Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Webová aplikace V tento den

2024

Filip Vepřík

Vedoucí práce:
Mgr. Jiří Zacpal, Ph.D.

Studijní program: Informatika,
Specializace: Programování a vývoj software

Bibliografické údaje

Autor:	Filip Vepřík
Název práce:	Webová aplikace V tento den
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Informatika, Specializace: Programování a vývoj software
Vedoucí práce:	Mgr. Jiří Zacpal, Ph.D.
Počet stran:	30
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Filip Vepřík
Title:	Web application On this day
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Computer Science, Specialization: Programming and Software Development
Supervisor:	Mgr. Jiří Zacpal, Ph.D.
Page count:	30
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Práce pojednává o vývoji webové aplikace V tento den. Tato aplikace slouží pro správu událostí, které se staly daný den v historii. Aplikace je implementována pomocí full-stack React frameworku Next.js.

Synopsis

This thesis is about the development of the web application On this day. This application is used to manage events that happened on a given day in history. The application is implemented using the full-stack React framework Next.js.

Klíčová slova: V tento den; webová aplikace; Next.js; Google Kalendář

Keywords: On this day; web application; Next.js; Google Calendar

Rád bych poděkoval panu Mgr. Jiřímu Zacpalovi, Ph.D. za jeho ochotu a cenné připomínky při vedení této práce. Dále bych rád poděkoval mé přítelkyni a rodině za jejich podporu.

Odevzdáním tohoto textu místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

1 Úvod

V dnešní době je přístup k historickým i budoucím událostem a informacím velmi důležitý, ať už pro osobní zájem, vzdělávání nebo profesionální potřeby. S rostoucím množstvím dostupných dat se stává stále náročnějším udržet přehled o událostech, které by nás mohli zajímat, ať už se jedná o důležité historické milníky, aktuální události nebo plánování budoucích akcí.

Webová aplikace V tento den je určena pro zjednodušení správy těchto událostí a jejich sdílení mezi uživateli. Je zaměřena na jednoduchost, přístupnost a přehlednost.

V práci se nejdříve zaměříme na specifikaci řešení, tedy jak jsou specifikovány různé části aplikace a jaké jsou jejich možnosti. Následně si ukážeme existující řešení, řekneme si jejich klady a zápory a přínosy pro vlastní řešení. Potom si uvedeme, v jakém jazyce a jaké technologie jsou pro řešení práce použity. Dále se budeme zabývat programátorskou příručkou, ta obsahuje strukturu databáze, nejdůležitější adresáře, funkcionalitu a možné rozšíření aplikace. Následuje uživatelská příručka, ta slouží jako návod pro uživatele. Nachází se v ní, jak aplikace vypadá, způsoby přihlášení a registrace uživatelů, jak se vytváří kategorie s událostmi a jak synchronizovat vybrané události s Google Kalendářem. Závěrem shrneme cíl práce a zhodnotíme její výsledek.

2 Specifikace řešení

2.1 Uživatelé

Uživatelé mají možnost přihlašování a registrování. Prvním způsobem registrace je pomocí uživatelského jména, hesla a emailu. Na zadaný email přijde potvrzovací odkaz, případně si díky němu můžeme změnit heslo. Další varianta přihlášení je pomocí Google účtu.

2.2 Události

U každé události evidujeme datum, popisek, dodatečný popisek a jaký uživatel ji vytvořil. Popisek je text, který uživatelům říká, co se v daném dni stalo. Dodatečné informace jsou nepovinné a zobrazí se po kliknutí na událost.

Události mohou přidávat, upravovat a mazat pouze uživatelé, kteří mají dostatečně velká oprávnění v kategorii. Každou událost bude moci uživatel vyhledat ať už pomocí data, textu nebo kategorie.

2.3 Kategorie

Všechny události jsou součástí nějaké kategorie. Každá je barevně odlišena, aby byla lépe rozpoznatelná. Dále je složena z názvu a popisu, který nám naznačuje, jaký druh událostí se v ní nachází.

Kategorie jsou rozděleny na dva druhy, na obecnou a uživatelskou. Obecnou událost vidí všichni uživatelé, ať už registrovaní nebo ne. Uživatelská je viditelná pouze pro toho uživatele, který ji vytvořil, případně má možnost kategorii sdílet se všemi nebo pouze s vybranými uživateli. Obecnou kategorii mohou vytvořit pouze administrátoři.

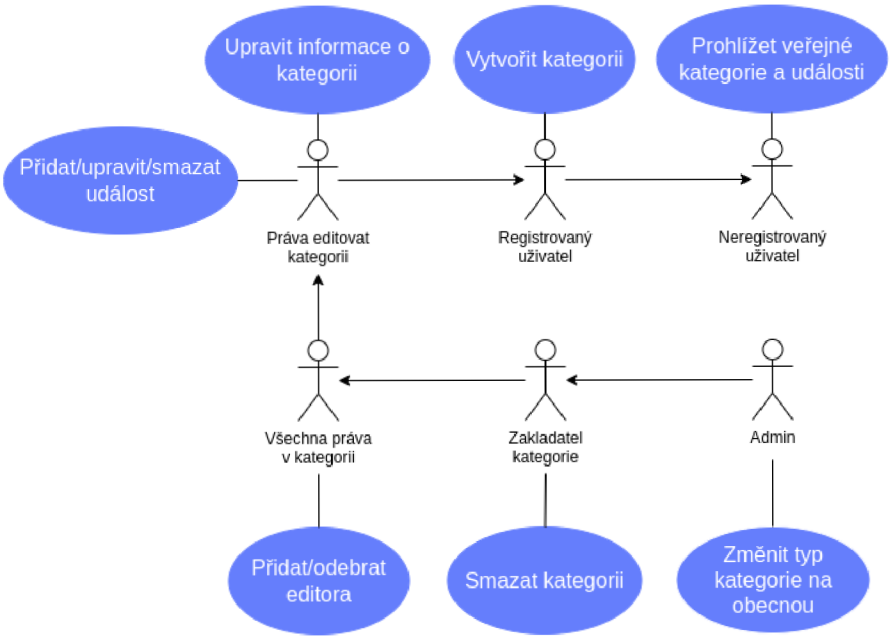
2.4 Import a export

Každý ověřený uživatel si může u odebírané kategorie exportovat všechny její události. Import uživatel může provést pouze do vlastní kategorie. Import a export probíhá ve formátu iCal, což je standard pro výměnu kalendářových dat.

2.5 Synchronizace s Google Kalendářem

Každý uživatel, který se zaregistroval pomocí Google účtu, si může vybrat, které události si chce synchronizovat se svým kalendářem. Uživatel má možnost zadat jméno kalendáře, do kterého se události vloží. Pokud takový kalendář neexistuje, tak se automaticky vytvoří.

2.6 Diagram případů užití



Obrázek 1: Diagram případů užití

3 Průzkum existujících řešení

V této části se budeme zabývat existujícími řešeními a popíšeme jejich klady a zápory. Následně si shrneme přínosy z existujících řešení pro tuto práci.

3.1 On This Day

On This Day[1] je přehledná webová aplikace, ve které lze vyhledávat velké množství událostí. Vyhledávání lze filtrovat na základě textu, data a typu události. Vyskytuje se zde i sekce s články, ve kterých se uživatelé mohou dozvědět detailnější informace o různých událostech. Kromě článků můžeme na stránce ještě najít sekci s kvízy.

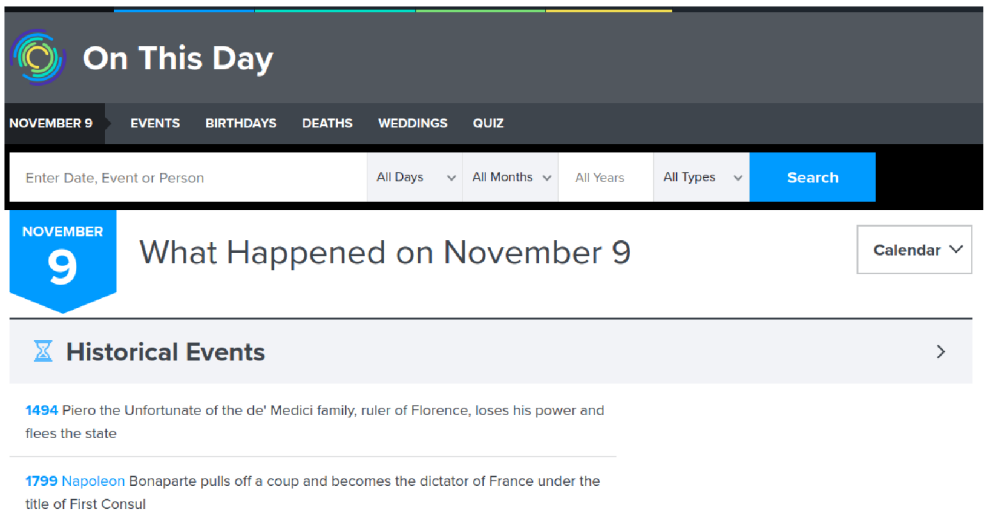
Uživatelé nemají možnost registrace, nemohou tedy přidávat vlastní události, není ani možnost synchronizování s kalendářem.

Klady

- Přehledné a jednoduché na ovládání
- Široké možnosti vyhledávání událostí
- Velké množství událostí

Zápory

- Nelze přidávat vlastní události
- Nelze synchronizovat s Google Kalendářem



Obrázek 2: On This day

3.2 Wikipedie

Na Wikipedii se vyskytuje článek „Kalendárium historických výročí“[2], kde nalezneme kalendář, na kterém má každý den svoji stránku. Záznamy jsou uspořádány do třech typů. Těmi typy jsou události, narození a úmrtí. Dále je každý typ rozdělen na české a světové. Jelikož se jedná o Wikipedii, tak se dá proklikávat do podrobnějších detailů.

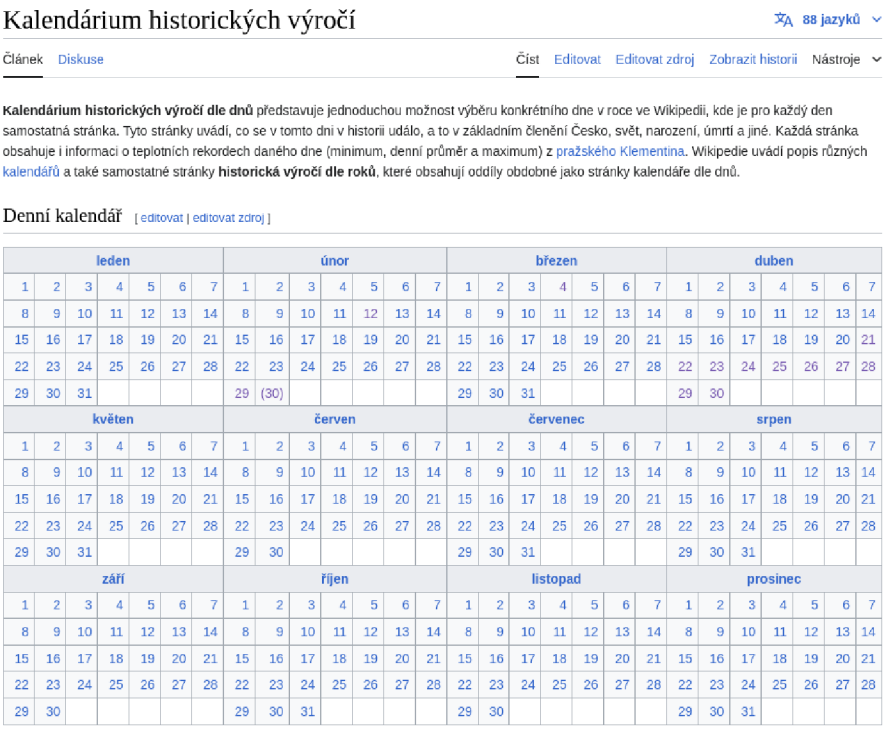
Všechny záznamy může editovat jakýkoliv uživatel, což může mít své nevýhody. V minulosti jsem se již setkal s pár záznamy, kde bylo špatně datum, proto je důležité si kontrolovat, jestli jsou údaje správné.

Klady

- Velké množství záznamů
- Proklikávání pro podrobnější informace

Zápory

- Údaje nejsou vždy pravdivé
- Nelze synchronizovat s Google Kalendářem


Kalendárium historických výročí

88 jazyků

Článek Diskuse Číst Editovat Editovat zdroj Zobrazit historii Nástroje

Kalendárium historických výročí dle dnů představuje jednoduchou možnost výběru konkrétního dne v roce ve Wikipedii, kde je pro každý den samostatná stránka. Tyto stránky uvádí, co se v tomto dni v historii událo, a to v základním členění Česko, svět, narození, úmrtí a jiné. Každá stránka obsahuje i informaci o teplotních rekordech daného dne (minimum, denní průměr a maximum) z pražského Klementina. Wikipedie uvádí popis různých kalendářů a také samostatné stránky **historická výročí dle roků**, které obsahují oddíly obdobné jako stránky kalendáře dle dnů.

Denní kalendář [editovat | editovat zdroj]

leden							únor							březen							duben						
1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
8	9	10	11	12	13	14	8	9	10	11	12	13	14	8	9	10	11	12	13	14	8	9	10	11	12	13	14
15	16	17	18	19	20	21	15	16	17	18	19	20	21	15	16	17	18	19	20	21	15	16	17	18	19	20	21
22	23	24	25	26	27	28	22	23	24	25	26	27	28	22	23	24	25	26	27	28	22	23	24	25	26	27	28
29	30	31					29	(30)						29	30	31					29	30					

květen							červen							červenec							srpen						
1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
8	9	10	11	12	13	14	8	9	10	11	12	13	14	8	9	10	11	12	13	14	8	9	10	11	12	13	14
15	16	17	18	19	20	21	15	16	17	18	19	20	21	15	16	17	18	19	20	21	15	16	17	18	19	20	21
22	23	24	25	26	27	28	22	23	24	25	26	27	28	22	23	24	25	26	27	28	22	23	24	25	26	27	28
29	30	31					29	30						29	30	31					29	30	31				

září							říjen							listopad							prosinec						
1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
8	9	10	11	12	13	14	8	9	10	11	12	13	14	8	9	10	11	12	13	14	8	9	10	11	12	13	14
15	16	17	18	19	20	21	15	16	17	18	19	20	21	15	16	17	18	19	20	21	15	16	17	18	19	20	21
22	23	24	25	26	27	28	22	23	24	25	26	27	28	22	23	24	25	26	27	28	22	23	24	25	26	27	28
29	30						29	30	31					29	30						29	30	31				

Obrázek 3: Wikipedie – Kalendárium historických výročí

3.3 On-This-Day

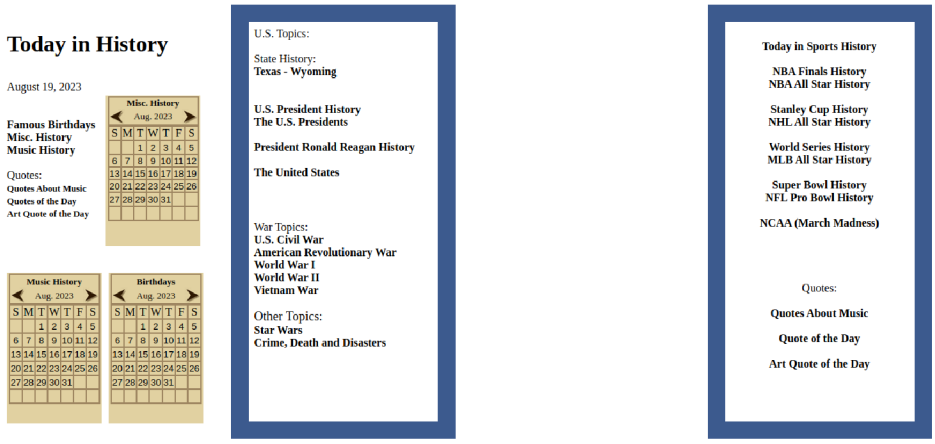
On-This-Day[3]má sice hodně možností, jak vyhledat co se stalo v daný den, ovšem stránka působí velice nepřehledně. Můžeme zde vyhledávat například v oblasti hudby, historie televize, sportu a další.

Klady

- Mnoho způsobů vyhledávání
- Spousta kategorií

Zápory

- Nepřehledné



Obrázek 4: On-This-Day

3.4 Přínosy pro vlastní práci

Pro vlastní práci bych použil podobné vyhledávání jako má webová aplikace On This Day, to znamená, že záznamy půjde vyhledávat pomocí data, typů (kategorií) a textů, ovšem uživatel si bude moci vytvořit vlastní typy a vyhledávat i podle nich.

U Wikipedie je možné se proklikávat pro více informací. V mé práci půjde kliknout na vybrané záznamy a po kliknutí se zobrazí dodatečné informace.

4 Použité technologie

4.1 Frontend a backend

Pro frontend a backend jsem použil full-stack React[4] framework Next.js[5], konkrétně ve verzi 14. Za vznikem stojí americká firma Vercel, tu založil v roce 2015 Guillermo Rauch, který mimo jiné vytvořil známou knihovnu Socket.IO.

Next.js je první framework využívající React Server Components. Jedná se o techniku, kdy se uživatelské rozhraní vykreslí na serveru. To má za následek několik výhod jako je například větší rychlost a bezpečnost. V těchto komponentách můžeme bez problému používat citlivá data, jako mohou být tokeny nebo API klíče, bez rizika vystavení klientovi.

Další výhodou Next.js je cachování, kdy si server pamatuje požadavky od klientů a ukládá si je do své cache. To nám může výrazně zrychlit výkon aplikace. Dále se framework soustředí pro lepší práci se SEO[1].

Next.js používám společně s Typescriptem[6]. Jedná se o nadstavbu jazyka JavaScript, kterou vyvinula společnost Microsoft. Hlavní výhodou je statické typování, což nám pomáhá odhalovat chyby v kódu při vývoji.

Existuje mnoho full-stack frameworků, ze kterých jsem si mohl vybrat. Jako například SvelteKit, ten nabízí SSR a spoustu dalších optimalizací, ale bohužel má menší ekosystém a komunitu. Dalším příkladem je Vue framework Nuxt, bohužel nemám žádné zkušenosti s Vue, tak pro mě byla vhodná volba Next.js.

4.2 Stylování

Pro stylování jsem si vybral svobodný framework Tailwind CSS[7], který využívá utility třídy. To jsou třídy, které slouží pouze k jednomu účelu. Výhodou je, že výsledný CSS soubor je co nejmenší, jelikož v něm jsou pouze ty třídy, které se opravdu používají. Používáním tohoto frameworku docíleme i toho, že naše aplikace bude vypadat ve všech prohlížečích stejně.

Tailwind používá mobile-first přístup, což je způsob návrhu uživatelského rozhraní, kdy jako první navrhujeme a stylujeme aplikaci na mobilní zařízení a poté pomocí breakpointů upravujeme na větší. Práce s tímto framworkem je mnohem rychlejší, jelikož programátor nemusí přemýšlet nad pojmenováváním tříd a také kam je umístí.

CSS frameworků existuje celá řada. Mezi ty největší a nejznámější patří Bootstrap. Ten nabízí velké množství předpřipravených šablon a programátor nemusí spoustu věcí stylovat od začátku. TailwindCSS je ovšem více přizpůsobitelný a více flexibilní.

[1] Optimalizace pro vyhledávače

4.3 Databáze

Jako databázový systém jsem si vybral PosgreSQL. Jedná se o svobodný relační databázový systém, nevlastní jej tedy žádná společnost, ale je vyvíjen komunitou vývojářů a firem. Podporuje standardní SQL s mnoha rozšířeními a funkcemi, díky čemuž můžeme vytvářet komplexnější dotazy a operace nad daty.

Databázový systém jsem využil společně s technologií Prisma ORM,[2][8] která slouží jako abstraktní vrstva mezi aplikací a databází. Díky tomu je práce s daty mnohem pohodlnější. Prisma ORM podporuje různé databázové systémy, což nám umožňuje případně jednoduchou migraci mezi těmito systémy. Výhodou je její typová bezpečnost, kdy se nám automaticky vygenerují typy na základě schématu databáze.

Prisma ORM jsem si vybral hlavně pro její přehlednost, jednoduchost a typovou bezpečnost. Mezi její hlavní konkurenty patří Drizzle ORM, kde mi přehlednost kódu nepřišla tak dobrá jako v Prisma ORM.

[2]Object Relational Mapper

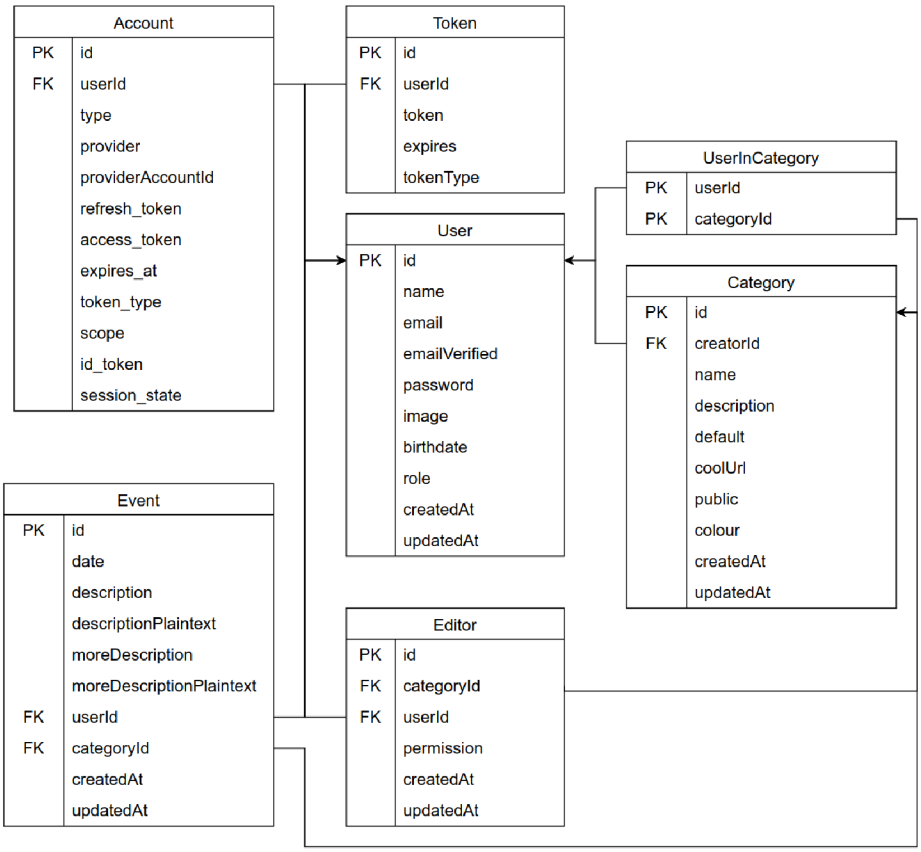
5 Programátorská příručka

Nyní se budeme zabývat technickými částmi aplikace. Nejdříve se zaměříme na strukturu databáze a její hlavní tabulky, poté na strukturu aplikace a nejdůležitější funkcionality aplikace.

5.1 Struktura databáze

Databáze obsahuje celkem sedm tabulek. Pro uživatele slouží tabulka `User`. Pokud se uživatel přihlásí pomocí Google účtu, tak jsou tyto informace uloženy v tabulce `Account`.

Další důležitou tabulkou je `Category`, kde ukládáme informace o kategoriích. Atribut `coolUrl` slouží pro elegantnější adresu, tím dosáhneme lepšího SEO. Následuje tabulka `Event`, která obsahuje všechny události. Atributy `descriptionPlaintext` a `moreDescriptionPlaintext` slouží pro optimalizaci vyhledávání.



Obrázek 5: Relační model

5.2 Adresář app

Next.js od verze 13 využívá *App routeru*, který nám definuje, jaké cesty budou v aplikaci dostupné. Jedná se o speciální složku `app`, která nám reprezentuje cestu na úvodní stránku[3]. Každá další složka reprezentuje další cestu. Adresář `app` funguje společně se starším způsobem tvorby Next.js aplikací a to adresářem `pages`, ale v této práci pracujeme pouze s novější variantou.

Stránky jsou reprezentovány souborem s názvem `page.tsx`. V každé složce může být pouze jeden soubor s tímto názvem. Pokud bychom chtěli, aby stránky sdílely část uživatelského rozhraní, tak k tomu slouží soubor `layout.tsx`. Mezi další speciální soubory patří `loading.tsx`, který se zobrazí, když se stránka načítá, tím lze docílit lepšího uživatelského zážitku.

Ještě se můžeme setkat se složkami, jejichž název je v kulatých závorkách. Tyto složky slouží pouze pro organizaci projektu a nijak neovlivní směrování aplikace. Další důležité složky jsou takové, jejichž název je v hranatých závorkách. Ty označují dynamické cesty[4].

5.3 Adresář components

Je dobrým zvykem rozdělovat části stránek do komponent kvůli přehlednosti. Ukládáme je do podsložek, které názvem vyjadřují, k čemu budou použity nebo na jaké stránce budou použity.

Speciální podsložkou je složka `ui`, ve které jsou uloženy základní upravené elementy. Některé jsou vytvořeny za pomocí knihovny Radix UI[9] a předpřipravených komponent shadcn/ui[10]. Díky těmto dvěma knihovnám docílíme lepšího uživatelského zážitku a lepší přístupnosti.

5.4 Adresář hooks

V této složce se vyskytují uživatelsky vytvořené *hooky* neboli funkce, které nám umožňují interagovat s životním cyklem a stavem komponenty.

Mezi základní vestavěné hooky patří `useState`. Tento hook přijímá počáteční hodnotu a vrací dvě hodnoty. První je proměnná a druhá je funkce, která proměnnou nastavuje, ta začíná vždy s předponou `set`. Jelikož používáme Typescript, tak definujeme i o jaký typ proměnné se jedná. Hlavní vlastnost proměnné je, že jakmile dojde ke změně její hodnoty, tak všechny komponenty, kde se proměnná používá, budou znovu vykresleny.

```
1 const [events, setEvents] = useState<Event[]>([]);
```

Zdrojový kód 1: Ukázka použití `useState`

[3] Tedy www.domena.cz/

[4] Například `/kategorie/[coolUrl]` označuje cestu, kde `[coolUrl]` je dostupný parametr, který nám slouží pro identifikaci kategorie

Další důležitý hook je `useEffect`. Ten slouží pro správu vedlejších efektů, jako je například volání API, ale i aby se nějaký kód vykonal při inicializaci. Jako své argumenty přijímá funkci a pole reaktivních proměnných. Jakmile nějaká z těchto proměnných změní svou hodnotu, spustí se funkce, která byla zadána jako první argument. Jestliže je pole prázdné, funkce se spustí pouze při inicializaci. V následujícím zdrojovém kódu vidíme použití hooku `useEffect` pro zavolání funkce onSubmit. V tomto případě funkce onSubmit ověří emailovou adresu uživatele.

```
1 useEffect(() => {
2   onSubmit();
3 }, []);
```

Zdrojový kód 2: Použití `useEffect` pro zavolání funkce `onSubmit` při inicializaci

5.5 Adresář lib

Tento adresář obsahuje dvě podsložky `data` a `actions`. V podsložce `data` se vyskytují soubory, ve kterých jsou funkce, které přistupují do databáze. Soubor se vždy jmenuje podle toho, k jakým datům přistupuje.

Next.js ve verzi 14 představilo serverové akce neboli funkce, které se vykonávají na straně serveru a používají se hlavně pro zpracovávání dat formulářů. Tyto akce definujeme v souborech v podsložce `actions`. Soubory musí začínat řádkem `"use server"`, který nám značí, že se funkce provede na serveru.

Většinou funkce v těchto souborech přijímají paramter `formData`, v něm jsou uloženy hodnoty z formuláře. Pokud bychom chtěli, aby funkce přijímala další parametr, musíme ho s funkcí svázat.

Pro lepší a bezpečnější práci s formuláři používáme knihovnu React Hook Form[11] a Zod[12]. Pomocí knihovny Zod si definujeme schéma formuláře. Toto schéma je poté validováno knihovnou React Hook Form a zajistí nám, že ve formuláři budou pouze hodnoty, které jsme definovali ve schématu.

```
1 const AddEditorSchema = z.object({
2   email: z.string().email({ message: "Není ve správném tvaru" }),
3   permissionId: z.string({ invalid_type_error: "Musí být řetězec" })
4   .min(1, "Oprávnění musí být vyplněno")
5   .max(20, "Oprávnění je moc dlouhé"),
6 });
```

Zdrojový kód 3: Schéma pro validaci přidání editora

5.6 Adresář prisma

Tato složka obsahuje konfigurační soubor `schema.prisma`, ve kterém najdeme definici *modelů*. Model je reprezentace datové struktury, která představuje konkrétní entitu v databázi. V následujícím zdrojovém kódu můžeme vidět definici modelu pro událost. Na řádku 2 definujeme unikátní identifikátor `id`, ten má výchozí hodnotu nastavenou na náhodně vygenerovaný řetězec. Na řádku 10 definujeme relaci s modelem `User`.

```
1  model Event {
2    id                  String @id @default(cuid())
3    date                DateTime @db.Date
4    description         String @db.Text
5    descriptionPlainText String @db.Text
6    moreDescription     String? @db.Text
7    moreDescriptionPlainText String? @db.Text
8
9    userId  String
10   user    User   @relation(
11     fields: [userId], references: [id], onDelete: Cascade
12   )
13   categoryId String
14   category Category @relation(
15     fields: [categoryId],references: [id], onDelete: Cascade
16   )
17
18   createdAt DateTime @default(now())
19   updatedAt DateTime @updatedAt
20
21   @@index([userId])
22   @@index([categoryId])
23 }
```

Zdrojový kód 4: Ukázka modelu `Event`

5.7 Autentizace uživatelů

Pro autentizaci uživatelů používáme knihovnu Auth.js[13], konkrétně ve verzi 5. Umožňuje autentizaci jak za pomocí klasického emailu a hesla, tak i pomocí oAuth[5] poskytovatelů.

Nejdříve musíme nadefinovat jaké způsoby přihlášení uživateli umožníme. V našem případě se jedná o `Credentials` a `Google`. Credentials znamená, že uživatel má možnost se přihlásit pomocí emailu a hesla.

Knihovna má dostupný Prisma adaptér, díky kterému se některé informace o uživatelech automaticky vkládají do databáze. Mezi tyto informace patří například přihlašování pomocí Google účtu, to se ukládá do tabulky `Account`.

[5]Protokol pro bezpečnou autentizaci a autorizaci

5.8 Univerzální modální okno

Veškeré úpravy se provádějí v univerzálním modálním okně. Díky tomuto oknu nemusíme vytvářet každý formulář zvlášť. Stačí pouze nadefinovat nadpis, jaké prvky má formulář obsahovat a akci, která se má vykonat po odeslání formuláře.

V následujícím zdrojovém kódu vidíme ukázku vytvoření modálního okna pro přidání editora. Definujeme typ formuláře, jestli se jedná o formulář pro přidávání nebo upravování. Dále název schématu, který je použit při validaci formuláře. Tyto schémata jsou uloženy v souboru `lib/schemas.ts` a definují se pomocí knihovny Zod.

Následuje definice samotných polí formuláře. Každé musí obsahovat název, typ, popis a počáteční hodnotu. Každý typ může mít navíc různé hodnoty. Nakonec přidáme akci, která se provede po odeslání formuláře.

Pro otevření tohoto okna používáme komponentu `OpenModalButton`, do které předáme data o okně. Následně se nám vykreslí tlačítko s funkcionalitou pro otevření modálního okna.

```
const modalDataAddEditor: ModalData = {
  universal: {
    header: "Přidat editora",
    form: {
      type: "add",
      formSchemaName: "AddEditorSchema",
      formFields: [
        {
          name: "email",
          type: "email",
          label: "Email",
          defaultValue: "",
        },
        {
          name: "permissionId",
          type: "select",
          label: "Oprávnění",
          placeholder: "Vyber oprávnění editora",
          defaultValue: "",
          options: permissionsArray,
        },
      ],
      action: addEditorWithCoolUrlAction,
    },
  },
};
```

Zdrojový kód 5: Data pro vytvoření univerzálního okna pro přidání editora

5.9 Tmavý a světlý režim

V dnešní době už je zvykem, že webové aplikace mají dostupný světlý i tmavý režim. Aby tato funkcionalita fungovala správně a spolehlivě, rozhodl jsem se použít knihovnu `next-themes`[14]. Jelikož používáme Tailwind CSS, tak využijeme prefixu `dark:`. Díky tomuto prefixu stylujeme elementy, jak mají vypadat v tmavém režimu.

```
1 <p className="text-black dark:text-white">Text</p>
```

Zdrojový kód 6: Ukázka stylování tmavého režimu pomocí Tailwind CSS

5.10 Lazy loading

Lazy loading je technika načítání webové aplikace, kdy část aplikace je načtena, až když je potřeba. Díky tomu dosáhneme větší rychlosti načítání aplikace. Na následujícím zdrojovém kódu můžeme vidět ukázku použití funkce `dynamic` pro dynamické načítání komponenty. Aby nedocházelo k posunům layoutu aplikace v moment, kdy se komponenta načítá, využívá se skeletonů. Skeleton je pouze napodobenina toho, jak vypadá finální vykreslená komponenta.

```
1  const CategItem = dynamic(() => import("./categories-item"), {
2    ssr: false,
3    loading: () => <Skeleton className="w-full h-40 rounded-2xl" />,
4  });
```

Zdrojový kód 7: Ukázka lazy loading

5.11 SEO a metadata

Optimalizace pro vyhledávače je v dnešní době čím dál důležitější téma. Jedná se o metody, které mají za cíl, aby se webová stránka zobrazovala na lepších místech vyhledávačů.

Jednou z možností, jak si vylepšit své výsledky ve vyhledávání, je vytvoření souboru `sitemap.xml`. Tento soubor napomáhá vyhledávacím robotům procházet stránky webové aplikace.

Další možností pro lepší pozici ve vyhledávání jsou metadata, která se většinou nacházejí v hlavičce stránky. V Next.js můžeme metadata vytvořit například v souboru `layout.tsx`. Stačí exportovat proměnnou s názvem `metadata` a vyplnit její obsah. Tyto metadata budou dostupné všem stránkám, které sdílejí stejný layout. Jednolivé stránky mohou metadata přepisovat.

```
export const metadata: Metadata = {
  title: {
    template: "%s | V tento den",
    default: "V tento den",
  },
  description: "Historické, aktuální a budoucí události.",
  applicationName: "V tento den",
  keywords: ["V tento den, kalendář, události, dnes, zajímavosti"]
};
```

Zdrojový kód 8: Metadata v souboru `app/layout.tsx`

5.12 Google Kalendář

Pro synchronizaci a import událostí z Google Kalendáře jsem využil knihovnu googleapis[15]. Synchronizace událostí probíhá tak, že nejdříve získáme všechny kalendáře uživatele. Pro ověření uživatele potřebujeme jeho `refresh_token` a `access_token`. Tyto tokeny máme automaticky k dispozici, když se uživatel zaregistruje pomocí Google účtu. Následně zjistíme, zda existuje kalendář se zadaným jménem, pokud ne, tak se vytvoří nový a do tohoto kalendáře vložíme události.

Import událostí z Google Kalendáře probíhá velmi podobně. Ověří se uživatel, zjistí se, zda existuje kalendář se zadaným jménem a pokud neexistuje, tak se zobrazí chyba. Získáme události z kalendáře a importujeme je do kategorie.

5.13 Rozšíření aplikace

I když aplikace nabízí širokou skálu funkcí, stále se dá rozšířit a optimalizovat. Každá tato nová funkcionalita by zajistila uživatelům větší možnosti při používání aplikace. Mezi tyto funkcionality by mohlo patřit:

- Možnost přidávání obrázků do událostí
- Nahlašování chyb v událostech
- Komunikační kanál pro editory

6 Uživatelská příručka

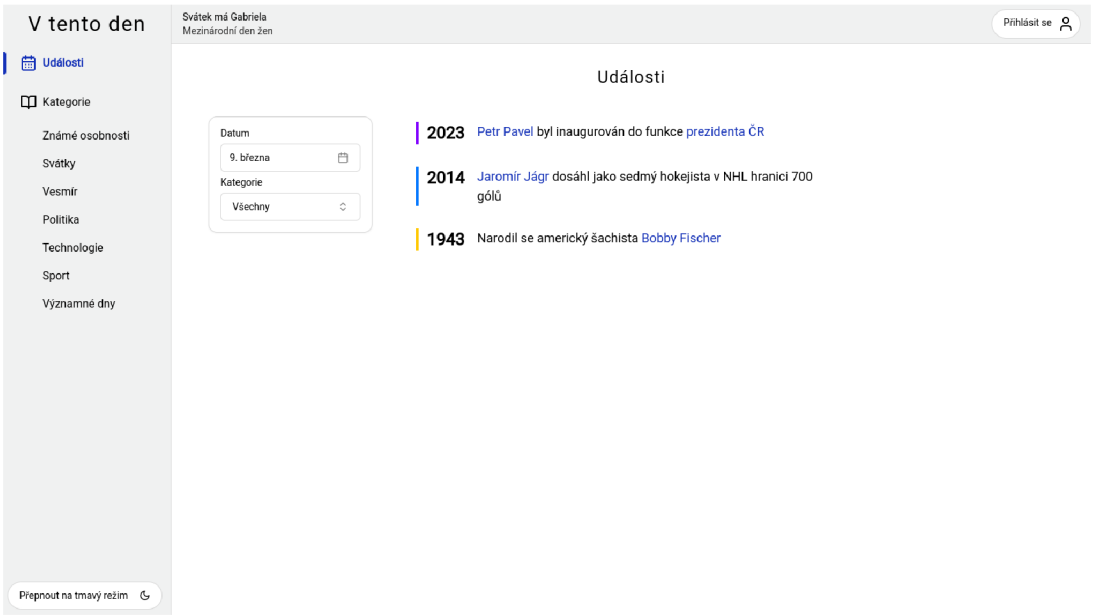
Následující část nám poslouží jako příručka pro uživatele, který s aplikací nemá žádné zkušenosti.

6.1 Rozhraní webové aplikace

Rozhraní je velice jednoduché a minimalistické. Skládá se ze tří hlavních částí. Tou první je levý postranní panel, na kterém se nachází název aplikace, navigace a přepínač světlého a tmavého režimu. První položkou této navigace je úvodní stránka, na které se zobrazují události aktuálního dne. Další položka „Kategorie“ odkazuje na stránku, která zobrazuje seznam všech kategorií, které jsou nám dostupné. Pod touto položkou jsou odkazy na kategorie, které odebíráme.

Druhou částí je horní lišta. Ta obsahuje informaci, kdo má dnes svátek a jaký je mezinárodní den. Dále se zde nachází tlačítko pro přihlášení uživatele. Pokud je uživatel přihlášen, tak se zobrazí tlačítko s ikonou, které po kliknutí zobrazí nabídku, ve které máme možnost přejít na svůj profil nebo se odhlásit. Je-li uživatel administrátorem aplikace, objeví zde i odkaz na stránku pro administrátory.

Třetí a nejdůležitější část je samotný obsah. V případě úvodní stránky jsou obsahem události. Ve výchozím stavu se zobrazují události pro dnešní den z odebíraných kategorií, ovšem to můžeme změnit pomocí filtru, ve kterém si zvolíme, který den a jaké kategorie chceme zobrazit. Každá událost patří do nějaké kategorie a pro lepší přehlednost, má přidělené barevné označení.

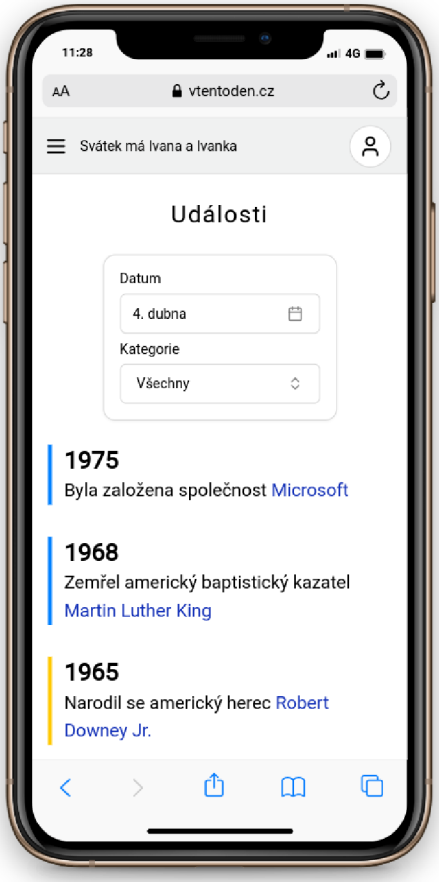


Obrázek 6: Layout webové aplikace

6.2 Mobilní verze

V dnešní době je čím dál důležitější, aby se webové aplikace zobrazovaly správně i na mobilních zařízení, jelikož spousta uživatelů nemusí mít ani počítač, nebo je pro ně pohodlnější mobilní zařízení.

Hlavním rozdílem mezi verzí pro desktop a mobilní zařízení je navigace. V případě mobilního zařízení je řešena pomocí hamburger menu, kdy se nám po kliknutí na ikonku navigace zobrazí levý postranní panel.



Obrázek 7: Webová aplikace na mobilním zařízení

6.3 Příhlášení a registrace

Pro registraci potřebuje uživatel vyplnit své uživatelské jméno, email a heslo. Následně na zadaný email přijde potvrzovací odkaz. Bez potvrzeného účtu se nelze přihlásit.

Dalším způsobem přihlášení je přes Google účet, kdy je účet automaticky ověřen. Pokud jsme zvolili tento způsob přihlášení, tak se nám otevře možnost synchronizovat si události se svým Google Kalendářem.

Pokud uživatel zapomene své heslo, může si ho samozřejmě obnovit. To udělá kliknutím na odkaz „Zapomněli jste heslo?“, kde zadá svůj email, na který mu přijde odkaz pro obnovu hesla, který má však omezenou platnost.

6.4 Kategorie

Na této stránce nalezneme seznam všech dostupných kategorií. Kategorie dělíme na tři typy. První jsou obecné, ty jsou vytvořeny administrátory aplikace a jsou dostupné všem uživatelům, kteří je automaticky odebírají při registraci. Poté jsou veřejné, to jsou kategorie, které vytvořili ostatní uživatelé a jejich viditelnost je nastavena pro všechny. Aby se kategorie mohla stát veřejnou, musí mít alespoň 10 událostí. Posledním druhem jsou privátní, ty jsou viditelné pouze pro daného uživatele a těm uživatelům, kterým to umožní.

U každé kategorie nalezneme tlačítko „Odebírat“ nebo „Přestat odebírat“. Toto tlačítko nám signalizuje, jestli uživatel kategorii odebírá, či nikoliv. Pouze ty kategorie, které uživatel odebírá, mu budou zobrazovány na stránce s událostmi a na bočním panelu.

Pokud je uživatel přihlášen, nalezne zde i tlačítko pro vytvoření vlastní kategorie. Bude muset vyplnit její název, barvu a krátký popis, který ostatním uživatelům shrne, co se v kategorii bude nacházet.

6.5 Konkrétní kategorie

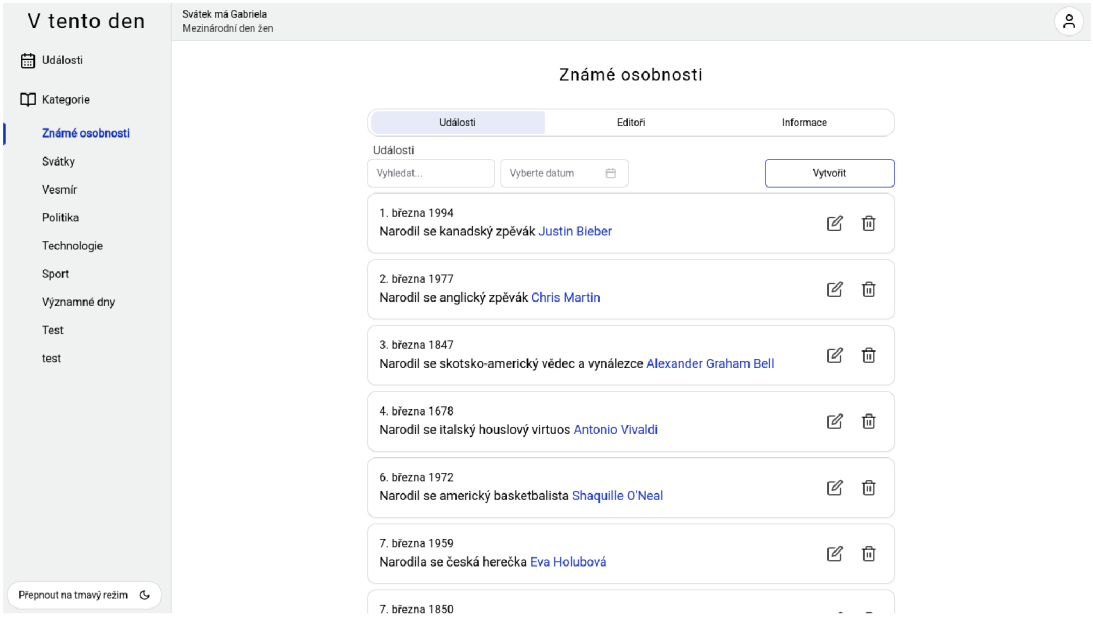
Stránka konkrétní kategorie obsahuje její veškeré informace. Je rozdělená na čtyři části, které přepínáme pomocí záložek. Ovšem některé záložky jdou vidět pouze za určitých podmínek.

Jako běžný uživatel vidíme všechny události a základní informace o kategorii a máme možnost události vyhledávat pomocí textu a data.

Zakladatel kategorie může ostatní uživatele pomocí emailu přidat do editorů. Editoři se dělí na dva typy podle pravomocí. Ti s nižšími pravomocemi mohou spravovat události a ti s vyššími mohou spravovat i ostatní editory. Pouze zakladatel může kategorii odstranit a importovat do ní události.

Pokud je kategorie privátní, tak zakladatel má dostupnou záložku „Uživatelé“, kde může přidat uživatele do své kategorie. Tito uživatelé začnou kategorii automaticky odebírat a samozřejmě mohou odběr i kdykoliv zrušit.

Pro vytvoření události stačí kliknout na tlačítko „Vytvořit“, tím se otevře modální okno s formulářem. U každé události je povinné vyplnit její datum a popis, ten se zadává v editoru, ve kterém je možné udělat text tučný, kurzívou, podrtžený, přeškrtnutý nebo zvýraznit barvou, ale můžeme vložit i seznam číselný nebo s odrážkami. Nepovinnou položkou je „Další popis“, ten se zobrazí po kliknutí na událost jako dodatečné informace.



Obrázek 8: Ukázka konkrétní kategorie

6.6 Profil

Přihlášený uživatel si může spravovat své informace na stránce profilu, na kterou se dostaneme kliknutím na ikonu v horní liště a zvolením „Profil“. Můžeme zde změnit své uživatelské jméno a datum narození. Je zde i možnost svůj účet odstranit.

6.7 Admin

Někteří uživatelé mohou mít zvýšené pravomoce. Tito uživatelé mají přístupnou stránku pouze pro adminy. Na této stránce mohou upravit pravomoce ostatním uživatelům. Je-li uživatel admin, tak může kategorii udělat obecnou.

6.8 Export událostí

Každý uživatel, který odebírá nějakou kategorii, si může exportovat všechny její události do formátu iCal, což je standard pro výměnu kalendářových dat. Můžeme tak učinit v záložce „Informace“, kde najdeme tlačítko „Exportovat události“. Po kliknutí na toto tlačítko se nám zobrazí modální okno. Události můžeme exportovat pouze jako samotný text nebo máme možnost zachovat i HTML značky, které byly automaticky přidány z Lexical editoru.

Zachování HTML značek se může hodit, pokud chceme události vložit do jiné kategorie a zachovat jejich formát.

6.9 Import událostí

Zakladatel kategorie má možnost importovat události ze souboru. Ten musí být ve formátu iCal. Další možností, jak importovat události je pomocí Google Kalendáře. Tato možnost je umožněna pouze těm uživatelům, kteří jsou přihlášení pomocí Google účtu. Pro import stačí zadat název kalendáře, ze kterého chceme události importovat. Pokud ovšem chceme importovat osobní kalendář, tak zadáme email uživatele. Pokud má událost v názvu rok, tak se při importu změní rok události na ten, který je v názvu a nepoužije se aktuální.

6.10 Synchronizace s Google Kalendářem

Uživatel, který je přihlášený pomocí Google účtu, si může u odebírané kategorie vybrat události, které se mu synchronizují s Google Kalendářem. Následně se mu vytvoří kalendář se zadaným názvem, kam se všechny události vloží.

Jelikož export událostí probíhá ve standardizovaném formátu iCal, tak uživatel má možnost si události exportovat a následně je naimportovat do jakékoli aplikace, která tento formát podporuje, včetně Google Kalendáře.

Závěr

Výsledkem práce je webová aplikace, která slouží pro správu událostí. Umožňuje registraci a přihlašování uživatelů a vytváření vlastních kategorií a událostí. Uživatel má možnost synchronizovat své události s Google Kalendářem, případně je může importovat a exportovat ve formátu iCal. Díky tomu může přenášet své události mezi různými aplikacemi.

Aplikace má jednoduché uživatelské rozhraní, je přístupná a použitelná bez ohledu na to, na jakém zařízení ji uživatel používá. Pro lepší uživatelský zážitek obsahuje i tmavý režim.

V budoucnu bych aplikaci rád rozšířil o několik dalších funkcionalit. Může se jednat například o hlášení chyb v události, kdy uživatel může nahlásit, že nějaká událost obsahuje chybu. Dále by bylo vhodné uživatelům umožnit vkládat obrázky do událostí.

Zpětně bych kladně zhotnotil výběr technologií. Ať už se jedná o framework Next.js s Typescriptem nebo Tailwind CSS, tak mi tyto technologie zpříjemnily práci na této aplikaci. Určitě bych tyto technologie zvolil i při dalším větším projektu.

Conclusions

The result of this work is a web application used for event management. It allows users to register and login, and create their own categories and events. The user has the possibility to synchronise his events with Google Calendar or to import and export them in iCal format. This allows them to transfer their events between different applications.

The application has a simple interface that is accessible and usable regardless of the device on which the user is using it. It also includes a dark mode for a better user experience.

In the future I would like to add some more functionality to the application. For example, event error reporting, where a user can report that an event contains an error. It would also be useful to allow users to insert images into events.

Looking back, I would rate the choice of technologies as positive. Whether it is the Next.js framework with Typescript or Tailwind CSS, these technologies have made working on this application more enjoyable for me. I would definitely choose these technologies for next larger project.

A Obsah elektronických dat

`text/`
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vytvoření PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu a příloh, vložené obrázky, apod.

`README.txt`
: Textový soubor s informacemi o zprovoznění webové aplikace, včetně požadavků potřebných pro bezproblémový provoz a webové adresy, na které je aplikace nasazena.

`src/`
: Zdrojový kód webové aplikace V tento den.

Literatura

[1] Ltd., On This Day Pte. *Today in History, Film, Music and Sport* [online]. [cit. 2023-5-6]. Dostupný z: ⟨https://www.onthisday.com⟩.

[2] Wikipedie. *Kalendárium historických výročí* [online]. [cit. 2023-5-6]. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Kalend%C3%A1rium_historick%C3%BDch_v%C3%BDro%C4%8D%C3%AD⟩.

[3] On-This-Day. *Daily history, famous birthdays and music history* [online]. [cit. 2023-5-6]. Dostupný z: ⟨https://on-this-day.com⟩.

[4] Meta. *React* [online]. [cit. 2024-2-26]. Dostupný z: ⟨https://react.dev⟩.

[5] Vercel. *Next.js by Vercel - The React Framework* [online]. [cit. 2024-2-26]. Dostupný z: ⟨https://nextjs.org⟩.

[6] Microsoft. *Typescript* [online]. [cit. 2024-3-5]. Dostupný z: ⟨https://www.typescriptlang.org/⟩.

[7] Wathan, Adam. *Rapidly build modern websites without ever leaving your HTML* [online]. [cit. 2024-2-26]. Dostupný z: ⟨https://tailwindcss.com⟩.

[8] Prisma. *Prisma | Simplify working and interacting with databases* [online]. [cit. 2024-2-26]. Dostupný z: ⟨https://www.prisma.io⟩.

[9] WorkOS. *Radix UI* [online]. [cit. 2024-2-27]. Dostupný z: ⟨https://www.radix-ui.com⟩.

[10] shadcn. *shadcn/ui* [online]. [cit. 2024-2-27]. Dostupný z: ⟨https://ui.shadcn.com⟩.

[11] BEEKAI. *React Hook Form* [online]. [cit. 2024-2-28]. Dostupný z: ⟨https://react-hook-form.com⟩.

[12] colinhacks. *Zod* [online]. [cit. 2024-2-28]. Dostupný z: ⟨https://zod.dev⟩.

[13] Orbán, Balázs. *Auth.js* [online]. [cit. 2024-2-26]. Dostupný z: ⟨https://authjs.dev⟩.

[14] pacocoursey. *next-themes* [online]. [cit. 2024-3-3]. Dostupný z: ⟨https://github.com/pacocoursey/next-themes⟩.

[15] Google. *googleapis* [online]. [cit. 2024-3-5]. Dostupný z: ⟨https://github.com/googleapis/google-api-nodejs-client⟩.