THINKLINK – APLIKACE NA PROPOJOVÁNÍ POZNÁMEK

Petr Kudrnovský

Bakalářská práce
Fakulta informačních technologií
České vysoké učení technické v Praze
Katedra softwarového inženýrství
Studijní program: Informatika
Specializace: Webové inženýrství
Vedoucí: Ing. David Bernhauer, Ph.D.
16. května 2025

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	ThinkLink - aplikace na propojování poznámek
Student:	Petr Kudrnovský
Vedoucí:	Ing. David Bernhauer, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2025/2026

Pokyny pro vypracování

Správa znalostí se v poslední době přesouvá z oblasti firemní do oblasti osobního rozvoje. Na rozdíl od firemního prostředí, kde se na správu know-how dbá více, správa osobních poznámek může trpět nedostatečnou provázaností jednotlivých poznámek. Cíl této práce je navrhnout aplikaci, která umožní automatické hledání souvisejících poznámek.

1. Proveďte rešerši existujících aplikací pro správu poznámek. Zaměřte se na to jakým způsobem umožní automatickou detekci souvislostí.
2. Proveďte softwarovou analýzu a návrh aplikace pro zobrazování poznámek a jejich vzájemných souvislostí, zaměřte se na rozšiřitelnost v oblasti automatického hledání souvisejících poznámek.
3. Implementujte prototyp takové aplikace na základě provedeného návrhu. Implementujte jednu naivní metodu pro provázání poznámek a alespoň jednu pokročilejší techniku.
4. Proveďte uživatelské testování prototypu a vyhodnoťte úspěšnost pokročilejší techniky v porovnání s naivním přístupem.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 6. ledna 2025 v Praze.

Chtěl bych poděkovat především Ing. Davidovi Bernhauerovi, Ph.D. za skvělé vedení práce, za čas, který mi věnoval a za sdílení cenných zkušeností a rad, které mě posunuly zase o krok dál. Zároveň bych chtěl poděkovat své rodině a přítelkyni Kateřině za jejich velkou podporu během psaní této práce. Dále bych chtěl poděkovat všem, kteří se mnou sdíleli své zkušenosti z oblasti poznámkových aplikací, ať už skrz dotazníkové šetření nebo mimo něj. Mé poděkování patří také těm, kteří prototyp aplikace otestovali a poskytli mi cennou zpětnou vazbu.

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.

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. Nástroje umělé inteligence jsem použil k následujícím účelům: kontrola textu z cizího jazyka, návrh lepší formulace a stylistiky psaného textu, vyhledávání správného LaTeX zápisu, inteligentní nápověda při psaní kódu v IDE (*autocomplete*) a vytváření obsahu testovacích sad poznámek.

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

V Praze dne 16. května 2025

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací webové aplikace ThinkLink, která uživateli umožňuje efektivní správu digitálních poznámek prostřednictvím automatického vyhledávání tematicky a obsahově souvisejících poznámek napříč celou kolekcí. Práce je zaměřena na výběr konkrétních přístupů k hledání relevantních poznámek, jejich rozšiřitelnost a porovnání jejich úspěšnosti pomocí uživatelského testování. Součástí práce je také rešerše existujících řešení a analýza potřeb uživatelů poznámkových aplikací. Výsledný prototyp aplikace byl vytvořen v jazyce PHP s využitím frameworku Symfony a databázového systému PostgreSQL. Pro účely testování byl prototyp aplikace nasazen do produkčního prostředí na vzdálený server, kde proběhlo uživatelské testování zaměřené jak na spolehlivost jednotlivých metod vyhledávání, tak na celkovou použitelnost systému. Výsledky testování poskytly podklad pro vyhodnocení nejvhodnější metody a další rozvoj aplikace ThinkLink.

Klíčová slova propojování poznámek, vektorová podobnost, sémantické vyhledávání, vektorové reprezentace, fulltextové vyhledávání, TF-IDF, webová aplikace

Abstract

This bachelor thesis deals with the design and implementation of ThinkLink, a web application that allows the user to efficiently manage digital notes by automatically searching for thematically and content-related notes across the entire collection. The thesis focuses on the selection of specific approaches to finding relevant notes, their extensibility and comparison of their success through user testing. The work also includes a search of existing solutions and an analysis of the needs of users of note taking applications. The resulting prototype application was developed in PHP using the Symfony framework and the PostgreSQL database system. For testing purposes, the prototype application was deployed to a production environment on a remote server, where user testing was conducted to test both the reliability of the individual search methods and the overall usability of the system. The results of the testing provided the basis for evaluating the most appropriate method and further development of the ThinkLink application.

Keywords note linking, vector similarity, semantic search, vector embeddings, full-text search, TF-IDF, web application

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Seznam zkratek

ACID	Atomicity, Consistency, Isolation and Durability
AI	Artificial Intelligence
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CODE	Capture, Organize, Distill, Express
CSRF	Cross-site Request Forgery
CSS	Cascading Style Sheets
GIN	Generalized Inverted Index
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LLM	Large Language Model
LTS	Long Term Support
MIME	Multipurpose Internet Mail Extensions
MVC	Model-View-Controller
NLP	Natural Language Processing
NDCG	Normalized Discounted Cumulative Gain
ORM	Object Relational Mapper
PDF	Portable Document Format
RDF	Resource Description Framework
SPA	Single Page Application
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

Úvod

Lidé si od nepaměti zaznamenávají informace. Počínaje malbami na stěnách, záznamy na pergamenu až po chytré mobilní aplikace. Poznámky existují ve mnoha formách a slouží k různým účelům. Od rychlých poznámek ve formě seznamu na nákup či uložení odkazu, přes zachycení nápadů a inspirací, až k tvoření komplexní znalostní báze. Jednotlivé poznámky lze organizovat pomocí tagů, kategorií, složek, projektů a dalších způsobů. Část uživatelů si poznámky vůbec neorganizuje, má je na jednom místě a spoléhá pouze na vyhledávání.

V současnosti roste zájem o tzv. „chytré poznámky“ – pokročilé systémy, které usnadňují tvorbu, propojování a efektivní využití poznámek. Tento trend dokládá například popularita knihy *Building a Second Brain* [1] od autora Tiago Forte nebo kniha *How To Take Smart Notes* [2] od autora Sönke Ahrens, jež popisuje poznámkový systém Zettelkasten. Tento trend je také podpořen rostoucí popularitou různých kurzů zaměřených na tvorbu chytrých poznámek a efektivní využívání poznámkových aplikací.

Tyto systémy a techniky se zaměřují nejen na způsoby psaní poznámek, ale i na jejich organizaci a strukturu. Klíčové je především propojování relevantních poznámek mezi sebou. Pokud se nová poznámka propojí s dalšími poznámkami, které ji doplňují nebo rozšiřují, vzniká síť vzájemně provázaných informací a roste celková informační hodnota systému.

Dnešní nástroje umožňují pokročilou správu poznámek včetně manuálního propojování poznámek. Aby mohl uživatel poznámky propojit, musí poznámky znát nebo je musí umět vyhledat podle názvu, klíčových slov či tagů. Aby to bylo možné, musí být poznámky buď pečlivě organizované nebo si uživatel musí obsah či názvy poznámek pamatovat. S rostoucím počtem poznámek se ale zvyšuje náročnost jejich správy, mohou vznikat duplicity a ubývá přehled o relevantních informacích.

Hlavní motivací této práce je uživateli co nejvíce usnadnit a urychlit vyhledávání relevantních poznámek. Pokud by uživatel při zobrazení poznámky ihned viděl seznam nejrelevantnějších poznámek, usnadnilo by mu to orien-

taci, omezilo duplicity a zefektivnilo práci s poznámkami, ať už jde o několik stručných záznamů či rozsáhlý systém chytrých poznámek.

Cíle práce

Cílem této práce je prozkoumat možnosti efektivnější práce s poznámkami prostřednictvím automatizovaného vyhledávání souvisejících poznámek. Cílem této funkcionality je usnadnit propojování relevantních informací, omezení duplicit a celkové zrychlení a zpřehlednění práce s poznámkovým systémem.

Hlavním záměrem je srovnání existujících přístupů k automatickému vyhledávání relevantních poznámek. Za tímto účelem bude navržena a implementována webová aplikace ThinkLink, která umožní správu poznámek rozšířenou o tuto funkcionalitu. Aplikace bude navržena s důrazem na rozšiřitelnost, zejména v oblasti přidávání nových metod pro vyhledávání souvisejících poznámek.

Součástí práce bude také uživatelské testování vybraných metod s cílem identifikovat tu, která dosahuje nejlepších výsledků z pohledu relevance doporučovaných poznámek.

Struktura práce

Nejprve bude provedena analýza potřeb uživatelů současných poznámkových aplikací, aby bylo jasné, jaké funkce v existujících aplikacích uživatelům vyhovují, jaké metody a nástroje pro organizaci poznámek používají. Důležitým cílem této analýzy je také zjistit, zda by uživatelé funkci automatického hledání souvisejících poznámek uvítali a jakými způsoby by ji využili.

V návaznosti na analýzu potřeb uživatelů proběhne rešerše a porovnání vybraných současných nástrojů na správu poznámek, s důrazem na jejich přístup k automatizovanému propojování obsahu. Tyto nástroje budou porovnány na základě pevně stanovených kritérií, která budou vycházet z analýzy potřeb uživatelů.

V rámci rešerše budou také popsány různé metody a přístupy pro hledání a propojování tematicky podobných textů. Na jejich základě se vyhodnotí, které přístupy jsou vhodné pro implementaci v aplikaci ThinkLink. Následovat bude výběr a podrobnější analýza vybrané poznámkové aplikace s cílem převzít ověřené a oblíbené funkce do návrhu nové aplikace ThinkLink.

Dále budou stanoveny funkční a nefunkční požadavky na aplikaci ThinkLink společně s modelem případů užití. Tyto části poslouží společně s doménovým modelem jako základ pro návrh a implementaci aplikace ThinkLink.

Při návrhu a implementaci je cílem klást důraz na rozšiřitelnost aplikace, aby bylo možné snadno přidávat další algoritmy automatického propojování poznámek a další typy souborů, se kterými bude umět aplikace ThinkLink pra-

covat. V práci budou také popsány implementované algoritmy pro propojování dokumentů včetně jejich výhod a nevýhod.

Závěrečná část této práce bude věnována testování prototypu webové aplikace ThinkLink. Uživatelské testování se zaměří na vyhodnocení celkové použitelnosti aplikace a porovnání úspěšnosti pokročilejších technik automatického propojování poznámek v porovnání s naivním přístupem.

Výsledkem práce bude funkční a otestovaný prototyp webové aplikace ThinkLink, který umožní uživatelům spravovat své poznámky, vkládat k nim soubory a při otevření poznámky automaticky nabídne relevantní poznámky na základě obsahové a tématické podobnosti. Takto pojatá automatizace přispěje k lepší provázanosti poznámek, omezí duplicity mezi poznámkami, zefektivní hledání relevantních informací, citací a zdrojů. Tímto způsobem se uživatelům ušetří čas, usnadní se opětovné použití starších poznámek a podpoří se propojení poznatků napříč různými oblastmi či projekty.

Kapitola 1
Rešerše

Tato kapitola se věnuje rešerši možností, jak zefektivnit práci s poznámkami prostřednictvím automatického propojování poznámek na základě obsahové a sémantické podobnosti. Rešerše má několik částí, které se navzájem doplňují a tvoří podklad pro následnou analýzu a návrh aplikace ThinkLink.

V první části se kapitola zabývá motivací k propojování poznámek a tím, jak je propojování důležité pro efektivní správu osobních znalostí. Tato část se také zabývá tím, jak by znalostním pracovníkům mohlo zautomatizování hledání relevantních poznámek pomoci.

Poté proběhne analýza potřeb uživatelů poznámkových aplikací prostřednictvím elektronického dotazníku. Tento průzkum zjišťuje, které funkce současných poznámkových aplikací jsou pro uživatele nejdůležitější, jak si uživatelé představují funkci automatického hledání souvisejících poznámek a za jakým účelem si poznámky vytvářejí.

Na základě získaných poznatků z dotazníku je provedena rešerše existujících řešení. Jednotlivá řešení jsou popsána se zaměřením na to, jaké nabízí možnosti propojování souvisejících poznámek a jestli společně s těmito funkcemi splňují i další kritéria, která jsou pro uživatele důležitá.

K získání kompletních podkladů pro analýzu a návrh aplikace ThinkLink a stanovení funkčních požadavků je provedena rešerše různých přístupů a metod k propojování textových dat na webu.

1.1 Důležitost propojování poznámek

V úvodní části rešerše jsou představeny teoretické základy a motivace k propojování digitálních poznámek. Správa osobních znalostí je zásadní zejména pro znalostní pracovníky a propojování poznámek je její důležitou součástí. Tato kapitola proto zdůrazňuje přínos vytváření vzájemných vazeb mezi poznámkami a celkový dopad na efektivitu celého systému správy informací.

Podle Willetta [3] v posledních dekádách vzrostla popularita a četnost využívání digitálních nástrojů pro tvorbu a správu poznámek. Tyto nástroje se oproti papírovým poznámkám vyznačují vyšší flexibilitou, snazší manipulací a širšími možnostmi dalšího využití.

Ačkoliv digitální poznámky nabízejí řadu výhod oproti papírovým poznámkám, tak s sebou přinášejí i nové problémy. Jedním z hlavních zjištění je skutečnost, že digitální formát často vede k tvorbě rozsáhlých kolekcí poznámek, které je náročné udržet organizované.

V rámci průzkumu bylo zjištěno, že poznámky vytvořené v minulosti, které se nacházejí hlouběji v kolekcích, nejsou často znovu navštěvovány a je snadné na ně zapomenout. To je podpořeno průzkumem [4], který uvádí, že průměrný americký zaměstnanec stráví 76 hodin ročně hledáním založených poznámek, věcí a souborů, což firmám působí značné finanční ztráty.

Průzkum [3] také ukázal, že velká část poznámek není po jejich vytvoření dále upravována, tedy potenciál pro jejich rozvoj a další rozpracování zůstává nevyužit. Týká se to také poznámek, které byly vytvořeny za účelem pozdějšího využití (např. referencí k webovým stránkám či myšlenkám a nápadům, které by mohly být dále rozpracovány).

V průzkumu je dále zmíněno, že v současných nástrojích chybí dostatečná podpora pro propojování poznámek napříč oddělenými složkami, projekty či tematickými okruhy. Z tohoto průzkumu také vyplynulo, že znalostní pracovníci a uživatelé nástrojů pro správu poznámek mají zájem o inteligentnější a vizuálně bohatší nástroje, které by jim pomohly:

- navrhovat a propojovat relevantní poznámky napříč celou kolekcí a vizualizovat jejich vzájemné vztahy,
- efektivně organizovat velké množství digitálních poznámek, aby nedocházelo ke ztrátě obsahu a
- mít nástroje pro lepší vybavování a reflexi starších a méně často navštěvovaných poznámek.

Toto podporuje i výzkum Schubmehla a Vesseta [5], který se věnuje problémům sdílení a propojování relevantních informací napříč systémy ve firmách. V rámci průzkumu bylo zjištěno, že typický znalostní pracovník stráví 16 % svého času pouze hledáním informací. Další čas věnuje konsolidaci a analyzování informací z různých zdrojů. Ne vždy je to produktivně strávený čas, protože podle článku znalostní pracovníci najdou potřebné informace pouze v 56 % případů. Čas ztrácejí také kombinováním dat z různých úložišť, formátováním rozmanitých dat z různých zdrojů a vytvářením nového obsahu, který nelze nalézt.

Podle autorů studie [3] lze tyto problémy řešit pomocí technik vizualizace informací. Na základě získaných dat proto navrhli různá řešení integrovatelná do aplikací pro správu poznámek. Jedním z nich je nástroj *Note Spiral*, který

umožňuje uživatelům sdružovat tematicky podobné poznámky do spirálového uspořádání. Ve spirále se pak k vybrané poznámce objeví tematicky nejpodobnější poznámky seřazené podle data poslední úpravy. Při přepnutí na jinou poznámku se spirála přeskládá tak, aby znovu zobrazovala relevantní obsah k nově zvolené poznámce. Tento přístup odráží požadavky uživatelů zjištěné v průzkumu a zjednodušuje objevování relevantních informací, které by jinak zůstaly skryty v hierarchii složek nebo by se těžko vyhledávaly textovým vyhledáváním.

Dále Schubmehl a Vesset [5] doporučují používat techniky a procesy zpracování informací (textová analýza, automatická kategorizace a štítkování informací) k získání hodnot z nestrukturovaných dat a jejich propojení s úložišti strukturovaných dat.

Další řešení nabízí Tiago Forte ve své knize *Building a Second Brain* [1], kde se věnuje systému, který nazývá „Druhý mozek“. Tento systém má za cíl vytvořit externí systém správy osobních znalostí, který uživateli umožní lépe zachytávat, organizovat, propojovat a využívat svoje poznámky.

1.1.1 Aplikace Scraps a propojování útržků informací s poznámkami

Dalším často zmiňovaným problémem digitálních poznámek je roztříštění obsahu do více nástrojů, což znesnadňuje jeho pozdější nalezení a použití [3, 6]. Jako řešení tohoto problému byla v rámci článku [6] vytvořena mobilní aplikace *Scraps*, která umožňuje zachycení multimediálního obsahu (poznámky, obrázky, záložky webů či audio) na jednom místě. Uživatel má možnost k uloženému obsahu doplnit kontextové informace pro pozdější zpracování. Sama aplikace automaticky k obsahu doplňuje další metadata (časová razítka, polohu atd.).

Podle průzkumu v tomto článku lidé v aplikaci *Scraps* nejčastěji zachytávali fotky (školní tabule, účtenky), webové záložky ve formě URL či nestrukturované rychlé poznámky (nápady, pracovní schůzky). Tím, že tyto informace byly uložené na jednom místě a opatřené přidaným kontextem, bylo mnohem jednodušší tyto informace později propojit s relevantními poznámkami či dokumenty.

Aplikace *Scraps* byla totiž propojena s editorem dokumentů, kde se zachycené zdroje včetně kontextu zobrazovaly na jednom místě. Toto propojení uživateli umožnilo během dne zachycovat různý obsah, opatřit ho kontextem a pak jej později jednoduše najít a využít v rámci tvorby poznámek či jiných dokumentů. Výsledkem pak bylo zvýšení produktivity a efektivního využití dříve zaznamenaného obsahu v nových dokumentech.

1.1.2 Zettelkasten – analogové propojování poznámek

Problematikou propojování poznámek se zabýval i německý sociolog a politolog Niklas Luhmann v druhé polovině 20. století. Ten položil základy moderních přístupů k efektivní správě znalostí. Vytvořil analogový systém *Zettelkasten* (česky *kartotéka*, vlastní překlad), kam zakládal jednotlivé poznámky. Systém byl zajímavý v tom, že každá poznámka měla svůj unikátní identifikátor určující její umístění v kartotéce. Klíčovým prvkem tohoto systému byla síť interních odkazů mezi jednotlivými poznámkami, kde každá poznámka byla součástí nějakého myšlenkového proudu, který jí poskytoval kontext. S podporou tohoto systému vzniklo kolem 50 knih a 550 vědeckých článků a Niklas Luhmann stal se jedním z nejvlivnějších sociologů dvacátého století. [2, 7]

Jeho *Zettelkasten* dokládá, jak promyšlená struktura a důsledné propojování poznámek mohou výrazně podpořit produktivitu a rozvoj znalostí – a to dokonce i v ryze analogovém prostředí. Současné digitální nástroje pro správu osobních znalostí se často inspirují právě těmito principy, což detailně popisuje například Sönke Ahrens ve své knize *How to Take Smart Notes* [2].

1.2 Analýza potřeb uživatelů

Většina lidí si vede poznámky v digitální podobě, avšak způsoby jejich organizace a správy se mohou lišit [3]. Cílem analýzy potřeb uživatelů není jen zjistit, jaké existující funkcionality poznámkových aplikací jsou pro uživatele klíčové, ale také zmapovat, jestli uživatelé vědí o pokročilejších způsobech a metodikách organizování poznámek a jestli by vůbec uvítali možnost automatického propojování poznámek na základě obsahové a sémantické podobnosti. Taková analýza je tedy klíčová, aby bylo možné provést kvalitní analýzu a návrh aplikace ThinkLink.

Analýza potřeb uživatelů byla provedena prostřednictvím elektronického dotazníku, který byl distribuován na sociálních sítích a mezi studenty vysokých škol.

1.2.1 Skladba dotazníku

Dotazník obsahuje celkem 18 otázek různých typů, které odpovídají charakteru zjišťovaných informací. Na začátku dotazníku se nachází úvod, kde je představen autor, bakalářská práce, vysvětlení účelu dotazníku a toho, jakým způsobem budou odpovědi použity a zpracovány. Dotazník je anonymní a je primárně určen lidem, kteří již s poznámkami pracují, mají zkušenosti s jejich správou a jednotlivé otázky s tím počítají. Je kladen velký důraz na variabilitu odpovědí a pokud jsou možnosti odpovědí uzavřené, tak vždy existuje možnost *Jiné*, kde může respondent specifikovat či rozšířit svoji odpověď.

V první části se nachází obecné otázky ohledně způsobů práce uživatelů s poznámkami, jakým účelem je organizují a jaké aplikace k tomu používají. Cílem této části je zmapovat, jakým způsobem je tvorba a správa poznámek rozšířená a jakými způsoby s poznámkami uživatelé pracují.

V druhé části uživatelé hodnotí různé funkce a vlastnosti poznámkových aplikací na stupnici 1–10 podle toho, zda je pro ně daná funkcionalita důležitá či nikoliv. Podle nejdůležitějších funkcionalit pak budou srovnány vybrané existující aplikace pro správu poznámek. Zároveň bude na tyto vybrané funkcionality kladen důraz při návrhu aplikace ThinkLink. Jednotlivých funkcionalit je celkem 9 včetně funkce automatického hledání souvisejících poznámek k aktuálně zobrazené poznámce. Vzhledem k tomu, že se jedná o uzavřený výčet, tak se v dotazníku nachází i otevřená a volitelná otázka, která nabízí respondentům specifikovat další funkcionality, které jsou pro ně důležité.

V poslední části se dotazník zaměřuje na stěžejní funkci automatického hledání souvisejících poznámek a nabízí respondentům možnost v otevřené otázce specifikovat, jak by tuto funkci používali či jak by očekávali, že jim pomůže.

1.2.2 Výsledky dotazníku

Elektronický dotazník vyplnilo 56 respondentů, primárně šlo o studenty vysokých škol. Všichni respondenti už mají nějakou zkušenost s tvorbou a správou poznámek. Veškeré otázky a výsledky jsou dostupné v přiloženém ZIP archivu. Z výsledků je zřejmé, že většina lidí využívá poznámky zejména k osobním a studijním účelům. Část respondentů poznámky také využívá k profesním účelům a pro ukládání zdrojů na pozdější přečtení.

Velká část respondentů pro správu poznámek využívá jednoduché nástroje jako *Apple Notes*[1] (44,6 %) či *Google Keep*[2] (17,9 %). Z pokročilejších nástrojů pro správu poznámek respondenti nejčastěji používají aplikaci *Microsoft OneNote*[3] (30,4 %), *Notion*[4] (17,9 %) a *Obsidian*[5] (14,3 %). Část uživatelů nepoužívá digitální nástroje, ale píše si ruční poznámky na papír. Vzhledem k tomu, že je práce zaměřena na tvorbu webové aplikace a implementaci funkcí, které lze využít pouze v digitální podobě, tak se dále papírovým nástrojům a technikám nebude věnovat pozornost.

Přes 60 % uživatelů svoji aplikaci používá více než 2 roky, což značí, že většina uživatelů často poznámkové aplikace nemění a výběr aplikace se kterou budou pracovat a svěří ji své poznámky, je pro ně důležitý. Že jde o důležitý nástroj, značí další výsledky, které uvádí, že přes 84 % uživatelů s poznámkami pracuje denně nebo několikrát týdně.

[1] Apple Notes: `https://www.icloud.com/notes/`
[2] Google Keep: `https://keep.google.com/`
[3] Microsoft OneNote: `https://www.onenote.com/`
[4] Notion: `https://www.notion.com/`
[5] Obsidian: `https://obsidian.md/`

58,9 % respondentů nemá žádnou zkušenost s pokročilými technikami organizování informací v poznámkách (např. *Zettelkasten* [2], *CODE* či *PARA* [1]), část uživatelů tyto techniky zná, ale nepoužívá je a pouze 10,7 % respondentů s nimi skutečně pracuje. Přes 67 % respondentů organizuje poznámky pomocí složek a projektů. Značná část (skoro 40 %) uživatelů poznámky neorganizuje vůbec a spoléhají se jen na vyhledávání. Automatické propojování a navrhování souvisejících poznámek by jim tedy mohlo značně usnadnit organizaci či vyhledávání informací v poznámkách.

V druhé části dotazníku respondenti hodnotili jednotlivé funkcionality poznámkových aplikací podle jejich důležitosti. Následuje seznam těchto funkcionalit seřazený podle důležitosti pro uživatele (od nejdůležitější):

1. možnost zobrazovat a upravovat poznámky offline (tj. bez přístupu k internetu),
2. automatická synchronizace poznámek napříč zařízeními,
3. dostupnost na různých platformách (mobilní zařízení, web, desktop),
4. možnost připojit k poznámkám multimédia (obrázky, nahrávky, PDF soubory apod.),
5. možnost využívat aplikaci zdarma,
6. ochrana a soukromí poznámek (např. šifrování, dvoufázové ověřování),
7. výkonné fulltextové vyhledávání (tj. vyhledávání v obsahu poznámek),
8. automatické zobrazování relevantních a souvisejících poznámek k aktuálně zobrazené poznámce a
9. funkcionalita obousměrného propojování poznámek (bidirektivní odkazy).

Ačkoliv se funcionalita automatického zobrazování relevantních poznámek umístila na předposlední příčce, odezva byla stále pozitivní. Pro 19,6 % respondentů by se jednalo o klíčovou funkci (hodnocení 10), pro 25 % respondentů by je jednalo o velmi důležitou funkci (hodnocení 9) a pro 19,6 % respondentů by se jednalo o užitečnou funkci (hodnocení 8). To poukazuje na to, že se také jedná o zajímavou funkcionalitu. Všechny navrhované vlastnosti jsou tedy pro uživatele důležité. Podle těchto funkcionalit budou v sekci *1.3 Rešerše existujících aplikací* srovnány vybrané existující aplikace pro správu poznámek.

Respondenti měli možnost specifikovat i další funkcionality, které se nenacházely v připraveném seznamu. Často byla zmiňována přehlednost a použitelnost aplikace, která by kombinovala jednoduché rozhraní a zároveň větší možnosti úpravy vzhledu poznámek. Pro část uživatelů je také důležitá možnost kombinovat různé možnosti zápisu poznámek (klávesnicí, perem či hlasem).

Při zaměření se na možnosti funkcionality automatického hledání souvisejících poznámek bylo na základě odpovědí respondentů identifikováno několik hlavních způsobů použití této funkce:

- Vyhledání starších a zapomenutých poznámek k danému tématu:
  - rychlé připomenutí obsahu, který by zůstal nevyužit,
  - možnost navázat na předchozí nápady a myšlenky a
  - efektivní archivace poznámek.
- Prevence duplicit a lepší organizace poznámek.
- Propojování a rozšiřování znalostí:
  - propojování znalostí z různých oborů, předmětů ve škole či osobních projektů a
  - doplnění informací z dříve vytvořených poznámek či archivovaných poznámek.
- Rychlé dohledávání citací a zdrojů:
  - automatické vyhledání místa, odkud daná informace pochází a
  - možnost kombinace různých relevantních zdrojů.
- Propojení různých druhů poznámek:
  - například propojení poznámek z pracovní schůzky s technickou dokumentací a předchozím projektem, kde se řešilo něco podobného.
- Lepší kategorizace vytvořených poznámek:
  - lepší organizace poznámek díky sjednocení roztříštěných, ale relevantních, informací k sobě.
- Inspirace a generování nápadů.
- Úspora času:
  - rychlejší nalezení potřebných informací a zredukování času pro ruční procházení existujících poznámek a
  - opětovné využití existujícího obsahu.
- Integrace s technikami rozpoznávání obrázků:
  - byla navržena i možnost propojování relevantních uložených obrázků či grafů s psaným textem.
- Integrace s AI asistentem, který by doplňoval kontext a shrnutí k jednotlivým poznámkám.

1.3 Rešerše existujících aplikací

Součástí rešerše je přehled, jaké různé aplikace a řešení pro správu poznámek již existují, jakým způsobem fungují a v čem se odlišují od ostatních. V současnosti je nabídka aplikací zaměřených na správu poznámek opravdu široká. Vybrány byly aplikace, které jsou dle odpovědí respondentů v rámci elektronického dotazníku nejčastěji využívané či takové aplikace, které jsou zajímavé svojí funkčností a jejich studium v rámci této rešerše je přínosné pro analýzu a návrh aplikace ThinkLink.

Tato práce se zaměřuje na automatické hledání souvisejících poznámek, proto je v rámci rešerše existujících aplikací důležité identifikovat, jakými způsoby tyto aplikace umí poznámky propojovat. U každé aplikace bude zjištěno, jestli umožňuje automatizaci propojování poznámek a případně bude tato funkce detailně popsána.

Na základě výsledků elektronického dotazníku byly vybrány klíčové funkce a vlastnosti poznámkových aplikací, jejichž důležitost byla ohodnocena respondenty dotazníku. Tyto aspekty budou použity jako kritéria pro srovnání vybraných existujících aplikací.

1.3.1 Obsidian

Obsidian [8] je aplikace pro tvorbu a správu poznámek. Kombinuje hierarchický a grafový přístup k jejich organizaci. Uživatel má možnost poznámky organizovat ve složkách a podsložkách, ale zároveň je může mezi sebou propojovat obousměrnými odkazy nezávisle na tom, kde jsou poznámky zrovna uložené. Tato propojení pak tvoří unikátní graf, který nabízí nový pohled na to, jak jsou mezi sebou poznámky propojené.

Obsidian ukládá jednotlivé poznámky pouze na disk zařízení, což zaručuje plnou kontrolu nad soubory a vysokou úroveň soukromí. Nevýhodou tohoto přístupu je absence synchronizace poznámek mezi jednotlivými zařízeními a automatického zálohování, což bylo v dotazníku analýzy potřeb uživatelů identifikováno jako klíčový faktor pro uživatele poznámkových aplikací. Obsidian nabízí možnost synchronizace pomocí placeného rozšíření *Sync*[6] nebo je možné využít alternativní možnosti synchronizace a zálohování mimo Obsidian. Tím může být například fyzické USB či cloudové služby jako *Google Drive*[7].

Obsidian nabízí desktopovou a mobilní aplikaci, ale nenabízí webovou verzi. Celá aplikace je zdarma, funkce synchronizace a publikování poznámek na web jsou placené. Obsidian je aktivně vyvíjená aplikace, kam vývojáři přidávají nové funkce a opravují chyby[8], má velkou komunitu uživatelů[9], kteří vytváří

[6] Záložka *Sync* v [8]: `https://obsidian.md/sync`
[7] Google Drive: `https://drive.google.com/`
[8] Záložka *Changelog* v [8]: `https://obsidian.md/changelog`
[9] Záložka *Community* v [8]: `https://obsidian.md/community`

velké množství pluginů a návodů, které pak všichni mohou bezplatně využívat. [8]

1.3.1.1 Komunitní rozšíření pro automatické propojování poznámek pro Obsidian

Obsidian jako samotný nenabízí žádné možnosti automatického propojování poznámek, ale zároveň díky otevřenému ekosystému pro komunitní rozšíření již existují komunitou vytvořené rozšíření zaměřené na tuto funkcionalitu. [8]

Propojení poznámek na základě výskytu názvu poznámky v textu jiné poznámky. Tato rozšíření propojují poznámky pomocí obousměrných odkazů a shody názvu poznámky s výskytem v textu jiné poznámky. Jedná se o rozšíření *Obsidian Note Linker* od uživatele *AlexW00* [9] a *Batch Obsidian Forward Linker* od uživatele *dxcore35* [10].

Obě rozšíření jsou dostupná na platformě *GitHub*[10] a zároveň v *Obsidian Plugin Store*[11], což je oficiální Obsidian platforma pro stahování a správu rozšíření do Obsidianu. Obě rozšíření fungují velmi podobně, rekurzivně najdou všechny názvy poznámek a pak v obsahu každé poznámky nahradí výskyt názvu jiné poznámky za přímý, obousměrný odkaz na danou poznámku. V obou případech se jedná o hromadné propojení poznámek najednou. Toto hromadné propojování nebere v potaz sémantickou relevanci jednotlivých poznámek. Ve většině případů se jedná o smysluplné propojení, ale může se stát, že takové propojení propojí dvě úplně nesouvisející poznámky jen na základě textové shody.

Obzvláště v případě velkého množství poznámek, může být tento přístup nevyhovující, protože může ztížit orientaci mezi důležitými, ručně vytvořenými, propojeními.

Propojení poznámek pomocí vektorových reprezentací. Pokročilejší přístup k automatickému propojování relevantních poznámek implementuje komunitní rozšíření *Smart Connections* od uživatele *brianpetro* [11]. Toto rozšíření využívá natrénovaných jazykových modelů pro vyhledávání sémanticky relevantních poznámek, které pak zobrazí v seznamu k aktuálně otevřené poznámce.

Rozšíření *Smart Connections* obsahuje funkci *Smart Views*, které pro každou poznámku vytvoří její vektorovou reprezentaci, což je číselná reprezentace dané poznámky, která zachytává sémantiku poznámky [12]. K vytvoření těchto reprezentací využívá předtrénované lokální jazykové modely. Díky tomu není potřeba odesílat žádná data třetím stranám a poznámky jsou zpracovávány pouze lokálně, což uživatelé ocení, zejména pokud jejich poznámky obsahují citlivé informace. Pomocí těchto vektorových reprezentací a počítání vzdáleností

[10] GitHub: `https://github.com/`
[11] Záložka *Plugins* v [8]: `https://obsidian.md/plugins`

mezi nimi je možné zobrazovat seznam nejrelevantnějších poznámek k aktuálně otevřené poznámce. [11]

Samotné rozšíření se nezaměřuje jen na hledání relevantních celých poznámek, ale i samostatných bloků v rámci poznámek, tedy pro aktuálně otevřenou poznámku může být relevantní pouze jeden blok z jiné poznámky a rozšíření *Smart Connections* umí uživatele nasměrovat přímo na daný blok.

Mimo funkci *Smart Views* rozšíření nabízí i funkci *Smart Chat*, pro kterou je potřeba do rozšíření vložit API klíč podporovaného LLM, který pak zajišťuje písemnou konverzaci o obsahu poznámek v reálném čase. Pokud uživatel tuto funkci povolí, tak už jsou data odesílána na servery třetích stran, což může nějaké uživatele z hlediska soukromí odradit.

Rozšíření *Smart Connections* implementuje podobnou funkcionalitu, která je cílem této práce, tedy automatické vyhledávání souvisejících poznámek. Úspěch tohoto rozšíření dokazuje, že automatické propojování je proveditelné a užitečné[12]. Nevýhoda tohoto rozšíření je taková, že je jedná o komunitní rozšíření, které je omezené pouze na platformu aplikace Obsidian a na datový typ Markdown. Pro jeho použití je nutné nainstalovat aplikaci Obsidian a poznámky spravovat v ní. Toto rozšíření spoléhá na rozhodnutí vývojářů aplikace Obsidian nadále podporovat komunitní rozšíření. Je implementován pouze jeden přístup pro hledání souvisejících poznámek, a to přes LLM a vektorové reprezentace poznámek.

Pro instalaci rozšíření je potřeba vypnout bezpečný režim aplikace Obsidian, který chrání poznámky a další soubory před potenciálními riziky ze strany komunitně vyvíjených rozšíření. Ty pak mají stejná přístupová práva jako Obsidian a mají přístup k souborům na počítači, mohou se připojovat na internet a instalovat další programy, což může být z bezpečnostního hlediska nevyhovující.[13]

V době rešerše ještě neexistuje podpora rozšíření pro mobilní zařízení, tedy veškerá funkcionalita je dostupná pouze v desktopové aplikaci Obsidian[14].

Oproti aplikaci Obsidian s rozšířením *Smart Connections* by se aplikace ThinkLink mohla více zaměřit na volnost v přidávání různých typů obsahu a dalších algoritmů pro automatické hledání souvisejících poznámek. Zároveň by ThinkLink mohl umožnit jednoduchou synchronizaci obsahu poznámek, která v základní aplikaci Obsidian chybí.

1.3.2 Mem.ai

Mem.ai [14] je nová poznámkovací aplikace, která integruje umělou inteligenci pro automatické nabízení relevantních poznámek uživateli v reálném čase. Aplikace je postavena na využívání velkých jazykových modelů a na

[12] Část *User Testimonials* v [11]
[13] Záložka *Plugin security* v [13]: `https://help.obsidian.md/plugin-security`
[14] Část *Limitations* v [11]

nehierarchickém přístupu k organizaci poznámek. Umožňuje jednoduše zapisovat poznámky, uspořádávat je do kolekcí a pak se nad poznámkami dotazovat pomocí AI asistenta.

Samotná aplikace je dostupná ve formě webové aplikace, desktopové aplikace pro operační systémy *macOS* a *Windows* a také jako nativní aplikace pro *iOS*. Mem.ai nepodporuje hierarchickou strukturu ukládání poznámek, uživatel má přímý přístup pouze k nedávno upraveným poznámkám, zatímco starší poznámky je nutné vyhledat pomocí fulltextového vyhledávání, AI asistenta či pomocí kolekcí, ve kterých lze poznámky seskupovat.

Aplikace se snaží o inteligentní propojování poznámek pomocí umělé inteligence. Při psaní poznámky automaticky hledá související poznámky pomocí sémantické podobnosti a klíčových slov, které pak zobrazuje v postranním panelu. Tímto způsobem usnadňuje uživatelům rychlé nalezení relevantních poznámek a zlepšuje jejich organizaci.

V aplikaci také fungují obousměrné odkazy. V rámci poznámky může uživatel zmínit jinou poznámku pomocí klíčového znaku „@“ a obě poznámky propojit. U poznámky, na kterou je odkazováno se pak v postranním panelu zobrazí seznam poznámek, kde je daná poznámka zmíněna. Narozdíl od aplikace Obsidian se ale už nezvýrazní konkrétní místo zmínky v původní poznámce, což může ztížit orientaci zejména při práci s větším množstvím delších poznámek.

Aplikace nabízí offline režim, který je podle průzkumu potřeb uživatelů velmi důležitý. V rámci offline režimu však nejsou dostupné žádné funkce založené na AI, tedy není možné konverzovat s poznámkami pomocí AI asistenta a automatické propojování nových poznámek s již existujícími poznámkami také nefunguje. Vzhledem k tomu, že je Mem.ai aplikace postavená hlavně na organizaci poznámek pomocí AI funkcí, absence těchto funkcí v offline režimu může výrazně omezit efektivitu práce s aplikací.

Aplikace Mem.ai také implementuje funkcionalitu, která je cílem této práce. Stejně jako u rozšíření *Smart Connections* platí, že je v aplikaci Mem.ai implementován pouze jeden přístup k automatickému hledání souvisejících poznámek. Během testování se ukázalo problematické to, že občas aplikace nabízela zcela nerelevantní poznámky jako související. Zároveň bylo zajímavé pozorování, že souvislost dvou poznámek není oboustranně stejná, tedy pro poznámku A se poznámka B objevila jako relevantní, ale opačně se žádná poznámka nezobrazila. Aplikace ThinkLink by se mohla pomocí srovnání více přístupů zaměřit na eliminaci těchto problémů.

1.3.3 Notion

Notion [15] je poznámková aplikace zaměřená hlavně týmovou spolupráci a organizaci informací. Nabízí přehledné uživatelské prostředí pro tvorbu stránek, tabulek, reportů či úkolů. Je to nástroj pro organizace či firmy, se kterým mohou organizovat důležité informace z projektů, dokumentaci, rozdělení práce

a sledování plnění cílů. Notion však mohou používat i jednotlivci pro osobní poznámky a plánování.

Přednost aplikace Notion oproti ostatním poznámkovým aplikacím je možnost organizovat informace a poznámky do přehledných databází a také je relačně propojovat mezi sebou, tvořit přehledy nad těmito databázemi a provádět jednoduché databázové operace. Další výhodou je rychlá synchronizace dat mezi jednotlivými zařízeními a možnost týmové spolupráce na jednotlivých poznámkách či databázích.

Notion také umožňuje propojení a integraci aplikace s velkým množstvím aplikací třetích stran, jako je *Google Drive*, *Dropbox*, *Trello*, *Confluence*, *Zapier* či *Slack* a jejich vzájemné propojení mezi sebou[15]. Integrace jsou velmi propracované a povyšují Notion na další úroveň, kdy v rámci svých poznámek uživatelé mohou využívat služby třetích stran přímo napojené v jedné aplikaci.

V rámci desktopové a mobilní aplikace Notion je možné zobrazovat a editovat obsah offline, který se pak synchronizuje, jakmile má zařízení přístup k internetu. Vzhledem k zaměření aplikace na spolupráci více lidí najednou je ale přístup k internetu klíčový. Notion také implementuje funkcionalitu obousměrných odkazů, takzvaných *backlinks* [16]. U každé poznámky je možné zobrazit, z jakých poznámek je na tu aktuální odkazováno. Po rozkliknutí obousměrného odkazu Notion zobrazí propojenou poznámku, ale nijak nezvýrazní přesné místo odkazu, což může znesnadňovat orientaci ve větším množství delších poznámek.

Notion také uchovává poznámky na svých cloudových serverech, což je výhodné pro rychlou synchronizaci, ale hrozí, že pokud Notion bude mít výpadek či nebude fungovat, tak ke svým poznámkám uživatelé nebudou mít přístup. Notion velmi dbá na bezpečnost dat a riziko úplné ztráty dat je nízké[16]. V případě potřeby je možné poznámky exportovat do formátů PDF, Markdown a HTML.

Notion také integruje svoji Notion AI[17] se kterou je možné konverzovat, vyhledávat informace jak v poznámkách, tak v připojených aplikacích třetí strany, generovat text na základě uživatelských vstupů a analyzovat texty, PDF a obrázky.

Nicméně Notion nenabízí možnost automatického propojování poznámek na základě obsahové a sémantické podobnosti. Uživatelé se musí spolehnout na vlastní znalost dokumentů a poznámek v aplikaci, vyhledávání či na AI asistenta.

1.3.4 RemNote

RemNote [17] je poznámková aplikace, která se od ostatních liší především tím, že se soustředí na proces rozloženého učení, což je jedna z efektivních metod

[15] Záložka *Integrations* v [15]
[16] Záložka *Security practices* v [15]
[17] Záložka *Meet the new Notion AI* v [15]

učení a účinného zapamatování si velkého množství informací [18] a integruje ho do tvorby jednotlivých poznámek.

RemNote nabízí webové rozhraní, desktopovou i mobilní aplikaci mezi kterými, na rozdíl od aplikace Obsidian, který nabízí pouze placenou funkci *Sync*, zdarma synchronizuje omezené množství dat a je možné aplikaci používat i offline. RemNote nabízí klasický textový editor, který je založený na Markdown formátu. Každou poznámku lze exportovat do různých formátů (Markdown, HTML, PDF či do Anki, což je oblíbená a specializovaná open-source aplikace pro tvorbu a procházení kartiček pro rozložené učení [19]).

Je možné RemNote využívat zdarma, aplikace nabízí i placenou verzi, ve které uživatel dostává přístup k tabulkám, většímu množství nahraných souborů či anotování PDF[18]. V rámci placeného RemNote je také k dispozici AI, která ale není využita na automatické propojování relevantních poznámek, ale na generování kartiček pro rozložené učení, shrnování obsahu poznámek, analýzu ručně psaných textů a dalších funkcionalit.

1.3.5 Shrnutí rešerše poznámkových aplikací

V rámci rešerše existujících aplikací bylo vybráno a vyzkoušeno několik různých poznámkových aplikací, které jsou si v mnoha ohledech podobné, ale každá se odlišuje svým zaměřením. Pochopení toho, jak zmíněné aplikace fungují, na co kladou důraz či jak implementují jednotlivé přístupy k tvorbě poznámek, je důležité pro následnou analýzu a návrh webové aplikace ThinkLink. Výsledky rešerše poznámkových aplikací a jejich srovnání podle definovaných kritérií v kapitole *1.2 Analýza potřeb uživatelů* jsou zobrazeny v tabulce 1.1. Uvažují se pouze funkcionality dostupné zdarma.

Nejlépe ze srovnání podle specifikovaných kritérií vychází aplikace Obsidian[19]. Žádná z aplikací nesplňuje všechna kritéria, která jsou pro uživatele důležitá. Ve srovnání také vidíme, že funkci automatického propojování poznámek implementuje jen aplikace Mem.ai nebo je možné rozšířit Obsidian o komunitní rozšíření, které tuto funkcionalitu nabízí. V samotné aplikaci Obsidian se ale taková funkce nenachází.

1.4 Různé přístupy v propojování dat na webu

Izolovaná data mají omezenou využitelnost. Propojováním dat na webu se zvyšuje informační hodnota těchto dat. Propojením získáme přehled o kontextu, o relevantních informacích a dalších zdrojích. Tato kapitola se zabývá různými přístupy k propojování textových dat. Získané poznatky z této oblasti následně slouží jako podklad pro analýzu a návrh aplikace ThinkLink zaměřené na automatické propojování poznámek.

[18] Záložka *Pricing* v [17]
[19] Kapitola *1.3.1 Obsidian*

Tabulka 1.1 Shrnutí analýzy existujících řešení podle implementace vybraných kritérií.

	Možnost zobrazovat a editovat poznámky offline	Automatická synchronizace poznámek	Dostupnost na různých zařízeních	Možnost připojit k poznámkám multimédia	Dostupné zdarma	Ochrana a soukromí poznámek	Fulltextové vyhledávání	Automatické hledání souvisejících poznámekext	Obousměrné propojování poznámek s označením místa propojení
Obsidian [8]	✓	–	✓	✓	✓	✓	✓	–	✓
Mem.ai [14]	–	✓	–	✓	–	–	✓	✓	–
Notion [15]	✓	✓	–	✓	✓	–	✓	–	✓
RemNote [17]	✓	✓	✓	✓	✓	–	✓	–	–

1.4.1 Propojování poznámek pomocí odkazů

Hypertextový odkaz je struktura, která odkazuje na nějaký zdroj na internetu. Typicky jde o zdroje definované pomocí URL (*Uniform Resource Locator*), což je unikátní adresa zdroje dostupného z webu [20]. Používání hypertextových odkazů je velmi přímočaré, uživatel na webu přechází pomocí hypertextových odkazů z jednoho zdroje na jiný. Hypertextový odkaz je jednosměrný a uchovává v sobě informaci, kam směřuje, ale samotný cíl nemá informaci o tom, odkud odkaz na daný cíl vedl. Jedná se o velmi jednoduché propojení, které ale s sebou nenese žádné informace či metadata navíc. [21]

Dobrý příklad a demonstrace využití hypertextových odkazů jsou webové stránky, např. *Wikipedia*[20], kde autoři článků manuálně vytvářejí odkazy na jiné články. Někdy se jedná o skutečně příbuzná témata, která se doplňují či rozvíjejí téma, o kterém čtenář čte, ale často se také jedná jen o shodu výskytu názvu jiného článku v aktuálně otevřeném článku, ale samotná témata vůbec nemusí být relevantní.

V rámci moderních poznámkových aplikací, jako je například Obsidian či Notion se rozšířil koncept bidirektivních odkazů [22]. Ty fungují tak, že poznámka A odkazuje na poznámku B, ale zároveň se v poznámce B automaticky zobrazí reference na poznámku A. Odkaz tedy propojuje poznámky oběma směry. Uživatel pak může jednoduše sledovat vzájemné vazby mezi poznámkami a pak získává intuitivnější přehled o struktuře obsahu. [23]

1.4.2 Linked Data a Semantic Web

Linked Data neboli propojená data je soubor principů, které definoval Tim Berners-Lee v roce 2006 a jejich cílem je publikování a propojování strukturovaných dat na webu. Cílem je propojit data z různých zdrojů tak, aby byla strojově čitelná, měla pevně definovaný význam, odkazovala na další externí data a mohla být odkazována z jiných externích datových setů. [24]

Tyto principy jsou (vlastní překlad z [25]):

1. Používejte URI pro pojmenování zdrojů
2. Používejte HTTP URI, aby si uživatelé dané zdroje mohli rovnou prohlédnout
3. Pokud si někdo vyhledá dané URI, poskytněte k němu kontext a informace. Používejte standardizované technologie (RDF, SPARQL)
4. Zahrňte odkazy na další URI, aby uživatelé mohli objevit více zdrojů

Díky těmto principům mohou být data na webu strojově čitelná a vzájemně propojitelná, což zlepšuje možnosti automatického vyhledávání vztahů mezi jednotlivými zdroji na webu. [24, 25]

[20] Wikipedia: `https://en.wikipedia.org/`

1.4.2.1 Linked Open Data (LOD)

Vedle *Linked data* existují i *Open data*, která lze volně používat a šířit. Jejich kombinací vznikla struktura zvaná *Linked Open Data* (LOD), kde jsou data jak propojená tak, aby jednotlivá propojení byly správně interpretovatelná i stroji, tak i volně distribuovaná a šiřitelná. Jedná se o data, která jsou uvolněna pod otevřenou licencí, která nebrání jejich bezplatnému a opětovnému použití. [25, 26]

Příkladem LOD je projekt *DBpedia*, což je komunitní projekt, jehož cílem je získat strukturovaný obsah z informací vytvořených v rámci Wikimedia projektů (např. Wikipedia, Wikibooks, Wiktionary apod.)[21]. Tyto informace pak tvoří otevřený znalostní graf, který je na webu dostupný všem. Znalostní graf je speciální druh databáze, který uchovává informace a znalosti ve strojově čitelné podobě a poskytuje prostředky ke sbírání, organizování, vyhledávání a využívání informací. [27]

1.4.2.2 Semantic Web

Semantic Web, také známý jako *Web of Data* (česky *Sémantický web*, vlastní překlad), je struktura, kde jednotlivě propojená data nejsou srozumitelná pouze pro člověka, ale i pro stroje. [26]

Současný stav webu, známý jako *Web of Documents*, kde jsou jednotlivé informace rozložené na jednotlivých „ostrovech“ či „silech“ a samotná data, zdroje a objekty mezi sebou nejsou jednoduše propojitelná. To zabraňuje efektivnímu provedení dotazů, které potřebují data z různých zdrojů. Pro zodpovězení takového dotazu je potřeba otevřít všechny relevantní dokumenty, přečíst si, co se v nich nachází a vyhodnotit, které informace jsou klíčové k zodpovězení daného dotazu, napsat a spustit program, který tato data z dokumentů extrahuje, vytvořit databázové schéma pro vybraná data, vytvořit transformační skript, který extrahovaná data z jednotlivých dokumentů vezme a vloží je do strukturované databáze a pak finálně vytvořit dotaz, jehož odpovědí bude výsledek, který jsme hledali. Tento postup je náročný a složitý. [28]

Cílem *Web of Data* je nepropojovat jednotlivé dokumenty, ale samotná data a zdroje mezi sebou pomocí technologií *Linked Data*, ve kterých se pomocí RDF implementuje decentralizovaný mechanismus pro popis vztahů mezi zdroji na webu. [26, 28]

1.4.3 Propojování pomocí outliningu

Outlining (česky *strukturování*, vlastní překlad) je přístup k organizaci a propojování poznámek, který se liší od klasického členění textu na nadpisy, podnadpisy a odstavce, kde jsou jednotlivé poznámky uložené ve složkách a pod-

[21] Wikimedia a projekty: `https://wikimediafoundation.org/about/`

složkách. V poznámkových aplikacích jako je *Logseq*[22] či *Roam Research*[23] je přístup k organizaci a propojování poznámek odlišný. V rámci poznámky se uživatel může odkazovat na jiné poznámky (podobně jako v dalších poznámkových aplikacích), ale také je možné se jednoduše odkazovat na samotné bloky informací v poznámkách. To se děje pomocí takzvaných blokových odkazů. Je to další z kroků pro lepší propojování jednotlivých částí informací nezávisle na jejich uložení v konkrétních poznámkách. [29, 30]

Jedná se o podobný přístup jako ve *Web of Data*, který také neřeší, v jakém dokumentu je informace uložena. Uživatel má možnost pospojovat různé myšlenky, fakta a názory bez manuální reorganizace souborů poznámek či kopírování.

Outlining funguje tak, že v poznámkovém systému existují samostatné bloky informací, které jsou do sebe zanořovány pomocí úrovní bodového seznamu. Úroveň zanoření určuje rodiče a potomky daného bloku a tímto způsobem je poznámka logicky organizovaná. Je možné zobrazit jenom danou úroveň bodového seznamu a její potomky (kteří se k ní vztahují). Díky tomu je možné poznámky rozdělit na více atomických a logicky propojených bloků, což usnadňuje pozdější vyhledávání informací. Jednotlivé bloky jde i zavírat a otevírat, takže si uživatelé mohou zobrazit jenom relevantní informace, což sníží jejich kognitivní zátěž. [29, 30]

1.4.4 Propojování dat pomocí vektorových reprezentací

Další z přístupů k propojování souvisejících informací využívá vektorový model. Jednotlivé dokumenty jsou v něm reprezentovány ve formě vektorových reprezentací (anglicky *vector embeddings*), které odráží jejich obsah a význam. Tyto vektory můžeme mezi sebou porovnávat, provádět výpočty a vyhodnocovat, jak moc jsou si podobné. Vektorové reprezentace jsou tedy způsob, jak převést nestrukturovaná data (text, zvuk nebo obrázky) na pole čísel, které je možné jednoduše algoritmicky zpracovat. [12, 31]

Vzhledem k zaměření práce na automatické hledání souvisejících textových poznámek, zpracování jiných forem obsahu (zvuk či obrázek) nebude popsáno.

Jednotlivé přístupy v rámci vektorového modelu se liší hlavně ve dvou směrech. Liší se hlavně tím, jakým způsobem se vytvářejí jednotlivé vektorové reprezentace textových dokumentů a na druhé straně se přístupy mohou lišit tím, jak se pak jednotlivé vektory mezi sebou porovnávají. Vektorový model nám podle vzájemné polohy vektorů umožňuje měřit podobnosti mezi dokumenty, vyhledávat texty podle toho, jak jsou obsahově podobné dotazu či propojovat a seskupovat jednotlivé dokumenty. [31, 32]

[22] Logseq: `https://logseq.com/`
[23] Roam Research: `https://roamresearch.com/`

1.4.4.1 Bag of Words (BoW)

Bag of words (česky *pytel slov*, vlastní překlad) je jeden z jednodušších způsobů, jak převést text na vektor. Za jednotlivé dokumenty se považují soubory slov bez žádného ohledu na pořadí či gramatiku. [32]

Každý dokument je reprezentován vektorem pevné délky. Délkou je většinou počet unikátních slov v celém korpusu dokumentů. A každá složka vektoru udává frekvenci výskytů daného slova v dokumentu. [32]

Výhodou je jednoduchost přístupu a implementace. Nevýhodou je to, že tento způsob ignoruje pořadí slov v dokumentu a tím, že jsou jednotlivá slova chápána jako nezávislé tokeny, tak není možné zachytit kontext, sémantiku ani význam dokumentu. Vzhledem k tomu, že v korpusu dokumentů je většinou opravdu velký počet slov a pouze malá podmnožina těchto slov se nachází v konkrétním dokumentu, tak se většinou pracuje s vysoko dimenzionálními vektory, které jsou velmi řídké. [32]

1.4.4.2 TF-IDF

TF-IDF (neboli *term frequency–inverse document frequency*) je přístup, který je rozšířením BoW a jeho myšlenkou je měřit důležitost slov v dokumentu vzhledem k celé kolekci dokumentů. Čím častěji je slovo v dokumentu, ale vzácněji v celé kolekci, tím vyšší má váhu. [31, 32]

První část (*term frequency*, česky *četnost termu*) vyjadřuje četnost slova neboli termu v dokumentu. Čím častěji se term v dokumentu objevuje, tím je pro daný dokument potenciálně důležitější. [31]

Druhá část (*inverse document frequency*, česky *inverzní četnost dokumentů*) udává míru důležitosti termu v rámci celého korpusu dokumentů. Pokud je term obsažen ve převážné většině dokumentů, není pro vzájemné odlišení dokumentů příliš důležitý. Často to bývají tzv. stop slova, což je seznam slov, které nemají velký sémantický význam pro analýzu dokumentů, protože jsou v textu příliš frekventovaná (většinou se jedná o spojky a předložky) [33]. [31]

Výhodou TF-IDF je, že tento přístup redukuje vliv frekventovaných slov (jako jsou spojky či zájmena) na podobnost dokumentů a obecně zohledňuje důležitost slov tak, že přisuzuje vyšší váhu slovům, která jsou častá pro daný dokument a zároveň vzácná v celém korpusu dokumentů. Další výhodou je i relativní jednoduchost implementace přístupu. [31]

Ale podobně jako BoW, TF-IDF stále ignoruje pořadí jednotlivých slov a nezachycuje kontext, fráze, idiomy a sémantické vztahy mezi slovy (např. synonyma), které jsou pro vyhodnocení podobnosti dokumentů důležité. [32]

1.4.4.3 Word2vec

Word2vec patří do třídy jazykových modelů pro výpočet souvislých vektorových reprezentací z velkých datových sad. Hlavní myšlenkou je, že slova s podobným významem budou mít podobné vektorové reprezentace a proto tyto

vektory mohou být použity k měření sémantické a syntaktické podobnosti mezi slovy. [32, 34]

Jednotlivé vektory umožňují provádění algebraických operací mezi nimi, aby bylo možné zodpovědět otázky týkající se sémantického významu slov a jejich vztahů. [34]

Například, Mikolov [34] uvádí příklad vztahu slov *velký–největší* a *malý–nejmenší*. Sémantický vztah mezi nimi je velmi podobný a pomocí operace:

$$\texttt{vector(„největší")}–\texttt{vector(„velký")} + \texttt{vector(„malý")}$$

nám vrátí vektor, který odpovídá `vector(„nejmenší")`.

Word2vec funguje tak, že se pomocí dvou hlavních modelů (*CBoW* nebo *Skip-gram*) učí reprezentace slov na základě jejich kontextu v rozsáhlém textovém korpusu. Způsobil revoluci v tom, že umožnil jednotlivá slova reprezentovat jako body v souvislém vektorovém prostoru, což umožnilo algoritmům lépe interpretovat význam a vztahy mezi slovy a položil základy pro současné jazykové modely. [32, 34]

1.4.4.4 Předtrénované modely

Zatímco modely jako Word2vec nebo GloVe[24] je možné vytrénovat od začátku, tak se nabízejí i jejich předtrénované varianty na velkém množství veřejných textových datech, jako je Wikipedia[25] a Common Crawl[26]. [12]

Stejně tak je možné pro získání vektorových reprezentací využít předtrénované velké jazykové modely jako je BERT [35] (v různých variantách – například S-BERT [36]) či specializované modely například od OpenAI [37] či Google Gemini [38] zaměřené na generování vektorových reprezentací textů.

Aby tyto modely byly schopné vytvořit odpovídající vektorovou reprezentaci, která zahrnuje sémantické informace a kontext daného textu, musí být natrénované na obrovském množství textových dat. [12]

Současné využití vektorových reprezentací textů je široké. Podle tvůrců modelů v OpenAI [37] a Google [38] se vektorové reprezentace používají na:

- **vyhledávání** – výsledky jsou řazené podle relevance dokumentu k zadanému dotazu,
- **shlukování** – dokumenty jsou seskupeny podle jejich relevance,
- **doporučování** – dokumenty blízké vybranému dokumentu jsou doporučeny,
- **detekce anomálií** – identifikace odlehlých hodnot, které nejsou moc podobné zbytku dokumentů,

[24] GloVe od Stanfordské univerzity: `https://nlp.stanford.edu/projects/glove/`
[25] Wikipedia: `https://www.wikipedia.org/`
[26] Common Crawl: `https://commoncrawl.org/`

- **měření diverzity** – analýza distribuce podobnosti a
- **klasifikace** – jednotlivé dokumenty jsou klasifikovány podle jejich nejpodobnějšího označení.

Jednotlivé reprezentace se pak většinou ukládají do vektorových databází, které umí efektivně pracovat s vektory a provádět s nimi různé operace. [38]

1.4.5 Hledání podobností mezi vektorovými reprezentacemi

Velké jazykové modely pouze převedou text na jeho vektorovou reprezentaci. Pro zjištění podobnosti různých dokumentů je potřeba vypočítat vzájemnou polohu odpovídajících vektorových reprezentací. Pro výpočet vzdáleností mezi vektory existuje několik různých metrik.

1.4.5.1 Kosinová podobnost

Pomocí kosinové podobnosti se podobnost mezi vektory počítá pomocí úhlu, který svírají jejich směry. Tato metrika není ovlivněna velikostí jednotlivých vektorů, pouze úhlem mezi nimi. [39]

Funguje tak, že se vypočítá kosinus úhlu, který mezi sebou dva vektory svírají. Pokud jsou oba vektory totožné či velmi podobné, tak svírají malý úhel a jejich podobnost je vysoká. Tato metrika se hodí porovnávání vektorových reprezentací textů s různou dimenzí, protože ignoruje celkovou velikost vektorů. Jedná se o standardní a nejběžnější způsob výpočtu sématické podobnosti mezi jednotlivými dokumenty. [39]

Naopak, tato metrika se nehodí pro porovnávání vektorů u kterých je jejich velikost důležitá a je potřeba ji zohlednit (např. porovnávání obrázků na základě intenzity pixelů). [40]

1.4.5.2 Euklidovská vzdálenost

Euklidovská vzdálenost je tradiční metrika počítání vzdáleností mezi vektory ve vektorovém prostoru. V tomto prostoru měří geometrickou vzdálenost mezi jednotlivými body. Čím menší je vzdálenost mezi body, tím jsou si vektory blíže. [40]

Tento přístup je citlivý na velikosti samotných vektorů. Pokud máme například dokumenty *d1* a *d2*, kdy *d2* obsahuje dvakrát zkopírovaný dokument *d1*, tedy je mu významově podobný, ale je dvakrát větší, tak Euklidovská vzdálenost počítá příliš velkou vzdálenost oproti kosinové podobnosti, pro kterou by se jednalo o dva vektory se stejným směrem. [40]

Kapitola 2
Analýza

Tato kapitola popisuje analýzu webové aplikace ThinkLink, která umožňuje vedle správy poznámek také automaticky hledat relevantní poznámky a zobrazovat je uživateli.

Nejdříve je provedena podrobná analýza poznámkové aplikace Obsidian, která ze srovnání poznámkových vyšla nejlépe. Cílem je z aplikace převzít část oblíbených funkcionalit a zajistit vzájemnou kompatibilitu s aplikací ThinkLink.

V rámci analýzy také vzniknou funkční a nefunkční požadavky na aplikaci ThinkLink. Společně s definovanými případy užití a doménovým modelem se bude jednat o základ pro návrh a implementaci.

2.1 Analýza aplikace Obsidian

Aplikace Obsidian [8] byla představena v části rešerše jako jedna z aplikací pro pokročilou správu poznámek a jejich propojování. Z porovnání s ostatními dostupnými aplikacemi v sekci *1.3.5 Shrnutí rešerše poznámkových aplikací* podle kritérií, která byla stanovena, vyšla jako nejvhodnější pro efektivní práci s poznámkami.

Podrobná analýza aplikace Obsidian je pro návrh aplikace ThinkLink klíčová, protože pracuje s otevřeným formátem Markdown, umožňuje k poznámkám přidávat multimédia či propojovat poznámky pomocí odkazů. To jsou vlastnosti, které by bylo vhodné v rámci aplikace ThinkLink zachovat, protože jsou obecně důležité pro uživatele poznámkových aplikací. Výsledky analýzy potřeb uživatelů popisuje sekce *1.2.2 Výsledky dotazníku*.

Naopak, aplikace Obsidian nenabízí webovou verzi a bezplatnou funkci synchronizace poznámek mezi jednotlivými zařízeními, což jsou vlastnosti, které jsou pro uživatele podle analýzy potřeb uživatelů důležité. Samotná aplikace Obsidian také nenabízí možnosti automatického hledání souvisejících poznámek.

Této funkce lze docílit formou komunitního rozšíření *Smart Connections*, které lze do aplikace dodatečně nainstalovat. Toto rozšíření s sebou ale přináší několik omezení a rizik. Ty jsou popsány v sekci *1.3.1 Obsidian*.

Cílem analýzy aplikace Obsidian je zjistit, jakým způsobem aplikace pracuje se soubory, jak interpretuje Markdown formát a jak implementuje propojování jednotlivých poznámek pomocí obousměrných odkazů. Pomocí této analýzy bude možné navrhnout webovou aplikaci ThinkLink tak, aby nabízela podobné funkcionality jako Obsidian, které jsou pro uživatele důležité a zároveň umožnila stávajícím uživatelům Obsidianu alternativu, která implementuje automatické hledání souvisejících poznámek a řeší nevýhody komunitního *Smart Connections* rozšíření.

Požadavkem je, aby aplikace ThinkLink uměla zpracovávat a zobrazovat jednotlivé soubory stejným nebo podobným způsobem, který je kompatibilní s aplikací Obsidian, čímž se předejde nutnosti provádět změny v jednotlivých souborech při paralelním využíváním obou aplikací. Pokud by se uživatel rozhodl přemístit celou práci s poznámkami nebo její část z aplikace Obsidian do aplikace ThinkLink, díky vzájemné kompatibilitě pak bude přechod velmi jednoduchý a snadný.

2.1.1 Markdown

Při zobrazování poznámek Obsidian vychází primárně z toho, jak Markdown funguje[1], což z Obsidianu dělá velmi univerzální nástroj pro správu poznámek, které pak jsou uložené v souborovém systému zařízení a v univerzálním Markdown formátu, který je lidsky čitelný i v textové podobě bez žádného interpretačního nástroje.

Aby byla aplikace ThinkLink kompatibilní s nástroji jako je Obsidian, tak také bude správně interpretovat Markdown a zobrazovat poznámky stejně jako Obsidian. Obsidian pokročile interpretuje Markdown soubory, včetně složitějších konstrukcí, jako jsou tabulky, matematické výrazy či grafy. K tomu využívá nástroje třetích stran, zejména Mermaid a MathJax[2].

2.1.2 Odkazy v rámci aplikace Obsidian

Zásadní funkcí Obsidianu jsou obousměrné odkazy mezi jednotlivými poznámkami. Obsidian akceptuje dva způsoby psaní odkazů: *Wikilink odkaz* a *Markdown odkaz*.[3]. Jejich syntax je následující:

- Wikilink odkaz:
  - `[[Odkaz na jinou poznámku]]`

[1] Záložka *Basic formatting syntax* v [13]: `https://help.obsidian.md/syntax`

[2] Záložka *Advanced formatting syntax* v [13]: `https://help.obsidian.md/advanced-syntax`

[3] Záložka *Internal links* v [13]: `https://help.obsidian.md/links`

- Markdown odkaz:
  - `[Odkaz na jinou poznámku](Odkaz%20na%20jinou%20poznamku.md)`

Pro *Markdown odkazy* je nutné název cílového souboru zakódovat pomocí ASCII. Standardně Obsidian používá formát *Wikilink odkazů*, hlavně kvůli přehlednějšímu a kompaktnějšímu zápisu. V nastavení je možné změnit používání *Wikilink* na *Markdown* formát.

Pokud chce uživatel přidat soubory (např. obrázek či PDF), tak také může použít *Wikilink* formát: `[[Obrazek1.png]]`. Tento zápis pouze vygeneruje odkaz na daný soubor, který se po kliknutí otevře na nové záložce. Mnohem praktičtější varianta, hlavně pro obrázky, je přidat k odkazu i náhled samotného obrázku. Toho je docíleno přidáním vykřičníku před definici odkazu: `![[Obrazek1.png]]`.

Dále uživatel může ovlivnit šířku náhledu obrázku tím, že do *Wikilink* odkazu přidá svislou čáru a za něj počet pixelů podle požadované šířky obrázku. Obsidian šířku upraví a zachová poměr stran obrázku.

- Obrázek se specifikovanou šířkou 200 px: `![[Obrazek1.png|200]]`.

Naopak, *Markdown* formát odkazu se používá pro přidávání externích odkazů do poznámek. Tedy: `[Obsidian Help](https://help.obsidian.md)`[4].

2.1.3 HTML v aplikaci Obsidian

Povolením HTML ve svých poznámkách má uživatel více možností zobrazit obsah tím způsobem, jakým přesně potřebuje. Může využít možnosti HTML, CSS i Javascriptu a zobrazit struktury, které by v rámci Markdown formátu nebylo možné definovat. Na druhou stranu, povolením HTML se uživatel vystavuje riziku začlenění nebezpečného Javascriptového kódu pomocí elementu `<script>`, který může zjistit citlivé informace ze zařízení a může je odeslat přes internet útočníkovi, který uživatele přesvědčil, aby do svých poznámek vložil poznámku s tímto kódem a zobrazil ji.

Aplikace Obsidian povoluje využití HTML, ale pouze k omezeným účelům (komentáře či stylování textu). HTML kód prochází úpravou, aby se předešlo jakýmkoliv škodlivým skriptům.[5]

2.1.4 Souborový systém v aplikaci Obsidian

Obsidian jako samotný soubory uživatele neukládá či si je nijak interně nedrží, pouze interpretuje soubory, které jsou uložené na souborovém systému zařízení. Pokud uživatel chce, aby Obsidian začal interpretovat jeho soubory, tak musí nejdříve založit tzv. *Obsidian Vault*, kde specifikuje kořenovou složku

[4] Záložka *Basic formatting syntax* v [13]: `https://help.obsidian.md/syntax`
[5] Záložka *HTML content* v [13]

v souborovém systému počítače, ke které bude mít aplikace Obsidian přístup a bude moci interpretovat veškeré soubory, které se v této složce a podsložkách nachází. V rámci Obsidianu je také možné nastavit speciální adresáře pro:

- pro nově přidané, a ještě nezařezené poznámky,
- pro nahrané soubory (např. obrázky či PDF) a
- pro definované šablony pro rychlejší a uniformnější vytváření nových poznámek se stejnou strukturou.

Obsidian kombinuje dva hlavní přístupy k organizaci poznámek. Poznámky lze organizovat pomocí hierarchického (složkového) přístupu anebo pomocí grafového přístupu (např. podle metodiky *Zettelkasten*[6]), kdy nezáleží na místě uložení poznámky v hierarchii, ale jednotlivé poznámky na sebe odkazují pomocí obousměrných odkazů.[7]

2.2 Analýza požadavků

V této sekci jsou stanoveny požadavky na webovou aplikaci ThinkLink. Tyto požadavky umožní zpřesnit odhad pracnosti projektu, vymezit hranice vytvářeného systému a definovat omezení kladená na samotnou aplikaci.

Požadavky jsou rozděleny do dvou hlavních částí. Funkční požadavky jsou požadavky ohledně funkčnosti aplikace a definují, co všechno by aplikace měla umět a splňovat. Nefunkční požadavky definují různá omezení na systém a dodržování standardů.

2.2.1 Funkční požadavky

F1: Přihlášení a správa uživatelů Aplikace umožní vytvářet a spravovat jednotlivé uživatelské účty. Běžný uživatel bude mít vytvořený účet a bude moci přistupovat ke svým poznámkám, souborům a nastavení vlastního účtu. Správa všech účtů bude zpřístupněna pouze uživatelům s administrátorskými právy. Určitá část aplikace bude přístupná pouze přihlášeným uživatelům a zároveň pro nepřihlášené uživatele bude k dispozici přihlašovací či registrační stránka.

- Priorita: **vysoká**.

F2: Správa poznámek Aplikace přihlášenému uživateli umožní vytvářet, zobrazovat, upravovat a mazat jednotlivé poznámky.

- Priorita: **vysoká**.

[6] Kapitola *1.1.2 Zettelkasten – analogové propojování poznámek*
[7] Záložka *How Obsidian stores data* v [13]: `https://help.obsidian.md/data-storage`

F3: Import a nahrávání souborů Aplikace umožní nahrávání či import jednotlivých poznámek a souborů, import více poznámek a souborů najednou či import celé složky poznámek a souborů zabalené v ZIP archivu. Aplikace umožní snadnou rozšiřitelnost pro nahrávání dalších typů souborů.

- Priorita: **vysoká**.

F4: Zobrazování obrázků a PDF souborů Nahrané obrázky a PDF soubory bude možné zobrazit a stáhnout. Do jednotlivých poznámek bude možné jednoduše vložit odkaz na nahraný soubor či obrázek. U obrázků bude možné zobrazit náhled obrázku v poznámce. Aplikace umožní snadnou rozšiřitelnost pro práci s dalšími typy souborů.

- Priorita: **vysoká**.

F5: Propojování jednotlivých poznámek pomocí odkazů V poznámce bude možné vytvořit jednoduchý funkční odkaz na jinou poznámku.

- Priorita: **vysoká**.

F6: Automatické propojování souvisejících poznámek na základě obsahové a sémantické podobnosti Aplikace bude implementovat několik různých přístupů k automatickému hledání a propojování poznámek. Při zobrazení poznámky nabídne seznam nejrelevantnějších poznámek s přímými odkazy na dané poznámky. Aplikace také umožní snadnou rozšiřitelnost pro nové způsoby a metody automatického hledání souvisejících poznámek.

- Priorita: **vysoká**.

F7: Cizí jazyky Aplikace umožní uživateli přepínat jazyk prostředí aplikace. Bude umožněno jednoduché přidání nových překladů.

- Priorita: **nízká**.

F8: Vlastní CSS stylování Aplikace umožní definici vlastních CSS stylů, které se pak aplikují na jednotlivé poznámky a umožní tak uživateli definovat unikátní vzhled poznámek.

- Priorita: **nízká**.

F9: Možnost zveřejnit část poznámek na web Aplikace umožní označit poznámky, které budou po zveřejnění dostupné k zobrazení i nepřihlášeným uživatelům. Kdykoliv bude možné zveřejnění poznámky vrátit zpět.

- Priorita: **nízká**.

F10: Export jednotlivých poznámek Aplikace umožní export vybraných poznámek do formátů PDF, HTML či Markdown.

- Priorita: **nízká**.

F11: Fulltextové vyhledávání Aplikace umožní uživatelům vyhledávat poznámky na základě fulltextového vyhledávání klíčových slov.

- Priorita: **střední**.

2.2.2 Nefunkční požadavky

N1: Webová aplikace Aplikace bude nabízet webové rozhraní a bude připravená na nasazení do produkčního prostředí na vzdáleném serveru.

N2: Responzitivita aplikace Webová aplikace bude responzivní a bude přizpůsobovat svůj vzhled v závislosti na rozlišení zařízení. Bude umožněna práce na počítači, tabletu a mobilním telefonu.

N3: Technologie Webová aplikace bude vyvíjena v jazyce PHP s využitím frameworku Symfony. Jako databázové řešení bude zvolena relační databáze PostgreSQL.

N4: Příprava na nasazení a jednoduchou spolupráci Aplikace bude vyvíjena v prostředí Docker, které umožní jednodušší spolupráci mezi vývojáři snadnou replikovatelností vývojového prostředí a zjednoduší proces nasazení na vzdálený server.

N5: Architektura aplikace Logická architektura aplikace bude mít formu MVC.

N6: Zabezpečení aplikace Aplikace bude respektovat zásady a postupy pro bezpečný provoz webové aplikace. Aplikace nabídne autentizační a autorizační mechanismy a uživatele s různými oprávněními. Soubory budou uložené mimo veřejnou složku serveru, aby k nim nebyl možný přístup neautorizovaným uživatelům. Aplikace také zamezí SQL injection útokům pomocí předpřipravených dotazů a CSRF ochranu u formulářů.

2.3 Model případů užití

V modelu případů užití jsou funkční požadavky zachyceny z hlediska interakce uživatele s aplikací. Tento model slouží k zpřesnění požadavků a lepšímu porozumění očekávanému chování systému. Model případů užití společně s funkčními požadavky a doménovým modelem tvoří základ pro implementaci samotné aplikace.

2.3.1 Seznam aktérů

Aktéři v modelu případů užití představují subjekty, které interagují s aplikací a využívají její funkce.

V rámci aplikace ThinkLink rozlišujeme následující aktéry: nepřihlášený uživatel, přihlášený uživatel a přihlášený administrátor.

Nepřihlášený uživatel

Nemá přístup do aplikace ThinkLink. Může zobrazit přihlašovací či registrační stránku aplikace, přihlásit se, případně změnit jazyk přihlašovacího prostředí. Pokud se pokusí zobrazit část aplikace ke které nemá přístup, nebude mu zpřístupněna a bude vyzván k autentizaci. Pokud se uživatel úspěšně přihlásí či registruje, podle přiřazené role se z něho stane přihlášený běžný uživatel nebo přihlášený administrátor.

Přihlášený uživatel

Má přístup do aplikace ThinkLink a může využívat všechny její funkcionality. Nemá přístup ke správě ostatních uživatelů ani k uloženým poznámkám, souborům a nastavením jiných uživatelů, pokud nejsou veřejné. Pokud se odhlásí, tak se stane nepřihlášeným uživatelem.

Přihlášený administrátor

Má přístup do aplikace ThinkLink ve stejném rozsahu jako přihlášený uživatel. K tomu má přístup ke správě ostatních uživatelů, ale z hlediska soukromí nemá přístup k jednotlivým poznámkám a souborům ostatních uživatelů, pokud nejsou veřejné. Po odhlášení se stane nepřihlášeným uživatelem.

2.3.2 Případy užití

Následuje seznam jednotlivých případů užití. Každý případ užití obsahuje textový popis, aktéra a pokud je to vhodné, tak i scénář případu užití.

UC1: registrace do aplikace ThinkLink

- **Aktér:** nepřihlášený uživatel

Případ užití umožňuje aktérovi zaregistrovat nový účet, který je potřeba k přístupu do aplikace ThinkLink. Aktér zobrazí registrační stránku, zadá svůj e-mail a nové heslo. Aplikace zkontroluje unikátnost e-mailu a daná omezení na heslo a pokud je kombinace e-mailu a hesla vyhovující, tak je aktérovi založen nový účet, který může použít pro přihlášení do aplikace ThinkLink (viz případ užití UC2).

UC2: přihlášení do aplikace ThinkLink

- **Aktér:** nepřihlášený uživatel

Aktér se přihlásí do aplikace ThinkLink pomocí svých přihlašovacích údajů (e-mail a heslo). Pokud jsou údaje platné, stane se z něj buď přihlášený uživatel, nebo přihlášený administrátor (dle role přiřazené v systému).

1. Nepřihlášený uživatel otevře přihlašovací stránku.
2. Zadá svoje přihlašovací údaje.
3. Aplikace ověří kombinaci e-mailu a hesla.
4. Pokud jsou přihlašovací údaje nesprávné, aplikace zobrazí chybovou hlášku a nabídne možnost přihlášení zopakovat.
5. Pokud je autentizace úspěšná, aplikace přiřadí uživateli odpovídající roli (běžný uživatel / administrátor).
6. Aplikace zobrazí hlavní stránku aplikace ThinkLink.

UC3: odhlášení uživatele

- **Aktér:** přihlášený uživatel

Aktér se odhlásí z aplikace, čímž se stane nepřihlášeným uživatelem a ztratí přístup do aplikace ThinkLink.

1. Aktér otevře uživatelské menu a zvolí volbu „Odhlásit“.
2. Aplikace ukončí uživatelskou relaci a vrátí uživatele na přihlašovací obrazovku.
3. Aktér se stává nepřihlášeným uživatelem.

UC4: zobrazení a správa ostatních uživatelů

- **Aktér:** přihlášený administrátor

Administrátor může zobrazovat přehled všech uživatelů v systému a provádět nad nimi akce – vytvoření nového uživatele, odstranění existujícího uživatele a úprava jejich údajů. Běžný přihlášený uživatel tyto práva nemá.

UC4.1: vytvoření nového uživatele

- **Aktér:** přihlášený administrátor

1. Administrátor v sekci „Správa uživatelů“ vybere možnost „Přidat uživatele“.
2. Vyplní registrační formulář: uživatelské jméno, e-mail, heslo, nastavení role (běžný uživatel / administrátor), podobně jako v UC1.
3. Aplikace zvaliduje zadaná data (délka hesla, jedinečnost e-mailu atd.).
4. Aplikace založí nový účet a uloží jej do databáze.
5. Administrátor vidí nově založeného uživatele v seznamu všech uživatelů.

UC4.2: odstranění existujícího uživatele

- **Aktér:** přihlášený administrátor

1. Administrátor v sekci „Správa uživatelů“ zobrazí seznam uživatelů.
2. Vybere uživatele, kterého chce odstranit, a zvolí „Odstranit uživatele“.
3. Aplikace zobrazí potvrzovací okno.
4. Administrátor potvrdí akci.
5. Aplikace odstraní uživatele (resp. jeho účet) z databáze a zobrazí aktualizovaný seznam.

UC4.3: úprava existujícího uživatele

- **Aktér:** přihlášený administrátor

1. Administrátor v sekci „Správa uživatelů“ zobrazí seznam uživatelů.
2. Vybere uživatele, kterého chce upravit, a zvolí „Upravit“.
3. Aplikace zobrazí editační formulář (změna jména, role atd.).
4. Administrátor provede požadované změny.
5. Aplikace validuje zadaná data a uloží je.
6. Aplikace zobrazí aktualizované údaje o uživateli.

UC5: správa poznámek

- **Aktér:** přihlášený uživatel

UC5.1: vytvoření poznámky

- **Aktér:** přihlášený uživatel

1. Aktér zvolí „Vytvořit novou poznámku“.
2. Aplikace zobrazí formulář s poli „Název“ a „Obsah“.
3. Aktér vyplní název a obsah poznámky.
4. Aplikace poznámku uloží do databáze a zobrazí ji v seznamu poznámek.

UC5.2: zobrazení poznámky

- **Aktér:** přihlášený uživatel

1. Aktér v seznamu poznámek klikne na vybranou poznámku.
2. Aplikace zobrazí detail poznámky společně s automaticky nalezenými souvisejícími poznámkami (UC7).

UC5.3: úprava poznámky

- **Aktér:** přihlášený uživatel

1. Aktér v detailu poznámky nebo v seznamu zvolí možnost „Upravit“.
2. Aplikace zobrazí editační formulář se stávajícím názvem a obsahem.
3. Aktér provede požadované změny.
4. Aplikace uloží změny do databáze a zobrazí aktualizovaný obsah.

UC5.3.1: vytvoření odkazu na jinou poznámku

- **Aktér:** přihlášený uživatel

1. Aktér v editačním formuláři vloží *Wikilink* s referenčním názvem cílové poznámky.
2. Aktér uloží změny.
3. Aplikace podle referenčního názvu v odkazu najde cílovou poznámku a vytvoří HTML odkaz na tuto poznámku.
4. Pokud se nepodaří cílovou poznámku v databázi najít, aplikace vytvoří neplatný odkaz, který bude vizuálně odlišený od platných odkazů.

UC5.3.2: přidání náhledu obrázku do poznámky

- **Aktér:** přihlášený uživatel

1. Aktér v editačním formuláři vloží *Wikilink* odkaz na soubor obrázku, který je již nahraný na serveru (UC6) pomocí jeho referenčního názvu.
2. Aktér uloží změny.
3. Aplikace podle referenčního názvu souboru nalezne daný soubor v databázi a pokud je to obrázek, tak zobrazí jeho náhled.

UC5.3.3: přidání odkazu na PDF soubor

- **Aktér:** přihlášený uživatel

1. Postup je stejný jako v UC5.3.2.
2. Pokud aplikace zjistí, že odkazovaný soubor je ve formátu PDF, tak do textu poznámky vloží odkaz na daný soubor, který se pak zobrazí v další záložce prohlížeče.

UC5.4: odstranění poznámky

- **Aktér:** přihlášený uživatel

1. Aktér zvolí v detailu poznámky volbu „Smazat“.
2. Aplikace zobrazí potvrzovací okno.
3. Aktér potvrdí akci.
4. Aplikace smaže poznámku z databáze.

UC6: import souborů a poznámek

- **Aktér:** přihlášený uživatel

UC6.1: import jedné a více poznámek či jednoho a více souborů

- **Aktér:** přihlášený uživatel

1. Aktér zvolí možnost „Nahrát poznámky a soubory“.
2. Vybere jeden či více souborů (ve formátu `.txt`, `.md`, `.pdf`, `.jpg` apod.).
3. Pokud uživatel nahraje nepodporovaný typ souboru, aplikace zobrazí chybovou hlášku a vybídne uživatele k opětovnému nahrání souborů.
4. Aplikace pro každý soubor určí, zda se jedná o poznámku či přílohu, a uloží je.
5. Aplikace zobrazí aktualizovaný seznam všech poznámek a nahraných souborů.

UC6.2: import ZIP archivu s poznámkami a soubory

- **Aktér:** přihlášený uživatel

1. Aktér zvolí možnost „Nahrát poznámky a soubory“.
2. Aktér vybere ZIP archiv a nahraje jej do aplikace.
3. Aplikace ZIP rozbalí na serveru, zpracuje a zvaliduje nalezené soubory a pro každý podporovaný soubor určí, jakým způsobem se uloží.
4. Pokud uživatel nahraje ZIP archiv, který obsahuje nepodporovaný typ souboru, aplikace ho na to upozorní pomocí chybové hlášky a vybídne uživatele k opětovnému nahrání aktualizovaného ZIP archivu.
5. Aplikace pro každý soubor vytvoří záznam v databázi a přiřadí je k uživateli.
6. Aplikace zobrazí aktualizovaný seznam všech poznámek a nahraných souborů.

UC7: zobrazení automaticky nalezených souvisejících poznámek

- **Aktér:** přihlášený uživatel

Aplikace automaticky analyzuje obsah poznámek a k aktuálně zobrazené poznámce zobrazuje seznam poznámek, které jsou podle zvoleného algoritmu relevantní.

1. Aktér zobrazí detail konkrétní poznámky (UC5.2).
2. Aplikace automaticky nalezne nejrelevantnější poznámky k aktuálně zobrazené poznámce.
3. Aplikace zobrazí seznam těchto poznámek, které jsou vyhodnoceny jako nejvíce související, v postranním panelu.
4. Aktér může na jednotlivé poznámky kliknout a zobrazit si je (UC5.2).

1. Aktér přejde do sekce „Nastavení aplikace“.
2. Změní hodnotu prahu skóre podobnosti pro aktuálně vybraný algoritmus (UC8).
3. Aplikace uloží změnu a následně filtruje zobrazené poznámky dle tohoto prahu.

UC8: změna jazyka prostředí aplikace

- **Aktér:** nepřihlášený uživatel, přihlášený uživatel

Uživatel v pravém horním rohu klikne na přepínač jazyků a vybere jím preferovanou jazykovou verzi prostředí aplikace. Aplikace se pak přepne do zvoleného jazyka (nikoliv však texty samotných poznámek nebo v nahraných souborech).

UC9: změna stylu poznámek pomocí CSS stylů

- **Aktér:** přihlášený uživatel

1. Aktér přejde do sekce „Nastavení aplikace“.
2. Aktér vloží nebo upraví CSS kód a potvrdí změny.
3. Aplikace dané změny uloží.
4. Aplikace nevaliduje správnost CSS kódu a nebere za jeho vykonání žádnou zodpovědnost, uživatele na to upozorní.
5. Aktér může zobrazit poznámku (UC5.2) a na danou poznámku budou aplikované specifikované CSS styly.
6. Pokud nejsou specifikované žádné vlastní CSS styly, aplikace zobrazí základní styly.

UC10: zveřejnění poznámky

- **Aktér:** přihlášený uživatel

1. Aktér otevře detail vybrané poznámky (UC5.2).
2. Zvolí akci „Zveřejnit“.
3. Aplikace nastaví poznámku jako veřejnou (tím získá veřejnou URL).
4. Nepřihlášení uživatelé si ji budou moci zobrazit (UC11).

UC10.1: zveřejnění souboru (obrázek, PDF soubor)

- **Aktér:** přihlášený uživatel

1. Aktér zobrazí seznam nahraných souborů.
2. U vybraného souboru klikne na „Zveřejnit“.
3. Aplikace zpřístupní soubor na veřejné adrese a zároveň budou soubory dostupné přes odkazy ve zveřejněných poznámkách.

UC10.2: odebrání veřejného statusu poznámce

- **Aktér:** přihlášený uživatel

1. Aktér v detailu zveřejněné poznámky zvolí „Zrušit zveřejnění“.
2. Aplikace poznámce odebere status „veřejná“.
3. Nepřihlášení uživatelé již nemají k poznámce přístup.

UC11: zobrazení zveřejněné poznámky

- **Aktér:** nepřihlášený uživatel

1. Nepřihlášený uživatel obdrží nebo nalezne odkaz na zveřejněnou poznámku (např. odkazem na webu).
2. Klikne na odkaz.
3. Aplikace zobrazí obsah poznámky a případných veřejných příloh (obrázky, PDF).
4. Nepřihlášený uživatel nemůže poznámku nijak upravovat či mazat. Pokud se v poznámce vyskytují odkazy na neveřejné poznámky či soubory, nemá k nim nepřihlášený uživatel přístup.

UC12: export poznámky

- **Aktér:** přihlášený uživatel

Aktér má možnost poznámky exportovat ve zvoleném formátu (Markdown, HTML atd.). Na detailu poznámky zvolí možnost „Exportovat“ a z nabídky vybere preferovaný formát. Aplikace pak vygeneruje soubor ve zvoleném formátu a nabídne jej ke stažení.

UC13: vyhledání poznámek pomocí fulltextového vyhledávání

- **Aktér:** přihlášený uživatel

Aktér zadá klíčové slovo či frázi a aplikace vyhledá odpovídající poznámky (fulltextově v názvu i obsahu poznámky).

1. Aktér otevře vyhledávací pole v horní části aplikace.
2. Zadá klíčová slova nebo frázi.
3. Aplikace provede fulltextové vyhledávání v názvech i obsahu poznámek.
4. Aplikace zobrazí seznam relevantních poznámek setříděných podle relevance.
5. Aktér vybere poznámku z výsledku a zobrazí její detail (UC5.2).

2.3.3 Incidenční matice funkčních požadavků a případů užití

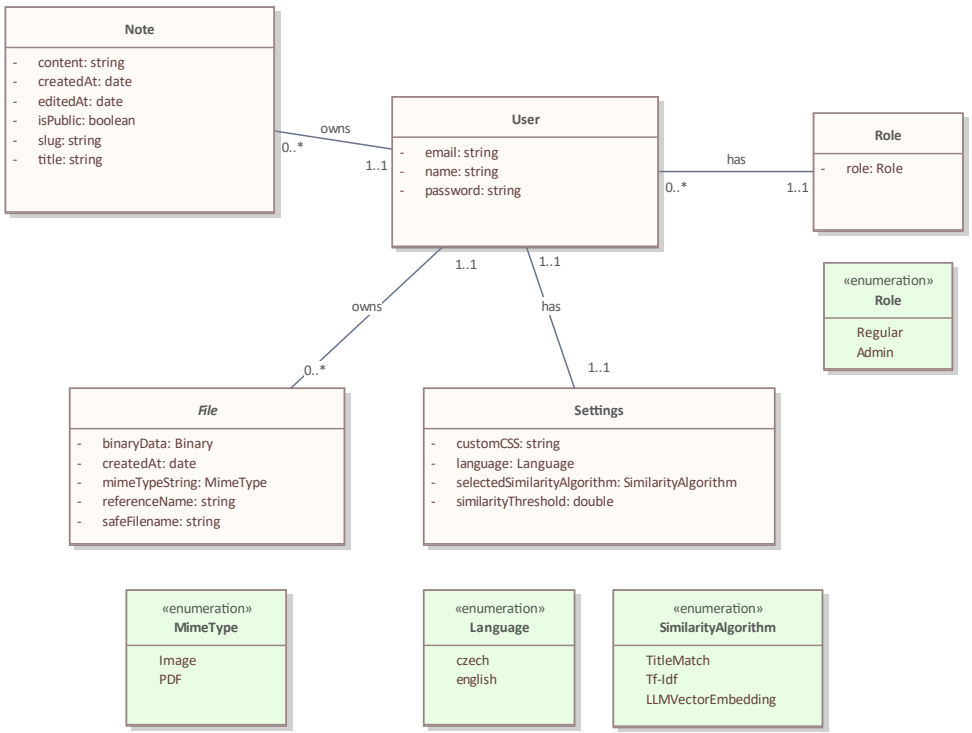
Tabulka 2.1 je vytvořena pro kontrolu, jestli jsou všechny specifikované případy užití pokryté funkčními požadavky. Z tabulky vyplývá, že funkční požadavky pokrývají všechny případy užití.

Tabulka 2.1 Incidenční matice pokrytí případů užití pomocí funkčních požadavků

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
UC1	✓	-	-	-	-	-	-	-	-	-	-
UC2	✓	-	-	-	-	-	-	-	-	-	-
UC3	✓	-	-	-	-	-	-	-	-	-	-
UC4	✓	-	-	-	-	-	-	-	-	-	-
UC5	-	✓	-	✓	✓	-	-	-	-	-	-
UC6	-	-	✓	-	-	-	-	-	-	-	-
UC7	-	✓	-	-	✓	✓	-	-	-	-	-
UC8	-	-	-	-	-	-	✓	-	-	-	-
UC9	-	-	-	-	-	-	-	✓	-	-	-
UC10	-	-	-	-	-	-	-	-	✓	-	-
UC11	✓	-	-	✓	-	-	-	-	✓	-	-
UC12	-	-	-	-	-	-	-	-	-	✓	-
UC13	-	-	-	-	-	-	-	-	-	-	✓

2.4 Doménový model

Doménový model na obrázku 2.1 znázorňuje rozdělení analyzovaného problému do jednotlivých tříd a zobrazuje jejich vzájemné vztahy. Tento doménový model doplňuje funkční požadavky a případy užití aplikace ThinkLink, definuje strukturu a vazby mezi jednotlivými entitami.



Obrázek 2.1 Doménový model aplikace ThinkLink, vlastní tvorba v [41]

Kapitola 3
Návrh

Tato kapitola se zabývá návrhem webové aplikace ThinkLink. Na základě stanovených funkčních a nefunkčních požadavků byla stanovena architektura aplikace a byly vybrány vhodné technologie pro vývoj, správu dat a implementaci aplikace s ohledem na vzájemnou kompatibilitu a udržitelnost.

Kapitola se také zabývá návrhem tří metod automatického hledání souvisejících poznámek, které budou v prototypu aplikace implementovány a následně otestovány. V kapitole jsou jednotlivé metody popsány a jsou zmíněny jejich výhody a nevýhody.

3.1 Architektura aplikace

Architektura systému představuje návrhová rozhodnutí, která se týkají se celkové struktury a chování systému. Cílem výběru architektury je zajištění udržitelnosti, rozšiřitelnosti a srozumitelnosti aplikace, proto je potřeba aplikaci rozdělit na menší, vzájemně spolupracující části, kde každá z těchto částí bude mít jasně vymezenou zodpovědnost. [42, 43]

V této sekci je popsána architektura webové aplikace ze dvou perspektiv:

- **Fyzická architektura**: definuje jaká část webové aplikace bude spuštěna na webovém serveru a jaká část bude spuštěna u klienta.
- **Logická architektura**: definuje rozdělení kódu a balíčků do logických komponent, modulů a jmenných prostorů.

3.1.1 Fyzická architektura

Navrhovaná aplikace bude webovou aplikací a bude mít architekturu klient-server. Tato architektura je rozdělena na dvě hlavní části, které spolu komunikují přes protokol HTTP. Jedna část této dvouvrstvé architektury je spuštěna

na vzdáleném serveru a přijímá jednotlivé požadavky od klientské části, zpracovává je a poskytuje na ně odpovědi. Klientská část aplikace pak běží na koncových zařízeních jednotlivých uživatelů, kde interaguje s uživateli, posílá požadavky na webový server a interpretuje odpovědi ze serveru. [44]

Je potřeba aplikaci rozdělit na tyto dvě části a definovat, jaké zodpovědnosti bude mít každá z nich. V praxi je možné ještě oddělit databázi a provozovat ji na samostatném uzlu, čímž se architektura stává třívrstvou (prezentace – webový server – databázový server) pro větší separaci odpovědnosti, abstrakce a snížení závislostí [42]. Pro potřeby tohoto projektu se zůstane u dvouvrstvé architektury.

Rozdělení dvouvrstvých architektur dle Sommerwilla [42]:

- **Thin-client / smart-server**, což je tradiční či monolitická aplikace, kde se veškerá logika zpracovává na straně serveru. Klientská část zde slouží k zobrazení uživatelského rozhraní pomocí HTML stránek generovaných serverem a k interakci s uživatelem. Mezi hlavní výhody tohoto přístupu patří nižší nároky na výkon klientského zařízení a jednodušší možnosti testování aplikace.
- **Thick-client / dumb-server**, což je modernější přístup (SPA – *Single Page Application*, česky *jednostránková aplikace*), kde je část logiky přesunuta na klientskou část a proto je třeba plnohodnotně implementovat obě části aplikace. Serverová část negeneruje hotovou HTML stránku pro klienta, ale jen posílá data, která jsou k vytvoření stránky potřeba a o samotné vytvoření se postará klientská strana aplikace. Výhodou je menší zátěž serverové části a zlepšení uživatelského zážitku při používání aplikace.

Aplikace ThinkLink bude implementována jako tradiční webová aplikace (tedy thin-client / smart-server). Uživateli se ze serveru poskytne plně vygenerovaná HTML stránka, se kterou může interagovat. Díky tomu bude vývoj a testování aplikace jednodušší a zařízení na straně klienta nebudou zatěžována velkými nároky na Javascript. Na druhou stranu bude v budoucnu náročnější aplikaci škálovat podle nárůstu počtu klientů a nároků na výkon.

V budoucím vývoji je možné přidat API vrstvu a nahradit prezentační vrstvu samostatnou SPA nebo mobilním klientem. To nabídne větší flexibilitu a umožní snadnou integraci s dalšími službami.

3.1.2 Logická architektura

Aplikace ThinkLink bude tedy vyvíjena jako tradiční webová aplikace, ale pro lepší udržitelnost, rozšiřitelnost a srozumitelnost návrhu bude aplikace rozdělena do více menších, jasně oddělených komponent, které spolu budou spolupracovat. Pro rozdělení aplikace bude použit zavedený návrhový vzor

MVC (Model-View-Controller), který rozděluje aplikaci do tří hlavních komponent [42]:

- Model (*model*) – definuje strukturu dat v aplikaci a obsahuje hlavní logiku aplikace.
- View (*pohled*) – definuje uživatelské rozhraní a to, jak mají být data zobrazena.
- Controller (*kontrolér*) – slouží jako moderátor, který dostává požadavky od uživatele, zpracovává je a jednotlivé úkony deleguje na jednotlivé služby modelu.

Díky tomuto rozdělení logiky, struktury dat a zobrazení bude aplikace ThinkLink přehlednější a lépe udržitelná. Vývoj aplikace bude jednodušší, protože bude možné pracovat na jednotlivých komponentách aplikace bez rizika ovlivnění jiných komponent. To pomáhá vývojářům vyvíjet efektivněji, snáze testovat jednotlivé součásti systému a celkově lépe organizovat práci. V budoucnu tedy nebude problém například vyměnit vrstvu *View* za jinou, aniž by byla ovlivněna struktura dat či logika aplikace.

Popularitu zvolené architektury také dokládá to, že na základě MVC architektury vznikly další návrhové vzory, jako MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter) a MVW (Model-View-Whatever). [45]

3.2 Použité technologie

Pro vývoj aplikace ThinkLink, v souladu s povahou systému, provedenou analýzou funkčních a nefunkčních požadavků a zvolenou architekturou je potřeba vybrat vhodné technologie, které budou hrát klíčovou roli v implementaci a realizaci celé aplikace. Patří sem především volba programovacího jazyka, odpovídajícího frameworku a dalších technologií a nástrojů. Důležité je také vybrat a definovat způsob uložení dat, protože bude potřeba bezpečně a efektivně ukládat poznámky a další soubory pro jednotlivé uživatele.

Zvolená řešení budou splňovat požadavky na webovou aplikaci, vzájemně spolupracovat a také mít dlouhodobou podporu do budoucna. Současně je důležité zohlednit znalosti vývojového týmu a dostupnost technologií na trhu, aby byla zajištěna snadná podpora a udržitelnost aplikace.

3.2.1 Programovací jazyk PHP

Na základě nefunkčního požadavku *N3: Technologie* byl hlavním implementačním jazykem zvolen jazyk PHP. Jedná se o populární open-source skriptovací jazyk pro všeobecné použití, který je vhodný zejména pro vývoj webových stránek a aplikací. PHP je zkratka pro *PHP: Hypertext Preprocessor*. [46]

V rámci navržené architektury thin-client / smart-server je PHP, které vykonává kód na serveru, dobrou volbou společně s jeho širokou podporou na všech hlavních operačních systémech i webových serverech, což minimalizuje náklady na infrastrukturu a usnadňuje nasazení aplikace.

Klíčovou výhodou PHP je snadná integrace s dalšími technologiemi, ať už se jedná o různé komunikační protokoly, podporu API a knihovny, které umožňují napojení na řadu externích služeb. PHP má také širokou podporu pro různé databázové stroje, a to umožňuje vybrat takový databázový stroj, který bude nejlépe vyhovovat potřebám aplikace ThinkLink. [47]

Jazyk PHP je tedy spolehlivou a stabilní volbou pro vývoj aplikace ThinkLink. Nabízí také velkou nabídku knihoven, pokročilých funkcí a frameworků (např. Symfony či Laravel), které jeho možnosti dále rozšiřují. PHP je základem mnoha úspěšných projektů (Wikipedia, WordPress, Facebook apod.) [48]. Jedná se tedy o spolehlivé řešení i v rozsáhlých produkčních instalacích. V projektu bude použita verze PHP 8.3.

V současnosti existuje mnoho alternativ jazyka PHP pro vývoj webových aplikací. Patří mezi ně oblíbený Node.js, což je výkonná open-source Javascriptová platforma pro vývoj webových aplikací jak na straně serveru, tak i klienta. Dále je možné využít jazyk Python společně s frameworky Flask nebo Django. Pro vývojáře zkušené v jazyce Java by mohlo být vhodné aplikaci vyvinout v Javě ve frameworku Spring Boot. Poslední zmíněnou a zajímavou alternativou může být jazyk C# v prostředí .NET (např. framework Blazor). [49]

3.2.2 Framework Symfony

Spolu s volbou jazyka PHP byl pro efektivnější, bezpečnější a strukturovanější vývoj zvolen framework Symfony. Vývoj v rámci zavedeného a rozšířeného frameworku je logickým krokem při vývoji moderní webové aplikace. Použitím frameworku nad jazykem PHP je možné využít již vytvořené a odladěné komponenty pro rychlejší, levnější a udržitelnější vývoj.[1]

Symfony nabízí sadu znovupoužitelných PHP komponent, které se dají využít jak samostatně, tak i v rámci celého frameworku. Framework Symfony je open-source, je aktivně spravován a vyvíjí se také pomocí široké a aktivní komunity. [50]

Pomocí Symfony [50] je možné jednoduše implementovat:

- autentizaci a autorizaci uživatelů,
- zpracování formulářů včetně validace,
- napojení na databázi přes Doctrine ORM[2],
- napojení na šablonovací systém Twig[3] a

[1] Záložka *Why use a framework* z [50]
[2] Sekce *3.3 Technologie pro ukládání a správu dat*
[3] Sekce *3.2.3 Twig*

- dalších funkcí (notifikace, logování, lokalizace atd.).

Pro vývoj aplikace ThinkLink je důležitá podpora architektonického vzoru MVC, který Symfony nativně podporuje. Dalším významným aspektem je podpora LTS (*Long-Term Support*) verzí, které garantují dlouholetou podporu, což je ideální pro dlouhodobý a stabilní vývoj aplikace ThinkLink.[4]

V neposlední řadě také Symfony podporuje PHP standardy (PSR – PHP Standards Recommendations). Díky tomu je zajištěna kompatibilita knihoven napříč celým ekosystémem PHP, což opět usnadňuje vývoj a údržbu aplikace. [51]

Jako alternativu frameworku Symfony je považován hojně používaný framework Laravel. Podle článku [52] je Laravel jednodušší a vhodnější pro menší projekty, zatímco Symfony je více modulárnější a vhodnější pro vývoj komplexních a dlouhodobějších projektů. V projektu bude použit framework Symfony verze 7.1.

3.2.3 Twig

Twig je moderní šablonovací systém pro jazyk PHP vybraný pro prezentační vrstvu aplikace ThinkLink především díky jeho vynikající kompatibilitě s frameworkem Symfony. Mezi jeho hlavní výhody patří:

- **jednoduchost a srozumitelnost** – šablony jsou přehledné, snadno se udržují a využívají sadu jasných direktiv a filtrů,
- **rychlost** – Twig dává důraz na optimalizaci pro rychlé generování jednotlivých stránek,
- **bezpečnost** – automaticky ošetřuje obsah proměnných, což je důležitá praktika při zobrazování uživatelských vstupů (např. poznámky uživatelů) a
- **možnosti rozšíření** – díky modulárnímu návrhu je možné Twig přizpůsobit či ho doplnit o další funkce dle potřeb konkrétního projektu.

V rámci projektu by bylo možné využít čisté PHP jako šablonovací systém, ale Twig ve spolupráci s frameworkem přináší spoustu výhod v podobě hotových nástrojů a zjednodušení, které usnadní vývoj celé aplikace. [53]

3.2.4 Bootstrap

Pro zlepšení uživatelského zážitku z používání aplikace je navrženo použití CSS frameworku, který umožňuje rychle navrhnout a implementovat styly webové aplikace ThinkLink. CSS frameworky nabízí sadu nástrojů, šablon a hotových tříd pro různé části webu, což usnadňuje jeho tvorbu a údržbu. Výhodou těchto

[4] Záložka *Six good reasons to use Symfony* z [50]

frameworků také je, že nabízejí řadu komponent, které se běžně ve webových aplikacích vyskytují, a tedy není potřeba je vytvářet zvlášť. Zároveň se použitím CSS frameworku nevylučuje možnost definovat vlastní CSS styly, které mohou definované styly měnit či doplňovat a vytvořit tím unikátní styl pro celou webovou aplikaci. [54]

Další předností CSS frameworků je zajištění responzivního návrhu, kdy se komponenty vhodně zobrazují na různých zařízeních, což je pro webovou aplikaci a její použitelnost nezbytné [54]. Správným použitím CSS frameworku bude tedy docíleno naplnění nefunkčního požadavku *N2: Responzitivita aplikace*.

Pro vývoj uživatelského prostředí aplikace ThinkLink byl zvolen CSS framework *Bootstrap 5* především pro jeho snadnou integraci do Symfony projektu [55], velkou popularitu a nabídce komponent, které zajistí dobrou uživatelskou zkušenost. [56]

3.2.5 Docker

Docker je open-source platforma pro vývoj, distribuci a provoz softwarových aplikací. Umožňuje oddělit aplikace od infrastruktury a zajistit stejné a replikovatelné vývojové prostředí nezávisle na rozdílných konfiguracích různých zařízení. Tímto přístupem je možné zásadně urychlit vývoj a připravit aplikaci na budoucí nasazení.

Klíčovým prvkem Dockeru jsou kontejnery, což jsou samostatné spustitelné jednotky, které obsahují potřebné závislosti, knihovny a konfigurační soubory nezbytné pro běh aplikací. Tyto kontejnery je možné mezi sebou propojovat a během vývoje tak simulovat prostředí podobné reálnému nasazení na vzdálených serverech. Tím, že kontejnery oddělují běžící aplikace od zbytku systému, tak v případě chyby v aplikaci se minimalizuje dopad na celý systém, na kterém Docker běží.

Docker byl vybrán jako součást vývoje aplikace ThinkLink, protože umožňuje rychlý vývoj se stejným a replikovatelným prostředím na různých zařízeních. Zvolením této technologie bude splněn nefunkční požadavek *N5: Příprava na nasazení a jednoduchou spolupráci*. Tím umožňuje jednodušší spolupráci vývojářů a snazší nasazení aplikace do produkčního prostředí. Jedná se také o široce známou technologii na trhu, která umožní dlouhodobý a udržitelný vývoj aplikace a podporuje všechny vybrané technologie v rámci projektu ThinkLink, což zajišťuje jejich bezproblémovou integraci a dlouhodobou udržitelnost projektu. [57, 58]

3.3 Technologie pro ukládání a správu dat

Ukládání a správa dat je klíčová část aplikace ThinkLink. Z provedené analýzy vyplývá, že je potřeba ukládat a spravovat jednotlivé poznámky a soubory pro různé uživatele, přičemž pro automatické hledání souvislostí mezi jednotlivými

poznámkami je také potřeba pracovat s jejich vektorovými reprezentacemi. Tyto požadavky vychází zejména z funkčních požadavků:

- *F1: přihlášení a správa uživatelů*,
- *F2: správa poznámek*,
- *F3: import a nahrávání poznámek a souborů*,
- *F4: zobrazování obrázků a PDF*,
- *F6: automatické propojování souvisejících poznámek na základě obsahové a sémantické podobnosti.*

V zásadě se jedná o tři druhy ukládaných dat: informace, soubory a vektory. U každého typu je potřeba zvolit jiný přístup ke správě a ukládání:

- **Soubory** jsou ukládány na disk serveru (sekce *3.3.1 Ukládání souborů*).
- **Informace** jsou ukládány do databáze PostgreSQL (sekce *3.3.2 PostgreSQL*).
- **Vektory** jsou ukládány a spravovány rozšířením *pgvector* databáze PostgreSQL (sekce *3.3.2.1 Rozšíření pgvector*).

Integrace databáze se zbytkem aplikace ne nachází v sekci *3.3.2.2 Doctrine ORM*. Data a soubory budou uloženy na vzdáleném serveru, aby k nim měl oprávněný uživatel přístup z libovolného zařízení. Vzhledem k tomu, že byla vybrána architektura thin-client / smart-server, tak umístění dat a souborů na serverové části dává smysl z hlediska omezení objemu přenesených dat, především při fázi zpracování poznámek pro jejich automatického hledání souvisejících poznámek.

3.3.1 Ukládání souborů

Aplikace umožní uživatelům nahrávat multimediální soubory (např. obrázky, PDF), které pak mohou být referencovány v poznámkách a zobrazovány přímo v prostředí aplikace. Tyto soubory se uloží na disk serveru do neveřejné složky, aby k nim neměli přístup neautorizovaní uživatelé.

Pro samotné ukládání bude použita PHP knihovna *Flysystem* [59], nabízející rozhraní a abstrakci nad různými typy úložných systémů. Díky integraci této knihovny nebude v budoucnu problém jednoduše změnit typ úložného systému z lokálního disku serveru na např. cloudová úložiště *AWS S3*, *Azure Blob Storage* či *Google Cloud Storage*. Možnost rychlé změny úložného systému je klíčové pro budoucí rozvoj aplikace.

3.3.2 PostgreSQL

Pro ukládání strukturovaných informací o poznámkách, uživatelích a dalších entitách byla zvolena relační databáze PostgreSQL [60]. Jedná se o open-source objektově-relační databázový systém se širokou a aktivní komunitou, která jej aktivně rozvíjí.

Vybraný framework Symfony podporuje práci s různými databázemi pomocí knihovny Doctrine. Použitím této knihovny lze pomocí konfigurace jednoduše změnit databázový stroj za například MySQL, SQLite či PostgreSQL[5], což projektu poskytuje flexibilitu. [61]

V PostgreSQL se budou ukládat veškerá aplikační data. Pro uložené soubory na disku budou vytvořeny záznamy v databázi obsahující metadata a informace ohledně uložení souboru.

Databáze PostgreSQL byla vybrána i pro její velkou podporu a široký výběr rozšíření, které rozšiřují její základní funkčnost. Jedním z těchto rozšíření je *pgvector*, který v rámci PostgreSQL zajišťuje ukládání a práci s vektory a vzdálenostmi mezi nimi, což je zásadní funkce pro aplikaci ThinkLink a její práci s vektorovými reprezentacemi jednotlivých poznámek.

3.3.2.1 Rozšíření pgvector

Rozšíření *pgvector* je open-source rozšíření databáze PostgreSQL umožňující ukládání vektorů společně s ostatními relačními daty a práci s nimi. Umožňuje provádět přesné či přibližné hledání nejbližších sousedů a výpočet různých metrik (Euklidovská[6], Kosinová[7], Manhattanská, Hammingova či Jaccardova). [62]

Výhodou je, že vedle speciálních funkcí pro práci s vektory splňuje ACID transakce, point-in-time recovery, podporu JOINů a další funkce relační databáze PostgreSQL. Jedná se o spojení relační databáze s vektorovými funkcemi, což aplikaci ThinkLink umožňuje ukládat vektory společně s ostatními daty do jedné databáze a eliminovat potřebu použití samostatné vektorové databáze. Toto řešení snižuje složitost a nároky na provoz. [62]

3.3.2.2 Doctrine ORM

Doctrine je sada PHP knihoven pro práci s databází. Tyto nástroje podporují jak relační databáze jako je MySQL či PostgreSQL, tak i NoSQL databáze jako například MongoDB. Tato široká podpora z Doctrine dělá užitečný nástroj pro spolupráci databáze a zbytku aplikace. [61]

Klíčovou částí knihovny Doctrine je ORM, neboli objektově relační mapování, které slouží k propojení objektově orientované aplikace s relační databází. Doctrine ORM je tedy vrstva, která mapuje jednotlivé objekty aplikace na relační databázi. Hlavním cílem je zjednodušit vývojářům práci s databází

[5] Záložka *Databases and the Doctrine ORM* z [50]
[6] Sekce *1.4.5.2 Euklidovská vzdálenost*
[7] Sekce *1.4.5.1 Kosinová podobnost*

a odstínit je od psaní a údržby SQL dotazů. Knihovna implementuje návrhový vzor *Data Mapper*, díky kterému vývojář pracuje pouze s entitami v kódu a knihovna se stará o jejich ukládání a načítání z databáze. [61]

Knihovna Doctrine je přímo podporovaná a využívaná v Symfony, ale lze ji integrovat i do dalších aplikací a frameworků. Jedná se tedy o vhodný nástroj pro vývoj aplikace ThinkLink.[8]

Velkou výhodou využití Doctrine ORM je podpora mnoha různých SQL databází (např. MySQL, PostgreSQL, MariaDB, SQLite aj.), které je možné pomocí konfiguračního nastavení měnit. Případná změna databázového stroje proto obvykle nevyžaduje zásah do zbytku kódu, pouze do konfiguračních souborů. [61]

Doctrine umožňuje generovat a spravovat tzv. migrace, což jsou třídy, které zachytávají změny v databázovém schématu. Doctrine umí generovat migrace na základě změn v entitách a tím umožňuje verzování a konzistentní údržbu databáze v průběhu vývoje. Pomocí migrací si také každý vývojář může sestavit stejné databázové schéma, což efektivně urychluje vývoj a eliminuje různé nekonzistence. V kombinaci s nástrojem Docker, který zajistí stejné vývojové prostředí se jedná o nápomocný nástroj co se týče spolupráce více vývojářů na projektu. [61]

3.4 Metody hledání souvisejících poznámek

Hlavní částí aplikace je implementace metod, které budou pro uživatele zajišťovat automatické hledání souvisejících poznámek. Bude implementována jedna naivní technika a alespoň jedna pokročilá technika pro provázání poznámek uživatele. Zároveň je třeba klást důraz na to, aby byla aplikace ThinkLink do budoucna rozšiřitelná pro další metody hledání souvisejících poznámek. V rámci návrhu jsou vybrány a implementovány tři různé způsoby automatického hledání souvisejících poznámek, které se liší způsobem provedení, složitostí a přístupem k problematice detekce souvisejících poznámek. V této části budou navrženy a představeny tři různé způsoby s krátkým popisem jejich výhod a nevýhod. Implementační podrobnosti budou popsány v sekci *4.6 Implementace metod pro automatické hledání souvisejících poznámek*. Uživatelské porovnání těchto metod se nachází v kapitole *5 Testování*.

3.4.1 Propojování podle výskytu názvu poznámky v obsahu jiných poznámek

Tato metoda je velmi naivní, ale v určitých případech může být přínosná a zajímavá pro srovnání s dalšími metodami. Funguje na jednoduchém principu hledání slov či slovních spojení z názvu jedné poznámky v obsahu jiných poznámek.

[8] Záložka *Databases and the Doctrine ORM* z [50]

Funkčnost metody je popsána následovně. Existuje poznámka A. Pokud existuje jiná poznámka B, která ve svém obsahu obsahuje slovo či slovní spojení z názvu poznámky A, tak se vyhodnotí jako související. Čím je slovo či slovní spojení v poznámce B častější, tím je poznámka relevantnější k poznámce A.

Výhody a nevýhody této metody jsou uvedeny v tabulce 3.1.

Tabulka 3.1 Výhody a nevýhody propojování poznámek podle výskytu názvu poznámky v obsahu jiných poznámek

Výhody	Nevýhody
Jednoduchost a snadná implementace.	Nezohledňuje sémantickou blízkost mezi poznámkami.
Funguje na základě běžného textového hledání bez potřeby speciálních modelů.	Nerozpozná synonyma, skloňování ani jazykové varianty názvů poznámek.
Lze efektivně využít možnosti databáze PostgreSQL a její fulltextové vyhledávání.	Nepodchytí koncepty vyjádřené jiným způsobem (např. NLP vs. zpracování přirozeného jazyka).
Dobrá dostupnost technologie a podpora ve zvoleném databázovém systému.	Riziko falešných shod u vícevýznamových slov (např. „Apple“ jako firma a „apple“ jako ovoce).

3.4.2 Propojování pomocí vektorových reprezentací a TF-IDF

Tato metoda využívá vektorové reprezentace poznámek a TF-IDF. Princip této metody je vysvětlen v sekci *1.4.4.2 TF-IDF*.

V rámci této metody se, narozdíl od prvního přístupu, už porovnávají jednotlivé poznámky na základě názvu i obsahu obou z nich. Nejdříve se vytvoří vektorové reprezentace pro každou z poznámek a pak se pomocí kosinové podobnosti[9] vypočítá vzájemný úhel mezi jednotlivými vektory a jednotlivé úhly pak reprezentují podobnost odpovídajících poznámek.

Výhodou tohoto přístupu je snadná implementace a možnost lokální tvorby vektorů. Zvolený jazyk PHP je dostatečně vybaven pro implementaci výpočtu částí TF-IDF a tvorby jednotlivých vektorů. Vektory budou ukládány do PostgreSQL databáze, která je umí efektivně ukládat a zpracovávat pomocí přidaného rozšíření *pgvector*, které rovnou nabízí hledání nejbližších vektorů pomocí kosinové podobnosti.

[9] Sekce *1.4.5.1 Kosinová podobnost*

Díky použití TF-IDF je k dispozici lepší přesnost pro klasický text, kdy výpočet TF-IDF potlačuje příliš častá slova (stop slova) a upřednostňuje slova (neboli termy), která odlišují text od ostatních textů, což zlepšuje přesnost oproti pouhému hledání počtu výskytů slov z názvu poznámky. V tabulce 3.2 jsou uvedeny výhody a nevýhody hledání poznámek pomocí TF-IDF.

Tabulka 3.2 Výhody a nevýhody propojování poznámek pomocí vektorových reprezentací a TF-IDF

Výhody	Nevýhody
Lepší přesnost oproti prostému hledání výskytů slov.	Nezachycuje sémantiku, ignoruje význam a pořadí slov.
Potlačuje častá slova (např. stop slova), která nejsou informačně přínosná.	Vyžaduje sledování globální frekvence termů napříč všemi dokumenty.
Upřednostňuje výrazy, které text odlišují od ostatních poznámek.	Náročnější na paměť a výpočetní výkon při větším množství poznámek.
Dobrá dostupnost technologie a podpora ve zvoleném databázovém systému.	Riziko falešných shod u vícevýznamových slov (např. „Apple“ jako firma vs. „apple“ jako ovoce).

3.4.3 Propojování pomocí vektorových reprezentací vygenerovaných LLM

Nejpokročilejší metoda automatického hledání souvisejících poznámek vychází z vektorových reprezentací, které jsou generované pomocí existujících LLM[10]. Každá poznámka se pomocí API třetí strany převede na vektorovou reprezentaci, který se uloží do databáze. Následně je možné mezi jednotlivými vektory počítat jejich vzájemnou polohu. K počítání podobností mezi vektory bude pro lepší srovnání zvolen stejný přístup jako u TF-IDF, tj. kosinová podobnost. Tabulka 3.3 přehledně shrnuje výhody a nevýhody tohoto přístupu.

V současnosti existuje více různých poskytovatelů API, které generuje vektorové reprezentace na základě vstupního textu [63]. Pro propojování poznámek pomocí vektorových reprezentací byly pro porovnání zvoleny dva různí poskytovatelé *embedding modelů* (tj. speciálních jazykových modelů pro generování vektorových reprezentací):

OpenAI [37] nabízí několik modelů, které umí generovat vektorové reprezentace pro vstupní text:

[10] Sekce *1.4.4.4 Předtrénované modely*

Tabulka 3.3 Výhody a nevýhody propojování poznámek pomocí vektorových reprezentací vygenerovaných LLM

Výhody	Nevýhody
Zachycuje sémantiku poznámek. Rozpoznává významová a kontextová spojení.	Závislost na externí (hostované) službě a nutnost síťového připojení.
Odolnost vůči synonymům. Poznámky s různou formulací mohou být relevantní.	Možné zpoplatnění API volání pro výpočet vektorových reprezentací.
Natrénované modely na obrovském množství různých textových dat.	Odesílání dat třetí straně – potřeba řešit bezpečnost, důvěryhodnost a soulad s GDPR.
Podpora více jazyků.	

- *text-embedding-3-small*,
- *text-embedding-3-large*,
- *text-embedding-ada-002*.

Na základě porovnání jednotlivých modelů byl zvolen model *text-embedding-3-small* podle nejlepšího poměru výkonu modelu (podle žebříčku MTEB [63]) a ceny.

Google [38] také nabízí v rámci *Gemini API* tři modely pro generování vektorových reprezentací:

- *gemini-embedding-exp-03-07*,
- *text-embedding-004*,
- *embedding-001*.

Za základě porovnání jednotlivých modelů byl zvolen model *text-embedding-004*, jelikož se jedná o nejnovější stabilní model. Uvedený *gemini-embedding-exp-03-07* je sice novější a výkonnější (podle žebříčku MTEB [63]), ale společnost Google zmiňuje, že se stále jedná o experimentální model a zatím nedoporučuje jeho použití v produkci [38].

Kapitola 4
Implementace

Tato kapitola popisuje průběh implementace prototypu webové aplikace ThinkLink. Nejdříve vymezí, které funkční a nefunkční požadavky jsou v prototypu skutečně implementovány, dále kapitola popíše strukturu a důležité části projektu.

Kapitola také popisuje způsob implementace funkcí z referenční aplikace Obsidian, která byla analyzována v kapitole *2.1 Analýza aplikace Obsidian*.

Hlavní část kapitoly se věnuje rozšiřitelnosti aplikace a implementačním detailům vybraných metod pro automatické hledání souvisejících poznámek.

4.1 Implementované části

Tato kapitola popisuje, které funkční požadavky (ze sekce *2.2.1 Funkční požadavky*) byly v rámci prototypu realizovány. Vývoj prototypu se zaměřil především na implementaci metod pro automatické vyhledávání souvisejících poznámek, jejich rozšiřitelnost a možnost jejich vzájemného porovnání.

Z definovaných funkčních požadavků byly splněny všechny, které měly přiřazenou vysokou prioritu:

- **F1:** Přihlášení a správa uživatelů,
- **F2:** Správa poznámek,
- **F3:** Import a nahrávání poznámek a souborů,
- **F4:** Zobrazování obrázků a PDF,
- **F5:** Propojování poznámek pomocí odkazů,
- **F6:** Automatické propojování poznámek na základě obsahové a sémantické podobnosti.

Ostatní funkční požadavky, označené střední nebo nízkou prioritou, poslouží jako podklad pro další vývoj aplikace.

4.2 Struktura projektu

Struktura projektu byla navržena v souladu s doporučenou strukturou Symfony projektů[1]. Architektura aplikace vychází z modelu MVC[2], který je v projektové struktuře reflektován a jednotlivé vrstvy jsou v rámci kódu odděleny.

Základní struktura projektu byla vytvořena pomocí příkazu dostupného v nástroji Symfony CLI[3]. Příkaz v ukázce 4.1 inicializuje tradiční webovou aplikaci včetně přednastavené adresářové struktury a základních komponent potřebných pro vývoj.

Výpis kódu 4.1 Vytvoření nové webové aplikace pomocí Symfony CLI

```
symfony new --webapp thinklink
```

4.2.1 Popis struktury projektu

Na následujícím seznamu je zachycena adresářová struktura projektu ThinkLink.

- **assets/** Obsahuje soubory pro prezentační vrstvu aplikace, jako jsou JavaScript, CSS styly a obrázky.
- **bin/** Obsahuje spustitelné skripty. Nejvýznamnější z nich je `console`, který slouží jako hlavní vstupní bod pro spouštění příkazů Symfony.
- **config/** Obsahuje konfigurační soubory aplikace. Zde jsou definovány například trasy, služby a parametry jednotlivých komponent. Klíčovým souborem je `services.yaml`, který obsahuje definice služeb a jejich konfigurační parametry, včetně strategií pro práci s různými datovými typy a vyhledávání souvisejících poznámek[4].
- **migrations/** Obsahuje migrační soubory sloužící k verzování změn v databázové struktuře. Využívá se knihovna Doctrine[5].
- **public/** Kořenový adresář pro webový server. Obsahuje soubor `index.php`, který funguje jako vstupní bod aplikace.
- **src/** Hlavní zdrojový kód aplikace. Obsahuje následující podsložky:

[1] Záložka *Best Practices* v [50]: `https://symfony.com/doc/current/best_practices.html`
[2] Kapitola *3.1.2 Logická architektura*
[3] Záložka *Download Symfony* v [50]: `https://symfony.com/download`
[4] Kapitola *4.5 Zajištění rozšiřitelnosti pomocí vzoru Strategie*
[5] Kapitola *3.3.2.2 Doctrine ORM*

- `Controller/` – kontroléry v komponentě *Controller* v MVC[6],
- `DataFixtures/` – Doctrine Fixtures[7] pro vytvoření výchozích dat,
- `Entity/` – definice databázových entit,
- `Form/` – definice formulářů, objektů pro přenos dat a validátorů,
- `Message/` – soubory pro asynchronní zpracování pomocí Symfony Messenger[8],
- `Repository/` – třídy pro práci s databází,
- `Service/` – aplikační logika (služby), včetně:
  - `FileHandler/` – strategie pro práci s různými typy souborů,
  - `RelevantNotes/` – strategie metod vyhledávání souvisejících poznámek,
- `Twig/` – vlastní Twig[9] filtry pro konverzi *Markdown* odkazů do HTML,
- `Voter/` – logika přístupových práv uživatelů[10]

- **templates/** Obsahuje Twig šablony pro vykreslování HTML. Reprezentuje komponentu *View* návrhového vzoru MVC.
- **tests/** Obsahuje testy aplikace psané pomocí knihovny PHPUnit[11].
- **translations/** Místo pro lokalizační soubory s překlady textů aplikace do různých jazyků.
- **var/** Složka pro ukládání cache a logů aplikace. Zároveň se zde pomocí knihovny Flysystem ukládají nahrané soubory jednotlivých uživatelů[12].
- **vendor/** Obsahuje veškeré závislosti nainstalované pomocí nástroje Composer.

4.3 Důležité části projektu

Tato sekce se zabývá popisem jednotlivých částí projektu, které jsou součástí vývoje prototypu aplikace ThinkLink. Tyto části zahrnují nástroje pro správu balíčků, sjednocení vývojářského prostředí na různých zařízeních či správu assetů potřebných ke správnému běhu aplikace. U každé části je zmíněno, jakým způsobem přispívá k vývoji aplikace.

[6] Kapitola *3.1.2 Logická architektura*
[7] Záložka *DoctrineFixturesBundle* v [50]: `https://symfony.com/bundles/DoctrineFixturesBundle/current/index.html`
[8] Záložka *Messenger* v [50]: `https://symfony.com/doc/current/messenger.html`
[9] Kapitola *3.2.3 Twig*
[10] Záložka *Voters* v [50]: `https://symfony.com/doc/current/security/voters.html`
[11] Knihovna *PHPUnit*: `https://phpunit.de/index.html`
[12] Kapitola *3.3.1 Ukládání souborů*

4.3.1 Composer

Composer je nástroj pro správu závislostí v jazyce PHP. Umožňuje instalaci balíčků a jejich závislostí do projektu a zároveň zajišťuje správu jejich verzí.

V projektu ThinkLink je Composer nainstalován během sestavování Docker kontejneru. V rámci běžícího Docker kontejneru je pak Composer využíván pro kompletní správu závislostí celého projektu.

Pro správnou funkci Composeru jsou klíčové dva soubory: `composer.json` a `composer.lock`. Soubor `composer.json` definuje veškeré závislosti projektu včetně požadovaných verzí a dalších metadat. Při instalaci závislostí se automaticky vytvoří či upraví soubor `composer.lock`, který zaznamenává přesné verze všech nainstalovaných balíčků.

Při týmovém vývoji je `composer.lock` nezbytný pro zajištění konzistence prostředí mezi vývojáři. Umožňuje všem členům týmu sestavit totožné vývojářské prostředí.

4.3.2 Docker

Použití nástroje Docker bylo navrženo v kapitole *3.2.5 Docker*, kde byl rovněž popsán jeho princip fungování a jeho výhody pro vývoj. Tato kapitola se zaměřuje na konkrétní využití Dockeru v rámci vývoje aplikace ThinkLink, včetně popisu jednotlivých komponent.

Celý projekt je možné spustit v prostředí Docker, čímž se eliminuje závislost na různých vývojových prostředích jednotlivých zařízení. Spolu s nástroji jako Git a Composer přispívá Docker ke stejnému vývojovému a běhovému prostředí, konfiguraci i chování aplikace napříč různými zařízeními.

Aplikace je rozdělena do více částí (tzv. kontejnerů), které spolu spolupracují. Pro správu těchto kontejnerů je využit nástroj *Docker Compose*[13], který umožňuje definici a současný běh více kontejnerů najednou. Konfigurace je uložena v souboru `docker-compose.yml`. Tento soubor definuje následující kontejnery:

- **php** – hlavní kontejner aplikace. Definuje prostředí pro běh aplikace a je postaven na oficiálním PHP image (verze `8.3-fpm`). Jeho kontext obsahuje soubor `Dockerfile`, který kromě potřebných závislostí instaluje i Composer a Symfony CLI, aby je bylo možné v rámci kontejneru používat.
- **nginx** – webový server, na kterém aplikace běží. Kontejner využívá image `nginx:alpine`[14].
- **postgres** – kontejner pro databázový systém PostgreSQL. V projektu je použit upravený image s předinstalovaným rozšířením *pgvector*[15], které hraje

[13] Nástroj *Docker Compose*: `https://docs.docker.com/compose/`

[14] Nginx v [64]: `https://hub.docker.com/_/nginx`

[15] PostgreSQL s *pgvector* rozšířením v [64]: `https://hub.docker.com/r/pgvector/pgvector`

klíčovou roli při vyhledávání souvisejících poznámek. Tento kontejner využívá *volumes*[16] pro perzistentní ukládání dat, čímž se předejde jejich ztrátě při vypnutí nebo restartu kontejneru.

- **pgAdmin**[17] a **Adminer**[18] – grafické nástroje pro správu databáze, které usnadňují její administraci a práci s daty.

4.3.3 AssetMapper a Bootstrap

AssetMapper je nástroj frameworku Symfony určený pro správu tzv. assetů, souborů jako jsou obrázky, CSS styly nebo skripty (JavaScript). Představuje jednodušší alternativu k nástroji Webpack Encore[19]. Jeho hlavní výhodou je možnost správy assetů bez nutnosti jejich kompilace či sestavení, čímž se výrazně zjednodušuje vývojový proces.[20]

Konfigurace assetů probíhá prostřednictvím souboru `importmap.php`, ve kterém jsou definovány cesty k jednotlivým souborům, které mají být zpřístupněny přes veřejnou složku serveru. Formát `importmap.php` je podporován ve všech moderních prohlížečích a je součástí HTML standardu.

Prostřednictvím nástroje AssetMapper je do projektu také integrován CSS framework Bootstrap [56, 55]. Ten zajišťuje jednotný a responzivní vzhled, funkčnost a použitelnost uživatelského rozhraní aplikace ThinkLink.

4.4 Funkce převzaté z aplikace Obsidian

V rámci sekce *2.1 Analýza aplikace Obsidian* byla jako referenční poznámková aplikace zvolen Obsidian, který byla podrobně analyzován se zaměřením na interpretaci Markdown souborů, práci s odkazy mezi poznámkami a způsob ukládání souborů.

Pro porovnání interpretace Markdown textu v Obsidianu a v rámci aplikace ThinkLink byl vytvořen srovnávací soubor v ukázce 4.2, který zachytává základní syntaxi a vymezuje rozsah podporovaných zápisů. Tento soubor by měl být stejným způsobem interpretovaný aplikací Obsidian i aplikací ThinkLink.

V aplikaci ThinkLink je možné vytvářet přímé odkazy na jiné poznámky. Ze dvou běžně používaných formátů byl zvolen formát *Wikilink*, který je výchozí v prostředí Obsidian[21].

[16] Docker Volumes v [58]: `https://docs.docker.com/engine/storage/volumes/`
[17] pgAdmin 4 v [64]: `https://hub.docker.com/r/dpage/pgadmin4/`
[18] Adminer v [64]: `https://hub.docker.com/_/adminer/`
[19] Webpack Encore v [50]: `https://symfony.com/doc/current/frontend/encore/index.html`
[20] AssetMapper v [50]: `https://symfony.com/doc/current/frontend/asset_mapper.html`
[21] Kapitola *2.1.2 Odkazy v rámci aplikace Obsidian*

Výpis kódu 4.2 Srovnávací Markdown dokument pro Obsidian a ThinkLink, inspirováno v záložce *Basic formatting syntax* v [13]

```
# Nadpis 1
## Nadpis 2
### Nadpis 3
#### Nadpis 4
##### Nadpis 5
###### Nadpis 6

Odstavec 1

Odstavec 2

Neuspořádaný seznam:
- **tučný text**
- *text kurzívou*
- ***tučný text s kurzívou***

Uspořádaný seznam:
1. [Obsidian Help - externí odkaz](https://help.obsidian.md)
2. obyčejný text
3. \*text s hvězdičkami, který není kurzívou\*

![Engelbart - externí odkaz na obrázek s náhledem obrázku](https://history-computer.com/ModernComputer/Basis/images/Engelbart.jpg)

***
- seznam
   - vložený seznam
   - další záznam vloženého seznamu
      - vložený seznam ve vloženém seznamu
- další odrážka seznamu

---
```js
function fancyAlert(arg) {
  if(arg) {
    $.facebox({div:'#foo'})
  }
}
```
```

Zobrazování poznámek je zajištěno prezentační vrstvou aplikace, která je postavena na šablonovacím systému Twig. Twig umožňuje použití různých filtrů pro úpravu výstupu dat směrem k uživateli. Pro aplikaci ThinkLink je klíčový filtr `markdown_to_html`, který převádí Markdown zápis do HTML podoby, aby mohl být správně interpretován v prohlížeči klienta [65].

Funkcionalita převodu *Wikilink* odkazů na platné HTML odkazy směřující na další poznámky či vložené soubory (např. obrázky nebo PDF) je zajištěna vlastním rozšířením Twig knihovny. Tato komponenta kontroluje, zda cílová poznámka nebo soubor existují, a v případě, že ano, zobrazí funkční odkaz. Pokud cílový soubor neexistuje, je místo něj vykreslen vizuálně odlišný odkaz, který uživatele upozorní na neplatnost daného odkazu.

V rámci implementace prototypu aplikace ThinkLink byl kladen důraz na bezpečnost a důsledné ošetření uživatelských vstupů. Podobně jako aplikaci Obsidian není ani zde povolena plná interpretace HTML kódu vloženého uživatelem, neboť by to mohlo představovat bezpečnostní riziko[22]. Vložený HTML kód je proto při zobrazení poznámky převeden pomocí Twig filtru `escape`[23] na neškodný textový výstup a není interpretován prohlížečem.

Ukládání souborů je v prototypu řešeno tak, že veškeré nahrané soubory jsou ukládány do jedné složky (podobně jako v aplikaci Obsidian[24]) bez hierarchického členění. Tento přístup je technicky realizován pomocí knihovny *Flysystem*[25]. Přestože absence složkové struktury uživatelům nenabízí tradiční formu hierarchického třídění poznámek a souborů, umožňuje flexibilní práci s poznámkami pomocí jiných přístupů, např. *Zettelkasten*[26].

4.5 Zajištění rozšiřitelnosti pomocí vzoru Strategie

Jedním z hlavních požadavků kladených na implementaci aplikace ThinkLink je její rozšiřitelnost. V prototypu je tato rozšiřitelnost zajištěna zejména prostřednictvím návrhového vzoru *Strategy* (česky *Strategie*). Tato kapitola se věnuje popisu tohoto návrhového vzoru a ukazuje, jakým způsobem je jeho použití v aplikaci realizováno a jak lze díky tomu snadno přidávat nové funkcionality bez nutnosti zásahu do stávající logiky.

4.5.1 Návrhový vzor Strategie

Návrhový vzor *Strategie* je objektový behaviorální vzor, který je vhodné použít v situaci, kdy je určitý problém řešen několika různými způsoby či algoritmy a je potřeba tyto algoritmy dynamicky střídat. Návrhový vzor *Strategie* umožňuje jednotlivé způsoby řešení zapouzdřit a dát jim jednotné společné rozhraní.

[22] Více informací v sekci *2.1.3 HTML v aplikaci Obsidian*
[23] Filtr `escape` v [53]: `https://twig.symfony.com/doc/3.x/filters/escape.html`
[24] Kapitola *2.1.4 Souborový systém v aplikaci Obsidian*
[25] Kapitola *3.3.1 Ukládání souborů*
[26] Kapitola *1.1.2 Zettelkasten – analogové propojování poznámek*

To pak umožňuje dynamickou volbu konkrétního algoritmu v závislosti na aktuálním kontextu nebo požadavcích aplikace. [66]

Každá strategie je implementována jako samostatná třída, která odpovídá společnému rozhraní. Díky tomu lze jednotlivé strategie vyvíjet nezávisle, testovat, přidávat nebo odebírat bez zásahu do jiné logiky systému.

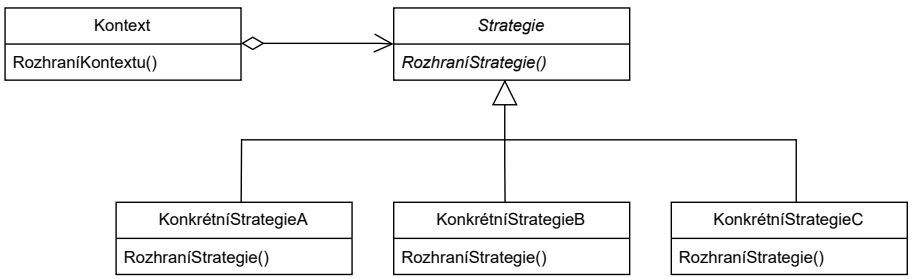
Návrhový vzor *Strategie* je podle slavné čtveřice *Gang of Four* [66] vhodný zejména tehdy, pokud:

- existuje více příbuzných tříd, které se liší pouze svým chováním,
- je potřeba poskytnout různé varianty algoritmu,
- je vhodné odstínit klienta od konkrétní implementace a datových struktur, které implementace používá,
- se v kódu vyskytuje množství podmíněných příkazů s různými operacemi.

V aplikaci ThinkLink se tento návrhový vzor využívá pro implementaci zpracování různých typů souborů (sekce *4.5.2 Rozšiřitelnost v oblasti typů podporovaných souborů*) či pro implementaci různých metod vyhledávání souvisejících poznámek (sekce *4.5.3 Rozšiřitelnost v oblasti vyhledávání souvisejících poznámek*).

4.5.1.1 Struktura

Obecná struktura návrhového vzoru *Strategie* je podle *Gang of Four* [66] definována podle diagramu 4.1.



Obrázek 4.1 Struktura návrhového vzoru *Strategie*, převzato z *Gang of Four* [66], vlastní tvorba v [67] a překlad.

Následující seznam popisuje jednotlivé části diagramu 4.1:

- **Strategie** – deklaruje společné rozhraní pro všechny podporované strategie. Aby bylo možné přidat novou strategii, musí implementovat toto rozhraní.
- **Kontext** – třída, která obsahuje referenci na konkrétní objekt typu *KonkrétníStrategie*.

- **KonkrétníStrategie** – konkrétní implementace algoritmu. Každá *KonkrétníStrategie* implementuje společné rozhraní definované ve třídě *Strategie*.

4.5.2 Rozšiřitelnost v oblasti typů podporovaných souborů

Prototyp aplikace ThinkLink podporuje nahrávání různých typů souborů: textových poznámek, obrázků, PDF dokumentů a ZIP archivů. Každý z těchto datových typů vyžaduje specifické zacházení v oblasti validace, ukládání a zobrazování.

Za účelem zajištění modulárního a snadno rozšiřitelného řešení byl zvolen návrhový vzor *Strategie*. Tento vzor umožňuje definovat pro každý podporovaný datový typ samostatnou strategii, která zajišťuje jeho zpracování.

Výhodou tohoto přístupu je možnost snadného rozšíření aplikace o podporu dalších datových typů (například audio nebo video) bez nutnosti zásahů do stávající aplikační logiky. Stejně tak lze jednoduše upravit způsob zpracování existujících typů souborů (například přidat funkci pro extrakci textu z PDF) bez rizika narušení ostatních částí systému.

Konkrétní ukázka použití návrhového vzoru Strategie

Konkrétní použití návrhového vzoru *Strategie* je ukázáno na strategii pro práci s obrázky: `ImageFileStrategy`.

Klíčovou roli hraje rozhraní `FileHandlerStrategyInterface`, které musí splňovat každá strategie pracující se soubory. Definice rozhraní je uvedena v ukázkovém kódu 4.3.

Rozhraní definované v 4.3 definuje následující metody:

- `supports(UploadedFile $file)`
  - Určuje, zda daná strategie podporuje práci s konkrétním typem souboru. Tato metoda je využívána při výběru vhodné strategie pro zpracování nahraného souboru.
- `validate(UploadedFile $file, ExecutionContextInterface $ctx)`
  - Odpovídá za validaci souboru. V případě nalezení chyb zapisuje příslušná chybová hlášení do validačního kontextu `$ctx`.
- `upload(UploadedFile $file)`
  - Provádí samotné nahrání souboru. V případě textové poznámky dochází k jejímu uložení do databáze, zatímco například obrázky nebo PDF soubory jsou ukládány do specifikované složky na serveru.
- `supportsServe(FilesystemFile $file)`

Výpis kódu 4.3 Rozhraní `FileHandlerStrategyInterface`

```php
<?php

interface FileHandlerStrategyInterface
{
    public function supports(UploadedFile $file): bool;

    public function upload(UploadedFile $file): void;

    public function validate(UploadedFile $file,
    ↪ ExecutionContextInterface $ctx): void;

    public function supportsServe(FilesystemFile $file): bool;

    public function serve(FilesystemFile $file, string
    ↪ $disposition = "inline"): Response;
}
```

  - Určuje, zda strategie podporuje obsluhu zadaného souboru při požadavku na jeho zobrazení nebo stažení.

- `serve(FilesystemFile $file, string $disposition = "inline")`
  - Zajišťuje výstup souboru k uživateli (např. obrázek nebo PDF ke stažení či zobrazení).

Validace každého nahraného souboru probíhá na základě:

- typu souboru (musí jít o jeden z podporovaných souborů),
- velikosti souboru (nesmí být překročen stanovený limit),
- unikátnosti referenčního názvu (ten slouží k jednoznačné identifikaci souboru, aby bylo možné vytvořit odkaz na tento soubor[27]).

Pokud soubor splňuje všechny požadavky, metoda `upload()` zajistí jeho uložení do lokálního souborového systému pomocí knihovny *Flysystem*[28].

Výběr správné strategie zajišťuje třída `FileHandlerCollection`, která má v konfiguračním souboru `services.yaml` nadefinované všechny dostupné implementace strategie jako parametry. Tato třída dle ukázky kódu 4.4 nabízí metodu pro získání konkrétní strategie pro zpracování souboru.

[27] Kapitola *2.1.2 Odkazy v rámci aplikace Obsidian*
[28] Kapitola *3.3.1 Ukládání souborů*

Výpis kódu 4.4 Hlavička metody `getFileHandler` vracející strategii pro zpracování souboru

```
getFileHandler(UploadedFile $file):
↪  ?FileHandlerStrategyInterface
```

Metoda v 4.4 iteruje nad všemi registrovanými strategiemi a pomocí metody `supports()` jejich rozhraní určí, která z nich podporuje zpracování daného souboru a vrátí ji. Tuto metodu používá např. kontrolér, který přijme nahrané soubory z formuláře a na základě jejich typu deleguje jejich zpracování na odpovídající strategii.

Strategie pro import ZIP archivů

Speciálním případem strategií pro práci se soubory je `ZIPArchiveStrategy`, která uživatelům umožňuje nahrát ZIP archiv obsahující poznámky a další soubory. Po nahrání ZIP archivu provede jeho rozbalení a pomocí iterace zjistí vhodnou strategii pro každý soubor. Následně deleguje validaci a nahrání souborů odpovídajícím třídám.

Vzhledem ke vzniku cyklické závislosti však nemůže být tato strategie součástí `FileHandlerCollection` a je tedy instanciována a využívána samostatně.

Přidání nové strategie

Přidání nové strategie pro datový typ je díky návrhovému vzoru *Strategie* velmi jednoduché a přímočaré. Pro přidání strategie je potřeba vytvořit třídu, která implementuje `FileHandlerStrategyInterface` a zaregistrovat ji v konfiguračním souboru `services.yaml` jako argument pro `FileHandlerCollection`. Aplikace ThinkLink následně tuto strategii automaticky použije při zpracování odpovídajících souborů.

4.5.3 Rozšiřitelnost v oblasti vyhledávání souvisejících poznámek

Důležitým požadavkem na prototyp aplikace ThinkLink je možnost rozšíření v oblasti metod pro automatické vyhledávání souvisejících poznámek. Tato rozšiřitelnost je v aplikaci zajištěna pomocí návrhového vzoru *Strategie*. Je důležité, aby bylo možné jednoduše přidávat, odebírat jednotlivé metody a upravovat jejich fungování.

Jednotlivé strategie pro vyhledávání jsou definovány jako služby v konfiguračním souboru `services.yaml`, kde jsou opatřeny štítkem[29]. Všechny služby označené tímto štítkem je možné automaticky vkládat do jiných služeb či kontrolérů. Konkrétní konfigurace je ukázána v kódu 4.5. Tento mechanismus umožňuje snadné rozšiřování systému o nové strategie, aniž by bylo nutné měnit existující kód.

Výpis kódu 4.5 Konfigurace štítku služby `app.relevant_notes_strategy`

```
# Konkrétní strategie se štítkem a prioritou
App\Service\RelevantNotes\TitleMatchStrategy\WebsearchTitleMat
↪  chStrategy::
  tags:
      - { name: 'app.relevant_notes_strategy', priority: 1 }

# ...

# Konkrétní strategie se štítkem a prioritou
App\Service\RelevantNotes\VectorEmbeddingStrategy\GeminiVector
↪  EmbeddingStrategy:
  tags:
      - { name: 'app.relevant_notes_strategy', priority: 10 }

# Kontext strategií
App\Controller\NoteController:
  arguments:
      - !tagged_iterator { tag: app.relevant_notes_strategy }
```

Za správu a delegování zodpovědností ohledně poznámek odpovídá kontrolér `NoteController`, který je v konfiguraci 4.5 nastaven tak, aby přijímal všechny strategie označené štítkem `app.relevant_notes_strategy`. Jakmile je do konfigurace přidána nová strategie s tímto štítkem, `NoteController` ji automaticky využije při generování seznamu souvisejících poznámek.

Každá strategie určená pro automatické vyhledávání souvisejících poznámek musí implementovat rozhraní `SearchStrategyInterface`. Toto rozhraní v ukázce 4.6 definuje dvě metody, jejichž implementace je nezbytná pro začlenění dané strategie do aplikace:

- `findRelevantNotes(Note $note, User $user)`

[29] Záložka *How to Work with Service Tags* v [50]: `https://symfony.com/doc/current/service_container/tags.html`

Výpis kódu 4.6 Rozhraní `SearchStrategyInterface`

```php
<?php

interface SearchStrategyInterface
{
  public function findRelevantNotes(Note $note, User $user):
  ↪  array;

  public function getStrategyMethodName(): string;
}
```

- Hlavní metoda, která na základě vstupní poznámky a přihlášeného uživatele vrací pole relevantních poznámek. Tyto výsledky jsou následně zobrazeny ve výstupu uživatelského rozhraní.

- `getStrategyMethodName()`
  - Tato metoda vrací název strategie, který se v prototypu zobrazuje v záhlaví nad výstupem jednotlivých metod. Slouží především v testovacím prostředí, kde jsou pro jednu poznámku zobrazovány výstupy více strategií vedle sebe a je potřeba je rozlišit.

Přidání nové strategie

Pro přidání nové metody automatického vyhledávání stačí vytvořit třídu implementující rozhraní `SearchStrategyInterface`, následně ji zaregistrovat v konfiguračním souboru `services.yaml` a opatřit ji štítkem podle ukázky 4.5.

Volitelně lze ke strategii vytvořit a připojit pomocné služby nebo entity, které jsou důležité pro zajištění její funkčnosti, rozdělení výpočtu apod.

Organizace strategií pomocí abstraktních tříd

V rámci prototypu aplikace ThinkLink je pro každou vybranou metodu automatického hledání souvisejících poznámek implementováno více konkrétních implementací (více informací v sekci *4.6 Automatické hledání souvisejících poznámek*). Pro zajištění přehledné organizace kódu jsou jednotlivé strategie reprezentované abstraktními třídami, které slouží jako základ pro konkrétní implementace. Tento přístup umožňuje sdílení společných metod a závislostí mezi konkrétními implementacemi strategie.

Například strategie využívající vektorové reprezentace poznámek z velkých jazykových modelů jsou postaveny nad abstraktní třídou, která implementuje rozhraní `SearchStrategyInterface` a definuje společné vlastnosti pro práci

s vektorovými reprezentacemi. Z této třídy dědí dvě konkrétní implementace této strategie:

- `GeminiVectorEmbeddingStrategy` – s modelem *Google Gemini API* [38],
- `OpenAIVectorEmbeddingStrategy` – s modelem *OpenAI* [37].

4.6 Automatické hledání souvisejících poznámek

Tato sekce se věnuje implementaci jednotlivých metod automatického hledání souvisejících poznámek v prototypu aplikace ThinkLink. U každé metody jsou popsány technické detaily realizace i jejich případná omezení.

4.6.1 Propojování podle výskytu názvu poznámky v obsahu jiných poznámek

Tato metoda využívá fulltextové vyhledávání nad databází PostgreSQL. To umožňuje identifikovat dokumenty, které odpovídají dotazu a jejich volitelné seřazení podle relevance. Nejběžnějším typem vyhledávání je hledání všech dokumentů obsahující hledané výrazy dotazu a jejich vrácení v pořadí podle podobnosti s dotazem.[30]

V aplikaci ThinkLink je při otevření poznámky použit její název jako vyhledávací dotaz. Tento dotaz se porovná s obsahem ostatních poznámek v databázi. Pokud metoda najde poznámky, které obsahují daný název ve svém textu, považuje je za relevantní.

Pro správné fungování této metody je v databázi vytvořen speciální sloupec typu `tsvector`, který obsahuje předzpracovaný text každé poznámky, aby se v nich dalo dobře vyhledávat. Pro zvýšení výkonu je nad tímto sloupcem vytvořen specializovaný GIN index, který funguje podobně jako invertované seznamy. Pro každý lexém udržuje seznam míst, kde se nachází.[31].

Pro převedení názvu a textu poznámky na formát vhodný pro fulltextové vyhledávání je využita databázová funkce `to_tsvector()`. V databázové migraci 4.7 je uvedeno, jak je pomocí jazyka SQL přidán sloupec typu `tsvector` a zároveň vytvořen odpovídající GIN index. Díky tomu se při každém vytvoření nebo aktualizaci poznámky automaticky aktualizuje i tento sloupec, aby odpovídal názvu a obsahu poznámky.[32]

Například pro testovací poznámku s názvem *Druhý mozek a digitální zahrada* v ukázce 4.8 vygeneruje PostgreSQL obsah sloupce `note_tsvector`, který se nachází v ukázce 4.9.

[30] Záložka *Chapter 12. Full Text Search* z [60]: `https://www.postgresql.org/docs/current/textsearch-intro.html`

[31] Záložka *Preferred Index Types for Text Search* z [60]: `https://www.postgresql.org/docs/current/textsearch-indexes.html`

[32] Záložka *Controlling Text Search* z [60]: `https://www.postgresql.org/docs/current/textsearch-controls.html`

Výpis kódu 4.7 Databázová migrace přidávající `tsvector` a GIN index

```php
<?php

$this->addSql("ALTER TABLE note ADD COLUMN note_tsvector
↪  tsvector GENERATED ALWAYS AS (to_tsvector('simple',
↪  coalesce(title, '') || ' ' || coalesce(content, '')))
↪  STORED");
$this->addSql("CREATE INDEX idx_note_tsvector ON note USING GIN
↪  (note_tsvector)");
```

Výpis kódu 4.8 Ukázka vložené poznámky (vlastní poznámka, zkráceno)

```
Systém, který doplňuje můj mozek, díky kterému jsem
↪  produktivnější. Hlavní část systému jsou uspořádané
↪  poznámky, kde si mohu uspořádávat svoje projekty, znalosti
↪  a myšlenky a propojovat je mezi sebou.

### Rozdíl mezi druhým mozkem a digitální zahradou
Rozdíl je minimální (aspoň pro mě), **digitální zahrada je
↪  obecnější pojem a je to místo, kde přemýšlím, mám svoje
↪  znalosti a můžu si ji kultivovat.** Část zahrady je
↪  přístupná veřejnosti. **Druhý mozek si spíš asociuji s
↪  výkonnou částí digitální zahrady, s tou akční částí, která
↪  mi pomáhá s projekty, připomíná mi věci a díky které jsem
↪  organizovanější a produktivnější.**
```

Výpis kódu 4.9 Výstup `tsvector` pro testovací poznámku 4.8

```
'a':3,27,29,38,52,61,93,98 'akční':83 'asociuji':75 'aspoň':44
↪  'digitální':4,39,47,79 'doplňuje':7 'druhý':1,71
↪  'druhým':36 'díky':10,94 'hlavní':14 'je':31,42,49,53,68
↪  'ji':64 'jsem':12,96 'jsou':17 'kde':20,56 'která':85
↪  'které':95 'kterému':11 'který':6 'kultivovat':65
↪  'mezi':32,35 'mi':86,91 'minimální':43 'mohu':22
↪  'mozek':2,9,72 'mozkem':37 'myšlenky':28 'mám':58
↪  'místo':55 'mě':46 'můj':8 'můžu':62 'obecnější':50
↪  'organizovanější':97 'pojem':51 'pomáhá':87 'poznámky':19
↪  'pro':45 'produktivnější':13,99 'projekty':25,89
↪  'propojovat':30 'přemýšlím':57 'připomíná':90
↪  'přístupná':69 'rozdíl':34,41 's':76,81,88 'sebou':33
↪  'si':21,63,73 'spíš':74 'svoje':24,59 'systému':16 'to':54
↪  'tou':82 'uspořádané':18 'uspořádávat':23 'veřejnosti':70
↪  'výkonnou':77 'věci':92 'zahrada':48 'zahradasystém':5
↪  'zahradou':40 'zahrady':67,80 'znalosti':26,60 'část':15,66
↪  'částí':78,84
```

Každý lexém je v 4.9 doplněn o seznam pozic v textu poznámky, což umožňuje efektivní vyhledávání.

Aby mohl být vyhledávací dotaz korektně zpracován, je potřeba název poznámky také převést do formátu `tsquery`. PostgreSQL podle dokumentace [68] poskytuje několik funkcí, které tento převod umožňují, přičemž každá z nich se liší způsobem zpracování vstupního textu:

- `plainto_tsquery()`,
- `phraseto_tsquery()`,
- `websearch_to_tsquery()`,
- `to_tsquery()`.

V prototypu aplikace ThinkLink jsou implementovány všechny výše uvedené metody, aby bylo možné je mezi sebou porovnat a zvolit tu nejvhodnější pro vyhledávání relevantních poznámek (v kapitole *5 Testování*). Následuje přehled jednotlivých strategií podle použité funkce.

PlainTitleMatchStrategy Funkce `plainto_tsquery()` převede neformátovaný text na `tsquery` tak, že text analyzuje a normalizuje: rozdělí ho na lexémy, odstraní speciální znaky a mezi lexémy vloží logický operátor `&` (AND).

Výsledkem je dotaz `tsquery`, který vyhledává dokumenty obsahující všechny lexémy z názvu poznámky bez ohledu na jejich pořadí. [68]

PhraseTitleMatchStrategy Funkce `phraseto_tsquery()` funguje stejně jako `plainto_tsquery()`, ale mezi jednotlivé lexémy vkládá operátor `<->` (FOLLOWED BY). Ten zajistí, že budou vyhledány pouze ty dokumenty, které obsahují přesnou posloupnost lexémů ve stejném pořadí, v jakém se vyskytují v názvu otevřené poznámky. [68]

WebsearchTitleMatchStrategy Funkce `websearch_to_tsquery()` má alternativní syntax, pomocí kterého ve vstupním neformátovaném textu identifikuje operátory, které ovlivňují, jakým způsobem bude text vyhledáván. Vstupní text je podle dokumentace [68] zpracován následovně:

- běžný text: zpracován stejně jako v `plainto_tsquery()`,
- text v uvozovkách: interpretován stejně jako v `phraseto_tsquery()`,
- slovo `OR` nebo `or`: převedeno na operátor `|` (OR),
- pomlčka (`-`): převedena na operátor `!` (NOT).

TsqueryORTitleMatchStrategy Tato strategie využívá základní funkci `to_tsquery()`, která očekává plně strukturovaný výraz s explicitně zadanými operátory (`&`, `|`, `!`, `<->`). V rámci strategie je název poznámky předzpracován tak, že je text manuálně rozdělen na lexémy, jsou odstraněna česká stop slova[33] a jednotlivé lexémy jsou spojeny pomocí operátoru `|` (OR). Výsledkem je dotaz, který vrací všechny poznámky obsahující alespoň jeden z lexémů z názvu otevřené poznámky. [68]

4.6.1.1 Řazení relevantních poznámek

Pro účely zobrazení těch nejrelevantnějších poznámek uživateli je třeba využít funkce, které umožňují měřit míru relevance jednotlivých dokumentů vzhledem k dotazu. PostgreSQL v této oblasti poskytuje dvě funkce, které zohledňují četnost výskytu hledaných lexémů v dokumentu, jejich vzájemnou vzdálenost a také relativní důležitost části dokumentu, ve které se lexémy nacházejí. [68]

Vzhledem k tomu, že pojem relevance není přesně definovaný a každý systém jej může interpretovat odlišně, PostgreSQL umožňuje vytvořit vlastní hodnoticí funkce přizpůsobené konkrétním potřebám daného systému. V implementaci metod pro vyhledávání poznámek jsou nicméně využity dvě standardní funkce PostgreSQL[34]:

[33] Český stop list od MUNI: `https://nlp.fi.muni.cz/cs/StopList`
[34] Sekce *Ranking Search Results* v [68]

- `ts_rank(...)` Řadí vektory na základě četnosti výskytu lexémů z dotazu v dokumentu.
- `ts_rank_cd(...)` Řadí vektory jak na základě četnosti lexémů (jako předchozí funkce), tak i na základě vzájemné blízkosti lexémů odpovídajících zadanému dotazu.

Příklad SQL dotazu, který využívá `plainto_tsquery()` k vytvoření dotazu a následně řadí výsledky podle skóre relevance pomocí funkce `ts_rank_cd()` se nachází v ukázce 4.10.

Výpis kódu 4.10 SQL dotaz vyhledávající nejrelevantnější poznámky pomocí funkce `plainto_tsquery()`.

```
1 SELECT note.*, ts_rank_cd(note.note_tsvector,
  ↪  plainto_tsquery(:searchTerm)) AS score
2 FROM note
3 WHERE note.owner_id = :userId
4   AND note.note_tsvector @@ plainto_tsquery(:searchTerm)
5 ORDER BY score DESC
```

V ukázce 4.10 představují výrazy `:searchTerm` a `:userId` parametry, které jsou do dotazu vkládány až v době jeho provedení pomocí nástroje *QueryBuilder* knihovny Doctrine[35]. Tento přístup zajišťuje ochranu proti útokům typu SQL injection.

4.6.2 Propojování pomocí vektorových reprezentací a TF-IDF

Tento přístup využívá vektorové reprezentace poznámek, mezi nimiž lze efektivně počítat vzájemnou podobnost. Každá poznámka je převedena na vektor v n-rozměrném prostoru a porovnání mezi poznámkami se pak provádí na základě vzdálenosti nebo úhlu mezi těmito vektory.

Pro určení vzdálenosti mezi jednotlivými vektory existuje několik metrik, přičemž v rámci tohoto projektu byla zvolena kosinová podobnost (anglicky *cosine similarity*), protože se jedná o vhodnou a nejběžnější metriku pro výpočet podobnosti dokumentů.[36] Také je tento přístup konzistentní s pojmenováním v rámci PostgreSQL (funkce podobnosti `ts_rank()` či dostupnost kosinové podobnosti v rozšíření *pgvector*[37]).

[35] Záložka *SQL Query Builder* v [61]: `https://www.doctrine-project.org/projects/doctrine-dbal/en/4.2/reference/query-builder.html`

[36] Kapitola *1.4.5.1 Kosinová podobnost*

[37] Sekce *Querying* v [62]

4.6.2.1 Zpracování poznámky a výpočet TF-IDF vektoru

Po vytvoření či úpravě poznámky je třeba pro poznámku vytvořit novou vektorovou reprezentaci. Proces tvorby vektoru má několik částí:

Předzpracování poznámky Před samotným výpočtem vektoru pro upravenou či přidanou poznámku musí proběhnout její předzpracování. Zpracovává se jak název, tak celý obsah poznámky. V aplikaci tento proces zajišťuje služba `TextPreprocessor`, která postupně provádí následující kroky:

- odstranění speciálních znaků a URL adres,
- převod veškerého textu na malá písmena,
- rozdělení textu na jednotlivé tokeny (slova),
- odstranění českých stop slov[38],
- úprava tvarů slov do kořenového tvaru s knihovnou *CzechStemmer* [69].

Výsledkem je pole upravených termů, které slouží jako základ pro výpočet vektorové reprezentace poznámky. První složkou výpočtu TF-IDF je *term frequency* (TF), která vyjadřuje relativní četnost výskytu daného termu v rámci dané poznámky. Ucelený popis fungování TF-IDF se nachází v kapitole *1.4.4.2 TF-IDF*.

Ke zpracovávané poznámce je vytvořena mapa termů, kde klíč odpovídá konkrétnímu termu a hodnota odpovídá počtu jeho výskytů v dané poznámce. Na základě těchto hodnot je spočtena relativní četnost výskytu termu (TF) podle vzorce 4.1[39].

$$\mathrm{TF}(t, d) = \frac{\text{Počet výskytů termu } t \text{ v poznámce } d}{\text{Celkový počet termů v poznámce } d} \tag{4.1}$$

Aktualizace globálního slovníku pro konkrétního uživatele Druhou složkou výpočtu TF-IDF je *inverse document frequency* (IDF), která vyjadřuje, jak často se daný term vyskytuje napříč všemi poznámkami určitého uživatele. Zatímco TF zachycuje lokální význam termu v rámci jedné poznámky, IDF poskytuje míru důležitosti termu v rámci celého souboru poznámek.

Pro výpočet IDF je nutné udržovat globální slovník termů, který se aktualizuje při každé změně nebo vytvoření poznámky. V rámci prototypu a testování různých přístupů k automatickému hledání relevantních poznámek je tento slovník vždy vytvářen znovu, aby výpočty hodnot pro určení relevantních poznámek vždy vycházely z nejaktuálnějších dat. Tento přístup zajišťuje

[38] Seznam českých stop slov od MUNI: `https://nlp.fi.muni.cz/cs/StopList`

[39] Převzato z `https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/` (vlastní překlad)

maximální přesnost výsledků, avšak nese s sebou i jistá omezení, která jsou podrobněji diskutována v kapitole *4.6.2.2 Omezení a možnosti řešení*.

Po předzpracování poznámky na pole tokenů dojde nejprve ke smazání starého globálního slovníku termů daného uživatele. Pro každou poznámku jsou postupně do nového slovníku přidávány nové termy spolu s informací, v kolika poznámkách se nacházejí. Tento aktualizovaný slovník pak slouží jako základ pro výpočet IDF části pomocí vzorce 4.2[40].

$$\text{IDF}(t, D) = \frac{\text{Celkový počet poznámek v korpusu } D}{\text{Počet poznámek obsahující term } t} \tag{4.2}$$

Vytvoření vektorové reprezentace pro každou poznámku Pro vytvoření vektorů, které budou reprezentovat jednotlivé poznámky je třeba, aby všechny vektory měly stejnou dimenzi a aby každá dimenze reprezentovala určitý term. Dimenze vektorů je zpravidla menší než celková velikost slovníku, takže je potřeba vybrat množinu termů o velikosti stanovené dimenze vektoru, které jsou dostatečně rozlišující pro všechny poznámky a každému z těchto termů přiřadit jeden index ve vektoru. Pomocí vzorce 4.3 se vypočítá finální TF-IDF hodnota pro konkrétní term t v poznámce d a v kolekci všech poznámek uživatele D[41].

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{4.3}$$

Způsob výběru množiny termů pro vektorovou reprezentaci Jedním ze způsobů výběru množiny termů, které budou tvořit vektorovou reprezentaci poznámek, je vybrat termy s nejvyšší průměrnou hodnotou TF-IDF. To jsou takové termy, které jsou pro danou kolekci nejvíce informačně důležité. Nejsou příliš běžné (nevyskytují se ve většině poznámek) a ani příliš unikátní (nevyskytují se pouze v jedné poznámce).

Konkrétně je pro každý term v globálním slovníku daného uživatele spočtena jeho průměrná TF-IDF hodnota napříč všemi jeho poznámkami. Tato hodnota je pak k danému termu uložena do databáze.

Při konstrukci vektorové reprezentace poznámky je následně vybráno N termů s nejvyššími průměrnými TF-IDF hodnotami (podle stanovené dimenze vektoru). Tento výběr zajistí, že výsledné vektory budou tvořeny právě těmi termy, které jsou nejvíce informačně hodnotné napříč celou kolekcí poznámek daného uživatele.

Tvorba finální vektorové reprezentace poznámek Poté je v cyklu přes všechny poznámky vypočítána jejich finální vektorová reprezentace. Pro kaž-

[40] Převzato z `https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/` (vlastní překlad)

[41] Vzorec převzat z `https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/` (vlastní překlad)

dou poznámku se iteruje přes vybranou množinu termů a ke každému se spočítá konkrétní TF-IDF hodnota vztažená k této poznámce a celkové kolekci. Výsledný vektor dané poznámky je následně uložen do databáze pomocí rozšíření *pgvector*[42].

Hledání relevantních poznámek Rozšíření *pgvector* poskytuje funkce pro výpočet vzdáleností mezi vektorovými reprezentacemi dokumentů včetně výpočtu kosinové podobnosti vektorů. [62]

Při otevření poznámky uživatelem se provede SQL dotaz v ukázce 4.11, který pomocí operátoru `<=>` (kosinová vzdálenost) zjistí nejbližší vektory k vektoru aktuálně zobrazené poznámky. [62]

Výpis kódu 4.11 SQL dotaz pro nalezení nejbližších vektorů pomocí kosinové vzdálenosti

```
1  SELECT
2    tf_idf_vector.*,
3    (tf_idf_vector.vector <=> (SELECT vector FROM tf_idf_vector
     ↪  WHERE note_id = :noteId)) AS distance
4  FROM tf_idf_vector
5  JOIN note ON note.id = tf_idf_vector.note_id
6  WHERE tf_idf_vector.note_id != :noteId AND note.owner_id =
   ↪  :userId
7  ORDER BY distance
8  LIMIT 10;
```

Dotaz v ukázce 4.11 porovnává vektor otevřené poznámky s vektory ostatních poznámek daného uživatele a vrací 10 poznámek s nejmenší kosinovou vzdáleností. Tyto poznámky jsou pak zobrazeny uživateli jako nejrelevantnější podle metody TF-IDF.

4.6.2.2 Omezení a možnosti řešení

Samotné předzpracování poznámek, a hlavně tvorba globálního slovníku při každé změně poznámky, představuje výpočetně náročnou operaci. Jak již bylo uvedeno, pro účely prototypu, jehož cílem je porovnat různé metody automatického hledání relevantních poznámek, se tento výpočet provádí po každém přidání nebo úpravě poznámky, aby bylo dosaženo maximální aktuálnosti dat.

Aby byla zajištěna plynulost aplikace pro uživatele, je tento výpočet prováděn asynchronně pomocí *Symfony Messenger*[43]. Po úpravě poznámky tak

[42] Sekce *3.3.2.1 Rozšíření pgvector*

[43] Záložka *Messenger: Sync & Queued Message Handling* v [50]: `https://symfony.com/doc/current/messenger.html`

může uživatel s aplikací dále pracovat, zatímco se výpočet spustí na pozadí. Tento přístup významně zlepšuje uživatelský komfort, avšak přináší potenciální riziko zobrazení neaktuálních výsledků. V případě, kdy si uživatel prohlíží relevantní poznámky, tak data o relevanci nemusí být ještě aktualizována. Jedná se o kompromis mezi aktuálností výsledků a uživatelským komfortem.

Alternativním řešením je využití plánovaných úloh pomocí nástroje *Scheduler* od Symfony[44], který umožňuje spouštění příkazů v předem definovaných intervalech. V tomto režimu by aplikace při změně poznámky provedlo pouze její předzpracování, aby byla připravena pro výpočet podobnosti, ale změny globálního slovníku by se do relevancí poznámek promítly až při dalším naplánovaném spuštění úlohy.

Tento přístup by přinesl významnou úsporu výpočetních zdrojů a snížil zátěž serveru, a to na úkor mírně snížené přesnosti a aktuálnosti dat. V produkčním prostředí by se mohlo jednat o vhodný kompromis mezi výkonem systému a kvalitou výsledků.

4.6.3 Propojování pomocí vektorových reprezentací vygenerovaných LLM

Tato metoda využívá vektorové reprezentace poznámek generované pomocí velkých jazykových modelů (LLM).

Metoda funguje tak, že po vytvoření nebo úpravě poznámky se její název a obsah odešlou pomocí POST požadavku na API vybraného poskytovatele jazykového modelu. V těle požadavku je obsažen text poznámky a případně další konfigurační parametry definované poskytovatelem. Služba následně vrátí odpověď ve formátu JSON, která obsahuje výsledný vektor reprezentující danou poznámku.

Tato vektorová reprezentace je následně převedena na datový typ `Vector` z rozšíření *pgvector*[45] a uložena do databáze. V ukázkách kódu 4.12 a 4.13 jsou uvedené ukázky požadavků na vybrané poskytovatele jazykových modelů: OpenAI a Google.

Z bezpečnostních důvodů je API klíč uložen v souboru `.env.local`, který není součástí verzovacího systému. Tím je zajištěno, že klíč nebude nechtěně zveřejněn v repozitáři se zdrojovým kódem.

Google Gemini API [38] umožňuje navíc specifikovat účel použití vektorové reprezentace pomocí parametru `taskType`. Pro účel projektu je použita hodnota `SEMANTIC_SIMILARITY` (viz ukázka požadavku 4.12), která optimalizuje vektorovou reprezentaci pro účely měření podobnosti textů.

Po získání vektorové reprezentace a po jejím uložení do databáze probíhá hledání relevantních poznámek stejně jako u přístupu založeného na TF-IDF. Při otevření poznámky se pomocí SQL dotazu najdou nejbližší vektory po-

[44] Záložka *Scheduler* v [50]: `https://symfony.com/doc/current/scheduler.html`
[45] Kapitola *3.3.2.1 Rozšíření pgvector*

Výpis kódu 4.12 Struktura požadavku na Google Gemini API [38]

```php
<?php

$response = $this->httpClient->request(
  'POST',
  'https://generativelanguage.googleapis.com/v1beta/models/text-embedding-004:embedContent?key=' .
  ↪ $_ENV['GOOGLE_API_KEY'],
  [
      'headers' => [
          'Content-Type' => 'application/json',
      ],
      'json' => [
          'content' => [
              'parts' => [
                  [
                      'text' => $note->getTitle() . ' ' .
                      ↪ $note->getContent(),
                  ]
              ],
          ],
          'taskType' => 'SEMANTIC_SIMILARITY'
      ],
  ]
);
```

Výpis kódu 4.13 Struktura požadavku na OpenAI API [37]

```php
<?php

$response = $this->httpClient->request(
  'POST',
  'https://api.openai.com/v1/embeddings',
  [
      'headers' => [
          'Content-Type' => 'application/json',
          'Authorization' => 'Bearer ' .
          ↪  $_ENV['OPENAI_API_KEY'],
      ],
      'json' => [
          'model' => 'text-embedding-3-small',
          'input' => $note->getTitle() . ' ' .
          ↪  $note->getContent(),
      ]
  ]
);
```

mocí kosinové podobnosti. Poznámky odpovídající těmto vektorům jsou poté zobrazeny uživateli jako nejrelevantnější.

Kapitola 5
Testování

Tato kapitola se věnuje testování prototypu webové aplikace ThinkLink. Testování mělo dva hlavní cíle:

- porovnat jednotlivé strategie automatického hledání souvisejících poznámek a vybrat tu nejspolehlivější,
- ověřit funkčnost a použitelnost aplikace.

Testování proběhlo formou přímé interakce uživatelů s prototypem aplikace ThinkLink. Aplikace byla nasazena na vzdálený server a uživatelé k ní přistupovali prostřednictvím svého prohlížeče. Každý uživatel měl vlastní účet, ve kterém byla připravena testovací sada poznámek, se kterou mohl uživatel pracovat.

5.1 Testované metody a postup testování

V kapitole *3.4 Metody hledání souvisejících poznámek* byly vybrány následující metody pro automatické hledání souvisejících poznámek:

- propojování podle výskytu názvu poznámky v obsahu jiných poznámek,
- propojování pomocí vektorových reprezentací a TF-IDF,
- propojování pomocí vektorových reprezentací vygenerovaných LLM.

Každá z uvedených metod má v rámci prototypu ThinkLink několik konkrétních implementací. Aby bylo možné objektivně vyhodnotit, která z těchto metod je nejspolehlivější pro automatické hledání souvisejících poznámek, bylo nutné provést následující kroky:

1. definice relevance v kontextu souvisejících poznámek,

2. výběr konkrétní implementace pro každou testovanou metodu hledání souvisejících poznámek,
3. provedení uživatelského testování a sběr hodnocení úspěšnosti jednotlivých metod,
4. vyhodnocení výsledků testování.

Po skončení uživatelského testování podle pevně daných scénářů měli uživatelé možnost s aplikací dále pracovat, aby mohli komplexněji zhodnotit funkčnost a použitelnost aplikace z pohledu běžného uživatele.

5.2 Definice relevance

Při porovnávání poznámek z hlediska jejich vzájemné podobnosti a významové blízkosti je nezbytné stanovit jasnou definici a kritéria relevance. V rámci této práce je relevance chápána jako míra tematické, významové či kontextové souvislosti mezi dvěma poznámkami. Je posuzována podle následujících hledisek:

- **Tematická blízkost**: poznámky se zabývají stejným nebo podobným tématem,
- **Stejná klíčová slova**: poznámky obsahují společné výrazy, termíny či pojmy,
- **Doplňující či rozšiřující obsah**: poznámka doplňuje či rozšiřuje obsah jiné poznámky,
- **Stejný účel**: poznámky slouží ke stejnému či podobnému účelu.

Pro objektivní výběr vhodné implementace pro všechny metody automatického hledání souvisejících poznámek byla vytvořena testovací sada poznámek a v tabulce 5.1 byly definovány konkrétní hodnoty relevancí mezi testovacími poznámkami (10 značí nejvyšší relevanci, nerelevantní poznámky jsou označeny pomlčkou). Všechny poznámky se nacházejí v přiloženém ZIP archivu. V následujícím seznamu poznámek jsou v závorce uvedeny zkrácené názvy pro snadnější orientaci v tabulce relevancí:

- **Red velvet dort** (zkratka: Red velvet) – recept na Red velvet dort, obsahuje zmínku o narozeninách Pavla.
- **Recept: Chilli con carne** (zkratka: Chilli) – standardní recept na Chilli con carne.
- **Gordon Ramsay omeleta** (zkratka: Omeleta) – zápisky z videa o přípravě omelety podle Gordona Ramsayho.

- **Jak zklidnit stres** (zkratka: Jak na stres) – poznámka o správném dýchání pro zvládání stresu.
- **Dýchání pomáhá** (zkratka: Dýchání) – poznámka o tom, jak správně dýchat pro zmírnění stresu.
- **Narozeniny Pavel** (zkratka: Pavel) – poznámka o plánování narozeninové oslavy pro Pavla, včetně zmínky o pečení Red velvet dortu a dárku v podobě knihy o investování.
- **Podcast Peníze dělají peníze** (zkratka: Podcast) – zápis myšlenek z fiktivního podcastu o investování.
- **Přednáška: Základy investování – Pasivní přístup** (zkratka: Pas. přístup) – zápisky z přednášky o pasivním investování.
- **Přednáška: Aktivní investování – Strategie, příležitosti a rizika** (zkratka: Strategie) – zápisky z přednášky o aktivním investování.
- **Uložený článek: Understanding Investor Behavior** (zkratka: Inv. behavior) – zkopírovaný anglický článek z webové stránky o investování.
- **Myšlenková poznámka – klid jako investiční výhoda** (zkratka: Myšlenka) – vlastní poznámka o významu klidu při poklesu trhu.
- **Výběr GPU do počítače** (zkratka: Výběr GPU) – stručná poznámka o výběru grafické karty.
- **Chilli con carne a výběr GPU – víc společného, než se zdá** (zkratka: GPU a Chilli) – poznámka spojující dvě jinak nesouvisející témata: recept a výběr hardware.

Jak je z poznámek patrné, byly navrženy tak, aby pokrývaly různé scénáře relevance, například:

- Recepty jsou tematicky odlišné, ale sdílí stejný účel a strukturu.
- Poznámky *Jak zklidnit stres* a *Dýchání pomáhá* jsou sémanticky i tematicky velmi blízké, ale liší se názvem a formulací obsahu.
- Poznámka *Narozeniny Pavel* obsahuje zmínky relevantní k poznámkám z jiných tématických skupin (Red velvet dort a investování).
- Skupina investičních poznámek obsahuje různé formy záznamů: přednášky, zápis z podcastu, zkopírovaný článek a vlastní myšlenku.
- Poznámka *Chilli con carne a výběr GPU – víc společného, než se zdá* slouží jako specifický případ, kdy je jeden text relevantní ke dvěma vzájemně nerelevantním poznámkám.

Tato sada poznámek byla sestavena tak, aby reflektovala různorodost typických poznámek běžných uživatelů. Výběr poznámek vychází z výsledků dotazníku v kapitole *1.2 Analýza potřeb uživatelů*, kde 75 % uživatelů poznámkových aplikací vytváří osobní poznámky (např. deník, rychlé zápisky, to-do list) a studijní poznámky (např. univerzita, kurzy, příprava do školy, výzkumné účely).

Testovací poznámky byly záměrně koncipovány jako jednoduché, nestrukturované a neorganizované, protože podle dotazníku více než 89 % respondentů nepoužívá pokročilé metody organizace poznámek (např. *CODE* [1] či *Zettelkasten* [2]) a téměř 60 % z nich tyto metody vůbec nezná.

5.3 Výběr implementací metod pro uživatelské testování

Následující srovnání porovnává jednotlivé implementace v rámci každé ze tří hlavních strategií samostatně. Cílem je vybrat pro každou strategii jednu nejvhodnější implementaci, která bude dále reprezentovat danou strategii při uživatelském testování.

5.3.1 Výběr implementace pro strategii propojování podle výskytu názvu poznámky v obsahu jiných poznámek

V této strategii bylo implementováno několik různých přístupů. Tyto přístupy jsou popsány v kapitole *4.6.1 Propojování podle výskytu názvu poznámky v obsahu jiných poznámek*. Tyto strategie budou porovnány na základě kvalitativního hodnocení.

Shrnutí chování testovaných funkcí:

- `phraseto_tsquery()` je příliš restriktivní, protože vyžaduje striktně stejnou posloupnost slov jako v názvu otevřené poznámky. V důsledku toho nezachytila relevantní poznámky obsahující stejná slova v jiném pořadí nebo s mezislovy. Například poznámka *Narozeniny Pavel* obsahovala slovní spojení „dort Red velvet“ a je relevantní k poznámce *Red velvet dort*, nicméně narozdíl od ostatních metod tuto relevanci kvůli jinému pořadí slov funkce nezachytila.
- `plainto_tsquery()` hledá pouze dokumenty obsahující všechny lexémy z názvu poznámky, což může vyloučit relevantní poznámky, které obsahují pouze část hledaných výrazů. Tato metoda, narozdíl od předchozí `phraseto_tsquery()` zachytí relevanci poznámky *Narozeniny Pavel*, která obsahuje slovním spojení „dort Red velvet“, ale pokud se v jiné poznámce

	Red velvet dort	Recept: Chilli con carne	Gordon Ramsay omeleta	Jak zklidnit stres	Dýchání pomáhá	Narozeniny Pavel	Podcast Peníze dělají peníze – poznámky	Přednáška: Základy investování – Pasivní přístup	Přednáška: Aktivní investování – Strategie, příležitosti a rizika	Uložený článek: Understanding Investor Behavior	Myšlenková poznámka – klid jako investiční výhoda	Výběr GPU do počítače	Chilli con carne a výběr GPU – víc společného, než se zdá
Red velvet	–	9	8	–	–	10	–	–	–	–	–	–	7
Chilli	8	–	9	–	–	–	–	–	–	–	–	–	10
Omeleta	9	10	–	–	–	–	–	–	–	–	–	–	–
Jak na stres	–	–	–	–	10	7	–	6	–	8	9	–	–
Dýchání	–	–	–	10	–	7	–	6	–	8	9	–	–
Pavel	10	–	–	4	3	–	8	7	6	9	5	–	–
Podcast	–	–	–	–	–	6	–	10	9	8	7	–	–
Pas. přístup	–	–	–	5	4	6	8	–	10	9	7	–	–
Strategie	–	–	–	–	–	6	7	10	–	9	8	–	–
Inv. behavior	–	–	–	9	8	4	5	7	6	–	10	–	–
Myšlenka	–	–	–	9	8	4	5	7	6	10	–	–	–
Výběr GPU	–	–	–	–	–	–	–	–	–	–	–	–	10
GPU a Chilli	7	10	8	–	–	–	–	–	–	–	–	9	–

Tabulka 5.1 Definice relevancí mezi jednotlivými testovacími poznámkami.

vyskytne spojení pouze „Red velvet“, tak bude tato poznámka považována za nerelevantní, protože neobsahuje lexém „dort“.

- `websearch_to_tsquery()` funguje obdobně jako `plainto_tsquery()`, ale interpretuje speciální znaky (uvozovky, pomlčky) a může vést k potenciálně nechtěným výsledkům. Například v poznámce *Přednáška: Základy investování – Pasivní přístup* pomlčka způsobila vyloučení klíčového slova „Pasivní“.
- `to_tsquery()` s operátorem `OR` je schopna detekovat samostatná slova z názvu poznámky v textech ostatních poznámek. To ji dává větší možnosti v hledání relevantních poznámek. Avšak u této metody je vyšší riziko nalezení nesouvisející poznámky obsahující pouze jedno slovo z názvu, které může být vytržené z kontextu.

Výběr řazení výsledků

Rozdíl mezi funkcemi `ts_rank()` a `ts_rank_cd()` je v tom, že `ts_rank_cd()` kromě frekvence výskytů bere v potaz také vzájemnou blízkost lexémů v dokumentu (kapitola *4.6.1.1 Řazení relevantních poznámek*). Tento přístup byl ověřen vložením klíčových slov „red“, „velvet“ a „dort“ do nerelevantní poznámky *Uložený článek: Understanding Investor Behavior* na vzdálená místa v textu. Výsledek ukázal, že `ts_rank_cd()` správně preferovalo relevantní poznámku *Narozeniny Pavel*, kde jsou hledaná slova blízko sebe, oproti upravené poznámce s nerelevantním článkem, kde slova byla od sebe vzdálená.

5.3.1.1 Výsledky kvalitativního hodnocení

Na základě kvalitativního hodnocení pomocí testovací sady poznámek a definovaných kritérií relevance se jako nejvhodnější ukázala kombinace:

- použití funkce `to_tsquery()` s operátorem `OR` mezi jednotlivými lexémy,
- řazení výsledků pomocí `ts_rank_cd()`.

Tato kombinace umožnila zachytit i částečné shody mezi názvem poznámky a obsahem jiných poznámek a zároveň při řazení výsledků upřednostnila dokumenty, kde se hledané lexémy nacházely blíže u sebe.

5.3.2 Výběr implementace pro strategii propojování pomocí vektorových reprezentací a TF-IDF

Implementace hledání relevantních poznámek pomocí TF-IDF je detailně popsána v kapitole *4.6.2 Propojování pomocí vektorových reprezentací a TF-IDF*. Každá poznámka má přiřazenou vektorovou reprezentaci a při hledání relevantních poznámek je využita kosinová podobnost mezi těmito vektory.

V rámci strategie založené na TF-IDF existuje pouze jedna implementace, a ta je proto přímo vybrána pro uživatelské testování.

Pro účely širšího testování a porovnání spolehlivosti je tato implementace zařazena také do kvantitativního porovnání spolu s implementacemi strategie využívající vektorové reprezentace generované pomocí velkých jazykových modelů. Ačkoliv se strategie TF-IDF nebude účastnit přímého výběru mezi implementacemi *OpenAI* a *Google Gemini*, její zahrnutí přináší dodatečný srovnávací rámec. Díky tomu je možné kromě uživatelského hodnocení získat také kvantitativní přehled o tom, jak si stojí přístup založený na TF-IDF ve srovnání s metodami využívajícími velké jazykové modely.

5.3.3 Výběr implementace pro strategii propojování pomocí vektorových reprezentací vygenerovaných LLM

V kapitole *4.6.3 Propojování pomocí vektorových reprezentací vygenerovaných LLM* je popsána funkčnost dvou přístupů k této strategii: *OpenAI API* a *Google Gemini API*.

Pro objektivní porovnání kvality výsledků jednotlivých modelů bude využita metrika *Normalized Discounted Cumulative Gain* (NDCG). Tato metrika je vhodná pro hodnocení systémů řazení a doporučování, protože bere v potaz nejen počet relevantních výsledků (podobně jako metrika *mean Average Precision*[1]), ale také jejich pořadí vzhledem k ideálnímu pořadí. [70]

Ideální pořadí relevantních poznámek pro každou testovací poznámku je definováno v tabulce 5.1 a slouží jako zdroj pravdy pro výpočet hodnot NDCG. Do uživatelského testování bude vybrán ten model, který pro více poznámek získá lepší NDCG skóre oproti druhému modelu.

5.3.3.1 Výpočet metriky NDCG

Pro výpočet NDCG slouží podle článku [70] vzorce 5.1 a 5.2.

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} \tag{5.1}$$

$$\text{DCG@K} = \sum_{k=1}^{K} \frac{rel_i}{\log_2(i+1)} \tag{5.2}$$

Kde K je počet zobrazovaných výsledků. V aplikaci ThinkLink je vždy zobrazeno $K = 10$ nejrelevantnějších poznámek.

Pro každou poznámku se vypočítá hodnota DCG@10, tedy *Discounted Cumulative Gain*, samostatně pro výsledky ze strategie *OpenAI* a *Google Gemini*.

[1] mAP: `https://www.evidentlyai.com/ranking-metrics/mean-average-precision-map`

Hodnota rel_i představuje relevanci poznámky na i-té pozici ve výsledném seznamu. Mezi testovacími poznámkami je relevance definována v tabulce 5.1 následovně:

- Relevance je určena pořadím – čím vyšší pozice, tím vyšší hodnota rel_i,
- Nerelevantní poznámky jsou označeny pomlčkou a mají hodnotu $rel_i = 0$.

Logaritmický jmenovatel snižuje váhu relevantních výsledků, které se umístily níže v seznamu (tedy mají vyšší pozici i), čímž reflektuje význam správného řazení. Tedy, pokud se relevantní poznámka s hodnotou relevance 10 doporučí na první pozici, bude mít větší vliv na výslednou hodnotu DCG@10 než kdyby byla doporučena na druhé pozici.

Po výpočtu hodnot DCG pro obě strategie (*OpenAI* a *Google Gemini*) se pro každou poznámku spočítá hodnota IDCG@10, která odpovídá ideálnímu seřazení podle definovaného pořadí relevance v tabulce 5.1.

Výsledná hodnota NDCG@10 = DCG@10/IDCG@10 vyjadřuje, jak blízko jsou skutečné výsledky ideálnímu scénáři. Implementace s vyšší počtem poznámek s lepší hodnotou NDCG bude následně vybrána pro uživatelské testování jako reprezentant strategie vyhledávání pomocí vektorových reprezentací z LLM.

Ukázkový výpočet metriky NDCG@10 pro poznámku *Přednáška: Základy investování – Pasivní přístup* se nachází v příloze B. V tabulce 5.2 jsou vypočtené hodnoty NDCG@10 pro všechny definované poznámky. Výpočet proběhl manuálně s pomocí Python skriptu v příloze A.

Poznámka	**NDCG$_{\text{Gemini}}$**	**NDCG$_{\text{OpenAI}}$**	**NDCG$_{\text{TF-IDF}}$**
Red velvet	**0,9969**	0.9942	0.9608
Chilli	0,9933	**1**	0.9558
Omeleta	**1**	0.9765	0.8930
Jak na stres	0.8021	**0.8310**	0.6001
Dýchání	**0.8230**	0,8143	0,7636
Pavel	0,6864	**0,7941**	0.7674
Podcast	0.9668	0.9661	**0.9681**
Pas. přístup	**0,9878**	0.9593	0.9113
Strategie	**0,9766**	0,9416	0,9416
Inv. Behavior	0,8841	**0,9386**	0,7724
Myšlenka	0,8707	**0,8896**	0,8413
Výběr GPU	1	1	1
GPU a Chilli	**0,9815**	0,9293	0,8856

Tabulka 5.2 Výsledky NDCG@10 pro jednotlivé poznámky a strategie

5.3.3.2 Výsledky kvantitativního hodnocení

Na začátku testu bylo definováno, že do uživatelského testování bude vybrán ten model, který pro více poznámek získá lepší NDCG skóre. Z tabulky 5.2 je vidět, že v porovnání mezi *Google Gemini* a *OpenAI* doporučilo *Google Gemini* lepší výsledky v sedmi případech a *OpenAI* pouze v pěti případech. Do uživatelského testování je tady vybrán model *text-embedding-004* od *Google Gemini*. U poznámky *Výběr GPU do počítače* všechny metody vrátili stejné a ideální pořadí relevantních poznámek.

5.3.4 Závěr výběru jednotlivých implementací metod pro uživatelské testování

Na základě kvalitativního hodnocení metod propojování poznámek dle výskytu názvu poznámky v obsahu jiných poznámek a kvantitativního testu strategií využívajících vektorové reprezentace byly pro uživatelské testování vybrány následující implementace:

- **PostgreSQL Full Text Search** využívající funkci `to_tsquery()` s operátorem `OR` mezi lexémy a řazením výsledků pomocí `ts_rank_cd()`.
- **TF-IDF** pro tvorbu vektorových reprezentací poznámek a použití kosinové podobnosti pro hledání nejbližších vektorů.
- **Google Gemini** s modelem *text-embedding-004* pro tvorbu vektorových reprezentací a kosinovou podobností pro hledání nejbližších vektorů.

Tyto vybrané implementace budou použity v rámci uživatelského testování k porovnání kvality automatického hledání souvisejících poznámek v aplikaci ThinkLink.

5.4 Uživatelské testování

Tato část popisuje průběh uživatelského testování prototypu aplikace ThinkLink. Testování mělo dvě hlavní části. V první části byla ověřována spolehlivost jednotlivých přístupů k automatickému hledání souvisejících poznámek. V druhé části, kdy uživatelé už měli zkušenosti s používáním prototypu, byla hodnocena celková použitelnost aplikace pomocí standardizovaného SUS dotazníku.

Každému uživateli byl pro testování vytvořen samostatný účet s rolí *Běžný uživatel*, který mu umožnil přístup do aplikace a zároveň zajistil, aby bylo testování oddělené a nebylo ovlivněno činností ostatních uživatelů.

Pro účely testování byl také vytvořen elektronický dotazník, který obsahoval vysvětlení účelu testování, instrukce pro přihlášení a postup celým testováním.

Samotný elektronický dotazník, testovací sada poznámek a kompletní výsledky testování se nacházejí v přiloženém ZIP archivu.

5.4.1 Uživatelské testování metod pro hledání souvisejících poznámek

Vzhledem k tomu, že vnímání relevance mezi dvěma poznámkami může být u každého uživatele mírně odlišné, bylo provedeno uživatelské testování, jehož cílem bylo zhodnotit úspěšnost různých přístupů k automatickému hledání souvisejících poznámek. Uživatelé během testování postupně hodnotili jednotlivé metody na škále od 1 do 10 podle toho, jak relevantní poznámky jim daná metoda doporučila.

Na základě výsledků tohoto testování bude vyhodnocena úspěšnost jednotlivých metod a ta, která dosáhne nejvyššího průměrného hodnocení, bude zvolena jako hlavní metoda pro automatické propojování poznámek v aplikaci ThinkLink.

Pro účely testování byla vytvořena sada testovacích poznámek a odpovídajících scénářů, které simulují reálné případy použití funkce automatického hledání relevantních poznámek. Scénáře byly sestaveny na základě odpovědí uživatelů v rámci analýzy potřeb uživatelů v kapitole *1.2.2 Výsledky dotazníku* a reflektují situace, ve kterých by uživatelé očekávali doporučení relevantního obsahu.

Prototyp aplikace ThinkLink byl upraven tak, aby u každé poznámky zobrazoval vedle sebe tři seznamy doporučených poznámek. Každý ze seznamů je generovaný jednou z vybraných metod pro hledání relevantních poznámek. V testovacím prototypu jsou zobrazeny výsledky konkrétních implementací, který byly v rámci předvýběru vybrány.

Aby bylo zajištěno objektivní hodnocení bez kognitivního zkreslení, byly metody v rozhraní aplikace označeny neutrálně jako *Metoda č. 1*, *Metoda č. 2* a *Metoda č. 3* a jejich pořadí bylo náhodně vybráno.

Aplikace byla pro účely testování nasazena do produkčního prostředí na veřejně dostupný server, aby k ní měli uživatelé přístup kdykoliv a kdekoliv.

5.4.1.1 Průběh testování

Po přihlášení do aplikace měl každý uživatel možnost se seznámit s připravenou sadou testovacích poznámek, která byla pro všechny stejná. Následně uživatelé postupně procházeli jednotlivé scénáře v dotazníku a u každého z nich hodnotili výstupy všech tří metod automatického vyhledávání souvisejících poznámek.

Hodnocení probíhalo na škále od 1 do 10, kde hodnota 1 znamenala, že daná metoda nenabídla žádné relevantní poznámky, zatímco hodnota 10 odpovídala situaci, kdy metoda zobrazila všechny relevantní poznámky v ideálním pořadí.

Příklad scénáře Pro ilustraci průběhu uživatelského testování uvádí výpis 5.1 ukázku scénáře č. 2, který testuje schopnost metody najít roztroušené poznámky k jednomu tématu, zaznamenané v delším časovém období.

Výpis kódu 5.1 Testovací scénář č. 2 – Inspirace pro osobní projekt

```
Název: Scénář č. 2:  Inspirace pro osobní projekt (hledání
↪ nápadů a inspirace)

V tomto scénáři se testuje schopnost zobrazit relevantní
↪ poznámky, které byly v průběhu času uloženy jako nápady
↪ či inspirace.

Máte více volna a chtěli byste dát dohromady svůj napůl
↪ vymyšlený osobní projekt: Aplikace pro správu
↪ přečtených knih. S tímto nápadem v hlavě chodíte už
↪ nějakou dobu a občas si uložíte poznámku s nápadem či
↪ inspirací pro tento projekt. Nyní je čas tyto poznámky
↪ dát dohromady. Otevřete si poznámku Nápad - Aplikace
↪ pro správu přečtených knih. Jak vám dostupné metody
↪ doporučily relevantní poznámky? Hodí se pro váš
↪ projekt?
```

Celkem bylo připraveno osm scénářů. Poslední z nich byl specifický tím, že uživatelům poskytoval volnost v testování aplikace. Uživatelé mohli vytvářet nové poznámky, upravovat existující nebo nahrát vlastní poznámky ze svého zařízení. I v tomto scénáři uživatelé následně hodnotili spolehlivost jednotlivých metod.

Na závěr elektronického dotazníku měli uživatelé možnost vložit slovní zpětnou vazbu prostřednictvím volitelné otevřené otázky.

5.4.1.2 Vyhodnocení spolehlivosti jednotlivých metod

Uživatelského testování se zúčastnilo celkem 32 účastníků, kteří u každého z osmi testovacích scénářů hodnotili všechny tři metody automatického hledání souvisejících poznámek na škále od 1 (nejhorší výsledek) do 10 (nejlepší výsledek). V následujícím seznamu jsou uvedena průměrná hodnocení jednotlivých metod přes všechny scénáře:

- Metoda propojování pomocí výskytu názvu poznámky v obsahu jiným poznámek: **3,008**
- Metoda propojování pomocí vektorových reprezentací a TF-IDF: **6,410**

- Metoda propojování pomocí vektorových reprezentací vygenerovaných pomocí LLM: **7,225**

Z výsledků testování jednoznačně vyplývá, že nejlépe hodnocenou metodou je přístup využívající vektorové reprezentace generované pomocí velkých jazykových modelů. Uživatelé tuto metodu v průměru napříč všemi scénáři včetně scénáře s vlastními poznámkami hodnotili jako nejúspěšnější při doporučování tematicky a významově souvisejících poznámek.

Metoda založená na výpočtu TF-IDF se také ukázala jako úspěšná a její hodnocení bylo mírně nižší než u metody s vektorovými reprezetacemi z velkých jazykových modelů. Vzhledem k tomu, že tato metoda nevyžaduje odesílání obsahu poznámek ke zpracování externí službě, může být vhodnou alternativou pro uživatele, kteří pracují s citlivými nebo důvěrnými informacemi.

Naopak metoda vyhledávání podle výskytu názvu poznámky v obsahu jiných poznámek dopadla v testování nejhůře. V některých scénářích ani nebyla schopna navrhnout žádné relevantní poznámky, a proto od uživatelů často dostávala velmi nízké hodnocení.

Aplikace byla po skončení testování upravena do finální podoby, kdy umožňuje správu poznámek a k tomu automaticky doporučuje relevantní poznámky pomocí nejúspěšnější metody využívající vektorové reprezentace vytvářené pomocí LLM. Vždy je ale možné jednoduše pomocí konfigurace metodu doporučování poznámek změnit.

Po skončení testování byla odhalena chyba v metodě hledání relevantních poznámek pomocí TF-IDF. V procesu předzpracování poznámky do pole jednotlivých tokenů proběhlo odstranění diakritiky. Toto odstranění diakritiky poté způsobilo nesprávnou funkci knihovny *CzechStemmer* [69]. Tato chyba mohla ovlivnit výkon této metody a v rámci budoucího vývoje aplikace je doporučeno nové testování. Samotná chyba je v aplikaci již opravena. I tak metoda fungovala správně a hledala relevantní poznámky, jen nebyl využit plný potenciál správného *stemmingu* českých slov.

5.4.2 Testování použitelnosti aplikace

V rámci uživatelského testování, během kterého měli účastníci možnost aktivně pracovat s prototypem aplikace ThinkLink, bylo provedeno také testování její použitelnosti. K tomuto účelu byl využit standardizovaný SUS (*System Usability Scale*) dotazník [71].

5.4.2.1 Formát SUS dotazníku

Dotazník obsahuje deset výroků, na které uživatelé odpovídají pomocí škály od 1 (zcela nesouhlasím) až 5 (zcela souhlasím). Jednotlivé výroky byly zadány v tomto pořadí (vlastní překlad):

1. Myslím, že bych tuto aplikaci rád/a používal/a opakovaně.

2. Myslím, že aplikace byla zbytečně komplexní.
3. Myslím si, že se aplikace používala snadno.
4. Myslím, že bych potřeboval/a pomoc někoho technicky zdatnějšího, abych mohl/a používat tuto aplikaci.
5. Jednotlivé funkce aplikace se mi zdají dobře zapracované.
6. Myslím si, že je aplikace neucelená/nejednotná.
7. Umím si představit, že se většina lidí s aplikací naučí pracovat velmi rychle.
8. Myslím si, že je používání systému těžko ovladatelné.
9. Při používání aplikace jsem se cítil/a velmi jistě.
10. Potřeboval/a bych se naučit hodně věcí, než bych mohl/a s aplikací pracovat.

Na závěr dotazníku měli účastníci možnost poskytnout volitelnou slovní zpětnou vazbu, kde mohli popsat případné potíže s používáním systému nebo navrhnout zlepšení. Výsledky testování použitelnosti a komentáře uživatelů budou zohledněny v dalším vývoji aplikace ThinkLink.

5.4.2.2 Výsledky SUS dotazníku

Podle Brooka [71] se výsledky SUS dotazníku interpretují následovně:

1. U lichých otázek (1, 3, 5, 7, 9) se od hodnoty uvedené na škále odečítá 1.
2. U sudých otázek (2, 4, 6, 8, 10) se hodnota na škále odečítá od 5.
3. Vypočítá se bodový výsledek pro každou otázku tak, že se sečtou body odpovědí všech uživatelů a tento součet je vydělen počtem uživatelů.
4. Vzniklé průměrné obodování jednotlivých otázek se sečte a celkový součet se následně vynásobí koeficientem 2,5.
5. Skóre SUS má rozsah od 0 do 100.

Výsledné SUS skóre pro aplikaci ThinkLink je **81,094**. Podle Saura [72] má průměrné SUS skóre hodnotu 68. Skóre aplikace ThinkLink tedy lze považovat za nadprůměrné, což značí dobrou celkovou použitelnost prototypu.

5.4.2.3 Kvalitativní vyhodnocení

Ve volitelné zpětné vazbě uživatelé uvedli několik konkrétních problémů souvisejících s uživatelským prostředím a použitelností aplikace:

- Chybějící možnosti řazení poznámek (například podle abecedy).
- Vyhledávání poznámek funguje pouze při přesné shodě včetně velikosti písmen, což je v praxi neintuitivní.
- Při vytváření poznámek chybí nápověda ke správné syntaxi odkazů na jiné poznámky nebo na vložení nahraných souborů.
- Některé z metod po relevantních poznámkách také doporučovaly úplně nerelevantní poznámky, což působilo matoucím dojmem.

Tyto připomínky budou dále analyzovány a zapracovány do další fáze vývoje aplikace ThinkLink s cílem zlepšit uživatelskou přívětivost a intuitivnost rozhraní.

Závěr

Cílem této práce bylo zefektivnit správu digitálních poznámek prostřednictvím funkce automatického hledání tematicky a obsahově souvisejících poznámek.

Na začátku byla provedena rešerše existujících aplikací pro správu poznámek. Rešerše byla zaměřena především na to, jakými způsoby existující aplikace umožňují automatickou detekci souvislostí. V rámci této rešerše ještě navíc proběhla analýza potřeb uživatelů, díky které bylo možné definovat seznam funkcionalit, které jsou pro uživatele skutečně důležité a podle kterých bylo možné existující aplikace porovnat. Součástí byla také rešerše možností, jak propojovat textové informace na webu.

Součástí práce byla také softwarová analýza a návrh aplikace ThinkLink, která umožňovala správu poznámek s funkcí automatického hledání relevantních poznámek. Při návrhu byl kladen důraz na snadnou rozšiřitelnost v oblasti zpracování různých datových typů i v oblasti metod pro vyhodnocování relevance. Aplikace byla navíc navržena tak, aby byla kompatibilní s existující aplikací Obsidian.

Aplikace ThinkLink byla na základě provedeného návrhu implementována. V rámci aplikace byla implementována jedna naivní a dvě pokročilé metody pro hledání relevantních poznámek. Výsledná aplikace byla úspěšně nasazena do produkčního prostředí na vzdálený server.

Za účelem vyhodnocení úspěšnosti jednotlivých přístupů bylo provedeno uživatelské testování. Do testování byly v rámci kvantitativního a kvalitativního testování vybrány tři implementace reprezentující jednotlivé přístupy k hledání relevantních poznámek. Uživatelé poté hodnotili jejich schopnost doporučení relevantních poznámek jak v předem definovaných scénářích, tak ve volném používání aplikace. Na základě výsledků testování byla vybrána metoda využívající vektorové reprezentace vytvářené pomocí LLM.

Na závěr byla testována celková použitelnost aplikace pomocí SUS dotazníku. Výsledné SUS skóre dosáhlo hodnoty 81, což značí nadprůměrnou úroveň použitelnosti prototypu.

Všechny stanovené cíle této práce byly splněny. Výsledkem je funkční a otestovaný prototyp aplikace ThinkLink, který slouží jako nástroj pro efektivní

správu poznámek a automatickou detekci jejich souvislostí. Na základě získané zpětné vazby lze konstatovat, že o takovou funkcionalitu je mezi uživateli zájem a aplikace má potenciál pro další rozvoj a rozšiřování o nové funkce.

Možné rozšíření aplikace ThinkLink

Aplikace ThinkLink je připravena na další vývoj a vylepšení. Díky zvoleným technologiím a architektuře je zajištěna stabilita, dlouhodobá udržitelnost a jednotné vývojové prostředí napříč různými zařízeními.

Již nyní existuje seznam funkčních požadavků s nižší prioritou, které nebyly v rámci této práce implementovány, ale jejich doplnění by přispělo k lepšímu uživatelskému zážitku z používání aplikace. Mezi takové funkce patří např. přidání podpory pro více jazyků, možnost zveřejnit vybrané poznámky na webu či možnost upravovat styl zobrazení poznámek pomocí vlastních CSS pravidel.

Také bude do budoucího vývoje potřeba zahrnout získanou zpětnou vazbu od uživatelů v testování. Zejména nastavení a implementaci prahových hodnot podobnosti (anglicky *threshold*) pro jednotlivé přístupy k hledání relevantních poznámek, aby bylo možné zobrazit pouze relevantní poznámky. Přesné hodnoty prahů podobnosti by byly předmětem dalšího testování.

Díky snadné rozšiřitelnosti aplikace je možné rychle a snadno přidávat nové typy souborů a nové metody automatického hledání souvisejících poznámek. To je zásadní výhoda zejména v kontextu rychlého vývoje technologií v této oblasti.

Dalším krokem v rámci vývoje by mohlo být pokročilé testování uživatelského rozhraní a použitelnosti aplikace k identifikaci nedostatků a zlepšení práce uživatele s aplikací. S ohledem na využitý Symfony framework lze zvážit integraci jeho rozšíření *Symfony UX Turbo*[2] pro zlepšení plynulosti a uživatelského zážitku z aplikace.

[2]Záložka *Symfony UX Turbo v [50]:* `https://symfony.com/bundles/ux-turbo/current/index.html`

Dodatek A

Python skript pro výpočet metriky NDCG

Tento pomocný skript slouží k výpočtu metrik DCG a NDCG pro konkrétní poznámku. Pro výpočet potřebných hodnot je potřeba zadat hodnoty relevance z tabulky 5.1.

Výpis kódu A.1 Výpočet DCG a NDCG pro porovnání relevance výsledků jednotlivých metod

```
import math

def addend(relevance, pos):
    return (relevance / (math.log2(1 + pos)))

def dcg(arr):
    index = 0
    sum = 0
    pos = 1
    while index != len(arr):
        sum += addend(arr[index], pos)
        pos += 1
        index += 1
    return sum

scoreIdeal = [10, 9, 8, 7, 6, 5, 4, 0, 0, 0]
scoreGemini = [10, 8, 7, 9, 6, 5, 0, 4, 0, 0]
scoreOpenAI = [10, 8, 7, 0, 6, 9, 4, 5, 0, 0]
scoreTfIdf =  [10, 7, 8, 9, 0, 6, 0, 0, 4, 0]

ideal = dcg(scoreIdeal)

print(f'DCG IDEAL je {dcg(scoreIdeal)}')
print(f'DCG Gemini je {dcg(scoreGemini)}; NDCG je
↪ {dcg(scoreGemini)/ideal}')
print(f'DCG OpenAI je {dcg(scoreOpenAI)}; NDCG je
↪ {dcg(scoreOpenAI)/ideal}')
print(f'DCG TF-IDF je {dcg(scoreTfIdf)}; NDCG je
↪ {dcg(scoreTfIdf)/ideal}')
```

Dodatek B

Ukázkový výpočet metriky NDCG

Následuje ukázkový výpočet metriky NDCG@10 pro poznámku *Přednáška: Základy investování – Pasivní přístup*. Na základě definované relevance v tabulce 5.1 je pro tuto poznámku určeno následující ideální pořadí relevantních poznámek (s hodnotami rel_i):

1. Přednáška: Aktivní investování – Strategie, příležitosti a rizika ($rel_1 = 10$)
2. Uložený článek: Understanding Investor Behavior ($rel_2 = 9$)
3. Podcast Peníze dělají peníze ($rel_3 = 8$)
4. Myšlenková poznámka – klid jako investiční výhoda ($rel_4 = 7$)
5. Narozeniny Pavel ($rel_5 = 6$)
6. Jak zklidnit stres ($rel_6 = 5$)
7. Dýchání pomáhá ($rel_7 = 4$)

Ideální pořadí: $[10, 9, 8, 7, 6, 5, 4, 0, 0, 0]$

Výpočet IDCG@10 v B.1:

$$\begin{aligned} \text{IDCG@10} &= \frac{10}{\log_2(1+1)} + \frac{9}{\log_2(2+1)} + \frac{8}{\log_2(3+1)} + \frac{7}{\log_2(4+1)} \\ &\quad + \frac{6}{\log_2(5+1)} + \frac{5}{\log_2(6+1)} + \frac{4}{\log_2(7+1)} + 0 + 0 + 0 \\ &= \frac{10}{1} + \frac{9}{\log_2 3} + \frac{8}{2} + \frac{7}{\log_2 5} + \frac{6}{\log_2 6} + \frac{5}{\log_2 7} + \frac{4}{\log_2 8} \\ &\approx 10 + 5.68 + 4 + 3.01 + 2.32 + 1.78 + 1.33 = 28.12 \end{aligned} \tag{B.1}$$

DCG@10 pro *Google Gemini API* v B.2:

Skutečné pořadí s rel_i hodnotami: $[10, 8, 7, 9, 6, 5, 0, 4, 0, 0]$

$$\begin{aligned}\text{DCG@10}_{\text{Gemini}} = &\frac{10}{\log_2(1+1)} + \frac{8}{\log_2(2+1)} + \frac{7}{\log_2(3+1)} \\ &+ \frac{9}{\log_2(4+1)} + \frac{6}{\log_2(5+1)} + \frac{5}{\log_2(6+1)} + 0 \\ &+ \frac{4}{\log_2(8+1)} + 0 + 0 \\ &\approx 10 + 5.05 + 3.5 + 3.87 + 2.33 + 1.78 + 0 + 1.27 \\ &+ 0 + 0 = 27.80\end{aligned} \tag{B.2}$$

DCG@10 pro *OpenAI API* v B.3:

Skutečné pořadí s rel_i hodnotami: $[10, 8, 7, 0, 6, 9, 4, 5, 0, 0]$

$$\begin{aligned}\text{DCG@10}_{\text{OpenAI}} = &\frac{10}{\log_2(1+1)} + \frac{8}{\log_2(2+1)} + \frac{7}{\log_2(3+1)} + 0 \\ &+ \frac{6}{\log_2(5+1)} + \frac{9}{\log_2(6+1)} + \frac{4}{\log_2(7+1)} \\ &+ \frac{5}{\log_2(8+1)} + 0 + 0 \\ &\approx 10 + 5.05 + 3.5 + 0 + 2.33 + 3.46 \\ &+ 1.33 + 1.58 = 27.25\end{aligned} \tag{B.3}$$

DCG@10 pro *TF-IDF* v B.4:

Skutečné pořadí s rel_i hodnotami: $[10, 7, 8, 9, 0, 6, 0, 0, 4, 0]$

$$\begin{aligned}\text{DCG@10}_{\text{TF-IDF}} = &\frac{10}{\log_2(1+1)} + \frac{7}{\log_2(2+1)} + \frac{8}{\log_2(3+1)} \\ &+ \frac{9}{\log_2(4+1)} + 0 + \frac{6}{\log_2(6+1)} + 0 + 0 \\ &+ \frac{4}{\log_2(9+1)} + 0 \\ &\approx 10 + 4.41 + 4 + 3.87 + 0 + 2.14 + 0 \\ &+ 0 + 1.32 = 25.74\end{aligned} \tag{B.4}$$

Výpočet NDCG@10 podle vzorce 5.1:

$$\text{NDCG@10}_{\text{Gemini}} = \frac{27.80}{28.13} \approx 0.9879$$
$$\text{NDCG@10}_{\text{OpenAI}} = \frac{27.25}{28.13} \approx 0.9593$$
$$\text{NDCG@10}_{\text{TF-IDF}} = \frac{25.64}{28.13} \approx 0.9113$$

Z tohoto srovnání vychází, že *Google Gemini API* přístup nejlépe seřadil nejrelevantnější poznámky k poznámce *Přednáška: Základy investování – Pasivní přístup*.

Dodatek C

Další dokumenty

Následující příloha obsahuje formální dokumenty vyžadované Metodickým pokynem č. 6/2023 pro zadávání, odevzdávání, ukládání a zpřístupňování bakalářských a diplomových závěrečných prací. Konkrétně obsahuje zadání ze systému KOS a prohlášení o využití nástrojů AI (Příloha 4 metodického pokynu) tak, aby bylo dosaženo splnění metodiky.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kudrnovský** Jméno: **Petr** Osobní číslo: **506876**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav: **Katedra softwarového inženýrství**

Studijní program: **Informatika**

Specializace: **Webové inženýrství**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

ThinkLink - aplikace na propojování poznámek

Název bakalářské práce anglicky:

ThinkLink - note linking app

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. David Bernhauer, Ph.D. katedra softwarového inženýrství FIT

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

Datum zadání bakalářské práce: **29.01.2025** Termín odevzdání bakalářské práce: **16.05.2025**

podpis vedoucí(ho) ústavu/katedry

podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

Datum převzetí zadání

Kudrnovský Petr

Podpis studenta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kudrnovský** Jméno: **Petr** Osobní číslo: **506876**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav: **Katedra softwarového inženýrství**

Studijní program: **Informatika**

Specializace: **Webové inženýrství**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

ThinkLink - aplikace na propojování poznámek

Název bakalářské práce anglicky:

ThinkLink - note linking app

Pokyny pro vypracování:

Správa znalostí se v poslední době přesouvá z oblasti firemní do oblasti osobního rozvoje. Na rozdíl od firemního prostředí, kde se na správu know-how dbá více, správa osobních poznámek může trpět nedostatečnou provázaností jednotlivých poznámek. Cíl této práce je navrhnout aplikaci, která umožní automatické hledání souvisejících poznámek.
1. Proveďte rešerši existujících aplikací pro správu poznámek. Zaměřte se na to jakým způsobem umožní automatickou detekci souvislostí.
2. Proveďte softwarovou analýzu a návrh aplikace pro zobrazování poznámek a jejich vzájemných souvislostí, zaměřte se na rozšiřitelnost v oblasti automatického hledání souvisejících poznámek.
3. Implementujte prototyp takové aplikace na základě provedeného návrhu. Implementujte jednu naivní metodu pro provázání poznámek a alespoň jednu pokročilejší techniku.
4. Proveďte uživatelské testování prototypu a vyhodnoťte úspěšnost pokročilejší techniky v porovnání s naivním přístupem.

Seznam doporučené literatury:

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY
Thákurova 9
160 00 Praha 6

PROHLÁŠENÍ

Já, níže podepsaný

Příjmení, jméno studenta:	Kudrnovský Petr
Osobní číslo:	506876
Název programu:	Informatika

prohlašuji, že jsem bakalářskou práci s názvem

ThinkLink - aplikace na propojování poznámek

vypracoval samostatně a uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací a Rámocovými pravidly používání umělé inteligence na ČVUT pro studijní a pedagogické účely v Bc a NM studiu.

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 08.05.2025

Petr Kudrnovský
.....
podpis

Bibliografie

1. FORTE, Tiago. *Building a Second Brain: A Proven Method to Organize Your Digital Life and Unlock Your Creative Potential*. Žádná velká věda. Přel. MOHELSKÁ, Libuše. Brno, Česká republika: Jan Melvil Publishing, 2022. ISBN 978-80-7555-189-4.
2. AHRENS, Sönke. *How to Take Smart Notes*. Žádná velká věda. Přel. EŠNEROVÁ, Kateřina. Brno, Česká republika: Jan Melvil Publishing, 2021. ISBN 978-80-7555-146-7.
3. WILLETT, Wesley; GOFFIN, Pascal; ISENBERG, Petra. Understanding digital note-taking practice for visualization. *IEEE Computer Graphics and Applications* [online]. 2015, roč. 35, č. 4, s. 38–51. ISSN 1558-1756. Dostupné z DOI: 10.1109/MCG.2015.52. [Citováno 8. 3. 2025].
4. MICROSOFT CORPORATION. *The Innovator's Guide to Modern Note Taking* [online]. 2017. Dostupné také z: https://info.microsoft.com/rs/157-GQE-382/images/EN-US%2017034_MSFT_WWSurfaceModernNoteTaking_ebookRefresh_R2.pdf. [Citováno 27. 4. 2025].
5. SCHUBMEHL, David; VESSET, Dan. The knowledge quotient: Unlocking the hidden value of information using search and content analytics. *International Data Corporation (IDC)*. 2014. Dostupné také z: https://pages.coveo.com/rs/coveo/images/IDC-Coveo-white-paper-248821.pdf. [Citováno 27. 4. 2025].
6. SWEARNGIN, Amanda; IQBAL, Shamsi; POZNANSKI, Victor; ENCARNACIÓN, Mark; BENNETT, Paul N; TEEVAN, Jaime. Scraps: Enabling mobile capture, contextualization, and use of document resources. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* [online]. New York, NY, USA: Association for Computing Machinery, 2021, s. 1–14. Dostupné z DOI: 10.1145/3411764.3445185. [Citováno 8. 3. 2025].

7. CEVOLINI, Alberto; SCHMIDT, Johannes F.K. *Forgetting Machines: Knowledge Management Evolution in Early Modern Europe* [online]. Leiden, The Netherlands: Brill, 2016. ISBN 978-90-04-32525-8. Dostupné z DOI: 10.1163/9789004325258. [Citováno 8. 3. 2025].
8. OBSIDIAN. *Obsidian* [online]. 2025. Dostupné také z: https://obsidian.md/. [Citováno 8. 3. 2025].
9. ALEXW00. *Obsidian Note Linker* [online]. 2025. Dostupné také z: https://github.com/AlexW00/obsidian-note-linker. [Citováno 8. 3. 2025].
10. DXCORE35. *Batch Obsidian Forward Linker* [online]. 2025. Dostupné také z: https://github.com/dxcore35/obs_batch-linkr. [Citováno 8. 3. 2025].
11. BRIANPETRO. *Smart Connections: AI-Powered Note Connections v2.3* [online]. 2025. Dostupné také z: https://github.com/brianpetro/obsidian-smart-connections. [Citováno 8. 3. 2025].
12. BERGMANN, Dave; STRYKER, Cole. *What is vector embedding?* [Online]. 2025. Dostupné také z: https://www.ibm.com/think/topics/vector-embedding. [Citováno 8. 3. 2025].
13. OBSIDIAN. *Obsidian Help* [online]. 2025. Dostupné také z: https://help.obsidian.md/. [Citováno 8. 3. 2025].
14. MEM. *Mem.ai* [online]. 2025. Dostupné také z: https://get.mem.ai/. [Citováno 8. 3. 2025].
15. NOTION LABS, INC. *Notion* [online]. 2025. Dostupné také z: https://www.notion.com/. [Citováno 8. 3. 2025].
16. CUNFF, Anne-Laure Le. *How to use bi-directional links (backlinks) in Notion* [online]. 2022. Dostupné také z: https://nesslabs.com/notion-backlinks. [Citováno 8. 3. 2025].
17. REMNOTE, INC. *RemNote* [online]. 2025. Dostupné také z: https://www.remnote.com/. [Citováno 8. 3. 2025].
18. TABIBIAN, Behzad; UPADHYAY, Utkarsh; DE, Abir; ZAREZADE, Ali; SCHÖLKOPF, Bernhard; GOMEZ-RODRIGUEZ, Manuel. Enhancing human learning via spaced repetition optimization. *Proceedings of the National Academy of Sciences* [online]. 2019, roč. 116, č. 10, s. 3988–3993. Dostupné z DOI: 10.1073/pnas.1815156116. [Citováno 8. 3. 2025].
19. ANKI. *Anki* [online]. 2025. Dostupné také z: https://apps.ankiweb.net/. [Citováno 8. 3. 2025].
20. MDN WEB DOCS GLOSSARY. *URL* [online]. 2021. Dostupné také z: https://developer.mozilla.org/en-US/docs/Glossary/URL. [Citováno 8. 3. 2025].

21. MDN WEB DOCS. *What is a hyperlink?* [Online]. 2021. Dostupné také z: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content/Creating_links#what_is_a_hyperlink. [Citováno 8. 3. 2025].

22. ZWAKI, Shahnewaz. *12 Best Bi-directional Linking Note-Taking App (2021)* [online]. 2021. Dostupné také z: https://deskoflawyer.com/best-bi-directional-linking-note-taking-apps/. [Citováno 8. 3. 2025].

23. GOLIGHTLY, Erica. *What Is Bi-directional Linking?* [Online]. 2022. Dostupné také z: https://clickup.com/blog/bidirectional-linking/. [Citováno 8. 3. 2025].

24. BIZER, Christian; HEATH, Tom; BERNERS-LEE, Tim. Linked Data – The Story So Far. In: *Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web* [online]. 2023, s. 115–143. Dostupné z DOI: 10.1145/3591366.3591378. [Citováno 8. 3. 2025].

25. BERNERS-LEE, Tim. *Linked Data* [online]. 2006. Dostupné také z: https://www.w3.org/DesignIssues/LinkedData.html. [Citováno 8. 3. 2025].

26. ONTOTEXT. *What Are Linked Data and Linked Open Data?* [Online]. 2025. Dostupné také z: https://www.ontotext.com/knowledgehub/fundamentals/linked-data-linked-open-data/. [Citováno 8. 3. 2025].

27. DBPEDIA. *About DBPedia* [online]. 2025. Dostupné také z: https://www.w3.org/DesignIssues/LinkedData.html. [Citováno 8. 3. 2025].

28. KLÍMEK, Jakub. *Graph data formats: RDF, RDFS, Linked Data* [online]. 2022. Dostupné také z: https://docs.google.com/presentation/d/1p65vUsNHxDnY0D-nkUCiMUapigGlhPaLNoJL1pAmBWo/. [Citováno 8. 3. 2025].

29. LOGSEQ, INC. *How to Get Started With Networked Thinking and Logseq* [online]. 2025. Dostupné také z: https://blog.logseq.com/how-to-get-started-with-networked-thinking-and-logseq/. [Citováno 8. 3. 2025].

30. LOGSEQ, INC. *The basics of block references* [online]. 2025. Dostupné také z: https://docs.logseq.com/#/page/the%20basics%20of%20block%20references. [Citováno 8. 3. 2025].

31. MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. *Introduction to Information Retrieval* [online]. Cambridge, England: Cambridge University Press, 2009. ISBN 0521865719. Dostupné z DOI: 10.1017/CBO9780511809071. [Citováno 8. 3. 2025].

32. ABUBAKAR, Haisal Dauda; UMAR, Mahmood. Sentiment Classification: Review of Text Vectorization Methods: Bag of Words, Tf-Idf, Word2vec and Doc2vec. *SLU Journal of Science and Technology* [online]. 2022. ISSN 2736-0903. Dostupné také z: https://api.semanticscholar.org/CorpusID:250936672. [Citováno 8. 3. 2025].

33. CENTRUM ZPRACOVÁNÍ PŘIROZENÉHO JAZYKA MUNI. *Český stoplist* [online]. 2025. Dostupné také z: https://nlp.fi.muni.cz/cs/StopList. [Citováno 25. 4. 2025].

34. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. *Efficient Estimation of Word Representations in Vector Space* [online]. 2013. Dostupné z DOI: 10.48550/arXiv.1301.3781. [Citováno 8. 3. 2025].

35. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [online]. 2019. Dostupné z DOI: 10.48550/arXiv.1810.04805. [Citováno 8. 3. 2025].

36. REIMERS, Nils; GUREVYCH, Iryna. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks* [online]. 2019. Dostupné z DOI: 10.48550/arXiv.1908.10084. [Citováno 9. 4. 2025].

37. OPENAI. *Vector embeddings* [online]. 2025. Dostupné také z: https://platform.openai.com/docs/guides/embeddings. [Citováno 9. 4. 2025].

38. GOOGLE AI FOR DEVELOPERS. *Embeddings* [online]. 2025. Dostupné také z: https://ai.google.dev/gemini-api/docs/embeddings. [Citováno 9. 4. 2025].

39. JURAFSKY, Daniel; MARTI, James H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* [online]. Třetí edice. 2025. Dostupné také z: https://web.stanford.edu/~jurafsky/slp3/. Online rukopis vydán 12. ledna 2025. [Citováno 8. 3. 2025].

40. SCHWABER-COHEN, Roie. *Vector Similarity Explained* [online]. 2023. Dostupné také z: https://www.pinecone.io/learn/vector-similarity/. [Citováno 9. 4. 2025].

41. SPARX SYSTEMS. *Enterprise Architect* [soft.]. 2025. Dostupné také z: https://sparxsystems.com/. [Citováno 8. 5. 2025].

42. SOMMERWILLE, Ian. *Software Engineering* [online]. Desátá globální edice. Edinburgh Gate, Harlow, England: Pearson Education Limited, 2016. ISBN 1-292-09613-6. Dostupné také z: https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf. [Citováno 14. 4. 2025].

43. MICROSOFT LEARN. *Common web application architectures* [online]. 2023. Dostupné také z: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures. [Citováno 14. 4. 2025].
44. MDN WEB DOCS. *Client-server overview* [online]. 2025. Dostupné také z: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps/Client-Server_overview. [Citováno 14. 4. 2025].
45. MDN WEB DOCS. *MVC* [online]. 2023. Dostupné také z: https://developer.mozilla.org/en-US/docs/Glossary/MVC. [Citováno 14. 3. 2025].
46. THE PHP FOUNDATION. *PHP* [online]. 2025. Dostupné také z: https://www.php.net/. [Citováno 14. 3. 2025].
47. THE PHP DOCUMENTATION GROUP. *PHP Manual - What is PHP and what can it do?* [Online]. 2025. Dostupné také z: https://www.php.net/manual/en/introduction.php. [Citováno 14. 3. 2025].
48. LAZARCHUK, Andrii. *Top 10 projects developed with PHP technology* [online]. 2018. Dostupné také z: https://cybercraftinc.com/top-10-projects-developed-with-php-technology/. [Citováno 14. 3. 2025].
49. STACK OVERFLOW. *Developer Survey 2024 – Technology – Web frameworks and technologies* [online]. 2024. Dostupné také z: https://survey.stackoverflow.co/2024/technology#1-web-frameworks-and-technologies. [Citováno 14. 3. 2025].
50. SYMFONY. *Symfony* [online]. 2025. Dostupné také z: https://symfony.com/. [Citováno 14. 3. 2025].
51. THE PHP FRAMEWORK INTEROP GROUP. *PHP-FIG* [online]. 2025. Dostupné také z: https://www.php-fig.org/. [Citováno 14. 3. 2025].
52. WICKRAMASINGHE, Shanika. *Symfony vs Laravel: Battle of the PHP Frameworks* [online]. 2024. Dostupné také z: https://kinsta.com/blog/symfony-vs-laravel/. [Citováno 14. 3. 2025].
53. SYMFONY. *Twig – The flexible, fast, and secure template engine for PHP* [online]. 2025. Dostupné také z: https://twig.symfony.com/. [Citováno 14. 3. 2025].
54. AYEBOLA, Joan. *CSS Frameworks vs Custom CSS – What's the Difference?* [Online]. 2023. Dostupné také z: https://www.freecodecamp.org/news/css-frameworks-vs-custom-css/. [Citováno 14. 3. 2025].
55. SYMFONY. *Bootstrap 5 Form Theme* [online]. 2025. Dostupné také z: https://symfony.com/doc/current/form/bootstrap5.html. [Citováno 14. 3. 2025].

56. BOOTSTRAP. *Bootstrap* [online]. 2025. Dostupné také z: https://getbootstrap.com/. [Citováno 14. 3. 2025].

57. DOCKER, INC. *Docker* [online]. 2025. Dostupné také z: https://www.docker.com/. [Citováno 14. 3. 2025].

58. DOCKER, INC. *What is Docker?* [Online]. 2025. Dostupné také z: https://docs.docker.com/get-started/docker-overview/. [Citováno 14. 3. 2025].

59. FLYSYSTEM. *File Storage Abstraction for PHP* [online]. 2025. Dostupné také z: https://flysystem.thephpleague.com/docs/. [Citováno 14. 3. 2025].

60. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL* [online]. 2025. Dostupné také z: https://www.postgresql.org/. [Citováno 14. 3. 2025].

61. DOCTRINE. *Doctrine* [online]. 2025. Dostupné také z: https://www.doctrine-project.org/index.html. [Citováno 14. 3. 2025].

62. PGVECTOR. *Pgvector – open-source vector similarity search for Postgres* [online]. 2025. Dostupné také z: https://github.com/pgvector/pgvector. [Citováno 14. 3. 2025].

63. HUGGING FACE. *MTEB – Embedding Leaderboard* [online]. 2025. Dostupné také z: https://huggingface.co/spaces/mteb/leaderboard. [Citováno 14. 3. 2025].

64. DOCKER HUB. *Docker Hub* [online]. 2025. Dostupné také z: https://hub.docker.com/. [Citováno 14. 4. 2025].

65. TWIG DOCS. *markdowntohtml* [online]. 2025. Dostupné také z: https://twig.symfony.com/doc/3.x/filters/markdown_to_html.html. [Citováno 14. 4. 2025].

66. GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Design Patterns: Elements of Reusable Object-Oriented Software* [online]. Boston, United States: Addison-Wesley, 1995. ISBN 0-201-63361-2. Dostupné také z: https://www.javier8a.com/itc/bd1/articulo.pdf. [Citováno 14. 4. 2025].

67. JGRAPH LTD. *draw.io* [soft.]. 2025. Dostupné také z: https://www.drawio.com/. [Citováno 8. 5. 2025].

68. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *Controlling Text Search* [online]. 2025. Dostupné také z: https://www.postgresql.org/docs/current/textsearch-controls.html#TEXTSEARCH-PARSING-QUERIES. [Citováno 14. 4. 2025].

69. X3WIL. *Czech stemmer* [online]. 2017. Dostupné také z: https://github.com/x3wil/czech-stemmer. [Citováno 25. 4. 2025].

70. EVIDENTLY AI. *Normalized Discounted Cumulative Gain (NDCG) explained* [online]. 2025. Dostupné také z: https://www.evidentlyai.com/ranking-metrics/ndcg-metric. [Citováno 25. 4. 2025].

71. BROOKE, John. SUS: A quick and dirty usability scale. *Usability Eval. Ind.* [Online]. 1995, roč. 189. Dostupné také z: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale. [Citováno 14. 4. 2025].

72. SAURO, Jeff. *Measuring Usability with the System Usability Scale (SUS)* [online]. 2011. Dostupné také z: https://measuringu.com/sus/. [Citováno 7. 5. 2025].

Obsah příloh

```
/
├── readme.txt ..... stručný popis obsahu média
├── analyza-uzivatelskych-potreb/ ..... adresář se soubory k dotazníku
│   ├── aup-otazky.pdf ..... seznam otázek v dotazníku
│   └── aup-odpovedi.csv ..... veškeré odpovědi (CSV)
├── testovani/ ..... adresář se soubory k testování
│   ├── predvyber-metod/
│   │   └── testovaci-sada-poznamek/ .... poznámky z předvýběru metod
│   └── uzivatelske-testovani/
│       ├── testovaci-sada-poznamek/ ..... poznámky z už testování
│       ├── ut-otazky.pdf ..... otázky v uživatelském testování
│       └── ut-odpovedi.csv ..... veškeré odpovědi (CSV)
├── thinklink/
│   ├── src/ ..... zdrojové kódy implementace
│   └── README.md ..... popis spuštění projektu
└── text/ ..... zdrojová forma práce ve formátu LaTeX
```