

Mobilní aplikace a systém pro správu konferencí a eventů

Jaroslav Knápek

Bakalářská práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Jaroslav Knápek
Osobní číslo: A23690
Studijní program: B0613A140020 Softwarové inženýrství
Forma studia: Kombinovaná
Téma práce: Mobilní aplikace a systém pro správu konferencí a eventů
Téma práce anglicky: Mobile Application and System for Conference and Event Management

Zásady pro vypracování

- Představte vývojový framework Vue.js, jazyk JavaScript a základní koncepty PWA (Progressive Web Apps).
- Zhodnoťte vhodnost výše uvedených technologií, pro vývoj responzivní webové aplikace pro správu konferencí.
- Stručně popište existující aplikace pro správu konferencí a shrňte jejich výhody či nevýhody.
- Stanovte funkcionální a nefunkcionální požadavky aplikace, navrhnete architekturu systému včetně struktury služeb API a definujte případy užití aplikace včetně aktérů.
- Implementujte aplikaci dle stanovených požadavků a důležité implementační kroky popište.
- V závěru popište výslednou aplikaci a prezentujte dosažené výsledky.

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

Seznam doporučené literatury:

1. DORMANN, Andreas, 2023. *Ionic 7*. Berlin: D&d Verlag Bonn.
2. GARAGUSO, Pablo David a Olaf ZANDER, 2023. *Vue.js 3 design patterns and best practices: Develop scalable and robust applications with vite, Pinia, and Vue Router*. Birmingham: Packt Publishing.
3. GRIFFITH, Christopher, 2017. *Mobile app development with Ionic: Cross-platform apps with Ionic, angular, and Cordova*. Sebastopol, CA: O'Reilly Media.
4. HALLIDAY, Paul, 2018. *Vue.js 2 design patterns and best practices: Build enterprise-ready, Modular vue.js applications with Vuex and Nuxt*. Birmingham: Packt Publishing.
5. KHANNA, Rahat, Hoc PHAN a Sani YUSUF, 2017. *Ionic: Hybrid mobile app development: Create cutting-edge, hybrid mobile applications using the Ionic Framework: A course in three modules*. Birmingham, UK: Packt Publishing.
6. SCHWARZMÜLLER, Maximilian, 2018. *Progressive web apps (PWA) the complete guide* Maximilian Schwarzmüller. B.m.: Packt Publishing.

Vedoucí bakalářské práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

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



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

L.S.

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

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

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

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Práce se zabývá vytvořením mobilní aplikace a systému pro správu konferencí a eventů. Cílem aplikace je umožnit organizátorům konference informovat účastníky o programu konference a umožnit účastníkům pokládat otázky pro prezentující. Na základě analýzy existujících řešení jsou definovány funkční a nefunkční požadavky. Aplikace a systém jsou následně navrženy a implementovány za použití Ionic frameworku, Python Flask backendu a Postgres databáze. Smyslem práce je nabídnout pořadatelům konferencí moderní opensource alternativu ke profesionálním a komerčním konferenčním systémům.

Klíčová slova: progresivní webová aplikace, Ionic framework, Vue.js, Docker, správa konferencí, otázky a odpovědi, architektura systému

ABSTRACT

The thesis addresses the development of a mobile application and system for managing conferences and events. The goal of the application is to enable conference organizers to inform participants about the conference program and allow participants to pose questions to presenters. Based on an analysis of existing solutions, functional and non-functional requirements are specified. The application and system are subsequently designed and implemented using the Ionic framework, a Python Flask backend, and a Postgres database. The aim of the thesis is to offer conference organizers a modern, open-source alternative to professional and commercial conference management systems.

Keywords: progressive web application, Ionic framework, Vue.js, Docker, conference management, questions and answers, system architecture

Na tomto místě bych rád poděkoval mému vedoucímu práce Ing. Radkovi Valovi, Ph.D. za poskytování hodnotné zpětné vazby, přínosných komentářů a také za představení frameworku Ionic a konceptu PWA.

OBSAH

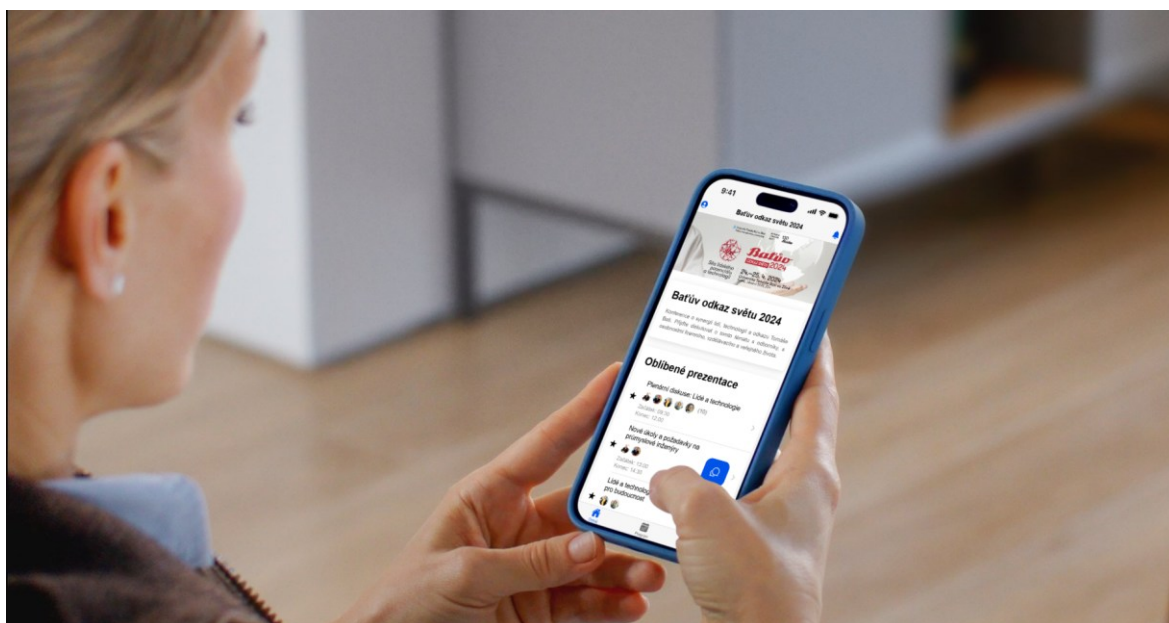
ÚVOD.....	13
I TEORETICKÁ ČÁST	15
1 ÚVOD DO TECHNOLOGIÍ.....	16
1.1 HTML.....	16
1.2 CSS.....	16
1.3 JAVASCRIPT	17
1.4 JSON	17
1.5 NODE.JS	18
1.6 JEST.....	18
1.7 VUE.JS FRAMEWORK.....	19
1.7.1 Vue a Vite	20
1.8 IONIC FRAMEWORK.....	20
1.8.1 Ionic a Vue	21
1.9 PROGRESIVNÍ WEBOVÉ APLIKACE.....	21
1.9.1 Manifest a instalace PWA.....	22
1.9.2 Struktura a obsah souboru manifest	22
1.9.3 Proces instalace PWA	23
1.9.4 Význam manifestu pro uživatelský dojem.....	23
1.9.5 Service Worker.....	24
1.10 EXPRESS.....	24
1.11 AXIOS.....	25
1.12 PRISMA.....	25
1.13 VHODNOST A VYUŽITELNOST VYBRANÝCH TECHNOLOGIÍ	25
1.13.1 Vhodnost Vue.....	26
1.13.2 Vhodnost Ionic Frameworku a PWA	26
1.13.3 Vhodnost technologií pro backend.....	26
2.1 MYIA.....	27
2.1.1 Popis funkcionalit z pohledu účastníka.....	28
2.1.2 Popis funkcionalit z pohledu organizátora	29
2.2 EVENTEE	29
2.2.1 Funkcionality z perspektivy účastníka	30
2.2.2 Funkcionality z perspektivy organizátora	31
2.3 CONFERO.....	32
2.3.1 Funkcionality z perspektivy účastníka	32
2.3.2 Funkcionality z perspektivy organizátora	32
2.4 SHRUTÍ A SROVNÁNÍ	33

II PRAKTICKÁ ČÁST	36
3 NÁVRH APLIKACE	37
3.1 FUNKCIONÁLNÍ A NEFUNKCIONÁLNÍ POŽADAVKY	37
3.1.1 Funkcionální požadavky.....	37
3.1.2 Nefunkcionální požadavky	40
3.2 NÁVRH SYSTÉMU.....	41
3.2.1 Návrh databázového diagramu	43
3.2.2 Návrh API.....	46
3.2.3 Návrh uživatelských obrazovek	47
3.3 PŘÍPADY UŽITÍ A AKTÉŘI.....	52
3.3.1 Aktéři systému.....	52
3.3.2 Případy užití	52
3.4 TOK DAT.....	53
4.1 INICIALIZACE PROJEKTU	55
4.1.1 Struktura repozitáře	55
4.1.2 Verzování repozitáře	55
4.2 IMPLEMENTACE BACKENDU.....	55
4.2.1 Testování backendu	56
4.2.2 Datová vrstva.....	57
4.2.3 Servisní vrstva	58
4.2.4 Prezentační vrstva.....	60
4.3 IMPLEMENTACE FRONTENDU	60
4.3.1 Navigace a routing.....	61
4.3.2 Ionic Modes a platformní styly	62
4.3.3 Komponenty a jejich struktura	64
4.3.4 Správa stavu aplikace	66
4.3.5 Komunikace s backendem.....	66
4.4 VYSTAVENÍ APLIKACE	66
4.4.1 Kontejnerizace aplikace	66
4.4.2 Spuštění kontejnerizovaného projektu	67
4.4.3 Spuštění lokální instance	68
4.4.4 Vystavení do produkčního prostředí	68
5.1 MANUÁL PRO POUŽITÍ APLIKACE	69
5.1.1 Manuál pro organizátora eventu.....	69
5.1.2 Manuál pro prezentujícího.....	69
5.1.3 Manuál pro uživatele aplikace.....	69
ZÁVĚR.....	71
SEZNAM POUŽITÉ LITERATURY	73
SEZNAM OBRÁZKŮ	75
SEZNAM TABULEK.....	76

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	77
SEZNAM PŘÍLOH.....	78

ÚVOD

Současná doba, ve které se mobilní zařízení stala nepostradatelnou součástí existence moderního člověka, představuje éru bezprecedentní technologické symbiózy. Tyto kapesní prostředníky, které nás propojují s globální sítí informací, formují nejen náš každodenní život, ale i způsob, jakým organizujeme společenské události. S rostoucím počtem chytrých telefonů, jehož dynamiku podtrhuje i rozvoj internetové infrastruktury, narůstá exponenciálně i počet mobilních aplikací – jen na platformě Android bylo v roce 2024 evidováno přes 2,7 milionu aplikací. [1] Tato práce si klade za cíl navrhnout a realizovat opensource mobilní aplikaci a systém pro správu konferencí a eventů, která by zaplnila mezeru v dostupnosti nekomerčních řešení a pomohla organizátorům v organizaci a usnadnit efektivní komunikaci s prezentujícími a účastníky.



Obrázek 1 Náhled aplikace pro konference

(zdroj: vlastní, vizualizace s využitím <https://www.canva.com>)

I TEORETICKÁ ČÁST

1 ÚVOD DO TECHNOLOGIÍ

Úvodem bych se rád věnoval krátkému představení technologií, které budu v průběhu mé práce využívat.

1.1 HTML

HTML (Hyper Text Markup Language) je jazyk používaný pro tvorbu a strukturování webových stránek a webových aplikací. Jeho hlavním účelem je definovat obsah webové stránky prostřednictvím různých elementů, jako jsou nadpisy, odstavce, odkazy, obrázky a další multimediální prvky. [2]

První verze HTML byla vyvinuta na počátku 90. let 20. století britským fyzikem Timem Berners-Leem, který pracoval v organizaci CERN. Jeho cílem bylo vytvořit jednoduchý způsob strukturování a sdílení dokumentů v rámci akademického prostředí. V roce 1991 Berners-Lee publikoval první specifikaci HTML, která obsahovala základní prvky jako nadpisy, odkazy a seznamy. S rostoucím významem internetu se HTML začal rychle vyvíjet. V roce 1995 byla vydána verze HTML 2.0, která zavedla standardizované formuláře a tabulky. Následovaly verze 3.2 a 4.01, které rozšířily možnosti jazyka o skriptování pomocí JavaScriptu a podporu kaskádových stylů. [3]

S nástupem chytrých telefonů a mobilních aplikací se objevila potřeba psát aplikace, které budou fungovat na různých platformách bez nutnosti odděleného vývoje pro iOS a Android. Právě zde se prosadily hybridní mobilní aplikace, které kombinují HTML, CSS a JavaScript s nativními prvky operačních systémů prostřednictvím tzv. WebView – ten slouží v mobilním telefonu jako vestavěný prohlížeč a umožňuje aplikacím zobrazit webovou stránku jako by byla její součástí. [3]

1.2 CSS

CSS (Cascading Style Sheets) je jazyk sloužící k definování vzhledu a formátování webových stránek. Byl představen organizací W3C v roce 1996 jako reakce na potřebu oddělit obsah HTML od jeho vizuální prezentace. Hlavním cílem CSS bylo umožnit lepší správu stylů, zjednodušit proces úpravy vzhledu webových stránek a snížit nadměrné používání složitých tabulek a inline stylů. [2]

Postupem času prošel CSS několika verzemi, přičemž nejvýznamnějšími milníky byla verze CSS2, která přinesla podporu pro media queries a lepší kontrolu nad rozvržením

stránky, a CSS3, jež byla rozdělena na několik modulů, což umožnilo nezávislý vývoj jednotlivých částí standardu. CSS3 přineslo podporu zaoblených rohů, přechodových efektů, animací a flexboxu, což významně zlepšilo možnosti návrhu moderních webových aplikací. [4]

1.3 JavaScript

JavaScript (JS) je programovací jazyk používaný především k vývoji interaktivních prvků webových stránek. Původně byl vytvořen Brendanem Eichem ve společnosti Netscape v roce 1995 pod názvem Mocha, poté LiveScript, a nakonec byl přejmenován na JavaScript. Jeho primárním cílem bylo umožnit manipulaci s HTML a CSS v reálném čase, což bylo klíčové pro vznik dynamických webových stránek. [2]

V následujících letech JavaScript prošel významnými změnami, přičemž jedním z klíčových momentů bylo vytvoření standardu ECMAScript, který zajišťuje jednotnost jazyka napříč různými implementacemi. ECMAScript 6 (ES6) v roce 2015 přinesl revoluční změny, jako jsou `let` a `const` pro deklaraci proměnných, šipkové funkce, moduly nebo destrukuralizace, čímž výrazně zlepšil čitelnost a efektivitu kódu. [2]

JavaScript se stal klíčovým prvkem nejen webových, ale i mobilních aplikací. Díky frameworkům, jako je Vue.js, Angular nebo React, umožňuje vývoj moderních single-page aplikací (SPA), které nabízejí plynulý uživatelský zážitek. [5]

JavaScript patří mezi vysokoúrovňové skriptovací interpretované jazyky. Interpretovaný jazyk se za běhu vykonává dle svého zdrojového kódu, nemusí se kompilovat. [6]

1.4 JSON

JSON (Java Script Object Notation) je univerzální datový formát používaný pro výměnu dat mezi serverem a klientem, který vznikl jako jednodušší alternativa k XML. Poprvé byl formálně definován Douglasem Crockfordem na počátku 21. století. Díky své jednoduchosti, lidské čitelnosti a široké podpoře v programovacích jazycích se JSON stal běžným formátem pro komunikaci mezi webovými a mobilními aplikacemi. [7]

JSON je možné ukládat do souborů nebo přenášet po síti, přičemž jeho výhodou je jednoduchost a univerzálnost. Mezi jeho omezení však patří absence nativní podpory pro

binární data a nemožnost explicitního určení znakové sady, ačkoli standardně se používá kódování UTF-8. [5]

1.5 Node.js

Node.js je open-source, multiplatformní běhové prostředí pro JavaScript, které umožňuje spouštět JavaScriptový kód mimo webový prohlížeč. Byl vytvořen v roce 2009 vývojářem Ryanem Dahlem a postaven na jádru V8 – výkonném JavaScriptovém enginu od Googlu, který je používán například v prohlížeči Google Chrome. Hlavní myšlenkou Node.js bylo rozšířit možnosti JavaScriptu ze striktně klientské strany na serverovou, čímž se z JavaScriptu stal univerzální jazyk pro vývoj celého webového stacku. [8]

Node.js je založen na asynchronním, událostmi řízeném modelu, což z něj činí velmi efektivní nástroj pro vývoj aplikací, které musí zpracovávat velké množství požadavků v reálném čase – typicky například chatovací aplikace, API servery nebo streamingové služby. Asynchronní architektura Node.js umožňuje neblokující operace, což je výhodné zejména při práci se souborovým systémem, databázemi nebo síťovou komunikací. [8]

Jedním z klíčových nástrojů, které Node.js poskytuje, je npm (Node Package Manager) – systém, který umožňuje snadnou správu závislostí a instalaci tisíců open-source knihoven, modulů a nástrojů. Díky tomu je vývoj s Node.js velmi flexibilní a rozšiřitelný. [8]

1.6 Jest

Testování softwaru je důležitou součástí vývoje moderních webových i mobilních aplikací. Umožňuje odhalit chyby dříve, než se dostanou k uživateli, zajišťuje vyšší kvalitu kódu a usnadňuje jeho údržbu a refaktoring. V prostředí JavaScriptu existuje několik nástrojů pro psaní a spouštění testů, přičemž jedním z nejpoužívanějších je Jest. [8]

Jest je open-source testovací framework vyvinutý společností Meta (dříve Facebook), který je optimalizovaný především pro práci s Reactem, ale lze ho využít i s jinými knihovnami nebo čistým JavaScriptem. Hlavními výhodami Jestu jsou jednoduchá konfigurace, rychlé spouštění testů, podpora tzv. snapshot testů a integrovaná funkcionalita pro práci s asynchronním kódem. [8]

Jest podporuje několik typů testů:

- Jednotkové testy (unit tests) – testují jednotlivé funkce nebo komponenty izolovaně.

- Integrační testy (integration tests) – ověřují správnou spolupráci více částí aplikace.
- Snapshot testy – ukládají výstup komponenty (například HTML strukturu) a při každém dalším spuštění testu kontrolují, zda nedošlo k nechtěné změně.

Základní test v Jestu může vypadat velmi jednoduše:

```
function soucet(a, b) {  
  return a + b;  
}  
  
test('funkce soucet sčítá dvě čísla', () => {  
  expect(soucet(1, 2)).toBe(3);  
});
```

Jest také poskytuje bohaté možnosti pro mockování funkcí nebo modulů, což je užitečné například při testování komponent, které komunikují s API nebo závisí na externích knihovnách. Díky těmto vlastnostem je Jest vhodným nástrojem nejen pro vývoj backendu webových aplikací, ale i pro větší projekty, kde je potřeba udržovat vysokou kvalitu a stabilitu kódu v čase. [8]

1.7 Vue.js Framework

Vue.js, moderní JavaScriptový framework, se stal nepostradatelným nástrojem pro tvorbu interaktivních webových aplikací. Jeho vznik lze vysledovat do období, kdy vývojáři čelili rostoucí potřebě vytvářet dynamické a modulární rozhraní, která by efektivně reagovala na uživatelské interakce. Vue.js byl navržen tak, aby kombinoval jednoduchost s výkonností, a to prostřednictvím modulárního a asynchronního přístupu, který umožňuje snadnou integraci do již existujících projektů. Framework se vyznačuje především svou lehkostí a flexibilitou. Základní princip Vue.js spočívá v deklarativním programování, kde se vývojáři soustředí na to, jak má uživatelské rozhraní vypadat, namísto komplikovaného manipulování s DOM (Document Object Model). Tato metodologie umožňuje výraznou diferenciaci mezi prezentací a logikou, což výrazně zjednodušuje údržbu a rozvoj aplikací. [9]

Historie Vue.js je také příběhem neustálé evoluce – od prvotních verzí, které nabízely základní reaktivitu, až po současnou verzi, které přináší komplexní ekosystém s podporou pro kompilaci, transpilaci i integraci s dalšími moderními technologiemi. Tento progresivní přístup umožňuje vývojářům pracovat v prostředí, kde je každá změna okamžitě

reflektována v uživatelském rozhraní, což zvyšuje interaktivitu a celkovou dynamiku aplikací. Vue.js se vyznačuje také robustní dokumentací a komunitou, která neustále přispívá k rozvoji frameworku. Ekosystém kolem Vue.js zahrnuje řadu podpůrných knihoven a nástrojů, jež umožňují snadnou injektáž funkcionality do aplikací. Díky modulární architektuře je možné framework snadno modifikovat a přizpůsobovat specifickým požadavkům každého projektu. [9; 10]

V rámci frameworku jsou implementovány různé inovativní přístupy, které zdůrazňují jeho transformativní a syntetickou povahu. Například reaktivní datové vazby umožňují automatickou aktualizaci uživatelského rozhraní, aniž by bylo nutné explicitně volat metody pro aktualizaci DOM. Tato funkce se stala paradigmatickým prvkem moderního vývoje webových aplikací a je často srovnávána s koncepty známými z jiných progresivních frameworků. [9]

1.7.1 Vue a Vite

Vite je moderní buildovací nástroj, který se v posledních letech stal úzce spjatým s vývojem aplikací ve frameworku Vue.js. Za jeho vývojem stojí stejný autor jako za Vue, Evan You [11], což zaručuje vysokou míru kompatibility a optimalizace pro tento framework. Hlavní devízou Vite je jeho rychlost – namísto tradičního bundlování využívá tzv. native ES modules, což umožňuje okamžité spuštění vývojového serveru a téměř okamžité aktualizace během vývoje (tzv. hot module replacement). Tato architektura příjemně usnadňuje vývoj. [12]

Pro Vue.js Vite nabízí speciální pluginy, které umožňují pokročilé možnosti, jako je automatické načítání komponent, optimalizace buildů a přímá integrace s TypeScriptem. Vite podporuje i rozdělení kódu na menší části, což má pozitivní dopad na výslednou velikost aplikace a rychlost načítání. [10]

1.8 Ionic Framework

Ionic Framework byl vytvořen v roce 2013 firmou Drifty Company a původně byl postaven na Angularu a Apache Cordova pro tvorbu hybridních mobilních aplikací. V roce 2018 byl představen Ionic verze 4, který podporoval Vue3 a v nadcházejících verzích tuto podporu rozvíjel a udržoval. V době tvorby této práce se bavíme o verzi 8, která přinesla aktualizace podpory pro Vue 3.0.6+.

Ionic Framework je nástroj, který pomáhá vytvářet multiplatformní mobilní aplikace pomocí webových technologií. Obsahuje sadu vizuálních komponent, které může vývojář využít k tvorbě jeho aplikace. Tyto komponenty jsou silně inspirované nativními prvky na mobilních platformách a dovedou přizpůsobit svůj vzhled dle toho, na jaké platformě jsou právě prohlíženy. Aplikace vyvinuté prostřednictvím frameworku Ionic kombinují nativní kód specifický pro danou platformu s webovými technologiemi, čímž umožňují programátorům přístup k periferním zařízením podle potřeby. Ionic efektivně využívá funkcionality nativní hardwarové akcelerace podporované webovými prohlížeči, například CSS animace, a optimalizuje proces vykreslování. [13]

1.8.1 Ionic a Vue

Ionic nejprve podporoval framework Angular. [14] Jakmile začal nabírat na popularitě framework React, rozhodl se Ionic rozšířit svou podporu i na něj. Třetím v pořadí pak byl Vue 3.

1.9 Progresivní webové aplikace

Progresivní webové aplikace (PWA – Progressive web app) představují moderní přístup k vývoji webových aplikací, jehož cílem je nabídnout uživatelům zážitek co nejbližší nativním mobilním aplikacím, a to při zachování flexibility a otevřenosti webových technologií. Koncept PWA se začal formovat kolem roku 2015, kdy Google poprvé definoval základní principy těchto aplikací. Klíčovou myšlenkou bylo umožnit webovým aplikacím fungovat offline, využívat push notifikace, být instalovatelné na domovskou obrazovku zařízení a spouštět se v samostatném okně bez adresního řádku prohlížeče – tedy tak, aby jejich uživatelské rozhraní i chování odpovídalo nativním aplikacím. [15]

Jednou z výhod PWA je jejich multiplatformnost a snadná přístupnost – uživatelé k nim nepotřebují přístup do aplikačních obchodů, protože se instalují přímo z webového prohlížeče. To významně snižuje nároky na vývoj i distribuci, čímž se PWA stávají atraktivním řešením zejména pro menší projekty nebo aplikace s omezeným rozpočtem. Díky podpoře klíčových API, jako je například Service Worker, lze v PWA efektivně spravovat cache, což umožňuje offline režim a rychlé načítání při opětovném otevření aplikace. [16]

PWA jsou v současnosti podporovány na většině hlavních platform a prohlížečů, jako jsou Google Chrome, Microsoft Edge, Firefox nebo Safari, a to jak na desktopových, tak

mobilních zařízeních. Podpora na platformě Android je výrazně pokročilá, neboť umožňuje přímou instalaci aplikací z webového rozhraní i přístup k mnoha nativním funkcím zařízení. Apple sice podporu PWA v systému iOS postupně rozšiřuje, nicméně stále existují významná omezení, například v oblasti push notifikací či přístupu k některým senzorům zařízení, což snižuje možnosti využití na této platformě.

Mezi nevýhody PWA patří právě neúplná podpora napříč operačními systémy a prohlížeči, což může komplikovat vývoj univerzální aplikace se stejnými funkcemi pro všechny uživatele. Další limitací je omezený přístup k některým nativním funkcím zařízení, jako jsou například Bluetooth, NFC, nebo pokročilé možnosti práce s kamerou a mikrofonom, které jsou u nativních aplikací běžně dostupné. Kromě toho některé platformy, jako například herní konzole (PlayStation, Xbox, Nintendo Switch) či chytré televizory, nepodporují PWA vůbec nebo jen velmi omezeně. Mimo to je obtížné sdělit koncovému uživateli aplikace, že si může tuto stránku nainstalovat jakožto PWA – na Windows a Androidu existuje možnost zobrazit uživateli dialogové okno s žádostí o instalaci například po stisku tlačítka na webové stránce, ale například zařízení s iOS tuto funkcionalitu nenabízí.

1.9.1 Manifest a instalace PWA

Jedním ze stěžejních prvků, které progresivní webové aplikace (PWA) odlišují od tradičních webových stránek, je soubor Web App Manifest. Tento soubor, běžně označovaný jako manifest.json, představuje sémantický popis aplikace, jenž umožňuje prohlížeči chápat webovou aplikaci jako instalovatelný software. Jeho přítomnost je podmínkou pro to, aby mohla být aplikace uživateli nabídnuta k instalaci na domovskou obrazovku nebo pracovní plochu zařízení. Další podmínkou je existující a správně v manifestu specifikovaná ikona aplikace, bez které se mi v Google Chrome nezobrazil dialog pro instalaci mé aplikace. [15]

1.9.2 Struktura a obsah souboru manifest

Manifest je soubor ve formátu JSON, který obsahuje řadu metainformací popisujících vzhled a chování aplikace po její instalaci. Jeho atributy jsou:

- **name** a **short_name**: Plný název a zkrácené označení aplikace, které se zobrazují v různých kontextech, např. v instalačním dialogu či na ploše.
- **icons**: Sada ikon různých rozměrů a formátů
- **start_url**: Adresa, na kterou bude prohlížeč navigován při spuštění ze zástupce.

- display: Mód zobrazení aplikace – například standalone, fullscreen, minimal-ui či browser. Tato vlastnost zásadním způsobem ovlivňuje, zda se PWA spustí v samostatném okně bez prvků prohlížeče, čímž se přibližuje vzhledu nativní aplikace.
- theme_color a background_color: Barvy používané při načítání a v titulku aplikace.

1.9.3 Proces instalace PWA

Instalace progresivní webové aplikace se zásadně liší od tradičních metod distribuce softwaru, jako jsou aplikační obchody. V případě PWA dochází k instalaci přímo z prostředí webového prohlížeče, za splnění několika podmínek – stránka musí být dostupná přes zabezpečený protokol HTTPS, musí mít registrovaný a aktivní Service Worker a validní manifest. [15]

Na podporovaných platformách, zejména v prohlížeči Google Chrome a na zařízeních s operačním systémem Android, může být uživateli nabídnuta možnost instalace pomocí tzv. "Add to Home screen" (Přidat na domovskou obrazovku). Tento mechanismus je iniciován buď automaticky – pokud jsou splněny všechny technické podmínky – nebo explicitně vývojářem prostřednictvím JavaScriptového rozhraní beforeinstallprompt. Tímto způsobem může aplikace prezentovat vlastní instalační dialog, čímž se zvyšuje pravděpodobnost, že si ji uživatel nainstaluje. [15]

Na platformě iOS je situace poněkud odlišná. Přestože Safari základní podporu PWA manifestu implementuje, proces instalace není doprovázen žádným automatizovaným dialogem. Uživatel musí ručně otevřít nabídku sdílení a zvolit možnost "Přidat na plochu". Tato skutečnost dle mě představuje značnou bariéru v šíření PWA mezi uživateli zařízení od společnosti Apple. Osobně však doufám, že Apple bude časem donucen z důvodu konkurenceschopnosti či Evropskou unií tyto důležité detaily implementovat.

1.9.4 Význam manifestu pro uživatelský dojem

Hlavním důsledkem instalace aplikace je to, že se zobrazuje na ploše zařízení jako samostatná ikona, otevírá se ve vlastním okně bez adresního řádku a lze ji spouštět bez nutnosti přístupu k prohlížeči. Tento způsob interakce posiluje uživatelský dojem z aplikace a vytváří iluzi, že se jedná o plnohodnotný software, nikoli jen o webovou stránku. To má

zásadní význam zejména pro aplikace zaměřené na konzumaci obsahu, produktivitu či služby, u nichž je důležitá rychlost přístupu a plynulost uživatelského prostředí. [15]

1.9.5 Service Worker

Dalším stavebním kamenem progresivních webových aplikací je Service Worker. kód v JavaScriptu, jenž operuje nezávisle a na pozadí. Jeho implementace umožňuje webovým aplikacím vystoupit z rámce tradičního modelu klient-server a vstoupit do domény asynchronních, událostmi řízených interakcí, čímž otevírá prostor pro pokročilé funkcionality jako je cachování obsahu, offline dostupnost, přijímání push notifikací či na pozadí probíhající synchronizace dat. [15]

Service Worker je registrován skrze specifické JavaScriptové API, přičemž jeho životní cyklus je důsledně oddělen od samotné webové stránky. Proces registrace se skládá z několika fází – instalace, aktivace a interceptace požadavků (fetch). Během fáze instalace lze předem připravit cache s potřebnými soubory, zatímco při aktivaci se staré verze skriptu nahrazují novějšími a je možné provádět úklid v zastaralých cache. Nejzásadnější roli však Service Worker sehrává v rámci tzv. fetch eventů, kdy zachytává síťové požadavky uživatelského rozhraní a může je dle potřeby směřovat buď do cache, na síť, či je úplně zpracovat sám. [16]

Z hlediska bezpečnostních praktik je nutné zdůraznit, že Service Worker může být registrován pouze na stránkách dostupných skrze zabezpečený protokol HTTPS (s výjimkou vývojového prostředí localhost). Navíc, vzhledem k tomu, že Service Worker funguje mimo běžný běh stránky, nemá přímý přístup k objektům jako window nebo document, což dále redukuje potenciální povrch pro útoky a neautorizovanou manipulaci s obsahem. [17]

1.10 Express

Express.js je minimalistický a flexibilní framework pro Node.js, navržený za účelem zjednodušení tvorby webových aplikací a RESTful API. Původně vyvinutý TJ Holowaychukem v roce 2010, se Express stal de facto standardem pro serverovou logiku v JavaScriptu a páteří mnoha moderních webových systémů. Express abstrahuje nízkourovňové vlastnosti Node.js, jako je správa požadavků a odpovědí, a nahrazuje je elegantní syntaxí a bohatým systémem middleware funkcí. Díky tomu mohou vývojáři rychle implementovat směřování, práci s požadavky typu JSON, statické soubory, autentizaci a další běžné operace. [8]

Významnou výhodou Expressu je jeho modularita. Middleware architektura umožňuje vkládat do řetězce zpracování požadavků libovolné funkce – od logování a validace až po autorizaci nebo práci s databází. [8]

1.11 Axios

Axios je populární HTTP klient pro prohlížeč i Node.js, využívající Promise API. Byl vytvořen s cílem zjednodušit komunikaci s externími API, a nabízí přehledné rozhraní pro realizaci asynchronních požadavků, jako jsou GET, POST, PUT či DELETE. [8]

Axios je široce podporován frontendovými knihovnami jako Vue.js a React a je běžnou volbou tam, kde je požadováno robustní řešení pro výměnu dat mezi klientem a serverem. Oproti nativnímu fetch API poskytuje Axios např. automatickou serializaci JSON dat, interceptory požadavků a odpovědí, nebo podporu time-outů a hlaviček. Axios lze využít nejen na severu, ale také na klientu. [8]

1.12 Prisma

Prisma je moderní ORM (Object-Relational Mapping) nástroj pro Node.js a TypeScript, jehož cílem je zpřístupnit databáze vývojářům pomocí typově bezpečného a intuitivního rozhraní. Prisma umožňuje pracovat s databázemi na vysoké úrovni abstrakce, bez nutnosti přímého psaní SQL dotazů. Díky silné integraci s TypeScriptem vývojář získává přístup ke statickému typování, které výrazně redukuje chyby v době vývoje. [8]

Prisma se skládá z několika komponent: Prisma Client – automaticky generovaný klient pro přístup k databázi, Prisma Migrate – nástroj pro správu databázových migrací a Prisma Studio – vizuální rozhraní pro prohlížení a editaci dat v databázi. Základem je schéma (schema.prisma), které definuje strukturu databáze, včetně relací, indexů, validací i hostitele databáze. [8]

1.13 Vhodnost a využitelnost vybraných technologií

Tuto sadu technologií jsem vybral na základě rozhodnutí, že aplikace bude implantována formou PWA. Původní plán byl vyvíjet nativní aplikaci pro Android a iOS, přičemž jednou z možností byl i Flutter, který také podporuje export do PWA. Nakonec jsem ale zvolil Ionic a Vue, protože jsem měl zkušenosti s webovým vývojem a práce s těmito nástroji mi byla bližší než Flutter, se kterým jsem neměl žádnou praxi. Další výhodou Ionicu byla možnost sdílení kódu mezi platformami bez nutnosti vše psát dvakrát. Ve výsledku byl

pro frontend využít Ionic framework a Vue, pro backend Node.js a Express, pro datovou vrstvu Prisma, pro testování backendu Jest a pro API Axios.

1.13.1 Vhodnost Vue

Vue.js je díky své modulární architektuře a jednoduchosti vhodný pro tvorbu webových aplikací, které mají být udržitelné a snadno rozšiřitelné. Jeho komponentový přístup umožňuje vývoj aplikace v jasně oddělených logických celcích, což zjednodušuje jak samotný vývoj, tak pozdější údržbu a testování. [10] Pro projekty, které jsou hostovány samostatně a nevyžadují nasazení prostřednictvím nativních obchodů s aplikacemi, je Vue.js ideálním kandidátem. Z pohledu dlouhodobé udržitelnosti je výhodou také silná komunita, rozsáhlá dokumentace a široká podpora nástrojů třetích stran.

1.13.2 Vhodnost Ionic Frameworku a PWA

Ionic Framework se vyloženě nabízí pro řešení aplikace pro konference, kterou si bude každá konference sama hostovat. Umisťování aplikace na Google Play a App Store je pro organizátory složitý proces vyžadující finanční vklad a možnost vystavit aplikaci prostřednictvím webu je tudíž velké usnadnění pro vývojáře a také ušetření času pro pořadatele konference. Zároveň ušetří čas při vývoji, jelikož framework disponuje velkou škálou předpřipravených komponentů, aby naše webová aplikace vypadala podobně, jako nativní mobilní aplikace.

1.13.3 Vhodnost technologií pro backend

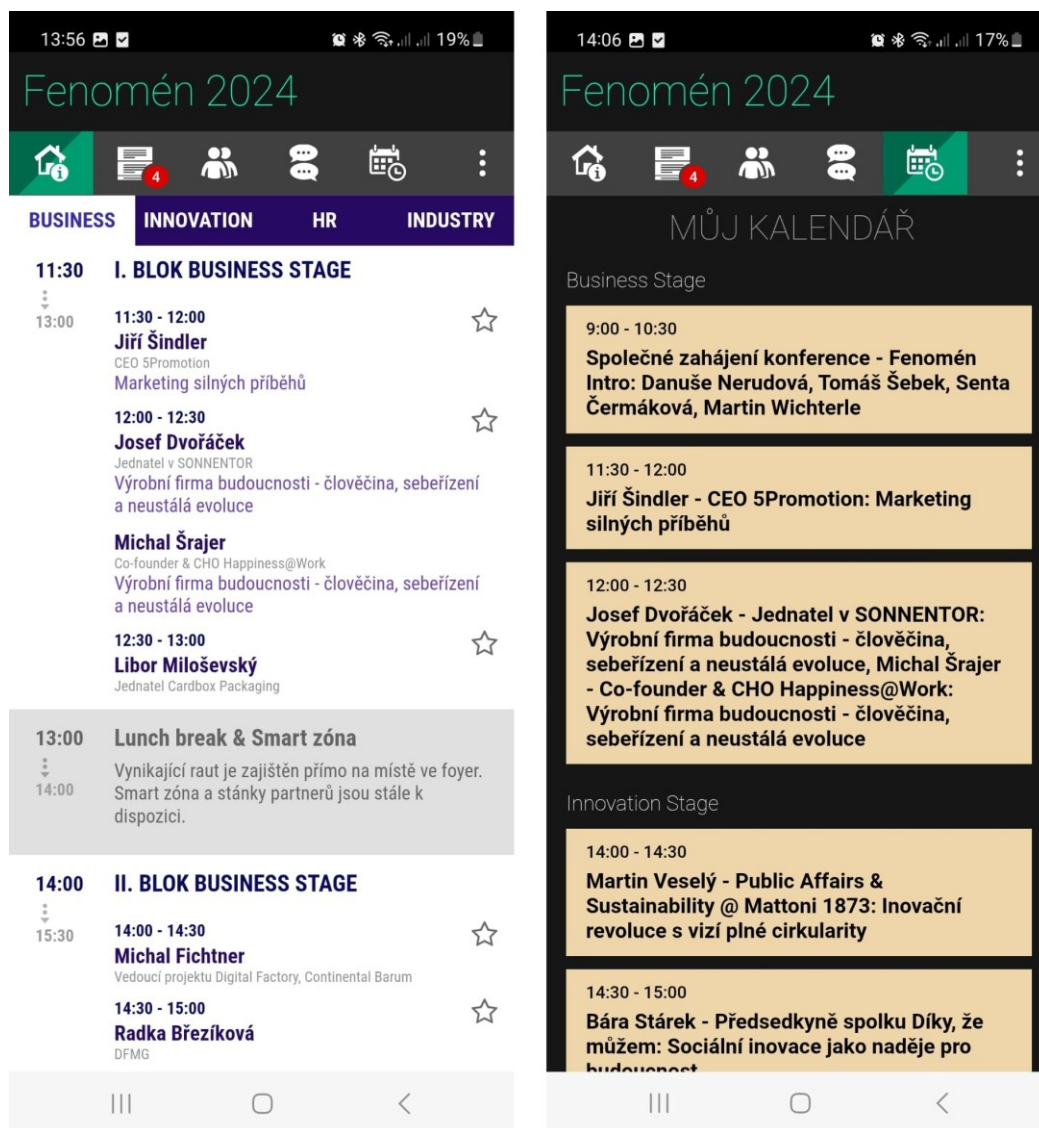
Node.js a Express jsou naprosto běžnými technologiemi pro JavaScriptový backend. Jest jsem vybral z důvodu jednoduchosti psaní testů. Jelikož databázová vrstva naší aplikace bude ORM, nabízí se využít modulu Prisma. Prisma nabízí typově bezpečný přístup k databázi, automatické generování klienta a přehlednou práci se schématy, což výrazně zvyšuje produktivitu vývoje.

2 PŘEHLED EXISTUJÍCÍCH ŘEŠENÍ

Pořadatelé prezenčních konferencí si mohou v dnešní době vybrat z opravdu velkého množství profesionálních komerčních řešení konferenčních aplikací. Z aplikací od českých vývojářů mohu jmenovat například aplikace Myia, Eventee, Attendu, Confero MICE, od zahraničních vývojářů jsem vyhledal aplikace Whova, COMS – Conference Management Software, Cvent, Eventify, vFairs, Bizzabo, Sched, EventMobi, EventMobi, Accelevents, EventsAIR, YesEvents a mnoho dalších aplikací.

2.1 Myia

Myia je eventová aplikace, se kterou mám osobní zkušenost na konferenci Fenomén 2024 ve Zlíně. Zde jsem aplikaci vyzkoušel, otestoval a provedení musím hodnotit na výbornou, protože všechny mnou testované funkcionality, které zmíním později, fungovaly podle předpokladu. Aplikace byla dostupná pro Android i iOS a zvládla obsloužit okolo tisíc účastníků bez menšího problému a neměla výpadky. Na Google play je aplikace hodnocena 4,1/5, dohromady hodnotilo 50 uživatelů. UI aplikace je navrženo intuitivně a přehledně. Aplikace je ve vlastnictví firmy MYIA SYSTEMS s.r.o., podrobnější informace lze nalézt na <https://myia.events/>. [18]



Obrázek 2 Náhled dvou obrazovek aplikace MYIA

(zdroj: vlastní, za využití aplikace Myia)

2.1.1 Popis funkcionalit z pohledu účastníka

Myia nabízí možnost pokládat dotazy na konkrétní přednášce tak, že je vidí ostatní účastníci. Ti na dotazy mohou reagovat prostřednictvím tlačítka „Líbí se“, kterým posouvají nejvíce oblíbené zprávy na vyšší pozice a mají tak vyšší šanci, že budou zodpovězeny prezentujícím. [18]

Mimo chatovací systém nabízí aplikace účastníkům sestavit si vlastní program v průběhu konání akce tím, že hvězdičkou označí danou prezentaci. V případě, že je více prezentací paralelně v jedné sekci si tak může účastník předem vybrat, na kterou prezentaci půjde. Účastníci jsou také informováni o aktuálním dění prostřednictvím nástěnky, kde aplikace umožňuje organizátorům mimo jiné přidávat i hlasování nebo fotky. Novinky jsou

jim doručeny prostřednictvím notifikace v telefonu. Dále aplikace umožňuje účastníkům konference sdílet svoji vizitku pomocí QR kódu a komunikovat prostřednictvím chat rozehrání. [18]

Po ukončení konference jsou data stále dostupná a uživatel má také možnost vracet se do pohledu minulých konferencí.

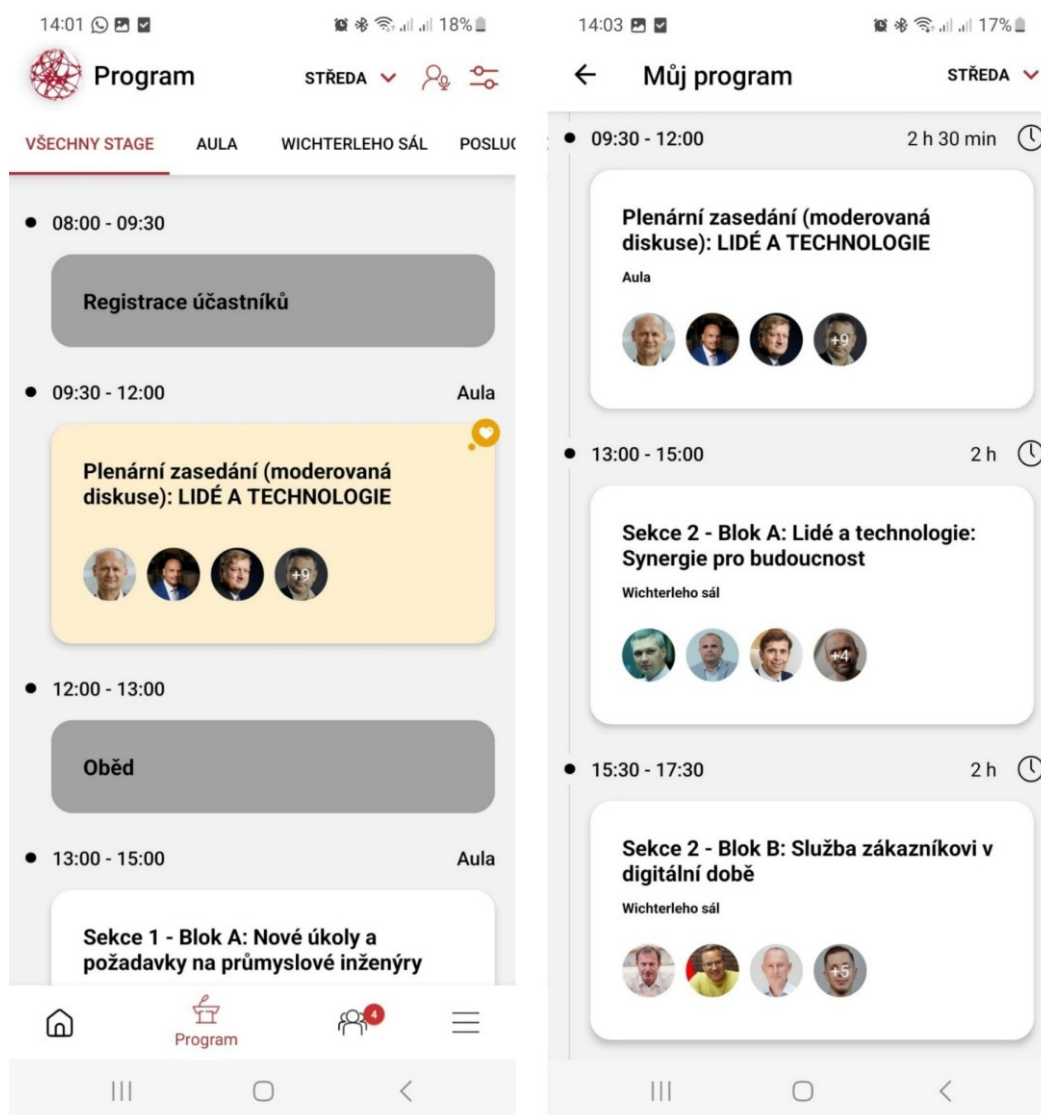
2.1.2 Popis funkcionalit z pohledu organizátora

Organizátor akce může využít služby Branding, díky které si vytvoří individuální úvodní stranu s vlastní grafikou. Organizátoři také velmi ocení, že aplikace splňuje nařízení GDPR, což určitě ocení i uživatelé aplikace. Prostřednictvím administrace si organizátor vytvoří program, kde mohou existovat i dvě prezentace současně. Předem zde zadá prezentující a další detaily prezentací. Také je možnost po ukončení konference exportovat data účastníků společně s jejich profily nebo exportovat aktivitu jednotlivých účastníků. [18]

Za aplikaci v tomto případě musí zaplatit organizátor, na webu ale neuvádějí konkrétní cenu. Na základě mé odeslané cenové poptávky mi byla nabídnuta cena 20 000 Kč za poskytnutí aplikace pro konferenci o 500 účastnících. Za každý profil přednášejícího musí organizátor doplatit 250 Kč.

2.2 Eventee

Eventee je mobilní aplikace od českých vývojářů (firma Eventee s.r.o.) určená pro správu událostí, která si klade za cíl zjednodušit organizaci konferencí, seminářů či jiných akcí a zároveň zvýšit komfort a zapojení jejich účastníků, podrobnější informace lze nalézt na <https://eventee.com/cs>. S aplikací mám osobní zkušenost z konference Baťův odkaz světu pořádané Univerzitou Tomáše Bati ve Zlíně v roce 2024. Na základě vysokého hodnocení této aplikace od uživatelů (například 4,7/5, 620 hodnocení na Google Play) lze usoudit, že jde také o profesionální a funkčně bohaté řešení. Aplikace je dostupná pro platformy Android i iOS a je navržena tak, aby zvládla obsloužit stovky až tisíce účastníků bez výrazných technických obtíží. Její uživatelské rozhraní je popsáno jako intuitivní a esteticky příjemné, což usnadňuje orientaci i méně technicky zdatným uživatelům. [19]



Obrázek 3 Náhled dvou obrazovek aplikace Eventee

(zdroj: vlastní, za využití aplikace Eventee)

2.2.1 Funkcionality z perspektivy účastníka

Eventee nabízí účastníkům širokou paletu nástrojů, které zvyšují interaktivitu a personalizaci jejich zážitku z události. Jednou z klíčových funkcí je možnost vytvořit si individuální program označením vybraných přednášek či workshopů, což je obzvláště praktické při paralelně probíhajících blocích. Účastníci tak mají přehled o svém plánu a mohou se lépe rozhodnout, které části akce navštíví. [19]

Další významnou funkcí je podpora networkingu – aplikace umožňuje vytvořit osobní profil, sdílet kontakty prostřednictvím nástroje pro spárování a komunikovat přes integrovaný chat nebo plánovač schůzek. Tato funkcionalita připomíná systém sdílení

vizitek pomocí QR kódu, jaký nabízí Myia, avšak Eventee jej rozšiřuje o sofistikovanější možnosti propojování účastníků bohatšími nástroji v chatu. [19]

Interakce během přednášek je zajištěna prostřednictvím živých otázek, které mohou účastníci pokládat a ostatní je mohou podpořit hlasováním – podobně jako u Myie s tlačítkem „Líbí se“, i zde se oblíbené dotazy dostávají na přední pozice. Kromě toho aplikace obsahuje sociální stěnu, kde lze sdílet příspěvky, fotografie či reagovat na průzkumy, což přispívá k dynamičtější atmosféře události. Notifikace v reálném čase informují o změnách programu nebo novinkách, přičemž offline režim zajišťuje přístup k datům i bez internetového připojení. Po skončení akce zůstávají informace dostupné, což umožňuje uživatelům vracet se k materiálům či kontaktům – v tomto ohledu se Eventee shoduje s Myiou, která rovněž uchovává data po ukončení konference.

2.2.2 Funkcionality z perspektivy organizátora

Z pohledu organizátorů představuje Eventee komplexní nástroj pro správu událostí s důrazem na jednoduchost a flexibilitu. Prostřednictvím samoobslužného editoru mohou organizátoři rychle vytvořit aplikaci přizpůsobenou konkrétní akci, včetně vlastního vizuálního stylu (logo, barvy, banner). Tato možnost brandingů je srovnatelná s funkcí Myie, avšak Eventee nabízí navíc variantu tzv. bílého štítku, díky čemuž může aplikace vystupovat pod zcela vlastní značkou organizátora na obou hlavních platformách (Google Play a App Store). [19]

Program události lze v administraci snadno nastavit, včetně podpory paralelních přednášek, což odpovídá možnostem Myie. Organizátoři mohou přidávat detaily o prezentujících, vytvářet průzkumy či hlasování a sdílet aktuální informace přes sociální stěnu a notifikace. Kromě toho Eventee přináší pokročilé analytické nástroje – například statistiky zapojení účastníků, analýzu síťení či export dat, což je užitečné pro hodnocení úspěšnosti akce a plánování budoucích událostí. Na rozdíl od Myie, kde cena není veřejně uvedena, Eventee nabízí transparentní cenové plány – například plán Solo za 29 999 Kč pro jednu událost s 500 účastníky. [19]

Další výhodou je soulad s nařízením GDPR, což je klíčové pro ochranu osobních údajů a shoduje se s přístupem Myie. Eventee navíc integruje gamifikaci (body, žebříčky), která motivuje účastníky k aktivitě. [19]

Obě aplikace – Myia i Eventee – sdílejí řadu podobností, jako je intuitivní rozhraní, podpora personalizovaného programu, interaktivní otázky během přednášek či uchování dat

po akci. Zatímco Myia má nejasnou cenovou politiku, Eventee nabízí jasné plány a neomezenou zkušební verzi, což může být pro organizátory rozhodujícím faktorem. [18; 19]

2.3 Confero

Confero MICE je softwarové řešení od společnosti HotelTime Solutions, jehož primárním účelem je usnadnění organizace konferencí, seminářů a dalších událostí, zejména v prostředí hotelů a kongresových center (podrobnější informace k dispozici na stránce <https://www.hoteltime.com/cs/mice/>). S touto aplikací nemám osobní zkušenost, avšak na základě dostupných informací lze konstatovat, že se jedná o nástroj zaměřený především na organizační aspekty akcí, nikoli na komplexní interakci s účastníky. Systém je dostupný prostřednictvím cloudového rozhraní, což zajišťuje jeho přístupnost z různých zařízení s internetovým připojením. Uživatelské rozhraní je navrženo s důrazem na jednoduchost a efektivitu, což usnadňuje práci organizátorům, avšak absence samostatné mobilní aplikace pro účastníky jej odlišuje od řešení jako Myia či Eventee. [20]

2.3.1 Funkcionality z perspektivy účastníka

Z pohledu účastníků nabízí Confero MICE omezenější sadu funkcí ve srovnání s Myia a Eventee, neboť není primárně určeno pro jejich přímé zapojení. Předpokládá se, že účastníci mohou prostřednictvím webového portálu nebo integrované hotelové aplikace získat přístup k programu události a sestavit si základní osobní rozvrh, což je srovnatelné s možností označení přednášek hvězdičkou v Myia či výběru workshopů v Eventee, avšak bez interaktivního charakteru těchto řešení. Účastníci dále mohou přijímat notifikace o změnách v programu či dalších důležitých oznámeních, což odpovídá standardu notifikací v reálném čase nabízených Myia i Eventee. Také lze sdílet materiály či dokumenty související s událostí, pokud je organizátor zpřístupní. Na rozdíl od Myia či Eventee, Confero MICE nenabízí nástroj k networkování. Chybí rovněž funkcionalita pro pokládání otázek během přednášek s možností hlasování, jakou mají obě konkurenční aplikace. [20]

2.3.2 Funkcionality z perspektivy organizátora

Z perspektivy organizátorů představuje Confero MICE nástroj pro správu událostí. Mezi jeho klíčové funkcionality patří možnost podrobného plánování programu, včetně podpory paralelních přednášek a alokace zdrojů, což je obdobné funkcím administrace v Myia a Eventee. Flexibilní úpravy rozvrhu jsou možné díky přednastaveným šablonám, které

zjednodušují reakci na neočekávané změny. Toto řešení dále nabízí integrované řešení pro správu událostí, inventáře a tvorbu nabídek, což přesahuje možnosti Myia a Eventee, jež se zaměřují spíše na interakci než na logistiku. [20]

Pro aplikaci Congero MICE je také unikátní kompatibilita s hotelovými systémy, což umožňuje plynulou koordinaci mezi konferenčními a ubytovacími službami – tato funkce je unikátní a nenachází se u předchozích řešení. [20]

Organizátoři rovněž ocení školení od HotelTime Solutions. Na rozdíl od Eventee, jež nabízí pokročilé analytické nástroje (např. statistiky zapojení), nebo Myia, která umožňuje export dat účastníků, o podobných funkcích u Confero MICE nejsou konkrétní zmínky, avšak lze předpokládat základní exportní možnosti vzhledem k jeho zaměření na správu. Cena služby je individuálně podle rozsahu události.

2.4 Shrnutí a srovnání

Vzhledem k analyzovaným řešením, které jsou již na trhu delší dobu, a dokonce přežily dobu Covidu, lze tvrdit, že po aplikaci pro spravování konferencí existuje poptávka. Níže uvádím v tabulce 1 přehledné srovnání existujících řešení na trhu.

Tabulka 1 Srovnání existujících řešení konferenčních aplikací

FUNKCIONALITA	MYIA	EVENTEE	CONFERO MICE
DOSTUPNOST PLATFÖRM	Android, iOS	Android, iOS	Cloud (webové rozhraní)
INTUITIVNÍ UŽIVATELSKÉ ROZHRANÍ	Ano	Ano	Ano (zaměřeno na organizátory)
PERSONALIZOVANÝ PROGRAM ÚČASTNÍKA	Ano (označení hvězdičkou)	Ano (výběr přednášek/workshopů)	Ano (základní rozvrh přes web)
NETWORKING (VIZITKY, CHAT)	QR kód, základní chat	Profil, pokročilý chat, plánovač	Ne
INTERAKTIVNÍ OTÁZKY S HLASOVÁNÍM	Ano (tlačítko „Líbí se“)	Ano (živé hlasování)	Ne
SOCIÁLNÍ STĚNA/NÁSTĚNKA	Ano (fotky, hlasování)	Ano (příspěvky, průzkumy)	Ne (pouze notifikace)
NOTIFIKACE V REÁLNÉM ČASE	Ano	Ano	Ano
PŘÍSTUP K DATŮM PO AKCI	Ano (i minulé konference)	Ano	Ano (materiály, pokud zpřístupněny)
OFFLINE REŽIM	Ano	Ano	Ne
BRANDING PRO ORGANIZÁTORY	Ano (úvodní strana)	Ano (včetně bílého štítku)	Ne (zaměření na logistiku)
PODPORA PARALELNÍCH PŘEDNÁŠEK	Ano	Ano	Ano
EXPORT DAT ÚČASTNÍKŮ/AKTIVITY	Ano	Ano (včetně analýz sítění)	Základní (předpoklad)
POKROČILÉ ANALYTICKÉ NÁSTROJE GAMIFIKACE	Ne	Ano (statistiky zapojení)	Ne (není zmíněno)
	Ne	Ano	Ne
SOULAD S GDPR	Ano	Ano	Ano (předpoklad)
CENA (500 ÚČASTNÍKŮ)	~20 000 Kč	29 999 Kč	Individuální (není uvedena)
ZKUŠEBNÍ VERZE	Ne (není zmíněno)	Ano (neomezená)	Ne (školení od dodavatele)

(zdroj: vlastní s využitím zdrojů [18; 19; 20])

Na základě provedené analýzy lze konstatovat, že současné konferenční aplikace sdílejí řadu společných rysů, které reflektují potřeby jak organizátorů, tak samotných účastníků akcí. Především je patrné, že všechny sledované aplikace kladou důraz na dostupnost napříč platformami – běžně se tak setkáváme s verzemi pro Android, iOS, případně s webovým rozhraním přístupným z jakéhokoli zařízení. Intuitivní uživatelské rozhraní představuje další klíčový standard, který napomáhá jednoduché orientaci a zvyšuje uživatelský komfort. Personalizace programu účastníka je samozřejmostí, přičemž jednotlivé aplikace umožňují sestavit si vlastní rozvrh podle zájmu o konkrétní přednášky či workshopy. Neméně důležitou funkcionalitou je podpora interakce mezi účastníky. Networking bývá zajištěn prostřednictvím sdílení vizitek, chatovacích funkcí a v některých případech i plánovačů schůzek. V oblasti zapojení účastníků během samotných akcí se často objevuje možnost pokládat otázky řečníkům či hlasovat o zodpovězení podaných otázek. Důraz na aktuálnost informací se promítá do schopnosti zasílat notifikace v reálném čase. Dalším společným prvkem je možnost přístupu k datům a materiálům i po skončení akce, což umožňuje účastníkům se k obsahu kdykoliv vrátit. V neposlední řadě je patrné, že většina analyzovaných řešení myslí i na offline režim, aby účastníci neztratili přístup k programu a důležitým informacím ani při omezeném internetovém připojení. Všechny aplikace deklarují soulad s požadavky GDPR, čímž se snaží garantovat ochranu osobních údajů.

II PRAKTICKÁ ČÁST

3 NÁVRH APLIKACE

Tuto aplikaci jsem se rozhodl vyvinout po konzultaci s prof. Dr. Ing. Drahomírou Pavelkovou pro potřeby Fakulty managementu a ekonomiky či dalších součástí UTB. Právě při řešení otázek ohledně organizace konference Bařův odkaz světu jsme nenalezli žádné volně přístupné řešení pro malé až střední konference. Většina řešení byla spojena s poměrně výraznými finančními částkami, a v případě, kdy jsou akce tohoto formátu organizovány pravidelně, je jejich finanční náročnost poměrně vysoká. Dospěl jsem tedy k závěru, že tvorba opensource řešení pro konference by mohla prospět i ostatním organizátorům konferencí a obdobných akcí v České republice. Finanční aspekt je zde také velmi důležitý, protože existující řešení aplikací pro konference stojí pořadatele v řádu vyšších desítek tisíc korun. To dále komplikuje nutnost minimálně poptávkových řízení pro instituce jako jsou univerzity.

3.1 Funkcionální a nefunkcionální požadavky

Každou softwarová aplikaci lze definovat souborem požadavků, které specifikují, co od systému očekáváme a co by systém dělat neměl, ať už je důvod jakýkoliv. Funkcionální požadavky popisují konkrétní chování systému – tedy, jaké služby poskytuje uživateli a jak reaguje na jednotlivé vstupy. Nefunkcionální požadavky se naproti tomu zaměřují na kvalitu systému – tedy na jeho výkon, bezpečnost, použitelnost a další obecné charakteristiky, které ovlivňují uživatelskou zkušenost, ale přímo neovlivňují samotnou funkci systému. Jelikož je stanovení funkcionálních a nefunkcionálních požadavků zásadou pro vypracování této práce, budu se jim věnovat i v dalších podkapitolách podrobněji. V rámci návrhu aplikace pro správu konferencí bylo nutné tyto dvě kategorie pečlivě vymezit, abychom na ně mohli později navázat konkrétnějším návrhem databázového schématu, api a uživatelských obrazovek.

3.1.1 Funkcionální požadavky

Mezi klíčové funkcionální požadavky naší aplikace patří především možnost správy konferenčního programu – tedy přidávání, editace a mazání bloků přednášek a jednotlivých prezentací. Dále bylo nezbytné umožnit správu prezentujících, včetně přidání jejich profilových informací a přiřazení k příslušným prezentacím. Z hlediska interakce s uživateli bylo důležité implementovat systém pro pokládání otázek během prezentací, včetně možnosti jejich lajkování ostatními účastníky. Administrátoři musí mít možnost spravovat

oznámení pro účastníky, aby mohli efektivně komunikovat změny v programu nebo důležitá upozornění. Funkcionalita správy mediálních souborů, jako jsou obrázky, byla rovněž zařazena mezi základní funkční prvky. Alternativně jsme mohli některé z těchto funkcí přesunout na externí služby (např. správa otázek přes externí chatovací platformu), avšak integrace těchto funkcí přímo do systému nám umožní plnou kontrolu nad uživatelským zážitkem.

Dále bych rád stanovil, co systém nebude umožňovat, abychom zarámovali jeho funkcionalitu. Aplikace nebude podporovat správu jednotlivých uživatelů. Tím opadl problém řešit registraci a přihlašování návštěvníků konference, správců a nutnost řešit uživatelská oprávnění. Rozdělením aplikace na zabezpečenou a nezabezpečenou část univerzálním administračním heslem řešíme požadavky aplikace dostatečně. Z vlastní zkušenosti se mi často stávalo, že na správu uživatelů padlo hodně úsilí a již nezbyly prostředky a čas na dokončení primární funkcionality aplikace a nechci tuto chybu opakovat, speciálně pak v projektu o rozměrech bakalářské práce jednoho řešitele.

V této práci se také nebudu věnovat překladu aplikace. Protože je konference, která vývoj této aplikace iniciovala, v češtině, neshledal jsem jako důležité překládat statický ani dynamický obsah aplikace. Jelikož tento text píšu již po implementaci technického řešení, narazil jsem v průběhu na možnost využít Google překladač zabudovaný přímo v aplikaci, který sice nepůsobí profesionálně a má občas nehodící-se překlady, ale je dostačujícím řešením pro návštěvníky, kteří trvají na překladu obsahu aplikace. Dle mého názoru je lepší věnovat se překladu pořádně a mimo tuto bakalářskou práci, například využitím modulu `i18n`, který je dostupný i pro framework Ionic. [21]

Tabulka 2 Funkcionální požadavky

ID	Požadavek	Popis	Role
FP01	Správa programu	Administrátor může vytvářet, upravovat a mazat časové bloky a přiřazovat k nim prezentace.	Administrátor
FP02	Správa prezentací	Administrátor může přidávat, editovat a mazat jednotlivé prezentace, včetně názvu, popisu, času a místnosti.	Administrátor

FP03	Správa prezentujících	Administrátor může přidávat, upravovat a mazat profily prezentujících (jméno, bio, foto) a přiřazovat je k prezentacím.	Administrátor
FP04	Správa oznámení	Administrátor může vytvářet, upravovat a mazat oznámení pro účastníky (změny programu, důležitá upozornění).	Administrátor
FP05	Správa médií	Administrátor může nahrávat a spravovat obrázky (např. profilové fotky, bannery).	Administrátor
FP06	Moderace otázek	Administrátor může zobrazit seznam otázek položených k prezentacím a označovat je jako zodpovězené.	Administrátor
FP07	Konfigurace konference	Administrátor může nastavit základní informace o konferenci (název, popis, úvodní obrázek).	Administrátor
FP08	Zobrazení programu	Účastník může procházet program konference, vidět časové bloky a detaily jednotlivých prezentací.	Účastník
FP09	Zobrazení prezentujících	Účastník si může prohlížet seznam prezentujících a jejich detailní profily.	Účastník
FP10	Zobrazení oznámení	Účastník vidí oznámení od administrátorů.	Účastník
FP11	Pokládání otázek	Účastník může k vybraným prezentacím pokládat otázky (včetně zadání svého jména).	Účastník
FP12	Lajkování otázek	Účastník může "lajkovat" otázky položené ostatními, aby zvýšil jejich prioritu.	Účastník
FP13	Mazání vlastních otázek	Účastník může smazat otázku, kterou sám položil (pomocí tokenu).	Účastník
FP14	Přístup do administrace	Administrátor se může přihlásit do chráněné sekce pomocí univerzálního hesla.	Administrátor

(zdroj: vlastní)

3.1.2 Nefunkcionální požadavky

Nefunkcionální požadavky byly formulovány s cílem zajistit, aby aplikace nebyla pouze funkční, ale také robustní, bezpečná a pohodlná pro uživatele. Mezi hlavní nefunkcionální požadavky patří například dostupnost napříč platformami, tedy schopnost fungovat jak na webových prohlížečích, tak v mobilních aplikacích (Android, iOS). Aplikace musí vykazovat nízkou odezvu, aby bylo možné plynule pokládat otázky během živých prezentací bez rušivých prodlev. Důraz byl kladen také na bezpečnost, zejména na oddělení aplikace na administrační a veřejnou část a také na ochranu osobních údajů v souladu s GDPR. Z hlediska architektury bylo požadováno, aby aplikace byla modulární a snadno rozšiřitelná, což umožní budoucí integraci nových funkcí, jako například hlasování nebo plánování schůzek. Důležitá je také implementace offline režimu pro případ výpadku připojení, díky kterému využijeme plný potenciál PWA aplikace, která je pro tento scénář připravená.

Aplikace by měla být přístupná i při velkém množství souběžných připojení. Předpokládejme, že na naší konferenci aplikace obsluhuje 500 účastníků a 10 administrátorů. Výsledná aplikace ani v tomto případě nesmí selhat a při komunikaci musí odbavit veškeré požadavky na server nejpozději do 3 sekund. Pro webové stránky platí, že přesahuje-li odezva 3 sekundy, uživatelé si toho začínají všimnout a mohou pociťovat frustraci. [22]

Tabulka 3 Nefunkcionální požadavky

ID	Požadavek	Popis	Metrika / Kritérium
NFP01	Dostupnost	Aplikace musí být dostupná jako Progresivní webová aplikace (PWA) na webových prohlížečích a mobilních zařízeních (Android, iOS).	Funkčnost na hlavních prohlížečích (Chrome, Firefox, Safari) a možnost instalace jako PWA.
NFP02	Odezva systému	Reakce systému, zejména při pokládání otázek a načítání dat, musí být rychlá.	Všechny požadavky na server musí být odbaveny do 3 sekund.
NFP03	Bezpečnost	Musí být zajištěno oddělení veřejné a administrační části. Přístup do administrace je chráněn univerzálním heslem.	Implementace middleware pro ověření hesla; oddělené API endpointy.

NFP04	Ochrana osobních údajů	Aplikace musí být v souladu s GDPR, zejména co se týče mazání uživatelských dat (otázek se jménem).	Možnost smazání otázky autorem.
NFP05	Rozšiřitelnost	Architektura musí být modulární a umožňovat snadné přidávání nových funkcí v budoucnu.	Oddělený frontend a backend, REST API, jasná struktura kódu.
NFP06	Offline režim	Aplikace musí být částečně použitelná i bez připojení k internetu (načtený program, PWA cache).	Základní obsah (program) je dostupný offline.
NFP07	Souběžnost / Zátěž	Aplikace musí zvládnout zátěž konference se cca 500 účastníky a 10 administrátory současně.	Systém nesmí selhat ani výrazně zpomalit při simulované zátěži 510 uživatelů.
NFP08	Použitelnost (UI)	Uživatelské rozhraní musí být intuitivní, přehledné a odpovídat standardům mobilních aplikací, ale bez skrytých prvků.	Pozitivní zpětná vazba od testovacích uživatelů; snadná navigace.
NFP09	Licence	Aplikace bude vyvíjena jako opensource řešení.	Kód bude zveřejněn pod vhodnou opensource licencí.
NFP10	Náklady	Řešení by mělo minimalizovat finanční náklady na provoz a nasazení.	Využití opensource technologií; PWA místo nativních app storů.
NFP11	Jazyk	Aplikace bude primárně v českém jazyce; překlad není prioritou.	Rozhraní a obsah jsou v češtině.
NFP12	Technologie	Využití specifických technologií: Ionic, Vue.js, Node.js, Express, Prisma, Axios.	Aplikace je implementována pomocí stanovených technologií.

(zdroj: vlastní)

Na základě definovaných požadavků bylo možné přistoupit k samotnému návrhu architektury systému a k návrhu databázového schématu, který tyto požadavky reflektuje.

3.2 Návrh systému

V případě vývoje konferenční aplikace bylo rozhodnuto rozdělit systém na dvě hlavní části – frontend a backend. Toto dělení vychází z osvědčených postupů moderního vývoje

softwaru a reflektuje potřebu oddělit prezentační vrstvu aplikace, se kterou pracuje uživatel, od vrstvy zajišťující logiku systému a správu dat. Takto zvolená architektura přináší nejen vyšší flexibilitu, ale rovněž umožňuje nezávislý vývoj a optimalizaci jednotlivých komponent.

Frontend aplikace zahrnuje vše, co uživatel vidí a s čím přímo interaguje. Patří sem uživatelské rozhraní (UI), vizuální design a všechny interaktivní prvky, které tvoří zážitek z používání systému. Pro vývoj uživatelské části naší aplikace jsem vyvinul frontend v Ionic frameworku, který umožňuje vytvářet multiplatformní aplikace s jedním společným kódem a poskytuje moderní vývojové nástroje, jako je podpora komponentového návrhu a snadná integrace s backendem.

Backend se stará o vše, co se odehrává na pozadí aplikace, tedy o logiku systému, zpracování dat, komunikaci se serverem a správu databáze. Historicky vznikla potřeba backendových řešení s rostoucí komplexitou webových stránek a nárůstem požadavků na personalizaci obsahu a zpracování většího objemu uživatelských dat. Původní serverové technologie, jako například PHP, umožnily obsluhovat dynamické požadavky klientů. Postupem času byly vyvinuty další technologie (Python, Ruby, Node.js), které přinesly nové možnosti v oblasti výkonu a škálovatelnosti. V rámci naší aplikace backend poskytuje důležité služby, jako je autentizace, správa databáze konferenčních dat, správa otázek, prezentací a oznámení.

Pro komunikaci mezi frontendem a backendem byl zvolen model Rest API, který se stal standardem pro výměnu dat mezi oddělenými částmi systému díky své jednoduchosti, široké podpoře a snadné rozšiřitelnosti. Alternativně jsem zvažoval i o použití websocketového připojení pro real-time interakce, konkrétně třeba při pokládání dotazů na přednáškách, což však v první fázi vývoje nebylo prioritou vzhledem k nižší složitosti požadovaných funkcí.

Pro vývoj frontendu jsme mohli alternativně zvolit například framework React Native nebo Flutter. Ionic však přináší výhodu jednoduché integrace s webovými technologiemi, na kterých máme postavenou i serverovou část, což značně zrychluje vývoj a snižuje nároky na školení vývojářů. Také bychom museli řešit další problematiku publikování aplikace přímo do obchodů Google Play a App Store, jak jsem již řešil v kapitole nefunkčních požadavků. Co se týče komunikace mezi frontendem a backendem, alternativa použití GraphQL byla zvážena, avšak její vyšší komplexita a nároky na návrh schémat nebyly v

tomto případě ospravedlnitelné. Důvodem pro volbu klasického Rest API je také snadná debuggovatelnost a kompatibilita s řadou existujících nástrojů a knihoven.

3.2.1 Návrh databázového diagramu

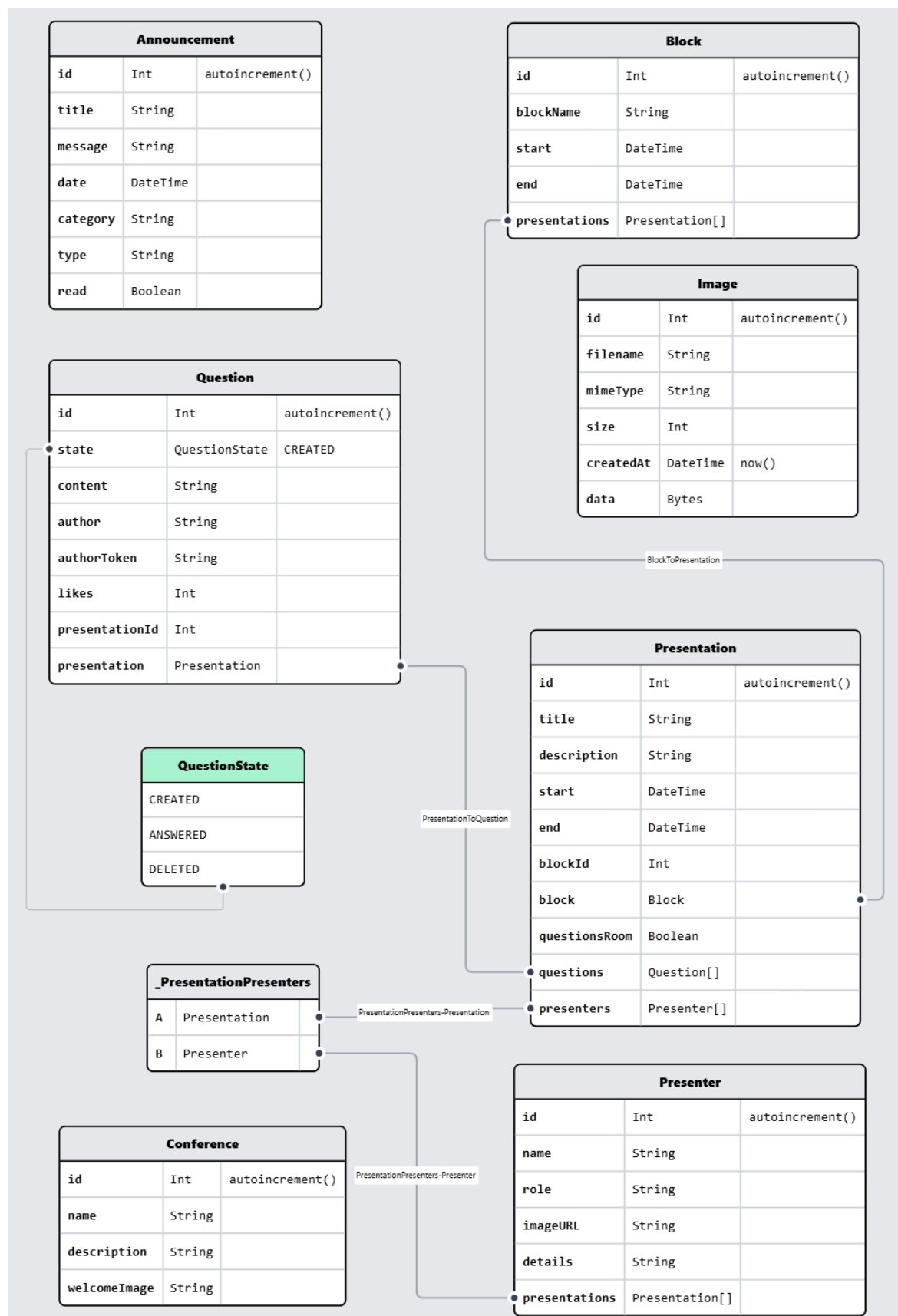
Databázový návrh je základem každé moderní aplikace, neboť správná organizace dat zásadně ovlivňuje nejen výkon systému, ale i jeho dlouhodobou udržitelnost a rozšiřitelnost. Při návrhu databázového schématu pro konferenční aplikaci jsem se držel funkcionálních požadavků a snažil jsem se o rozdělení systému na samostatné a dobře definované entity. Tento přístup umožňuje jednak snadnější pochopení celkové struktury a jednak usnadňuje případné budoucí rozšíření systému.

Klíčovými entitami návrhu jsou zejména modely Announcement, Block, Presentation, Presenter a Question. Tyto modely představují základní stavební kameny celého systému a reflektují potřeby jak organizátorů, tak účastníků konference. Například model Announcement slouží k uchovávání důležitých oznámení, přičemž každé oznámení má svůj titul, zprávu, datum a kategorii. Mohli jsme zde zvažovat možnost mít oznámení jako volný text bez struktury, ovšem díky oddělení jednotlivých atributů lze snadno filtrovat například pouze důležitá upozornění nebo technické zprávy. Model Block zastřešuje jednotlivé časové úseky programu, například současně probíhající přednášky. Do každého bloku pak patří jednotlivé prezentace (Presentation), které mají svůj popis, časový rámec a možnost zařadit místnost pro pokládání otázek. Prezentace vždy náleží existujícímu prezentačnímu bloku a v jednom prezentačním bloku může být žádná, nebo libovolné množství prezentací.

Model Presentation je ústředním bodem interakce účastníků, protože kromě základního popisu uchovává také vztahy na prezentující (Presenter) a otázky (Question). Prezentující a prezentace spojuje vztah M: N, což bylo nutné řešit pomocí explicitní definice relace s názvem PresentationPresenters. Tento způsob nám umožňuje přidělit více prezentérů k jedné přednášce a zároveň jednoho prezentéra k více přednáškám, což je běžné například při panelech nebo diskuzních sekcích. Otázky (Question) pak účastníci pokládají přímo k jednotlivým prezentacím, a každá otázka nese kromě obsahu i autora, token (pro pozdější identifikaci autora při mazání otázky), počet „lajků“ a stav otázky (vytvořená, zodpovězená, smazaná).

Další důležitou součástí systému je práce s multimediálním obsahem. Pro ukládání obrázků (například profilových fotografií prezentujících nebo úvodního banneru konference) byl vytvořen samostatný model Image. Tento model umožňuje kromě

základních informací o souboru (název, typ, velikost, datum vytvoření) také uložení samotných binárních dat. Zde jsem uvažoval i o možnosti ukládat pouze odkazy na soubory uložené na externím úložišti (například Amazon S3), což by mohlo zlepšit výkon databáze, nebo využít externí CDN či samostatnou microservice. S ohledem na požadovanou jednoduchost nasazení první verze systému a snahu minimalizovat závislost na třetích stranách, bylo rozhodnuto o uložení obrázků přímo do databáze. Tento přístup je vhodný zejména pro malé až střední aplikace, jakou má být právě tato konferenční platforma. Výsledný databázový návrh neomezuje vývojáře v budoucích možných změnách ve schématu. Jednotlivé části systému jasně oddělené a návrh umožňuje například snadné přidání nových funkcí jako jsou workshopy, hlasování během prezentací nebo přidání přihlašování heslem a víceúrovňovou správu uživatelských rolí. Přehledná vizualizace navrženého schématu databáze je znázorněna na obrázku 4.



Obrázek 4 Vizualizace schématu databáze

(zdroj: vlastní, vizualizace s využitím <https://prismaliser.app>)

3.2.2 Návrh API

Již ve funkcionálních požadavcích jsem zmínil záměr rozdělit backendovou část aplikace na veřejnou a chráněnou sekci. Stejným způsobem jsem postupoval i při návrhu samotného API. Tento přístup umožňuje nejen lepší přehlednost v kódu a vyšší úroveň organizace jednotlivých částí systému, ale také jasné a transparentní oddělení operací dostupných běžným uživatelům od operací vyhrazených pouze administrátorům a organizačnímu týmu konference. Cílem bylo, aby každý vývojář pracující na projektu měl okamžitě jasnou představu o tom, které části API jsou přístupné veřejnosti a které jsou chráněny autentizací.

Motivací pro toto rozdělení nebyla pouze bezpečnost, ale rovněž snaha o udržení čisté a modulární struktury kódu. V praxi se tedy veřejná část API stará o zpřístupnění informací, které jsou určeny široké veřejnosti – například seznam prezentací, program konference nebo profil prezentujících osobností. Otevřenou API také může využít i administrační portál, abychom nemuseli duplikovat kód pro router. Chráněná část API by měla naproti tomu sloužit k administrativním operacím, jako je správa uživatelů, tvorba a editace oznámení, správa otázek a aktualizace informací o konferenci. Takto navržené členění výrazně napomáhá i implementaci oprávnění a zabezpečení systému.

Struktura samotného projektu by měla dodržet tento návrh přímo ve své adresářové hierarchii. Middleware vrstva (`middleware/auth.cjs`) by měla zajistit ověřování přístupu k chráněné části API. Konkrétní jednotlivé cesty (`routes`) by měly být logicky rozděleny do dvou hlavních skupin – `protected` a `public`. Ve složce `routes/protected` by se měly nacházet všechny endpointy, které vyžadují autentizaci, tedy například:

- `admin.routes.cjs` – kontrola přístupu do administračního portálu
- `announcements.routes.cjs` – tvorba a správa oznámení
- `conference.routes.cjs` – úprava meta informací o konferenci
- `people.routes.cjs` – správa osobností a prezentujících
- `presentations.routes.cjs` – správa přednášek a prezentací, dělba přednášek do bloků
- `questions.routes.cjs` – moderování otázek pokládaných účastníky

Ve složce routes/public by se pak měly nacházet všechny veřejně přístupné endpointy, které poskytují pouze čtecí operace bez nutnosti autentizace. Tyto endpointy by měly sloužit například pro:

- načítání oznámení (announcements.routes.cjs)
- získání informací o konferenci (conference.routes.cjs)
- stahování profilových obrázků (images.routes.cjs)
- seznam prezentujících osobností (people.routes.cjs)
- program prezentací (presentations.routes.cjs)
- pokládání nových otázek (questions.routes.cjs)

Tento přístup by měl umožnit uživatelům přistupovat ke všem potřebným veřejným informacím, zatímco chráněné operace jsou dostupné pouze po úspěšném ověření pomocí přihlašovacích údajů. Pro oddělení přístupu byl měl být využit middleware auth.cjs, který u každého požadavku prověřuje, zda má uživatel oprávnění operaci provést.

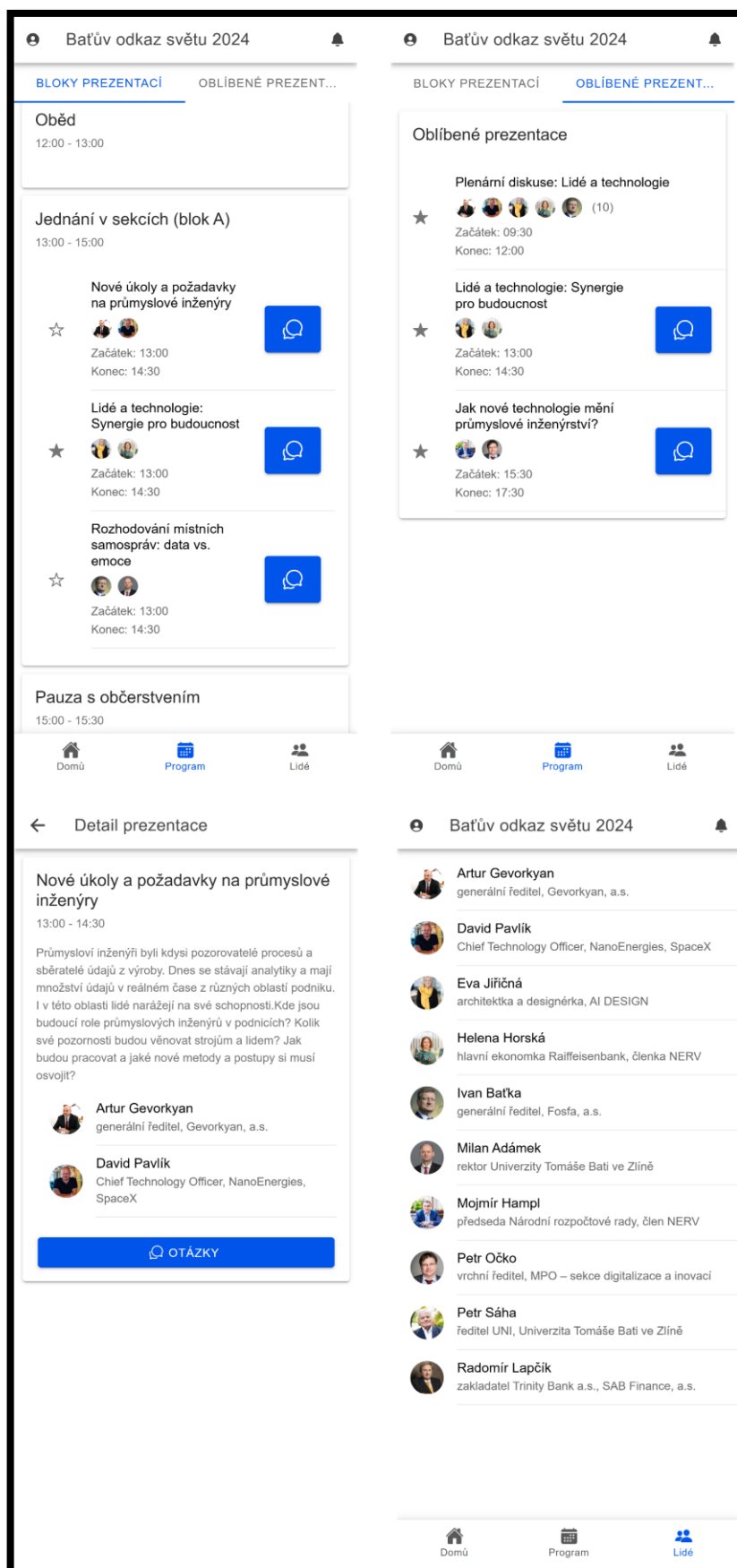
3.2.3 Návrh uživatelských obrazovek

V rámci návrhu uživatelských obrazovek jsem přistoupil k logickému rozdělení aplikace na dvě hlavní části – sekci určenou pro účastníky konference a sekci určenou pro administrátory a organizátory. Stejně jako u návrhu API, i zde bylo mým cílem zajistit maximální přehlednost, jednoduchost a intuitivní ovládání jednotlivých obrazovek.

Uživatelská část aplikace byla navržena s důrazem na požadavek, aby výsledné rozhraní plně odpovídalo standardům mobilních aplikací. Dle mého názoru některé běžné prvky mobilních UI nejsou pro běžného uživatele příliš intuitivní. Proto jsem při návrhu záměrně nevyužil všechny dostupné komponenty frameworku Ionic, ale pouze ty nejčastěji používané a intuitivně srozumitelné. Jako příklad nevyužitého prvku lze uvést tzv. slider pro seznamové položky, který umožňuje mazat položky přejetím doprava či doleva. Tento prvek však není vizuálně nijak označen, a jeho existence tak může být pro běžného uživatele zcela skrytá. Snažil jsem se proto, aby každá akce v aplikaci byla jasně patrná a nevyžadovala žádné „zkoumání“ rozhraní.

Pro návštěvníky konference jsou klíčové zejména obrazovky zajišťující přístup k programu, jednotlivým přednáškám, seznamu osobností a možnosti pokládat otázky – viz obrázek 5. Z hlediska struktury navigace byla aplikace navržena tak, aby hlavní orientační

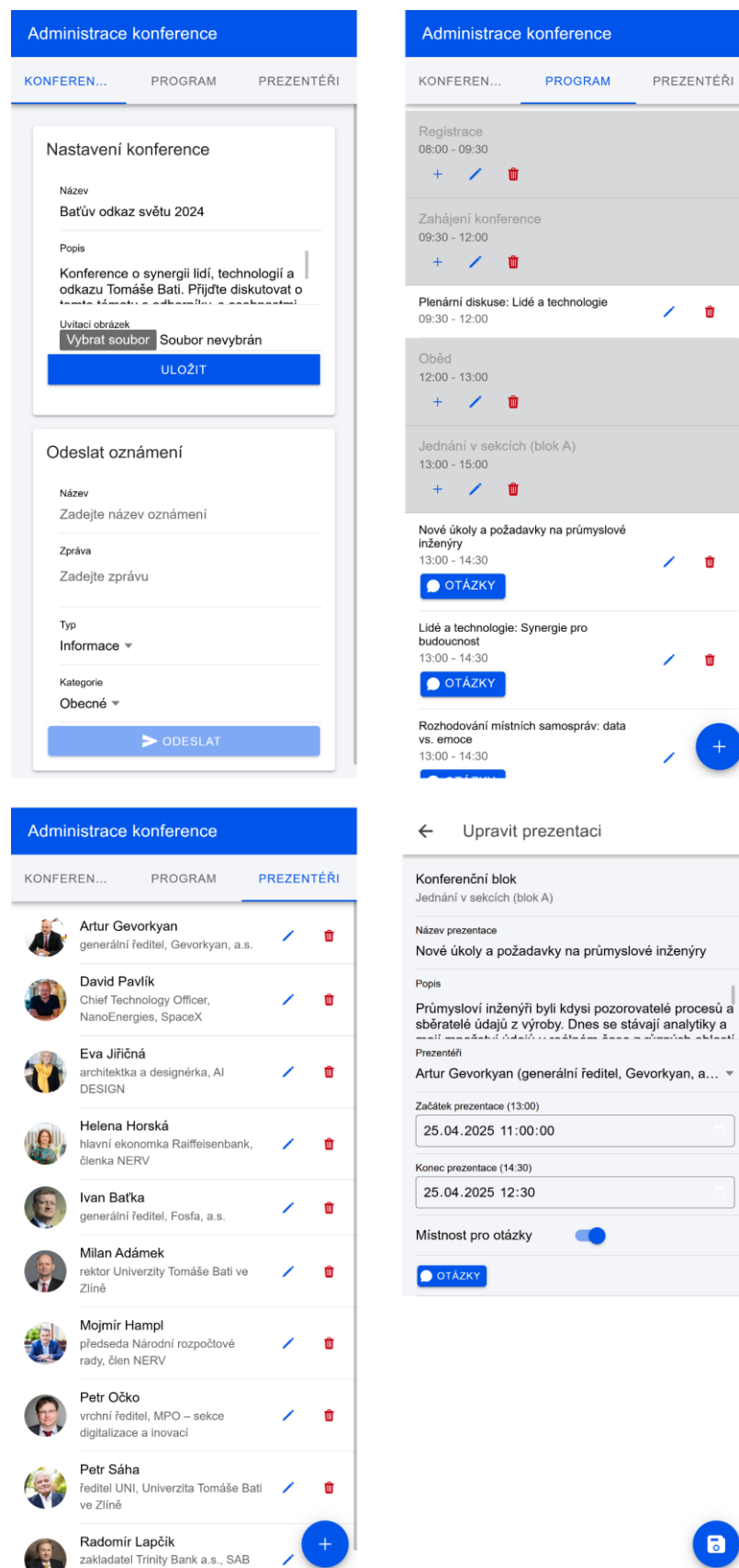
body byly vždy snadno dostupné skrz spodní navigační lištu. Úvodní přesměrování (/) vede uživatele na domovskou obrazovku (/home), která slouží jako hlavní rozcestník. Z této obrazovky se uživatel snadno dostane například na stránku programu (/program), kde jsou jednotlivé bloky a prezentace konference. Každá prezentace má detailní zobrazení (/presentations/:id), včetně možnosti přechodu na sekci otázek (/questions/:id), kde mohou účastníci aktivně pokládat dotazy řečníkům. Součástí aplikace je rovněž sekce osobností (/people), kde si uživatel může prohlédnout seznam prezentérů a po kliknutí zobrazit detailní informace o každém z nich (/people/:id). Uvítací obrazovka (/welcome) účastníky provede instalací aplikace.



Obrázek 5 Náhled obrazovek – bloky prezentací, oblíbené prezentace, detail prezentace a seznam prezentujících

(zdroj: vlastní)

Administrátorská část aplikace byla navržena tak, že přímo nezasahuje do aplikace návštěvníkem konference. Vstupní bod administrátorské části je stránka /admin, která umožňuje přístup k editaci jednotlivých bloků programu (/admin/blocks/:id nebo /admin/blocks/new), spravování prezentací (/admin/presentations/:id a /admin/presentations/new) a správu prezentérů (/admin/presenters/:id a /admin/presenters/new). Každý editor je navržen jako samostatná obrazovka, která umožňuje rychlé přidávání, editaci i mazání jednotlivých záznamů. Pro lepší správu interakce s účastníky byla vytvořena také specializovaná administrátorská sekce otázek (/admin/presentations/:id/questions), kde administrátor vidí všechny zaslané dotazy a může je označovat jako zodpovězené.



Obrázek 6 Náhled obrazovek, zleva na nastavení konference, nastavení programu, nastavení prezentujících a úprava prezentace

(zdroj: vlastní)

3.3 Případy užití a aktéři

Tato kapitola se věnuje definici aktérů systému a odvození případů užití, které konkretizují interakce mezi aktéry a systémem za účelem dosažení specifických cílů. Při formulaci těchto případů užití vycházíme přímo z funkcionálních a nefunkcionálních požadavků definovaných v předchozích kapitolách, čímž zajišťujeme, že všechny klíčové funkce a vlastnosti systému jsou pokryty a zasazeny do kontextu reálného použití.

3.3.1 Aktéři systému

Identifikace aktérů a jejich typických scénářů použití je fundamentálním krokem v návrhu softwaru. Hlavním aktérem našeho systému je bezesporu účastník konference. Účastník je oprávněn načítat informace o konferenci, programu, prezentujících a periodicky si musí načítat notifikace. Dále je oprávněn načítat, vytvářet, lajkovat otázky. Také se nabízí přidat možnost mazat své otázky, a to nejen kvůli možnosti vzít svůj dotaz zpátky, ale také kvůli splnění GDPR o mazání svých osobních dat, čímž může být i jméno účastníka konference zobrazované u otázky.

Druhým aktérem je administrátor konference. Administrátoři mají na starosti konfiguraci aplikace, zvolení názvu, obrázku a popisu konference, vytvoření a konfiguraci konferenčních bloků a prezentací a navedení údajů o prezentujících. Administrátoři mohou vytvářet notifikace, prostřednictvím kterých informují účastníky o novinkách. U prezentací s povolenými dotazy mohou administrátoři zobrazit seznam všech otázek a označovat otázky jako zodpovězené.

3.3.2 Případy užití

Následující odstavce popisují typické scénáře interakce definovaných aktérů se systémem, přičemž tyto scénáře přímo reflektují a rozvíjejí dříve stanovené požadavky.

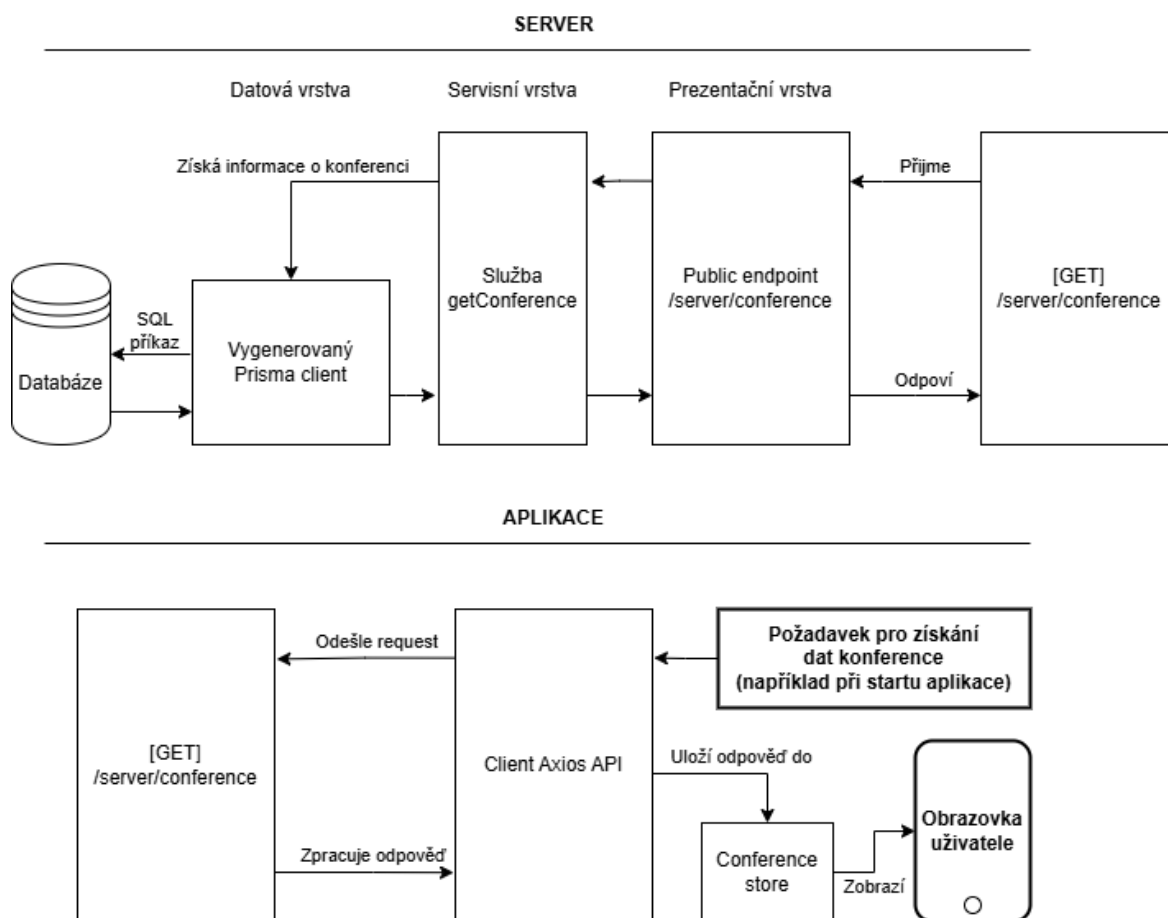
Pro účastníka konference začíná interakce typicky seznámením se s aplikací, což může zahrnovat i její instalaci ve formě PWA pro snazší přístup. Jeho hlavním nástrojem je následně přehled programu, kde může detailně zkoumat jednotlivé bloky a prezentace, zjišťovat jejich časy, místa konání a obsah. S programem úzce souvisí i možnost seznámit se s osobnostmi konference – účastník si může prohlížet profily prezentujících, aby získal kontext k jednotlivým vystoupením. Klíčovou interaktivní funkcí je možnost pokládat otázky k prezentacím, čímž se účastník aktivně zapojuje do diskuse. Tento proces je doplněn možnostmi procházet a lajkovat dotazy ostatních, což pomáhá identifikovat nejžádanější

témata. V průběhu celé konference je účastník příjemcem oznámení, která ho informují o důležitých aktualitách. V souladu s právem na smazání dotazů pak může v případě potřeby odstranit své vlastní dotazy.

Na straně administrátora je prvním krokem autorizovaný přístup do chráněné sekce systému. Po úspěšném přihlášení se mu otevírají možnosti komplexní správy konference. Jeho činnost obvykle začíná nastavením globálních parametrů konference. Následně přechází k detailnímu plánování programu, což zahrnuje vytváření a organizaci časových bloků a naplňování těchto bloků konkrétními prezentacemi, včetně jejich popisů a přiřazení k místnostem. Souběžně spravuje databázi prezentujících, aktualizuje jejich údaje a propojuje je s příslušnými prezentacemi. Během konference administrátor aktivně využívá systém oznámení pro komunikaci s účastníky a v reálném čase monitoruje a moderuje sekci otázek, kde zajišťuje jejich přehlednost a označuje zodpovězené dotazy. Veškeré tyto operace (vytváření, čtení, úprava, mazání) jsou prováděny prostřednictvím specializovaných obrazovek v administrátorské části aplikace.

3.4 Tok dat

Celý proces začíná na straně aplikace, kde klient prostřednictvím Axios API odešle požadavek na získání dat o konferenci (například při spuštění aplikace). Tento požadavek směřuje na veřejný endpoint /conference, který je součástí prezentační vrstvy backendu. Endpoint předá požadavek do servisní vrstvy, kde služba getConference zajišťuje logiku získání potřebných dat. Ta dále komunikuje s datovou vrstvou, konkrétně s generovaným Prisma klientem, který provede odpovídající SQL dotaz na databázi. Výsledkem této interakce jsou získaná konferenční data, která jsou zpětně předána skrz jednotlivé vrstvy až na frontend. Zde Axios klient zpracuje odpověď a uloží data do aplikačního stavu (conference store), odkud jsou následně zobrazena uživateli na jeho zařízení.



Obrázek 7 Diagram datového toku

(zdroj: vlastní, vizualizace s využitím draw.io)

4 IMPLEMENTACE

Nyní známe specifikace aplikace, v teoretické části jsme vybrali a obhájili technologie, takže můžeme přejít k samotné implementaci aplikace. Začal jsem vývojem mock frontendu v Ionic, který zatím nebyl napojen na backend. V dalším kroku jsem přidal server s REST API. V konečném kroku jsem napojil existující mock frontend na server.

4.1 Inicializace projektu

Začneme tím, že inicializujeme nový projekt v nové složce pomocí Ionic CLI.

```
npm install -g @ionic/cli  
ionic start konference
```

Ionic nám nabízí 3 výchozí projekty – blank, sidemenu a tabs. Nám se nejvíce hodil template tabs, který již od začátku disponuje třemi stránkami, mezi kterými lze přepínat ve spodním tabu. Pro instalaci všech nezbytných modulů jsem využil příkaz npm install. Výsledné dependence je možné zobrazit v souboru package.json ve složce projektu, která je pro čtenáře poskytnuta v příloze. Další detaily a postup instalace projektu byly poskytnuty uvnitř souboru README.md.

4.1.1 Struktura repozitáře

Uvnitř našeho nově inicializovaného repozitáře jsem umístil složku server, do které budeme později implementovat backend naší aplikace. Ve složce prisma nalezneme schéma databáze a skript pro naplnění databáze výchozími daty. Struktura složek repozitáře až na tyto dvě výjimky odpovídá běžnému členění Vue projektu.

4.1.2 Verzování repozitáře

Jednotlivé pokroky v repozitáři jsem se rozhodl verzovat za pomoci verzovacího systému GIT. Protože je projekt opensource, je poslední verze aplikace vystavena na adrese <https://github.com/knapejar/conference-app>.

4.2 Implementace backendu

Při vývoji backendové části systému jsem se rozhodl řídit principy vývoje řízeného testy (Test-Driven Development, dále jen TDD), což je metodologie, při níž samotné testy předcházejí implementaci funkčního kódu. Hlavní výhodou tohoto přístupu je možnost návrhu rozhraní a chování jednotlivých částí systému ještě před jejich samotným

vytvořením, čímž se podstatně snižuje pravděpodobnost, že výsledná implementace nebude odpovídat stanoveným požadavkům.

4.2.1 Testování backendu

Testovací framework Jest byl doplněn o mock rozhraní Prisma Client pro přístup k databázi, což umožnilo simulovat databázové operace bez nutnosti přístupu k reálnému úložišti. Tento přístup zajistil deterministické a rychlé testování. Všechny jsou ve složce `/server/__tests__`.

V rámci testovací strategie jsem implementoval několik unit testů, které testují izolované služby a jejich metody. Testy jsou rozděleny do dvou kategorií:

- Veřejné služby (bez prefixu `protected-`)
- Chráněné služby (s prefixem `protected-`)

Nejprve jsem do každého testovacího souboru integroval mockování objektu Prisma Client. Tím jsem vytvořil prostředí, které simuluje reálné databázové operace bez nutnosti připojení ke skutečné databázi. Dále jsem u každého testu pokryl tzv. „happy path“ scénáře, tedy běžné případy, kdy aplikace funguje podle očekávání, mezní hodnoty, ale rovněž i „error scenarios“, které simulují neočekávané vstupy či selhání systému. Součástí každého testovacího bloku byly také přípravné a úklidové kroky, implementované pomocí metod `beforeEach` a po případě `afterEach`. Tyto kroky zajišťovaly, že každý test probíhá ve zcela izolovaném a deterministickém prostředí, bez vlivu předchozích testovacích běhů. Tato struktura testů se v praxi osvědčila jako klíčová z několika důvodů: Přispívá k dlouhodobé stabilitě jádra aplikace, umožňuje rychlou detekci regresních chyb při úpravách kódu, poskytuje formu živé dokumentace toho, jaký je očekávaný výstup jednotlivých funkcí, a výrazně usnadňuje refaktoring, protože vývojář má okamžitou zpětnou vazbu o případném narušení funkcionality.

Pro spuštění testů v projektu použijte tento příkaz:

```
npm run test:server
```

Očekávaný výstup je následující:

```
> conference-app@0.0.0 test:server
```

```
> jest server/__tests__
```

```
PASS  server/__tests__/people.test.js
```

```
PASS  server/__tests__/announcements.test.js
```

```
PASS  server/__tests__/presentations.test.js
PASS  server/__tests__/protected-announcements.test.js
PASS  server/__tests__/protected-questions.test.js
PASS  server/__tests__/protected-conference.test.js
PASS  server/__tests__/protected-people.test.js
PASS  server/__tests__/questions.test.js
PASS  server/__tests__/protected-presentations.test.js
PASS  server/__tests__/conference.test.js
PASS  server/__tests__/protected-admin.test.js
PASS  server/__tests__/questions.routes.test.cjs
PASS  server/__tests__/protected-images.test.js
```

```
Test Suites: 13 passed, 13 total
Tests:      68 passed, 68 total
Snapshots:  0 total
Time:       4.466 s
Ran all test suites matching /server\\__tests__/i.
```

V tomto případě všechny testy prošly a vývojář získal nárok na jedno kafe za odměnu.

4.2.2 Datová vrstva

Též známá jako perzistentní vrstva – v našem projektu bude databáze řešena technologií PostgreSQL – opensource relační databáze, ke které budeme přistupovat pomocí Prisma clientu. V předešlém kroku jsme již vytvořili databázové schéma aplikace. Pro vygenerování Prisma klienta a migraci databáze jsem použil příkaz:

```
npx prisma generate
npx prisma migrate dev --name init
```

Databáze je nyní prázdná, bez žádných dat. Pro testovací účely jsem ve složce prisma vytvořil script seed.ts, kterým databázi naplníme mock daty. Část scriptu seed.ts:

```
const conference = await prisma.conference.create({
  data: {
    name: 'Baťův odkaz světu 2024',
    description: `Konference o ...`,
    welcomeImage: './assets/batuv-odkaz-2024.jpg',
  },
});
```

Tento skript se spouští sám při první migraci. Jednotlivé migrace pak můžeme pozorovat ve složce /prisma/migrations.

4.2.3 Servisní vrstva

Poté, co jsem připravil databázovou vrstvu aplikace jsem pokračoval vývojem jednotlivých služeb. Kód služby pro načítání dat konference ze souboru `/server/services/public/konference.service.cjs`:

```
const { PrismaClient } = require('@prisma/client');
const { createError, HttpError } = require('../../utils/errors.cjs');

const prisma = new PrismaClient();

const getConference = async () => {
  try {
    const conference = await prisma.conference.findFirst();
    if (!conference) {
      throw createError('No conference found', 404);
    }
    return conference;
  } catch (error) {
    if (error instanceof HttpError) {
      throw error;
    }
    console.error('Error getting conference:', error);
    throw createError('Failed to get conference', 500);
  }
};

module.exports = {
  getConference
};
```

Správné chování této služby kontroluje test `/server/__tests__/konference.test.js`, zde je jeho část, která demonstruje, že služba správně vrací detail konference:

```
describe('Conference Module', () => {
  const mockConference = {
    id: 1,
    name: 'Test Conference',
    description: 'Test Description',
    welcomeImage: 'test-image.jpg'
  };
});
```

```
beforeEach(() => {
  jest.clearAllMocks();
});

describe('getConference', () => {
  it('should return conference data', async () => {
    prisma.conference.findFirst.mockResolvedValue(mockConference);

    const result = await getConference();
    expect(result).toEqual(mockConference);
    expect(prisma.conference.findFirst).toHaveBeenCalled();
  });

  it('should throw error when no conference exists', async () => {
    prisma.conference.findFirst.mockResolvedValue(null);
    await expect(getConference()).rejects.toThrow('No conference found');
  });
});
```

Tímto způsobem jsem implementoval všechny veřejné i neveřejné endpointy a v testech jsem se snažil pokrýt běžné scénáře aplikace. Kontrolu přístupu zajišťuje middleware `auth.cjs`, který porovná příchozí hash s hashem administrátorského hesla. Hash hesla je kontrolován při každém přístupu k API. Následující ukázka ze souboru `/server/__tests__/protected-admin.js` demonstruje ověření, že přístup k zabezpečeným endpointům má pouze ten uživatel, který zná heslo:

```
test('should allow access with correct password', async () => {
  const response = await request(app)
    .get('/admin/test')
    .set('Authorization', `Bearer ${TEST_PASSWORD_HASH}`);

  expect(response.status).toBe(200);
  expect(response.body).toEqual({ success: true });
});

test('should deny access with incorrect password', async () => {
  const wrongPassword = 'wrong_password';
  const wrongPasswordHash = crypto.createHash('sha256').update(wrongPassword).digest('hex');
```

```
const response = await request(app)
  .get('/admin/test')
  .set('Authorization', `Bearer ${wrongPasswordHash}`);

expect(response.status).toBe(403);
});
```

Kvalitním otestováním backendu jsem se snažil předejít budoucím problémům při vývoji a zpětně musím zhodnotit, že využití TDD byl správný postup.

4.2.4 Prezentační vrstva

Prezentační vrstvu jsem umístil do složky `/server/routes` a opět jsem oddělil zabezpečenou a nezabezpečenou část umístěním do složek `protected` a `public`. Opět přidávám část kódu – konkrétně endpointu `/konference`:

```
router.get('/', async (req, res, next) => {
  try {
    const conference = await getConference();
    res.json(conference);
  } catch (error) {
    next(error);
  }
});
```

To, že je tento endpoint dostupný na adrese „`/konference`“ a ne „`/`“ stanovuje až `server.js`, ve kterém se prefix přiřadí pomocí:

```
app.use('/conference', publicConferenceRoutes);
```

4.3 Implementace frontendu

Projekt je strukturován do několika klíčových adresářů:

- `src/views/` - obsahuje hlavní stránky aplikace
- `src/components/` - znovupoužitelné komponenty
- `src/router/` - konfigurace routování
- `src/store/` - správa stavu aplikace
- `src/api/` - komunikace s backendem
- `src/theme/` - styly a témata aplikace

4.3.1 Navigace a routing

Dle návrhu uživatelských obrazovek jsem navrhl navigaci uvnitř aplikace. Navigace v aplikaci je řešena pomocí Ionic Routeru. Hlavní struktura routování je definována v souboru `src/router/index.ts`, ve kterém jsou jednotlivé cesty definované následovně:

```
const routes: Array<RouteRecordRaw> = [
  {
    path: '/',
    redirect: '/home'
  },
  {
    path: '/program',
    name: 'Program',
    component: Program
  },
  {
    path: '/questions/:id',
    name: 'Questions',
    component: Questions
  },
  {
    path: '/people',
    name: 'People',
    component: People
  }
  // ... další routy
]
```

Zajímavým prvkem je implementace uvítací obrazovky, která se zobrazí novým uživatelům. Toto je řešeno pomocí navigation guardu:

```
router.beforeEach((to, from, next) => {
  if (to.path === '/welcome') {
    next()
    return
  }

  const savedSettings = localStorage.getItem('userSettings')
  if (!savedSettings) {
    next('/welcome')
    return
  }
})
```

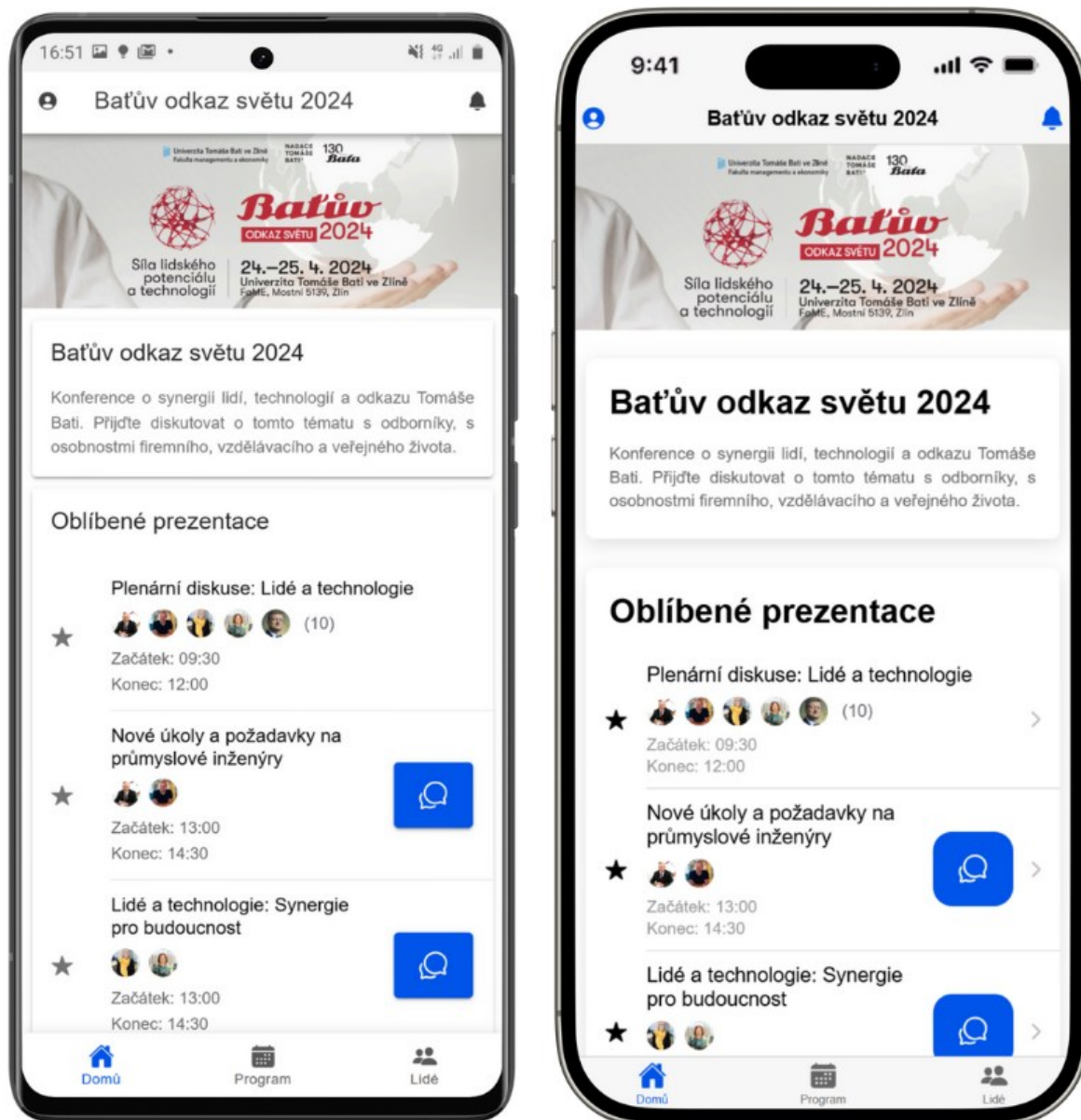
```
}  
})
```

4.3.2 Ionic Modes a platformní styly

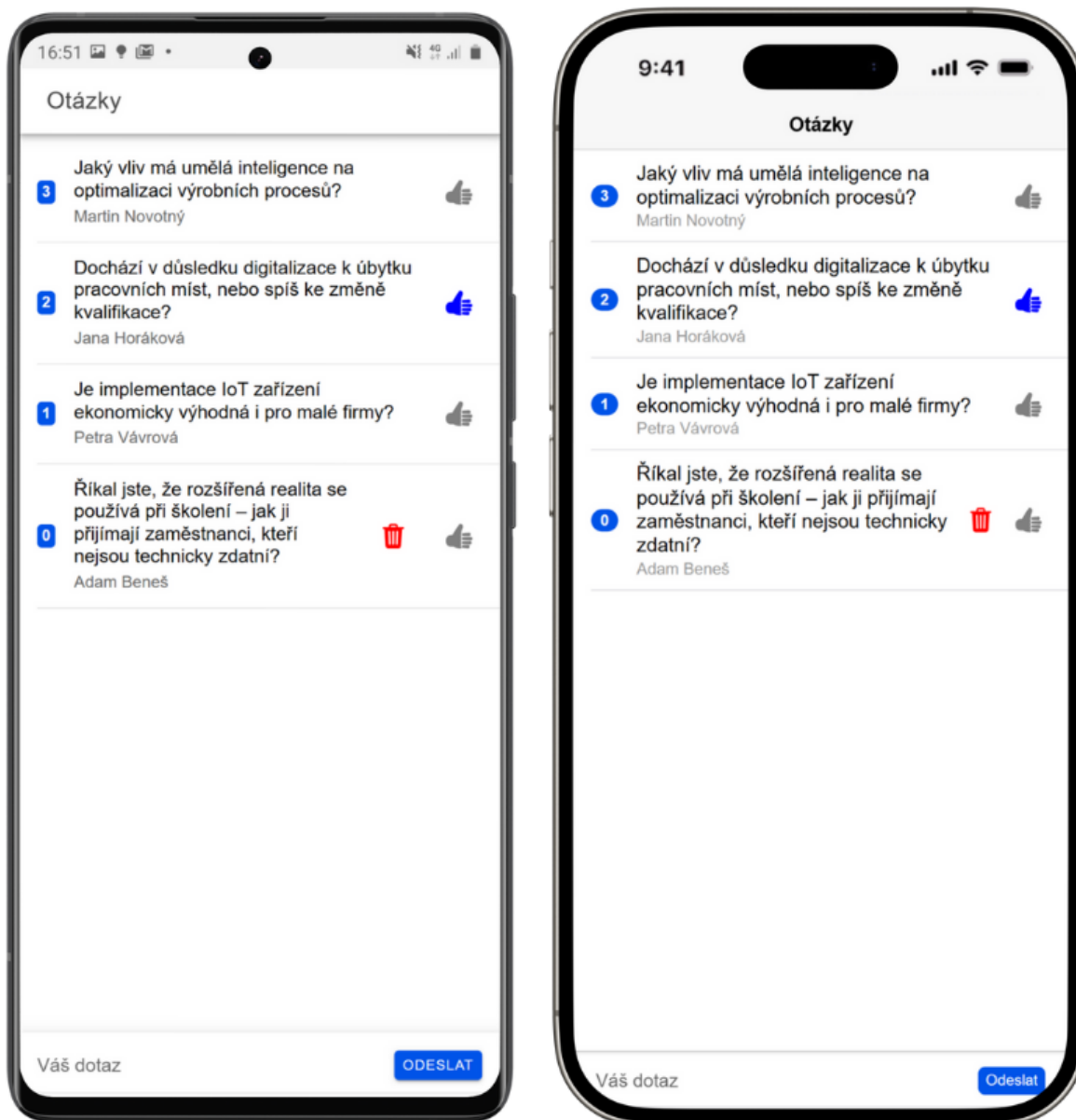
Ionic framework nabízí dva hlavní módy zobrazení – iOS a Material Design (Android). Tyto módy se automaticky přizpůsobují platformě, na které aplikace běží. V projektu aplikace jsem využil tuto funkcionalitu pro zajištění nativního vzhledu na obou platformách. Hlavní rozdíly mezi módy zahrnují:

- iOS mode: zaoblené rohy, tenčí fonty, specifické umístění tlačítek
- Material Design mode: větší prvky, stíny, specifické animace

Tyto styly jsou automaticky aplikovány na všechny Ionic komponenty. Například tlačítko v iOS módu bude mít zaoblené rohy a tenčí font, zatímco v Material Design módu bude mít výraznější stíny a větší padding. Na následujícím obrázku lze vidět tento rozdíl na vyvíjené aplikaci.



Obrázek 8 Rozdíl zobrazení aplikace na Android (vlevo) a iOS (vpravo)



Obrázek 9 Rozdíl zobrazení pohledu otázek na Android (vlevo) a iOS (vpravo)

(zdroj: vlastní)

4.3.3 Komponenty a jejich struktura

Aplikace je postavena na modulárním systému komponent, jejich využití jsem znázornil v diagramu viz. obrázek 10.

- Administrační komponenty (src/components/admin/)
 - AdminPanel, AdminPanelLogin, AdminPanelSettings, AdminQuestionsView, AdminQuestionsList, AdminQuestionsListItem, BlockDateSelector, ConferenceBlockEditor,

ConferenceBlocksManager, ConferenceSettings, NotificationSender,
PresenterEditor, PresentersManager, PresentationEditor

- Komponenty pro prezentace (src/components/presentations/)
 - StarredPresentations, PresentationBlocks, PresentationDetail, PresentationItem, PresentationBlock
- Komponenty pro otázky (src/components/questions/)
 - QuestionsComponent, QuestionsFooterSendQuestion, QuestionsList, QuestionsListItem, QuestionsIconButton, AdminQuestionsList, AdminQuestionsListItem
- Komponenty pro osoby (src/components/people/)
 - PeopleList, PeopleListItem, PersonDetail
- Komponenty pro nastavení (src/components/settings/)
 - Settings
- Komponenty pro oznámení (src/components/announcements/)
 - AnnouncementsBlock, AnnouncementsBlocks
- Komponenty pro layout (src/components/MainLayout/)
 - MainLayout, MainLayoutRefresher, MainLayoutHeaderToolbarBell, MainLayoutHeaderToolbarSettings, MainLayoutNotificationsMenu, MenuLayoutFooterToolbar
- Základní layout komponenty (src/components/base/)
 - BaseLayout
- Komponenty pro ikony (src/components/icons/)
 - AddIcons
- Komponenty pro toasty a upozornění o chybě (src/components/toasts/)
 - ErrorOccured, OfflineWarning, ToastParent
- Komponenty pro úvod a instalaci (src/components/welcome/)

4.3.4 Správa stavu aplikace

Stav aplikace je spravován pomocí Vuex store, který je umístěn v adresáři src/store/. Store obsahuje jednotlivé moduly pro různé části aplikace:

- Úschova hashe hesla pro administraci
- Notifikace
- Detaily konference
- Prezentující
- Prezentační bloky a prezentace
- Otázky na jednotlivých prezencích
- Uživatelské jméno účastníka

Všechny moduly ukládají svůj stav do localStorage, takže pokud aplikace není připojena k internetu, zobrazí svůj poslední stav a toast s neúspěšným připojením. Tím je uživatel upozorněn, že se nedívá na aktuální obsah.

4.3.5 Komunikace s backendem

Komunikace s backendem je umístěna v adresáři src/api je dělena do dvou soborů – pro zabezpečenou a nezabezpečenou část. API volání jsou implementována pomocí Fetch API a jsou obalena v Promise-based rozhraní.

4.4 Vystavení aplikace

V této kapitole se věnuji procesu nasazení aplikace do produkčního prostředí.

4.4.1 Kontejnerizace aplikace

Od samého začátku vývoje jsem se snažil udržet aplikaci jednoduchou, snadno upravitelnou a jednoduše nasaditelnou. Abych usnadnil organizátorům konferencí její spuštění bez zbytečných komplikací, rozhodl jsem se využít technologii Docker, která výrazně zjednodušuje proces nasazení.

Docker umožňuje zabalit celou aplikaci spolu se všemi jejími závislostmi a nastavením prostředí do přenositelného kontejneru. Díky tomu se aplikace chová stejně ve všech prostředích – na vývojářském počítači i v produkci. Pomocí Dockerfile můžeme přesně

definovat prostředí, ve kterém aplikace běží a ve kterém se sestavuje, včetně základního operačního systému, potřebných knihoven a způsobu spuštění aplikace.

Díky využití Dockeru získáváme opakovatelné, izolované prostředí, které omezuje chyby způsobené rozdíly v konfiguraci a zpřístupňuje spuštění aplikace i méně technickým uživatelům. To pomáhá zajistit stabilitu a konzistentní fungování aplikace bez ohledu na to, kde nebo kým je provozována.

4.4.2 Spuštění kontejnerizovaného projektu

Zde jsem popsal spuštění aplikace v případě, že chcete spustit aplikaci za pomoci Dockeru. Postupoval jsem dle souboru README.md v repozitáři. Pro spuštění projektu potřebujete instanci s nainstalovaným Dockerem.

Dále budeme potřebovat přístup k databázi PostgreSQL. Jedna z možností je spustit kontejner s databází za pomoci příkazu:

```
docker run -d \  
  --name my_postgres \  
  -e POSTGRES_USER=postgres \  
  -e POSTGRES_PASSWORD=password \  
  -e POSTGRES_DB=konference \  
  -p 5432:5432 \  
  -v pgdata:/var/lib/postgresql/data \  
  postgres:latest
```

Tímto příkazem jsem vytvořil prázdnou databázi, která ale zatím nic neobsahuje. Abychom vytvořili schéma konference, jeho tabulky a přidali ukázková data konference, musíme použít následující příkazy:

```
npm install prisma --save-dev  
npx prisma generate  
npx prisma migrate dev --name init
```

Poté sestavíme kontejner aplikace spuštěním tohoto příkazu ze složky projektu, což může trvat kolem 3 minut nebo více:

```
docker build -t conference_image .
```

A pro spuštění kontejneru na portu 8000 spustíme tento příkaz:

```
docker run -d \  
  --name conference_demo \  
  -p 8000:80
```

```
-e  
DATABASE_URL=postgresql://postgres:password@host.docker.internal:5432/konference \  
-e VITE_API_BASE=http://localhost:8000 \  
-e ADMIN_PASSWORD=admin123 \  
--link my_postgres \  
konference_image
```

Poté navštívíme stránku <http://localhost:8000> pro ověření, že je aplikace nasazena správně. Do administrace se dostaneme na adrese <http://localhost:8000/admin>.

Při spouštění kontejneru je nezbytné specifikovat tři proměnné:

- DATABASE_URL slouží ke specifikaci databáze, ke které se má aplikace připojit. Tento parametr je povinný a bez něj nebude aplikace fungovat.

- VITE_API_BASE specifikuje adresu, na které je hostovaný backend. Opět se jedná o povinný parametr. Pokud je aplikace spuštěna v Docker image, pak je nutné zde zadat celou URL adresu aplikace + /server – například <https://konference.jk9.eu/server>. Alternativní možností je otevřít kontejneru port 3000, na kterém je server hostován. Uvnitř kontejneru je /server přesměrován na port 3000, čímž se zajišťuje oddělení serveru a klientu.

- ADMIN_PASSWORD je heslo pro přístup do administrace aplikace. Pokud není specifikováno, výchozí hodnota hesla je default_admin_password.

4.4.3 Spuštění lokální instance

Docker lze využít jak k vystavení aplikace do produkčního prostředí, tak pro vývoj. V našem projektu se ale budeme bavit pouze o prvním zmiňovaném způsobu využití.

4.4.4 Vystavení do produkčního prostředí

Pro vystavení do produkčního prostředí můžeme využít například službu <https://render.com/>, která umožňuje vystavení Docker kontejneru a zároveň také řeší plně spravované TLS certifikáty, čímž zajistíme zabezpečené připojení s aplikací s https:// na začátku naší adresy. Služba Render také nabízí Postgres databázi s prostorem 1 GB na měsíc zdarma, takže je alternativním řešením pro vystavení bezplatné instance aplikace.

K vyzkoušení je aplikace vystavena za pomoci Dockeru zde: <https://konference.jk9.eu/>.

5 VÝSLEDNÁ APLIKACE

Výsledkem implementační části této práce je funkční aplikace splňující zadané požadavky. Aplikaci jsem testoval na několika Android zařízeních, emulovaném i reálném iOS zařízení i na zařízení s Windows za pomoci různých prohlížečů. Aplikaci se mi ve všech případech podařilo nainstalovat a bez problému ji využívat.

5.1 Manuál pro použití aplikace

V této kapitole stručně popíšu, jak aplikace funguje a jak jí využít, ať už jste organizátor konference, prezentující, nebo účastník.

5.1.1 Manuál pro organizátora eventů

Pro vystavení aplikace doporučuji organizátorovi oslovit DevOps inženýra, který dle předchozí kapitoly věnované vystavení aplikace připraví novou instanci aplikace a případně na ní nasměruje doménu dle požadavků organizátora.

Pro přihlášení do administrace musí administrátor navštívit stránku /admin a zadat administrátorské heslo. Na první obrazovce administrace se nachází menu pro nastavení domovské obrazovky konference a odeslání nových notifikací. Pro úvodní obrázek doporučuji dobře čitelný obrázek ve rozlišení a formátu připraveném pro publikaci na web v poměru 2:1–3:1, aby se správně ukazoval jak na mobilních zařízeních, tak na laptotech. Další dvě obrazovky pak intuitivně umožňují návrh programu konference a správu přednášejících.

5.1.2 Manuál pro prezentujícího

Jak otevřít stránku s dotazy od diváků

Pro zobrazení místnosti s dotazy se přihlaste do aplikace. Po navštívení stránky aplikace nahraďte v adresní řádce stránku /admin a zadat administrátorské heslo. Následně vyberte menu prezentací, vyhledá jeho prezentaci a stiskne tlačítko, které otevře místnost s otázkami. Tyto otázky můžete označovat jako zodpovězené.

5.1.3 Manuál pro uživatele aplikace

Instalace aplikace – dle OS:

V případě Android prosím využijte prohlížeč Google Chrome. Po navštívení stránky aplikace stiskněte tři tlačítka a zvolte tlačítko přidat na domovskou obrazovku. Následně vyberte první možnost – instalovat.

V případě iOS prosím využijte prohlížeč Safari. Po navštívení stránky aplikace klepněte na ikonku sdílení a vyberte možnost přidat na plochu.

V případě Windows také doporučuji prohlížeč Google Chrome. V adresním řádku se zobrazí ikona „Instalovat aplikaci“, stačí na ní kliknout a potvrdit dialog tlačítkem Instalovat.

Dále už jen zbývá zadat uživatelské jméno, které bude zobrazováno u otázek a návštěvník se dostane na domovskou obrazovku. Spodní navigační lištou můžete přecházet na přehled prezentací a na seznam prezentujících.

Pokud prezentace nabízí místnost pro otázky, zobrazí se tlačítko pro vstoupení do místnosti pro pokládání otázek. V této místnosti může účastník položit jeho dotaz, lajkovat dotazy ostatních uživatelů a mazat své položené dotazy.

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout a úspěšně realizovat mobilní aplikaci a systém pro komplexní správu konferencí a eventů. Výsledkem je plně funkční, moderní a flexibilní platforma, která je připravena k okamžitému nasazení v reálném provozu.

Jedním z nejvýznamnějších aspektů vytvořeného řešení je jeho opensource charakter. Veškerý zdrojový kód je volně dostupný v oficiálním repozitáři na adrese <https://github.com/knapejar/conference-app>. To nejen umožňuje komukoliv na světě kód převzít, libovolně upravit a využít bez jakýchkoliv omezení, ale také otevírá dveře pro komunitní vývoj a další vylepšování. Hlavní přidanou hodnotou vzniklé aplikace je její široká podpora zařízení, které je dosaženo využitím konceptu Progresivní webové aplikace (PWA), zajišťující bezproblémový chod na desktopových i mobilních platformách. Tato technologická volba rovněž přispívá k potenciálně delšímu životnímu cyklu aplikace. Za zmínku rovněž stojí vysoká míra nastavitelnosti aplikace, která pro svou základní konfiguraci a přizpůsobení konkrétní události nevyžaduje žádné programátorské znalosti ze strany organizátora.

V průběhu práce se podařilo naplnit veškeré stanovené funkcionální požadavky. Aplikace tak disponuje nástroji pro správu konferenčního programu, prezentujících, mediálních souborů, systémem pro pokládání a lajkování otázek během prezentací či efektivní distribuci oznámení účastníkům. Neméně důležité bylo i splnění nefunkcionálních požadavků, kde byl kladen důraz na rychlou odezvu systému, bezpečnost dat, oddělení administrační a veřejné části a částečnou funkcionalitu v offline režimu.

Teoretická část práce detailně seznámila čtenáře s použitými moderními technologiemi, včetně frameworků Vue.js pro reaktivní uživatelské rozhraní a Ionic pro tvorbu multiplatformní PWA. Dále byly vzneseny a podloženy argumenty pro jejich vhodnost a popsána architektura backendu postaveného na Node.js, Express a ORM Prisma. Součástí teoretické části byla také důkladná analýza a srovnání funkcionalit vybraných existujících aplikací pro správu konferencí, což pomohlo definovat klíčové vlastnosti vlastního řešení.

Praktická část práce představovala proces softwarového inženýrství, zahrnující návrh architektury systému, databázového schématu a API, až po samotnou implementaci jednotlivých modulů a jejich důkladné testování. Vývoj backendové části byl veden principy Test-Driven Development (TDD), což přispělo ke kvalitě a spolehlivosti výsledného kódu.

Pro zájemce o tuto aplikaci či její budoucí uživatele byl sepsán také podrobný uživatelský manuál, který usnadňuje orientaci v systému a jeho administraci.

Pro zajištění co nejjednoduššího nasazení a distribuce projektu byla implementována kontejnerizace aplikace pomocí Dockeru. Tento přístup minimalizuje problémy s kompatibilitou prostředí a zjednodušuje proces spuštění aplikace na téměř jakémkoliv serveru. Navíc, pro maximální uživatelské pohodlí, je přímo v souboru README.md hlavního repozitáře umístěno tlačítko "Deploy to Render". Tato funkcionality umožňuje komukoliv, i bez hlubších technických znalostí, vystavit plně funkční instanci aplikace na internet během několika minut. Stačí zadat výchozí heslo pro administraci a aplikace je dostupná, a to i bez nutnosti počáteční finanční investice – zcela zdarma. Pro větší množství účastníků pak doporučuji využít placeného hostingu pro vyšší spolehlivost aplikace.

Věřím, že výsledná aplikace představuje hodnotný a prakticky využitelný nástroj, který má potenciál zjednodušit organizaci mnoha konferencí a eventů, a to nejen na Zlínské akademické půdě. Její otevřenost a snadná dostupnost jsou příslibem pro její další organický růst a adaptaci komunitou.

SEZNAM POUŽITÉ LITERATURY

- [1] Number of available apps in the Google Play Store from 2nd quarter 2015 to 2nd quarter 2024. *Statista* [online]. [2025] [cit. 2025-05-07]. Dostupné z: <https://www.statista.com/statistics/289418/number-of-available-apps-in-the-google-play-store-quarter/>
- [2] DEAN, John. *Web Programming with HTML5, CSS, and JavaScript*. Jones & Bartlett Learning, 2018. ISBN 9781449395926.
- [3] BARTOŠ, Pavel a David ZEJDA. *Html: průvodce tvorbou www stránek v HTML*. Praha: Evropská akademie vzdělávání SE, 2024. ISBN 978-80-909154-7-3.
- [4] MICHÁLEK, Martin. *CSS: moderní layout*. [Praha]: Martin Michálek - Vzhůru dolů, [2022]. ISBN 978-80-88253-07-5.
- [5] BROWN, Ethan. *Learning JavaScript*. 3rd edition. O'Reilly Media, 2016. ISBN 9781491914939.
- [6] *JavaScript: řešené úlohy*. Praha: Česká technika - nakladatelství ČVUT, 2025. ISBN 978-80-01-07377-3.
- [7] POWERS, Shelley. *JavaScript Cookbook*. O'Reilly Media, 2010. ISBN 9781449395926.
- [8] MCGREGOR, Milecia. *Full Stack JavaScript Strategies*. O'Reilly Media, 2025. ISBN 9781098122218.
- [9] NELSON, Brett. *Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch*. 18th. Apress, 2018. ISBN 9781484237816.
- [10] FRANKLIN, Jack, Michael WANYOIKE, Ahmed BOUCHEFRA, Kingsley SILAS, Nilson JACQUES, Chad A. OMOLE a Michiel MULDER. *Working with Vue.js*. Sitepoint, 2019. ISBN 978-1-4920-7144-0.
- [11] Meet the Team. *Vue.js - The Progressive JavaScript Framework* [online]. c2014 - 2025 [cit. 2025-05-09]. Dostupné z: <https://vuejs.org/about/team>
- [12] Getting Started. *Vite* [online]. c2019 - 2025 [cit. 2025-05-09]. Dostupné z: <https://vite.dev/guide/>
- [13] MANGLIK, Rohit. *Cross Platform Mobile Application Development*. EduGorilla Publication, 2024. ISBN 9789358804317.
- [14] DABIT, Nader. Ionic React - First Look. *DEV Community* [online]. 2019, 17. 10. 2019 [cit. 2025-05-05]. Dostupné z: <https://dev.to/dabit3/ionic-react-first-look-104l>
- [15] SINGH, Baljeet. *Ionic 4 - Build PWA and Mobile Apps with Angular*. Packt Publishing, 2019. ISBN 9781839218002.
- [16] SHRESTHA, Pritam Suwal. Service Workers | Architecture and Usage. *Medium* [online]. 2021 [cit. 2025-05-08]. Dostupné z: <https://medium.com/@pritam.shrestha/service-workers-architecture-and-usage-5b1e98731cc3>
- [17] Using Web Workers. *MDN Web Docs* [online]. c1998 - 2025 [cit. 2025-05-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers
- [18] MYIA SYSTEMS S.R.O. *MYIA* [online]. c2015 - 2025 [cit. 2025-05-11]. Dostupné z: <https://myia.events/>

- [19] EVENTEE S.R.O. *Eventee* [online]. [2025] [cit. 2025-05-11]. Dostupné z: <https://eventee.com>
- [20] HOTELTIME SOLUTIONS A.S. *Confero MICE* [online]. [2025] [cit. 2025-05-11]. Dostupné z: <https://www.hoteltime.com/cs/mice/>
- [21] KUTANOV, Roman. Ionic internationalization i18n tutorial with ngx-translate. *Lokalise* [online]. 2024, 20. září 2024 [cit. 2025-05-05]. Dostupné z: <https://localise.com/blog/ionic-translation-and-internationalization/>
- [22] SCHULTZ, Dominic. The Impact Of Loading Speed On User Experience. *DIGON DESIGN* [online]. 28 October 2024 [cit. 2025-05-14]. Dostupné z: <https://digondesign.com/blog/web-design/the-impact-of-loading-speed-on-user-experience/>

SEZNAM OBRÁZKŮ

Obrázek 1 Náhled aplikace pro konference	13
Obrázek 2 Náhled dvou obrazovek aplikace MYIA.....	28
Obrázek 3 Náhled dvou obrazovek aplikace Eventee	30
Obrázek 4 Vizualizace schématu databáze	45
Obrázek 5 Náhled obrazovek – bloky prezentací, oblíbené prezentace, detail prezentace a seznam prezentujících.....	49
Obrázek 6 Náhled obrazovek, zleva na nastavení konference, nastavení programu, nastavení prezentujících a úprava prezentace	51
Obrázek 7 Diagram datového toku	54
Obrázek 8 Rozdíl zobrazení aplikace na Android (vlevo) a iOS (vpravo).....	63
Obrázek 9 Rozdíl zobrazení pohledu otázek na Android (vlevo) a iOS (vpravo).....	64

SEZNAM TABULEK

Tabulka 1 Srovnání existujících řešení konferenčních aplikací.....	34
Tabulka 2 Funkcionální požadavky	38
Tabulka 3 Nefunkcionální požadavky	40

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API Application Programming Interface

CSS Cascading Style Sheets

GDPR General Data Protection Regulation pdf

HTML HyperText Markup Language

JS JavaScript

JSON JavaScript Object Notation

PWA Progressive Web Application

REST API Representational State Transfer Application Programming Interface

SDK Software Development Kit

TDD Test-Driven Development

UI User Interface

UX User Experience

W3C World Wide Web Consortium

SEZNAM PŘÍLOH

Příloha P I: Zdrojový kód