UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2025 Dmytrii Vezerian

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Mobilní aplikace kulturní akcí

Bakalářská práce

2025 Dmytrii Vezerian

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Dmytrii Vezerian**
Osobní číslo:	**I20291**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Mobilní aplikace kulturní akcí**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvořit mobilní aplikaci pro operační systém Android, která bude uživatelům nabízet informace o kulturních akcích v různých městech.
Uživatel bude moci akce filtrovat a řadit podle různých kritérií. Zadávání kulturní akcí bude řešeno buď manuálním vkládáním nebo importem z veřejně dostupných zdrojů.
Nedílnou součástí práce mimo implementace bude rešerše podobných služeb, analýza, výběr technologií, navržení architektury aplikace a realizace testů.

Rozsah pracovní zprávy: **30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

WARGO John M. Cordova API Cookbook (Mobile Programming): Addison-Wesley Professional; 1st edition (July 9, 2014), ISBN 978-0321994806
SIMS, Gary. Android authority [online]. Dostupné z: https://www.androidauthority.com/develop-android-apps-languages-learn-391008/
HARKIRAN, Kaur. Top Programming Languages for Android App Development Dostupné z: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/

Vedoucí bakalářské práce: **Ing. Martin Pozdílek, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2025

Prohlašuji:

Práci s názvem Mobilní aplikace kulturní akcí jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

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

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

V Pardubicích dne 22. 08. 2025

Dmytrii Vezerian

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Martinu Pozdílkovi, Ph.D. za odborné vedení, cenné rady a trpělivost během celého procesu tvorby této práce. Jeho podpora a konstruktivní zpětná vazba mi výrazně pomohly při zpracování tématu i při praktické části vývoje aplikace.

Poděkování patří také mé rodině a přátelům za jejich podporu, motivaci a trpělivost během celého studia.

ANOTACE

Tato bakalářská práce se zaměřuje na návrh a vývoj mobilní aplikace pro vyhledávání a správu kulturních a společenských událostí. Aplikace je postavena na multiplatformním frameworku Apache Cordova, který umožňuje vývoj pomocí webových technologií jako HTML, CSS a JavaScript. Backend je řešen pomocí PHP a databáze MySQL.

KLÍČOVÁ SLOVA

Mobilní aplikace, Apache Cordova, kulturní události, multiplatformní vývoj, JavaScript, HTML, CSS, PHP, MySQL

TITLE

Mobile Application for Cultural Events

ANNOTATION

This bachelor thesis focuses on the design and development of a mobile application for searching and managing cultural and social events. The application is built on the multiplatform framework Apache Cordova, which enables development using web technologies such as HTML, CSS, and JavaScript. The backend is implemented using PHP and a MySQL database.

KEYWORDS

Mobile application, Apache Cordova, cultural events, multiplatform development, JavaScript, HTML, CSS, PHP, MySQL

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK A ZNAČEK

AJAX – Asynchronous JavaScript and XML

API – Application Programming Interface

Cordova – Apache Cordova (multiplatformní framework)

CSS – Cascading Style Sheets

DOM – Document Object Model

ERD – Entity-Relationship Diagram

FTP – File Transfer Protocol

HTML – HyperText Markup Language

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

ID – Identifikátor

JS – JavaScript

MVC – Model–View–Controller

MySQL – Open source relační databázový systém

PDO – PHP Data Objects

PHP – Hypertext Preprocessor

REST – Representational State Transfer

SQL – Structured Query Language

UI/UX – User Interface / User Experience

XSS – Cross-Site Scripting

ÚVOD

V současné době hrají mobilní aplikace klíčovou roli v každodenním životě. Tato práce se zaměřuje na návrh a vývoj mobilní aplikace, která uživatelům umožní snadné vyhledávání veřejných událostí. Aplikace je určena široké veřejnosti, která se zajímá o kulturní, sportovní či společenské akce. Výběr tohoto tématu byl motivován snahou vytvořit nástroj, jenž zlepší dostupnost informací o kulturních událostech, sportovních akcích i dalších společenských aktivitách.

Vývoj nativních aplikací pro jednotlivé platformy, jako je Android, může být časově i finančně náročný. Alternativním řešením je využití multiplatformních nástrojů, jako je Apache Cordova, které umožňují vývoj aplikací pomocí běžných webových technologií, jako jsou HTML, CSS a JavaScript.

Cílem této práce je návrh a vývoj mobilní aplikace zaměřené na správu osobních údajů uživatelů a zobrazení přehledu dostupných událostí. Aplikace je navržena tak, aby byla uživatelsky přívětivá a snadno rozšiřitelná o další funkce. Pro zajištění ukládání a správu dat byla použita technologie PHP a relační databáze MySQL, což zajišťuje stabilitu a flexibilitu celého systému.

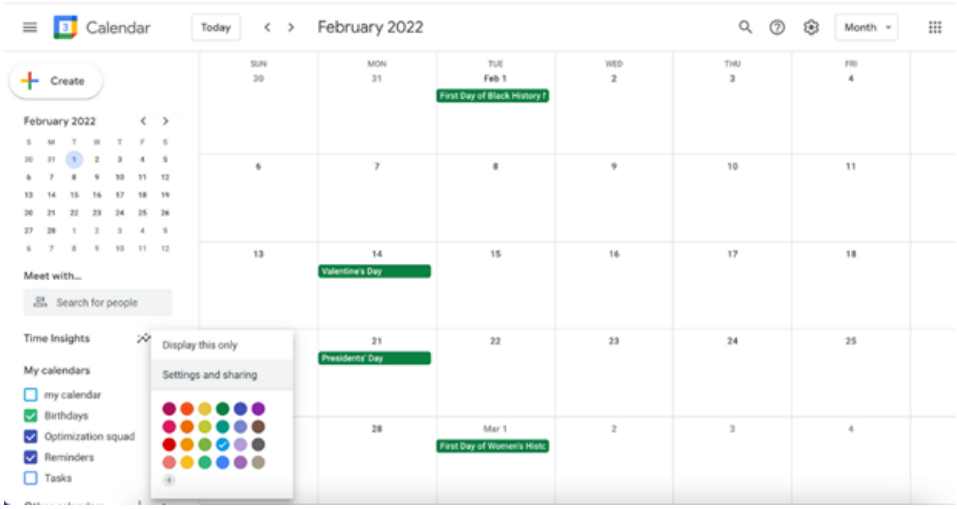
Práce popisuje použité technologie a jejich výhody při vývoji mobilních aplikací. Dále zahrnuje návrh, implementaci a testování aplikace, přičemž jsou podrobně popsány jednotlivé fáze vývoje. V závěru je zhodnoceno výsledné řešení a jeho přínos.

1 Rešerše

V současnosti existuje řada mobilních aplikací zaměřených na správu uživatelských údajů a organizaci událostí. Některé aplikace jsou obecné a slouží pro správu kalendářů, zatímco jiné jsou specializované a umožňují registraci a správu uživatelů pro specifické akce. Mezi nejznámější aplikace patří Google Calendar, Eventbrite, Facebook Events a Simple Calendar.

1.1 Google Calendar

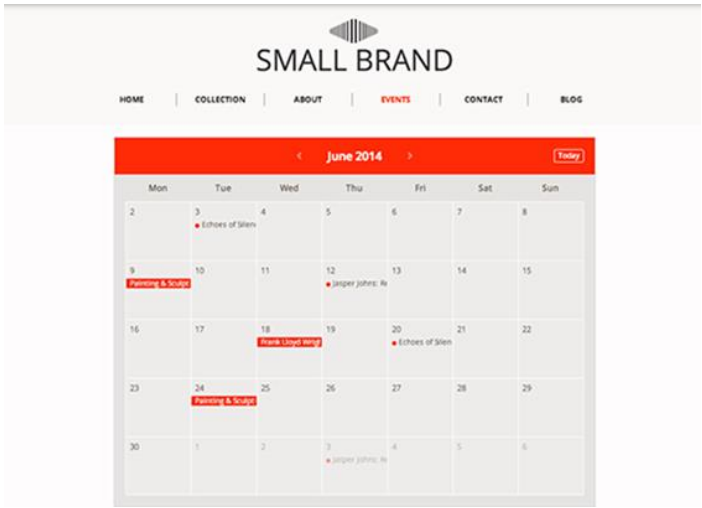
Google Calendar patří mezi nejpoužívanější digitální kalendáře současnosti. Uživatelům umožňuje snadno vytvářet a spravovat události s možností nastavení připomínek, opakování a sdílení kalendáře s ostatními. Díky integraci s dalšími službami Google a možností synchronizace napříč zařízeními se jedná o velmi praktický nástroj jak pro osobní, tak pracovní využití. Přestože nabízí široké spektrum funkcí, chybí mu pokročilá správa uživatelských účtů přímo v rámci aplikace, což může být omezením zejména při týmovém nasazení [1].



Obrázek 1: Google Calendar, zdroj: [1]

1.2 Eventbrite

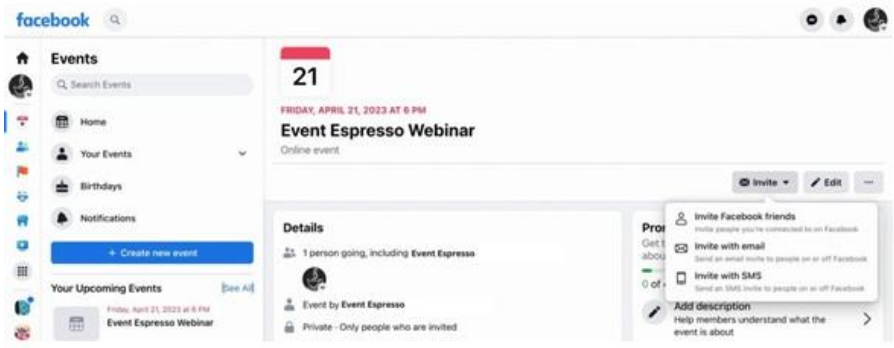
Eventbrite je populární platforma určená k organizaci jak veřejných, tak soukromých událostí, přičemž klade důraz zejména na správu účastníků a prodej vstupenek. Organizátorům poskytuje nástroje pro tvorbu detailních popisů akcí, jednoduchou správu registrací i přehledné rozhraní pro komunikaci s přihlášenými. Systém rovněž nabízí analytické funkce, které pomáhají sledovat návštěvnost, úspěšnost marketingových kampaní a celkový zájem o akci v reálném čase. Omezením může být skutečnost, že úprava osobních údajů uživatelů je možná pouze v rámci konkrétní události a není řešena na úrovni celého uživatelského profilu [2].



Obrázek 2: Eventbrite, zdroj: [2]

1.3 Facebook Events

Facebook Events představují integrovanou součást této sociální sítě, která umožňuje uživatelům snadno vytvářet a spravovat události s možností pozvání přátel a sdílení informací v rámci vlastní komunity. Hlavní důraz je kladen na sociální interakci – účastníci mohou události komentovat, sdílet je dále a dostávat automatická upozornění na blížící se termíny. Výhodou je rychlá distribuce informací v rámci existující sítě kontaktů, nevýhodou pak určitá omezenost pro organizátory, kteří

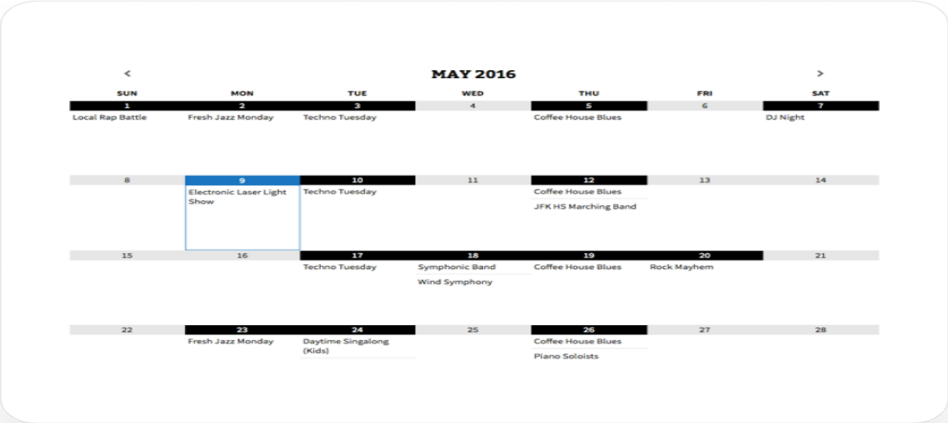


Obrázek 3: Facebook Events, zdroj: [3]

jsou plně závislí na funkcionalitě a pravidlech samotné platformy Facebooku, bez možnosti větší vlastní přizpůsobitelnosti [3].

1.4 Simple Calendar

Tato minimalistická kalendářová aplikace je navržena pro uživatele, kteří preferují jednoduchost a funkčnost bez zbytečných komplikací. Nabízí přehledné a intuitivní rozhraní, které umožňuje vytvářet a spravovat události i bez nutnosti připojení k internetu nebo propojení s účtem Google. Právě absence závislosti na externích službách může být pro některé uživatele klíčovou výhodou, zejména v případě požadavku na větší míru soukromí. Na druhou stranu aplikace neobsahuje pokročilé funkce jako je synchronizace mezi zařízeními či možnost sdílení událostí s ostatními, což ji činí vhodnější spíše pro individuální použití [4].



Obrázek 4: Simple Calendar, zdroj: [4]

1.5 Odlišnosti navrhované aplikace

Navržená aplikace se od výše zmíněných řešení odlišuje především svou zaměřeností na jednoduchou správu osobních údajů a přehledné zobrazení událostí. Zatímco existující aplikace často integrují široké množství funkcí, tato aplikace se soustředí na specifické potřeby uživatelů s důrazem na administrativní kontrolu a uživatelsky přívětivé prostředí.

Hlavní rozdíly oproti existujícím aplikacím:

- Správa uživatelských účtů – uživatelé mohou registrovat, upravovat a spravovat své osobní údaje přímo v aplikaci.
- Administrativní přístup – správci mohou přímo upravovat obsah aplikace, kontrolovat uživatelské účty a spravovat události.
- Přehledné zobrazení událostí – jednoduchý systém pro strukturované zobrazování a vyhledávání informací o událostech.

- Nezávislost na externích službách – aplikace funguje jako samostatné řešení, nevyžaduje integraci se sociálními sítěmi nebo jinými platformami.

Díky těmto vlastnostem aplikace cílí na různé typy uživatelů, zejména na účastníky akcí, kteří hledají jednoduché a efektivní řešení pro správu osobních údajů a informací o událostech, bez nutnosti využívat rozsáhlé a komplexní platformy, které mohou být pro některé účely nadbytečné .

1.6 Funkční a nefunkční požadavky

1.6.1 Funkční požadavky

Mobilní aplikace poskytuje uživatelům možnost prohlížet a vyhledávat události, spravovat svůj uživatelský profil a interagovat se systémem v bezpečném a uživatelsky přívětivém prostředí.

Registrace a správa uživatelského profilu

- V rámci aplikace je uživatelům umožněno vytvořit vlastní účet prostřednictvím jednoduchého registračního formuláře, kde se zadávají základní osobní údaje, jako je jméno, telefonní číslo a datum narození.
- Během registrace systém automaticky vygeneruje unikátní identifikační číslo (promo). Jedná se o náhodné unikátní číslo, které může při registraci použít jiný nový uživatel.
- Při registraci je možné vložit promo číslo jiného uživatele. Tomu se automaticky připočítají hvězdy. Informace o provázání uživatelů je uložena v databázi.
- Při registraci uživatel zároveň získává jeden event kód.
- Uživatel má možnost kdykoliv upravovat a aktualizovat své osobní údaje v rámci uživatelského profilu.

Správa událostí

- Přihlášený uživatel se může prohlížeč různé události v systému a filtrovat a třídit je podle různých kritérií jako jsou kategorie, místo a datum konání.
- U každé události si uživatel může zobrazit detailní informace včetně popisu, odkazu na zakoupení vstupenek, mapy místa konání a kontaktních údajů.
- Uživatel se může na akci registrovat. V jeden okamžik může být registrovaný na více akcích najednou.

Event kód

- Během registrace na akci uživatel může využít event kód.
- Po použití je event kód zneplatněn.
- Další event kód je možné získat jako bonus za účast na událostech, za pozvání nových uživatelů prostřednictvím referenčního systému nebo ručně – dle uvážení administrátora.

Bodový systém (stars)

- Součástí profilu je sledování aktuálního stavu bodového systému („stars“), kde uživatel sbírá body například za účast na akcích
- Body se připisují automaticky po skončení události, pokud byl uživatel na událost registrován, nebo pokud se na ni zaregistroval někdo prostřednictvím jeho referenčního systému.

Administrativní funkce

- Administrátor má v systému možnost vytvářet nové události, editovat jejich obsah jako například změnu data, místa konání či popisu, případně je mazat.
- Administrátor disponuje možnostmi vyhledávání uživatelů, správy a aktualizace jejich údajů.
- Administrátor může ručně uživatelům přidávat hvězdy do bodového systému.
- Administrátor může ručně uživatelům přidělovat další event kódy.
- Administrátor spravuje seznam kategorií akce.
- Dále systém umožňuje zobrazování statistik týkajících se registrací a aktivity uživatelů za účelem efektivnější správy aplikace (například pohledy na seznam registrovaných uživatelů na akci, seznam uživatelů, kteří použili na akci event kód atd.)
- Součástí funkcionality je i možnost automatického importu událostí z externích zdrojů pomocí technik parsování webových stránek.
- Systém podporuje integraci externích API podle specifických požadavků.

1.6.2 Nefunkční požadavky

Výkonnost

- Aplikace je optimalizována, aby doba odezvy jednotlivých akcí byla do 1 sekundy.

Uživatelská přívětivost (UI/UX)

- Uživatelské rozhraní je navrženo s důrazem na intuitivní ovládání a přehlednost.
- Design aplikace je minimalistický a je přizpůsoben různým velikostem obrazovek včetně zařízení s nižším výkonem. Design aplikace je minimalizovaný a přizpůsoben různým velikostem obrazovek, včetně mobilních telefonů, tabletů a běžných počítačů. Optimalizace je zohledněna i pro zařízení s nižším výpočetním výkonem.

Bezpečnost a komunikace

- Pro výměnu dat mezi klientskou a serverovou částí systému je využíváno rozhraní REST API s datovým formátem JSON.
- Bezpečnost přenosu dat je zajištěna prostřednictvím šifrovaného protokolu HTTPS.
- Načítání dat probíhá asynchronně (využitím technologií AJAX/Fetch API), což přispívá k plynulému uživatelskému zážitku.
- Veškerá vstupní data podléhají validaci za účelem prevence bezpečnostních hrozeb, jako jsou útoky typu SQL injection či XSS.
- Osobní údaje uživatelů jsou v systému ukládány v zašifrované podobě.
- Přístup k API je omezen na důvěryhodné zdroje pomocí CORS politiky. CORS (Cross-Origin Resource Sharing) je bezpečnostní mechanismus, který definuje, jaké domény mají povolený přístup k API. Server na základě hlavičky Origin určuje, zda požadavek z externího původu (např. z jiné domény nebo portu) může být obsloužen. Tím je zabráněno neoprávněnému přístupu k citlivým údajům ze škodlivých aplikací. CORS hlavičky se nastavují na straně serveru, typicky pomocí Access-Control-Allow-Origin.

Podporovaná platforma

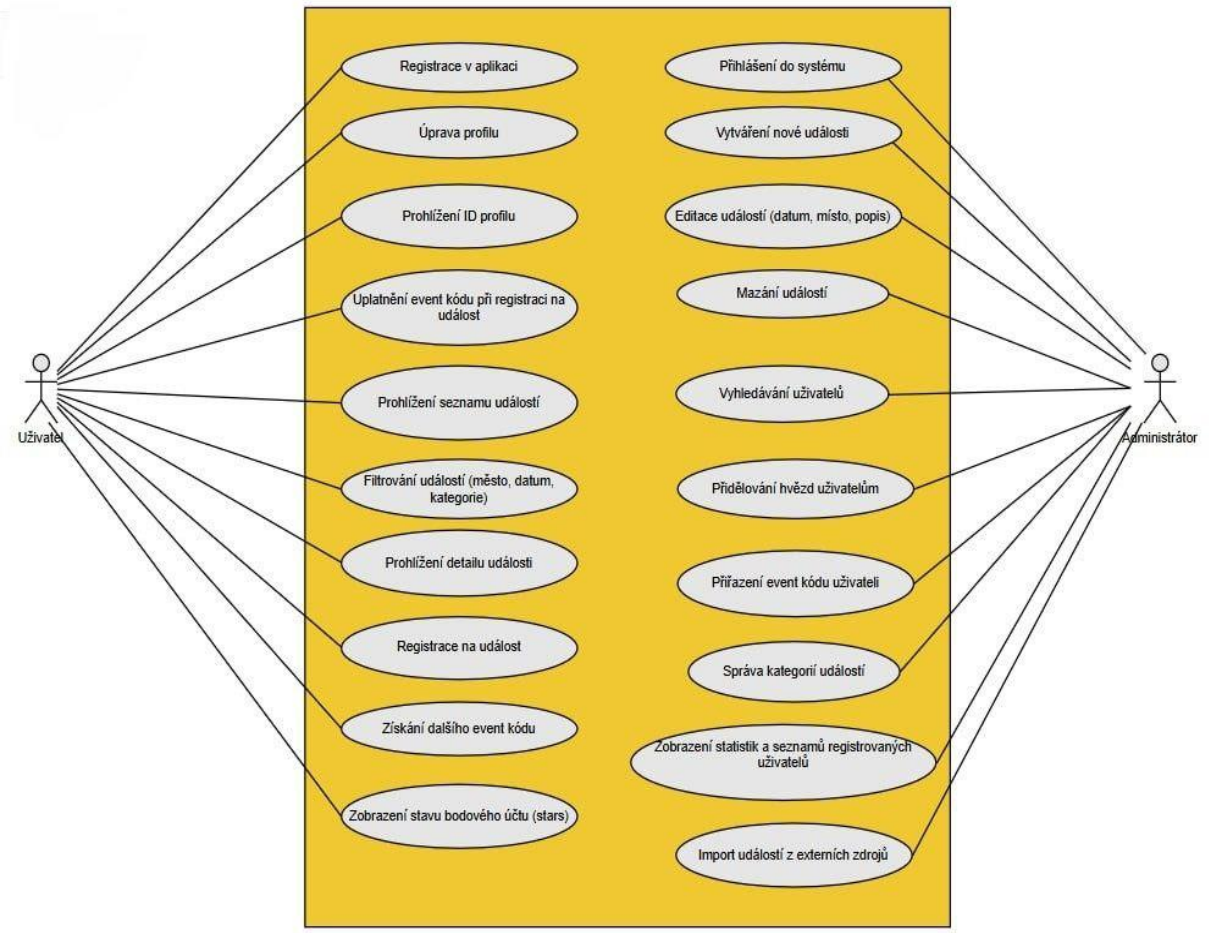
- Aplikace je plně kompatibilní s operačním systémem Android.

Další funkcionality

- Aplikace podporuje offline režim, který umožňuje prohlížení dříve načtených událostí i bez aktivního připojení k internetu. Po obnovení připojení dochází k automatické synchronizaci dat.

2 Analýza

2.1 Use case



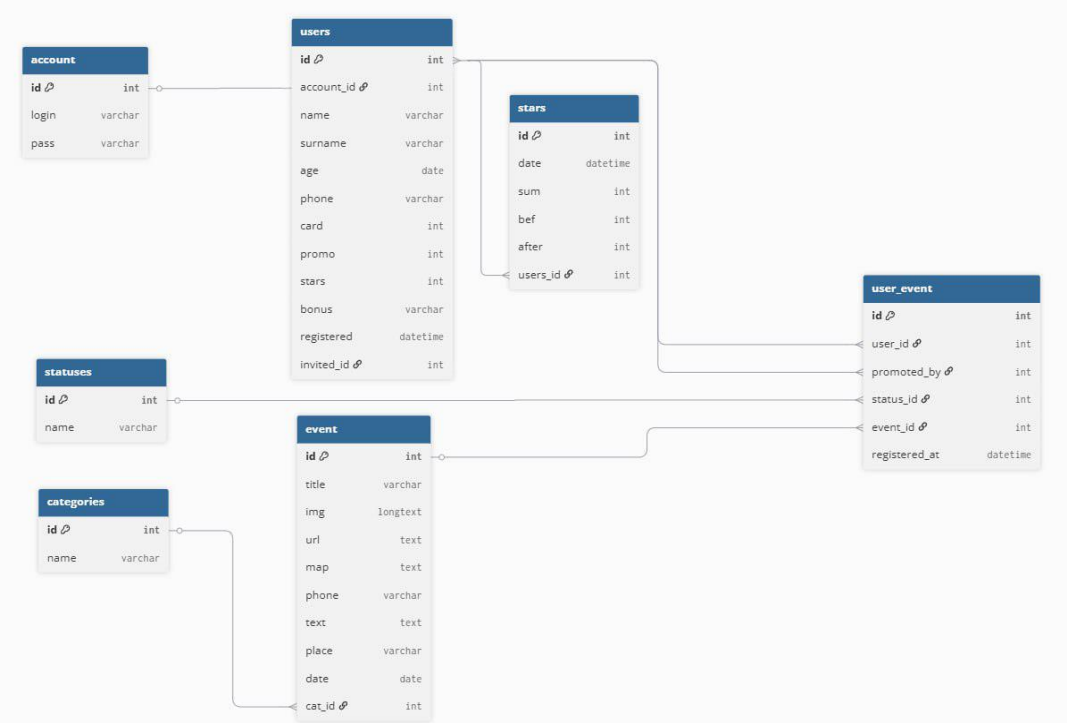
Obrázek 5: Scénáře použití (Use Cases), zdroj: vlastní zpracování

Use Case diagram popisuje interakci uživatelů s mobilní aplikací. Aplikace slouží ke správě uživatelských účtů, prohlížení a filtrování událostí a administrativní správě obsahu. Každý uživatel může provádět různé akce, které odpovídají jeho roli v systému.

V systému jsou definovány dva hlavní typy uživatelů:

- Běžný uživatel (User) – může se registrovat, přihlašovat, prohlížet obsah a spravovat své osobní údaje.
- Administrátor (Admin) – má oprávnění pro správu událostí, kategorií, statistik a základních údajů uživatelských účtů.

2.2 Datový model



Obrázek 6: Databázový model (ERD diagram) , zdroj: vlastní zpracování

Databázový model aplikace je založen na relačním modelu a obsahuje sedm hlavních tabulek, které definují správu uživatelů, příspěvků a kategorií [17, 18].

Tabulka account

Tato tabulka slouží k autentizaci uživatelů při přihlašování do systému:

- login – přihlašovací jméno
- pass – heslo uložené pomocí algoritmus bcrypt, který je speciálně navržen pro hashování hesel. Tento algoritmus:
    - automaticky přidává kryptografickou sůl ke každému hashi
    - umožňuje nastavení výpočetní náročnosti (cost)
    - generuje hashe pevné délky bez ohledu na délku hesla

Tabulka categories

Slouží ke kategorizaci událostí (akce) v systému:

- name – název kategorie, např. "Music Events", "Educational Events", "Outdoor and Adventure", apod.

Každá akce patří do jedné kategorie.

Tabulka statuses

Tabulka slouží k evidenci základního stavu uživatele na události. Například může označovat, zda je uživatel registrován, zda se ukončené akce zúčastnil apod.

- name – název stavu (např. registred, completed, …)

Tabulka users

Hlavní tabulka uchovávající podrobné informace o uživatelích systému:

- name, surname, age, phone – základní osobní údaje o uživateli
- stars – aktuální počet bodů, které uživatel vlastní (součást bodového systému)
- promo – referenční kód jiného uživatele, který uživatel zadá při registraci a tím mu přidělí bonusové body (*referral systém*)
- invited_id – id uživatele v tabulce users na uživatele, který tohoto uživatele pozval
- bonus – speciální „event“ kód pro přihlášení na akci, díky kterému uživatel může získat výhody nebo přístup
- registered – datum a čas registrace uživatele do systému

Tabulka event

Tato tabulka uchovává informace o jednotlivých akcích (událostech), které jsou zveřejněny v systému:

- title – název události
- img – obrázek události (uložený jako base64 string)
- url – odkaz na stránku s více informacemi
- map – souřadnice nebo mapa místa konání (např. Google Maps)
- phone – kontaktní telefon na pořadatele
- text – podrobný popis události (může obsahovat HTML)
- place – místo konání události
- date – datum konání
- cat_id – cizí klíč na tabulku categories, určující do jaké kategorie událost patří

Tabulka je jádrem systému pro správu a zobrazování událostí. Uživatelé se na tyto události registrují prostřednictvím tabulky user_event.

Tabulka stars

Tato tabulka slouží k sledování historie změn bodového konta jednotlivých uživatelů:

- sum – hodnota změny (kladná nebo záporná)
- bef – počet bodů před změnou
- after – počet bodů po změně
- users_id – cizí klíč odkazující na tabulku users
- date – datum a čas provedené změny

Každý záznam představuje jednu změnu ve stavu bodů daného uživatele.

Tabulka user_akce

Tato tabulka slouží k uchování informací o registraci uživatelů na jednotlivé události:

- user_id – cizí klíč na tabulku users, označující, kdo se registroval
- akce_id – cizí klíč na tabulku akce, označující, na jakou akci
- promoted_by – identifikátor uživatele, který jiného uživatele zaregistroval nebo pozval na událost při vlastní registraci zůstává NULL (odkazuje na tabulku users)
- status_id – cizí klíč na tabulku statuses, označující stav registrace
- registered_at – datum a čas registrace na akci

Každý záznam reprezentuje vztah mezi uživatelem a událostí, na kterou se registroval.

3 Implementace

3.1 Architektura

3.1.1 Uživatelská a administrativní část

Aplikace byla navržena jako modulární systém, který je rozdělen do dvou hlavních částí – uživatelské a administrativní. Toto členění umožňuje oddělit běžné činnosti koncových uživatelů od správy obsahu a systémových funkcí.

Uživatelská část

Uživatelská část slouží k interakci návštěvníků s aplikací. Jejím cílem je umožnit snadný přístup k informacím o veřejných kulturních, sportovních a společenských událostech. Uživatelům je umožněno registrovat se prostřednictvím jednoduchého vícekrokového formuláře, kde jsou zadávány základní osobní údaje. Po registraci je vygenerován unikátní identifikátor, který může být dále využit v referenčním systému.

Po úspěšném přihlášení má uživatel přístup ke svému profilu, ve kterém může upravovat své údaje, sledovat stav bodového systému („stars“) a získávat přehled o vlastní aktivitě v rámci aplikace. Součástí profilu je také možnost využít event kódy k registraci na konkrétní akce. Uživateli je rovněž nabídnuto filtrování a vyhledávání událostí podle kategorií, lokality nebo data konání. Z každé události je možné zobrazit detailní informace a provést registraci. Systém umožňuje registraci na více akcí současně a následně sleduje účast na základě zadaného statusu.

Administrativní část

Administrativní rozhraní je určeno pro správce systému, kterému poskytuje nástroje pro řízení obsahu a dohled nad uživatelskou aktivitou. Správce má možnost vytvářet a upravovat události, měnit jejich vlastnosti, jako je název, datum, místo konání nebo popis, a v případě potřeby je zcela odstranit.

V rámci administrace může správce spravovat jednotlivé uživatelské účty, včetně vyhledávání uživatelů, aktualizace jejich údajů a ručního přidělování bonusových bodů či event kódů. Rovněž má k dispozici správu seznamu kategorií událostí a monitoring registrací. Aplikace poskytuje přehledné statistiky o aktivitě uživatelů, včetně počtu registrací na konkrétní akce nebo využití referenčního systému.

Součástí administrativní části je také mechanismus pro importování událostí z externích zdrojů. Tento proces probíhá automaticky pomocí interního parseru, který extrahuje potřebná data, provádí

jejich transformaci do požadovaného formátu a vkládá je do databáze. Systém zajišťuje, že duplicitní události nejsou zobrazeny, a rovněž provádí čištění a validaci importovaného obsahu [15].

Administrativní prostředí je navrženo s důrazem na přehlednost, snadné ovládání a efektivní správu systému, čímž je zajištěna vysoká úroveň kontroly nad celým chodem aplikace.

3.1.2 Backend, frontend

Struktura aplikace byla navržena jako oddělený systém skládající se z klientské (frontendové) a serverové (backendové) části.

Frontendová část

Frontendová část představuje vrstvu, se kterou přímo interaguje uživatel. Zajišťuje vizuální rozhraní aplikace, umožňuje zadávání vstupních údajů a zprostředkovává přístup ke všem dostupným funkcím systému. Rozhraní bylo navrženo s důrazem na přehlednost, intuitivní navigaci a kompatibilitu s různými typy zařízení. Důležitým prvkem návrhu bylo přizpůsobení aplikace pro použití na mobilních telefonech, tabletech i zařízeních s omezeným výkonem.

Uživatelské rozhraní je rozděleno do více sekcí, které zahrnují registraci a přihlášení, hlavní stránku s přehledem událostí, profil uživatele, správu osobních údajů a zobrazování event kódů. Zvláštní pozornost byla věnována možnosti filtrování událostí podle kategorií, místa konání a časového období.

Backendová část

Backendová část zajišťuje zpracování veškerých požadavků, správu dat, řízení aplikační logiky a komunikaci s databázovým systémem. Systém byl navržen podle architektonického vzoru, který odděluje datovou vrstvu od prezentační a řídicí logiky, což výrazně zjednodušuje správu kódu a budoucí rozšiřování funkcionality.

Databázová vrstva zahrnuje tabulky pro správu uživatelů, událostí, kategorií, registrací, stavů a bodového systému. Tyto tabulky jsou propojeny přes vazby a referenční klíče, které umožňují sledování historie, propojení mezi uživateli a jejich akcemi, evidenci získaných bodů a uplatnění referenčních nebo bonusových kódů.

Na straně serveru je implementována kompletní logika pro autentizaci uživatelů, zpracování formulářů, správu profilů, filtrování událostí, záznam registrací, kontrolu stavu účasti a přidělování bodových odměn. Zajištěna je rovněž správa obsahu ze strany administrátora, včetně importu událostí z externích zdrojů. Tento import probíhá automatizovaně, přičemž systém analyzuje externí obsah, identifikuje relevantní údaje, filtruje duplicity a aktualizuje databázi.

Backend je navržen jako rozšiřitelný systém s možností přidání nových funkcí, jako jsou další filtry, export dat, rozšířená analýza aktivity uživatelů nebo podpora více jazykových mutací.

3.2 Volba technologií

Při vývoji mobilní aplikace byl kladen důraz na volbu technologií, které zajistí multiplatformní kompatibilitu, snadnou údržbu a efektivní komunikaci mezi klientskou a serverovou částí. Systém byl záměrně rozdělen na dvě hlavní části: uživatelskou (klientská mobilní aplikace) a administrátorskou (serverová aplikace s databází), pro které byly zvoleny specifické technologie.

3.2.1 Uživatelská část

Pro vývoj klientské, uživatelsky orientované části aplikace byla zvolena technologie Apache Cordova [5]. Jedná se o open-source framework, jehož primárním cílem je umožnit vývojářům vytvářet multiplatformní mobilní aplikace s využitím jejich stávajících znalostí standardních webových technologií. Konkrétně se pro definování základní struktury a sémantiky prvků uživatelského rozhraní používá jazyk HTML, pro vizuální stylizaci, rozložení a zajištění responzivního designu CSS [11] a veškerá aplikační logika, interaktivita a komunikace se serverem je implementována v jazyce JavaScript [9]. Cordova funguje jako most (bridge), který propojuje webovou část aplikace s nativními funkcemi mobilního zařízení prostřednictvím systému pluginů.

Architektura klientské části je tedy plně postavena na tomto frameworku. Ten umožňuje provozovat takto vytvořenou webovou aplikaci uvnitř nativního kontejneru, který je specifický pro cílový operační systém, v tomto případě Android. Jádrem tohoto přístupu je systémová komponenta WebView, což je v podstatě vestavěný prohlížeč bez viditelných ovládacích prvků, který může aplikace využít k vykreslování webového obsahu. Pro vytvoření samotné stránky, která je v komponentě WebView zobrazena, a pro implementaci její logiky jsou použity právě zmíněné technologie HTML, CSS a JavaScript [11, 12]. Tato komponenta je tak plně zodpovědná za vykreslení veškerého uživatelského rozhraní a spuštění aplikační logiky přímo na zařízení. Celá aplikace je tak ve své podstatě webová stránka „zabalená“ do nativního rámce, což bylo motivováno především snahou o zefektivnění a zrychlení vývojového procesu.

Tento hybridní přístup přináší řadu významných výhod, ale také určitá omezení, která byla při volbě technologie zvážena.

- Výhody: Klíčovou předností je multiplatformní kompatibilita, která umožňuje spravovat jednotnou kódovou základnu pro více operačních systémů (Android, iOS). To vede k výrazně rychlejšímu vývoji a nižším nákladům, jelikož není nutné vyvíjet a udržovat oddělené verze pro každou platformu ani se učit specifické programovací jazyky jako Swift

či Kotlin. Dalším pozitivem je snadná rozšiřitelnost pomocí širokého ekosystému pluginů, které poskytují JavaScriptové API pro přístup k nativním funkcím zařízení, jako je souborový systém, síťové informace nebo systémová schránka, což bylo v rámci tohoto projektu využito.

- Omezení: Hlavním kompromisem je potenciálně nižší výkon ve srovnání s plně nativními aplikacemi. Komunikace mezi JavaScriptem a nativní vrstvou přes zmíněný most představuje určitou režii a vykreslování v WebView nemusí být tak plynulé jako u nativních UI komponent. Dalším omezením je závislost na verzi WebView, která je součástí operačního systému. Na starších zařízeních s neaktualizovanou komponentou WebView se může aplikace chovat odlišně nebo nemusí podporovat moderní webové standardy, což může vést k nekonzistentnímu zobrazení a funkčnosti.

3.2.2 Administrátorská část

Administrativní rozhraní bylo realizováno jako rozšířená webová část systému, přístupná oprávněným uživatelům (administrátorům) prostřednictvím standardního webového prohlížeče. Stejně jako uživatelské rozhraní byla i tato část vytvořena pomocí HTML, CSS a JavaScriptu [11, 12], avšak s důrazem na správu dat a přístup ke komplexnějším funkcím. Pro administrativní část jsem framework Apache Cordova nepoužil.

V rámci administračního panelu je umožněna:

- správa uživatelských účtů (editace údajů, přidělování bodů a kódů),
- vytváření a úprava událostí,
- správa kategorií,
- import událostí z externích webových zdrojů pomocí vlastního parseru.

Administrativní rozhraní komunikuje s backendem prostřednictvím REST API [19], přičemž přenášená data jsou ve formátu JSON. Díky tomuto rozhraní lze data získávat i aktualizovat bez nutnosti opakovaného načítání celé stránky, což zajišťuje rychlejší odezvu systému a pohodlnější uživatelskou zkušenost.

3.2.3 Databázové řešení

Databázová vrstva byla postavena na relačním databázovém systému MySQL, který zajišťuje spolehlivé uložení, integritu a rychlý přístup k datům. Pro komunikaci s databází na serverové straně

byl použit jazyk PHP, přičemž datová vrstva byla zprostředkována pomocí ORM knihovny RedBeanPHP [14].

Použití RedBeanPHP umožnilo automatické generování tabulek, snadnou manipulaci s daty bez nutnosti přímého psaní SQL dotazů a podporu pokročilých databázových operací.

3.2.4 Zabezpečení a komunikace API

Klíčovou součástí systému je spolehlivá a bezpečná komunikace mezi klientskými aplikacemi a serverovým backendem. Veškerá komunikace probíhá přes rozhraní API, přičemž byl kladen důraz na zavedení standardních bezpečnostních opatření:

- Šifrovaný přenos: Pro ochranu přenášených dat před odposlechem je veškerá komunikace vedena výhradně přes šifrovaný protokol HTTPS [20].
- Validace vstupních dat: Všechna data přicházející od uživatele jsou na straně serveru důsledně validována a ošetřena (sanitizována). Tento krok je zásadní pro ochranu proti běžným zranitelnostem, jako jsou útoky typu SQL Injection (vkládání škodlivého SQL kódu) a Cross-Site Scripting (XSS) (vkládání škodlivých skriptů do webové stránky) [20].
- Řízení přístupu (CORS): Přístup k API je řízen pomocí politiky CORS (Cross-Origin Resource Sharing). Tento mechanismus zajišťuje, že k serverovému rozhraní mohou přistupovat pouze důvěryhodné klientské aplikace z povolených domén, čímž se brání neoprávněnému využití API [8].

Díky tomuto kombinovanému přístupu bylo dosaženo vysoké úrovně bezpečnosti, tak i flexibility a škálovatelnosti celého systému.

3.3 Modulární architektura aplikační logiky

Pro zajištění přehlednosti, snadné údržby a budoucího rozšíření byla aplikační logika v JavaScriptu navržena jako modulární. Každý modul má jasně definovanou zodpovědnost a specializuje se na konkrétní oblast funkcionality, což odděluje jednotlivé části systému a přispívá k čistotě kódu.

3.3.1 Mechanismus importu událostí z externích zdrojů

Jednou z klíčových funkcí administrativního rozhraní je možnost automatizovaného importu událostí z externích webových stránek. Cílem této funkcionality je minimalizovat manuální práci administrátora, zrychlit proces přidávání nového obsahu a zajistit aktuálnost dat v aplikaci. Tento proces je realizován pomocí vlastního parseru implementovaného na straně serveru v jazyce PHP [16].

Celý proces importu probíhá v několika navazujících krocích:

Získání HTML obsahu

Proces je iniciován administrátorem, který v administrativním rozhraní zadá URL adresu externí webové stránky obsahující seznam nebo detail události. Serverová aplikace následně pomocí PHP funkce (např. file_get_contents nebo knihovny cURL) stáhne kompletní HTML obsah dané stránky.

Parsování a extrakce dat (Web Scraping)

Po získání HTML kódu nastupuje samotný parser. Ten využívá PHP knihovny pro práci s DOM (Document Object Model), jako jsou DOMDocument a DOMXPath. Parser je naprogramován tak, aby procházel HTML strukturu a na základě předem definovaných selektorů (např. názvů HTML značek, tříd nebo ID) vyhledával a extrahoval konkrétní data [11]:

- Název události: Typicky z nadpisů <h1> nebo <h2>.
- Popis události: Z odstavců <p> nebo specifických kontejnerů.
- Datum a místo konání: Z tabulek <table> nebo ze strukturovaných seznamů <ul>.
- Obrázek: Značka <img> s relevantním atributem src.
- Odkaz na web: Značka <a> s atributem href.

Transformace a čištění dat

Data získaná v surové podobě jsou následně zpracována a převedena do požadovaného formátu. Tento krok zahrnuje například:

- Převod textového data (např. "9. srpen 2025") na standardní databázový formát (YYYY-MM-DD).
- Odstranění nežádoucích HTML značek, stylů a skriptů z textu popisu, aby se zabránilo narušení vzhledu aplikace.
- Převod relativních URL adres obrázků nebo odkazů na absolutní.

Kontrola duplicit

Před uložením nové události do databáze systém provede kontrolu, aby se zabránilo duplicitním záznamům. Porovná extrahovaný název a datum události s již existujícími záznamy v tabulce event [19]. Pokud je nalezena shoda, proces importu pro danou položku se zastaví.

Validace a uložení do databáze

Očištěná a zkontrolovaná data jsou nakonec validována (např. kontrola, zda není název prázdný). Pokud jsou všechna data v pořádku, jsou prostřednictvím ORM knihovny RedBeanPHP uložena jako nový záznam do tabulky event v databázi MySQL [15, 13].

Tento automatizovaný proces výrazně zefektivňuje správu obsahu a umožňuje snadno rozšiřovat databázi událostí bez nutnosti zdlouhavého manuálního zadávání.

3.3.2 Přehled modulů a jejich odpovědnosti

Struktura JavaScriptového kódu je rozdělena do několika klíčových souborů, které vzájemně spolupracují na zajištění plynulého chodu aplikace [12]:

- app.js: Tento soubor představuje hlavní vstupní bod aplikace. Jeho primární rolí je inicializace frameworku Cordova a správa životního cyklu aplikace. Naslouchá klíčové události deviceready, která signalizuje, že všechny nativní komponenty a pluginy jsou připraveny k použití. Dále zpracovává systémové události jako pause (minimalizace aplikace) a resume (návrat do popředí), což umožňuje aplikaci správně reagovat na změny stavu.
- main.js: Funguje jako centrální koordinátor aplikační logiky. Po úspěšné inicializaci v app.js přebírá řízení a volá ostatní moduly podle aktuální potřeby. Zajišťuje orchestraci procesů, jako je zobrazení správné obrazovky nebo spuštění akcí na základě interakce uživatele.
- dom.js: Tento modul je plně zodpovědný za veškerou manipulaci s prvky DOM (Document Object Model). Jeho úkolem je dynamicky aktualizovat uživatelské rozhraní, vykreslovat data získaná ze serveru a registrovat posluchače událostí (např. kliknutí na tlačítka, odeslání formulářů). Tím odděluje prezentační vrstvu od aplikační logiky [13].
- data.js: Tento modul má na starosti správu dat. Zajišťuje komunikaci se serverovým REST API prostřednictvím asynchronních HTTP požadavků pro načítání a ukládání dat. Dále spravuje lokální úložiště (localStorage) pro podporu offline režimu, kdy jsou dříve načtená data dostupná i bez připojení k internetu.
- reg.js: Specializovaný modul, který obsahuje veškerou funkcionalitu spojenou s registrací a autorizací uživatelů. Zpracovává a validuje data z registračních formulářů a komunikuje s modulem data.js za účelem jejich bezpečného uložení na server.

3.3.3 Schéma interakce mezi moduly

Spolupráce mezi jednotlivými moduly probíhá podle přesně definovaného schématu, které zajišťuje efektivní zpracování uživatelských požadavků:

- Spuštění aplikace: Po startu aplikace soubor app.js inicializuje prostředí Cordova. Jakmile je zařízení připraveno, je spuštěna událost deviceready, na kterou app.js zareaguje a předá řízení hlavnímu modulu main.js.
- Zobrazení a interakce: Modul main.js následně instruuje dom.js, aby vykreslil výchozí uživatelské rozhraní. Modul dom.js zároveň nastaví posluchače událostí na interaktivní prvky. Když uživatel provede akci (např. kliknutí na tlačítko), dom.js tuto událost zachytí a předá ji main.js ke zpracování.
- Zpracování dat: Na základě provedené akce main.js rozhodne, jaký další krok je potřeba. Pokud je nutné pracovat s daty (např. načíst seznam událostí), zavolá příslušnou funkci v modulu data.js.
- Komunikace se serverem a offline režim: Modul data.js nejprve zkontroluje stav síťového připojení. Je-li zařízení online, odešle HTTP požadavek na server. V případě offline režimu se pokusí načíst data z lokálního úložiště localStorage.

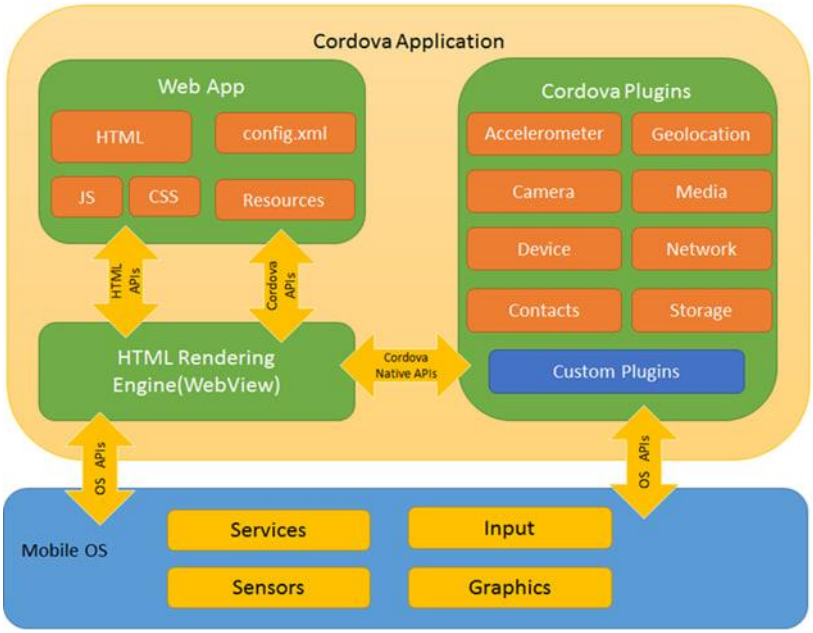
Aktualizace rozhraní probíhá následovně. Po obdržení dat (ať už ze serveru nebo z lokálního úložiště) data.js vrátí výsledek modulu main.js. Ten následně znovu zavolá dom.js, aby aktualizoval uživatelské rozhraní a zobrazil nová data uživateli.

3.4 Architektura a klíčové nativní komponenty frameworku Cordova

JavaScriptový kód aplikace, popsaný v předchozí kapitole, je spouštěn v rámci specifického prostředí, které zajišťuje framework Apache Cordova. Jeho architektura umožňuje, aby byla aplikace, vyvinutá pomocí standardních webových technologií, spuštěna uvnitř nativního aplikačního rámce a mohla přistupovat k funkcím a prostředkům hostitelského operačního systému [5].

3.4.1 Vrstvený model architektury

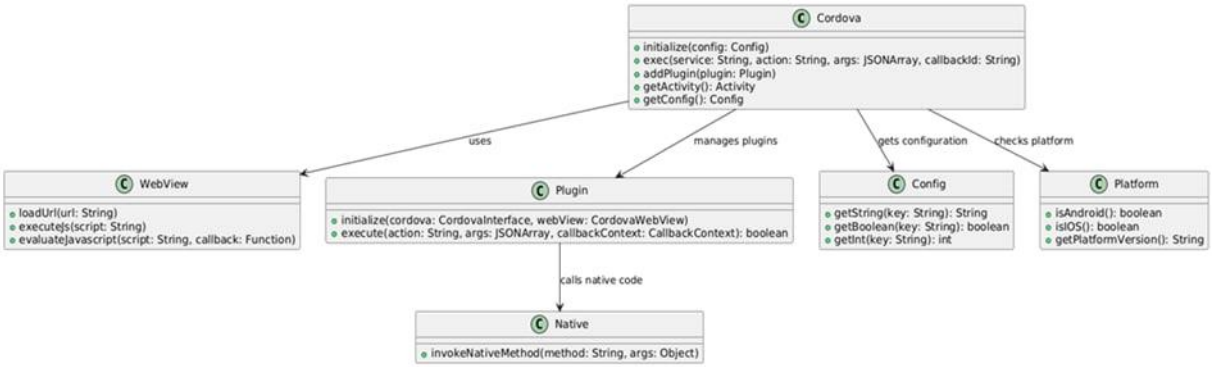
Architektura frameworku je tvořena čtyřmi hlavními, vzájemně provázanými komponentami, které společně tvoří funkční celek [1].



Obrázek 7: Popis diagramu, zdroj: [5]

- Web App: Představuje jádro aplikace z pohledu vývojáře. Obsahuje veškeré soubory, které definují uživatelské rozhraní a logiku (HTML, CSS, JavaScript)[11, 12] a konfigurační soubor config.xml.
- WebView: Klíčová systémová komponenta, která je zodpovědná za vykreslení obsahu z vrstvy Web App a spouštění JavaScriptových skriptů [7, 12].
- Cordova Plugins: Moduly nativního kódu, které fungují jako most mezi webovou a nativní vrstvou [6].
- Mobile OS: Nejnižší vrstva představující samotný operační systém (v tomto případě Android).

3.4.2 Hlavní nativní třídy a správa životního cyklu



Obrázek 8: Class diargam aplikace, zdroj: [6]

Celý tento balíček je hostován v minimální nativní aplikaci, jejímž jádrem je na platformě Android třída CordovaActivity. Ta slouží jako hlavní vstupní bod a orchestrátor celé aplikace. Její vztah k ostatním klíčovým třídám nativní vrstvy je znázorněn na diagramu níže.

Jak diagram ukazuje, CordovaActivity spravuje instanci WebViewEngine, která je zodpovědná za načtení webového obsahu do komponenty WebView. Klíčovou odpovědností CordovaActivity je správa životního cyklu aplikace prostřednictvím metod, které reagují na systémové události [7]:

- onCreate(): Volána při prvním vytvoření aplikace, inicializuje klíčové komponenty a načítá WebView.
- onResume(): Spuštěna, když se aplikace vrací do popředí. Aplikace v JavaScriptu obdrží událost resume.
- onPause(): Reaguje na minimalizaci aplikace. Do WebView je vyslána událost pause.
- onDestroy(): Volána při úplném ukončení aplikace, uvolňuje systémové prostředky.

Třída CordovaActivity rovněž hraje roli dispečera – když JavaScriptový kód zavolá funkci pluginu, metoda exec() v CordovaActivity tento požadavek přijme a předá ho ke zpracování správné instanci třídy Plugin. Tím je překlenuta technologická mezera mezi webovým a nativním světem.

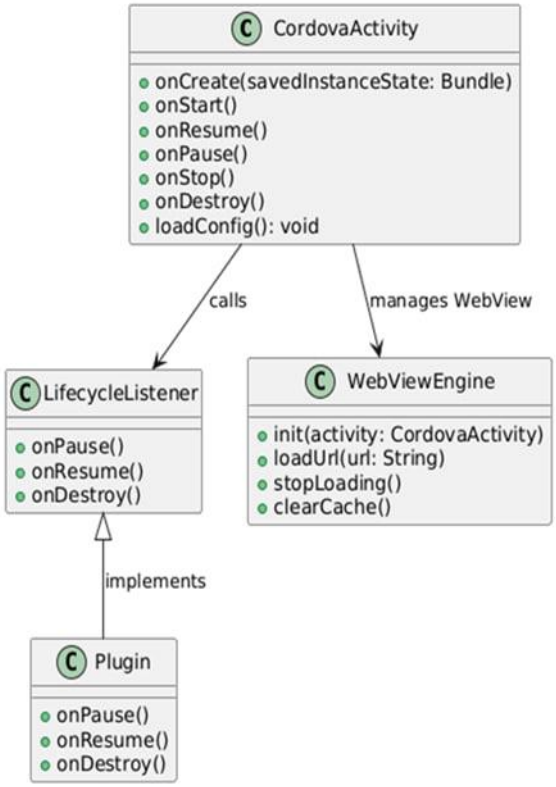
3.5 Správa a využití pluginů v aplikaci

Zásadní výhodou frameworku Cordova [6] je jeho schopnost propojit webovou část aplikace s nativními funkcemi zařízení. Toto propojení je realizováno prostřednictvím systému rozšiřujících pluginů. Pluginy jsou samostatné moduly nativního kódu, které fungují jako most mezi JavaScriptovým kódem běžícím ve WebView a nativními API operačního systému. Tím umožňují aplikaci využívat funkce, které jsou pro standardní webové stránky nedostupné.

3.5.1 Role PluginManageru a životní cyklus pluginu

Správu celého ekosystému pluginů zajišťuje centrální třída PluginManager. Při spuštění aplikace je jednou z jejích úloh načíst všechny pluginy definované v konfiguračním souboru config.xml. Když následně JavaScriptový kód požaduje provedení nativní akce, PluginManager je zodpovědný za nalezení správné instance pluginu a spuštění jeho hlavní metody execute() [6].

Pluginy v ekosystému Cordova navíc nemusí fungovat pouze jako pasivní nástroje. Mohou být aktivními komponentami, které jsou přímo napojeny na nativní životní cyklus aplikace. To je realizováno tak, že základní třída Plugin implementuje rozhraní LifecycleListener.



Obrázek 9: Životní cyklus aplikace, zdroj: [6]

Díky tomuto návrhu jsou pluginy informovány o změnách stavu aplikace. Mohou tak reagovat na její minimalizaci (v metodě onPause()), například pozastavením probíhajících operací, nebo na její návrat do popředí (v metodě onResume()), a efektivně tak hospodařit se systémovými prostředky.

3.5.2 Přehled implementovaných pluginů a jejich funkce

Pro zajištění požadované funkcionality byly v aplikaci implementovány následující čtyři klíčové pluginy:

- cordova-plugin-network-information: Tento plugin je nezbytný pro robustní implementaci offline režimu. Aktivně monitoruje stav síťového připojení a umožňuje aplikaci detekovat,

zda je zařízení online. Na základě této informace se modul data.js dynamicky rozhoduje, zda má data načítat ze serveru, nebo z lokální cache [9].

- cordova-plugin-advanced-http: Veškerá komunikace se serverovým API je realizována prostřednictvím tohoto pluginu. Zajišťuje bezpečný přenos dat díky nativnímu zpracování HTTPS a pokročilé správě SSL certifikátů, čímž zároveň obchází problémy spojené s politikou CORS [21].
- cordova-plugin-file: Pro podporu offline režimu a optimalizaci výkonu byl využit plugin pro práci se souborovým systémem. Umožňuje aplikaci ukládat, číst a mazat soubory na zařízení, což je využíváno především pro účely cacheování dat načtených ze serveru [10].
- cordova-clipboard-plugin: Pro zlepšení uživatelského komfortu byl integrován plugin pro práci se systémovou schránkou. Uživatelé tak mohou snadno zkopírovat své unikátní ID (referral kód) a sdílet ho, aniž by ho museli ručně opisovat.

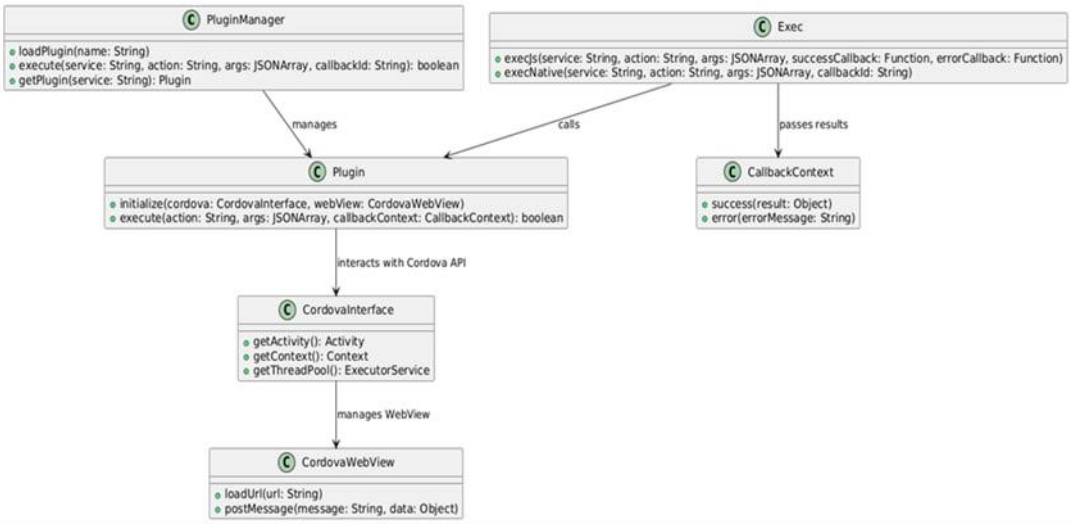
3.6 Komunikace se serverovým rozhraním (API)

Pro zajištění dynamické funkčnosti, jako je registrace uživatelů nebo načítání aktuálních dat, je nezbytná spolehlivá komunikace mezi klientskou aplikací a serverovým backendem. Tato interakce je realizována prostřednictvím rozhraní REST API [19].

3.6.1 Principy a zvolená technologie

Veškerá datová komunikace probíhá přes zabezpečený protokol HTTPS[20] a jako datový formát pro výměnu informací je použit JSON. Pro realizaci samotných HTTP požadavků byl implementován plugin cordova-plugin-advanced-http, který přesouvá zpracování požadavků na nativní úroveň operačního systému. Tento přístup nabízí několik klíčových výhod, jako je zvýšená bezpečnost, obcházení omezení CORS [21] a vyšší spolehlivost, protože operace neblokují uživatelské rozhraní.

3.6.2 Mechanismus volání pluginu



Obrázek 10: Příklad HTTP požadavku pomocí Cordova pluginu, zdroj: [6]

Proces komunikace začíná v JavaScriptové části aplikace. Když je potřeba provést nativní akci, kód zavolá příslušnou funkci pluginu. Tento požadavek je interně předán třídě Exec, která komunikuje s PluginManagerem. PluginManager identifikuje správný plugin a spustí jeho metodu execute(), čímž se řízení dostane na nativní úroveň. Tento obecný mechanismus je znázorněn na obrázku níže [8].

3.6.3 Příklad HTTP požadavku

```
cordova.plugin.http.get(
  "https://server.site/user/data",
  {},   // parameter (GET data)
  {},   // heavily
  function(response) {
    console.log("OK:", response.data);
  },
  function(error) {
    console.error("Error:", error);
  }
);
```

Obrázek 11: Cordova plugin předává požadavek na nativní úroveň, zdroj: vlastní zpracování

Volání pluginu využívá asynchronní model s callback funkcemi. Tento přístup zajišťuje, že aplikace během čekání na odpověď ze serveru nezamrzne. Příklad konkrétního volání GET požadavku pro získání uživatelských dat pomocí pluginu cordova-plugin-advanced-http je vidět na následujícím obrázku [9].

Jak je z ukázky kódu patrné, volání obsahuje cílovou URL adresu a dvě anonymní funkce. První je volána v případě úspěšného přijetí odpovědi (response), druhá v případě chyby (error).

3.6.4 Zpracování odpovědi ze serveru

Obrázek 12: Ukázka JSON odpovědi ze serveru, zdroj: vlastní zpracování

Po úspěšném zpracování požadavku vrací serverový backend odpověď, která obsahuje požadovaná data ve formátu JSON, jak ilustruje následující příklad.

Tento textový řetězec je na straně klienta převeden (parsován) na nativní JavaScriptový objekt. Datový modul data.js následně předá tento objekt řídícímu modulu main.js, který zajistí aktualizaci uživatelského rozhraní prostřednictvím modulu dom.js [12].

3.7 Implementace offline režimu

Klíčovou vlastností moderní mobilní aplikace je schopnost poskytovat alespoň základní funkčnost i bez aktivního připojení k internetu. V rámci tohoto projektu byla implementována strategie pro offline režim, která zajišťuje, že uživatel má přístup k dříve načteným datům a aplikace zůstává použitelná [5].

Tato funkcionalita je postavena na synergii několika komponent popsaných v předchozích kapitolách. Základním principem je dynamická reakce na aktuální stav síťového připojení.

- Detekce stavu sítě: Před každým pokusem o načtení dat z externího zdroje využívá modul data.js plugin cordova-plugin-network-information. Tento plugin poskytuje informaci, zda je zařízení aktuálně připojeno k internetu [9].
- Strategie načítání dat: Na základě zjištěného stavu se aplikace rozhoduje podle následující logiky:
    - Online režim: Pokud je zařízení připojeno, aplikace odešle standardní požadavek na server prostřednictvím pluginu cordova-plugin-advanced-http. Po úspěšném přijetí jsou nová data nejen zobrazena uživateli, ale zároveň jsou uložena do lokální cache pro budoucí použití [11].

    - Offline režim: Pokud zařízení není připojeno k internetu, aplikace se nepokouší kontaktovat server. Místo toho se obrátí na lokální úložiště a načte data, která byla uložena během poslední úspěšné synchronizace.

- Mechanismy pro ukládání dat: Pro dočasné uložení dat na zařízení jsou využívány dvě metody. Pro menší objemy dat, jako jsou uživatelská nastavení nebo textové informace, je použito localStorage. Pro účely robustnějšího kešování, například pro dočasné ukládání souborů nebo větších datových sad, je využíván plugin cordova-plugin-file, který umožňuje přímou práci se souborovým systémem zařízení [10].

Tímto způsobem je zajištěno, že aplikace poskytuje plynulý uživatelský zážitek a minimalizuje dopady dočasné nedostupnosti sítě.

3.8 Architektura serverové části

Serverová část aplikace, implementovaná v jazyce PHP s využitím databáze MySQL [13], nebyla vytvořena jako čistá implementace bez struktury. Pro zajištění přehlednosti kódu, snadné údržby a budoucí škálovatelnosti byla pro ni navržena architektura podle osvědčeného návrhového vzoru MVC (Model-View-Controller). Tento přístup striktně odděluje datovou vrstvu (Model), prezentační vrstvu (View) a aplikační logiku (Controller) a plní dvě klíčové role:

- Poskytuje REST API pro komunikaci s mobilní klientskou aplikací.
- Zobrazuje webové administrační rozhraní pro správu obsahu a uživatelů.

Pro zajištění přehlednosti kódu, snadné údržby a budoucí škálovatelnosti byla její architektura navržena podle osvědčeného návrhového vzoru MVC (Model-View-Controller). Tento přístup striktně odděluje datovou vrstvu (Model), prezentační vrstvu (View) a aplikační logiku (Controller) [13].

Komponenty MVC

Architektura byla implementována prostřednictvím čtyř hlavních, vzájemně spolupracujících komponent:

- Model: Tato komponenta zapouzdřuje veškerou práci s daty a komunikaci s databází. Pro zjednodušení a zrychlení vývoje byla využita knihovna RedBeanPHP, která funguje jako ORM (Object-Relational Mapping). Díky tomu nebylo nutné psát komplexní SQL dotazy ručně; ORM se stará o vytváření, načítání, aktualizaci a mazání datových záznamů prostřednictvím objektově orientovaného rozhraní [15, 14].

- View (Zobrazení): Komponenta View je zodpovědná za generování finálního výstupu pro uživatele, typicky ve formě HTML stránek pro administrační rozhraní. Umožňuje načítání souborů šablon, předávání dynamických dat z kontroléru a využití hlavního layoutu pro zajištění konzistentního vzhledu napříč celou administrací. Klíčovou vlastností je možnost vypnout tento hlavní layout, což se využívá při obsluze AJAXových požadavků z mobilní aplikace, kdy je potřeba vrátit pouze čistá data ve formátu JSON, nikoliv celou HTML stránku [21].
- Controller (Kontrolér): Kontrolér funguje jako prostředník, který zpracovává příchozí HTTP požadavky. Komunikuje s Modelem pro získání nebo uložení dat a následně předává tato data komponentě View k zobrazení. Všechny kontroléry v aplikaci dědí od společné abstraktní třídy Controller, která centralizuje opakující se logiku, jako je inicializace databázového připojení nebo kontrola přihlášení administrátora.
- Router (Směrovač): Jedná se o centrální prvek, který analyzuje URL adresu každého příchozího požadavku a na základě předdefinovaných pravidel rozhoduje, který Controller a která jeho metoda (akce) má být spuštěna. Tím je zajištěno, že například URL adresa .../user/profile je správně nasměrována na metodu profileAction() uvnitř třídy UserController.

Frontový kontroler a inicializace systému

Celý systém využívá návrhový vzor Front Controller, což znamená, že veškeré požadavky na server jsou směrovány na jediný vstupní bod – soubor index.php. Tento soubor je zodpovědný za inicializaci celého prostředí aplikace:

- Spuštění relace (session): Pro správu přihlášení administrátora je inicializována session.
- Definování konstant: Jsou zde definovány globální konstanty (např. ROOT pro kořenový adresář nebo URL pro základní adresu), což usnadňuje konfiguraci a psaní přenositelného kódu.
- Automatické načítání tříd: Pomocí funkce spl_autoload_register() je nastaveno automatické načítání souborů s definicemi tříd. To eliminuje potřebu manuálního vkládání souborů pomocí require nebo include a zpřehledňuje kód.
- Inicializace směrovače: Nakonec je vytvořena instance Routeru a je mu předáno řízení, aby zpracoval aktuální požadavek.

Cesta požadavku: Inicializace a směrování

Každý HTTP požadavek na server je nejprve zpracován podle návrhového vzoru Front Controller. Všechny požadavky bez výjimky směřují na jediný vstupní bod – soubor index.php. Zde proběhne inicializace celého aplikačního prostředí [15]:

- Je spuštěna relace (session) pro správu stavu přihlášeného administrátora.
- Jsou definovány systémové konstanty pro centralizovanou konfiguraci (např. cesty k souborům).
- Je aktivován mechanismus pro automatické načítání tříd (spl_autoload_register), který zjednodušuje správu kódu.

Po dokončení inicializace je řízení předáno komponentě Router (směrovač). Úkolem směrovače je analyzovat URL adresu příchozího požadavku, porovnat ji se sadou předdefinovaných tras pomocí regulárních výrazů a na základě shody určit, který

Controller a která jeho metoda (akce) má být pro zpracování požadavku spuštěna. Například požadavek na example.com/user/profile je nasměrován na metodu profileAction() v rámci UserController.

Provedení logiky: Role Kontroléru a Modelu

Jakmile směrovač určí cíl, je spuštěna příslušná metoda v daném Kontroléru. Kontrolér funguje jako koordinátor – nepřistupuje přímo k databázi ani negeneruje HTML kód, ale deleguje tyto úkoly na specializované komponenty. Jeho úkolem je přijmout požadavek, vyžádat si potřebná data od Modelu a předat je View k zobrazení [16].

Veškeré databázové operace jsou zapouzdřeny v komponentě Model. Pro interakci s databází byla zvolena ORM (Object-Relational Mapping) knihovna RedBeanPHP. Tato volba výrazně zjednodušila práci s daty, jelikož abstrahuje psaní složitých SQL dotazů a umožňuje pracovat s databázovými záznamy jako s objekty [14, 16].

Generování odpovědi: Flexibilita vrstvy View

Po získání dat z Modelu předává Kontrolér řízení komponentě View, která je zodpovědná za finální prezentaci. Vrstva View je navržena flexibilně, aby mohla obsluhovat oba typy klientů:

- Pro webové administrační rozhraní generuje kompletní HTML stránku s využitím systému šablon a hlavního layoutu, který zajišťuje jednotný vzhled.

- Pro REST API (požadavky z mobilní aplikace) je generování hlavního layoutu vypnuto a View vrací pouze čistá data, typicky ve formátu JSON.

Tímto způsobem je zajištěno, že stejná logika v Kontroléru a Modelu může sloužit pro různé typy výstupů, což je klíčový prvek pro efektivní a udržitelný kód.

Zabezpečení aplikace

Bezpečnostní aspekty byly řešeny na několika úrovních:

- Řízení přístupu a autorizace: Přístup do administračního rozhraní je chráněn přihlašovacím systémem, který využívá PHP sessions pro ověření identity a autorizaci administrátora.
- Ochrana proti přímému přístupu: Díky návrhovému vzoru Front Controller, kde všechny požadavky procházejí přes index.php, je zabráněno přímému přístupu k souborům s aplikační logikou z webu.
- Ochrana proti běžným zranitelnostem: Veškerá data přicházející od uživatele podléhají validaci a ošetření (sanitizaci), aby se předešlo útokům typu SQL Injection a Cross-Site Scripting (XSS).
- Zabezpečený přenos dat: Komunikace mezi mobilní aplikací a serverem probíhá výhradně přes šifrovaný protokol HTTPS [20].

Zpracování chyb

```php
public static function dispatch($url){
  $url = self::removeQueryString($url);
  if (self::matchRoute($url)) {
      $controller = 'controllers\\' . self::$route['prefix'] . self::$route['controller'] . 'Controller';
      if (class_exists($controller)) {
          $cObj = new $controller(self::$route);
          $action = self::lowerCamelCase(self::$route['action']) . 'Action';
          if (method_exists($cObj, $action)) {
              $cObj->$action();
              $cObj->getView();
          } else {
              echo "Miss <b>$controller::$action</b>";
          }
      } else {
          echo "Miss <b>$controller</b>";
      }
  } else {
      http_response_code(404);
      include '404.html';
  }
}
```

Obrázek 13: Funkce zpracování adresy žádosti, zdroj: vlastní zpracování

Aplikace obsahuje mechanismus pro elegantní zpracování chyb, zejména těch, které souvisí s nesprávným směrováním požadavků.

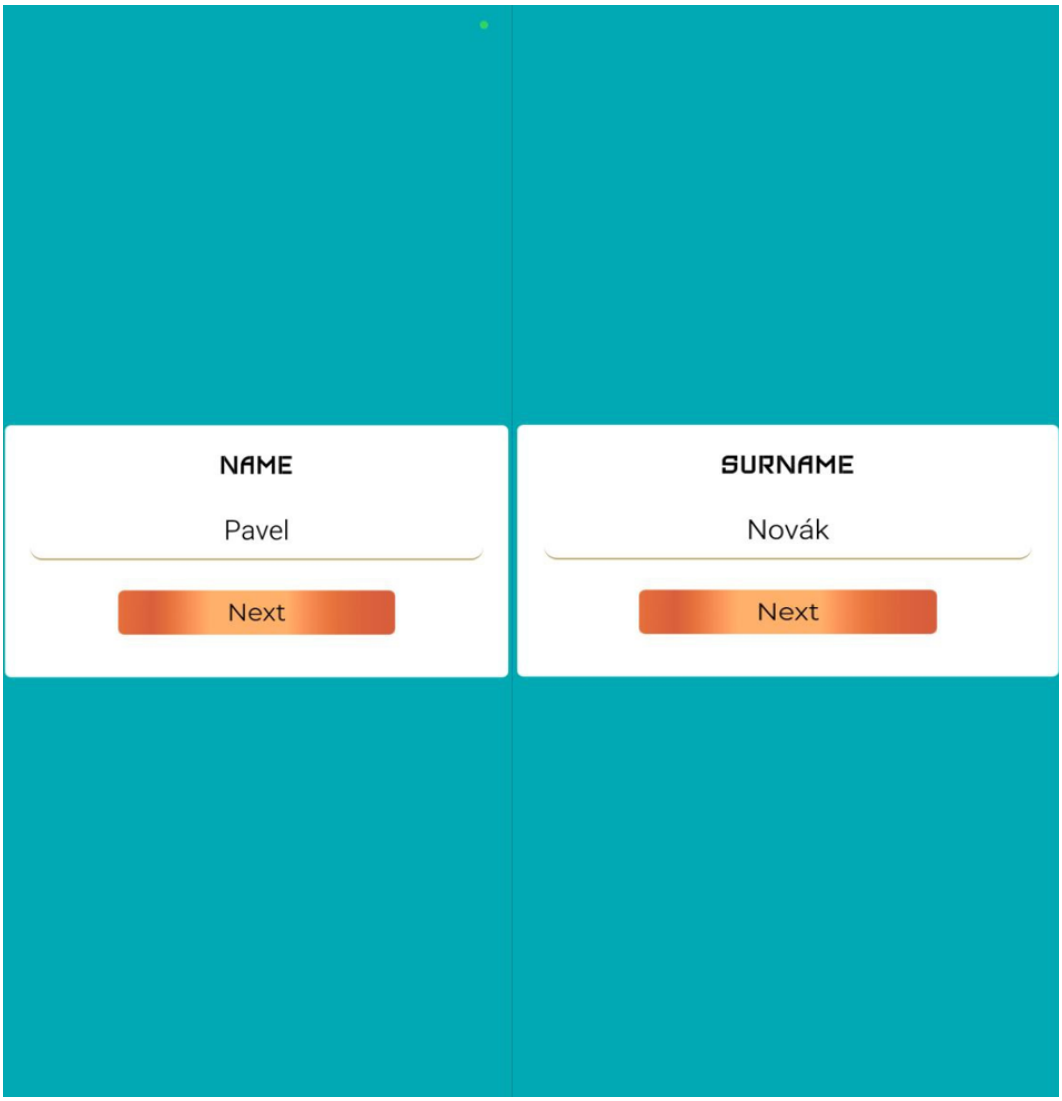
Jak je vidět na ukázce kódu z třídy Router, systém je navržen tak, aby neodpovídal neošetřenou chybou.

- Pokud je nalezena odpovídající trasa, ale neexistuje daný kontrolér nebo jeho metoda, je vypsána pouze informativní hláška pro účely ladění.
- V případě, že pro danou URL adresu neexistuje žádná definovaná trasa, aplikace vrátí standardní chybový stav HTTP 404 Not Found a zobrazí uživatelsky přívětivou chybovou stránku 404.html. Tento přístup zajišťuje robustnost a profesionální chování aplikace i v případě nevalidního požadavku.

4 Popis uživatelského rozhraní

4.1 Uživatelská část

4.1.1 Registrační formulář

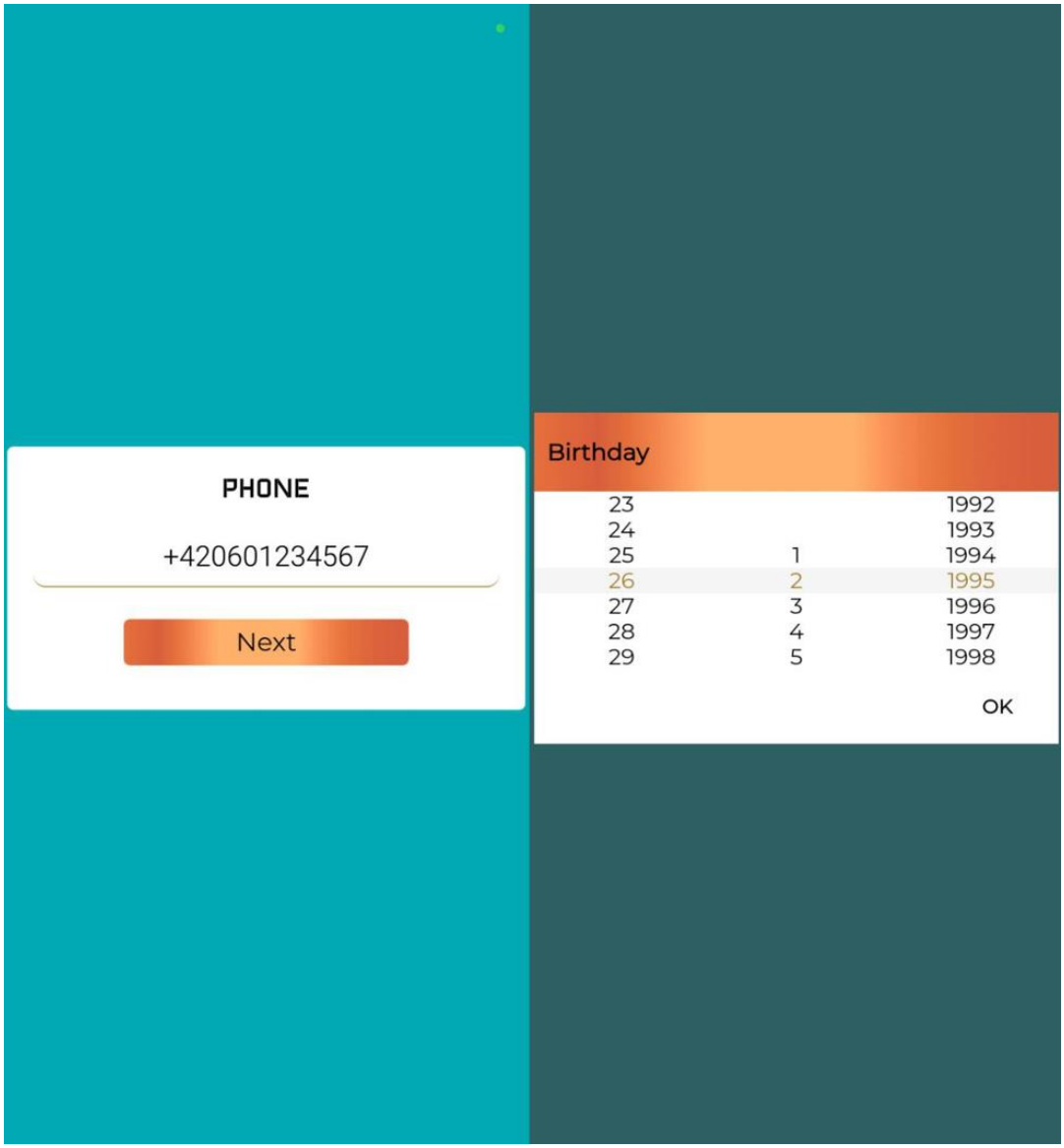


Obrázek 14: Registrační formulář – zadání jména a příjmení, zdroj: vlastní zpracování

Po úvodní obrazovce je uživatel, který dosud nemá účet, přesměrován na registrační proces. V souladu s funkčními požadavky je pro vytvoření účtu nutné zadat základní osobní údaje. Tento proces byl navržen jako vícekrokový formulář, aby bylo jeho vyplnění pro uživatele co nejjednodušší a nejpřehlednější.

Prvními kroky registrace jsou zadání jména (NAME) a příjmení (SURNAME). Každý údaj se zadává samostatně do příslušného pole. Po vyplnění každého údaje uživatel stiskne tlačítko „Next“ pro přechod k dalšímu kroku. Tento postupný přístup minimalizuje množství informací zobrazených najednou a vede uživatele intuitivně celým procesem registrace.

4.1.2 Dokončení registrace



Obrázek 15: Registrační formulář – zadání telefonu a data narození, zdroj: vlastní zpracování

Po zadání jména a příjmení pokračuje vícekrokový registrační proces zadáním telefonního čísla (PHONE) a data narození (Birthday). Uživatel nejprve vyplní své kontaktní telefonní číslo a potvrdí jej stisknutím tlačítka „Next“.

Následně je zobrazeno rozhraní pro výběr data narození. Pro zajištění maximální uživatelské přívětivosti byl implementován přehledný roletkový kalendář, který umožňuje snadný výběr dne, měsíce a roku bez nutnosti ručního vypisování. Po nastavení správného data uživatel potvrdí volbu tlačítkem „OK“, čímž je registrační proces dokončen. Veškeré zadané údaje jsou poté odeslány na server, kde dojde k vytvoření nového uživatelského účtu v databázi.

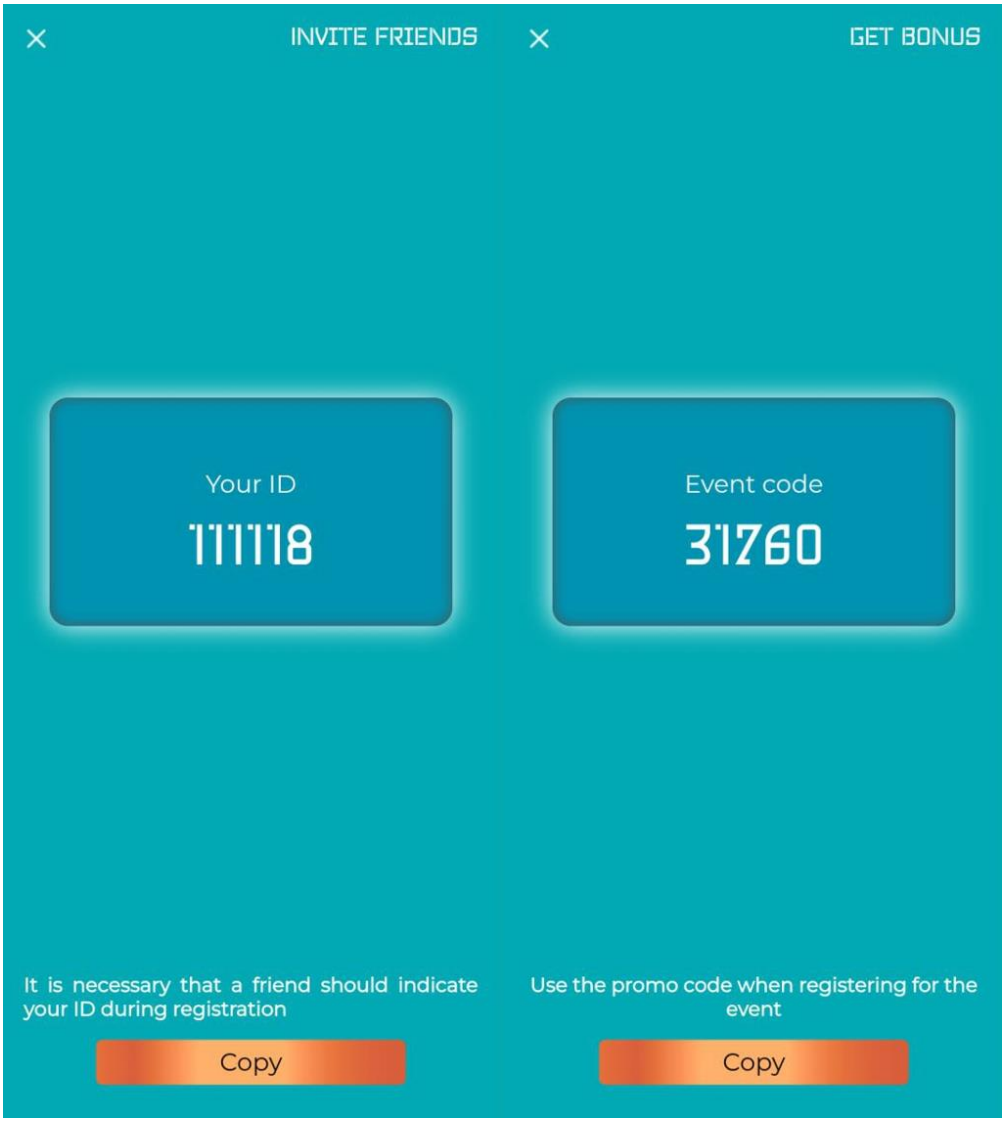
4.1.3 Hlavní menu a profil

Obrázek 16: Hlavní menu a obrazovka pro úpravu profilu, zdroj: vlastní zpracování

Po úspěšné registraci nebo přihlášení je uživatel přesměrován na hlavní obrazovku, která slouží jako centrální navigační bod aplikace. V horní části je umístěno jméno aplikace a hlavní menu, které poskytuje přístup ke klíčovým funkcím: PROFILE (úprava osobních údajů), PERSONAL ID (zobrazení unikátního referenčního kódu), EVENT CODE (správa bonusových kódů pro akce), EVENT LIST (přehled registrovaných událostí) a CONTACT US (kontaktní informace). Pod menu se nachází sekce „INVITE FRIENDS!“, která motivuje uživatele k využívání referenčního systému pro získání bonusů, jak je definováno ve funkčních požadavcích.

Zvolením možnosti PROFILE v menu je uživateli zobrazena obrazovka pro správu jeho osobních údajů. Formulář automaticky načte a zobrazí stávající data z databáze. Pole PERSONAL ID je zobrazeno pouze pro informaci a nelze jej měnit, jelikož se jedná o unikátní identifikátor v systému. Ostatní údaje, jako jsou NAME (jméno), SURNAME (příjmení), PHONE (telefon) a BIRTHDAY (datum narození), může uživatel libovolně upravit. Provedené změny se do databáze uloží po stisknutí tlačítka „Save“.

4.1.4 Osobní ID a Event kód



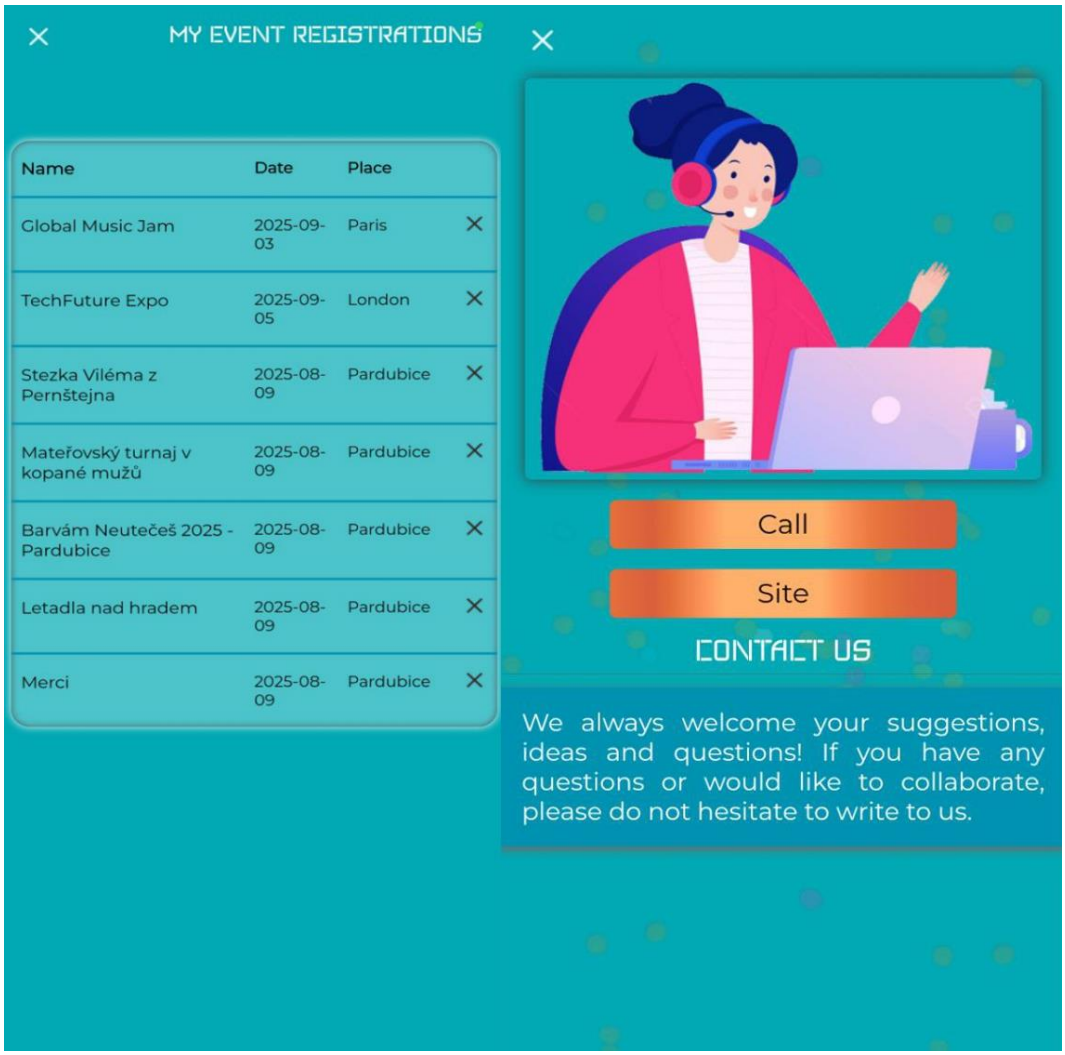
Obrázek 17: Zobrazení osobního ID a bonusového kódu, zdroj: vlastní zpracování

Aplikace implementuje referenční a bonusový systém, jak je specifikováno ve funkčních požadavcích. Pro interakci s těmito systémy jsou uživateli k dispozici dvě specializované obrazovky.

První obrazovka, přístupná z menu pod volbou PERSONAL ID, zobrazuje unikátní referenční kód uživatele („Your ID“). Tento kód slouží k pozvání nových uživatelů do aplikace. Pokud nový uživatel zadá tento kód při své registraci, původnímu uživateli se přičtou bonusové body („stars“). Pro usnadnění sdílení je pod kódem umístěno tlačítko „Copy“, které pomocí pluginu cordova-clipboard-plugin zkopíruje ID do systémové schránky zařízení.

Druhá obrazovka, nazvaná „GET BONUS“, zobrazuje speciální „Event code“. Tento kód může uživatel získat za aktivitu v aplikaci nebo od administrátora a použít jej při registraci na konkrétní událost pro získání definovaných výhod. Stejně jako u osobního ID, i zde je k dispozici tlačítko „Copy“ pro snadné zkopírování kódu.

4.1.5 Registrací na události a kontakt



Obrázek 18: Seznam registrací na události a kontaktní obrazovka, zdroj: vlastní zpracování

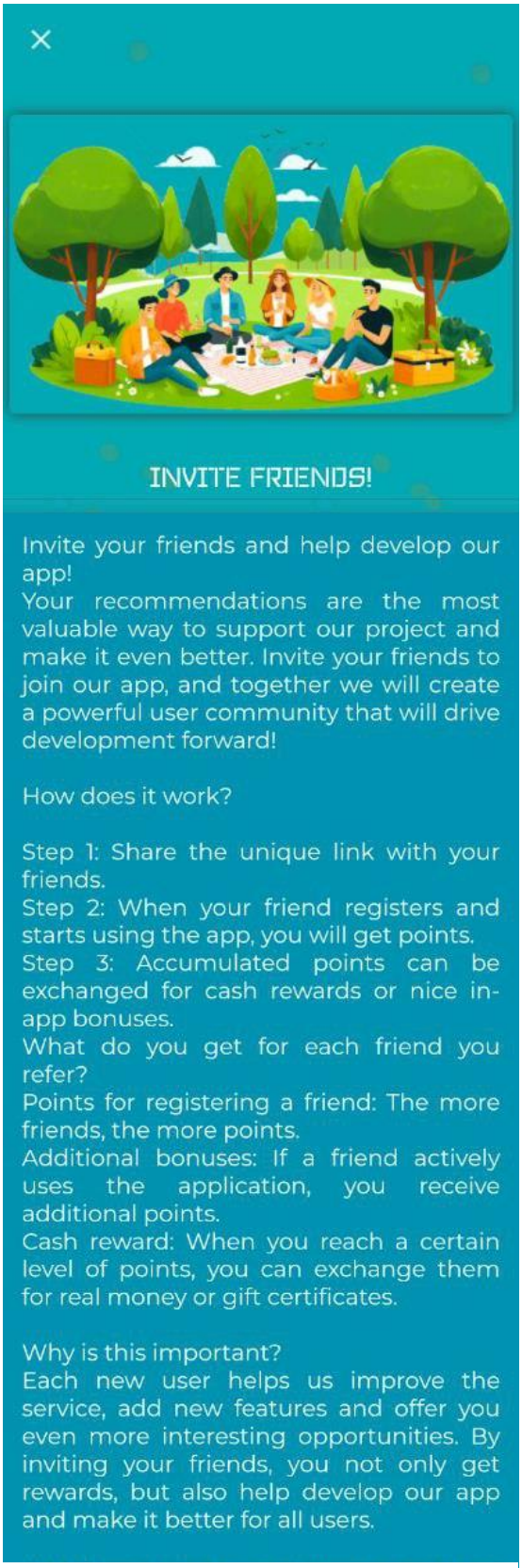
Pro kompletní správu uživatelské aktivity a zajištění zpětné vazby byly implementovány obrazovky pro přehled registrací a přímý kontakt na správce aplikace.

Obrazovka „MY EVENT REGISTRATIONS“, přístupná z hlavního menu, poskytuje uživateli centralizovaný přehled všech událostí, na které je aktuálně registrován. Seznam je strukturován do tabulky, která zobrazuje klíčové informace: název události (Name), datum konání (Date) a místo (Place). Každý záznam je doplněn o ikonu pro zrušení registrace, což uživateli umožňuje flexibilně spravovat svou účast na akcích. Tato funkcionalita je přímo napojena na tabulku `user_event` v databázi.

Sekce „CONTACT US“ slouží jako přímý komunikační kanál mezi uživatelem a týmem aplikace. Obsahuje dvě hlavní interaktivní tlačítka: „Call“ pro zahájení telefonního hovoru s podporou a „Site“ pro přesměrování na oficiální webové stránky. Pod tlačítky se nachází výzva pro uživatele,

aby zasílali své nápady a dotazy, čímž aplikace podporuje komunitní zpětnou vazbu pro svůj další rozvoj.

4.1.6 Popis referenčního systému „Invite Friends“



Obrázek 19: Referenčny systém „Invite Friends“, zdroj: vlastní zpracování

Klíčovou součástí strategie pro růst uživatelské základny a komunity je implementovaný referenční systém, který je uživateli prezentován na obrazovce „Invite Friends“. Tato sekce detailně vysvětluje mechaniku a výhody pozvání nových uživatelů do aplikace.

Systém funguje na principu sdílení unikátního osobního ID. Stávající uživatel sdílí svůj kód s přáteli, a pokud jej nový uživatel zadá při registraci, oba získají definované bonusy. Obrazovka přehledně popisuje celý proces a motivuje k zapojení tím, že specifikuje možné odměny, jako jsou bonusové body („stars“), které lze později směnit za věcné ceny nebo dárkové certifikáty. Zdůrazněn je také komunitní aspekt – zapojením přátel uživatelé nejen získávají odměny, ale také aktivně přispívají k rozvoji a vylepšování celé služby.

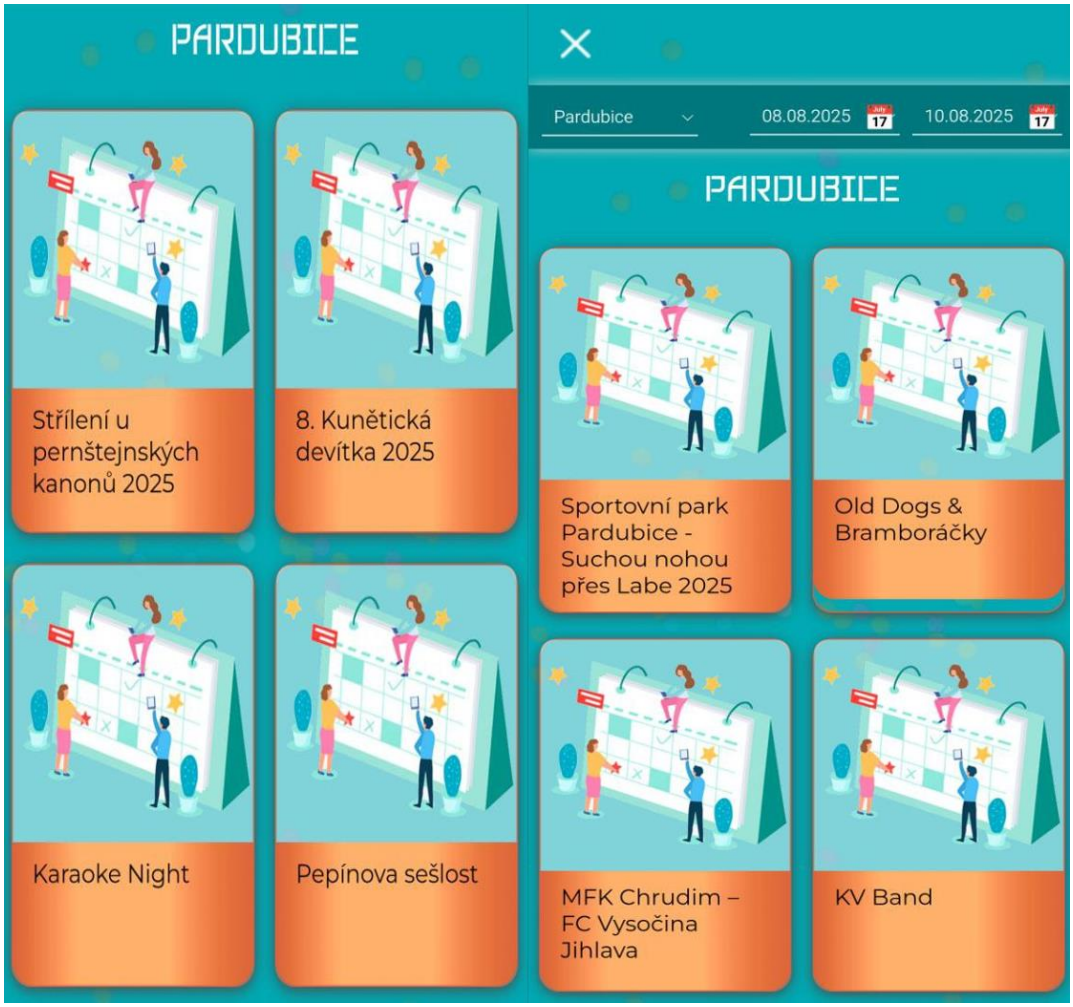
4.1.7 Přehled kategorií událostí

Obrázek 20: Kategorie událostí, zdroj: vlastní zpracování

Hlavní obrazovka aplikace slouží jako rozcestník pro procházení dostupných událostí. Pro maximální přehlednost a intuitivní navigaci jsou veškeré akce roztříděny do tematických kategorií, které jsou uživateli prezentovány ve formě vizuálně oddělených bloků. Každá kategorie, jako například EDUCATIONAL EVENTS, MUSIC EVENTS nebo OUTDOOR AND ADVENTURE, je reprezentována názvem a sadou dlaždic s konkrétními událostmi.

Tato struktura umožňuje uživatelům rychle se zorientovat a zaměřit se na typ akcí, které je zajímají. Kliknutím na název kategorie nebo na ikonu plus (+) se rozbalí kompletní seznam událostí v dané sekci. Toto členění odpovídá datovému modelu, kde je každá událost v tabulce event propojena s příslušnou kategorií z tabulky categories. V horní části obrazovky je rovněž integrován panel „INVITE FRIENDS!“, který konzistentně napříč aplikací motivuje k využívání referenčního systému.

4.1.8 Seznam událostí a filtrování



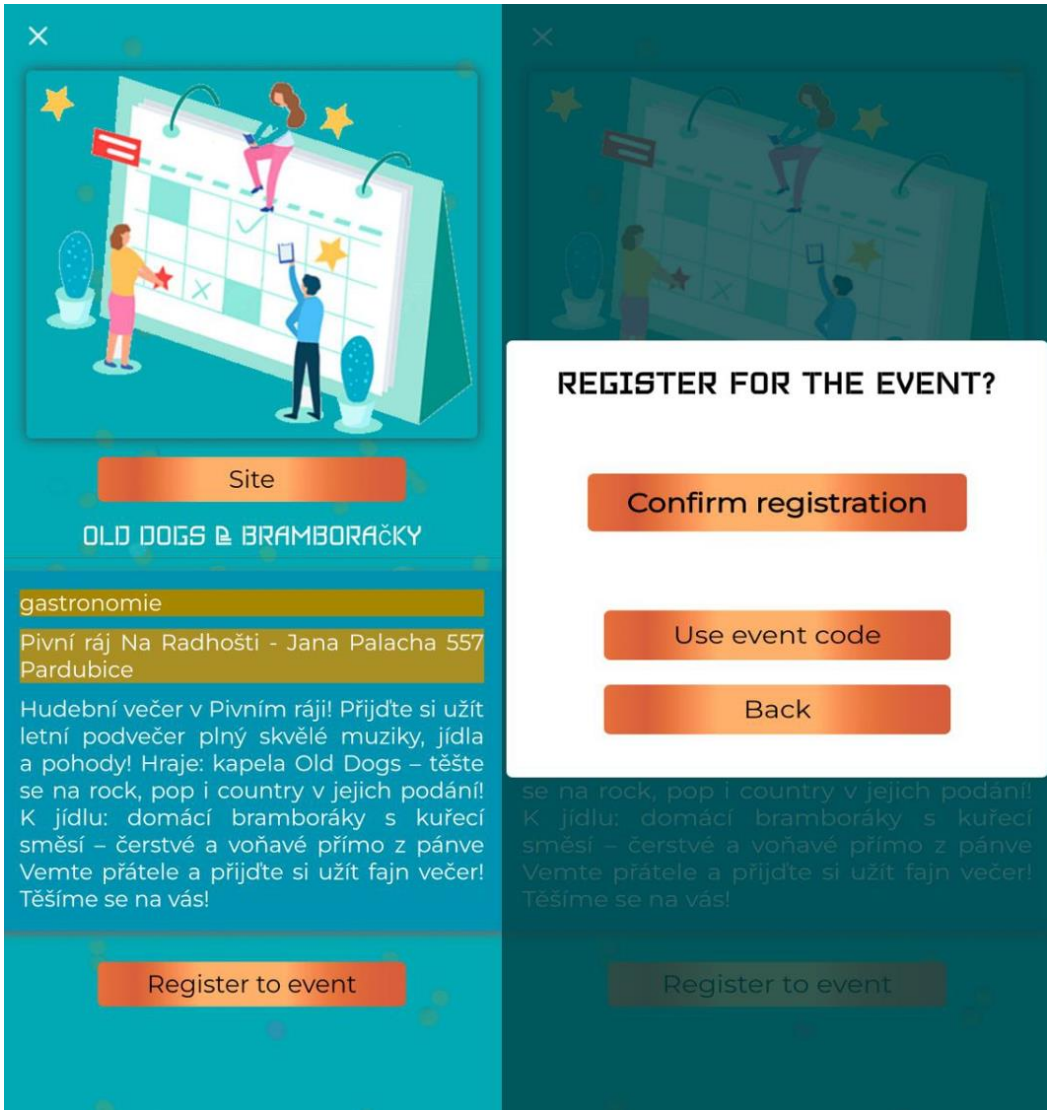
Obrázek 21: Seznam událostí s aktivním filtrem, zdroj: vlastní zpracování

Po výběru konkrétní kategorie, v tomto případě událostí pro město Pardubice, je uživateli zobrazen přehledný seznam relevantních akcí. Každá událost je prezentována formou vizuální dlaždice, která obsahuje název a tematický obrázek, což usnadňuje rychlou orientaci.

V horní části obrazovky je umístěn panel pro pokročilé filtrování, který umožňuje dále specifikovat výběr. Uživatel může omezit zobrazené události podle lokality (výběrem města z roletového menu) a podle časového období (nastavením počátečního a koncového data pomocí kalendářových ikon). Tato funkcionalita je klíčová pro efektivní plánování a vyhledávání akcí a odpovídá funkčním

požadavkům na filtrování a třídění událostí podle různých kritérií. Po nastavení filtru se seznam dynamicky aktualizuje a zobrazí pouze ty události, které odpovídají zadaným parametrům.

4.1.9 Detail a registrace na událost



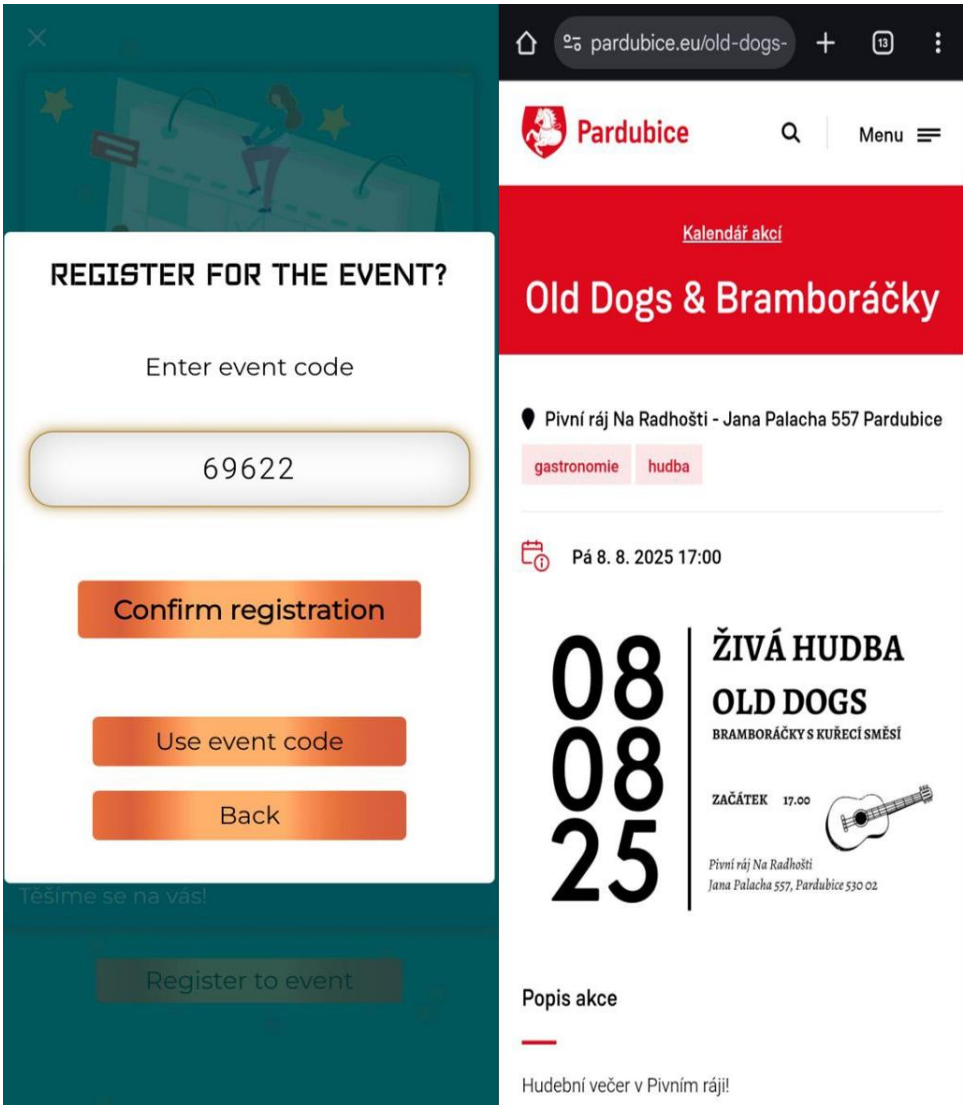
Obrázek 22: Detail události a proces registrace, zdroj: vlastní zpracování

Po kliknutí na konkrétní událost v seznamu je uživateli zobrazena její detailní stránka. Tato obrazovka poskytuje veškeré podstatné informace: název akce, odkaz na webové stránky organizátora („Site“), zařazení do kategorie (např. „gastronomie“) a podrobný textový popis. Ve spodní části obrazovky se nachází tlačítko „Register to event“, které iniciuje proces registrace na danou akci.

Po stisknutí tohoto tlačítka se zobrazí modální okno s výzvou „REGISTER FOR THE EVENT?“. Uživatel má na výběr ze tří možností. Tlačítko „Confirm registration“ slouží pro standardní přihlášení na akci. Volba „Use event code“ umožňuje zadat speciální bonusový kód, jak je definováno ve funkčních požadavcích. Tlačítkem „Back“ se lze vrátit na detail události bez

provedení registrace. Tento vícekrokový proces zajišťuje, že registrace je provedena vědomě a zároveň nabízí možnost uplatnění bonusů.

4.1.10 Zadání kódu a webová stránka



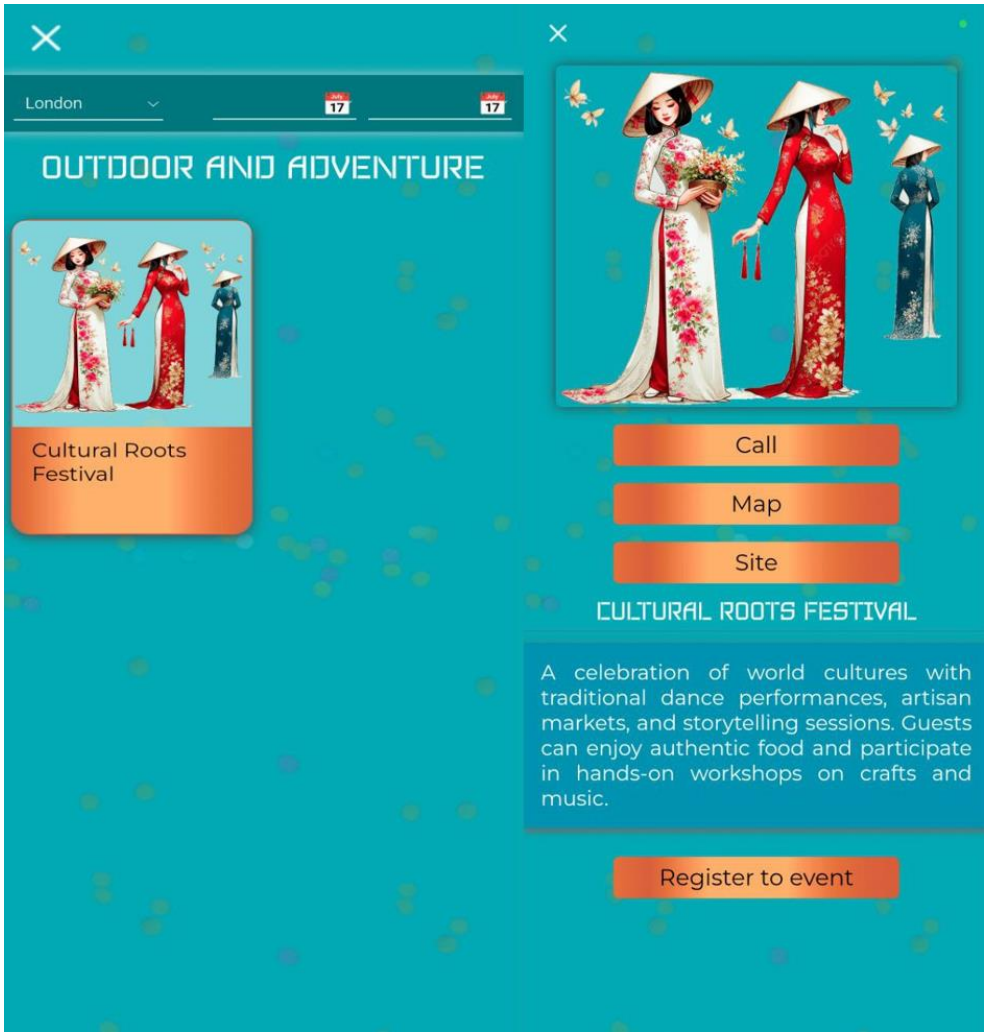
Obrázek 23: Zadání event kódu a zobrazení webové stránky události, zdroj: vlastní zpracování

Proces registrace na událost nabízí pokročilé možnosti, které zvyšují uživatelský komfort a propojují aplikaci s externími zdroji. Pokud uživatel v modálním okně zvolí možnost „Use event code“, rozhraní se dynamicky změní a zobrazí pole pro zadání bonusového kódu. Po jeho vložení a potvrzení tlačítkem „Confirm registration“ systém ověří platnost kódu a v případě úspěchu dokončí registraci s příslušnými výhodami.

Další klíčovou funkcí je přímé propojení na webové stránky organizátora (parsování). Po kliknutí na tlačítko „Site“ na detailu události aplikace plynule přesměruje uživatele na externí webovou stránku, kde si může prohlédnout doplňující informace, které nejsou součástí mobilní aplikace, nebo případně zakoupit vstupenky. Toto propojení zajišťuje, že uživatel má přístup ke všem relevantním

informacím na jednom místě, aniž by musel aplikaci opouštět a ručně vyhledávat stránku v prohlížeči.

4.1.11 Výpis a detail události



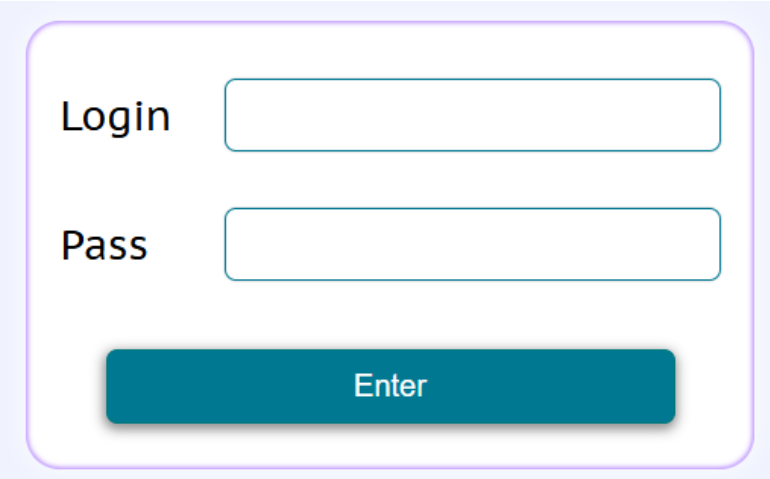
Obrázek 24: Zadání event kódu a zobrazení webové stránky události, zdroj: vlastní zpracování

Po výběru kategorie, například OUTDOOR AND ADVENTURE, je uživateli zobrazen seznam příslušných událostí. Kliknutím na konkrétní akci, v tomto případě „Cultural Roots Festival“, se otevře její detailní obrazovka, která centralizuje veškeré potřebné informace pro uživatele.

V horní části detailu dominuje tematický obrázek, pod nímž jsou umístěna interaktivní tlačítka s klíčovými funkcemi: „Call“ pro přímé telefonické spojení s organizátorem, „Map“ pro zobrazení místa konání v mapové aplikaci a „Site“ pro přesměrování na oficiální web. Níže je uveden název události a její podrobný popis. Celá obrazovka je zakončena tlačítkem „Register to event“, které uživatele provede procesem registrace na danou akci.

4.2 Administrační část

4.2.1 Přihlášení do administrace

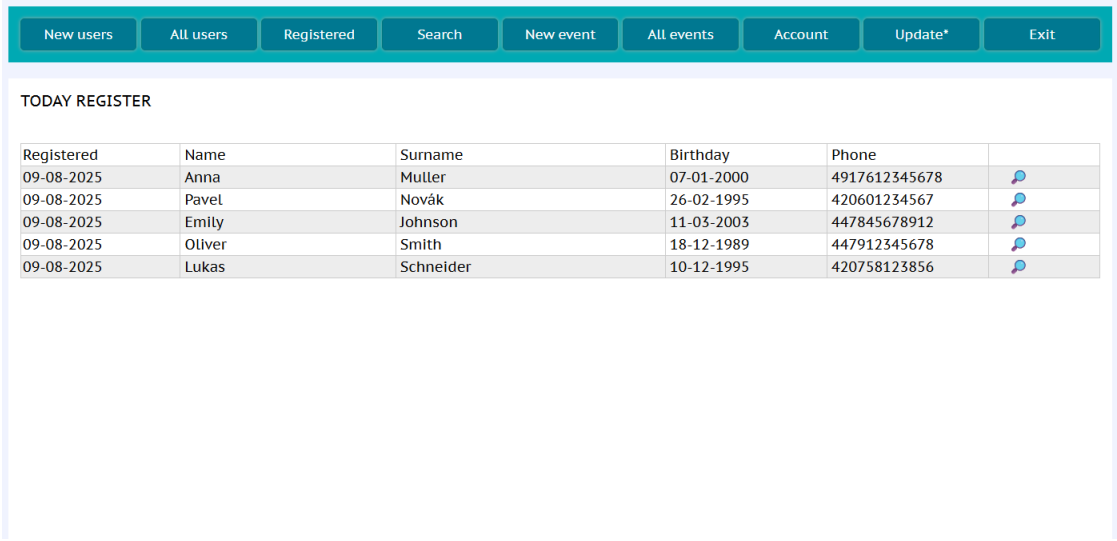


Obrázek 25: Přihlašovací formulář do administrativního rozhraní, zdroj: vlastní zpracování

Přístup do administrativní části aplikace, která slouží ke správě obsahu a uživatelů, je zabezpečen přihlašovacím formulářem. Má přístup pouze oprávněná osoba (administrátor).

Rozhraní je záměrně minimalistické a obsahuje pouze dvě vstupní pole – pro zadání přihlašovacího jména (Login) a hesla (Pass) – a tlačítko pro potvrzení (Enter). Po odeslání formuláře jsou zadané údaje zaslány na server, kde dojde k jejich ověření. Systém porovná zadané heslo s hashem uloženým v databázové tabulce account. V případě shody je pro administrátora vytvořena PHP session, která ho autorizuje pro přístup do všech částí administrace. Pokud údaje nesouhlasí, přístup je odepřen a uživatel zůstává na přihlašovací stránce.

4.2.2 Přehled nových uživatelů

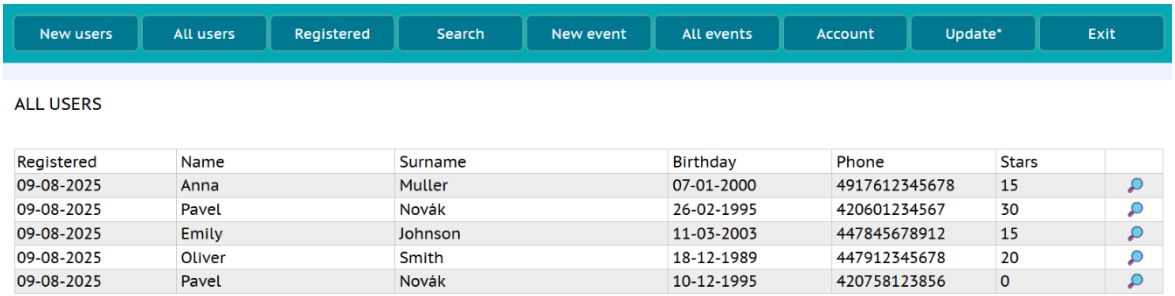


Obrázek 26: Přehled nově registrovaných uživatelů, zdroj: vlastní zpracování

Po úspěšném přihlášení je administrátor přesměrován na hlavní panel administrativního rozhraní. V horní části se nachází navigační lišta, která poskytuje rychlý přístup ke všem klíčovým modulům správy: New users, All users, Search, New event, All events, Account, Update a Exit.

Výchozí zobrazenou sekcí je „TODAY REGISTER“, která poskytuje okamžitý přehled o nově registrovaných uživatelích za aktuální den. Data jsou prezentována v přehledné tabulce, která obsahuje základní údaje o uživateli: datum registrace (Registered), jméno (Name), příjmení (Surname), datum narození (Birthday) a telefonní číslo (Phone). Na pravé straně každého řádku se nachází ikona lupy, která slouží k zobrazení detailních informací o daném uživateli, včetně možnosti správy jeho účtu. Tento přehled umožňuje administrátorovi efektivně monitorovat aktivitu a růst uživatelské základny.

4.2.3 Seznam uživatelů

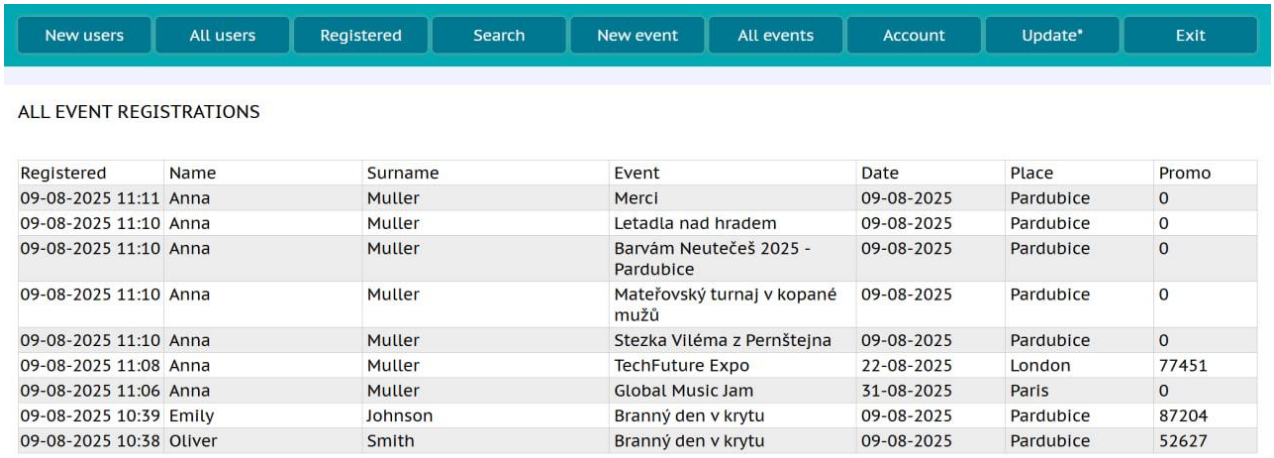


Obrázek 27: Seznam všech uživatelů, zdroj: vlastní zpracování

Kromě přehledu nově příchozích uživatelů poskytuje administrativní rozhraní také kompletní výpis všech uživatelských účtů v systému. Tato sekce, přístupná přes záložku All users, je klíčovým nástrojem pro celkovou správu uživatelské databáze.

Tabulka zobrazuje nejen základní údaje jako datum registrace, jméno, příjmení, datum narození a telefon, ale navíc obsahuje i sloupec Stars. Tento sloupec vizualizuje aktuální stav bodového konta každého uživatele, což je přímé propojení na bodový systém popsaný ve funkčních požadavcích. Administrátor tak má okamžitý přehled o aktivitě a nasbíraných bonusech jednotlivých uživatelů. I zde je u každého záznamu k dispozici ikona lupy pro přechod na detailní správu konkrétního účtu.

4.2.4 Přehled registrací na události



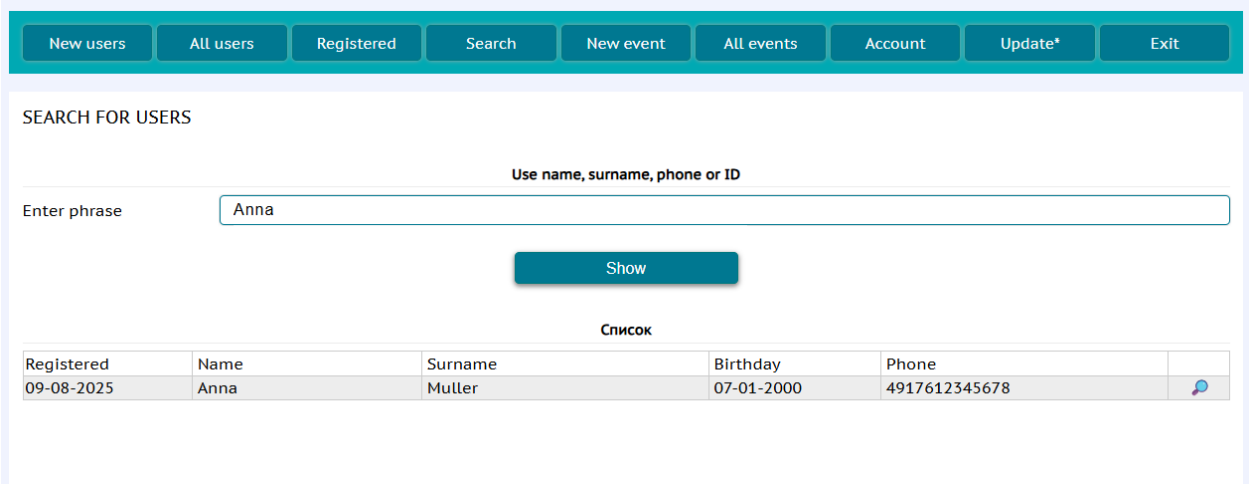
Registered	Name	Surname	Event	Date	Place	Promo
09-08-2025 11:11	Anna	Muller	Merci	09-08-2025	Pardubice	0
09-08-2025 11:10	Anna	Muller	Letadla nad hradem	09-08-2025	Pardubice	0
09-08-2025 11:10	Anna	Muller	Barvám Neutečeš 2025 - Pardubice	09-08-2025	Pardubice	0
09-08-2025 11:10	Anna	Muller	Mateřovský turnaj v kopané mužů	09-08-2025	Pardubice	0
09-08-2025 11:10	Anna	Muller	Stezka Viléma z Pernštejna	09-08-2025	Pardubice	0
09-08-2025 11:08	Anna	Muller	TechFuture Expo	22-08-2025	London	77451
09-08-2025 11:06	Anna	Muller	Global Music Jam	31-08-2025	Paris	0
09-08-2025 10:39	Emily	Johnson	Branný den v krytu	09-08-2025	Pardubice	87204
09-08-2025 10:38	Oliver	Smith	Branný den v krytu	09-08-2025	Pardubice	52627

Obrázek 28: Přehled všech registrací na události, zdroj: vlastní zpracování

Administrativní rozhraní poskytuje v sekci Registered centralizovaný přehled všech registrací, které uživatelé provedli na jednotlivé události. Tato funkcionalita je klíčová pro monitorování popularity akcí a sledování aktivity uživatelů.

Zobrazená tabulka „ALL EVENT REGISTRATIONS“ detailně vypisuje každý záznam o registraci. Obsahuje sloupce s datem registrace (Registered), jménem a příjmením uživatele (Name, Surname), názvem události (Event), datem a místem jejího konání (Date, Place). Dále je zde sloupec Promo, který eviduje, zda byl při registraci použit speciální event kód. Tento přehled je přímou vizualizací dat z propojovací tabulky user_event (v práci také označované jako user_akce), která v databázi uchovává vztahy mezi uživateli a událostmi.

4.2.5 Vyhledávání uživatelů

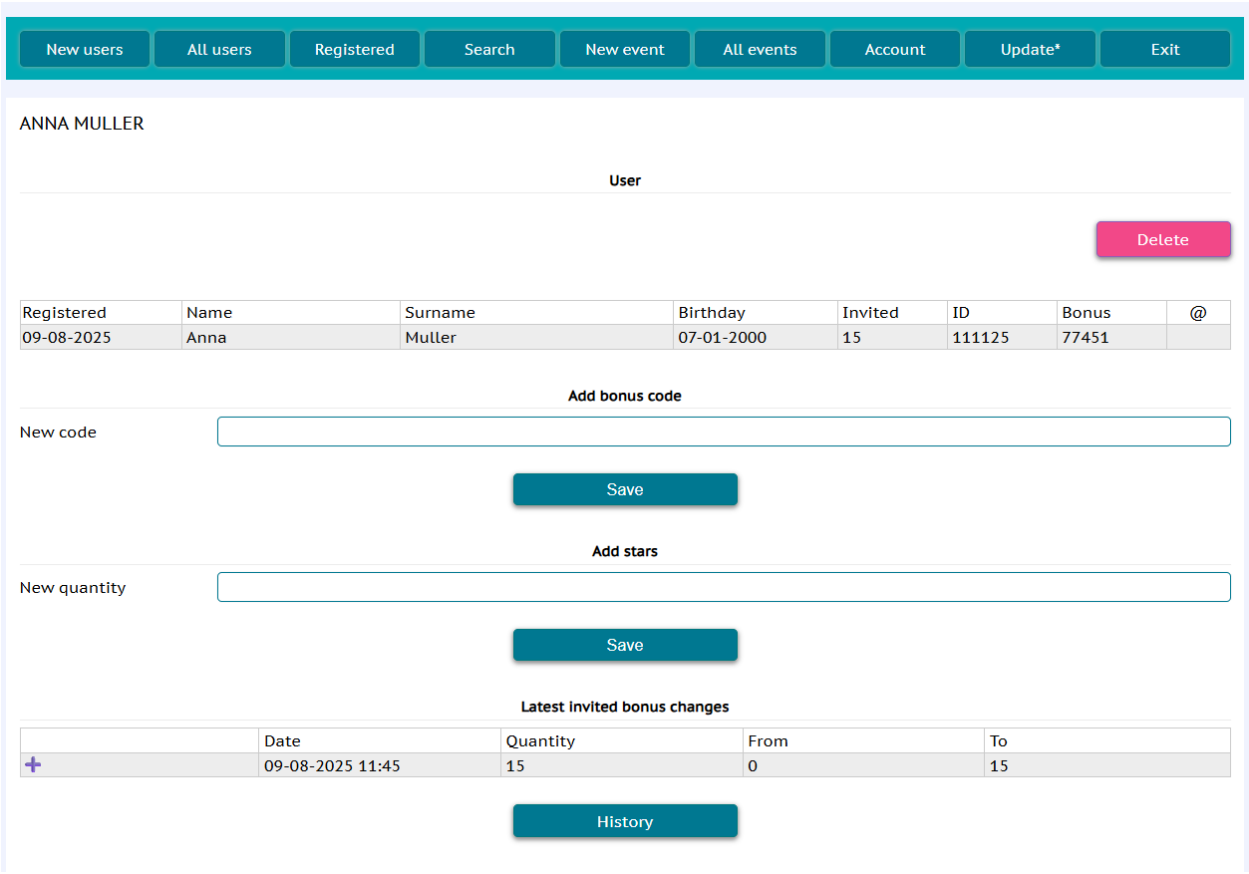


Obrázek 29: Vyhledávání uživatelů v administrativním rozhraní, zdroj: vlastní zpracování

Pro efektivní práci s rozsáhlou databází uživatelů je v administrativním rozhraní implementována funkce vyhledávání, přístupná přes záložku Search. Tento nástroj umožňuje administrátorovi rychle lokalizovat konkrétní uživatelské účty bez nutnosti manuálního procházení kompletních seznamů.

Rozhraní obsahuje jednoduché textové pole „Enter phrase“, do kterého může administrátor zadat hledaný výraz. Jak napovídá instrukce „Use name, surname, phone or ID“, systém je navržen tak, aby prohledával databázi podle několika klíčových atributů. Po zadání dotazu a stisknutí tlačítka „Show“ provede aplikace dotaz na server, který vrátí všechny odpovídající záznamy. Výsledky jsou následně zobrazeny v přehledné tabulce pod vyhledávacím formulářem, přičemž struktura tabulky je konzistentní s ostatními přehledy uživatelů.

4.2.6 Detailní správa uživatelského účtu



Obrázek 30: Správa uživatelského účtu, zdroj: vlastní zpracování

Po kliknutí na ikonu lupy v seznamu uživatelů je administrátor přesměrován na obrazovku detailní správy konkrétního uživatelského účtu. Tato sekce poskytuje komplexní nástroje pro správu všech aspektů uživatelského profilu, v souladu s administrativními funkcemi definovanými v zadání práce.

V horní části jsou zobrazeny souhrnné informace o uživateli, včetně jeho ID, počtu pozvaných přátel (Invited) a aktuálního bonusového kódu (Bonus). Administrátor zde má možnost účet zcela odstranit pomocí tlačítka Delete.

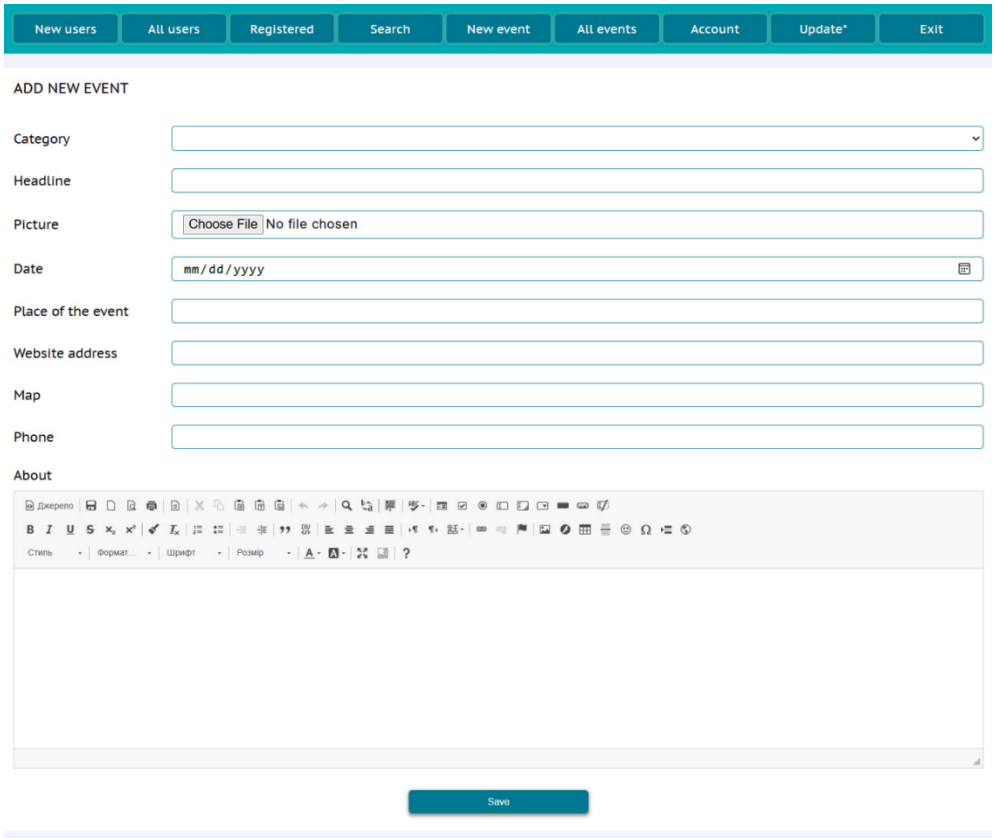
Níže se nacházejí moduly pro specifické akce:

- Add bonus code: Tento formulář umožňuje administrátorovi ručně přidělit nebo změnit speciální „event kód“ uživatele.
- Add stars: Zde může administrátor manuálně upravovat stav bodového konta uživatele, například jako odměnu za aktivitu nebo pro korekci.

Ve spodní části je umístěn přehled posledních změn v bodovém systému („Latest invited bonus changes“), který pro transparentnost zobrazuje datum, množství a stav bodů před a po změně.

Tlačítko History pak vede na kompletní historii bodových transakcí daného uživatele, což je vizualizace dat z tabulky stars.

4.2.7 Formulář pro vytvoření nové události



Obrázek 31: Formulář pro vytvoření nové události, zdroj: vlastní zpracování

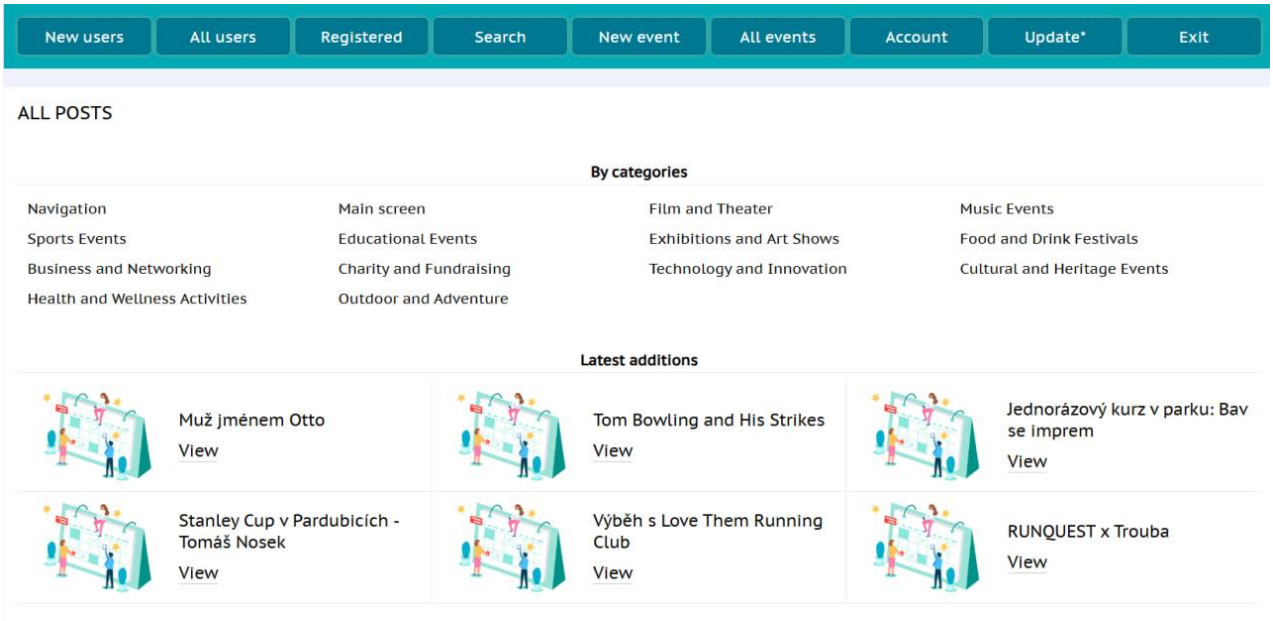
Pro správu obsahu aplikace je klíčovou funkcí administrace možnost přidávat nové události. K tomuto účelu slouží formulář „ADD NEW EVENT“, přístupný z hlavní navigační lišty. Tento formulář umožňuje administrátorovi zadat veškeré potřebné informace o nové akci, které se následně zobrazí uživatelům v mobilní aplikaci.

Formulář je strukturován do několika polí, která odpovídají atributům v databázové tabulce event:

- Category: Roletové menu pro výběr kategorie, do které událost spadá. Seznam kategorií je dynamicky načítán z tabulky categories.
- Headline: Název události.
- Picture: Možnost nahrát obrázek, který bude událost vizuálně reprezentovat.
- Date, Place of the event, Website address, Map, Phone: Pole pro zadání praktických informací, jako je datum a místo konání, odkaz na web, mapu a kontaktní telefon.

- About: Pro podrobný popis události je implementován WYSIWYG (What You See Is What You Get) editor, který umožňuje formátování textu, vkládání odkazů či seznamů pro vytvoření přehledného a strukturovaného obsahu.

Po vyplnění všech údajů a stisknutí tlačítka Save jsou data odeslána na server, kde proběhne jejich validace a následné uložení do databáze.

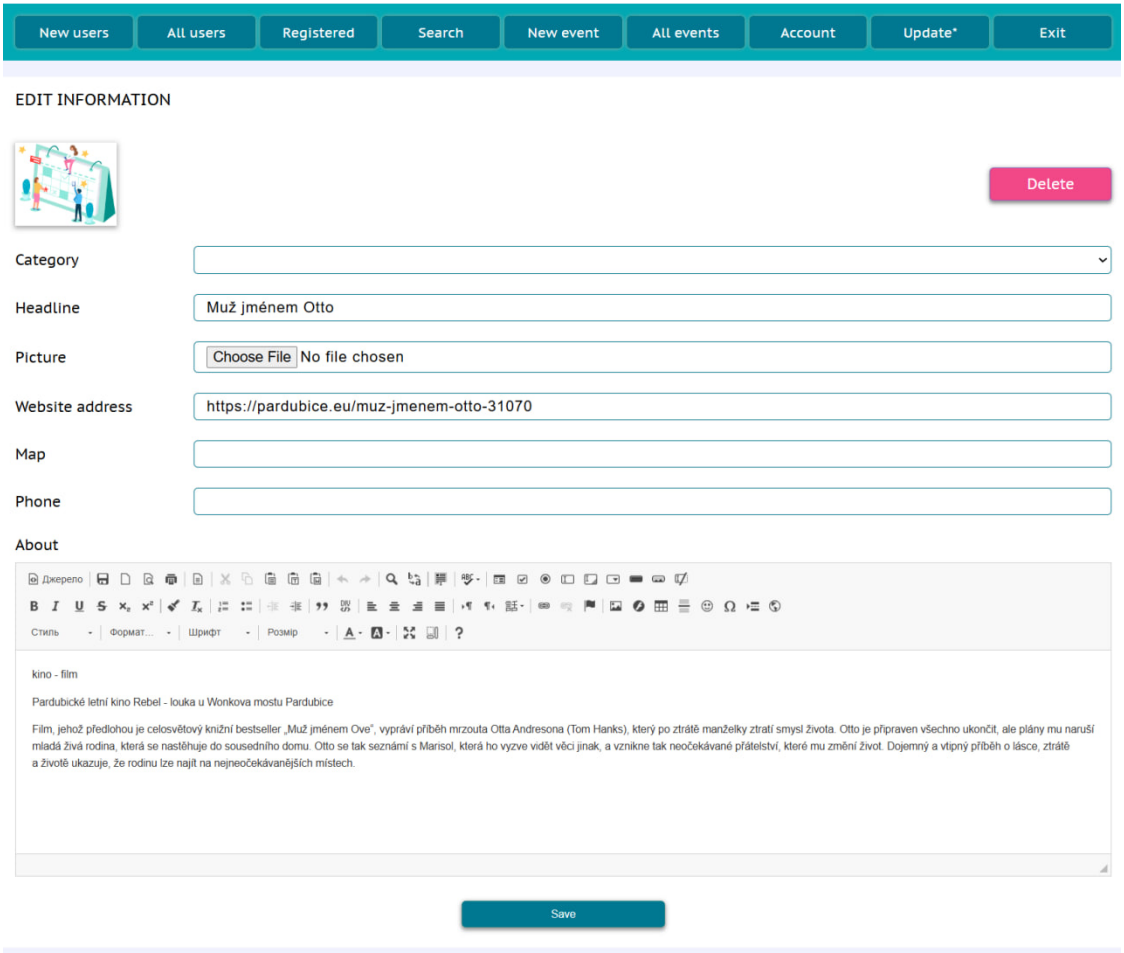


Obrázek 32: Přehled událostí podle kategorií, zdroj: vlastní zpracování

4.2.8 Správa událostí v administraci

V sekci All events má administrátor k dispozici souhrnný přehled všech událostí vložených do systému. Rozhraní je navrženo pro snadnou orientaci a rychlou správu obsahu. V horní části je zobrazen seznam všech dostupných kategorií, které jsou načítány z databázové tabulky categories. Kliknutím na název kategorie může administrátor filtrovat události a zobrazit pouze ty relevantní.

Pod seznamem kategorií se nachází sekce „Latest additions“, která zobrazuje nejnověji přidané události ve formě přehledných dlaždic. U každé události je uveden její název a odkaz View pro přechod na detailní zobrazení a úpravu.



Obrázek 33: Formulář pro úpravu existující události, zdroj: vlastní zpracování

Po kliknutí na odkaz View u konkrétní události je administrátor přesměrován na formulář „EDIT INFORMATION“. Tento formulář je vizuálně i funkčně shodný s formulářem pro přidání nové události, avšak s klíčovým rozdílem: všechna pole jsou již předvyplněna aktuálními daty z databáze.

Administrátor zde může upravit jakýkoliv údaj o události – od kategorie a názvu, přes obrázek, až po detailní popis v WYSIWYG editoru. Provedené změny se uloží do databáze po stisknutí tlačítka Save. V případě, že je událost již neaktuální nebo byla zadána chybně, je možné ji ze systému trvale odstranit pomocí výrazně označeného tlačítka Delete.

5 Testování

Testování představuje klíčovou fázi vývojového cyklu, jejímž cílem je ověření funkčnosti, stability, bezpečnosti a celkové kvality aplikace před jejím nasazením. Pro zajištění komplexního pokrytí byla zvolena vícevrstvá strategie testování, která zahrnovala různé metody a techniky.

5.1 Jednotkové testování

Jednotkové testy (Unit Testing) byly zaměřeny na ověření funkčnosti nejmenších, izolovaných částí kódu – typicky jednotlivých funkcí nebo metod v rámci tříd. Cílem bylo zajistit, že každá taková jednotka funguje správně a vrací očekávané výsledky pro různé vstupní hodnoty. V kontextu serverové části (backendu) byly testovány především metody v modelech, které jsou zodpovědné za validaci dat a přípravu dat pro uložení do databáze.

5.2 Integrační testování

Po ověření jednotlivých jednotek následovalo integrační testování (Integration Testing), jehož úkolem bylo ověřit správnou spolupráci a komunikaci mezi různými moduly systému. Nejdůležitější částí bylo testování propojení mezi klientskou mobilní aplikací (frontend) a serverovým rozhraním (backend API). Ověřovaly se kompletní scénáře, které vyžadovaly součinnost více komponent, například:

- Registrace uživatele: Odeslání dat z registračního formuláře v mobilní aplikaci, jejich správné zpracování na serveru, uložení do databáze MySQL a odeslání potvrzovací odpovědi zpět do aplikace.
- Vytvoření události administrátorem: Odeslání dat z formuláře pro novou událost v administrativním rozhraní, uložení do databáze a následné ověření, že se nová událost správně zobrazí v seznamu událostí v mobilní aplikaci pro běžného uživatele.
- Přihlášení na událost: Zpracování požadavku na registraci na událost z mobilní aplikace, vytvoření správného záznamu v propojovací tabulce user_event a ověření, že se tato registrace projeví jak v profilu uživatele, tak v přehledech pro administrátora.
- Aktualizace profilu: Odeslání upravených údajů z profilu uživatele, jejich aktualizace v databázi a ověření, že se po novém načtení v aplikaci zobrazí již nová data.

5.3 Uživatelské akceptační testování

Tato fáze testování byla zaměřena na ověření aplikace z pohledu koncového uživatele (User Acceptance Testing – UAT). Cílem nebylo testovat kód, ale ověřit, že je aplikace intuitivní,

srozumitelná a splňuje všechny funkční požadavky definované v zadání. Byly připraveny a provedeny testovací scénáře, které simulovaly reálné použití aplikace oběma typy rolí.

Scénáře pro běžného uživatele

- Úspěšné vytvoření nového uživatelského účtu přes vícekrokový formulář.
- Přihlášení do aplikace a následná úprava údajů ve vlastním profilu.
- Zobrazení a zkopírování osobního ID a event kódu.
- Procházení seznamu událostí, jejich filtrování podle města a data.
- Zobrazení detailu konkrétní události.
- Úspěšná registrace na událost (standardní i s použitím event kódu).
- Zrušení registrace na událost.
- Ověření správného zobrazení všech sekcí (kontakty, pozvání přátel).

Scénáře pro administrátora

- Úspěšné přihlášení do administrativního rozhraní.
- Vytvoření nové události, včetně nahrání obrázku a formátování popisu.
- Úprava a následné smazání existující události.
- Vyhledání konkrétního uživatele podle jména, příjmení a telefonu.
- Zobrazení detailu uživatele a kontrola historie jeho bonusových bodů.
- Ruční přidělení bonusových bodů ("stars") a event kódu konkrétnímu uživateli.
- Kontrola přehledů (noví uživatelé, všechny registrace na akce).
- Ověření, že změny provedené v administraci (např. úprava názvu události) se okamžitě a správně projevily v mobilní aplikaci pro běžného uživatele.

5.4 Výkonnostní testování

V rámci výkonnostních testů byla ověřována odezva a stabilita aplikace, aby splňovala stanovené nefunkční požadavky. Byl měřen čas odezvy serverového API na klíčové požadavky (např. načtení seznamu událostí), aby se zajistilo, že nepřesahuje stanovený limit jedné sekundy. Dále byla sledována plynulost a rychlost reakcí uživatelského rozhraní na mobilních zařízeních.

6 Nasazení

Hosting

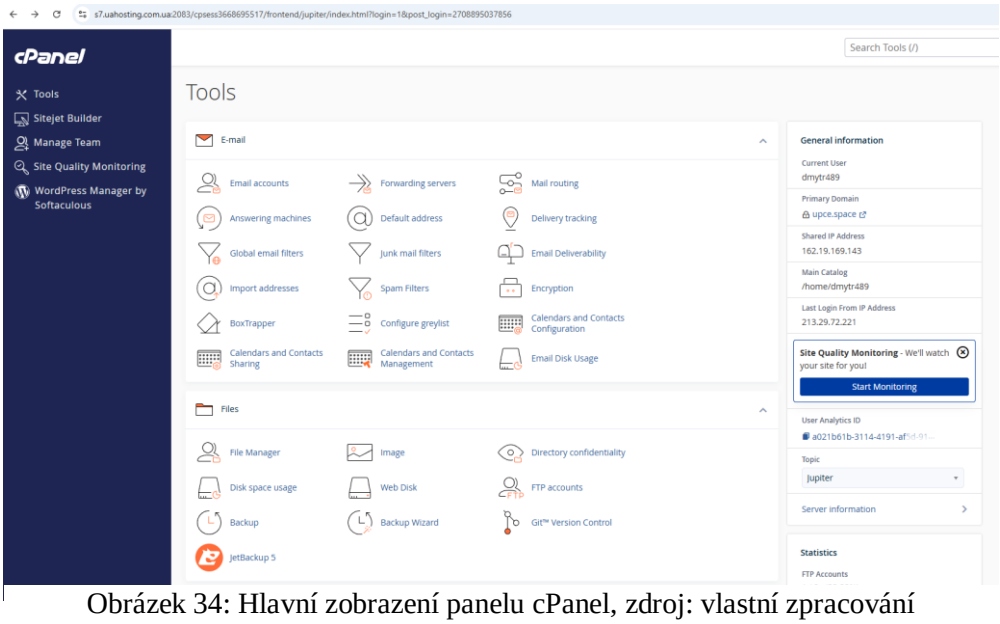
Aby byla webová aplikace dostupná uživatelům přes internet, je potřeba ji nahrát na hosting. To znamená, že veškeré soubory projektu a databáze musí být umístěny na server, který je dostupný 24/7 a podporuje potřebné technologie (např. PHP, MySQL atd.) [13].

Dnes existuje spousta typů hostingu – od jednoduchého sdíleného hostingu, přes VPS až po vyhrazené servery. V našem případě byl použit klasický sdílený (virtuální) hosting s cPanelem, což je rozhraní, které výrazně usnadňuje správu webu.

Pomocí cPanelu je možné:

- spravovat soubory přes vestavěný správce nebo FTP,
- vytvářet a upravovat databáze (např. přes phpMyAdmin),
- připojit doménu a nastavit e-mailové adresy,
- nastavovat přístupy, zálohování a další funkce – vše bez potřeby práce přes příkazový řádek.

cPanel je mezi hostingovými panely jeden z nejrozšířenějších a nejpřívětivějších. Díky svému grafickému rozhraní umožňuje i méně technicky zdatným uživatelům bez problémů nasadit a spravovat webovou aplikaci.

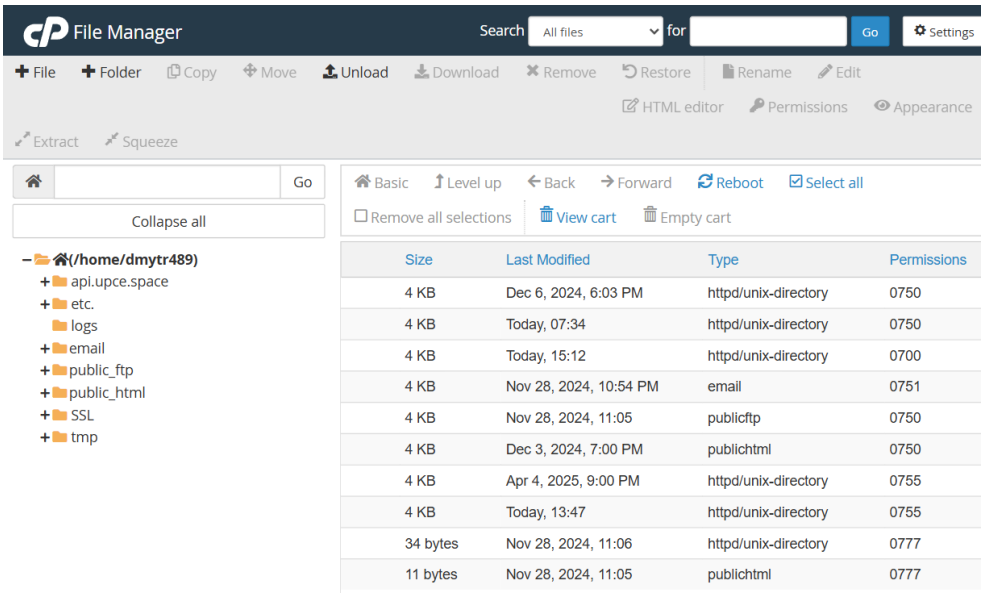


Obrázek 34: Hlavní zobrazení panelu cPanel, zdroj: vlastní zpracování

Součástí cPanelu je také integrovaný správce souborů, který umožňuje nahrávat, mazat, přesouvat nebo upravovat soubory přímo přes webové rozhraní. Tento nástroj se hodí především pro rychlé úpravy jednotlivých souborů nebo pro kontrolu struktury složek přímo na serveru.

Pro nahrání většího množství souborů (např. celého projektu) je ale efektivnější využít FTP klienta – například oblíbený WinSCP. Ten umožňuje pohodlně připojit se k serveru a hromadně přetahovat soubory či složky z lokálního disku do hostovaného prostoru. Tento způsob je rychlejší a vhodnější pro plnohodnotné nasazení aplikace.

Díky tomu lze snadno přenést celý projekt na server, nahrát databázi a zpřístupnit aplikaci veřejnosti.



Obrázek 35: Správce souborů, zdroj: vlastní zpracování

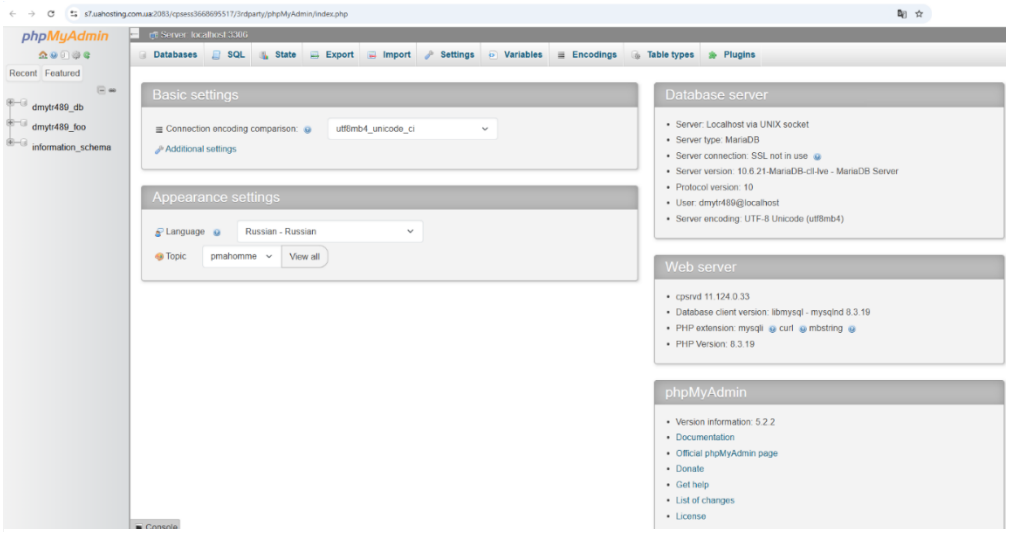
Kromě správy souborů nabízí cPanel také jednoduché nástroje pro práci s databázemi. Hlavním z nich je phpMyAdmin – webová aplikace, která umožňuje pohodlnou správu databází typu MySQL přímo z prohlížeče [13].

Pomocí phpMyAdminu lze například:

- vytvářet a upravovat databázové tabulky,
- prohlížet a editovat jednotlivé záznamy,
- spouštět vlastní SQL dotazy,
- importovat/exportovat databáze (např. ve formátu .sql při nasazení aplikace).

Vedle toho umožňuje cPanel i správu samotných databází – novou databázi lze vytvořit pomocí několika kliknutí. Stejně snadno se přidávají i uživatelé databáze, kterým lze nastavit různá práva (např. pouze čtení, nebo úplná správa) [19].

To vše se dá udělat bez nutnosti psát jediný řádek kódu nebo používat příkazový řádek, což výrazně zjednodušuje nasazení a údržbu aplikace.



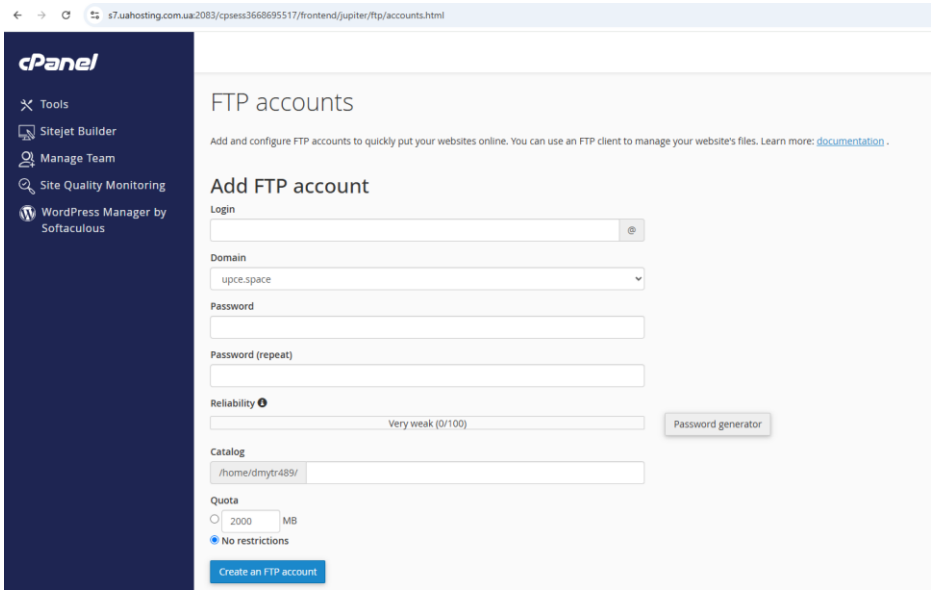
Obrázek 36: Rozhraní phpMyAdmin, zdroj: vlastní zpracování

Díky rozhraní cPanel je práce s webem na hostingu výrazně jednodušší. Vývojář se tak může soustředit na samotnou aplikaci, aniž by musel řešit složité serverové příkazy nebo konfigurace.

Aby se předešlo zbytečným chybám, byl ještě před nasazením projektu sestaven přehledný pracovní postup:

1. Vytvoření FTP účtu v cPanelu
2. Nastavení připojení přes WinSCP
3. Nahrání souborů projektu na server
4. Vytvoření databáze v cPanelu
5. Vytvoření uživatele databáze a přiřazení oprávnění
6. Import databáze pomocí phpMyAdmin
7. Kontrola funkčnosti webu a případné opravy

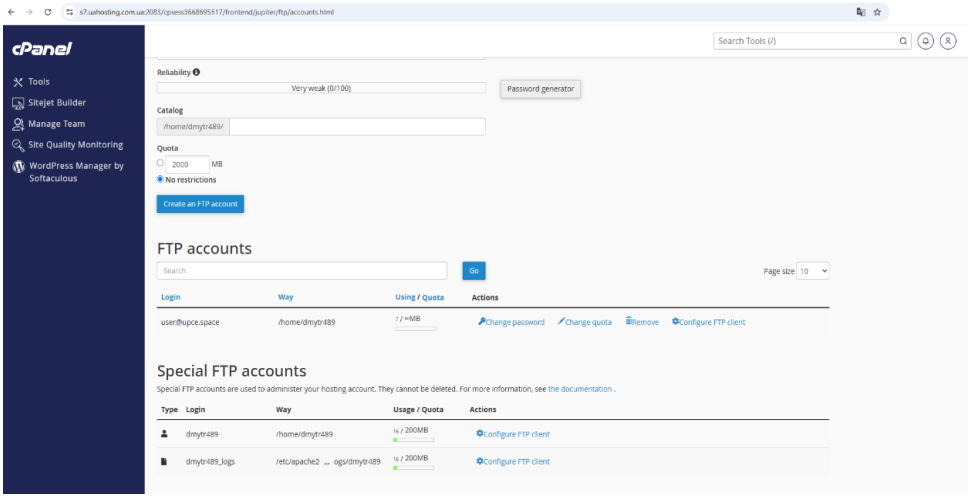
Vytvoření FTP účtu v cPanelu



Obrázek 37: Formulář pro vytvoření účtu, zdroj: vlastní zpracování

Postup vytvoření FTP účtu probíhal následovně:

- Přihlášení do administrace cPanel
- Otevření sekce „FTP Accounts“ (nachází se v bloku Files)
- Vyplnění formuláře pro vytvoření nového účtu:
    - Login – uživatelské jméno FTP účtu
    - Domain – výběr domény, pod kterou má účet fungovat
    - Password – silné heslo pro přístup
    - Directory – výchozí složka, ke které bude mít účet přístup (např. public_html/ftpuser, s možností úpravy)

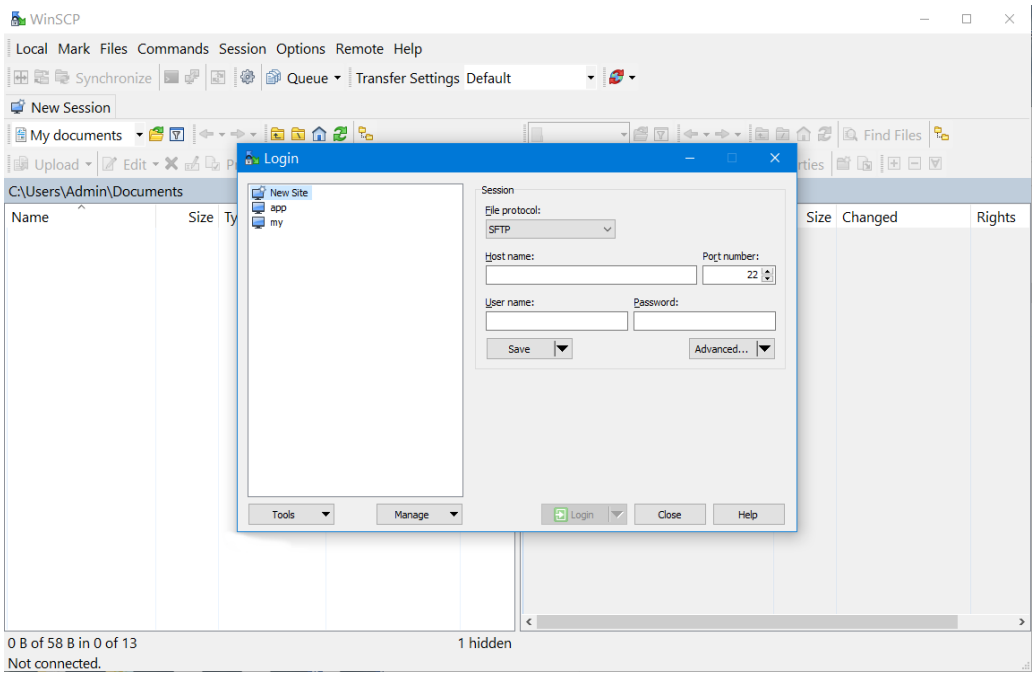


Obrázek 38: Seznam účtů, zdroj: vlastní zpracování

Po potvrzení tlačítkem „Create FTP Account“ byl účet vytvořen a následně zobrazen v dolní části stránky v seznamu existujících FTP účtů.

Pomocí takto vytvořeného účtu bylo možné se pohodlně připojit k serveru prostřednictvím programu WinSCP a nahrát celý projekt.

Nastavení FTP připojení pomocí WinSCP



Obrázek 39: Nastavení přístupu k ftp, zdroj: vlastní zpracování

Po vytvoření FTP účtu v rozhraní cPanel bylo možné navázat připojení k serveru prostřednictvím aplikace WinSCP, která nabízí přehledné grafické rozhraní pro správu souborů na vzdáleném hostingu.

Postup konfigurace:

- Spuštění programu WinSCP

  Po otevření aplikace bylo kliknuto na tlačítko „New Site“, čímž se vytvořila nová relace pro připojení.

- Volba protokolu

  V poli „File protocol“ byla vybrána možnost FTP. Pro bezpečnější přenos dat byla zvolena varianta FTP with TLS/SSL Explicit encryption, která zajišťuje šifrovanou komunikaci mezi klientem a serverem [35].

- Vyplnění přístupových údajů

  Do příslušných polí byly zadány následující údaje:

- Host name – doména nebo IP adresa serveru (např. s7.uahosting.com.ua)
- Port number – ponechána výchozí hodnota 21
- User name – e-mailová adresa nebo název FTP účtu vytvořeného v cPanelu (např. user@upce.space)
- Password – heslo zadané při vytvoření účtu

Připojení k serveru

Po zadání všech údajů bylo kliknuto na tlačítko „Login“. V případě správné konfigurace se zobrazila adresářová struktura na serveru, což znamenalo úspěšné navázání spojení.

Uložení relace ve WinSCP a nahrání webových souborů

Aby se předešlo opakovanému zadávání přihlašovacích údajů, byla relace ve WinSCP uložena pomocí tlačítka „Save“, přičemž jí byl přiřazen název a zaškrtnuta možnost „Save password“, která zajistila uložení hesla. Následně bylo možné relaci buď ihned použít kliknutím na „Login“, nebo ji uložit pro pozdější připojení.

Při prvním připojení zobrazil systém bezpečnostní upozornění týkající se certifikátu, které bylo potvrzeno s cílem důvěřovat serveru. Po úspěšném přihlášení se otevřelo hlavní rozhraní aplikace WinSCP, kde:

- levá část zobrazuje soubory v místním počítači,
- pravá část zobrazuje adresářovou strukturu na serveru.

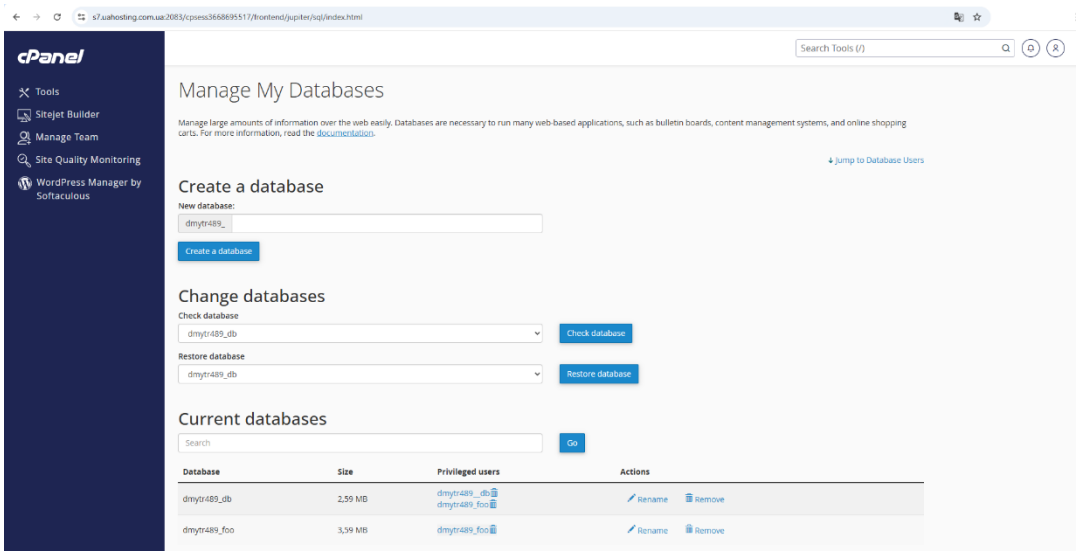
V tomto rozhraní bylo možné začít pracovat se soubory webu – nahrávat je, upravovat, mazat nebo organizovat složky.

Nahrání webových souborů přes WinSCP – postup:

- Po připojení k FTP:
    - levá část – místní počítač
    - pravá část – soubory na serveru
- V pravé části byla otevřena složka public_html, která slouží jako kořenová složka webu
- Soubory webové aplikace byly přetaženy z levé strany (PC) do pravé (server)
- Během přenosu aplikace WinSCP zobrazila průběh a stav nahrávání

- Po dokončení byly soubory připraveny k použití na serveru

Vytvoření databáze v cPanelu



Obrázek 40: Stránka práce s databází, zdroj: vlastní zpracování

Pro vytvoření nové databáze byla v cPanelu otevřena sekce „MySQL Databases“. Do pole „Create New Database“ byl zadán název databáze (například site_db) a následně bylo kliknuto na tlačítko „Create Database“.

Systém automaticky přidal předponu odpovídající uživatelskému jménu v cPanelu (například dmytr489_), takže výsledná databáze nesla název ve formátu dmytr489_site_db.

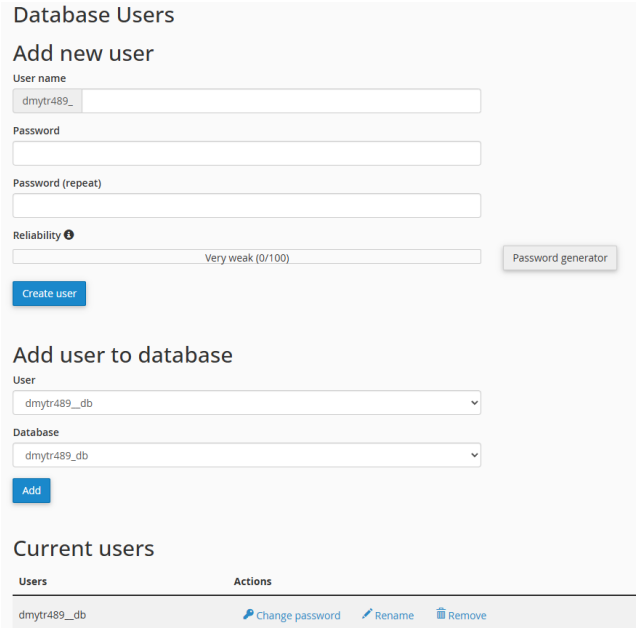
Po vytvoření databáze bylo kliknuto na „Go Back“, čímž se uživatel vrátil zpět na stránku s dalším nastavením.

Dalším krokem bylo vytvoření samostatného uživatele databáze a přiřazení přístupu. Tento úkon byl proveden na stejné stránce v sekci „Add New User“.

Do příslušných polí bylo zadáno:

- uživatelské jméno (například dbuser),
- heslo, které bylo buď zadáno ručně, nebo vygenerováno automaticky. V případě automatického generování bylo heslo zkopírováno a bezpečně uloženo.

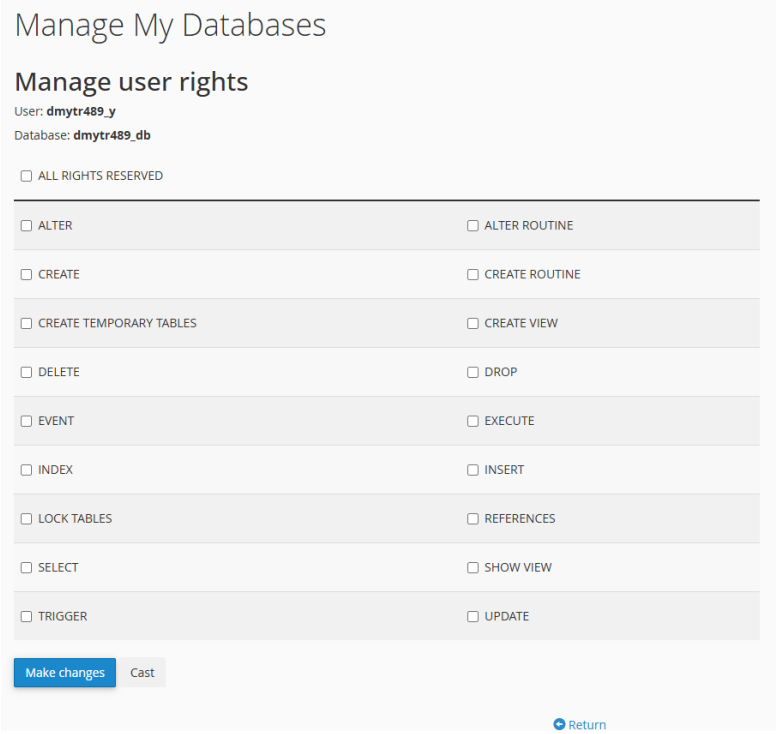
Po vyplnění všech údajů bylo kliknuto na tlačítko „Create User“, čímž došlo k vytvoření databázového uživatele.



Obrázek 41: Uživatel databáze, zdroj: vlastní zpracování

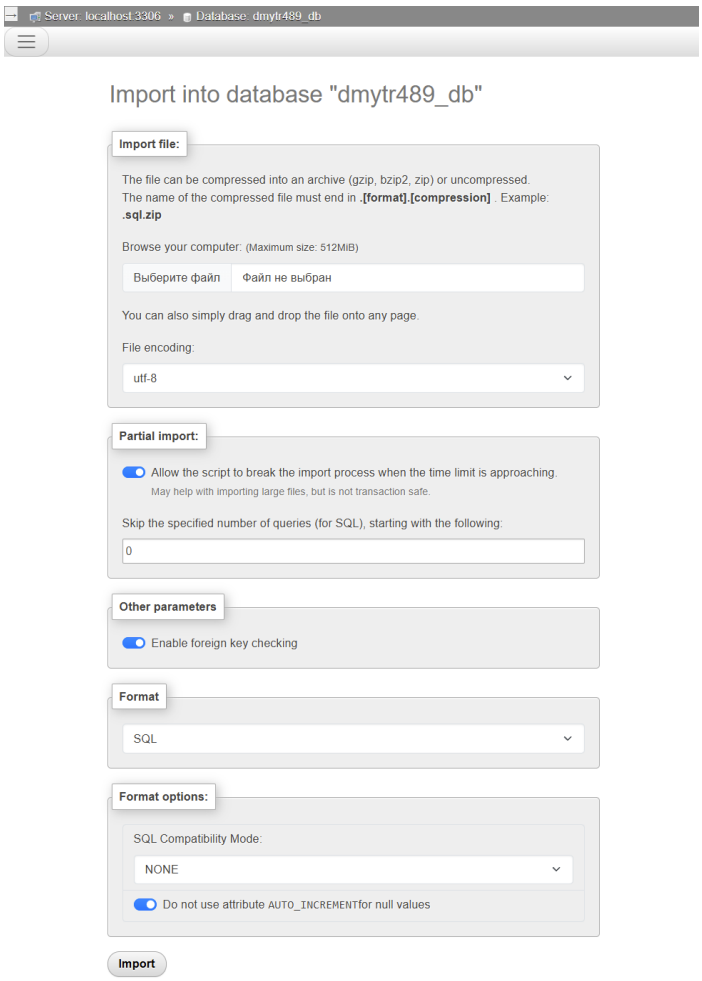
Pro přidělení přístupu nově vytvořenému uživateli bylo na stejné stránce v sekci „Add User to Database“ vybráno konkrétní jméno uživatele a databáze. Následně bylo kliknuto na tlačítko „Add“.

Po tomto kroku se otevřela stránka pro nastavení oprávnění. Ve většině případů bylo dostačující zaškrtnout možnost „ALL PRIVILEGES“, která zajišťuje plný přístup k databázi.



Obrázek 42: Udělení práv uživateli databáze, zdroj: vlastní zpracování

Změny byly potvrzeny kliknutím na tlačítko „Make Changes“. Tímto způsobem byl uživateli úspěšně udělen plný přístup a tato část nastavení byla tímto dokončena.



Obrázek 43: Formulář pro import databáze, zdroj: vlastní zpracování

Po dokončení vytvoření databáze a přiřazení uživatele bylo možné přejít k importu její struktury nebo obsahu. K tomu slouží nástroj phpMyAdmin, který je dostupný přímo z hlavní stránky cPanelu.

V levé části rozhraní phpMyAdmin byla vyhledána a kliknutím vybrána právě vytvořená databáze, čímž došlo k jejímu otevření. Následně byla zvolena záložka „Import“, kde se nachází rozhraní pro nahrání souboru s databázovým obsahem.

Pomocí tlačítka „Browse“ byl vybrán soubor databáze ve formátu .sql nebo .sql.zip. Většina importních parametrů zůstala ve výchozím nastavení – formát SQL a kódování utf-8.

Proces importu byl spuštěn kliknutím na tlačítko „Import“ ve spodní části stránky. Po jeho dokončení byla databáze připravena k použití aplikací.

Po dokončení importu se v phpMyAdmin zobrazilo hlášení „Import has been successfully finished“ a v levém panelu se objevily tabulky obsažené v databázi. To znamená, že import proběhl úspěšně a databáze je připravena k použití.

Posledním krokem bylo ověření dostupnosti webové stránky a samotné aplikace. Pokud byla všechna nastavení provedena správně, aplikace se po zadání domény načetla bez chyb a veškerá funkcionalita byla dostupná v plném rozsahu.

ZÁVĚR

V rámci této bakalářské práce byla navržena a vytvořena mobilní aplikace zaměřená na jednoduché a intuitivní vyhledávání a správu událostí pro uživatele. Aplikace byla realizována pomocí moderní multiplatformní technologie Apache Cordova, která umožňuje použití jednotného kódu pro platformy Android a iOS, což vedlo k výraznému zkrácení času na vývoj a usnadnění budoucí údržby aplikace.

Přínosem této aplikace je především snadná dostupnost informací o aktuálních událostech, rychlá navigace mezi kategoriemi a možnost filtrování dle města a data. Uživatelé mají díky jednoduchému uživatelskému rozhraní okamžitý přístup k podrobným informacím o akcích, včetně lokalizace místa konání a odkazu na zakoupení vstupenek.

Další výhodou je použití běžných webových technologií (HTML, CSS, JavaScript), které umožnily vytvořit responzivní design vhodný pro široké spektrum zařízení. Použitá architektura MVC na serverové části s využitím PHP a databáze MySQL přinesla flexibilní řešení, schopné jednoduše spravovat obsah a uživatele, a zároveň zajišťovat vysokou úroveň zabezpečení a výkonu.

Během procesu vývoje byly provedeny rozsáhlé testy, včetně uživatelského testování, unit testů a základních bezpečnostních kontrol. Výsledky těchto testů potvrdily stabilitu aplikace a připravenost k reálnému nasazení.

V budoucnu je plánováno rozšíření funkcionality aplikace, například implementace pokročilých notifikací o blížících se událostech či integrace s externími kalendáři a sociálními sítěmi, což dále zvýší uživatelský komfort a praktickou využitelnost.

Celkově tato bakalářská práce dokazuje vhodnost použitých technologií pro rychlý a efektivní vývoj multiplatformních mobilních aplikací, které splňují moderní požadavky uživatelů na snadnou obsluhu, dostupnost informací a vysokou výkonnost.

POUŽITÁ LITERATURA

[1] GOOGLE. *Google Calendar Help*. Google [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://support.google.com/calendar

[2] EVENTBRITE. *Eventbrite Help Center*. Eventbrite [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://www.eventbrite.com/support

[3] META PLATFORMS. *Facebook Help Center — Events*. Meta [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://www.facebook.com/help/152652248136178

[4] SIMPLE MOBILE TOOLS. *Simple Calendar App — Official Website*. Simple Mobile Tools [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://simplecalendar.io

[5] APACHE SOFTWARE FOUNDATION. *Apache Cordova Docs — Overview.* Apache Software Foundation [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/overview/index.html

[6] APACHE SOFTWARE FOUNDATION. *Apache Cordova — Plugin Development Guide*. Apache Software Foundation [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://cordova.apache.org/docs/en/latest/guide/hybrid/plugins/

[7] GOOGLE. *Android Developers — WebView.* Google [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://developer.android.com/reference/android/webkit/WebView

[8] WARGO, John M. *Apache Cordova API Cookbook*. Sebastopol: O'Reilly Media, 2015. ISBN 978-1491923812.

[9] MOZILLA FOUNDATION. *JavaScript Promises*. MDN Web Docs [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://developer.mozilla.org/docs/Web/JavaScript/Reference/Global_Objects/Promise

[10] MOZILLA FOUNDATION. *Responsive design — basics*. MDN Web Docs [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://developer.mozilla.org/docs/Learn/CSS/CSS_layout/Responsive_Design

[11] DUCKETT, Jon. *HTML and CSS: Design and Build Websites*. Indianapolis: Wiley, 2011. ISBN 978-1118008188.

[12] FLANAGAN, David. *JavaScript: The Definitive Guide*. 7. vyd. Sebastopol: O'Reilly Media, 2020. ISBN 978-1491952027.

[13] ORACLE. *MySQL 8.0 Reference Manual*. Oracle [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/

[14] REDBEANPHP. *RedBeanPHP ORM — Documentation*. RedBeanPHP [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://redbeanphp.com/index.php?p=/welcome

[15] ELMASRI, Ramez, a Shamkant B. NAVATHE. *Fundamentals of Database Systems*. 7. vyd. Harlow: Pearson, 2016. ISBN 978-0133970777.

[16] CONNOLLY, Thomas, a Carolyn BEGG. *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6. vyd. Harlow: Pearson, 2015. ISBN 978-1292061184.

[17] BATINI, Carlo, Stefano CERI a Shamkant NAVATHE. *Conceptual Database Design: An Entity-relationship Approach*. Redwood City: Benjamin/Cummings, 1992. ISBN 978-0805303984.

[18] TEOREY, Toby J. *Database Modeling and Design*. 5. vyd. Burlington: Morgan Kaufmann, 2011. ISBN 978-0123820204.

[19] FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures (REST)*. Dissertation. University of California, Irvine, 2000.

[20] OWASP FOUNDATION. *OWASP Top Ten Security Risks*. OWASP [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://owasp.org/www-project-top-ten/

[21] MOZILLA FOUNDATION. *Using the Fetch API*. MDN Web Docs [online]. © 2025 [cit. 2025-08-18]. Dostupné z: https://developer.mozilla.org/docs/Web/API/Fetch_API