Gymnázium Thomase Manna

MATURITNÍ PRÁCE

Přihlašovací systém pro žáky Gymnázia Thomase Manna

Filip Troníček

Vedoucí práce: Mgr. Martin Jelínek

Ročník/předmět : 8. / Informační a komunikační technologie

Školní rok: 2023/2024

Obsah

Úvod

Gymnázium Thomase Manna nabízí několikrát ročně odlehčení od každodenního vyučování ve formě různorodých akcí. Navíc nabízí žákům ve vyšších ročnících i možnost výběru některých předmětů, tzv. seminářů. Jednou z hlavních výzev nejen školních akcí, ale i organizace rozvrhů žáků vyšších ročníků, je proces přihlašování. Jelikož každé přihlašování má své vlastní požadavky, těžko se přihlášky realizují v generických nástrojích, jako jsou on-line formulářové nabídky Microsoftu či Googlu. Na základě mé zkušenosti jako žáka gymnázia, studujícího na škole již osmým rokem, jsem se s různými řešeními problému přihlašování seznámil a pocítil náročnost v používání jak proprietárních intranetů, tak i online formulářů. Po konzultaci s vedením školy jsem také poznal, v čem je náročné přihlašování organizovat a kde se dá tento proces optimalizovat.

Přirozeným řešením tohoto problému je vývoj vlastní přihlašovací aplikace podléhající požadavkům gymnázia, která je navržena pomocí moderních webových standardů, je přístupná a nenarušená sezónním používáním jejích přihlašovacích služeb[1]. Taková služba musí být také dostatečně generická, aby se dokázala adaptovat na změny pravidel přihlašování, ale zároveň také dostatečně přizpůsobitelná, aby nepřestala být lepší alternativou k online formulářům.

Práce je rozdělena na teoretickou a praktickou část. Teoretická část se zaměřuje na obhajobu volby a popis technologií vybraných pro vývoj aplikace, zatímco praktická část, začínající kapitolou 4, dokumentuje doporučené postupy používání aplikace a dovysvětluje implementační detaily jejích funkcí.

Cílem práce je popsat softwarovou architekturu Aplikace a zdůvodnit mnohá rozhodnutí, která byla klíčová pro její zhotovení. Dále by práce měla sloužit jako částečná projektová dokumentace a praktická příručka, která popisuje funkcionalitu aplikace a vztahy datového modelu, na kterém je postavena.

[1] Problém sezónnosti se dá představit podobně jako vytížení e-shopů o Vánocích: aplikace bude mít předvídatelné špičky a měla by se umět naškálovat na požadovanou velikost pro plynulé a nepřerušené používání.

1 Metodologie

Na základě cíle práce byly stanoveny výzkumné otázky, které byly zodpovězeny v popisu vybraných technologií, tvořící hlavní část teoretické části práce (kapitola 2 Využité technologie). Otázky byly formulovány tak, aby nejen popisovaly a argumentovaly výběr technologií, ale i zdůvodňovaly jejich následné použití v projektové implementaci a zajistily z dlouhodobého hlediska Aplikaci stabilní běh. Závěry z těchto otázek byly v kapitole rozpracovány do podkapitol a nejdůležitější aspekty nástrojů popsány ve formě tabulky (viz kapitola 2.8 Shrnutí). Výzkumné otázky zněly následovně:

1. Jak se tento nástroj / přístup liší od podobných, které se nejčastěji v oboru používají?
2. Jaké jsou limitace nástroje?
3. Jaká je škálovatelnost nástroje v kontextu Aplikace?

Pro hledání dokumentací byl použit internetový vyhledávač Google, Bing a databáze otevřeného zdrojového kódu GitHub. Na dokumentačních webech nástrojů byla použita metoda kvalitativní obsahové analýzy, která pomohla podrobit technologie podrobné analýze.

Popsány byly následující technologie:

- PostgreSQL (databáze)
- Prisma a Prisma Studio (podpůrné databázové nástroje)
- Typescript (programovací jazyk)
- tRPC (API protokol)
- zod (nástroj na validaci dat)
- Next.js (frontendový framework)
- Tailwind CSS (frontendová stylovací knihovna)
- shadcn/ui (sada frontendových komponentů)
- Supabase Auth (nástroj pro autentifikaci uživatelů)

Obsahová analýza textu je metoda zaměřená na důkladné porozumění původnímu textu. Jejím cílem je rychle a precizně proniknout k podstatě textu, efektivně jej zredukovat a extrahovat z něj klíčové informace, jako jsou obsah, časová a prostorová perspektiva,

cílová čtenářská skupina a formát dokumentu[2]. Tento proces umožňuje oddělit určité části původního textu, zachovat jejich strukturu a případně je přeformulovat do nové podoby. Výsledkem je slovní vyjádření obsahu dokumentu, které nabývá formy anotace. (Lidmila, 2021)

Vyhledávání relevantních dokumentů probíhalo mezi lednem a březnem roku 2024 a k převážné většině dotazů korespondoval jen jeden výsledek v dokumentačních databázích popsaných technologií.

[2] Pro potřeby této práce byly některé údaje pominuty.

2 Využité technologie

Tato kapitola se věnuje technologiím, které byly použity pro implementaci Aplikace a také zdůvodňuje jejich volbu oproti relevantním alternativám. V každé následující podkapitole je opodstatněn výběr technologií, které byly v dané oblasti vybrány. Kapitola 2.8 Shrnutí poté tato rozhodnutí zpracovává ve formátu tabulky.

2.1 Databáze

Roli centrálního bodu perzistence dat hraje v aplikaci databáze. V našem případě se zde ukládají informace včetně údajů o uživatelích, jako je jméno, e-mailová adresa a třída. Zároveň jsou zde informace o Akcích a všechny vztahové informace mezi žáky a Akcemi (např. jaké možnosti si žák na dané Akci vybral, či kolik žáků již má zájem o daný seminář).

Výběr databáze spočívá v rámci moderních aplikací nejen ve výběru konečného serveru – ty bývají všechny již velmi vyspělé a liší se většinou jen v ceně, dostupnosti a feature setu. Hlavní rozlišení bývá v nástrojích kolem databáze, jako jsou například systémy ORM (Tošovský, 2010).

2.1.1 Databáze a její poskytovatel

Pro realizaci projektu byla zvolena databáze PostgreSQL, která kombinuje přednosti mnohých dokumentových (či NoSQL / nerelačních) a tradičních relačních databázových systémů, jako je MySQL. PostgreSQL vyniká svou podporou pro funkce jako je ukládání dat ve formátu JSON či velmi dobrou nativní podporou pro fulltextové vyhledávání (Ellingwood, 2024) a pyšní se vysokou mírou rozšiřitelnosti s přídavky jako je PostGIS na ukládání zeměpisných souřadnic, či hstore pro jednoduché key-value databázové sloupce (Prakash, 2023).

Pro hosting databáze byla vybrána nabídka od firmy Supabase. Supabase nabízí obohacenou verzi databáze PostgreSQL o funkce jako je Realtime, který umožňuje reagovat na změny dat v reálném čase, a Storage, což je nástroj pro správu oprávnění k souborům založený na Postgresovém modelu oprávnění. Jejich řešení bylo vybráno také proto, že nabízí modul Auth, který se v aplikaci stará o přihlašování uživatelů přes SSO

(více v kapitole 2.6 Autentifikace). Jejich open-source[3] řešení pro databázové systémy poskytuje flexibilní možnosti pro vyzkoušení databáze zdarma, což bylo klíčové pro jednoduchý vývoj aplikace po dobu mnoha měsíců.

2.1.2 Podporující nástroje

Za účelem zjednodušení integrace databáze do celkové struktury aplikace byla zvolena technologie Prisma. Prisma je sada nástrojů pro jednodušší práci s databázemi, zahrnující svůj vlastní ORM (Object-Relational Mapping) systém, který je navržen tak, aby bezproblémově fungoval s aplikačním kódem. Tento systém automaticky generuje typescriptové typy na základě databázových schémat, což přináší výhody typové bezpečnosti a konzistence mezi databází a kódem aplikace.

Díky Prismě může vývojář pracovat s danou databází na vyšší úrovni abstrakce a nemusí se zabývat psaním dlouhých SQL dotazů (které s sebou také nesou riziko útoků jako je SQL injection[4]). ORM systémy také zlepšují obecnou čitelnost a revidovatelnost kódu pomocí syntaktického cukru[5].

Prisma také obsahuje další databázové abstrakce a funkce: v balíčku je webový prohlížeč databáze připomínající PhpMyadmin (Prisma Studio[6]), vlastní formát pro zapisování databázových schémat (Prisma schema[7]) a také adaptéry pro více než 10 různých databází, což je užitečné i pro účely Aplikace.

Tyto adaptéry jsou pro vývoj Aplikace zcela zásadní: sice je PostgreSQL databáze používána jak pro produkci, tak pro vývoj, ale nebylo tomu vždy tak. V prvních iteracích Aplikace byla použita jednoduchá databáze SQLite, která umožňuje databázi zachovat v jednom souboru (např. db.sqlite), který nepotřebuje žádné nastavování, uživatelské účty či server. S rostoucí komplexitou Aplikace bylo nutné databázi změnit na systém PostgreSQL, přičemž byla tato změna změnou jen jednoho řádku v konfiguraci

[3] "open-source" je model otevřeného a veřejně dostupného zdrojového kódu aplikací. V případě Supabase to znamená, že i pokud by firma zkrachovala, databázi si může škola na svých serverech rozběhnout sama, bez porušení jakýchkoliv licenčních podmínek.
[4] Lekce 2 - Technika útoku SQL injection
[5] Syntaktický cukr je metoda usnadnění práce programátorovi zjednodušením a zpříjemněním syntaxe. V tomto případě je to absence nutnosti využívat SQL klíčová slova jako JOIN nebo ALTER.
[6] Dokumentace: Prisma Studio | Prisma Docs
[7] Dokumentace: Prisma schema | Prisma Docs

databázového schématu. Díky Prismě není při vývoji typ databáze zásadní a databáze je k programátorovi transparentní.[8]

2.2 Jazyk

Programovací jazyk pro projekt byl vybrán Typescript. Typescript je v dnešní době webu de facto standardem ve vývoji webových aplikací velkými firmami, jelikož nabízí (i když omezenou) typovou bezpečnost a užitečnou nadstavbu Javascriptové syntaxe pro komplexní logiku dnešních obrovských aplikací (Branscombe, 2022). Protože nebyl Typescript vybrán jen na frontend[9], ale i backend[10] aplikace, dají se typy mezi prostředími sdílet a docílit tak konzistentnějšího procesu vývoje a méně produkčních překvapení.

Typescript nabízí narozdíl od Javascriptu funkce jako jsou pokročilé možnosti statického typování, generické a výčtové typy, podporu pro dekorátory, modifikátory přístupu pro metody tříd a další. Zároveň ale benefituje Typescript z bohatých webových API funkcionalit Javascriptu, které se dají často využít jak v prohlížeči, tak na serveru.

V porovnání s jazyky jako je PHP nebo Python má Typescript výhodu v bohaté komunitě a mnoha komunitních balíčcích v databázi npmjs.com. Další výhodou je také jednodušší hosting na populárních službách jako je Vercel, Netlify nebo Cloudflare Pages, které pro Javascript a Typescript mají velmi jednoduchou integraci serverless funkcí.

2.3 API

API rozhraní je komunikační vrstva mezi dvěma nebo více komponenty. V kontextu projektu je to rozhraní mezi serverem a prohlížečem (koncovým uživatelem), které určuje formát požadavků a jim korespondujícím odpovědím. API protokoly mohou být navrženy velmi mnoha způsoby a jsou často samy se sebou nekonzistentní (např. formát odpovědí, konformita k nativním funkcionalitám prohlížečů apod.). Častým problémem tak bývá například používání HTTP slovesa POST pro všechny požadavky (takto funguje například gRPC) nebo nutnost si o každý prostředek požádat zvlášť (na tom je založena architektura REST a tento problém se snaží řešit architektura GraphQL).

[8] Samozřejmě mají ale různé databáze a různá vlastní omezení, což prakticky znamená, že např. sloupcový typ JSON není pro SQLite podporován a nejde tak jednoduše přemigrovat z PostgreSQL na SQLite, i když z SQLite na PostgreSQL je migrace snadná (Prisma, 2023).
[9] Webové rozhraní viditelné uživateli
[10] Server, na který webové rozhraní posílá požadavky

Moderní API řešení se snaží balancovat DevX (vývojářskou přívětivost) a výkonnost při aplikaci do projektů, které mohou mít API metod stovky či tisíce. Pro projekt byla využita poměrně nová API architektura s názvem tRPC (Typescript Remote Procedure Call), která tento dobrý balanc slibuje a pokouší se řešit dříve zmíněné časté nástrahy.

tRPC využívá vestavěných funkcí prohlížečů například tak, že API metody rozděluje na dva základní typy: dotazy a mutace. Pro dotazy využívá HTTP sloveso GET (což mimochodem umožňuje prohlížeči jednoduše cachovat odpovědi serveru pomocí HTTP hlavičky Cache-Control) a pro mutace, které nějak data na serveru mění (např. přihlašují uživatele na možnost Akce), používá sloveso POST (tRPC, 2024). Podobně jako HTTP slovesa respektuje tRPC i konvence HTTP stavových kódů: mnohé alternativy (jako je GraphQL) chyby zakódují do odpovědi, která z vnějška vypadá jako vydařená (odpovídají stavovým kódem HTTP 200 OK) a uživatel si musí odpověď přečíst a až potom vyhodnotit, jestli jsou mu data k něčemu užitečná. tRPC využívá třídu *TRPCError*, která dokáže využít většinu nejpoužívanějších stavových kódů specifikace HTTP bez převlékání odpovědí (tRPC, 2023), takže například pro neexistující prostředek dostane uživatel v prohlížečové konzoli HTTP 400 Not Found a pro nepřihlášeného uživatele HTTP 403 Forbidden.

tRPC také automaticky spojuje dotazy, které by jinak byly provedeny najednou pomocí tzv. batchingu. Navíc se stará o konstrukci takto složitých API dotazů a dekonstrukci jejich odpovědí tak, aby kód projektu mohl zůstat stejný, nezávisle na tom, jestli se pro požadavek batching využil. Tím řeší problém žádání si o každý prostředek zvlášť poněkud elegantním způsobem, který nevyžaduje detailní vyjmenování všech potřebných dat jako v GraphQL.

Velkou nevýhodou přístupů jako je REST či GraphQL je tzv. configuration drift. Tento problém se projevuje, když například autor kódu změní API definici a přejmenuje metodu či formát vstupních dat. Kompilace serveru i klienta pořád proběhne v pořádku, ale při běhu aplikace bude najednou frontend požadovat metodu, která již neexistuje nebo jí

posílat data, kterým už nerozumí. Toto řeší přístupy jako gRPC či tRPC statickým typováním metod a typů jejich vstupních a výstupních dat[11].

tRPC staví API na základních entitách, kterým říká *routery*, pod kterými se implementují jednotlivé metody (v tRPC *procedures*). Routery se dají analogicky přirovnat například ke službám (*services*) v gRPC. Takto vypadá úryvek z projektové implementace metody `list` (pod routerem event), která uživateli podle jeho třídy a popřípadě preference statusu akce[12] vrátí všechny akce, které se ho týkají:

```
JavaScript
import { createTRPCRouter, publicProcedure } from "~/server/api/trpc";

export const eventRouter = createTRPCRouter({
	list: publicProcedure
		.input(
			z.object({
				active: z.boolean().optional(),
				class: z.enum(CLASSES),
			}),
		)
		.query(({ ctx, input }) => {
			const now = new Date();
			return ctx.db.event.findMany({
				...
```

Pro definici tvaru metod se používá místo klasických Typescript typů schéma vytvořené nástrojem zod, který umožňuje nejen více konzistentní definice při kompilaci kódu (build time), nýbrž také za jeho běhu (runtime). tRPC je aktivně používán v produkčních službách firmami jako je Netflix, Cal.com či Ping.gg (Johansson, 2023).

[11] gRPC problém řeší generováním kódu, což vyžaduje změnu definičních souborů pro jakoukoliv změnu. tRPC využívá stejnojazykosti všech vrstev a typy používá direktně pro implementaci i konzumaci.
[12] Status akce indikuje, zda je akce v budoucnu, zrovna probíhající, či již proběhla.

Knihovna zod je také používána pro extenzivní validaci formulářů v aplikaci, což pomáhá k odstranění klasického problému ve vývoji webu, kterým je nutnost validovat data v prohlížeči a poté ještě jednou na serveru: data se pomocí zodu sice pořád validují dvakrát, ale schéma, proti kterému je validace prováděna, je jen jedno.

2.3.1 Využití systému CRUD

Jako základní stavební kámen architektury tRPC metod v projektu byl vybrán model CRUD (Create, Read, Update, Delete), který popisuje základní operace vytváření, čtení, aktualizace a mazání prostředků, které by měly být konzumentům API umožněny. Na příkladu routeru pro Akce tedy můžeme najít následující metody:

- `list` (read pro více entit najednou)
- `create`
- `get` (read pro jednu entitu)
- `update`
- `delete`

Podobné metody mají i routery uživatele a možností Akcí. CRUD s sebou přináší výhody konzistence (kód je čitelnější a je hned jasné, co daná metoda s prostředkem dělá) a zjednodušení nastavení oprávnění (např. uživatelé Aplikace mohou *číst* Akce, ale všechny ostatní metody jsou jim zakázány).

2.4 Framework

Pro projekt byl vybrán framework Next.js, který vyvíjí firma Vercel. Next.js nejen podstatně zjednodušuje práci s Reactem pomocí vlastního file-based routeru, dobrých výchozích nastavení a jednoduchou integrací pro serverless funkce, ale i zlepšuje výkon a snižuje velikost Javascriptových bundlů, které putují po síti ke koncovým klientům. Next.js totiž umí stránky renderovat nejen přes Reactu nativní CSR[13], ale přináší i jednoduché nastavení pro alternativní metody, jako je SSG[14], SSR[15] a ISR[16], včetně jejich různých kombinací.

[13] Client Side Rendering. Dokumentace
[14] Static Site Generation. Dokumentace
[15] Server-side Rendering. Dokumentace
[16] Incremental Static Regeneration. Dokumentace

I díky těmto výhodám je na Nextu postaven tzv. T3 Stack, který byl vybrán pro bootstrapping projektu. Hlavním komponentem aplikací, založených na stacku T3 je již zmíněný Next.js, komplementovaný tRPC, Prismou a Tailwind CSS. Hlavní změna na tomto stacku proběhla v oblasti přihlašování a managementu uživatelů (viz kapitola 2.6 Autentifikace).

V rámci integrace tRPC do Next.js je využita populární knihovna TanStack Query, která umožňuje optimální zacházení s daty na frontendu. Kromě nativní integrace s React hooks nabízí na požadovaná data pomocné dotazy, jako "načítají se tato data?", nebo "naskytl při načítání dat nějaký problém?". Toto užitečné řešení serverového state managementu je také obohaceno o automatické doplňování tRPC metod a nástroje k jednoduché invalidaci dat či jejich znovuvyžádání (tzv. refetching).

2.5 Styly

V oblasti stylování webových stránek se z tradičních UI knihoven jako Material UI, Chakra UI či Bootstrap přechází na více flexibilní a přizpůsobitelné komponentové knihovny jako je shadcn/ui. Těmito knihovnami chtějí projekty zachovat svůj branding a originalitu a zároveň vyřešit problém často nestandardních nároků pro stavební bloky jejich webových prezentací (Manupa, 2023).

Tyto flexibilní UI knihovny se nejčastěji přizpůsobují pomocí atomických stylovacích knihoven jako je Panda CSS (Chakra UI, 2024), či StyleX (Goel a Gallagher, 2023). Jedním z nejpoužívanějších z těchto nástrojů je ale již pár let Tailwind CSS (Tailwind Labs, 2024), který umožňuje komponenty stylovat v markupu atomickými třídami[17], plně nativní prohlížečům. Tyto třídy Tailwind detekuje a podle těch, které se v projektu používají, vygeneruje CSS soubor. To s sebou přináší řadu výhod, včetně:

- Malých souborů stylů bez nadbytečných definic.
- Pohodlného upravování stylů ve stejných souborech, jako jsou stylované komponenty.

[17] Atomická třída znamená, že namísto tradičního pojetí třídy jako celé stylovací jednotky (např. třída *tlačítko* pro všechny styly tlačítka) se pro stylování takového komponentu používá mnoho různých tříd, každá většinou popisující jen jedno pravidlo kaskádových stylů.

- Jednoduchého plugin systému pro rozšiřování stylování – ať už jde o stylování formulářů, animace, či komplexní typografie, existuje na to přídavek na rozšíření Tailwind syntaxe. (Tailwind Labs, 2023)

Komponentovou knihovnou projektu se stal shadcn/ui, který se přizpůsobuje pomocí Tailwind CSS. Díky tomuto stylování mohou mít komponenty jako tlačítka či odkazy v navigačním menu barvy gymnázia a nemusí se omezovat předpřipravenými barevnými schématy.

2.6 Autentifikace

Přihlašování do aplikace probíhá přes školní Google účty. Implementačně je Google přihlašování prováděno pomocí Supabase Auth, která podporuje přihlašování přes různé poskytovatele, a kromě Google přihlášení podporuje přihlašování například přes e-mail a heslo, Apple ID či Microsoft účet (Supabase, 2024).

Na úrovni databáze je nastavené pravidlo, které umožní přihlášení jen uživatelům s doménou gtmskola.cz. Pokud e-mailová adresa tuto podmínku splňuje, přesměruje Supabase uživatele zpět do aplikace a do URL adresy zakóduje autentifikační token. Tento token prohlížeč uloží do svých cookies[18] a pošle ho s každým požadavkem na server.

2.7 Autorizace

Pro již přihlášené uživatele existuje ještě distinkce jejich práv v aplikaci, čímž jsou pravidla autorizace. Aplikace podporuje dvě základní role, uživatele a správce, které jsou nastaveny každému uživateli zvlášť (každý nový uživatelský účet dostává výchozí roli standardního uživatele). Změnu rolí ovládá kolonka “admin” v tabulce “Students” v PostgreSQL databázi.

Přístup jednoduchého sloupce v databázi oproti řešením jako je Google Zanzibar či SpiceDB byl vybrán z důvodu snahy o snížení komplexity celé Aplikace a o to, aby neměla kromě datového modelu ještě model mapující uživatelská oprávnění k možnostem změn a čtení databázi. Tato řešení jsou pro produkční aplikace velice užitečnými technologiemi,

[18] Cookies jsou jednou z metod ukládání dat v prohlížeči. Pomocí HTTP hlavičky Set-Cookie může server prohlížeči naznačit o jakou hodnotu s jakým názvem se jedná, na jaké stránky ji má posílat (cookies se posílají s každým požadavkem, který splňuje podmínku domény) a na jak dlouho ji má uložit, než ji opět smaže.

ale při zohlednění komplexity Aplikace a jejího rámce byla metoda manuálního ověřování oprávnění praktičtější.

2.8 Shrnutí

Následující tabulka shrnuje data, která byla nejdůležitější při analýze nástrojů popsanou v kapitole 1 Metodologie. Zdroje a vysvětlivky jsou pro lepší přehled v tabulce zmíněny pod čarou.

Kategorie	Vybraný nástroj / přístup	Relevantní alternativy	Výhody oproti alternativám
Databáze	PostgreSQL	MySQL, SQLite, řešení NoSQL	- Podpora pro RLS[19] - Lepší podpora pro JSON pole[20]
API protokol	tRPC	REST, GraphQL	- Užší integrace s Next.js - Statické typování API metod bez nutnosti generování kódu - Nelze použít v aplikacích nenapsaných v jazyce Typescript[21] - API metody musí na požadavky odpovídat ve formátu JSON[22]
Programovací jazyk	Typescript	Javascript, PHP	- Statické typování - Stejný jazyk pro frontend i backend
Stylování	Tailwind CSS s shadcn/ui	Bootstrap, StyleX, Panda CSS, Bootstrap	- Vysoká míra přizpůsobitelnosti - Generování jen definic tříd, které se opravdu používají
Autentifikace	Supabase Auth	Clerk, NextAuth.js	- Jednoduchost užití a integraci s frameworkem

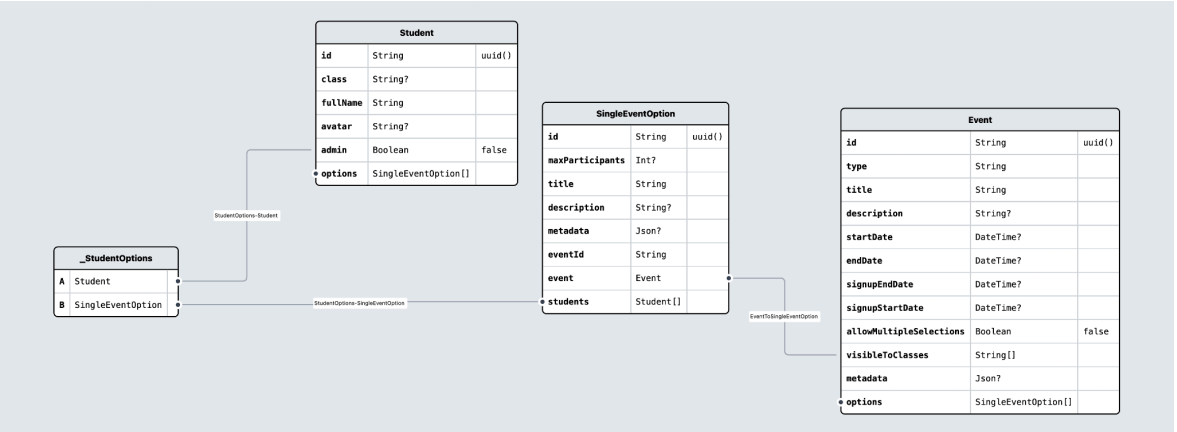
[19] Row Level Security je princip, který umožňuje konfiguraci oprávnění na úrovni databáze: PostgreSQL: Documentation: 16: 5.8. Row Security Policies
[20] (PostgreSQL, 2024)
[21] (Moya, 2023)
[22] (Berry, 2022)

Autorizace	sloupec *admin* v databázi	Google Zanzibar, Authzed SpiceDB	- Redukce komplexity

3 Datový model

Tato kapitola popisuje schéma databáze a to, jak spolu její tabulky souvisí. Pro více informací o technologii a implementační stránce okolo perzistence dat viz kapitola 2.1 Databáze.

Vizualizace databázového schématu za pomocí nástroje Prismaliser



3.1 Databázové entity

Databáze je rozdělena na základní entity, z kterých jsou pomocí nástroje Prisma vygenerovány Postgresové tabulky. Těmito základními entitami jsou:

3.1.1 Student

Tato entita reprezentuje jednoho žáka v databázi. Entita se vytvoří při vytvoření prvního API autorizovaného požadavku na server.

Databázové schéma entity *Student*

Název pole	Typ	Účel	Výchozí hodnota
id	UUID v4 podle RFC 4122	Primární klíč entity. Pomáhá od sebe rozlišit všechny uživatelské účty.	uuid()
class	textový řetězec	Určení třídy uživatele. Databáze pole nevyžaduje, ale aplikace požaduje, aby si žák třídu vybral.	NULL

fullName	textový řetězec	Jméno a příjmení uživatele. Použito v uživatelském nastavení, administrátorském rozhraní a vyexportovaných tabulkách s účastníky. Pole je inicializováno pomocí dat z Google SSO.	NENÍ
avatar	textový řetězec	URL adresa odkazující na profilový obrázek uživatele. Pole může být prázdné a je inicializováno pomocí Google SSO.	NULL
admin	boolean	Indikace zvýšených oprávnění uživatele (role správce).	FALSE
options	Pole odkazů na entitu SingleEventOption	Odkázání na možnosti, které byly uživatelem vybrány. Relace M:N (implicitní, pomocí tabulky _StudentOptions).	

3.1.2 Event

Entita *Event* popisuje v databázi záznam o Akci.

Databázové schéma entity *Event*

Název pole	Typ	Účel	Výchozí hodnota
id	UUID v4 podle RFC 4122	Primární klíč entity. Pomáhá od sebe rozlišit všechny Akce.	uuid()
type	textový řetězec	Kategorizace Akce. Databáze nepřiděluje výchozí hodnotu, ale aplikace dává výchozí hodnotu “other”.	NENÍ
title	textový řetězec	Název Akce. Je určen pro koncové uživatele	NENÍ
description	textový řetězec	Detailní popis Akce. Je určen pro koncové uživatele	NULL
startDate	časové razítko	Určení data a času začátku Akce.	NULL

endDate	časové razítko	Určení data a času konce Akce.	NULL
signupStartDate	časové razítko	Určení data a času začátku přihlašování na Akci.	NENÍ
signupEndDate	časové razítko	Určení data a času konce přihlašování na Akci.	NENÍ
allowMultipleSelections	boolean	Povolení pro uživatele k zvolení více možností jedné Akce.	FALSE
visibleToClasses	pole textových řetězců	Omezení tříd, které se mohou na Akci přihlásit a v Aplikaci ji zobrazit.	[]
metadata	Data ve formátu JSON	Přídavná data potřebná pro funkcionalitu Akcí, které jsou nastavené jako semináře.	NULL
options	Pole odkazů na entitu SingleEventOption	Odkázání na možnosti, které Akci náleží. Relace 1:N.	

3.1.3 SingleEventOption

Entita *SingleEventOption* popisuje jednu možnost jedné Akce.

Databázové schéma entity *SingleEventOption*

Název pole	Typ	Účel	Výchozí hodnota
id	UUID v4 podle RFC 4122	Primární klíč entity. Pomáhá od sebe rozlišit všechny možnosti všech Akcí.	uuid()
maxParticipants	celé číslo	Nastavení maximálního počtu uživatelů, kteří se mohou na možnost přihlásit.	NULL
title	textový řetězec	Název možnosti Akce. Je určen pro koncové uživatele	NENÍ
description	textový řetězec	Detailní popis / anotace možnosti Akce. Je určen pro koncové uživatele	NULL

metadata	Data ve formátu JSON	Přídavná data potřebná pro funkcionalitu Akcí, které jsou nastavené jako semináře.	NULL
eventId	UUID v4 podle RFC 4122	Identifikátor Akce, ke které možnost náleží.	NENÍ
event	Odkaz na entitu Event	Odkázání na entitu Akce, korespondující k hodnotě sloupce eventId.	
students	Pole odkazů na entitu Student	Určení uživatelů, kteří se na danou možnost přihlásili. Relace M:N (implicitní, pomocí tabulky _StudentOptions).	

3.1.4 _StudentOptions

Pomocná tabulka *_StudentOptions* pomáhá vytvořit vztah M:N (*many to many*) mezi možnostmi a uživateli. Tabulka je vygenerována pomocí implicitního many-to-many vztahu[23].

3.2 Interakce Aplikace s databází

Tato podkapitola popisuje operace, které probíhají na úrovni databáze v různých stavech Aplikace, a které mají vliv na jiné entity než ty, které operaci přímo náleží.

3.2.1 Smazání Akce

Po smazání Akce přes webové rozhraní je před smazáním samotného databázového řádku smazána každá možnost, která Akci náleží (také je přes tabulku *_StudentOptions* smazáno spojení každé možnosti Akce s každým přihlášeným uživatelem).

3.2.2 Smazání možnosti Akce

Po smazání jedné možnosti Akce je před smazáním dat v databázi vymazán vztah mezi uživatelem a možností v tabulce *_StudentOptions* a možnost je disociována od záznamu v tabulce Event.

[23] Tyto implicitní vztahy jsou zapisovány do databázového schématu nástroje Prisma. Dokumentace implicitních many-to-many vztahů.

3.2.3 Smazání uživatele

Před smazáním uživatele v Aplikaci je vynucena disociace od všech přihlášených možností všech Akcí (odstranění hodnot *_StudentOptions*, kde se studentId rovná identifikátoru uživatele, kterého mažeme).

3.2.4 Změna třídy uživatele

Z archivačních důvodů změna třídy u uživatele **neodstraní** uživatele z možností, na které se již kvůli změně nemůže přihlásit.

4 Příručka

Tato kapitola slouží jako stručná a praktická příručka k používání aplikace. Je rozdělena do podkapitol a každá z nich popisuje user flow, kterým uživatel prochází pro dosažení cíle naznačeného v jejím nadpisu. Příručka je rozdělena na dvě hlavní pasáže: příručku pro správce a příručku pro uživatele / žáky.

4.1 Pro správce

4.1.1 Vytváření a úprava Akcí

Pro vytváření a upravování Akcí je důležité být do aplikace přihlášený. Je také důležité obdržet roli správce, což je možné po kontaktování it@gtmskola.cz[24]. Po změně a navýšení role uživatele se v horním menu objeví položka "Správa Akcí".

Na této stránce je tabulka se všemi Akcemi, které se v databázi vyskytují. Obsahuje nejen nadcházející, ale i všechny historické Akce pro archivní účely. V seznamu se dá hledat pomocí textového pole "Vyhledat", které dokáže hledat ve sloupcích "Název", "Typ", dat začátku a konce přihlašování a v seznamu zúčastněných tříd[25].

Úprava Akce je možná na její podstránce, která je dostupná pod tlačítkem "Zobrazit". Data, která se dají na stránce upravit, jsou:

- **Název**: krátký popis Akce, který se zobrazí pro žáky jak na hlavní stránce, tak na podstránce jejího detailu. Je doporučeno mít názvy deskriptivní a věcné, aby bylo na první pohled jasné, o jaké přihlašování se jedná. Příklad doporučeného formátu názvu volby seminářů je "Semináře (8G) 2024/2025", jelikož je nejen jasné, o jakou třídu se jedná, ale i to, v jakém školním roce budou semináře z přihlašování relevantní. Inkluze školního roku je potom užitečná hlavně pro účely hledání v přihláškách z minulých let.

[24] Pro instrukce pro správce o tom, jak přidat správcovská oprávnění dalším uživatelům konzultujte kapitolu 4.1.4 Povýšení uživatele na správce a oprava výběru třídy.
[25] Třídy se identifikují jejich dvouznakovou zkratkou (např. 1G, 1N).

- **Popis**: detailnější anotace Akce, která je ve svém celku viditelná na její podstránce. Toto pole je užitečné pro obecné informace k přihlašování, jako kde a s kým se Akce koná, kontakt na zodpovědnou osobu či často kladené dotazy.
- **Více možností na uživatele**: toto políčko zaškrtněte, pokud chcete, aby si jeden žák mohl vybrat více možností této Akce najednou. Například pro výběr projektového dne je užitečné nechat políčko nezaškrtnuté, jelikož se jeden žák může zúčastnit jen jednoho projektu. Narozdíl od toho je volba seminářů ideální pro umožnění volby několika možností jedním žákem.
- **Typ Akce**: toto pole pomáhá systému v adaptaci na dané přihlašování. Například typ "Semináře" rozšiřuje možnosti nastavení o zadání počtu týdenní hodinové dotace, učitele a větve[26].
- **Přihlášky**: určení doby, ve které bude možné pro žáky se na Akci přihlásit. Přihlašování musí být v rozmezí alespoň jednoho dne a platí vždy od půlnoci dne začátku přihlášek do 23:59 dne konce přihlášek[27]. Mimo rozmezí těchto dat nedovolí uživatelům systém se přihlásit na kteroukoliv jimi vybranou možnost a zobrazí jim možnosti, které si vybrali. Pro vybrání časového okna trvající jen jeden den je možné na datum v kalendáři poklepat dvakrát.
- **Konání**: rozmezí dat, mezi kterými se Akce koná. Tento údaj se ukazuje na stránce detailu Akce a také umožňuje systému ukazovat Akce na domovské stránce uživatele i přes to, že přihlašování již skončilo pro účely zkontrolování jejich volby, jakou jsou jejich semináře na příští rok.
- **Hodin týdně**: vztahuje se jen na semináře. Určuje počet hodin, kterého musí dohromady žáci dosáhnout kombinací všech jejich seminářů.
- **Omezení tříd**: Akce mohou omezit to, jaké třídy je vidí a mohou zobrazit. I zde se identifikují třídy podle jejich dvouznakového názvu (1G, 6G, 2N). Pokud pole zůstane prázdné, Akce bude dostupná k zobrazení a přihlašování od všech uživatelů bez omezení.

[26] Kategorie, do které daný seminář zapadá.

[27] Tato data se vztahují k časovému pásmu počítače, na kterém se Akce vytváří. To znamená, že kdyby vytvářel Akci uživatel v Anglii, přihlašování by v Česku začalo již ve 23:00 předchozího dne a skončilo ve 22:59 v posledním dni přihlašování

Nové Akce se dají vytvářet za pomocí tlačítka “Vytvořit Akci” na stránce “Správa Akcí”. Typ dat, které do formuláře uživatel vyplní se shodují s těmi zobrazenými na podstránce Akce, které jsou popsané výše. Odstranění Akce je možné na spodní části detailu Akce tlačítkem “Smazat Akci” a následným potvrzením ve vyskakovacím okně.[28]

4.1.2 Vytváření a úprava možností Akce

Každé Akci náleží libovolný počet možností, na které se žáci mohou přihlásit. Možnosti se vytvářejí až po vytvoření odpovídající Akce na její podstránce. Podobně jako samostatné Akce se možnosti u vytváření a upravování shodují: vytváření je dostupné pomocí tlačítka “Přidat možnost” nad tabulkou s možnostmi a upravování možnosti je možné tlačítkem “Upravit” v posledním sloupci tabulky. Data, která se dají upravit u jedné možnosti, jsou:

- **Název**: krátký titulek možnosti, který se bude zobrazovat na výčtu možností u jedné Akce.
- **Popis**: krátký popis možnosti, který se bude zobrazovat na výčtu možností u jedné Akce. Dá se použít například pro krátkou anotaci a bližší informace jako například učitele zodpovědného za daný projekt projektového týdne.
- **Maximální počet účastníků**: maximální kapacita možnosti, která určuje počet přihlášených, po jehož dosažení zakáže systém se přihlašovat všem dalším. Toto pole je užitečné například pro projekty na projektových týdnech nebo výlety.
- **Hodin týdně**: vztahuje se jen na semináře. Určuje počet hodin, který zabírá daný předmět v rozvrhu žáka. Pro semináře, které se týden od týdne liší (např. jeden dvouhodinový blok za 14 dnů) se uvádí údaj průměru přes toto období (v našem příkladě tedy 1 hodina týdně).[29]
- **Větev**: vztahuje se jen na semináře. Rozhoduje o zaměření semináře, přičemž ve výchozím nastavení je na výběr z větve přírodovědné, humanitní a univerzální. Pravidla zaměření jsou:
    - Na předměty v univerzální větvi se může uživatel zapsat kdykoliv.

[28] Odstranění Akce neodstraní jen Akci jako takovou, ale i všechny její možnosti a registrační data. To znamená, že mazání Akcí je nenávratné a mělo by se provádět jen v krajních případech.
[29] Systém nepodporuje pro počty hodin necelé číselné údaje, což znamená, že předmět, který má 1 hodinu za čtrnáct dní není validní seminář.

- Na předměty ve větvi přírodovědné a humanitní se může uživatel zapsat jen v případě, že se již nezapsal na kterýkoliv předmět z větve druhé. Tyto větve se vzájemně vylučují a není možné jejich předměty kombinovat.
- Každý předmět může mít právě jedno zaměření.

Implementační poznámka k větvím seminárních předmětů: v aktuální verzi aplikace se dají na předměty seminářů aplikovat jen tři předem zmíněné větve – univerzální, humanitní a přírodovědná. Funkcionalita vzájemného vylučování (mutual exclusivity) je indikována typem větve: pokud je její typ “oneof”, může si uživatel vybrat jen jednu větev ze všech větví, které mají také typ “oneof”. Ostatní větve mají typ “unbound” a jejich možnosti jsou volitelné vždy, nezávisle na ostatních větvích. Takto vypadá výchozí nastavení větví:

```JavaScript
const AVAILABLE_BRANCHES = [
        { id: "universal", label: "Univerzální", type: "unbound" },
        { id: "humanitarian", label: "Humanitní", type: "oneof" },
        { id: "science", label: "Přírodovědná", type: "oneof" },
]
```

4.1.3 Exportování dat Akce do souboru .xlsx

Aplikace umožňuje správcům z jednotlivých Akcí vygenerovat tabulkový soubor kompatibilní s aplikacemi jako je Microsoft Excel, Tabulky Google, iWork Numbers či LibreOffice Calc.

Vyexportovaný soubor obsahuje právě tolik listů, jako je dostupných možností u zvolené Akce. Každý list je pojmenován podle názvu možnosti a obsahuje dva sloupce: “Jméno” a “Třída”. Řádky tabulky korespondují ke všem uživatelům, kteří se na danou akci přihlásili.

Vygenerování tabulky a jeho následné stažení je možné na podstránce Akce ve správcovském rozhraní, stisknutím tlačítka “Stáhnout jako .xlsx”.

Poznámka: informace o třídě jsou do souboru vloženy tak, jak platí v moment vygenerování. To znamená, že při vygenerování tabulky v příštím školním roce bude třída žáka z tercie kvarta, i když přihlášení proběhlo v tercii.

4.1.4 Povýšení uživatele na správce a oprava výběru třídy

Pro správce je v hlavní nabídce dostupná položka “Správa Uživatelů”. Na této stránce najdete seznam všech uživatelů, kteří se kdy přihlásili do aplikace, spolu s jejich třídou a statusem správcovství.

U každého uživatele je tlačítko “Upravit”, po jehož stisknutí je možné pro uživatele vybrat jakoukoliv třídu, nebo mu změnit roli (povýšení na či degradování ze) správce.

Poznámka: správce nemůže sám sebe degradovat ze správce na uživatele.

4.1.5 Odstraňování uživatelů

V krajních případech je možné databázovou reprezentaci uživatelských účtů odstranit. Je důležité zmínit, že vymazání nesmaže korespondující Google účet a nezakáže uživatelům si založit účet znovu – odstranění vymaže všechny informace o účtu (jméno, třída, status správcovství) a konexe k Akcím (všechny přihlášky na možnosti Akcí).

Vymazání účtu je možné přes vyskakovací okno k úpravě uživatele na stránce “Správa Uživatelů”.

Poznámka: správce může odstranit všechny uživatele kromě sebe, takže pro smazání všech uživatelů v systému je nutné využít přímý přístup k databázi.

4.1.6 Diverzifikace oprávnění uživatelů a správců

Uživatelé mají následující oprávnění:

- Účet
    - Přihlásit se
    - Změnit svou třídu
- Akce

        - Zobrazit Akce své třídy
        - Zobrazit možnosti jakékoliv Akce
        - Přihlásit se na možnost Akce / odhlásit se z Akce, u které právě probíhá přihlašování, a která je relevantní pro zvolenou třídu.

Správci mají ze základu všechna oprávnění jako uživatelé. Navíc mají ještě tato:

- Účet
    - Změnit třídu jiných uživatelů
    - Změnit roli jiných uživatelů (přidání správcovských oprávnění)
- Akce
    - Vytvořit novou Akci
    - Upravit Akci (viz kapitola 4.1.1 Vytváření a úprava Akcí)
    - Odstranit Akci
    - Vytvořit novou možnost Akce
    - Upravit možnosti Akce (viz kapitola 4.1.2 Vytváření a úprava možností Akce)
    - Odstranit možnosti Akce
    - Vygenerovat soubory .xlsx z Akcí (viz kapitola 4.1.3 Exportování dat Akce do souboru .xlsx)

4.2 Pro uživatele

4.2.1 Přihlašování se do Aplikace

Žáci jsou hlavním publikem přihlašovacího portálu. Ze začátku jsou uvítáni přihlašovací stránkou, která je přesměruje na přihlášení přes jejich školní Google účet. Po úspěšném přihlášení musí uživatel zadat jeho třídu.

V případě přihlášení pod Google účtem nepatřícím pod doménu gtmskola.cz bude přihlášení zamítnuto a uživatel instruován k novému přihlášení s účtem správným.

4.2.2 Přihlašování se na Akce

Žáci se mohou přihlašovat pouze na Akce, které jsou označeny jako *aktivní*. To znamená, že jejich přihlašovací okno je otevřené, a že se akce koná v budoucnosti. Jako příklad se

dají použít projektové dny, na které se standardně přihlašuje 3 týdny před jejich konáním a přihlašování na ně trvá jeden týden. Pro Akce, které jsou sice v budoucnu, ale jejich přihlašování se již uskutečnilo, nabízí Aplikace žákům přehled jimi vybraných možností, které už ale nemohou měnit. Podobně je to u Akcí, které ještě nemají přihlašování otevřené a žáci se mohou podívat jen na možnosti budoucího výběru.

Dalším omezením dostupnosti možnosti přihlášení na Akci je omezení tříd, které se ke každé Akci dá nakonfigurovat. Pokud je tedy Akce omezena na třídu A, žák ze třídy B ji neuvidí v seznamu nadcházejících Akcí. V případě, že se žák B na stránku Akce dostane přes přímý odkaz, bude mít přístup k zobrazení detailů (název, anotace, výpis možností apod.), ale tlačítka pro přihlášení zůstanou skrytá.

Přihlašování probíhá přes podstránku Akce, na kterou je odkázáno ve výpisu Akcí (úvodní stránka, na kterou je uživatel po přihlášení přesměrován). Při vybrání aktivní Akce ze seznamu získá uživatel možnost se přihlásit na libovolnou možnost Akce (v případě některých Akcí kombinaci určitých možností), s výjimkou těch, které mají vyčerpanou kapacitu (na možnosti je nastaven maximální počet účastníků a tato kvóta je naplněna).

4.2.2.1 Přihlašování se na semináře

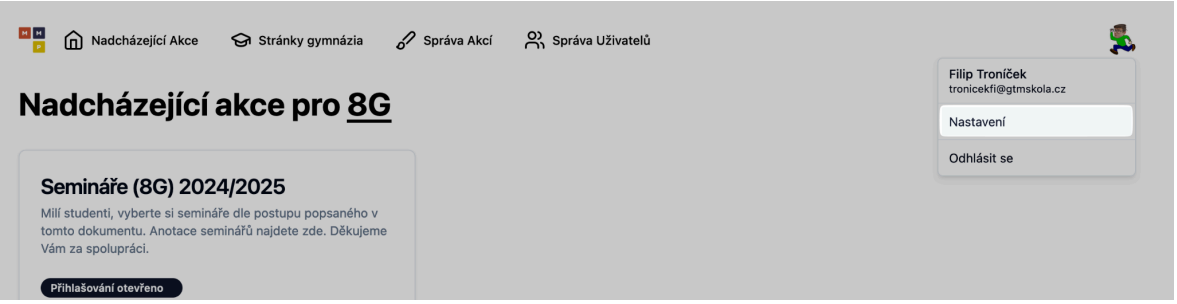
Přihlašování se na semináře probíhá podobně jako na ostatní Akce, s tím, že existují malé změny v uživatelském rozhraní a omezení výběru možností.

Na přihlašovací stránce na semináře je pod nadpisem dynamický indikátor, který zobrazuje, kolik týdenních hodin seminářů zbývá danému uživateli k vybrání. Při vybrání všech požadovaných hodin indikátor zezelená a uživatel je na semináře oficiálně zapsán (přičemž své volby může změnit až do konce termínu přihlášek).

Na některých volbách seminářů (zejména těch pro vyšší ročníky) jsou také omezeny kategorie (větve) seminářů, na které se uživatel může zapsat. Například pokud se uživatel zapíše na seminář fyziky, který je kategorizován do větve přírodovědné, nemůže se již přihlásit na seminář literatury (na humanitní větev po zvolení předmětu z větve přírodovědné již nepůjde kliknout). Ve většině případů je ale také volitelná větev univerzální, která obsahuje předměty volitelné nezávisle na všech ostatních.

4.2.3 Změna třídy uživatele

Třída uživatele určuje Akce, které může uživatel zobrazit a na které se může přihlásit. Každý uživatel si může svoji třídu změnit v uživatelském nastavení, které je dostupné jako odkaz po kliknutí profilového obrázku v pravém horním rohu Aplikace.



Volbu může uživatel vytvořit ze seznamu všech dostupných tříd a poté své změny potvrdí tlačítkem “Uložit”.

Závěr

Tato práce se zabývala popisem, implementací a dokumentací webové aplikace pro přihlašování žáků Gymnázia Thomase Manna na školní akce a semináře. Prostřednictvím teoretické části vysvětlila softwarovou architekturu projektu a odůvodnila výběr jejích nejdůležitějších komponent. V praktické části poté ve formátu příručky popsala, jak se s Aplikací zachází a zmínila důležité netriviální implementační detaily.

Maturitní práce splnila všechny své stanovené cíle, kterými bylo popsání softwarové architektury vyvinuté aplikace, obhájení a analýza využitých technologií včetně dokumentace průzkumu relevantních alternativ a vytvoření praktické dokumentace ve formě příručky.

U naplnění cíle je třeba zohlednit problematiku chybějící či nedostatečné dokumentace mnoha softwarových nástrojů, které byly v práci popisovány: velkému počtu technologií chybí stránky dokumentace o technologii obecně a je tak nutné dostávat obecné či úvodní informace o nich od třetích stran. Oficiální dokumentace a návody také často nezmiňují porovnání s podobnými nástroji, což dělá proces objektivního zhodnocení podstatně složitější a nutí čtenáře hledat informace u méně důvěryhodných zdrojů.

Limitem výsledků teoretické části práce byla poměrně malá velikost vzorku vybraných alternativních technologií v porovnání s často i stovkami různých možností a přístupů ke každému problému, které jsou na internetu k dispozici.

Výsledky praktické části limitovala skutečnost, že Aplikace v době dokončení práce (březen 2024) nebyla prakticky vyzkoušena žáky školy a příručka tak nemusí být dostatečně dopodrobna a srozumitelně napsána a odrážet opravdové problémy žáků a vedení školy s projektem.

Seznam použitých informačních zdrojů

BERRY, James, 2022. Support for additional `Content-Type`s · Issue #1937 · trpc/trpc. GitHub [online] [vid. 2024-03-10]. Dostupné z: https://github.com/trpc/trpc/issues/1937

BRANSCOMBE, Mary, 2022. How TypeScript Helps Enterprise Developers. The New Stack [online]. [vid. 2024-02-18]. Dostupné z: https://thenewstack.io/how-typescript-helps-enterprise-developers/

ELLINGWOOD, Justin, 2024. PostgreSQL Advantages: Benefits of Using PostgreSQL. Prisma's Data Guide [online] [vid. 2024-03-31]. Dostupné z: https://www.prisma.io/dataguide/datamodeling/intro-dont-panic

GOEL, Naman a Nicolas GALLAGHER, 2023. Introducing StyleX. StyleX [online] [vid. 2024-02-25]. Dostupné z: https://stylexjs.com/blog/introducing-stylex/

CHAKRA UI, 2024. Panda CSS [online]. TypeScript. 25. únor 2024. B.m.: Chakra UI. [vid. 2024-02-25]. Dostupné z: https://github.com/chakra-ui/panda

JOHANSSON, Alex, 2023. 👋 Who's using tRPC? · Issue #1290 · trpc/trpc. GitHub [online] [vid. 2024-03-17]. Dostupné z: https://github.com/trpc/trpc/issues/1290

LIDMILA, Jan, 2021. Formální a obsahová analýza textu, rychlé čtení [online]. 2021. B.m.: Moravská zemská knihovna. [vid. 2024-03-10]. Dostupné z: https://www.mzk.cz/sites/mzk.cz/files/souboryMZK/formalni_a_obsahova_analyza_textu.pdf

MANUPA, 2023. The anatomy of shadcn/ui [online] [vid. 2024-03-02]. Dostupné z: https://manupa.dev/blog/anatomy-of-shadcn-ui

MOYA, Jose, 2023. tRPC: TypeScript Remote Procedure Calls - Capicua [online] [vid. 2024-03-10]. Dostupné z: https://www.wearecapicua.com/blog/all-about-trpc

POSTGRESQL, 2024. JSON Functions and Operators. PostgreSQL Documentation [online] [vid. 2024-03-17]. Dostupné z: https://www.postgresql.org/docs/16/functions-json.html

PRAKASH, Aditi, 2023. Top 10 PostgreSQL Extensions You Should Know About | Airbyte [online] [vid. 2024-03-31]. Dostupné z: https://airbyte.com/data-engineering-resources/postgresql-extensions

PRISMA, 2023. SQLite database connector. Prisma [online] [vid. 2024-02-18]. Dostupné z: https://www.prisma.io/docs/orm/overview/databases/sqlite

SUPABASE, 2024. Auth | Supabase Docs [online] [vid. 2024-03-02]. Dostupné z: https://supabase.com/docs/guides/auth

TAILWIND LABS, 2023. Plugins - Tailwind CSS [online] [vid. 2024-02-25]. Dostupné z: https://tailwindcss.com/docs/plugins

TAILWIND LABS, 2024. Tailwind CSS [online]. HTML. 25. únor 2024. B.m.: Tailwind Labs. [vid. 2024-02-25]. Dostupné z: https://github.com/tailwindlabs/tailwindcss

TOŠOVSKÝ, Martin, 2010. Objektově relační mapování [online] [online]. 2010. B.m.: Univerzita Hradec Králové, UHK, Fakulta informatiky a managementu Hradec Králové. Dostupné z: https://theses.cz/id/rur7qo/

TRPC, 2023. Error Handling | tRPC [online] [vid. 2024-03-31]. Dostupné z: https://trpc.io/docs/server/error-handling

TRPC, 2024. Quickstart | tRPC [online] [vid. 2024-03-31]. Dostupné z: https://trpc.io/docs/quickstart

Seznam příloh

Příloha 1 – Použité zkratky

API — Aplikační programovací rozhraní (Application Programming Interface)
CRUD — Create Read Update Delete
CSS — Kaskádové styly (Cascading Style Sheets)
Google SSO — Jednotné přihlášení přes Google (Google Single Sign-On)
HTTP — HyperText Transfer Protocol
JSON — JavaScriptový objektový zápis (JavaScript Object Notation)
ORM — Objektově relační mapování (Object Relational Mapping)
REST — REpresentational State Transfer
UUID — Univerzální unikátní identifikátor (Universally Unique IDentifier)

Poznámka: obrázkové přílohy byly vloženy přímo do textu, aby byl zřetelný kontext, který je obklopuje.