UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2025 Bc. Lukáš Bajer

Univerzita Pardubice
Fakulta elektroniky a informatiky

Mobilní aplikace pro události pořádané v okolí

Diplomová práce

2025 Bc. Lukáš Bajer

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

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Lukáš Bajer**
Osobní číslo:	**I23351**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Mobilní aplikace pro události pořádané v okolí**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem práce je tvorba mobilní aplikace pro zařízení Android, která umožňuje informovat uživatele o událostech v jejich okolí, které pořádají vybrané podniky. Mobilní aplikace bude pracovat s různými API jako jsou mapové podklady tak, aby bylo možné uživatele informovat dle jeho aktuální polohy a také denním reportem formou notifikací o jednotlivých událostech. Aplikace bude využívat databázi PostgreSQL a bude splňovat požadavky kladené na uživatelskou přívětivost a moderní design mobilní aplikace. Součástí řešení bude i analýza stávajících nebo obdobných řešení a jejich komparace, podcílem práce je rešerše v oblasti vývoje moderních trendů mobilních aplikací.

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

Seznam doporučené literatury:

PRASUN, Barua. *Android App Development: From Concept to Code*. Amazon Digital Services LLC – Kdp, 2023. ISBN 9798860554177.
BAILEY, Noah. *Mobile App Development: Mobile App Development 101: A Step-by-Step Guide for Beginners*. Noah Bailey, 2023. ISBN 9798869080035.

Vedoucí diplomové práce: **Ing. Monika Borkovcová, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2024**
Termín odevzdání diplomové práce: **23. května 2025**

L.S.

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

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

V Pardubicích dne 29. listopadu 2024

Prohlašuji:

Práci s názvem Mobilní aplikace pro události pořádané v okolí 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 29. 08. 2025

Bc. Lukáš Bajer v.r.

ANOTACE

Diplomová práce „Mobilní aplikace pro události v okolí“ se zaměřuje na vývoj mobilní aplikace pro zařízení se systémem Android, která uživatelům zprostředkovává informace o kulturních a společenských událostech v jejich blízkosti. Aplikace umožňuje vyhledávání a filtrování událostí podle data, času, kategorie či názvu, jejich tvorbu, úpravu a správu, a zároveň nabízí zobrazení na mapě. Součástí řešení je také systém denních notifikací, který uživatele pravidelně upozorňuje na nadcházející akce. Teoretická část práce se věnuje rešerši současných trendů ve vývoji mobilních aplikací a analýze existujících řešení, zatímco praktická část se zaměřuje na použité technologie, návrh a implementaci výsledné aplikace.

KLÍČOVÁ SLOVA

Mobilní aplikace, Android, Flutter, API, Spring Boot, PostgreSQL, PostGIS, kulturní události, mapová vizualizace

TITLE

Mobile Application for Nearby Events

ANNOTATION

The thesis “Mobile Application for Nearby Events” focuses on the development of a mobile application for Android devices that provides users with information about cultural and social events in their vicinity. The application allows searching and filtering of events by date, time, category, or name, as well as their creation, editing, and management, while also offering map visualization. The solution includes a daily notification system that regularly informs users about upcoming events. The theoretical part of the thesis deals with current trends in mobile application development and the analysis of existing solutions, while the practical part focuses on the applied technologies, design, and implementation of the final application.

KEYWORDS

Mobile application, Android, Flutter, API, Spring Boot, PostgreSQL, PostGIS, cultural events, map visualization

OBSAH

SEZNAM ILUSTRACÍ A TABULEK 11
SEZNAM ZKRATEK 13
ÚVOD 14
1 REŠERŠE SOUČASNÝCH TRENDŮ VE VÝVOJI MOBILNÍCH APLIKACÍ 15
1.1 Technologické trendy 15
1.1.1 Multiplatformní vývoj 15
1.1.2 Backend as a service 18
1.1.3 Integrace externích API 18
1.2 UX/UI trendy v mobilních aplikacích 19
1.2.1 Funkční design, UI/UX, Responzibilita 19
1.2.2 Tmavý a světlý režim 20
1.2.3 Web Content Accessibility Guidelines 20
1.3 Bezpečnost a ochrana soukromí 21
1.3.1 Šifrování dat 21
1.3.2 Autentizace a autorizace uživatelů 22
1.3.3 Práce s osobními údaji a oprávněními 22
1.3.4 Distribuce softwarových produktů 23
1.4 Umělá inteligence a její využití v mobilních aplikacích 23
1.5 Inovativní přístupy ve vývoji aplikací 24
1.5.1 Spolupráce v rámci ekosystémů 24
1.5.2 Rozšířená a virtuální realita v mobilních zařízeních 25
1.6 Průzkum klíčových pojmů rešerše a implikace pro vlastí řešení 25
1.6.1 Průzkum a volba frontendového frameworku pro implementaci aplikace 27
1.6.2 Průzkum a výběr externích API pro implementaci v aplikaci 29
1.6.3 Výběr způsobu implementace backendu 32
2 ANALÝZA OBDOBNÝCH ŘEŠENÍ 33
2.1 Facebook Events 33

2.1.1 Cílová skupina a účel aplikace....33
2.1.2 Hlavní funkce....34
2.1.3 Uživatelské rozhraní a UX....35
2.1.4 Uživatelská hodnocení....37
2.1.5 SWOT analýza....37
2.1.6 Zhodnocení....37
2.2 Presentie....38
2.2.1 Cílová skupina a účel aplikace....38
2.2.2 Hlavní funkce....38
2.2.3 Uživatelské rozhraní a UX....39
2.2.4 Uživatelská hodnocení....41
2.2.5 SWOT Analýza....42
2.2.6 Zhodnocení....42
2.3 Liiiv....42
2.3.1 Cílová skupina a účel aplikace....43
2.3.2 Hlavní funkce....43
2.3.3 Uživatelské rozhraní a UX....44
2.3.4 SWOT analýza....45
2.3.5 Hodnocení uživatelů....45
2.3.6 Zhodnocení....46
3 PRAKTICKÁ ČÁST....47
3.1 Použité technologie a software....47
3.1.1 Vývojové prostředí Visual Studio Code....47
3.1.2 Vývojové prostředí IntelliJ Idea....48
3.1.3 Testovací nástroj Postman....49
3.1.4 Docker....49
3.1.5 Spring framework....50

3.1.6 Java 51
3.1.7 PostgreSQL 51
3.1.8 Flyway 52
3.1.9 Flutter 52
3.1.10 Programovací jazyk Dart 53
3.2 Funkční a nefunkční požadavky 54
3.3 Databázový návrh 54
3.3.1 Základní principy návrhu 55
3.3.2 Databázový model 56
3.3.3 Popis entit a vazeb 57
3.4 Implementace backendu 60
3.4.1 Použité knihovny pro Spring Boot 61
3.4.2 Architektura backendu 61
3.4.3 REST API 62
3.4.4 Testování 65
3.5 Implementace frontendu 69
3.5.1 Architektura aplikace 69
3.5.2 Uživatelské rozhraní aplikace 70
3.6 Práce s API třetích stran 76
3.6.1 Firebase Cloud Messaging 76
3.6.2 Mapbox Search Box API 77
3.6.3 Cloudflare Images API 78
3.6.4 ARES API 78
3.6.5 Brevo SMTP 79
3.7 Denní notifikace 79
3.8 Zabezpečení, správa oprávnění a ochrana osobních údajů 80
3.8.1 Validace uživatelských vstupů a zpracování výjimek 80

3.8.2 Authentizace a správa JWT tokenů 81
3.8.3 Ochrana osobních údajů 81
3.8.4 Požadovaná oprávnění aplikace a způsob jejich získávání 82
3.8.5 Zabezpečení databázové vrstvy 83
3.8.6 Omezení přístupu k API a povolené endpointy 83
3.8.7 Instalace z třetích stran 83
3.8.8 Práce s API klíči 83
3.8.9 Uživatelské role a oprávnění 84
ZÁVĚR 85
POUŽITÁ LITERATURA 86

SEZNAM ILUSTRACÍ A TABULEK

Obrázek 1 - Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2023. zdroj: (45) 27
Obrázek 2 - Webové uživatelské rozhraní aplikace Facebook Events. zdroj: (43) 35
Obrázek 3 - Mobilní uživatelské rozhraní aplikace Facebook Events. zdroj: (43) 36
Obrázek 4 - Webové uživatelské rozhraní aplikace Presentie, zdroj: (46) 40
Obrázek 5 - Mobilní uživatelské rozhraní aplikace Presentie, zdroj: (47) 41
Obrázek 6 - Mobilní uživatelské rozhraní aplikace Liiiv. zdroj: vlastní 45
Obrázek 7 - Unit testy AccountService. zdroj: vlastní 65
Obrázek 8 - Unit testy AuthService. zdroj: vlastní 66
Obrázek 9 - Unit testy LocationService. zdroj: vlastní 66
Obrázek 10 - Unit testy EventService. zdroj: vlastní 67
Obrázek 11 - Integrační testy EventRepository. zdroj: vlastní 69
Obrázek 12 - Obrazovky přihlášení, obnova hesla, registrace. zdroj: vlastní 71
Obrázek 13 - Obrazovky event feed, filtry: datum, lokace, kategorie. zdroj: vlastní 72
Obrázek 14 - Obrazovka mapy. zdroj: vlastní 73
Obrázek 15 - Obrazovky karta události a detail události. zdroj: vlastní 74
Obrázek 16 - Obrazovky menu, úprava oblíbených kategorií, úprava účtu. zdroj: vlastní 75
Obrázek 17 - Obrazovka editor událostí. zdroj: vlastní 75
Obrázek 18 - Obrazovky moje události, editace události. zdroj: vlastní 76
Obrázek 19 - Vzhled emailu pro obnovu hesla. zdroj: vlastní 79
Obrázek 20 - Denní notifikace. zdroj: vlastní 80
Obrázek 21 - Validace uživatelských vstupů a zpracování výjimek. zdroj: vlastní 81

Tabulka 1 - Porovnání multiplatformních frameworků. zdroj: (62; 63) 29
Tabulka 2 - Porovnání notifikačních služeb. zdroj: (64; 65) 30
Tabulka 3 - Porovnání mapových služeb. zdroj: (66; 67; 68) 31
Tabulka 4 - Porovnání služeb pro ukládání obrázků. Zdroj: (69; 70) 31
Tabulka 5 - SWOT analýza aplikace Facebook Events. zdroj: vlastní 37
Tabulka 6 - SWOT analýza aplikace Presentie. zdroj: vlastní 42
Tabulka 7 - SWOT analýza aplikace Liiiv. zdroj: vlastní 45
Tabulka 8 - Relační model. zdroj: vlastní 56
Tabulka 9 - Popis tabulky Account. zdroj: vlastní 57

Tabulka 10 - Popis tabulky Category. zdroj: vlastní 58
Tabulka 11 - Popis tabulky Currency. zdroj: vlastní 58
Tabulka 12 - Popis tabulky Device_token. zdroj: vlastní.....59
Tabulka 14 - Popis tabulky Location. zdroj: vlastní.....59
Tabulka 13 - Popis tabulky Event. zdroj: vlastní.....60
Tabulka 15 - Popis controlleru Account controller. zdroj: vlastní 62
Tabulka 16 - Popis controlleru Auth controller. zdroj: vlastní.....63
Tabulka 17 - Popis controllerů Category a Currency controller. zdroj: vlastní.....63
Tabulka 18 - Popis controleru device_token controller. zdroj: vlastní.....63
Tabulka 20 - Popis controlleru Location controller. zdroj: vlastní.....63
Tabulka 19 - Popis controlleru Event controller. zdroj: vlastní.....64
Tabulka 21 - Popis controlleru Mapbox controller. zdroj: vlastní.....64
Tabulka 22 - Popis controlleru CloudflareImages controller. zdroj: vlastní 64
Tabulka 23 - Uložené osobní údaje uživatele. zdroj: vlastní.....82

SEZNAM ZKRATEK

UX	User Experience, Uživatelská zkušenost
UI	User Interface, Uživatelské rozhraní
WCAG	Web Content Accessibility Guidelines
OLED	Organic Light-Emitting Diode
LCD	Liquid Crystal Display
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
POUR	Perceivable, Operable, Undersandable, Robust
GDPR	General Data Protection regulation
HTTPS	Hypertext Transfer Protocol Secure
TLS	Transport Layer Security
AI	Artificial Inteligence
NLP	Natural Langugage Processing
AR	Augmented Reality
VR	Virtual Reality
BaaS	Backend as a Service
KMP	Kotlin Multiplatform
PWA	Progressive Web App
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
API	Application Programming Inteface
REST	Representational State Transfer

ÚVOD

V posledních letech se mobilní zařízení, zejména chytré telefony a tablety, staly neoddělitelnou součástí každodenního života. Jejich rozšíření zásadně ovlivnilo způsoby, jakými lidé komunikují, vyhledávají informace či tráví volný čas. Mobilní aplikace se tak staly univerzálním prostředkem, který propojuje uživatele s okolním světem a nabízí nové možnosti v oblasti vzdělávání, práce i kulturních či společenských aktivit.

Významnou roli mohou mobilní technologie sehrát také při objevování kulturních a komunitních událostí. Informace o těchto akcích jsou často roztříštěné napříč různými platformami, což snižuje jejich dostupnost a přehlednost. Specializovaná mobilní aplikace dokáže tento problém překlenout a nabídnout uživatelům komfortní a rychlý přístup k aktuálnímu dění v jejich okolí.

Z pohledu podnikatelů a organizátorů událostí představuje taková aplikace nástroj pro efektivní propagaci aktivit a potenciální zvýšení návštěvnosti. Díky systému notifikací a možnosti označovat kategorie jako oblíbené se uživatelé snadněji dostanou k událostem, které odpovídají jejich zájmům. Tím se zvyšuje pravděpodobnost, že si jich všimnou právě ti, pro které jsou určeny. V širším společenském kontextu může aplikace přispět ke zviditelnění lokálních iniciativ, podpořit kulturní rozmanitost a posílit komunitní život ve městech a obcích.

Tato diplomová práce se proto zaměřuje na oblast lokálních událostí a jejich propojení s moderními mobilními technologiemi. Jejím cílem je zmapovat současné trendy ve vývoji mobilních aplikací, porovnat existující řešení dostupná na českém trhu a na základě těchto poznatků navrhnout a implementovat vlastní aplikaci. Ta bude uživatelům poskytovat přehledný přístup k informacím o kulturních a společenských akcích a zároveň podpoří podniky a prostory, jako jsou kavárny, bary, kulturní centra či galerie, které hrají významnou roli v utváření komunitního života.

1 REŠERŠE SOUČASNÝCH TRENDŮ VE VÝVOJI MOBILNÍCH APLIKACÍ

Význam mobilních aplikací v současné společnosti neustále roste, jelikož se staly nedílnou součástí každodenního života jednotlivců i klíčovým nástrojem firem pro komunikaci se zákazníky. S tím souvisí i dominance dvou hlavních mobilních platforem, iOS a Android, které určují směr vývoje, nastavují technické standardy a ovlivňují volbu vývojových nástrojů. V posledních letech lze pozorovat posun od tradičního nativního vývoje k multiplatformním řešením, jako jsou například Flutter nebo React Native, které umožňují rychlejší a efektivnější vývoj pro obě platformy současně. Rostoucí očekávání uživatelů zároveň klade větší důraz na kvalitní UX/UI a personalizaci obsahu, což se stává jedním z klíčových faktorů úspěchu aplikací. S tím však přichází i zvýšené požadavky na ochranu osobních údajů a bezpečnost celého ekosystému, které nelze v moderním vývoji opomíjet. Významnou roli začíná hrát také integrace umělé inteligence, stejně jako inovativní přístupy typu integrace rozšířené a virtuální reality, které přinášejí nové možnosti interakce a posouvají hranice uživatelského zážitku.

1.1 Technologické trendy

V oblasti technologických trendů vývoje mobilních aplikací hrají klíčovou roli přístupy a nástroje, které zefektivňují vývoj a zkracují čas potřebný k uvedení produktu na trh. Mezi nejvýraznější patří multiplatformní vývoj, který umožňuje vytvářet aplikace pro více platforem z jedné kódové základny, a tím šetří náklady i zdroje. Stále větší význam má také využití konceptu Backend-as-a-Service (BaaS), jenž vývojářům poskytuje hotovou infrastrukturu pro správu dat, autentizaci či notifikace. Nedílnou součástí moderního vývoje je i integrace externích API, která rozšiřují funkčnost aplikací a umožňují jejich napojení na různé služby třetích stran.

1.1.1 Multiplatformní vývoj

Multiplatformní vývoj mobilních aplikací představuje přístup, jehož cílem je vytvořit jednu společnou kódovou základnu, kterou lze nasadit na více operačních systémů, typicky Android a iOS, ale nástroje začínají podporovat i zavedené operační systémy jako jsou Windows a Linux, nebo webové prostředí. Tento koncept reaguje na rostoucí poptávku po rychlejším, levnějším a udržitelnějším vývoji, který zároveň zachová vysokou kvalitu uživatelského rozhraní i výkon aplikace. Tradiční nativní vývoj, kdy se pro každou platformu vytváří samostatná aplikace, je sice optimalizovaný pro konkrétní prostředí, ale často znamená vyšší náklady a složitější údržbu. Multiplatformní přístupy naproti tomu umožňují vývojářům

sdílet značnou část logiky a uživatelského rozhraní, čímž zefektivňují celý vývojový cyklus. V následujících podkapitolách budou podrobněji rozebrány nejvýznamnější nástroje a technologie, které tento přístup dnes umožňují neboli Flutter, React Native, Kotlin Multiplatform a Progressive Web Apps (1; 2).

React Native je open-source framework vyvinutý společností Meta, který umožňuje vývoj nativně působících multiplatformních aplikací pomocí programovacího jazyka JavaScript a knihovny React. Jeho hlavním přínosem je schopnost sdílet velkou část kódu mezi platformami iOS a Android, zatímco výsledná aplikace vykresluje skutečné nativní komponenty dané platformy, nikoliv webové prvky. React Native využívá architekturu založenou na tzv. „bridge“ vrstvě, která zprostředkovává komunikaci mezi JavaScriptem a nativním kódem zařízení. Tento přístup umožňuje flexibilní kombinaci společného a platformě specifického kódu podle potřeby (3).

Díky své architektuře je React Native vhodný zejména pro projekty, kde je důležité zachovat nativní vzhled a odezvu aplikace, ale zároveň optimalizovat náklady a čas vývoje. Framework podporuje i integraci s nativními moduly napsanými v Objective-C, Swift nebo Java/Kotlin, což umožňuje rozšíření funkcionality mimo rámec samotného JavaScriptu. Mezi klíčové výhody patří také aktivní vývojářská komunita, bohatý ekosystém knihoven a možnost využití nástrojů jako Expo, které zjednodušují nasazení a testování. React Native tak představuje pragmatický kompromis mezi plně nativním vývojem a plně sdíleným multiplatformním přístupem (4).

Flutter je open-source framework vyvinutý společností Google, který umožňuje tvorbu multiplatformních aplikací z jedné společné kódové základny, přičemž podporuje nejen Android a iOS, ale i webové aplikace a desktopové prostředí. Flutter využívá programovací jazyk Dart a je založen na vlastním renderovacím jádru (Skia), díky čemuž není závislý na nativních komponentách operačního systému. Místo toho vykresluje uživatelské rozhraní sám, což zajišťuje konzistentní vzhled a chování napříč platformami.

Jedním z klíčových rysů Flutteru je tzv. widget-based architecture, kde je vše, od tlačítek po celé obrazovky, reprezentováno pomocí widgetů. Tento přístup umožňuje vývojářům vytvářet flexibilní a vysoce přizpůsobitelná uživatelská rozhraní. Díky podpoře nástrojů jako hot reload je možné okamžitě reflektovat změny v kódu, což významně zrychluje vývoj a ladění aplikací. Flutter navíc umožňuje přímou integraci s platformově specifickým kódem (např. v Javě nebo Swiftu) přes tzv. platform channels, čímž vývojáři získávají přístup k funkcím, které nejsou standardně součástí frameworku. Výsledkem je robustní nástroj pro vývoj moderních, výkonných a vizuálně konzistentních aplikací napříč platformami (5; 6).

Kotlin Multiplatform (KMP) je moderní přístup k vývoji multiplatformních aplikací vyvíjený společností JetBrains, který umožňuje sdílení aplikační logiky mezi různými platformami jako jsou Androidu, iOS, desktop nebo web, přičemž je zachována plná flexibilita pro tvorbu nativních uživatelských rozhraní. Na rozdíl od frameworků jako Flutter nebo React Native, které se snaží sjednotit vývoj i na úrovni UI, Kotlin Multiplatform se zaměřuje především na sdílení tzv. business logic, jako jsou datové modely, síťová komunikace, databáze nebo zpracování dat.

KMP využívá programovací jazyk Kotlin, který je oficiálně podporovaný Googlem pro vývoj Android aplikací, a díky podpoře Kotlin Native je možné překládat sdílený kód i pro iOS bez potřeby virtuálního stroje. Vývojáři tak mohou psát aplikační logiku jednou a následně ji znovu využít na více platformách, přičemž uživatelské rozhraní a interakce zůstávají plně nativní. Tento přístup respektuje rozdílnost platforem a umožňuje zachovat vysoký výkon i uživatelský komfort. Významnou výhodou KMP je také snadná integrace do stávajících projektů, kde není nutné aplikaci přepisovat od základu, ale je možné multiplatformní architekturu zavádět postupně. Kotlin Multiplatform tak představuje flexibilní a výkonné řešení, které propojuje výhody sdíleného kódu s možnostmi nativního vývoje (2; 7).

Progressive Web Apps (PWA) představují moderní přístup k vývoji webových aplikací, které díky pokročilým technologiím poskytují uživatelský zážitek srovnatelný s nativními mobilními aplikacemi. PWAs jsou webové aplikace, které využívají standardy jako Service Workers, Web App Manifest a HTTPS, čímž umožňují funkce jako offline režim, push notifikace, instalaci na domovskou obrazovku či rychlé načítání bez ohledu na kvalitu připojení. Hlavním cílem tohoto přístupu je sjednocení výhod webových technologií s nativními funkcemi zařízení, bez nutnosti distribučního kanálu typu App Store nebo Google Play.

Vývojářům PWA umožňují psát aplikaci v jazycích jako HTML, CSS a JavaScript, přičemž výsledná aplikace běží ve webovém prohlížeči, ale chová se jako běžná mobilní aplikace, která může být spuštěna na celou obrazovku, funguje offline a zachovává rychlou odezvu. Výhodou je také multiplatformnost bez potřeby duplikovat kód pro různé operační systémy. PWA jsou oblíbeným řešením zejména pro firmy, které chtějí minimalizovat náklady na vývoj a zároveň oslovit široké publikum. Úspěšné nasazení PWAs lze vidět například u společností jako Twitter, Pinterest nebo Starbucks, kde tyto aplikace dosahují výkonnostních i uživatelských metrik srovnatelných s nativními aplikacemi (8; 9).

1.1.2 Backend as a service

Backend as a Service (BaaS) představuje moderní přístup k vývoji mobilních aplikací, který vývojářům umožňuje soustředit se primárně na vývoj klientské části aplikace, zatímco backendová logika a infrastruktura jsou zajištěny třetí stranou. BaaS poskytuje hotové služby jako autentizace uživatelů, správa databáze, cloudové funkce, notifikace nebo file storage skrze jednotné API. Díky tomu odpadá potřeba budovat vlastní serverovou infrastrukturu, což výrazně zkracuje čas vývoje a snižuje náklady (7).

Mezi nejznámější zástupce BaaS platforem patří Firebase od společnosti Google a Supabase, open-source alternativa postavená na PostgreSQL. Firebase nabízí komplexní balík nástrojů včetně real-time databáze, cloudového úložiště, autentizace uživatelů a analytiky (8). Supabase se profiluje jako vývojářsky přívětivá alternativa, která poskytuje plně spravovanou PostgreSQL databázi s automaticky generovaným REST a GraphQL API, podporou real-time událostí a integrovanou autentizací (9).

1.1.3 Integrace externích API

Integrace externích rozhraní API (Application Programming Interfaces) představuje jeden z významných směrů současného vývoje softwarových systémů. Tato rozhraní umožňují aplikacím využívat funkcionalitu externích služeb bez nutnosti jejich vlastního vývoje, což přináší úsporu času i zdrojů a zároveň rozšiřuje možnosti výsledného řešení. V kontextu softwarového inženýrství jde o praktické uplatnění principů modularity a znuvupoužitelnosti, kdy jsou jednotlivé části systému skládány z existujících komponent s jasně definovaným rozhraním, přičemž právě rozhraní API plní v této architektuře roli standardizovaného spojovacího prvku (13).

Typickým příkladem integrace externího API jsou mapové podklady, například pomocí Google Maps Platform nebo Mapbox, které umožňují vizualizaci geografických dat, navigaci a geolokační funkce bez nutnosti vlastní kartografické infrastruktury (14; 15). Další běžnou oblastí jsou cloudová úložiště pro obrázky a dokumenty, jako například Amazon S3, která poskytují optimalizaci souborů, transformace médií a bezpečné ukládání (16). Významné postavení mají rovněž platební brány jako PayPal API, které umožňují bezpečně zpracovávat platby s minimální námahou na straně vývojáře (17). Mimo tyto příklady lze nalézt řadu dalších integrací, například služby pro zasílání notifikací (Firebase Cloud Messaging), ověřování identity (OAuth) nebo analytiku (Google Analytics).

1.2 UX/UI trendy v mobilních aplikacích

Trendy v oblasti UX/UI designu mobilních aplikací se v posledních letech soustředí na propojení estetiky s funkčností. Minimalistický design, který klade důraz na přehlednost a intuitivní ovládání, se stal standardem, jenž usnadňuje uživatelskou orientaci a zvyšuje celkovou použitelnost. Stále častěji se také uplatňuje možnost přepínání mezi světlým a tmavým režimem, což nejen přispívá k lepšímu vizuálnímu komfortu, ale zohledňuje i individuální preference uživatelů, nebo lepší hospodaření s energií u mobilních zařízení. Významnou roli hraje i responzivita, tedy schopnost designu přizpůsobit se různým velikostem a rozlišením obrazovek. V neposlední řadě nabývá na důležitosti dodržování principů přístupnosti podle standardů WCAG, které zajišťují, že aplikace budou srozumitelné a ovladatelné i pro uživatele se specifickými potřebami.

1.2.1 Funkční design, UI/UX, Responzibilita

V oblasti návrhu mobilních aplikací je klíčovým požadavkem tvorba funkčního designu, který propojuje prvky UI (user interface) a UX (user experience) s důrazem na responzivitu. Funkční design znamená, že vizuální podoba aplikace není pouze estetická, ale především účelná, to znamená že každé rozhraní musí uživateli umožnit snadnou a efektivní interakci. UX se zaměřuje na celkový uživatelský zážitek, tedy na to, jak snadno a příjemně lze aplikaci používat, zatímco UI řeší konkrétní vzhled jednotlivých prvků, jako jsou tlačítka, typografie či barevná schémata. Společně tyto oblasti tvoří základ pro intuitivní, konzistentní a přístupný produkt (18).

Jedním z klíčových aspektů moderního designu je také responzivita, tedy schopnost uživatelského rozhraní dynamicky se přizpůsobit různým velikostem obrazovek a orientacím zařízení. Vzhledem k rozmanitosti mobilních zařízení, od malých telefonů až po tablety, je responzivní design nezbytný pro zajištění konzistentního a přívětivého zážitku napříč platformami. Responzivní návrh nejen zvyšuje přístupnost aplikace, ale má také přímý vliv na míru jejího využívání a uživatelskou spokojenost (19).

Významným principem funkčního designu je i dodržování tzv. platformových doporučení, jako jsou Human Interface Guidelines od Apple a Material Design od Googlu, které vývojářům a designérům poskytují rámec pro tvorbu aplikací odpovídajících standardům dané platformy. Tyto principy podporují konzistenci, snižují kognitivní zátěž a napomáhají lepší orientaci uživatele (20; 21).

1.2.2 Tmavý a světlý režim

V současném vývoji mobilních aplikací představuje integrace světlého a tmavého režimu významný trend v oblasti uživatelského rozhraní (UI) a uživatelské zkušenosti (UX). Tyto režimy, původně vnímané jako estetická preference, dnes plní důležité funkce z hlediska čitelnosti, vizuálního komfortu i energetické efektivity. Zatímco světlý režim, založený na kontrastu tmavého textu na světlém pozadí, připomíná tradiční tištěná média a bývá vhodnější pro denní světlo, tmavý režim využívá světlého textu na tmavém pozadí, což redukuje oslnění a vizuální únavu, zejména v nočních podmínkách. Při volbě mezi světlým a tmavým režimem hraje roli více faktorů, například typ displeje (OLED nebo LCD), světlo v okolí nebo rozdíly ve zraku uživatele. Moderní mobilní aplikace proto reflektují rostoucí požadavky na personalizaci prostřednictvím možnosti přepínání režimů nebo automatického přizpůsobení podle světelných podmínek. Současné operační systémy, jako Android a iOS, spolu s návrhovými frameworky typu Material Design nebo iOS Human Interface Guidelines, tuto funkcionalitu aktivně podporují (20; 21). Kromě estetické hodnoty přispívá tmavý režim také ke snížení spotřeby energie u OLED displejů, kde tmavé pixely zůstávají vypnuté. S rostoucím důrazem na přístupnost a digitální pohodlí se volba mezi světlým a tmavým režimem stává klíčovým prvkem kvalitního a inkluzivního UX designu, jehož cílem je přizpůsobit se dynamickým potřebám a kontextu uživatele (22)

1.2.3 Web Content Accessibility Guidelines

Web Content Accessibility Guidelines (WCAG) jsou mezinárodně uznávané standardy pro zajištění přístupnosti digitálního obsahu. Tyto směrnice byly vytvořeny konsorciem W3C (World Wide Web Consortium) v rámci iniciativy Web Accessibility Initiative (WAI) s cílem zpřístupnit webové a mobilní aplikace osobám se zdravotním postižením. Základní filozofie.

WCAG je postavena na čtyřech hlavních principech označovaných zkratkou POUR:

- **Perceivable (vnímatelné)**: obsah musí být prezentován tak, aby jej uživatel mohl vnímat, např. pomocí textových alternativ pro vizuální prvky
- **Operable (ovladatelné)**: uživatel musí být schopen ovládat rozhraní, například prostřednictvím klávesnice
- **Understandable (srozumitelné)**: informace a rozhraní by měly být snadno pochopitelné a předvídatelné
- **Robust (robustní)**: obsah musí být kompatibilní s různými asistivními technologiemi, například čtečkami obrazovky

WCAG standardy se dělí podle úrovně shody na tři stupně: úroveň A (základní přístupnost), úroveň AA (doporučovaná úroveň pro většinu aplikací, včetně veřejné správy v EU) a úroveň AAA (nejvyšší standard přístupnosti, který však bývá obtížně dosažitelný). Například pro úroveň A je vyžadováno, aby měl každý obrázek textový popis (alt text), pro úroveň AA je požadován minimální kontrast mezi textem a pozadím, a úroveň AAA vyžaduje mimo jiné možnost přizpůsobit rozhraní pro různé úrovně porozumění.

V praxi se požadavky WCAG promítají například do dostatečného barevného kontrastu textu, možnosti ovládat aplikaci pouze klávesnicí, přidávání alternativních popisků k obrázkům nebo označování formulářových polí. Implementace těchto pravidel významně zlepšuje přístupnost aplikace nejen pro osoby s postižením, ale i pro starší uživatele nebo osoby v dočasně ztížených podmínkách (např. hlučné prostředí nebo malý displej). Dodržování WCAG navíc často představuje právní požadavek, zejména v rámci veřejných institucí nebo projektů financovaných z veřejných prostředků. Kromě toho přístupné rozhraní přispívá ke zlepšení celkového uživatelského zážitku, čímž může pozitivně ovlivnit i obchodní úspěšnost dané aplikace (23) .

1.3 Bezpečnost a ochrana soukromí

Bezpečnost a ochrana soukromí patří mezi nejzásadnější oblasti vývoje mobilních aplikací, zejména s ohledem na množství citlivých dat, která aplikace často zpracovávají. Klíčovým prvkem je šifrování dat, a to jak při jejich přenosu, tak při ukládání, které chrání informace před neoprávněným přístupem. Stejně důležitá je i správně navržená autentizace a autorizace uživatelů, zajišťující, že ke konkrétním funkcím a datům mají přístup pouze oprávněné osoby. Zvláštní pozornost je třeba věnovat nakládání s osobními údaji v souladu s platnou legislativou (např. GDPR), což zahrnuje jak transparentnost zpracování, tak důraz na minimalizaci sběru dat a správu uživatelských oprávnění.

1.3.1 Šifrování dat

Šifrování dat je proces, při kterém se běžná (čitelná) data převádějí do nečitelné podoby za účelem ochrany před neoprávněným přístupem. Pouze subjekty, které mají odpovídající dešifrovací klíč, mohou data opět převést zpět do původní podoby. Tento princip hraje klíčovou roli v oblasti kybernetické bezpečnosti a je široce využíván i v mobilních aplikacích, zejména při přenosu citlivých informací přes internet, jako jsou osobní údaje, hesla nebo finanční transakce (24).

V kontextu mobilních aplikací je šifrování běžně využíváno jak při komunikaci se servery (např. pomocí protokolu HTTPS/TLS), tak pro lokální ukládání dat (např. šifrování

databází nebo souborů uložených v zařízení). Mnohé moderní operační systémy, jako Android a iOS, poskytují vývojářům nástroje pro šifrování uložených dat pomocí systémových API, jako jsou Keychain (iOS) nebo Keystore Systém pro Android (25; 26).

Typickým příkladem využití šifrování v praxi je end-to-end šifrování chatů v aplikacích pro zasílání zpráv. Například aplikace Messenger od společnosti Meta začal v roce 2023 ve výchozím nastavení používat end-to-end šifrování pro všechny osobní konverzace, což znamená, že zprávy může číst pouze odesílatel a příjemce, takže ani samotná platforma k nim nemá přístup (27, 29).

1.3.2 Autentizace a autorizace uživatelů

Autentizace a autorizace představují dva základní bezpečnostní mechanismy, které se sice často využívají společně, avšak plní rozdílné úlohy. Autentizace slouží k ověření identity uživatele, typicky prostřednictvím hesla, biometrických údajů či přihlašovacího tokenu. V moderních aplikacích je běžné využití JWT (JSON Web Token), který po úspěšném přihlášení uživatele slouží jako digitální důkaz identity a umožňuje bezpečné předávání ověřovacích informací mezi klientem a serverem. Pro zvýšení úrovně bezpečnosti lze autentizační proces rozšířit o dvoufaktorovou autentizaci (2FA), při níž uživatel kromě znalosti (hesla) musí doložit i vlastnictví (například mobilního zařízení pro SMS kód či autentizační aplikaci). Tímto způsobem je výrazně sníženo riziko zneužití účtu v případě úniku přihlašovacích údajů.

Autorizace naopak určuje, jaké zdroje nebo funkce může již ověřený uživatel využívat. Jinými slovy, autentizace odpovídá na otázku *„Kdo jsi?“*, zatímco autorizace řeší *„Co smíš dělat?“*. Autorizace je v praxi realizována pomocí uživatelských rolí a oprávnění, které definují úrovně přístupu k jednotlivým částem systému. Například běžný uživatel může mít přístup pouze k prohlížení událostí, zatímco podnikový účet může navíc vytvářet a spravovat vlastní obsah. Správně nastavený autorizační systém zajišťuje, že citlivé funkce a data jsou dostupné pouze těm uživatelům, kteří k nim mají oprávnění, což je nezbytné pro ochranu informačních systémů a prevenci neoprávněného přístupu. (28).

1.3.3 Práce s osobními údaji a oprávněními

Zpracování osobních údajů a správa uživatelských oprávnění v mobilních aplikacích představuje klíčovou oblast, která vyžaduje vysokou míru právní a technické odpovědnosti. Vývojáři a provozovatelé aplikací musí zajistit, aby bylo se všemi daty uživatelů nakládáno transparentně, účelně a bezpečně. To zahrnuje informování uživatelů o rozsahu a účelu sběru údajů nebo oprávnění, době jejich uchování i případném sdílení s třetími stranami.

Mobilní aplikace často přistupují k citlivým údajům, jako jsou informace o poloze, kontakty, obrazová data nebo biometrie. Z tohoto důvodu je nezbytné uplatňovat zásady tzv. privacy by design a privacy by default, tedy zajištění ochrany soukromí již ve fázi návrhu a prostřednictvím výchozího nastavení aplikace.

Nedodržení pravidel ochrany osobních údajů může vést nejen ke ztrátě důvěry uživatelů, ale i k právním a finančním důsledkům ze strany regulačních orgánů (29; 30; 31).

1.3.4 Distribuce softwarových produktů

Distribuce softwarových produktů je nedílnou součástí jejich životního cyklu a má zásadní vliv na bezpečnost i důvěryhodnost výsledné aplikace. Moderní software je zpravidla šířen prostřednictvím oficiálních kanálů, jako jsou webové stránky vývojáře nebo oficiální obchody s aplikacemi (např. Google Play Store či Apple App Store). Tyto platformy zajišťují, že distribuovaný software prochází určitým procesem kontroly, čímž je minimalizováno riziko šíření škodlivého kódu, podvodných aplikací nebo neautorizovaných úprav.

1.4 Umělá inteligence a její využití v mobilních aplikacích

Umělá inteligence (AI) představuje oblast informatiky, která se zabývá vývojem systémů schopných vykonávat úkoly vyžadující lidskou inteligenci jako je například učení, rozhodování, zpracování jazyka či rozpoznávání obrazů. S rozvojem výpočetního výkonu a dostupnosti velkých dat se AI technologie staly běžnou součástí komerčních produktů, zejména v oblasti mobilních aplikací, kde pomáhají zvyšovat komfort, personalizaci i bezpečnost.

V mobilních aplikacích se AI uplatňuje v celé řadě scénářů, a to od automatizovaného zákaznického servisu přes doporučovací systémy až po detekci anomálií nebo rozpoznávání objektů na základě kamerového vstupu. Jedním z nejrozšířenějších typů AI v mobilních aplikacích je Natural Language Processing (NLP), který umožňuje porozumění lidské řeči a její zpracování. Díky tomu mohou aplikace komunikovat s uživatelem přirozeným jazykem, což je základem virtuálních asistentů nebo chatbotů. Dále se uplatňuje strojové učení (machine learning) pro analýzu uživatelského chování, prediktivní funkce nebo inteligentní filtrování obsahu (32; 33).

Typickým příkladem mobilní aplikace postavené přímo na AI je ChatGPT, aplikace vyvinutá společností OpenAI. Tato aplikace využívá pokročilé jazykové modely pro vedení konverzací, vysvětlování konceptů, generování textu a mnoho dalších úloh. Díky NLP a trénování na rozsáhlých datech dokáže interagovat v přirozeném jazyce způsobem, který připomíná konverzaci s člověkem (34).

Vedle čistě konverzačních aplikací existují i aplikace, které využívají AI k jiným účelům jako například Lensa, mobilní editor fotografií, využívá neuronové sítě k úpravám portrétů, generování stylizovaných avatarů a inteligentnímu rozmazávání pozadí. Aplikace tohoto typu kombinují počítačové vidění a generativní AI, čímž přinášejí uživatelům nástroje, které byly dříve dostupné jen profesionálům (35).

Podle výzkumu společnosti GoodFirms z roku 2024 patří AI mezi hlavní trendy, které formují budoucnost vývoje aplikací. AI nejen zvyšuje uživatelský zážitek a efektivitu, ale také umožňuje vývojářům lépe reagovat na individuální potřeby uživatelů prostřednictvím adaptivních a personalizovaných řešení (36).

1.5 Inovativní přístupy ve vývoji aplikací

Inovativní přístupy ve vývoji mobilních aplikací přinášejí nové možnosti, jak zlepšit uživatelský zážitek a rozšířit funkcionalitu aplikací nad rámec tradičních řešení. Významným trendem je spolupráce v rámci technologických ekosystémů od společností, jako jsou například Apple nebo Samsung, které umožňují plynulé propojení aplikací napříč různými zařízeními, jako jsou chytré hodinky, tablety či televize. Do popředí se rovněž dostává využití rozšířené a virtuální reality (AR/VR), jež přináší nové formy interakce a nachází uplatnění v oblastech jako vzdělávání, obchod nebo zdravotnictví.

1.5.1 Spolupráce v rámci ekosystémů

Spolupráce v rámci ekosystémů znamená provázanost zařízení, služeb a aplikací jednoho výrobce tak, aby uživatelský zážitek byl co nejplynulejší a funkčně propojený napříč různými zařízeními. Typickým příkladem je Apple ekosystém, kde aplikace mohou využívat funkce jako Handoff, Universal Clipboard, AirDrop nebo iCloud synchronizaci, což umožňuje začít činnost na jednom zařízení a bez přerušení pokračovat na jiném. Vývojáři mohou tyto vazby implementovat díky jednotné platformě a nástrojům jako App Groups, CloudKit nebo Continuity API (37). Samsung na druhé straně propojuje svá zařízení prostřednictvím platformy SmartThings a dalších specializovaných funkcí, které umožnují například sdílení myši a klávesnice mezi Galaxy zařízeními, plynulé přepínání sluchátek mezi telefonem a tabletem nebo zrcadlení obrazovky (38; 39). Obě firmy staví ekosystém jako konkurenční výhodu, která podporuje loajalitu uživatelů a zároveň otevírá nové možnosti pro vývojáře, ti tak mohou vytvářet aplikace, které překračují hranice jednoho zařízení a využívají sílu propojeného prostředí.

1.5.2 Rozšířená a virtuální realita v mobilních zařízeních

AR (Augmented Reality) a VR (Virtual Reality) představují technologie, které zásadně mění způsob, jakým lidé interagují s digitálním obsahem. Zatímco AR obohacuje reálné prostředí o digitální prvky, jež jsou zobrazovány například skrze kameru mobilního zařízení, VR vytváří zcela virtuální prostředí, do kterého je uživatel plně ponořen pomocí speciálních zařízení, jako jsou VR brýle. Obě technologie nacházejí využití nejen v herním průmyslu, ale i v oblastech jako je vzdělávání, zdravotnictví, architektura nebo obchod (37).

V kontextu mobilních aplikací se v praxi uplatňuje především rozšířená realita, a to díky rostoucím možnostem výpočetního výkonu moderních chytrých telefonů a vývojářským nástrojům jako ARKit (Apple) nebo ARCore (Google). Tyto frameworky umožňují vývojářům jednoduše vytvářet aplikace, které reagují na okolní prostředí, rozpoznávají roviny, objekty nebo gesta a umisťují digitální obsah do reálného prostoru. Uživatelé tak mohou například vizualizovat produkty přímo ve svém domově, učit se interaktivní formou nebo procházet virtuální expozice. AR/VR tak nejsou jen futuristickým trendem, ale praktickým nástrojem, který přináší nový rozměr interakce a uživatelského zážitku i do běžných mobilních aplikací (38) (39). Dva dobře známé a úspěšné příklady integrace AR do mobilních aplikací jsou:

Pokémon GO: Hra od společnosti Niantic, která způsobila světový fenomén, využívá rozšířenou realitu k tomu, aby hráči mohli „chytat“ pokémony ve skutečném světě. Hra kombinuje GPS lokalizaci a kameru telefonu s AR prvky a ukazuje, jak může být zážitek z rozšířené reality přirozeně začleněn do herního prostředí (37).

IKEA Place: Aplikace od nábytkářského giganta IKEA umožňuje uživatelům zobrazit 3D modely nábytku ve skutečném prostředí jejich domácnosti pomocí AR. Uživatel tak může vidět, jak se konkrétní produkt bude hodit do jeho interiéru, a to v reálném měřítku a perspektivě (37).

1.6 Průzkum klíčových pojmů rešerše a implikace pro vlastí řešení

Jedním z klíčových rozhodnutí v počáteční fázi vývoje nové mobilní aplikace je volba vhodného programovacího jazyka či frameworku, ve kterém bude aplikace implementována. Neméně významnou součástí návrhu je také výběr API třetích stran, která musí být zvolena nejen s ohledem na požadované funkce, ale rovněž na jejich cenovou politiku a dlouhodobou udržitelnost. Dalším zásadním rozhodnutím je volba způsobu implementace backendové části aplikace. Lze zvolit buď přístup typu BaaS (Backend as a Service), který poskytuje hotovou infrastrukturu a škálovatelné služby, nebo klasický přístup, kdy je backend vyvíjen od základu.

V rámci rešerše této práce byly identifikovány čtyři hlavní kandidáti pro vývoj mobilní aplikace: React Native, Flutter, Kotlin Multiplatform a Progressive Web Apps (PWAs). Cílem této práce je vytvořit mobilní aplikaci pro operační systém Android, a proto musíme možnost využití PWAs vyloučit. Přestože se jedná o výkonný nástroj umožňující převést webovou aplikaci do podoby, která se svým vzhledem a chováním blíží mobilním aplikacím, ve své podstatě zůstává webovou aplikací běžící v prohlížeči, a tedy nesplňuje požadavek na plně mobilní řešení.

Další klíčovou oblastí návrhu aplikace je výběr API třetích stran. Jako první jsou **notifikační služby**, mezi zavedená řešení patří Firebase Cloud Messaging, OneSignal a Apple Push Notifications, přičemž ta neumožnuje spolupráci s jiným operačním systémem, něž iOS proto není do průzkumu zahrnuta. **Mapové služby** pro vyhledávání a práci s mapovými daty, přičemž uvažovanými technologiemi jsou MapBox, Google Maps a OpenStreetMap. Jako poslední je **uložiště obrazových dat**, toto API musí poskytovat CRUD endpointy pro práci s obrázky, přičemž výhodou je možnost nahrávání přímo z klienta. Při výběru je nutné zohlednit i cenovou politiku. Zvažovanými kandidáty jsou Cloudflare Images a Google Cloud Storage.

Poslední významnou součástí návrhu je volba přístupu k implementaci backendu, jednou z možností je využití BaaS (Backend as a Service) druhou pak klasický postup vývoje backendu.

Následující část textu se zaměří na porovnání zvolených tří frameworků, a to z hlediska jejich rozšíření a míry využití v průběhu posledních let, přičemž budou posouzeny také jejich funkční vlastnosti z různých perspektiv. Současně bude provedeno porovnání jednotlivých zvažovaných API třetích stran prostřednictvím srovnávacích tabulek, které umožní přehledné vyhodnocení jejich výhod, nevýhod a klíčových parametrů. Jako poslední bude na základě rešerše rozhodnuto o způsobu implementace backendu.

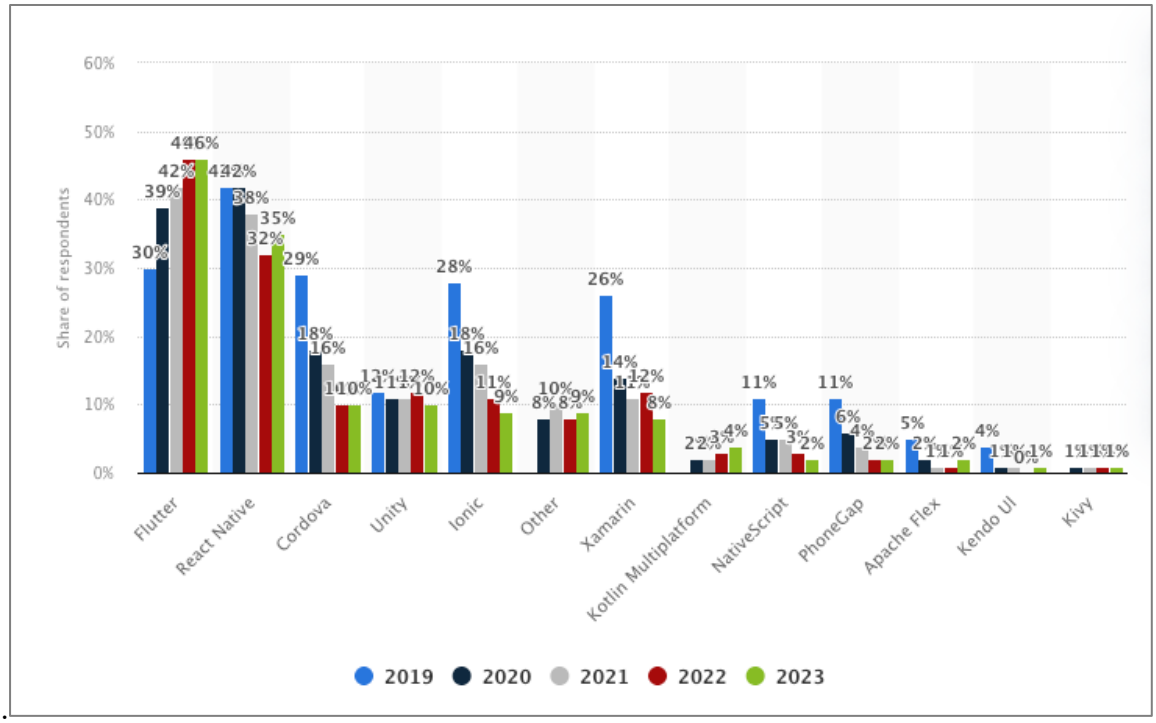
Výsledkem provedené analýzy bude výběr nejvhodnějších technologií a nástrojů, které budou nejlépe odpovídat požadavkům a cílům této práce. Zvolená řešení budou představovat optimální kompromis mezi technickými možnostmi, finanční náročností a dlouhodobou udržitelností projektu. Následně budou tato řešení aplikována při samotné implementaci, čímž se zajistí efektivní a kvalitní realizace finální mobilní aplikace.

1.6.1 Průzkum a volba frontendového frameworku pro implementaci aplikace

Předložený graf poskytuje přehled o popularitě a míře využití vybraných multiplatformních frameworků v období od roku 2019 do roku 2023. Data vycházejí z celosvětového průzkumu realizovaného na webovém portálu statista.com, jehož se zúčastnilo více než 29 tisíc respondentů.

Z grafu je patrné, že z uvedených kandidátů, ale i v porovnání s ostatními možnostmi, výrazně dominují frameworky Flutter a React Native. Naopak Kotlin Multiplatform nedosahuje zdaleka takové míry rozšíření. Tento stav může být do značné míry ovlivněn skutečností, že Kotlin Multiplatform vstoupil na trh až v roce 2020 v alfa verzi, tedy později než React Native (2015) či Flutter (2018).

Při bližším pohledu na oba vedoucí frameworky lze zaznamenat odlišný trend vývoje jejich popularity. Zatímco u Flutteru je patrný postupný nárůst, React Native zaznamenává mírný pokles, a to s výjimkou roku 2023, kdy došlo k dočasnému nárůstu. Na základě těchto údajů lze konstatovat, že jak Flutter, tak React Native představují relevantní a široce využívané možnosti pro vývoj multiplatformních mobilních aplikací. Kotlin Multiplatform se oproti tomu jeví jako perspektivní, avšak dosud méně zavedená technologie, jejíž rozšíření je ve srovnání s ostatními dvěma frameworky zatím omezené (45).



Obrázek 1 - Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2023. zdroj: (45)

Následující tabulka porovnává dva celosvětově nejpoužívanější frameworky pro vývoj multiplatformních aplikací Flutter a React Native z několika klíčových kritérií. Tím poskytuje komplexní přehled jejich hlavních předností i slabších stránek.

V rámci výkonnostního kritéria jednoznačně vychází lépe Flutter, a to především díky vlastnímu renderovacímu enginu Skia a efektivní architektuře. Kritérium konzistence uživatelského rozhraní taktéž hovoří ve prospěch Flutteru, díky enginu Skia jsou všechny komponenty vykreslovány shodně na každé platformě. Oproti tomu React Native převádí kód na nativní komponenty dané platformy, což v praxi znamená, že například tlačítka na Androidu a iOS mohou mít odlišný vzhled, pokud nejsou explicitně sjednocena stylováním.

V kategorii cílových platforem také vítězí Flutter, neboť nabízí nativní podporu širšího spektra platforem. React Native sice nativně podporuje pouze hlavní platformy, avšak prostřednictvím komunitních rozšíření lze dosáhnout podpory i dalších. Jedinou oblastí, ve které má React Native navrch, je podpora televizních operačních systémů (Apple TV, Android TV).

Popularita byla posuzována podle počtu udělených hvězdiček na GitHubu, kde Flutter dosáhl přibližně 172 tisíc hvězd oproti zhruba 123 tisícům u React Native. Přesto toto měřítko může být zkreslující, lze proto porovnání doplnit i počtem veřejných repozitářů na GitHubu. Při vyhledávání na GitHubu bylo nalezeno přibližně 878 tisíc repozitářů obsahujících klíčové slovo „flutter“, zatímco „react native“ se vyskytuje u přibližně 530 tisíc repozitářů. Tyto údaje naznačují vyšší míru využití a popularity frameworku Flutter.

Co se týče programovacího jazyka, jedná se do značné míry o otázku osobní preference. Dart, vyvinutý současně s Flutterem, je určen výhradně pro tento framework, a proto je nutné jej osvojit jako nový jazyk. Naproti tomu JavaScript, používaný v React Native, je zavedený a široce známý, což může být pro vývojáře výhodou.

Na základě provedené analýzy lze konstatovat, že ačkoli je Flutter mladším frameworkem než React Native, nabízí vysoký výkon, jeho popularita a komunita rychle rostou, podporuje širší spektrum platforem a zajišťuje konzistentní vzhled uživatelského rozhraní napříč platformami. Z těchto důvodů byl pro implementaci mobilní aplikace v rámci této práce zvolen právě Flutter (58; 59; 60; 61).

Tabulka 1 - Porovnání multiplatformních frameworků. zdroj: (62; 63)

Kritérium	Flutter	React Native
Výkon	Vysoká výkonnost, vysoká plynulost animací a efektivita se zdroji.	Vysoká výkonnost, bridge bottleneck (komunikace RN a nativních modulů), méně efektivní práce se zdroji, nižší plynulost.
UI konzistence	Renderovací engine Skia, vypadá na vše platformách totožně.	Převádí kód na nativní komponenty dané platformy, UI nemusí být konzistentní napříč platformami.
Cílové platformy	Android, iOS, macOS, Windows, Linux, web	Android, iOS, web, TV
Komunita a ekosystém	Mladší komunita, poslední roky velmi vysoký nárust knihoven.	Dospělý produkt, velké množství knihoven.
Popularita	172 000 hvězd na GitHub	123 000 hvězd na Githubu
Programovací jazyk	Dart (pouze pro Flutter)	JavaScript (zavedený jazyk)
Oficiální rok vydání	2018	2015

1.6.2 Průzkum a výběr externích API pro implementaci v aplikaci

Prvním klíčovým API pro doručování zpráv do aplikace jsou notifikační služby. Při jejich výběru je vhodné porovnat zejména podporované platformy, cenu a snadnost integrace do vyvíjené aplikace.

Z provedeného srovnání vyplývá, že obě zvažované služby umožňují zasílání notifikací na mobilní platformy, přičemž primárním cílem je operační systém Android. Integrace obou řešení do aplikace je poměrně snadná, avšak FCM má výhodu v tom, že nabízí nativní API pro Flutter, a to v rámci stejného ekosystému od společnosti Google, což usnadňuje implementaci a zajišťuje lepší kompatibilitu.

Hlavní rozdíl mezi oběma řešeními spočívá v ceně. Zatímco Firebase Cloud Messaging je poskytován zcela zdarma, služba OneSignal může při vyšším objemu využívání generovat dodatečné náklady. Z tohoto důvodu, a s ohledem na technickou provázanost s ostatními použitými technologiemi, bylo rozhodnuto implementovat do aplikace právě Firebase Cloud Messaging.

Tabulka 2 - Porovnání notifikačních služeb. zdroj: (64; 65)

Kritérium	**Firebase Cloud Messaging (FCM)**	**OneSignal**
Podporované platformy	Android, iOS, Web	Android, iOS, Web, Huawei
Cena	Zdarma	Free tier (do 10k odběratelů), poté od $9/měs.
Snadnost integrace	Dobrá, nativní SDK, podrobná dokumentace.	Dobrá, SDK pro více platforem, rychlá implementace.

Dalším nezbytným API pro potřeby aplikace jsou mapové podklady, přičemž nejvyšší prioritou je funkce doporučení míst na základě části nebo celého názvu vyhledávané lokality. Pro samotné načítání mapových podkladů je možné využít službu OpenStreetMap, která je bezplatná. Pro vyhledávání a doporučování míst je však nutné zvolit specializované API, které bude porovnáno především z hlediska kvality poskytovaných výsledků a ceny, s cílem nalézt optimální řešení.

Z provedeného srovnání vyplývá, že cenově nejvýhodnější variantou je využití služby Nominatim (součást OpenStreetMap). Tato služba je sice dostupná zdarma, avšak není navržena pro masivní zátěž, má striktně omezené kvóty a neposkytuje požadovanou úroveň přesnosti výsledků. Teoreticky je možné nasadit vlastní instanci Nominatimu a tyto limity tak obejít, nicméně takové řešení není součástí rozsahu této práce a ani by tím nebyl vyřešen problém s přesností.

Alternativou je Google Places Autocomplete API, které nabízí velmi vysokou přesnost a rozsáhlou databázi míst. Díky měsíčnímu kreditu $200 může být pro menší až střední objemy dotazů cenově výhodnější než Mapbox.

Mapbox Search Box API představuje dobrou volbu pro integraci s vlastními mapovými podklady a jednoduchou správu služeb na jedné platformě. Cenově je při preview tarifu srovnatelný s Google, avšak po zavedení standardních cen může být dražší.

Na základě provedeného průzkumu se služby Mapbox i Google ukázaly jako vhodné možnosti. Pro implementaci aplikace byl však zvolen Mapbox, a to zejména z důvodu předchozích zkušeností s jeho integrací a využíváním.

Tabulka 3 - Porovnání mapových služeb. zdroj: (66; 67; 68)

Kritérium	Mapbox Search Box API	Google Places Autocomplete API	OpenStreetMap (Nominatim/Photon)
Cena	500 session/měs. zdarma, poté od $3,00 za variabilní počet sessions.	$200 kredit/měs. (~70k req zdarma), poté $2,83 / 1000 req.	Zdarma ale značně omezující limity.
Funkce endpointu	Návrhy při psaní, přesné výsledky s metadaty (lat/long, typ, relevance, vzdálenost).	Návrhy při psaní, metadata, velmi vysoká přesnost.	Základní návrhy, méně přesné, závisí na instanci.

Poslední potřebnou službou je uložiště obrazových dat, toto uložiště musí poskytovat API pro CRUD operace nad jednotlivými obrázky, musí být cenově akceptovatelné a rychle dostupné po světě, například pomocí CDN.

Následující tabulka vypovídá, že obě porovnávané splňují požadavek na podporu CRUD operací pro práci s obrázky. Z hlediska cenové politiky však vychází výhodněji Cloudflare Images, které za fixní poplatek 5 USD měsíčně nabízí úložiště až pro 100 000 obrázků, a to včetně doručení prostřednictvím vestavěné CDN. Naproti tomu Google Cloud Storage účtuje samostatně poplatky za ukládání dat, odchozí přenos a síťové požadavky, přičemž CDN není zahrnuta v základní ceně.

S ohledem na tyto skutečnosti lze konstatovat, že Google Cloud Storage v cenové i infrastrukturní oblasti zaostává. Z tohoto důvodu bylo pro implementaci aplikace zvoleno řešení Cloudflare Images.

Tabulka 4 - Porovnání služeb pro ukládání obrázků. Zdroj: (69; 70)

Kritérium	Cloudflare Images	Google Cloud Storage (GCS)
CRUD	Ano, vestavěné API pro nahrávání, mazání a správu.	Ano, REST/JSON API, široké možnosti správy objektů.
Cena	$5/měs. za 100k obrázků + $1/měs. za každý další 100k; $1/měs. za 100k transformací.	Platba za uložený objem + počet požadavků + odchozí data; cena závisí na regionu.
Optimalizace a CDN	Vestavěná CDN a optimalizace velikosti obrázků.	Možná integrace s Cloud CDN (placené zvlášť).

1.6.3 Výběr způsobu implementace backendu

V rámci rozhodování o způsobu implementace backendu byly zvažovány dvě hlavní varianty, využití služby Backend as a Service (BaaS) a vývoj vlastního backendového řešení. Obě možnosti byly posouzeny z několika hledisek, zejména z technického, ekonomického, provozního a uživatelského pohledu. Na základě těchto kritérií bylo možné objektivně stanovit, která z variant nejlépe odpovídá potřebám projektu.

Technický pohled:

BaaS nabízí hotové moduly (autentizace, databáze, notifikace, file storage) a umožňuje rychlé spuštění aplikace bez rozsáhlé implementace backendu. Vlastní backend však poskytuje plnou kontrolu nad architekturou, logikou i bezpečnostními mechanismy, což umožňuje lépe přizpůsobit řešení specifickým potřebám projektu a integracím třetích stran.

Ekonomický pohled:

BaaS může být finančně výhodný při malém rozsahu aplikace, ale s rostoucím počtem uživatelů a transakcí mohou náklady rychle narůstat. Vlastní backend vyžaduje investici do vývoje a správy infrastruktury, avšak dlouhodobě může být stabilnější z hlediska nákladů, zejména pokud je možné optimalizovat provoz podle konkrétních potřeb.

Provozní pohled:

BaaS minimalizuje provozní starosti díky automatickému škálování a správě infrastruktury. Vlastní backend vyžaduje nasazování, monitoring, řešení incidentů a průběžnou optimalizaci, ale nabízí větší nezávislost na poskytovateli služeb a flexibilitu při volbě technologií.

Uživatelský pohled:

Pro koncového uživatele není rozdíl mezi BaaS a vlastním backendem přímo viditelný. Vlastní backend však může lépe reagovat na specifické funkční požadavky, rychleji přizpůsobit API vývoji aplikace a snížit závislost na změnách podmínek poskytovatele BaaS.

Na základě těchto hledisek bylo rozhodnuto implementovat **vlastní backend**. Toto řešení sice vyžaduje vyšší časovou i technickou investici, ale poskytuje maximální kontrolu nad vývojem, možnost přizpůsobení a lepší prostor pro prezentaci komplexních technických řešení v rámci diplomové práce.

2 ANALÝZA OBDOBNÝCH ŘEŠENÍ

Události a akce tvoří důležitou součást společenského života a digitální nástroje pro jejich vyhledávání se staly běžnou součástí každodenního používání mobilních zařízení. Přestože existuje několik zavedených řešení, jejich kvalita, zaměření a přístup k uživateli se výrazně liší. Analýza těchto nástrojů může pomoci lépe pochopit potřeby uživatelů a současně odhalit slabiny, které je možné v novém návrhu překonat.

V následující kapitole je provedena analýza stávajících mobilních a webových aplikací, které slouží k objevování, správě a sdílení událostí v blízkém okolí. Cílem této analýzy je zmapovat aktuální stav trhu, identifikovat klíčové funkce, porovnat uživatelské zkušenosti a odhalit příležitosti, které by mohly být využity při návrhu vlastní aplikace.

Do analýzy byly zahrnuty tři vybrané aplikace:

- **Facebook Events**: součást známé sociální sítě, globálně rozšířená platforma pro správu a objevování událostí
- **Presentie**: méně známá, ale inovativní aplikace s důrazem na komunitní organizaci akcí
- **Liiiv**: novější aplikace která se zaměřuje inovativní přístup prozkoumávání událostí, navrch umožnuje propagaci umělců, či podniků jako takových

Analýza každé aplikace je provedena samostatně a zahrnuje základní informace, uživatelské rozhraní, klíčové funkce, technické vlastnosti, zpětnou vazbu od uživatelů a základní SWOT analýzu. SWOT analýza byla provedena na základě osobního testování aplikace, které umožnilo komplexní zhodnocení jejích silných a slabých stránek, příležitostí a hrozeb. Následně jsou jednotlivá řešení porovnána a vyhodnocena z hlediska silných a slabých stránek s důrazem na aspekty důležité pro návrh vlastní aplikace.

2.1 Facebook Events

Aplikace Facebook Events je nástroj vyvinutý společností Meta, který slouží k plánování, organizaci a správě různých typů událostí. Dnes je již plně integrovaná do hlavní platformy Facebook a dostupná jak ve webové, tak i mobilní verzi. Uživatelům nabízí širokou škálu funkcí, které umožňují efektivní koordinaci mezi organizátory a účastníky, a tím výrazně usnadňují správu a sdílení událostí v online prostoru.

2.1.1 Cílová skupina a účel aplikace

Aplikace je určena široké veřejnosti všech věkových kategorií, především uživatelům Facebooku. Jejím cílem je pomoci lidem objevovat události ve svém okolí, sledovat,

co se chystá, organizovat vlastní akce a jednoduše se připojit k aktivitám svých přátel nebo komunit.

2.1.2 Hlavní funkce

Jednou ze základních funkcí je možnost vytváření událostí, kdy uživatel může snadno založit veřejnou nebo soukromou událost podle potřeby. V rámci procesu tvorby lze specifikovat datum, čas, místo konání, popis události, přidat tematický obrázek či určit kategorii události. Dále je možné nastavit událost jako opakující se, což je užitečné například pro pravidelné setkání nebo cyklické akce.

Další klíčovou funkcionalitou je správa hostů a systém pozvánek. Organizátor může oslovit jednotlivé přátele, skupiny nebo širokou veřejnost. Uživatelé mají možnost potvrdit svou účast, označit nejistotu nebo se z účasti omluvit, čímž vzniká přehledná zpětná vazba o očekávané účasti. Organizátoři mají zároveň možnost informovat přihlášené uživatele o případných změnách či aktualizacích prostřednictvím notifikací.

Funkce objevování událostí umožňuje uživatelům procházet veřejné akce na základě lokace, data, tématu nebo zájmů. Facebook využívá své algoritmy k personalizaci doporučení, přičemž zohledňuje předchozí interakce uživatele, jeho záliby i účast přátel na jiných událostech. Tato schopnost přispívá k vyšší míře zapojení uživatelů do komunitního dění.

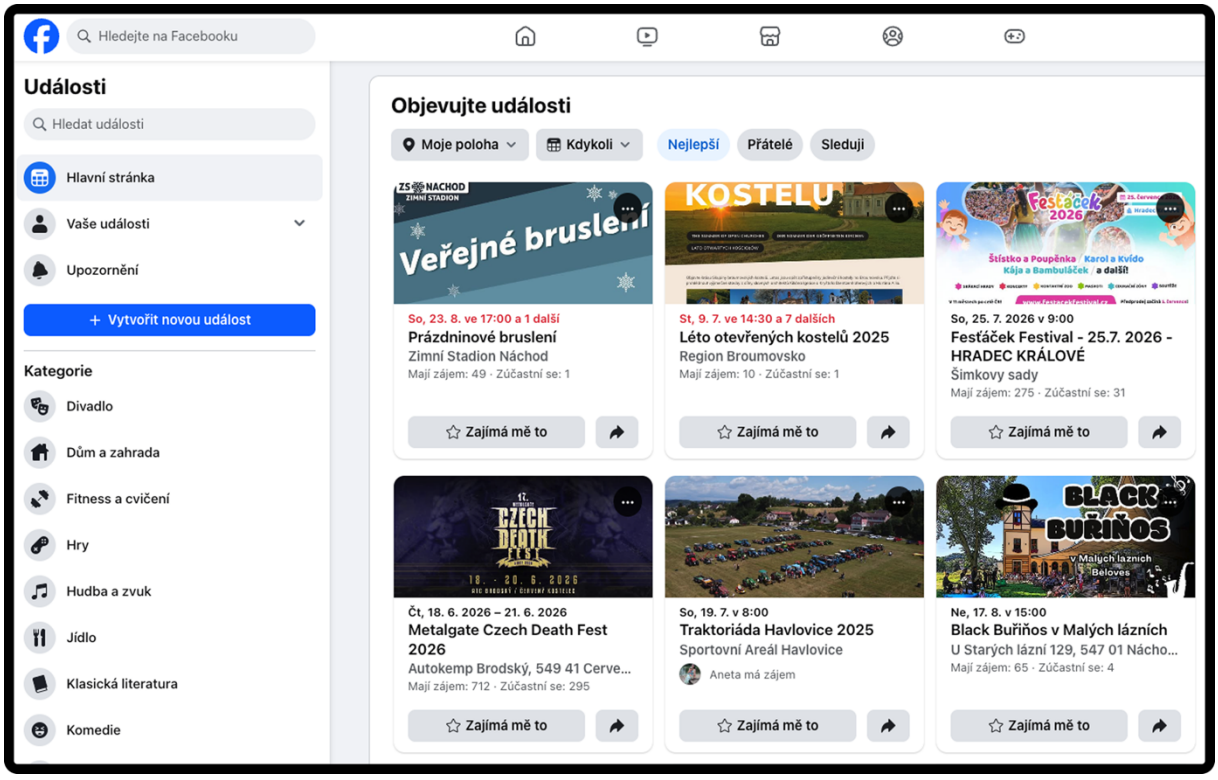
Důležitým prvkem aplikace je integrace s kalendářními systémy a upozorněními. Uživatelé si mohou přidávat události do svých externích kalendářů, jako je Google Calendar nebo Apple Calendar. Facebook zároveň automaticky generuje připomínky a upozornění na nadcházející události, což napomáhá lepší organizaci času a snižuje riziko opomenutí plánované aktivity.

Součástí funkcionality je také mapové zobrazení místa konání události. Díky propojení s geografickými daty mohou uživatelé snadno identifikovat polohu události, zobrazit trasu a využít navigační služby. Pokud dojde ke změně místa, informace se automaticky aktualizují a uživatelé jsou o změně informováni.

V neposlední řadě nabízí aplikace prostor pro sdílení obsahu v rámci samotné události. Organizátoři i účastníci mohou přidávat příspěvky, fotografie, videa či komentáře, čímž vzniká dynamický komunikační kanál. Tato možnost podporuje interakci, budování vztahů a sdílení zážitků spojených s událostí jak před jejím konáním, tak i po jejím skončení (43).

2.1.3 Uživatelské rozhraní a UX

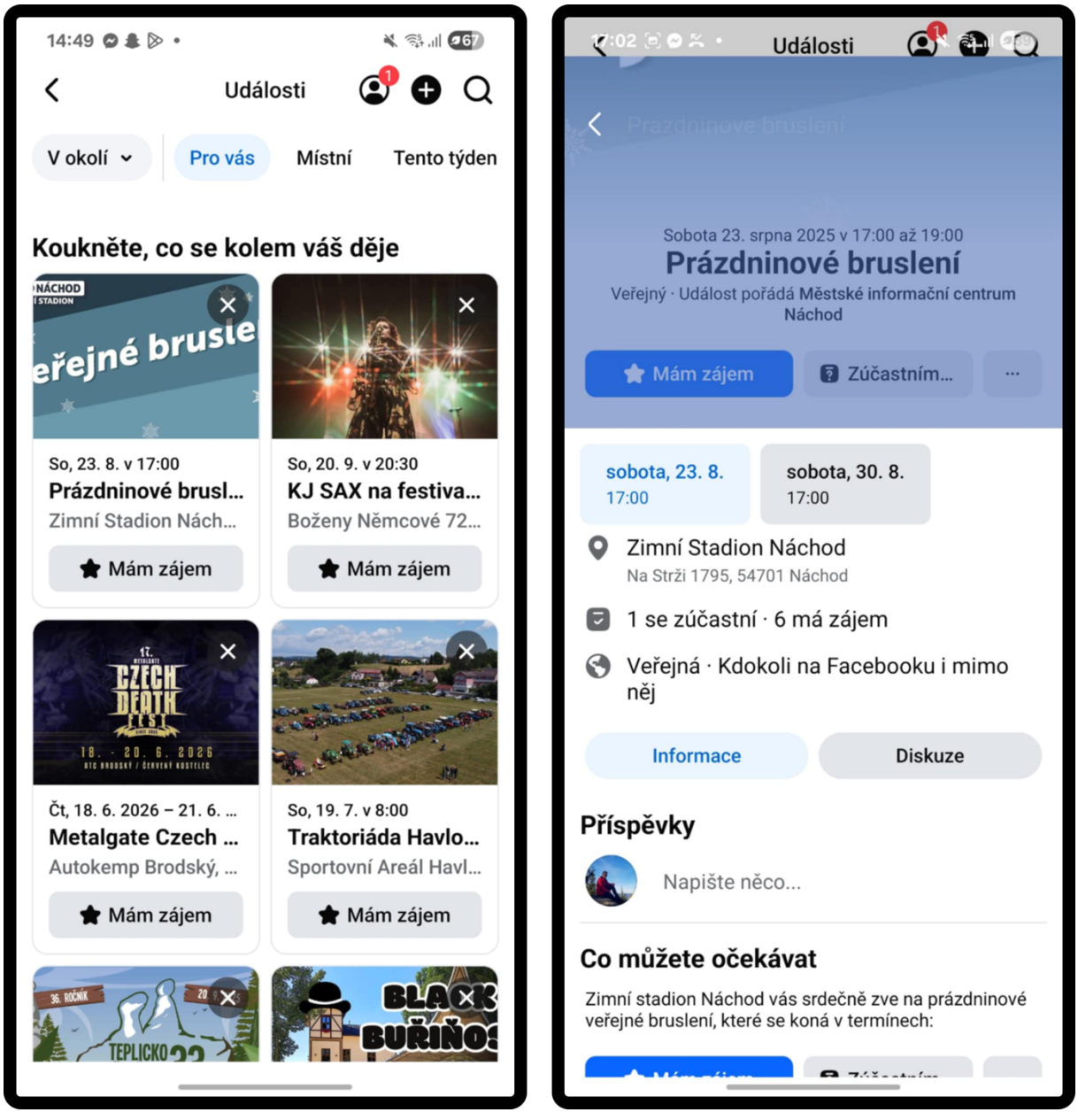
Webové uživatelské rozhraní a uživatelská zkušenost aplikace Facebook Events je navržena s důrazem na přehlednost, dostupnost a snadnou orientaci v široké nabídce událostí. Rozhraní využívá vertikální navigační menu vlevo, kde jsou umístěny základní sekce jako „Vaše události“, „Upozornění“ či tlačítko pro vytvoření nové události. Pod tímto menu je uživatelům umožněno filtrovat události dle kategorií (např. Divadlo, Hudba, Jídlo, Sport atd.), což přispívá k efektivnímu vyhledávání. V hlavní části obrazovky je dominantním prvkem přehledný grid událostí, který obsahuje vizuálně výrazné náhledy, základní informace (název, datum, místo, počet účastníků) a akční tlačítko „Zajímá mě to“. Filtrační panel v horní části umožňuje dále omezit zobrazené události dle polohy, času nebo relevantnosti (např. Nejlepší, Přátelé, Sleduji), čímž se zvyšuje personalizace obsahu. Celkové uživatelské prostředí působí intuitivně, vizuálně konzistentně a podporuje rychlou interakci s událostmi bez nutnosti složité navigace.



Obrázek 2 - Webové uživatelské rozhraní aplikace Facebook Events. zdroj: (43)

Mobilní uživatelské rozhraní aplikace Facebook Events je navrženo s ohledem na omezený prostor displeje a potřebu jednoduchého, ale efektivního přístupu k informacím. Horní část obrazovky obsahuje základní navigaci, včetně návratu zpět, notifikací, vytvoření nové události a nastavení polohy („V okolí“). Níže jsou umístěny filtrační přepínače (Pro vás, Místní, Tento týden), které umožňují rychlé omezení zobrazovaných událostí dle

relevance a času. Hlavní obsah tvoří kartový systém událostí s vertikálním scrolováním, kde každá karta zobrazuje vizuální náhled, název, datum, lokaci a akční tlačítko „Mám zájem“. Rozhraní je přehledné, optimalizované pro ovládání jednou rukou a navrženo tak, aby minimalizovalo nutnost hluboké navigace. Z hlediska uživatelské zkušenosti je kladen důraz na rychlý přístup k personalizovanému obsahu, intuitivní interakci a okamžitou orientaci v tom, co se „kolem uživatele děje“. Design zachovává vizuální kontinuitu s desktopovou verzí a přináší plnohodnotný zážitek i na mobilních zařízeních.



Obrázek 3 - Mobilní uživatelské rozhraní aplikace Facebook Events. zdroj: (43)

2.1.4 Uživatelská hodnocení

Hodnocení uživatelů v Obchodě Play dosahuje 4,5 z 5 hvězd a vychází z více než 163 milionů recenzí. Je však důležité poznamenat, že toto hodnocení se vztahuje ke kompletní aplikaci Facebook, v rámci, které je funkce Facebook Events pouze jednou z mnoha integrovaných součástí. Samostatné hodnocení pouze událostí proto není dostupné. Aplikace Facebook jako celek byla stažena více než 10 miliardkrát, což svědčí o její globální rozšířenosti a širokém záběru uživatelů.

2.1.5 SWOT analýza

Tabulka 5 - SWOT analýza aplikace Facebook Events. zdroj: vlastní

Silné stránky	**Slabé stránky**
Velká uživatelská základna. Součástí silné platformy (Meta). Integrace s kalendáři, notifikace.	Chybí uložení uživatelských preferencí. Pouze součástí hlavní aplikace. Mapové zobrazení událostí.
Příležitosti	**Hrozby**
Osamostatnění aplikace. Lepší správa uživatelských preferencí.	Odchod uživatelů ke komunitním aplikacím. Ztráta mladší cílové skupiny. Přesycení funkcemi v hlavní aplikaci.

2.1.6 Zhodnocení

Facebook Events představuje funkčně bohatý nástroj pro správu a objevování veřejných i soukromých událostí. Mezi její klíčové přednosti patří robustní vyhledávání, které umožňuje filtrovat události podle polohy, času nebo relevance, dále jednoduché a přehledné uživatelské rozhraní, které je konzistentní napříč zařízeními. Aplikace těží z napojení na hlavní platformu Facebook, což jí zajišťuje velkou uživatelskou základnu a možnost využití online komunikačního kanálu prostřednictvím komentářů a příspěvků v rámci události. Nechybí ani notifikační systém a mapové podklady zobrazující lokaci události.

Na druhé straně však aplikace postrádá pokročilejší funkce, jako je například personalizace typů událostí na základě zájmů uživatele nebo interaktivní mapa s přehledem všech událostí v okolí. Tyto limity mohou omezit uživatelskou zkušenost při aktivním vyhledávání nových akcí dle individuálních preferencí.

2.2 Presentie

Presentie je moderní aplikace zaměřená na objevování a správu událostí v reálném světě s důrazem na osobní zapojení a přehlednost. Za vývojem stojí česká společnost Presentmania s. r. o., která ji provozuje pod vlastní značkou. Aplikace je dostupná jak jako mobilní aplikace pro Android a iOS, tak i ve formě webové aplikace, čímž rozšiřuje možnosti přístupu pro různé typy uživatelů. Aplikace si klade za cíl usnadnit organizaci i účast na událostech v osobním i komunitním životě, přičemž zdůrazňuje jednoduchost, soukromí a přirozené propojení lidí mimo tradiční sociální sítě.

2.2.1 Cílová skupina a účel aplikace

Presentie cílí především na jednotlivce, skupiny přátel, školy, organizátory menších komunitních akcí a kulturní nadšence, kteří chtějí efektivně plánovat a sdílet události ve svém okolí. Aplikace je vhodná pro neformální prostředí, kde je klíčová přehlednost, jednoduché přizvání a potvrzení účasti. Účelem aplikace je vytvořit prostor pro osobní propojení a reálné setkávání, bez rušivých vlivů velkých sociálních sítí. Díky přehlednému a soukromí respektujícímu přístupu si aplikace nachází místo u uživatelů, kteří hledají nástroj pro sdílení událostí bez zahlcení.

2.2.2 Hlavní funkce

První klíčovou funkcí aplikace Presentie je vyhledávání a objevování událostí v okolí. Uživatelé si mohou jednoduše zobrazit přehled akcí, které se konají v jejich městě nebo regionu, a to prostřednictvím přehledného seznamu. Nejčastějšími typy událostí jsou společenské hry, komunitní setkání a volnočasové aktivity.

Aplikace umožňuje přizpůsobitelné filtrování událostí podle místa, času nebo kategorie. Díky tomu si každý uživatel může zobrazit pouze ty akce, které jsou pro něj skutečně relevantní, jako například události konané tento týden nebo pouze akce v konkrétním městě.

Presentie podporuje rezervaci i nákup vstupenek přímo v aplikaci, čímž zjednodušuje proces přihlášení na akci. Platba může proběhnout pohodlně online, a vstupenka je pak dostupná v profilu uživatele bez nutnosti tisku nebo přeposílání e-mailem.

Pro organizátory akcí nabízí aplikace přístup do webového administrativního rozhraní, které umožňuje vytvářet nové události, upravovat jejich obsah, sledovat zájem účastníků a spravovat vlastní databázi kontaktů. Toto rozhraní je přehledné a vhodné i pro uživatele bez technických znalostí.

Každá událost má svůj vlastní detail, který obsahuje důležité informace o akci jako je čas, místo, popis, cenu vstupu a další podrobnosti. Tento detail je dostupný jak v mobilní aplikaci, tak na webové verzi, a slouží jako hlavní informační stránka pro účastníky.

Aplikace je dostupná ve formě mobilní aplikace i webového rozhraní, což zajišťuje její snadnou dostupnost jak pro běžné uživatele na cestách, tak pro organizátory, kteří spravují obsah z počítače (46; 47).

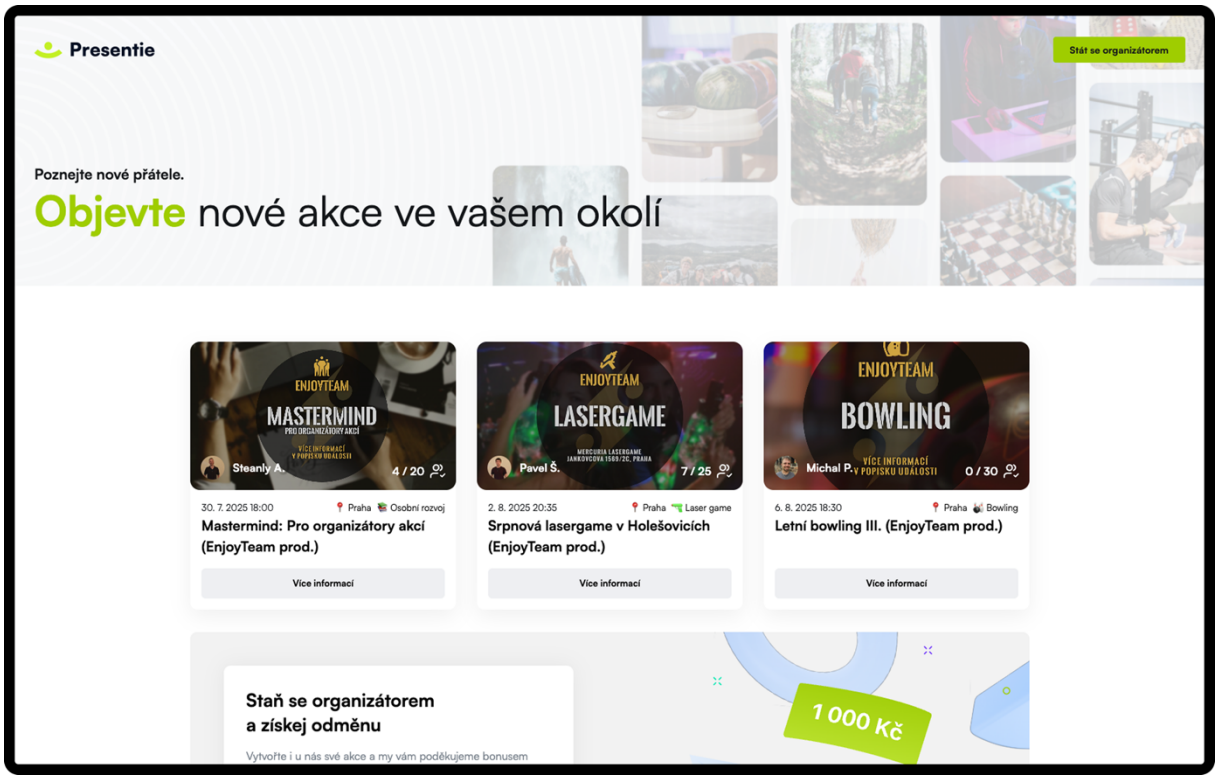
2.2.3 Uživatelské rozhraní a UX

Webové rozhraní aplikace Presentie je navrženo velmi jednoduše a přehledně, s důrazem na rychlý přístup k událostem bez rušivých prvků. Úvodní stránka zobrazuje slogan „Objevte nové akce ve vašem okolí“ a hned pod ním přehled dostupných akcí ve formě kartiček, z nichž každá zobrazuje název události, datum, místo, kategorii, počet přihlášených účastníků a základní popis.

Uživatelské rozhraní je orientováno na pasivní prohlížení akcí, nabízí tlačítko „Více informací“, které vede na detail události. Z hlediska designu využívá moderní a čisté prvky, s dobře čitelným fontem, ikonami a decentními barvami, přičemž důraz je kladen na vizuální náhledy událostí.

Funkcionalita webové verze je omezenější oproti mobilní aplikaci, jelikož chybí například pokročilé filtry nebo správa profilu. Pro organizátory je však na webu k dispozici samostatné rozhraní pro přidávání a správu událostí.

Celkově působí webová verze jako doplněk k mobilní aplikaci, vhodný pro rychlé nahlédnutí na nadcházející akce, prozkoumání detailu události a případné koupi vstupenek (46).



Obrázek 4 - Webové uživatelské rozhraní aplikace Presentie, zdroj: (46)

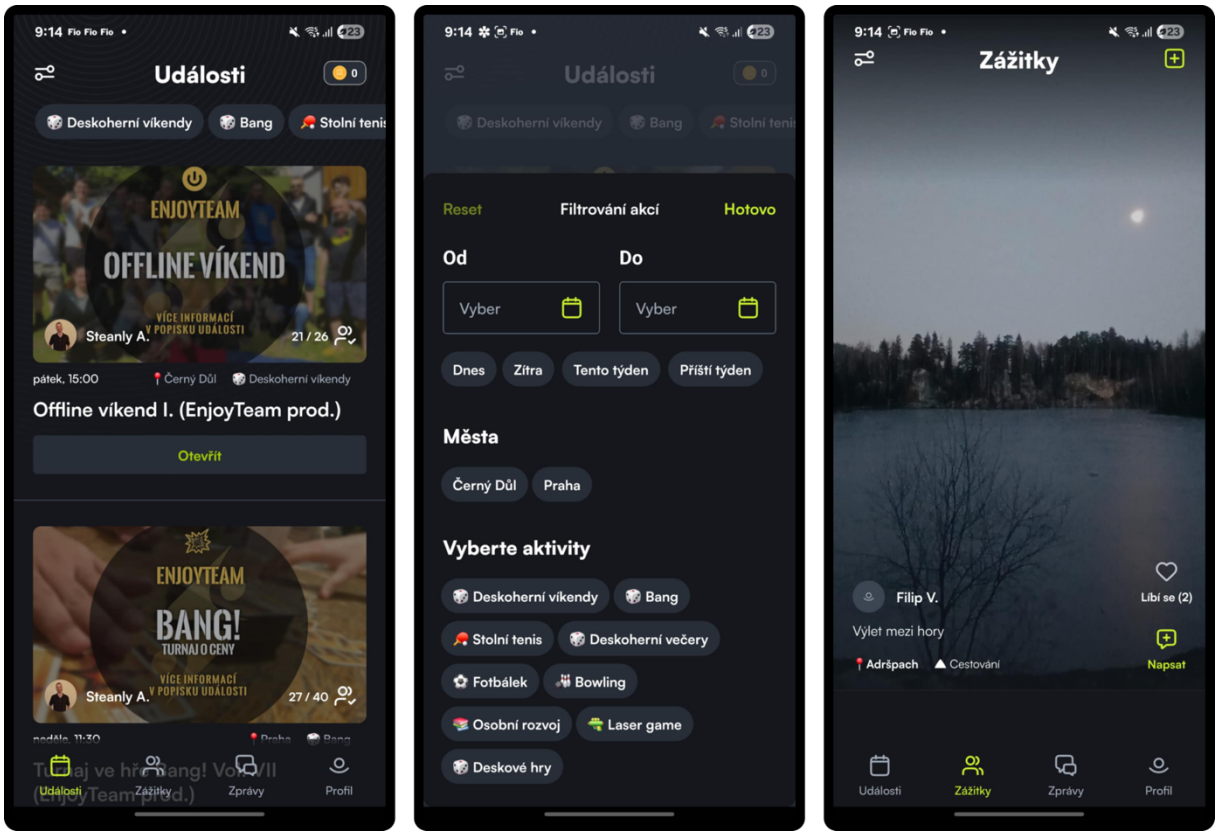
Mobilní aplikace Presentie nabízí kompletní uživatelské rozhraní určené pro běžné uživatele, kteří chtějí procházet, filtrovat a sledovat události ve svém okolí. Navigace je intuitivně řešena prostřednictvím spodního panelu se čtyřmi hlavními sekcemi: Události, Zážitky, Zprávy a Profil. Úvodní obrazovka zobrazuje aktuální nebo nadcházející akce formou přehledných karet, které obsahují název, místo, čas konání, kategorii a počet přihlášených účastníků.

Jednou z hlavních funkcí aplikace je pokročilá filtrace událostí, která umožňuje uživatelům rychle vyhledat akce podle data (např. dnes, zítra, tento týden), města a konkrétní aktivity. Filtrování je dostupné přímo z horní části obrazovky a probíhá pomocí jednoduchých ovládacích prvků a štítků, které se dají snadno kombinovat. Díky tomu je možné snadno zobrazit např. jen akce typu „Bowling v Praze příští týden“ bez nutnosti složitého procházení.

Sekce Zážitky přináší komunitní rozměr aplikace – zde mohou uživatelé sdílet své fotografie nebo dojmy z navštívených akcí, a zároveň si prohlížet příspěvky ostatních. Každý příspěvek lze označit jako „Líbí se“ nebo okomentovat, čímž aplikace podporuje sdílení zážitků i mimo samotné události.

Další funkce zahrnují přehled zpráv, který umožňuje komunikaci mezi uživateli a organizátory, a také profilovou sekci, kde má uživatel přístup ke svému přehledu plánovaných akcí a nastavení účtu. Oproti webové verzi aplikace, která je spíše statická, nabízí mobilní

aplikace plnohodnotný interaktivní zážitek včetně potvrzení účasti a přímé práce s událostmi (47).



Obrázek 5 - Mobilní uživatelské rozhraní aplikace Presentie, zdroj: (47)

2.2.4 Uživatelská hodnocení

Aplikace Presentie má v Obchodě Google Play hodnocení 4,1 z 5 hvězd na základě 17 uživatelských recenzí. Celkový počet stažení překročil 1 000 instalací, což ukazuje na počáteční zájem, ale stále omezený dosah. Uživatelé hodnotí aplikaci převážně pozitivně, přičemž chválí především originální nápad, přehledné zpracování a potenciál aplikace spojovat lidi skrze lokální události. Negativní komentáře se nejčastěji týkají technických problémů či drobných chyb ve funkčnosti. Významnou výzvou, kterou někteří uživatelé zmiňují, je nízký počet aktivních organizátorů a událostí, což může negativně ovlivnit celkový uživatelský zážitek, zejména v méně aktivních regionech (47).

2.2.5 SWOT Analýza

Tabulka 6 - SWOT analýza aplikace Presentie. zdroj: vlastní

Silné stránky	**Slabé stránky**
Originální komunitní koncept. Komunitní přehled zážitků.	Malá uživatelská základna. Mapové zobrazení událostí. Chybí uložení uživatelských preferencí.
Příležitosti	**Hrozby**
Růst zájmu o lokální a komunitní akce. Funkce a monetizace pro organizátory.	Silná konkurence zavedených platforem. Omezené šíření bez marketingu. Nízká aktivita uživatelů, organizátorů.

2.2.6 Zhodnocení

Aplikace Presentie představuje zajímavý a promyšlený koncept, který se zaměřuje na budování komunit skrze lokální události, zejména ve formě společenských her a volnočasových aktivit. Tento důraz na osobní propojení uživatelů mimo digitální svět ji odlišuje od větších, univerzálních platforem. Významným přínosem je i funkce uživatelského feedu, kde mohou účastníci sdílet fotografie a zážitky z uskutečněných akcí, čímž se posiluje komunitní aspekt. Dalším pozitivním prvkem je integrovaný chat, který umožňuje snadnou komunikaci mezi účastníky i organizátory přímo v aplikaci.

Navzdory těmto kvalitám aplikace zatím nenaplňuje svůj plný potenciál, především kvůli nízkému počtu aktivních uživatelů a organizátorů. Tento faktor omezuje nabídku událostí a snižuje atraktivitu platformy pro nové uživatele. Přitom právě zde by mohl být prostor pro další rozvoj a monetizaci, a to například prostřednictvím placených nástrojů pro organizátory, propagace událostí nebo prodeje vstupenek. Celkově lze konstatovat, že Presentie stojí na pevných základech s dobře navrženým uživatelským rozhraním a jasně definovaným zaměřením, ale pro širší uplatnění bude klíčové rozšíření její uživatelské základny a posílení viditelnosti na trhu.

2.3 Liiiv

Liiiv je mobilní aplikace vyvinutá společností Creative Dock Czech s.r.o., která usiluje o rychlé a zábavné objevování lokálních událostí, míst a tvůrců. Spuštěna začátkem roku 2025, funguje pro Android i iOS, a jejím hlavním cílem je načasovat a doporučit aktivity, které uživatele autenticky „osloví“. Aplikace nabízí funkčnost typu swipe, pomocí které si uživatelé vybírají zajímavé události na základě osobního vkusu, a to od pub kvízů, přes open-mic večery nebo yoga vína, na které následně mohou rezervovat vstupenky, stoly či ukládat oblíbené akce.

2.3.1 Cílová skupina a účel aplikace

Aplikace Liiiv cílí především na mladší generaci uživatelů, zejména ve věku 18–35 let, kteří hledají zábavu, inspiraci a zážitky ve svém okolí. Typickým uživatelem je člověk, který chce objevovat nové podniky, netradiční akce nebo kulturní zážitky, aniž by musel procházet dlouhé seznamy. Svou formou a jazykem aplikace oslovuje zejména městské publikum s otevřeným přístupem ke spontánním zážitkům a sociálnímu kontaktu.

Účelem aplikace je zjednodušit a zatraktivnit objevování lokálních akcí a zajímavých míst, a to pomocí personalizovaného výběru a vizuálně přívětivého rozhraní. Liiiv propojuje uživatele s akcemi prostřednictvím intuitivního posouvání (swipování), čímž spojuje principy známé ze seznamovacích aplikací s objevováním kulturního a společenského dění. Vedle toho umožňuje i přímé rezervace a ukládání akcí, čímž posiluje jejich přístupnost a podporuje účast.

2.3.2 Hlavní funkce

Aplikace Liiiv využívá pro procházení událostí systém swipování, inspirovaný populárním principem známým například z aplikace Tinder. Uživatel přejíždí kartami jednotlivých akcí do všech čtyř stran podle toho, zda ho daná událost zaujme, nezaujme nebo chce vidět její detail. Tento interaktivní způsob výběru podporuje rychlé rozhodování a zábavné objevování nových zážitků v okolí.

Při zájmu o konkrétní událost má uživatel možnost přejít na rezervaci vstupenek nebo míst. Po kliknutí na příslušné tlačítko je přesměrován na externí webovou stránku organizátora, kde může nákup dokončit. Aplikace tak usnadňuje proces rezervace bez zbytečného hledání mimo platformu.

Události, které uživatele zaujmou, si může ukládat mezi oblíbené. Tím získá vlastní přehled událostí, ke kterým se může kdykoliv vrátit, aniž by je musel znovu vyhledávat. Funkce slouží jako nástroj pro plánování a snadné sledování osobních tipů.

Zobrazování událostí je v aplikaci personalizováno podle uživatelských preferencí. Každý si může při registraci nebo v profilu zvolit tematické okruhy, které ho zajímají, jako například kultura, hry, hudba nebo gastronomie, a aplikace podle těchto voleb sestavuje obsah ve feedu.

Liiiv dále umožňuje objevovat zajímavá místa a tvůrce, kteří pořádají akce. Každé místo má vlastní profil s popisem a seznamem aktuálních i budoucích událostí, čímž aplikace propojuje uživatele nejen s jednorázovými akcemi, ale i s lokální scénou.

2.3.3 Uživatelské rozhraní a UX

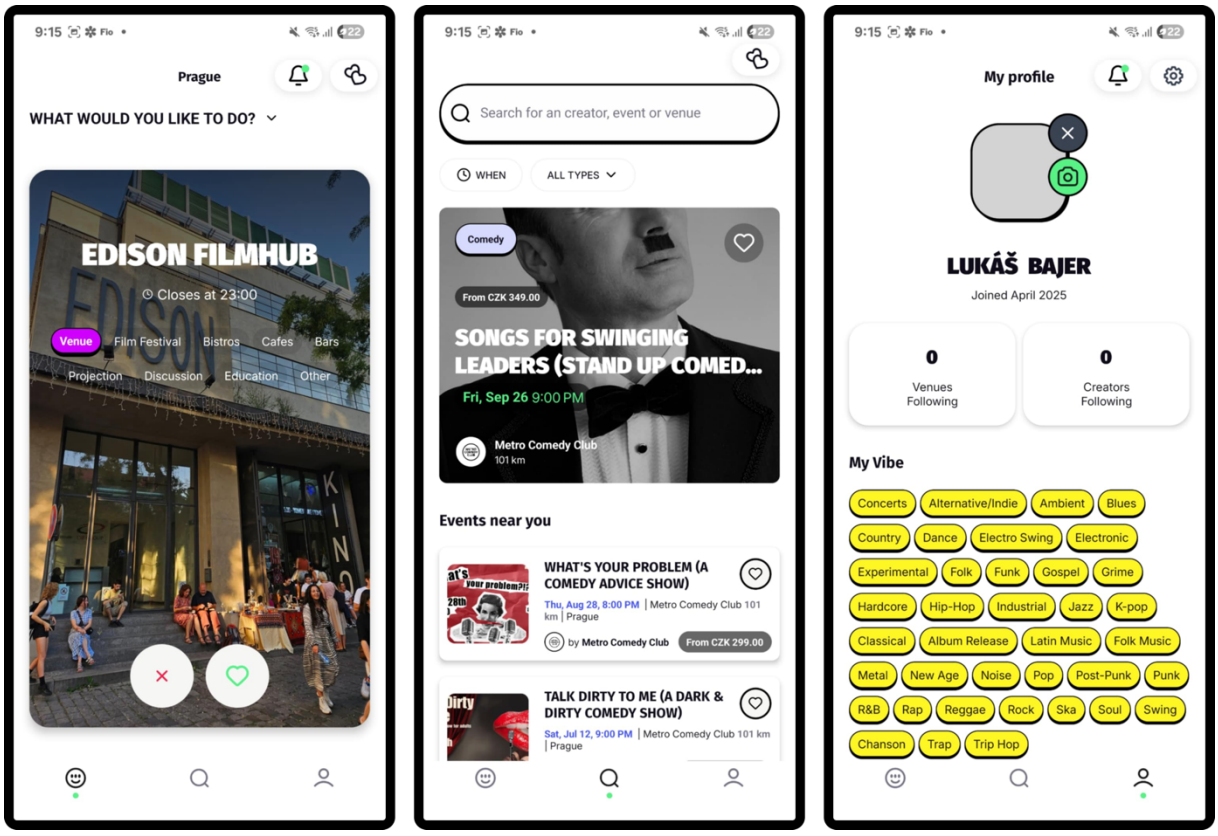
Mobilní aplikace Liiiv je navržena s důrazem na jednoduchost, přehlednost a moderní vizuální styl. Rozhraní je čisté, vizuálně kontrastní a pracuje s velkým množstvím prostoru, díky čemuž působí vzdušně a příjemně. Barevná paleta je minimalistická, s výraznými prvky (např. fialové štítky kategorií, žluté bubliny s preferencemi), které slouží k navigaci i zvýraznění obsahu.

Domovská obrazovka nabízí uživatelům možnost „What would you like to do?“, kde se zobrazují jednotlivá místa či události formou velkých karet, které lze pomocí swipování rychle procházet. Tento prvek je klíčovou součástí UX a přináší zábavný a přirozený způsob interakce s aplikací a zároveň připomíná mechanismus známý z Tinderu.

Vyhledávací sekce umožňuje uživatelům filtrovat akce podle data a typu a zároveň nabízí seznam událostí v okolí, které jsou vizuálně rozděleny podle žánru či tématu. Události jsou prezentovány formou přehledných náhledů s datem, místem konání a případnou cenou, což podporuje rychlou orientaci v nabídce.

Profilová obrazovka nabízí přehled o sledovaných organizátorech a místech, a zároveň zobrazuje uživatelovy preferované zájmy ve formě žlutých bublin pod nadpisem My Vibe. Tyto zájmy (např. Jazz, Indie, Stand-up, Folk Music) lze snadno přidávat nebo měnit, a tvoří základ pro personalizované doporučování obsahu v aplikaci.

Ovládací prvky aplikace jsou umístěny intuitivně. Spodní lišta obsahuje ikony pro přechod mezi hlavními sekcemi aplikace jako swipe stránka, feed s filtry a události a profil s preferencemi uživatele. přičemž celkové rozložení podporuje snadné ovládání jednou rukou. Aplikace působí moderně a mladistvě nejen vzhledem, ale také formou komunikace, která využívá neformální tón a hravý jazyk.



Obrázek 6 - Mobilní uživatelské rozhraní aplikace Liiiv. zdroj: vlastní

2.3.4 SWOT analýza

Silné stránky	**Slabé stránky**
Inovativní swajpovací systém. Stylový vizuální design. Personalizace obsahu podle preferencí. Možnost objevovat i tvůrce a místa.	Funguje aktuálně pouze v Praze. Omezené možnosti interakce mezi organizátory a uživateli. Malá uživatelská základna.
Příležitosti	**Hrozby**
Rozšíření do dalších regionů. Využití preferencí pro doporučovací systém.	Velká konkurence zavedených platforem. Závislost na kvalitě a kvantitě obsahu od organizátorů.

Tabulka 7 - SWOT analýza aplikace Liiiv. zdroj: vlastní

2.3.5 Hodnocení uživatelů

Aplikace má v Obchodě Google Play hodnocení 3,4 z 5 hvězd, a to na základě 10 uživatelských recenzí. Celkový počet stažení přesahuje 1 000 instalací, což odpovídá časné fázi vývoje a uvádění platformy na trh. Ve veřejně dostupných recenzích převládají kritické komentáře, které se nejčastěji zaměřují na technické nedostatky aplikace. Uživatelé zmiňují problémy se stabilitou, načítáním obsahu nebo chybami v rozhraní. Dalším častým tématem

je omezený geografický dosah, protože aplikace v současné době funguje pouze pro akce konané v Praze. Tato skutečnost výrazně omezuje uživatelskou zkušenost pro zájemce z jiných regionů

2.3.6 Zhodnocení

Aplikace Liiiv představuje na trhu s událostmi a kulturním obsahem výrazně odlišný koncept, který zaujme především svým vizuálně atraktivním zpracováním a moderním přístupem k uživatelskému zážitku. Stylová grafika a výrazný jazyk ji odlišují od běžných platforem a jasně cílí na mladší generaci městských uživatelů. Vedle vzhledu zaujme také inovativními funkcemi, jako je výběr událostí formou swipování nebo nastavení osobních preferencí zájmů.

Významným rysem aplikace je, že nezobrazuje pouze události, ale také místa a tvůrce, čímž propojuje kulturní scénu z obou stran, kde umožňuje podnikům hledat vystupující umělce, a naopak tvůrcům nabízí nové příležitosti. Tím se Liiiv staví nejen jako nástroj pro objevování, ale také jako platforma pro propojení a spolupráci.

Za hlavní omezení lze považovat to, že aplikace aktuálně funguje výhradně v Praze, a tedy nemá pro většinu uživatelů mimo hlavní město praktické využití. Vzhledem k tomu, že se stále nachází v rané fázi vývoje, je rovněž patrná nízká uživatelská základna, což může negativně ovlivnit dynamiku obsahu i motivaci k opakovanému používání. Přesto má Liiiv díky svému originálnímu pojetí a designu potenciál stát se výrazným hráčem na poli kulturních a komunitních aplikací, pokud dojde k dalšímu rozšíření a technickému vylepšení.

3 PRAKTICKÁ ČÁST

Cílem praktické části této diplomové práce je návrh a implementace mobilní aplikace pro operační systém Android, která umožňuje uživatelům přehledně a efektivně získávat informace o kulturních a společenských událostech ve svém okolí. Aplikace je navržena s důrazem na moderní uživatelské prostředí, přehlednost a pohodlné ovládání. Její hlavní funkcí je zobrazování aktuálních událostí, jako jsou koncerty, večírky, výstavy a další akce pořádané kluby, podniky či kulturními centry, a to na základě polohy uživatele i dalších volitelných filtrů.

Na základě dříve provedeného průzkumu a analýzy dostupných technologií byly pro jednotlivé části systému zvoleny následující nástroje a služby. Pro vývoj mobilní aplikace byl vybrán framework Flutter, a to z důvodu jeho schopnosti vytvářet responzivní a vizuálně moderní uživatelská rozhraní napříč platformami. Backendová část systému byla realizována v jazyce Java s využitím frameworku Spring, který byl zvolen pro svou robustnost, široké možnosti rozšíření a spolehlivou integraci s databází PostgreSQL.

Pro správu a distribuci obrazových dat byla vybrána služba Cloudflare Images, jež poskytuje cenově výhodné úložiště s vestavěnou CDN. Mapové podklady jsou zajištěny prostřednictvím OpenStreetMap a MapBox s využitím odpovídajícího API. Pro doručování notifikací, například o nových událostech v okolí uživatele, bylo vybráno řešení Firebase Cloud Messaging, a to díky jeho bezplatnému modelu a nativní integraci s Flutterem.

3.1 Použité technologie a software

Výběr správných technologií má zásadní vliv na kvalitu, udržitelnost a rozšiřitelnost aplikací. Vhodně zvolený technologický základ umožňuje rychlejší vývoj, snadnější ladění chyb, lepší výkon i bezpečnost. V této kapitole budou představeny konkrétní nástroje a technologie použité při vývoji této aplikace.

3.1.1 Vývojové prostředí Visual Studio Code

Visual Studio Code je výkonné, kódem orientované vývojové prostředí, které bylo navrženo tak, aby co nejvíce usnadnilo psaní všech různých typů aplikací. VS Code je multiplatformní nástroj, který běží na Windows, Linuxu a macOS, a je dostupný jako open-source software zdarma.

Významnou vlastností tohoto nástroje je podpora pro velké množství programovacích jazyků, integrovaný debugger a nativní integrace verzovacího systému Git. Na rozdíl od plně integrovaných vývojových prostředí, jako je například plnohodnotné Visual Studio od Microsoftu, je VS Code lehčí, otevřenější a vhodný pro vývojáře pracující na různých operačních systémech i s různými technologiemi.

Mezi jeho hlavní přednosti patří:

- rozšířená editace kódu včetně IntelliSense, vyhledávání symbolů, rychlého přechodu na definice typů apod
- univerzální použití bez závislosti na konkrétním vývojovém prostředí
- možnost rozšíření o nové jazyky, nástroje, ladicí prostředí a další funkcionalitu

Přestože VS Code neposkytuje grafické návrháře uživatelského rozhraní ani zabudovanou kompilaci pro všechny jazyky, nabízí flexibilní a výkonné prostředí pro psaní kódu, které lze přizpůsobit různým vývojovým scénářům (48).

V rámci této diplomové práce sloužilo Visual Studio Code jako primární nástroj při vývoji frontendové části aplikace. Byl využit ke psaní a správě kódu, ladění aplikace pomocí integrovaného debuggeru a také ke spouštění emulátorů a testování funkčnosti na různých zařízeních.

3.1.2 Vývojové prostředí IntelliJ Idea

IntelliJ IDEA je pokročilé integrované vývojové prostředí, navržené s důrazem na inteligentní podporu vývojářské práce. Jeho cílem je zjednodušit a automatizovat běžné vývojové úkoly, minimalizovat chyby a zvýšit produktivitu. IntelliJ přináší širokou sadu funkcí, které přesahují běžné možnosti jiných editorů jako např. chytrou nápovědu při psaní kódu, automatické refaktoringy, kontrolu kódu v reálném čase a efektivní nástroje pro navigaci a vyhledávání.

IDE nabízí podporu pro velké množství jazyků jako Java, Scala, Groovy, SQL, JavaScript a další. Díky své rozsáhlé integraci se jedná o nástroj vhodný především pro vývoj webových, mobilních a podnikových aplikací. IntelliJ disponuje bohatou podporou moderních frameworků, jako například Spring, Hibernate, Play, Grails, JSF, Struts a dalších. Zároveň nabízí nativní integraci s nástroji jako Maven, Gradle, Git, a s databázovými systémy (49).

Mezi významné vlastnosti IntelliJ IDEA patří:

- chytré doplňování kódu s podporou kontextového řetězení a vyhledávání
- analýza kódu za běhu s možností upozornění na chyby a návrhy vylepšení
- pokročilé refaktoringy, včetně změn signatur, typů, extrakcí tříd nebo konstant
- pokročilá navigace v projektu pomocí zkratek, symbolů a rychlého hledání
- integrované databázové a modelovací nástroje (např. UML editor, správce tabulek)

- podpora verzovacích systémů jako například Git
- plná integrace s buildovacími nástroji jako Maven, Gradle, Ant apod.

V rámci této diplomové práce bylo prostředí IntelliJ IDEA využito pro vývoj backendové části aplikace postavené na frameworku Spring. Díky své nativní podpoře Springu, možnostem pohodlné správy závislostí a přehlednému rozhraní umožňujícímu efektivní práci s kontrolou kódu, byl IntelliJ ideálním nástrojem pro návrh a implementaci aplikační logiky, REST API a připojení k databázi. Funkce jako inteligentní doplňování kódu, refaktoringy a automatická kontrola syntaxe zásadně zefektivnily vývojový proces a pomohly minimalizovat chybovost (49).

3.1.3 Testovací nástroj Postman

Postman je multiplatformní nástroj určený pro vývoj, testování a dokumentaci API. Původně vznikl jako jednoduchý klient pro testování HTTP požadavků, dnes se jedná o komplexní platformu pro práci s API, dostupnou ve formě desktopové i webové aplikace. Postman umožňuje vývojářům sestavovat, odesílat a analyzovat API požadavky, a tím efektivně ladit komunikaci mezi systémy (50).

Mezi jeho klíčové funkce patří:

- přehledné uživatelské rozhraní pro sestavení a odesílání HTTP požadavků
- sledování odpovědí serveru a zobrazení návratových hodnot, hlaviček a stavových kódů
- správa kolekcí, prostředí a předdefinovaných požadavků pro opakované testování
- podpora automatizovaného testování pomocí skriptů a CI/CD integrací

V této práci byl Postman využit jako klíčový nástroj pro testování REST API. Sloužil k ověřování správného fungování backendových endpointů, testování různých vstupních dat a sledování odpovědí serveru. Díky intuitivnímu rozhraní a podpoře správy testovacích scénářů bylo možné rychle ladit a iterovat jednotlivé endpointy, a tak testovat jejich stabilitu a správnou funkčnost (50).

3.1.4 Docker

Docker je open-source nástroj určený k automatizaci nasazení aplikací do kontejnerů. Vyvíjený společností Docker, Inc., poskytuje lehké a rychlé prostředí pro spouštění kódu i efektivní workflow pro jeho přenos z vývojového prostředí do testování a produkce. Docker

přidává vlastní vrstvu nad virtualizované kontejnery, přičemž se zaměřuje na jednoduchost, rychlost a oddělení vývoje od provozu.

Kontejnery spuštěné pomocí Dockeru jsou extrémně rychlé a díky absenci hypervizoru umožňují efektivnější využití systémových prostředků. Docker používá model copy-on-write, který výrazně urychluje úpravy aplikací a tvorbu docker image. Vývojáři se tak mohou soustředit na aplikace běžící uvnitř kontejnerů, zatímco administrátoři spravují samotné kontejnery, čímž se zvyšuje konzistence mezi vývojovým a produkčním prostředím.

Docker rovněž podporuje mikroslužbovou architekturu, kdy každá služba nebo aplikace běží samostatně v odděleném kontejneru. To usnadňuje škálování, ladění i správu distribuovaných systémů. Hlavní předností Dockeru je zkrácení vývojového cyklu, kód lze rychle sestavit, testovat, nasadit a sdílet mezi členy týmu i prostředími.

V této práci byl Docker využit k nasazení backendové aplikace a databáze ve formě samostatných kontejnerů. Pomocí Docker sítě byla zajištěna jejich vzájemná komunikace bez potřeby složité konfigurace. Tento přístup umožnil rychlé nasazení, přenositelnost prostředí a jednoduché testování v odděleném, izolovaném prostředí (52).

3.1.5 Spring framework

Spring Framework je široce používaný open-source framework pro vývoj aplikací v jazyce Java. Často bývá označován jako lehký framework, ačkoliv toto označení se nevztahuje na jeho velikost, ale spíše na filozofii nízké závislosti a minimálního zásahu do aplikačního kódu. Spring umožňuje stavět různé typy Java aplikací, a to od samostatných konzolových nástrojů až po webové a enterprise řešení bez nutnosti vázat se na konkrétní prostředí nebo architekturu.

Základním principem Springu je tzv. Inversion of Control (IoC), známý také jako Dependency Injection. Místo toho, aby si komponenty vytvářely své závislosti samy (např. pomocí new), jsou jim tyto závislosti předávány zvenčí, obvykle při spuštění aplikace. Tento přístup přispívá k větší modularitě, testovatelnosti a čistotě kódu.

Spring se postupně rozrostl o množství dalších modulů a projektů (např. Spring Data, Spring Security, Spring Boot), které dále rozšiřují jeho možnosti a zjednodušují vývoj rozsáhlých a škálovatelných aplikací. Díky své flexibilitě, komunitní podpoře a bohaté sadě nástrojů patří Spring mezi nejvýznamnější technologie v ekosystému Java.

V rámci této diplomové práce byl Spring Framework využit jako základní stavební prvek pro vývoj backendové části aplikace. Konkrétně byl použit modul Spring Boot, který výrazně zjednodušuje konfiguraci a spouštění aplikace. Pomocí Springu byla vytvořena REST

API vrstva, řešeno mapování entit, validace vstupů a propojení s databází. Díky principu IoC bylo možné snadno strukturovat aplikaci do jednotlivých vrstev a oddělit odpovědnosti mezi komponentami (53).

3.1.6 Java

Java je objektově orientovaný programovací jazyk, virtuální stroj (JVM) a vývojová platforma, která umožňuje vytvářet přenositelné, robustní a bezpečné aplikace. Při vysvětlování pojmu Java je důležité rozlišovat mezi jazykem Java, Java Virtual Machine (JVM) a Java platformou, každá z těchto částí hraje v ekosystému specifickou roli.

Programovací jazyk Java je moderní, objektově orientovaný jazyk se syntaxí podobnou jazyku C. Byl navržen tak, aby umožňoval psaní výkonného, a přitom jednoduchého kódu. Jeho návrháři se záměrně vyhnuli složitostem, které zatěžují jiné objektově orientované jazyky, jako je C++. Díky svému čistému návrhu a novým funkcím, jako například generickým typům v Javě 5.0, se jazyk stal velmi oblíbeným mezi vývojáři, kteří oceňují jeho stabilitu, čitelnost a schopnost vytvářet bezchybný kód.

Java Virtual Machine je jádrem každé instalace Javy. Je odpovědná za interpretaci a vykonávání bajtkódu, do kterého je každý Java program po kompilaci převeden. Díky tomu je Java platformně nezávislá, takže programy nejsou vázány na konkrétní operační systém, ale pouze na to, zda daný systém podporuje JVM. Moderní JVM používají technologii just-in-time kompilace, která převádí bajtkód do nativního strojového kódu při běhu, čímž zvyšuje výkon aplikací.

Java platforma představuje předdefinovanou sadu tříd, které jsou dostupné v každé instalaci Javy a tvoří základní stavební bloky aplikací, a to od vstupu/výstupu až po uživatelské rozhraní, síťovou komunikaci či bezpečnost. Důležité je, že Java není operační systém, ale alternativní vývojová platforma, která poskytuje rozhraní srovnatelná s těmi, jež obvykle definuje samotný OS.

Výhodou je, že aplikace napsaná v Javě může běžet na jakémkoli zařízení s podporou Java platformy, bez nutnosti přepisovat ji pro různá prostředí (Windows, macOS, Linux). Tato vlastnost vedla ke známému mottu: „Write once, run anywhere“, které vystihuje hlavní přínos jazyka Java neboli nezávislost na cílovém operačním systému (54).

3.1.7 PostgreSQL

PostgreSQL je výkonný, spolehlivý a škálovatelný open-source relační databázový systém, který je často označován jako nejpokročilejší open-source databáze na světě. Díky podpoře vlastností ACID (atomicita, konzistence, izolace, trvanlivost), pokročilé správě

konkurence a silnému důrazu na datovou integritu je PostgreSQL vhodný nejen pro běžné aplikace, ale i pro podniková nasazení s vysokými nároky.

PostgreSQL se neomezuje pouze na základní databázovou funkcionalitu, ale jedná se o celý ekosystém nástrojů, rozšíření a integrovaných jazyků, jako je PL/pgSQL, Python, Java nebo Bash. Díky rozsáhlé komunitě a otevřenému vývoji pod licencí BSD jej může používat i rozšiřovat kdokoli, bez omezení na typ projektu. Systém je extrémně flexibilní, podporuje např. triggery, pohledy, materializované pohledy, partitioning, plánování úloh i specializované datové typy nebo rozšíření jako například GIS.

Technicky dokáže jedna instance PostgreSQL pojmout více než 4 miliardy databází, každou s miliardou tabulek o velikosti až 32 TB. Tabulky mohou mít až 1 600 sloupců, každý až 1 GB, což potvrzuje schopnost PostgreSQL zpracovávat extrémní objemy dat. Díky výkonnému just-in-time zpracování a přizpůsobitelnosti různým systémům je PostgreSQL vhodný pro jakékoli prostředí, a to od cloudových řešení po zařízení jako Raspberry Pi.

V této diplomové práci byl PostgreSQL využit jako primární databázový systém pro ukládání aplikačních dat. Byl spuštěn jako samostatný kontejner v rámci Docker infrastruktury a komunikoval přímo s backendovou aplikací vyvinutou ve Spring Frameworku (55).

3.1.8 Flyway

Flyway je populární open-source nástroj pro automatizované migrace databázového schématu. Umožňuje udržovat konzistentní verzi databáze prostřednictvím verzovaných SQL skriptů, které se aplikují v definovaném pořadí. Tím je zajištěna kontrola nad historií změn a možnost automatizace migrací v různých prostředích. Flyway dále podporuje automatické zálohy, minimalizaci rizika nechtěné ztráty dat a bezproblémové integrování do ekosystému Java, např. přes Spring či CI/CD nástroje

V této práci byl Flyway použit k verzování databázového schématu PostgreSQL. Změny databáze byly ukládány do SQL skriptů, které se při spuštění aplikace automaticky provedly. Díky propojení Flyway se Spring Bootem a Mavenem byla zajištěna automatická synchronizace databáze od vývoje až po produkci (56).

3.1.9 Flutter

Flutter je moderní open-source framework od společnosti Google určený pro vývoj nativních, responzivních a multiplatformních uživatelských rozhraní. Umožňuje tvorbu aplikací z jednoho společného kódu pro Android, iOS, web i desktopová prostředí (Windows, macOS, Linux). Google aktivně rozvíjí také nasazení Flutteru na vestavěných zařízeních, jako jsou Raspberry Pi nebo systémy v chytré domácnosti či automobilech.

Flutter využívá programovací jazyk Dart, který je objektově orientovaný a kompilovaný jak ahead-of-time pro maximální výkon, tak just-in-time pro rychlé zobrazení změn v kódu. Funkce hot reload umožňuje vývojářům okamžitě vidět změny v běžící aplikaci bez nutnosti restartu. Výsledné aplikace jsou kompilovány do nativního ARM kódu, mohou využívat výkon GPU a komunikovat s nativními API systémů prostřednictvím tzv. platform channels.

Flutter využívá deklarativní způsob tvorby uživatelského rozhraní, který zjednodušuje návrh a aktualizaci UI na základě aktuálního stavu dat. Uživatelské rozhraní je tvořeno pomocí widgetů, přičemž každý vizuální nebo funkční prvek (např. tlačítka, text, formuláře, animace) představuje widget. Pro vykreslování rozhraní využívá Flutter výkonný engine Skia, který se používá i v prohlížečích Google Chrome nebo Mozilla Firefox.

V této diplomové práci byl Flutter využit pro tvorbu frontendové mobilní aplikace cílené na platformu Android. Vývoj probíhal v prostředí macOS, což umožnilo pohodlné testování iOS i Android verzí aplikace. Díky Flutteru bylo možné efektivně navrhnout moderní a přizpůsobivé uživatelské rozhraní s jednotným základem, čímž se urychlil vývoj a zjednodušila správa kódu (57).

3.1.10 Programovací jazyk Dart

Dart je moderní objektově orientovaný programovací jazyk vytvořený společností Google. Slouží jako základní jazyk pro vývoj aplikací ve frameworku Flutter, a kromě toho se využívá také pro tvorbu webových a serverových aplikací. Dart má jednoduchou, čitelnou syntaxi podobnou jazykům jako Java, C# nebo Kotlin, a je tak vhodný pro začátečníky i zkušené vývojáře.

Díky ahead-of-time kompilaci se Dart převádí přímo do nativního kódu, což zajišťuje vysoký výkon výsledných aplikací. Zároveň podporuje just-in-time kompilaci, která výrazně urychluje vývoj, a to zejména díky funkci hot reload, která umožňuje okamžité zobrazení změn bez restartu aplikace. Dart se také vyznačuje reaktivním přístupem k tvorbě uživatelského rozhraní, které je možné psát přímo v kódu bez nutnosti používat dodatečné značkovací jazyky nebo vizuální návrháře. To zjednodušuje vývoj, sjednocuje kódovou základnu a umožňuje přesnější kontrolu nad chováním aplikace.

V této diplomové práci byl jazyk Dart využit při vývoji mobilní aplikace pomocí Flutteru. Celé uživatelské rozhraní i aplikační logika byly vytvořeny v Dart kódu, což umožnilo efektivní vývoj z jednoho zdrojového souboru a snadné testování (57).

3.2 Funkční a nefunkční požadavky

Funkční požadavky popisují konkrétní chování systému, tedy co má aplikace dělat.

- R1 – Uživatel se může zaregistrovat a přihlásit do systému.
- R2 – Aplikace zobrazí seznam aktuálních událostí ve scrolovatelném feedu.
- R3 – Aplikace zobrazí seznam aktuálních událostí na mapě.
- R4 – Aplikace umožnuje zobrazit detail události (popis, čas, místo, obrázek).
- R5 – Uživatel může filtrovat události podle času, místa, názvu a typu.
- R6 – Aplikace umožňuje vyhledávání událostí dle názvu.
- R7 – Aplikace umožnuje posílat denní notifikaci o relevantních událostech.
- R8 – Aplikace umožňuje změnit nastavení filtrovacích preferencí.
- R9 – Administrátor může vytvářet nové události prostřednictvím formuláře.
- R10 – Administrátor může upravovat a mazat již vytvořené události.

Nefunkční požadavky definují vlastnosti systému, jako je výkon, použitelnost, bezpečnost nebo dostupnost.

- R1 – Backend je implementován pomocí frameworku Spring.
- R2 – Backend komunikuje s databází PostgreSQL.
- R3 – Aplikace má moderní a uživatelsky přívětivé rozhraní vytvořené ve Flutteru.
- R4 – Rozhraní je optimalizováno pro běžná Android zařízení od verze Android 8.0.
- R5 – Systém je navržen modulárně a umožňuje budoucí rozšíření o další funkce.
- R6 – Administrace je dostupná pouze autorizovaným uživatelům.
- R7 – Obrázky událostí jsou ukládány a zobrazovány pomocí služby Cloudflare Images.
- R8 – Aplikace používá systém pro správu migrací databáze Flyway.
- R9 – JWT tokeny musí mít omezenou platnost a musí být zabezpečeně uloženy na klientovi.
- R10 – Systém neumožnuje vytvořit dva účty s totožným emailem nebo ičem

3.3 Databázový návrh

Databáze tvoří nedílnou součást téměř každé moderní softwarové aplikace, neboť zajišťuje strukturované uchovávání, správu a efektivní zpřístupnění dat napříč celým systémem. Z hlediska funkčnosti i bezpečnosti představuje databázová vrstva klíčový prvek infrastruktury, nejenže musí umožňovat rychlý a spolehlivý přístup k datům, ale zároveň musí chránit citlivé informace před neoprávněným přístupem.

V rámci tohoto projektu byla zvolena relační databáze PostgreSQL, která díky své robustnosti, výkonnosti a podpoře pokročilých datových typů představuje ideální řešení pro

rozsáhlejší aplikace. Pro práci s prostorovými a geografickými daty, jako je lokalizace událostí na mapě, bylo využito rozšíření PostGIS, které je nadstavbou PostgreSQL a přidává plnohodnotné geoprostorové funkce.

Databáze je nasazena v prostředí Docker a spolu s backendem běží ve stejné dockerové síti. Backendová část aplikace se k databázi připojuje přímo přes nativní protokol PostgreSQL (JDBC/Spring Data JPA) na příslušném portu. Konfigurace připojení je řešena přes environment proměnné a Docker Compose.

Tato kapitola podrobně popisuje návrh databázového schématu, klíčové entity, jejich vzájemné vazby a specifické konstrukce použité s ohledem na cíle aplikace.

3.3.1 Základní principy návrhu

Návrh databázového modelu reflektuje specifické požadavky systému zaměřeného na lokalizované vyhledávání, pokročilou filtraci a rychlý přístup k datům. Hlavním cílem bylo zajistit konzistenci dat, podporu prostorového vyhledávání a možnost efektivního dotazování pomocí funkcí s relevantními parametry.

Normalizace dat

Struktura databáze byla navržena v souladu se zásadami normalizace. Data jsou rozdělena do samostatných tabulek podle logických entit, mezi nimiž jsou definovány vazby pomocí cizích klíčů. Tento přístup minimalizuje redundanci a zajišťuje integritu uložených informací napříč celým systémem.

Práce s geografickými daty a rozšíření PostGIS

Pro podporu prostorových operací je v systému využito rozšíření PostGIS, které rozšiřuje databázový systém PostgreSQL o pokročilé geografické funkce. Díky tomu je možné efektivně pracovat s lokalizačními dotazy, jako je například určení vzdálenosti mezi dvěma body nebo výběr všech událostí v daném okruhu od zadané polohy. Ačkoli jsou geografické souřadnice míst a událostí v databázi uloženy jako dvojice hodnot latitude a longitude, rozšíření PostGIS umožňuje jejich dynamické využití prostřednictvím funkcí jako ST_MakePoint, ST_DWithin nebo ST_Distance. Tento přístup zajišťuje potřebnou flexibilitu a zároveň zachovává jednoduchost datového modelu.

Využití pokročilých funkcí pro komplexní dotazy

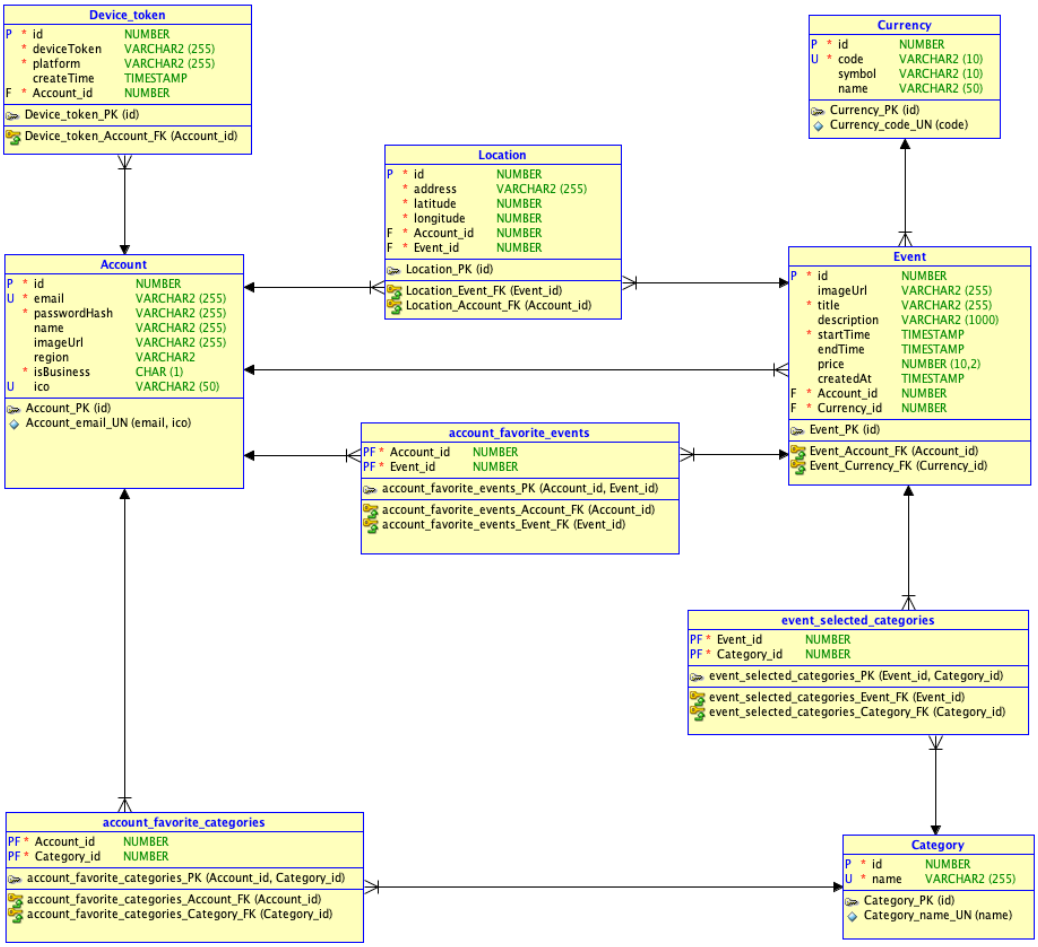
Systém podporuje filtrování a vyhledávání událostí podle více parametrů současně, včetně časového rozsahu, názvu, kategorie nebo geografické polohy. K tomu slouží kombinace klasických SQL dotazů a prostorových funkcí PostGIS. Tyto dotazy jsou navrženy

tak, aby umožňovaly dynamickou práci s uživatelskými vstupy a poskytovaly výsledky odpovídající reálnému kontextu.

3.3.2 Databázový model

Následující diagram znázorňuje úplný databázový model navržený pro tuto aplikaci. Vizuálně zobrazuje hlavní entity, jejich atributy a vzájemné vztahy, které budou podrobně popsány v následující kapitole. Model slouží jako přehledná reprezentace struktury databáze a usnadňuje pochopení logiky uložení dat i vzájemných vazeb mezi jednotlivými částmi systému.

Fyzická struktura databáze je v průběhu vývoje generována automaticky pomocí nástroje Hibernate DDL, na základě JPA anotací ve zdrojovém kódu. Pro produkční nasazení je ze skutečné databáze následně vygenerován DDL skript, který slouží jako vstup pro nástroj Flyway zajišťující verzování a správu migrací databázového schématu.



Tabulka 8 - Relační model. zdroj: vlastní

3.3.3 Popis entit a vazeb

Databázový model navržený pro tuto aplikaci odráží hlavní funkční části systému jako například správu uživatelů, podniků, lokalit a kulturních událostí. Každá tabulka představuje samostatnou entitu s jasně definovanou rolí, přičemž mezi jednotlivými entitami existují logické vazby vyjádřené pomocí cizích klíčů. Následující přehled popisuje strukturu všech tabulek, jejich atributy a účel, který v systému plní.

Tabulka Account v systému reprezentuje uživatelský účet a osahuje atributy související s autentizací a profilem. Každý účet jak podnikatelský, tak osobní využívají stejné atributy, na frontendu jsou poté zobrazovány v kontextu typu účtu. Tabulka je rovněž propojena s dalšími klíčovými entitami systému prostřednictvím množinových a jednosměrných vazeb.

Tabulka 9 - Popis tabulky Account. zdroj: vlastní

Název sloupce / vztahu	**Datový typ / Kardinalita**	**Popis**
Id	LONG	Primární klíč účtu, automaticky generovaný identifikátor.
email	VARCHAR	Unikátní e-mailová adresa sloužící jako přihlašovací údaj.
password_hash	VARCHAR	Zahashované heslo uložené pro autentizaci uživatele.
name	VARCHAR	Zobrazené jméno uživatele nebo název podnikatelského účtu.
image_url	VARCHAR	URL adresa profilového obrázku.
region	VARCHAR	Preferovaný region uživatele pro filtrování obsahu a notifikace.
is_business	BOOLEAN	Určuje, zda se jedná o podnikatelský účet nebo běžného uživatele.
ico	String	Identifikátor podniku ičo (ověřený)
favorite_categories	M:N	Vazba na tabulku category, uživatelem preferované typy událostí.
device_tokens	1:N	Vazba na tabulku device_token, zařízení spojená s tímto účtem pro FCM notifikace.
created_events	1:N	Vazba na tabulku event, události vytvořené tímto účtem (pouze pro podniky).

Tabulka Category slouží k evidenci jednotlivých tematických kategorií, do kterých mohou být zařazeny kulturní nebo společenské události. Kategorie zároveň slouží jako nástroj pro personalizaci, uživatelé si mohou vybrat preferované typy událostí a na základě toho jim jsou zobrazovány odpovídající události. Rozšíření tabulky o nové kategorie není možné provést uživatelským zásahem, doplnění nabídky kategorií vyžaduje kontaktování technické podpory.

Tabulka 10 - Popis tabulky Category. zdroj: vlastní

Název sloupce / vztahu	**Datový typ / Kardinalita**	**Popis**
id	LONG	Primární identifikátor kategorie, automaticky generovaný.
name	VARCHAR	Unikátní název kategorie.
favoriteByAccount	M:N	Vazba na tabulku account, reprezentuje uživatele, kteří si kategorii přidali mezi oblíbené.
selectedByEvent	M:N	Vazba na tabulku event, události, které spadají pod danou kategorii.

Tabulka Currency uchovává informace o měnách používaných v systému při zobrazování cen kulturních a společenských událostí. Umožňuje oddělit konkrétní částky od jejich měnové reprezentace, čímž podporuje mezinárodní rozšiřitelnost systému a správnou lokalizaci cenových údajů. Každá událost je navázána na jednu měnu prostřednictvím cizího klíče. Rozšíření tabulky o nové měny není možné provést uživatelským zásahem, doplnění měnové nabídky vyžaduje kontaktování technické podpory.

Tabulka 11 - Popis tabulky Currency. zdroj: vlastní

Název sloupce / vztahu	**Datový typ / Kardinalita**	**Popis**
id	LONG	Primární identifikátor měny, automaticky generovaný.
code	VARCHAR	Kód měny (např. CZK, EUR), unikátní a povinný.
symbol	VARCHAR	Symbol měny pro zobrazování (např. Kč, €).
name	VARCHAR	Plný název měny.
usedBy	1:N	Vazba na tabulku event, označuje události využívající danou měnu.

Tabulka Device_token slouží k uchovávání informací o zařízeních, která jsou spojena s uživatelskými účty za účelem doručování push notifikací. Každý záznam v tabulce reprezentuje jedno mobilní zařízení (např. telefon nebo tablet) registrované prostřednictvím služby Firebase Cloud Messaging (FCM).

Tabulka 12 - Popis tabulky Device_token. zdroj: vlastní

Název sloupce / vztahu	**Datový typ / Kardinalita**	**Popis**
id	LONG	Primární identifikátor zařízení, automaticky generovaný.
deviceToken	VARCHAR	Unikátní token zařízení vygenerovaný službou FCM.
platform	VARCHAR	Platforma zařízení (např. android, ios).
createTime	TIMESTAMP	Datum a čas registrace zařízení.
tokenOwner	N:1	Vazba na tabulku account, uživatel, kterému dané zařízení patří.

Tabulka Location slouží k uchovávání informací o konkrétních místech konání událostí a zároveň k evidenci sídel podnikatelských účtů. Každý záznam obsahuje adresu a geografické souřadnice, které umožňují prostorové vyhledávání a zobrazení událostí na mapě. Lokace jsou vytvářeny podniky a mohou být využívány opakovaně pro různé události.

Tabulka 13 - Popis tabulky Location. zdroj: vlastní

Název sloupce / vztahu	**Datový typ / Kardinalita**	**Popis**
id	LONG	Primární identifikátor lokace, automaticky generovaný.
address	VARCHAR	Textová reprezentace adresy místa.
latitude	DOUBLE	Zeměpisná šířka.
longitude	DOUBLE	Zeměpisná délka.
createdBy	N:1	Vazba na tabulku account, účet, který lokaci vytvořil.
hostingEvent	1:N	Vazba na tabulku event, události konané na daném místě.

Tabulka Event představuje klíčovou entitu systému. Eviduje veškeré kulturní a společenské události, které jsou zobrazovány uživatelům v mobilní aplikaci. Každá událost obsahuje základní popisné informace, je přiřazena ke konkrétní lokaci, tvůrci a případně jedné či více tematickým kategoriím.

Tabulka 14 - Popis tabulky Event. zdroj: vlastní

Název sloupce / vztahu	**Datový typ / Kardinalita**	**Popis**
id	LONG	Primární identifikátor události, automaticky generovaný.
image_url	VARCHAR	URL adresa obrázku ilustrující danou událost.
title	VARCHAR	Název události zobrazovaný v přehledech.
description	TEXT	Podrobný popis události.
start_time	TIMESTAMP	Datum a čas začátku konání události.
end_time	TIMESTAMP	Datum a čas ukončení události (volitelný).
price	DECIMAL	Cena vstupného.
created_at	TIMESTAMP	Datum a čas vytvoření záznamu o události.
createdBy	N:1	FK na tabulku account, autor nebo pořadatel události.
happeningOn	N:1	FK na tabulku location, kde se událost koná.
currency	N:1	FK na tabulku currency, měna ceny vstupného.
selectedCategories	M:N	Vazba na tabulku category, kategorie, pod které událost spadá.

3.4 Implementace backendu

Tato kapitola popisuje implementaci serverové části systému, která tvoří aplikační logiku a zajišťuje komunikaci mezi mobilní aplikací a databází. Backend byl navržen jako RESTful webová služba postavená na frameworku Spring Boot. Hlavní úlohou backendu je zpracovávat požadavky přicházející z klientské aplikace, provádět validaci vstupních dat, přistupovat k databázové vrstvě a poskytovat strukturované výstupy ve formátu JSON.

3.4.1 Použité knihovny pro Spring Boot

Pro implementaci backendu byla využita platforma Spring Boot, která zajišťuje jednoduchou konfiguraci a spouštění celé aplikace. Práce s databází je realizována pomocí knihovny Spring Data JPA, která umožňuje efektivní mapování objektů na relační databázové struktury. Pro validaci vstupních dat je použito rozšíření Spring Boot Starter Validation. Zabezpečení aplikačních rozhraní zajišťuje Spring Security, zatímco autentizace uživatelů probíhá pomocí JWT tokenů. Správa migrací databáze je řešena pomocí nástroje Flyway, který umožňuje verzování schématu. Pro testování backendových komponent je využita knihovna Spring Boot Starter Test a Spring Security Test. K usnadnění práce s datovými objekty a eliminaci opakujícího se kódu je použita anotacemi řízená knihovna Lombok. Pro odesílání push notifikací je integrována knihovna Firebase Admin.

3.4.2 Architektura backendu

Architektura backendu vychází ze zavedených konvencí a osvědčených návrhových vzorů, které jsou běžné v rámci vývoje webových aplikací v prostředí Spring Boot. Cílem je dosažení přehlednosti, oddělení odpovědností a snadné údržby systému. Celá aplikace je členěna do více vrstev, přičemž každá vrstva má jasně definovanou roli:

- **Controller vrstva** zpracovává příchozí HTTP požadavky z klientské aplikace, provádí základní validaci vstupů a předává data dále do servisní vrstvy.
- **Service vrstva** obsahuje obchodní logiku aplikace. Zde dochází k transformaci dat, řízení transakcí a implementaci pravidel specifických pro doménu aplikace.
- **Repository vrstva** slouží pro přístup k databázi prostřednictvím rozhraní Spring Data JPA. Tato vrstva umožňuje definovat metody pro vyhledávání, ukládání a mazání dat bez nutnosti psaní explicitních SQL dotazů.

Veškeré doménové entity jsou umístěny ve složce Entity, přičemž každá entita má vlastní podsložku. V ní se nachází nejen samotná JPA entita, ale také související datové přenosové objekty (DTO), které slouží k oddělení interní reprezentace dat od struktury přenášené na klienta.

Chybové stavy jsou řešeny pomocí vlastních výjimek, které se nacházejí ve složce Exception. Každá výjimka dědí ze třídy RuntimeException, což umožňuje jejich jednoduché odchytávání. Obsluha výjimek je centralizována pomocí tříd označených anotací @ControllerAdvice, ve kterých jsou jednotlivé metody označeny anotací @ExceptionHandler pro zpracování konkrétních typů chyb.

Aplikace obsahuje také složku Scheduled, kde se nacházejí komponenty s pravidelně spouštěnými úlohami. Třídy v této části jsou označeny anotací @Component a jejich metody anotací @Scheduled, přičemž spouštění probíhá dle definovaných CRON výrazů. Tyto úlohy slouží například k odesílání pravidelných notifikací.

Konfigurační logika je centralizována ve složce Config. Nacházejí se zde třídy označené anotacemi @Component, @Configuration nebo @Bean. Tyto třídy zajišťují konfiguraci bezpečnostních mechanismů (např. JWT autentizace, CORS, heslování), zpracování tokenů, nastavení plánovače úloh a dalších aplikačních parametrů.

3.4.3 REST API

Backendová část systému je navržena jako RESTful služba poskytující datové rozhraní mezi mobilní aplikací a databází. API je strukturováno podle principů REST, které zajišťují konzistenci, jednoduchost a snadnou rozšiřitelnost. Každý endpoint reprezentuje konkrétní operaci nad entitou, jako je vytvoření, čtení, aktualizace nebo mazání záznamu. Výměna dat probíhá ve formátu JSON pomocí jednotlivých http metod. Pro přístup k chráněným zdrojům je vyžadována autentizace pomocí JWT. Následuje výčet jednotlivých REST endpointů a jejich význam.

Account controller zajišťuje správu uživatelských účtů, přístup k informacím o přihlášeném uživateli, oblíbeným událostem a kategoriím.

Tabulka 15 - Popis controlleru Account controller. zdroj: vlastní

Endpoint	Popis
GET /api/account/me	Vrací detail aktuálně přihlášeného uživatele.
PUT /api/account/me	Aktualizuje profil přihlášeného uživatele.
GET /api/account/favorite-categories	Vrací seznam oblíbených kategorií přihlášeného uživatele.
PUT /api/account/favorite-categories	Nahrazuje celý seznam oblíbených kategorií novým.
POST /api/account/favorite-categories/{categoryId}	Přidá kategorii do oblíbených.
DELETE /api/account/favorite-categories/{categoryId}	Odebere kategorii z oblíbených.

Auth controller zajišťuje autentizaci uživatelů, konkrétně registraci nových účtů, přihlášení pomocí e-mailu a hesla a obnovu hesla pomocí odeslání emailu s krátkodobě platícím tokenem, a endpointem který token ověří a nastaví nové heslo.

Tabulka 16 - Popis controlleru Auth controller. zdroj: vlastní

Endpoint	Popis
POST /api/auth/register	Zaregistruje nového uživatele a vrátí JWT token.
POST /api/auth/login	Přihlásí existujícího uživatele a vrátí JWT token.
POST/api/auth/password-reset/request	Odešle email pro obnovu hesla na požadovaný email.
POST/api/auth/password-reset/reset	Provede změnu hesla ověřenou tokenem.

Currency a **Category controllery** poskytují veřejné rozhraní pro načtení podpůrných dat, jako jsou měny a kategorie událostí. Obě entity slouží jako referenční seznamy využívané při tvorbě a filtrování událostí v systému.

Tabulka 17 - Popis controllerů Category a Currency controller. zdroj: vlastní

Endpoint	Popis
GET /api/currency	Vrací seznam všech dostupných měn v systému.
GET /api/category	Vrací seznam všech kategorií událostí.

Device token controller slouží ke správě zařízení registrovaných pro příjem push notifikací. Umožňuje správu zařízení podle tokenu vygenerovaného službou Firebase.

Tabulka 18 - Popis controleru device_token controller. zdroj: vlastní

Endpoint	Popis
POST /api/device-token	Uloží token zařízení pro aktuálně přihlášeného uživatele.
DELETE /api/device-token	Odstraní zadaný token zařízení ze systému.

Location controller slouží ke správě míst, na kterých se konají události. Umožňuje vytváření, aktualizaci, mazání i načítání konkrétní lokace podle ID. Lokace jsou vytvářeny podniky a mohou být využívány opakovaně.

Tabulka 19 - Popis controlleru Location controller. zdroj: vlastní

Endpoint	Popis
GET /api/location/{id}	Vrací detail konkrétní lokace podle jejího ID.
POST /api/location	Vytváří novou lokaci pod přihlášeným podnikatelským účtem.
PUT /api/location/{id}	Aktualizuje existující lokaci podle ID.
DELETE /api/location/{id}	Odstraňuje lokaci ze systému podle ID.

Event controller zajišťuje kompletní správu kulturních a společenských událostí. Umožňuje vytváření, úpravu, mazání a filtrování událostí podle různých kritérií (např. preference uživatele, geografická poloha nebo stránkování pomocí kurzoru).

Tabulka 20 - Popis controlleru Event controller. zdroj: vlastní

Endpoint	Popis
GET /api/event/{id}	Vrací detail konkrétní události podle ID.
GET /api/event/my	Vrací seznam událostí vytvořených přihlášeným uživatelem.
POST /api/event	Vytvoří novou událost pod účtem aktuálně přihlášeného uživatele.
PUT /api/event/{id}	Aktualizuje vybranou událost podle ID.
DELETE /api/event/{id}	Odstraní událost podle ID.
GET /api/event/filtered	Vrací filtrované události s podporou kurzorového stránkování.
GET /api/event/events-in-bounds	Vrací události nacházející se v daném geografickém obdélníku (mapa).

Mapbox Controller slouží jako backendová proxy, která zajišťuje volání příslušných endpointů služby Mapbox API. Jeho úkolem je doplnit požadavek o potřebné API klíče tak, aby tyto klíče nebyly vystaveny veřejně v klientských aplikacích. Tímto způsobem je zajištěna bezpečná komunikace se službou Mapbox při zachování integrity a ochrany citlivých přístupových údajů.

Tabulka 21 - Popis controlleru Mapbox controller. zdroj: vlastní

Endpoint	Popis
GET/suggestions ?{suggestion} &advancedSearch	Vrácí méně detailní objekty s limitem 3 podle obsahu suggestion. advancedSearch zajištje detailnější vyhledávání zaměřené na města
GET /retrieve/{id}	Získá detailní objekt hledaného pojmu podle mapboxID

CloudflareImages controller slouží k zajištění komunikace s Cloudflare Images API. Tento kontroler byl implementován především z důvodu ochrany citlivých API klíčů, aby nedošlo k jejich vystavení v klientských aplikacích. Díky tomu probíhá komunikace s Cloudflare bezpečně prostřednictvím backendu, který má klíče k dispozici, zatímco klienti využívají pouze zprostředkované endpointy, nebo bezpečné upload adresy.

Tabulka 22 - Popis controlleru CloudflareImages controller. zdroj: vlastní

Endpoint	Popis
GET /cloudfalre-images/direct-ipload	Requestem na cloudflare API vygeneruje časově omeznou url pro nahrání obrázku
DELETE/cloudflare-images/{id}	Odstraní obrázek podle jeho id

3.4.4 Testování

Testování backendové části aplikace bylo realizováno kombinací unit testů a integračních testů. Tento přístup umožňuje jednak izolovaně ověřit správnost jednotlivých metod a business logiky, a zároveň prověřit funkčnost celého systému nad skutečnou databází a běžící aplikací.

3.4.4.1 Unit testy

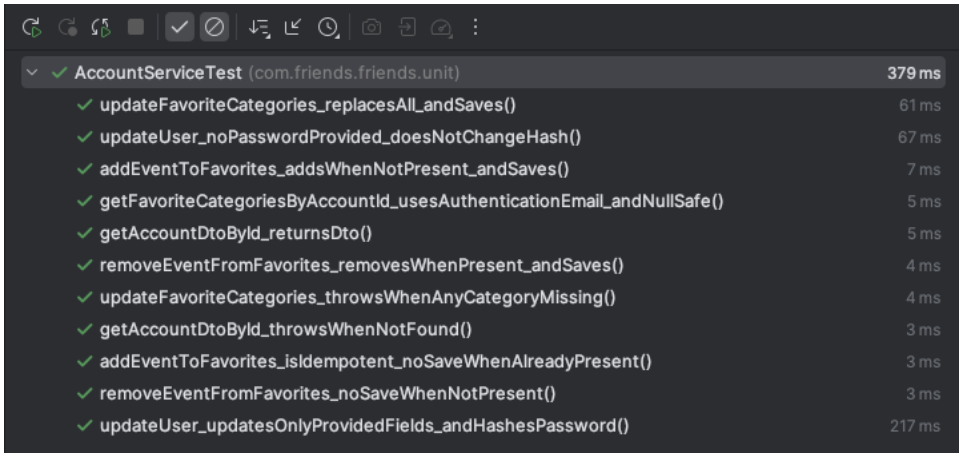
Unit testy byly zaměřeny na servisní vrstvu aplikace, která představuje jádro business logiky. Testy byly psány v prostředí JUnit 5 s využitím knihovny Mockito pro vytváření mocků závislostí. Hlavním cílem bylo ověřit, že jednotlivé služby pracují korektně i bez reálné databáze, a správně reagují na různé vstupy či chybové stavy.

Unit testy pokrývají klíčové scénáře v servisní vrstvě aplikace. Byly otestovány jak pozitivní případy (správné zadání), tak i negativní scénáře (neexistující záznamy, neplatné vstupy, duplicitní hodnoty).

AccountServiceTest

Testování této služby se zaměřilo na:

- získání uživatelských účtů podle ID včetně správného mapování na AccountDto
- aktualizaci účtu uživatele: ověření, že se mění pouze předané atributy
- správu oblíbených kategorií: nahrazování celého seznamu kategorií, detekce neexistujících kategorií a ošetření null hodnot.

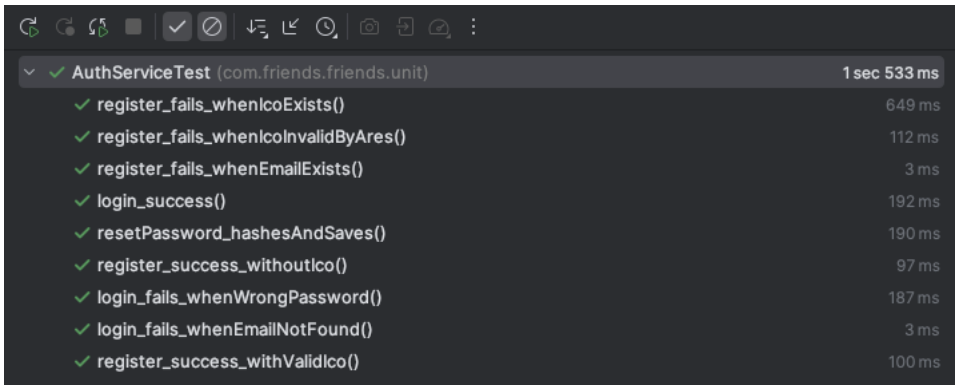


Obrázek 7 - Unit testy AccountService. zdroj: vlastní

AuthServiceTest

Tato třída byla testována zejména z pohledu autentizace a registrace:

- registrace běžného i podnikatelského účtu, včetně validace IČO
- ošetření duplicitních údajů (existující e-mail nebo IČO)
- přihlášení uživatele pomocí ověření hashe hesla a generování JWT tokenu
- obsluha chybových stavů: neexistující uživatel, neplatné heslo
- resetování hesla pomocí JWT reset tokenu a následné uložení nového hashe

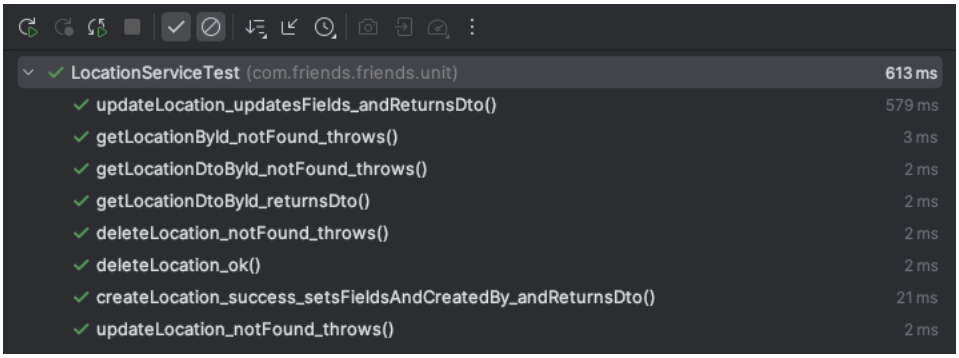


Obrázek 8 - Unit testy AuthService. zdroj: vlastní

LocationServiceTest

Testy lokací byly zaměřeny na:

- získání lokace dle ID a mapování na DTO
- vytváření nové lokace navázané na přihlášeného uživatele
- aktualizaci existující lokace s ověřením změny všech atributů
- mazání lokace s validací existence v databázi



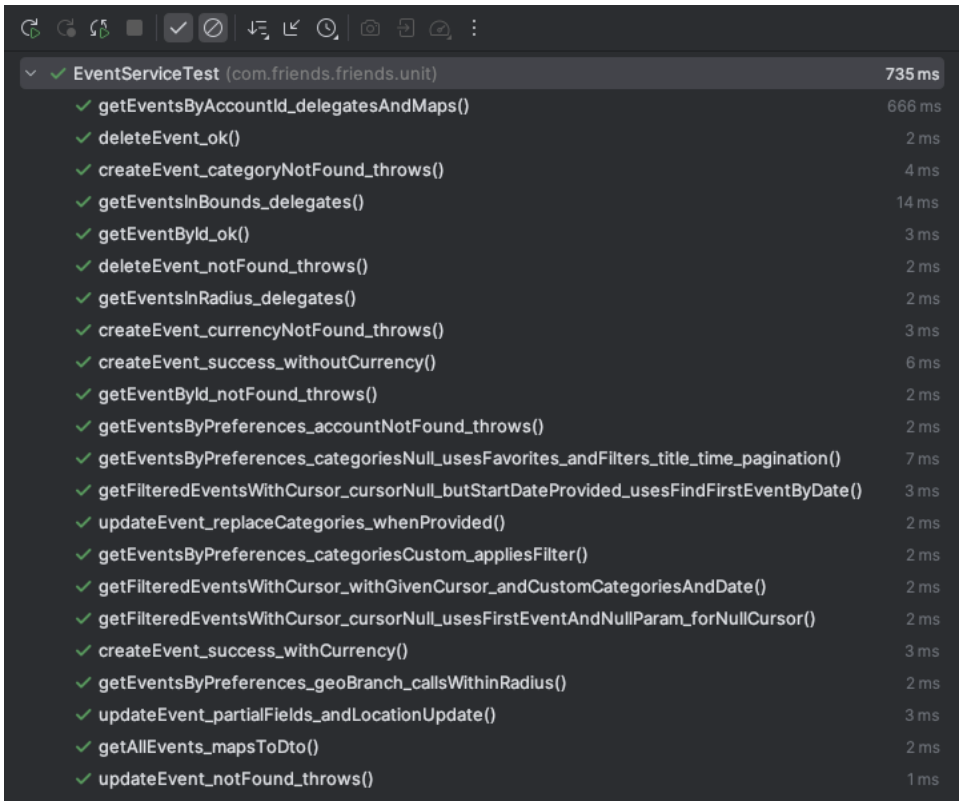
Obrázek 9 - Unit testy LocationService. zdroj: vlastní

EventServiceTest

Nejrozsáhlejší testy byly věnovány správě událostí:

- získání všech událostí i detailu konkrétní události
- vytváření nové události s kontrolou povinných vazeb (kategorie, lokalita, měna)
- aktualizace událostí včetně dílčí změny polí a nahrazování kategorií

- mazání událostí a ošetření případů, kdy událost neexistuje
- filtrování událostí podle preferencí uživatele, vybraných kategorií, času nebo geografické polohy
- stránkování a práce s tzv. cursor-based načítáním událostí
- zpracování mapových endpointů getEventsInBounds a getEventsInRadius



Obrázek 10 - Unit testy EventService. zdroj: vlastní

3.4.4.2 Integrační testy

Integrační testy byly spuštěny pomocí anotace @SpringBootTest s vypnutou náhradou databáze @AutoConfigureTestDatabase aby běžely proti reálné testovací databázi a konfiguraci JPA/Hibernate. Každý test probíhá v rámci transakce (@Transactional), která je po jeho dokončení automaticky rollbackována, takže prostředí zůstává vždy čisté. V metodě @BeforeEach se připravuje minimální, ale realistická datová sada zahrnující uživatelský účet (organizátora), kategorie (např. Rock, Pop), měnu (CZK), několik lokací v Praze s různou vzdáleností (v radiusu i mimo něj, v bboxu i mimo něj) a pět událostí rozmístěných v čase (včera, dnes, zítra, specifické datum za 3 dny) i prostoru (blízko/daleko).

Testovací třída EventRepositoryIT tak využívá plnohodnotný Spring kontext včetně repozitářů EventRepository, LocationRepository, CategoryRepository, CurrencyRepository a AccountRepository. Díky tomu integrační testy ověřují reálné SQL dotazy nad skutečným databázovým schématem, což unit testy s mocky nepostihnou, a zároveň prověřují geografické

výpočty (radius/bounding box) v kombinaci s ostatními filtry a cursor-based stránkováním, které jsou klíčové pro výkon i uživatelský zážitek (feed, mapové zobrazení).

Provedené integrační testy byly zacíleny primárně na kritické a komplexní části logiky, zejména nad repozitářovou vrstvou událostí.

Hledání událostí v radiusu

findByLocationWithinRadius_returnsOnlyEventsInsideRadius_sortedByDistanceOrTime

- Ověřuje, že dotaz vrací pouze události ležící uvnitř zadaného poloměru a vyloučí vzdálenější.

První „dnešní“ událost a první událost od daného data

- findFirstTodayEvent_returnsEarliestToday: vrací nejčasnější dnešní událost.
- findFirstEventByDate_returnsEarliestAtGivenDayOrAfter: vrací první událost v konkrétní den (nebo po něm).

Komplexní filtrování s kurzorem (cursor-based pagination)

findFilteredEvents_withCursorAndFilters_respectsAll

- Kombinuje kurzor (vynechání položky kurzoru), textový filtr (case-insensitive „rock“), kategoriální filtr (např. jen Rock), časové okno (od dneška do +4 dny), geofilter (např. do 6 km) a limit výsledků.
- Ověřuje se, že dotaz vrací jen odpovídající události (např. „Rock“ v okně a v dosahu) a zároveň správně nevrátí kurzorovou položku či nerelevantní záznamy (minulost, jiné kategorie, mimo textový filtr).

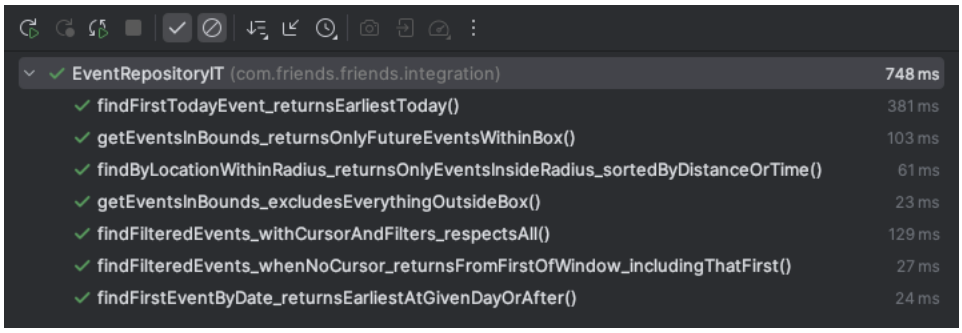
První stránka bez kurzoru

findFilteredEvents_whenNoCursor_returnsFromFirstOfWindow_includingThatFirst

- Simuluje první načtení feedu, pokud kurzor není k dispozici, test ověřuje, že je zahrnuta i první událost v aktuálním okně.

Události v mapovém výřezu (bounding box)

- getEventsInBounds_returnsOnlyFutureEventsWithinBox: vrací pouze budoucí události ležící uvnitř zadaného bboxu; test navíc verifikuje, že všechny vrácené body mají souřadnice uvnitř obdélníku.
- getEventsInBounds_excludesEverythingOutsideBox: pro malý bbox očekávaně vrací prázdnou množinu.



Obrázek 11 - Integrační testy EventRepository. zdroj: vlastní

3.5 Implementace frontendu

Frontend je realizován pomocí mobilní aplikace napsané ve Flutteru. Tato mobilní aplikace představuje primární rozhraní, kterým uživatel interaguje se systémem. Jejím úkolem je přehledně zobrazovat kulturní a společenské události podle aktuální polohy uživatele, umožňovat jejich filtrování, zobrazení detailů a navigaci na místo konání. Dále zajišťuje personalizované notifikace s denním přehledem událostí a poskytuje jednoduchý způsob správy uživatelského účtu. Aplikace tak propojuje uživatele s kulturním děním v jejich okolí rychlým, moderním a uživatelsky přívětivým způsobem

3.5.1 Architektura aplikace

Aplikace byla vytvořena pomocí nástroje flutter create, který vygeneruje základní souborovou strukturu pro nový Flutter projekt. Ta zahrnuje platformně specifické složky s konfiguračními soubory nezbytnými pro kompilaci kódu na jednotlivé cílové platformy. Samotný aplikační kód je soustředěn v adresáři lib/hotspot_app.

Projekt je následně organizován do logických celků podle zodpovědností jednotlivých vrstev:

- **core**: obsahuje kód nezávislý na konkrétních modulech aplikace, který lze považovat za základní stavební prvky projektu.
- **exceptions**, **functions**, **hooks**: samostatné složky, které zahrnují znovupoužitelný kód využívaný na více místech aplikace. Díky své obecnosti mají globálnější charakter a přispívají k jednotnému řešení chyb, opakovaně využívaným funkcím a správě stavového cyklu komponent.
- **presentation**: zahrnuje uživatelské rozhraní aplikace. Struktura je rozdělena na složku **pages**, která obsahuje jednotlivé obrazovky aplikace, a složku **widgets**, v níž se nachází sdílené komponenty. Každá stránka je koncipována jako samostatný modul, který může definovat vlastní widgety, hooky, funkce a stav, využívané pouze v rámci dané stránky.

- **services**: obsahuje logiku pro komunikaci s backendem a dalšími API. Jednotlivé podsložky odpovídají konkrétním backendovým kontrolerům. Nachází se zde modely návratových objektů (entity a DTO), dále třída **service.dart** zajišťující volání endpointů prostřednictvím HTTP metod a nadřazená vrstva **query.dart**, která s využitím knihoven *cached_query* a *Riverpod* přidává podporu cachování a zpřístupňuje data aplikaci prostřednictvím globálního stavu.
- **state**: obsahuje globální stav aplikace, definovaný pomocí providerů a datových tříd. Tyto objekty reprezentují sdílený stav aplikace, který je využíván v různých částech uživatelského rozhraní.

Tato architektura zajišťuje přehlednou organizaci projektu, jasné oddělení jednotlivých vrstev a zároveň podporuje modularitu, znuvupoužitelnost i dlouhodobou udržitelnost kódu.

3.5.2 Uživatelské rozhraní aplikace

Před samotnou implementací mobilní aplikace je vhodné navrhnout a vizuálně si představit její uživatelské rozhraní například pomocí design mockupů. Mockupy (nebo také drátové modely či návrhové šablony) slouží jako vizuální prototypy jednotlivých obrazovek aplikace, které pomáhají definovat rozmístění prvků, navigaci, barvy, typografii i celkový vzhled.

Hlavním cílem tvorby mockupů je ověřit návrh uživatelského prostředí ještě před zahájením programování a tím předejít zbytečným chybám, přepracovávání a neefektivnímu vývoji. Design mockupy usnadňují komunikaci mezi vývojářem, designérem a případnými zadavateli a zároveň slouží jako podklad pro ladění uživatelské přívětivosti a vizuální konzistence celé aplikace.

Sekundárním cílem tvorby mockupů v rámci této práce je vizualizace jednotlivých prvků na každé stránce a zároveň identifikace jejich požadavků na backend. Vytváření UI návrhů slouží jako základ pro analýzu potřebných dat, jejich struktur a akcí, které s nimi bude nutné provádět. Tento přístup následně pomáhá definovat seznam entit, datových modelů, API endpointů a souvisejících funkcionalit, které bude třeba implementovat pro správnou funkčnost celé aplikace.

Obrazovky přihlášení a registrace

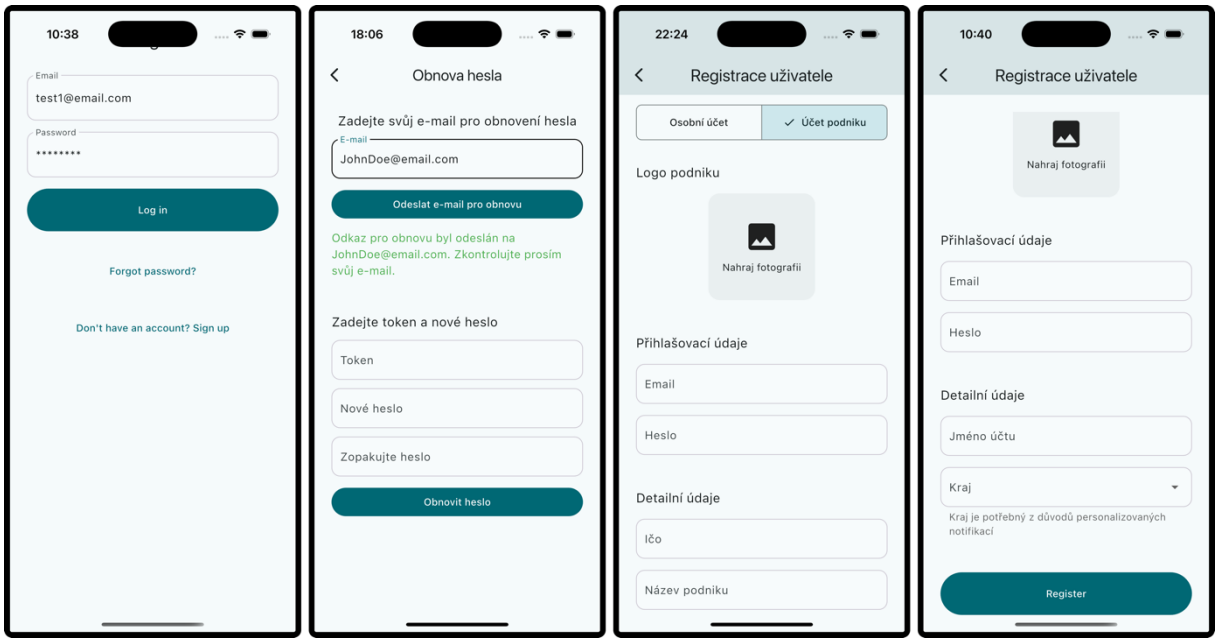
První obrazovka představuje přihlašovací formulář, který umožňuje uživateli zadat své přihlašovací údaje ve formě e-mailové adresy a hesla. Pod hlavním tlačítkem „Log in“, které slouží k odeslání údajů a ověření uživatele, se nachází odkaz „Forgot password?“, který

uživatele přesměruje na obrazovku pro reset hesla. V dolní části obrazovky je odkaz „Don't have an account? Sign up", pomocí kterého se může uživatel přesunout na registrační formulář.

Druhá obrazovka představuje formulář pro resetování hesla, obsahuj epole pro zadání emailu a tolačítko, pomocí kterého je na zadaný email odeslán kód pro obnovu. Druhá část formuláře poté slouží k vyvoření nového hesla, obsahuje pole pro token z emailu a nové heslo. Nové heslo je nastaveno po zvalidování formuláře stisknutím tlačítka Obnovit heslo.

Třetí obrazovka zobrazuje registrační rozhraní, ve kterém si uživatel volí typ účtu, který si přeje vytvořit. K dispozici jsou dvě možnosti: „Osobní účet" a „Účet podniku". Podle zvolené varianty se dynamicky mění podoba registračního formuláře.

Třetí obrazovka představuje zbytek registračního formuláře. Obsahuje pole pro zadání e-mailu, hesla, jména účtu a výběr kraje, ve kterém se uživatel nachází. Celý formulář je zakončen tlačítkem „Register", které spustí proces registrace a odeslání údajů na backend.



Obrázek 12 - Obrazovky přihlášení, obnova hesla, registrace. zdroj: vlastní

Hlavní obrazovka, eventy a filtry

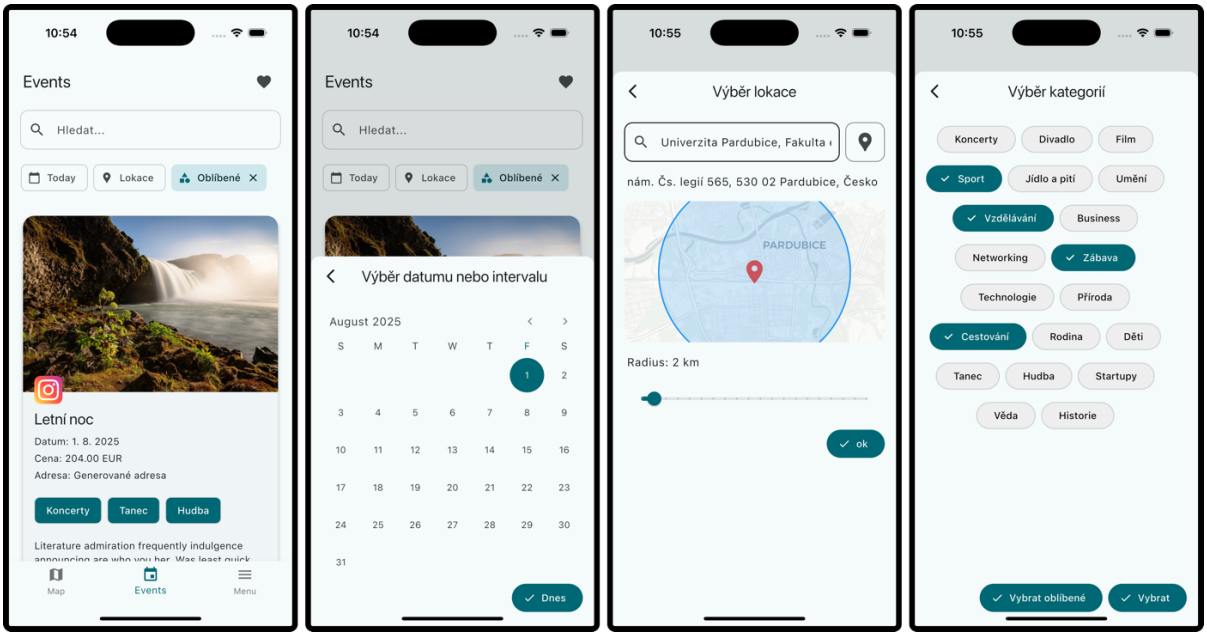
Hlavní obrazovka aplikace slouží k zobrazování veřejných kulturních a společenských událostí ve formě karet ve feedu. Každá karta obsahuje název události, obrázek, typ události, místo konání, datum a čas. Kliknutím na kartu se uživatel proklikne na zobrazení detailu události. V horní části obrazovky se nachází vyhledávací pole pro zadání názvu události a tlačítka pro otevření jednotlivých filtrů (datum, lokalita, kategorie). Pomocí těchto filtrů může uživatel přizpůsobit zobrazovaný obsah svým preferencím.

První filtr slouží k výběru konkrétního data nebo časového rozmezí, během kterého chce uživatel zobrazit události. Zobrazena je kalendářová mřížka umožňující snadný výběr jednoho dne nebo intervalu. Ve spodní části obrazovky je tlačítko, které aplikuje vybrané datum a zavře filtr.

Druhý filtr umožňuje vybrat konkrétní lokalitu, odkud se mají události zobrazovat. Místo je možné vybrat ze zadané adresy. Součástí rozhraní je interaktivní mapa s posuvníkem určujícím poloměr vyhledávání, ten lze nastavit v rádiusu od 1 do 20 kilometrů od vybraného boduu. Uživatel tak může přesně určit, jak daleko od vybraného bodu chce události zobrazit. Tento filtr je užitečný pro lokalizované hledání například v rámci města či regionu.

Třetí filtr umožňuje výběr typu událostí podle tematických kategorií. Kategorie jsou zobrazeny ve formě zaoblených tlačítek (chipů), mezi kterými může uživatel libovolně vybírat více možností najednou (např. Koncerty, Výstavy, Party, Divadlo apod.). Ve spodní části se nachází tlačítko pro potvrzení výběru, nebo aplikaci uživatelských preferencí. Tento filtr zajišťuje tematické cílení obsahu.

Jednotlivé filtry se poté na hlavní obrazovce zobrazují jako aktivní, společně se zvolenou hodnotou filtru. Je zde možné je také vyresetovat jejich defaultní hodnotou.



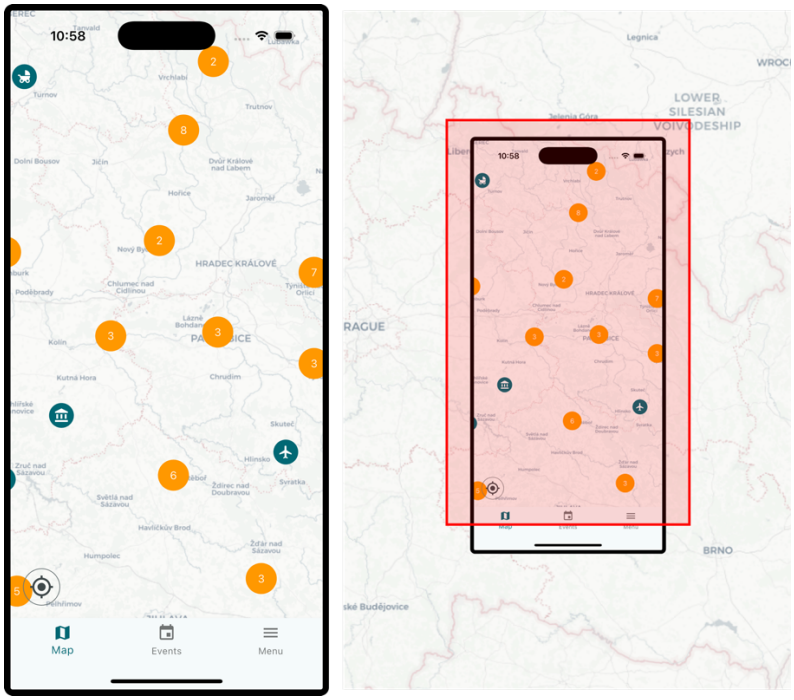
Obrázek 13 - Obrazovky event feed, filtry: datum, lokace, kategorie. zdroj: vlastní

Obrazovka mapy

Tato obrazovka představuje mapový pohled aplikace, který slouží k vizuálnímu zobrazení kulturních a společenských událostí podle jejich geografického umístění. Základem obrazovky je mapový podklad, na kterém jsou vyznačeny události formou barevných kruhových značek s čísly. Každé číslo představuje počet událostí v dané oblasti. Tato metoda

tzv. „clusteringu“ zajišťuje přehlednost i při větším množství událostí v jednom regionu. Při přiblížení mapy se klastry rozpadnou a zobrazí jednotlivé události. V levém dolním rohu je umístěno tlačítko pro lokalizaci uživatele, které po stisknutí přesune mapu na aktuální pozici.

Obrazovka mapy je navržena tak, aby efektivně pracovala s backendem a nezatěžovala systém stahováním všech událostí naráz. Prostřednictvím databázového endpointu **/api/event/events-in-bounds** jsou načítány pouze ty události, které se nacházejí v rámci aktuálně zobrazeného bounding boxu mapy na daném zařízení. Při pohybu v mapě je v krátkém intervalu (jedna vteřina) odesílán nový požadavek s aktualizovanými souřadnicemi bounding boxu. Tímto způsobem se zobrazují vždy pouze relevantní události pro danou oblast, což zajišťuje plynulost aplikace a současně šetří datové i výpočetní zdroje.



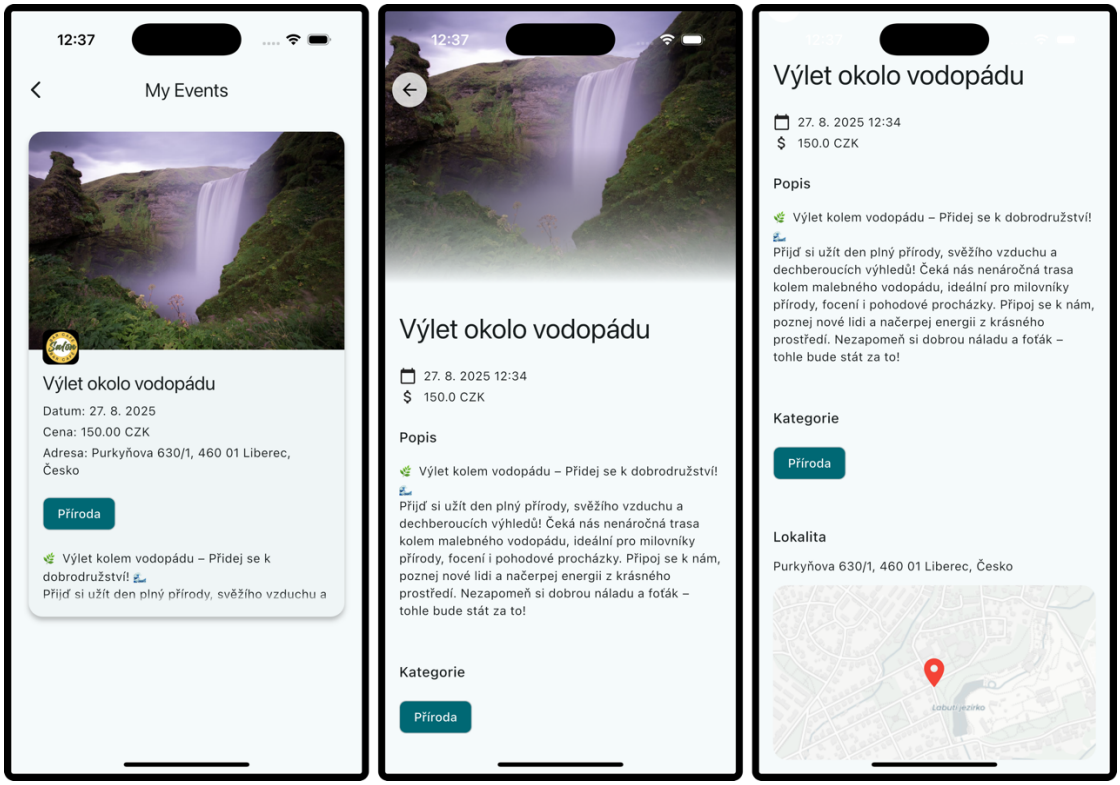
Obrázek 14 - Obrazovka mapy. zdroj: vlastní

Karta a detail události

Každá událost je reprezentována výraznou kartou s ilustračním obrázkem, názvem, datem a časem konání, cenou, adresou a štítkem kategorie. Ve spodní části karty je také krátký úryvek z popisu události. Karta působí vizuálně atraktivně a přehledně a slouží jako vstupní bod do podrobného zobrazení události, kde kliknutím na kartu se uživatel přesune na detailní obrazovku.

Druhá a třetí obrazovka představuje detail vybrané události s velkým úvodním obrázkem a plynulým přechodem do obsahu. Pod obrázkem jsou uvedeny klíčové informace jako, název události, datum a čas konání, cena vstupu, podrobný popis, kategorie události

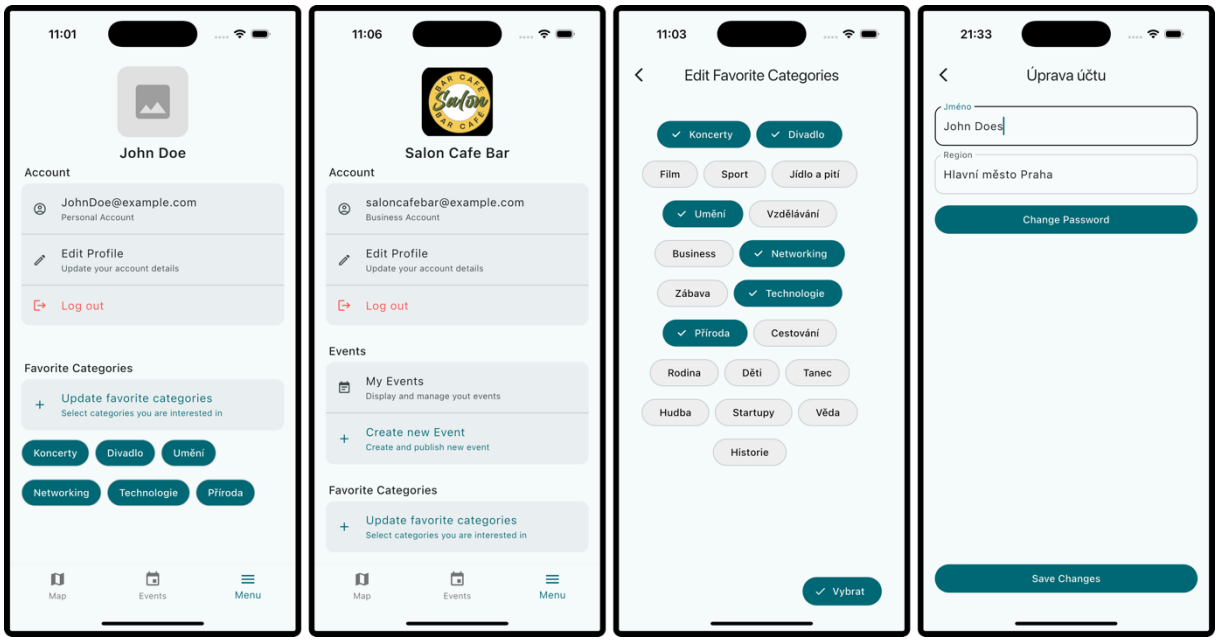
a adresa místa konání. Texty jsou rozděleny do tematických sekcí s důrazem na čitelnost a strukturu. Tato obrazovka slouží k podrobnému seznámení s obsahem události. V dolní části se nachází mapa s vyznačeným místem konání pomocí mapového podkladu a červeného bodu.



Obrázek 15 - Obrazovky karta události a detail události. zdroj: vlastní

Obrazovka menu

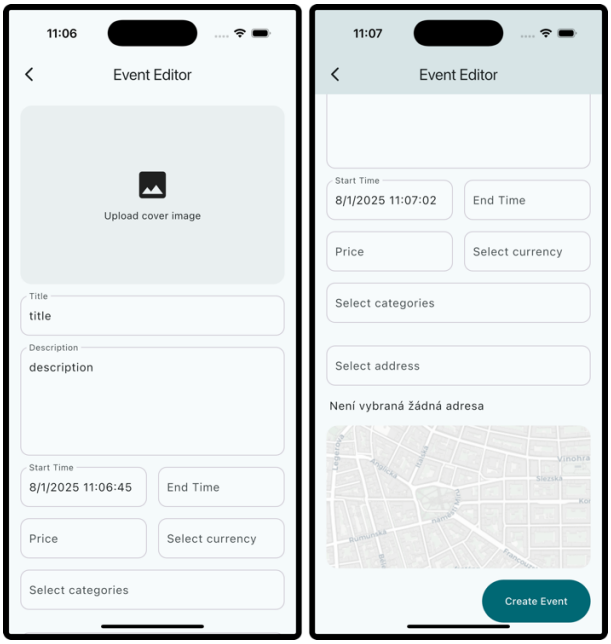
Obrazovka menu je rozdělena do několika částí podle typu funkcionality. V horní části se nachází profilová sekce, která obsahuje základní údaje o uživateli jako profilový obrázek, jméno, e-mail a typ účtu. Součástí této sekce je také akce pro úpravu profilu a akce „Log out“ pro odhlášení z aplikace. Pod touto částí se nachází sekce Events, která je zobrazena pouze v případě podnikatelských účtů. Obsahuje dvě akce: „My events“, jež zobrazí seznam událostí vytvořených daným uživatelem, a „Create event“, která uživatele naviguje do editoru pro vytvoření nové události. Spodní část obrazovky tvoří sekce „Favorite Categories“, kde je zobrazen seznam oblíbených kategorií událostí vybraných uživatelem. Každá kategorie je reprezentována barevným štítkem. Součástí sekce je také možnost „Update favorite categories“, která přesměruje na obrazovku pro správu a úpravu uživatelských preferencí.



Obrázek 16 - Obrazovky menu, úprava oblíbených kategorií, úprava účtu. zdroj: vlastní

Editor událostí

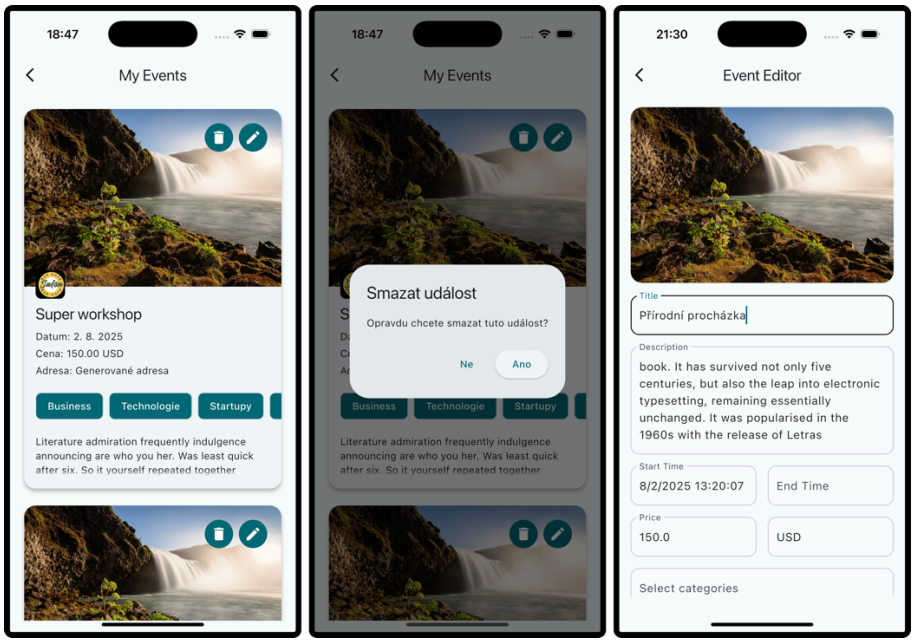
Tyto obrazovky představují editor událostí, který slouží k vytvoření nové akce. V horní části editoru se nachází pole určené pro nahrání fotografie události. Následují vstupní formulářová pole pro zadání základních informací – titulek, popis, čas začátku a konce události, cenu a příslušnou měnu. Součástí editoru je také vyhledávací pole pro výběr kategorií, do nichž událost spadá, a pole pro zadání adresy místa konání. V dolní části obrazovky je umístěno tlačítko „Create event“, které po stisku provede validaci všech vyplněných údajů. Pokud je formulář vyplněn správně, dojde k uložení a publikaci události.



Obrázek 17 - Obrazovka editor událostí. zdroj: vlastní

Moje eventy a jejich editace

Tyto obrazovky jsou dostupné výhradně pro podnikatelské účty prostřednictvím sekce „Events“ v menu. První z obrazovek představuje přehled událostí vytvořených daným účtem. Jednotlivé události jsou zde zobrazeny ve formě karet, které jsou doplněny o ovládací prvky pro editaci nebo odstranění. Funkce odstranění události je realizována prostřednictvím potvrzovacího dialogu, který zabraňuje nechtěnému smazání. Po potvrzení dojde k odstranění události z databáze i uživatelského přehledu. Funkce editace naopak otevře editor událostí předvyplněný existujícími údaji, které může uživatel upravit. Změny se uloží stiskem tlačítka „Update event“, čímž dojde k aktualizaci záznamu a jeho opětovnému publikování.



Obrázek 18 - Obrazovky moje události, editace události. zdroj: vlastní

3.6 Práce s API třetích stran

Pro zajištění klíčových funkcí aplikace bylo integrováno několik API třetích stran. Tato rozhraní rozšiřují schopnosti systému, aniž by bylo nutné implementovat vlastní řešení například pro notifikace, mapové služby nebo práci s obrázky. V aplikaci jsou konkrétně využívána tři rozhraní: Firebase Cloud Messaging, Mapbox Search Box API a Cloudflare Images API, ARES REST API.

3.6.1 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) je služba poskytovaná společností Google, která umožňuje spolehlivé doručování push notifikací do uživatelských zařízení. V rámci vyvíjené aplikace je FCM využíváno k zasílání denních přehledů událostí ve formě notifikací.

Celý proces začíná získáním uživatelského deviceTokenu, který je uložen do databáze pouze v případě, že uživatel explicitně povolí přijímání notifikací. Další nezbytnou součástí je určení přibližné polohy uživatele, přičemž při registraci je zadáván kraj, z něhož chce uživatel dostávat informace o kulturních a společenských událostech.

Samotná distribuce notifikací je následně plně automatizována. Na backendové straně je spuštěn cron job, který v předem definovaný čas získá seznam všech uživatelů společně s jejich deviceTokeny. Na základě zadaného kraje jsou vyhledány aktuální události v dané oblasti, ze kterých je sestaven text notifikace. Ten je následně odeslán prostřednictvím FCM na příslušné deviceTokeny.

Uživatel tak na svém zařízení obdrží notifikaci s denním přehledem událostí. Po jejím rozkliknutí je aplikace otevřena a uživatel získá možnost zobrazit detailnější informace vyhledáním konkrétních událostí.

3.6.2 Mapbox Search Box API

Mapbox Search Box API slouží k inteligentnímu vyhledávání míst a adres s podporou automatického doplňování a validace. V rámci vyvíjené aplikace je toto API využíváno především při vytváření nové události, kdy uživatel musí zvolit konkrétní adresu místa konání. Druhým případem jeho použití je uživatelský filtr polohy, kde si uživatel může určit svou aktuální polohu, případně ji vyhledat prostřednictvím vyhledávacího pole doplněného o návrhy právě z tohoto API.

Z bezpečnostních důvodů není možné uchovávat API klíč k Mapboxu přímo na straně klienta. Proto byla implementována backendová proxy, která zajišťuje bezpečné volání Mapbox služeb. Klientská aplikace posílá požadavky na vlastní backend, endpointy **/mapbox/suggestions** a **/mapbox/retrieve**, ten provede volání Mapbox API pomocí uloženého klíče a vrátí výsledky zpět klientovi. Tím je zajištěno, že klíč zůstává chráněn na serverové straně a nedochází k jeho vystavení uživatelům.

Výsledkem volání API je objekt obsahující užitečné geografické údaje, jako je název místa, úplná adresa a souřadnice (zeměpisná šířka a délka), které slouží například k vykreslení pinu na mapě. API poskytuje dva hlavní endpointy:

https://api.mapbox.com/search/searchbox/v1/suggest?q={search_text}

- vrací doporučení pro vyhledávací pole na základě zadaného textu. Tento endpoint obsahuje řadu konfiguračních parametrů, z nichž aplikace využívá zejména limit (omezení počtu doporučení) a country (omezení výsledků pouze na území České republiky).

https://api.mapbox.com/search/searchbox/v1/retrieve/{id}

- vrací detailní informace o místě na základě ID, které je získáno z výsledků předchozího dotazu *suggest*. Tento endpoint poskytuje rozšířený objekt místa, včetně názvu, adresy a souřadnic, přičemž tyto údaje jsou následně ukládány do databáze a dále využívány v aplikaci.

3.6.3 Cloudflare Images API

Cloudflare Images API představuje řešení pro správu, ukládání a optimalizaci obrázků. V rámci vyvíjené aplikace slouží tato služba k nahrávání a doručování obrázků událostí, profilových fotografií či log jednotlivých účtů. Jejím hlavním přínosem je zajištění rychlého načítání optimalizovaných obrázků prostřednictvím vestavěné CDN a zároveň jednoduchá integrace nahrávání ze strany klienta.

V aplikaci je tato služba využívána pouze pro uchovávání obrazových souborů a jejich načítání prostřednictvím veřejných URL adres. Proces nahrávání probíhá tak, že klientská část (mobilní aplikace) ukládá soubory přímo na servery Cloudflare. Aby nedošlo k ohrožení bezpečnosti zveřejněním API klíče, je na backendu implementován speciální endpoint, **/cloudflare-images/direct-upload**, který pro klienta vygeneruje jednorázovou upload URL adresu s omezenou platností. Mobilní aplikace následně tuto adresu využije k bezpečnému nahrání souboru přímo na Cloudflare Images. Tento mechanismus je v API označován jako Direct Creator Upload.

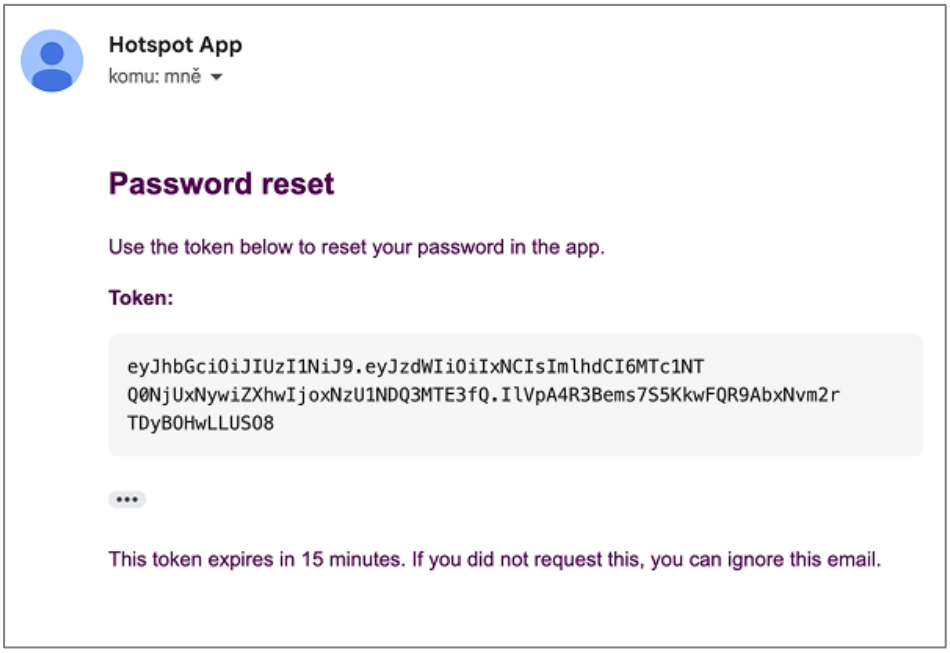
3.6.4 ARES API

ARES (Administrativní registr ekonomických subjektů) je veřejný registr, který poskytuje několik API endpointů umožňujících programový přístup k datům a vyhledávání v databázi ekonomických subjektů. V rámci vyvíjené aplikace plní toto API klíčovou roli zejména při registraci nových podniků. Každý podnik se registruje prostřednictvím svého unikátního IČO, přičemž jeho validace probíhá právě pomocí ARES API přes endpoint **https://ares.gov.cz/ekonomicke-subjekty-v-be/rest/ekonomicke-subjekty/{ico}.**

Pokud tento endpoint vrátí stavový kód 404, znamená to, že zadané IČO není v registru evidováno a registrace není povolena. V případě kódu 200 je IČO platné a registrace může pokračovat. Na backendové straně je zároveň ošetřena duplicita záznamů – v systému je IČO evidováno jako unikátní identifikátor, a proto není dovoleno, aby se registrovaly dva podniky se stejným IČO. V takovém případě je registrace zamítnuta.

3.6.5 Brevo SMTP

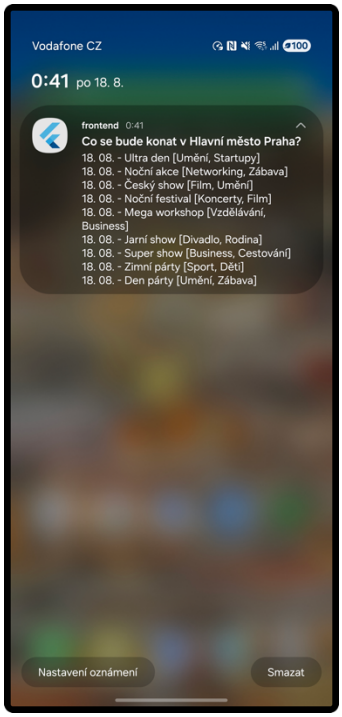
Brevo je e-mailová marketingová platforma, která umožňuje hromadné i cílené rozesílání e-mailů. V rámci vyvíjené aplikace je tato služba využívána pro rozesílání emailů sloužících k obnově hesla. Obsah e-mailu je konstruován na základě oficiální dokumentace služby a následně je odesílán prostřednictvím endpointu: **https://api.brevo.com/v3/smtp/email**. Tímto způsobem jsou zprávy doručovány ke koncovým uživatelům společně s pokyny k provedení obnovy hesla.



Obrázek 19 - Vzhled emailu pro obnovu hesla. zdroj: vlastní

3.7 Denní notifikace

V rámci aplikace je implementován systém denních notifikací, které uživatelům přinášejí pravidelný přehled o nadcházejících událostech v jejich okolí. Notifikace jsou generovány a odesílány na straně backendu prostřednictvím CRON úlohy definované pomocí mechanismu @Scheduled v prostředí Spring Framework. Odesílání probíhá každý den v 9:00 a je určeno všem uživatelům, kteří si povolili přijímání notifikací. Obsah notifikace vychází z dat uložených v systému. Pro každého uživatele se vyberou události konající se od aktuálního dne směrem do budoucna v rámci kraje, který uživatel zvolil při registraci. Notifikace má jednotnou strukturu – nadpis ve tvaru „Co se bude konat v [název kraje]?“, například *„Co se bude konat v Hlavním městě Praha?“*. Pod tímto nadpisem je uveden seznam až deseti událostí s klíčovými informacemi: datem konání, názvem a kategorií. Po doručení na zařízení je notifikace zobrazena uživateli a kliknutím na ni je uživatel přesměrován přímo do aplikace, kde si může zobrazit podrobnější informace o vybraných událostech.



Obrázek 20 - Denní notifikace. zdroj: vlastní

3.8 Zabezpečení, správa oprávnění a ochrana osobních údajů

Bezpečnost byla jedním z klíčových aspektů zohledněných už ve fázi návrhu celé aplikace. Cílem bylo vytvořit systém, který chrání uživatelská data, minimalizuje rizika neoprávněného přístupu a současně neomezuje uživatele zbytečnými kroky.

3.8.1 Validace uživatelských vstupů a zpracování výjimek

Aby nedocházelo k ukládání nekonzistentních nebo neplatných dat, jsou uživatelské vstupy kontrolovány jak na frontendové, tak na backendové straně. Na frontendu probíhá validace přímo v textových polích s využitím knihoven FormBuilder a FormBuilderValidators. Tyto nástroje zajišťují základní ověřování, jako je kontrola povinných polí nebo správného formátu e-mailové adresy. Okrajové případy a chyby, které vrátí backend, jsou uživateli prezentovány prostřednictvím vyjížděcího dialogu. Na backendové straně jsou vstupy dále kontrolovány na neplatné hodnoty a ověřována je i jejich validita. Typickým příkladem je ověření IČO prostřednictvím služby ARES. Pokud dojde k chybě, backend vyhodí výjimku a prostřednictvím objektu ResponseEntity vrátí odpovídající chybovou zprávu. Ta je následně zobrazena uživateli opět formou vyjížděcího dialogu.

Obrázek 21 - Validace uživatelských vstupů a zpracování výjimek. zdroj: vlastní

3.8.2 Authentizace a správa JWT tokenů

Autentizace v aplikaci je v současné verzi realizována prostřednictvím JWT access tokenů s omezenou dobou platnosti. Po jejich vypršení je vyžadováno nové přihlášení uživatele. Do budoucna je zvažováno rozšíření tohoto mechanismu o podporu refresh tokenů, které by umožnily prodloužení relace bez nutnosti opakovaného přihlášení. Součástí tohoto řešení by mohla být také jejich evidence na backendu, implementace rotace a případné rozšíření o dvoufaktorovou autentizaci (2FA) jako doplňkový bezpečnostní prvek.

3.8.3 Ochrana osobních údajů

Při návrhu aplikace byl od počátku uplatňován princip Privacy by Design, tedy takový přístup, kdy je ochrana soukromí a osobních údajů začleněna přímo do architektury a funkční logiky systému. Aplikace sbírá pouze ty údaje, které jsou nezbytné pro její fungování, personalizaci obsahu a poskytování služeb uživateli. Všechny osobní údaje jsou svázány s tabulkou Account a uloženy v bezpečné databázi. Následující tabulka definuje všechny sbírané osobní údaje, jejich popis a účel použití.

Tabulka 23 - Uložené osobní údaje uživatele. zdroj: vlastní

Osobní údaj	Popis	Účel použití
email	E-mailová adresa uživatele	Slouží k autentizaci, komunikaci a případnému obnovení přístupu k účtu
region	Přibližná poloha zvolená uživatelem	Personalizace obsahu a filtrování událostí podle geografické oblasti
imageUrl	URL profilového obrázku nahraného uživatelem	Zobrazení vizuální identity uživatele v aplikaci
name	Reálné nebo fiktivní jméno uživatele	Identifikace uživatele v rámci aplikace, např. v komentářích nebo profilech
favoriteCategories	Vazba na tabulku Category s oblíbenými typy událostí uživatele	Přizpůsobení doporučeného obsahu podle preferencí
deviceTokens	Vazba na tabulku DeviceToken obsahující FCM token pro zařízení	Umožnění zasílání push notifikací prostřednictvím služby Firebase Cloud Messaging

3.8.4 Požadovaná oprávnění aplikace a způsob jejich získávání

Aplikace vyžaduje pouze ta oprávnění, která jsou nezbytná pro poskytování hlavních funkcí uživateli. V současné verzi se jedná o oprávnění pro zasílání notifikací a přístup k poloze zařízení. Ani jedno ze zmíněných oprávnění však není povinné a aplikace zůstane funkční i bez jejich poskytnutí.

Oprávnění pro zasílání notifikací je vyžádáno při prvním spuštění aplikace po registraci nebo přihlášení uživatele. Pokud jej uživatel udělí, je do databáze uložen příslušný deviceToken, který slouží k zasílání push notifikací prostřednictvím služby Firebase Cloud Messaging. V případě, že oprávnění není uděleno, deviceToken se neukládá a notifikace nejsou doručovány.

Oprávnění k přístupu k poloze není ukládáno do databáze a aplikace jej vyžaduje pouze v okamžiku, kdy je nezbytné pro konkrétní akci, například při zobrazení aktuální polohy uživatele na mapě po kliknutí na příslušnou funkci. Tento přístup zajišťuje, že uživatel poskytuje přístup k citlivým údajům pouze tehdy, když je to skutečně potřeba.

3.8.5 Zabezpečení databázové vrstvy

Databázová vrstva je navržena tak, aby byla chráněna před neoprávněným přístupem a minimalizovala riziko úniku dat. Backendová aplikace k databázi přistupuje výhradně prostřednictvím interní sítě Dockeru a není dostupná z veřejného internetu. Přístup je omezen na dedikovaný databázový účet chráněný uživatelským jménem a heslem. Databázový server PostgreSQL tak přijímá spojení pouze z důvěryhodného aplikačního kontejneru, čímž je vyloučeno přímé připojení z neautorizovaných zdrojů. Tento přístup výrazně snižuje potenciální útočnou plochu a eliminuje rizika spojená s externím přístupem k databázi.

3.8.6 Omezení přístupu k API a povolené endpointy

API je veřejně přístupné v síti, avšak přístup k jednotlivým částem je striktně omezen podle úrovně oprávnění. Veřejně dostupné jsou pouze autentizační endpointy, které slouží k registraci, přihlášení a obnovení tokenů. Všechny ostatní endpointy vyžadují předání platného, neexpirovaného **access tokenu**, jehož validita je kontrolována při každém požadavku. Tento model zajišťuje, že s chráněnými zdroji mohou pracovat pouze ověření uživatelé.

3.8.7 Instalace z třetích stran

Aplikace je distribuována výhradně prostřednictvím oficiálních obchodů s aplikacemi, jako je Google Play. Instalace z neoficiálních zdrojů nebo třetích stran není doporučena, protože takové balíčky nemusí pocházet z důvěryhodného zdroje a mohou obsahovat škodlivý kód. Dodržování oficiálního způsobu distribuce zajišťuje, že uživatel získá autentickou a ověřenou verzi aplikace, která prošla bezpečnostní kontrolou daného obchodu.

3.8.8 Práce s API klíči

Správné zacházení s API klíči a konfiguračními soubory je klíčové pro zajištění bezpečnosti systému a prevenci jejich zneužití. Všechny citlivé hodnoty, jako jsou přístupové tokeny nebo konfigurační objekty, jsou ukládány výhradně na backendové straně a nejsou nikdy součástí frontendové aplikace. Na backendu jsou tyto údaje spravovány mimo veřejně dostupný zdrojový kód – přístupové tokeny pro službu Cloudflare Images jsou uloženy v souboru .env, zatímco konfigurace pro službu Google, obsahující klíče a URL pro komunikaci s Firebase, je uložena v samostatném konfiguračním souboru. Oba tyto soubory nejsou verzovány a jsou k aplikačnímu prostředí přidávány až při nasazení pomocí speciálního postupu. Tento přístup zaručuje, že žádné citlivé údaje se neobjeví v distribuovaném balíčku aplikace ani ve veřejně sdíleném kódu.

3.8.9 Uživatelské role a oprávnění

V systému jsou definovány dvě uživatelské role, osobní účet a podnikový účet. Role určují rozsah oprávnění uživatele při práci s aplikací. Největším rozdílem je, že podnikové účty mají rozšířená oprávnění k provádění operací typu CRUD nad událostmi, které spravují, zatímco osobní účty tuto možnost nemají. Ostatní části API jsou zabezpečeny tak, aby žádný uživatel neměl přístup k cizím údajům a nemohl je jakkoli upravovat nebo mazat. Tento model zajišťuje, že uživatelé mohou manipulovat pouze s daty, k nimž mají výslovně přidělená oprávnění.

ZÁVĚR

Cílem diplomové práce bylo navrhnout a vyvinout mobilní aplikaci pro zařízení se systémem Android, která uživatelům zprostředkovává informace o kulturních a společenských událostech v jejich okolí, a zároveň vytvořit podpůrný backend systém. Vedle toho byla pozornost věnována rešerši moderních trendů ve vývoji mobilních aplikací, zejména v oblasti multiplatformního přístupu a integrace API třetích stran. Výsledkem je funkční mobilní aplikace umožňující správu uživatelských účtů s oddělenými rolemi, plnohodnotný CRUD nad událostmi, pokročilé filtrování, mapovou vizualizaci událostí a systém denních notifikací. Součástí řešení je také backend s implementovanými prostorovými funkcemi PostGIS pro vyhledávání dle lokace a kurzorovým stránkováním podporujícím nekonečný feed.

Frontendová část aplikace byla realizována pomocí frameworku Flutter, který umožnil efektivní vývoj moderního a responzivního uživatelského rozhraní. Backend byl vybudován nad technologií Spring Boot v jazyce Java. Využití databáze PostgreSQL s rozšířením PostGIS pak poskytlo pokročilé možnosti práce s geografickými daty, které byly klíčové pro filtraci událostí na základě polohy.

Funkčnost a spolehlivost systému byla ověřena prostřednictvím unit a integračních testů. Unit testy se zaměřily na business logiku a prověřovaly správnost jednotlivých metod a Service tříd, zatímco integrační testy testovaly komplexnější funkce, zejména mechanismus filtrování a práci s geografickými daty v repository vrstvě. Díky této kombinaci bylo možné ověřit, že řešení funguje jak na úrovni izolovaných komponent, tak i při jejich vzájemné spolupráci v rámci celého systému.

Aplikace oproti existujícím řešením přináší přidanou hodnotu zejména díky denním notifikacím, které uživatele aktivně informují o aktuálním dění, mapovému zobrazení událostí a možnosti ukládání oblíbených kategorií s automatickými doporučeními.

Možný budoucí rozvoj aplikace zahrnuje především rozšíření interakcí mezi uživateli, aby aplikace nesloužila pouze jako informační nástroj, ale také jako platforma pro sdílení zkušeností v průběhu událostí. Dalšími směry jsou vylepšení zabezpečení prostřednictvím dvoufaktorové autentizace či rozšíření škály podporovaných kategorií. Jako limitní faktor se ukázala především závislost na placených plánech externích API, přičemž do budoucna by tento faktor mohl být u některých služeb řešen implementací vlastní cache vrstvy.

Celkově lze konstatovat, že vyvinuté řešení naplnilo stanovené cíle a představuje uživatelsky přívětivý a perspektivní systém, který může výrazně zjednodušit přístup k informacím o kulturním a společenském dění v okolí uživatele.

POUŽITÁ LITERATURA

1. ABHISHEK, Jain. Scalable Frameworks for Cross-Platform Mobile App Development. Online. *Journal of Emerging Technologies and Innovative Research*. 2025, article 6. ISSN 2349-5162. [cit. 2025-08-20].
2. JETBRAINS. *Kotlin Mlutiplatform*. Online. Kotlin Mlutiplatform. Dostupné z: https://www.jetbrains.com/kotlin-multiplatform/. [cit. 2025-08-20].
3. META PLATFORMS, INC. *About the New Architecture*. Online. Reactnative.dev. Dostupné z: https://reactnative.dev/docs/intro-react-native-components. [cit. 2025-08-20].
4. META PLATFORMS, INC. *Core Components and Native Components*. Online. Reactnative.dev. Dostupné z: https://reactnative.dev/docs/intro-react-native-components. [cit. 2025-08-20].
5. FLUTTER. *Flutter architectural overview*. Online. Flutter Docs. Dostupné z: https://docs.flutter.dev/resources/architectural-overview. [cit. 2025-08-20].
6. FLUTTER. *Writing custom platform-specific code*. Online. Flutter Docs. Dostupné z: https://docs.flutter.dev/platform-integration/platform-channels#architecture. [cit. 2025-08-20].
7. GOOGLE. *Develop Android apps with Kotlin*. Online. Developers. Dostupné z: https://developer.android.com/kotlin/first. [cit. 2025-08-20].
8. GOOGLE CHROME. *Welcome to Learn Progressive Web Apps!*. Online. Web.dev. Dostupné z: https://web.dev/learn/pwa/welcome. [cit. 2025-08-20].
9. MOZILLA. *Progressive web apps*. Online. Mdn. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps. [cit. 2025-08-20].
13. STRELEC, Michal. *Co je to API?* Online. Michal Strelec. Dostupné z: https://www.strelec.pro/napsal/co-je-to-api. [cit. 2025-08-20].
14. GOOGLE. *Maps products*. Online. Google Maps Platform. Dostupné z: https://mapsplatform.google.com/maps-products/. [cit. 2025-08-20].
15. MAPBOX. *Mapbox Documentation*. Online. Mapbox. Dostupné z: https://docs.mapbox.com/. [cit. 2025-08-20].
16. AMAZON WEB SERVICES, INC. *Amazon Simple Storage Service (S3)*. Online. Aws. Dostupné z: https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html. [cit. 2025-08-20].

17. PAYPAL. *Get started with PayPal REST APIs*. Online. Developer. Dostupné z: https://developer.paypal.com/api/rest/. [cit. 2025-08-20].
18. FIGMA. *What is the difference between UI and UX?* Online. Figma. Dostupné z: https://www.figma.com/resource-library/difference-between-ui-and-ux/. [cit. 2025-08-20].
19. ETHAN, Marcotte. *Our responsive web*. Online. 2014. Dostupné z: https://ethanmarcotte.com/books/responsive-web-design/full/chap01/. [cit. 2025-08-20].
20. APPLE INC. *Human Interface Guidelines*. Online. Developer. Dostupné z: https://developer.apple.com/design/human-interface-guidelines. [cit. 2025-08-20].
21. GOOGLE. *Get Started Get to know Material 3 – from UX guidance and tools to reusable components and open-source code*. Online. Dostupné z: https://m3.material.io/get-started. [cit. 2025-08-20].
22. CHEN, Junxiang; ZHU, Pengyu; YUN, Jing; TIAN, Baixuan; YANG, Yalan et al. *Text Readability of Smartphone in Dark Mode: Effects of Font Type, Font Weight and Color*. Online. Springer, Cham, 2023. ISBN 978-3-031-34866-2. Dostupné z: https://doi.org/https://doi.org/10.1007/978-3-031-34866-2_2. [cit. 2025-08-20].
23. SHAWN LAWTON, Henry. *WCAG 2 Overview*. Online. W3C Web Accessibility Initiatiative WAI. 2005, 06. 05. 2025. Dostupné z: https://www.w3.org/WAI/standards-guidelines/wcag/#whatis2. [cit. 2025-08-20].
24. KUMAR, Keshav a KUMAR PANDEY, Bishwajeer. *Next generation mechanism for data encryption*. Abingdon: CR Press, 2025. ISBN 9781032832845.
25. APPLE INC. *Keychain services*. Online. Developer. Dostupné z: https://developer.apple.com/documentation/security/keychain-services. [cit. 2025-08-20].
26. GOOGLE. *Android Keystore system*. Online. Developers. Dostupné z: https://developer.android.com/privacy-and-security/keystore. [cit. 2025-08-20].
27. LOREDANA, Crisan. *Launching Default End-to-End Encryption on Messenger*. Online. 2023. Dostupné z: https://about.fb.com/news/2023/12/default-end-to-end-encryption-on-messenger/. [cit. 2025-08-20]
28. KIM, Hokeun a A. LEE, Edward. Authentication and Authorization for the Internet of Things. Online. *IEEE Xplore*. 2017, roč. 19, č. 5. ISSN 1941-045X. [cit. 2025-08-20].

29. NÚKIB, NAKIT, MINISTERSTVO VNITRA ČESKÉ REPUBLIKY. *MINIMÁLNÍ BEZPEČNOSTNÍ STANDARD*. Online. 2023. Dostupné z: https://nukib.gov.cz/download/publikace/podpurne_materialy/minimalni-bezpecnostni-standard_v1.2.pdf. [cit. 2025-08-20].
30. EUROPEAN COMMISION. *What does data protection 'by design' and 'by default' mean?* Online. Dostupné z: https://commission.europa.eu/law/law-topic/data-protection/rules-business-and-organisations/obligations/what-does-data-protection-design-and-default-mean_en. [cit. 2025-08-20].
31. EUROPEAN COMMISION. *Data protection explained*. Online. Dostupné z: https://commission.europa.eu/law/law-topic/data-protection/data-protection-explained_en. [cit. 2025-08-20].
32. RUSSELL, Stuart a NORVIG, Peter. *Artificial Intelligence: A Modern Approach 2Nd Ed., chapter 1*. Prentice-Hall Of India Pvt, 2007. ISBN 8120323823.
33. JIM, Holdsorth. *What is NLP (natural language processing)?* Online. IBM. 2024. Dostupné z: https://www.ibm.com/think/topics/natural-language-processing. [cit. 2025-08-20].
34. OPENAI. *Explore ChatGPT*. Online. OpenAI. 2024, 2025. Dostupné z: https://openai.com/chatgpt/overview/. [cit. 2025-08-20]
35. LENSAAI. *Influencers' Best Kept Secret*. Online. 2025. Dostupné z: https://lensa.app/#top. [cit. 2025-08-20].
36. SOPHIA, Jayden. *5 App Development Trends that Could Change How You Develop an App in 2025: GoodFirms Survey*. Online. GoodFirms. 2025. Dostupné z: https://www.goodfirms.co/resources/app-development-trends-change-how-you-develop-an-app. [cit. 2025-08-20].
37. APPLE INC. *Continuity features and requirements for Apple devices*. Online. 2025, 07. 05. 2025. Dostupné z: https://support.apple.com/en-us/108046. [cit. 2025-08-20].
38. SAMSUNG. *Co je Galaxy ekosystém, který umožňuje propojené žití?* Online. Samsung. 2025. Dostupné z: https://www.samsung.com/cz/support/mobile-devices/co-je-galaxy-ekosystem-ktery-umoznuje-propojene-ziti/. [cit. 2025-08-20].
39. SAMSUNG. *Nový způsob, jak si užít svůj domov*. Online. Samsung SmartThings. 2025. Dostupné z: https://www.samsung.com/cz/smartthings. [cit. 2025-08-20].
43. META PLATFORMS, INC. *Facebook Events*. Online. Dostupné z: https://www.facebook.com/events. [cit. 2025-08-20].

45. LIONEL SUJAY, Vailshery. *Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2023*. Online. In: . 2025. Dostupné z: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/. [cit. 2025-08-20].
46. PRESENTMANIA, S. R. O. *Objevte nové akce ve vašem okolí*. Online. Presentie. Dostupné z: https://presentie.cz/. [cit. 2025-08-20].
47. PRESENTMANIA S.R.O. *Presentie*. Online. Google Play. Dostupné z: https://play.google.com/store/apps/details?hl=cs&id=com.presentmania.presentie. [cit. 2025-08-20].
48. ALESSANDRO DEL, Sole. *Visual Studio Code Distilled*. Apress, 2019. ISBN 978-1-4842-4223-0.
49. ASSUMPÇÃO a ORSINE, Hudson. *Getting started with Intellij IDEA*. BIRMINGHAM - MUMBAI: PACKT Publishing, 2013. ISBN 978-1-84969-961-7.
50. POSTMAN. *AI needs context. APIs deliver it*. Online. Dostupné z: https://www.postman.com/. [cit. 2025-08-20].
52. TURNBULL, James. *The Docker Book*. Turnbull, James, 2014. ISBN 978-0-9888202-0-3.
53. COSMINA, Iuliana; HARROP, Rob; SCHAEFER, Chris a HO, Clarence. *Pro Spring 5*. Apress, 2017. ISBN 978-1-4842-2807-4.
54. DAVID, Flanagan. *JAVA IN A NUTSHELL*. Beijing: O`REILLY, 2005. ISBN 978-0-596-00773-7
55. FERRARI, Luca a PIROZZI, Enrico. *Learn PostgreSQL*. BIRMINGHAM - MUMBAI: Packt, 2020. ISBN 978-1-83898-528-8.
56. TIDBTEAM. *Mastering Flyway for Seamless Database Schema Migrations*. Online. TiDB. 2024. Dostupné z: https://www.pingcap.com/article/mastering-flyway-seamless-database-schema-migrations/. [cit. 2025-08-20].
57. MARCO L., Napoli. *Beginning Flutter*. Indianapolis: John Wiley & Sons, 2020. ISBN 978-1-119-55082-2.
58. *GOOGLE. Supported deployment platforms*. Online. Flutter Docs. 2025. Dostupné z: https://docs.flutter.dev/reference/supported-platforms. [cit. 2025-08-20].
59. SKUZA, Bartosz; JANIEC, Jaukub a BARTOSINKA, Inez. *Flutter vs React Native: Complete 2025 Framework Comparison Guide*. Online. In: . Dostupné z: https://www.thedroidsonroids.com/blog/flutter-vs-react-native-comparison. [cit. 2025-08-20].

60. META PLATFORMS, INC. *Get Started with React Native*. Online. Dostupné z: https://reactnative.dev/docs/environment-setup. [cit. 2025-08-20].
61. BHAT, Natesh. *Flutter Vs React Native : Performance Benchmarks you can't miss !*. Online. Dostupné z: https://nateshmbhat.medium.com/flutter-vs-react-native-performance-benchmarks-you-cant-miss-%EF%B8%8F-2e31905df9b4. [cit. 2025-08-20].
62. NOWAK, Maja. *Flutter vs. React Native in 2025 — Detailed Analysis*. Online. Nomtek. 23. 01. 2025. Dostupné z: https://www.citacepro.com/dok/8lco36wjCEAN5GZG. [cit. 2025-08-20].
63. USAMA RIAZ, Muhammad. *Comparative Analysis of React Native, Kotlin, and Flutter for Cross-Platform Mobile Development*. Online, Diplomová práce, vedoucí Adnan Ashraf. Åbo Akademi University: Åbo Akademi University, 2025. Dostupné z: https://www.doria.fi/bitstream/handle/10024/192774/riaz_usama.pdf. [cit. 2025-08-20].
64. GOOGLE. *Pricing plans*. Online. Dostupné z: https://firebase.google.com/pricing. [cit. 2025-08-20].
65. ONESIGNAL. *Transparent pricing for every stage of growth*. Online. OneSignal. Dostupné z: https://onesignal.com/pricing. [cit. 2025-08-20].
66. MAPBOX. *Mapbox pricing*. Online. Mapbox. Dostupné z: https://www.mapbox.com/pricing. [cit. 2025-08-20].
67. GOOGLE. *Google Maps Platform core services pricing list*. Online. Google Maps Platform. Dostupné z: https://developers.google.com/maps/billing-and-pricing/pricing#places-pricing. [cit. 2025-08-20].
68. NOMINATIM. *Open-source geocoding with OpenStreetMap data*. Online. Nominatim. Dostupné z: https://nominatim.org/. [cit. 2025-08-20].
69. CLOUDFLARE. *Pricing*. Online. Cloudflare Docs. Dostupné z: https://developers.cloudflare.com/images/pricing/. [cit. 2025-08-20].
70. GOOGLE. *Cloud Storage pricing*. Online. Google Cloud. Dostupné z: https://cloud.google.com/storage/pricing#europe. [cit. 2025-08-20].