3.3.1 PHP

PHP je skriptovací programátorský jazyk, který je určen především pro vývoj webových aplikací. Nejčastěji je PHP kód spouštěn na webovém serveru na základě požadavku od klienta, kterým může být například webový prohlížeč. Po vykonání kódu je výsledek zaslán zpět klientovi. [17, 18] Nejnovější stabilní verze PHP (ke dni 5. 2. 2022) je verze 8.1 [19], avšak pro vývoj tohoto systému bude využita verze 8.0. Toto rozhodnutí bylo učiněno z důvodu chyby ve verzi 8.1, která znemožňovala načítání dat z databáze v závislosti na specifikovaném datumu.

Podobně jako v jiných programovacích jazycích, tak i pro jazyk PHP byla vytvořena celá řada frameworků a knihoven, které by měly usnadnit vytváření nových aplikací. Dle průzkumu za rok 2021 se mezi nejpoužívanější PHP frameworky řadí Laravel a Symfony [20].

3.3.2 Symfony

Symfony se řadí mezi nejstarší PHP frameworky, neboť se prvního vydání dočkalo již v roce 2005 [21]. Od té doby se však neustále vyvíjí a v roce 2021 se dle průzkumu od společnosti JetBrains jednalo o druhý nejpoužívanější PHP framework [20]. Symfony však není pouze samotný framework, ale je to sada znovupoužitelných komponent. To dokazuje i skutečnost, že Laravel a mnoho dalších projektů některé Symfony komponenty využívá. Mezi takové projekty se například řadí známé systémy pro správu obsahu (CMS) Drupal a Joomla! nebo řešení pro internetové obchody PrestaShop. [22]

Pokud bychom porovnávali Laravel a Symfony, tak zjistíme, že to jsou v základu podobné frameworky a liší se spíše v drobnostech. Oba jsou založeny na MVC architektuře, v obou jsou k dispozici nástroje pro objektově relační mapování, šablonovací procesory a podobně. Zásadnější rozdíly bychom například mohli nalézt v možnostech konfigurace. Symfony v porovnání s Laravelem umožňuje pokročilejší nastavení, díky čemuž však může být obtížnější na naučení a vývoj aplikace většinou zabere i více času. U větších projektů však tyto důsledky ustupují do pozadí a vyvíjená aplikace může být přizpůsobena přesně dle potřeb cílových skupin. [23]

Jedním z důvodů, proč jsem si pro implementaci této aplikace vybral Symfony, je skutečnost, že Laravel využívá „magii“ (například magickou metodu `__get()` pro přístup k atributům), která stěžuje statickou analýzu kódu [24]. Dalším faktorem byla již zmíněná větší volnost při využívání Symfony frameworku a také fakt, že s používáním Symfony jsem měl již nějaké zkušenosti.

3.3.3 Databáze

Pro ukládání dat lze využít několik typů databází. V současnosti jsou stále nejpopulárnější relační databáze, které jsou občas z důvodu využívaného dotazovacího jazyka také označovány jako SQL databáze. Jejich základem jsou tabulky, které mají pevně danou strukturu. Jednotlivé řádky tabulky představují uložené záznamy a sloupce obsahují vlastnosti daných záznamů. [25] Dnes se však kromě relačních databází můžeme setkat i s NoSQL databázemi, které mají své uplatnění například při ukládání nestrukturovaných nebo velkých dat [26].

Vyvíjená aplikace bude pracovat s pevně struktorovanými daty, a i proto bude využita relační databáze. Vzhledem k nefunkčnímu požadavku **N2** se bude jednat o databázi MariaDB, která vznikla v roce 2009 jako kopie MySQL kvůli obavám jejího dalšího směřování po odkoupení společností Oracle [27].

3.3.4 Doctrine

Jednou z dalších zvolených technologií pro vývoj nové aplikace je Doctrine. Výběr Doctrine je úzce spojen s výběrem frameworku, neboť již samotné Symfony poskytuje možnost integrace s touto komponentou. Jedná se o sadu knihoven, které umožňují objektově relační mapování (ORM) a které nabízí objektově orientované API (a řadu dalšího) pro manipulaci s daty a se strukturou databáze. [28, 29]

Při vývoji aplikací v objektově orientovaných jazycích se často využívají objekty, které nám sdružují vlastnosti předmětů podobně jako tomu je v reálném světě. Tento přístup k datům však neodpovídá tomu, jakým způsobem se data ukládají v relačních databázích. V nich je jeden konkrétní objekt reprezentován jako jeden řádek s odpovídajícím počtem sloupců. Rozdíly mezi těmito přístupy vyrovnává právě ORM, které zajišťuje automatickou konverzi dat mezi objekty a relační databází. [28]

Další částí Doctrine je již zmíněná abstraktní vrstva nad databází (DBAL). Díky ní nemusíme k databázi přistupovat napřímo, ale můžeme využít připravené metody, které poskytují jednotný přístup k datům a struktuře databáze nezávisle na využívaném řešení. V současnosti jsou podporovány všechny nejpoužívanější typy relačních databází zahrnující MySQL, Oracle, PostgreSQL, SQLite a Microsoft SQL Server. [29]

3.3.5 Bootstrap

Poslední technologií, která bude představena v této podkapitole, je Bootstrap. Jedná se svobodný a otevřený software, který usnadňuje tvorbu uživatelského rozhraní ve webových aplikacích. Zahrnuje šablony napsané v HTML, CSS a další rozšíření implementované v jazyku JavaScript. Šablony jsou k dispozici pro všechny základní HTML elementy a umožňují jednoduše vytvářet responzivní uživatelské rozhraní. [30]

Kapitola 4
Implementace

Tato kapitola se zabývá procesem implementace nové webové aplikace. Nejdříve budou představeny nástroje, které jsem při vývoji využíval, a bude popsána struktura projektu. Následně budou přiblíženy vybrané části implementace a nakonec budou diskutovány další možná rozšíření.

4.1 Použité nástroje

Při implementaci jsem využíval několik nástrojů, které zefektivňují práci při vytváření a následných úpravách aplikací. Prvním takovým nástrojem je verzovací systém Git. Jde o svobodný a otevřený software, který umožňuje zaznamenávání jednotlivých revizí během vývoje aplikací. Jeho síla se nejvíce projeví v případě, kdy na projektu pracuje více vývojářů a je potřeba nějakým způsobem synchronizovat jejich práci. Pro své vlastnosti je využíván i největšími technologickými společnostmi jako jsou Google, Facebook nebo Microsoft. [31]

Git byl na tomto projektu využit ve spojení s webovou službou GitHub, která nabízí bezplatné umístění repozitáře spolu s dalšími souvisejícími funkcemi jako je ticketovací systém nebo možnost tvorby stránek s dokumentací. Právě i na webové službě GitHub budou po obhájení práce dostupné kompletní zdrojové kódy aplikace pro případné zájemce.

Dalším využitým nástrojem je program PhpStorm od firmy JetBrains. Jedná se o chytrý editor zdrojového kódu, který však obsahuje spoustu dalších podpůrných funkcí, které zjednodušují vývoj webových aplikací. Mezi tyto funkce se například řadí analýza zdrojového kódu, podpora pro ladění a testování a spousta dalších.

4.2 Struktura projektu

Projekt následuje standardní adresářovou strukturu Symfony aplikací, a tedy orientace by lidem, kteří se již s nějakou aplikací vytvořenou v Symfony frameworku setkali, neměla dělat problém.

- `config` ..... konfigurační soubory aplikace
- `public` ..... kaskádové styly, kód v jazyku JavaScript, ikony
- `src`
  - `Controller` ..... controllery
  - `Form` ..... formuláře
  - `Entity` ..... modely
- `templates` ..... šablony stránek
- `composer.json` ..... soubor se seznamem závislostí

4.3 Ukázky vybraných částí

V následující sekci je popsáno několik vybraných částí z implementované aplikace, které jsou pro vytvořený systém důležité nebo jsou nějakým způsobem zajímavé. Jako první bude představen způsob komunikace s informačním systémem Českého svazu orietačních sportů ORIS, poté bude popsána tvorba formuláře pro přidání nové události a jako poslední bude demonstrován způsob zobrazení aplikace na různých typech klientských zařízení.

4.3.1 Komunikace s IS ORIS

Jednou z nových funkcí oproti současně používanému systému je napojení na IS ORIS. Konkrétně je v nové aplikaci umožněno automatické načítání informací o závodu evidovaného v celorepublikovém systému a také odesílání evidovaných přihlášek. Propojení je dosaženo s využitím dostupného API, které je popsáno v sekci 4.3.1.3.

4.3.1.1 Webové API

Zkratku API lze do češtiny přeložit jako „rozhraní pro programování aplikací“. Z tohoto slovního spojení však člověk, který se v oblasti informačních technologií nepohybuje, pravděpodobně stále nebude mít moc představu, k čemu to API vlastně je. V případě webových aplikací se jedná o rozhraní, které umožňuje komunikaci mezi různými systémy nebo různými částmi systému. Jedna strana musí toto rozhraní, jež se může skládat z více koncových bodů (endpointů), poskytovat a druhá následně může na některý z těchto endpointů posílat povolené instrukce, a tím s první interagovat. V dnešní době je pro API orientovaná na data nejčastěji využívána architektura REST, avšak informační systém ORIS své API založil na vlastní architektuře, jejíž popis je součástí sekce 4.3.1.3. [32]

4.3.1.2 HTTP

S webovými API je úzce spojen protokol HTTP. Jedná se o protokol aplikační vrstvy, který byl navržen pro komunikaci mezi webovým serverem a webovým prohlížečem. Je založen na principu klient–server, kdy klient iniciuje vznik spojení za účelem zaslání požadavku a následně čeká na odpověď od serveru. Každý HTTP požadavek mimo dalších informací obsahuje i název metody, která může specifikovat, jakým způsobem bude požadavek zpracován. Mezi nejdůležitější metody patří metoda GET, která slouží pro získání dat, a metoda POST, jež se nejčastěji využívá při odesílání dat. [33]

4.3.1.3 ORIS API

I přes zvyklosti uvedené v sekci 4.3.1.2 probíhá komunikace s ORIS API pouze pomocí GET požadavků. Každý požadavek musí obsahovat query parametr s názvem metody a názvem formátu, ve kterém chceme data obdržet. V závislosti na vybrané metodě může být nutné uvést i další query parametry, které jsou pro ni specifické. V současné době je podporováno 28 různých metod a 2 datové formáty (JSON a XML). Příklad požadavku na získání všech klubů v Českém svazu orientačních sportů ve formátu JSON můžeme vidět na výpisu kódu 4.1.

Výpis kódu 4.1 Požadavek na získání všech klubů v ČSOS

```
GET /API/?format=json&method=getCSOSClubList HTTP/1.1
Host: oris.orientacnisporty.cz
Accept: */*
```

V nové aplikaci jsou využity tři z nabízených metod. Konkrétně se jedná o metody `getEvent`, `getEventList` a `createEntry`. Metoda `getEvent` se využívá pro načtení informací o závodě při vytváření nové události, `getEventList` se volá při hromadné kontrole času uzávěrky přihlášek a metoda `createEntry` se podílí na odesílání přihlášek do IS ORIS.

4.3.1.4 Importování nového závodu

Jak již bylo zmíněno, při přidávání nového závodu je možné využít automatické načítání informací z celorepublikového systému. Procesy vykonávané v rámci této funkcionality lze rozdělit na 4 logické fáze:

1. Získání ORIS ID závodu od uživatele
2. Odeslání požadavku na ORIS API
3. Zpracování přijatých dat
4. Načtení zpracovaných dat do formuláře

V rámci první fáze uživatel vyplní vstupní pole pro ORIS ID ve formuláři a klikne na tlačítko *Importovat závod*. Kliknutím na uvedené tlačítko se spustí mechanismus, který na ORIS API odešle požadavek na zaslání dat týkající se závodu s odpovídajícím ORIS ID. Příklad validního požadavku je na výpisu kódu 4.2 a následná odpověď na výpisu kódu 4.3.

Výpis kódu 4.2 Požadavek na získání informací o závodu

```
GET /API/?format=json&method=getEvent&id=6734 HTTP/1.1
Host: oris.orientacnisporty.cz
Accept: */*
```

Výpis kódu 4.3 Začátek odpovědi na požadavek na získání informací o závodu

```
HTTP/1.1 200 OK
Date: Sun, 17 Apr 2022 12:06:08 GMT

{
    "Method": "getEvent",
    "Format": "json",
    "Status": "OK",
    "ExportCreated": "2022-04-17 14:06:09",
    "Data": {
        "ID": "6734",
        "Name": "Oblastní žebříček",
        "Date": "2022-04-02",
        "Place": "Lomnice nad Popelkou",
        "Map": "Tábor, 1:10000",
```

Z přijatých dat je následně pomocí speciální třídy sestaven nový objekt typu `Race`, jenž je poté předán formuláři. Ten si již jen načte data z tohoto objektu a výsledná stránka včetně předvyplněných hodnot je zaslána uživateli.

4.3.1.5 Odesílání přihlášek

Pro odeslání přihlášek do IS ORIS je zapotřebí poslat GET požadavek, který kromě názvu metody a formátu dat obsahuje i query parametry `username`, `password`, `class` a `clubuser`. Takovýto požadavek je potřeba poslat pro každou evidovanou přihlášku, poněvadž systém ORIS nepodporuje posílání informací o více přihláškách najednou.

Parametry `username` a `password` slouží k autorizaci požadavku a jedná se o jméno a heslo, pomocí kterých uživatel přistupuje do IS ORIS. Tento způsob autentizace však není doporučovaný z několika důvodů. Prvním z nich je skutečnost, že na straně serveru bývá požadovaná URL včetně query parametrů ukládána do přístupových záznamů. Ke stejné situaci dochází i na straně klienta, kde je URL ukládána do historie prohlížeče. Další problém může vzniknout, pokud by stránky využívaly JavaScript, CSS nebo jiné zdroje třetích stran, protože požadavek na tyto zdroje může obsahovat hlavičku `Referer` s kompletní URL. Ani jedna z těchto vlastností není žádoucí, a proto by se měly využívat jiné dostupné možnosti autentizace. [34]

Dalším povinným parametrem je `class`, který představuje identifikátor kategorie v IS ORIS a který je uložen v databázi vytvořené aplikace v entitě `Category`. Posledním povinným query parametrem je `clubuser`, jenž identifikuje přihlašovaného uživatele. I tento identifikátor byl nakonec uložen do databáze vytvořeného systému, poněvadž pro jeho získání by bylo potřeba poslat dva další požadavky. Pro každou přihlášku by tedy bylo potřeba celkově poslat tři požadavky místo jednoho. Příklad požadavku na vytvoření přihlášky pro člena klubu s ID *12007* do kategorie *156818* je na výpisu kódu 4.4.

Výpis kódu 4.4 Požadavek na vytvoření přihlášky

```
GET /API/?format=json&method=createEntry&username=Paukert&password=XXXX␣
↪  &clubuser=12007&class=156818
↪  HTTP/1.1
Host: oris.orientacnisporty.cz
Accept: */*
```

4.3.2 Formuláře

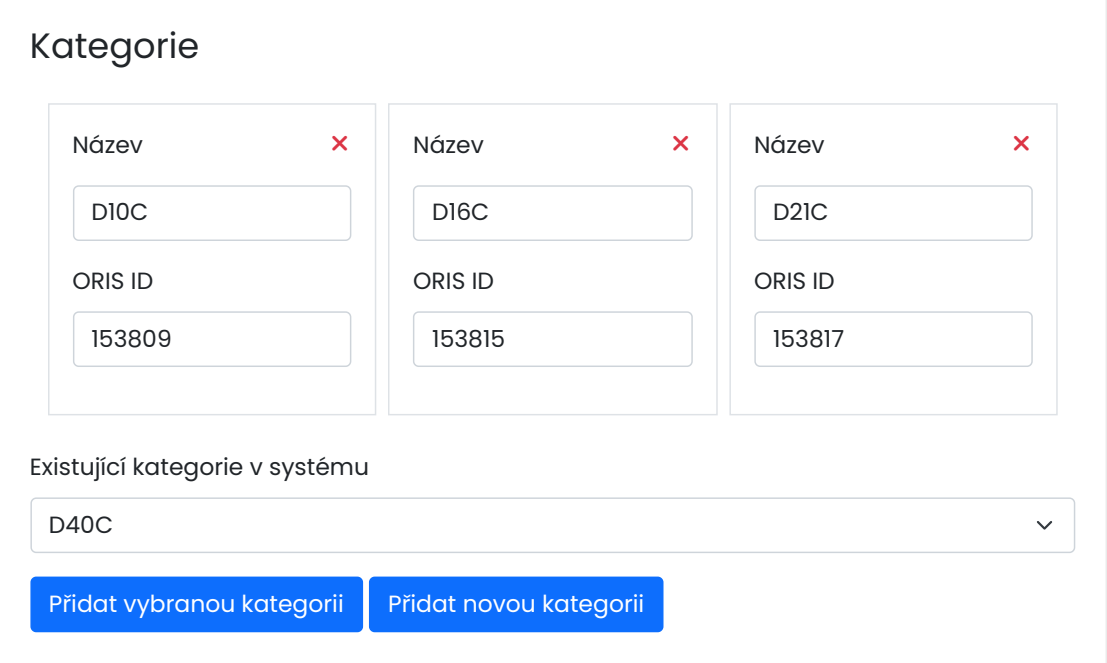
Jednou z komplikovanějších částí implementace bylo vytvoření formulářů pro přidávání a úpravu událostí. Symfony poskytuje propracovaný systém pro práci s formuláři, který zahrnuje jejich vytváření, renderování, zpracování nebo i validaci. Součástí zpracování formuláře je i možnost jeho propojení s konkrétním objektem, díky čemuž se vyplněné hodnoty do tohoto objektu automaticky propisují. Validace vyplněných hodnot probíhá jak na straně klienta díky speciálním atributům u tagu `input`, tak na straně serveru, kde je řešena s využitím Symfony validátorů, konkrétně pomocí „Symfony Validation Constraints“.

Pro vlastnosti objektu `Event`, jež jsou typu `string`, `integer` nebo `datetime`, nebylo vytváření formuláře nijak speciální. Zajímavější bylo vymyslet, jakým způsobem do formuláře přidat vlastnosti `categories` a `organizers`, která jsou v databázi reprezentované vazbou M:N mezi entitami `Event` a `Category`, respektive `Event` a `Organizer`. Uvedený typ vazby M:N v případě entit `Event` a `Category` znamená, že událost může mít libovolné množství kategorií a naopak konkrétní kategorie může být přiřazena k libovolnému počtu událostí.

Po zvážení dostupných možností jsem se rozhodl vytvořit formulář, ve kterém je možné dynamicky přidávat a odebírat nové kategorie a organizátory. Nové položky lze do formuláře přidat vybráním již existujícího záznamu registrovaného u jiné události, nebo vytvořením zcela nové položky. Finální vzhled části formuláře obsahující sekci s kategoriemi je zobrazen na obrázku 4.1.

Logiku ukládání a odebírání položek nebylo třeba vymýšlet od začátku, poněvadž Symfony formuláře obsahují podporu pro přidávání libovolného počtu položek ke kontrétní vlastnosti. Pro zprovoznění této funkcionality však bylo nezbytné přidat vlastní kód v jazyku JavaScript, který zařídí samotné přidávání a odebírání položek z objektového modelu dokumentu (DOM).

Obrázek 4.1 Část formuláře na přidání události



V souvislosti s dynamickým přidáváním nových položek bylo nutné navíc zamezit jejich duplikování v databázi v případě shody hodnot jejich vlastností.

4.3.3 Responzivní design

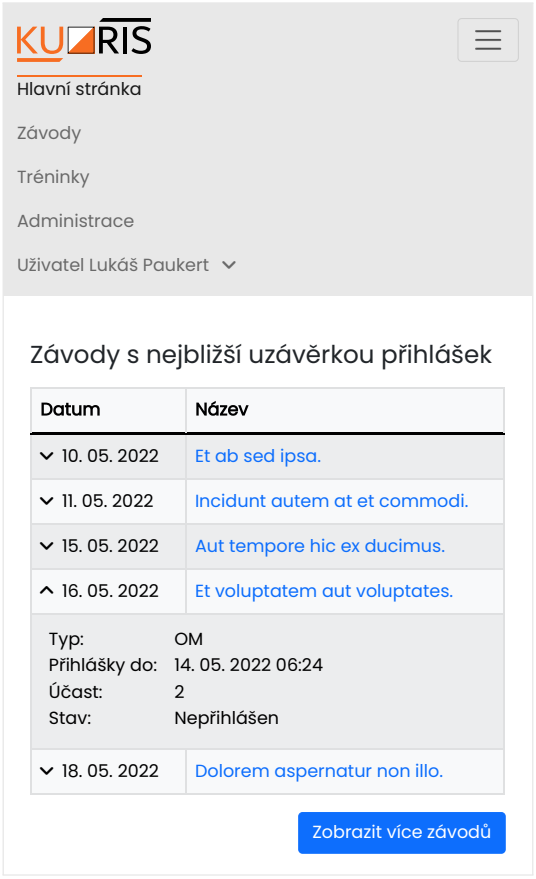
Jednou z důležitých vlastností nové aplikace je nepochybně responzivní design, díky kterému je vzhled systému optimalizován pro klientské zařízení s různým rozlišením. S ohledem na skutečnost, že v dnešní době přistupuje na web nejvíce lidí z mobilních zařízení [5], je tato vlastnost o to zásadnější. Na obrázcích 4.2 a 4.3 je vidět způsob zobrazení hlavní stránky a stránky s formulářem pro přidání události na chytrém telefonu. Další ukázky z vytvořené aplikace jsou umístěny v příloze A.

Vytvořený responzivní vzhled stojí z velké části na frameworku Bootstrap, který již byl představen v sekci 3.3.5. Příkladem části, jež Bootstrap kompletně neřeší, jsou tabulky. Pro ty byl s využitím jazyku JavaScript a CSS vytvořen mechanismus, který u zařízení s menším rozlišením postupně skrývá jednotlivé sloupce a naopak umožňuje zobrazení informací z těchto sloupců „rozbalením“ odpovídajícího řádku. Tato funkcionalita je demonstrována na obrázku 4.2.

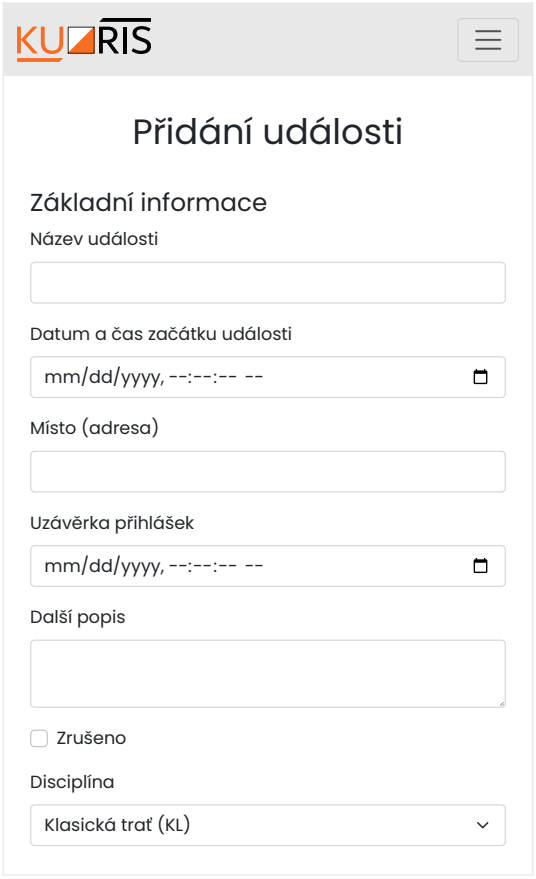
4.4 Možná rozšíření

Výslednou aplikaci lze v budoucnu rozšířit a vylepšit mnoha způsoby. Jeden z velkých nedostatků se týká evidence stavu klubových kont. Dnes je jejich stav po určitém časovém období zasílán vedoucím klubu na e-mailovou adresu ve formě XLSX souboru. V současné verzi aplikace je následně nutné manuálně zadat hodnotu klubového konta u každého ze členů dle zaslaného souhrnu. Tento nedostatek by mohl být například napraven možností importu stavu klubových kont v CSV formátu. Jako další řešení by se nabízelo automatické odečítání ze zůstatku konta v závislosti na evidovaných přihláškách, avšak tento přístup by stále vyžadoval ruční zásahy

Obrázek 4.2 Hlavní stránka



Obrázek 4.3 Formulář pro přidání události



ve chvíli, kdy si člen chce přidat prostředky na své osobní konto.

Mezi další možná vylepšení se jistě může zařadit zasílání informačních e-mailů při přidávání nových událostí, zobrazování statistik pro trenéry a administrátory nebo pokročilejší propojení s IS ORIS. Konkrétně by například mohla být přidána možnost výběru přihlášek, které se mají do celorepublikového systému odeslat, nebo podpora pro jejich úpravu a odstranění. V neposlední řadě by bylo dobré systém připravit pro lokalizaci do dalších jazyků.

Kapitola 5
Uživatelské testování

Tato kapitola se zabývá testováním vytvořeného prototypu aplikace. Jak už název napovídá, tak při uživatelském testování ověřují použitelnost aplikace samotní uživatelé. Hlavním cílem je zjistit, zda je uživatelské rozhraní dostatečně intuitivní, a získat případné návrhy na jeho vylepšení.

Testování proběhlo za osobní přítomnosti a zúčastnily se ho celkem 4 osoby včetně předsedy klubu KOB Ústí nad Orlicí. Své zastoupení zde měly různé generace a byli přizváni jak uživatelé současného systému, tak i lidé, kteří s ním nikdy nepracovali. Každý tester byl na začátku seznámen s podstatou systému a následně obdržel sadu úkolů ze sekce 5.1, které za dozoru postupně procházel.

5.1 Testovací scénáře

5.1.1 Registrace a přihlášení do systému

Modelová situace

Představte si, že jste nový člen klubu a chcete získat přístup do webové aplikace.

Postup

1. Přejděte na stránku s registračním formulářem, vyplňte požadované údaje a klikněte na tlačítko *Registrovat*.
2. Vyčkejte na schválení vytvořeného účtu administrátorem.
3. Pomocí dříve zadaných údajů se přihlaste do aplikace.

5.1.2 Změna kontaktního e-mailu a hesla

Modelová situace

Na váš oblíbený internetový obchod zaútočili hackeři a ukradli údaje ke stovkám účtů. Bohužel e-shop využíval k hashování hesel zastaralý mechanismus a byl vám z tohoto důvodu ukraden i e-mail, ke kterému jste používali stejné heslo. Nyní si chcete v systému nastavit novou e-mailovou adresu a změnit přihlašovací heslo.

Postup

1. Přejděte pomocí hlavní nabídky na stránku *Nastavení*.
2. Zadejte svou novou e-mailovou adresu a heslo a změny uložte.

5.1.3 Přidání nového závodu a jeho následná úprava

Modelová situace

Představte si, že jste trenér klubu KOB Ústí nad Orlicí a chcete přidat do systému nový závod Východočeského poháru.

Postup

1. Přejděte do administrace systému a zvolte možnost pro přidání nového závodu.
2. Využijte možnosti načtení informací o závodu ze systému Českého svazu orientačních sportů ORIS, ve kterém má závod přidělený identifikátor *6738*.
3. Přidejte tuto událost se všemi načtenými informacemi a názvem *7. kolo VčP* do systému.
4. Na následně zobrazené stránce jste si všimli chybného data uzávěrky přihlášek, vraťte se do administrace a změňte ho na *7. 5. 2022 23:59*.

5.1.4 Přihlášení na závod a napsání komentáře

Modelová situace

Jako člen klubu se chcete přihlásit na nadcházející závod Východočeského poháru a napsat k němu komentář.

Postup

1. Z hlavní stránky systému nebo ze seznamu všech závodů si zobrazte detailní informace o závodu s názvem *7. kolo VčP*.
2. Vyberte si požadovanou kategorii a přihlášení potvrďte.
3. Napište k této události libovolný komentář.

5.2 Průběh a výsledky

První modelová situace ze scénáře 5.1.1 se zabývá registrací a přihlášením do systému. Všichni testeři se pomocí položky v hlavní nabídce případně s využitím tlačítka na uvítací obrazovce bez problému dostali na stránku s registračním formulářem. Samotný proces registrace do systému a i následného přihlášení považovali respondenti za jednoduchý.

Scénář 5.1.2 měl za úkol ověřit, zda uživatelé bez potíží najdou sekci s nastavením, kde si mohou změnit některé ze svých údajů. Bod ve scénáři zmiňuje, aby přešli pomocí hlavní nabídky na stránku *Nastavení*. Jeden z testerů, jehož považuji za průměrně zdatného uživatele počítače, chvilku přemýšlel, jak se do nastavení dostat, nicméně i jemu se to nakonec podařilo. Pro rozbalovací menu byla z tohoto důvodu nakonec zvolena jiná ikona, kterou oslovení respondenti hodnotí jako názornější.

Přidání a následná úprava závodu byly testovány v rámci scénáře 5.1.3. Rozložení administrace přišlo respondentům přehledné a s vyhledáním možnosti pro přidání nového závodu neměl nikdo problém. Načtení informací o události z IS ORIS nedělalo uživatelům taktéž obtíže a i následná úprava závodu není dle jejich slov složitá.

Poslední modelová situace uvedená ve scénáři 5.1.4 ověřovala intuitivnost uživatelského rozhraní při přihlašování se na události. Přechod na stránku s detailními informacemi o závodu nečinil potíže žádnému z respondentů. Při samotném přihlašování na událost jeden z testerů navrhl, aby se mu v nabídce zobrazovaly pouze kategorie, jež jsou pro něj relevantní (tj. aby se mužům nezobrazovaly kategorie pro ženy a aby proběhlo filtrování na základě věku přihlašované osoby). Tuto funkcionalitu jsem sám plánoval v budoucnu naimplementovat s využitím ORIS

API, které nabízí metodu `getValidClasses` vracející seznam platných kategorií pro konkrétního uživatele a závod. Kromě této připomínky hodnotili respondenti proces přihlašování a psaní komentářů jako jednoduchý.

Aplikaci oslovení testeři až na dvě výjimky popsané výše hodnotili jako přehlednout a uživatelsky přívětivou. Jedna ze dvou připomínek již byla zapracována a druhá bude vyřešena v rámci budoucího vylepšení integrace se systémem Českého svazu orientačních sportů ORIS.

Kapitola 6
Nasazení

Proces zprovoznění aplikace může být mnohdy poměrně náročný a problémy způsobené vzájemnou nekompatibilitou jednotlivých verzí nástrojů rozhodně dokáží tento proces velice znepříjemnit. Podobným komplikacím můžeme v dnešní době naštěstí předcházet například díky balíčkovacím systémům a nástroji Docker, který bude představen v podkapitole 6.1. Následně bude také popsáno, jak lze právě s jeho pomocí spustit vytvořenou aplikaci a taktéž bude přiblíženo, jakým způsobem je možné aplikaci zprovoznit na současně využívaných hostingových službách.

6.1 Docker

V dnešní době je Docker využíván jak při vývoji, tak při samotném nasazování aplikací do produkčního prostředí. Jedná se o nástroj, který poskytuje odlehčenou formu virtualizace. K tomu využívá takzvané kontejnery, které zahrnují samotný kód aplikace, všechny její závislosti a i její nastavení. Na rozdíl od klasických virtualizací tyto kontejnery neobsahují operační systém a přímo pomocí Docker engine komunikují s hostitelským operačním systémem. [35]

Díky této vlastnosti jsou kontejnery nástroje Docker méně náročné na hardware než klasické virtualizace. Mezi další důležité vlastnosti patří větší bezpečnost, poněvadž aplikace z různých kontejnerů jsou od sebe izolovány, a přenositelnost, která zaručuje, že aplikace budou fungovat na různých strojích s nainstalovaných nástrojem Docker vždy stejně. [35]

Ve chvíli, kdy se v rámci jednoho projektu využívá více různých kontejnerů, může s jejich správou pomoci nástroj Docker Compose, který se stará o jejich orchestraci. Jako hlavní konfigurační soubor využívá `docker-compose.yml`, v němž se definují a nastavují jednotlivé služby. Ty je následně možné spouštět a vypínat pomocí jediného příkazu.

6.2 Požadovaný software

Pro zprovoznění aplikace dle postupu uvedeného v podkapitole 6.3 je zapotřebí mít nainstalován následující software:

- Docker (popsán v předchozí sekci 6.1)
- Docker Compose (nadstavba nad nástrojem Docker, taktéž popsána v sekci 6.1)

6.3 Postup pro zprovoznění

Díky nástroji Docker je zprovoznění aplikace velice jednoduché. Po provedení níže uvedených kroků bude systém dostupný na adrese `https://localhost/`.

1. sestavení kontejnerů příkazem `docker-compose build --pull --no-cache`
   - instalace závislostí pomocí balíčkovacího systému Composer proběhne automaticky
2. spuštění kontejnerů pomocí příkazu `docker-compose up -d`
   - databázové migrace se spustí také automaticky
   - v rámci migrací bude do databáze vložen administrátor se jménem `KUO9801` a heslem `KUO9801` a i několik dat pro číselníky disciplín a úrovní
3. testovací data obsahující další členy a události včetně přihlášek lze vygenerovat pomocí příkazu `docker-compose exec php php bin/console doctrine:fixtures:load --append`
4. všechny kontejnery lze naopak zastavit pomocí `docker-compose down --remove-orphans`

6.4 Webhosting

Již od počátku bylo cílem vyvinout aplikaci, která by se mohla provozovat na aktuálně využívaných hostingových službách, jakými jsou sdílený webhosting a databázový server MariaDB od společnosti WEDOS. Z tohoto důvodu byl i specifikován nefunkční požadavek N2, který tento cíl reflektoval. S provozem webové aplikace na těchto službách však přichází řada úskalí, neboť k webhostingu například není možné přistupovat přes SSH protokol. Jenom toto omezení způsobuje řadu komplikací, poněvadž je tímto například znemožněno instalovat závislosti pomocí balíčkovacího systému Composer nebo spouštět standardním způsobem databázové migrace.

Pro zprovoznění vytvořené aplikace na produkčním serveru bylo tedy třeba provést několik dalších úprav a nastavení. Prvním krokem bylo přidání závislosti `symfony/apache-pack`, jež zajistila, že systém bude fungovat na webovém serveru Apache, který využívají sdílené webhostingy od společnosti WEDOS. S ohledem na omezení, která jsou kladena na tyto webhostingy, muselo být z vygenerovaného `.htaccess` souboru odebráno nepodporované nastavení, jež je na výpisu kódu 6.1. [36]

Výpis kódu 6.1 Nepodporované nastavení ve vygenerovaném .htaccess souboru

```
<IfModule mod_negotiation.c>
    Options -MultiViews
</IfModule>
```

Další úpravy kódu bylo třeba udělat ve spojení s voláním PHP funkce `putenv()`, která je na webhostingu zakázána. Poslední nutné nastavení spočívalo ve vytvoření dalšího `.htaccess` souboru s obsahem uvedeném na výpisu kódu 6.2 v kořenové složce projektu. Pravidlo uvedené v tomto souboru zajišťuje přesměrování požadavků na přístupový bod naší aplikace, kterým je soubor `index.php` umístěný ve složce `public`. [36]

Výpis kódu 6.2 Obsah .htaccess souboru umístěného v kořenové složce projektu

```
RewriteEngine On
RewriteCond %{REQUEST_URI} !^public
RewriteRule ^(.*)$ public/$1 [L]
```

Jak již bylo zmíněno, k webhostingu není možné přistupovat přes SSH protokol. Z tohoto důvodu je na něj nutné nahrát všechny soubory vytvořené aplikace včetně jejích závislostí ručně přes FTP protokol. Posledním nutným úkonem je vytvoření odpovídající struktury v databázi a přidání dat pro číselníky. Toho může být docíleno například spuštěním databázových migrací v lokálním prostředí, vytvořením zálohy této databáze a následným importováním této zálohy do produkčního prostředí.

Kapitola 7
Závěr

Cílem práce bylo navrhnout a implementovat prototyp webové aplikace, jež bude sloužit sportovním klubům, a to především těm, které se věnují orientačnímu běhu.

Na základě cílů práce byla nejprve provedena analýza, která zahrnovala popis již existujících řešení a specifikaci funkčních a nefunkčních požadavků. Následoval návrh nové aplikace a výběr vhodných technologií. S ohledem na omezení, jež byla stanovena v rámci nefunkčních požadavků, byl systém vyvinut v programovacím jazyku PHP s využitím frameworku Symfony.

Vytvořená aplikace umožňuje registraci a správu uživatelů a evidenci událostí, na které je možné se přihlašovat a psát k nim komentáře. S využitím dostupného API informačního systému Českého svazu orientačních sportů byl vytvořen mechanismus pro snadné importování závodů do nově naprogramované aplikace. Do systému ORIS lze naopak z vyvinutého systému odeslat všechny přihlášky evidované u konkrétní události.

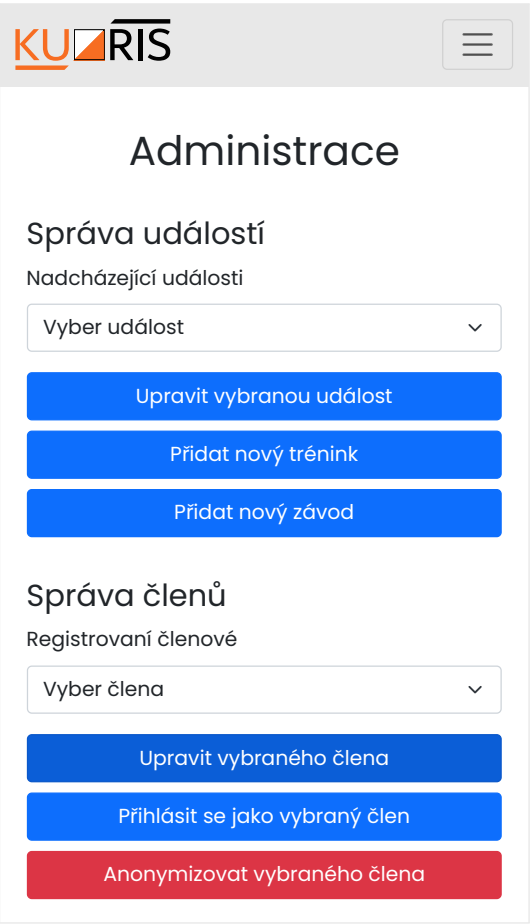
Po dokončení implementace byla webová aplikace podrobena uživatelskému testování a na základě výsledků byly navrženy drobné úpravy. Systém byl zároveň připraven pro nasazení místo aktuálně využívané aplikace, kterou by jistě mohl již v současném stavu nahradit.

Všechny cíle této práce byly splněny. Zdrojové kódy výsledné aplikace budou veřejně přístupné na internetové službě GitHub a kdokoliv bude mít možnost si vytvořené řešení stáhnout a případně upravit podle svých potřeb. Osobně plánuji ve vývoji nadále pokračovat i po odevzdání této práce. Vytvořenou aplikaci je možné například vylepšit o možnost importování stavu osobních kont, zasílání informačních e-mailů při přidávání nových událostí, zobrazování statistik pro trenéry a administrátory nebo o větší integraci s informačním systémem ORIS.

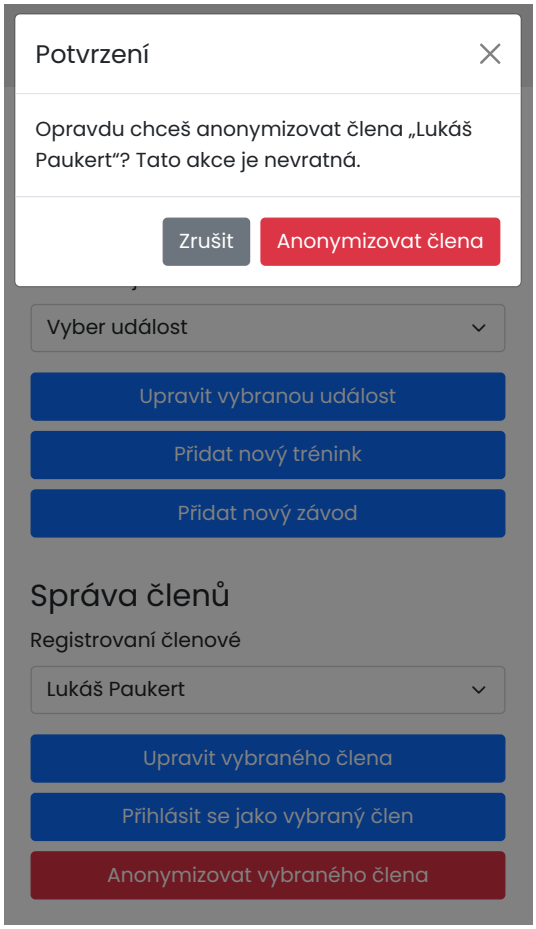
Příloha A

Ukázky z vytvořené aplikace

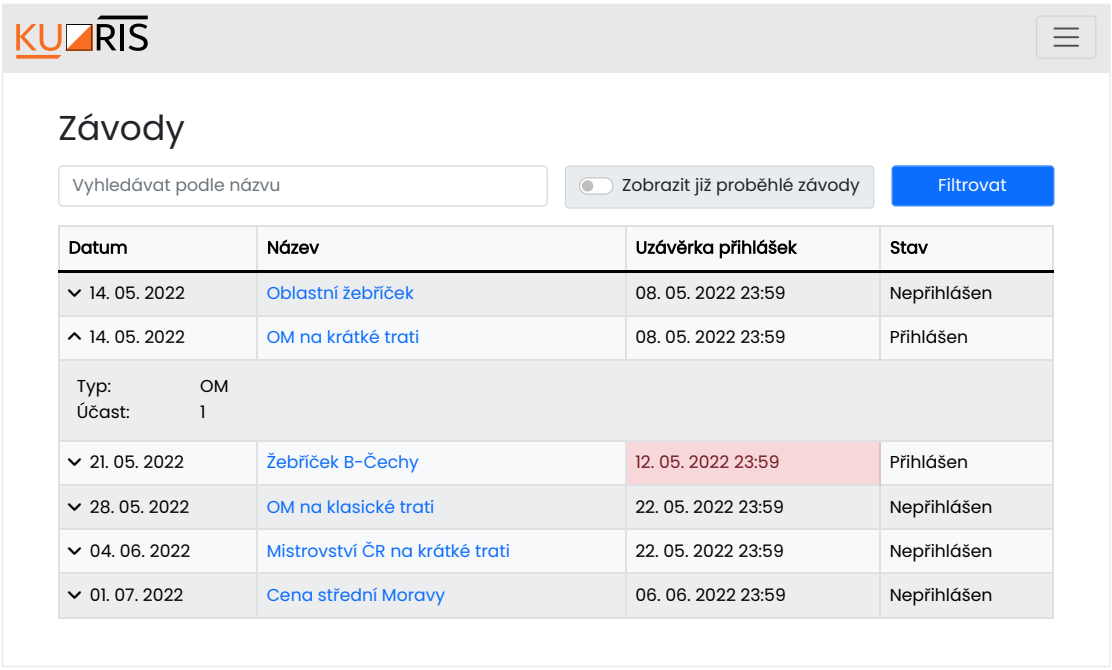
Obrázek A.1 Administrace



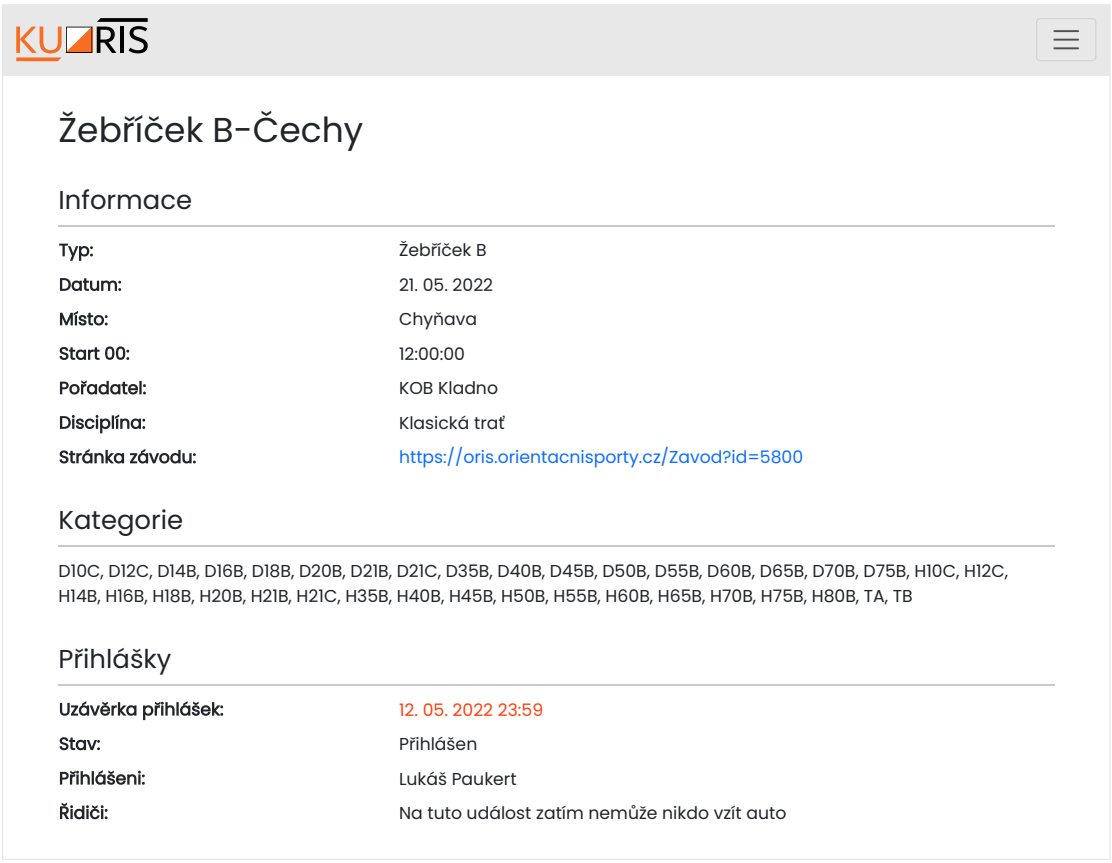
Obrázek A.2 Potvrzovací dialog



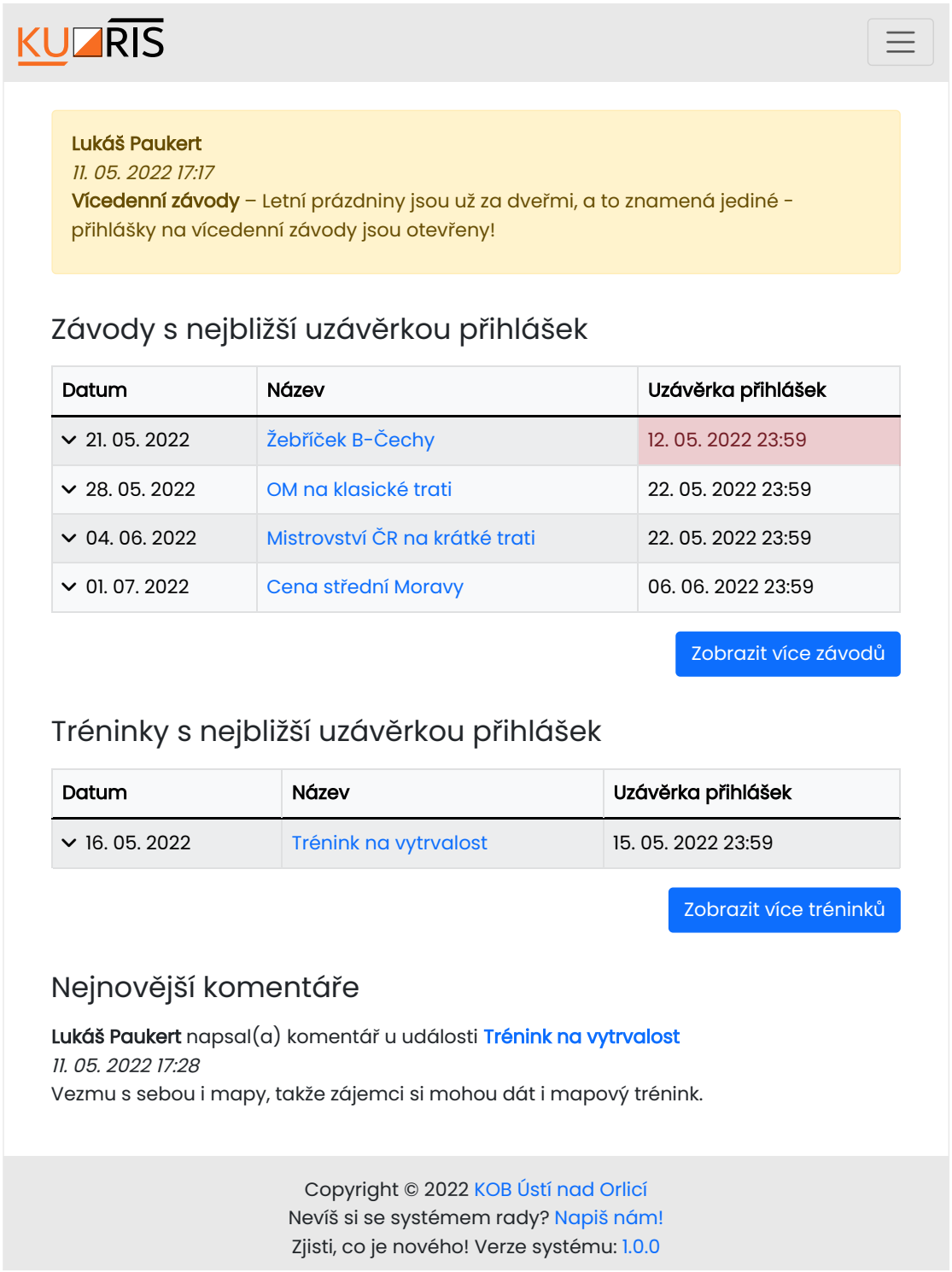
Obrázek A.3 Stránka se závody



Obrázek A.4 Stránka s detailními informacemi o závodu



Obrázek A.5 Hlavní stránka



Bibliografie

1. PANTLÍK, Dalibor. *Informační systém sportovního klubu Taekwonda* [online]. Brno, 2021 [cit. 2021-12-06]. Dostupné z: `https://is.muni.cz/th/vz3sr/`. Bakalářská práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedoucí práce Jaromír PLHÁK.
2. JIRKA, Michal. *Informační systém pro sportovní klub* [online]. Praha, 2016 [cit. 2021-12-06]. Dostupné z: `https://dspace.cvut.cz/handle/10467/66510`. Bakalářská práce. ČVUT v Praze, Fakulta informačních technologií, Katedra softwarového inženýrství. Vedoucí práce Michal ŠOCH.
3. KRUG, Jan. *Informační systém pro sportovní kluby* [online]. Praha, 2013 [cit. 2021-12-06]. Dostupné z: `https://vskp.vse.cz/35996_informacni_system_pro_sportovni_kluby`. Diplomová práce. VŠE v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií. Vedoucí práce Zbyněk ŠLAJCHRT.
4. *About phpBB* [online]. 2022. [cit. 2021-04-23]. Dostupné z: `https://www.phpbb.com/about/`.
5. ENGE, Eric. *Mobile vs. Desktop Usage in 2020* [online]. 2021. [cit. 2021-12-06]. Dostupné z: `https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage`.
6. ESPORTS, S.R.O. *KIS - Klubový Informační Systém, online správa sportovního oddílu* [online]. 2021. [cit. 2021-11-11]. Dostupné z: `https://www.esports.cz/kis/`.
7. *Klubový informační systém je oporou pro sportovní kluby* [online]. 2020. [cit. 2021-11-11]. Dostupné z: `https://www.ceskyhokej.cz/clanky/klubovy-informacni-system-je-oporou-pro-sportovni-kluby`.
8. JML GROUP S.R.O. *Administrativní a evidenční systém pro sportovní kluby - Sportes* [online]. 2021. [cit. 2021-11-12]. Dostupné z: `https://www.sportes.cz/`.
9. *Týmuj* [online]. 2020. [cit. 2021-12-02]. Dostupné z: `https://tymuj.cz/about-us`.
10. MATTHEW, Martin. *Functional Vs. Non Functional Requirements: Differences* [online]. 2022. [cit. 2022-05-11]. Dostupné z: `https://www.guru99.com/functional-vs-non-functional-requirements.html`.
11. MLEJNEK, Jiří. *Analýza a sběr požadavků* [online]. 2021. [cit. 2022-01-10]. Dostupné z: `https://moodle-vyuka.cvut.cz/pluginfile.php/437169/mod_resource/content/5/03.prednaska.pdf`.
12. BERNHAUER, David. *Architektury webových aplikací* [online]. 2021. [cit. 2022-04-10]. Dostupné z: `https://courses.fit.cvut.cz/BI-TWA/media/topics/t08-architecture.pdf`.

13. ELLIOTT, Roxana. *Monolithic vs microservice architecture: Which is best?* [Online]. 2022. [cit. 2022-05-04]. Dostupné z: https://www.digitalocean.com/blog/monolithic-vs-microservice-architecture.
14. ČÁPKA, David. *MVC architektura* [online]. 2020. [cit. 2022-04-10]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.
15. *MVC* [online]. 2022. [cit. 2022-04-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/MVC.
16. Q-SUCCESS. *Historical yearly trends in the usage statistics of server-side programming languages for websites* [online]. 2022. [cit. 2022-04-11]. Dostupné z: https://w3techs.com/technologies/history_overview/programming_language/ms/y.
17. PHP GROUP. What is PHP? In: *PHP Documentation* [online]. 2022 [cit. 2022-02-05]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php.
18. PHP GROUP. What can PHP do? In: *PHP Documentation* [online]. 2022 [cit. 2022-02-05]. Dostupné z: https://www.php.net/manual/en/intro-whatcando.php.
19. PHP GROUP. *PHP: Downloads* [online]. 2022. [cit. 2022-02-05]. Dostupné z: https://www.php.net/downloads.
20. JETBRAINS S.R.O. *PHP Programming - The State of Developer Ecosystem in 2021 Infographic* [online]. 2022. [cit. 2022-04-11]. Dostupné z: https://www.jetbrains.com/lp/devecosystem-2021/php/.
21. *symfony 1.x legacy website* [online]. 2022. [cit. 2022-04-11]. Dostupné z: https://symfony.com/legacy.
22. *Projects using Symfony* [online]. 2022. [cit. 2022-04-11]. Dostupné z: https://symfony.com/projects.
23. KING, Thomas. *Laravel Vs Symfony – An Honest Comparison* [online]. 2021. [cit. 2022-04-11]. Dostupné z: https://reachstudios.co.uk/blog/laravel-vs-symfony-an-honest-comparison/.
24. REDMOND, Paul. *Larastan: Discover Bugs in Your Code Before Running It* [online]. 2018. [cit. 2022-04-15]. Dostupné z: https://laravel-news.com/larastan.
25. CHAND, Mahesh. *Most Popular Databases In The World* [online]. 2022. [cit. 2022-04-15]. Dostupné z: https://www.c-sharpcorner.com/article/what-is-the-most-popular-database-in-the-world/.
26. MONGODB, INC. *What is NoSQL?* [Online]. 2022. [cit. 2022-04-15]. Dostupné z: https://www.mongodb.com/nosql-explained.
27. PEARCE, Rohan. *Dead database walking: MySQL's creator on why the future belongs to MariaDB* [online]. 2013. [cit. 2022-04-15]. Dostupné z: https://www2.computerworld.com.au/article/457551/dead_database_walking_mysql_creator_why_future_belongs_mariadb/.
28. Getting Started with Doctrine - Doctrine Object Relational Mapper (ORM). In: *Doctrine Documentation* [online]. 2022 [cit. 2022-04-12]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.11/tutorials/getting-started.html.
29. Introduction - Doctrine Database Abstraction Layer (DBAL). In: *Doctrine Documentation* [online]. 2022 [cit. 2022-04-12]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/introduction.html.
30. *Bootstrap Get Started* [online]. 2022. [cit. 2022-04-15]. Dostupné z: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp.
31. *Git* [online]. 2022. [cit. 2022-04-16]. Dostupné z: https://git-scm.com/.

32. RED HAT, INC. *What is an API?* [Online]. 2017. [cit. 2022-04-17]. Dostupné z: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces.

33. *An overview of HTTP* [online]. 2022. [cit. 2022-05-10]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview.

34. *How Secure Are Query Strings Over HTTPS?* [Online]. 2009. [cit. 2022-04-26]. Dostupné z: https://blog.httpwatch.com/2009/02/20/how-secure-are-query-strings-over-https/.

35. DOCKER INC. *What is a Container? - Docker* [online]. 2022. [cit. 2022-05-07]. Dostupné z: https://www.docker.com/resources/what-container/.

36. VRANÝ, Lukáš. *Rozběhnutí Symfony na Wedos multihostingu* [online]. 2017. [cit. 2022-05-08]. Dostupné z: https://archiv.pehapkari.cz/blog/2017/05/01/symfony-a-wedos-multihosting/.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `app` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Paukert.pdf` ..... text práce ve formátu PDF