FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Implementace webové aplikace pro pořádání táborů
Student:	Jan Převrátil
Vedoucí:	Ing. Jiří Mlejnek
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem práce je vytvořit webovou aplikaci, která jako informační systém propojí tábory, rodiče a organizátory a bude tak podporovat vybrané procesy pořádání táborů. Táborům přinese možnost centralizovat jejich data a definovat politiky pro jednotlivé účastníky. Organizátorům umožní s těmito informacemi tábora pracovat a usnadní jim přípravu vybraných táborových aktivit. V neposlední řadě, rodičům zjednoduší a zpřehlední přihlašování dětí na tábory využívající tuto službu.
Pokyny pro vypracování:
- Analyzujte požadavky pro vytvoření této aplikace. Při specifikaci požadavků vyjděte z analýzy procesů vytvářené v rámci závěrečné práce Tomáše Kasala – Podpora efektivity pořádání táborů.
- Navrhněte architekturu a zvolte vhodné technologie pro její realizaci. Výsledná aplikace by měla běžet na technologii JVM (Java Virtual Machine).
- Aplikaci implementujte, důkladně otestujte a zdokumentujte.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 21. října 2022 v Praze.

Bakalářská práce

IMPLEMENTACE WEBOVÉ APLIKACE PRO POŘÁDÁNÍ TÁBORŮ

Jan Převrátil

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jiří Mlejnek
11. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych tímto poděkovat mému vedoucímu Ing. Jiřímu Mlejnkovi za poskytnuté konzultace, podnětné rady a cenné vedení v průběhu psaní této bakalářské práce. Dále bych chtěl poděkovat svým rodičům a přítelkyni Veronice za podporu, trpělivost a pochopení během celého studia. Děkuji vám.

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.

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 tohoto autorského díla, a to včetně všech počítačových programů 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, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 11. května 2023 .....

Abstrakt

Teoretická část této bakalářské práce se zabývá problémem aktuálních informačních systémů určených pro tábory. Z pohledu uživatele detailně analyzuje český trh táborových katalogů i informačních systémů. Práce dále identifikuje chybějící funkcionality existujících řešení a poté se věnuje porovnání vhodných technologií. Pro implementaci systému je zvolen multiplatformní jazyk Kotlin s technologiemi Spring (pro server) a KVsion (pro klienta). Praktická část práce se zabývá návrhem systému, který řeší vybrané požadavky. Poté následuje popis jeho implementace s vybranými ukázkami. Přínosem této práce je prozkoumání možností multiplatformního vývoje s použitím jazyka Kotlin. Dalším přínosem je navržená aplikace propojující rodiče, organizátory a tábory, která respektuje aspekty softwarového inženýrství a lze na ní navázat. Závěrem je zhodnocení práce a nastínění dalších směrů, kterými je aplikaci vhodné rozšířit.

Klíčová slova tábory, webová aplikace, informační systém, návrh a implementace, JVM jazyky, Kotlin, Spring, KVision

Abstract

The theoretical part of this bachelor thesis deals with the problem of current information systems intended for camps. It analyzes in detail the Czech market of camp catalogs and information systems from the user's prespective. The thesis further identifies the missing functionalities of existing solutions and then compares suitable technologies for implementation. The multi-platform Kotlin language with Spring (for the server) and KVsion (for the client) technologies is chosen for the system implementation. The practical part of the thesis deals with the design of the system that solves the selected requirements. This is followed by a description of its implementation with selected examples. The benefit of this thesis is the exploration of multiplatform development possibilities using the Kotlin language. Another benefit is a designed application that connects parents, organizers, and camps while respecting software engineering aspects and can be built upon. In conclusion, there is an evaluation of the thesis and an outline of future directions in which the application is suitable for expansion.

Keywords camps, web application, information system, design and implementation, JVM languages, Kotlin, Spring, KVision

Seznam zkratek

API	Application Programming Interface
CSR	Client Side Rendering
IDE	Integrated Development Environment
IS	Informační Systém
IT	Informační Technologie
JPA	Java Persistence API
JS	JavaScript
JVM	Java Virtual Machine
MVC	Model-View-Controller
OOP	Objektově Orientované Paradigma
ORM	Object Relation Mapping
PoC	Proof of Concept
SSR	Server Side Rendering
UI	User Interface
UML	Unified Modeling Language

Překladový slovník

Activity	Aktivita
Address	Adresa
Allergy	Alergie
App role	Aplikační role
Asthmatic	Astma
Batch	Turnus
Camp	Tábor
Camp organization	Táborová organizace
Camper	Táborník
Camper application	Přihláška táborníka
Category	Kategorie
Contract	Kontrakt
Country	Země
Creature	Bytost
Diet	Dieta
Drug	Léčivo
Event	Událost
Focus	Zaměření
Food	Jídlo
Food allergy type	Typ potravinové alergie
Health info	Zdravotní informace
Health insurance	Zdravotní pojišťovna
Housing	Ubytování
Medicament	Medikament
Mental	Psychické
Organizer	Organizátor
Other	Ostatní
Particle	Částice
Physical	Fyzické
Restriction	Omezení
Season	Roční období
Squad	Oddíl
Squad score	Body oddílu
Subdivision	Podrozdělení
Swimmer	Plavec
Swimmer type	Typ plavce
Theme	Téma
Transport	Doprava
User	Uživatel

Úvod

Kapitola se zabývá úvodem do problematiky informačních systémů pro tábory, dále popisuje motivaci k volbě tématu, stručně cíle práce, hlavní přínos práce a v neposlední řadě přibližuje strukturu tohoto dokumentu.

V České republice se ročně uskuteční řádově stovky různých táborů [1]. Přesto na českém trhu existuje pouze několik aplikací, které se specializují na usnadnění pořádání táborů. Aktuální stav lze popsat tak, že větší tábory mají často na míru vyvinuté IT systémy pro svou vlastní potřebu a oproti tomu většina menších táborů pracuje se všemi daty v papírové podobě, protože si drahý IT systém nemohou dovolit.

Situace na trhu informačních systémů pro tábory by mohla být lepší, k čemuž se pokusí přispět tato práce.

Autorovou osobní motivací je i fakt, že do nedávna pomáhal jako oddílový vedoucí na menším táboře. Během tohoto působení na vlastní kůži pociťoval absenci informačního systému, jenž by pomáhal se správou dat. V rámci tábora má oddílový vedoucí zodpovědnost za skupinu nezletilých, jejichž informace (od potravinových alergií přes léky či fyzická omezení) musí mít vždy po ruce, aby zajistil jejich bezpečí. Dále nosí domluvený harmonogram umožňující mu být s táborníky včas na správném místě a se správnou výbavou (např. vhodná obuv, oblečení dle počasí). Též přijde vhod znát aktuální stav bodování, pravidla táborových her atd. Zkrátka dat kolem tábora není málo a informační systém, který by dokázal přinést řád do různě nacházejících se papírů, tabulek a nástěnek, by našel své uplatnění.

Cílem práce je dle zadání analyzovat, navrhnout a vytvořit webovou aplikaci, která bude podporovat vybrané procesy pořádání táborů.

Kapitola 1
Cíle práce

Kapitola se zabývá popisem cílů teoretické a praktické části bakalářské práce.

Cílem teoretické části práce je identifikovat požadavky, navrhnout a zvolit technologie pro webovou aplikaci, jež v podobě informačního systému sjednotí tábory, rodiče a organizátory. Aplikace bude podporovat pořádání táborů, proto budou požadavky pro aplikaci specifikovány na základě táborových procesů. Specifikované požadavky budou analyzovány. Součástí analýzy požadavků bude též volba prioritních k implementaci v rámci praktické části.

Dílčím cílem je analyzovat existující JVM jazyky. Vypracování analýzy umožní porovnání jejich výhod a nevýhod. Na základě analýzy bude pro implementaci aplikace zvolen jeden z analyzovaných jazyků.

V neposlední řadě je cílem teoretické části zvolit vhodné související technologie pro usnadnění typických implementačních problémů, například práci s databází, vytváření instancí, provázání závislostí a podobně.

Hlavním cílem praktické části bakalářské práce je navrhnout a poté implementovat zvolené požadavky aplikace. Implementace využije vybrané technologie a JVM jazyk. Dílčími cíli jsou řádně aplikaci zdokumentovat a otestovat.

Kapitola 2
Rešerše existujících řešení

Kapitola se zabývá rešerší existujících řešení – definuje pojmy katalog a informační systém, následuje analýza vybraných táborových katalogů a táborového informačního systému.

Pro analýzu existujících řešení byly zvoleny 3 táborové katalogy, které jsou na českém trhu relevantní – každý z nich je provozován přes 10 let a pro rok 2023 eviduje více než 400 táborů a 700 turnusů [1]. Informační systém zvolený pro analýzu je pouze jeden, jelikož v době psaní práce nebylo možné na českém trhu najít žádný jiný volně dostupný a veřejně nabízený informační systém, který by se specializoval přímo na tábory.

Pojmy „táborový katalog“ a „informační systém“ budou definovány v následujících podkapitolách před analýzou konkrétních webů.

2.1 Táborové katalogy

Dle definice internetového slovníku se pojmem katalog rozumí utříděný seznam určitých položek [2]. Typickým zástupcem v reálném světě jsou katalogy produktů firem vyrábějící různé zboží. Konkrétně se jedná například o katalog cihel vyráběných cihlářským průmyslem.

Pojem katalog se přenesl i do IT světa. Virtuální katalog[1] je též utříděný seznam položek, ovšem oproti katalogu z fyzického světa umožňuje lepší interakci s uživatelem.

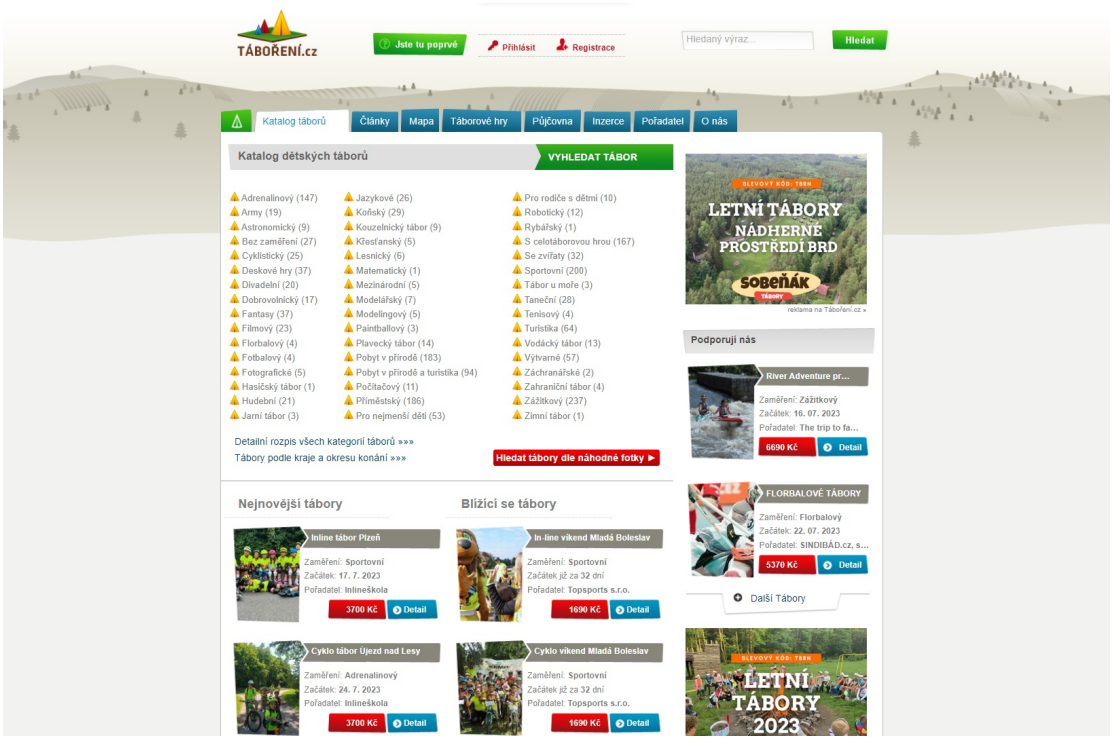
Virtuální katalog umožňuje uživateli:

- vyhledávat prvky katalogu dle specifikovaného textu,
- filtrovat prvky katalogu dle dostupných parametrů,
- seřadit výsledky vyhledávání a filtrování na základě hodnoty zvoleného parametru.

Základem uživatelské přívětivosti katalogu[2] je poskytnutí co nejvíce parametrů, podle nichž může uživatel položky filtrovat a řadit dle svých potřeb a preferencí. Množství parametrů by samozřejmě nemělo být na úkor přehlednosti katalogu, protože cílem každého katalogu, ať už toho fyzického či virtuálního, je především poskytnout uživateli přehlednou nabídku katalogových položek. Uživatel si poté může v této nabídce vyhledat položku pro něj nejvhodnější.

Táborovým katalogem se v tomto textu rozumí virtuální katalog, jehož položky tvoří konkrétní tábory, případně jejich turnusy. V následujících kapitolách bude slovo katalog mít význam virtuálního katalogu.

[1] či pouze katalog (v kontextu IT)
[2] tedy toho, aby byl pro uživatele katalog dobře použitelný



Obrázek 2.1 Titulní strana webu táboření.cz

2.1.1 Táboření.cz

Webový katalog taboreni.cz je provozován od roku 2011, čemuž odpovídá i jeho zastaralý vzhled. Katalog se od svého vzniku moc nezměnil [3]. Mobilní verze, která pravděpodobně přibyla v průběhu, není moc dobře realizovaná. Na většině mobilních zařízení přetéká šířku zařízení, a proto je nutné rolovat do strany. Další nevýhodou jsou malá písmenka, která jsou na mobilním zařízení špatně čitelná. Na druhou stranu má oproti ostatním katalogům obsáhlou titulní stranu.

Katalog v mobilní verzi, stejně tak jako v počítačové, zobrazuje sice jeden tábor na řádek (mimo titulní stranu), ale v kartě nejsou zbytečně velké prázdné mezery. Největší nevýhodou jsou zobrazované informace – místo strukturovaných informací jako je doporučený věk či lokalita pořádání zabírá nejvíce místa začátek popisu tábora, který je u každého tábora odlišný.

Web obsahuje kromě táborů i další zajímavé sekce, například táborové hry, rejstřík pořadatelů nebo články věnující se táborům.

V katalogu je pro rok 2023 evidováno okolo 1000 turnusů, což vypovídá o tom, že je katalog hojně využívaný. Inzerce táborů je zdarma. Provoz je financován čistě reklamou, která je pouze v pravém sloupci webu a nezabírá moc prostoru. Díky svému umístění i táborové tématice nenarušuje vzhled webu ani uživatele.

Základní filtrování je možné dle dopravy, ubytování, zaměření nebo zdravotních omezení a funguje dobře, což se nedá říct o dalších možnostech. Fulltextové vyhledávání ani pokročilé filtrování se během testování stránky nepodařilo zprovoznit. Po zadání parametrů, které několika výsledkům odpovídaly, vždy web ukázal zprávu „Nic nebylo nalezeno“. Ze zmíněného je možné usoudit, že vyhledávání nefunguje. Řazení výsledků je pouze výchozí.

Katalog pro vyhledání táborů nelze doporučit. Vzhled je zastaralý, mobilní verze přetéká, filtrování funguje pouze v základním režimu a zobrazené turnusy je nutné zkoumat přes detail, jelikož v náhledu chybí důležité informace. Plusy je možné vidět ve veliké nabídce táborů, v nerušivé reklamě a přidruženém obsahu. [4]

Obrázek 2.2 Titulní strana webu cesketabory.cz

2.1.2 České tábory.cz

Internetová stránka cesketabory.cz s katalogem táborů je v provozu již od roku 2008. Začátkem roku 2023 byl web předělán z původního tabulkového vzhledu do novějšího [5]. Bohužel se během aktualizace vzhledu moc nehledělo na celkový design webu. Web obsahuje spoustu nekonzistentních prvků. Pole filtrů jsou zaoblená, tlačítka v menu naopak hranatá a karty s tábory mají jako jediný prvek stín. Nové logo vypadá zvláštně, co má symbolizovat, zůstává otázkou.

V mobilní verzi je katalog dobře čitelný. Karty se vypisují v jednom sloupci. Nevýhodou je, že mezi zobrazenými parametry karty není zaměření, které by mohlo být místo zbytečně velké mezery, která je za obrázkem. Na počítačové verzi je pro změnu zbytečně velký samotný obrázek.

Ostatní sekce, které web obsahuje, jsou docela prázdné a působí dojmem, že nejsou moc používané. Sekce táborové vybavení, bazar a nabídka vzdělání jsou dokonce v době psaní prázdné úplně.

Katalog pro rok 2023 eviduje 770 turnusů, z čehož lze usoudit, že je katalog též dost využívaný. Vložení tábora je zdarma, platí se za jeho propagaci. Web ale obsahuje i reklamu, která je z pohledu uživatele velmi vyrušující. První 3 vypsané tábory jsou vždy ty sponzorované, a nad nimi je místo pro další reklamu. Reklama je navíc i v hlavičce webu.

Filtrování je až na vizuální stránku dobré, umožňuje filtrovat více než 10 parametry. Funguje správně, do té doby, než v polích, které umožňují vybrat více hodnot, vybereme zároveň více hodnot. V takovém případě se nám vrátí chyba od Microsoft SQL Serveru, která není ošetřena a běžnému uživateli nic neřekne. Řadit výsledky též není možné.

Katalog má značné množství nedostatků. Těmi jsou především nekonzistentní vzhled, neošetřené chyby a otravná reklama. Eviduje sice menší množství táborů, ale oproti předchozímu se dají lépe vyfiltrovat. Pro vyhledání tábora je použitelný, ale díky zmíněným problémům ho nelze doporučit. [6]

2.1.3 Dětské-tábory.info

Katalog detske-tabory.info, provozován od roku 2007, je nejstarším katalogem. V roce 2017 však prošel výrazným předěláním, které dopadlo velice dobře [7]. Z prezentovaných katalogů díky jeho aktualizaci vypadá nejlépe.

K mobilní verzi lze říci, že má několik menších nedostatků. Vypisované informace mají širokou mezeru od okrajů a jsou mezi nimi větší mezery, než by bylo nutné. Tábory se vždy vypisují jeden na řádek, stejně jako v počítačové, a zaměření tábora je až opět v detailu tábora. Jinak je zobrazení přehledné, čemuž pomáhají ikony i různá zvýraznění textu.

Web neobsahuje další sekce, což je lepší, než aby byly prázdné. Obsahuje pouze další stránky s informacemi a od roku 2021 nabízí evidenci účastníků (viz kapitola 2.2.1) [8].

Katalog pro rok 2023 eviduje přes 550 táborů a přes 1380 turnusů. Z čistých čísel tedy vyplývá, že je nejpoužívanější. Inzerce táborů je zdarma, web je financován reklamou, která je na začátku i na konci stránky. Reklamní blok je zatím největší ze všech webů. Na druhou stranu při procházení stránek se web vrátí jen na začátek sekce, takže reklamu uživatel nevidí pořád znovu.

Filtrování i vyhledávání funguje dobře až na pár detailů. Nastavování filtrů je zdlouhavé, jelikož se při každém kliknutí (zbytečně) obnovuje stránka (tábory se zobrazí až po finálním potvrzení). Počet táborů v kategorii ne vždy sedí s počtem, který se nakonec zobrazí, což uživatele může znepokojit, především, pokud se mu nakonec nezobrazí žádný tábor. Řazení není možné.

Katalog lze pro vyhledání táboru jako jediný doporučit, ale pouze pro základní vyhledávání, protože uživatelská zkušenost s ním nebude při použití více filtrů najednou ideální. Jeho výhodou je ve srovnání s ostatními nejmodernější vzhled a největší množství turnusů. [9]

Obrázek 2.3 Titulní strana webu detske-tabory.info

2.2 Táborový informační systém

Pro úplné pochopení termínu „informační systém“ je nezbytné porozumět pojmům „data“ a „informace“. Proto nejdříve následuje definice těchto dvou pojmů.

▶ **Definice 2.1.** *Data jsou získané a zachycené údaje, které lze fyzicky zaznamenat pomocí textové, grafické, zvukové nebo jiné formy.*

Data jsou výsledky získané na základě monitorování reality. Monitorováním může být myšleno klasické pozorování člověkem, ale může se jednat i o sledování a zaznamenávání dat počítačem.

Mezi zástupce dat se řadí zaznamenané poznatky, informace, znalosti i vědomosti.

Záznam dat lze provést na různé druhy nosičů například na kamenné destičky, papír s použitím tužky nebo digitální média pomocí počítače. [10]

▶ **Definice 2.2.** *Informace jsou data, kterým člověk přisuzuje význam.*

Informace vznikají interpretací dat a jejich vztahů konkrétním člověkem za pomocí jeho znalostí a vědomostí. Z toho lze usoudit, že co jsou pro jednoho člověka užitečné informace, mohou být pro druhého zbytečná data. [11]

Informace mezi sebou lidé komunikují. Tuto skutečnost lze nazvat výměnou zpráv.

▶ **Definice 2.3.** *Informační systém je systém sběru, uchování, analýzy a prezentace dat, který poskytuje různé informace různým uživatelům. Tento systém je souborem technicko-organizačních opatření a pravidel.*

Informační systém sbírá a uchovává data získaná většinou prostřednictvím jeho uživatelů, ale často též od jiných zdrojů. Zaznamenaná data poté zpracovává a prezentuje v širších souvislostech jeho uživatelům. Ti tato data interpretují do informací. Uživatelé jsou tedy nepostradatelnou součástí informačního systému.

Vzhledem k obecnosti definice, z ní vyplývá, že katalogy uvedené v předchozí sekci tvoří též informační systémem, který ovšem poskytuje pouze malé množství funkcionalit. Z toho důvodu byly vyčleněny do samostatné sekce.

Z definice také vyplývá, že informační systém nemusí být nutně podpořen počítačem. Ovšem podpoření počítačem může v mnohých ohledech dodržování a chod systému usnadnit, což může vést ke zlepšení fungování organizace. Zavedení informačního systému může zajistit:

- centralizaci zpracování informací,
- zjednodušení evidence,
- optimalizaci a zvýšení efektivity práce,
- snížení nákladů,
- zrychlení získávání informací,
- nárůst objemu a kvality získávaných informací.

Informační systém je ale jen nástroj. Samotné nasazení informačního systému všechny problémy nevyřeší, navíc žádný z předchozích bodů není garantovaný – vždy záleží na konkrétním případu. Na druhou stranu, pokud je zavedení informačního systému do organizace úspěšné, vždy s sebou nese pozitivní dopady, protože cílem informačního systému je pomáhat. [12]

Táborovým informačním systémem se v tomto textu rozumí informační systém, který se přímo specializuje na táborové organizace.

2.2.1 Pohodlně.info

Na stránkách táborového katalogu detske-tabory.info je nabízena služba „Evidence účastníků“, která je rozšířením správy táborů a turnusů. Tato služba je součástí pronajímaného informačního systému pro správu spolku pohodlne.info. Cena ročního poplatku činí 3 500 Kč. Za uvedenou částku systém nabízí:

- on-line přihlášku pro zájemce včetně zpracování platby,
- evidenci účastníků tábora (i náhradníků) a jejich kontaktních údajů,
- stav přihlášek účastníků, sledování úhrad,
- hromadné kontaktování účastníků (z minulých i letošních turnusů),
- připravení podkladů pro nahlášení tábora úřadům. [13] [14]

Popis systému pohodlne.info je zde uveden velmi stručně a existuje spoustu dalších drobnějších funkcionalit, které systém zvládá. Ovšem žádná zásadní věc zde není vyloženě vynechána, a proto lze o systému říci, že je vhodný hlavně pro ulehčení práce s přihláškami.

Vzhledem k ceně, kterou je navíc nutné platit každý rok, se systém vyplatí středním až velkým táborům. Pro ty menší tábory, které za rok uspořádají 1 až 3 turnusy, by byl spíše zbytečně drahý.

2.3 Vyhodnocení

V této kapitole byly zhodnoceny 3 různé katalogy. Ani jeden z představených katalogů není úplně ideální. Soustředí se hlavně na organizace a koncovému účastníkovi (jeho zákonnému zástupci) nabízí v rámci registrace pouze možnost hodnotit tábory a komentovat příspěvky. Přihlašování na tábory nesjednocuje žádný z katalogů. Hledání tábora dle parametrů má u každého z nich nějaké nedostatky. Vyhodnocení přehledně shrnuje tabulka 2.1.

Poslední představený katalog nabízí propojení na informační systém, který je ale určený pouze pro správce tábora a užitečný především pro práci s přihláškami před táborem. Organizátorům pro přípravu harmonogramu, stejně jako zákonným zástupcům, nic nenabízí.

Z uvedených skutečností vyplývá, že webová aplikace, která nebude soustředěná pouze na správce táborů, ale též na rodiče a organizátory, aktuálně nemá konkurenci. Proto má zmíněná aplikace šanci na trhu uspět a proto má smysl ji vytvořit.

Tabulka 2.1 Srovnání popisovaných řešení

Vlastnost / Funkcionalita	Táboření.cz	České tábory.cz	Pohodlne.info + Dětské-tábory.info
Správa táborů a turnusů	+	+	+
Cena za vložení tábora	0 Kč	0 Kč	0 Kč
Evidence účastníků	–	–	+
– Zpracování přihlášek a platby			+
– Kontaktování účastníků			+
– Cena modulu			3500 Kč/r.
Procházení kategorií katalogu	+	+	+
Mobilní verze katalogu	přetéká	velké mezery	velké mezery
Pokročilé filtrování	–	s chybami	zdlouhavé
Možnosti řazení v katalogu	–	–	–
Organizace turnusu (harmonogram, ...)	–	–	–
Usnadnění přihlášení táborníka	–	–	+

Kapitola 3
Analýza

Kapitola se zabývá analýzou navrhovaného systému. První dvě podkapitoly se věnují potřebné teorii využívané v dalších částech práce a to analýze požadavků a jazyku UML. Následuje podkapitola zabývající se definicí doménových pojmů a vypracováním doménového modelu. Poslední dvě podkapitoly prezentují funkční a nefunkční požadavky na systém.

3.1 Analýza požadavků

Prvními činnostmi v procesu vývoje software jsou sběr a analýza požadavků. Zmíněné činnosti jsou důležitým faktorem úspěchu každého softwarového projektu, jelikož v případě jejich chybného provedení může skončit projekt neúspěchem.

Cíle analýzy požadavků zahrnují:

- vymezení hranic systému,
- zachycení omezení systému,
- zpřesnění odhadu pracnosti,
- prioritizace požadavků k implementaci[1]. [15]

Požadavky se v rámci analýzy rozřazují dle různých kritérií. Nejznámějším kritériem pro kategorizaci požadavků je zaměření požadavku (někdy též základní kategorizace).

Zaměření rozděluje požadavky na:

Funkční požadavky popisují chování systému při jeho interakci s okolím – co musí/může systém a/nebo uživatel dělat a za jakých podmínek.

Nefunkční požadavky [2] popisují vlastnosti či omezení systému. Spadají sem všechny požadavky, které nejsou funkční. Typicky se jedná o požadavky na výkon, použitelnost, spolehlivost či rozšiřitelnost. [15]

Velmi důležitým kritériem pro rozdělení požadavků je priorita. Pro ohodnocení priority lze využít různých stupnic, časté bývá použití stupnice se třemi hodnotami.

[1] většinou dle potřeb zákazníka, v případě bakalářské práce je prioritizace požadavků na základě cílů práce
[2] někdy též uváděné jako obecné požadavky

Priorita v kontextu této práce nabývá dvou hodnot:

Zásadní bude označen takový požadavek, který je nedílnou součástí výsledné implementace pro naplnění cílů práce. Požadavky s touto prioritou tvoří jádro systému a proto je nutné je v každém případě implementovat.

Vedlejší bude označen každý, bez kterého se implementace sice obejde, ale zvyšuje hodnotu a použitelnost výsledného systému.

Posledním kritériem v rámci této práce pro kategorizaci požadavků je odhadovaná pracnost. Podobně jako v předchozím lze využít různých stupnic.

Odhadovaná pracnost v rámci práce nabývá následujích hodnot:

Malá pracnost zahrnuje požadavky s odhadem pracnosti < 1 MD.

Střední pracnost je pro požadavky s odhadem pracnosti > 1 MD a zároveň < 3 MD.

Velká pracnost slouží pro požadavky s odhadem pracnosti > 3 MD.

V rámci analýzy je často užitečné zobrazovat popisované prvky vizuální formou. V oblasti vývoje software se pro analýzu a návrh běžně používá grafický standard UML, proto se před samotnou analýzou text v další kapitole věnuje přiblížení tohoto jazyka. [16]

3.2 Jazyk UML

Unified Modeling Language, zkratkou UML, je systémem grafických notací určeným pro modelování softwarových systémů. Jeho hlavním přínosem je schopnost srozumitelně vizualizovat procesy, struktury a další koncepty, které by za použití běžného textového popisu působily mnohem složitěji. Snazší vizualizace přispívá k rychlejšímu pochopení modelované problematiky a díky tomu i k rychlejšímu vypracování odhadu či odhalení chyby. [16]

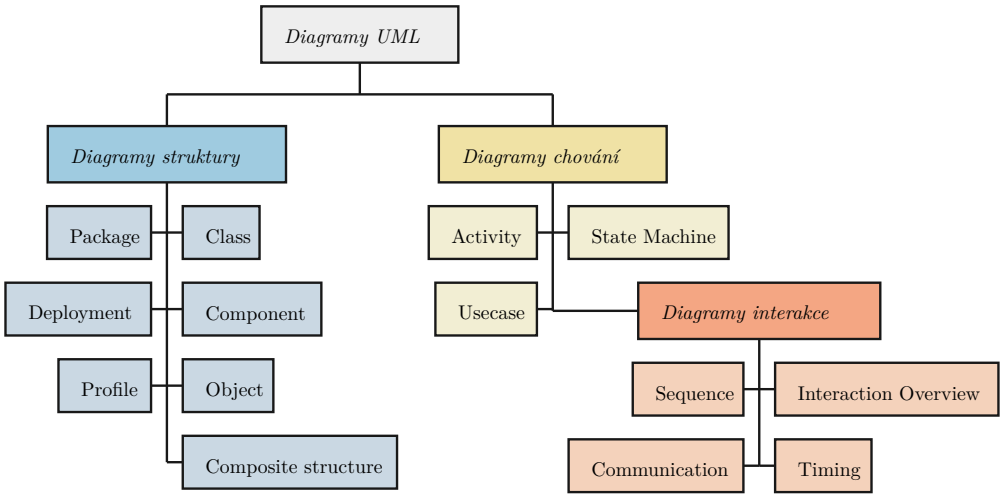
3.2.1 Historie UML

Historie vzniku UML sahá až do 90. let 20. století a je spjatá s objektově orientovanými jazyky, jejichž rozšíření mezi programátory přineslo nový pohled na návrh informačních systémů. Objektově orientované paradigma (ve zkratce OOP), které tyto jazyky přináší, umožňuje systémy dělit na objekty, jež omezují přístup ke svým datům pouze skrz striktně definovaná rozhraní, přes která si objekty zasílají zprávy. OOP přidává nové možnosti, jak celý systém strukturalizovat na jednotlivé funkční celky, které lze v ideálním případě snadno rozšířit nebo vyměnit. Důvodem vzniku OOP byla rostoucí komplexita systémů a s tím související udržovatelnost těchto systémů.

Nové možnosti dělení systémů umožněné objektově orientovanými jazyky, ale též pracování na stále složitějších systémech, přináší potřebu nového grafického zápisu. Díky tomu během první poloviny 90. let vzniká nezávisle na sobě mnoho různých metod a návrhových formátů. To vede k potřebě sjednocení a vytvoření jednoho univerzálního jazyka, čehož se v roce 1994 ujímá firma Rational Software. Cílem nebylo vytvořit novou notaci, ale spojit, zjednodušit a případně rozšířit již existující notace do jednoho standardu.

Díky rozsáhlosti a též formální přesnosti první zveřejněné verze UML, se do práce na dalších verzích připojují velké společnosti jako IBM, Microsoft, Oracle i další. Vzniká tak konsorcium s názvem Object Management Group (ve zkratce OMG), které dohlíží na UML standard. Společně tyto (a další firmy) s vizitkou OMG vydávají nové verze jazyka UML.

Dnes je UML rozšířený po celém světě a používá se pro modelování softwarových systémů v různých oblastech, jako jsou například informační systémy nebo webové aplikace. [16] [17]



Obrázek 3.1 Druhy diagramů jazyka UML

3.2.2 Význam UML

UML bylo původně vytvořeno pro tvorbu přesného a jednoznačného návrhu. Díky jeho formální přesnosti ho lze v dnešní době použít i jako programovací jazyk. Stále však převládá jeho použití v oblasti návrhu, kde lze jeho význam a užitečnost shrnout do následujících bodů. Nástroj UML umožňuje:

- přehlednou vizualizaci navržených oblastí systému,
- snazší komunikaci v rámci týmu i s klientem,
- jednodušší odhadování ceny systému. [16]

Především díky prvním dvěma bodům a faktu, že UML je v oblasti softwarového inženýrství standardem, budou v tomto textu vybrané UML diagramy využívány.

3.2.3 Druhy diagramů

Standard UML obsahuje ve své specifikaci 14 diagramů. Každý patří do jedné ze dvou skupin:

Diagramy struktury popisující strukturu systému, respektive z jakých částí je složený, a jak jsou tyto části na sebe navázány.

Diagramy chování popisující chování systému, respektive vnitřní logiku fungování systému nebo jeho části. Tato skupina obsahuje následující samostatnou podskupinu.

Diagramy interakce jsou diagramy chování popisující interakce systému s uživatelem nebo interakce jednotlivých částí systému mezi sebou. [16]

Rozdělení diagramů do zmíněných skupin je též přehledně znázorněno na obrázku 3.1.

Při využití jazyka UML jakožto nástroje pro modelování systému není potřeba využít všechny typy diagramů. Stejně tak je tomu i v této bakalářské práci, kde jsou využívány pouze některé z diagramů představených v následujících sekcích.

3.2.3.1 Usecase diagram

Usecase diagram, česky nazývaný diagram případů užití, zobrazuje systém z perspektivy uživatelů. Jeho účelem je znázornit funkcionality, kterými systém disponuje. Diagram popisuje, jaké možnosti použití jsou poskytovány různým aktérům, kterými jsou v kontextu tohoto diagramu různé skupiny uživatelů, ale může jimi být i systém samotný nebo čas (například pro automaticky spuštěné úlohy). Diagram tedy vizualizuje, co má systém dělat, ale již neznázorňuje, jak to má dělat. [16]

3.2.3.2 Class diagram

Třída je základní jednotkou class diagramu, česky nazývaného diagram tříd. Každá třída je v diagramu znázorněna v obdélníku, který má tři sekce. První z nich slouží hlavně pro název třídy, ale může obsahovat i další deklarace o třídě zvané stereotypy (nejčastěji „abstraktní“ či „rozhraní“). Druhá sekce poté slouží pro atributy a třetí pro metody. Metody a atributy mají nejdříve uvedený modifikátor přístupu, který je jeden z následujících:

\+ pro public (veřejné) atributy, viditelné pro jakékoliv třídy

~ pro package atributy, viditelné pro třídy v rámci balíčku

\# pro protected atributy, viditelné v třídě samotné a jejích potomcích

– pro private (privátní) atributy, viditelné pouze v třídě samotné

Po modifikátoru přichází název atributu nebo hlavička metody[3], následovaný typem atributu nebo typem návratové hodnoty metody. Typ je od názvu vždy oddělen dvojtečkou.

Třídy jsou mezi sebou propojeny pomocí vztahů. Těch UML nabízí hned několik, mezi nejpoužívanější patří:

Asociace je základní vazba, zakreslená plnou čarou, mezi dvěma entitami, které mohou existovat nezávisle na sobě. Přidáním jednoduché šipky se určuje, na jakou třídu je držen odkaz.

Agregace modeluje vazbu zachycující vztah celek-část. Zakresluje se plnou čarou s prázdným kosočtvercem u třídy celku (ten drží odkaz na část). Třída části může existovat i sama o sobě.

Kompozice se velmi podobá agregaci, ale oproti ní nemá třída části smysl bez celku – sama o sobě neexistuje (zaniká společně s celkem). Rozdíl v zakreslení dělá plný kosočtverec.

Generalizace reprezentuje vazbu dědičnosti. Značí se plnou čarou s šipkou, v podobě prázdného trojúhelníku, směřující z potomka na rodiče.

Realizace je vztahem mezi rozhraním a třídou, jenž jej implementuje. Značí se přerušovanou čarou s šipkou, v podobě prázdného trojúhelníku, směřující na rozhraní.

U vazeb asociace, agregace a kompozice (zde pouze u části) má smysl určit multiplicitu. Ta popisuje počet objektů jedné třídy, které jsou prostřednictvím vazby spojeny s objekty druhé třídy. Multiplicita může být reprezentována konkrétním číslem (1, 2, atd.), intervalem (nejčastěji 1..* s významem 1 až N) nebo znakem * (odpovídá intervalu 0 až N). [16]

3.2.3.3 Doménový model

Doménový model je formou diagramu tříd vytvářenou v rámci analýzy. Jedná se o diagram prezentující základní návrh entit systému a vazeb mezi nimi. Model lze nazvat zjednodušeným, protože třídy v modelu neobsahují metody, ani datové typy, ale pouze atributy a vazby na ostatní třídy. To, že model neobsahuje žádné datové typy, ho činí platformě nezávislým. [16]

[3] skládající se z názvu metody a kulatých závorek, uvnitř kterých jsou názvy argumentů metody a jejich typy

3.2.4 Sekvenční diagram

Sekvenční diagram patří do skupiny diagramů interakce. Slouží k vizualizaci interakce mezi dvěma a více účastníky v průběhu času. Účastníci symbolizují například objekty v systému, jednotlivé systémy či uživatele. Diagram zobrazuje výměnu zpráv nebo volání operací mezi účastníky a lze jím popsat například interakci programu s API nebo klienta se serverem.

V rámci sekvenčního diagramu se pro jednotlivé účastníky využívá čára života. Značí se přerušovanou čárou a reprezentuje časový průběh interakce. V horní části sekvenčního diagramu se horizontálně vedle sebe umisťují účastníci a pod nimi se vertikálně zaznamenávají jejich čáry života.

Dalším prvkem, který se na čáru života umisťuje, jsou activation. Značí se obdélníkem a reprezentují okamžiky, kdy je daný účastník aktivní, tedy vykonává akci nebo aktivně čeká na odpověď na základě předchozí akce.

V sekvenčním diagramu rozlišujeme několik typů zpráv. Pro potřeby této práce postačí následující:

Synchronní zpráva se používá nejčastěji. Znázorňuje odeslání zprávy či zavolání metody a umožňuje i specifikovat parametry. Při odeslání synchronní zprávy čeká odesílatel na dokončení metody na straně příjemce. V sekvenčním diagramu je synchronní zpráva zakreslena jako plná čára s vyplněnou šipkou směrem doprava.

Asynchronní zpráva funguje podobně jako synchronní zpráva. Nicméně, na rozdíl od synchronní zprávy se nečeká na odpověď nebo dokončení metody. V sekvenčním diagramu je asynchronní zpráva zakreslena jako plná čára s jednoduchou šipkou napravo.

Odpověď na zprávu zobrazuje odpověď na zprávu nebo návratovou hodnotu volání synchronní metody. V sekvenčním diagramu je odpověď zakreslena jako přerušovaná čára s šipkou směrem doleva. Obvykle zapisujeme proměnnou, do které se návratová hodnota ukládá.

Sekvenční diagramy mají ještě několik dalších možností notace, které v tomto textu nejsou využity, proto jsou zde vynechány. [16]

3.2.5 Diagramu balíčků

Diagram balíčků je jednoduchým diagramem ze skupiny diagramů struktury. Vizualizuje organizaci aplikace a na úrovni balíčků a vazby mezi nimi, což zahrnuje informace o závislostech a vnoření těchto prvků. Jednoduše řečeno, znázorňuje hierarchickou strukturu aplikace.

V diagramu balíčků se rozlišují 3 typy vazeb. Vazby jsou znázorněny přerušovanou čarou s jednoduchou šipkou. Nad šipkou může být jeden z následujících stereotypů:

«Import» zpřístupňuje balíčky veřejně – pokud balíček A importuje balíček B, který importuje balíčky C a D, jsou v balíčku A přístupné balíčky B, C i D. Pokud v diagramu balíčků není uveden žádný stereotyp, jedná se o «Import».

«Access» zpřístupňuje balíčky privátně – pokud má balíček A «Access» na balíček B, který importuje balíčky C a D, je v balíčku A přístupný pouze balíček B.

«Merge» slučuje několik implementací stejných členů v různých balíčcích. Pokud tedy dva balíčky obsahují stejnou třídu, mohou být tyto třídy spojeny pomocí vazby «Merge». Výsledkem je třída, která obsahuje vše z obou implementací. [16]

3.3 Doménový model

Během analýzy požadavků je vhodné vytvořit slovníček používaných doménových i obecných pojmů, což upřesní význam následovaných požadavků. S pomocí zmíněného slovníčku a požadavků lze následně sestrojit doménový model, který poskytuje základní datovou logiku pro naplnění požadavků.

Z předchozího odstavce vyplývá, že by měl doménový model následovat až po výčtu funkčních požadavků. Jelikož by popis tříd doménového modelu duplikoval informace uvedené v popisu doménových pojmů, je tato kapitola definicí pojmů a současně představením doménového modelu.

Vzhledem k tomu, že jsou v rámci funkčních požadavků doménové pojmy využívány, je tato kapitola zařazena před funkční požadavky.

Pojmy v doménovém modelu jsou v anglickém jazyce, aby korespondovaly s následujícími modely v kapitolách návrhu a implementace. Ve zmíněných kapitolách jsou pojmy v anglickém jazyce, protože reflektují názvy v kódu, kde jsou v angličtině kvůli univerzálnosti. V textu je o pojmech psáno v češtině, aby text nekombinoval dva jazyky. Překlad vychází z překladového slovníku.

3.3.1 Doménové pojmy

Sekce obsahuje výčet doménových pojmů seřazených v logickém sledu – složitějším pojmům, které odkazují na ostatní, předcházejí jednodušší pojmy. Související pojmy jsou blízko sebe, pokud je to možné. Následuje výčet pojmů:

Plnoletý je osobou, které již bylo 18 let, a proto se na tábor může přihlásit sama.

Nezletilý je osobou, které ještě nebylo 18 let, a proto je pro její přihlášení a účast na táboře potřeba souhlas jejího zákonného zástupce. Sám se tedy přihlásit nemůže, přihlašuje ho jeho zákonný zástupce.

Zákonný zástupce může na tábor přihlašovat nezletilé, které zastupuje.

Táborová organizace též táborová společnost je organizace založená za účelem pořádaní táborů. Organizace má množství různých kontaktů.

Slovo „táborová“ může být v tomto textu vynecháváno, protože je z kontextu zřejmé, o jakou organizaci se jedná.

Správce organizace je zakladatelem (jednatelem) táborové organizace. Spravuje data o registrované organizaci – pořádané tábory a jednotlivé turnusy táborů. Správce si hledá organizátory, kteří mu pomáhají s organizováním tábora. Těm může poslat pozvánku, ve které jim přidělí práva v rámci organizace.

Organizátor je osoba, která chce pomáhat s přípravou a realizací tábora. Proto je potřeba, aby se nejdříve stal členem libovolné táborové společnosti, která ho využije při pořádní táborů.

Pozvánka je prostředkem správce organizace, kterým může do své organizace přidat organizátory. Pozvánka může být odeslaná, přijatá nebo odmítnutá a lze během jejího odeslání nastavit práva případnému členovi.

Pověřený organizátor je již členem společnosti (přijal pozvánku) v rámci níž participuje v plánování turnusů táborů a připravování táborových aktivit. Je zodpovědný za různé aspekty, například může být oddílovým vedoucím, praktikantem či zdravotníkem.

Táborník je typicky nezletilý (ale může být i plnoletý), se zájmem přihlásit se (nebo již přihlášený) na tábor. Je součástí tábora jakožto účastník, což znamená, že se aktivně podílí na programu připraveném organizátory.

Tábor je naplánovaný dočasný pobyt táborníků a organizátorů s daným účelem. Stejný tábor lze naplánovat vícekrát, pak mluvíme o různých turnusech tábora. Tábor lze tedy chápat jako abstraktní plán a turnus je jeho konkrétní realizací. Často se ale pojmem tábor myslí konkrétní turnus tábora, například přihlášení na (turnus) tábor(a).

Turnus je konkrétní termín určitého tábora. Turnus přebírá z tábora jeho název, téma a účel, konkrétní harmonogram se ale může lišit, například z důvodu jiného počasí. Dále se z organizačních důvodů může lišit místo a s tím související kapacita turnusu, doprava a ceny.

Oddíl tvoří skupina účastníků tábora, tedy táborníků a organizátorů. Organizátoři v této skupině přebírají zodpovědnost za nezletilé táborníky. Oddíl slouží též pro hraní táborových her, za které může být obodován.

Aktivita je znovupoužitelná činnost s danou náplní. Může být prováděna táborníkem nebo skupinou a lze ji (i opakovaně) naplánovat. Typicky se jedná o skupinovou sportovní hru, která má daná pravidla, například fotbal, florbal nebo táborovou hru. Aktivita může být bodovatelná, což znamená, že po jejím naplánování a skončení za ní lze oddílům udělit body. Aktivita má nějakou minimální a maximální očekávanou délku, pro kterou jí má smysl plánovat.

Událost je jednorázová činnost s přesně definovaným začátkem a koncem, která nutně nemusí mít danou náplň. Událost může být naplánovanou aktivitou.

Body oddílu jsou uděleny oddílu po skončení předem naplánované aktivity (tedy události), která byla bodovatelná.

3.3.2 Hlavní část

Popis se vztahuje k obrázku 3.2. Většina tříd odpovídá pojmům definovaným v předchozí sekci, proto následuje pouze výčet tříd, které nebyly zmíněny nebo mají oproti pojmům odlišnosti:

Uživatel je plnoletá osoba registrovaná do systému. Během registrace si mimo vyplnění svých osobních informací zvolí, v rámci kterých aplikačních rolí chce systém používat.

Aplikační role jsou role, v rámci kterých lze systém využívat. Dostupné role jsou 3 a to následující: správce organizace, zákonný zástupce a organizátor. Uživatel si tyto role může zvolit během registrace.

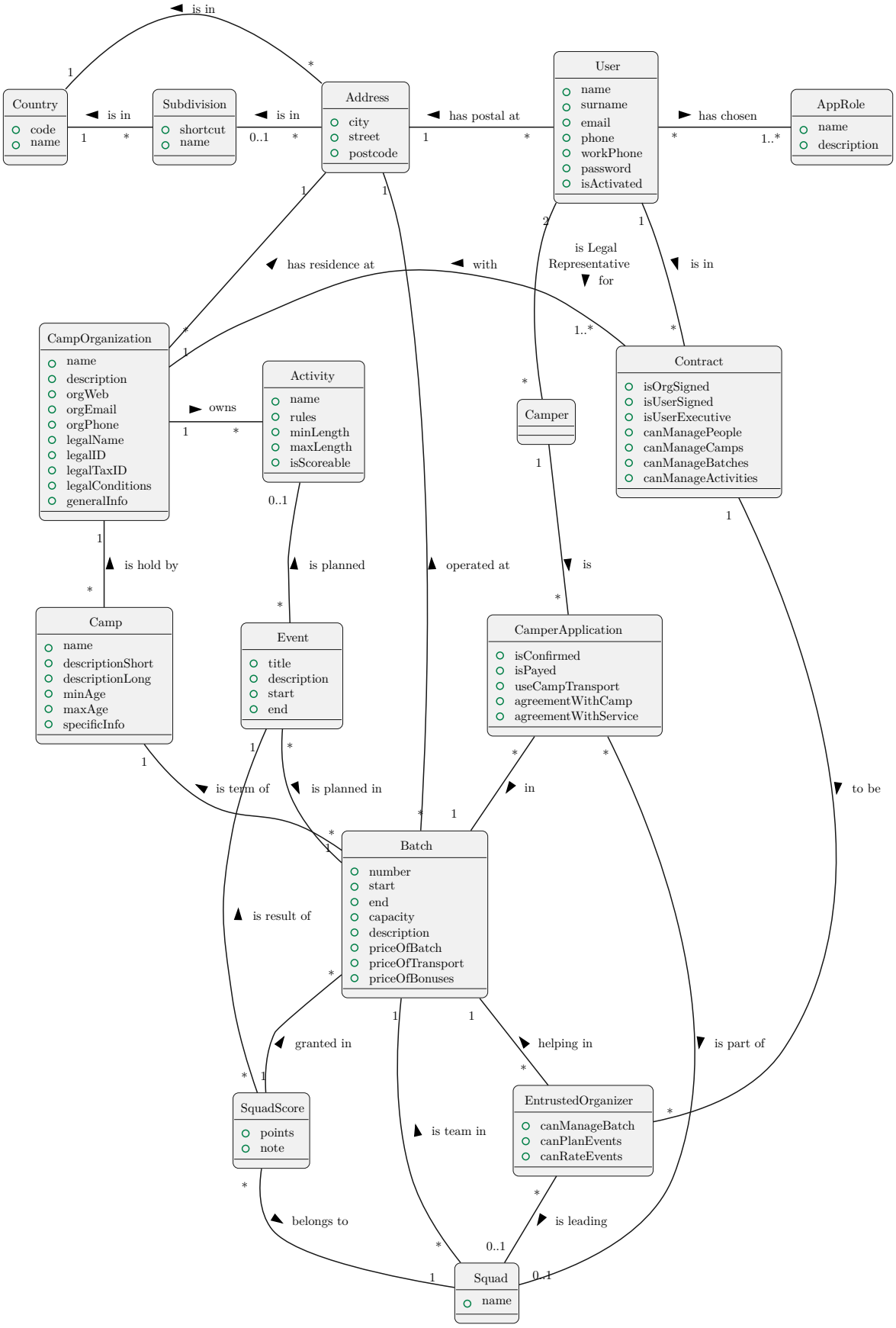
Kontrakt definuje vztah mezi uživatelem a organizací. Jedná se o abstrakci nad pozvánkou a možností být správce organizace. roli uživatele v organizaci a jeho oprávnění.

Adresa slouží pro definici adresy jedné ze tří navázaných tříd. U uživatele je jeho korespondenční adresou, u organizace udává sídlo firmy a u turnusu znamená místo pobytu.

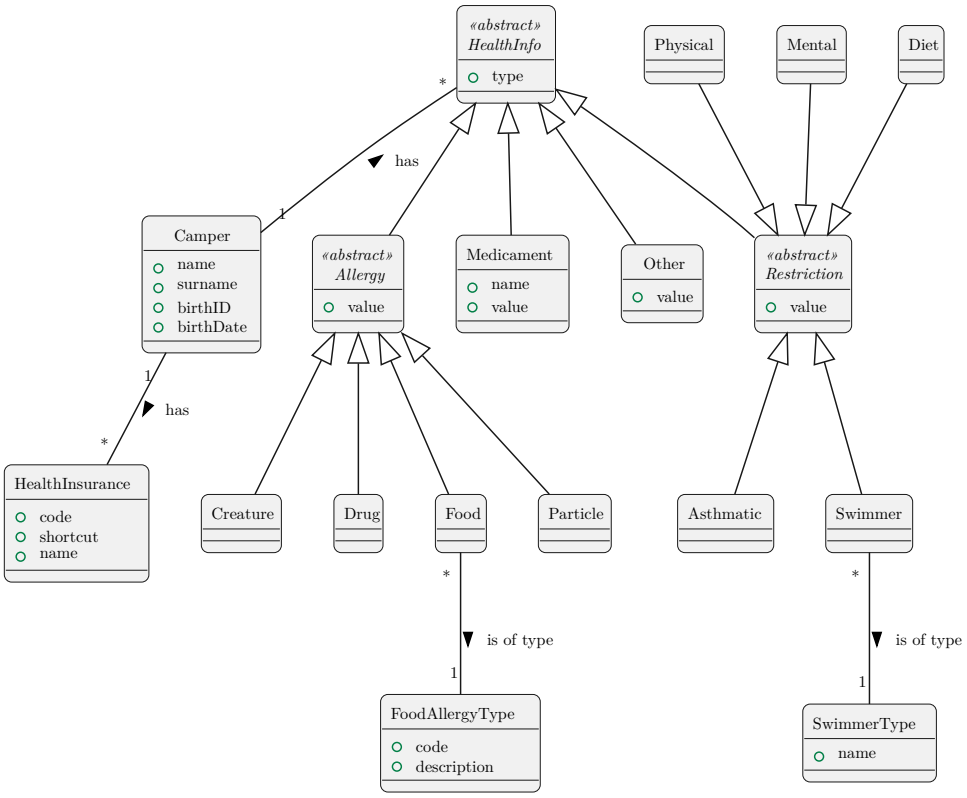
Země má úlohu pro specifikování země adresy. Většina adres bude v České republice, ale existují i výjimky, jimiž jsou tábory s výjezdem do zahraničí.

Podrozdělení má význam rozdělení země první úrovně. Jedné se o abstrakci nad Českými kraji, pro kterou bude primárně využito, ale umožňuje evidovat například Italské regiony.

Přihláška táborníka umožňuje přihlášení táborníka na turnus tábora jeho zákonným zástupcem. Přihláška může být (ne)potvrzená a (ne)zaplacená, obsahuje souhlasy s podmínkami.



Obrázek 3.2 Doménový model – Hlavní část



Obrázek 3.3 Doménový model – Zdravotní stav táborníka

3.3.3 Zdravotní stav táborníka

Sekce se věnuje obrázku 3.3. Tato část modelu se zabývá modelováním zdravotního stavu táborníka. Vyjma třídy táborníka žádná nebyla zmíněna v doménových pojmech, proto jsou tu popsány všechny třídy modelu:

Táborník obsahuje základní údaje o táborníkovi jako je jméno, příjmení, rodné číslo a datum narození.

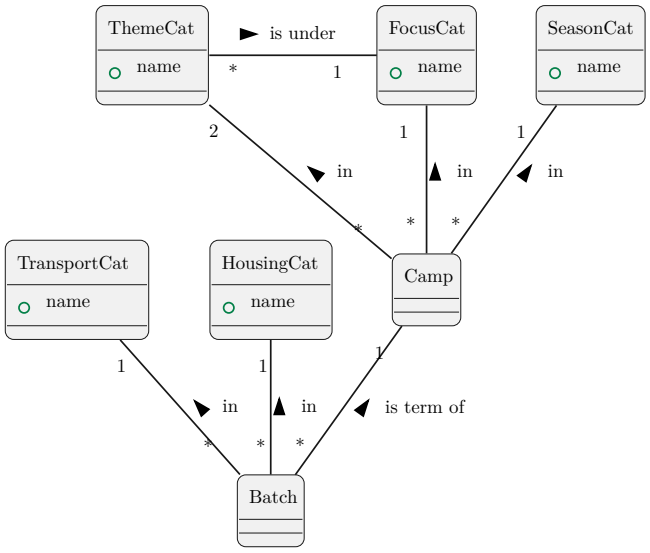
Zdravotní pojišťovna slouží jako číselník českých zdravotních pojišťoven. Obsahuje číslo, celé jméno pojišťovny a její zkratku.

Zdravotní informace je abstraktní třídou tvořící rodiče všech zdravotních informací, váže se na táborníka.

Alergie je abstraktní třídou pro všechny druhy alergií. Potomek *bytost* zahrnuje alergie na zvířata a hmyz, potomek *částice* zahrnuje alergie na prach a pyl. Potomek *léčivo* zahrnuje alergie na léky. Poslední potomek *jídlo* zahrnuje alergie na potraviny a má vazbu na třídu *typ potravinové alergie*, která slouží jako číselník potravinových alergií.

Medikament slouží pro zaznamenání užívaných léků. Zahrnuje název léku a způsob užívání.

Omezení je abstraktní třída nad všemi druhy omezení. Potomci zahrnují *fyzické* a *psychické* omezení, dále *diety*, *astma* a nakonec *plavce*. U poslední uvedené třídy je vazba na *typ plavce*, která slouží pro rozlišení úrovně plavectví.



Obrázek 3.4 Doménový model – Kategorie

3.3.4 Kategorie

Třídy v této sekci jsou popisem k obrázku 3.4. Část v tomto obrázku modeluje kategorie pro tábory a turnusy pro využití v katalogu. Třídy táboru a turnusu byly popsány v doménových pojmech 3.3.1, proto zde jejich popis není uveden.

Kategorie má význam pro abstrakci všech ostatních poddruhů kategorií.

Roční období je třídou pro roční období, ve kterém se tábor pořádá. Tábor se může pořádat v jednom ze čtyř ročních období – na jaře, v létě, na podzim nebo v zimě.

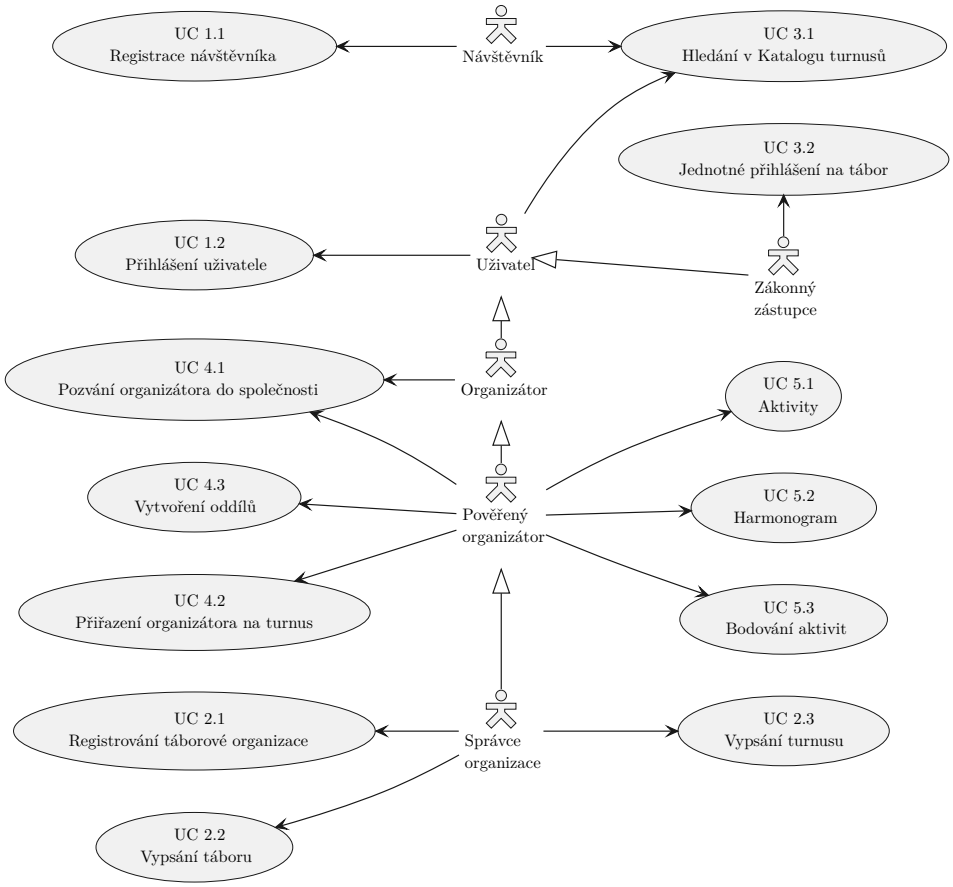
Zaměření slouží pro obecné zaměření tábora. Zaměření by měla být co nejobecnější a celkově by jich mělo být 5 až 10.

Téma je bližší specifikací zaměření tábora. Témata spadají pod jedno obecné zaměření a každý tábor spadá pro jednoduchost a spravedlivost právě do 2 témat. Témata by měla konkretizovat zaměření – ke každému zaměření by mělo být zhruba 3 až 15 témat.

Ubytování obsahuje typy ubytování, převážně různé typy budov. Spadá sem i příměstský typ[4] ubytování.

Doprava určuje organizátorem poskytovaný druh dopravy. Typy dopravy zahrnují dopravu autobusem, autem, vlakem, ostatním MHD a vlastními prostředky (pokud není poskytována žádná doprava).

[4] což znamená, že tábor neobstarává ubytování



Obrázek 3.5 Model případů užití pro navrhovaný systém

3.4 Funkční požadavky

Kapitola obsahuje výčet funkčních požadavků, které jsou specifikovány pomocí případů užití. Z toho důvodu je na začátku kapitoly model případů užití s popisem aktérů a následují podrobnosti o jednotlivých případech užití, které jsou shlukovány a číslovány dle obecnějšího cíle.

Model 3.5 zachycuje, kdo může vykonávat specifikované případy užití. Systém sám neiniciuje žádný případ (i když participuje ve všech) proto není v modelu zachycen jako aktér. Následuje výčet a popis aktérů modelu:

Návštěvník je osoba neregistrovaná do systému. V případě, že se registruje, se stává uživatelem.

Uživatel je plnoletá osoba registrovaná do systému.

Organizátor je uživatel s aplikační rolí organizátor.

Pověřený organizátor je uživatel s aplikační rolí organizátor a zároveň je již členem dané organizace, v rámci které mu správce přidělil potřebná práva pro konkrétní činnosti.

Správce organizace je uživatel s aplikační rolí správce organizace a zakladatelem dané táborové organizace.

Zákonný zástupce je uživatel s aplikační rolí zákonný zástupce.

3.4.1 Evidence uživatelů

Základem každého informačního systému je evidence uživatelů, kteří s daným systémem pracují. Navazující nedílnou součástí IS, též týkající se uživatelů, je dle definice 2.3 zanesení technicko-organizačních opatření do daného systému. To lze vyřešit pomocí rozdělení uživatelů dle jejich rolí v IS. Jelikož by systém mělo být možné provozovat jako veřejně přístupný, uživatelé si mohou jejich role sami zvolit během registrace. Cíle evidence uživatelů proto zahrnují:

- Návštěvník se může do systému registrovat vyplněním a odesláním registračního formuláře.
- Návštěvník si během registrace zvolí z dostupných rolí, v rámci kterých bude systém využívat.
- Dostupné role jsou **správce organizace**, **zákonný zástupce** a **organizátor**.
- Po registraci přijde uživateli email s odkazem pro dokončení jeho registrace.
- Uživatel se může do systému přihlásit vyplněním a odesláním přihlašovacího formuláře.

3.4.1.1 UC 1.1 – Registrace návštěvníka

- Priorita: Zásadní
- Odhadovaná pracnost: Střední

Návštěvník se prostřednictvím svého webového prohlížeče dostane na stránky systému. Zde si může přečíst obecné informace, jako například k čemu systém slouží nebo pro koho je určen. Klikne na tlačítko „Registrovat se“ pro registraci do systému. Zobrazí se mu formulář pro registraci. Zde vyplní své osobní informace, jako jméno, příjmení a své kontaktní údaje jako email a telefonní číslo. Dále si návštěvník zvolí, v rámci kterých rolí chce systém používat. Musí si zvolit minimálně jednu a maximálně všechny role. Nakonec vyplní heslo a klikne na tlačítko „Registrovat se“. V případě vyplnění validních dat z návštěvníka stal uživatel.

3.4.1.2 UC 1.2 – Přihlášení uživatele

- Priorita: Zásadní
- Odhadovaná pracnost: Střední

Uživatel se prostřednictvím svého webového prohlížeče dostane na stránky systému. Jelikož je již registrovaný, lze předpokládat, že zná informace na webu. Klikne na tlačítko „Přihlásit se“ pro přihlášení do systému. Zobrazí se mu formulář pro přihlášení. Zde vyplní svůj email a heslo z registrace. Systém ho přihlásí.

3.4.2 Evidence táborů a organizací

Zásadním prvkem informačního systému pro tábory je umožnit jejich pořadatelům, aby mohli registrovat svojí organizaci a vypisovat pořádané tábory a termíny turnusů. Cíle evidence táborů zahrnují:

- Správce organizace může zaregistrovat táborovou organizaci.
- Správce organizace může vypsat tábor (1 i více), jehož pořadatelem je registrovaná organizace.
- Správce organizace může tábor smazat, pokud neobsahuje žádné vypsané turnusy.
- Správce organizace může vypisovat turnusy (1 i více) k vypsanému táboru.
- Správce organizace může v systému interagovat (mazat, editovat) pouze s tábory a turnusy, které jsou pořádány jím spravovanou organizací.

3.4.2.1 UC 2.1 – Registrování táborové organizace

- Priorita: Zásadní
- Odhadovaná pracnost: Malá

Uživatel musí být přihlášený a na úvodní stránce aplikace. Klikne na dané tlačítko registrace. Vyplní požadované informace a data o organizaci. Klikne na tlačítko registrovat organizaci a v případě validních dat systém vytvoří novou organizaci a informuje ho o tom. Pokud jsou data nevalidní, formulář ho o tom informuje.

3.4.2.2 UC 2.2 – Vypsání táboru

- Priorita: Zásadní
- Odhadovaná pracnost: Malá

Uživatel musí být přihlášený a na úvodní stránce aplikace. Přejde na stránku tábory. Klikne na dané tlačítko vypsání táboru. Systém vytvoří formulář. Uživatel vyplní požadované informace a data o táboru. Klikne na tlačítko vypsat tábor a v případě validních dat systém vytvoří nový tábor a informuje ho o tom. Pokud jsou data nevalidní, formulář ho o tom informuje. V případě nějaké změny lze celý formulář upravit, nebo smazat.

3.4.2.3 UC 2.3 – Vypsání turnusu

- Priorita: Zásadní
- Odhadovaná pracnost: Střední

Uživatel musí být přihlášený a na úvodní stránce aplikace. Přejde na stránku turnusy. Klikne na dané tlačítko vypsání turnusu. Systém vytvoří formulář. Uživatel vyplní požadované informace a data o turnusu. Klikne na tlačítko vypsat turnus a v případě validních dat systém vytvoří nový turnus a informuje ho o tom. Pokud jsou data nevalidní, formulář ho o tom informuje. Pokud potřebuje uživatel upravit nějaké informace, tak musí kliknout na tlačítko upravit.

3.4.3 Katalog táborů

Registrované turnusy táborů jednotlivých organizací tvoří katalog táborů. Katalog je dostupný pro všechny návštěvníky a registrovaným rodičům nabízí možnost přihlásit jejich děti na tábor prostřednictvím jednotné přihlášky. Rodič (zákonný zástupce) si do svého účtu může zaevidovat informace o dětech (zastupovaných nezletilých). To mu následně umožňuje, přihlásit nezletilé na několik táborů bez nutnosti vyplňovat přihlášku pro každý tábor zvlášť. Cíle katalogu turnusů jednotlivých táborů zahrnují:

- Návštěvníci i uživatelé systému si mohou zobrazit katalog vypsaných turnusů táborů všech registrovaných organizací.
- Katalog poskytuje alespoň 3 parametry, podle kterých je možné turnusy filtrovat.
- Katalog poskytuje alespoň 3 parametry, podle kterých je možné řadit turnusy.
- Zákonný zástupce může do svého účtu evidovat nezletilé, které zastupuje.
- Zákonný zástupce může přihlásit nezletilého na vypsaný turnus (1 i více).
- Zákonný zástupce může v systému interagovat (editovat, mazat) pouze s nezletilými, kteří jsou evidováni v jeho účtu.

3.4.3.1 UC 3.1 – Hledání v Katalogu turnusů

- Priorita: Zásadní požadavek
- Odhadovaná pracnost: Velká

Uživatel musí být přihlášený a na úvodní stránce aplikace. Poté přejde na stránku s nabídkou katalogů. Systém zobrazí vypsané jednotlivé turnusy, které uživatel může rovnou procházet. Uživatel aplikuje jím požadované filtry. Systém nakonec zobrazí vyfiltrované turnusy a uživatel si už jen vybere ten nejvhodnější.

3.4.3.2 UC 3.2 – Jednotné přihlášení na tábor

- Priorita: Zásadní požadavek
- Odhadovaná pracnost: Velká

Uživatel musí být přihlášený a mít ve svém účtu vyplněné informace alespoň o jednom táborníkovi. Potom si najde požadovaný tábor a klikne na přihlásit táborníka. Pokud má ve svém účtu více táborníků, tak zvolí, kterého chce přihlásit a potvrdí svůj výběr. Systém vytvoří přihlášku. V případě lze přihláška upravit nebo smazat.

3.4.4 Správa uživatelů v organizaci

Jelikož správce organizace nezvládne uspořádat celý turnus sám, potřebuje pomoc organizátorů. S plánováním a realizací harmonogramu tábora mu pomáhají pověření organizátoři. Ty může správce pověřit pozváním do své organizace a přidělením příslušných práv v rámci organizace. Organizátoři mohou pozvání přijmout i odmítnout. V případě, že organizátor pozvání přijme, stane se členem společnosti a může tak být přiřazen k libovolnému turnusu pořádaným danou organizací. Osoby v turnusu, tedy táborníky a organizátory, lze poté rozdělit do oddílů. Cíle správy uživatelů v organizaci zahrnují:

- Správce organizace a pověřený organizátor můžou pozvat organizátora (1 i více) do táborové organizace, kterou spravují.
- Během pozvání lze nastavit práva v různých kategoriích, která pozvaný organizátor dostane. Maximálně lze pozvanému organizátorovi přidělit stejná práva jako správci organizace.
- Pozvaný organizátor může pozvánku přijmout a stát se tak členem společnosti, což mu umožní účastnit se turnusů a pomáhat s jejich organizováním.
- Pozvaný organizátor může pozvánku odmítnout.
- Správce organizace nebo pověřený organizátor může přiřadit organizátora (1 i více) k vybranému turnusu (k 1 i více).
- Během přiřazení lze organizátorovi nastavit práva v různých kategoriích, která organizátor v rámci daného turnusu dostane.
- Správce organizace nebo pověřený organizátor může rozdělit táborníky do oddílů.
- Každý táborník může být právě v jednom oddílu.
- Oddíl se váže k danému turnusu a skládá se pouze z táborníků z daného turnusu.
- Správce organizace nebo pověřený organizátor může v rámci oddílů rozdělovat pouze táborníky a organizátory, kteří jsou součástí daného turnusu pořádaného danou organizací.

3.4.4.1 UC 4.1 – Pozvání organizátora do společnosti

- Priorita: Zásadní
- Odhadovaná pracnost: Malá

Uživatel musí být přihlášený a na úvodní stránce aplikace. Přejde na stránku se členy, klikne na pozvat, vyplní požadované informace (email), vyplní role a systém pozvánku odešle. Adresát poté, co uvidí pozvánku na úvodní stránce, tak ji může přijmout, nebo odmítnout.

3.4.4.2 UC 4.2 – Přiřazení organizátora na turnus

- Priorita: Zásadní
- Odhadovaná pracnost: Malá

Uživatel musí být přihlášený a vyskytovat se na úvodní stránce aplikace. Dále přejde na stránku se členy, klikne na přiřadit na turnus u vybraného člena a systém zobrazí okno, ve kterém vybere turnus a potvrdí svoji volbu.

3.4.4.3 UC 4.3 – Vytvoření oddílů

- Priorita: Vedlejší
- Odhadovaná pracnost: Velká

Uživatel musí být přihlášený a být přítomný na úvodní stránce aplikace. Poté přejde na stránku s turnusy. Systém zobrazí nabídku vybraných turnusů a uživatel si zvolí ten preferovaný. Klikne na tlačítko vytvořit oddíly, zvolí jejich počet a poté rozřadí jednotlivé členy a potvrdí. Systém vytvoří oddíly a rozřadí do nich táborníky. Počet i seznam táborníků lze v případě potřeby upravovat, nebo mazat.

3.4.5 Organizování turnusu

Pro každý turnus tábora je potřeba připravit harmonogram, podle něhož se bude řídit chod tábora. Harmonogram se skládá z událostí, které mají daný začátek a konec. Událost může obsahovat odkaz na (znovupoužitelnou) táborovou aktivitu, kterou daná společnost dokáže uspořádat. Například pokud společnost vlastní nafukovací hrad, může si zaevidovat aktivitu „Skákání v nafukovacím hradu“, k té přidat pravidla a další podrobnosti. Takto evidovanou aktivitu lze poté libovolně plánovat jako událost v jednotlivých táborových turnusech. Pokud je aktivita evidována jako bodovatelná, po skončení události za ní lze oddílům přidělit body.

- Správce organizace a pověřený organizátor můžou přidávat táborové aktivity do organizace, kterou spravují.
- Aktivity slouží pro plánování harmonogramu, stejná aktivita může být naplánovaná vícekrát.
- Správce organizace a pověřený organizátor mohou plánovat harmonogram turnusu.
- Plánování harmonogramu zahrnuje vytvoření události obsahující konkrétní čas. Událost může mít vazbu i na aktivitu, která se bude v rámci události dělat.
- Správce organizace může plánovat harmonogram pouze u turnusů pořádaných jím spravovanou organizací.
- Organizátor může plánovat harmonogram pouze u turnusů, které organizuje (přijal na ně pozvánku).
- Správce organizace a organizátor můžou bodovat aktivity zařazené v harmonogramu.
- Body se vážou ke zvolenému oddílu.
- Správce může bodovat aktivity pouze u turnusů pořádaných jím spravovanou organizací.
- Organizátor může bodovat aktivity pouze u turnusů, u kterých přijal pozvánku.

3.4.5.1 UC 5.1 – Aktivity

- Priorita: Zásadní
- Odhadovaná pracnost: Střední

Uživatel musí být přihlášený a pobývat na úvodní stránce aplikace. Jako další krok přejde na stránku aktivit, kde vyplní potřebná data. Systém založí novou aktivitu.

3.4.5.2 UC 5.2 – Harmonogram

- Priorita: Zásadní
- Odhadovaná pracnost: Velká

Uživatel musí být přihlášený v systému a být na úvodní stránce konkrétního turnusu. Poté si uživatel zvolí záložku harmonogramu, ve které může přidat novou událost a napsat časový sled událostí na další dny. Systém vytvoří nový harmonogram. V případě potřeby lze harmonogram smazat či jen upravit.

3.4.5.3 UC 5.3 – Bodování aktivit

- Priorita: Vedlejší
- Odhadovaná pracnost: Střední

Uživatel musí být přihlášený v aplikaci a vyskytovat se na stránce konkrétního turnusu. Pak si zvolí záložku určenou k zaznamenávání bodování aktivit. Systém zobrazí odehrané aktivity a jejich body. Na této stránce lze zadávat hodnocení jednotlivým oddílům. Body se dají upravit.

3.5 Nefunkční požadavky

Kapitola obsahuje výčet nefunkčních požadavků, ty jsou rozděleny do čtyřech kategorií, a to použitelnost, spolehlivost, výkon a podporovatelnost.

3.5.1 Použitelnost

Požadavky na použitelnost se posuzují z hlediska uživatele systému. Definují, jak aplikace vypadá, nebo jakým dojmem působí. Snaží se systém definovat tak, aby byl snadno použitelný.

- Systém je možné provozovat přes internet jako veřejně přístupnou webovou aplikaci.
- Systém lze použít v počítačové i mobilní verzi prohlížeče.
- Rozhraní systému obsahuje barvy monochromatického nebo dichromatického spektra[5].
- Intenzita barev v rozhraní systému je mírná a není zde přehnaný kontrast mezi jednotlivými barvami.

3.5.2 Spolehlivost

Kategorie spolehlivosti se týká závažnosti vyskytujících se chyb a jejich dopadu na systém. Požadavky na spolehlivost mohou též hodnotit schopnost systému na zotavení se v případě výpadku.

- Systém umožňuje naplánovat a realizovat turnus bez závažných výpadků.
- Při výpadku je systém schopný obnovit perzistentně uložená data[6] (data rozpracované akce mohou být při výpadku ztracena[7]).

3.5.3 Výkon

Výkonnostní požadavky se soustředí na stanovení minimální rychlosti odezvy systému a počtu uživatelů zároveň využívajících systém.

- Systém by měl zvládnout minimálně řádově desítky najednou aktivních uživatelů.
- Odezva systému je při zátěži desítek uživatelů maximálně v nízkých jednotkách sekund.

3.5.4 Podporovatelnost

Požadavky v kategorii podporovatelnosti se zaměřují na rozšiřitelnost systému o nové funkce.

- Implementaci lze bez nutnosti zásahu do datového modelu rozšířit o další role, například roli pro táborníka.
- Implementaci lze bez nutnosti zásahu do datového modelu rozšířit tak, aby mohla mít organizace více správců.

[5] dvoubarevné spektrum obsahující dvě barvy v různých úrovních jasu a sytosti
[6] například již uložené tábory
[7] například data vyplněných polí v neodeslaném formuláři

Kapitola 4
Výběr technologií

Kapitola se zabývá představením technologie JVM, jejích výhod a existujících programovacích jazyků s ní kompatibilních. Následuje vyhodnocení, které volí konkrétní jazyk pro implementaci. Poté následuje volba aplikačních rámců pro tvorbu serverové a klientské části aplikace.

Vzhledem k tomu, že zadání požaduje použití technologie JVM, je nezbytné tuto technologii představit a věnovat pozornost jejím výhodám zasluhujícím se o to, že je i v současné době stále aktuální a hojně využívanou platformou.

4.1 Technologie JVM

Java Virtual Machine (JVM), česky též virtuální stroj Javy, je moderní technologie poskytující virtuální prostředí pro vývoj a nasazení aplikací. Tato platforma vznikla v roce 1995 ve společnosti Sun Microsystems, která vydala i její první implementaci.

JVM se rychle stala populární díky svému zaměření na internetové aplikace, jež v této době byly na vzestupu. Další zásadní výhodou platformy JVM je to, že byla navržena s ideologií „write once, run everywhere“ (napiš jednou, spusť všude), tedy aby aplikace napsaná pro tuto platformu mohla běžet nejen na různých operačních systémech, ale i na různém hardwaru. JVM byla původně zamýšlena pro využití v set-top boxech, ale kvůli nezralosti tohoto trhu byla nakonec dodána i na stolní počítače.

Pro umožnění popsané platformní nezávislosti vytvořila společnost Sun Microsystems vlastní binární spustitelný formát nazvaný Java bajtkód. Pro spuštění programů zkompilovaných do Java bajtkódu musí být na daném systému nainstalována implementace JVM, která Java bajtkód dokáže zpracovat. Základem implementací JVM je interpret, jenž při běhu aplikace překládá bajtkód do strojového kódu pro dané zařízení. Jelikož je ale interpretování vždy pomalejší, než přímé spuštění nativního kódu, obsahují JVM implementace často i JIT kompilátor. Ten překládá do nativního kódu metodou „just-in-time“ (právě včas) ty sekce bajtkódu, které jsou často volány. Interpret pak spouští verzi přeloženou do nativního kódu místo verze v bajtkódu. Díky této optimalizaci jsou programy běžící na JVM srovnatelně rychlé s těmi nativními.

Původně byla technologie JVM navržena pro aplikace napsané v programovacím jazyce Java, což lze vyvodit už z jejího názvu. Návrháři programovacích jazyků ale brzy zjistili, že pokud dokáží svůj jazyk zkompilovat do Java bajtkódu, mohou využívat výhod a rozsáhlé knihovny tříd technologie JVM. Proto se tato platforma stala pro návrháře programovacích jazyků oblíbenou a během let vzniklo několik zajímavých alternativ k jazyku Java. [18]

4.1.1 Výhody JVM

Mimo výhody popsané v předchozích odstavcích disponuje technologie JVM mnoha dalšími. Tři nejdůležitější z nich, mající největší podíl na její stálé relevantnosti, shrnuje tato sekce.

4.1.1.1 Silná standardní knihovna

Třídy v „Java Class Library“ (standardní knihovně tříd Javy) poskytují mnoho užitečných funkcí usnadňujících práci programátorů. Knihovna obsahuje třídy pro komunikaci se servery, spouštění vláken operačního systému či čtení z a zápis do konzolového okna i souborů. Dále obsahuje třídy definující základní datové struktury, jako jsou seznamy, mapy a mnoho dalších. To umožňuje návrhářům programovacích jazyků soustředit se především na samotný návrh jazyka. Vytvoření plně otestované multiplatformní knihovny srovnatelné s tou Javy je velmi náročný úkol. [18]

4.1.1.2 Přizpůsobování se změnám na trhu

V prvních verzích byla standardní knihovna tříd navržena pouze pro jednojádrové procesory a počítače s omezenou pamětí. S nástupem vícejádrových procesorů přibyly nové třídy pro podporu práce s více vlákny. Když se začalo prosazovat funkcionální programovací paradigma, Java byla rozšířena o lambdy a proudy. Součástí přizpůsobování se trhu je i to, že občas musí něco odejít. V případě JVM šlo například o podporu appletů, což byly aplikace napsané v Javě, které mohly být spouštěny v prohlížeči. Ty byly kompletně vytlačeny jazykem JavaScript. [18]

4.1.1.3 Rozsáhlý ekosystém

Sebelepší knihovna tříd samozřejmě nemůže pokrýt všechny potřeby programátora. Proto se vývojáři často obrací na knihovny a nástroje již vyvinuté a vydané ostatními. Těch je pro platformu JVM široké spektrum, především díky úspěchu Javy v průběhu let. Její ekosystém je nesrovnatelný – lze jen těžko nalézt platformu s takovou škálou kvalitních knihoven a nástrojů, jakou má JVM. [18]

4.2 JVM kompatibilní jazyky

Pro platformu JVM existuje více programovacích jazyků, které lze pro vývoj zvolit.

Tato sekce objektivně pojednává o vlastnostech vybraných programovacích jazyků kompatibilních s technologií JVM. Jednotlivé jazyky jsou seřazeny dle data jejich prvního vydání a nakonec následuje shrnutí reflektující autorův názor s výběrem konkrétního jazyka.

4.2.1 Java

Java je staticky typovaný, objektově orientovaný jazyk. Obsahuje několik základních primitivních typů, všechno ostatní jsou objekty, ke kterým je přistupováno pomocí referencí. Java nevyžaduje manuální správu paměti, protože paměť automaticky spravuje JVM pomocí algoritmu „Garbage Collectoin“ (sběr odpadu), který periodicky uvolňuje nepoužívané úseky paměti.

První verze jazyka byla vydána, stejně jako její virtuální stroj, v roce 1995 společností Sun Microsystems. Vedoucím designérem jazyku Java byl James Gosling. V současné době vlastní Javu společnost Oracle.

Aktualizace jazyku Java přinesly během let mnoho vylepšení. Významnou verzí je Java 8, díky které Java získala podporu pro prvky funkcionálního programování, kterou následné verze dále rozšířily. Ovšem i přes všechny aktualizace je Java oproti ostatním výřečný jazyk a nejvíce striktní – vše musí být zabaleno ve třídě. [19]

4.2.2 Groovy

Groovy byl první alternativou k jazyku Java. Vznikl v roce 2003 jako dynamicky typovaný plně objektově orientovaný[1] jazyk, který s jazykem Java sdílí celou syntax. Díky tomu je plně kompatibilní s Javou, v tom smyslu, že validní kód v jazyce Java je validní kód v jazyce Groovy.

Hlavním rozdílem oproti Javě je, že to, co je v jazyce Java syntakticky povinné, je v Groovy volitelné. Jazyk Groovy umožňuje vynechávat specifikaci typu, středníky na konci řádky, veřejný modifikátor přístupu a dokonce závorky pro volání metody. Navíc lze kód psát mimo třídu i funkci. Kód se tak stává kratším, ale v případě přehnaného vynechávání typů i závorek může být až nepřehledný.

Zmíněné vynechávání závorek umožňuje jazyku Groovy tvořit „domain specific languages“ (doménově specifické jazyky), zkratkou DSL. Toho využívá nástroj pro sestavování aplikace Gradle. Ten je jedním z hlavních sestavovacích nástrojů v kontextu JVM, navíc je napsaný právě v jazyku Groovy. [18] [19]

4.2.3 Scala

V roce 2004 vytvořil Martin Odersky, profesor informatiky ze Švýcarska, programovací jazyk Scala. Díky jeho předchozí práci na parametrickém polymorfismu s generickými typy v Javě měl již před vydáním jazyku Scala seriózní zkušenosti v oblasti návrhu programovacích jazyků.

Scala je kombinací plně objektově orientovaného a funkcionálního jazyku. Jedná se o staticky typovaný jazyk, ale díky typové inferenci (odvozování typů) není typy potřeba vždy uvádět. Scala dokonce umožňuje definovat vlastní implicitní konverze, čímž se potřeba explicitního specifikování typů ještě snižuje.

Jedním z hlavních cílů jazyku Scala bylo pevnější spojení funkcionálního s objektově orientovaným přístupem, což se profesorovi Oderskymu nepochybně podařilo. „Currying“, „pattern matching“, „immutable data“ či ad-hoc polymorfismus pomocí „type classes“ patří mezi techniky pocházející z funkcionálního světa, které do jazyka Scala skvěle zapadají. Zároveň Scala nabízí lepší řešení pro mnohonásobný polymorfismus (pomocí „traits“).

V neposlední řadě je zajímavostí jazyka Scala možnost zkompilovat její kód do jazyka JavaScript, což umožňuje využití pro tvorbu interaktivních webových aplikací. [18] [19] [20]

4.2.4 Kotlin

Programovací jazyk Kotlin je téměř o dekádu mladší než ostatní zde zmíněné jazyky. Byl vydán v roce 2011 společností JetBrains, která je významným hráčem v oblasti nástrojů a vývojových prostředí pro programátory. K jeho popularitě zajisté přispělo i rozhodnutí společnosti Google učinit ho preferovaným programovacím jazykem pro vývoj aplikací pro Android.

Kotlin je plně objektově orientovaný jazyk s funkcionálními prvky. Jedná se zároveň o multiplatformní jazyk – necílí pouze na platformu JVM, ale též na Web a mobilní platformy Android a iOS. Přestože se jedná o staticky typovaný jazyk, obsahuje podporu i pro dynamické typy.

Hlavním cílem jazyka Kotlin je pragmatičnost. Jeho konstrukty nabízí praktická řešení pro typické vývojářské problémy. To reflektuje například typový systém jazyka, který má pro každý typ dvě varianty – jedna může obsahovat hodnotu null a druhá nikoliv, což umožňuje lepší ošetření časté chyby způsobené převážně programátorem, v kontextu JVM nazývané „NullPointerException“. Dalšími příklady pragmatičnosti Kotlinu jsou konstrukt „corutines“ (koprogramy), který je abstrakcí vláken usnadňující asynchronní programování, nebo možnost jednoduše vytvářet „extension functions“ (rozšiřující funkce) a s jejich pomocí DSL jazyky. Též stojí za zmínku i skvělá kompatibilita jazyka Kotlin s knihovnami v jazyce Java. [18] [19] [21]

[1] všechno je objekt, včetně primitivních typů - ty existují pouze ve verzi zabalené do objektů

4.2.5 Výběr jazyka pro vývoj

Výběr vhodného programovacího jazyka, knihoven či aplikačních rámců zahrnuje mnoho faktorů od rozsahu projektu až po osobní preference. Proto tato závěrečná sekce a zbytek kapitoly reflektuje autorův názor.

Jazyk Java je nejstarší a pravděpodobně stále nejrozšířenější jazyk pro JVM. Z tohoto důvodu pro něj existuje velké množství knihoven a aplikačních rámců, a také je dobře podporován vývojářskými nástroji. Nicméně, jeho syntaxe působí ve srovnání s ostatními jako zdlouhavá, zastaralá a omezující. Proto má smysl pro nový projekt zvolit některou z alternativ.

Jazyk Groovy působí velmi flexibilně díky jeho syntaxi a dynamickému typování, což z něj dělá populární volbu pro tvorbu skriptů, případně na testováni. Na druhou stranu, pro větší projekty je potřeba kvůli jeho flexibilitě stanovit pravidla pro psaní kódu, protože výsledkem může být kód, který vypadá z jedné části jako Java a z druhé jako Groovy.

Funkcionálně objektový jazyk Scala, poskytuje vývojářům silné konstrukty z obou paradigmat. Nicméně je těžší se ho naučit a plně mu porozumět, protože vyžaduje více znalostí od programátorů. Navíc díky velké svobodě v implicitních konverzích a přetěžování operátorů je možné vytvořit velice krátký kód, ovšem těžko srozumitelný.

Nakonec, Kotlin se jeví jako nejlepší volba. Je z uvedených jazyků nejmladší, v důsledku čehož v něm lze vidět převzetí úspěšných prvků z ostatních jazyků, stejně jako poučení se z jejich chyb. Jeho syntax je zlatým středem mezi restrikcí a volností. I když není plně funkcionální, nemá implicitní konverze a neumožňuje tak volně přetěžovat operátory, jako jazyk Scala, stále má výhodu v tom, že zůstává jednoduchým a snadno srozumitelným. Kromě toho, díky podpoře společností Google a JetBrains, pro něj dnes existuje už mnoho nástrojů a spousta knihoven jazyku Java s ním je kompatibilní.

Jazykem pro implementaci praktické části práce bude tedy jazyk Kotlin.

4.3 Aplikační rámce

Aplikační rámec, z anglického „framework“, zajišťuje základní strukturu aplikace s potřebnými nástroji a knihovnami. Rámec obstarává spouštění kódu a často i propojení jednotlivých součástí vyvíjené aplikace. Tento efekt, nazvaný „inversion of controll“ (inverze závislostí), snižuje provázanost komponentů vyvíjené aplikace, což v důsledku vede k menší náchylnosti na změnu. Výhodou použití aplikačního rámce je především urychlení vývoje, nevýhodou je často počáteční seznámení se s fungováním rámce, které může naopak vývoj brzdit.

Oproti tomu knihovna[2] poskytuje znovupoužitelné funkce, třídy a objekty, jež následně volá vyvíjená aplikace využívající knihovnu. [22]

4.3.1 Backend

Přestože standardní knihovna tříd Javy je skvěle vybavená pro webové aplikace, má podobný problém jako Java samotná – je velmi výřečná. Pro tvorbu webových aplikací či jejich API[3] je potřeba mnoho řádků kódu, které se napříč projekty opakují. Proto během let vzniklo v ekosystému JVM velké množství aplikačních rámců. Nejzajímavější z nich jsou uvedeny v tabulce 4.1.

Rozhodnutí použít aplikační rámec bylo učiněno s vizí usnadnění vývoje. Dobrou zprávou je, že vybraný programovací jazyk neomezuje výběr aplikačního rámce, jelikož jsou všechny uvedené kompatibilní s jazykem Kotlin.

[2] v kontextu vývoje software
[3] aplikačních rozhraní z anglického „Aplication Programming Interface“

Tabulka 4.1 Aplikační rámce pro JVM dle zdroje [23]

Název	Rok vydání	Vývojářská firma
Spring	2014	Pivotal
Micronaut	2018	Object Computing
Ktor	2018	JetBrains
Helidon SE	2019	Oracle
Quarkus	2019	Red Hat

4.3.1.1 Spring Boot

Po přečtení srovnání ve zdroji [23] a vlastním uvážením autora byl pro vývoj zvolen aplikační rámec Spring, konkrétně jeho nová edice Spring Boot. Hlavními přesvědčovacími argumenty byly:

Jednoduchá konfigurace S použitím konceptů „starters“ a konvence nad konfigurací, které jsou dostupné v moderním vydání Spring Boot, lze urychlit úvodní konfiguraci a spuštění projektu. Konkurenční aplikační rámce naproti tomu využívají explicitnější konfigurování, které je sice čitelnější, ale zároveň i pracnější.

Dospělost Spring je nejstarším z uvedených aplikačních rámců a patří k nejpoužívanějším. Tento fakt vede k tomu, že existuje velké množství dostupných materiálů a návodů, což je pro vývojáře využívající Spring výhodné. Díky této popularitě je větší poptávka po vývojářích s ním pracujících i na trhu práce.

Flexibilita Jedná se o flexibilní aplikační rámec, který poskytuje řešení pro většinu běžných architektur a úkolů. V důsledku toho se Spring stal oblíbeným mezi komunitami vývojářů. Rámec se také pružně přizpůsobuje požadavkům trhu a mnoho lidí se shoduje na tom, že Spring zůstane významným aplikačním rámcem v platformě JVM i v budoucnu. [24]

Při porovnání s ostatními aplikačními rámci lze považovat rychlost spuštění aplikace používající Spring za nevýhodu. Tento fakt může být způsoben rozsáhlostí poskytovaných funkcionalit a s tím spojenou spotřebou systémových zdrojů. [23]

4.3.2 Frontend

Jelikož má být výsledkem práce webová aplikace pro koncového uživatele, je třeba vytvořit UI[4]. V kontextu webových aplikací existují dva základní způsoby vykreslování UI:

SSR z anglického „Server Side Rendering“ je vykreslování uživatelského rozhraní na straně serveru. Výhody zahrnují:

- Rychlé úvodní načtení stránky, protože webový prohlížeč klienta nemusí zpracovávat skripty v jazyce JavaScript, pomocí kterých následně sestavuje stránku.
- Lepší SEO[5], důsledkem čehož je webová stránka umístěna blíže ve výsledcích vyhledávání.
- Vývoj webové aplikace je díky blízkému spojení vrstev rychlejší a jednodušší.

Nevýhody zahrnují:

- Vysoké zatížení serveru, jež způsobuje zpomalení jeho odezvy při větším množství uživatelů. Díky většímu zatížení je aplikace hůře škálovatelná[6].

[4] z anglického „User Interface“ (uživatelské rozhraní)
[5] z anglického „Search Engine Optimalization“ (optimalizace pro vyhledávače)
[6] což je schopnost systému zvládat rostoucí objem požadavků

- Pomalejší přechod mezi stránkami zaviněný vykreslováním celé stránky vždy od znovu, i když jsou rozdíly mezi nimi minimální.
- Užší závislost mezi uživatelským rozhraním a logikou, v případě, že není striktně dodržováno oddělení těchto vrstev. V případě tvorby nových klientů pro jiné platformy (např. nativní mobilní aplikace) je často potřeba přidat webové API.

CSR z anglického „Client Side Rendering“ je vykreslování uživatelského rozhraní na straně klienta. Výhody zahrnují:

- Minimální zátěž serveru, jelikož jsou stránky renderovány u klienta v prohlížeči, a tak nepředstavují větší zátěž pro server, který aplikaci poskytuje data přes webové API. Z toho plyne i lepší škálovatelnost aplikace.
- Rychlejší renderování stránek po prvním načtení. Při přepnutí mezi stránkami jsou překresleny pouze nové informace. Aplikace tak reaguje rychleji.
- Jasné oddělení klienta a serveru, což umožňuje jednodušeji vytvářet klienty pro další platformy bez nutnosti zasahovat do kódu serveru.

Nevýhody zahrnují:

- Delší čas na první načtení. Při první návštěvě stránky trvá aplikaci déle, než stáhne a zpracuje všechny soubory v jazyce JavaScript, zejména pokud používají zastaralý prohlížeč nebo pomalé připojení k internetu.
- Horší SEO, i navzdory tomu, že vyhledávače zpracovávají i webové stránky vykreslované pomocí JavaScriptu.
- Vývoj je pomalejší, jelikož je vždy potřeba mezi klientem a serverem navrhnout API rozhraní, přes které probíhá komunikace.

Z logiky uvedených informací i z použitého zdroje [25] vyplývá, že pro navrhovanou webovou aplikaci je vhodnější využít CSR. Proto pro něj bylo rozhodnuto i v této práci.

Za nejpoužívanější aplikační rámce pro tvorbu uživatelských rozhraní jsou považovány Ember, Angular, React a Vue [26]. Nicméně vzhledem ke zvolenému programovacímu jazyku Kotlin, který umožňuje kompilaci do JavaScriptu a snaží se být platformově nezávislým, bude tato práce zkoumat jeho aktuální možnosti v oblasti multiplatformních projektů.

Společností JetBrains jsou pro modul Kotlin/JS doporučeny čtyři aplikační rámce: KVision, fritz2, Doodle a Compose for Web [27]. V blízké budoucnosti bude zajímavou volbou Compose for Web, protože je produktem JetBrains. V době psaní práce se však nachází v experimentálním stádiu, navíc bez dokumentace a návodů. Pro tento projekt se nejvíce hodí rámec KVision, který plně využívá potenciál multiplatformního vývoje pro jazyk Kotlin a vypadá stabilně, včetně celkem rozsáhlé dokumentace.

4.3.2.1 Kvision

Aplikační rámec Kvision se prezentuje jako „Fullstack[7] ready“ – součástí rámce jsou inovativní rozhraní s podporou pro připojení serverových aplikačních rámců Ktor, Jooby, Spring Boot, Javalin, Vert.x a Micronaut. Tato rozhraní umožňují vytvářet fullstack aplikace se sdíleným kódem pro datový model a obchodní logiku. [28]

Po prozkoumání možností aplikačních rámců pro tvorbu klientské části aplikace byl zvolen KVision. Hlavní důvody zahrnují často aktualizovanou a celkem rozsáhlou dokumentaci, včetně několika příkladných projektů, dále napojení na zvolený Spring Boot.

[7] je termín pokrývající frontend (zahrnující implementaci UI aplikace, tedy klienta) i backend (zahrnující implementaci aplikační logiky a dat, tedy serveru)

Kapitola 5
Návrh

Kapitola se v úvodu zabývá důležitostí návrhu při vývoji software. Následují sekce věnující se návrhu architektury pro serverovou a klientskou část aplikace. V jednotlivých sekcích jsou popsány architektonické vzory použité pro návrh.

5.1 Návrh architektury aplikace

Návrh architektury je důležitým krokem při vývoji softwaru, především u větších projektů. Klíčovým krokem je rozdělit si aplikaci na menší části, z nichž každá má danou zodpovědnost. To napomáhá splnit cíl návrhu architektury, kterým je dosáhnutí toho, aby byla aplikace srozumitelná, rozšířitelná a udržovatelná. Srozumitelnost vede ke snadné lokalizaci chyb, což umožňuje efektivní provedení oprav i úprav aplikace. Rozšířitelnost zajišťuje, že změny v jedné části systému nemají nežádoucí dopady na nesouvisející části. Udržovatelnost vychází z předchozích vlastností, jelikož z dobré srozumitelnosti a rozšířitelnosti často vyplývá i snadná udržovatelnost. [29]

5.1.1 Server

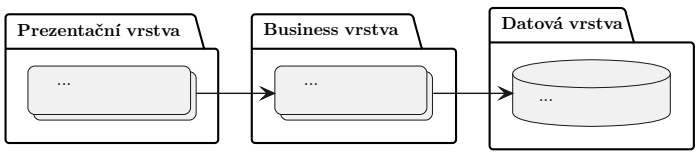
Pro realizaci architektury serveru byla zvolena třívrstvá architektura, kterou popisuje tato sekce.

Třívrstvá architektura je častou volbou pro podnikové aplikace, protože je robustní a dobře odděluje zodpovědnosti. Architektura se skládá ze 3 vrstev znázorněných v obrázku 5.1:

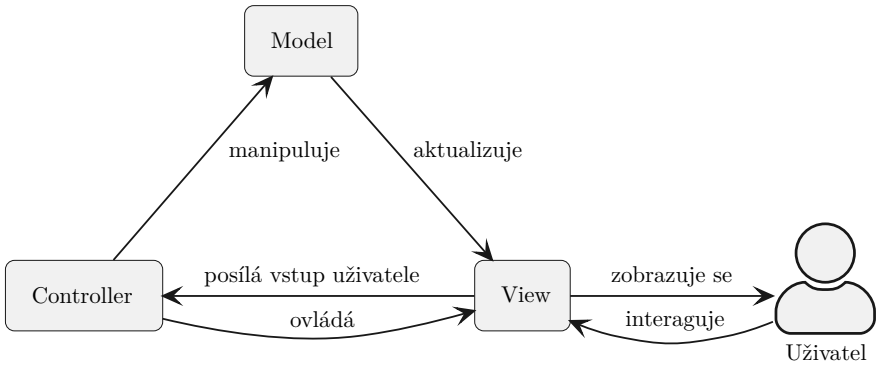
Datová vrstva tvoří spodní vrstvu modelu, zajišťující persistenci dat. Její zodpovědností je poskytnutí rozhraní pro ukládání a načítání dat. Typicky je realizována pomocí databáze.

Business vrstva je střední vrstvou modelu se závislostí na datové vrstvě. Obaluje datovou vrstvu business logikou, která může zahrnovat autentizaci, autorizaci a podobně.

Prezentační vrstva je nejvyšší vrstvou se závislostí na business vrstvě. Obstarává logiku pro příjem požadavků a následnou prezentaci získaných výsledků. [30]



Obrázek 5.1 Třívrstvá architektura



Obrázek 5.2 Návrhový vzor Model-View-Controller

5.1.2 Klient

Vybraný aplikační rámec KVision nediktuje ani nedoporučuje dodržování konkrétního architektonického vzoru. Po prozkoumání poskytnutých ukázkových projektů využívajících rámec KVision byl zvolen návrhový vzor MVC (Model-View-Controller), který bude představen v této sekci. Posuzovanými alternavami k MVC byli MVVM (Model-View-Viewmodel) a MVP (Model-View-Presenter). Obě tyto alternativy jsou z MVC odvozeny, a výrazně se neliší, proto se jimi tato práce nezabývá.

Návrhový vzor zvaný Model-View-Controller, zkratkou MVC, je se často využívá v aplikačních rámcích pro tvorbu uživatelského rozhraní. Klíčovou vlastností MVC (i z něj odvozených alternativ) je oddělení zobrazovací logiky od business logiky. To přispívá ke všem zmíněným cílům návrhu architektury. MVC je tvořen 3 vrstvami, jejichž vazby znázorňuje obrázek 5.2:

Model spravuje business logiku a data aplikace. V případě změny dat notifikuje vrstvu View, která zareaguje aktualizováním UI, proto lze říci, že aktualizuje View. V některých případech může dokonce notifikovat i vrstvu Controller pro aktivování jiné logiky [31]. Vrstva model by měla být nezávislá na vrstvách View a Controller.

View definuje rozložení obrazovky UI a způsob zobrazení dat. Stará se o překreslování dat, v případě, že dojde k jejich změně. Vrstva View by měla být nezávislá na vrstvách Controller a Model. Je povolena pouze závislost na datech poskytovaných vrstvou Model.

Controller má na starost propojení vrstev View a Model. Pokud má být manipulováno s daty[1], Controller zařizuje předání požadavku na vrstvu Model. Zároveň může i přímo ovládat View, například při změně obrazovky UI. [31] [32]

[1] což zahrnuje jejich čtení, změnu atd.

Kapitola 6
Realizace

Kapitola se zabývá popisem činností provedených během realizace praktické části. V úvodu je přiblížena příprava a konfigurace projektu. Následuje popis realizované architektury a jednotlivých modulů projektu. Nakonec se kapitola věnuje dokumentaci, testování a nasazení.

Vzhledem k rozsáhlosti cílů této bakalářské práce i provedené analýzy bylo po dohodě s vedoucím rozhodnuto, že se implementační část práce bude soustředit pouze na realizaci požadavků pro vytvoření katalogu táborů.

6.1 Příprava

Před zahájením práce na praktické části bylo potřeba se seznámit se zvolenými technologiemi. Dosavadní zkušenosti autora (z oblasti analyzovaných technologií) zahrnovaly pouze dobrou znalost jazyků Java, Kotlin a Scala, poté základy webů (CSS, html, javascript) a pouze elementární znalosti technologií Spring, JPA a Hibernate, které budou povrchově popsány v následujících kapitolách. Proto bylo potřeba, seznámit se alespoň základně s použitou technologií Kotlin Multiplatform[1] pro víceplatformní projekty a poté s aplikačním rámcem KVision[2]. Zde se naplno ukázala nevýhoda používání nových technologií, které jsou využívány pouze menším množstvím vývojářů – neexistuje moc návodů ke studiu, případně odpovědí v komunitách a diskuzních webech.

Následujícím krokem přípravy byla volba a instalace potřebných vývojových nástrojů.

Jako vývojové prostředí, zkráceně IDE, z anglického „Integrated Development Environment“, bylo zvoleno IntelliJ Idea[3], které je od společnosti JetBrains stejně jako jazyk Kotlin, díky čemuž s ním velice dobře spolupracuje. S tímto IDE měl autor práce díky předchozím absolvovaným předmětům pozitvní zkušenosti, a proto se jej rozhodl využít.

Na sestavování projektu byl zvolen nástroj Gradle[4], jelikož je momentálně jediným podporovaným nástrojem pro multiplatformní projekty [33]. Gradle je možné nainstalovat samostatně, ale vzhledem k tomu, že je součástí zvoleného IDE, tak to není potřeba. Použitým jazykem pro Gradle byl zvolen Kotlin, jelikož je v několika málo návodech a ukázkových projektech též používán. Navíc je výhodou, že je pro celý vývoj použit jeden programovací jazyk.

[1] https://kotlinlang.org/lp/multiplatform/
[2] https://kvision.io/
[3] https://www.jetbrains.com/idea/
[4] https://gradle.org

6.2 Vytvoření projektu

Po volbě nástrojů a prostudování dostupných materiálů k rámci KVision byl s pomocí pluginu KVision Project Wizard[5] vytvořen prázdný multiplatformní Kotlin projekt s navolenými KVision moduly. Vytvořený projekt, cílící na platformy JS a JVM, je rozdělen do tří složek, v kontextu použitého sestavovacího nástroje Gradle nazývaných moduly:

backendMain je modul obsahující zdrojové kódy pro platformu JVM, proto zde lze využívat knihovny[6] z ekosystému JVM.

commonMain je modul s kódem nezávislý na platformě. To je důvod, proč je zde možné využívat pouze knihovny napsané v jazyce Kotlin, které nemají závislost na konkrétní platformě. Na kódu z této složky mají závislost obě platformy a kompiluje se dvakrát, tedy pro každou platformu.

frontendMain je modul, jenž zahrnuje zdrojové kódy pro platformu JS (JavaScript), což umožňuje zde využívat knihovny cílené na tuto platformu. Ty mohou být napsané buď v Kotlinu nebo v jazyce JavaScript.

KVision, jakožto fullstack aplikační rámec, poskytuje kód do všech tří modulů, i když primárně se jedná o rámec pro tvorbu UI. Proto je většina jeho funkcionality cílena na platformu JS.

6.3 Proof of Concept

Pro zahájení realizace bylo potřeba do projektu přidat všechny zvolené technologie a s jejich pomocí vytvořit jednoduchou ukázku pro vyzkoušení, zda je s kombinací těchto technologií možné projekt vyvíjet. Toto ověření proveditelnosti návrhu zvané „Proof of Concept“, zkratkou PoC, se v oblasti vývoje software běžně dělá [34], protože napomáhá včasnému odhalení chybného návrhu, například při zvolení nekompatibilních technologií.

6.3.1 Přístup k databázi

Ve vygenerované projektové šabloně byl použit reaktivní přístup k databázi pomocí API R2DBC. Vzhledem k tomu, že již předchozí zvolené technologie byly pro autora práce novinkou, se kterou se musel seznámit, rozhodl se zde využít technologie, se kterými se již setkal. Proto byl pro napojení k databázi využit modul Spring Data JPA, který integruje a usnadňuje práci s JPA. Vysvětlení významu zmíněných technologií jsou pro přehlednost oddělena do samostatných sekcí.

Prvním krokem PoC bylo tedy změnit rozhraní využívané pro přístup do databáze z R2DBC na Spring Data JPA.

6.3.1.1 JPA

„Java Persistence API“, zkratkou JPA, je specifikace API v ekosystému JVM, pro ukládání a mapování objektů do relační databáze, anglicky nazývaném „Object Relation Mapping“ se zkratou ORM. Ve světě JVM je často využíváno, protože zjednodušuje typický implementační problém – aplikace využívají pro reprezentaci dat objekty naproti tomu relační databáze řádky v tabulkách. Tyto dva způsoby reprezentace je potřeba převádět mezi sebou.

[5] https://plugins.jetbrains.com/plugin/16533-kvision-project-wizard
[6] v kapitole Realizace má tento pojem obecnější význam, knihovnou může být i aplikační rámec

6.3.1.2 Spring Data JPA

Modul Spring Data JPA umožňuje napojení JPA do rámce Spring a usnadňuje jeho konfiguraci, například tím, že pro implementaci JPA dodává Hibernate, což je nejpoužívanější implementace JPA. [35]

6.3.1.3 Problém: Kotlin s JPA

Při nastavování JPA entit se začaly objevovat první problémy. Při zkoumání příčiny byl nalezen zajímavý zdroj[7] zabývající se konfigurací JPA při použití Kotlinu. Problém byl způsoben tím, že JPA vyžaduje, aby entity bylo možné rozšiřovat pomocí dědičnosti. V opačném případě není možné využít techniky „Lazy loading“, která umožňuje načítat vazby na objekty až v době, kdy je k nim přistupováno. Jazyk Kotlin má oproti jazyku Java ve výchozím stavu třídy i jejich atributy nastavené jako neměnné (v kódu jazyka Java označeny „final“), proto je potřeba explicitně specifikovat možnost dědičnosti, což se v jazyce Kotlin dělá pomocí „open“. Video se též zabývá problematikou použití datových tříd jazyka Kotlin s různými typy kolekcí a závěrem doporučením tuto konstrukci nepoužívat.

Řešení problému bylo provedeno dle popsaných kroků.

6.3.2 Zabezpečení aplikace

Z provedené analýzy aplikace vyplývá striktní zabezpečení, například v sekci „Evidence táborů a organizací“ je uvedeno: „Správce organizace může v systému interagovat (mazat, editovat) pouze s tábory a turnusy, které jsou pořádány jím spravovanou organizací.“

Následujícím krokem PoC bylo ověřit, že je možné provést zabezpečení serverového API vytvořeného s pomocí rámce KVision. Zabezpečení musí na vybrané adresy povolit přístup pouze přihlášenému uživateli. Dále je potřeba dle přihlášeného uživatele a požadované akce rozhodnout, zda bude provedení akce vykonáno nebo zamítnuto.

Součástí zabezpečení aplikace je též jeho ošetření v klientovi. Proto bylo třeba zabezpečit, že v případě vstupu uživatele na zabezpečenou url adresu nevrátí klient error, ale zobrazí přihlašovací okno.

6.3.2.1 Problém: Zabezpečení serveru

V dokumentaci rámce KVision je zabezpečení pro jednotlivé backendové rámce věnováno minimální množství prostoru. Z toho důvodu se podle něj nepovedlo zabezpečení replikovat do projektu. Proto se autor textu obrátil na ukázkové projekty, aby pochopil konfiguraci zabezpečení z nich. Zde se výrazně projevilo riziko volby málo používaných knihoven, jelikož na internetu nebyly k nalezení žádné rady či jiné zdroje vysvětlující konfigurace v těchto projektech.

Pokus replikovat zabezpečení do aktuálního projektu, dle jednoho z ukázkových projektů na githubu[8] autora rámce KVision, též selhal. Vytvořené přihlašování do aplikace přestalo fungovat. Server nevracel žádnou chybu, a proto bylo velice obtížné vystopovat problém.

Nakonec se problém povedlo vytrasovat při pokusu zkopírovat zmíněný ukázkový projekt a postupně do něj přenést již vytvořený kód. Chyba byla ve špatně nastavené adaptaci JPA do reaktivní služby „ReactiveUserDetailsServiceImpl“, která obstarává přihlášení. Použitá implementace JPA je blokující (blokuje vlákno, než se vrátí odpověď z databáze), ale vyžaduje se neblokující přístup (který vlákno neblokuje, ale provede notifikaci až po vrácení dat). To způsobilo, že služba neprovedla přihlášení, protože nepočkala na výsledek z databáze.

Problém byl tedy nakonec vyřešen opravou adaptace blokující technologie a do dokumentačních komentářů kódu bylo doplněno vysvětlení aktuálního kódu. Dokumentační komentáře byly

[7] https://www.youtube.com/watch?v=a_6V8xwiv04
[8] https://github.com/rjaros/kvision-examples/tree/master/addressbook-fullstack-spring-boot

též rozšířeny o důležité odkazy na konkrétní místa v dokumentacích použitých technologií, jež tento problém zmiňují.

6.3.2.2 Problém: Zabezpečení klienta

Pro ošetření zabezpečení klienta byla součástí rámce KVision poskytnutá třída implementující „SecurityMgr“. Tato implementace byla velmi jednoduchá a z uživatelského hlediska velmi nepřívětivá. Při přístupu na url, která vyžadovala zabezpečení, vyskočilo přihlašovací okno. To by bylo v pořádku, ovšem aplikace neumožňovala okno zavřít nebo se jinak vrátit na domovskou obrazovku. Uživatel se tedy mohl buď přihlásit, nebo zavřít celou stránku. Při přidání možnosti zavřít okno vyhodila klientská aplikace výjimku a přestala reagovat. I při odchycení výjimky, se nepodařilo implementovanou třídu použít tak, aby měla přívětivé chování pro uživatele a umožnila vrácení na domovskou obrazovku. Dalším problémem implementace bylo, že neposkytovala způsob, jak vrátit informaci, zda je již uživatel přihlášený, která byla potřeba například pro nastavení navigace aplikace. V neposlední řadě porušovala návrh zvolené architektury MVC, jelikož třída (model) pro přihlášení a registraci měla přímou závislost na okně (view) pro přihlášení a registraci.

Kvůli zmíněným problémům poskytnuté implementace bylo potřeba implementovat vlastní zabezpečení klienta obstarávající přihlášení a registraci. To respektuje architekturu MVC – protože view „SecurityWindow“ ani model „SecurityModel“ na sobě nemají přímou závislost. Propojuje je „SecurityController“, který navíc poskytuje metody pro vyvolání okna. Ošetření, zda je uživatel pro danou URL přihlášen, provádí controller „AppRouting“, který deleguje požadavky na zmíněný controller či model.

Řešení problému bylo realizováno pomocí vlastní implementace pro ošetřování bezpečnosti u uživatele.

6.3.3 Vyhodnocení PoC

Vypracování PoC ukázky čelilo několika nelehkým problémům. Nejzajímavější z nich byly popsány včetně jejich řešení v předcházejících sekcích. Přestože vyřešit nastalé problémy nebylo snadné, nakonec se to podařilo.

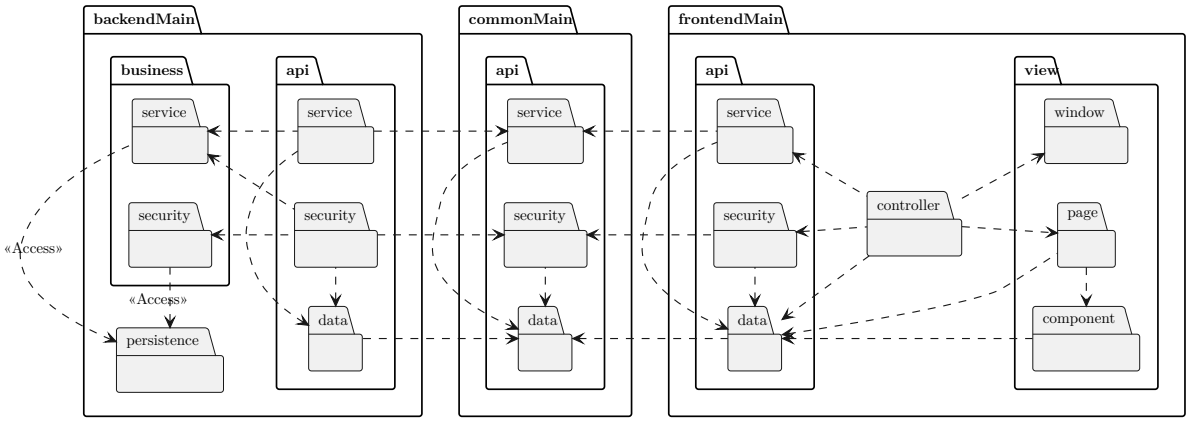
Následující podkapitoly řeší jednotlivé aspekty implementované části aplikace a prezentují její vybrané části.

6.4 Realizace architektury

Provedený návrh stanovil použití třívrstvé architektury pro server a MVC architektury pro klienta. Obě architektury byly dodrženy, což lze vidět v diagramu balíčků 6.1, který mapuje strukturu závislostí v projektu. Následuje popis modulů a balíčků aplikace. Pro přehlednost jsou pro architekturu nedůležité balíčky v obrázku 6.1 vynechány:

commonMain je modul obsahující sdílený kód. Skládá se z balíčků:

- **api** definuje kontrakt mezi klientem a serverem. Obsahuje 3 balíčky:
  - **data** je balíček definující objekty pro přepravu dat mezi klientem a serverem, anglicky nazývané pojmem „Data Transfer Object“ se zkratkou DTO. Kromě definice DTO obsahuje i funkce pro jejich validaci, což umožňuje validovat DTO na klientovi i serveru.
  - **service** definuje rozhraní pro služby mezi klientem a serverem, jehož součástí jsou i definice cílových adres („endpointů“).
  - **security** definuje rozhraní a cílové adresy pro přihlášení a registraci.
- **simplevalidation** implementuje jednoduchou multiplatformní knihovnu pro validaci dat. Třídy v této knihovně nemají závislost mimo tento balíček, díky čemuž je znovupoužitelná.



Obrázek 6.1 Diagram balíčků aplikace

backendMain je modul realizující server pomocí třívrstvé architektury:

persistence tvoří datovou vrstvu. Obsahuje definici datových entit pro ORM a dále Spring JPA repozitáře pro přístup k datům.

business reprezentuje business vrstvu, která vytváří abstrakci nad poskytovanými daty – závislost na perzistentní vrstvě není vystavena, což je znázorněno vazbou «Access». Obsahuje implementaci zabezpečení aplikace (security), realizaci business logiky (service), mapování entit do DTO (mapping) a plnění databáze (runner).

api zastupuje prezentační vrstvu. Jednotlivé balíčky uvnitř odpovídají balíčkům ve sdílené vrstvě a implementují rozhraní v nich. Tyto implementace tvoří webové API.

frontendMain je modul realizující klienta pomocí MVC architektury

view obsahuje uživatelské rozhraní aplikace zahrnující komponenty (component), stránky (page) a okna (window). Třídy v balíčcích page a component jsou závislé pouze na datech modelu (api.data), což neporušuje architekturu MVC – vrstva view potřebuje závislost na datech, které má zobrazit.

controller obsahuje třídy propojující vrstvy view a model. Dále obsahuje zmíněné ovladače pro navigaci aplikace a ovládání zabezpečení.

api reprezentuje vrstvu model. Jednotlivé balíčky uvnitř odpovídají balíčkům ve sdílené vrstvě a obalují poskytnutou implementaci rozhraní, která obstarává posílání požadavků na webové API a zpracování odpovědí.

kvisionext obsahuje rozšíření pro rámec KVision. Stejně, jako v případě balíčku simplevalidation, nemají třídy v tomto balíčku závislost mimo něj, díky čemuž je znovupoužitelná.

6.5 Implementace webového API

Webové API je rozhraní pro webové prostředí, které slouží k přístupu a manipulaci s daty v aplikacích. Na webu jsou často zastoupeny API vycházející ze specifikace REST API.

REST API je bezstavové webové API skládající se z jednotlivých zdrojů, které mají své jedinečné adresy. Tyto zdroje jsou pak přístupné pomocí HTTP metod. Při implementaci REST API je vhodné využívat stavových kódů a metod HTTP protokolu, které slouží k označení výsledku požadavku. [36]

Implementace webového API vychází ze specifikace REST API s tím rozdílem, že není plně bezstavové. Aplikace udržuje pomocí relace, anglicky „session“, data přihlášeného uživatele. Dalším rozdílem je, že metoda GET se používá jen v případě absence parametrů. Důvodem je, že rámec KVision, pomocí kterého je API implementováno, neumožňuje v použité verzi s metodou GET předávat parametry. Tento problém je řešen přidáním cesty „/get“ na konec adresy a využitím metody PUT, viz řádky 4 až 6 a 15 až 16 ve výpisu kódu 6.1.

Zkrácený výpis kódu 6.1 z modulu commonMain demonstruje ukázku sdíleného rozhraní:

Výpis kódu 6.1 Multiplatformní rozhraní z modulu commonMain

```
1  package cz.cvut.fit.campsystem.api.service
2
3  object Route {
4      private const val GET = "/get"
5      const val ORGANIZATION = "organization"
6      const val ORGANIZATION_GET = ORGANIZATION + GET
7  }
8
9  @KVService
10 interface ICampOrgService {
11
12     @KVBinding(Method.POST, Route.ORGANIZATION)
13     suspend fun create(campOrg: CampOrgDTO): ContractDTO
14
15     @KVBinding(Method.PUT, Route.ORGANIZATION_GET)
16     suspend fun readOne(id: Int): CampOrgDTO
17
18     @KVBinding(Method.PUT, Route.ORGANIZATION)
19     suspend fun update(campOrg: CampOrgDTO): Boolean
20
21     @KVBinding(Method.DELETE, Route.ORGANIZATION)
22     suspend fun delete(campOrgId: Int): Boolean
23 }
```

Díky anotaci KVService dokáže rámec KVision rozpoznat, že se jedná o specifikaci API. Anotace KVBinding poté umožňuje pro každou metodu definovat HTTP metodu a cestu. Rozhraní ICampOrgService je pro vytvoření API potřeba implementovat v serverové části aplikace, což ukazuje zkrácený výpis kódu 6.2.

Výpis kódu 6.2 Implementace rozhraní v modulu backendMain

```
package cz.cvut.fit.campsystem.api.service

@Service
@Scope(value = ConfigurableBeanFactory.SCOPE_PROTOTYPE)
actual class CampOrgService(
    override val serverRequest: ServerRequest,
    private val campOrgDAO: CampOrgDAO,
) :
    ICampOrgService, WithLoggedInUser {

    override suspend fun create(campOrg: CampOrgDTO): ContractDTO {
        campOrg.isValidOrException()
        return withContext(Dispatchers.IO) {
            campOrgDAO.create(getLoggedInUser(), campOrg.toCampOrg(null))
                ?.toContractDTO().accessDeniedOnNull()
        }
    }

    override suspend fun readOne(id: Int): CampOrgDTO {
        return withContext(Dispatchers.IO) {
            campOrgDAO.readById(getLoggedInUser(), id)
                ?.toCampOrgDTO().accessDeniedOnNull()
        }
    }
    ...
}
```

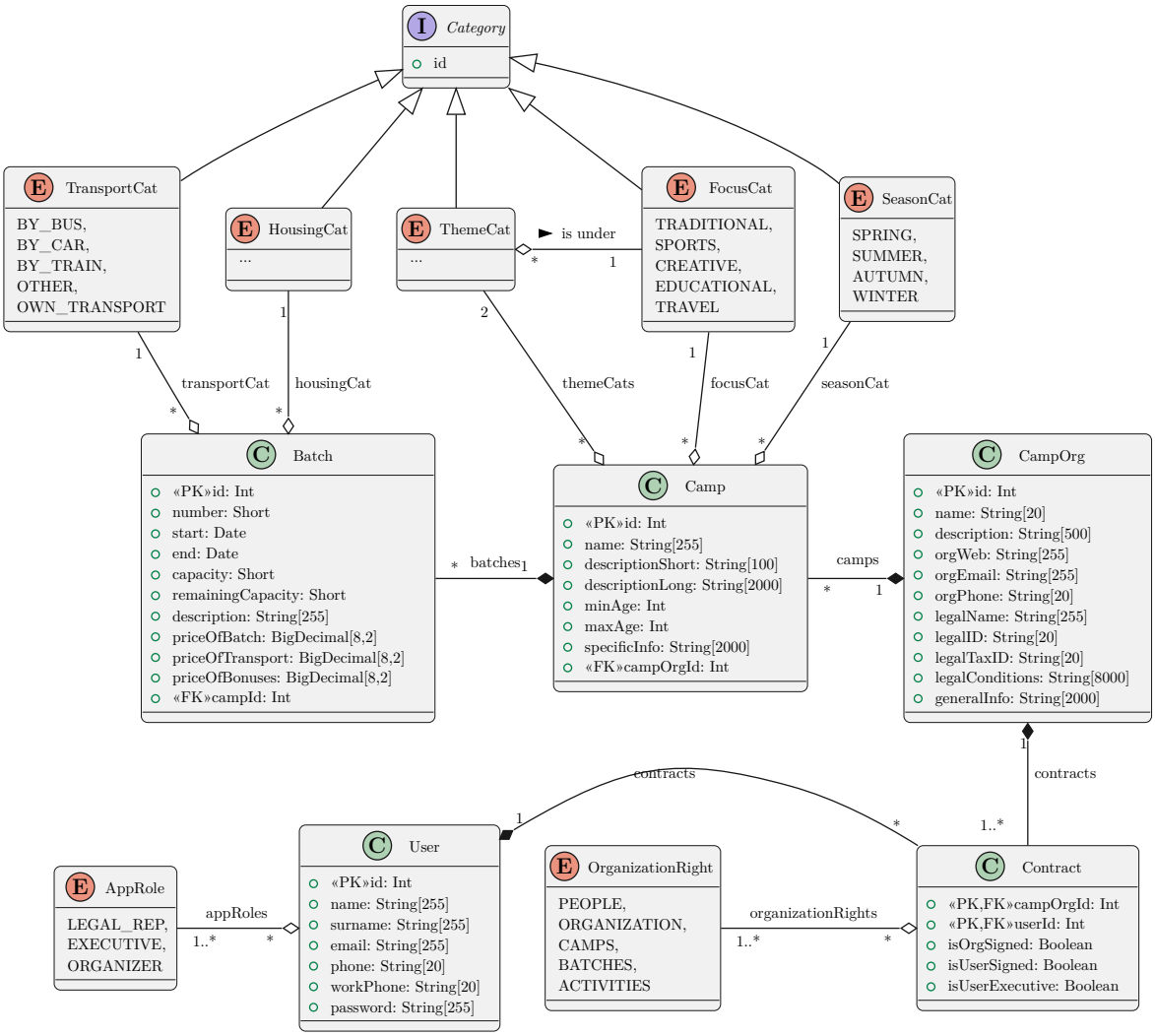
Třída znázorněná ve výpisu kódu 6.2 CampOrgService implementuje multiplatformní rozhraní ICampOrgService. Na začátku metod se zavolá validace objektu a následně předá vyhodnocení dat požadavku i data přihlášeného uživatele do business vrstvy, konkrétně do CampOrgDAO.

Rámec KVision zajistí, že pokud sedí HTTP metoda, url cesta a povedou se deserializovat argumenty pro metodu, předá se vyhodnocení požadavku implementované metody. Například při použití HTTP metody PUT na cestu /organization/get a odeslání ve formátu JSON dat znázorněných ve výpisu 6.3 se na instanci třídy CampOrgService zavolá metoda readOne(1).

Výpis kódu 6.3 Ukázka JSON dat pro dotaz

```
{
  "id": 1,
  "method": "/kv/organization/get",
  "params": ["1"]
}
```

Na klientské části aplikace je poté možné obdržet od rámce KVision automatickou implementaci, která umí na toto API odeslat požadavek a vrátit jeho odpověď, případně vyhodit výjimku. Serializace a deserializace argumentů pro metodu, návratových hodnot či případné výjimky a její přehození zajišťuje rámec KVision. Jednoduše by se dalo říct „stačí zavolat metodu serveru“.



Obrázek 6.2 Diagram tříd pro reprezentaci databáze

6.6 Realizace databáze

Realizace databáze v této bakalářské práci byla provedena na základě vypracovaného doménového modelu. Návrh databáze se omezuje na třídy pro realizaci funkcionalit katalogu.

Vytvoření databáze provede použitý rámec Spring, konkrétně modul Spring Data JPA, který využívá technologii Hibernate. To umožňuje automatické vytvoření databázového schématu z definovaných tříd. Tento přístup usnadňuje vývoj a údržbu databáze, a zároveň minimalizuje riziko chyb spojených s ručním vytvářením databázových tabulek.

Pro účely ukázky byla zvolena neprodukční H2 databáze, kterou není potřeba instalovat či nasazovat. Díky tomu bylo možné rychle ověřit funkčnost a správnost návrhu databáze, aniž by bylo nutné provádět složitější konfiguraci a nasazení do produkčního prostředí. Zároveň lze tuto databázi pro účely nasazení jednoduše vyměnit za jinou, stejně tak jako vypnout automatické vytvoření databáze, prostou změnou konfigurace.

Databáze se vytváří z tříd označených anotací „Entity“, které jsou znázorněny v diagramu 6.2 se značkou (C) v hlavičce. Tnačka (E) reprezentuje konstanty („enum“) a značka (I) rozhraní („interface“).

6.7 Uživatelské rozhraní

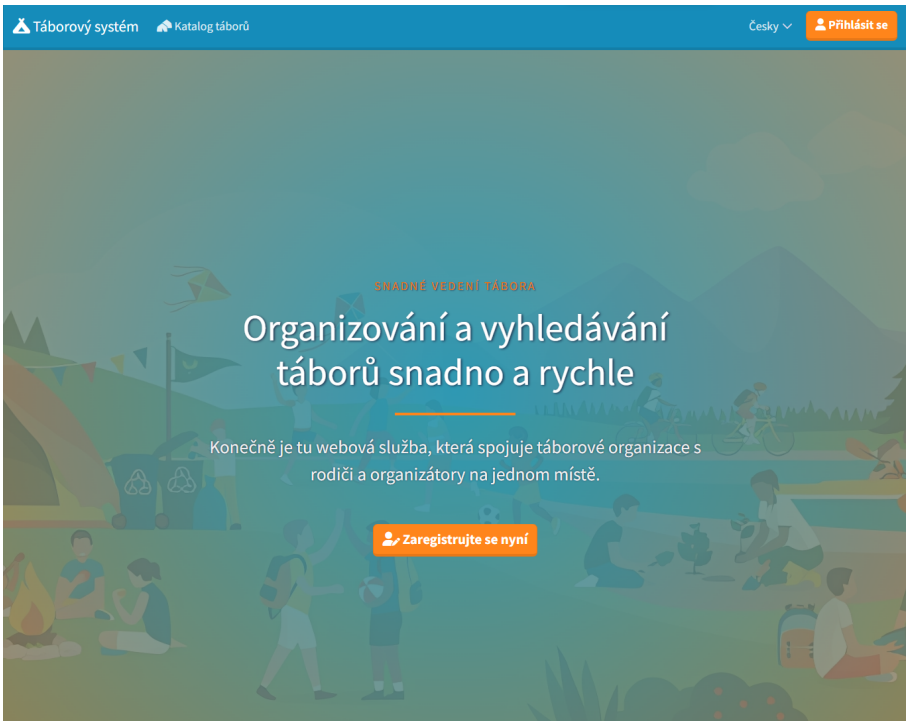
Tvorba uživatelského rozhraní je důležitým krokem vývoje webové aplikace. Z důvodu použití rámce KVision, pro který nejsou k dispozici hotová rozložení obrazovek, bylo potřeba navrhout a sestavit obrazovky z poskytnutích základních komponent. Některé komponenty, které byly v rámci této práce vytvořeny, lze opakovaně použít a proto byly umístěny do balíčku KVisionext.

Navržení rozložení obrazovek je sice pro vývojáře náročnější a vyžaduje pečlivou práci s CSS. Nicméně díky této flexibilitě lze dosáhnout vysoké úrovně přizpůsobení uživatelského rozhraní konkrétním potřebám aplikace.

Uživatelské rozhraní je implementováno ve dvou jazycích – v Češtině a v Angličtině. Rozhodnutí implementovat více jazyků bylo kvůli tomu, aby aplikace neobsahovala české texty v kódu. Všechny texty v kódu jsou proto v angličtině a součásti aplikace je překladový slovník, který dokáže při změně jazyka všechny texty přeložit.

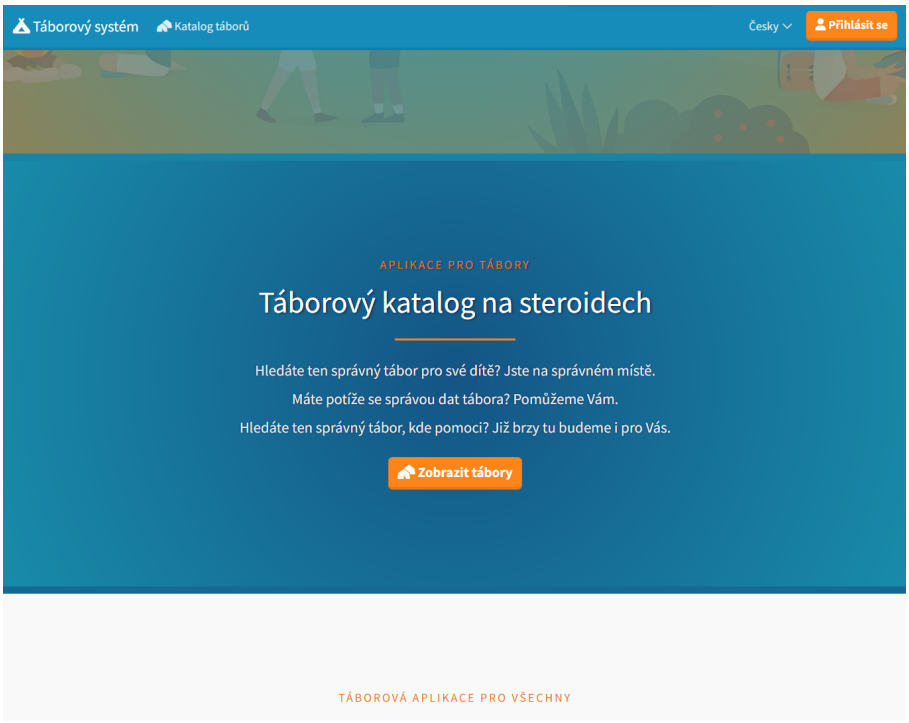
Navržená a realizovaná byla úvodní obrazovka. Ta se skládá ze 3 sekcí, které v krátkosti popisují službu, přibližují, komu je určena a popisují její vlastnosti:

- První sekce v obrázku 6.3 se snaží uživatele upoutat a v krátkosti mu sdělit, v čem je služba prospěšná. Zároveň obsahuje tlačítko pro registraci, pro vyvolání akce.
- Druhá sekce, v obrázku 6.4 o něco rozvádí pro koho je služba určena a obsahuje odkaz na katalog táborů.
- Třetí sekce, v obrázku 6.5, popisuje nejpodrobněji zamýšlené funkcionality aplikace.

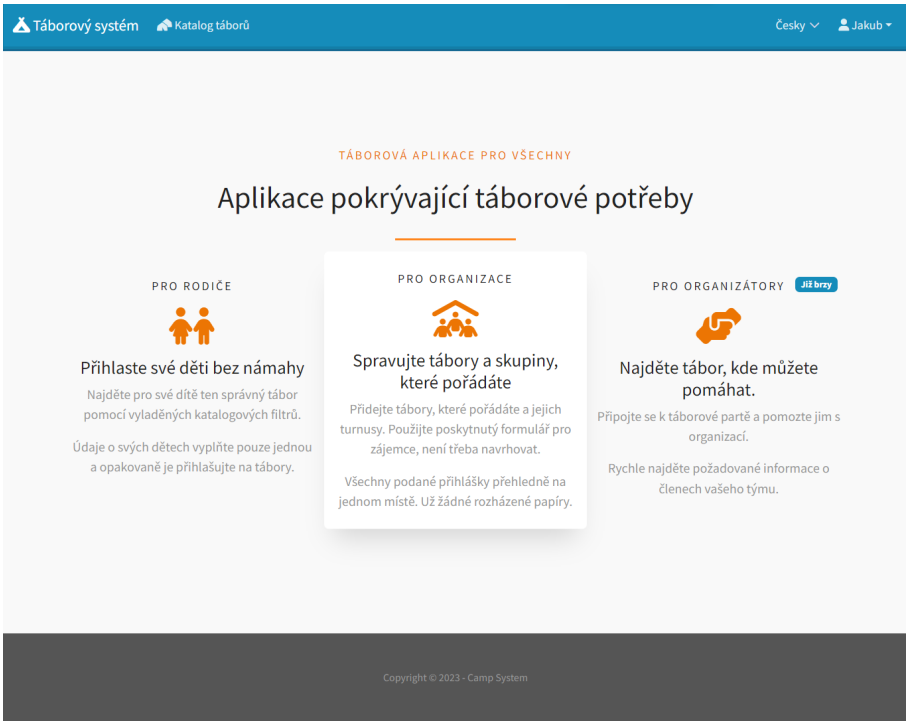


Obrázek 6.3 Navržená domovská obrazovka - první sekce

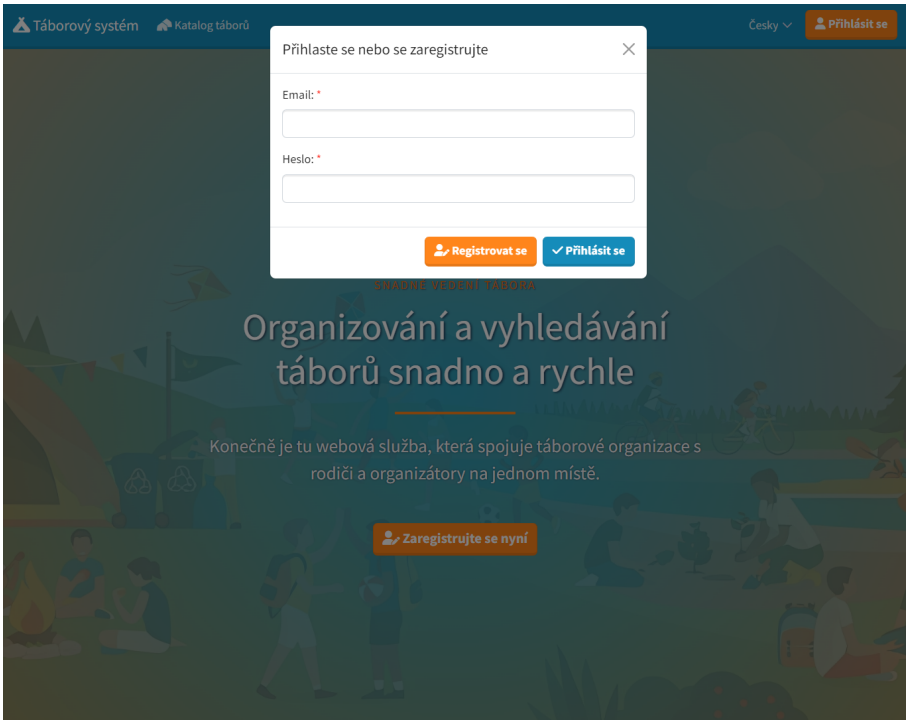
Všechny formuláře mají implementované validace. Validace je realizována v modulu commonMain, aby ji mohl využít klient i server. Obrázek 6.8 ukazuje, jak se jednotlivá pole formuláře zabarví, pokud jejich validace neuspěje. V případě neúspěchu validace formuláře nedojde k jeho odeslání.



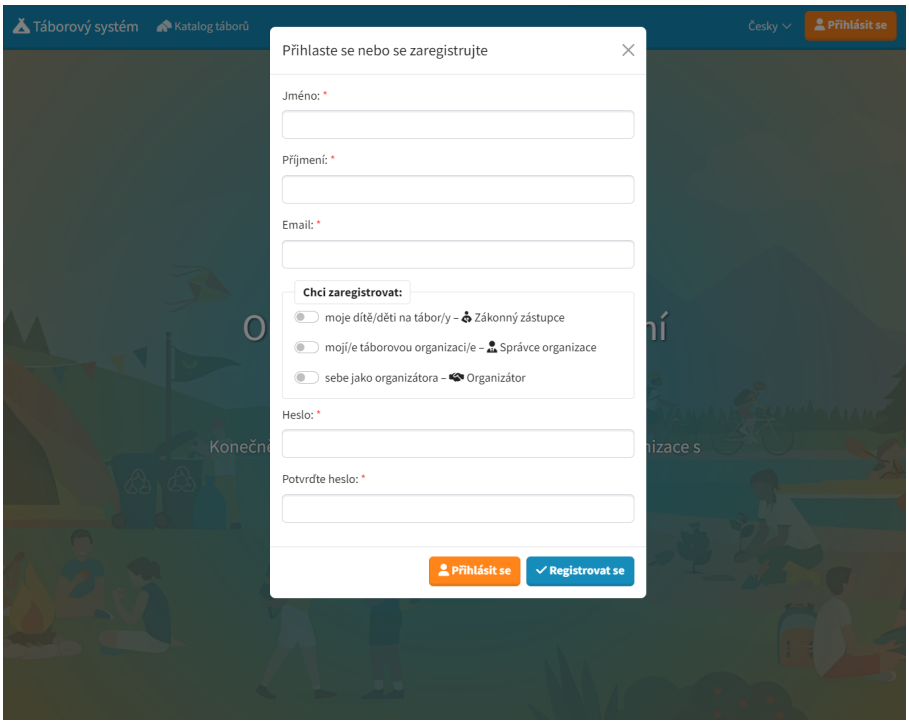
Obrázek 6.4 Navržená domovská obrazovka - druhá sekce



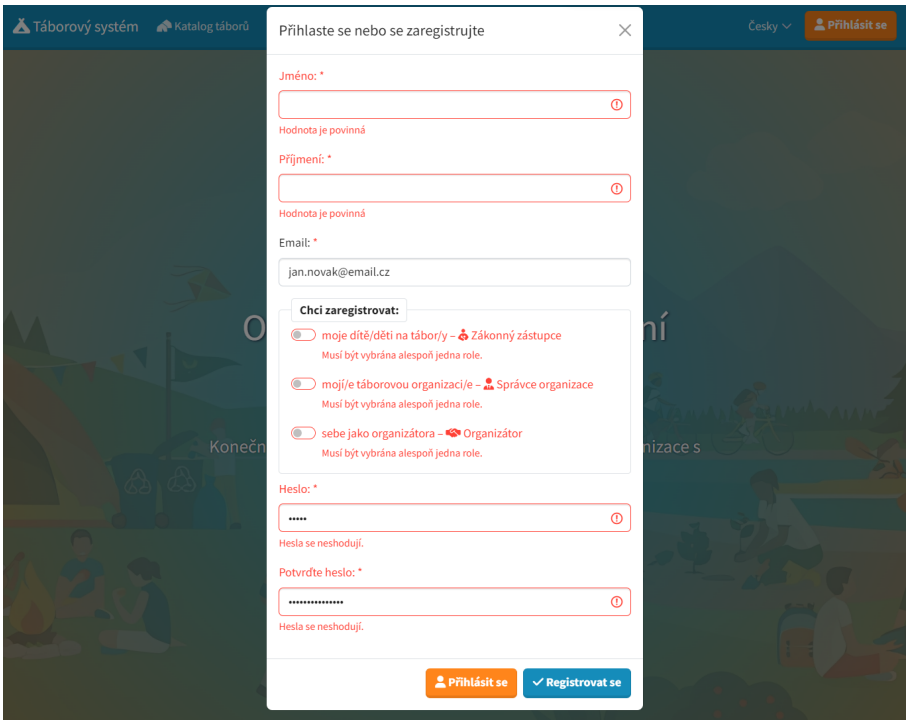
Obrázek 6.5 Navržená domovská obrazovka - třetí sekce



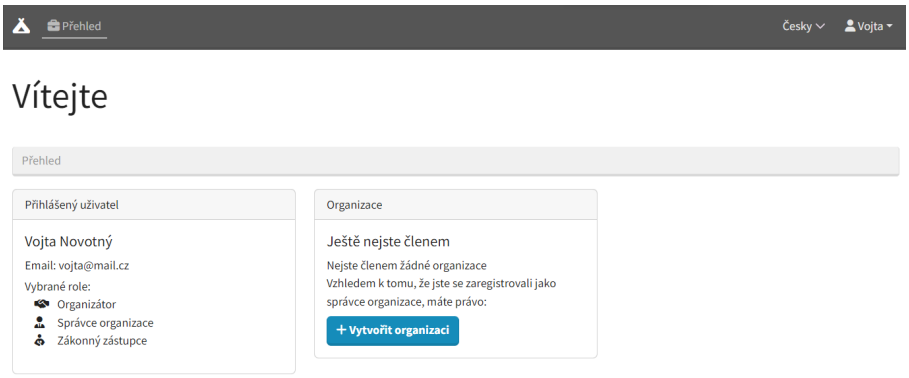
Obrázek 6.6 Formulář pro přihlášení



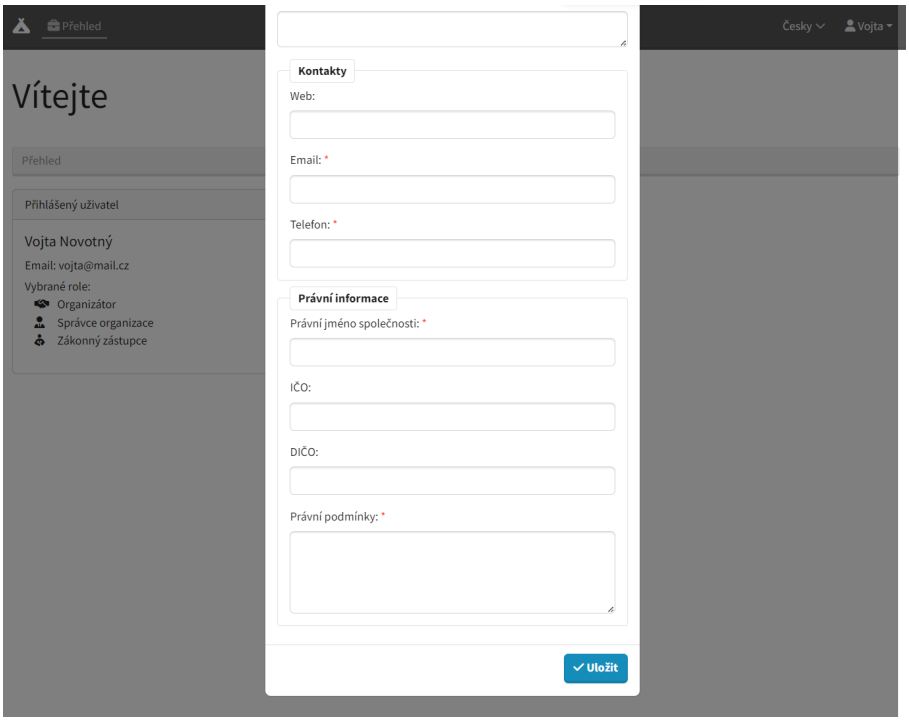
Obrázek 6.7 Formulář pro registraci



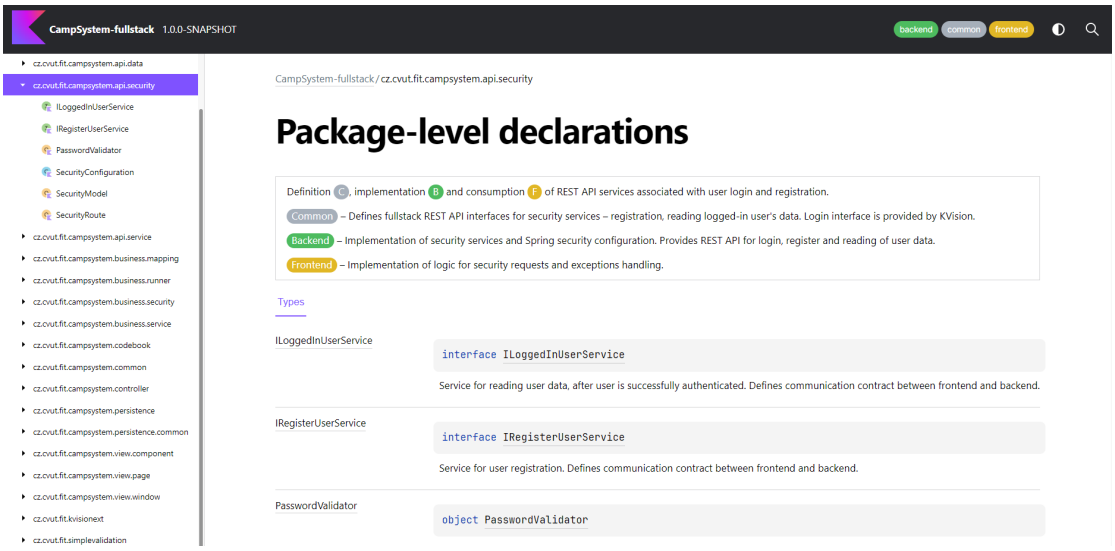
Obrázek 6.8 Ukázka validace



Obrázek 6.9 Aplikace – stránka přehledu



Obrázek 6.10 Aplikace – formulář pro vytvoření organizace



Obrázek 6.11 Ukázka vygenerované dokumentace

6.8 Dokumentace a testování

Dokumentace a testování jsou dva klíčové prvky vývojového procesu softwaru. Dobrá dokumentace může ušetřit spoustu času a úsilí při vývoji, údržbě a rozšíření softwaru. Testování zase pomáhá zajistit, že software funguje správně a splňuje požadavky uživatelů.

V obrázku 6.11 je ukázka vygenerované dokumentace. Jedná se o multiplatformní dokumentaci, která umožňuje filtrovat balíčky pomocí tlačítek v horní liště. Dále byly pro potřebu této dokumentace přidány do popisu jednotlivých modulů barevná zvýraznění, která označují, jaké platformy se popis týká.

Testování bylo provedeno formou uživatelských testů. Následuje souhrn zpětné vazby pro jednotlivé funkcionality:

- Úvodní stránka webu – Sekce působily na uživatele přehledně, jen by očekávali jejich rozšíření o další, které budou lépe ukazovat možnosti služby.
- Registrace a přihlášení – Uživatel by mohl být po registraci rovnou přihlášen. To však nebylo implementováno z toho důvodu, aby mohla být aplikace jednoduše rozšiřitelná o možnost ověření emailu.
- Tvorba organizace – Formulář by pro stolní verzi bylo vhodnější roztáhnout na celou obrazovku.

6.9 Nasazení

Po nasazení aplikaci aplikace poskytuje aplikační rámec KVision gradle task JAR. Ten zkompiluje server do bytekódu a klienta do JavaScriptu a zabalí celou aplikaci do jednoho JAR archivu. Tento archiv lze následně spustit na serveru, na kterém je nainstalováno JRE 17 a vyšší.

Závěr

Cílem bakalářské práce bylo analyzovat, navrhnout a vytvořit webovou aplikaci v podobě informačního systému, která umožní propojení táborů, rodičů i organizátorů na jednom místě a zároveň podpoří vybrané procesy pořádání táborů.

V první řadě byla provedena rešerše konkurence v oblasti táborových katalogů a informačních systémů, která poukázala na chybějící nedostatky na trhu. Na základě průzkumu trhu byla poté provedena detailní analýza specifikující funkční a nefunkční požadavky pro požadovanou aplikaci. Souvisejícím krokem analýzy bylo též vypracování rozsáhlého doménového modelu pro realizaci těchto požadavků.

Dalším neméně důležitým krokem bylo vypracování přehledu mapujícího existující programovací jazyky pro platformu JVM. V první řadě byly přiblíženy výhody této platformy a důvody její aktuálnosti. Následně byla vytvořena rešerše, která mapovala nejpoužívanější jazyky pro technologii JVM, která zahrnovala jazyky Java, Groovy, Scala a Kotlin. Díky vypracované rešerši bylo nakonec zjištěno, že pro vývoj webové aplikace bude nejvhodnější zvolit jazyk Kotlin a s tím související technologie.

Poté následovalo zhotovení samotného návrhu architektury. Daná kapitola se zabývala nejen důležitostí návrhu při vývoji software, ale také zahrnovala navrhnutí architektury klientské a serverové části webové aplikace.

Vzhledem k rozsáhlosti provedené analýzy a cílů práce, bylo po dohodě s vedoucím rozhodnuto, že se implementace bude věnovat pouze realizaci požadavků pro vytvoření katalogu. Implementace se nejprve věnovala ověření, zda je se zvolenými technologiemi možné aplikaci realizovat. Z důvodu čelení několika problémům, jejichž řešení zabralo mnoho času, se nepodařilo úspěšně implementovat všechny zamýšlené funkcionality. Příčinou dlouhého řešení problémů byla volba málo používaných technologií, což prodloužilo čas strávený jejich řešením.

Praktická část aplikace byla detailně zdokumentována pomocí komentářů v kódu, ze kterých byla vygenerovaná programátorská příručka. Aplikace byla též základně otestována uživateli a jejich zpětná vazba shrnuta v této práci.

Přestože se nepovedlo splnit rozsáhlé cíle této práce, obsahuje tato práce všechny aspekty softwarového inženýrství od rešerše technologií a trhu, přes návrh a analýzu až po samotnou implementaci, dokumentaci a testování. Analýza, návrh a implementace byly též podpořeny UML diagramy, usnadňující vizualizaci a pochopení problematiky. Všechny požadované kroky byly provedeny a na vývoj aplikace lze navázat.

Přínos této práce lze vidět především v rozsáhle vypracované analýze. Dále může být přínosná pro vývojáře hledající alternativy k jazyku Java, jelikož provedená rešerše programovacích jazyků usiluje o nezávislý pohled na problematiku. V neposlední řadě může být praktická část využita jako základ pro jiný multiplatformní projekt.

Bibliografie

1. *Táborová sezóna 2022* [online]. Fajn Táboření o. s., 2022 [cit. 2023-03-08]. Dostupné z: https://www.taboreni.cz/clanek/5250/Taborova%20sezona%202022.html.
2. *Pojem katalog* [online]. ABZ knihy a.s., ©2005–2023 [cit. 2023-02-16]. Dostupné z: https://slovnik-cizich-slov.abz.cz/web.php/slovo/katalog.
3. *Táboření* [online]. Internet Archive, 2012 [cit. 2023-03-08]. Dostupné z: https://web.archive.org/web/20120414170715/https://www.taboreni.cz/tabor/.
4. *Táboření* [online]. Fajn Táboření o. s., ©2011–2023 [cit. 2023-02-16]. Dostupné z: https://www.taboreni.cz/tabor/.
5. *České tábory* [online]. Internet Archive, 2022 [cit. 2023-03-08]. Dostupné z: https://web.archive.org/web/20221006030216/http://www.cesketabory.cz/tabory/.
6. *České tábory* [online]. COODY OUTDOOR s.r.o., ©2008–2023 [cit. 2023-02-16]. Dostupné z: https://www.cesketabory.cz/tabory/.
7. *Dětské tábory* [online]. Internet Archive, 2017 [cit. 2023-03-08]. Dostupné z: https://web.archive.org/web/20171119081326/https://detske-tabory.info/.
8. *Dětské tábory* [online]. Internet Archive, 2021 [cit. 2023-03-08]. Dostupné z: https://web.archive.org/web/20211209095310/https://detske-tabory.info/.
9. *Dětské tábory* [online]. VRK plus s.r.o., ©2007–2023 [cit. 2023-02-16]. Dostupné z: https://detske-tabory.info/katalog-taboru.
10. *Data* [online]. Wilmington (DE): ManagementMania.com, 2018 [cit. 2023-02-16]. Dostupné z: https://managementmania.com/cs/data.
11. *Informace* [online]. Wilmington (DE): ManagementMania.com, 2017 [cit. 2023-02-16]. Dostupné z: https://managementmania.com/cs/informace.
12. NÁPLAVA, Pavel. *BI-TIS – Úvod do problematiky IS* [online]. Praha: ČVUT FIT, 2022 [cit. 2023-04-11]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/531177/mod_page/content/18/Prednaska01.pdf.
13. *Evidence účastníků* [online]. VRK plus s.r.o., ©2007–2023 [cit. 2023-02-16]. Dostupné z: https://detske-tabory.info/sluzby/evidence-ucastniku.
14. *Informační systém pro správu spolku* [online]. VRK plus s.r.o., ©2016–2023 [cit. 2023-02-16]. Dostupné z: https://pohodlne.info/.
15. MLEJNEK, Jiří. *BI-SI1 – Analýza a sběr požadavků* [online]. Praha: ČVUT FIT, 2022 [cit. 2023-04-11]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/506241/mod_resource/content/7/03.prednaska.pdf.

16. ČÁPKA, David. *UML – Online kurz* [online]. Praha: ITnetwork.cz, [b.r.] [cit. 2023-03-21]. Dostupné z: https://www.itnetwork.cz/navrh/uml.
17. BAUMANN, H.; GRÄSSLE, P.; BAUMANN, P. *UML 2.0 in Action: A Project-based Tutorial*. Birmingham: Packt Publishing, 2005. ISBN 9781904811558.
18. LEUN, Vincent van der. *Introduction to JVM Languages*. Birmingham: Packt Publishing, 2017. ISBN 9781787126589.
19. HENDRICK, Josh. *How to Choose Your JVM-Based Language* [online]. Palo Alto: Rookout, 2022 [cit. 2023-05-06]. Dostupné z: https://www.rookout.com/blog/how-to-choose-your-jvm-based-language/.
20. *The Scala Programming Language* [online]. Lausanne: École Polytechnique Fédérale, ©2002–2023 [cit. 2023-05-06]. Dostupné z: https://www.scala-lang.org/.
21. *Concise. Cross-platform. Fun.* [Online]. Praha: JetBrains s.r.o., ©2000-2023 [cit. 2023-05-06]. Dostupné z: https://kotlinlang.org/.
22. ROY, Sandip. *The difference between a framework and a library* [online]. Bucharest: Tarnum Java SRL, 2022 [cit. 2023-05-06]. Dostupné z: https://www.baeldung.com/cs/framework-vs-library.
23. KUDRYASHOV, Roman. *Not only Spring boot: A review of alternatives* [online]. 2020. [cit. 2023-05-06]. Dostupné z: https://romankudryashov.com/blog/2020/01/heterogeneous-microservices/#_micronaut_service.
24. *Why Spring?* [Online]. Palo Alto: VMware, Inc., ©2000-2023 [cit. 2023-05-06]. Dostupné z: https://spring.io/why-spring.
25. SYDORKINA, Anastasiia. *Key Differences Between Client-Side, Server-Side and Pre-rendering* [online]. Dněpr: Clockwise Software, 2023 [cit. 2023-05-06]. Dostupné z: https://clockwise.software/blog/client-side-vs-server-side-vs-pre-rendering/.
26. *Introduction to client-side frameworks* [online]. MDN Web Docs, 2023 [cit. 2023-05-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Introduction#what_frameworks_are_out_there.
27. *Kotlin for JavaScript* [online]. Praha: JetBrains s.r.o., 2023 [cit. 2023-05-06]. Dostupné z: https://kotlinlang.org/docs/js-overview.html#kotlin-js-frameworks.
28. JAROS, Robert. *KVision - object oriented web framework for Kotlin/JS* [online]. 2023. [cit. 2023-05-06]. Dostupné z: https://kvision.io/.
29. MLEJNEK, Jiří. *BI-SI1 – Návrh softwarových systémů* [online]. Praha: ČVUT FIT, 2022 [cit. 2023-04-11]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/506252/mod_resource/content/6/05.prednaska.pdf.
30. MLEJNEK, Jiří. *BI-SI1 – Architektonické vzory* [online]. Praha: ČVUT FIT, 2022 [cit. 2023-04-11]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/506256/mod_resource/content/3/06.prednaska.pdf.
31. *MVC* [online]. MDN Web Docs, 2023 [cit. 2023-05-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/MVC#model_view_controller_example.
32. ČÁPKA, David. *MVC architektura* [online]. Praha: ITnetwork.cz, [b.r.] [cit. 2023-05-06]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.
33. BELKOV, Alexey. *Fullstack Kotlin with Maven?* [Online]. Praha: JetBrains s.r.o., 2020 [cit. 2023-05-06]. Dostupné z: https://discuss.kotlinlang.org/t/fullstack-kotlin-with-maven/16008/2.
34. *Co je to Proof of Concept?* [Online]. IT-Slovnik.cz, ©2008–2022 [cit. 2023-05-06]. Dostupné z: https://it-slovnik.cz/pojem/proof-of-concept.

35. DINESHCHANDGR. *Hibernate vs JPA VS Spring Data JPA* [online]. medium.com, 2022 [cit. 2023-05-06]. Dostupné z: https://medium.com/javarevisited/hibernate-vs-jpa-vs-spring-data-jpa-ff4485aaa780.

36. DRAHOMÍR, Hanák. *Stopařův průvodce REST API* [online]. Praha: ITnetwork.cz, [b.r.] [cit. 2023-05-06]. Dostupné z: https://www.itnetwork.cz/programovani/nezarazene/stoparuv-pruvodce-rest-api/.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `diagrams` ..... zdrojové kódy diagramů ve formátu puml
  - `doc` ..... programátorská dokumentace ve formátu html
- `text` ..... text práce
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `thesis.pdf` ..... text práce ve formátu PDF