



REIMPLEMENTACE DATABÁZE KONFERENCÍ A PUBLIKACÍ II

Bc. David Kocman

Diplomová práce
Fakulta informačních technologií
České vysoké učení technické v Praze
Katedra softwarového inženýrství
Studijní program: Informatika
Specializace: Softwarové inženýrství
Vedoucí: doc. Ing. Petr Fišer, Ph.D.
6. května 2025

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kocman** Jméno: **David** Osobní číslo: **526300**
Fakulta/ústav: **Fakulta informačních technologií**
Zadávající katedra/ústav: **Katedra softwarového inženýrství**
Studijní program: **Informatika**
Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Reimplementace databáze konferencí a publikací II

Název diplomové práce anglicky:

Reimplementation of the database of conferences and publications II

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Petr Fišer, Ph.D. katedra číslicového návrhu FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **26.11.2024**

Termín odevzdání diplomové práce: **09.05.2025**

podpis vedoucí(ho) ústavu/katedry

podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Bc. Kocman David

Podpis studenta

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kocman** Jméno: **David** Osobní číslo: **526300**
Fakulta/ústav: **Fakulta informačních technologií**
Zadávající katedra/ústav: **Katedra softwarového inženýrství**
Studijní program: **Informatika**
Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Reimplementace databáze konferencí a publikací II

Název diplomové práce anglicky:

Reimplementation of the database of conferences and publications II

Pokyny pro vypracování:

Přeprogramujte část stávající webové aplikace pro správu publikací a konferencí do jazyka podporovaného ICT FIT, tj. TypeScript. Původním jazykem je PHP (Nette Framework) a JavaScript. Jako databázi použijte PostgreSQL (původní je MySQL). Jedná se o přeprogramování front-endu i back-endu aplikace. Zaměřte se konkrétně na:

- migraci databáze z MySQL do PostgreSQL,
- správu konferencí (konferencí, ročníků konferencí),
- vytváření publikací s návazností na konference a časopisy,
- import (BibTeX, Springer API) a export (citační formáty) publikací.

Dále vytvořte důkladnou dokumentaci vytvořené části práce (uživatelskou i programátorskou), dle požadavků stanovených vedoucím práce.

Výslednou aplikaci důkladně otestujte, proveďte funkční a uživatelské testy. Ze zpětné vazby uživatelů navrhnete témata pro další vylepšení aplikace.

Seznam doporučené literatury:

České vysoké učení technické v Praze
Fakulta informačních technologií

© 2025 Bc. David Kocman. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.

Odkaz na tuto práci: Kocman David. *Reimplementace databáze konferencí a publikací II*. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2025.

Chtěl bych poděkovat především svému vedoucímu, panu doc. Ing. Petrovi Fišerovi PhD. za jeho ochotu, zpětnou vazbu, profesionální přístup a poskytnuté konzultace při zpracovávání této diplomové práce. Také bych chtěl srdečně poděkovat mé rodině a přátelům za nepřetržitou podporu při mých studiích na FIT ČVUT.

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.

Prohlašuji, že jsem v průběhu příprav a psaní závěrečné práce použil nástroje umělé inteligence. Vygenerovaný obsah jsem ověřil. Stvrzuji, že jsem si vědom, že za obsah závěrečné práce plně zodpovídám.

V Praze dne 6. května 2025

Abstrakt

Cílem této diplomové práce je přeprogramování části webové aplikace pro správu konferencí a publikací, která je původně napsána v jazycích PHP a JavaScript, do jazyka TypeScript a provedení migrace databáze z MySQL do PostgreSQL. Nová aplikace je vyvinuta ve full-stack frameworku Next.js, který na front-endu používá knihovnu React a pro komunikaci s databází se používá Prisma ORM. Práce se zaměřuje konkrétně na implementaci správy konferencí a jejích ročníků, vytváření publikací s návazností na konference, časopisy, vydavatele a autory, a import (citační formáty, Springer API) a export (citační formáty) publikací. K aplikaci jsou vytvořeny automatizované integrační testy a nakonec je také celá aplikace otestována uživatelskými testy použitelnosti.

Klíčová slova TypeScript, Uživatelské rozhraní, Full-stack, React, Next.js, Publikace, Odborná práce, Konference, PostgreSQL, Citace, Databáze, Prisma

Abstract

This thesis aims to reimplement a part of a web application for managing conferences and publications, which was written in PHP and JavaScript, to the TypeScript language and the migration of the database from MySQL to PostgreSQL. The new application is developed in the Next.js full-stack framework, which uses the React library on the front-end, and to communicate with the database, the Prisma ORM is used. The thesis specifically focuses on implementing the management of conferences and their annual editions, creating publications linked to conferences, journals, publishers, and authors, and importing (citation formats, Springer API) and exporting (citation formats) publications. Automatic integration tests were developed for the application and finally, the entire application was tested through usability tests.

Keywords TypeScript, User interface, Full-stack, React, Next.js, Publications, Academic paper, Conferences, PostgreSQL, Citations, Database, Prisma

Obsah

Úvod	1
Cíle práce	1
Vymezení vypracovaných částí	2
1 Analýza	4
1.1 Obecný popis původní aplikace	4
1.2 Návrh	6
1.2.1 Volba technologií	7
1.2.2 Architektura	8
1.2.3 Databáze	9
1.2.4 Návrh uživatelského rozhraní	11
1.2.4.1 Vytváření publikací	11
1.2.4.2 Konference	15
1.3 Požadavky na novou funkcionalitu	19
1.4 Dokumentace	21
2 Implementace	22
2.1 Struktura projektu	22
2.2 Migrace databáze	23
2.3 Zapracované Issues z původní aplikace	24
2.4 Publikace	25
2.4.1 Vytváření	25
2.4.1.1 Formulář	26
2.4.1.2 Zápis do databáze	34
2.4.1.3 Implementace správy souvisejících entit z formuláře	35
2.4.1.4 Implementace správy souborů	40
2.4.1.5 Implementace vlastní rozbalovací sekce	44
2.4.2 Úprava publikace	45
2.4.3 Import přes definice	46
2.4.4 Import přes Springer API	50
2.4.5 Export	51
2.4.6 Detail publikace	53
2.5 Konference	56
2.5.1 Seznam	56
2.5.1.1 Modály a akční tlačítka	60

2.5.2	Vytváření a úprava	61
2.5.3	Detail konferencí a ročníků konferencí	66
2.5.3.1	Sjednocení konferencí	67
2.5.4	Správa a sjednocení kategorií	68
2.5.5	Správa databází dokumentových indexů	71
2.6	Nová podoba databáze	71
3	Testování	73
3.1	Uživatelské testování	73
3.1.1	Scénáře Publikace (SP)	74
3.1.2	Scénáře Konference (SK)	75
3.1.3	Vyhodnocení testování	78
3.1.3.1	Tabulka připomínek	81
3.2	Funkční testování	82
3.3	Zpracované změny	83
3.4	Návrhy na vylepšení aplikace	84
	Závěr	86
	A Závěrečný dotazník 1. uživatele	87
	B Závěrečný dotazník 2. uživatele	89
	C Závěrečný dotazník 3. uživatele	91
	D Závěrečný dotazník 4. uživatele	93
	E Mapa Citation.js typů na PubConf typy publikací u importu	95
	F Mapa PubConf typů na Citation.js typy publikací u export	97
	G Uživatelská dokumentace	99
	Obsah příloh	128

Seznam obrázků

1.1	Příklad klikatelného záznamu v seznamu publikací.	5
1.2	Seznam ročníků konference v původní aplikaci. V levém horním rohu lze vidět zavádějící cesta. Také horní lišta říká, že se jedná o stránku s „konferencemi“.	6
1.3	ER diagram databáze původní aplikace před změnami. Převzato z [14].	10
1.4	Neideální pozice popisku vstupního pole v původní aplikaci. . .	11
1.5	První část formuláře pro vytváření publikací. Lze zde vidět formát chybových hlášek.	12
1.6	Druhá a třetí část formuláře pro vytváření publikací.	12
1.7	Design multifunkčního tlačítka.	13
1.8	Design modálu pro import publikace pomocí citačního formátu. . .	14
1.9	Design modálu pro import publikace přes Springer API.	14
1.10	Design seznamu konferencí a ročníků konferencí.	16
1.11	Design formuláře pro vytváření konferencí. Na obrázku lze vidět sekci pro vyplnění informací samotné konference.	17
1.12	Design formuláře pro vytváření konferencí. Na obrázku lze vidět sekci pro vytváření jednotlivých ročníků konferencí.	17
1.13	Design detailu ročníku konference.	18
1.14	Design seznamu databází dokumentových indexů. Vychází z designu ostatních stránek správy sekundárních entit [1].	19
1.15	Ilustrace sjednocení kategorií do jedné tabulky, přemapování na ročníky a nová tabulka pro uchovávání historie jmen. Přerušované tabulky a vztahy byly z databáze vymazány.	20
1.16	Ilustrace sjednocení dvou konferencí. Sjednocená konference nabyde ročníky obou konferencí. Nový ročník sjednocené konference (2025) nabyde všech kategorií, které měli poslední ročníky cílové a zdrojové konference (2022 a 2024) a také nabyde spojeného popisu z těchto let. Zdrojová konference je z databáze smazána.	21
2.1	Vyplnění políčka Booktitle a vybrání vydavatele ročníku. . . .	29
2.2	Design výběru autorů se seznamy zvolených a nových autorů. .	29
2.3	Upozorňující text na modálu pro vytvoření kategorie.	30
2.4	Výpis stávajících nebo nově nahraných souborů.	32
2.5	Rozšíření vstupního pole o tlačítka pro správu entity.	36

2.6	Modál pro vytvoření nové související entity.	39
2.7	Detail nové související entity.	39
2.8	Zobrazení definice na formuláři pro vytvoření publikace.	49
2.9	Formulář pro získání článku ze Springer API.	50
2.10	Záložka export na detailu publikace.	51
2.11	První část detailu publikace obsahující informace o publikaci. .	55
2.12	Druhá, třetí a čtvrtá část detailu publikace obsahující informace o souborech, tlačítka a metadata.	55
2.13	Finální podoba seznamu konferencí.	56
2.14	Ukázka správy předešlých jmen s varováním a „rollback“ nabídkou.	62
2.15	Konferenční formulář s novým ročníkem.	64
2.16	Detail konference.	67
2.17	Seznam souvisejících publikací konference.	67
2.18	Kontrola sjednocení dvou konferencí.	68
2.19	ER diagram aktuální databáze po změnách. Modře jsou vy- značeny nové tabulky.	72
3.1	Nová podoba tlačítek pro vytvoření či import publikací.	84

Seznam tabulek

2.1	Entity a atributy objektu IForm, které slouží pro ukládání nově vytvořených nebo upravených entit.	37
2.2	Návratové hodnoty POST metody.	41
2.3	Návratové hodnoty DELETE metody.	42
2.4	Návratové hodnoty GET metody.	42
2.5	Tabulka akcí v posledním sloupci záznamu podle typu a kde se nachází.	60
3.1	Výsledná klasifikace připomínek.	82

Seznam výpisů kódu

2.1	Špatný formát data u publikace s id 438 a možnost opravy. . .	23
2.2	Implementace logiky odeslání formuláře	25

2.3	Vytvoření <code>Form Contextu</code> a <code>form</code> objektu.	27
2.4	Získání ročníků konference při změně výběru.	28
2.5	Logika otevření modálu se souvisejícími publikacemi.	30
2.6	Obalení funkce <code>renderTreeNodeWithActions</code> za účelem předání hooků.	31
2.7	Podmínečný <code>mount</code> a <code>unmount</code> tlačítek.	31
2.8	Implementace odeslání formuláře pro správu kategorií z formuláře.	32
2.9	Odeslání souborů na koncový bod API za účelem nahrání.	33
2.10	Provedení strategie.	34
2.11	Transakce, která zapíše změny souvisejících objektů a vytvoří novou publikaci.	36
2.12	Ilustrace získání dat pro úpravu souvisejících entit.	38
2.13	Vypočítání budoucího indexu ročníku pro navázání a úprava konference.	41
2.14	Implementace mazání souborů na serveru.	43
2.15	Implementace serializace souboru pro stažení ze serveru.	44
2.16	Přečtení složky obsahující soubory publikace a získání jejich statistik.	45
2.17	Optimalizační kroky pro <code>Collapse</code> komponentu. Inspirováno [17].	46
2.18	Propsání úpravy publikace do databáze a manipulace souborů.	47
2.19	Zpracování definice a logika jejího rozeznání typu	48
2.20	Zpracování definice a vytvoření objektu pro další zpracování pomocí knihovny <code>Zod</code>	49
2.21	Vytvoření <code>EndNote</code> export formátu.	53
2.22	Funkce pro vytváření textové citace.	54
2.23	Implementace seznamu konferenčních ročníků.	57
2.24	Vytvoření kontextu pro seznam konferencí a ročníků.	58
2.25	Implementace použití funkce pro získávání dat do seznamu.	59
2.26	Filtrace ročníku podle kategorií.	60
2.27	Logika otevření modálu nebo provedení akce po kliknutí na tlačítko.	61
2.28	Logika dědění kategorií a popisů.	63
2.29	Implementace přepsání funkce a odeslání formuláře.	65
2.30	Kešovaná funkce <code>fetchConferenceCategoryTree</code>	69
2.31	Naplnění dočasné „ <code>new_acm_categories</code> “ tabulky a přemapování <code>parent_id</code> atributu. Naplnění je docíleno za použití <code>Common Table Expression</code> , tedy vytvoření pojmenovaných poddotazů.	70
3.1	Průběh testu úpravy konference a jejího ročníku.	83
E.1	Enumy obsahující mapování <code>Citation.js</code> typů na <code>PubConf</code> typy.	95
E.2	Enum obsahující mapování <code>Citation.js</code> typů na <code>PubConf</code> typy.	96
F.1	Enum obsahující mapování <code>PubConf</code> typů na <code>Citation.js</code> typy.	97
F.2	Enum obsahující mapování <code>PubConf</code> typů na <code>Citation.js</code> typy.	98

Seznam zkratek

ORM	Object–Relational Mapping
ACM	Association for Computing Machinery
CRUD	Create Read Update Delete
DAL	Data Access Layer

Úvod

Vědecké práce, konference a publikace jsou nedílnou součástí každé akademické a výzkumné půdy. Sdílejí se přes ně důležitá data, poznatky a objevy, které posouvají hranice lidského poznání a přispívají k řešení problémů. Vědecké instituce, mezi které patří univerzity, produkují velké množství těchto prací, ať již v rámci vlastního výzkumu, ve spolupráci s ostatními institucemi nebo v návaznosti na předešlé práce. Konference následně nabízejí prostor pro sdílení těchto poznatků a pokroků s jinými akademiky. Proto je třeba mít relevantní publikace a konference na jednom místě, které jednoduše umožňuje správu, uložení a vyhledávání těchto prací.

Takový software pro správu publikací a konferencí již na FIT ČVUT existuje, je ale zastaralý a místy nefunguje, jak by měl. Po správě publikací a konferencí, zejména na univerzitě, je velká poptávka, tedy je na místě tento program zmodernizovat a přepsat do aktuálních technologií. Toto přepracování přinese řadu výhod, zejména z hlediska funkčnosti, uživatelské přívětivosti a bezpečnosti.

Cíle práce

Hlavním cílem této diplomové práce je kompletní přepsání části databáze pro správu publikací a konferencí z technologií nepodporovaných oddělením ICT FIT do technologií jím podporovaných. Kompletního přepracování se dočká jak back-end, tak i front-end aplikace, s důrazem na lepší uživatelskou použitelnost a přívětivost. Důvodem reimplementace je následné převzetí tohoto softwaru do péče oddělení ICT FIT.

Jedná se o týmovou diplomovou práci, kde ve dvoučlenném týmu společně s kolegou Bc. Tomášem Vondrou [1] je kladen důraz na implementaci **základních funkcionalit** celého původního systému, aby se nová aplikace následně dala použít v praxi. Důvodem volby týmové implementace byla náročnost a rozsáhlost celého systému. Více k vymezení práce v sekci níže.

Nejprve je v kapitole 1 představena původní aplikace, její vlastnosti a ne-

dostatky, a následně se provede návrh nové aplikace včetně výběru technologií, návrhu architektury a uživatelského rozhraní.

Dále je v kapitole 2 v prvotní fázi provedena migrace databáze z MySQL do PostgreSQL. Znamená to konverzi všech tabulek, vztahů a integritních omezení do PostgreSQL syntaxe a přesun dat.

Následně je kapitola zaměřena na přeprogramování vytváření publikací s návazností na autory, časopisy, vydavatele a konference. Dále na kompletní správu konferencí a ročníků konferencí, což zahrnuje vytváření, úpravu a mazání konferencí a ročníků a také jejich zobrazení, vyhledávání a filtrování. A v poslední řadě i na implementaci importu publikací pomocí publikačních formátů (BibTeX, RefWorks a EndNote) a exportu publikací v citačních formátech (BibTeX, RefWorks, EndNote, IEEE, ACM a ISO 690).

Výsledná aplikace je funkčně a uživatelsky otestována a seznam nedostatků nalezených při testování bude použit pro návrh budoucích vylepšení aplikace, což lze najít v kapitole 3.

Vymezení vypracovaných částí

V této diplomové práci jsou implementovány části systému pro vytváření, import a export publikací, správu konferencí a ročníků konferencí. Konkrétně se jedná o:

- Publikace:
 - Vytváření publikací s návazností na autory, časopisy, vydavatele a konference.
 - Import publikací pomocí citačních formátů (BibTeX, RefWorks a EndNote).
 - Import publikací pomocí Springer API.
 - Export publikací v citačních formátech (BibTeX, RefWorks, EndNote, IEEE, ACM a ISO 690).
 - Správa souborů spojených s publikací.
- Konference a ročníky konferencí:
 - Vytváření konferencí a ročníků konferencí.
 - Vyhledávání, filtrování a zobrazení konferencí a ročníků konferencí.
 - Úprava, mazání a zobrazení detailů konferencí a ročníků konferencí uživateli.
 - Správa databází dokumentových indexů a navázání na ročníky.
 - Správa konferenčních kategorií s návazností na konference.

Diplomová práce Tomáše Vondry [1] se zabývá vyhledáváním, filtrací a zobrazením publikací, správou autorů, časopisů, publikačních kategorií a uživatelských skupin a správou uživatelů, včetně autentizace a autorizace. Konkrétně se jedná o:

- Publikace:
 - Vyhledávání, filtrace a zobrazení publikací.
 - Správa autorů, časopisů, uživatelských skupin, vydavatelů a publikačních kategorií.
- Uživatelé:
 - Správa uživatelů a jejich přihlášení.
 - Správa uživatelských práv a jejich aplikování.
- Design úvodní stránky a společného rozhraní.
- DevOps pipelines a infrastruktura projektu.

Kapitola 1

Analýza

Následující kapitola se zabývá nejprve obecným shrnutím aplikace, dále návrhem, kde se rozebere volba použitých technologií a architektur, a v poslední řadě představením lo-fi prototypů jednotlivých sekcí a stránek. Analýza se zejména zabývá částmi aplikace, které jsou relevantní k zadání.

1.1 Obecný popis původní aplikace

Aplikace s produkčním názvem *PubConf* je webová platforma pro ukládání záznamů publikací a konferencí spjatých s ČVUT FIT na jednom místě. Software je navržen tak, aby vyhovoval potřebám vedení záznamů o publikacích a konferencích, orientaci v nich a umožňoval exportování (respektive citování) těchto záznamů pro specifickou skupinu lidí. Přístup k nim mají zaměstnanci, doktorandi, studenti a externisté fakulty. Historicky se jednalo o dvě oddělené aplikace, ale z toho důvodu, že měly velký průnik, byly spojeny v jednu.

Dále také ukládá záznamy o záležitostech spojených s publikacemi a konferencemi, tedy v aplikaci jsou drženy záznamy o autorech, vydavatelích a časopisech. Dále jsou to kategorie, kterých mohou různé publikace a konference nabývat pro účely filtrování, záznamy o databázích pro indexování dokumentů, publikační skupiny, které slouží pro hromadné exporty publikací a konferenční skupiny, které slouží pro doporučování konferencí uživatelům.

Aplikace je psána v jazyce PHP ve frameworku Nette. Na front-endu se nachází i JavaScript pro interaktivitu stránky. Databáze používá technologii MySQL. Aplikace je hostována na serveru `ddd.kcn.in.fit.cvut.cz`, nepoužívá žádnou virtualizaci a neběží v žádném kontejneru. Samotná webová aplikace se poté nachází ve složce `/var/www/html/PubConf`. MySQL server, kde se nachází databáze, je hostován také na stejném serveru.

Každá publikace může nabývat jednoho ze 13 možných BibTeX typů uvedených v [2], kromě typu `conference`. Dále si publikace ukládá zbylé informace, jako je například svazek či strany, v ekvivalentních BibTeX attributech (zde by to tedy bylo `volume` a `pages`). Publikaci je možné také rozšířit o vlastní

atributy.

Publikace je možné zobrazovat, filtrovat nebo vyhledávat podle určitých kritérií na hlavním seznamu. Před tímto seznamem je navíc mezistránka, kde si uživatel nakliká tyto filtry a kritéria. Následně jsou publikace předvedeny uživateli v klikatelném IEEE citačním formátu. Po kliknutí na název publikace je uživatel přenesen na její detail, ale citace obsahuje i jiné klikatelné odkazy na související záznamy, například na autory či časopisy, které uživatele přenesou na detail, viz obrázek 1.1.

• R. Monasson et al. "2+p-SAT: relation of typical-case complexity to the nature of the phase transition", *Random Structures & Algorithms*, vol. 15, no. 3-4, pp. 414-435, 10, 1999

• I. Pomeranz, S. M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits", *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 7, pp. 1050-1058, 7, 1993

■ Obrázek 1.1 Příklad klikatelného záznamu v seznamu publikací.

Na detailu publikace se nachází více záložek, kromě záložky obsahující informace je k dispozici také záložka s exporty, referencemi (publikace, které cituje) a citacemi (publikace, ve kterých je citována). Na první záložce detailu lze vidět informace o publikaci a lze přidávat, modifikovat a mazat anotace a štítky. Z detailu lze navigovat na úpravu publikace.

Jednotlivé publikace je možné také importovat pomocí citačních formátů, mezi které patří *BibTeX*, *EndNote tagged file* [3] a *RefWorks tagged format* [4]. V původní aplikaci ale tato funkcionality je omezena, nehlásí chybové hlášky při špatné syntaxi a pokud obsahuje špatný formát, tak aplikace vyhodí error 404.

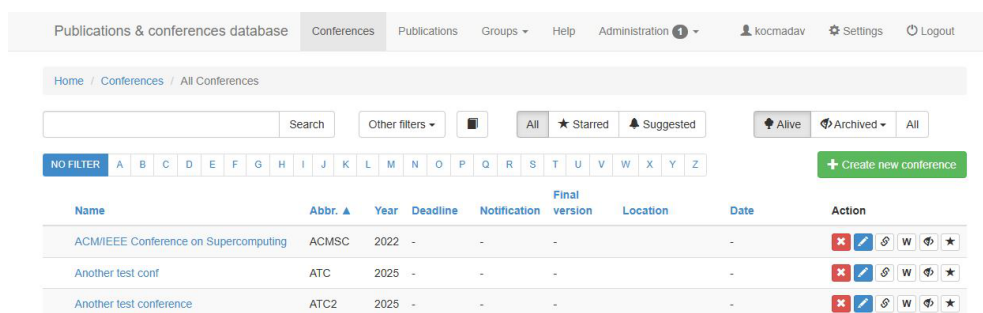
Export těchto publikací, nacházející se na jejich detailu, je možný v různých citačních formátech. Kromě formátů zmíněných výše při importu, export obsahuje i textové formáty IEEE, ACM a ISO 690.

K dispozici je také možnost získání článku ze Springer API [5] pomocí jedinečného DOI, ISSN nebo ISBN identifikátoru. V tomto případě je ale tato funkcionality v původní aplikaci kompletně nedostupná, pravděpodobně kvůli propadlému Springer API klíči.

Konference si u sebe ukládají jméno, zkratku a datum prvního ročníku a slouží zejména jako „seskupovatel“ jednotlivých ročníků této konference. Každý ročník si ukládá kromě jména a zkratky i rok, kdy se odehrával. Dále mimo jiné k těmto atributům patří i datum konání, vydavatel spojený s tímto ročníkem, kategorie a v jaké databázi pro indexování dokumentů je zahrnut. Ročníky konference si také mohou vést informaci o tom, jaké jiné ročníky menší konference jsou k nim přidružené, tedy vedou si záznamy o *workshopech*. Konference mohou být ve stavu **alive**, tedy ještě probíhají a jsou aktivní, nebo **dead**, což znamená, že již jsou neaktivní. Ročníky také mohou být ve stavu **alive**, tedy probíhají nebo probíhat budou, nebo ve stavu **archived** – již proběhly.

Jednotlivé ročníky konferencí je možné vyhledávat a filtrovat podle jména, zkratky, roku, datumů (jedná se zejména o **Deadline**, **Notification**, **Final**

version a datum konání), lokace, kategorií a grup. Je možné si zobrazit také oblíbené, archivované a živé ročníky. Aplikace seznam ročníků konferencí zobrazuje pod cestou Home/Conferences/All Conferences, což může být matoucí, jak lze vidět na obrázku 1.2. Seznam samotných konferencí pak lze najít v administrativě, kam se ale dostane jen administrátor.



Obrázek 1.2 Seznam ročníků konference v původní aplikaci. V levém horním rohu lze vidět zavádějící cesta. Také horní lišta říká, že se jedná o stránku s „konferencemi“.

Vytváření a editace konferencí a jejich ročníků je koncipováno jako modál a v jednom okamžiku nelze skupinově zadat více ročníků.

Detail jednotlivých ročníků v sobě obsahuje všechny potřebné informace z databáze. Nechybí přehled ostatních ročníků, pokud existují, workshopů, metadat (tedy kdo ročník založil a kdy ho naposledy upravil) a také je k dispozici záložka se souvisejícími publikacemi tohoto ročníku (opět ve formátu IEEE). Detail pro samotnou konferenci v aplikaci ale neexistuje.

V poslední řadě uživatelé mohou nabývat následujících rolí:

- Administrator – má neomezený přístup a pravomoce.
- Conference user – může číst a vyhledávat konference.
- Conference moderator – může navíc provádět správu konferencí.
- Publication reader – může číst a vyhledávat publikace.
- Publication submitter – může navíc provádět správu publikací.

1.2 Návrh

Následující kapitoly projednávají volbu technologií použitých při implementaci, navrženou architekturu a v poslední řadě návrh a lo-fi prototypy uživatelského rozhraní.

1.2.1 Volba technologií

Zadání definuje požadavek přepsat aplikaci do technologií podporovaných oddělením ICT FIT. Níže jsou citovány podmínky relevantní k této práci, které se nacházejí na stránce pokynů oddělení pro převzetí externích softwarů (stránka se nachází za přihlášením):

- *Aplikace je napsaná v jazyku JavaScript/TypeScript. Výjimky podléhají rozhodnutí vedoucího ICT.*
- *Pokud používá relační databázi, tak PostgreSQL nebo SQLite (MySQL/MariaDB není podporovaná). Výjimky podléhají rozhodnutí vedoucího ICT.*
- *Aplikace je verzovaná nástrojem git a projekt s aplikací je umístěn na fakultním GitLabu.*

Tedy povolenými jazyky implementace jsou JavaScript a jeho nadmnožina TypeScript. Nicméně prvotní myšlenkou architektury bylo napsat back-end v jazyce Java a frameworku Spring a frontend v Reactu. Spring byl vybrán z toho důvodu, že by back-end byl zamýšlen jako REST API, což by například nabízelo i budoucí integraci s vícero systémy. Jak je výše v odrážkách řečeno, „*Výjimky podléhají rozhodnutí vedoucího ICT.*“, ale s panem vedoucím jsme se bohužel nestihli spojit včas, takže jazyk TypeScript se stal finální volbou.

Každý z nás navíc bude své části vyvíjet „vertikálně“, tedy full-stackově. Důvodem zvolení full-stacku bylo vyvarování se situací, kdy by jeden člen týmu musel čekat na dokončení práce toho druhého. Tento přístup umožňuje každému z nás pracovat na všech vrstvách aplikace vlastním tempem. Dalším rázným důvodem byl fakt, že back-endové logiky v aplikaci je méně než logiky na front-endu.

Z tohoto důvodu jsme zvolili pro vývoj aplikace moderní framework Next.js [6]. Jedná se o full-stack framework postavený na Reactu pro tvorbu webových aplikací a oproti jiným frameworkům nabízí optimalizovanější a výkonnější produkty. Mezi hlavní výhody tohoto frameworku, které velmi ovlivnily naši volbu, patří [6]:

- **Server Side Rendering (SSR)** – jedná se o generování HTML stránky na serveru při každém požadavku. Tato vlastnost zajišťuje méně práce pro klienta, protože dostane už vygenerovanou stránku. Díky SSR také můžeme přesunout získávání dat na server blíže ke zdroji, což má za důsledek optimalizaci a snížení požadavků, které musí klient vykonat.
- **Získávání dat** – Next poskytuje vylepšenou **fetch** API rozšířenou o kešování a takzvané **Server actions**, což jsou asynchronní funkce vykonávané na serveru.
- **TypeScript** – Next poskytuje efektivnější kompilaci a lepší kontrolu typů.

Jak bylo zmíněno, Next.js je postaven na Reactu, což je velmi populární front-end framework, který nám umožňuje vytvářet dynamické a interaktivní webové aplikace. Abychom docílili skvělého uživatelského a vývojářského zážitku, používáme Reactí komponentovou knihovnu Mantine [7]. Tato knihovna poskytuje širokou škálu komponent a stylů, což nám usnadňuje vývoj uživatelského rozhraní.

Dále požadavky definují použití databázových technologií SQLite nebo PostgreSQL. Zde volba byla jasná a zvolili jsme PostgreSQL. Důvod byl zejména ten, že PostgreSQL je komplexnější databázový systém, který je vhodný pro silný provoz, tedy se více hodí pro webové aplikace [8].

Pro napojení a komunikaci aplikace s databází a zjednodušení práce s ní jsme se rozhodli pro použití Prisma ORM [9]. Prisma je nástroj pro zjednodušenou správu databáze za pomoci intuitivního datového modelu, automatických migrací a možností psaní dotazů přímo v kódu, protože umožňuje psát dotazy jako vnořené objekty.

Celá aplikace bude verzována v novém GitLab repozitáři, kde se také budou nacházet CI/CD pipelines.

Nová aplikace bude hostována na stejném serveru definovaného adresou `ddd.kcn.in.fit.cvut.cz`, ale nebude uložena jako dosavadní aplikace, bude virtualizovaná v Docker kontejneru. V tomto kontejneru bude také běžet PostgreSQL server s databází.

1.2.2 Architektura

Bude se opět jednat o webovou aplikaci, tedy původní koncepce zůstane stejná. Tentokrát ale v aplikaci nebude použita žádná z populárních architektur, jako v té původní, tedy žádný Model-View-Controller nebo Model-View-Presenter, ale bude se dodržovat konvence definovaná Next.js.

Next.js používá *file-system based routing*, což znamená to, že cesta v souborové struktuře definuje cestu stránky na webu, respektive složky souborového stromu definují segmenty cesty a ty se mapují na segmenty URL. V nejnovější verzi 15 je tento routing známý jako **App Router**, tedy ve složce s názvem `app/` jsou uloženy všechny stránky a tato složka slouží jako relativní kořenová složka stránek. Tento způsob i definuje, kdy je souborová cesta platnou cestou na webu a uživateli vykreslí UI – pokud segment obsahuje soubor `page.tsx` [6, sekce Layouts and Pages]. Zmíněná konvence navíc umožňuje příjemnější organizaci velkého projektu, jako je například tento.

Samozřejmě **App Router** nabízí i dynamické parametry, nazývané **slugs**, které mají jmennou konvenci následující: `[nazev_slozky]`. Tyto segmenty se používají pro získávání parametrů z URL, které se mění, nejčastěji identifikátory zdrojů [6, sekce Layouts and Pages].

Samotný obsah stránky bude tedy vykreslen ze souboru `page.tsx`, který může obsahovat další komponenty v sobě. Segment může obsahovat i soubor `layout.tsx`, který definuje jednotné rozhraní pro všechny stránky v dětských

segmentech.

Pokud si uživatel bude chtít zobrazit na webu stránku

```
/publications/122/edit,
```

pak tedy soubor obsahující stránku, která se uživateli vykreslí, bude definován cestou (kromě `page.tsx` se jedná o názvy složek v souborovém systému):

```
/app/publications/[id]/edit/page.tsx.
```

Složka `app/` se nachází v kořenové složce projektu.

Pro manipulaci s daty či jejich získávání budou sloužit **Server actions**, což, jak bylo zmíněno výše, jsou funkce prováděné na serveru, volané buď z klientských, či serverových komponent¹ (částí UI). Tyto akce budou tvořit back-end a budou obsahovat přímý přístup do databáze a náročnou výpočetní logiku.

V původní aplikaci se hojně používá OOP, TypeScript sice OOP jazykem je, ale ve webové architektuře se spíše používá koncept modulů pro deklaraci funkcionalit, respektive **import/export** syntax. Každý soubor, který obsahuje importy nebo exporty na nejvyšší úrovni, je považován za modul. Moduly jsou oddělené a mají vlastní rozsah, což znamená, že proměnné, funkce, třídy a další deklarace uvnitř modulu nejsou viditelné mimo něj, pokud nejsou explicitně exportovány [10].

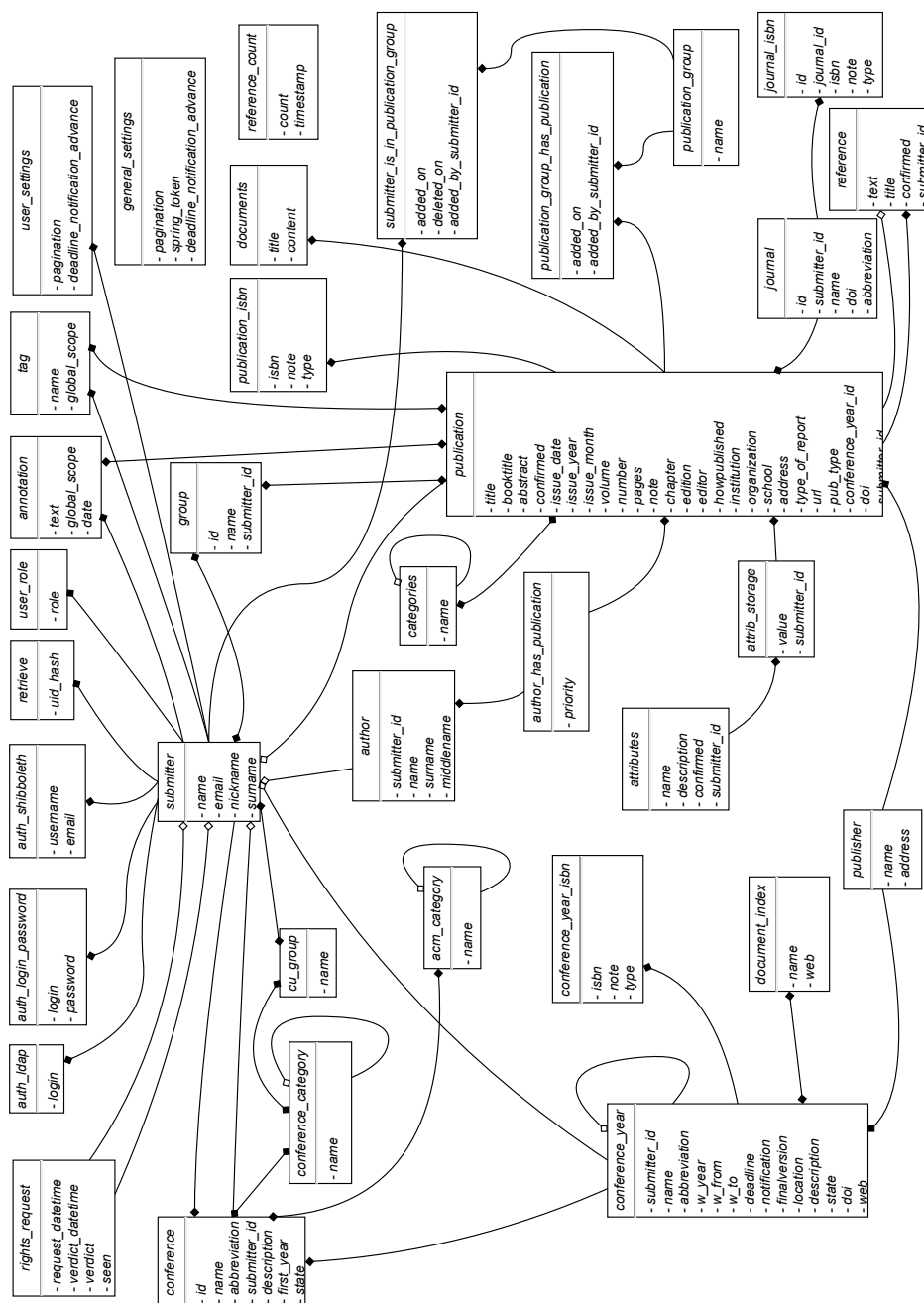
Tedy nebudou existovat CRUD třídy, jako v původní aplikaci, ale soubory s CRUD funkcemi, které jsou poté exportovány a použity v kódu.

Pár tříd ve spojení se dvěma návrhovými vzory bude v kódu použito. Konkrétně se bude jednat o návrhové vzory **Strategy** [11], pro změny typu publikace před zapsáním do databáze, a **Builder** [12] pro postavení objektu, který bude obsahovat data pro export publikace.

1.2.3 Databáze

Jak bylo již několikrát avizováno, bude třeba migrace databáze z MySQL do PostgreSQL. K tomuto bude použit nástroj **pgloader**, který slouží právě k migraci databázových schémat a dat do PostgreSQL [13]. Datový model bude jen zlehka upraven, což bude popsáno v sekci 1.3. Zbylé úpravy nebudou velké, bude se jednat především o přidávání integračních omezení, kaskádové mazání a přejmenování políček či modelů (tabulek). ER diagram databáze původní aplikace se nachází na obrázku 1.3 a je převzat z [14].

¹Komponenta je klientská, pokud její předvykreslení proběhne na serveru a na klientovi umožní interaktivitu za pomoci JavaScriptu [6, sekce Rendering/Client Components], serverová komponenta je celá vykreslena na serveru a finální HTML obsah je odeslán klientovi [6, sekce Rendering/Server Components].



Obrazek 1.3 ER diagram databáze původní aplikace před změnami. Převzato z [14].

1.2.4 Návrh uživatelského rozhraní

Největší důraz je třeba věnovat front-endu nové aplikace. Oproti původní aplikaci je nutné vylepšit jak funkcionální, tak i stylistické aspekty. Například v původní aplikaci byly popisky vstupních polí vlevo, namísto nad nimi, což ztěžuje čitelnost, jak ukazuje obrázek 1.4, nebo některým akcím chybí odezva, zda-li proběhly úspěšně či neúspěšně. Cílovým jazykem aplikace zůstane angličtina.

The image shows a portion of a web form. It has two input fields. The first field is labeled 'Type of publication*' on the left side. The second field is labeled 'Title*' on the left side. Both labels are positioned to the left of the input fields, which is noted as a poor design choice in the text.

■ **Obrázek 1.4** Neideální pozice popisku vstupního pole v původní aplikaci.

1.2.4.1 Vytváření publikací

Pro potřeby vytváření publikací s návazností na autory, vydavatele, časopisy a konference je navržen formulář, jehož obsah se bude měnit na základě zvoleného typu publikace. Bude se skládat ze tří částí:

- **Obecné a povinné atributy** – tato sekce bude obsahovat důležité informace sdílené všemi typy, jako je například název, autoři, rok či samotný typ.
- **Volitelné atributy** – tato sekce bude obsahovat informace, které jsou volitelné pro každý typ, jako například svazek, adresa vydavatele či organizace, která vydala publikaci.
- **Vlastní atributy** – poslední sekce obsahuje prostor pro vytváření, editaci a vyplňování vlastních atributů.

Lo-fi návrhy formuláře pro vytváření publikací mohou být nalezeny na obrázcích 1.5 a 1.6. Vstupní pole mají označení nad nimi pro lepší čitelnost. Povinné atributy jsou vyznačeny červenou hvězdičkou vedle názvu pole. Každá část je vizuálně rozdělena horizontálními liniemi a finální tlačítko pro odeslání formuláře je znázorněno dole. Pokud selže validace formuláře, tedy v nějakém políčku je nepovolená hodnota či povinné pole je prázdné, celé políčko se zabarví červeně a vypíše se pod ním hláška. Toto chování je základní chybné chování komponenty `Input` knihovny Mantine [7, sekce Core/Input].

Pro lepší orientaci uživatele budou k dispozici také tzv. „breadcrumbs“, neboli drobečková navigace. Ta znázorňuje cestu, na které se uživatel aktuálně nachází. Tato navigace bude všudypřítomná na všech stránkách.

Uživateli bude vždy po nějaké akci poskytnuta zpětná vazba. Po úspěšné či neúspěšné tvorbě publikace se ukáže notifikace s odpovídající zprávou. Pokud bude uživatel cokoli na formuláři mazat (například nově vytvořeného autora) ukáže se mu potvrzovací modál.

The screenshot shows the 'Create a publication' form. At the top, there's a navigation bar with 'Home / Publications / New'. The form is titled 'Create a publication' with a help icon. It is divided into two main columns. The left column contains a 'Title' field (marked as required), an 'Abstract' field, a 'Categories' dropdown (with an 'Add new category' button), a 'DOI' field (with a 'wrong format' error), and a 'Pages' field. The right column contains a 'Type' dropdown (with a 'Select type' placeholder), an 'Author' dropdown (with 'Edgar Allan Poe' selected), 'Month of publication' and 'Year of publication' dropdowns (both set to 'January' and '2024' respectively), a 'Groups' dropdown (with an 'Add new group' button), an 'ISSN/ISBN' field (with a 'wrong format' error), a 'URL' field (with a 'wrong format' error), and a 'Note' field. At the bottom, there is a 'Create' button.

Obrázek 1.5 První část formuláře pro vytváření publikací. Lze zde vidět formát chybových hlášek.

The screenshot shows the second and third parts of the 'Create a publication' form. The top part is a large 'Note' field. Below it is a 'File' section with a 'Drag & Drop File or select' button. The bottom part is the 'Further attributes' section, which includes an 'Attribute1' field and a 'Create' button. The footer of the page reads '2024, FIT CTU'.

Obrázek 1.6 Druhá a třetí část formuláře pro vytváření publikací.

Tento formulář byl v rámci předmětu Návrh uživatelského rozhraní (NINUR) uživatelsky testován a vzniklá připomínka, která byla zapracována do tohoto formuláře, byla také použita při návrhu formuláře pro vytváření konferencí a ročníků konferencí. Uživatelé zmínili, že by bylo dobré, aby se daly druhá a třetí část složit pro ušetření místa.

Pro navigaci na tento formulář je navrženo tlačítko, které se bude nacházet na hlavním seznamu publikací. Tlačítko se skládá ze dvou částí, nejvíce viditelná část naviguje na samotný formulář a druhá část obsahuje rozbalovací nabídku, jejíž možností otevřou modál buď pro import publikace z definice, nebo pro import ze Springer API. Design tlačítka může být nalezen na obrázku 1.7.



■ **Obrázek 1.7** Design multifunkčního tlačítka.

Modál pro import přes definici bude obsahovat pole pro vyplnění textu citační definice a poté tlačítka pro import či zrušení akce. Oproti původní verzi aplikace nebude mít pod sebou tři přepínače pro výběr, o jakou definici se jedná, ale typ definice se bude automaticky rozpoznávat. Chybné hlášky se budou vypisovat hned pod vstupní pole. Design modálového okna je znázorněn na obrázku 1.8.

Modál pro import přes Springer API je již trochu komplexnější. Obsahuje jedno menší vstupní pole pro zadání jedinečného identifikátoru a jedno větší vstupní, respektive neměnné výstupní pole pro zobrazení Springer API odpovědi. K dispozici jsou tři tlačítka – pro zrušení operace, „vyzkoušení“ identifikátoru a importu získaných dat. Tlačítko pro „vyzkoušení“ identifikátoru provede dotaz na koncový bod API, získá data a ukáže navrácená data ve výstupním poli pro kontrolu uživatelem. Dokud uživatel nenatáhne nějaká data ze Springer API, nebude mu umožněn import. Chybové hlášky se vypíší pod výstupním polem. Design modálového okna je znázorněn na obrázku 1.9.

V původní verzi aplikace tento modál opět obsahuje přepínače pro určení typu identifikátoru. Zde se typ nebude nějak automaticky rozpoznávat, ale bude vyžadováno, aby uživatel sám zadal, o jaký typ se jedná. K určení typu poslouží prefixy `doi:`, `issn:` a `isbn:`. Více k chování tohoto modálu v sekci 2.4.5.



Create a publication from one of the supported formats

Input text *

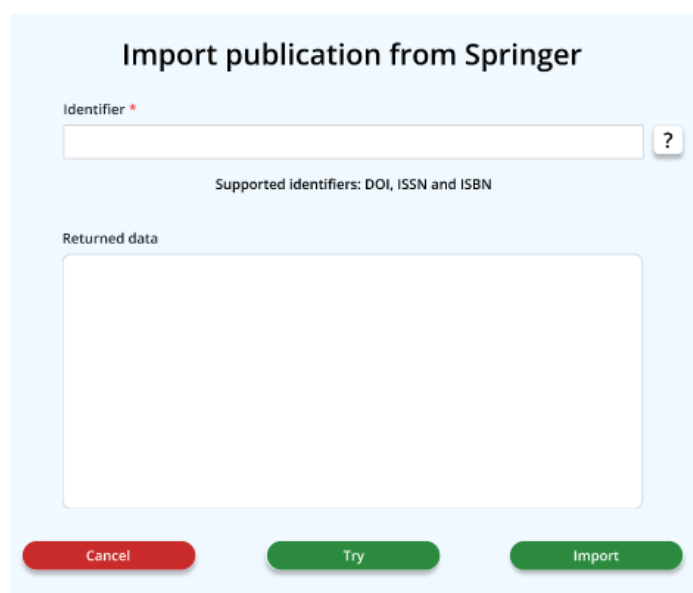
?

Supported formats: BibTeX, RefWorks, EndNote

Cancel Import

This is a design of a modal window. It has a light blue background. At the top, it says "Create a publication from one of the supported formats". Below this is a large white text input area with a red asterisk and a question mark icon. Underneath the input area, it lists "Supported formats: BibTeX, RefWorks, EndNote". At the bottom, there are two buttons: a red "Cancel" button and a green "Import" button.

■ **Obrázek 1.8** Design modálu pro import publikace pomocí citačního formátu.



Import publication from Springer

Identifier *

?

Supported identifiers: DOI, ISSN and ISBN

Returned data

Cancel Try Import

This is a design of a modal window. It has a light blue background. At the top, it says "Import publication from Springer". Below this is a white text input field with a red asterisk and a question mark icon. Underneath the input field, it lists "Supported identifiers: DOI, ISSN and ISBN". Below this is a large white text area labeled "Returned data". At the bottom, there are three buttons: a red "Cancel" button, a green "Try" button, and a green "Import" button.

■ **Obrázek 1.9** Design modálu pro import publikace přes Springer API.

1.2.4.2 Konference

Zobrazení konferencí, oproti publikacím, bude mít dvě části, mezi kterými se bude přepínat pomocí záložek. První záložka bude obsahovat seznam všech konferencí, druhá seznam všech jejich ročníků. Pod záložkami se bude nacházet vyhledávací pole, které bude filtrovat aktuální seznam. Samotný seznam pro konference a ročníky konferencí bude stejný, jen se bude lišit v počtu sloupců. Tato stránka bude obsahovat také tlačítko pro navigaci na formulář pro vytváření konferencí.

Pod vyhledávacím polem se budou nacházet dodatečná tlačítka pro filtrování konferencí a ročníků. Na obrázku 1.10 lze vidět pohled na konference, k nimž případnou skupiny tlačítek, kde první skupina bude obsahovat tyto možnosti:

- všechny konference,
- živé konference,
- mrtvé konference.

Druhá skupina bude obsahovat následující možnosti:

- všechny konference,
- oblíbené konference,
- doporučené konference – dle toho, do jaké konferenční skupiny uživatel patří.

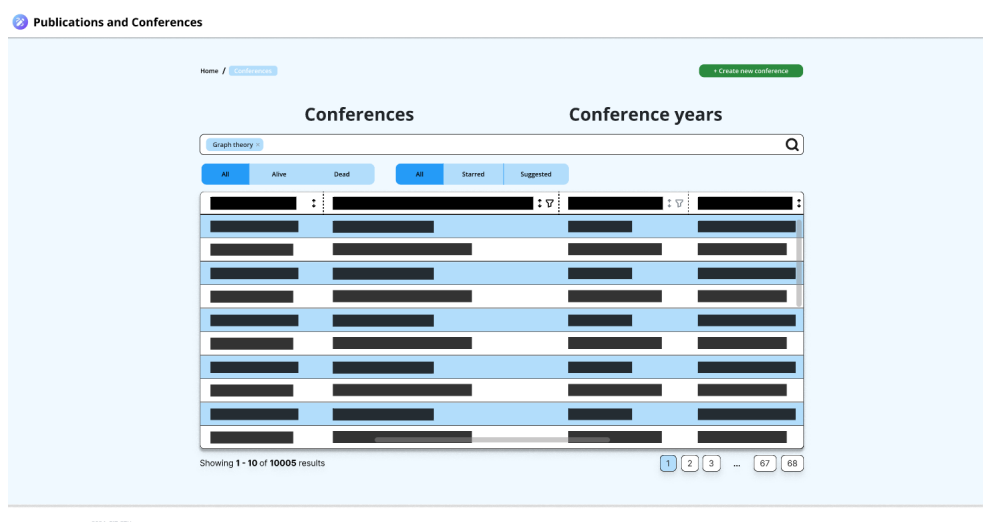
U ročníků konferencí již bude možností více, skupiny ale budou obsahovat filtry podobného duchu, jako jsou u konferencí. První skupina tlačítek se bude skládat z:

- všechny ročníky,
- živé ročníky,
- archivované ročníky,
- poslední ročníky archivovaných ročníků.

Druhá skupina tlačítek je identická s druhou skupinou tlačítek u konferencí. Poslední tlačítko bude sloužit pro filtrování kategorií konferenčních ročníků.

Toto řešení vzniklo na bázi jedné z GitLab Issues (viz 2.3, #165), která požadovala zpřístupnění konferencí pro všechny uživatele. Proto bylo rozhodnuto, že jak konference tak i ročníky budou na jedné stránce a bude možno se mezi nimi jednoduše přepínat. Tento design lze vidět na obrázku 1.10.

Pro vytváření konferencí a ročníků konferencí bude k dispozici formulář, který bude rozdělen na dvě části:



Obrázek 1.10 Design seznamu konferencí a ročníků konferencí.

- První část bude dedikována samotné konferenci, design této části lze vidět na obrázku 1.11.
- Druhá část bude obsahovat seznam souvisejících ročníků.

Jak bylo zmíněno, v původní aplikaci nejde vytvářet více ročníků naráz. Zde byl formulář navržen tím způsobem, aby tento nedostatek vyřešil a to tak, že druhá část bude obsahovat seznam ročníků. Každý záznam v seznamu bude v podobě pilulky, která se bude dát roztáhnout, aby šlo vyplnit informace ročníku. Návrh seznamu lze vidět na obrázku 1.12. Na každé pilulce bude rok a tlačítko pro smazání záznamu. Nový ročník bude možné přidat po kliknutí na tlačítko + vedle nadpisu sekce.

Na konci formuláře se budou nacházet tlačítka pro odeslání či pro zrušení vytváření konference a ročníků.

Při mazání ročníku ze seznamu vyskočí na uživatele potvrzovací modál, zda-li opravdu chce tento ročník smazat. Zbývá mechanika zpětné vazby je stejná jako u formuláře pro vytváření publikací.

Tento formulář bude použit také pro úpravu konference a jejích ročníků.

Detail bude mít jak samotná konference, tak samotné ročníky. Na detailu budou obsaženy všechny dostupné informace, informace o tom, kdo záznam vytvořil, a kdo a kdy záznam naposledy upravil. K dispozici také budou tlačítka na navigaci na úpravu konference a smazání konference.

Každý detail bude členěn na dvě části: první část obsahuje tedy samotný detail a druhá část bude obsahovat seznamy. Seznamy mohou nebo nemusí být navigovatelné (nenavigovatelné platí pouze pro ISSN a ISBN identifikátory ročníku). Navigovatelné seznamy znamenají, že po kliknutí na pilulku se uživatel přesune na adresu definovanou v tlačítku.

The screenshot shows the 'Create a conference' form, specifically the 'General' section. The title 'Create a conference' is centered at the top. Below it, the 'General' section is marked with a red asterisk. It contains three input fields: 'Name *', 'Abbreviation *', and a larger 'Description' field. At the bottom of this section, there are two buttons labeled 'ACM Categories' and 'Custom Categories', each followed by a green plus icon.

■ **Obrázek 1.11** Design formuláře pro vytváření konferencí. Na obrázku lze vidět sekci pro vyplnění informací samotné konference.

The screenshot shows the 'Years of conference' section, marked with a red asterisk and a green plus icon. It is a light blue box with a 'Title' dropdown and a red 'X' icon in the top right corner. Inside, there are several input fields: 'Year *', 'From - To', 'Submission deadline', 'Notification', 'Final version', 'DOI', 'Web', 'Location', and a large 'Description' field. Below these are 'Publisher' and 'Indexed at' dropdowns. At the bottom of the box, there is an 'ISSN/ISBN' section with a green plus icon, followed by three 'Name *' input fields and a red 'X' icon. Below the light blue box, there is a 'Title >' dropdown. At the very bottom, there are two buttons: 'Cancel' (red) and 'Create' (green).

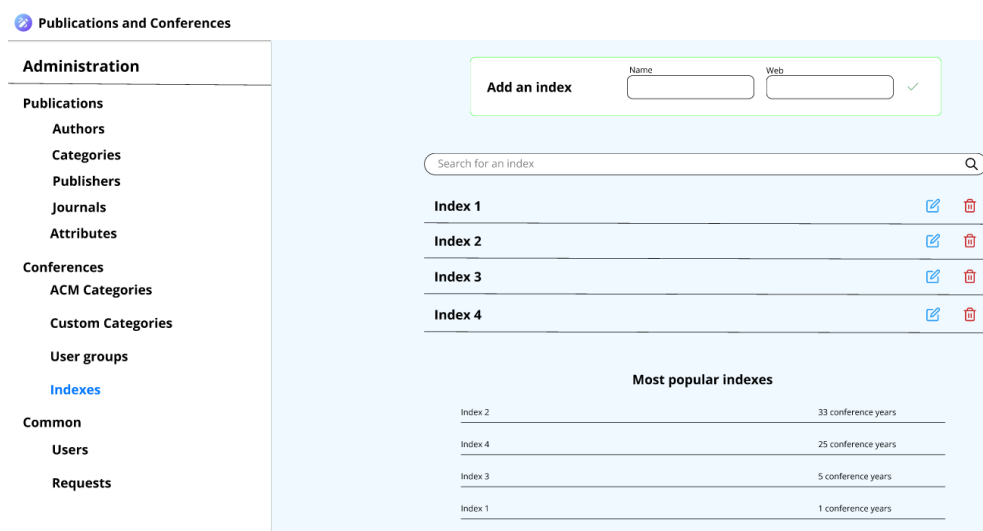
■ **Obrázek 1.12** Design formuláře pro vytváření konferencí. Na obrázku lze vidět sekci pro vytváření jednotlivých ročníků konferencí.

Detail konference bude v druhé části obsahovat pouze navigovatelný seznam všech ročníků té konference. Na druhou stranu, detail ročníku konference bude obsahovat nenavigovatelný seznam identifikátorů a navigovatelné seznamy workshopů, ostatních ročníků konferencí a navíc seznam souvisejících publikací. Design detailu ročníku konference lze vidět na obrázku 1.13.

Obrázek 1.13 Design detailu ročníku konference.

V poslední řadě je třeba představit design správy databází dokumentových indexů a konferenčních kategorií. Design vychází z návrhu stránek správy autorů či časopisů kolegy [1], z důvodu zachování jednotného designového stylu. Na správu indexů se půjde dostat proklikem z postranního panelu, zde uvidí uživatel seznam všech indexů a možnost přidat nový záznam. Také bude k dispozici seznam populárních indexů, tedy které indexy mají nejvíce vztahů. Po kliknutí na záznam v tabulce indexů se uživatel naviguje na detail konkrétního indexu, kde jej může editovat či smazat a uvidí zde seznam ročníků konferencí, se kterými je asociován. Design úvodního seznamu indexů lze nalézt na obrázku 1.14.

Design a funkcionality všech konferenčních kategorií bude identická s databázemi dokumentových indexů.



Obrázek 1.14 Design seznamu databází dokumentových indexů. Vychází z designu ostatních stránek správy sekundárních entit [1].

1.3 Požadavky na novou funkcionalitu

Oproti původní aplikaci bude provedeno pár změn v logice a datovém modelu konferencí a konferenčních ročníků.

V první řadě si konference nově budou udržovat historii o předešlých jménech a zkratkách a také podle historických jmen a zkratek mohou být vyhledávány. Bude možné se vrátit na jedno z předešlých jmen v úpravě konference (operace **undo**).

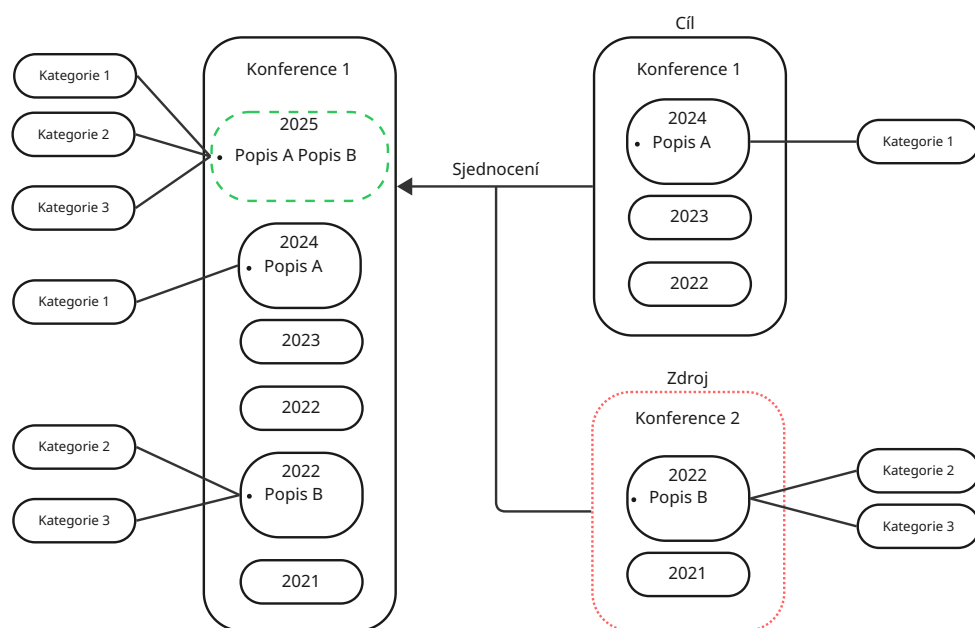
Dále si konference již nebude udržovat její popis, ale místo něj se vyskytne web konference. Webové stránky a popisy v ročnících zůstanou.

Následně všechny kategorie, tedy ACM kategorie a vlastní kategorie, se sjednotí do jedné tabulky `Conference_category` a místo konferencí se budou přiřazovat jednotlivým ročníkům konference. Ilustrace změny databázového modelu může být viděna na obrázku 1.15.

Dále se změní logika sjednocování konferencí, tedy když jedna pohltí druhou. V původní aplikaci z kódu vyplývá, že tato logika byla přímočará, tedy cílová konference přebrala všechny ročníky, kategorie a zbylé entity. Bohužel ale tato funkcionalita nejde v aplikaci otestovat, neboť kliknutí na tlačítko sjednocení konferencí vyhodí HTTP error 500.

Nový požadavek na logiku bude vycházet z původní, tedy cílová konference přebere všechny původní entity zdrojové konference, navíc ale se sjednotí historie jmen a zkratek a přiřadí se této cílové. Úplně pozměněnou logiku očekávají kategorie a popisy ročníků. Jak bylo výše zmíněno, kategorie teď budou jedny a budou se uchovávat u ročníků.

Tedy kategorie a popisy u ročníků, které vznikly před sjednocením kon-



Obrázek 1.16 Ilustrace sjednocení dvou konferencí. Sjednocená konference nabyde ročníky obou konferencí. Nový ročník sjednocené konference (2025) nabyde všech kategorií, které měli poslední ročníky cílové a zdrojové konference (2022 a 2024) a také nabyde spojeného popisu z těchto let. Zdrojová konference je z databáze smazána.

posledního ročníku cílové konference.

Požadována je také implementace některých z **Issues**, které byly vytvořeny na GitLabu původní aplikace. Seznam těchto **Issues** lze nalézt v sekci 2.3.

1.4 Dokumentace

Součástí práce má být i uživatelská a programátorská dokumentace. Jako programátorská dokumentace slouží celá kapitola 2 a také komentáře v samotném kódu.

Uživatelskou dokumentaci lze najít v příloze G, která obsahuje návod pro uživatele, jak používat aplikaci *PubConf*.

[illegible]

Implementace

Následující kapitola obsahuje důkladný popis implementace části aplikace. Kapitola má lehký charakter programové dokumentace pro příští ročníky. Prvotně se popisuje způsob migrace databáze z MySQL do PostgreSQL, následně implementace části s publikacemi a nakonec správa konferencí.

2.1 Struktura projektu

Níže je struktura klíčových složek první hloubky projektu:

```
pubconf-2.0
├── app..... stránky aplikace
├── auth..... autentizační služba
├── components.....sdílené komponenty mezi všemi sekcemi
├── hooks..... sdílené vlastní hooks
├── lib..... back-end služby aplikace a DAL
├── prisma.....obsahuje Prisma schéma a složku s migracemi
└── public.....statické soubory
```

Složka `lib/` obsahuje moduly implementující back-endovou logiku pomocí `Server actions`. Moduly s CRUD operacemi se nachází ve složce `lib/prisma`. Tato složka představuje DAL této aplikace, neboť zde program sahá přímo do databáze.

Stránky aplikace PubConf jsou ve složce `app/` a rozděleny jsou následovně.

```

app
├── conferences ..... konference
├── home ..... domovská stránka
├── publications ..... publikace
├── users ..... uživatelský profil
├── api ..... API endpointy
└── providers ..... poskytovatelé kontextu

```

Tyto složky a jejich podsložky mohou obsahovat složku `components/`, které obsahují front-end komponenty té sekce.

2.2 Migrace databáze

Prvotním úkolem je migrace databáze z technologie MySQL do technologie PostgreSQL. Tento krok je nezbytné udělat jako první, abychom měli k dispozici funkční databázi a data pro testování. Jak bylo zmíněno v sekci 1.2.3, k migraci byl použit nástroj **pgloader** [13], který je právě pro migrace databází do PostgreSQL určen.

První migrace byla pro jistotu provedena lokálně, tedy na serveru byla vytvořena kopie celé MySQL databáze, následně stažena a vložena do lokální instance MySQL. Původní databáze ale obsahuje několik integračních omezení, která jsou porušena. Jedná se o:

- Špatné formáty dat – některé záznamy obsahují datum ve formátu YYYY-00-00, jak je ukázáno v kódu 2.1.
- Odkazy na neexistující záznamy:
 - Publikace obsahují cizí klíč odkazující na záznam časopisů nebo vydavatelů, který v databázi neexistuje.
 - Některé záznamy v tabulce **ACM Category** odkazují na rodičovskou kategorii s id 0, která v databázi neexistuje.
 - Tabulka **Documents** obsahuje záznamy odkazující na neexistující publikace.

```
1 mysql> SELECT issue_date FROM publication WHERE id = 438;
2 +-----+
3 | issue_date |
4 +-----+
5 | 2002-00-00 |
6 +-----+
7 1 row in set (0.00 sec)
8
9 mysql> UPDATE publication SET issue_date = '2002-01-01'
10 WHERE id = 438;
```

■ **Výpis kódu 2.1** Špatný formát data u publikace s id 438 a možnost opravy.

Tedy těsně před migrací je třeba přímý zásah do databáze pro úpravu těchto porušených integračních omezení. Za tímto účelem byl vytvořen skript v jazyce **Python**, který podle uvedeného přepínače zkontroluje integritu databáze, nebo smaže soubory neexistujících publikací, nebo obojí.

Skript má následující přepínače:

- d: jméno databáze
- p: heslo k databázi

```
-u: databázové uživatelské jméno
-y: automaticky odpoví "ano" na všechny výzvy
-h: ukáže nápovědu
--delete-pub-pdfs: zkontroluje složku pro ukládání souborů
    a smaže všechny soubory publikací, které již nejsou
    v databázi
--check-db-integrity: zkontroluje a opraví všechny
    porušení integritních omezení
```

Po úpravě porušení integritních omezení je možné zmigrovat databázi. Použito bylo základní nastavení nástroje `pgloader` [15]:

```
./pgloader mysql://user:password@localhost/PubConf
postgres://user:password@localhost:5432/pubconf
```

Tento příkaz provede přeměnu databázových schémat ze syntaxe MySQL do syntaxe PostgreSQL, vytvoří v cílové databázi náležité tabulky, vztahy a omezení a pomocí PostgreSQL protokolu `COPY` přesune všechna data [13].

Na serveru se skript pro opravu původní databáze nachází ve složce

```
/var/www/html/PubConf2
```

pro finální a případné budoucí migrace a není verzován, protože by se zbytečně v kontejneru při každém přenosu musel instalovat Python interpret. Tato složka také obsahuje složku pro ukládání záloh, která hlavně obsahuje prvotní (a zároveň aktuální) PostgreSQL `dump` databáze.

Jak bylo zmíněno v sekci 1.2.2, pro komunikaci s databází slouží Prisma ORM, která datový model databáze `pubconf` abstrahuje do souboru `prisma/schema.prisma`. Tento soubor obsahuje modely, což jsou abstrakce tabulek v databázi, a definice vztahů a integritních omezení. Složka `prisma` také obsahuje složku s migracemi, což jsou SQL skripty obsahující inkrementální změny v databázi.

2.3 Zpracované Issues z původní aplikace

Níže lze nalézt seznam `Issues` z původního GitLab repozitáře, které byly v této práci zpracovány a představují vylepšení původní funkcionality nebo něco, na co by se mělo dát pozor při implementaci:

- #184 – nahlášení všech chyb ve formulářích najednou.
- #183 – hromadný zápis do databáze až při odeslání formuláře.
- #180 – nový typ `Online`.
- #179, #178, #177, #176 – triviální chyby u importu publikací. Většinu z nich bude řešit knihovna.

- #169 – povinný `Editor` u typu `Book`.
- #165 – přesunutí seznamu z admin sekce do veřejné sekce.
- #142 – vyhledávání v kategoriích.
- #2 – chronologické řazení dat u ročníku konference, tedy pokud se změní `Submission deadline`, pak se kaskádově nastaví stejné období (měsíc a rok) do `Notification`, atd.

2.4 Publikace

Následující sekce popisuje implementaci vytváření publikací s návazností na autory, časopisy, vydavatele a konference, import a export publikací pomocí citačních formátů a import pomocí Springer API. V poslední řadě je zde popsána implementace detailu a úpravy publikace, která nebyla definována v zadání.

2.4.1 Vytváření

Stránka pro vytváření publikací se vykresluje ze souboru `app/publications/create/page.tsx`, který obsahuje základní rozložení stránky a drobnou navigaci. Za zmínku zejména stojí closure `handleSubmit`, což podle názvu obsahuje logiku vytváření publikace poté, co uživatel odešle formulář.

```
1  const CreatePublicationWithNotifications =  
2    useErrorSuccessNotification(  
3      createPublication,  
4      /*onSuccess message*/,  
5      /*onError message*/  
6    );  
7  
8  const handleSubmit = async (values: IForm) => {  
9    const ret = await CreatePublicationWithNotifications(  
10     values);  
11    if (ret.state === 'success') {  
12      if (values.uploadedFiles.length > 0) {  
13        await uploadFiles(values, ret.data.id);  
14      }  
15  
16      setData(EmptyForm);  
17      router.push(Publications.getDetail(ret.data.id!));  
18    }  
19  };
```

- **Výpis kódu 2.2** Implementace logiky odeslání formuláře

Kód 2.2 obsahuje implementaci této closure. První se zavolá objekt vytvořený pomocí `useErrorSuccessNotification` hooku (vytvořen v [1]), následně pokud se bude jednat o úspěšné provedení akce, tak se nahrají soubory, vymaže se importovací kontext a uživatel je přesměrován na detail nově vytvořené publikace. Ke kontextu import dat více v sekci 2.4.3.

Následně uvnitř formuláře (v komponentě `PublicationForm`) se volá tato funkce uvnitř `handleSubmit` closure (shoda jmen docílena náhodně). V této closure se navíc vykoná úprava vyplněných políček formuláře pomocí návrhového vzoru `Strategy`. Více o tomto vzoru a implementaci v sekci 2.4.1.2.

Samotný formulář se nachází v komponentě `PublicationForm`, která se nachází ve složce `app/publications/components/`. Jedná se o klientí komponentu z důvodu používání hooků a interaktivity spojené s formuláři. Komponenta přijímá dvě *props*, což je React termín pro argumenty komponent:

- `onSubmitAction: (values: IForm) => void` – funkce, která určuje, co se má dít po odeslání formuláře, jako je například implementováno v kódu 2.2.
- `fetchedData?: IForm` – použije se při úpravě publikace, formulář se naplní přiloženými daty.

Celá logika formuláře je implementována pomocí `Mantine Form context` [7, sekce Mantine Form/Form Context], což nám umožňuje přistupovat ke všem funkcionalitám formuláře ve vícero komponentách souborech. Tento kontext je vytvořen v souboru `lib/publicationForm/createPubFormService.ts` a následuje rozhraní `IForm`, které definuje hodnoty a možná políčka formuláře. Tento soubor také obsahuje podpůrné funkce využívané ve formuláři. Definice rozhraní `IForm` se nachází v `lib/publicationForm/types.ts`. Toto rozhraní bylo navrženo podle jednoho z požadavků v *Issues* (2.3, #183), tedy že by se související hodnoty hned neměly propsat do databáze. Tedy toto rozhraní (a `form` objekt) si uchovává všechny možné nové hodnoty v sobě a vše se až po odeslání propíše do databáze.

Následně je vytvořen `form` objekt, který zajišťuje funkcionalitu formuláře. Nachází se v `uncontrolled` módu [7, sekce Mantine Form/Uncontrolled Mode] a jeho výchozí data se liší podle toho, zda se jedná o import publikace (data se poté nachází v proměnné `data`), o úpravu publikace (data se nachází v proměnné `fetchedData`) nebo o vytvoření (pak se formulář „naplní“ objektem `EmptyForm`). Tuto logiku zajišťuje funkce `initializeForm`, která se nachází v souboru `createPubFormService.ts`. Validace jednotlivých políček se poté nachází v atributu `validate` a probíhá nejen při odeslání, ale také při změně hodnoty políčka. Ústřížek kódu 2.3 obsahuje vytvoření a následné použití kontextu.

2.4.1.1 Formulář

Samotný formulář je dle analýzy členěn do tří logických částí. Pokud nebude řečeno jinak, všechny komponenty formuláře se nachází ve složce `/app`

```

1 // lib/publicationForm/createPubFormService.ts
2 export const [PublicationsFormProvider,
3   usePublicationsFormContext,
4   usePublicationsForm] = createFormContext<IForm>();
5
6 //app/publications/components/PublicationForm.tsx
7 const form = usePublicationsForm({
8   mode: 'uncontrolled',
9   initialValues: initializeForm(fetchedData ?? data),
10  validateInputOnChange: true,
11  validate: {
12    // form validation
13  },
14 });
15
16 // wrapping the form with the context
17 <PublicationsFormProvider form={form}>
18   <form onSubmit={...}>
19     ...
20   </PublicationsFormProvider>
21
22 //app/publications/components/formSections/*
23 // access to the context from any component
24 const form = usePublicationsFormContext();

```

■ **Výpis kódu 2.3** Vytvoření Form Contextu a form objektu.

/publications/components/formSections.

První logická část je implementována v komponentě `TopInputs.tsx` a obsahuje všechna povinná a obecná pole pro všechny publikace. Zde se také nachází `Select` [7, sekce Mantine Core/Select] s výběrem typu publikace, který ovlivňuje, jaká políčka se mají uživateli vykreslit dle zvoleného typu. Obsahuje nejen 13 [2] původních typů publikace, ale také přibyl nový typ `online`, což byl jeden z požadavků v *Issues* (2.3, #180). Online bude nabývat povinných a volitelných políček dle **BibLaTeX** specifikace [16].

Následně stojí za zmínku komponenta `MandatoryRelatedFields.tsx`, která obsahuje podmíněčné vykreslení vydavatelů a časopisů. Tyto komponenty jsou obaleny komponentou `InputWithActions`, která zajišťuje rozhraní pro vytváření a editaci souvisejících entit z formuláře. Detailní popis implementace této správy v sekci 2.4.1.3.

Pokud se typ publikace rovná typu `InProceedings` nebo `Proceedings` pak je k dispozici nepovinný výběr konference a ročníku konference, do které tato publikace spadá. Výběr konferencí je opět obalen komponentou `InputWithActions`.

Při výběru konference se ročníky naplní buď z pole upravených konferencí (pokud byla konference změněna), nebo se získají z databáze, jak je znázorněno

v ukázce kódu 2.4. V tomto kódu lze také vidět forma všech dat do `Select` komponent, které se používají v aplikaci. Jedná se o objekty s atributy `value` a `label`, kde `value` vždy obsahuje id toho záznamu, ale místo `BigInt` nabývá typu `string`.

```

1 // On input change
2 onChange={ (value) => {
3   ...
4   getConferenceYears(BigInt(value));
5 }}
6 ...
7 const getConferenceYears = async (id: bigint) => {
8   startTransitionYear(async () => {
9     ...
10    // If the conference was edited, get the years from
11    here.
12    if (form.getValues().editedConferences.length > 0) {
13      foundIndex = form
14        .getValues()
15        .editedConferences.findIndex((val) => BigInt(val.
16        conferenceId!) === BigInt(id));
17    }
18    if (foundIndex >= 0) {
19      ret = form.getValues().editedConferences[foundIndex
20        ].conference_years;
21
22      const auxYears = ret.map((key, index) => ({
23        label: `${key.year} ? (${key.year.getFullYear()})`
24        : '}', ${key.name}',
25        value: key.yearId ? key.yearId.toString() : `new-
26        ${index}`,
27      }));
28      ...
29    }
30    });
31  });

```

■ **Výpis kódu 2.4** Získání ročníků konference při změně výběru.

`InProceedings` má ale dodatečné chování při výběru ročníku, kde do políčka `Booktitle` se dosadí jeho jméno. Jedná se o vylepšení uživatelské přívětivosti, protože často se stává, že sborník obsahuje jméno konference. Navíc, pokud

vybraný ročník má přiřazeného vydavatele, pak také políčko s vydavatelem v publikaci bude zašedlé a bude se používat vydavatel vybraného ročníku. Toto chování ilustruje obrázek 2.1.

Name	Abbreviation
30th International Conference on Compu	CAV

Year	Publisher
2018	Springer-Verlag

Obrázek 2.1 Vyplnění políčka Booktitle a vybrání vydavatele ročníku.

Výběr autorů podléhá rozšířené logice správy souvisejících entit. Z důvodu, že se jedná o **MultiSelect** [7, sekce Mantine Core/MultiSelect] komponentu, je třeba více sofistikovaný přístup pro přístup k jednotlivým autorům. Z toho důvodu se hned pod výběrem autorů nachází rozbalovací sekce, která obsahuje v levé půlce pilulky s vybranými autory a v pravé půlce nové autory. Na každé pilulce se nachází akce pro úpravu autora, v levé sekci následně pilulky obsahují akce na zobrazení publikací autora a v pravé sekci pilulky obsahují akci na odstranění nového autora, jak lze vidět na obrázku 2.2.

Obrázek 2.2 Design výběru autorů se seznamy zvolených a nových autorů.

Pod výběrem se ještě nachází tlačítko, které otevře modál pro vytvoření nového autora. Logika vytváření a úpravy autorů je následně identická se správou souvisejících entit v sekci 2.4.1.3.

Implementace obsahu rozbalovací sekce detailu nových autorů se nachází v komponentě `/components/relatedFormInputs/details/AuthorsDetails`. Obsah kromě obou seznamů autorů obsahuje i modál pro zobrazení publikací autora. Tento modál je logicky identický se zobrazením souvisejících publikací konference nebo ročníků konference a nachází se v souboru `AuthorRelatedPublications.tsx`, tedy získá související publikace a vykreslí je v IEEE citačním formátu. Modál se otevře kliknutím na šedé tlačítko u zvolených autorů a zavoláním funkce `open()` z `useDisclosure` hooku. Pomocí `useState` hooků se nastaví id a jméno autora. Kód lze vidět v ukázce 2.5.

```

1 <Button
2   variant="transparent"
3   color="gray"
4   onClick={() => {
5     open();
6     setName(...);
7     setId(author.id);
8   }}
9 >
10 <IconLink size={16} />
11 </Button>

```

■ **Výpis kódu 2.5** Logika otevření modálu se souvisejícími publikacemi.

Funkce

```

// Server Action from /lib/prisma/actions
// /authors/authorActions.ts
getAuthorsRelatedPublications(authorId: BigInt)

```

v `useEffect` hooku získá seznam souvisejících publikací a následně v komponentě `Card` je vykreslen text získaný z export funkce

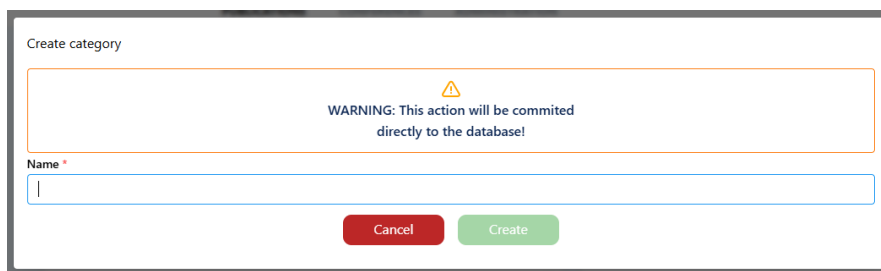
```

exportInIEEEClickableTextFormat(publication:
  PublicationWithRelations, navigateNewTab?: boolean).

```

K dispozici je i tlačítko pro zkopírování citace, které volá textovou verzi IEEE export funkce. Více k export funkcím v sekci 2.4.5.

Následuje výběr publikačních kategorií. Kategorie (jak publikační tak konferenční) jsou jediná entita, jejíž správa se zapisuje přímo do databáze, proto všechny modály obsahují upozorňující text, jak lze vidět na obrázku 2.3.



■ **Obrázek 2.3** Upozorňující text na modálu pro vytvoření kategorie.

Pro zobrazení kategorií slouží komponenta `TreeDataFilter`, která se nachází ve složce `/components/filter/`. Ta obsahuje komponentu `CheckBox-TreeView`, která ve svém jádru obsahuje Mantine komponentu `Tree` [7, sekce Mantine Core/Tree], jež se dá dobře upravovat. Pro vykreslení elementů zde

ve formuláři se používá `renderTreeNodeWithActions` closure, jejíž definice se nachází ve stejném souboru jako tree view komponenta. Tato funkce dodatečně vykreslí tlačítka pro provedení akcí vedle textu záznamu, pokud na něj uživatel najede myší. Otevření modálu a předání dat zajišťují `useState` hooky, které jsou předány do této funkce přes argumenty, což je ilustrováno v ukázce kódu 2.6.

```
1 const render = (props: RenderTreeNodePayload
2   & { search?: string }) =>
3   renderTreeNodeWithActions({
4     ...props,
5     setData: setCategoryModalData,
6     setOpen: setOpenCategoryModal,
7   });
```

■ **Výpis kódu 2.6** Obalení funkce `renderTreeNodeWithActions` za účelem předání hooků.

Tato `render` closure je následně předána do komponenty `TreeDataFilter`. Pro optimalizované vykreslení uzlů je použito podmíněčné vykreslení při vznášení myší nad záznamem, znázorněno v kódu 2.7.

```
1 {elementProps['data-hovered']} && (
2   <Group gap={2}>
3     // buttons for creating a subcategory and editing or
4     deleting a category
5   </Group>
6 )}
```

■ **Výpis kódu 2.7** Podmínečný mount a unmount tlačítek.

Po kliknutí na jakékoli z akcí se otevře modál, jehož mód se odvíjí od zvolené akce. Změnu módu zajišťují přepínače ve `values` objektu `isEdited` nebo `isDeleted`. Tento modál je implementován v souboru `/components/related-FormInputs/modals/CategoryActionModal.tsx`. Modál obsahuje form objekt pro manipulaci s formulářem a podle módu je formulář adekvátně naplněn daty. Následně po odeslání dat jsou data propsána do databáze, jak je znázorněno v kódu 2.8. Po úspěšné operaci se zavolá `refreshFunction`, což v této situaci je funkce `getCategories`, která volá serverovou akci `fetchCategoryTree`.

Funkce `startLoading` pochází z hooku `useTransition`, který se v celé aplikaci používá pro ilustrování načítajícího se stavu u tlačítek či dat při vykonávání asynchronní funkce.

Druhá logická část formuláře je schována v rozbalovací sekci, jejíž implementace je detailně popsána v sekci 2.4.1.5. Tato sekce se nadále dělí na `MidInputs` komponentu a `BottomInputs` komponentu.

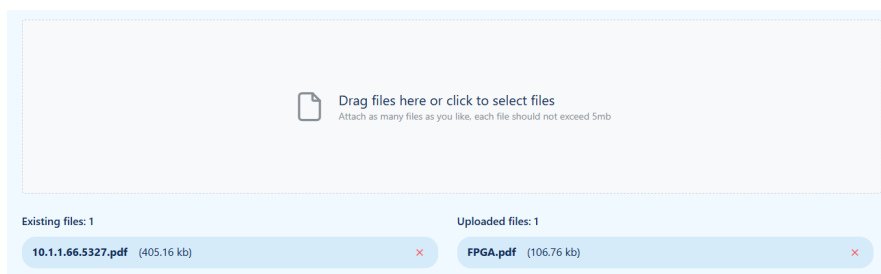
```
1  const handleSubmit = async () => {  
2    ...  
3    startLoading(async () => {  
4      if (categoryForm.getValues().isEdited &&  
5        categoryForm.getValues().id) {  
6        // edit  
7      } else if (categoryForm.getValues().isDeleted &&  
8        categoryForm.getValues().id) {  
9        // delete  
10     } else {  
11       // create  
12     }  
13  
14     if (ret && ret.state === 'success') {  
15       await refreshFunction();  
16       setOpen(false);  
17       setValues(/* clear data */);  
18     }  
19   });  
20 };
```

■ **Výpis kódu 2.8** Implementace odeslání formuláře pro správu kategorií z formuláře.

MidInputs komponenta se skládá ze **switche**, který na základě typu publikace zvolené výše vykreslí adekvátní vstupní pole. Všechny komponenty, které jsou vykresleny na základě typu se nachází ve složce `app/publications/components/formSections`.

BottomInputs komponenta obsahuje jen políčka pro URL publikace (neplatí při typu **online**, který vyžaduje URL povinně), poznámku k publikaci a nahrávání souborů publikace.

Pro nahrávání souborů publikace je použita komponenta **Dropzone** [7, sekce Extensions/Dropzone] a samotné soubory jsou uloženy v rozhraní **IForm** jako pole **File** objektů. Pod **Dropzone** je místo pro vypsání jak nově nahraných souborů, tak již stávajících souborů publikace, jak je znázorněno na obrázku 2.4. Druhá možnost je využita jen u úpravy (sekce 2.4.2).



■ **Obrázek 2.4** Výpis stávajících nebo nově nahraných souborů.

Nahrání souborů na server se provede po odeslání formuláře, jak je naznačeno v kódu 2.2. Funkce

```
uploadFiles(values: IForm, id: bigint): Promise<void>
```

akceptuje `IForm` objekt, ze kterého získá soubory a id nové publikace. Následně funkce serializuje tyto data do nového `FormData` objektu a odešle je pomocí HTTP metody POST na koncový bod API definovaný URL `.../Pub-Conf2/api/files` pomocí `fetch` API. Tato logika pochází ze souboru `/lib/publicationForm/fileHandlers.ts` a je znázorněna v kódu 2.9.

```
1  const formData = new FormData();
2  values.uploadedFiles.forEach((file) => {
3    formData.append('files', file);
4  });
5  formData.append('id', id.toString());
6
7  let response: Response;
8  try {
9    response = await fetch(
10     `${process.env.NEXT_PUBLIC_FILE_STORAGE_URL}`,
11     {
12       method: 'POST',
13       body: formData,
14     }
15   );
16 } catch (e) {
17   // Warning notification
18   return;
19 }
```

■ **Výpis kódu 2.9** Odeslání souborů na koncový bod API za účelem nahrání.

Nahrání souborů není součástí žádné transakce a pokud se operace nepodaří, pak se jen ukáže varovná zpráva, a aplikace pokračuje dál. Konkrétní logika nahrávání souborů a struktury úložiště souborů lze najít v sekci 2.4.1.4.

Třetí část formuláře je také schována v rozbalovací sekci a obsahuje prostor pro správu vlastních atributů. Tyto atributy se používají pro rozšíření již existujících atributů publikace. Celá logika se nachází v komponentě `Attributes.tsx`. Objekt `IForm` si udržuje pole `attributes`, ve kterém se nachází tyto atributy. U každého z nich je příznak `modified`, tedy pokud byl upraven. Dále si pole `deletedAttributes` udržuje atributy s platným id, které budou smazány.

Při vytváření se nový atribut vloží na konec do `attributes` pole. Pokud tento nový atribut obsahuje navíc nějakou hodnotu, je také zapsán do databáze, pokud ne, nic se nestane.

Při úpravě se do modálu pošle index upravovaného atributu a následně se v poli `attributes` nahradí novým objektem a nastaví se příznak `modified`.

Do databáze se pak propíše `modified` atribut pouze s platným id.

Při vyplnění některé z hodnot vlastních atributů se po odeslání formuláře přidá nový záznam do tabulky `Attrib_storage`. Modály vlastních atributů se nachází ve složce `app/publications/components/modals/attributes`.

2.4.1.2 Zápis do databáze

Při odeslání formuláře jsou před zapsáním do databáze data „osekána“ o hodnoty, které do daného typu nepatří, a tím se zamezí ukládání nevyžádaných hodnot. Toto chování zajišťuje návrhový vzor `Strategy` [11], který přemapuje hodnoty. Nejprve se v komponentě `PublicationForm` vytvoří objekt `PublicationTypeStrategyContext`, který následně v `handleSubmit` closure ve stejnojmenné komponentě vykoná strategii, jak je zobrazeno v úkázce kódu 2.10.

```
1 context.setStrategy(form.getValues().type);  
2 values = context.executeStrategy(values);  
3 // DB Commit  
4 onSubmitAction(values);
```

■ **Výpis kódu 2.10** Provedení strategie.

Kontextovou třídu a konkrétní strategii lze najít ve složce `/lib/publicationForm/typeStrategies`. `PublicationTypeStrategyContext` obsahuje funkci `setStrategy`, jež obsahuje `switch`, který vytvoří adekvátní strategii podle typu publikace a následně funkci `executeStrategy`, která deleguje vykonání přemapování. Každá konkrétní strategie následuje rozhraní `Strategy` a implementuje funkci `execute`, která vytvoří nový `IForm` objekt s adekvátními atributy.

Po nastavení strategie je zavolána funkce `onSubmitAction`, která je předána zvenčí. Konkrétní implementace této funkce pro vytvoření publikace je znázorněna v kódu 2.2.

Funkce pro vytváření publikací

```
createPublication(data: IForm): Promise<Publication>
```

ze souboru `/lib/prisma/actions/publications/createActions.ts`, je obalena autentizačním wrapperem `withAuthUserCanEdit`, který zjišťuje, zda-li má uživatel práva na to vykonávat tuto funkci. Více o autentizaci v [1].

Tato funkce pro vytváření publikací nejprve na začátku vytváří objekty, které v následné `Prisma create query` vytváří vztah na dané související záznamy, jak je znázorněno v kódu 2.11. Následně pomocí interaktivní transakce [9, sekce `Prisma Client/Queries/Transactions and batch queries`] je proveden zápis do databáze.

Tuto transakci lze abstrahovat na následující fáze:

1. Vytvoření nové konference či úprava konference, pokud byla vytvořena či upravena ve formuláři a vytvoření objektů pro navázání ročníku a vydavatele.
2. Úprava či mazání objektů, se kterými bylo manipulováno v rámci správy souvisejících entit, viz 2.4.1.3. Docíleno pomocí `Promise.all()` funkcí.
3. Vytvoření publikace.

Funkce

```
constructPublicationDependencies(data: IForm):  
    PublicationDependencies
```

se nachází ve souboru `/lib/prisma/actions/publications/publicationUtils.ts` a slouží pro vytváření objektů, které obsahují atributy, podléhající Prisma koncepci pro vytváření či napojení souvisejících záznamů [9, sekce Prisma Client/Queries/Relation Queries].

2.4.1.3 Implementace správy souvisejících entit z formuláře

Ve formuláři při vytváření či úpravě konference nebo publikace je možnost také vytvoření související entity, pokud se v databázi nenachází. Jak bylo zmíněno již výše ve 2.4.1, jedna z GitLab Issues požadovala, aby tyto související entity nebyly propsány hned do databáze. Z tohoto důvodu nové entity a informace o nich jsou dočasně ukládány v rozhraní `IForm`. Pro implementaci správy souvisejících entit bylo navrženo integrační API, které se skládá z následujících logických částí:

- Wrapper komponenty, která vstupnímu poli přidá tlačítka pro vytvoření, úpravu nebo smazání (viz obrázek 2.5) a umožňuje definovat akce těchto tlačítek.
- Místo pro detail nově vytvořené entity.
- Atribut typu `Object` nebo `Array` pro ukládání nových, upravených či vymazaných entit.
- Modál obsahující formulář pro vytvoření či úpravu dané entity.

Související entity se ve formuláři vybírají ze seznamu v komponentě `Select`. Seznam entit se skládá z dvojic `value` a `label`, kde `value` obsahuje id entity a `label` obsahuje to, co vidí uživatel při výběru.

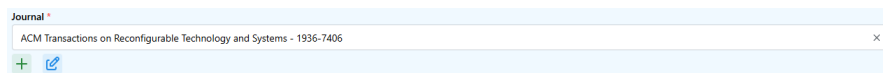
Nově vytvořené entity se ukládají v `IForm` attributech začínající na `new*`, typu `Object`. Výjimce podléhá atribut `newAuthors`, který je typu pole. Upravené entity se ukládají do atributu začínajícího na `edited*`, typu pole. Tabulka 2.1 obsahuje názvy konkrétních atributů pro správu entit.

```

1  const { journal, authors, modifiedAttributes, attributes,
2    validDeletedAttributes, identifiers } =
3    constructPublicationDependencies(data);
4
5  return prisma.$transaction(
6    async (tx) => {
7      // 1. and 2. phase
8      ...
9      return tx.publication.create({
10        data: {
11          // Publication Attributes
12          ...
13          // Relations
14          publisher,
15          journal,
16          conference_year,
17          categories_has_publication: {
18            create: data.categories.map((id) => ({
19              categories: {
20                connect: {
21                  id: BigInt(id),
22                },
23              },
24            })),
25          },
26          author_has_publication: {
27            create: authors,
28          },
29          attrib_storage: {
30            create: attributes,
31          },
32        },
33      });
34    },
35  );
36 }

```

■ **Výpis kódu 2.11** Transakce, která zapíše změny souvisejících objektů a vytvoří novou publikaci.



■ **Obrázek 2.5** Rozšíření vstupního pole o tlačítka pro správu entity.

Všechny potřebné front-end komponenty, tedy modály, detaily a samotná wrapper komponenta, se nachází ve složce `components/relatedFormInputs`.

Entita	Create atribut	Edit atribut
Autor	newAuthors	editedAuthors
Vydavatel	newPublisher	editedPublishers
Časopis	newJournal	editedJournals
Konference	newConference	editedConferences

■ **Tabulka 2.1** Entity a atributy objektu `IForm`, které slouží pro ukládání nově vytvořených nebo upravených entit.

Wrapper komponenta `InputWithActions` poskytuje rozhraní pro definování akcí po kliknutí na akční tlačítko. Tyto akce ve formě closures jsou předány pomocí následujících props: `editEntryAction`, `createEntryAction` a `deleteEntryAction`.

Funkce `deleteEntryAction` a tlačítko pro smazání je vidět pouze, pokud se jedná o nově vytvořenou entitu. Dle požadavků není třeba možnost mazání záznamů z databáze z formuláře. Implementace funkcí `createEntryAction` a `editEntryAction` naplní pomocný `Action` objekt relevantními daty (tento objekt je následně použit ve formuláři, tedy když se jedná o úpravu, obsahuje data na úpravu) a pomocí `useState` hooku otevřou formulář.

Pro vytvoření entity je naplnění tohoto objektu přímočaré. U úpravy je to ovšem jinak. Je třeba rozeznat, zda-li se uživatel snaží upravit nově vytvořenou entitu nebo již existující entitu. Kód 2.12 ukazuje konkrétní příklad implementace `editEntryAction` pro entitu časopisu. Výběr upravované entity probíhá následovně:

- Uživatel se snaží upravit nově vytvořenou entitu – data se vezmou z `new*` atributu.
- Uživatel se snaží upravit již existující entitu:
 - Pokud se v seznamu vyskytují všechna data, jsou naplněna ze seznamu.
 - Pokud se v seznamu nevyskytují všechna data, jsou naplněna záznamem v `edited*`, pokud uživatel tuto entitu v této session¹ již upravoval nebo se získají z databáze, pokud v této session uživatel entitu ještě neupravoval.

Uživateli se zobrazí modál vyobrazený na obrázku 2.6. Všechny modály se nachází ve složce `/components/relatedFormInputs/modals`. Každý modál se skládá z formuláře, který zejména dohlíží na validaci výstupních dat. Jak bylo výše zmíněno, každému modálu náleží objekt pro předávání dat, který se přejímá přes `initialValues` prop. Každý objekt obsahuje boolean atribut `isEdited`, dle kterého se posuzuje výsledná akce při stisknutí „Submit“ tlačítka.

¹Session se zde myslí průběh vytváření publikace – od otevření formuláře do jeho zavření či odeslání.

```

1  const editJournal = async () => {
2      if (form.getValues().newJournal) {
3          setJournalModalData({
4              // Get data from newJournal
5              ...
6          });
7      } else if (form.getValues().journal) {
8          const index = form
9              .getValues()
10             .editedJournals.findIndex((val) => val.id! ===
11             BigInt(form.getValues().journal!));
12
13             if (index >= 0) {
14                 setJournalModalData({
15                     // Get data from editedJournals array
16                     ...
17                 });
18             } else {
19                 const ret = await getJournalById(BigInt(form.
20                 getValues().journal!));
21                 if (!ret) return;
22
23                 setJournalModalData({
24                     // Get data from the database
25                     ...
26                 });
27             }
28         }
29
30         setOpenJournalModal(true);
31     };

```

■ **Výpis kódu 2.12** Ilustrace získání dat pro úpravu souvisejících entit.

Pokud se jedná o vytvoření nové entity, tedy `isEdited` je `false`, do `IForm` atributu `new*` se vloží nový objekt s poskytnutými daty. Pokud se ale jedná o úpravu buď již existující nebo nové entity, mohou se stát následující akce:

- Uživatel se snaží upravit nově vytvořenou entitu – data vymění objekt v `new*` atributu.
- Uživatel se snaží upravit již existující entitu:
 - Pokud uživatel v této session entitu již editoval, vymění se data na správném indexu v `edited*` poli.
 - Jinak se vloží nový záznam do `edited*` atributu.
 - Nakonec se změní záznam v seznamu.

Create journal

Name *

Abbreviation

DOI

ISSN/ISBN Identifiers +

ISSN/ISBN Value

Note

ISSN ISBN X

Cancel Create

Obrázek 2.6 Modál pro vytvoření nové související entity.

Detail nově vytvořené entity se objeví pod vstupním **Selectem** dané entity. Obsahuje všechna data, která byla vyplněna. Samotný **Select** zašedne a nejde použít, dokud existuje nová entita, jak je zobrazeno na obrázku 2.7. Také se objeví tlačítko pro smazání nové entity, pokud se uživatel rozmyslí.

Journal

+ ✎ 🗑

New journal

Name Test DOI Abbreviation TST

ISSN/ISBN Identifiers

ISSN/ISBN Value 12345678 Note

ISSN ISBN

Obrázek 2.7 Detail nové související entity.

Po vykonání akce se zavolají funkce `postEditAction` a `postCreateAction`, které definují akce, které se mají provést po úpravě či vytvoření. Většinou jsou buď prázdné nebo se jedná o zašednutí vstupního políčka.

Všechny související entity následují tuto integrační API pro jejich správu, mezi ně patří: autoři, časopisy, vydavatelé a konference. Logika vytváření a editace entit obecně následuje výše popsané algoritmy, můžou se ale lišit v implementačních detailech.

Vlastní atributy mají rozšířenou funkcionalitu o možnost smazání z databáze, neboť celkově mají v aplikaci trochu jiný účel. Více k implementaci vlastních atributů v sekci 2.4.1.1.

U správy souvisejících konferencí je logika komplikovanější. Stále se následuje API, ale je třeba dbát na několik implementačních detailů. Nejprve je třeba po úspěšné úpravě naplnit seznam ročníků dané konference v `postEditAction` nebo `postCreateAction` funkci. Konference se upraví již v modálových funkcích, tedy v `post*` funkcích se sahá pouze do atributů `form` objektu.

Je třeba zmínit, že pokud uživatel upraví vydavatele při vytváření či editaci konference, tito upravení vydavatelé jsou předáni objektu `IForm`.

Pokud se jedná o vytvoření či úpravu nové konference, získají se ročníky

z `newConference` objektu, přetransformují se do seznamových dat a pomocí `useState` se tato data naplní do `Select` komponenty. Ale pokud se jedná o úpravu již existující konference pak program zavolá funkci `getConferenceYears`, která se používá i pro získání ročníků při změně konference. Tato funkce se v tomto případě dívá do `editedConferences` pole a odtud získá ročníky. Viz ukázka kódu 2.4.

Následně v serverové akci jak pro vytvoření publikace, tak pro úpravu publikace, musí program identifikovat, zda-li se má vytvářet či upravovat konference. Z optimalizačních důvodů je upravována jen ta konference, k jejíž ročníku bude poté publikace napojena.

Pokud se má konference vytvořit, transakce vytvoří konferenci a z vrácených hodnot se naplní objekty `publisher` a `conference_year`, které slouží jako objekty pro napojení těchto entit v Prisma `create query`.

Pokud se má konference upravit, pak nejprve se vypočítá budoucí index nového ročníku, následně se konference upraví. Z toho důvodu, že Prisma vkládá do databáze v opačném pořadí, než je specifikováno v poli ročníků, se musí nejprve odhadnout index, který ročník bude zaujímat ve vrácených datech po úpravě konference. Po navrácení dat se z pole ročníků získá id ročníku pro navázání. Tuto logiku znázorňuje ukázka kódu 2.13. Po přidání nové konference či přidání nového ročníku úpravou je poslední (nejnovější) ročník následně automaticky vybrán v `Select` komponentě.

2.4.1.4 Implementace správy souborů

Soubory se na serveru ukládají v Docker kontejneru ve složce `/uploadedFiles`. Struktura této složky je identická s původní strukturou úložiště, tedy soubory jsou ukládány ve složkách, které nesou jako název id dané publikace, ke kterým patří. V nové aplikaci je tato volba ukládání zachována.

Nahrávání, získávání a mazání souborů publikací je implementováno pomocí *Route handlers*, což je způsob jak v Next.js vystavit vlastní koncový bod API [6, sekce Routing/Route Handlers].

Ve složce `/app/api/files/` se v souboru `route.ts` (název následuje Next.js konvenci) nachází implementace bodu pro POST metody nahrávání souborů. V této stejnojmenné funkci se nejprve získají soubory a id publikace z obsahu požadavku, vytvoří se složka, která má v názvu získané id, a soubory se zapíší do této složky. Návrátové kódy metody POST jsou znázorněny v tabulce 2.2.

Dále se v této složce nachází `slug [id]`, jež v souboru `route.ts` obsahuje implementace GET a DELETE metod. `Slug` obsahuje id publikace, jejíž soubory si uživatel přeje číst nebo mazat.

Obě metody podporují query parametr `filename`, který definuje jméno konkrétního souboru pro práci. DELETE ale tento parametr nemá povinný. Pokud je parametr vynechán, metoda DELETE smaže celou složku, zatímco s parametrem smaže pouze konkrétní soubor. Toto chování znázorňuje ukázka

```
1 // Finds index of first year with null id
2 const newYearsStart = editedConference.conference_years.
  findIndex((value) => !value.yearId);
3 // Filter array, get only the elems with null ids and get
  their length
4 const newYearsLength = editedConference.conference_years.
  filter(
5    (value) => !value.yearId
6  ).length;
7
8 const calculatedEditedIndex =
9   newYearsStart + (newYearsLength - 1 - (editedIndex -
10    newYearsStart));
11
12 // Fetch former conference from db
13 ...
14 // Purge all edited publishers => they are in the
  publication now
15 editedConference.editedPublishers = [];
16 const updateConferencePromises = await
  updateConferenceRaw(
17   editedConference,
18   initialConference,
19   {},
20   tx
21 );
22
23 [conf] = await Promise.all(updateConferencePromises);
24 ...
25 const year = updated.conference_year[
  calculatedEditedIndex];
```

■ **Výpis kódu 2.13** Vypočítání budoucího indexu ročníku pro navázání a úprava konference.

Kód	Důvod
200	Pokud se soubory úspěšně nahrají, nebo pokud žádné soubory na nahrání poslány nebyly (nic se neprovedlo).
401	Uživatel není přihlášen.
500	Chyba při nahrávání.

■ **Tabulka 2.2** Návrátové hodnoty POST metody.

kódu 2.14. Návrátové kódy metody DELETE jsou znázorněny v tabulce 2.3. Ideální kód úspěšně provedeného smazání je 204, ale tento kód házel error (neznámý kód), takže se používá 200.

Kód	Důvod
200	Úspěšně smazáno nebo se nic neprovedlo.
400	Chybí id publikace.
401	Uživatel není přihlášen.
500	Chyba při mazání.

■ **Tabulka 2.3** Návrátové hodnoty DELETE metody.

Metoda GET slouží spíše pro stahování souborů, než pro zjištění obsahu složky, z toho důvodu je `filename` povinný. Funkce najde vyžádaný soubor, přečte jej do byte bufferu a tento stream odešle klientovi. Snaží se také donutit prohlížeč, aby `.pdf` soubory zobrazil v okně. Zbylé soubory jsou staženy. Implementace je znázorněna v ukázce kódu 2.15 a návratové kódy metody GET jsou znázorněny v tabulce 2.4.

Kód	Důvod
200	Úspěšné stáhnutí souboru.
400	Chybí parametr <code>filename</code> nebo id publikace.
401	Uživatel není přihlášen.
404	Soubor nebyl nalezen.
500	Chyba při stahování.

■ **Tabulka 2.4** Návrátové hodnoty GET metody.

Pro získání souborů publikace (obsah složky s id publikace) slouží serverová akce

```
getPublicationFileNames(id: bigint): Promise<
  PublicationFiles[]>,
```

která se nachází v souboru `/lib/prisma/actions/publications/files-Actions.ts`. Tato akce přijímá id publikace, jejíž soubory si uživatel přeje vidět a vrací `Promise` s polem `PublicationFiles` objektů, které obsahují jméno a velikost souboru (v bytech). Pro získání velikosti je třeba získat statistiky souborů v souborovém systému, což je implementováno pomocí asynchronní funkce `fs.stats`, v jejímž callbacku bylo potřeba provést `resolve` všech `Promise` ručně. Toto chování je znázorněno v ukázce kódu 2.16.

```
1 export async function DELETE(  
2   request: NextRequest,  
3   { params }: { params: Promise<{ id: string }> }  
4 ) {  
5   const session = await auth();  
6   if (!session) {  
7     return NextResponse.json(null, { status: 401 });  
8   }  
9   ...  
10  if (!id) {  
11    return NextResponse.json({ error: 'ID is missing!' },  
12      { status: 400 });  
13  }  
14  // Delete file or folder if exists  
15  try {  
16    if (filename) {  
17      // Delete just a file  
18      const filePath = path.join(`${process.env.  
19        FILE_STORAGE}`, `${id}`, filename);  
20      if (fs.existsSync(filePath)) {  
21        fs.rmSync(filePath);  
22        // For some reason 204 threw an error that it is  
23        an unknown status...  
24        return NextResponse.json(null, { status: 200 });  
25      }  
26    } else {  
27      // Delete whole folder  
28      const folderPath = path.join(`${process.env.  
29        FILE_STORAGE}`, `${id}`);  
30      if (fs.existsSync(folderPath)) {  
31        fs.rmSync(folderPath, { recursive: true, force:  
32          true });  
33        return NextResponse.json(null, { status: 200 });  
34      }  
35    }  
36  } catch (e) {  
37    if (e instanceof Error) {  
38      return NextResponse.json({ message: e.message as  
39        string }, { status: 500 });  
40    }  
41  }  
42  ...  
43 }
```

■ Výpis kódu 2.14 Implementace mazání souborů na serveru.

```
1  ...
2  try {
3    if (fs.existsSync(filePath)) {
4      const buffer = fs.readFileSync(filePath);
5      const extension = path.extname(filePath).toLowerCase();
6      let contentType = 'application/octet-stream';
7      let disposition = 'attachment';
8
9      // When the file is a pdf then try to force the
10     browser to display it in a window.
11     // Otherwise download it.
12     if (extension === '.pdf') {
13       contentType = 'application/pdf';
14       disposition = 'inline';
15     }
16
17     const headers = new Headers();
18     headers.append('Content-Disposition', `${disposition}; filename="${filename}"`);
19     headers.append('Content-Type', contentType);
20
21     return new Response(buffer, {
22       headers,
23     });
24   }
25   ...
```

■ **Výpis kódu 2.15** Implementace serializace souboru pro stažení ze serveru.

2.4.1.5 Implementace vlastní rozbalovací sekce

Pro účely skrývání obsahu neměla knihovna Mantine žádnou vhodnou komponentu, proto byla vyvinuta komponenta `Dropdown`, která se nachází v souboru `/components/common/pubConfDropdown/Dropdown.tsx`.

Komponenta se rozvine kliknutím na ní, ale lze ji předat i externí hook pro otvírání „zvenčí“. Na pravé straně také lze přidat vlastní akční tlačítko.

Skrytý obsah je v Mantine komponentě `Collapse` [7, sekce Mantine Core/-Collapse], která nabízí animovaný přechod při rozevírání či uzavírání. U této komponenty ale neprobíhá `un-mount` skrytého obsahu, což znamená, že obsah je stále vykreslen na stránce, jen je vizuálně skryt [17]. Toto způsobovalo velké problémy s výkonem na formuláři pro vytváření konferencí a konferenčních ročníků, když se v seznamu nacházelo už jen 10 ročníků. Z toho důvodu bylo třeba samotný `Collapse` trochu modifikovat, aby implementoval *Lazy loading* a skryté komponenty se `un-mountly`. Ukázka kódu 2.17 ukazuje toto chování, samotný kód byl inspirován [17].

```
1  const files = fs.readdirSync(path);
2  if (files.length < 1) {
3    return publicationFiles;
4  }
5  const fileStatsPromises = files.map(
6    (file) =>
7      new Promise<{ file: string; stats: fs.Stats }>((
8        resolve, reject) => {
9          const filePath = `${path}/${file}`;
10         fs.stat(filePath, (err, stats) => {
11           if (!err) {
12             resolve({
13               file,
14               stats,
15             });
16           }
17           reject(err);
18         });
19       })
20 );
21
22 const stats = await Promise.all(fileStatsPromises);
23 stats.forEach((entry) =>
24   publicationFiles.push({
25     name: entry.file,
26     size: entry.stats.size,
27   })
28 );
```

■ **Výpis kódu 2.16** Přečtení složky obsahující soubory publikace a získání jejich statistik.

2.4.2 Úprava publikace

Úprava publikace se odehrává na stejném formuláři jako vytváření. Pro implementaci formuláře viz sekce 2.4.1. Stránka s úpravou publikace se nachází v server komponentě `/app/publications/[id]/edit/page.tsx`, která také formulář naplní daty pomocí funkce

```
fetchPublication(id: bigint): Promise<Publication>,
```

která vrátí publikaci podle id. Následně je vytvořen nový `IForm` objekt, který je předán komponentě `PublicationForm`. Také pomocí funkce `fetchPublicationFileNames`, popsána v sekci 2.4.1.4, jsou získány soubory dané publikace.

Klientská komponenta `/app/publications/[id]/edit/PublicationEditForm.tsx` obsahuje funkci `handleDirtySubmit`, která imple-

```
1  useEffect(() => {
2    if (props.in) {
3      // First mount the children
4      setChildrenMounted(true);
5      // Then trigger the animation in the next frame
6      const animationId = requestAnimationFrame(() =>
7        setIsAnimating(true));
8      return () => cancelAnimationFrame(animationId);
9    }
10
11    // When closing, animate first
12    setIsAnimating(false);
13  }, [props.in]);
14
15  return (
16    <Collapse
17      ...
18      onTransitionEnd={() => {
19        if (!props.in) {
20          setChildrenMounted(false);
21        }
22      }}
23    >
24      {childrenMounted && children}
25    </Collapse>
26  );
```

■ **Výpis kódu 2.17** Optimalizační kroky pro Collapse komponentu. Inspirováno [17].

mentuje logiku odeslání formuláře. Nejprve zapíše úpravu publikace do databáze a pokud byla operace úspěšná tak smaže soubory určené ke smazání či nahraje nové soubory. Následně přejde na detail publikace. Ukázka 2.18 kódu ilustruje tuto funkcionalitu.

2.4.3 Import přes definice

Importem přes definice se v této aplikaci myslí proces zpracování záznamu textového publikačního formátu za účelem uložení si publikace na kterou tento formát odkazuje. Podporované formáty jsou tři:

- BibTeX [2],
- EndNote tagged file [3] a
- RefWorks tagged format [4].

```
1  const ret = await EditPublicationWithNotifications(id,
    values);
2  if (ret.state === 'success') {
3    if (values.deletedFiles.length > 0) {
4      for (const file of values.deletedFiles) {
5        await deleteFile(id, file.name);
6      }
7    }
8
9    if (values.uploadedFiles.length > 0) {
10     await uploadFiles(values, id);
11   }
12   ...
13 }
```

■ **Výpis kódu 2.18** Propsání úpravy publikace do databáze a manipulace souborů.

Pro zpracování formátů je použita knihovna Citation.js [18], která umožňuje zpracování několika známých citačních formátů, včetně formátů uvedených výše. Citation.js je postaven na pluginech a zde byly použity pluginy pro zpracování BibTeX [19], enw (EndNote tagged file) [20] a RefWorks [21].

Oproti původní aplikaci, kde tlačítka pro import byly až na samotném formuláři, zde si uživatel může vybrat požadovaný import v tlačítku na seznamu publikací. Pokud klikne na šipečku vedle tlačítka „Create from blank“, ukáže se rozbalovací nabídka s možnostmi „Import from definition“ a „Import from Springer API“. K importu ze Springer API v sekci 2.4.4.

Tato sekce se zaměří na funkcionalitu schovanou za tlačítkem „Import from definition“. Uživateli se zobrazí modál s textovým polem, kde vyplní požadovanou definici. Po kliknutí na „Import“ se provede validace a uživatel je přesměrován na formulář pro vytvoření publikace s již dosazenými atributy, které se nacházely v definici.

Při kliknutí na výše zmíněné tlačítko se do URL přidá **query** parametr **?imp=**, podle kterého se otevře požadovaný modál:

- **def** – zobrazí se modál pro import z definice.
- **spr** – zobrazí se modál pro import ze Springer API.

Toto rozhodování, jaký modál se má zobrazit, se provádí v komponentě `/app/publications/components/modals/import/ImportModal.tsx`, která obsahuje samotný modál. Formulář pro import přes definici se nachází v komponentě `/app/publications/components/modals/import/DefinitionForm.tsx`.

Při odeslání import formuláře se volá serverová akce

```
parseDefinition(definition: string):
```

Promise<ImportResponseObject>,

která zpracuje daný definiční řetězec a následně vrátí objekt, který kromě dat v objektu IForm vrátí i informaci, zda-li operace proběhla úspěšně. Nachází se v souboru `/lib/importFormatParser/definitionFormatParserService.ts` a ukázce kódu 2.19. Následně jsou data předána do import kontextu, aby mohla být dosazena do formuláře.

```
1  const handleOnSubmit = async () => {
2    const response = await parseDefinition(form.getValues()
      .definition.trim());
3
4    if (!response.success || !response.data) {
5      // set errors
6    }
7
8    // Set data to the context
9    setData(response.data);
10   router.replace(Publications.getCreate());
11 };
12
13 // Server action parseDefinition
14 let result: ImportResponseObject;
15
16 const refworks = definition.match(refworksRegex);
17 const enw = definition.match(enwRegex);
18 const bibtex = definition.match(bibtexRegex);
19
20 if (refworks) {
21   result = await refworksParser(definition);
22 } else if (enw) {
23   result = await enwParser(definition);
24 } else if (bibtex) {
25   result = await bibtexParser(definition, bibtex[1]);
26 } else {
27   result = {
28     data: null,
29     success: false,
30     message: 'Unknown definition!',
31   };
32 }
33 return result;
```

 **Výpis kódu 2.19** Zpracování definice a logika jejího rozeznání typu .

V serverové akci se pomocí regexu rozpozná, o jaký formát se jedná, a následně se provede požadované zpracování. Řetězec se prožene funkcí `Cite` z knihovny `Citation.js`, která vytvoří `CSL-JSON` [22] objekt, který je následně zpra-

cován a verifikován knihovnou Zod [23]. Tento Zod objekt s názvem `CitationsEntrySchema` je následně použit pro tvorbu objektu s rozhraním `IForm`. Definice Zod schématu se nachází v souboru `/lib/importFormatParser/types.ts`. Tuto logiku ilustruje ukázka kódu 2.20. Použité chybové hlášky jsou z knihoven `Zod` nebo `Citation.js` a ty jsou následně ukázány uživateli.

```

1 try {
2   const data = Cite(definition);
3   data.format('data', { format: 'object' });
4   entry = CitationsEntrySchema.parse(data.data[0]);
5 } catch (e) {
6   return handleError(e, response);
7 }

```

■ **Výpis kódu 2.20** Zpracování definice a vytvoření objektu pro další zpracování pomocí knihovny `Zod`.

Finální `IForm` objekt je vytvořen ve funkci

```

createFormObjectBase(entry: Entry,
  definition: string,
  type: PubType): Promise<IForm>,

```

která vykoná dodatečné zpracování dat na míru aplikace, včetně získání autorů, vydavatelů, časopisů a konferencí z databáze a nachází se v souboru `/lib/importFormatParser/utils.ts`. V této funkci se také naplní `issues-Array`, který obsahuje nějaké upozornění a nesrovnalosti v definici a ukáže je následně uživateli na začátku formuláře pro vytváření publikace, společně s definičním řetězcem jak je znázorněno na obrázku 2.8.



■ **Obrázek 2.8** Zobrazení definice na formuláři pro vytvoření publikace.

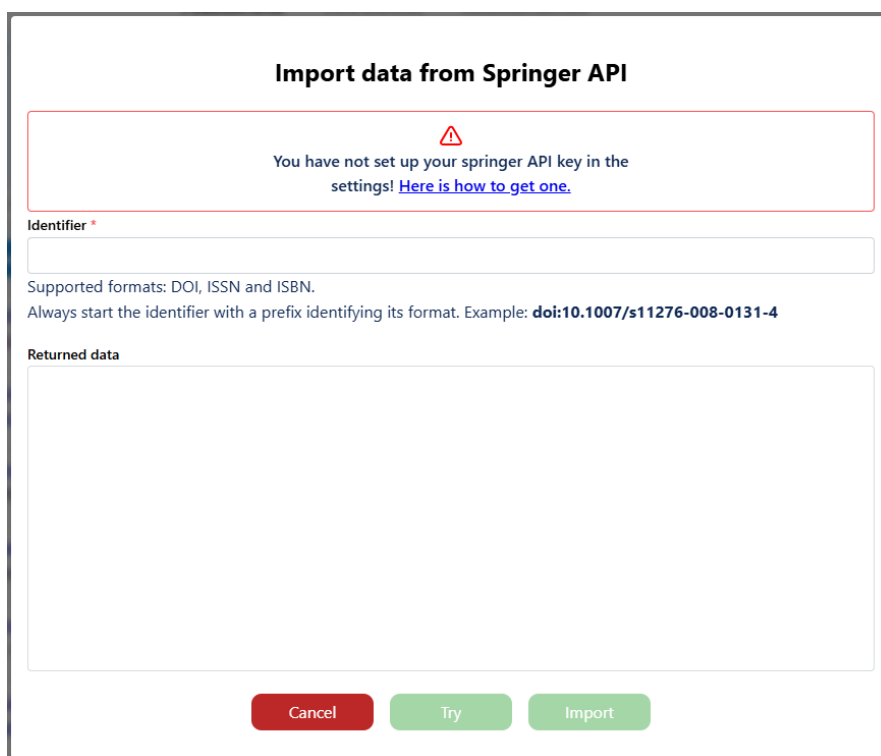
Po vytvoření objektu je potřeba publikaci přiřadit odpovídající typ. Knihovna `Citation.js` se snaží být univerzální a obsahuje mnoho různých typů publikací, které nejsou relevantní aplikaci, ale jedná se například o bratrské či související typy. Program se snaží minimalizovat mnohotvárnost tím, že byly vybrány relevantní typy a ručně se následně mapují na typy relevantní aplikaci. Mapa typů může být nalezena v příloze E.

U BibTeXu je typ zpracován dvakrát. Poprvé je třeba předělat knihovní typ na typ používaný v aplikaci, díky kterému je následně naplněn `IForm` objekt. Následně je původní typ definice (který je vypreparován z definice a předán do funkce) přemapován na ekvivalent použitý v aplikaci. U zbylých dvou formátů je první zpracování také použito, aby se jednodušeji pracovalo s již používaným `enumem`.

Je třeba podotknout, že **EndNote** a **RefWorks** formáty neobsahují některé typy, jaké jsou používány v aplikaci, jako je například **InCollection**.

2.4.4 Import přes Springer API

Import článků ze Springer API funguje na podobný způsob jako přes definici, který je vysvětlen v sekci 2.4.3. Opět se uživateli ukáže modál s formulářem. Tentokrát ale obsahuje políčko pro vyplnění identifikátoru ve formátu DOI, ISSN nebo ISBN. Pod tímto políčkem se nachází textové pole pro získaná data a pod ním tři tlačítka: „Cancel“, „Try“ a „Import“, jak je znázorněno na obrázku 2.9. Může se zde objevit také chybová hláška, že uživatel nemá nastavený API klíč, který je potřeba pro autorizované stahování dat. Tento klíč je možné si zřídit a uložit si jej v nastavení.



Obrázek 2.9 Formulář pro získání článku ze Springer API.

Samotný formulář se nachází v souboru `app/publications/components/`

`modals/import/SpringerForm.tsx`. Uživateli je povolen import do formuláře pouze pokud úspěšně získá data z koncového bodu API. To docílí pomocí tlačítka „Try“, které odešle požadavek a vypíše data nebo chybovou hlášku. Tlačítko volá funkci

```
fetchSpringerDefinition(identifier: string):  
    Promise<any>,
```

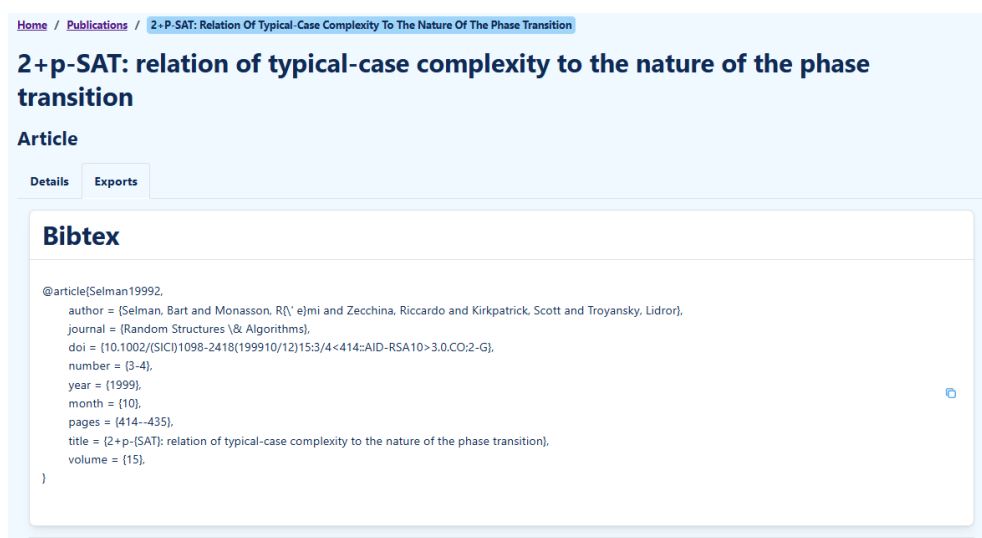
která odešle požadavek pomocí `fetch` API a zpracuje vrácená data do `ApiResponse` objektu. Nachází se v souboru `app/publications/actions/importModalActions.ts`. Aplikace odesílá požadavky na koncový bod `/meta/v2/json`.

Po zmáčknutí tlačítka „Import“ se vytvoří `IForm` objekt a předá se kontextu. Vracená data jsou ukázána uživateli na začátku formuláře, stejně jako to bylo u importu přes definici.

2.4.5 Export

Exportem publikací se v této aplikaci myslí vytvoření citačního formátu z informací publikace za účelem jejího citování. Pro export se opět používá knihovna `Citation.js` a stejné pluginy jako pro import přes definice v sekci 2.4.3.

Připravené formáty pro export lze najít na detailu publikace v záložce „Export“, jak je znázorněno na obrázku 2.10. K dispozici jsou stejné formáty jako u importu, tedy BibTeX, EndNote tagged file a RefWorks tagged format. Dále jsou k dispozici i textové formáty IEEE, ACM a ISO 690.



Obrázek 2.10 Záložka export na detailu publikace.

Exportní API pro aplikaci PubConf je implementována v souboru `lib/ci-`

`tationsExport/citationsExportService.ts`. Tento soubor obsahuje celkem sedm exportních funkcí.

Průběh každé z funkcí pro vytvoření citačního formátu je velmi podobný, nejprve je postaven CSL-JSON objekt pomocí `CitationsObjectBuilder` třídy, která implementuje návrhový vzor `Builder` [12], následně je vygenerován citační formát, opět pomocí funkce `Cite` a v poslední řadě je tento `string` případně doupřaven. Průběh vytvoření EndNote formátu lze vidět na ukázce kódu 2.21.

Doupřavením řetězce se zejména myslí odstranění nechtěných atributů, nebo přejmenování či přidání některých atributů. Nejvíce úpravy řetězce se nachází u BibTeX formátu, kde knihovna `Citation.js` některé atributy nazývala nesprávně, případně tam úplně chyběly. Mezi jednu z výhrad knihovny `Citation.js` patří špatné rozpoznávání zkratk a přebytečné závorkování u BibTeX definic. Správně přidá závorky ke slovům s velkými písmeny (například `{IEEE}`), jenže knihovna nerozezná zkratky, které mají být správně malými písmeny (jsou například chybně psány velkým) a navíc závorky dává zbytečně ke slovům pouze s velkým počátečním písmenem.

`Builder` se nachází v souboru `lib/citationsExport/citationsObjectBuilder.ts` a jedná se o třídu, která následuje `CitationsBuilder` rozhraní a implementuje definované metody.

U exportních funkcí pro textové citace probíhá jen sestavení objektu a následné vytvoření řetězce. Zde se opět používá funkce `Cite`, ale navíc se zde používají šablony textových citací, převzaty z [24]². Tyto šablony jsou ve formátu XML a slouží pro vygenerování adekvátní textové citace. Lze je najít v souboru `/lib/citationsExport/templates.ts` v podobě `string` proměnných.

Podle vyžádaného typu citace se zvolí šablona a použitý jazyk. Tyto funkce navíc přijímají `boolean` argument `asText`, který říká, zda-li má být citace v textové nebo HTML podobě. Textová podoba se použije v tlačítku pro zkopírování citace, zatímco HTML verze je vysázena na stránce. Použití šablon je ilustrováno v ukázce kódu 2.22.

Za zmínku stojí zejména funkce `exportInIEEEClickableTextFormat`, která vytvoří klikatelný textový formát. Používá se k tomu HTML element `<a>` a speciální funkce v třídě `Builder` s názvy `addClickable*`. Tyto funkce vytvoří klikatelné položky na: autory, detail publikace (přes název), časopis, ročník konference či URL. Funkce `exportInIEEEClickableTextFormat` navíc obsahuje `boolean` argument `navigateNewTab`, který říká, zda-li kliknutím na odkaz se má stránka otevřít v novém okně či nikoli. Tyto klikatelné citace se používají v seznamu publikací a v seznámech souvisejících publikací.

Stejně jako u importu přes definici i zde je potřeba namapovat `PubConf` publikační typy na `Citation.js` publikační typy. Mapa je k dispozici v příloze F.

²Jedná se o soubory `ieee.csl`, `acm-sig-proceedings.csl` a `iso690-author-date-en.csl`. Šablony byly také lehce upraveny aby více odpovídaly požadavkům.

```
1 export function exportInEnw(publication:
    PublicationWithRelations): string | null {
2     const builder = new CitationsObjectBuilder(publication)
3     ;
4     const citationObject = builder
5         .addTitle()
6         .addEnwType()
7         // More attributes
8         ...
9     const ret = generateCitationFormat(citationObject, 'enw
10     ');
11     if (!ret) return null;
12
13     let lines = ret.split('\n');
14     // Final string fixes
15     ...
16     return lines.join('\n');
17 }
18
19 // File /lib/citationsExport/parsers/
20 // generateCitationFormat.ts
21 export default function generateCitationFormat(
22     citationObject: CitationObject,
23     type: 'bibtex' | 'enw' | 'refworks'
24 ): string | null {
25     let aux: string | null = null;
26     try {
27         const data = new Cite(citationObject);
28         aux = data.format(type as string, { format: 'text',
29             lineEnding: '\n' });
30     } catch (e) {
31         if (e instanceof Error) {
32             console.log(e.message as string);
33         }
34     }
35
36     return aux;
37 }
```

■ **Výpis kódu 2.21** Vytvoření EndNote export formátu.

2.4.6 Detail publikace

Po kliknutí na záznam v seznamu publikací je uživatel přenesen na jeho detail. Stránka obsahuje dvě záložky: detail a exporty. Podoba exportu a implementace se nachází v sekci 2.4.5. Záložka detail se skládá ze čtyř částí:

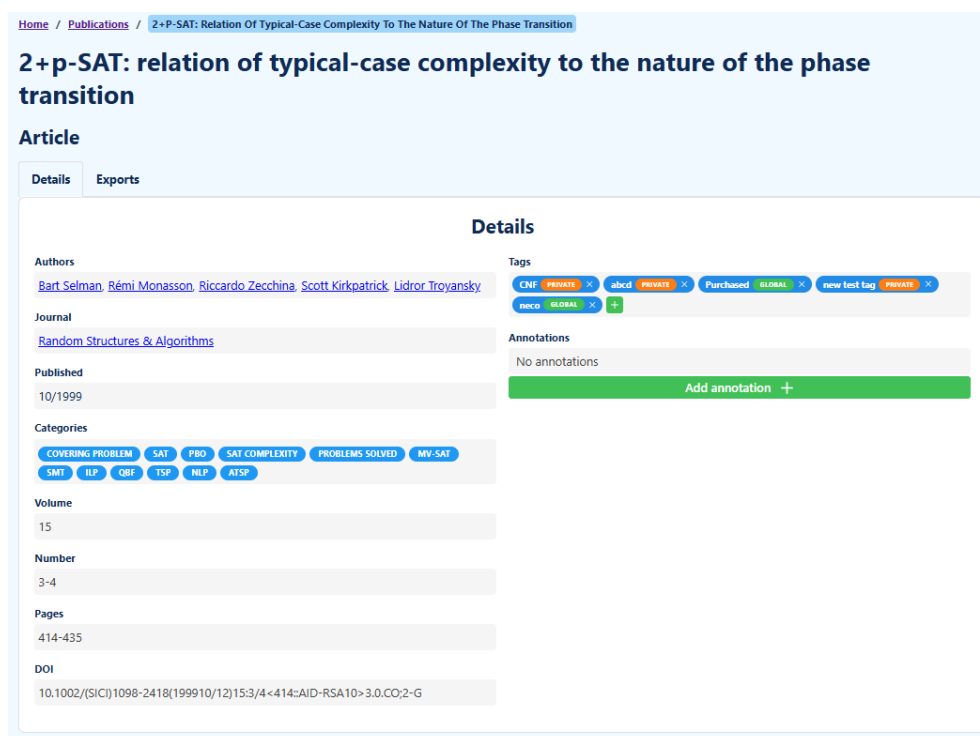
```
1 export default function generateCitationText(  
2   citationObject: CitationObject,  
3   type: 'ieee' | 'acm' | 'iso690',  
4   asText: boolean = false  
5 ): string | null {  
6   const templateName = type;  
7   const locale = csLocale;  
8  
9   let template;  
10  switch (type) {  
11    case 'acm':  
12      template = acmTemplate;  
13      break;  
14    case 'iso690':  
15      template = iso690Template;  
16      break;  
17    case 'ieee':  
18      template = ieeeTemplate;  
19      break;  
20  }  
21  
22  const config = plugins.config.get('@cs1');  
23  config.templates.add(templateName, template);  
24  config.locales.add('cs-CZ', locale);  
25  
26  let ret: string | null = null;  
27  try {  
28    const data = new Cite(citationObject);  
29    ret = data.format('bibliography', {  
30      format: asText ? 'text' : 'html',  
31      template: templateName,  
32      lang: type === 'iso690' ? 'cs-CZ' : 'en-US',  
33    });  
34  } catch (e) {  
35    if (e instanceof Error) {  
36      console.log(e.message as string);  
37    }  
38  }  
39  
40  return ret;  
41 }
```

■ **Výpis kódu 2.22** Funkce pro vytváření textové citace.

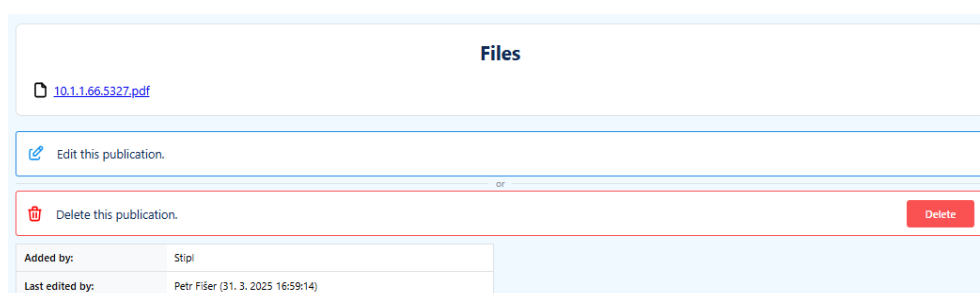
- Informace o publikaci a správa anotací a štítků. Správa vytvořena v práci [1].
- Soubory publikace.

- Tlačítka pro úpravu a smazání publikace.
- Metadata – kdo ji vytvořil a kým naposledy byla upravena.

V informacích o publikaci se ukazují pouze data, která nejsou null v databázi. Obrázky 2.11 a 2.12 ukazují finální verzi detailu publikace, kde pozice tlačítek pro úpravu a smazání byly přesunuty na spodní část stránky.



■ **Obrázek 2.11** První část detailu publikace obsahující informace o publikaci.



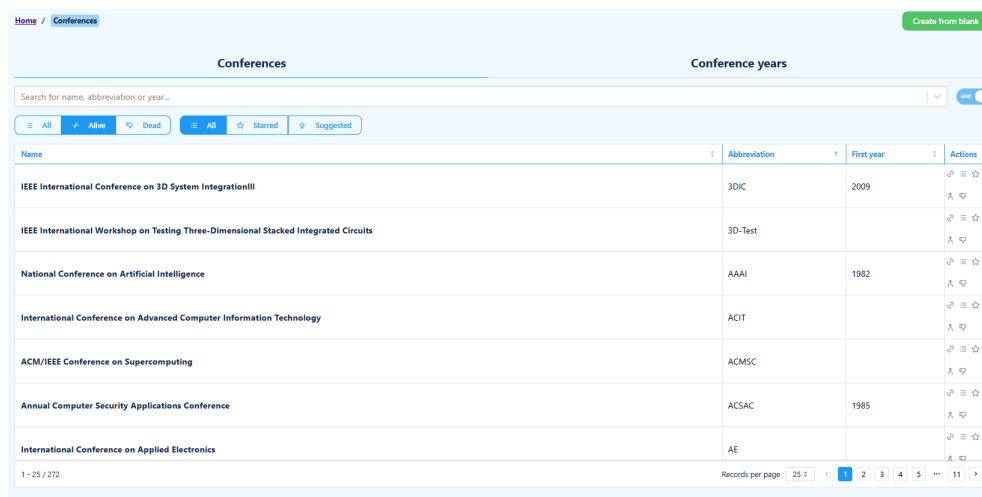
■ **Obrázek 2.12** Druhá, třetí a čtvrtá část detailu publikace obsahující informace o souborech, tlačítka a metadata.

2.5 Conference

Následující sekce se zaměří na implementaci správy konferencí, což zahrnuje vyhledávání a filtrování, vytváření, úpravu a mazání konferencí a ročníků konferencí. Dále sekce pokrývá implementaci správy dokumentových indexů a konferenčních kategorií, což zahrnuje jejich sjednocení s ACM kategoriemi.

2.5.1 Seznam

První co uživatel vidí, když naviguje na conference, je seznam konferencí. Samotný seznam je rozdělen na dvě záložky – „Conferences“ a „Conference years“. Jak názvy napovídají, jedná se o pohledy na dva různé seznamy – seznam samotných konferencí a seznam ročníků konferencí. Součástí seznamu je vyhledávací pole s filtračními tlačítky. Dále se na vrchní straně stránky nachází tlačítko pro navigaci na formulář pro vytvoření konference a jejích ročníků. Samotná stránka se nachází v souboru `app/conferences/page.tsx` a na obrázku 2.13 se nachází tento seznam.



The screenshot shows a web application interface for managing conferences. At the top, there are two tabs: "Conferences" (active) and "Conference years". Below the tabs is a search bar with the placeholder text "Search for name, abbreviation or year...". To the right of the search bar is a "Create from blank" button. Below the search bar are several filter buttons: "All", "Alive", "Dead", "All", "Starred", and "Suggested". The main content is a table with the following columns: "Name", "Abbreviation", "First year", and "Actions". The table contains several rows of conference data, including "IEEE International Conference on 3D System IntegrationIII", "IEEE International Workshop on Testing Three-Dimensional Stacked Integrated Circuits", "National Conference on Artificial Intelligence", "International Conference on Advanced Computer Information Technology", "ACM/IEEE Conference on Supercomputing", "Annual Computer Security Applications Conference", and "International Conference on Applied Electronics". At the bottom of the table, there is a pagination bar showing "1 - 25 / 272" and "Records per page" with a dropdown menu set to "25".

Name	Abbreviation	First year	Actions
IEEE International Conference on 3D System IntegrationIII	3DIC	2009	⌕ ⌵ ⭐
IEEE International Workshop on Testing Three-Dimensional Stacked Integrated Circuits	3D-Test		⌕ ⌵ ⭐
National Conference on Artificial Intelligence	AAAI	1982	⌕ ⌵ ⭐
International Conference on Advanced Computer Information Technology	ACIT		⌕ ⌵ ⭐
ACM/IEEE Conference on Supercomputing	ACMSC		⌕ ⌵ ⭐
Annual Computer Security Applications Conference	ACSAC	1985	⌕ ⌵ ⭐
International Conference on Applied Electronics	AE		⌕ ⌵ ⭐

Obrázek 2.13 Finální podoba seznamu konferencí.

Komponenta `ConferenceTabView` obsahuje implementaci záložek a podmíněné vykreslování seznamů. Samotné seznamy jsou následně v komponentách `ConferencesDataGrid` a `ConferenceYearsDataGrid`. Obě komponenty jsou implementací komponenty `MainSearchDataGrid`, která byla vytvořena kolegou [1], ale následně autorem práce upravena a zobecněna pro použití jak u publikací, tak konferencí. Mezi významné změny této komponenty patří přidání `ReactNode` pro filtrační tlačítka a zobecnění props. Ukázka kódu 2.23 představuje implementaci `MainSearchDataGrid` komponenty jako seznam ročníků konferencí.

V kódu 2.23 lze také vidět prop `columns`. Tento prop přijímá hook s návra-

```

1  return (
2    <MainSearchDataGrid
3      columns={useConferenceYearsColumns(
4        setOpenModalObject)}
5      noHeader={false}
6      searchPlaceholder={ConferenceSearchPlaceholder}
7      customSearchParams={
8        // Filter buttons
9        <Group>
10         <ConferencesFilterButtons
11           buttonLabels={visibilityButtonLabels}
12           activeButton={yearVisibilityActiveButton}
13           setActiveButtonAction={
14             setYearVisibilityActiveButton}
15           buttonIcons={visibilityButtonIcons}
16         />
17         ...
18       </Group>
19     }
20     onClick={(({ record }) => {
21       router.push(Conferences.getYearDetail(record.
22         conference_id, record.id));
23     })}
24     height="70dvh"
25   />
26 );

```

■ **Výpis kódu 2.23** Implementace seznamu konferenčních ročníků.

ovou hodnotou `DataTableColumn<T>[]`, kde generický typ `T` definuje možné sloupce a použitelné hodnoty v seznamu. Sloupcový hook pro seznam ročníků konferencí se nachází v souboru `components/datagrid/conferenceYears-DataGrid/useConferenceYearsColumns.tsx` a pro seznam konferencí v souboru `.../useConferenceColumns.tsx`, jež sdílí stejnou cestu.

Celý seznam, včetně vyhledávání a filtrování, je obalen poskytovatelem kontextu s názvem `ConferenceFilterContextProvider`. Jedná se o kontext, který obsahuje hooks pro udržování aktivní záložky a stavů filtračních tlačítek a je podřazený `MainSearchDataGridContextProvider` poskytovateli, který zastřešuje funkcionalitu komponenty `MainSearchDataGrid`. Ukázka kódu 2.24 ilustruje použití poskytovatele `MainSearchDataGridContextProvider` při vytvoření kontextu pro konference.

`MainSearchDataGridContextProvider` poskytuje kontext pro `MainSearchDataGrid`, zejména pro získávání dat, získávání nabídky v seznamu a další pomocné funkce. Získávání dat se zařizuje pomocí `useCallback` hooku, který v sobě volá prop `loadingFunc`, kde `loadingFunc` je funkce pro získávání dat. Kód 2.25 obsahuje tuto implementaci. Tato funkce `loadDataFunc` je následně

```

1  return (
2    <MainSearchDataGridContextProvider
3      loadingFunc={filterConferenceDataGrid}
4      storageKey={props.storageKey}
5      loadingFuncCustomArgs={[
6        activeTab as any,
7        conferenceVisibilityActiveButton,
8        yearVisibilityActiveButton,
9        preferenceActiveButton,
10       checkedCategoryIds,
11     ]}
12     suggestionsLoadingFunc={
13       activeTab === ConferenceTabLabels.conferences
14         ? fetchConferenceSuggestions
15         : fetchConferenceYearsSuggestions
16     }
17     defaultSortAccessor="abbreviation"
18   >
19     // New conference context
20     <Context.Provider
21       value={{
22         ...

```

■ **Výpis kódu 2.24** Vytvoření kontextu pro seznam konferencí a ročníků.

volána při každé změně periferií za účelem získávání dat.

Jak je tedy vidět v kódu 2.25, `loadingFunc` definuje rozhraní pro funkce, které jej musí následovat, aby se mohly zde použít. Pro zbylé vlastní argumenty funkcí slouží právě prop `loadingFuncCustomArgs`, kde se jedná o pole, které tyto argumenty obsahuje. Funkce `suggestionsLoadingFunc` je funkce pro získávání dat do našeptávače ve vyhledávacím poli. Seznam konferencí nabývá data pomocí serverové akce

```

filterConferenceDataGrid<T>(
  ...
  viewButtonLabel: string | null,
  ...):
  Promise<{ data: any[]; totalCount: number }>,

```

která se nachází v souboru `/lib/prisma/actions/conferences/getActions.ts`. Tato funkce na základě hodnoty `viewButtonLabel`, která obsahuje hodnotu aktuálně zvolené záložky, zavolá buď funkci `filterConferences`, pokud se jedná o konference, nebo `filterConferenceYears`, pokud se jedná o ročníky. Tyto funkce následně postaví objekty podle předaných filtrů a tyto objekty se následně použijí v Prisma `findMany` query jako filtry.

```
1 const loadDataFunc = useCallback(async () => {
2   setTransition(async () => {
3     try {
4       const response = await props.loadingFunc(
5         search?.map((opt) => opt.value) ?? [],
6         sortStatus,
7         chaining,
8         pageSize,
9         pageNumber,
10        ...props.loadingFuncCustomArgs
11      );
12
13      setData(response.data);
14      setTotalCount(response.totalCount);
15    } catch (err) {
16      console.log('filter provider error', err);
17    }
18  });
19 }, [search, pageNumber, pageSize, chaining, sortStatus,
  ...props.loadingFuncCustomArgs]);
```

■ **Výpis kódu 2.25** Implementace použití funkce pro získávání dat do seznamu.

Stavy filtračních tlačítek jsou předány zbylými vlastními argumenty:

- `conferenceVisibilityButtonLabel` – zobrazení mrtvých či živých konferencí.
- `yearVisibilityButtonLabel` – zobrazení archivovaných či živých ročníků.
- `preferenceButtonLabel` – zobrazení oblíbených či doporučených konferencí/ročníků.
- `checkedCategoryIds` – zvolené kategorie u ročníků.

Oba `*visibilityButtonLabel` argumenty porovnávají, zda-li se `state`³ atribut rovná danému filtru. Argument `preferenceButtonLabel` se dívá, zda-li konference má relaci v tabulce `Submitter favourite_conference`. Tato relace je pouze u konference, ale filtr je k dispozici i u ročníků. V tomto případě filtruje ročníky té konference, kterou má uživatel mezi oblíbenými.

Nakonec atribut `checkedCategoryIds` je pole idček zvolených kategorií, kterými se filtruje ročník. Jak se filtrují ročníky podle kategorií je znázorněno v ukázce kódu 2.26. Berou se ročníky, které mají relace pouze na kategorie definované v poli idček. Kvůli chybě v prismě bylo v objektu třeba udělat trik s prázdným `some` až následně definovat `every`.

³Tento atribut obsahují jak ročníky tak konference.

```

1 year_has_categories: {
2   some: {},
3   every: {
4     category_id: {
5       in: checkedCategoryIds.map((val) => BigInt(val)),
6     },
7   },
8 },

```

■ **Výpis kódu 2.26** Filtrace ročníku podle kategorií.

Za zmínku dále stojí filtrační možnost „Archived – last years“ u ročníků. Ročníky se nejprve seskupí podle `conference_id` atributu a udrží se u nich největší roky. Následně se získají identifikátory těchto ročníků a podle tohoto pole se získají záznamy.

2.5.1.1 Modály a akční tlačítka

U každého záznamu v seznamu se v posledním sloupci vyskytují akční tlačítka, která mohou otevřít buď modál, nebo provést změnu v databázi. Tabulka 2.5 ukazuje, jaká tlačítka jsou u konferencí a konferenčních ročníků k dispozici.

Typ	Konference	Ročníky
Modál	Ukázat související publikace	Ukázat související publikace
	Ukázat ročníky této konference	Ukázat workshopy tohoto ročníku
	Sjednotit konference	–
Akce	Přidat do oblíbených	Přidat do oblíbených
	Archivovat/oživit	Umrtnvit/oživit

■ **Tabulka 2.5** Tabulka akcí v posledním sloupci záznamu podle typu a kde se nachází.

Sloupec s akcemi je definován ve sloupcových hooks, probírány výše v sekci 2.5.1. Tyto hooks také obsahují kód logiky, která se má provést po kliknutí na tlačítko.

Tlačítka modálového typu vykreslí uživateli modál s požadovanými daty. Po kliknutí na tlačítko se naplní pomocný objekt vyžadovanými daty ze záznamu, jak je znázorněno v ukázce kódu 2.27. Komponenty s modály se nachází ve složce `app/conferences/components/datagrid/datagridModals`. Každý modál je obalen komponentou `DataGridActionModal`, která pomocí `switche` na základě typu vyžadovaného modálu (atribut `kind` v pomocném objektu) vykreslí modál, který v `useEffect` hooku získá data pomocí serverové akce. Modál se sjednocením konferencí se bude probírat v sekci 2.5.3.1.

Tlačítka akčního typu pouze zavolají požadovanou serverovou akci a pokud

je výsledek úspěšný, zavolají refresh funkci, která aktualizuje data v seznamu, jak lze vidět na ukázce kódu 2.27.

```
1 // Modal
2 onClick={ (e) => {
3   e.stopPropagation();
4   setOpenModalObject({
5     open: true,
6     conferenceId: id,
7     yearId: null,
8     kind: ModalType.years,
9     name,
10  });
11 }}
12
13 // Action
14 const EditFavouriteConferenceWithNotifications =
15   useErrorSuccessNotification(
16     updateFavouriteConference,
17     // Messages
18   );
19 ...
20 const starConference = async (id: bigint) => {
21   // Called in the onClick
22   const ret = await
23     EditFavouriteConferenceWithNotifications(id);
24   if (ret.state === 'success') {
25     refresh()
26   }
27 };
28
```

■ **Výpis kódu 2.27** Logika otevření modálu nebo provedení akce po kliknutí na tlačítko.

2.5.2 Vytváření a úprava

Pro účely vytváření konferencí a jejich ročníků je k dispozici formulář, jehož implementace je obdobná formuláři pro vytváření publikací, a na který se uživatel dostane kliknutím na tlačítko „Create from blank“, které se nachází na seznamu. Tedy formulář je opět obalen poskytovatelem kontextu do kterého je předán form objekt. Poskytovatel se vytváří v souboru `/lib/conferences/createConfFormService.ts` a následuje `ConferenceFormInterface` rozhraní, které obsahuje definice políček formuláře a pomocné atributy. Více obecných informací k formulářovému poskytovateli kontextu v sekci 2.4.1.1. U tohoto formuláře je opět respektován požadavek z `Issues`, že všechna data jsou do

databáze zapsána až po odeslání formuláře⁴.

Formulář se skládá ze dvou částí – část pro konference a část pro vytváření ročníků. Konferenční část obsahuje pouze políčka pro jméno, zkratku a web konference. V původní aplikaci se zde místo webu vyskytoval popis konference, který ale na přání vedoucího práce byl nahrazen.

Zde se také vyskytuje tlačítko pro správu předešlých jmen. Jedná se o tlačítko vedle políčka pro zkratku, které po kliknutí otevře seznam předešlých jmen a zkratk. Kliknutím na záznam v seznamu se původní jméno a zkratka dosadí do políček. Toto tlačítko slouží jako „rollback“ k některému z původních jmen. Pokud se jedná o vytváření nové konference, při odeslání formuláře se žádné jméno neneviduje. Naopak při úpravě konference, pokud uživatel změní jméno či zkratku, tato změna se eviduje a vloží se do nové tabulky `Conference_previous_names`. Uživateli se také ukáže varování, že mění jméno, které bude evidováno, a také tlačítko pro vymazání úpravy. Tento pohled lze vidět na obrázku 2.14. Funkce evidence předešlých jmen je implementována následně tak, že se porovná jméno a zkratka s původními hodnotami. Pokud je jeden z atributů jiný, vytvoří se záznam.

The screenshot shows a web form titled 'Conference'. It has two main input fields: 'Name' and 'Web'. The 'Name' field contains the text 'IEEE International Conference on 3D System Integration EDIT'. Above this field, a small black box displays the warning 'Changing the name!'. To the right of the 'Name' field, there is a button labeled 'rollback'. The 'Web' field contains the text 'IEEE International Conference on 3D System Integration | 3DIC'. Below the 'Web' field, there is a small table with two columns: 'Name' and 'Web'. The first row contains the text 'IEEE International Conference on 3D System Integration | 3DIC'.

Obrázek 2.14 Ukázka správy předešlých jmen s varováním a „rollback“ nabídkou.

Druhá část se skládá ze seznamu ročníků. Tento přístup byl zvolen proto, aby všechny ročníky dané konference byly na jednom místě a daly se spravovat všechny naráz. Seznam je tvořen komponentami `Dropdown` (viz 2.4.1.5), které obsahují rok, zkratku ročníku a tlačítko pro smazání ročníku. Dokud není zvolen rok, je místo něj pouze nápis „Year“.

Nové ročníky lze vytvořit kliknutím na + vedle nadpisu sekce „Years of conference“. Každý nový ročník má tmavší pozadí a vedle roku nápis „New“. Také se automaticky celý `Dropdown` rozevře a pohled stránky se přesune na nově vytvořený záznam v seznamu. Toto bylo zajištěno pomocí `useRef` hooku a `scrollIntoView` funkce.

Nový ročník při vytvoření automaticky dědí název a zkratku konference. Následuje logika dědění kategorií a popisů, která byla detailněji popsána v sekci 1.3 a je implementována ve funkční komponentě `AddNewYear`, která se nachází v komponentě `ConferenceForm`. Ukázka kódu 2.28 tuto logiku ilustruje.

Komponenta `Dropdown` po rozevření obsahuje vstupní pole pro vyplnění atributů daného ročníku. Mezi ně patří výběr vydavatelů, obalený komponentou `ImpoutWithAction` (viz 2.4.1.3), nabízející jejich správu a také výběr a správu konferenčních kategorií (stejný princip jako správa publikačních ka-

⁴Výjimku mají opět konferenční kategorie.

```

1  if (
2    form.getValues().created_new_year_after_merge &&
3    form.getValues().
4      latest_src_conference_year_id_after_merge &&
5    form.getValues().
6      latest_trgt_conference_year_id_after_merge
7  ) {
8    // Get the categories and description from the latest
9    // source year and from the latest target year
10   const highestSourceYear = ...
11   const highestTargetYear = ...
12
13   // Array without duplicates
14   value.checkedCategories = Array.from(
15     new Set([
16       ...(highestTargetYear?.checkedCategories ?? []),
17       ...(highestSourceYear?.checkedCategories ?? []),
18     ])
19   );
20   // Concat the descriptions
21   value.description = `${highestTargetYear?.description
22     ?? ''} ${highestSourceYear?.description ?? ''}`;
23 } else if (form.getValues().conference_years.some((val)
24   => val.yearId)) {
25   // Else infer the categories from the most recent year
26   // By the expression above at least one year should
27   // have an id.
28   const highestTargetYear = ...
29
30   value.checkedCategories = highestTargetYear.
31     checkedCategories;
32   value.description = highestTargetYear.description;
33 }
34 value.name = form.getValues().name;
35 value.abbreviation = form.getValues().abbreviation;

```

■ **Výpis kódu 2.28** Logika dědění kategorií a popisů.

tegorií popsána v 2.4.1.1). Obrázek 2.15 ukazuje finální podobu formuláře s ukázkou nového ročníku.

Oproti publikačnímu formuláři, metody, které jsou spuštěny po odeslání formuláře, se nachází v samotné komponentě formuláře v souboru `/app/conferences/components/ConferenceForm.tsx`. Která akce se provede, se rozhoduje podle `formMode` prop, který může být buď `create` nebo `edit`. Do komponenty lze předat vlastní funkce pro vytvoření či úpravu, které přepíší stávající. Po úspěšném vykonání je uživatel přenesen na detail konference nebo je tato akce opět přepsána propem `onSuccessAction`. Kód ilustrující

[Home](#) / [Conferences](#) / [IEEE International Conference On 3D System IntegrationIII](#) / [Edit](#)

IEEE International Conference on 3D System IntegrationIII

Fields and sections with * are required!

Conference

Name * × Abbreviation * ↩

Web

Years of conference (6) +

2009 : (3DIC) >

2015 : (3DIC) >

2016 : (3DIC) >

2018 : (3DIC) >

2028 : (3DIC) >

(New) Year : (3DIC) v

Name *	Abbreviation *	Year *
<input type="text" value="IEEE International Conference on 3D System Integration EDIT"/>	<input type="text" value="3DIC"/>	<input type="text" value=""/>
Submission deadline <input type="text"/>	Notification <input type="text"/>	
Final version <input type="text"/>	From-To <input type="text"/>	
Publisher <input type="text"/>	Indexed at <input type="text"/>	

+

Obrázek 2.15 Konferenční formulář s novým ročníkem.

toto chování se nachází v ukázce kódu 2.29.

```

1  const EditConferenceWithNotifications =
    useErrorSuccessNotification(
2      // Function override
3      updateConferenceAction ?? updateConference,
4      // Notification
5  );
6
7  const handleOnSubmit = async (values:
    ConferenceFormInterface) => {
8      startLoading(async () => {
9          let ret;
10         if (formMode === FormMode.create) {
11             ret = await CreateConferenceWithNotifications(
                values);
12         } else {
13             ret = await EditConferenceWithNotifications(values,
                initialData!, form.getDirty());
14         }
15         if (ret.state === 'success' && ret.data) {
16             typeof onSuccessAction === 'function'
17                 ? onSuccessAction()
18                 : router.push(Conferences.getConferenceDetail(ret
                .data.id));
19         }
20     });
21 };

```

■ Výpis kódu 2.29 Implementace přepsání funkce a odeslání formuláře.

Funkce

```

createConferenceAndYears(values: ConferenceFormInterface):
    Promise<Conference>

```

```

updateConference(
    data: ConferenceFormInterface,
    initialData: ConferenceFormInterface,
    dirtyFields: FormStatus): Promise<Conference>

```

jsou základní funkce, které vytvoří, případně upraví konference a související ročníky, pokud nejsou přepsány.

Při vytváření konference a ročníků se stránka s formulářem vykreslí ze souboru `app/conferences/create/page.tsx`. Po odeslání formuláře je spuštěna funkce `createConferenceAndYears`, která se nachází v souboru `/lib/prisma/actions/conferences/createActions.ts` a jedná se o transakci, ve které se nejprve upraví vydavatelé a následně se vytvoří konference s ročníky.

Formulář pro úpravu konference a ročníků se uživateli vykreslí ze souboru `app/conferences/[conferenceId]/edit/page.tsx`. Nejprve se z databáze získá konference se všemi jejími souvisejícími entitami, vytvoří se objekt následující `ConferenceFormInterface` rozhraní a ten se předá do komponenty přes `initialValues` prop.

Následně po odeslání formuláře je spuštěna funkce `updateConference`, která se nachází v souboru `/lib/prisma/actions/conferences/updateActions.ts` a jedná se o transakci, která nejprve upraví konferenci, následně smaže ISSN/ISBN identifikátory určené ke smazání (nové jsou vytvořeny či stávající upraveny při úpravě ročníku v posledním kroku pomocí Prisma metody `upsert`), upraví upravené vydavatele (nový je případně vytvořen a napojen v posledním kroku) a nakonec iterací přes všechny ročníky, které mají ve formuláři `id`, vymaže všechny relace ke každému ročníku, které pak ale v následující `update query` navrátí zpátky a upraví ročník (tímto je elegantně zaručeno propování jakékoli změny bez potřeby provnávání objektů).

Obě funkce jsou rozděleny ještě do funkcí `updateConferenceRaw` a `createConferenceAndYearsRaw`, které slouží pro postavení Prisma promises, které jsou následně vykonány v transakcích. Důvodem je nutnost vytvoření či úpravy konference v jiných transakcích a není rozumné tyto transakce vnořovat.

Přepsání funkcí se využije u sjednocení konferencí (viz sekce 2.5.3.1) a při úpravě či vytvoření konference z publikačního formuláře (sekce 2.4.1.3).

2.5.3 Detail konferencí a ročníků konferencí

Po kliknutí na záznam v seznamu konferencí či ročníku konferencí je uživatel navigován na odpovídající detail. Zde se dá bavit o tom, že detail ročníku konference je podřazen detailu konference, protože v první části obsahuje její informace. Navíc v URL a drobné navigaci je ročník vždy podvojen konferencí ve formátu

```
conferences/[conference_id]/year/[year_id]
```

a jinou cestou se na detail ročníku dostat nelze. Oba detaily obsahují dvě záložky:

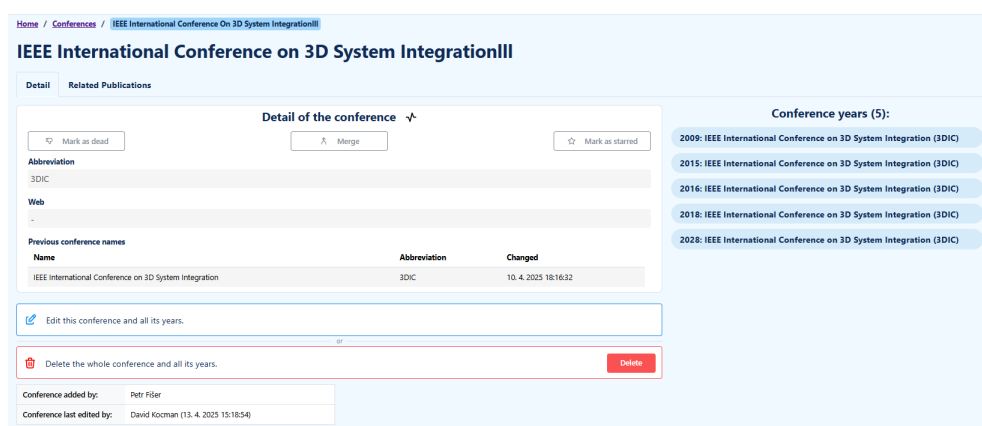
- Záložka Detail – obsahuje informace o konferenci či ročníku.
- Záložka Related publications – obsahuje související publikace konference (přes všechny ročníky) nebo ročníku.

Následně záložka detail je dále rozdělena na dvě části, kde dvě třetiny stránky jsou dedikovány zmíněným informacím a zbylá třetina obsahuje seznamy souvisejících entit. Pro konferenci je to seznam jejích ročníků. Pro ročníky jsou to seznamy ISSN/ISBN identifikátorů, ostatní ročníky konference a workshopy ročníku.

Detail konference se nachází v serverové komponentě v souboru `/app/conferences/[conferenceId]/page.tsx` a je obalen poskytovatelem kontextu s názvem `ConferenceDetailContextProvider`, do kterého je předána konference získaná z databáze. Mezi informacemi, které jsou zobrazeny na první záložce, je i tabulka předchozích jmen a metadata, která obsahují, kdo tuto konferenci vytvořil, a kdy a kým byla upravena.

Detail ročníku, kromě dat nadřazené konference, obsahuje všechny informace o ročníku (i pokud nejsou v databázi). Stránka se nachází v serverové komponentě `/app/conferences/[conferenceId]/year/[yearId]/page.tsx` a je obalena `ConferenceDetailContextProvider` poskytovatelem a také `ConferenceYearDetailContextProvider` poskytovatelem.

Na obou detailech je možné také vidět akční tlačítka, která se nacházela na seznamu. Nakonec ve spodní části se nachází tlačítka pro navigaci na editaci nebo smazání ročníku či celé konference. Obrázek 2.16 obsahuje detail konference a obrázek 2.17 pohled na její související publikaci.



Obrázek 2.16 Detail konference.



Obrázek 2.17 Seznam souvisejících publikací konference.

2.5.3.1 Sjednacení konferencí

Funkční požadavky pro sjednocení konference se nachází v sekci 1.3. Pro sjednocení konferencí je k dispozici modál, který se uživateli objeví po kliknutí

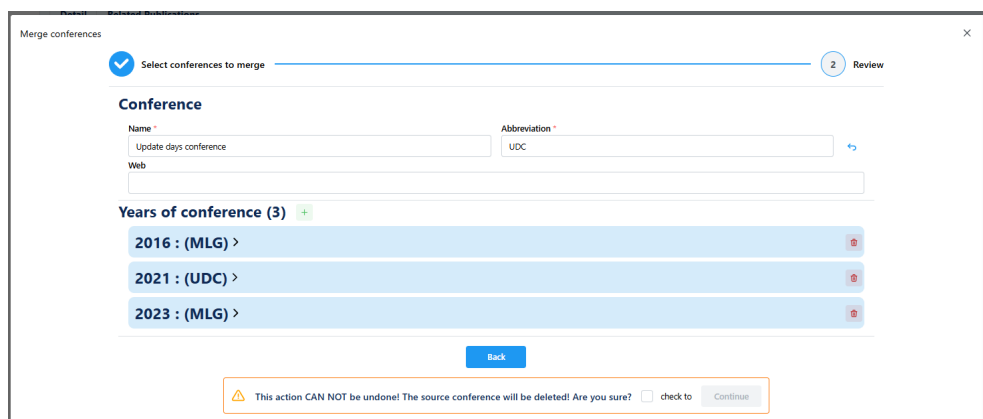
na akční tlačítko v záznamu nebo na tlačítko „Merge“ na detailu. Sjednocení konferencí má tři části:

- Vybrání zdrojové konference.
- Zkontrolování sjednocených konferencí.
- Závěrečná obrazovka se zprávou že vše proběhlo úspěšně.

Všechny funkce pro sjednocení konferencí se nacházejí v souboru `/lib/prisma/actions/conferences/mergeActions.ts` a modál s jeho sekcemi se nachází ve složce `/app/conferences/components/detailsSections/detailsModals`.

Cílová konference je vždy ta, u které stiskl uživatel tlačítko „Merge“ a nedá se změnit. Po vybrání zdrojové konference jsou následně obě na serveru sjednoceny ve funkci `fetchMergedConferences` a uživateli je představena výsledná podoba, která se nachází na obrázku 2.18. Zde nové atributy v `Conference` tabulce nabydou dané hodnoty. Tyto atributy, jakých hodnot mohou nabýt a k čemu slouží, lze následně nalézt v sekci 1.3.

Pokud je uživatel spokojen, pokračuje dále a funkce `updateMergedConference` v rámci transakce tyto změny propíše do databáze, což zahrnuje přemapování ročníků a předchozích jmen a smazání zdrojové konference.



■ Obrázek 2.18 Kontrola sjednocení dvou konferencí.

2.5.4 Správa a sjednocení kategorií

Na správu souvisejících entit konferencí, včetně konferenčních kategorií, se uživatel dostane přes postranní nabídku. Úvodní stránka správy byla inspirována ostatními úvodními stránkami správy entit, které byly vypracovány v [1], se záměrem zachování jednotného uživatelského rozhraní. Stránka se nachází v souboru `/app/conferences/categories/page.tsx` a skládá se ze

seznamu konferenčních kategorií, seznamu populárních kategorií⁵ a prostoru pro vytvoření nové kategorie. Tato stránka je při načítání nahrazena definicí *loading UI fallbacku* [6, sekce Routing/Loading UI and Streaming] v souboru `loading.tsx`.

Po kliknutí na záznam v jednom ze seznamů se uživatel přesune na detail dané kategorie, který také následuje rozhraní ostatních stránek správy, a obsahuje prostor pro úpravu a smazání kategorie a také prostor pro vytvoření podkategorie. Většina stránky je napsána v serverové komponentě v souboru `layout.tsx` a obalena komponentou `ConferenceRelatedEntitiesLayout`, která toto rozhraní definuje pro všechny stránky správy souvisejících entit.

Pro získávání stromu kategorií byla naprogramována serverová akce

```
fetchConferenceCategoryTree(categoryId: bigint | null):  
  Promise<ConferenceCategoryWithSubCategories[]>,
```

která ale kvůli velkému objemu dat zpomalovala načítání konferenčních stránek, protože se vždy vytvářel rekurzivně nový strom. Proto na výsledek této funkce je použita `unstable_cache` [6, sekce Functions/unstable_cache] funkce, která výsledek `fetchConferenceCategoryTree` kešuje buď po dobu jedné hodiny nebo dokud se ručně nerevaliduje. Kód 2.30 ilustruje definici funkce `getCachedConferenceCategoryTree`, která by se měla používat místo `fetchConferenceCategoryTree`.

```
1 export const getCachedConferenceCategoryTree =  
  unstable_cache(  
2    async (categoryId: bigint | null) =>  
      fetchConferenceCategoryTree(categoryId),  
3    ['conference-categories'],  
4    {  
5      revalidate: 3600,  
6      tags: ['conference-categories'],  
7    }  
8  );
```

■ **Výpis kódu 2.30** Kešovaná funkce `fetchConferenceCategoryTree`.

Jeden z funkčních požadavků na novou funkcionalitu bylo sjednocení a přemapování dosavadních konferenčních kategorií. V původní aplikaci mohla konference nabývat dvou druhů kategorií – `conference_category` a `acm_category`. Zde zanikne tabulka `acm_category`, všechny její kategorie se přesunou do tabulky `Conference_category` a všechny kategorie z této tabulky se přemapují na tabulku `Conference_year`.

Přemapování je implementováno pomocí Prisma migrací, ve kterých se nacházejí PostgreSQL skripty modifikací databázového schématu a pohybu dat.

⁵Seznam kategorií seřazených podle počtu všech relací.

Sjednocení probíhá ve třech krocích:

1. Přesun dat – pro uchování původních identifikátorů slouží dočasná tabulka, do které se vloží nově vložené ACM kategorie z tabulky `Conference_category`. Tyto identifikátory jsou třeba, protože následně se musí také přemapovat atributy `parent_id`, které odkazují na nadřízenou kategorii. Dočasná tabulka si uchovává tedy jak nové id, tak staré id. Následně proběhne zmíněné přemapování atributu `parent_id`, kde se upravují všechny nové kategorie, které mají id větší nebo rovno nejmenšímu novému id těchto kategorií. V ukázce kódu 2.31 se nachází SQL skript, který toto chování implementuje.

```
1 WITH selected AS (  
2     SELECT id AS old_id, name, parent_id  
3     FROM "Acm_category"  
4 ),  
5 inserted AS (  
6     INSERT INTO "Conference_category" (name, parent_id)  
7     SELECT name, parent_id  
8     FROM selected  
9     RETURNING id AS new_id, name, parent_id  
10 )  
11 INSERT INTO "new_acm_categories" (old_id, new_id, name)  
12 SELECT selected.old_id, inserted.new_id, inserted.name  
13 FROM selected  
14 JOIN inserted ON selected.name = inserted.name AND  
15 (selected.parent_id = inserted.parent_id OR (selected.  
16     parent_id IS NULL AND inserted.parent_id IS NULL));  
17 -- Update the parent_id in Conference_category based on  
18     the mapping  
19 UPDATE "Conference_category" c  
20 SET parent_id = acms.new_id  
21 FROM "new_acm_categories" acms  
WHERE c.parent_id = acms.old_id AND c.parent_id IS NOT  
    NULL AND c.id >= (SELECT MIN(new_id) FROM  
        new_acm_categories);
```

■ **Výpis kódu 2.31** Naplnění dočasné „new_acm_categories“ tabulky a přemapování `parent_id` atributu. Naplnění je docíleno za použití Common Table Expression, tedy vytvoření pojmenovaných poddotazů.

Nakonec je třeba také přesunout relace z tabulky `Conference_has_acm_category` do tabulky `Conference_has_category`. To se provede nejprve tak, že se relační idčka v tabulce `Conference_has_acm_category` přepíší na nové a následně všechna data jsou přesunuta do `Conference_has_category` tabulky.

2. Přemapování relací z konference na ročníky konference – je vytvořena nová relační tabulka `Conference_year_has_category`, která spojuje ročník a konferenční kategorie. Přemapování z konference na ročník je následně provedeno tak, že se do dočasné tabulky vloží idčka ročníku a kategorie získaná díky spojení tabulek `Conference_year` a `Conference_has_category` přes atribut `conference_id`. To má za důsledek, že kategorie, která byla přiřazena konferenci teď bude přiřazena každému ročníku. Následně jsou tato data přesunuta do nové tabulky.
3. Smazání nadbytečných tabulek – tabulky `ACM_category`, `Conference_has_acm_category` a `Conference_has_category` jsou smazány z databáze.

Všechny tyto migrace lze najít ve složkách:

- `/prisma/migrations/20250307114024_merge_categories_remap_relations,`
- `/prisma/migrations/20250307140916_remap_categories_to_years,`
- `/prisma/migrations/20250307142830_new_conf_col_drop_redundant_cat_tables,`

2.5.5 Správa databází dokumentových indexů

Na správu databází dokumentových indexů se uživatel dostane přes postranní nabídku. Úvodní stránka, nacházející se v souboru `/app/conferences/document-indexes/page.tsx` se skládá ze stejných elementů, jako správa konferenčních kategorií, viz sekce 2.5.4. Stránka detailu indexu se nachází v souboru `/app/conferences/document-indexes/(document-indexes)/[document-Id]/page.tsx`.

Stejně jako u kategorií je zde většina stránky definována v souboru `layout.tsx` a obě stránky jsou nahrazeny `loading.tsx` fallbackem, dokud jsou data načítána z databáze a stránky renderovány.

O zápis do databáze a získávání dat se starají serverové akce, které se nacházejí v souboru `lib/prisma/actions/documentIndexActions.ts`.

2.6 Nová podoba databáze

Obrázek 2.19 znázorňuje ER diagram aktuální databáze. Modře jsou vyznačeny nové tabulky.

Tato práce vytvořila novou tabulku `Conference_previous_names` pro ukládání historie jmen a zkratk konferencí a smazala tabulku `acm_category`, jejíž data jsou teď v tabulce `Conference_category`.



Obrázek 2.19 ER diagram aktuální databáze po změnách. Modře jsou vyznačeny nové tabulky.

Kapitola 3

Testování

Následující kapitola pojednává zejména o uživatelském testování použitelnosti. Nejprve budou předvedeny scénáře, kterých se testeři drželi, a následně proběhne shrnutí testů a návrh vylepšení. Nakonec se kapitola zaměří na funkční testování a automatické integrační testy.

3.1 Uživatelské testování

Pro uživatelské testování jsme zvolili testování použitelnosti, tedy jak je snadné systém používat a jak rychle se uživatelé naučí jeho ovládání.

Sekce 3.1.1 a 3.1.2 obsahují scénáře pro část aplikace vytvořenou v této diplomové práci a snaží se pokrýt všechny její vlastnosti a hlavní use-cases. Pro scénáře druhé části viz [1].

Testování uživatelé jsou dva studenti FITu, jeden čerstvý absolvent a studentka VŠE, která slouží jako externí uživatel.

Uživatelé před zahájením testu dostali vstupní dotazník, který obsahuje místo pro vyjádření, jak moc jsou s doménou konferencí a publikací seznámeni, a jestli souhlasí s tím, že jsou natáčeni. Jeho znění lze najít v [1]. Po vypracování scénářů byl uživatelům dán výstupní dotazník, kde odpověďmi na pár otázek podali zpětnou vazbu. Vyplněné dotazníky lze vidět v přílohách A, B, C, D a níže v sekci 3.1.3 lze vidět vyhodnocení testování, které čerpalo jak ze zpětné vazby z dotazníků, tak z výsledovaného chování uživatelů.

Provádění testu probíhalo tak, že jeden z nás vždy dělal moderátora, který četl scénáře toho druhého, a druhý sledoval reakce uživatelů při řešení scénářů. Pouze u posledního testování se studentkou VŠE tyto činnosti dělal jeden člověk.

V sekci 3.1.3 budou průběhy testování popsány. Nejprve se sepíše popis testování – jak uživatel testy procházel, jestli splnil očekávané kroky a jaké byly postřehy uživatele. Následně bude sepsána zpětná vazba ze závěrečného dotazníku, kde bude kladen důraz na názory, které nezazněly u testování. První uživatel bude sepsán detailně, u zbylých se bude nacházet jen výtažek nej-

zajímavějších chyb či komentářů.

3.1.1 Scénáře Publikace (SP)

Následující scénáře pokrývají vytváření publikací se správou přidružených entit, import publikace přes definici a Springer API a v poslední řadě také smazání a úpravu publikace.

SP1: Vytvoření publikace

Chcete si uložit článek s názvem „Pokroky v AI“ od autora Bryana Johnsona z časopisu „Nature“, co vás nedávno zaujal. Veškeré informace o něm máte k dispozici, tedy víte, že byl vydán v červenci 2021, jednalo se o 2. svazek (volume) a máte u sebe i PDF soubor s daným článkem. Při vytváření ale zjistíte, autor Bryan Johnson v databázi PubConf není, takže ho musíte přidat. Vytvořte nový záznam v aplikaci PubConf.

Očekávané kroky:

- Uživatel na seznamu publikací klikne na tlačítko „Create from blank“.
- Uživatel zadá název „Pokroky v AI“, změní typ publikace na „Article“ a vyplní rok a měsíc vydání.
- Uživatel klikne na tlačítko + pod autory a vytvoří nového autora se jménem Bryan Johnson.
- Uživatel najde a zvolí časopis.
- Uživatel rozbalí druhou sekci formuláře, vyplní svazek (volume) a nahraje soubor do vyznačené zóny.
- Po kliknutí na tlačítko „Submit“ uživatel vytvoří publikaci.

SP2: Úprava publikace

Po vytvoření předchozí publikace jste zjistili, že jste zadali špatně název a správně má znít „AI v kostce“.

Očekávané kroky:

- Uživatel na detailu publikace klikne na tlačítko „Edit this publication“.
- Uživatel změní název na „AI v kostce“.
- Uživatel klikne na tlačítko „Submit“ a upraví publikaci.

SP3: Import publikace přes definici

K dispozici máte BibTeX definici *inproceedings* záznamu. Po vytvoření záznamu chcete tuto publikaci citovat v jedné ze svých prací ve formátu IEEE. Importujte tuto publikaci do aplikace PubConf, vytvořte nový záznam a exportujte tento záznam ve formátu IEEE.

Očekávané kroky:

- Uživatel zkopíruje poskytnutou definici a po kliknutí na šipečku vedle tlačítka „Create from blank“ na seznamu publikací vybere možnost **Import from definition**.
- Uživatel vloží definici do pole a klikne na tlačítko „Import“.
- Uživatel zkontroluje vložené hodnoty, klikne na tlačítko „Submit“ a vytvoří publikaci.
- Na stránce detailu uživatel naviguje na záložku „Exports“, najde IEEE export a kliknutím na tlačítko „Copy the citation“ zkopíruje citaci.

SP4: Smazání publikace

Chcete smazat předchozí *inproceedings* záznam. Smažte publikaci v aplikaci PubConf.

Očekávané kroky:

- Uživatel najde předchozí publikaci a na stránce detailu zaškrtně políčko pro odemčení smazání publikace a smaže publikaci.

SP5: Import publikace přes Springer API

Chcete importovat článek ze Springeru. Je to v aktuálním stavu možné?

Očekávané kroky:

- Uživatel klikne na šipečku vedle „Create from blank“ tlačítka na seznamu publikací a zvolí možnost „Import from Springer API“.
- Podle informací zobrazených na modálu uživatel odpoví ne, protože chybí v nastavení uživatelů Springer API klíč.

3.1.2 Scénáře Konference (SK)

Následující scénáře pokrývají vyhledávání samotných konferencí se zobrazením souvisejících publikací, následně také vyhledávání ročníků pomocí filtračních

tlačítek a možnosti změny stavu ročníků a přidání do oblíbených. Dále vytvoření konference a ročníků konference, vytvoření přidružené entity přes její detail a následné úpravy ročníku za účelem napojení této nové entity. V poslední řadě scénáře pokryjí sjednocení dvou konferencí.

SK1: Vyhledání konference a zobrazení publikací

Chcete si vyhledat konferenci s názvem „IEEE European Test Symposium“ a zobrazit si publikace, které s touto konferencí souvisí. Vyhledejte tuto konferenci v aplikaci PubConf.

Očekávané kroky:

- Uživatel na seznamu konferencí zadá do vyhledávacího pole název „IEEE European Test Symposium“.
- Uživatel v seznamu klikne na tlačítko „Show associated publications with this conference“, které se nachází v posledním sloupci.

SK2: Vyhledání ročníku podle kategorie

Chcete si zobrazit všechny archivované ročníky, které obsahují kategorii „Cryptography“ a jsou z roku 2016. Tyto ročníky chcete nastavit na živé a dát si je do oblíbených (ohvězdičkovat). Nalezněte ročníky podle těchto kategorií, proveďte předem zmíněné úkony a zobrazte si detail libovolného z nich.

Očekávané kroky:

- Uživatel na seznamu konferencí překlikne na záložku „Conference years“, klikne na tlačítko „Categories“, vybere kategorii „Cryptography“ a do vyhledávacího pole zadá rok 2016.
- Uživatel překlikne filtrovací tlačítko z „Alive“ na „Archived“.
- V posledním sloupci klikne na tlačítko „Set as alive“ a „Star“.
- Poté klikne na řádek a naviguje na detail.

SK3: Vytvoření konference

Chcete si uložit záznam o nové konferenci a dvou jejích ročnících, které proběhly v letech 2022 a 2023. Konference se nazývá „Update days conference“ se zkratkou UDC. Jednotlivé ročníky se nazývají stejně a mají stejnou zkratku. Každý ročník probíhá vždy první týden v květnu a spadá do kategorie definované cestou „HW/Asynchronous circuits“. Vytvořte nový záznam v aplikaci PubConf.

Očekávané kroky:

- Uživatel klikne na „Create from blank“ tlačítko, které se nachází na seznamu konferencí a vyplní název a zkratku konference.
- Rozbalí první ročník a vyplní název, zkratku, rok a datum ročníku. Poté vybere kategorii v nabídce kategorií.
- Kliknutím na + vedle nadpisu „Years of conference“ vytvoří další ročník a vyplní stejné informace.
- Uživatel klikne na tlačítko „Submit“ a vytvoří novou konferenci.

SK4: Vytvoření nové indexové databáze

Chcete přidat novou Document index database do aplikace PubConf s názvem „Science files“, mající web „www.files.com“. Vytvořte nový záznam dokumentového indexu v aplikaci PubConf.

Očekávané kroky:

- Uživatel v levé nabídce klikne na „Document index databases“ a naviguje na správu dokumentových indexů.
- V části „Add document index“ uživatel vyplní jméno a web a kliknutím na „Add“ vytvoří nový index.

SK5: Úprava konference

Chcete, aby skutečnost, že poslední ročník „Update Days Conference“ je indexován ve „Science files“ dokumentovém indexu, byla uložena v aplikaci PubConf. Upravte poslední ročník konference „Update days conference“.

Očekávané kroky:

- Uživatel naviguje na seznam konferencí a buď vyhledá celou konferenci „Update days conference“ a naviguje na detail, nebo se přepne na záložku „Conference years“, zadá do vyhledávacího pole „Update days conference“, v seznamu ročníků najde poslední ročník a klikne na řádek pro navigaci na detail.
- Na stránce detailu klikne na modré tlačítko „Edit this conference and all its years“, na stránce úpravy konference uživatel rozbalí poslední ročník a v políčku „Indexed at“ vybere nově vytvořený index.
- Uživatel po kliknutí na tlačítko „Submit“ upraví konferenci.

SK6: Sjednocení konferencí

Po upravení jste zjistili, že se vlastně jedná o aktuální pokračování konference „Update weeks conference“, kterou jste již přidali a je třeba tuto konferenci sjednotit do aktuální konference. Sjednoťte tyto dvě konference v aplikaci PubConf.

Očekávané kroky:

- Uživatel na stránce detailu konference klikne na tlačítko „Merge“ a na první stránce modálu vyhledá zdrojovou konferenci „Update weeks conference“.
- Po kliknutí na tlačítko „Continue“ uživatel zkontroluje, zda jsou informace sjednocených konferencí správně a po zaškrtnutí políčka klikne na odemčené tlačítko „Continue“.
- Po sjednocení konferencí klikne na tlačítko „Finish“.

3.1.3 Vyhodnocení testování

Jak bylo zmíněno, testu se zúčastnili čtyři lidé, kteří byli nahrávání a tyto nahrávky lze najít v příloženém médiu. Scénáře testerům zabralo projet v průměru 21 minut a 50 sekund.

U každého testu byly psány poznámky a postřehy jednotlivých testovaných subjektů. Následovně budou testeři nazýváni jako uživatelé a průběh testování uživatele číslo 1 bude popsán detailně. U zbývajících bude k dispozici shrnutí.

Uživatel 1: student FIT

- **SP1** – Uživatel se na začátku seznámil s rozhraním tlačítka pro vytváření publikací, což velmi usnadnilo orientaci při budoucím importování, a úspěšně přešel na formulář. S vyplněním atributů (i skrytých ve druhé sekci) si poradil a také úspěšně přidal nového autora. Soubor nahrál a vytvořil publikaci. Zapomněl ale změnit typ publikace na *Article* a s tím i přiřadit časopis. Tato chyba se v průběhu testování opakuje často.
- **SP2** – Splnil. Uživatel našel tlačítko pro navigaci na úpravu publikace a úspěšně změnil jméno.
- **SP3** – Uživatel se nejprve snažil importovat publikaci přes metodu „drag and drop“, následně ale text zkopíroval, vložil do vstupního pole a po kliknutí na tlačítko „Import“ jej nahrál do formuláře. Následně se snažil publikaci exportovat ještě ve vytvářecím formuláři, ale po ujistění úspěšně publikaci vytvořil. Následně na detailu publikace navigoval na záložku s exporty a požadovaný formát zkopíroval přes tlačítko.
- **SP4** – Splnil. Uživatel úspěšně našel tlačítko pro smazání publikace.

- **SP5** – Splnil. Uživatel správně odpověděl „ne“.
- **SK1** – Splnil. Uživatel úspěšně našel konferenci „IEEE European Test Symposium“, ale následně nevyužil možnosti zobrazení souvisejících publikací přímo ze seznamu, ale navigoval na detail a tam přešel na záložku se souvisejícími publikacemi, což ale v konečném důsledku není špatně.
- **SK2** – Splnil. Uživatel úspěšně vyfiltroval ročníky, změnil jejich stav a přidal do oblíbených. Následně navigoval na detail.
- **SK3** – Uživatel úspěšně navigoval na formulář pro vytvoření konference a konferenčních ročníků, úspěšně přidal ročník, vyplnil atributy a přiřadil kategorie. Šlo vidět, že uživatel se pozastavil nad výběrem datumu „From–To“, který i přes zvolený ročník ukazuje aktuální měsíc a rok. Následně uživatel vytvořil konferenci.
- **SK4** – Uživatel nejprve hledal správu databáze dokumentových indexů v sekci administrace. Po nevyhledání správy v této sekci přešel zpět do sekce s konferencemi a na správu úspěšně navigoval. Následně vytvořil dokumentový index.
- **SK5** – Uživatel úspěšně našel konferenci a po navigaci na detail úspěšně našel v seznamu poslední ročník. Nejprve se ale snažil databázi dokumentových indexů přiřadit na samotném detailu. Po zorientování se na stránce přešel na úpravu konference a úspěšně dokumentový index přiřadil.
- **SK6** – Uživatel nejprve začal hledat zdrojovou konferenci, tedy snažil se sjednotit „Update days conference“ do „Update weeks conference“, což je naopak. Po seznámení se s uživatelským rozhraním si uživatel ujistil otázku, zjistil, že proces dělá naopak a následně jej dokončil správně. Toto pochybení je u ostatních uživatelů časté a důvodem byla neupřesněná otázka.

Ve výsledku si uživatel číslo jedna vedl velmi výborně. Jde to vidět zejména u publikací, kde úspěšně splnil tři scénáře z pěti. Nejzávažnější chyba byla u scénáře **SP1**, což bylo nezměnění typu publikace a nevyplnění časopisu.

V dotazníku uživatel navrhl i pár vylepšení pro aplikaci, mezi které patří kopírování ročníků a možnost výměny zdrojové a cílové konference při sjednocení. Následně vytkl koncept exportu v této aplikaci. Závěrečný dotazník prvního uživatele je k dispozici v příloze A.

Uživatel 2: absolvent FIT

U scénáře **SP1** udělal druhý uživatel stejnou chybu, že opět nezměnil typ publikace na „Article“. Následně uživateli chvíli trvalo najít skrytou sekci a při vytváření zapomněl vyplnit rok, což vyhodilo error. Zbylé publikační scénáře byly úspěšně splněny.

U **SK1** uživatele zmátla omezená nabídka našeptávání při vyhledávání a kvůli tomu měl připomínky k vyhledávání. Následně nevyužil tlačítka pro zobrazení souvisejících publikací, ale také šel na detail, jako první uživatel.

U druhého konferenčního scénáře **SK2** uživatel hned nenašel záložku s konferenčními ročníky. U vyhledávání kategorií uživateli vadilo, že je neoptimalizované a seká se. Uživatel nefiltroval podle roku, protože nebylo nikde očividně napsáno, že vyhledávací pole mimo jiné také filtruje podle roku. Nakonec uživatel změnil stav ročníku až na jeho detailu a vytkl, že nejde vidět zpětná vazba (měnící se ikonku vedle nadpisu „Detail of this year“ a v samotném tlačítku přehlédl).

U třetího scénáře **SK3** si nebyl uživatel jistý, kam zadat datum od–do a jak zprvu přidat kategorii. Při filtrování kategorií uživateli vadilo, že nemizí, ale že se vyhledávaný výraz jen zvýrazňuje.

U posledního scénáře, tedy **SK6**, uživatel sjednotil konference naopak.

V dotazníku uživatel vytkl pomalé vyhledávání našeptávače u konferencí a publikací a rozdílné tooltipy u akčních tlačítek. Dotazník lze najít v příloze B.

Uživatel 3: student FIT

Třetí uživatel si u prvního publikačního scénáře **SP1** vůbec nebyl jist, jak má začít. Po přechodu na formulář pro vytváření publikací nevěděl, kam má zadat časopis. Po pár ujasněních změnil typ a následně vyplnil atributy. Úspěšně našel skryté atributy a nahrál soubor.

Ve třetím publikačním scénáři **SP3** si nebyl jist, jak exportovat a hledal export na formuláři před vytvořením.

U konferenčního scénáře **SK1** opět publikace vyhledal přes detail.

U třetího konferenčního scénáře **SK3** uživateli chvíli trvalo, než našel tlačítko pro přidání ročníku. Uživateli vadil výběr data od–do, že se výběr nepřizpůsobil zvolenému roku. Nakonec při kliknutí na tlačítko „Submit“ uživatele překvapila chybová hláška chybného políčka, které nemohl najít, protože bylo schováno v rozevíratelné sekci ročníku.

U posledního scénáře **SK6** uživatel sjednotil konference opačně.

V dotazníku následně uživatel vytkl způsob vybírání typu publikace, výběr data při vytváření ročníku konference a že by nové ročníky mohly od ostatních dědit některé atributy, když se vytváří (což se již děje, viz sekce 2.5.2, uživatel si ale této funkcionality nevšiml). Navíc se uživateli zdály některé obrázky na tlačítkách v aplikaci zavádějící a tlačítka na detailu konference „mrtvé“, tedy výběr barvy se uživateli zdá nevhodný. Závěrečný dotazník třetího uživatele je k dispozici v příloze C.

Uživatel 4: studentka VŠE

Uživatel oproti ostatním první publikační scénář **SP1** zvládl mnohem lépe, zvládl úspěšně změnit typ a vyplnit atributy. Následně u třetího publikačního

scénáře **SP3** uživatel nebyl seznámen s formátem IEEE a nechápal koncepci exportování publikací, což vedlo k nedorozumění.

U druhého konferenčního scénáře **SK2** se uživatel nejprve nepřeklikl na ročníky. Následně uživatel zprvu zadával kategorii do vyhledávacího pole a navíc nevyhledal podle roku¹.

U třetího konferenčního scénáře **SK3** uživatel váhal, kam má zadat od-do. Uživatel navíc poprvé zapomněl přidat nový ročník a vytvořil konferenci pouze s jedním, po ujasnění úkolu uživatel úspěšně přidal další ročník.

Ve scénáři **SK4** uživatel nenašel stránku se správou databází dokumentových indexů, po nápovědě ale úkol dokončil úspěšně.

Znění posledního scénáře **SK6** bylo od posledního testování upraveno, aby uživateli byl lépe vysvětlen záměr sjednocení, a díky tomu uživatel dokončil tento scénář úspěšně.

V dotazníku uživatel nevytkl žádný nový nedostatek. Naopak ocenil „drag and drop“ funkcionalitu nahrávání souborů. Závěrečný dotazník čtvrtého uživatele je k dispozici v příloze D.

3.1.3.1 Tabulka připomínek

Tabulka 3.1 níže obsahuje souhrn připomínek, které byly vypořádány při testování nebo zmíněny v dotaznících. Připomínka bude nabývat jedné ze tří úrovní závažnosti:

1. **Nedostatek** – chyba, která má minimální nebo žádný dopad na práci uživatele.
2. **Chyba** – chyba, která nezamezuje chodu systému, ale ztěžuje práci uživateli.
3. **Závažná chyba** – chyba, která zamezuje dalšímu chodu systému a musí být opravena.

Připomínka	Závažnost	Návrh řešení
Špatné přepínání typu publikace	Chyba	Při kliknutí na tlačítko „Create from blank“ zobrazit modál, kde si uživatel zvolí prvotní typ publikace. Následně výběr typu přesunout na viditelnější místo.

¹V nahrávce lze vidět, že při ohvězdičkování ročníku se ohvězdičkují také další ročníky, které patří té samé konferenci. Toto je moderátorem mylně označeno jako chyba. V aplikaci se do oblíbených přidává **konference**, tedy tím pádem se označí jako oblíbené všechny ročníky, které do té konference patří.

Připomínka	Závažnost	Návrh řešení
Nevyužití tlačítka pro zobrazení souvisejících publikací u záznamu konference.	Nedostatek	Zvýraznění této možnosti.
Nezadání roku ročníku konference do vyhledávacího políčka.	Chyba	Napsání možností, které atributy lze vyhledávat do zástupného textu vyhledávacího pole.
Neoptimalizovaná filtrace konferenčních kategorií.	Chyba	Tato chyba se blíží závažné chybě. Z důvodu velkého počtu kategorií přesunout filtraci z front-endu na back-end, implementace pomocí serverové akce.
Kategorie nemizí při filtrování, ale místo toho se hledaná fráze zvýrazní.	Chyba	Osekávat získané pole kategorií.
Aktuální rok při výběru od-do v konferenčním ročníku.	Chyba	Při změně roku při vytváření ročníků nastavit tuto hodnotu do komponenty.
Přehlédnutí od-do políčka při zakládání ročníku konference.	Nedostatek	Možná jen zvýraznit a zdůraznit účel.
Nevýraznost postranní nabídky.	Nedostatek	Zvýraznění postranní nabídky.
„Mrtvé“ tlačítka v detailu konference.	Nedostatek	Změna barvy, například na modré, jako jsou na seznamu.
Nepochopení koncepce exportu v této aplikaci.	Nedostatek	Přidat na vršek export záložky pomocný text.
Nesjednocené nápovědy u akčních tlačítek.	Nedostatek	Sjednotit Tooltip komponentu.

■ **Tabulka 3.1** Výsledná klasifikace připomínek.

3.2 Funkční testování

Celá aplikace byla při vývoji řádně funkčně testována vývojáři a vedoucím práce. Také byly vyvinuty automatické integrační testy, které se spouští při každém sestavení aplikace (v rámci CI/CD pipeline).

Připomínky vzniklé při funkčním testování byly zaznamenány v GitLab issues a některé byly zapracovány. Seznam zapracovaných issues z nového GitLabu lze vidět jak v uzavřených **Issues**, tak níže v sekci 3.3.

Integrační testy jsou napsány ve frameworku **Vitest** [25] a nachází se ve

složce `__tests__`. Testují se serverové CRUD akce, na začátku před každým testem se připraví kontext, tedy se vyčistí adekvátní tabulky a naplní se testovacími daty. Následně po provedení testované funkce se provede aserce. Pro účely testování je použita testovací databáze. Vzorový průběh testu lze vidět v ukázce kódu 3.1. V této práci byly vytvořeny testy na funkce vytvořené v rámci zadání.

```
1 test('updateConference updates a conference and year',
2     async () => {
3       const newValues: ConferenceFormInterface = {
4         ...
5       };
6
7       const updatedConference = await updateConference(
8         newValues, originalConference, {});
9
10      expect(updatedConference.name).toBe('Test Conference
11        New Name');
12
13      const years = await prisma.conference_year.findUnique({
14        where: {
15          id: BigInt(1),
16        },
17      });
18
19      expect(years).not.toBeNull();
20      expect(years!.name).toBe('...');
21    });
```

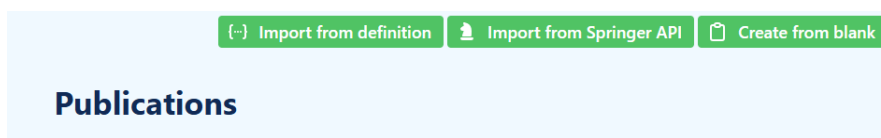
■ **Výpis kódu 3.1** Průběh testu úpravy konference a jejího ročníku.

Jak bylo řečeno, testy se pustí při každém běhu CI/CD pipeline. Nastavení této pipeline a Vitest frameworku lze najít v [1].

3.3 Zpracované změny

Některé jednoduché a potřebné změny byly zpracovány hned po testování. Níže je seznam zpracovaných změn, na kterých pracoval jak autor práce, tak kolega [1]. Změny z uživatelských připomínek:

- Vylepšení zástupného textu ve vyhledávacím políčku konferencí.
- Změna tlačítka pro vytváření či importaci publikace – místo jednoho tlačítka s rozbalovací nabídkou jsou všechny tři tlačítka viditelné, jak lze vidět na obrázku 3.1.
- Změna designu postranní nabídky



Obrázek 3.1 Nová podoba tlačítek pro vytvoření či import publikací.

- Seznam požadavků pro vytvoření nového hesla a další vstupní pole pro potvrzení hesla.
- Vylepšené vytváření ISSN/ISBN identifikátorů u časopisů.
- Vylepšení našeptávačů.
- Odstranění zaškrtačovacího tlačítka u mazání.
- Vylepšení funkce „Me“ tlačítka – vyplní i uživatelské jméno.

Následně byly také zapracovány připomínky z funkčních testování a **Issues** vytvořených v GitLab repozitáři:

- Přidání možnosti editace konference z publikačního formuláře².
- Změna pozadí nově přidaného ročníku a automatické přesunutí pohledu³.
- Možnost změny hierarchie všech kategorií.
- Přesunutí tlačítek pro úpravu či smazání na spodek stránky.
- Persistence publikačních a konferenčních filtrů při změnách stránek.
- Opravy akčních tlačítek u seznamu publikací.
- Přesunutí výběru typu publikace na první místo pod nadpis „General“.
- Na DOI v detailu publikace lze klikat.

3.4 Návrhy na vylepšení aplikace

Nejprve by bylo třeba dodělat zbylé funkcionality původní aplikace, které nebyly požadovány za základní. Mezi ně patří:

- Počítání referencí – jedná se o náročnou operaci, která zjišťuje vzájemné citace mezi publikacemi, tedy které publikace citují jiné a ve kterých jsou citovány.
- Konferenční skupiny

²Tato logika byla důkladně zdokumentována v sekci 2.4.1.3.

³Zdokumentováno v sekci 2.5.2.

- Vlastní atributy – správa již je implementována, ale chybí implementace příznaku viditelnosti v různých publikačních skupinách.

Následně některé z připomínek výše:

- Optimalizace filtrování kategorií v seznamu konferenčních ročníků.
- Mizení vyfiltrovaných kategorií ve výběru na seznamu konferenčních ročníků.
- Další vylepšení vybírání typu publikace (například implementace zmíněného modálu z tabulky 3.1).
- Sjednotit design nápovědy u akčních tlačítek.
- Zvýraznění akčních tlačítek.
- Nastavení vybraného roku u výběru dat při vytváření ročníků a vylepšení pozic výběrů dat.

V tabulce 3.1 se nachází sloupec s návrhem řešení pro možnou inspiraci.

V poslední řadě zpracovat všechny **Issues** v novém GitLab repozitáři (zejména vyladění rozpoznávání zkratk při exportu publikace) a některé z **Issues** z původní aplikace, které se budou zdát stále relevantní.

Závěr

Cílem diplomové práce bylo přepsat původní databázi publikací a konferencí do jazyka TypeScript a migraci databáze do PostgreSQL. Práce zahrnuje analýzu původní aplikace, tedy k čemu slouží a co všechno dokáže, výběr správných technologií, návrh architektury a uživatelského rozhraní a následné vypracování programu. Program byl otestován jak funkčně, tak uživatelsky, přičemž pro testování uživatelské přívětivosti byla zvolena metoda testování použitelnosti. Některé výsledné připomínky byly zapracovány a ze zbylých byla navržena témata pro další vylepšení aplikace.

Analýza původní aplikace včetně jejího popisu, návrhu nové aplikace a výběru technologií byla rozebrána v kapitole 1.

Kapitola 2 se zabývala implementací nové aplikace. Nejprve byla popsána migrace databáze, přičemž zbývající část kapitoly obsahuje dokumentaci všech částí vyhrazených pro tuto práci, což zahrnuje implementační detaily a významné části kódu. Tato kapitola slouží jako programátorská příručka pro tuto část aplikace a navíc příloha G obsahuje uživatelskou příručku celého softwaru.

Poslední kapitola 3 obsahuje scénáře a průběh testování aplikace. V této kapitole v sekci 3.4 byla také navržena témata pro budoucí vylepšení aplikace. V rámci funkčního testování byly také vyvinuty back-endové integrační testy, které se spouštějí s každým sestavením aplikace.

Cíl práce byl splněn. Výsledná aplikace je plně použitelný software pro správu publikací a konferencí, který se v praxi využije na Fakultě informačních technologií ČVUT. Oproti původní aplikaci má modernější a uživatelsky přívětivější rozhraní, vylepšenou zpětnou vazbu a obsahuje všechny základní funkce, které jsou plně funkční.

Závěrečný dotazník 1. uživatele

1. Líbila se Vám aplikace PubConf jako celek? Dokážete si představit tuto aplikaci používat pro každodenní správu publikací a konferencí?

Áno, páčil sa mi dizajn, rozloženie stránky aj jednotlivé funkcionality. Z toho čo som mal možnosť vidieť a použiť by som si vedel predstaviť používať aplikáciu. Pre pravidelne používanie by sa možno hodila možnosť konferencie/publikácie alebo roky konferencie kopírovať - napr tak urýchliť pridávanie rokov konferencii ak sa niektoré polia opakujú.

2. Co se Vám na aplikaci pro změnu nelíbilo, případně co Vám přišlo zbytečné, krkolonné či obtížné?

Pri zadávaní hesla pri registrovaní nového používateľa by sa mali podmienky pre heslo vypísať hneď a naraz, nie postupne. To je potom zbytočne frustrujúce. Taktiež by sa asi novému userovi malo heslo vygenerovať automaticky a poslať na email a nemal by ho vytvárať administrátor. Pri merge konferencií by bolo vhodné mať možnosť prehodiť destination/source v prípade, že vie aké dve konferencie chce mergnúť len na to šiel zlou stranou.

- 3.** Přišel Vám způsob vyhledávání, filtrace a zobrazování konferencí a ročníků konferencí srozumitelný?

Áno s tým som nemal problém.

4. Je způsob přepínání mezi konferencemi a ročníky konferencí jasný?

Áno.

5. Je pro Vás koncepce vytváření publikací a konferencí srozumitelná? Tedy rozdělení formuláře na sekce, vytváření ročníků konferencí v podobě se-

znamu, vytváření nových souvisejících záznamů z formuláře (např. vytvoření nového autora při vytváření publikace) atd.

Áno, s tým som nemal žiadny problém.

6. Je kopírovací tlačítko vedle citačních formátů užitečný doplněk?

Áno, tam mi skôr nesesedelo, že sa tá sekcia volala export ale možno som to len nepochopil. Pod exportom si predstavím nejaké uloženie niečoho v pdf alebo inom formáte.

7. Byly obrázky u tlačítek a vysvětlivky akcí v aplikaci dostatečně srozumitelné?

Nestihol som sa na to počas testovania veľmi zamerať ale keďže tlačidlá mali tooltips tak sa to dalo pochopiť.

8. Jiné připomínky:

Myslím, že všetko čo mi napadlo som spísal vyššie. Takže nič:)

Závěrečný dotazník 2. uživatelé

1. Líbila se Vám aplikace PubConf jako celek? Dokážete si představit tuto aplikaci používat pro každodenní správu publikací a konferencí?

V podstate ano, po miernych úpravách UI si ju dokážem predstaviť používať pre každodennú správu publikácií a konferencií.

2. Co se Vám na aplikaci pro změnu nelíbilo, případně co Vám přišlo zbytečné, krkolonné či obtížné?

Ono to asi nebolo vašou úlohou, ale to UI aplikácie mi prišlo pomerne zmätené. Taktiež sa mi nepáčil ten dlho trvajúci delete, ktorý nebol okamžitý, ale trval 3s?? Ďalej pridanie ISBN a ISSN bolo pre mňa confusing, kedy ma zmiatol ten switch button medzi ISBN a ISSN (viz. Testovanie), to + podtým som si žiaľ nevyšmol. Potom to HESLO, prosím pridajte všetky podmienky validácie hesla naraz + pridajte ďalší input type password pre overenie hesla. Vyhľadávanie konerencií / publikácií by mohlo byť lepšie, hlavne nech sa to filtruje rýchlejšie - boli tam časté záseky pri hľadaní (veľa záznamov). Taktiež editačná vzdialenosť strikes again, ale to je už mimo vaše DP. Nakoniec všetko to čo som vám hovoril počas testovania.

3. Přišel Vám způsob vyhledávání, filtrace a zobrazování konferencí a ročníků konferencí srozumitelný?

Vyhľadávanie ok az na to pomalé hľadanie. Filtrovanie conference year som úprimne nepochopil, skúšal som to aj teraz a neviem ako filtrovať napr. rok 2016. Keď to dám do toho poľa na hľadanie, tak tam podľa roku mi to nejde a automaticky sa mi zvolí nejaká konferencia, ale ja chcem len rok (s tým som mal problém aj pri testovaní myslím). Inak asi OK

4. Je způsob přepínání mezi konferencemi a ročníky konferencí jasný?

No, najprv mi to jasné nebolo, ale je to tam celkom do oči bijúce, takže asi by to malo byť jasné.

5. Je pro Vás koncepce vytváření publikací a konferencí srozumitelná? Tedy rozdělení formuláře na sekce, vytváření ročníků konferencí v podobě seznamu, vytváření nových souvisejících záznamů z formuláře (např. vytvoření nového autora při vytváření publikace) atd.

Asi ano, počas testovania som tam mal taktiež pár zaváhání, ale overall asi ok

6. Je kopírovací tlačítko vedle citačních formátů užitečný doplněk?

To som si nikdy nevšimol (ani pri testovaní), takže neviem posúdiť.

7. Byly obrázky u tlačítek a vysvětlivky akcí v aplikaci dostatečně srozumitelné?

Asi ano, ale všimol som si u zoznamu publikácií, ako sú tam tie 4 tlačítka napravo, tak to na kopírovanie má nejaký vlastný tooltip, ale ostatné majú len ten default HTML alt.

8. Jiné připomínky:

Všetky ktoré som vám poveda počas testovania, nech sa to použije aspoň na niečo, keď je to s mojimi nadávkami nepoužiteľné.

Závěrečný dotazník 3. uživatele

- Líbila se mi hlavní tabulka s konferencemi, přišla mi přehledná a bylo fajn, že se dá na všechno kliknout a je to interaktivní. Celkově že je to propojené a dá se dostat na všechno různými způsoby.*

Líbila se mi možnost vyhledávat v konferencích fulltextově, přičemž po zmáčknutí enteru se vyhledaný výraz zobrazil jako buňka ve vyhledávacím poli a věděl jsem, co jsem vyhledal, zároveň oceňuji přítomnost našeptávače.

- Schovaný journal, dokud jsem nevybral article, mohl by být zašedlý (disabled). Check u mazání mi přijde navíc. Tlačítka na detailu ročníku mi přijdou mrtvé. Nelíbil se mi jak fungoval `DatePicker` u vytváření konference.*

- Asi ano.

- Trošku divné, jak se jiné ročníky stejné konference jmenujou jinak. Jinak dobré.*

- Je pro Vás koncepce vytváření publikací a konferencí srozumitelná? Tedy rozdělení formuláře na sekce, vytváření ročníků konferencí v podobě seznamu, vytváření nových souvisejících záznamů z formuláře (např. vytvoření nového autora při vytváření publikace) atd.

Bylo by super kdyby jednotlivé ročníky od sebe něco přebíraly, když se vytváří.

6. Je kopírovací tlačítko vedle citačních formátů užitečný doplněk?

Určitě ano, ale nejsem si jistý jeho umístěním a velikostí.

7. Byly obrázky u tlačítek a vysvětlivky akcí v aplikaci dostatečně srozumitelné?

Ano, ale některé mi nedávaly smysl, např. pšenice u autorů.

 [Bart Selman](#)

8. Jiné připomínky:

Žádné.

Závěrečný dotazník 4. uživatele

1. Líbila se Vám aplikace PubConf jako celek? Dokážete si představit tuto aplikaci používat pro každodenní správu publikací a konferencí?

Ano, nejsem v této oblasti zrovna zběhlá, ale věřím, že by to bylo lepší, kdybych tomu lépe rozuměla nebo s aplikací alespoň chvíli už pracovala.

2. Co se Vám na aplikaci pro změnu nelíbilo, případně co Vám přišlo zbytečné, krkolonné či obtížné?

Neušimla jsem si členění konferencí vlevo, ale zpětně mi tam přijdou fajn.

3. Přišel Vám způsob vyhledávání, filtrace a zobrazování konferencí a ročníků konferencí srozumitelný?

Ano.

4. Je způsob přepínání mezi konferencemi a ročníky konferencí jasný?

Už ano.

5. Je pro Vás koncepce vytváření publikací a konferencí srozumitelná? Tedy rozdělení formuláře na sekce, vytváření ročníků konferencí v podobě seznamu, vytváření nových souvisejících záznamů z formuláře (např. vytvoření nového autora při vytváření publikace) atd.

Ano, ale opět - vypadalo by to jinak, kdybych se s aplikací seznámila už dříve. Plnit úkoly, aniž bych moc věděla co kde je a ještě před kamerou a poměrně dlouhému zadání, kterému jsem ne vždy 100% rozuměla tak pro mě nebylo zrovna jednoduché, nicméně už bych věděla, jak na to.

- 6.** Je kopírovací tlačítko vedle citačných formátů užitečný doplněk?

To je super.

7. Byly obrázky u tlačítek a vysvětlivky akcí v aplikaci dostatečně srozumitelné?

Ano, pěkné.

8. Jiné připomínky:

Je super rychlé přetáhnutí souboru do daného prostoru v publikacích.

Mapa Citation.js typů na PubConf typy publikací u importu

```
1 export const refworksType = {
2   'article-journal': PubTypes_db.article,
3   book: PubTypes_db.book,
4   chapter: PubTypes_db.inbook,
5   manuscript: PubTypes_db.unpublished,
6   'paper-conference': PubTypes_db.inproceedings,
7   report: PubTypes_db.techreport,
8   thesis: PubTypes_db.phdthesis,
9   webpage: PubTypes_db.online,
10 };
11
12 export const enwType = {
13   article: PubTypes_db.unpublished,
14   'article-journal': PubTypes_db.article,
15   book: PubTypes_db.book,
16   chapter: PubTypes_db.inbook,
17   'paper-conference': PubTypes_db.inproceedings,
18   report: PubTypes_db.techreport,
19   thesis: PubTypes_db.phdthesis,
20   webpage: PubTypes_db.online,
21 };
```

■ **Výpis kódu E.1** Enumy obsahující mapování Citation.js typů na PubConf typy.

```
1 export const bibType = {
2   article: PubTypes_db.article,
3   'article-journal': PubTypes_db.article,
4   'article-magazine': PubTypes_db.article,
5   'article-newspaper': PubTypes_db.article,
6   book: PubTypes_db.book,
7   chapter: PubTypes_db.inbook,
8   classic: PubTypes_db.unpublished,
9   collection: PubTypes_db.misc,
10  document: PubTypes_db.misc,
11  event: PubTypes_db.misc,
12  manuscript: PubTypes_db.unpublished,
13  'paper-conference': PubTypes_db.inproceedings,
14  post: PubTypes_db.online,
15  'post-weblog': PubTypes_db.online,
16  report: PubTypes_db.techreport,
17  thesis: PubTypes_db.phdthesis,
18  phdthesis: PubTypes_db.phdthesis,
19  webpage: PubTypes_db.online,
20 };
```

■ **Výpis kódu E.2** Enum obsahující mapování Citation.js typů na PubConf typy.

Mapa PubConf typů na Citation.js typy publikací u export

```
1 export const reverseRefworksType = {  
2   misc: 'manuscript',  
3   book: 'book',  
4   article: 'article-journal',  
5   inproceedings: 'paper-conference',  
6   proceedings: 'paper-conference',  
7   incollection: 'chapter',  
8   inbook: 'chapter',  
9   booklet: 'book',  
10  manual: 'report',  
11  techreport: 'report',  
12  mastersthesis: 'thesis',  
13  phdthesis: 'thesis',  
14  unpublished: 'manuscript',  
15  online: 'webpage',  
16 };
```

■ **Výpis kódu F.1** Enum obsahující mapování PubConf typů na Citation.js typy.

```

1 export const reverseEnwType = {
2   misc: 'manuscript',
3   book: 'book',
4   article: 'article-journal',
5   inproceedings: 'paper-conference',
6   proceedings: 'paper-conference',
7   incollection: 'chapter',
8   inbook: 'chapter',
9   booklet: 'book',
10  manual: 'report',
11  techreport: 'report',
12  mastersthesis: 'thesis',
13  phdthesis: 'thesis',
14  unpublished: 'article',
15  online: 'webpage',
16 };
17
18 export const reverseBibType = {
19   online: 'webpage',
20   article: 'article-journal',
21   book: 'book',
22   booklet: 'book',
23   inbook: 'chapter',
24   incollection: 'chapter',
25   inproceedings: 'paper-conference',
26   manual: 'report',
27   mastersthesis: 'thesis',
28   misc: 'document',
29   phdthesis: 'thesis',
30   proceedings: 'book',
31   techreport: 'report',
32   unpublished: 'manuscript',
33 };

```

■ **Výpis kódu F.2** Enum obsahující mapování PubConf typů na Citation.js typy.

[illegible]

Uživatelská dokumentace

Příloha níže obsahuje PDF soubor, v němž se nachází uživatelská příručka aplikace PubConf v anglickém jazyce.

CZECH TECHNICAL UNIVERSITY
FACULTY OF INFORMATION
TECHNOLOGY

PubConf
User Documentation

May 3, 2025

David Kocman, Tomáš Vondra

Creating a publication

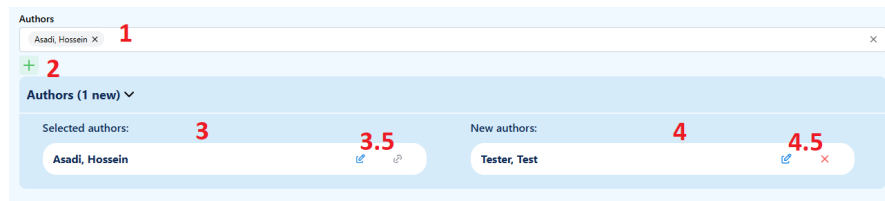
For the purpose of creating a publication a form is available, that changes its content based on the chosen type of publication. The first part of the form that contains some general and mandatory fields can be seen of the Figure 1.

The screenshot shows the 'Create publication' form in the PubConf system. The form is titled 'Create publication' and includes a sidebar menu on the left with options like Home, Publications, Conferences, and Users. The main form area has a 'General' section with fields for Type (labeled 3), Title (labeled 4), Abstract, Year (labeled 5), Month, DOI, ISSN/ISBN Identifiers (labeled 6), Note, and Authors (labeled 8). A red message at the top right says 'Fields and sections with * are required!'.

Figure 1 Form for creating a publication.

Figure 1 contains these labels:

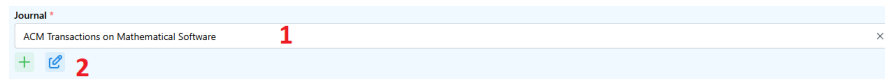
1. Breadcrumb navigation (ubiquitous).
2. Side menu.
3. Type of publication – selecting the type changes the input fields that are shown to the user. Required.
4. Title of the publication – required.
5. Year of the publication – required.
6. Add new ISSN/ISBN identifier – maximum number of identifiers is 3.
7. Select for the type of identifier and a button for deleting an identifier entry – the user defines the identifier type.
8. Authors – required for all type except for Misc and Manual, in Proceedings the author select is not shown at all.



■ **Figure 2** Selection and management of authors from the publication form.

Figure 2 contains the administration of authors from the publication form:

1. The author Select.
2. Add new author button that opens the modal for creation.
3. The list of selected authors. These entries contain action buttons for editing the author (opens the edit modal) or opening a modal with the author's related publications (3.5).
4. The list of new authors. These entries contain action buttons for editing the new author (opens the edit modal) or deleting the new author entry (4.5).



■ **Figure 3** Selection and administration of a related entity from the publication form.

The Figure 3 shows the general functionality of the administration of related entities:

1. The list of entities to select.
2. The action buttons for editing or creating a new entity. The edit button is not visible unless an entity is selected. Both buttons open a modal with a form.

Figure 4 shows the detail of a new related entity created from the publication form:

1. The action buttons – now a delete button can be seen to delete the new record.
2. Details of the newly created entity.

Figure 5 shows the behavior of the `Booktitle` and the `Publisher` fields when the *inproceedings* type is selected.

Journal

+ [edit] [delete] 1

New journal

Name: TEST JOURNAL DOI: Abbreviation:

ISSN/ISBN Identifiers 2

ISSN/ISBN Value: Note: ISSN ISBN

Figure 4 Administration and detail of a new entity from publication form.

Booktitle *

IEEE International Workshop on Testing Three-Dimensional Stacked Integrated Circuits 1

Publisher:

+

Conference

3D-Test - IEEE International Workshop on Testing Three-Dimensional Stacked Integrated Circuits 2

Year of conference

(2016) IEEE International Workshop on Testing Three-Dimensional Stacked Integrated Circuits X

Conference year 3

Name: IEEE International Workshop on Testing 1 Abbreviation: 3D-Test

Year: 2016 Publisher: 4

Figure 5 The behavior of the Booktitle and the Publisher fields when *proceedings* type is selected.

1. The Booktitle field is automatically filled with the name of the conference year.
2. Conference list and action buttons to create a new conference or edit the conference (e.g. create new years, rename).
3. The list of conference years of the selected conference and its detail.
4. When the year has an assigned publisher, the publication's publisher is overridden with this one.

Figure 6 shows the selection input of the publication categories and their administration interface:

1. Search bar.
2. The list of categories to choose from.
3. When hovering over a category, some action buttons are shown. These buttons open a modal with a form to create a subcategory, edit this category or delete the category.
4. Add a new category button that opens a modal with a form.

Figure 7 contains the two hidden parts of the publication form:

The screenshot shows a form titled 'Categories'. At the top is a search bar labeled 'Search for a category' with a magnifying glass icon. Below the search bar is a list of categories, each with a checkbox and a dropdown arrow. The categories listed are: Approximate computing, Asynchronous logic, Benchmarking, Boolean network, Cellular automata (highlighted), CGP, Codes, Complexity theory, CPU, CSP, Data structures, Delay estimation, Dependability, reliability, and Design diversity. To the right of the 'Cellular automata' entry are three icons: a plus sign, a pencil, and a trash can. At the bottom of the list is a button labeled '+ Add a new category'.

Figure 6 Selection and administration of publication categories. Categories are the only entity for which any changes are committed directly to the database.

The screenshot shows two sections of a form. The first section is titled 'Misc's optional arguments' and has a right-pointing arrow. The second section is titled 'Further custom attributes' and also has a right-pointing arrow. At the bottom of the form are two buttons: a red 'Cancel' button and a green 'Submit' button.

Figure 7 Second and third parts of the form that are hidden by default.

1. Optional arguments section – this section changes its contents based on the chosen type. It contains fields that are not mandatory for the publication entry and a file input.
2. Further attributes – this part of the form contains a list of further attributes that are used for an extension of already existing attributes.
3. Submit button – this button validates the form inputs and then creates a publication and uploads any new files.

Figure 8 illustrates the view on the optional arguments of the publication and file input:

1. The optional arguments.

Book's optional arguments ▾

Volume Number

Edition

URL

Note

Drag files here or click to select files
Attach as many files as you like, each file should not exceed 5mb

Existing files: 0

Uploaded files: 1

Conclusion.pdf (21.42 kb)

Figure 8 The view of the second part of the publication form.

2. The file input.
3. List of already existing files – when a publication is edited, any files associated with this publication that are already uploaded are shown here.
4. List of new files to be uploaded.

Further custom attributes ▾

Further attributes + 1

_parts (number of parts)	<input type="text"/>	X	pencil
_test (toto je test)	<input type="text"/>	X	pencil

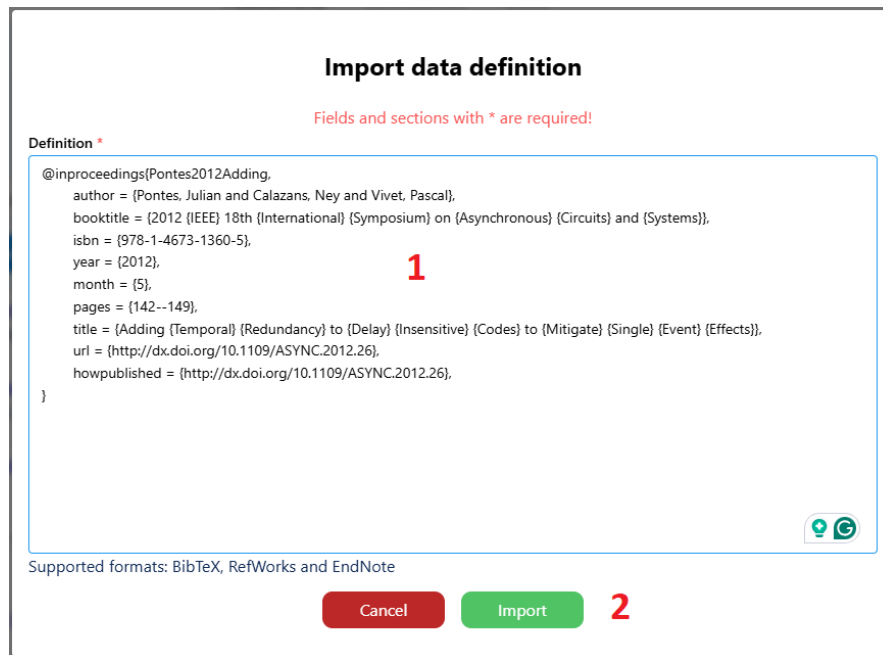
Figure 9 Further attributes and their administration.

Figure 9 illustrates the view on the third part of the publication form – further custom attributes:

1. Add a new attribute button that opens a modal with a form.
2. The input for the value of this attribute for this publication.
3. Delete or edit an attribute – again, opens a modal with a form.

A publication record can also be added to the database with the help of citation definition or imported from Springer.

Figure 10 shows the import by a citation definition modal:



Import data definition

Fields and sections with * are required!

Definition *

```
@inproceedings{Pontes2012Adding,
  author = {Pontes, Julian and Calazans, Ney and Vivet, Pascal},
  booktitle = {2012 {IEEE} 18th {International} {Symposium} on {Asynchronous} {Circuits} and {Systems}},
  isbn = {978-1-4673-1360-5},
  year = {2012},
  month = {5},
  pages = {142--149},
  title = {Adding {Temporal} {Redundancy} to {Delay} {Insensitive} {Codes} to {Mitigate} {Single} {Event} {Effects}},
  url = {http://dx.doi.org/10.1109/ASYNC.2012.26},
  howpublished = {http://dx.doi.org/10.1109/ASYNC.2012.26},
}
```

Supported formats: BibTeX, RefWorks and EndNote

Cancel
Import

Figure 10 Import publication by a citation definition.

1. Textarea for the raw definition to be pasted in.
2. After pasting the definition the "Import" button checks for any errors in the string and fills the publication form with the provided attributes.

After importing, the raw definition or Springer response can be seen on the top of the publication form. When the definition or response contains any warnings, the warning messages are rendered below the raw definition. Figure 11 shows this behavior:

1. The raw definition/returned data. Here are also any warnings rendered.
2. The pre-populated fields with the corresponding values from the input attributes.

The modal for importing the publication from Springer can be seen on Figure 12:

1. When the user has no API key set then this error message will be displayed.
2. The DOI, ISSN or ISBN identifier of the article.
3. In this textarea the returned data will be displayed for the user to review.
4. The "Try" button sends the request to the Springer API and fills the textarea with a response.

Home / Publications / Create

Create publication

Imported this definition:

```
@inproceedings{Pontec2012Adding,
  author = {Pontec, Julian and Calazans, Ney and Vivet, Pascal},
  booktitle = {2012 (IEEE) 18th (International) Symposium on (Asynchronous) (Circuits) and (Systems)},
  isbn = {978-1-4673-1360-5},
  year = {2012},
  month = {5},
  pages = {142--149},
  title = {Adding (Temporal) (Redundancy) to (Delay) (Insensitive) (Codes) to (Mitigate) (Single) (Event) (Effects)},
  url = {http://dx.doi.org/10.1109/ASYNC.2012.26}.}
```

Fields and sections with * are required!

General

Type *
InProceedings

Title *
Adding Temporal Redundancy to Delay Insensitive Codes to Mitigate Single Event Effects

Abstract

Year *
2012

Month
5

DOI

■ **Figure 11** The raw definition can be seen at the top of the publication form.

Import data from Springer API

You have not set up your springer API key in the settings! [Here is how to get one.](#)

Identifier *
doi:10.1145/321229.321232

Supported formats: DOI, ISSN and ISBN.
Always start the identifier with a prefix identifying its format. Example: doi:10.1007/s11276-008-0131-4

Returned data

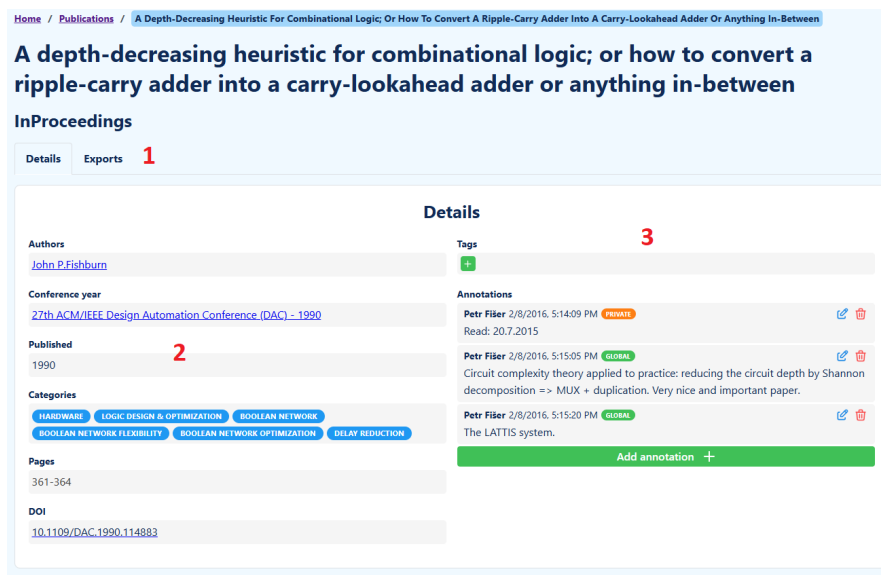
Cancel Try Import

■ **Figure 12** Import publication from the Springer API.

5. When the user fetched the correct data, the "Import" button navigates the

user to the filled-out form.

Detail of the publication



■ **Figure 13** The detail of a publication, part 1.

When clicking on a record in the publication list, the user is navigated to a publication detail page. Figure 15 describes the first part of the page:

1. The detail tabs – detail or exports page.
2. The information about the publication.
3. Annotations or tags administration.

Figure 15 contains the second part of the details page:

1. The files of the publication.
2. Edit or delete buttons.
3. Metadata – created by and modified by.

Exporting a publication means creating a citation format from publication information for the purpose of citing it. Figure 15 describes the export tab:

The screenshot shows a user interface for managing a publication. At the top, there is a 'Files' section with a link to '54_00114883.pdf' (labeled 1). Below this is an 'Edit this publication.' button (labeled 2). Underneath is a 'Delete this publication.' button with a trash icon and a 'Delete' button (labeled 3). At the bottom, there is a table with two columns: 'Added by:' and 'Last edited by:'. The 'Added by:' row shows 'Petr Filer'. The 'Last edited by:' row shows a dash '-' (labeled 3).

Added by:	Last edited by:
Petr Filer	-

■ **Figure 14** The detail of a publication, part 2.

The screenshot shows the 'Exports' tab of a publication page. The title is 'A comprehensive structural-based similarity measure in directed graphs'. Below the title, there is a 'Bibtex' section. It contains a BibTeX citation (labeled 1) and a copy button (labeled 2).

```
@article{Zhang2015comprehensive,
  author = {Zhang, Mingxi and Hu, Hao and He, Zhenying and Gao, Liping and Sun, Liujie},
  journal = {Neurocomputing},
  doi = {https://doi.org/10.1016/j.neucom.2015.04.084},
  year = {2015},
  pages = {147--157},
  title = {A comprehensive structural-based similarity measure in directed graphs},
  url = {http://www.sciencedirect.com/science/article/pii/S0925231215006359},
  howpublished = {http://www.sciencedirect.com/science/article/pii/S0925231215006359},
  volume = {167},
}
```

■ **Figure 15** The publication exports page.

1. The final citation.
2. A button for copying the citation.

Conferences

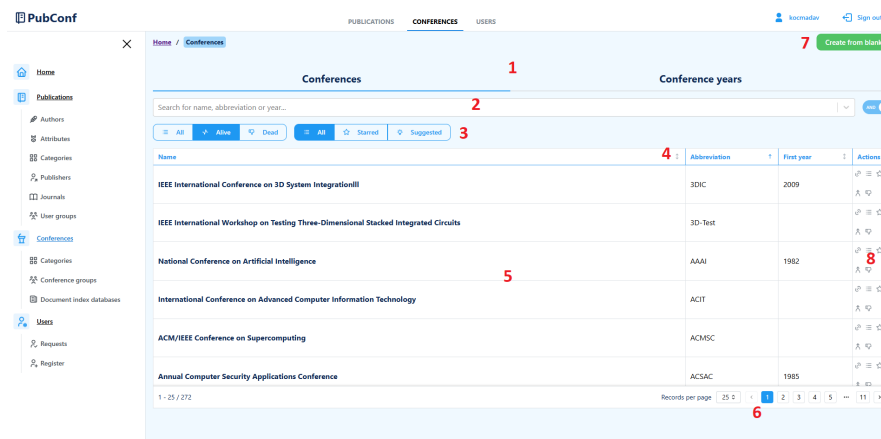


Figure 16 List of conferences and their filters.

Figure 16 contains the list of conferences and their filters to view, search and filter conferences:

1. The conference and conference years tab – clicking the tab changes the view.
2. Search bar - search by title, abbreviation or year.
3. Filter buttons – see dead or alive or all conferences or see favourites or suggested conferences.
4. Headers with order by switches.
5. The data.
6. Pagination options.
7. Button that navigates to the create conference form.
8. The action buttons – show associated publications or this conference's years or set this year as alive/dead or this conference as favourite or merge this conference with another one.

Figure 17 contains the list of conference years and their filters:

PUBICATIONS **CONFERENCES** USERS

Home / **Conferences**

Conferences

Conference years

Search for name, abbreviation or year...

1

Year name	Abbreviation	Year	Categories	Deadline	Notification	Final version	Date	Actions
IEEE International Conference on 3D System Integration	3DIC	2009						+ - ☆
IEEE International Conference on 3D System Integration	3DIC	2018		29. 6. 2018			9. 10. 2018 - 11. 10. 2018	+ - ☆
IEEE International Conference on 3D System Integration	3DIC	2020						+ - ☆
IEEE International Conference on 3D System Integration	3DIC	2016		30. 6. 2016			8. 11. 2016 - 11. 11. 2016	+ - ☆
IEEE International Conference on 3D System Integration	3DIC	2015		14. 5. 2015			31. 8. 2015 - 2. 9. 2015	+ - ☆
IEEE International Workshop on Testing Three-Dimensional Stacked Integrated Circuits	3D-Test	2016		1. 10. 2016	15. 10. 2016	1. 11. 2016	17. 11. 2016 - 18. 11. 2016	+ - ☆

1 / 25 / 2066

Records per page: 25 | 50 | 100 | 200 | 500 | 1000 | All

■ **Figure 17** List of conference years and their filters.

1. Conference year filters – filter by archived, alive or last years of archived or see favourite (this filters favourite conferences, not years) or suggested years or filter by categories.
2. The data.
3. The action buttons – show associated publications or workshops or set this year as alive/archived or this conference as favourite.

Details

[Home](#) / [Conferences](#) / [IEEE International Conference On 3D System IntegrationIII](#)

IEEE International Conference on 3D System IntegrationIII

[Detail](#) | [Related Publications](#)

Detail of the conference ↗

Mark as dead
 Merge
 Mark as starred

Abbreviation

3DIC

Web

-

Previous conference names

Name	Abbreviation	Changed
IEEE International Conference on 3D System Integration	3DIC	10. 4. 2025 18:16:32

[Edit this conference and all its years.](#)

Delete the whole conference and all its years.

Conference added by: Petr Fíler

Conference last edited by: David Kooman (13. 4. 2025 15:18:54)

Conference years (5):

- 2009: IEEE International Conference on 3D System Integration (3DIC)
- 2015: IEEE International Conference on 3D System Integration (3DIC)
- 2016: IEEE International Conference on 3D System Integration (3DIC)
- 2018: IEEE International Conference on 3D System Integration (3DIC)

■ **Figure 18** Detail of a conference.

After clicking on any of the record in the conference or conference year list, the user is navigated to a detail page. Figure 18 contains the detail page of a conference:

1. The tabs –detail or export page.
2. Action buttons – set as alive/dead or merge with another category worse as favourite.
3. Information about the conference.
4. Previous names.
5. Edit or delete buttons.
6. Metadata – created by and modified by.
7. List of conference years of this conference.

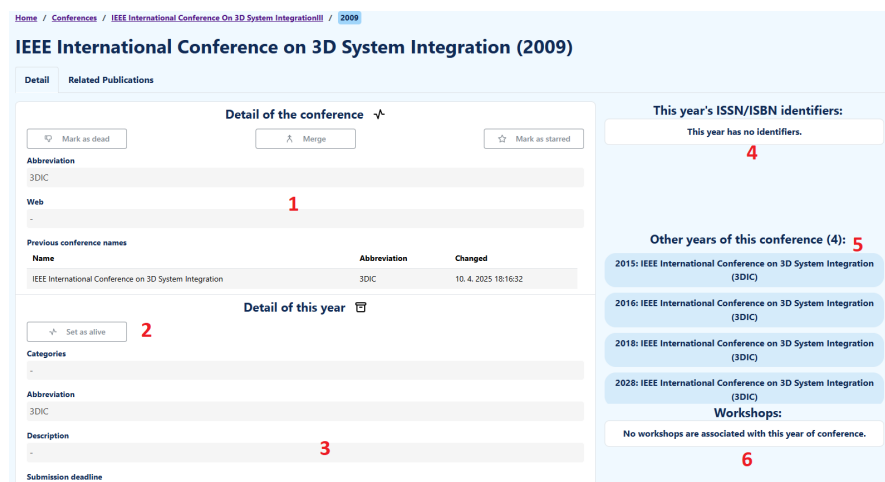


Figure 19 Detail of a conference year, part 1.

Figure 19 contains the detail page of a conference year, part one:

1. The conference detail
2. Action button – set as alive/archived.
3. Information about the year.
4. List of ISSN/ISBN identifiers of this year.
5. List of other conference years.
6. List of workshops.

Figure 19 contains the detail page of a conference year, part two:

1. Edit or delete buttons.
2. Metadata of the conference and the year of conference – created by and modified by.

Submission deadline

-

Notification

-

Final version

-

Date

-

Location

-

Web

-

Indexed at

-

Publisher

-

Edit this conference and all its years.

1 Delete only this year of conference. Delete

2

Conference added by:	Petr Fišer	Year added by:	Petr Fišer
Conference last edited by:	David Kocman (13. 4. 2025 15:18:54)	Year last edited by:	David Kocman (13. 4. 2025 15:18:54)

■ **Figure 20** Detail of a conference year, part 2.

Creating a conference

Home / Conferences / Advanced Computer Systems / Edit

Advanced Computer Systems

Fields and sections with * are required!

Conference

Name * 1 Advanced Computer Systems

Abbreviation * ACS

Web

Years of conference (2) + 2

1998 : (ACS) >

(New) Year : (ACS) > 3

Cancel Submit 4

■ **Figure 21** The form for creating or editing a conference.

Figure 21 contains the form for creating or editing conferences and all its years:

1. Conference information – next to the abbreviation field is the rollback to one of the previous names button. After clicking, a dropdown with all previous names is shown, and the user can change the current name and abbreviation to one of the previous ones.

2. Conference years list and create a new year button.
3. The year dropdown – for the sake of saving space the years are hidden in a dropdown. The trash can on the right removes the entry.
4. Submit button – validates the form and creates a new conference.

■ **Figure 22** The contents of the year dropdown.

Figure 22 contains the contents of the year dropdown:

1. The dropdown header – the year and abbreviation is shown.
2. Mandatory fields – name, abbreviation, and year.
3. Same category select and administration interface as in Figure 6.

Navigation and Home Page



Home Page

The Home page contains lists of recently edited and visited publications and conferences:

1. [Recently edited and visited publications](#)
2. [Recently edited and visited conferences](#)

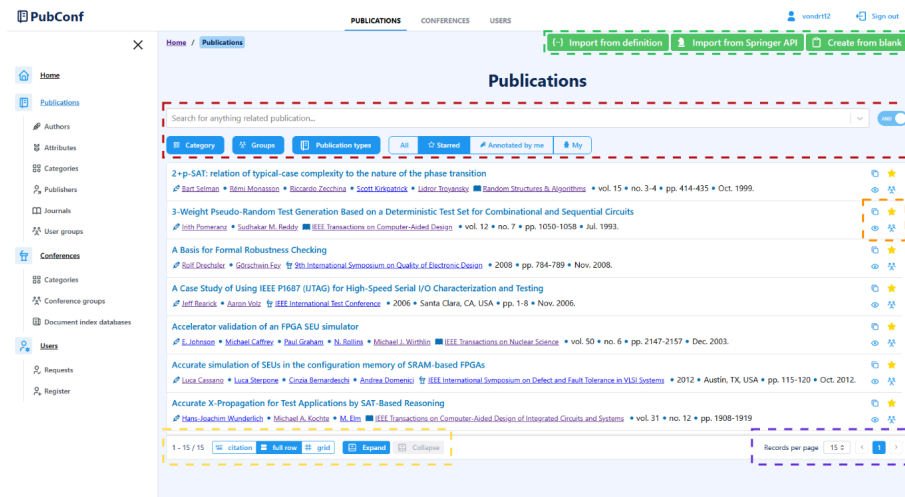
Navigation

The [navigation](#) lets the user switch between application pages.

The top navigation contains links for the main pages and user-specific actions:

1. [Home](#) – Home page
2. [Publications](#) – [Publications list](#)
3. [Conferences](#) – Conferences list
4. [Users](#) – Users list (accessible only to `admin` users)
5. [vondrt12](#) – Signed-in user's profile page
6. [Sign out](#) – Logs out the current user

Publication List



Create Publication

The top three buttons let the user create a new publication in three ways:

1. **Import from definition** – Create from a [BibTeX definition](#).
2. **Import from Springer** – Create using DOI, ISSN, or ISBN via the Springer API.
3. **Create from blank** – Create from scratch.

Filters



Users can use these options to filter the entries shown in the table below:

1. **Search input** – Enter multiple text values (author, title, category, group, year). Results must satisfy *all* specified values (logical AND).
2. **AND/OR** – Determine whether the search values are combined using logical AND or OR.
3. **Category** – Opens a popup to filter by one or more *Categories*.
4. **Groups** – Opens a popup to filter by one or more *Groups*.
5. **Publication Types** – Opens a popup to filter by one or more *Publication types*.
6. **Relation to user filters:**
 7. **All** – No filter applied.
 8. **Starred** – Show only publications *starred* by the user.
 9. **Annotated by** – Show only publications *annotated* by the user.
 10. **My** – Show only publications *created* by the user.

List Item Functions




Each result table row has four actions in the item's actions section:



1. **Copy** – Copy the publication's citation.
2. **Star** – Star the publication (add to favorites).

3.  – Show additional information (if available).
4.  – Opens a popup to select the *Groups* for the publication.

Result Table Controls

On the bottom left, the user can control the *view* of the table. Options:

1.  **Citation** – Citation view.
2.  **Full row** – Full-row view with icons and dividers.
3.  **Grid** – Grid view with individual cells.

The user can also  **Expand** to show additional information for all results or  **Collapse** to hide it.

On the bottom right, the user can control table *pagination*.

Entity Picker

Entity administration always starts with a list view that lets the user search for an entity and navigate to its detail page.

Publication Authors

Search for author



Name

Petr Fišer

Hana Kubátová

Pavel Kubalík

Vishwani D. Agrawal

A. V.S.S. Prasad

V. Madhusudan

Sooryong Lee

Brad Cobb

Jennifer Dworak

1 - 15 / 1565

Records per page

15



<

1

2

3

4

5

...

105



>

or

Choose from most popular authors

Petr Fišer

86 publications →

Alan Mishchenko

45 publications →

Robert K. Brayton

43 publications →

Jan Schmidt

39 publications →

Rolf Drechsler

33 publications →

Hana Kubátová

20 publications →

Giovanni De Micheli

20 publications →

Pierre-Emmanuel Gaillardon

15 publications →

Luca Gaetano Amarú

15 publications →

Jan Hlavička

14 publications →

Mathias Soeken

13 publications →

Stephan Eggersglüß

13 publications →

Janusz Rajski

12 publications →

Franc Brglez

11 publications →

Satrajit Chatterjee

11 publications →



Add author



Name *

Surname *

Middle name

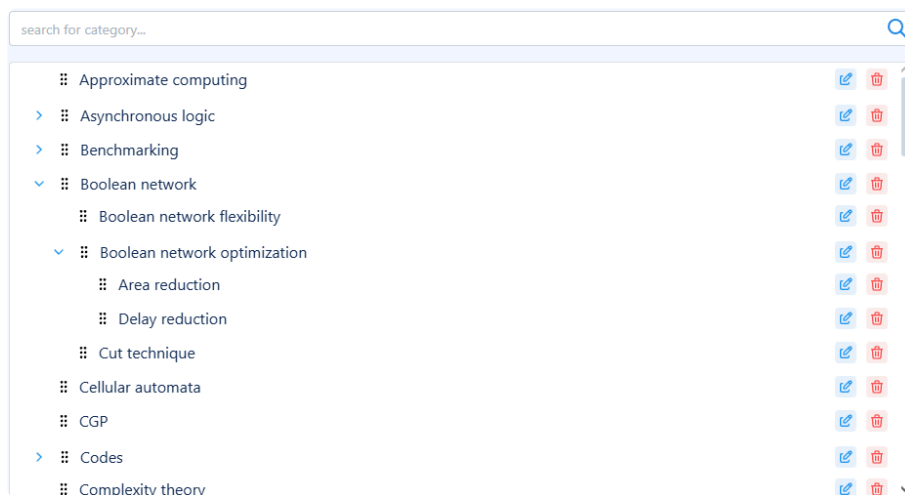
Associated user



The list view contains:

1. Entities picker – Search by name; click a row to view details.
2. Entities popular list – Shows entities with the most publication connections.
3. Add entity form – Form to add a new entity.

Category Picker



The category picker offers extra functionality:

1. **:: Reordering** – Drag & drop to reorder categories; moving a parent moves its children.
2. **✎ Inline editing** – Edit a category name directly in the picker.
3. **🗑 Inline deleting** – Delete a category (and its children) directly in the picker.

Entity Detail Page

Home / Publications / Authors / Petr Fišer

Petr Fišer

Associated user

Petr Fišer

Publications with this author

Search for anything related publication...

A Comprehensive Set of Logic Synthesis and Optimization Examples

Petr Fišer

Jan Schmidt

International Workshop on Boolean Problems

2016

Freiberg, Germany

pp. 151-158

Sep. 2016.

A Difficult Example Or a Badly Represented One?

Petr Fišer

Jan Schmidt

International Workshop on Boolean Problems

2012

Freiberg, Germany

pp. 115-122

Sep. 2012.

A Fast SOP Minimizer for Logic Functions Described by Many Product Terms

Petr Fišer

David Toman

12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools

2009

Patras, Greece

pp. 757-764

Aug. 2009.

A Flexible Minimization and Partitioning Method

Petr Fišer

Jan Hlavicka

5th International Workshop on Boolean Problems

2002

Freiberg, Germany

pp. 83-90

Sep. 2002.

A Heuristic method of two-level logic synthesis

Petr Fišer

Jan Hlavicka

The 5th World Multiconference on Systemics, Cybernetics and Informatics

2001

Orlando, Florida, USA

pp. 283-288

Jul. 2001.

Algorithm for Minimization of Partial Boolean Functions

Petr Fišer

Jan Hlavicka

IEEE Design and Diagnostics of Electronic Circuits and Systems

1 - 15 / 86

citation

full row

grid

Expand

Collapses

Records per page 15

1

2

3

4

5

6

Edit author

Delete Author

Search for author

Name

Petr Fišer

Hana Kubátová

Pavel Kubalik

Vishwani D. Agrawal

A. V.S.S. Prasad

V. Madhusudan

Sooryong Lee

Brad Cobb

Jennifer Dworak

Michael R. Grimaldi

M. Ray Mercer

Malgorzata Marek-Sadowska

M. K. Iyer

G. Parthasarathy

L. C. Wang

1 - 15 / 1565

Records per page 15

1

2

3

4

5

The side picker provides the same actions as on the [entity picker](#) page. Additionally:































- [Detail information](#) – Displays additional data about the entity.
- [Publication list](#) – Shows publications linked to the entity.
- [Entity actions](#) – Edit or delete the entity; for categories, add a subcategory.

Users List

Home / Users

+ Register user

Search users...

Name	Email	Level	Actions
Jakub Kozubek (kozubjak)	kozubjak@fit.cvut.cz	submitter ✕	 
Hana Kubátová (kubatova)	kubatova@fit.cvut.cz	submitter ✕	 
Jiří Kvasnička (kvasnjir)	kvasnjir@fit.cvut.cz	submitter ✕	 
Latka (latkamat)	latkamat@fit.cvut.cz		 
Lepic (lepict1)	lepict1@fit.cvut.cz		 
Magda Friedjungova (friedmag)	magda.friedjungova@fit.cvut.cz		 
Marek Langos (marek.langos)	marek.langos@email.com	submitter ✕	 
Adam Marheřka (marheada)	marheada@fit.cvut.cz	submitter ✕	 
Martin Bílý (bily)	martin.bily@fit.cvut.cz	submitter ✕	 
Martin Danhel (danhema12)	martin.danhel@fit.cvut.cz	submitter ✕	 
Martin Kohlík (kohlimar2)	martin.kohlik@fit.cvut.cz	submitter ✕	 
Martin Novotný (novotnym2)	martin.novotny@fit.cvut.cz	submitter ✕	 
Matej Bartík (bartimat2)	matej.bartik@fit.cvut.cz	submitter ✕	 
Valenta (valenta)	Michal.Valenta@cvut.cz		 
Miloslav Bečvář (becvarm)	milos.becvar@centrum.cz	submitter ✕	 

61 - 75 / 129

Records per page 15 1 4 5 6 9

On this page, an administrator can search registered users. Each row includes actions to remove roles, edit, or delete the account.

The **+ Register user** button navigates to the registration page for adding external users. External users sign in with email and password and require administrator registration. FIT ČVUT students and employees sign in via **Microsoft Entra ID**.

User Requests

Home / Users / Requests

Search for user...

User	Request date	Requested role	Verdict	Actions
Tomáš Vondra (vondrt12)	26. 04. 2025	submitter	waiting	approve reject
Marek Langos (mareklangos)	10. 04. 2025	admin	approved	
(triksiand)	08. 04. 2025	admin	rejected	
(triksiand)	07. 04. 2025	admin	approved	
(triksiand)	07. 04. 2025	submitter	approved	
Tomáš Vondra (vondrt12)	05. 11. 2024	admin	approved	
Testovací Tester (test123)	25. 04. 2019	admin	approved	

1 - 7 / 7

Records per page 15 1

Administrators can [approve](#) or [reject](#) role requests. The table holds up to *100* requests; when full, the oldest handled request is deleted.

User Profile

vondrt12

Profile

User Detail

Name

Tomáš Vondra

E-mail

vondrt12@fit.cvut.cz

Login

vondrt12

Permissions

admin

submitter denied

Account settings

Name

Tomáš

Surname

Vondra

Import settings

Springer token (optional)

Spring token

Publications table settings

Table default view

Citation

Full row

Grid

Default Expansion state

Collapsed

Expanded


Pagination

15

Save

Permissions

In this section, users can request the [admin](#) and [submitter](#) roles:

- [Admin](#) – manage other users' permissions and requests.
 - [Submitter](#) – edit publications, conferences, and related entities.
- If a request is denied, the user can reapply with the  **Reapply** button.

User Preferences

- **Account settings** – Edit first and last name.
- **Import settings** – Enter a personal Springer token for importing publications via the [Import from Springer](#) option.
- **Publication table settings:**
 1. **Table default view** – Choose the default [table view](#).
 2. **Default expansion state** – Set whether additional information is visible by default.
 3. **Pagination** – Specify how many items each table displays at once.

Bibliografie

1. VONDRA, Tomáš. *Reimplementace databáze konferencí a publikací I.* 2025. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
2. BIBTEX.EU. *BibTeX Entry Types*. 2025. Dostupné také z: <https://bibtex.eu/types/>. cit. 2025-04-06.
3. CITAVI. *Importing an EndNote Tagged File*. 2025. Dostupné také z: https://www1.citavi.com/sub/manual5/en/importing_an_endnote_tagged_file.html. cit. 2025-04-06.
4. REFWORKS. *RefWorks Tagged Format*. 2025. Dostupné také z: https://refworks.com/RWShibboleth/help/508help/RefWorks_Tagged_Format.htm. cit. 2025-04-06.
5. NATURE, Springer. *Springer Nature API Documentation*. [B.r.]. Dostupné také z: <https://dev.springernature.com/docs/introduction/>. cit. 2025-04-10.
6. VERCEL, Inc. *Next.js Documentation* [<https://nextjs.org/docs>]. 2025. cit. 2025-04-06.
7. RTISHCHEV, Vitaly et al. *Mantine*. 2021. Dostupné také z: <https://mantine.dev/overview/>. cit. 2025-04-06.
8. FERREL, Ray. *SQLite vs PostgreSQL: Choosing the Right DB*. 2023. Dostupné také z: <https://sqldocs.org/sqlite-database/sqlite-vs-postgresql/>. cit. 2025-04-20.
9. PRISMA DATA, Inc. *Prisma ORM Documentation*. 2025. Dostupné také z: <https://www.prisma.io/docs/orm/overview/introduction/what-is-prisma>. cit. 2025-04-09.
10. MICROSOFT. *Modules*. 2015. Dostupné také z: <https://www.typescriptlang.org/docs/handbook/2/modules.html>. cit. 2025-04-11.

11. REFACTORING GURU. *Strategy Design Pattern*. 2025. Dostupné také z: <https://refactoring.guru/design-patterns/strategy>. cit. 2025-04-11.
12. REFACTORING GURU. *Builder Design Pattern*. 2025. Dostupné také z: <https://refactoring.guru/design-patterns/builder>. Accessed: 2025-04-11.
13. FONTAINE, Dimitri. *Pgloader Documentation*. 2022. Dostupné také z: <https://pgloader.readthedocs.io/en/latest/intro.html>. cit. 2025-04-10.
14. HAJTOL, Peter. *Databáze konferencí a publikací IV*. 2022. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
15. FONTAINE, Dimitri. *MySQL to Postgres*. 2022. Dostupné také z: <https://pgloader.readthedocs.io/en/latest/ref/mysql.html>. cit. 2025-04-11.
16. PHILIPP LEHMAN Philip Kime, Moritz Wemheuer. *The biblatex Package*. 2024. Dostupné také z: <https://mirrors.nic.cz/tex-archive/macros/latex/contrib/biblatex/doc/biblatex.pdf>. cit. 2025-04-06.
17. MANTINE DEVELOPERS. *Collapse unmountOnExit/mountOnEnter support*. 2025. Dostupné také z: <https://github.com/orgs/mantinedev/discussions/2401>. GitHub Diskuse, cit. 2025-04-12.
18. WILLIGHAGEN, Lars G. Citation.js: a format-independent, modular bibliography tool for the browser and command line. *PeerJ Computer Science*. 2019, roč. 5, e214. Dostupné z DOI: 10.7717/peerj-cs.214.
19. WILLIGHAGEN, Lars G. *@citation-js/plugin-bibtex*. 2025. Dostupné také z: <https://www.npmjs.com/package/@citation-js/plugin-bibtex>. cit. 2025-04-20.
20. WILLIGHAGEN, Lars G. *@citation-js/plugin-enw*. 2024. Dostupné také z: <https://www.npmjs.com/package/@citation-js/plugin-enw>. cit. 2025-04-20.
21. WILLIGHAGEN, Lars G. *@citation-js/plugin-refworks*. 2022. Dostupné také z: <https://www.npmjs.com/package/@citation-js/plugin-refworks>. cit. 2025-04-20.
22. D'ARCUS, Bruce et al. *Schemas describing the Citation Style Language*. 2020. Dostupné také z: <https://github.com/citation-style-language/schema/tree/master>. cit. 2025-04-20.
23. MCDONNEL, Colin. *Zod*. 2020. Dostupné také z: <https://zod.dev/>. cit. 2025-04-20.

24. D'ARCUS, Bruce et al. *Citation Style Language - Style Repository*. 2020. Dostupné také z: <https://github.com/citation-style-language/styles>. cit. 2025-04-20.
25. ANTHONY, Fu et al. *Vitest – Getting started*. 2021. Dostupné také z: <https://vitest.dev/guide/>. cit. 2025-05-01.

Obsah příloh

/	
└─ README.txt.....	popis obsahu média s odkazy na nahrávky a aplikaci
└─ src.....	složka se zdrojovými kódy
└─ pubconf-2.0-main.....	složka obsahující zdrojový kód aplikace
└─ thesis	složka se zdrojovou formou práce ve formátu L ^A T _E X
└─ script...	složka s Python skriptem pro upravení původní databáze
└─ thesis-kocmadav.pdf	text práce ve formátu PDF