FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Portál pro organizování vědeckých konferencí
Student:	Tomáš Cabák
Vedoucí:	Ing. Petr Pauš, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem práce je vytvořit webový portál pro organizování vědeckých konferencí. Portál by měl umět zobrazit informace o konferenci, seznam účastníků a abstrakty a dodatečné info stránky včetně fotogalerie. Měl by též umožňovat registraci účastníků s abstrakty a měl by obsahovat administrační část s editací dat o konferenci, jednotlivých infostránek a údajů o účastnících. Data se budou ukládat do databáze.

Pokyny k vypracování:
1. Analyzujte vhodné technologie pro tvorbu front-endu webového portálu včetně administrace.
2. Analyzujte možnosti ukládání dat do databáze a přístup k nim.
3. Navrhněte prototyp portálu včetně uživatelského rozhraní a administrační části.
4. Implementujte prototyp.
5. Proveďte vhodné testování.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 20. listopadu 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Portál pro organizování vědeckých konferencí

Tomáš Cabák

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petr Pauš, Ph.D.

10. května 2023

Poděkování

Chtěl bych poděkovat všem, kteří mi pomohli při psaní této bakalářské práce. Nejprve bych chtěl poděkovat svému vedoucímu práce za cenné rady, trpělivost a podporu v průběhu psaní práce. Dále bych rád poděkoval svým přátelům a rodině za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 10. května 2023

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Cabák, Tomáš. *Portál pro organizování vědeckých konferencí.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Práce se zabývá postupem při vývoji nového Webového portálu pro organizaci každoroční vědecké konference, který nahradí jeho současně používanou podobu. Vývoj začíná analýzou současného řešení, která je následována návrhem a volbou technologií nového řešení, jeho implementací, nasazením a testováním. Nově vytvořený Webový portál je implementován jako full-stack webová aplikace ve frameworku Next.js. Zájemcům a účastníkům přehledně poskytuje informace o konferenci a možnost se registrovat. Organizátorům umožňuje plánování konference včetně úpravy obsahu organizačních stránek a galerie, správy účastníků a tvorby programu. Webový portál je spolu s databází nasazen na platformě Digital Ocean, ale nabízí i možnost lokálního nasazení pomocí nástroje Docker.

Klíčová slova webový portál, jednostránková webová aplikace, organizování konferencí, full-stack, TypeScript, Next.js

Abstract

The work deals with the process of developing a new Web portal for organizing the annual scientific conference, which will replace its currently used form. The development begins with an analysis of the current solution, followed by the design and selection of technologies for the new solution, its implementation, deployment, and testing. The newly created Web portal is implemented as a full-stack web application in the Next.js framework. It provides clear information about the conference and the possibility to register for interested parties and participants. It allows organizers to plan the conference, including editing the content of organizational pages and gallery, managing participants, and creating the program. The Web portal is deployed on the Digital Ocean platform along with the database, but also offers the possibility of local deployment using Docker.

Keywords web portal, single page web application, conference organization, full-stack, TypeScript, Next.js

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

Úvod

Již 14. rokem se spolu se studentskou konferencí koná konference Workshop on Scientific Computing jako multilaterální setkání pod záštitou ČVUT v Praze. Organizování těchto akcí může být velmi složitý a časově náročný proces, který vyžaduje pečlivé plánování. Spolehlivá platforma, jako je webová aplikace, je pro organizátory důležitým nástrojem, který umožňuje zefektivnění procesu plánování, minimalizování chyb a nejasností a zlepšení komunikace s účastníky. Webová aplikace umožní organizátorům soustředit se na ty nejdůležitější aspekty, jako jsou obsah programu celé konference a její účastníci.

Práce je určena zejména pro organizátory, ale také pro všechny, kteří se o každoroční konferenci zajímají. Webový portál pro organizování vědeckých konferencí (dále Webový portál) zájemcům a účastníkům konference přehledně poskytne organizační informace, denní program, seznam registrovaných účastníků s jejich kontribucemi a fotogalerii. Zájemci budou mít rovněž možnost registrovat se do konference s vlastním příspěvkem. Správci prostřednictvím administrační části Webového portálu budou moci spravovat veškerá data a informace týkající se konference a jejich účastníků, nahrávat fotografie a vytvářet/upravovat program.

Cílem bakalářské práce je vytvořit Webový portál jako full-stack webovou aplikaci a analyzovat jeho aktuálně používanou podobu, která vznikala v průběhu několika let, a jejíž návrh a implementace neumožňuje snadné rozšiřování a údržbu.

Práce je rozdělena do několika kapitol popisujících postup při realizaci Webového portálu. Kapitoly následují typickou strukturu při vývoji softwarového projektu.

První kapitola *Analýza* se zabývá rozborem aktuálního řešení a jeho problémů, analýzou požadavků a analýzou problémové domény. Druhou kapitolou je *Návrh prototypu*, kde dojde k volbě přístupu při tvorbě Webového portálu, k volbě vhodných technologií a architektury rozhraní koncových bodů serveru, a k návrhu uživatelského rozhraní. Následně jsou v kapitole *Realizace* popsány

použité nástroje a stěžejní části implementace prototypu. V kapitole *Nasazení* je přiblížen postup při nasazení výsledného prototypu a v kapitole *Testování* postup při testování během vývoje a po nasazení.

Kapitola 1

Analýza

Tato kapitola je rozdělena na tři sekce zabývající se analýzou aktuálního řešení, požadavků a problémové domény. Většina informací v této kapitole vychází z aktuálně používaného Webového portálu [1] a z konzultací s jeho autorem.

1.1 Analýza aktuálního řešení

V této sekci dojde k analýze aktuálně používaného Webového portálu. Popis a architektura současného řešení pomohou přiblížit problematiku této práce a jsou následovány rozborem nejzávažnějších problémů, které se v současném řešení nachází.

1.1.1 Popis

Současný Webový portál vznikal od roku 2010 jako platforma naplňující specifické požadavky organizátorů a účastníků každoroční konference. Hlavní motivací pro jeho vznik bylo předávání informací účastníkům konference a usnadnění samotné organizace.

Stránky Webového portálu jsou v závislosti na oprávnění uživatele rozděleny na veřejnou a administrátorskou část. Všechny stránky mají společnou hlavičku obsahující název konference, datum a místo konání, a název fakulty a katedry, která konferenci pořádá.

1.1.1.1 Veřejná část

Veřejná část Webového portálu je tvořena stránkami, které zájemcům a účastníkům poskytují informace o konferenci a jejím průběhu. Tyto stránky jsou *Home*, *Register*, *Participants*, *Abstracts*, *Programme* a *Travel*.

Domovská stránka *Home* uvádí, čeho se konference týká a pro koho je určena. Obsahuje informace o organizátorech a registraci a poskytuje důležité odkazy.

Stránka *Register* obsahuje registrační formulář a detailní informace k registraci, jako jsou datum uzavření možnosti registrování a poplatek za účast. Prostřednictvím formuláře se uživatel může registrovat a současně vložit svůj příspěvek do konference. Formulář je rozdělen na dvě části. V první části se vyplňují osobní informace, jako jsou jméno, e-mailová adresa, příslušnost k určité univerzitě, způsob účasti (on-line nebo prezenční), poštovní adresa pro zaslání faktury, zda je uživatel studentem a dodatečné informace pro organizátory. Ve druhé části se vyplňují informace týkající se nepovinného příspěvku do konference. Těmito informacemi jsou název příspěvku, jména spoluautorů, příslušnost spoluautorů k určité univerzitě a samotný abstrakt příspěvku.

Stránka *Participants* je seznamem účastníků. U každého účastníka je jeho příslušnost k univerzitě a odkaz na stránku *Abstracts*, který po kliknutí naviguje k příspěvku účastníka.

Stránka *Abstracts* obsahuje seznam příspěvků všech účastníků. U každého příspěvku je uveden jeho název, jméno autora a spoluautorů, příslušnost k univerzitě a abstrakt příspěvku. Zároveň se od každého příspěvku dá navigovat na stránku *Programme*, kde se konkrétní příspěvek v denním programu zvýrazní.

Poslední veřejnou stránkou je stránka *Travel*, která popisuje cestu na místo konference. Instrukce jsou doplněny interaktivní mapou.

1.1.1.2 Administrátorská část

Administrátorská část Webového portálu je přístupná pouze administrátorům po přihlášení. Administrátoři jsou v ní schopni upravovat informace o konferenci, spravovat účastníky a jejich příspěvky, vytvářet program a nahrávat fotografie.

Po přihlášení se zobrazí úvodní stránka sloužící jako rozcestník mezi jednotlivými editačními stránkami. Současně také obsahuje seznam všech účastníků s jejich e-mailovými adresami. Editační stránky jsou *Edit settings*, *Edit abstracts*, *Edit programme* a *Upload files*.

Stránka *Edit settings* slouží k úpravě obsahu veřejných stránek a k úpravě globálních údajů v hlavičce každé stránky. Úpravy jsou realizovány prostřednictvím textového pole, do kterého se vkládá text a případně řetězce HTML (HyperText Markup Language) kódu.

Stránka *Edit abstracts* zobrazuje tabulku účastníků se všemi jejich údaji, které byly při registraci zadány. Pod tabulkou se nachází editační sekce pro každého účastníka, umožňující změnu údajů nebo úplné odstranění účastníka.

Stránka *Edit programme* je editor denního programu konference. Umožňuje dynamické přidávání dní konference a vytváření struktury programu. Program dne je tvořen jednotlivými bloky, kterým zasedá předseda, a ve kterých probíhají prezentace příspěvků s pevně danou délkou v minutách. Mimo prezentací příspěvků je do programu také možné přidat přestávky a další libovolné události.

Poslední administrátorskou stránkou je stránka *Upload files* umožňující nahrávání obrázků, které jsou následně uloženy na disku serveru. Přestože je možné fotografie nahrávat, Webový portál neobsahuje galerii.

1.1.2 Architektura a implementace

Z pohledu logické architektury dle [2] je Webový portál jednovrstvá monolitická aplikace. Problémy této architektury budou spolu s ostatními problémy detailněji rozebrány v sekci 1.1.3.

Webový portál je implementován převážně v programovacím jazyce PHP (PHP: Hypertext Preprocessor). Každou stránku tedy reprezentuje její PHP soubor, ze kterého se na straně serveru, před odesláním odpovědi uživateli, dynamicky vytváří výstupní HTML stránky [3]. Interakce uživatele v klientské části jsou obohaceny JavaScript skripty, které využívají funkcí knihovny jQuery [4]. Data se ukládají do souborů uložených na disku serveru a přistupuje se k nim za použití klasických PHP funkcí.

1.1.3 Problémy

Nejzávažnějšími problémy jsou samotná architektura, způsob úpravy organizačních informací a obsahu stránek, ukládání a přístup k datům a přihlašování do administrátorské části Webového portálu. Další problém je způsob správy účastníků a příspěvků.

1.1.3.1 Architektura

Jak již bylo zmíněno v sekci 1.1.2, Webový portál je jednovrstvá monolitická aplikace. Kód prezentační logiky je tak míchán s kódem řídícím aplikační logiku (viz. ukázka kódu 1.1). To ho činí nepřehledným a neudržovatelným a změny prezentační logiky mohou způsobit nutnost úpravy i samotné aplikační logiky.

Prezentační logika je v některých PHP souborech stránek implementována jako výpis čistého HTML řetězce PHP funkcí *echo*, což opět způsobuje velmi nepřehlednou strukturu kódu bez možnosti využití zvýrazňování a napovídání editoru. V ostatních souborech jsou použity běžné HTML značky prokládané PHP skripty.

1.1.3.2 Úprava organizačních informací

Organizační informace reprezentující obsah stránek se vkládají do textového pole jako čistý HTML kód a vložení delších řetězců způsobí zamrznutí celé záložky prohlížeče. Navíc je takto vkládaný kód bez dodatečného manuálního zarovnání velice nepřehledný a nepohodlný na úpravu.

```
...
</div>
<?php
$program = load_file('settings/program_data.txt');
$abstracts = load_file('settings/abstract_data.txt');

if (is_array($program) && is_array($abstracts))
{

if (isset($program['intro']) && strlen($program['intro'])>0) {
  echo '<div class="row">';
  echo '<div class="col s12 m12
  ↪  l12">'.$program['intro'].'</div>'."\n";
  echo '</div>';
}

foreach ($program as $id => $day)
...
```

Ukázka kódu 1.1: Prezentační a aplikační logika v jednom souboru monolitické aplikace

1.1.3.3 Ukládání a přístup k datům

Data o konferenci jsou ukládána do tří textových souborů ve formátu JSON (JavaScript Object Notation) uložených v jednom adresáři společně s celou aplikací. Jednotlivé soubory obsahují obecné informace o konferenci, data o programu a data o účastnících. K souborům se přistupuje při každém vykreslování stránky, kdy se z nich získají jen informace pro stránku potřebné. Dle [5] má takovýto způsob ukládání dat několik nevýhod:

- Soubory jsou přístupné všem, kteří mají přístup k souborovému systému serveru.
- Znemožnění efektivní práce s daty a vyhledávání v nich. V případě větších souborů může dojít k výraznému zpomalení aplikace.
- Soubory nejsou přizpůsobeny pro simultánní přístup několika uživatelů, což může způsobovat konflikty a inkonzistence mezi daty.
- Riziko poškození souborů a ztráty přístupu k uloženým datům.
- Je obtížné dodržovat strukturu dat a typy konkrétních hodnot.

1.1.3.4 Přístup do administrátorské části

Webový portál poskytuje pouze jeden sdílený administrátorský profil. To může vést k několika problémům:

- Riziko úniku přihlašovacích údajů, protože všichni administrátoři musí používat stejné.
- V případě nechtěných změn je obtížné dohledat, kdo tyto změny provedl. [6]

1.1.3.5 Správa účastníků a příspěvků

V administrátorské tabulce registrovaných účastníků nelze vyhledávat a řadit. Rovněž zde není stránkování a všechna data se tak zobrazují pod sebou na jedné stránce. Stejný problém je s formuláři pro úpravu účastníků a jejich příspěvků. Pod tabulkou se pro každého účastníka vykreslí formulář pro jeho úpravu, kvůli čemuž je výsledná stránka příliš dlouhá a nepřehledná.

1.2 Analýza požadavků

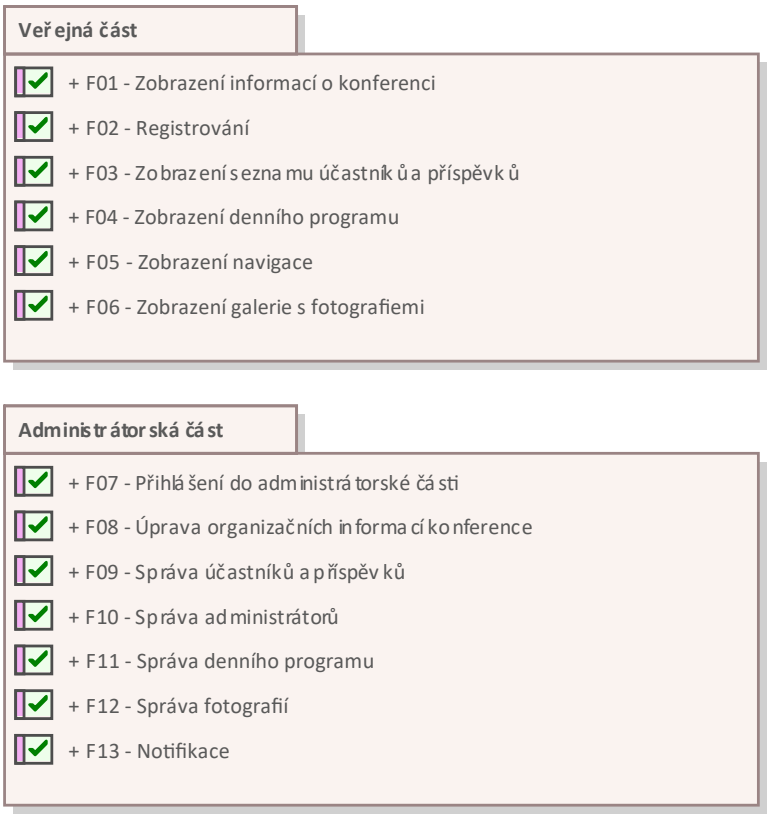
V této sekci se definují funkční a nefunkční (obecné) požadavky, které současný Webový portál splňuje, a které by měl splňovat i nově vytvořený Webový portál. Přidají se také nové požadavky rozšiřující funkcionality systému.

1.2.1 Funkční požadavky

Funkční požadavky popisují chování samotného systému. Specifikují funkce nebo operace, které musí být systém schopen provést jako reakce na určité vstupy. Pokud se tyto požadavky nesplní, systém nebude fungovat. [7]

Během diskuze s vedoucím této práce byly vzhledem k současnému Webovému portálu vzneseny nové požadavky. Přibyly funkční požadavky na vyhledávání účastníků a jejich příspěvků, zobrazení galerie, správu administrátorských účtů a notifikace.

Podle veřejné a administrátorské části Webového portálu jsou tyto požadavky rozděleny do dvou balíčků. Přehled požadavků je možné vidět na obrázku 1.1.



Obrázek 1.1: Přehled funkčních požadavků rozdělených do balíčků v nástroji Enterprise Architect

1.2.1.1 Veřejná část

F01 Zobrazení informací o konferenci
Webový portál musí zobrazovat všechny organizační informace týkající se konference. Nejdůležitější z informací, jako jsou název konference, místo a čas konání, se musí nacházet v globální hlavičce každé veřejné stránky.

F02 Registrování
Uživatel musí mít možnost registrovat se jako účastník konference. Při registraci se vyplňují osobní informace, jako jsou jméno a příjmení, e-mailová adresa, příslušnost k určité univerzitě, způsob účasti, poštovní adresa pro zaslání faktury, zda je účastník studentem a dodatečné informace pro organizátory. Volitelně je při registraci možné vložit příspěvek do konference, u kterého se uvádí název příspěvku, jména spoluautorů, příslušnost spoluautorů k určité univerzitě a samotný abstrakt příspěvku. V registračním formuláři musí být jasně vidět, jaká pole jsou povinná.

F03 Zobrazení seznamu účastníků a příspěvků
Webový portál musí zobrazovat seznam všech účastníků a jejich příspěvků. Ve veřejném přehledu nesmí být vidět citlivé informace účastníků, jako je e-mailová adresa. V seznamu musí být možné vyhledávat. Výsledky vyhledávání musí být možné seřadit podle příjmení účastníků nebo titulku příspěvků.

F04 Zobrazení denního programu
Webový portál musí zobrazovat denní program konference. Každý den musí mít uveden datum a čas konání. Jednotlivé položky dne musí mít časový rozsah a obsah. Položkou může být přednáška nebo doplňující akce. U přednášky je zobrazeno jméno přednášejícího účastníka a titulek jeho příspěvku, přes který se lze navigovat na přehled všech příspěvků, ve kterém se ten konkrétní příspěvek zvýrazní. U doplňující akce je její krátký popis.

F05 Zobrazení navigace
Webový portál musí poskytovat informace o místě konání konference a způsobu dopravy na něj. Textový popis musí být doplněn interaktivní mapou ukazující na místo konání.

F06 Zobrazení galerie s fotografiemi
Webový portál musí obsahovat galerii s fotografiemi z konference. Jednotlivé fotografie musí být možno zvětšit a následně ve zvětšeném režimu procházet ostatní fotografie. Fotografie musí jít stáhnout v plné velikosti.

1.2.1.2 Administrátorská část

F07 Přihlášení do administrátorské části
Webový portál musí administrátorům umožnit přihlásit se do administrátorské části. Přihlášení je pomocí přihlašovacího jména a hesla. Po přihlášení zůstane administrátor přihlášen a má možnost se prostřednictvím položky administrátorské navigace odhlásit.

F08 Úprava organizačních informací konference
Veškeré informace týkající se konference musí jít upravit prostřednictvím administrátorské části Webového portálu. Do těchto informací se počítají informace v globální hlavičce a obsah všech veřejných stránek. Zároveň musí být možnost vložit do obsahu stránek fotografie a interaktivní mapu.

F09 Správa účastníků a příspěvků
Webový portál musí administrátorům zobrazit seznam účastníků a jejich příspěvků. V seznamu musí jít vyhledávat a řadit. Každého účastníka a jeho příspěvek musí být možno upravit, případně smazat. Administrátor musí mít rovněž možnost přidat nového účastníka.

F10 Správa administrátorů
Webový portál musí administrátorům zobrazit seznam účtů administrátorů. V seznamu musí jít vyhledávat, řadit a jednotlivé účty upravovat. Každý účet, s výjimkou toho, ve kterém je administrátor přihlášený, musí jít odstranit. Administrátor musí mít zároveň možnost přidat nový administrátorský účet.

F11 Správa denního programu
Webový portál musí poskytovat editor denního programu. V editoru musí být možnost volby začátku konference, přidávání dní s přednáškami a dalšími akcemi a přidávání předsedů zasedajících jednotlivým sekcím dní.

F12 Správa fotografií
Webový portál musí umožňovat nahrávání a ukládání fotografií. Fotografie musí jít nahrávat po jedné, ale i ve větším množství. U každé fotografie musí být možnost nastavit její titulek a fotografii smazat.

F13 Notifikace
Administrátor musí být po každém provedení změn upozorněn o výsledku jeho akce. Upozornění musí být také před provedením nezvratitelné akce, jako je odstranění dat.

1.2.2 Nefunkční požadavky

Nefunkční požadavky se netýkají chování systému, pouze určují omezení kladená na systém. Takováto omezení se obvykle týkají použitelnosti, spolehlivosti, výkonu a dodržení standardů při implementaci systému. I v případě nesplnění nefunkčních požadavků bude systém fungovat. [7]

N01 Webové rozhraní
Webový portál bude mít uživatelské rozhraní, které bude přístupné prostřednictvím webového prohlížeče. Uživatel tak bude schopen používat všechny funkce Webového portálu bez nutnosti stahování nebo instalace jakéhokoliv softwaru na svém počítači.

N02 Responzivní design
Webový portál bude navržen tak, aby se přizpůsoboval různým velikostem obrazovek a zařízení, na kterých je zobrazen. To znamená změnu rozvržení a velikosti prvků na stránce, aby byla zajištěna optimální čitelnost a použitelnost na všech zařízeních.

N03 Úprava informací
Pro úpravu organizačních informací a obsahu stránek musí být použit editor umožňující psaní strukturovaného textu s nadpisy, seznamy, odkazy, zvýrazněným textem a vloženými fotografiemi a interaktivní mapou.

N04 Ukládání dat
Veškerá data, s výjimkou fotografií, musí být ukládána do databáze. Fotografie jsou ukládány na disk serveru a do databáze jsou ukládány pouze odkazy na ně.

N05 Nasazení
Webový portál a databáze musí jít nasadit jako Docker kontejnery.

1.2.3 Aktéři

Aktéři jsou skupiny uživatelů, kteří systém používají stejným způsobem [8]. Webový portál obsahuje dva aktéry – uživatel a administrátor.

Uživatel
Uživatel je nepřihlášená osoba, která navštěvuje Webový portál za účelem získání informací o konferenci a registrování se.

Administrátor
Administrátor je osoba, která je přihlášená a má přístup do administrátorské části Webového portálu. Spravuje informace o konferenci a účastnících, vytváří program a nahrává fotografie.

1.2.4 Případy užití

Případy užití jsou detailnější specifikací požadavků. Poskytují popis jednotlivých kroků, které bude uživatel při používání systému, za účelem dosažení svého cíle, vykonávat. [8]

Z detailního popisu jsou vynechány jednoduché případy užití, které se většinou týkají pouhého zobrazení informací. Pořadí případů užití je zanecháno tak, aby odpovídalo funkčním požadavkům. Stejně tak, jak jsou funkční požadavky rozděleny na veřejnou a administrátorskou část, jsou rozděleny i případy užití. *V* značí případ užití veřejné a *A* případ užití administrátorské části.

Případy užití bez detailního popisu:

- V01 Zobrazení informací o konferenci
- V04 Zobrazení účastníků a příspěvků
- V06 Zobrazení programu konference
- V07 Zobrazení detailu příspěvku z programu
- V08 Zobrazení informací o dopravě na místo konference
- V09 Zobrazení galerie

- A15 Detailní seznam účastníků a příspěvků
- A16 Přidání účastníka a příspěvku
- A20 Seznam administrátorů
- A21 Přidání administrátora
- A22 Úprava administrátora
- A23 Odstranění administrátora
- A25 Nahrání fotografií
- A25 Úprava informací fotografie

1.2.4.1 V02 Registrace do konference s příspěvkem

Předpoklad: Uživatel se nachází na veřejné stránce *Register*.

1. V úvodu stránky *Register* si uživatel přečte informace ohledně registrace a pokračuje k níže umístěnému registračnímu formuláři.
2. Uživatel vyplní část formuláře požadující osobní informace. Osobní informace jsou jméno a příjmení, e-mailová adresa, příslušnost k určité univerzitě, způsob účasti, poštovní adresa pro zaslání faktury, zda je studentem a dodatečné informace pro organizátory.
3. Uživatel vyplní část formuláře týkající se jeho příspěvku do konference. Informace o příspěvku jsou název příspěvku, jména spoluautorů, příslušnost spoluautorů k určité univerzitě a samotný abstrakt příspěvku.
4. Uživatel potvrdí registraci odesláním formuláře.
5. Systém v případě správně vyplněného formuláře přidá uživatele, včetně jeho příspěvku, do seznamu všech účastníků.

1.2.4.2 V03 Registrace do konference bez příspěvku

Předpoklad: Uživatel se nachází na veřejné stránce *Register*.

1. V úvodu stránky *Register* si uživatel přečte informace ohledně registrace a pokračuje k níže umístěnému registračnímu formuláři.
2. Uživatel vyplní část formuláře požadující osobní informace. Osobní informace jsou jméno a příjmení, e-mailová adresa, příslušnost k určité univerzitě, způsob účasti, poštovní adresa pro zaslání faktury, zda je studentem a dodatečné informace pro organizátory.

3. Uživatel zruší implicitně vybranou položku *Contribution* indikující zájem o přidání příspěvku.
4. Uživatel potvrdí registraci odesláním formuláře.
5. Systém v případě správně vyplněného formuláře přidá uživatele do seznamu všech účastníků.

1.2.4.3 V05 Vyhledání účastníka a příspěvku

Předpoklad: Uživatel se nachází na veřejné stránce *Participants & Abstracts*.

1. Uživatel do vyhledávacího pole zadá text, podle kterého chce vyhledávat. Vyhledávat lze podle všech informací, které jsou v seznamu účastníků a příspěvků zobrazeny.
2. Systém zobrazí výsledky relevantní k hledanému výrazu.
3. Uživatel zvolí způsob řazení výsledků. Řazení je možné podle příjmení účastníka a titulku příspěvku, a to vzestupně nebo sestupně.
4. Systém seřadí výsledky podle volby uživatele a zobrazí je.

1.2.4.4 V10 Zvětšení a uložení fotografie

Předpoklad: Uživatel se nachází na veřejné stránce *Gallery*.

1. Uživatel klikne na libovolnou fotografii.
2. Systém otevře vyskakovací okno se zvětšenou fotografií, které uživateli umožňuje přecházet mezi jednotlivými fotografiemi.
3. Uživatel klikne na tlačítko stáhnout.
4. Systém spustí akci stažení a uložení fotografie v plné velikosti na zařízení uživatele.

1.2.4.5 A11 Přihlášení a odhlášení administrátora

Předpoklad: Uživatel se nachází na libovolné veřejné stránce.

1. Uživatel se odkazem v patičce libovolné veřejné stránky dostane na přihlašovací stránku *Login*.
2. Systém zobrazí přihlašovací formulář.
3. Uživatel zadá své uživatelské jméno a heslo a potvrdí přihlášení.
4. Po úspěšném přihlášení má uživatel roli administrátora a systém ho přesměruje do administrátorské části.

5. Administrátor vykoná své změny a z administrátorské části se odhlásí.

6. Administrátor po odhlášení ztrácí roli administrátora a má opět pouze roli uživatele. Systém ho přesměruje na domovskou stránku.

1.2.4.6 A12 Úprava organizačních informací v hlavičce

Předpoklad: Administrátor je přihlášen a nachází se na hlavní administrátorské stránce *General*.

1. Systém zobrazí náhled hlavičky s aktuálními informacemi a formulář pro jejich změnu.

2. Administrátor informace ve formuláři změní.

3. Systém aktualizuje informace v náhledu hlavičky.

4. Administrátor změny uloží.

5. Systém zobrazí upozornění o výsledku uložení změn a v případě úspěchu hlavičku překreslí.

1.2.4.7 A13 Úprava obsahu stránek

Předpoklad: Administrátor je přihlášen a nachází se na administrátorské stránce *Pages*.

1. Systém zobrazí editor pro každou veřejnou stránku s jejím obsahem.

2. Administrátor změní obsah libovolné stránky a změny uloží.

3. Systém zobrazí upozornění o výsledku uložení změn a v případě úspěchu překreslí změněnou stránku.

1.2.4.8 A14 Vložení interaktivní mapy

Předpoklad: Administrátor je přihlášen a nachází se na administrátorské stránce *Pages*.

1. Systém zobrazí editor pro každou veřejnou stránku s jejím obsahem.

2. Administrátor vybere libovolnou stránku a v editoru otevře možnost vložení interaktivní mapy.

3. Administrátor vloží odkaz z libovolné on-line mapy, vybere velikost a změny uloží.

4. Systém zobrazí upozornění o výsledku uložení změn a v případě úspěchu překreslí změněnou stránku s přidanou interaktivní mapu.

1.2.4.9 A17 Vyhledání účastníka a příspěvku

Předpoklad: Administrátor je přihlášen a nachází se na administrátorské stránce *Participants & Abstracts*.

1. Systém zobrazí tabulku se všemi účastníky.
2. Administrátor zadá text, podle kterého chce vyhledávat.
3. Systém zobrazí výsledky relevantní k hledanému výrazu.
4. Administrátor klikne na hlavičku sloupce, podle kterého chce vzestupně nebo sestupně řadit.
5. Systém účastníky v tabulce seřadí a zobrazí.

1.2.4.10 A18 Odstranění účastníka

Předpoklad: Administrátor je přihlášen a nachází se na administrátorské stránce *Participants & Abstracts*.

1. Systém zobrazí tabulku se všemi účastníky a jejich příspěvky.
2. Administrátor vybere účastníka, kterého chce odstranit, a klikne na tlačítko smazat.
3. Systém zobrazí potvrzovací okno ve kterém je popis odstraňovaného účastníka.
4. Administrátor potvrdí odstranění.
5. Systém zobrazí upozornění o výsledku odstranění účastníka.
6. Systém v případě úspěchu účastníka, včetně jeho příspěvku, ze seznamu všech účastníků odstraní.

1.2.4.11 A19 Odstranění příspěvku účastníka

Předpoklad: Administrátor je přihlášen a nachází se na administrátorské stránce *Participants & Abstracts*. Vybraný účastník má příspěvek.

1. Systém zobrazí tabulku se všemi účastníky a jejich příspěvky.
2. Administrátor vybere účastníka, jehož příspěvek chce odstranit. Zobrazí si jeho detail, kde zruší vybranou položku *Contribution* indikující přidaný příspěvek.
3. Systém zobrazí upozornění, že účastník měl příspěvek a uložením dojde ke smazání příspěvku.

4. Administrátor potvrdí uložení změn.
5. Systém zobrazí upozornění o výsledku odstranění příspěvku.
6. Systém v případě úspěchu odstraní příspěvek a u účastníka v tabulce indikuje, že žádný příspěvek nemá.

1.2.4.12 A24 Vytvoření denního programu

Předpoklad: Administrátor je přihlášen a nachází se na administrátorské stránce *Programme*.

1. Systém zobrazí položku pro výběr data začátku konference.
2. Administrátor vybere datum začátku konference.
3. Systém zobrazí možnost přidání dne konference.
4. Administrátor přidá den konference, zvolí čas jeho začátku a vyplní doplňující informace k tomuto dni.
5. Systém zobrazí možnost přidat položky do dne.
6. Administrátor přidá předsedu, přednášku nebo doplňující akci a uloží změněný program.
7. Systém zobrazí upozornění o výsledku uložení změn programu a v případě úspěchu překreslí změněný program.

1.2.5 Pokrytí funkčních požadavků

V tabulkách 1.1 a 1.2 lze vidět, jak případy užití pokrývají funkční požadavky. První tabulka zobrazuje pokrytí požadavků veřejné části, zatímco druhá zobrazuje pokrytí požadavků části administrátorské.

1.3 Analýza problémové domény

Tato sekce se zabývá analýzou problémové domény. Doména představuje oblast, na kterou se aplikace zaměřuje [9]. Doménový model je objektový model zachycující chování a data v doméně [10].

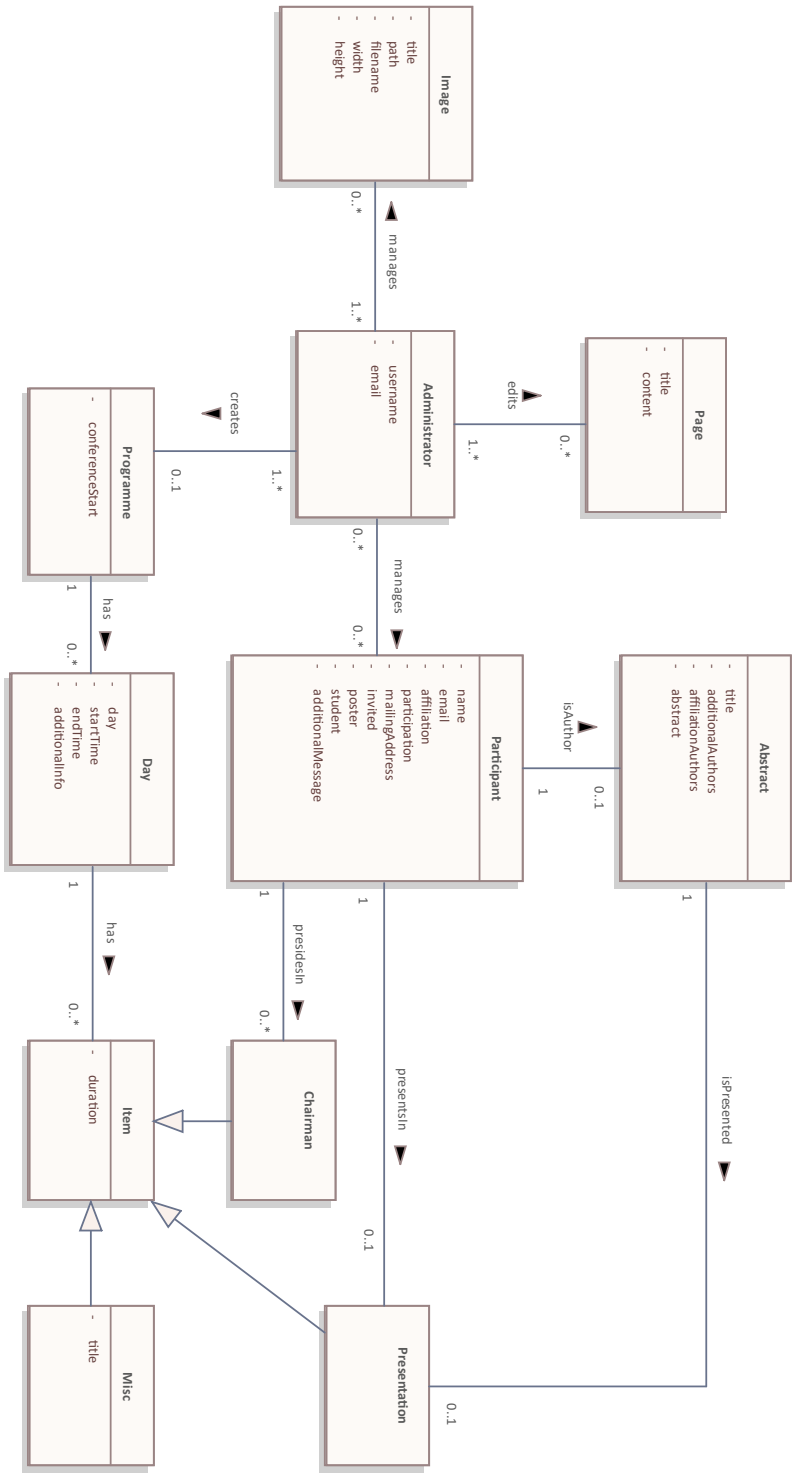
Cílem analýzy je identifikovat relevantní koncepty, vztahy a omezení domény, které nám samotnou doménu pomohou lépe pochopit a umožní spolehlivě navrhnout infrastrukturu. Výsledkem analýzy je doménový model (viz. obrázek 1.2), reprezentovaný diagramem tříd v jazyce UML (Unified Modeling Language), který může sloužit jako základ při tvorbě databázového modelu [11].

Tabulka 1.1: Pokrytí funkčních požadavků veřejné části případy užití

	F01	F02	F03	F04	F05	F06
V01	✓					
V02		✓				
V03		✓				
V04			✓			
V05			✓			
V06				✓		
V07				✓		
V08					✓	
V09						✓
V10						✓

Tabulka 1.2: Pokrytí funkčních požadavků administrátorské části případy užití

	F07	F08	F09	F10	F11	F12	F13
A11	✓						
A12		✓					✓
A13		✓					✓
A14		✓					✓
A15			✓				
A16			✓				✓
A17			✓				
A18			✓				✓
A19			✓				✓
A20				✓			
A21				✓			✓
A22				✓			✓
A23				✓			✓
A24					✓		✓
A25						✓	✓
A26						✓	✓



Obrázek 1.2: Doménový model v nástroji Enterprise Architect

Kapitola 2

Návrh prototypu

V této kapitole dojde k návrhu prototypu Webového portálu. Před volbou technologií je důležité zvolit přístup při tvorbě samotné aplikace. Poté následuje volba technologií, návrh uživatelského rozhraní a volba architektury API (Application Programming Interface).

2.1 MPA vs. SPA

V současné době se při tvorbě full-stack webových aplikací používají dva hlavní přístupy – MPA (Multi-Page Application) a SPA (Single-Page Application). Hlavním rozdílem mezi těmito přístupy je způsob, kterým aplikace reaguje na akce uživatele a vykresluje změny, a také místo, kde se vykonává většina aplikační logiky. [12]

2.1.1 MPA

V případě MPA způsobí většina interakcí uživatele kompletní načtení nové stránky. Webová aplikace se tak skládá z více HTML stránek, které server vykresluje a posílá klientovi jako odpověď na jeho požadavek. Většina logiky aplikace se nachází na straně serveru. [12]

Mezi výhody MPA patří rychlejší prvotní načtení aplikace, lepší SEO (Search Engine Optimization) výsledky, podpora starších prohlížečů, jelikož není nutné podporovat JavaScript na straně klienta, a snadnější implementace bezpečnostních operací z důvodu samostatného posílání požadavků pro každou stránku. [12]

Hlavními nevýhodami jsou načítání celé stránky po každé akci, vyšší náročnost na server z důvodu zpracování větších a častějších požadavků, složitější implementace dynamických funkcionalit a složitější vývoj, údržba a škálování. [12]

MPA jsou příkladem klasických webových aplikací používaných od počátků internetu. Nejvíce se hodí pro stránky s velkým množstvím obsahu, jako jsou zpravodajské stránky a stránky internetových obchodů. [12]

2.1.2 SPA

U SPA se celá webová aplikace načítá pouze jednou, a to při prvotním načtení. Následné akce jsou prováděny dynamicky na straně klienta díky technologii AJAX (Asynchronous JavaScript and XML), která umožňuje obnovení jen té části aplikace, ve které došlo ke změně. [12]

Zatímco u MPA má každá stránka svůj HTML dokument, SPA staví na modulárním designu, ve kterém jsou stránky sestavovány z různých komponent. Každá komponenta se stará o svou funkcionalitu a může být upravována nebo nahrazena bez vlivu na zbytek aplikace. [12]

Mezi výhody SPA patří nutnost pouze prvotního načtení celé stránky, snadnější škálování a údržba díky nezávislému použití technologií pro klientskou a serverovou část, modulární design a minimalizace počtu požadavků na server. [12]

Hlavními nevýhodami jsou potenciálně pomalejší prvotní načtení celé aplikace, negativně ovlivněné SEO výsledky z důvodu dynamického načítání obsahu, nutnost podpory JavaScriptu na straně klienta a vyšší riziko odhalení citlivých dat, jelikož aplikace běží u uživatele. [12]

SPA se využívá zejména v případě sociálních sítí, navigačních aplikací a obecně stránek s velkým množstvím dynamického obsahu. V posledních letech se rozšířili i mezi běžné webové aplikace. [12]

2.1.3 Závěr

Vzhledem k potřebě tvorby administrátorské části obsahující dynamické prvky (správa fotografií, vytváření programu, editor obsahu, ...), je vhodnější volbou SPA. Tyto funkcionality by bylo možné implementovat i v rámci MPA, ale celý proces by byl ve srovnání s SPA náročnější. Volba SPA tak umožní rychlejší vývoj, v budoucnosti případné snadnější rozšiřování a také poskytne uživatelům responzivnější zážitek bez prodlev a načítání.

Pro veřejnou část Webového portálu by mohl být problém se SEO výsledky a JavaScript podporou v prohlížeči. To ale v dnešní době řeší meta-frameworky (viz. sekce 2.2.2) pomocí různých způsobů vykreslování stránek, jako jsou SSG (Static-Site Generating) a SSR (Server-Side Rendering). [13]

2.2 Volba technologií

Z důvodu volby SPA jako přístupu při tvorbě full-stack webové aplikace se v první podsekci vybere nejvhodnější framework pro klientskou část. Pro tvorbu serverové části bude výběr pokračovat meta-frameworky, které běžné frameworky rozšiřují. Následně se zvolí knihovna komponent a databáze.

2.2.1 Framework

Vývojová platforma neboli *framework* je sada nástrojů usnadňující vývoj aplikace poskytnutím hotových řešení k problémům, se kterými se při vývoji běžně setkává. Rovněž definováním konvencí pro psaní a strukturování kódu standardizují způsob, jakým se kód píše. *Frontend framework* (dále framework) je framework specializovaný na tvorbu klientské části webových aplikací. [14]

Dle [15] jsou za rok 2022 nejpoužívanějšími frameworky React.js, Angular a Vue.js. Všechny tyto framework staví na jazyce JavaScript, případně jeho rozšíření TypeScript. Nejvhodnějším frameworkem z těchto tří je React.js.

React.js je deklarativní, komponentově orientovaná knihovna pro tvorbu uživatelských rozhraní, většinou už ale označovaná za framework. Uživatelské rozhraní se skládá z jednotlivých komponent, jejichž propojení umožní vznik komplexní aplikace. Pro manipulaci webové stránky používá virtuální DOM (Document Object Model), umožňující efektivní změny v uživatelském rozhraní. Je relativně jednoduchý na používání a velmi flexibilní. Jeho rozsáhlá a aktivní komunita nabízí řešení k téměř každému problému, na který se během vývoje narazí. [16]

React.js navíc umožňuje volbu mezi jazyky JavaScript a TypeScript. Zvolen byl TypeScript, který JavaScript rozšiřuje a přináší statické typování, kontrolu chyb v době kompilace a mnoho dalších zlepšení. Díky rozšířením v editorech také nabízí kvalitnější našeptávání, které přispívá k efektivnějšímu a příjemnějšímu vývoji. [17]

2.2.2 Meta-framework

Zvolený framework React.js obsahuje nástroje pouze k tvorbě SPA. Při tvorbě rozsáhlejších aplikací, které potřebují i serverovou část, kombinaci způsobů vykreslování stránek a další pokročilé funkcionality, stojí za to zhodnotit, zda by nebylo vhodné využít nějaký meta-framework. Alternativním řešením k meta-frameworku by bylo vytvořit samostatnou aplikaci pouze pro serverovou část a propojit ji s klientskou částí pomocí nějakého API. Stále by ale zůstaly problémy s různými způsoby vykreslování stránek a vlastní implementací pokročilých funkcionalit, jako je optimalizace obrázků.

Meta-frameworky jsou v jednoduchosti frameworky frameworků. Spojují víc menších frameworků, knihoven a balíčků, a vytvářejí jeden funkční celek, který poskytuje nástroje pro tvorbu komplexních webových aplikací. [18]

Mezi nejpopulárnější meta-frameworky rozšiřující React.js v současnosti patří Next.js, Gatsby a Remix. [19]

Gatsby se specializuje na vykreslování stránek, jejichž obsah se téměř nemění (SSG). Takové stránky jsou obvykle blogy a portfolia. Nově podporuje i SSR, ale tato funkcionalita není úplně odladěná. [20]

Remix naopak SSG nepodporuje vůbec a zaměřuje se na SSR. Je to relativně nový meta-framework a jeho adopce stále není tak velká. [20]

Next.js je momentálně nejoblíbenější a nejpoužívanější meta-framework [19]. Podporuje souběžně SSG, SSR a navíc i ISR (Incremental Static Regeneration), díky čemuž se dají na vyžádání překreslovat konkrétní statické stránky. Dalšími rozšířeními jsou směrování na základě adresářové struktury, optimalizace obrázků, možnost v jednom projektu vytvořit klientskou i serverovou část (realizovanou jako API) a mnoho dalších. [18][20]

Volbou Next.js se vyřeší potřeba samostatné serverové části, která bude nahrazena API koncovými body. SSG se využije pro vykreslování veřejných stránek, čímž se vyřeší problém se SEO výsledky a přesto bude v kombinaci s ISR umožněno obsah kdykoliv upravovat. Automatická optimalizace fotografií usnadní nahrávání fotografií a tvorbu galerie. Další přiložená vylepšení oproti „holému“ SPA frameworku navíc výrazně zpříjemní a urychlí vývoj.

2.2.3 Knihovna komponent

Dalším významným rozhodnutím je volba knihovny komponent. Knihovna komponent je soubor nejčastěji používaných komponent, jako jsou formuláře, tlačítka a tabulky, které lze v projektu využít a sestavit z nich výsledné uživatelské rozhraní. Přispívají k rychlosti a konzistenci při vývoji a předcházejí potenciálním problémům, které by při vlastnoručním vytváření jednotlivých komponent mohly vzniknout. [21]

V nefunkčním požadavku *N03 Úprava informací* je pro úpravu organizačních informací požadován editor strukturovaného textu. Takový editor je možno zvolit jako nezávislou knihovnu, ale vhodnější je použití knihovny komponent, která ho má sama zakomponovaný a upravený tak, aby odpovídal vzhledu a chování ostatních komponent.

Po průzkumu několika knihoven komponent se jako vhodná jeví pouze knihovna Mantine [22]. Obsahuje požadovaný editor a nabízí velké množství různých komponent a funkcí usnadňujících psaní aplikační logiky. Její autor také vytvořil sesterskou knihovnu Mantine UI [23], ve které jsou již hotové větší části uživatelského rozhraní, jako je například responzivní navigační lišta a přihlašovací formulář.

2.2.4 Databáze a přístup k datům

Nefunkční požadavek *N04 Ukládání dat* požaduje ukládání dat do databáze, což současně řeší problém s ukládáním a přístupem k datům popsaným v sekci 1.1.3.3.

V současné době existují dva populární typy databází – SQL (Structured Query Language) a NoSQL. „*Relační (SQL) databáze ukládají strukturovaná data, která odpovídají předem definovanému schématu. Databáze s dokumentovým modelem (nebo NoSQL) ukládají dokumenty, které mohou obsahovat nestrukturovaná data bez schématu.*“ [24] (překlad vlastní)

Vzhledem k pevně dané struktuře dat a vazbám mezi entitami je vhodnější typ databáze relační. Relačních databází existuje mnoho a pro požadavky této práce na rozdílech mezi nimi nezáleží. Zvolena bude aktuálně druhá nejpoužívanější a zároveň první nejoblíbenější databáze, a to PostgreSQL [25]. K volbě také přispěly osobní zkušenosti s touto databází vedoucí k rychlejšímu vývoji.

K datům uloženým v databázi lze následně ze serverové části přistupovat prostřednictvím databázového klienta a SQL dotazů. Lepším způsobem je ale použití ORM (Object-Relational Mapping) knihovny, která poskytuje abstrakci nad komunikací s databází a umožňuje snadné dotazování a ukládání dat. Takových knihoven existuje mnoho a vzhledem k tomu, že nejsou žádné specifické požadavky na konkrétní funkcionality, na volbě opět nezáleží. Zvolena byla jedna z nejoblíbenějších ORM knihoven knihovna Prisma. [26]

2.3 Uživatelské rozhraní

Tato sekce se zabývá návrhem uživatelského rozhraní pomocí tzv. *wireframes*. Návrh navazuje na analýzu provedenou v kapitole 1, zejména pak na analýzu požadavků v sekci 1.2, a je opět rozdělen na veřejnou a administrátorskou část. Wireframes pro veřejnou část lze vidět v příloze B a pro administrátorskou část v příloze C.

2.3.1 Veřejná část

Všechny stránky veřejné části mají společnou hlavičku obsahující název, datum a místo konání. Stránky *Home* a *Travel* se strukturou oproti původnímu Webovému portálu nezměnily, jelikož pouze zobrazují informace vytvářené administrátory. Jejich návrhy tedy budou vynechány.

Na stránce *Register* došlo ke změně struktury registračního formuláře (obrázek přílohy B.1). Stránky *Participants* a *Abstracts* prošly největšími změnami (obrázek přílohy B.2) – byly spojeny do nové stránky *Participants & Abstracts* a zároveň bylo přidáno vyhledávání a řazení. Zcela novou stránkou je stránka *Gallery* (obrázek přílohy B.3).

2.3.2 Administrátorská část

Uživatelské rozhraní administrátorské části prošlo kompletní přestavbou. Společná hlavička, která je ve veřejné části, byla odstraněna za účelem získání více prostoru. Navigace mezi stránkami administrátorské části, ve formě velkých dlaždic na hlavní stránce, byla nahrazena navigací v postranním panelu. V postranním panelu je navíc možnost odhlásit se.

Stránka *General* (obrázek přílohy C.1) slouží k úpravě nejdůležitějších informací v hlavičce veřejné části a je domovskou administrátorskou stránkou. Obsahuje náhled, jak bude hlavička s aktuálně zadanými informacemi vypadat.

Stránky *Administrators* (obrázek přílohy C.2) a *Participants & Abstracts* (obrázek přílohy C.3) se skládají z formulářů pro vytváření nových záznamů a z tabulek, které již existující záznamy zobrazují. V tabulce lze vyhledávat a dle jednotlivých sloupců řadit. Kliknutím na konkrétní záznam se otevře okno s možností úpravy.

Stránka *Programme* (obrázek přílohy C.4) slouží k sestavování denního programu konference. Na vrchu stránky je volba data začátku konference a uložení vytvořeného programu. Pod nimi se nacházejí jednotlivé dny, ke kterým lze vkládat doplňující informace a vytvářet v nich položky programu. Za posledním dnem je možnost přidat další.

Stránka *Pages* (obrázek přílohy C.5) obsahuje editory s obsahem každé veřejné stránky. Zároveň zde lze upravit titulky stránek, které se uživateli zobrazují v záložce prohlížeče.

Stránka *Gallery* (obrázek přílohy C.6) umožňuje nahrání a správu fotografií. Kliknutím na libovolnou fotografii se zobrazí její název, titulek a možnost ji smazat.

2.4 API

Přestože byl zvolen meta-framework umožňující tvorbu klientské i serverové části v jednom projektu, obě vrstvy mezi sebou musí nějakým způsobem komunikovat. V této sekci se popíše pojem API a vybere se vhodná architektura pro jeho realizaci.

API je rozhraní aplikace realizované za pomoci specifikací a protokolů a popisuje význam a strukturu zpráv posílaných mezi dvěmi softwarovými komponenty. Bude tedy sloužit jako most pro komunikaci mezi klientskou a serverovou částí. V současné době patří mezi nejpopulárnější API architektury REST, RPC a GraphQL. [27]

REST (Representational State Transfer) je architektonický styl pro návrh webových API. Pro komunikaci se typicky využívá HTTP (Hypertext Transfer Protocol). Staví na použití unikátních identifikátorů pro identifikaci a manipulaci zdrojů v systému. REST API jsou bezstavová – server si neukládá stav o relaci a v každém požadavku musí být všechny informace potřebné pro jeho

provedení. Tato architektura je populární pro svou jednoduchost, flexibilitu a škálovatelnost, a je vhodná pro aplikace, které nepotřebují flexibilitu v samotných požadavcích. [27]

RPC (Remote Procedure Call) je protokol umožňující vzdálené spouštění funkcí. Funguje tak, že klient zavolá (*invokuje*) funkci na serveru a počká na její dokončení, po kterém server klientovi pošle výsledek. Pro komunikaci se obvykle používá HTTP protokol. Tato architektura požaduje, aby se klient a server shodly na typech a procedurách, které mohou být zavolány, což ho činí méně flexibilní. Je vhodný pro propojování mikroslužeb. [27]

GraphQL je dotazovací jazyk a běhové prostředí pro API. Klient neposílá požadavky na konkrétní koncové body, ale požadavkem přímo určuje, jaká data chce. Server v odpovědi pošle pouze požadovaná data. Výhodou je minimalizace přenosu nechtěných dat a dodržování přesně daných typů. Požadavky jsou obvykle velké, protože se specifikuje přesně to, co klient chce. Vhodné pro grafová data. Téměř nemožné ukládání požadavků do mezipaměti. [27]

Z důvodu jednoduchosti a univerzálnosti byla zvolena REST architektura. RPC a GraphQL mají ve srovnání složitější architekturu a jsou obtížnější na implementaci.

Kapitola 3

Realizace prototypu

V této kapitole se přiblíží samotný vývoj prototypu Webového portálu, při kterém se využívá technologií zvolených během návrhu. V první sekci proběhne krátké seznámení s použitými nástroji, v druhé pak ukázka struktury projektu. Další sekce přiblíží stěžejní části Webového portálu s problémy, na které se při vývoji narazilo. Ukázky kódu jsou za účelem ilustrace problémů zjednodušeny a nemusí odpovídat skutečně použitému kódu.

3.1 Použité nástroje

Během vývoje byly použity různé nástroje pro jeho usnadnění a současné dodržení určitých pravidel a struktury při psaní kódu.

3.1.1 Správce balíčků

K instalaci a správě knihoven a frameworků v podobě balíčků byl použit nástroj NPM (Node Package Manager). Pomocí NPM příkazu *npx* lze spouštět balíčky pro vytváření projektů a spouštění lokálního serveru pro vývoj. [28]

3.1.2 Vytvoření projektu

Pro vytvoření projektu byl použit terminálový nástroj Create Next App od autorů Next.js. Příkazem *npx create-next-app* se spustí interaktivní průvodce instalace, během kterého se sestaví a nakonfiguruje Next.js projekt podle vybrané šablony. Pro tento projekt byla zvolena šablona *Next.js with TypeScript*, která obsahuje přednastavený TypeScript a jeho závislosti. [29]

3.1.3 Kontrola a formátování kódu

Pro dodržení jednotné struktury a pravidel při psaní kódu byly použity nástroje Prettier [30] a ESLint [31]. ESLint nabízí velké množství veřejných kon-

figurací používaných ve velkých projektech, které definují pravidla psaní JavaScript a TypeScript kódu. Konkrétní zvolenou konfigurací byla vzhledem k použitým knihovnám kombinace konfigurací Next a Mantine. Prettier slouží k formátování kódu a společnou konfigurací s ESLint umožňuje automatické formátování podle nastavených pravidel.

3.1.4 Verzování

Pro verzování projektu byl použit nástroj Git a platforma GitHub. Git umožňuje zaznamenávání historie změn v kódu a GitHub nabízí platformu pro ukládání takto verzovaných projektů (tzv. repositářů). Často lze u hostovacích služeb využít propojení s GitHub repositářem pro snadné nasazení s automatickým hlídáním změn. [32]

3.1.5 Responzivita uživatelského rozhraní

Nefunkční požadavek N02 požaduje responzivní design uživatelského rozhraní. Ke snadnému testování responzivity uživatelského rozhraní byl použit nástroj Polypane. Polypane zobrazuje konkrétní stránku v několika různě velkých oknech simulovaného prohlížeče a nabízí předpřipravené skupiny nejpoužívanějších velikostí obrazovek. To umožňuje testování responzivity bez nutnosti neustálých změn velikosti okna prohlížeče a potřeby dalších zařízení. [33]

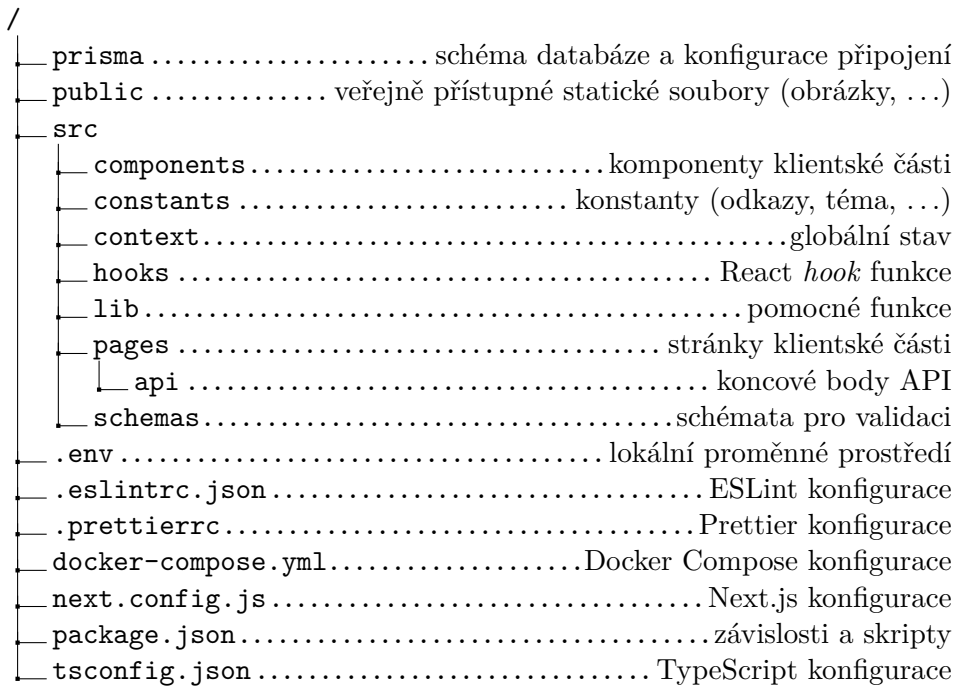
3.2 Struktura projektu

Struktura projektu je rozšířením typické struktury Next.js projektu s jazykem TypeScript. Navíc jsou zde konfigurační soubory pro nástroje Docker Compose (viz. sekce 4.2), Prettier a ESLint (viz. sekce 3.1.3). Struktura složky *pages* definuje navigační strukturu klientské části a struktura složky *api* definuje navigační strukturu serverové části a jejích koncových bodů [29].

3.3 Autentizace

Pro vyřešení problému přístupu do administrátorské části (viz. sekce 1.1.3.4) bylo nutné implementovat autentizaci více nezávislých administrátorů. Použita byla knihovna NextAuth, která poskytuje flexibilní API pro autentizaci na straně klienta i serveru a nabízí propojení přihlašovacích služeb třetích stran a tvorbu vlastního ověření. [34]

Administrátoři se přihlašují uživatelským jménem a heslem. Od tohoto způsobu přihlašování se autoři knihovny NextAuth snaží z bezpečnostních důvodů odradit, proto není dostatečně zdokumentován a má omezenou funkcionalitu. Relaci vzniklou tímto způsobem přihlášení není možné ukládat do databáze, ale pouze jako JWT (JSON Web Token) do prohlížeče uživatele. Vlastní logika ověření zadaných přihlašovacích údajů je pak v podobě funkce



Obrázek 3.1: Struktura projektu

předané vlastnímu poskytovateli přihlášení. Návratová hodnota této funkce je objekt uložený do cookies v podobě JWT sloužící pro samotnou autentizaci. [34]

První zamýšlený a nejjednodušší způsob kontroly, zda je uživatel přihlášen, měl být za použití tzv. *middleware* funkce (viz. ukázka kódu 3.1). Kód v *middleware* funkci *withAuth* proběhne pokaždé, když přijde požadavek na adresu odpovídající cestě definované v konfiguraci *config* [29]. V případě, že uživatel přihlášen není, je přesměrován na adresu */login*.

```
1 export default withAuth({
2   callbacks: { authorized: ({ token }) => !!token },
3   pages: { signIn: '/login' },
4 });
5
6 export const config = { matcher: '/admin/:path*' };
```

Ukázka kódu 3.1: Autentizace pomocí middleware

Tento způsob ve vývojovém prostředí fungoval bez jediného problému. V případě produkčního prostředí ale přestalo fungovat přesměrování na administrátorskou stránku po té, co se uživatel přihlásil. Přesměrování zafungovalo až po úplném přenačtení stránky, což není v případě SPA přijatelné.

Druhá, a fungující, možnost využívá toho, že jsou všechny administrátorské stránky vykreslovány způsobem SSR. Ověření tokenu tak probíhá na serveru ve funkci *getServerSideProps*, těsně před začátkem vykreslování stránky. Pokud nebyl součástí požadavku uživatele požadovaný token, je uživatel přesměrován na přihlašovací stránku.

3.4 Práce s databází

V sekci 2.2.4 byla jako ORM knihovna zvolena knihovna Prisma. Hlavními kroky při konfiguraci Prisma je nastavení generátoru a zdroje dat a následné definování modelů. Model určuje strukturu, kterou bude mít odpovídající tabulka v databázi. Příkazem *npx prisma generate* se na základě schématu vytvoří Prisma klient obsahující kód potřebný pro práci s definovanými modely. Změny schématu jsou pak zaneseny do databáze příkazem *npx prisma migrate*, který vytvoří soubor s SQL skripty provádějícími změny odpovídající změnám ve schématu, a aplikuje je na databázi. [26]

Pro účely testování a prezentace výsledného Webového portálu byl také vytvořen soubor *seed.ts*, který obsahuje skripty pro vkládání testovacích dat. Příkazem *npx prisma db seed* jsou pak tyto skripty spuštěny nad databází, která se daty naplní. Tento skript je navíc automaticky spouštěn při vytváření Docker kontejnerů během lokálního nasazení (viz. sekce 4.2).

3.5 Práce s fotografiemi

Vzhledem k funkčním požadavkům F06 a F12, které požadují galerii a její správu, bylo potřeba vyřešit nejen nahrávání a ukládání fotografií, ale i jejich následné zobrazení.

3.5.1 Nahrávání a ukládání

Pro zpracování fotografií byla použita knihovna Formidable, která se zaměřuje na zpracování formulářových dat, zejména pak souborů [35]. Administrátorem nahrané fotografie jsou poslány v těle formuláře na vlastní API koncový bod. Přijaté soubory jsou knihovnou Formidable zpracovány, přičemž se kontroluje jejich formát a velikost. V případě úspěchu při zpracování se fotografie uloží na server do *public* složky a odkazy na ně se uloží do databáze. V databázi je u každého odkazu na fotografii také uložen její popisek. Aby se předešlo případným konfliktům v názvech souborů, je před název při ukládání připojena časová značka. Při stahování je naopak odstraněna.

Toto řešení ukládání fotografií má zásadní vliv na finální nasazení Webového portálu (více v kapitole 4).

3.5.2 Přístup k fotografiím

Next.js při sestavování produkční verze aplikace optimalizuje fotografie, které se ve chvíli zahájení sestavování nachází ve složce *public*. U takto zpracovaných fotografií není třeba řešit cokoliv ohledně jejich optimalizace. K obsahu složky *public* se pak přistupuje jako ke statickým veřejným souborům, a to prostřednictvím kořenové adresy webové aplikace následované názvem souboru. Tím ale vznikl problém, kdy se dalo přistupovat pouze k fotografiím, které v *public* složce byly v době kompilace. Pokud administrátor nahrál fotografii po spuštění aplikace, Next.js ji nijak nezpracoval a žádný takový veřejný soubor pro uživatele neexistoval. [29]

Řešením bylo vytvoření dalšího koncového bodu, který má jako parametr cestu k fotografii. Pokud fotografie existuje, načte se ze souborového systému serveru, nastaví se potřebné hlavičky, jako jsou formát a velikost, a odešle se v odpovědi klientovi. Koncový bod poslouchá na HTTP požadavky metody GET, takže pro zobrazení fotografie v klientské části stačí vytvořit klasické HTML *img* elementy a jako zdroj jim dát adresu vlastního koncového bodu s názvem fotografie. Navíc se ale na elementu *img*, pro vykreslení optimální velikosti, musí atributem *sizes* určit velikosti fotografií v závislosti na různých šířkách obrazovky.

3.5.3 Galerie

Pro zobrazení nahraných fotografií ve veřejné části byly použity knihovny React Photo Album [36] a Yet Another React Lightbox [37]. Obě jsou od stejného autora, díky čemuž spolu dobře fungují.

Knihovna React Photo Album zobrazuje responzivní dlaždicovou galerii a umožňuje definování vlastního elementu, který je použit pro vykreslení fotografií. Tím lze využít optimalizaci fotografií nabízenou Next.js.

Pomocí Yet Another React Lightbox se kliknutím na fotografii otevře detailní prohlížení fotografií. Zde byla, stejně jako u React Photo Album, použita Next.js optimalizace fotografií. Aby tato část aplikace nebyla součástí balíčku JavaScript skriptů, které se načítají při prvním načtení celé aplikace, bylo využito Next.js funkcionality pro dynamické načítání komponent. Dynamické načítání umožňuje odložit stahování JavaScript balíčku až do chvíle, kdy je skutečně potřeba. [29]

Příklad implementace dynamického načítání je v ukázce kódu 3.2. Komponentě *PhotoAlbum* se předává funkce pro změnu stavu, který indikuje, zda uživatel interagoval s nějakou z fotografií galerie. Ve chvíli, kdy uživatel na fotografii klikne, se se změnou stavu zahájí dynamické načítání a zobrazí se od-

povídající indikátor načítání. Po dokončení načítání se komponenta *Lightbox* vykreslí.

```
const Lightbox = dynamic(() => (
  import('../components/Lightbox'), {
    loading: () => <LoadingOverlay />
  }
));

const GalleryPage = () => {
  const [interactive, setInteractive] = useState(false);

  return (
    <>
      <PhotoAlbum setInteractive={setInteractive} />
      {interactive && <Lightbox />}
    </>
  );
};
```

Ukázka kódu 3.2: Dynamické načtení komponenty

3.6 Vyhledávání, řazení, stránkování

Pro tabulky v administrátorské části a seznam účastníků ve veřejné části bylo nutné implementovat logiku pro vyhledávání, řazení a stránkování. Pro implementaci takové logiky jsou ideálním nástrojem tzv. *hooks*.

Hooks jsou speciální React funkce, které umožňují přidat hotovým komponentám specifické chování a funkcionalitu. To vede ke snadnému znovupoužití stejné logiky ve více částech aplikace. Nejpoužívanější hook funkce *useState*, která je implementována v knihovně React, umožňuje komponentám ukládat stav. [38]

V ukázce kódu 3.3 je příklad implementace vlastní hook funkce pro stránkování. Funkce *setPage* a *setPerPage* se následně využijí v ovládacích prvcích tabulky pro přechod mezi stránkami a nastavení počtu záznamů zobrazených na jedné stránce. Stejným způsobem byly implementovány hook funkce pro vyhledávání a řazení.

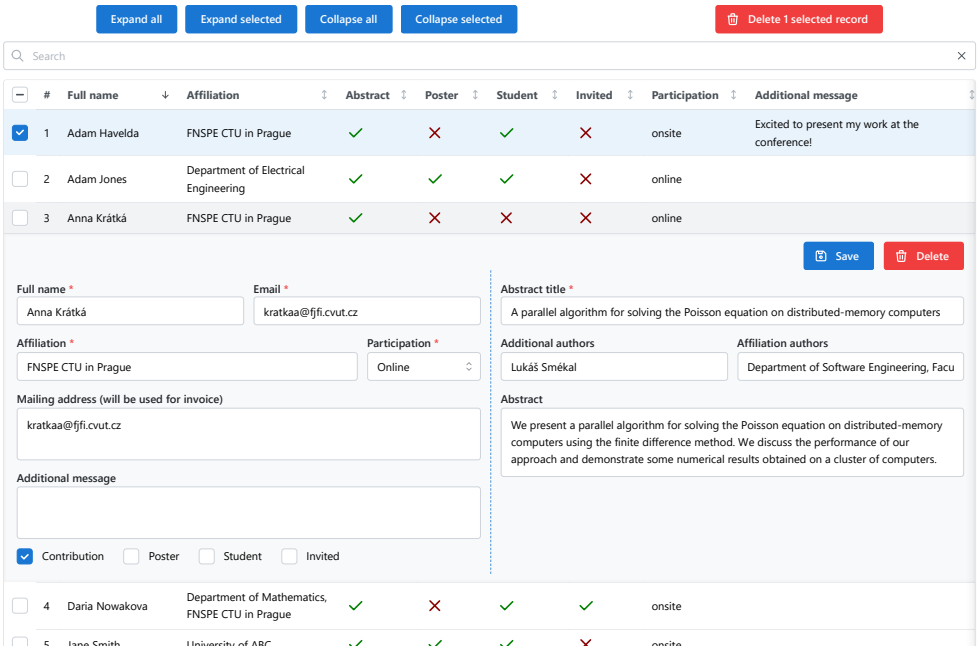
```
const usePaginate = <T>({ data, initialPage = 1 }) => {
  const [page, setPage] = useState(initialPage);
  const [perPage, setPerPage] = useState(defaultPagesNum);
  const [pagiResults, setPagiResults] = useState<T[]>([]);

  useEffect(() => {
    const fromIndex = (page - 1) * perPage;
    const toIndex = fromIndex + perPage;
    setPagiResults(data.slice(fromIndex, toIndex));
  }, [page, perPage, data]);

  useEffect(() => {
    setPage(initialPage);
  }, [initialPage, perPage]);

  return {page, setPage, perPage, setPerPage, paginateResults};
};
```

Ukázka kódu 3.3: Hook funkce pro stránkování dat

3.7 Tabulky

K přehlednému tabulkovému zobrazení dat o administrátorských účtech a registrovaných účastnících byla použita knihovna Mantine Datatable, která staví na použité knihovně komponent Mantine. Neřeší ale logiku vyhledávání, řazení a stránkování, pro kterou byly využity hooks popsané v sekci 3.6. [39] Tabulkovým zobrazením dat, spolu s řazením a vyhledáváním, je vyřešen problém popsaný v sekci 1.1.3.5.

Administrátorský účet tolik informací neobsahuje, takže se přehledně vejde do jednoho řádku tabulky. Proti tomu účastník informací obsahuje hodně a do jednoho řádku se nevejde. Řešením bylo zobrazit pouze ty nejdůležitější informace, podle kterých by administrátor zároveň mohl chtít řadit. Zbytek informací a možnost jejich úpravy se zobrazí po kliknutí na konkrétní řádek tabulky rozšířením toho řádku. Takovýto způsob úpravy informací, ve srovnání s běžným řešením v podobě vyskakovacího modálního okna, umožní upravovat víc detailů účastníků současně. Tabulka účastníků navíc umožňuje zvolit víc jednotlivých účastníků a hromadně zobrazit nebo skrýt jejich detail, případně je hromadně odstranit. Tabulku účastníků lze vidět na obrázku 3.2.



Obrázek 3.2: Administrátorská tabulka účastníků

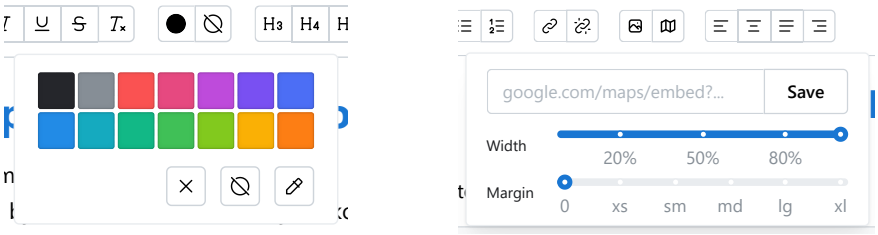
3.8 Úprava obsahu stránek

Administrátoři mají možnost upravovat obsah stránek prostřednictvím WYSIWYG (What You See Is What You Get) editoru ze zvolené knihovny komponent Mantine [22], který staví na WYSIWYG editoru TipTap [40]. WYSIWYG editor (dále pouze editor) je editor, který obsahuje textové pole a panel nástrojů, ve kterém jsou ovládací prvky umožňující provádění různých změn nad textem, a zobrazuje upravovaný obsah rovnou tak, jak bude ve výsledku vypadat [41]. Použití tohoto editoru řeší problém týkající se úpravy organizačních informací (viz. sekce 1.1.3.2).

Použitými ovládacími prvky z Mantine editoru jsou: tučný text, kurzíva, podtržení, přeškrtnutí, barva a zarovnání textu, nadpisy, seznamy, odkazy, fotografie a interaktivní mapa. Všechny zmíněné ovládací prvky kromě fotografie a interaktivní mapy jsou do Mantine editoru integrovány. Zbylé dva ovládací prvky bylo potřeba implementovat vlastnoručně za použití základních rozšíření z rodičovského TipTap editoru.

Po přidání rozšíření Image a Iframe je možné v editoru zobrazovat fotografie a mapy, ale chybí ovládací prvek, který by poskytoval uživatelsky přívětivé rozhraní pro jejich vkládání. Takové rozhraní musí nabízet možnost vložení odkazu (pro mapu) nebo názvu (pro fotografii) a možnost volby velikosti a odsazení prvku. Mantine umožňuje snadné přidání vlastního ovládacího prvku editoru v případě, že se akce má vykonat bezprostředně po kliknutí na něj [22]. Rozhraní pro vkládání mapy a fotografie tímto způsobem realizovat nelze.

Mantine editor rozhraní podobné požadovanému nabízí při volbě barvy textu (viz. obrázek 3.3a), čímž byla vlastní implementace ovládacích prvků inspirována. Vytvořené ovládací prvky po aktivaci otevřou vyskakovací okno, které obsahuje textové pole pro zadání odkazu nebo názvu vkládaného prvku a dva posuvníky pro volbu šířky a odsazení. K tomu bylo nutné rozšířit i základní TipTap elementy *Image* a *Iframe* přidáním atributů *src*, *width* a *margin*. Rozšíření elementů umožňuje tyto atributy přidat na výsledné HTML elementy a zpět je z nich získat. Rozhraní pro vkládání mapy je vidět na obrázku 3.3b. Obdobně funguje vkládání fotografií.



(a) Volba barvy (b) Vkládání mapy

Obrázek 3.3: Ovládací prvky editoru

3.9 Program konference

Pro sestavení denního programu konference byl vytvořen vlastní editor programu. Po volbě data začátku konference je umožněno přidávat jednotlivé dny.

Každý den může obsahovat dodatečné informace vkládané prostřednictvím WYSIWYG editoru popsaného v sekci 3.8. Následuje volba času začátku dne, ke kterému se přičítá délka všech položek programu toho dne. Položky programu jsou dvou typů – *item* nebo *chairman*. Položka *Item* zastupuje přednášku nebo libovolnou doplňující akci a může obsahovat titulek a účastníka s příspěvkem. Položka *Chairman* (předseda) slouží k volbě účastníka, který zasedá určité sekci programu (neboli všem položkám dne do dalšího předsedy). Pořadí položek dne je možno měnit pomocí tzv. *drag'n'drop* funkcionality, tedy přetažením myší.

Při mazání prázdné položky dne se položka rovnou smaže. Pokud ale již nějaká data obsahuje, zobrazí se upozornění s potvrzením smazání. Upozornění se také zobrazí při mazání dní.

Pokud dojde ke změně data začátku konference a jsou již vytvořené dny programu, jejich datum se změní vzhledem k novému datu začátku konference.

Administrátor má možnost program uložit a smazat. Uložením programu dojde k překreslení veřejné stránky *Programme*. Tam jednotlivé položky typu přednáška umožňují navigaci na stránku s *Participants & Abstracts*, kde se zobrazí detail konkrétního účastníka a příspěvku.

3.10 API koncové body

Vzhledem k doménovému modelu byly podle zvolené REST architektury API vytvořeny koncové body. Všechny zdroje tak podporují CRUD operace, tj. *Create* (vytvoření), *Read* (získání/přečtení), *Update* (změnu/aktualizaci) a *Delete* (smazání).

abstracts Zdroj pro manipulaci s příspěvky. Koncový bod */abstracts* umožňuje získání všech příspěvků pomocí GET požadavku a vytvoření příspěvku pomocí POST požadavku. Koncový bod */abstracts/{id}* umožňuje získání, změnu a smazání konkrétního příspěvku pomocí požadavků GET, PUT a DELETE (ve stejném pořadí).

admins Zdroj pro manipulaci s administrátory. Koncový bod */admins* umožňuje získání všech administrátorů pomocí GET požadavku a vytvoření administrátora pomocí POST požadavku. Koncový bod */admins/{id}* umožňuje získání, změnu a smazání konkrétního administrátora pomocí požadavků GET, PATCH a DELETE (ve stejném pořadí). Koncový bod */admin/{id}/password* umožňuje změnu hesla administrátora pomocí požadavku PATCH.

auth Koncové body vytvořené knihovnou NextAuth [34] sloužící pro autorizaci.

images Zdroj pro manipulaci s fotografiemi. Koncový bod */images* umožňuje získání metadat všech fotografií pomocí GET požadavku a nahrání nové fotografie pomocí POST požadavku. Koncový bod */images/{id}* umožňuje získání metadat, změnu metadat a smazání konkrétní fotografie pomocí požadavků GET, PUT a DELETE (ve stejném pořadí). Koncový bod */images/download/{...path}* umožňuje získání (stažení) konkrétní fotografie.

pages Zdroj pro manipulaci se stránkami Webového portálu. Koncový bod */pages* umožňuje získání všech stránek pomocí GET požadavku a vytvoření nové stránky pomocí POST požadavku. Koncový bod */pages/{id}* umožňuje získání, změnu a smazání konkrétní stránky pomocí požadavků GET, PUT a DELETE (ve stejném pořadí).

participants Zdroj pro manipulaci s účastníky. Koncový bod */participants* umožňuje získání všech účastníků pomocí GET požadavku, vytvoření nového účastníka pomocí POST požadavku a hromadné smazání účastníků pomocí DELETE požadavku s povinným parametrem *ids*. Parametr *ids* je seznam identifikátorů účastníků k odstranění, oddělených čárkami. Koncový bod */participants/{id}* umožňuje získání, upravení a smazání konkrétního účastníka pomocí požadavků GET, PUT a DELETE (ve stejném pořadí).

settings Zdroj pro manipulaci s globálními informacemi Webového portálu. Koncový bod */settings* umožňuje získání všech globálních informací pomocí GET požadavku a hromadnou úpravu pomocí PUT požadavku. Vzhledem k pevně daným typům globálních informací neumožňuje přidávání a odebírání.

programme Zdroj pro manipulaci s programem Webového portálu. Koncový bod */programme* umožňuje získání programu pomocí GET požadavku, vytvoření nového programu a změnu stávajícího pomocí PUT požadavku a odstranění programu pomocí DELETE požadavku.

Kapitola 4

Nasazení

V této kapitole se popíše postup při nasazení realizovaného prototypu Webového portálu. Kromě samotné webové aplikace je potřeba nasadit i databázi.

4.1 Hostovací platforma

Pro testovací účely byl Webový portál nasazen prostřednictvím hostovací platformy. Možnosti pro nasazení jsou omezeny způsobem ukládání fotografií (viz. sekce 3.5.1). Některé platformy, jakou je například platforma Vercel [42] od autorů Next.js, neumožňují ukládání souborů na disk serveru.

Po průzkumu několika platforem pro hostování webových aplikací, které ukládání souborů na disk umožňují, byla zvolena plaftorma DigitalOcean. Velké množství aplikací lze jednoduše nasadit prostřednictvím DigitalOcean App Platform, což je PaaS (Platform-as-a-Service), která odstiňuje od potřeby řešit infrastrukturu serveru. Pokud není požadovaná technologie podporována tímto způsobem nasazení, je další možností nasazení prostřednictvím virtuálních strojů (tzv. Droplets). [43]

Webová aplikace byla nasazena přes DigitalOcean App Platform, a to propojením s GitHub repositářem. Databáze byla vytvořena a nasazena jako tzv. *spravovaná databáze*, tedy server s předpřipravenou databází. Po spuštění databáze se k ní stačilo z lokálního vývojového prostředí připojit prostřednictvím knihovny Prisma a nahrát databázové schéma s daty (viz. sekce 3.4). Nasazená aplikace a databáze byly v rámci DigitalOcean přidány do společného projektu, což umožnilo jejich jednoduché propojení. [43]

DigitalOcean nenabízí žádné členství s bezplatným hostováním. Každý uživatel ale po registraci dostane dárek v podobě kreditu ve výši $200, který je platný po dobu 60 dní. Náklady na hostování aplikace a databáze činí dohromady $20 měsíčně, což kredit z bonusu po registraci plně pokrývá. [43]

4.2 Docker

Vzhledem k požadavku N05 bude možno Webový portál a databázi nasadit, nebo spustit lokálně, také v podobě Docker kontejnerů. Docker je platforma umožňující vytváření, nasazování a spouštění aplikací v kontejnerech. Kontejner je samostatný spustitelný balíček, který obsahuje vše potřebné pro běh aplikace, včetně kódu, knihoven a závislostí. Pro souběžné spuštění dvou a více kontejnerů, které spolu mají nějakým způsobem komunikovat, je vhodné využít nástroje Docker Compose. V konfiguračním souboru *docker-compose.yml* se definuje nastavení kontejnerů a příkazem *docker compose up* se všechny kontejnery společně spustí. [44]

Výsledný *docker-compose.yml* obsahuje dvě služby, databázi a aplikaci, které poběží ve svých kontejnerech. Kontejner pro databázi je sestavený jako běžný PostgreSQL kontejner za použití oficiálního obrazu [44]. Při sestavování aplikace je potřeba získat data, kterými se naplní statické stránky. Proto bylo nutné vytvořit vlastní *Dockerfile*, ve kterém se definuje postup sestavení kontejneru aplikace. Nejdříve se stáhne rodičovský obraz, na kterém se bude stavět. Z důvodu potřeby čekání na spuštění databáze se následně stáhne nástroj docker-compose-wait, který umožňuje čekat na otevření portu na cílovém kontejneru [45]. Po otevření portu se spustí skript *setup.sh*, který zahájí instalaci závislostí aplikace, nahraje schéma do databáze a naplní ji daty, načež dojde k sestavení a finálnímu spuštění aplikace.

Návod k lokálnímu spuštění se nachází v přiloženém archivu v souboru *impl/README.md*. V přiloženém archivu v adresáři *impl/* je rovněž možné nalézt soubory *docker-compose.yml*, *Dockerfile* a *setup.sh*.

Kapitola 5

Testování

V této kapitole se popíše testování, které proběhlo nad Webovým portálem v průběhu vývoje a po nasazení.

5.1 Během vývoje

Během vývoje byl Webový portál testován manuálně a s použitím vytvořených skriptů. Manuální testování zahrnuje testování responzivity uživatelského rozhraní nástrojem Polypane (viz. sekce 3.1.5) a průběžnou interakci s vyvíjenou verzí Webového portálu. Z počátku bylo také API testováno manuálně. To se ale při rostoucím počtu koncových bodů stalo zdlouhavým procesem, což vedlo k vytvoření skriptů v nástroji Postman [46]. Příklad testu, který se spustí po odeslaném požadavku a přijaté odpovědi, lze vidět v ukázce kódu 5.1.

5.2 Uživatelské testování

Po nasazení Webového portálu došlo k uživatelskému testování. „*Uživatelské testování je analytická metoda reflektující interakci reálných uživatelů s webem, eshopem nebo aplikací. Pracuje tedy se skutečnými zákazníky a pomáhá přeměnit původní dohady a předpoklady v podložená data. Designer nebo programátor tak získává tolik potřebnou zpětnou vazbu přímo od své cílové skupiny.*“ [47]

Testování se účastnily dvě skupiny uživatelů – běžní uživatelé a administrátoři původního Webového portálu. Skupina běžných uživatelů dostala scénáře pokrývající veřejnou část Webového portálu, zatímco skupina administrátorů testovala část administrátorskou. Scénáře první skupiny pokrývaly registraci jako účastník, najití konkrétní organizační informace, vyhledání registrovaného účastníka a procházení galerie se stažením fotografie. Scénáře druhé skupiny pokrývaly přihlášení a změnu hesla, úpravu informací v globální hlavičce, nahrání a vložení fotografie do textu stránky, úpravu textu stránky, tvorbu programu a vytvoření a úpravu účastníka.

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});

pm.test("Participant is created", function () {
    const participant = pm.response.json();
    pm.expect(participant).to.have.property("id");
    pm.expect(participant)
        .to.have.property("fullName", "Jan Novák");
    pm.expect(participant)
        .to.have.property("email", "novak@example.com");
    pm.expect(participant)
        .to.have.property("affiliation", "ČVUT");
});
```

Ukázka kódu 5.1: Test API v nástroji Postman

Výsledkem testování jsou následující připomínky a poznatky:

- Při vkládání fotografie do obsahu stránky by bylo vhodné napovídat, jaké všechny fotografie jsou k dispozici. Aktuální implementace poskytuje pouze pole pro vložení názvu fotografie, takže si ho uživatel musí pamatovat nebo je nucen kopírovat ho z administrační stránky *Gallery*.
- Pořadí položek konkrétního dne programu lze měnit přetažením myší (tzv. *drag'n'drop*) a bylo by vhodné mít stejnou možnost i při změně pořadí celých dní.
- Po registraci by účastník mohl obdržet potvrzující e-mail se soupisem informací zadaných při registraci.
- Před odesláním vyplněného registračního formuláře by měl uživatel vyplnit ověření, zda není robot (tzv. *captcha*).
- Přidání možnosti exportu dat o konferenci a účastnících.

Závěr

Cílem bakalářské práce bylo vytvořit Webový portál pro organizování vědeckých konferencí v podobě full-stack webové aplikace a zároveň provést analýzu jeho aktuálně používaného řešení, při které se identifikují problémy, na základě kterých dojde k návrhu a implementaci nové verze. Účelem bylo poskytnout organizátorům, účastníkům a zájemcům konference uživatelsky přívětivé rozhraní a systém, který by umožňoval snadné rozšiřování a údržbu.

Cíle práce byly splněny typickým postupem při realizaci softwarového projektu. Nejprve byla provedena analýza aktuálně používaného Webového portálu, byly identifikovány jeho nedostatky a možnosti zlepšení. Během analýzy požadavků se rovněž požadavky, které splňovala aktuální verze Webového portálu, rozšířily o požadavky získané od organizátorů konference. Na základě analýzy se zvolila vhodná architektura a technologie a došlo k návrhu uživatelského rozhraní. Realizace prototypu přiblížila použité nástroje a stěžejní části implementace a byla následována nasazením vytvořeného prototypu.

Během vývoje docházelo k průběžnému testování, a to manuálnímu a s pomocí skriptů. Po nasazení prototypu došlo k uživatelskému testování, kterého se účastnili běžní uživatelé a administrátoři původního Webového portálu. Výstupem testování jsou možná zlepšení a rozšíření, jako přidání napovídání při vkládání fotografií do obsahu stránek, možnost změny pořadí dne přetažením myší, zaslání potvrzujícího e-mailu po registraci a přidání exportu dat o konferenci a účastnících.

Literatura

[1] Mathematical Modelling Group: Student workshop on scientific computing [online]. 2023, [vid. 2023-03-24]. Dostupné z: https://geraldine.fjfi.cvut.cz/wsc2023/

[2] Mlejnek, J.: Architektonické vzory [online]. 2023, [vid. 2023-03-25]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/600986/mod_resource/content/5/06.prednaska.pdf

[3] Davis, M. E.; Philips, J. A.: *Learning PHP & MySQL*. O'Reilly Media, Inc., druhé vydání, 2007, ISBN 9780596514013, [vid. 2023-03-25]. Dostupné z: https://www.oreilly.com/library/view/learning-php/9780596514013

[4] OpenJS Foundation: jQuery [online]. 2023, [vid. 2023-03-25]. Dostupné z: https://jquery.com/

[5] Singh, A. J.: File system vs DBMS [online]. 2022, [vid. 2023-03-26]. Dostupné z: https://www.scaler.com/topics/file-system-vs-dbms/

[6] Miller, M.: 2 Dangers Of Shared Accounts [online]. 2023, [vid. 2023-03-26]. Dostupné z: https://www.triaxiomsecurity.com/2-dangers-of-shared-accounts/

[7] Kompaniets, A.: Functional Vs. Non-Functional Requirements: Why Are Both Important? [online]. 2023, [vid. 2023-03-27]. Dostupné z: https://www.uptech.team/blog/functional-vs-non-functional-requirements

[8] Milani, F.: *Digital Business Analysis* [online]. Springer, 2019, ISBN 978-3-030-05718-3, [vid. 2023-04-5]. Dostupné z: https://link.springer.com/book/10.1007/978-3-030-05719-0

[9] Evans, E.: *Domain-Driven Design Reference* [online]. Domain Language Inc., 2015, [vid. 2023-03-28]. Dostupné z: https://www.domainlanguage.com/wp-content/uploads/2016/05/DDD_Reference_2015-03.pdf

[10] Fowler, M.: *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002, ISBN 978-0321127426, [vid. 2023-03-28]. Dostupné z: https://martinfowler.com/books/eaa.html

[11] Sparks, G.: Database Modelling in UML [online]. [vid. 2023-03-28]. Dostupné z: https://sparxsystems.com/downloads/whitepapers/Database_Modeling_In_UML.pdf

[12] Davidson, T.: Single Page Application (SPA) vs Multi Page Application (MPA): Which Is The Best? [online]. 2023, [vid. 2023-04-02]. Dostupné z: https://cleancommit.io/blog/spa-vs-mpa-which-is-the-king/

[13] Skólski, P.: Single-Page Application vs. Multiple-Page Application [online]. 2016, [vid. 2023-04-02]. Dostupné z: https://neoteric.eu/blog/single-page-application-vs-multiple-page-application/

[14] Spittel, A.: What is a Web Framework, and Why Should I use one? [online]. [vid. 2023-04-02]. Dostupné z: https://welearncode.com/what-are-frontend-frameworks/

[15] Krotoff, T.: Front-end frameworks popularity (React, Vue, Angular and Svelte) [online]. 2023, [vid. 2023-04-02]. Dostupné z: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

[16] Ekainu, G. A.: 9 Best JavaScript Frameworks to Use in 2023 [online]. 2023, [vid. 2023-04-03]. Dostupné z: https://ninetailed.io/blog/best-javascript-frameworks/

[17] Agrawal, H.: What is TypeScript and why should you use it? [online]. 2022, [vid. 2023-04-03]. Dostupné z: https://www.contentful.com/blog/what-is-typescript-and-why-should-you-use-it/

[18] D'Avanzo, L.: What is a JavaScript Meta-framework? [online]. 2022, [vid. 2023-04-03]. Dostupné z: https://www.ombulabs.com/blog/javascript/what-is-a-javascript-meta-framework.html

[19] State of JS: Front-end frameworks [online]. 2022, [vid. 2023-04-02]. Dostupné z: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/

[20] Hofmann, F.: Getting Started: Gatsby vs. Next.js vs. Remix [online]. 2022, [vid. 2023-04-03]. Dostupné z: https://satellytes.com/blog/post/getting-started-gatsby-next-remix/

[21] New Target: What Is a Component Library? [online]. 2022, [vid. 2023-04-03]. Dostupné z: https://www.newtarget.com/web-insights-blog/what-is-a-component-library/

[22] Rtishchev, V.: Mantine [online]. 2023, [vid. 2023-04-03]. Dostupné z: https://mantine.dev/

[23] Rtishchev, V.: Mantine UI [online]. 2023, [vid. 2023-04-03]. Dostupné z: https://ui.mantine.dev/

[24] Walker, J.: SQL vs. NoSQL: Which is right for your workload? [online]. 2022, [vid. 2023-04-04]. Dostupné z: https://www.cockroachlabs.com/blog/document-store-vs-relational-database/

[25] Stack Overflow: Stack Overflow Developer Survey 2022 [online]. 2022, [vid. 2023-04-04]. Dostupné z: https://survey.stackoverflow.co/2022/

[26] Nnakwue, A.: The best TypeScript ORMs [online]. 2022, [vid. 2023-04-16]. Dostupné z: https://blog.logrocket.com/best-typescript-orms

[27] AltexSoft: Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC [online]. 2020, [vid. 2023-04-09]. Dostupné z: https://www.altexsoft.com/blog/soap-vs-rest-vs-graphql-vs-rpc/

[28] npm, Inc.: npm [online]. 2023, [vid. 2023-04-15]. Dostupné z: https://www.npmjs.com/

[29] Vercel Inc.: Next.js [online]. 2023, [vid. 2023-04-15]. Dostupné z: https://nextjs.org/

[30] Prettier: Prettier [online]. 2023, [vid. 2023-04-15]. Dostupné z: https://prettier.io/

[31] OpenJS Foundation: ESLint [online]. 2023, [vid. 2023-04-15]. Dostupné z: https://eslint.org/

[32] Novoseltseva, E.: Benefits Of Using Github [online]. 2020, [vid. 2023-04-15]. Dostupné z: https://apiumhub.com/tech-blog-barcelona/using-github/

[33] Polypane: Polypane, The browser for ambitious web developers [online]. 2023, [vid. 2023-04-20]. Dostupné z: https://polypane.app/

[34] Auth.js: NextAuth [online]. 2023, [vid. 2023-04-15]. Dostupné z: https://next-auth.js.org/

[35] Reagent, C. M.: formidable [online]. 2023, [vid. 2023-04-16]. Dostupné z: https://www.npmjs.com/package/formidable

[36] Danchenko, I.: React Photo Album [online]. 2023, [vid. 2023-04-16]. Dostupné z: https://react-photo-album.com/

[37] Danchenko, I.: Yet Another React Lightbox [online]. 2023, [vid. 2023-04-16]. Dostupné z: https://yet-another-react-lightbox.com/

[38] Kříž, P.: React Hooks, které potřebujete znát [online]. 2019, [vid. 2023-04-16]. Dostupné z: https://zdrojak.cz/clanky/react-hooks-ktere-potrebujete-znat/

[39] Florescu, I.-C.: Mantine Datatable [online]. 2023, [vid. 2023-04-16]. Dostupné z: https://icflorescu.github.io/mantine-datatable/

[40] überdosis: Headless WYSIWYG Text Editor – Tiptap Editor [online]. 2023, [vid. 2023-04-16]. Dostupné z: https://tiptap.dev/

[41] Štráfelda, J.: Co je WYSIWYG [online]. 2022, [vid. 2023-04-16]. Dostupné z: https://www.strafelda.cz/wysiwyg

[42] Vercel Inc.: Vercel [online]. 2023, [vid. 2023-04-18]. Dostupné z: https://vercel.com/

[43] DigitalOcean, LLC: DigitalOcean [online]. 2023, [vid. 2023-04-18]. Dostupné z: https://www.digitalocean.com/

[44] Docker Inc.: Docker [online]. 2023, [vid. 2023-04-18]. Dostupné z: https://www.docker.com/

[45] Cinà, F.: docker-compose-wait [online]. 2023, [vid. 2023-04-18]. Dostupné z: https://github.com/ufoscout/docker-compose-wait/

[46] Postman, Inc.: API testing [online]. 2023, [vid. 2023-04-20]. Dostupné z: https://www.postman.com/api-platform/api-testing/

[47] Koďousková, B.: Jak na uživatelské testování webu a aplikací [online]. 2021, [vid. 2023-04-20]. Dostupné z: https://www.rascasone.com/cs/blog/uzivatelske-testovani-webu-aplikace

Příloha A

Seznam použitých zkratek

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

DOM Document Object Model

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

ISR Incremental Static Regeneration

JSON JavaScript Object Notation

JWT JSON Web Token

MPA Multi-Page Application

NPM Node Package Manager

ORM Object-Relational Mapping

PaaS Platform-as-a-Service

PHP PHP: Hypertext Preprocessor

REST Representational State Transfer

RPC Remote Procedure Call

SEO Search Engine Optimization

SPA Single Page Application

SQL Structured Query Language

SSG Static-Site Generating

SSR Server-Side Rendering

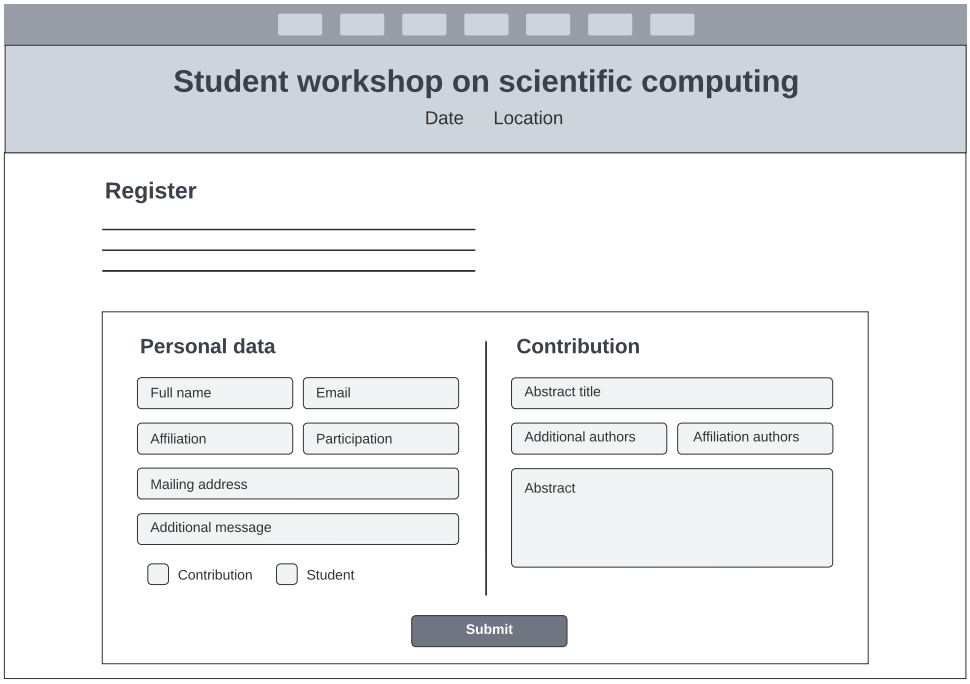
UML Unified Modeling Language

Webový portál Webový portál pro organizování vědeckých konferencí

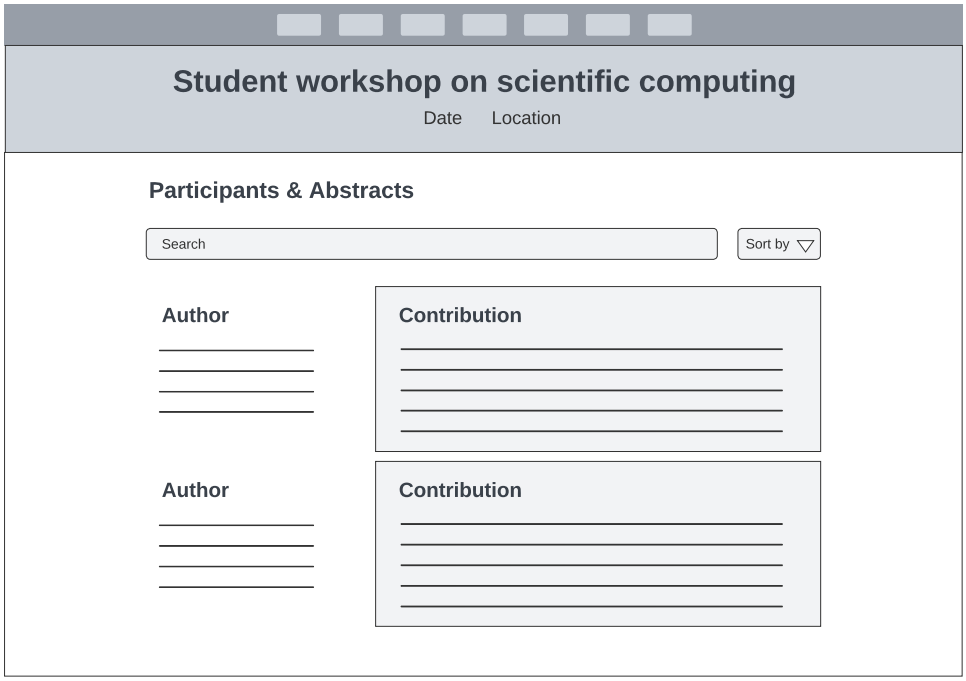
WYSIWYG What You See Is What You Get

Příloha B

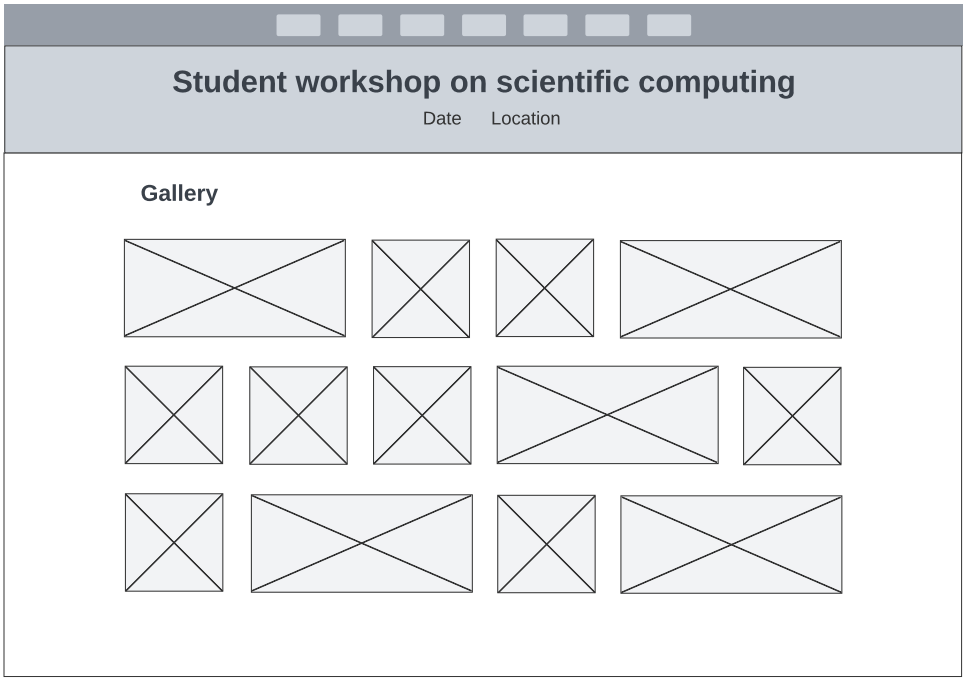
Návrh uživatelského rozhraní veřejné části



Obrázek B.1: Návrh veřejné stránky Register



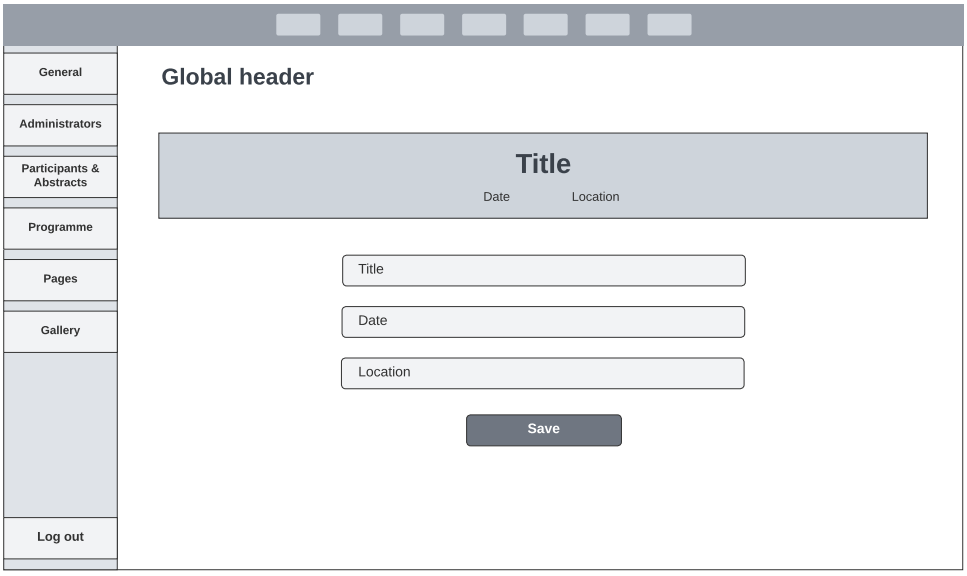
Obrázek B.2: Návrh veřejné stránky Participants & Abstracts



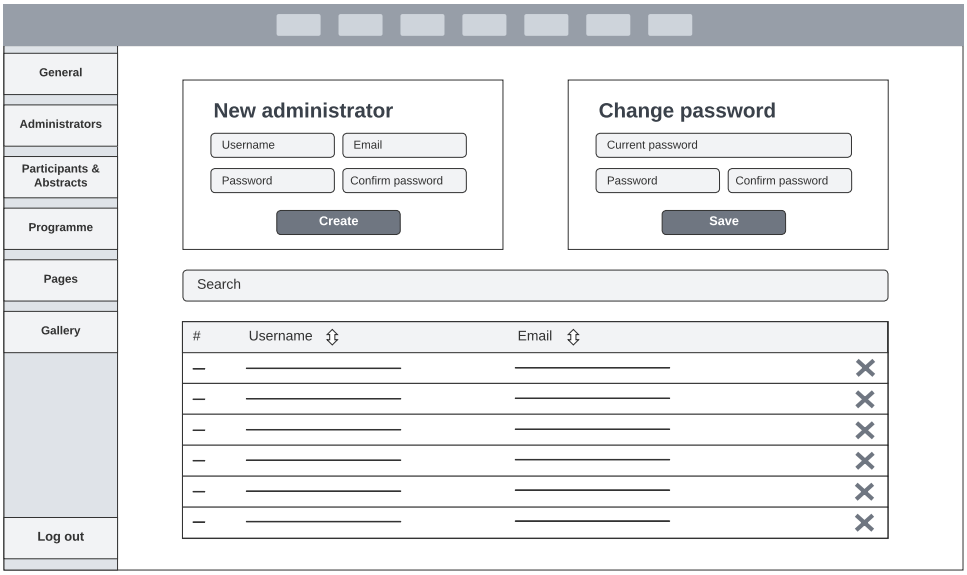
Obrázek B.3: Návrh veřejné stránky Gallery

Příloha C

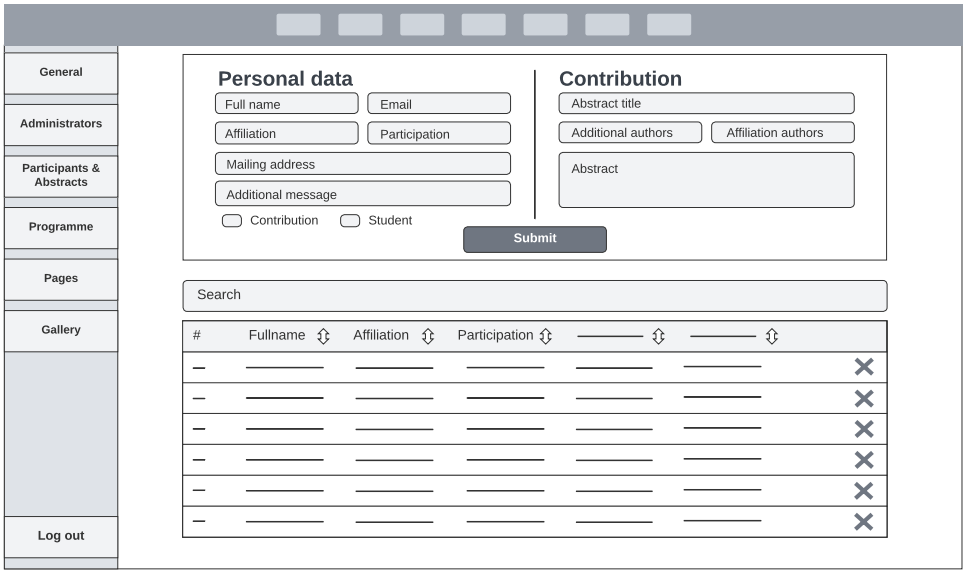
Návrh uživatelského rozhraní administrátorské části



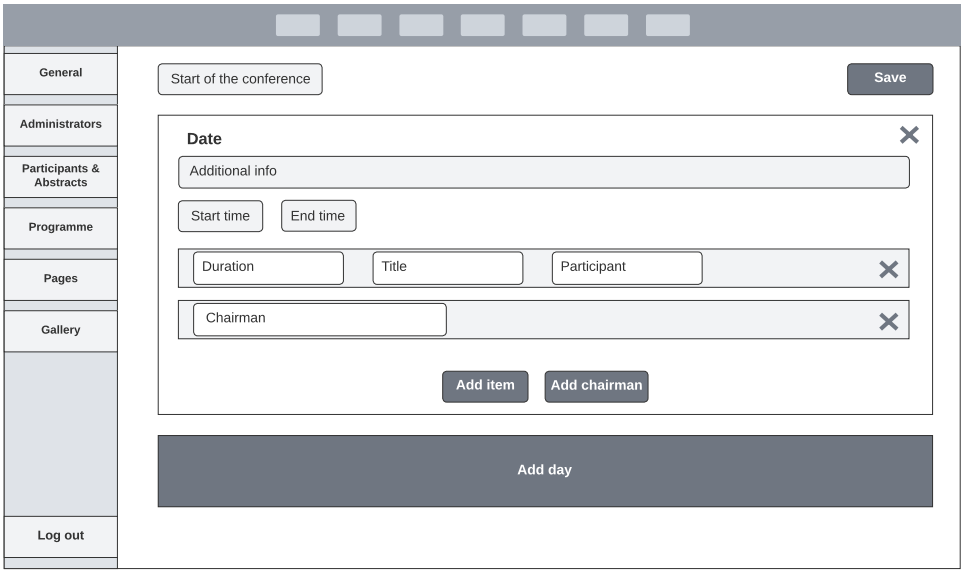
Obrázek C.1: Návrh administrátorské stránky General



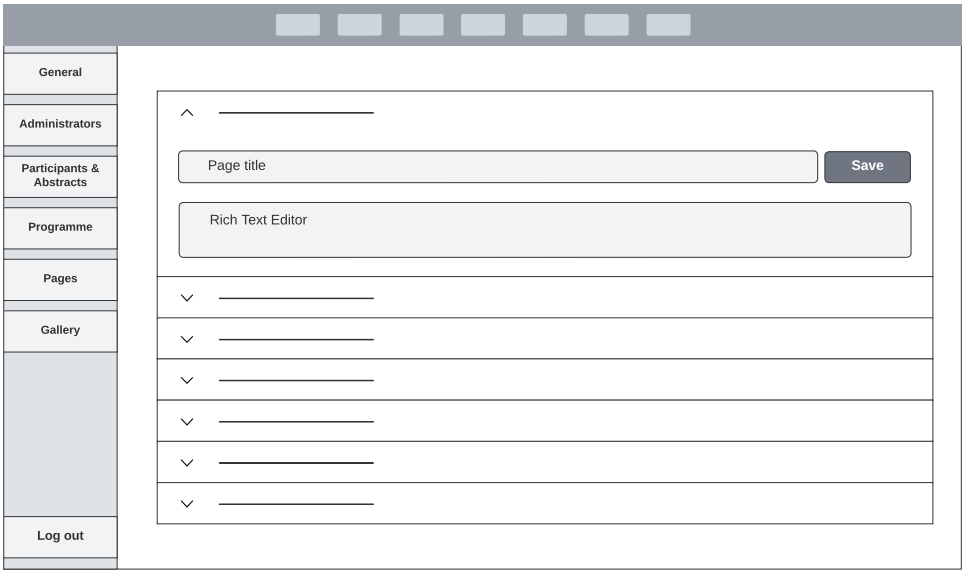
Obrázek C.2: Návrh administrátorské stránky Administrators



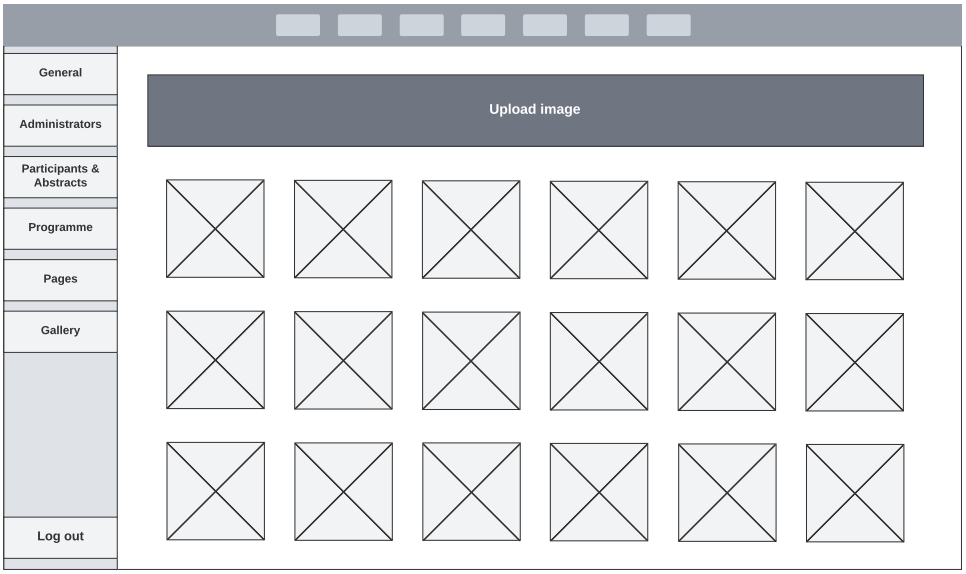
Obrázek C.3: Návrh administrátorské stránky Participants & Abstracts



Obrázek C.4: Návrh administrátorské stránky Programme



Obrázek C.5: Návrh administrátorské stránky Pages



Obrázek C.6: Návrh administrátorské stránky Gallery

Příloha D

Obsah přiloženého archivu

/
- `README.md` ..... stručný popis obsahu
- `impl` ..... implementace práce
  - `README.md` ..... návod k lokálnímu spuštění
  - `src` ..... zdrojové kódy
- `text` ..... text práce
  - `BP_Cabak_Tomas_2023.tex` zdrojová forma práce ve formátu LaTeX
  - `BP_Cabak_Tomas_2023.pdf` ..... text práce ve formátu PDF