FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Backend k interaktivní mapě průchodu studiem na FIT ČVUT
Student:	Raian Samerkhanov
Vedoucí:	doc. Ing. Robert Pergl, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Zadání práce navazuje na úspěšně obhájenou diplomovou práci, v rámci které vznikl návrh vizualizací zpřístupňující přehledným způsobem problematiku průchodu studiem na FIT ČVUT. Výsledkem práce byl prototyp webové aplikace, která však pracovala pouze nad mockup daty. Cílem této práce je vytvořit serverovou část (backend), která umožní nasazení celého projektu.

1. Seznamte se s uvedenou diplomovou prací a analyzujte požadavky na backend pro ni.
2. Vyberte vhodné technologie pro vývoj backendu -- databáze a REST API -- na platformě JVM.
3. Proveďte návrh a implementaci řešení.
4. Naplňte databázi dostupnými daty.
5. Backend otestujte a napojte frontendovou část.
6. Výsledky zdokumentujte.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 31. října 2023 v Praze.

Bakalářská práce

BACKEND K INTERAKTIVNÍ MAPĚ PRŮCHODU STUDIEM NA FIT ČVUT

Raian Samerkhanov

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: doc. Ing. Robert Pergl, Ph.D.
16. května 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Rád bych poděkoval mému vedoucímu doc. Ing. Robertu Perglovi, Ph.D za odborné vedení a konzultace k této práci. Dále děkuji své rodině za podporu během mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 16. května 2024

Abstrakt

Cílem této bakalářské práce je vývoj backendové části webové aplikace „interaktivní mapa studijního postupu na FIT ČVUT“. V rámci práce byla provedena analýza požadavků na serverovou část, vybrány a použity technologie na platformě JVM, včetně Spring Boot pro implementaci REST API. Výsledkem je spolehlivá a škálovatelná serverová část, která zajišťuje efektivní zpracování a správu dat ve webové aplikaci.

Klíčová slova FIT, ČVUT, sylabus, studijní předměty, backend, java, spring boot

Abstract

The aim of this bachelor's thesis is the development of the backend part for the web application "interactive map of study progress at FIT CTU". The work involved analyzing the requirements for the server part, selecting and using technologies on the JVM platform, including Spring Boot for implementing the REST API. The result is a reliable and scalable server part that ensures efficient processing and management of data in the web application.

Keywords FIT, CTU, sylab, courses, backend, java, spring boot

Seznam zkratek

AOP	Aspect Oriented Programming
CRUD	Creat Read Update Delete
CSRF	Cross-Site Request Forgery
IoC	Inversion of Control
JBDC	Java DataBase Connectivity
JPA	Java Persistence API
JVM	Java Virtual Machine
JWT	JSON Web Token
MSW	Mock Service Worker
REST API	Representational State Transfer Application programming interface
XSS	Cross-Site Scripting

Úvod

Digitalizace a informační technologie jsou již dlouhodobě integrovanou součástí akademického světa. Vysoké školy a univerzity neustále hledají způsoby, jak tyto nástroje využívat co nejefektivněji pro zlepšení studijního prostředí a podpory studentů. Jedním z příkladů takového úsilí je projekt interaktivní mapy průchodu studiem na Fakultě informačních technologií Českého vysokého učení technického v Praze (FIT ČVUT), který si klade za cíl usnadnit studentům orientaci ve studijních programech a plánech. Projekt však stále čelí výzvám, jako je například nedostatek plně funkčního backendu, což komplikuje jeho kompletní nasazení a praktické využití.

Významnost tohoto projektu spočívá v potenciálu značně ulehčit studijní plánování pro studenty FIT ČVUT, což přináší přínos nejen pro současné, ale i budoucí generace studentů. Tato bakalářská práce přímo navazuje na dřívější úspěšně obhájenou diplomovou práci Bc. Kláry Matouškové, která se zaměřila na vizualizace a první kroky k realizaci této myšlenky, ale byla limitována na práci s neautentickými (mockup) daty.

Motivací pro zvolení tohoto tématu bylo uvedené omezení a potřeba vyvinout robustní řešení, které by mapu učinilo plně funkční a přínosnou pro akademickou komunitu. Tím se tato práce stává nejen pokračováním předchozích snah, ale i důležitým krokem k realizaci celkové vize interaktivní mapy průchodu studiem.

Cíle práce a metodika

Hlavním cílem této bakalářské práce je navrhnout a implementovat serverovou část (backend) pro interaktivní mapu průchodu studiem na FIT ČVUT, což umožní plnohodnotné nasazení a integraci celého projektu s reálnými daty. Práce navazuje na úspěšně obhájenou diplomovou práci Bc. Kláry Matouškové[1] a jejím záměrem je překonat omezení předchozího prototypu tím, že poskytne dynamické zpracování dat. Klíčové úkoly zahrnují:

Teoretická část:

1) Důkladné seznámení se s existujícím návrhem interaktivní mapy: Podrobné seznámení s předchozí diplomovou prací, která představila koncept interaktivní mapy, včetně její struktury, funkcionalit a omezení. To zahrnuje analýzu požadavků na backend systému, aby bylo možné adekvátně navrhnout jeho architekturu a funkcionalitu.

2) Výběr vhodných technologií pro vývoj backendu: Provedení rešerše a analýzy dostupných technologií na platformě JVM, které by byly nejvhodnější pro vývoj backendu, včetně databázových systémů a REST API.

Praktická část:

1) Návrh a implementace serverové části: Tento proces zahrnuje vytvoření databázových schémat, vývoj API endpointů a zajištění integrace s frontendovou částí. Rozšíření funkčnosti již existujícího frontendu.

2) Naplnění databáze aktuálními daty: Naplní databázi reálnými daty.

3) Testování backendu a integrace s frontendovou částí: Provedení testování funkcionalit backendu. Součástí je i testování integrace s frontendovou částí interaktivní mapy, aby byla zajištěna bezproblémová spolupráce obou částí systému a optimální uživatelská zkušenost.

4) Dokumentace vývoje a použitých technologií: Celý proces vývoje, od analýzy požadavků přes výběr technologií, návrh, implementaci a testování, bude detailně zdokumentován. Tato dokumentace poskytne přehled o použitých technologiích, architektuře systému, a také návod pro další vývoj a údržbu projektu.

Výsledkem této práce bude plně funkční a integrovaný systém interaktivní mapy průchodu studiem, který bude sloužit jako důležitý nástroj pro podporu studijní orientace studentů FIT ČVUT. Projekt si klade za cíl nejen zlepšit navigaci v studijních plánech, ale také motivovat studenty k lepšímu plánování jejich akademické kariéry a pomoci jim dosáhnout jejich vzdělávacích cílů efektivněji.

Část I

Teoretická část

Kapitola 1
Sylabus

Tato kapitola je věnována analýze struktury a obsahu Syllabu na Fakultě informačních technologií Českého vysokého učení technického v Praze (FIT ČVUT). Hlavním zdrojem pro výzkum je Bílá kniha ČVUT, která poskytuje podrobné údaje o akreditaci a obsahu vzdělávacích programů [2].

1.1 Studijní program

Studijní program je strukturovaný soubor vzdělávacích předmětů, postupů a požadavků, které musí studenti splnit, aby získali titul nebo kvalifikaci v určité oblasti znalostí. V rámci FIT CVUT lze programy rozdělit na úrovně:

Bakalářské studium Tento základní stupeň vysokoškolského vzdělání. Studium obvykle trvá tři roky a je označováno písmenem B.

Magisterské studium Tento stupeň je určen pro ty, kteří pokračují ve vzdělávání po získání bakalářského titulu. Délka studia činí rovněž tři roky a je označena písmenem M, což odráží systém nové akreditace.

Doktorandské studium Nejvyšší akademický stupeň, pro který je vyžadováno čtyřleté studium, je symbolizován písmenem P.

Bakalářské i magisterské studium je nabízeno v českém i anglickém jazyce, což rozšiřuje možnosti pro zahraniční studenty a podporuje mezinárodní výměnu znalostí.

1.2 Katedra

Odpovědnost za specifické studijní zaměření a předměty je svěřena jednotlivým *katedrám*, které se specializují na určité tematické oblasti. Každá katedra má svého vedoucího, který zodpovídá za kvalitu a obsah vyučovaných předmětů a slouží jako hlavní kontaktní bod pro zaměstnance fakulty. Učitelé a další akademický personál jsou klíčovými členy kateder a aktivně přispívají k rozvoji studijního programu. Fakulta informačních technologií dnes zahrnuje následující katedry:

- **KTI** - katedra teoretické informatiky
- **KSI** - katedra softwarového inženýrství
- **KCN** - katedra číslicového návrhu

- **KIB** - katedra informační bezpečnosti
- **KPS** - katedra počítačových systémů
- **KAM** - katedra aplikované matematiky

1.3 Specializace

Každý studijní program také zahrnuje různé *specializacemi* – jedná se o specifické oblasti v rámci široké disciplíny, které studentům umožňují soustředit se na konkrétní témata nebo profesní směry. Specializace pomáhají studentům rozvíjet hlubší a specifické dovednosti v oblastech, které je zajímají.

Každá specializace v akademickém programu univerzity je úzce spojena s konkrétní katedrou, která je zodpovědná za její obsah a kvalitu vzdělávání. Katedra může vést jednu nebo více specializací a poskytuje rozmanitost a hloubku učení v příslušné oblasti znalostí. To umožňuje fakultám navrhovat studijní plány, organizovat předměty a vědecký výzkum tak, aby co nejlépe vyhovovaly zájmům a potřebám studentů, kteří si vybrali konkrétní specializaci.

Katedra	Specializace studijního programu Informatika
KIB	Informační bezpečnost 2021 (BI-IB21) – bakalářský
KSI	Webové inženýrství 2021 (BI-WI21) – bakalářský
KSI	Manažerská informatika 2021 (BI-MI21) – bakalářský
KSI	Počítačová grafika 2021 (BI-PG21) – bakalářský
KCN	Počítačové inženýrství 2021 (BI-PI21) – bakalářský
KPS	Počítačové sítě a Internet 2021 (BI-PS21) – bakalářský
KPS	Počítačové systémy a virtualizace 2021 (BI-PV21) – doktorský
KSI	Softwarové inženýrství 2021 (BI-SI21) – bakalářský
KTI	Teoretická informatika 2021 (BI-TI21) – bakalářský
KAM	Umělá inteligence 2021 (BI-UI21) – bakalářský

Tabulka 1.1 Specializace studijních programů informatiky

1.4 Studijní plán

Studijní plán je seznam předmětů, které musí student absolvovat na univerzitě, aby dokončil své studium ve zvoleném oboru. Každý obor má alespoň jeden studijní plán. Studijní plán se dělí na dva typy podle formy studia:

Prezenční Fyzická účast studenta ve škole.

Kombinovaný Smíšená forma výuky, zahrnující jak prezenční, tak i online výuku.

1.4.1 Skupina předmětů

Studijní plány zahrnují různé *skupiny předmětů*, které mají své specifické účely v rámci celkového vzdělávacího procesu studenta. Tyto skupiny předmětů jsou následovně kategorizovány:

- **PE** - Povinné ekonomické
- **PJ** - Povinná zkouška z angličtiny
- **PO** - Povinné předměty oboru

- **PP** - Povinné předměty programu
- **PS** - Povinné předměty specializace
- **PT** - Povinná tělesná výchova, sportovní předměty
- **PV** - Povinně volitelné předměty
- **PZ** - Povinně předměty zaměření
- **V** - Volitelné předměty
- **VE** - Povinně volitelné ekonomicko-manažerské
- **VH** - Povinně volitelné humanitní
- **VO** - Volitelné předměty oboru/specializace

1.4.2 Doporučený průchod

Doporučený průchod je předem stanovená posloupnost v osnově studijního plánu, poskytující optimální řazení předmětů. Je navržen tak, aby zajišťoval logické následování dovedností a znalostí, kde každý následující předmět staví na předcházejících.

1.5 Předmět

Studijní plán se skládá z mnoha *předmětů* a konkrétní předmět může být součástí několika studijních plánů. Každý předmět v rámci studijního plánu hraje svou roli, což určuje jeho důležitost ve vzdělávacím procesu. Některé předměty jsou povinné v určitých plánech, ale mohou být volitelné v jiných.

1.5.1 Kod předmětu

Kód předmětu je specifický identifikátor používaný vzdělávacími institucemi k jednoznačné identifikaci předmětu. Tento kód je vlastně zkratkou, která shrnuje základní informace o předmětu, jako je typ programu, ke kterému patří (B, N, D), nebo forma studia (prezenční nebo kombinovaná), a vyučovací jazyk (český nebo anglický).

1.5.2 Semestr výuky

Semestr obsahuje celkem 13 výukových týdnů, z nichž 7 týdnů je lichých a 6 sudých. Podle kalendáře je první týden výuky lichý[3]. Vzdělávací instituce dělí akademický rok na zimní a letní semestr. Zimní semestr obvykle začíná ve druhé polovině září a končí na konci prosince, zatímco letní semestr začíná ve druhé polovině února a končí ve druhé polovině května. Některé předměty jsou nabízeny pouze během jednoho semestru, zatímco jiné se mohou vyučovat během obou. Například předmět BI-TDP.21 a BI-BPR.21 mohou být nabízeny jak v zimním, tak v letním semestru, což studentům poskytuje flexibilitu v plánování jejich akademického rozvrhu.

1.5.3 Kredity

Po úspěšném dokončení předmětu student získává za tento předmět *kredity*. Za celé studium student musí získat celkem 180 kreditů. Během studia se student setká s řadou etap, v nichž musí dosáhnout minimálního množství kreditů, což je podmínkou pro další postup ve studiu[4].

DOBA STUDIA	POČET KREDITŮ
ZA 1. SEMESTR	15
ZA 1. AKADEMICKÝ ROK (2 SEMESTRY)	30
ZA KAŽDÝ DALŠÍ AKADEMICKÝ ROK (2 SEMESTRY)	40

Tabulka 1.2 Počet kreditů potřebných za určité období studia.

1.6 Weby

Současné studijní programy poskytují studentům bohaté množství informací, od akademických předmětů a studijních plánů po rozvrh hodin a nejnovější zprávy univerzity. K úspěšné orientaci v tomto informačním prostoru univerzita nabízí řadu webových zdrojů, které se stávají neocenitelným nástrojem jak pro studenty, tak pro vyučující. V této práci se podívám na některé z těchto webových stránek a jejich význam pro vzdělávací proces.

1.6.1 KOS

Systém komponenta studium [5] (dále jen KOS) představuje klíčový nástroj pro každého studenta. Je to platforma, kde studenti se zapisují do předmětů a plánují svůj studijní plán. To umožňuje přehledně spravovat své studijní povinnosti. Stránka také slouží pro sestavení individuálního rozvrhu hodin, což studentům usnadňuje koordinaci a optimalizaci jejich akademického dne. Další nezbytnou funkcí je možnost zápisu na zkoušky, což umožňuje studentům efektivně plánovat svůj studijní postup a spravovat termíny zkoušek. Kromě toho stránka obsahuje podrobné informace o všech nabízených předmětech, jako :

- typ programu
- vyučovací jazyk
- forma zakončení předmětu
- počet kreditů
- rozsah
- seznam vyučujících a jejich role

Informace o obsahu předmětů, včetně témat, která budou probírána, jsou také dostupné, což studentům pomáhá lépe se připravit na akademický rok.

1.6.2 FIT ČVUT Course Pages

Pro podrobnější seznámení se s předmětem slouží webová stránka *FIT ČVUT Course Pages* [6]. Tyto stránky poskytují studijní materiály, včetně obsahu přednášek a cvičení. Kromě toho slouží jako platforma pro oznámení aktuálních změn, které mohou nastat během semestru.

1.6.3 Usermap

Systém *Usermap* [7] spravuje identity zaměstnanců a studentů ČVUT, umožňuje vyhledávání osob spojených s univerzitou a poskytuje přístup k detailům jejich profilů.

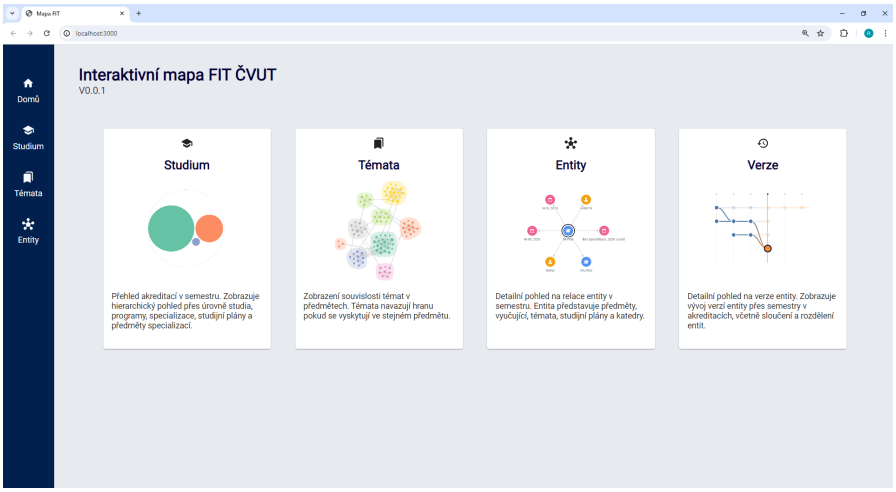
Kapitola 2
Analýza

2.1 Seznámení s web front-end částí aplikace

Po důkladném prostudování studijního programu FIT ČVUT se stalo důležitým krokem seznámit se a analyzovat front-end část aplikace. Tato část aplikace hraje klíčovou roli ve vizualizaci studijních plánů a kurzů, poskytující studentům pohodlný nástroj pro navigaci a plánování akademické činnosti.

Pro hluboké porozumění interakci uživatelského rozhraní a serverové části jsem provedl analýzu klíčových prvků rozhraní. Také jsem zvážil a přinesl příklady snímků hlavních obrazovek aplikace, což umožní lépe popsat požadavky na back-end.

2.1.1 Domovská stránka



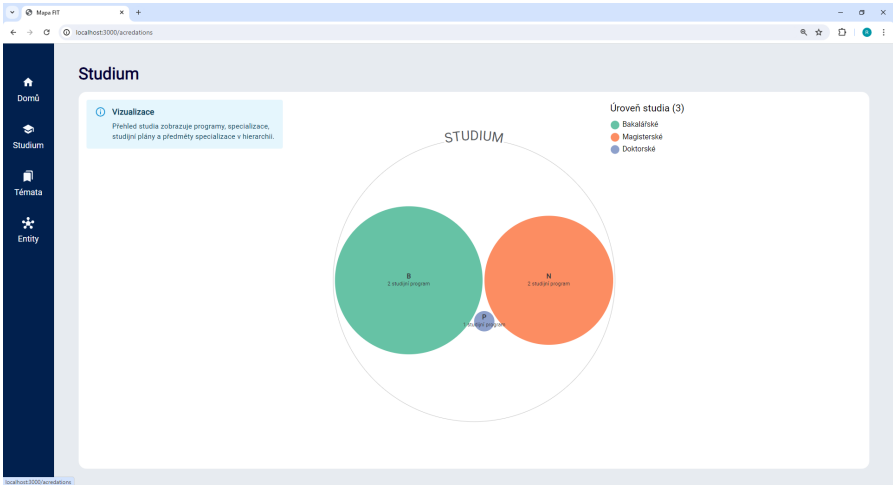
Obrázek 2.1 Domovská stránka interaktivní mapy FIT ČVUT.

Úvodní stránka interaktivní mapy FIT ČVUT (obrázek 2.1) slouží jako výchozí bod pro uživatele, poskytuje přehled a přístup k hlavním funkčním částem aplikace. Na snímku obrazovky jsou vidět čtyři hlavní moduly: „Studium“, „Témata“, „Entity“ a „Verze“, z nichž každý je vizuálně reprezentován ikonami a krátkým popisem.

- Modul „Studium“. Přehled akreditací v semestru. Zobrazuje hierarchický pohled přes úrovně studia, programy, specializace, studijní plány a předměty specializací.
- Modul „Témata“. Zobrazení souvislosti témat v předmětech. Témata navazují hranu pokud se vyskytují ve stejném předmětu.
- Modul „Entity“. Detailní pohled na relace entity v semestru. Entita představuje předměty, vyučující, témata, studijní plány a katedry.
- Modul „Verze“. Detailní pohled na verze entity. Zobrazuje vývoj verzí entity přes semestry v akreditacích, včetně sloučení a rozdělení entit.

Každý modul je určen k vizualizaci specifických dat a umožňuje uživatelům hlouběji proniknout do struktury a obsahu studijních materiálů.

2.1.2 Studium



Obrázek 2.2 Stránka akreditací interaktivní mapy FIT ČVUT.

Na stránce akreditací (obrázek 2.2) jsou uživatelům prezentovány studijní programy nabízené na fakultě. Tato sekce umožňuje prozkoumat dostupné směry vzdělávání a jejich strukturu. Studijní programy jsou vizualizovány ve formě kruhových diagramů, kde každý program je klikatelný a při interakci odhaluje další podrobnosti o specializacích. Při výběru konkrétní specializace se otevře nová úroveň informací, kde jsou zobrazeny studijní plány.

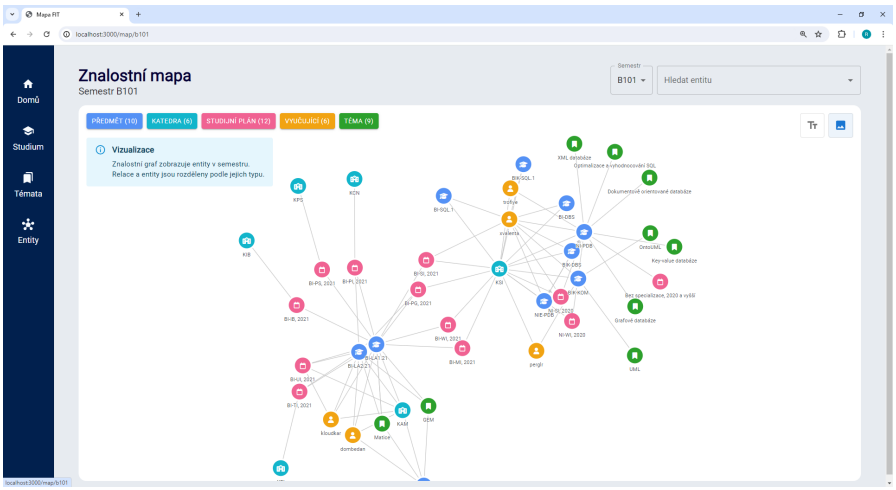
2.1.3 Znalostní mapa

Stránka „Znalostní mapa“ (obrázek 2.3) v interaktivní mapě FIT ČVUT představuje vizualizaci ve formě grafu, která spojuje různé entity vzdělávacího procesu, jako jsou :

- předměty
- katedry
- studijní plány
- vyučující

- témata

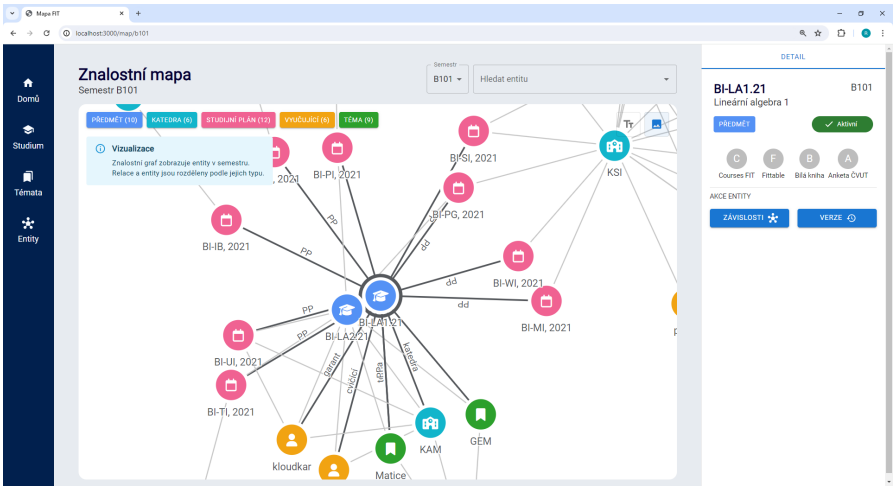
a zároveň ukazuje vztahy mezi nimi.



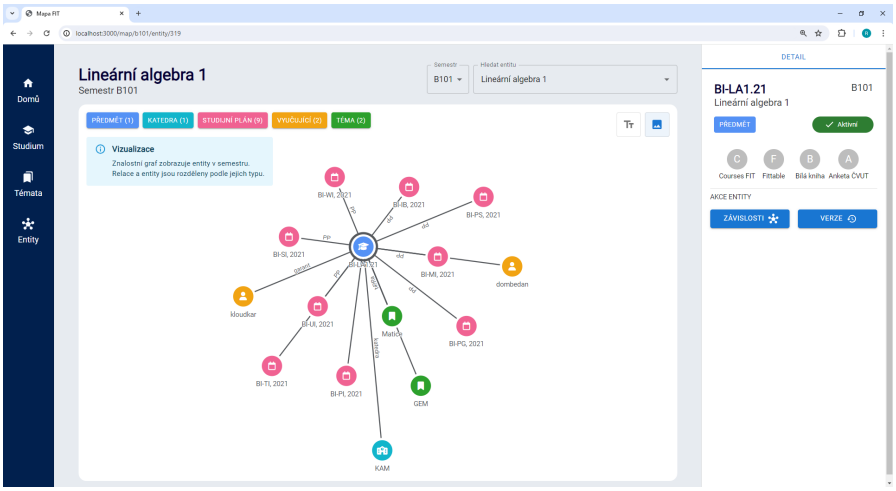
Obrázek 2.3 Znalostní mapa interaktivní mapy FIT ČVUT.

Na horním panelu jsou umístěna tlačítka pro výběr kategorií, což uživatelům umožňuje vybírat zobrazení typů entit. Toto uživateli umožňuje zobrazovat pouze ty informace, které potřebuje, což zlepšuje vzhled mapy a její čitelnost.

Všechny prvky na mapě jsou interaktivní: kliknutím na prvek se zobrazí další informace (obrázek 2.4) o něm , jako jsou s ním spojené kurzy, katedry nebo studijní plány. V pravé části obrazovky se zobrazí podrobné informace o objektu, odkazy na vzdělávací zdroje a tlačítka „Závislosti“ a „Verze“. Po kliknutí na tlačítko „Závislosti“ se zobrazí objekt a všechny jeho vazby na ostatní objekty (obrázek 2.5) .

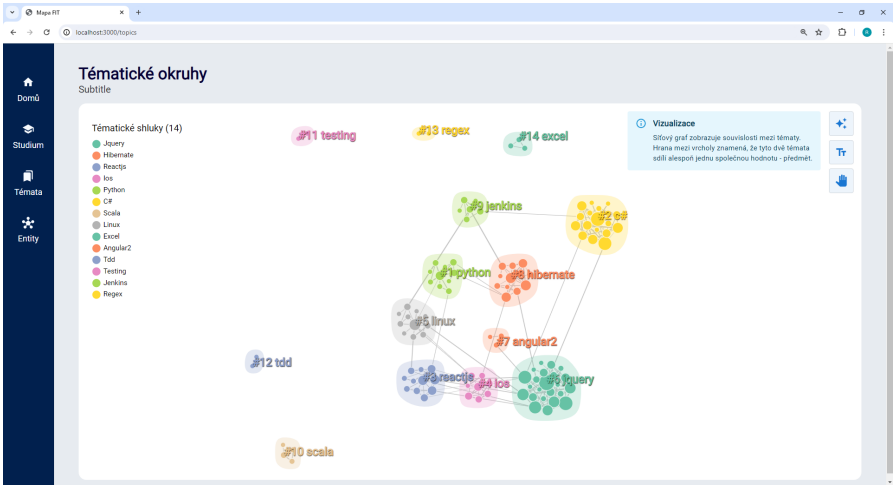


Obrázek 2.4 Informace entity



Obrázek 2.5 Zobrazení závislostí entity

2.1.4 Tématické okruhy

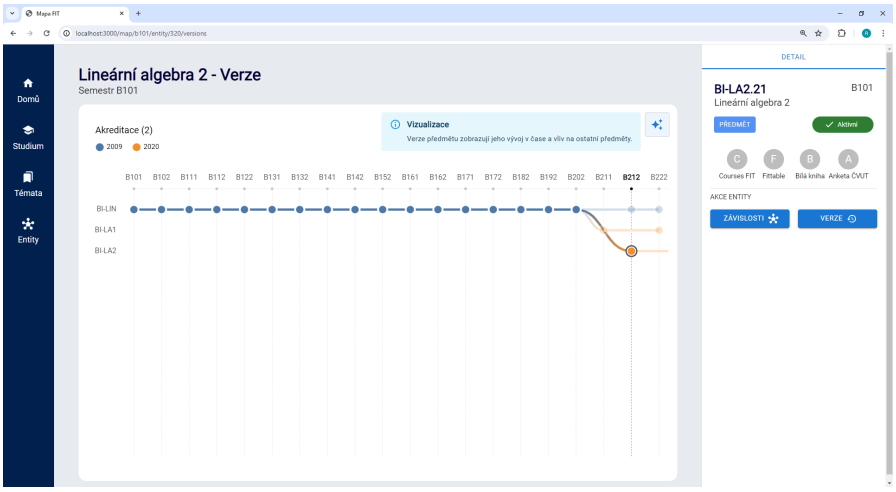


Obrázek 2.6 Mapa tématických okruhu

Na stránce „Tématické okruhy“ (obrázek 2.6) interaktivní mapy FIT ČVUT je prezentována vizualizace vazeb mezi vzdělávacími tématy. Každý tematický klastr je zobrazen kruhem určité barvy, což ho vizuálně odlišuje od ostatních klastrů. Uživatelé mohou kliknout na klastr, aby viděli podrobnější informace o tématech, která do něj patří, a jejich vzájemných vazbách.

2.1.5 Verze

Na stránce „Verze“ (obrázek 2.7) interaktivní mapy FIT ČVUT je prezentován vývoj vzdělávacích předmětů ve formě časové osy. Každý sloupec osy odpovídá určitému semestru. Verze předmětů jsou seskupeny podle roku akreditace a různé barvy označují příslušnost každé verze k určité skupině.



Obrázek 2.7 Zobrazení verzí předmětů BI-LA2.21

2.2 Výběr technologií pro vývoj backendu na platformě JVM

Ve vývoji moderního softwaru je výběr technologií pro backend rozhodující. Efektivní řešení ovlivňuje rychlost vývoje, zjednodušení podpory, bezpečnost, výkon a škálovatelnost aplikací. Nejběžnější platformou pro vytváření backendu webových aplikací zůstává Java Virtual Machine (JVM), díky jejím silným možnostem a rozsáhlé komunitě vývojářů. Tato kapitola představuje podrobný přehled klíčových technologií pro vytváření REST API a správu databází na této platformě.

2.2.1 Výběr technologií pro vývoj REST API

V rámci své bakalářské práce jsem se zaměřil na studium a srovnání několika populárních technologií pro vývoj REST API na platformě JVM. Konkrétně budou zváženy následující technologie:

- Spring Boot
- Micronaut
- Quarkus
- Vert.x

2.2.1.1 Spring Boot

„*Spring Boot* usnadňuje vytváření samostatných, produkčně kvalitních aplikací založených na Springu, které můžete „jednoduše spustit“. Zaujímáme stanovisko k platformě Spring a knihovnám třetích stran, abyste mohli začít s minimálními obtížemi. Většina aplikací Spring Boot vyžaduje minimální konfiguraci Springu.“[8] Poskytuje řadu „starterů“, které jsou předem nakonfigurovány pro automatizaci nastavení různých technologií, jako jsou databáze a systémy pro výměnu zpráv. To umožňuje vývojářům vyhnout se mnoha běžným potížím při nastavování a konfiguraci projektů. Funkce Spring Boot:

Automatická konfigurace Spring Boot se snaží automaticky nakonfigurovat vaši aplikaci na základě přidaných závislostí v projektu, což snižuje potřebu ručního nastavení.

Vestavěné aplikační servery Spring Boot obsahuje vestavěné servery, jako jsou Tomcat a Jetty, které nevyžadují samostatnou instalaci a nastavení, což umožňuje rychlé spuštění webových aplikací.

Rozsáhlá dokumentace a podpora komunity Díky velké popularitě Spring Boot mají vývojáři přístup k rozsáhlé dokumentaci a komunitě, což usnadňuje řešení jakýchkoli vzniklých problémů.

2.2.1.2 Micronaut

„Framework *Micronaut* je moderní, založený na JVM, full-stack Java framework navržený pro stavbu modulárních, snadno testovatelných JVM aplikací s podporou pro Java, Kotlin a Groovy.“ [9] Micronaut je navržen tak, aby poskytoval komplexní sadu nástrojů potřebných pro vývoj aplikací na platformě JVM. To zahrnuje techniky jako:

- *Dependency Injection* and *Inversion of Control (IoC)*
- *Aspect Oriented Programming (AOP)*

Micronaut snaží vyhnout běžným komplikacím spojeným s platformami jako Spring a Grails. Například nabízí rychlé spouštění, sníženou paměťovou stopu, jednoduché jednotkové testování.

2.2.1.3 Quarkus

„ *Quarkus* je Kubernetes-nativní Java framework přizpůsobený pro GraalVM a HotSpot, který je vytvořen z nejlepších Java knihoven a standardů.“[10]

Vývoj s podporou živého načítání (live coding) Aktualizace zdrojových kódů, zdrojů a konfigurací aplikace okamžitě odráží změny v běžící aplikaci, což usnadňuje vývoj zahrnující uživatelské rozhraní a databázi, umožňující okamžité zobrazení změn. [11]

Imperativní a reaktivní programování Quarkus je postaven na reaktivním jádru, což umožňuje aplikaci kombinovat reaktivní a imperativní komponenty, podporující cestu k reaktivnímu programování.[12]

2.2.1.4 Vert.x

Eclipse *Vert.x* je sada nástrojů pro vytváření reaktivních aplikací na JVM, známá svou škálovatelností a odolností. „Reaktivní aplikace jsou navrženy tak, aby efektivně zvládaly velké pracovní zatížení a udržovaly odezvu navzdory selháním. Vert.x nabízí rozsáhlý ekosystém zahrnující webový stack, reaktivní ovladače databází, zasílání zpráv, proudy událostí, clustering, metriky a distribuované sledování. Podporuje různé asynchronní programovací modely, včetně zpětných volání, promises/futures, RxJava a Kotlin coroutines.“ [13]

2.2.2 PostgreSQL

„*PostgreSQL* je výkonná, otevřená a bezplatná relační databáze s podporou SQL a mnoha pokročilými funkcemi.“[14]

PostgreSQL se vyznačuje svou rozšiřitelností, spolehlivostí a podporou mnoha funkcí pro vývojáře a správce. Systém je zdarma, podporuje uživatelsky definované datové typy a funkce, funguje na všech hlavních operačních systémech. PostgreSQL je ideální pro projekty jakékoli velikosti, nabízí pokročilé možnosti, jako jsou geoprostorová rozšíření, víceúrovňové správě transakcí a rozsáhlé bezpečnostní možnosti.

2.2.3 Integrace a bezpečnost

V této části prozkoumáme dva klíčové aspekty vývoje bezpečného a integrovaného backendu: použití Spring Data pro práci s databázemi a aplikaci Spring Security pro zajištění bezpečnosti webových aplikací.

2.2.3.1 Spring Data JPA

Spring Data JPA je modul v rámci rozsáhlé rodiny Spring Data, který usnadňuje integraci technologie *Java Persistence API* (JPA) do aplikací na platformě Spring. Zjednodušuje proces vytváření JPA repozitářů, což umožňuje vývojářům soustředit se na hlavní logiku aplikace a minimalizovat potřebu psaní šablonového kódu pro provádění dotazů na databáze.[15]

Repositáře bez kódu: Spring Data JPA automatizuje vytváření repositářů, eliminuje potřebu psaní šablonového kódu pro standardní CRUD operace. To významně snižuje množství kódu, který musí vývojáři napsat a udržovat. [16]

Redukce šablonového kódu: Spring Data JPA poskytuje standardní implementace pro každou metodu definovanou v rozhraních repositářů, což usnadňuje základní operace čtení a zápisu dat. [16]

Automaticky generované dotazy: Tato funkčnost umožňuje automaticky vytvářet dotazy do databáze na základě názvů metod v rozhraních repositářů, což zjednodušuje proces psaní dotazů.[16]

2.2.3.2 Spring Security

„*Spring Security* je framework, který se zaměřuje na poskytování autentizace a autorizace pro aplikace Java. Stejně jako u všech projektů Spring, skutečná síla Spring Security spočívá v tom, jak snadno lze rozšířit pro splnění specifických požadavků.“[17] Základními aspekty bezpečnosti aplikací jsou autentizace a autorizace (neboli kontrola přístupu). Tyto dvě klíčové oblasti jsou cílem Spring Security.[18]

Autentizace pomáhá potvrdit, že subjekt je skutečně tím, za koho se vydává, ať už jde o uživatele, zařízení nebo systém působící v aplikaci.

Autorizace určuje, zda má subjekt právo provádět určité akce v aplikaci. Identifikace subjektu musí být potvrzena v procesu autentizace, než bude rozhodnuto o autorizaci.

2.3 Požadavky

2.3.1 Funkční požadavky

- **F1 – Zobrazování informací bez přihlášení**
  - **Popis:** Systém poskytne široký přístup k informacím o studijních programech, tematických okruzích, mapě znalostí, informacích o objektech na mapě znalostí a verzích předmětů bez nutnosti přihlášení.
  - **Priorita:** Vysoká
  - **Složitost:** Střední
  - **Poznámka:** Umožní rychlý a snadný přístup k informacím pro všechny uživatele.
- **F2 – Správa dat administrátorem**
  - **Popis:** Administrátor bude moci přidávat, upravovat a mazat informace prostřednictvím zabezpečeného administračního rozhraní.

- **Priorita:** Vysoká
- **Složitost:** Střední
- **Poznámka:** Zabezpečené rozhraní zaručí, že pouze autorizované osoby mohou měnit důležité informace.

- **F3 – Autorizace a autentizace administrátora**
  - **Popis:** Systém bude podporovat robustní mechanismy pro ověření totožnosti a autorizaci administrátora k ochraně přístupu k datům.
  - **Priorita:** Vysoká
  - **Složitost:** Střední
  - **Poznámka:** Zabezpečení přístupu je nezbytné pro ochranu před neautorizovanými změnami.

2.3.2 Nefunkční požadavky

- **NF1 – Bezpečnost**
  - **Popis:** Backend systému musí zabezpečit všechna data a operace prostřednictvím bezpečných protokolů, autentizace a kontroly přístupu.
- **NF2 – Výkon**
  - **Popis:** Backend musí rychle zpracovávat příchozí požadavky a efektivně spravovat velký objem dat, aby zajistil rychlou odezvu systému i při vysokém zatížení.
- **NF3 – Spolehlivost**
  - **Popis:** Backend musí garantovat stabilní a nepřetržitý provoz systému, minimalizovat pravděpodobnost a dopady výpadků.
- **NF4 – Škálovatelnost**
  - **Popis:** Backend by měl podporovat možnost škálování v závislosti na změnách zatížení, aby systém mohl reagovat na růst počtu uživatelů a dat.
- **NF5 – Vzájemná interakce s Front-endem**
  - **Popis:** Backend by měl poskytovat stabilní a dobře zdokumentované API, které umožňuje front-endu efektivně komunikovat se systémem, zajišťující konzistenci a přesnost dat.

Část II

Praktická část

Kapitola 3
Návrh

3.1 Use Case

Use Case diagram je diagram, který ukazuje, kdo a jak bude systém používat. Jedná se o jednoduchý způsob, jak vizuálně ukázat, jaké akce lze v programu vykonat a kdo za ně nese odpovědnost. V rámci této práce jsem popsal scénáře pro backend a zobrazil use case diagramy, které byly dříve popsány v diplomové práci o frontendové části. Dále jsem předložil tabulku mapování scénářů, která ukazuje vztah mezi frontedovými a backendovými funkcemi.

3.1.1 Aktéři

V rámci mé práce je jediným aktérem Server, protože veškeré zpracování dat probíhá na serverové straně.

Server je účastník systému, který zpracovává požadavky od klientů a provádí operace s daty. Server je zodpovědný za provádění obchodní logiky, zabezpečení a integrity dat. Také spravuje chyby.

3.1.2 Znalostní mapa

3.1.2.1 UC1: Předání dat znalostní mapy

Aktéři: Server
Počáteční podmínka: Server přijímá a zpracovává `HTTP GET` požadavky na endpoint `/api/map/semesterID`.
Scénář:

1. Server přijme požadavek na zobrazení mapy.
2. Server získá data z databáze.
3. Data jsou serializována do formátu JSON.
4. Server odešle data zpět klientovi jako odpověď na požadavek.

3.1.2.2 UC2: Přidání entity/relace

Aktéři: Server
Počáteční podmínka: Server přijímá a zpracovává `HTTP POST` požadavky na endpoint `/api/entity` pro entity a `/api/map/links` pro relace.
Scénář:

1. Server přijme požadavek na vytvoření nové entity nebo relace.
2. Server ověří poskytnutá data.
3. Nová entita nebo relace je vytvořena v databázi.
4. Server vrací odpověď s úspěchem a detaily vytvořené entity nebo relace.

Alternativní scénář:

1. Server přijme požadavek na vytvoření nové entity nebo relace
2. Server ověří poskytnutá data a zjišťuje chyby.
3. Server vrací chybu při vytváření entity nebo relace.

3.1.2.3 UC3: Smazání entity/relace

Aktéři: Server
Počáteční podmínka: Server přijímá a zpracovává `HTTP DELETE` požadavky na endpoint `/api/entities/{id}` pro entity a `/api/map/links/{id}` pro relace.
Scénář:

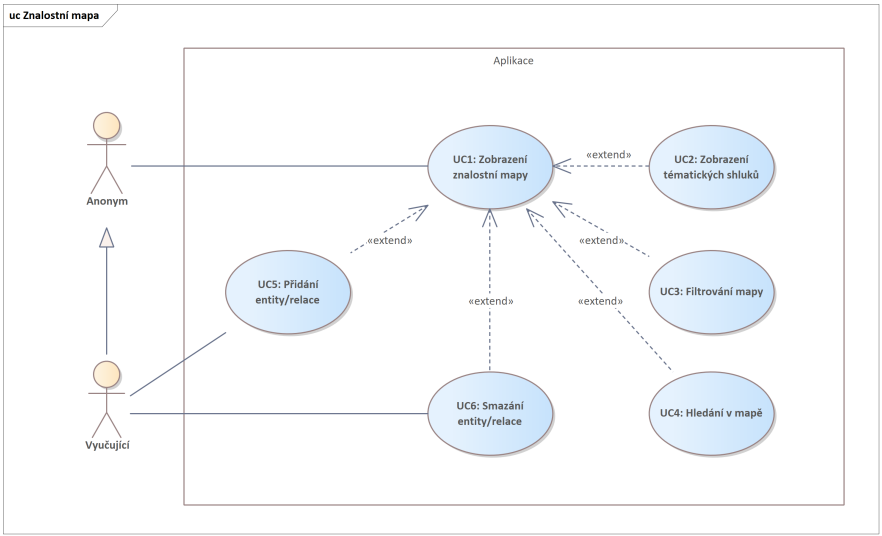
1. Server přijme požadavek na smazání entity nebo relace.
2. Server ověří, že daná entita nebo relace existuje.
3. Entita nebo relace je smazána z databáze.
4. Server vrací odpověď s úspěchem o odstranění entity nebo relace.

3.1.2.4 Pokrytí případů použití frontendových scénářů

Tabulka 3.1 Tabulka krytí případů použití : Znalostní mapa

	UC1	UC2	UC3	UC4	UC5	UC6
UC 1	✓		✓	✓		
UC 2					✓	
UC 3						✓
pokrytí	✓		✓	✓	✓	✓

Scénář „UC2 Zobrazení tématických shluků“ je specifikován na úrovni frontendu, ale nebyl tam implementován. Z tohoto důvodu nebyl tento scénář realizován ani na úrovni backendu. „UC3 Filtrování Mapy“ a „UC4 Hledání v Mapě“ jsou scénáře realizované na úrovni frontendu, kde probíhá vyhledávání a filtrování dat. Jelikož tyto činnosti závisí na datech poskytovaných serverem, v tabulce mapování pro backend je uveden „UC1 Předání dat znalostní mapy“, který popisuje předávání dat ze serveru, nezbytných pro provádění funkcí vyhledávání a filtrování na klientské straně.



Obrázek 3.1 Diagram případu užití: Znalostní mapa

3.1.3 Informace entity

3.1.3.1 UC4: Předání dat entity

Aktéři: Server
Počáteční podmínka: Server přijímá a zpracovává `HTTP GET` požadavky na endpoint `/api/entity/id`.
Scénář:

1. Server přijme požadavek na získání informací o entitě.
2. Server získá data o entitě z databáze.
3. Data jsou serializována do formátu JSON.
4. Server odešle data zpět klientovi jako odpověď na požadavek.

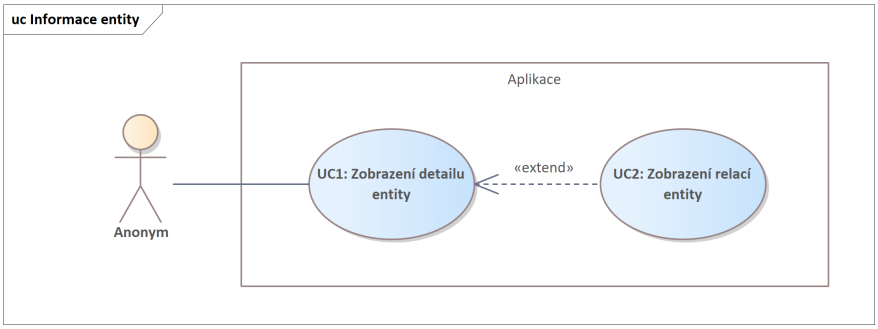
Alternativní scénář:

1. Server přijme požadavek na získání informací o entitě.
2. Server hledá data o entitě v databázi a zjišťuje, že entita neexistuje.
3. Server vrátí odpověď s chybou, informující o tom, že entita nebyla nalezena.

3.1.3.2 Pokrytí případů použití frontendových scénářů

Tabulka 3.2 Tabulka krytí případů použití : Informace entity

	UC1	UC2
UC 4	✓	✓
pokrytí	✓	✓



Obrázek 3.2 Diagram případu užití: Informace o entitě

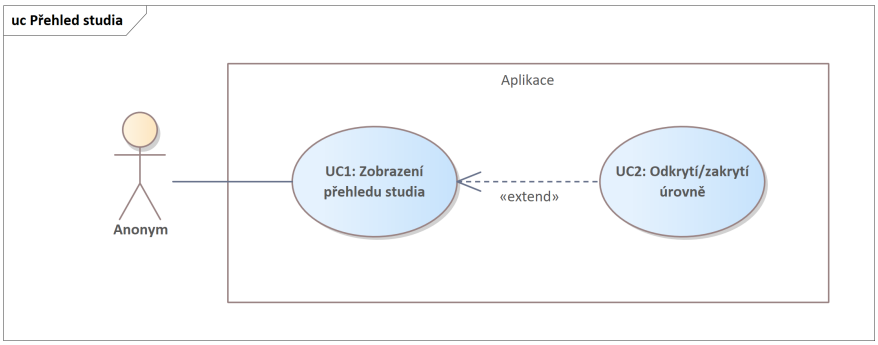
3.1.4 Přehled studia

3.1.4.1 UC5: Předání dat studijního přehledu

Aktéři: Server
Počáteční podmínka: Server přijímá a zpracovává `HTTP GET` požadavky na endpoint `/api/program`.
Scénář:

1. Server přijme požadavek na získání přehledu studia.
2. Server získá informace o studijních programech, specializacích a aktuálním semestru z databáze.
3. Data jsou serializována do formátu JSON.
4. Server odešle data zpět klientovi jako odpověď na požadavek.

3.1.4.2 Pokrytí případů použití frontendových scénářů



Obrázek 3.3 Diagram případu užití: Přehled studia

Tabulka 3.3 Tabulka krytí případů použití : Přehled studia

	UC1	UC2
UC 5	✓	✓
pokrytí	✓	✓

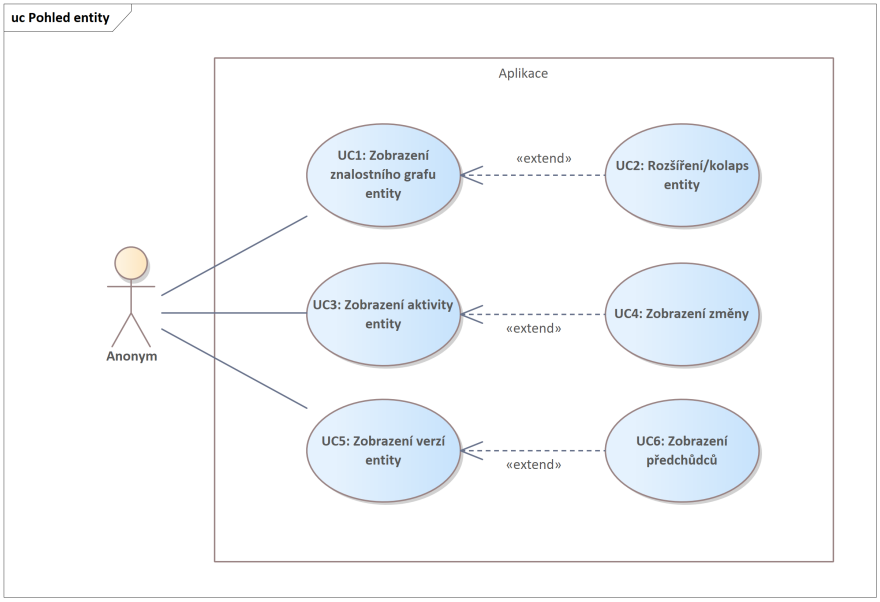
3.1.5 Pohledy entity

3.1.5.1 UC6: Předání dat verzí entity

Aktéři: Server
Počáteční podmínka: Server přijímá a zpracovává `HTTP GET` požadavky na endpoint `/api/version/id`.
Scénář:

1. Server přijme požadavek na zobrazeni verzí entity.
2. Server získá informace o verzích předmětů, specializacích a aktuálním semestru z databáze.
3. Data jsou serializována do formátu JSON.
4. Server odešle data zpět klientovi jako odpověď na požadavek.

3.1.5.2 Pokrytí případů použití frontendových scénářů



Obrázek 3.4 Diagram případu užití: Pohledy entity

Tabulka 3.4 Tabulka krytí případů použití : Pohledy entity

	UC1	UC2	UC3	UC4	UC5	UC6
UC 4	✓	✓				
UC 6					✓	✓
pokrytí	✓	✓			✓	✓

Scénáře „UC3 Zobrazení aktivit entity“ a „UC4 Zobrazení změn“ jsou specifikovány na úrovni frontendu, ale nebyly tam implementovány. Z tohoto důvodu nebyly tyto scénáře realizovány ani na úrovni backendu.

3.1.6 Autorizace uživatele

3.1.6.1 UC7: Autorizace uživatele

Aktéři: Server
Počáteční podmínka: Server přijímá `HTTP POST` požadavek na endpoint `/auth` pro autorizaci.
Scénář:

1. Server přijímá požadavek na autorizaci s přihlašovacími údaji (uživatelské jméno, heslo).
2. Server ověřuje poskytnutá data na správnost.
3. Jsou-li data správná:
   a. Server vytváří session pro uživatele.
   b. Server vrací odpověď se stavem úspěchu a autorizačním tokenem.

Alternativní scénář: Nesprávné údaje

1. Server ověřuje data a zjišťuje chybu (nesprávné údaje).
2. Server vrací odpověď s chybou autorizace.
3. Server poskytuje zprávu o chybě, například "Nesprávné přihlašovací údaje"nebo "Uživatel nenalezen".

3.1.7 Realizace funkčních požadavků

požadavky	UC1	UC2	UC3	UC4	UC5	UC6	UC7
F1	✓			✓	✓	✓	
F2		✓	✓				
F3							✓

3.2 Architektura aplikace

Pro úspěšnou realizaci a podporu jakéhokoli softwarového projektu je důležité promyslet jeho architekturu. To je základ, na kterém je celý projekt postaven, a definuje jak strukturu aplikace, tak její funkcionalitu. Vytvoření správné architektury zajišťuje škálovatelnost systému, což usnadňuje provedení následných změn v dalších iteracích. V mé práci jsem použil *třívrstvou architekturu*, která je ustálenou strukturou softwarových aplikací, rozdělující aplikace na tři logické a fyzické úrovně výpočtů: úroveň prezentace nebo uživatelské rozhraní; aplikační úroveň, na které probíhá zpracování dat; a úroveň dat, kde jsou data aplikace ukládána a spravována[19]. Dále podrobně popíšu každou z tří úrovní architektury, abyste lépe pochopili, jak spolu jednotlivé úrovně interagují a jaké funkce každá z nich plní v kontextu mého projektu.

3.2.1 Prezentační

V rámci mé práce je hlavní důraz kladen na vývoj backendu. Tato úroveň aplikace byla již dříve realizována v diplomové práci bakalářky Kláry Matouškové. V kontextu mé práce je nesmírně důležité zajistit správnou integraci backendu s frontendem. Tento aspekt je podrobněji zkoumán v sekci věnované integraci.

3.2.2 Business

V této části své práce podrobně popíši navrhování business logiky pro každou stránku, kterou představuje frontend. Tímto způsobem budu moci demonstrovat, jak systém zpracovává a řídí business procesy, a také jaká pravidla a algoritmy jsou používána pro provádění business operací na každé fázi interakce s uživatelem. Podrobnější popis jednotlivých stránek je uveden v sekci 2.1

3.2.2.1 Studium

Pro zachování hierarchie a správu dat jsem vyvinul následující strukturu tříd, která odráží úrovně vzdělávacího programu:

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor programu
name	String	Název programu
label	String	Štítek programu
group	String	Typ vzdělávacího programu
children	List<Accreditation>	Seznam souvisejících akreditací

Tabulka 3.5 Třída Program

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor akreditace
name	String	Název akreditace
label	String	Štítek akreditace
group	String	Vyučovací jazyk akreditace
children	List<Specialization>	Seznam souvisejících specializací
parent	Program	Odkaz na rodičovský program

Tabulka 3.6 Třída Accreditation

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor specializace
name	String	Název specializace
label	String	Štítek specializace
group	String	Katedra specializace
children	List<StudyPlan>	Seznam souvisejících studijních plánů
parent	Accreditation	Odkaz na rodičovskou akreditaci

Tabulka 3.7 Třída Specialization

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor studijního plánu
name	String	Název studijního plánu
label	String	Štítek studijního plánu
group	String	Typ podle formy studia
parent	Specialization	Odkaz na rodičovskou specializaci

Tabulka 3.8 Třída StudyPlan

Program Controller

- `GET /program/{id}` - Získává program podle ID.
- `PUT /program/{id}` - Aktualizuje existující program.
- `DELETE /program/{id}` - Maže program.
- `GET /program` - Získává všechny programy.
- `POST /program` - Vytváří nový program.

Accreditation Controller

- `GET /program/{programId}/accreditation/{id}` - Získává akreditaci podle ID.
- `PUT /program/{programId}/accreditation/{id}` - Aktualizuje akreditaci.
- `DELETE /program/{programId}/accreditation/{id}` - Maže akreditaci.
- `GET /program/{programId}/accreditation` - Získává všechny akreditace.
- `POST /program/{programId}/accreditation` - Vytváří novou akreditaci v programu.

Specialization Controller

- `GET /accreditation/{accreditationId}/specialization/{id}` - Získává specializaci podle ID.
- `PUT /accreditation/{accreditationId}/specialization/{id}` - Aktualizuje specializaci.
- `DELETE /accreditation/{accreditationId}/specialization/{id}` - Maže specializaci.
- `GET /accreditation/{accreditationId}/specialization` - Získává všechny specializace.
- `POST /accreditation/{accreditationId}/specialization` - Vytváří novou specializaci v akreditaci.

Study Plan Controller

- `GET /specialization/{specializationId}/studyplan/{id}` - Získává studijní plán podle ID.
- `PUT /specialization/{specializationId}/studyplan/{id}` - Aktualizuje studijní plán.
- `DELETE /specialization/{specializationId}/studyplan/{id}` - Maže studijní plán.
- `GET /specialization/{specializationId}/studyplan` - Získává všechny studijní plány.
- `POST /specialization/{specializationId}/studyplan` - Vytváří nový studijní plán v specializaci.

3.2.2.2 Tématické okruhy

Topic: Mapa uzlů a vazeb.

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor mapy témat
nodes	List<NodeTopic>	Seznam uzlů v mapě
links	List<LinkTopic>	Seznam vazeb mezi uzly v mapě

Tabulka 3.9 Třída Topic

NodeTopic: Odráží uzly mapy, kde každý uzlů reprezentuje téma. Uzly mohou být seskupeny (atribut *cluster*), což umožňuje vizuálně je rozdělit do kategorií.

Proměnná	Typ dat	Popis
name	String	Jméno uzlu, slouží jako unikátní identifikátor
cluster	int	Číselný identifikátor skupiny, umožňuje vizuální seskupení uzlů
topic	Topic	Reference na objekt Topic, ke kterému uzl patří
source_links	List<LinkTopic>	Seznam vazeb, kde uzlů je zdrojem
target_links	List<LinkTopic>	Seznam vazeb, kde uzlů je cílem

Tabulka 3.10 Třída NodeTopic

LinkTopic: Vazby mezi uzly.

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor vazby
source	NodeTopic	Reference na zdrojový uzlů
target	NodeTopic	Reference na cílový uzlů
value	List<String>	Seznam hodnot asociovaných s vazbou
topic	Topic	Reference na téma, ke kterému vazba patří

Tabulka 3.11 Třída LinkTopic

Topic Controller

- `GET /topics/nodes/{id}` - Získává uzel podle ID.
- `PUT /topics/nodes/{id}` - Aktualizuje existující uzel.
- `DELETE /topics/nodes/{id}` - Maže uzel.
- `GET /topics/links/{id}` - Získává odkaz podle ID.
- `PUT /topics/links/{id}` - Aktualizuje existující odkaz.
- `DELETE /topics/links/{id}` - Maže odkaz.
- `POST /topics/nodes` - Vytváří nový uzel.
- `POST /topics/links` - Vytváří nový odkaz mezi uzly.
- `GET /topics` - Získává všechny témata.

3.2.2.3 Znalostní mapa

Map: Třída, která zobrazuje celou mapu znalostí obsahující entity a vazby mezi nimi.

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor mapy znalostí
entities	List<InstitutionalEntity>	Seznam entit v mapě znalostí
links	List<LinkMap>	Seznam vazeb mezi entitami

Tabulka 3.12 Třída Map

InstitutionalEntity: Tato třída představuje základní entity v rámci mapy znalostí, které mohou být různých typů, jako jsou učitelé, témata, ústavy, kurzy nebo plány. Obsahuje informace o odkazech na externí zdroje a verze pro entitu (více v sekci 3.2.2.4).

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor entity
name	String	Název entity
label	String	Popisek entity
type	String	Typ entity (učitel, téma, institut, kurz, plán)
active	Boolean	Stav aktivity entity
semester	Semester	Semestr, ke kterému je entita přiřazena
source_links	List<LinkMap>	Seznam odkazů, kde tato entita je zdrojem
target_links	List<LinkMap>	Seznam odkazů, kde tato entita je cílem
web_links	List<LinkEntity>	Webové odkazy asociované s entitou
map	Map	Mapa, ke které je entita přiřazena

Tabulka 3.13 Třída InstitutionalEntity

LinkEntity: Představuje třídu odpovídající webovým odkazům entit.

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor web odkazu
ref	String	URL odkaz
name	String	Název odkazu
entityId	InstitutionalEntity	Entita, s kterou je odkaz asociován

Tabulka 3.14 Třída LinkEntity

LinkMap: Reprezentuje vazby mezi různými entitami.

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor odkazu
source	NodeTopic	Zdrojový uzel odkazu
target	NodeTopic	Cílový uzel odkazu
value	String	Hodnota nebo popis odkazu
map	Map	Mapa, ke které je odkaz přiřazen

Tabulka 3.15 Třída LinkMap

Semester: Představuje akademický semestr.

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor semestru

Tabulka 3.16 Třída Semester

Map Controller

- `GET /map/links` - Získat všechny spoje.
- `POST /map/links` - Vytvořit nový spoj.
- `PUT /map/links/{id}` - Aktualizovat spoj podle identifikátoru.
- `DELETE /map/links/{id}` - Odstranit spoj podle identifikátoru.
- `GET /map/{semester}` - Získat mapu podle semestru.

Institutional Entity Controller

- `GET /entity/{id}` - Získat institucionální entitu podle identifikátoru.
- `POST /entity` - Vytvořit institucionální entitu.
- `PUT /entity/{id}` - Aktualizovat institucionální entitu podle identifikátoru.
- `DELETE /entity/{id}` - Odstranit institucionální entitu podle identifikátoru.
- `GET /entity` - Získat všechny institucionální entity.
- `POST /entity/{id}/web-links` - Vytvořit webový odkaz pro entitu.
- `DELETE /entity/{id}/web-links/{link-id}` - Odstranit webový odkaz z entity.

Semester Controller

- `GET /semesters` - Získat všechny semestry.
- `POST /semesters` - Vytvořit nový semestr.
- `DELETE /semesters/{id}` - Odstranit semestr podle identifikátoru.

3.2.2.4 Verze

Version: Třída reprezentující verze předmětů. Třída má rodičovské a dětské verze pro sledování historie změn.

Proměnná	Typ dat	Popis
id	Long	Unikátní identifikátor verze
semester	Semester	Semestr, ke kterému verze patří
group	String	Akreditace, ke které verze patří
entity	InstitutionalEntity	Entita, ke které verze patří
parent	List<Version>	Rodičovské verze této verze
children	List<Version>	Dětské verze této verze

Tabulka 3.17 Třída Version

Version Controller

- `PUT /version/{id}` - Aktualizace existující verze entity. Předává ID verze v URL pro identifikaci a aktualizaci.
- `DELETE /version/{id}` - Smazání verze entity. Také využívá ID verze v URL pro určení mazané verze.
- `POST /version` - Vytvoření nové verze entity. Data pro novou verzi jsou předána v těle požadavku.
- `GET /version/{entity}` - Získání všech verzí podle ID entity. ID entity je předáno v URL, a jsou vráceny všechny s touto entitou spojené verze.

3.2.3 Datová

Základem datové vrstvy mé aplikace je databáze, která byla automaticky generována díky funkcím Spring Boot a jeho integraci s *Java Persistence API* (dále jen JPA). Pro interakci s databází jsem připojil potřebné závislosti prostřednictvím systému správy závislostí Maven, přidáním následujících knihoven Spring Boot Starter Data JPA a ovladače databáze pro PostgreSQL:

Výpis kódu 3.1 Závislosti Maven pro PostgreSQL a Spring Boot JPA.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

Díky anotacím poskytovaným Spring Data JPA byla struktura databáze automaticky vytvořena. Tyto anotace zahrnují:

- **@Entity**: Definuje třídu jako entitu, která má být mapována na databázovou tabulku. Každá instance @Entity odpovídá řádku v tabulce.
- **@Table**: Specifikuje konkrétní tabulku v databázi, na kterou bude entita mapována. Pokud není název tabulky specifikován, použije se název třídy.
- **@Id**: Označuje pole jako primární klíč entity, unikátní identifikátor každého záznamu v tabulce.

Anotace vztahů v JPA umožňují popsat vztahy mezi entitami v databázi:

- **@OneToOne**: Používá se pro označení vztahu jeden k jednomu mezi dvěma entitami. Například, každý uživatel může mít pouze jeden pas.
- **@OneToMany**: Definuje vztah jeden k mnoha. Například, jeden autor může napsat mnoho knih.
- **@ManyToOne**: Označuje vztah mnoho k jednomu. Například, mnoho studentů může patřit k jedné fakultě.

- **@ManyToMany**: Používá se pro popis vztahu mnoho k mnoha. Například, mnoho studentů může navštěvovat mnoho kurzů.

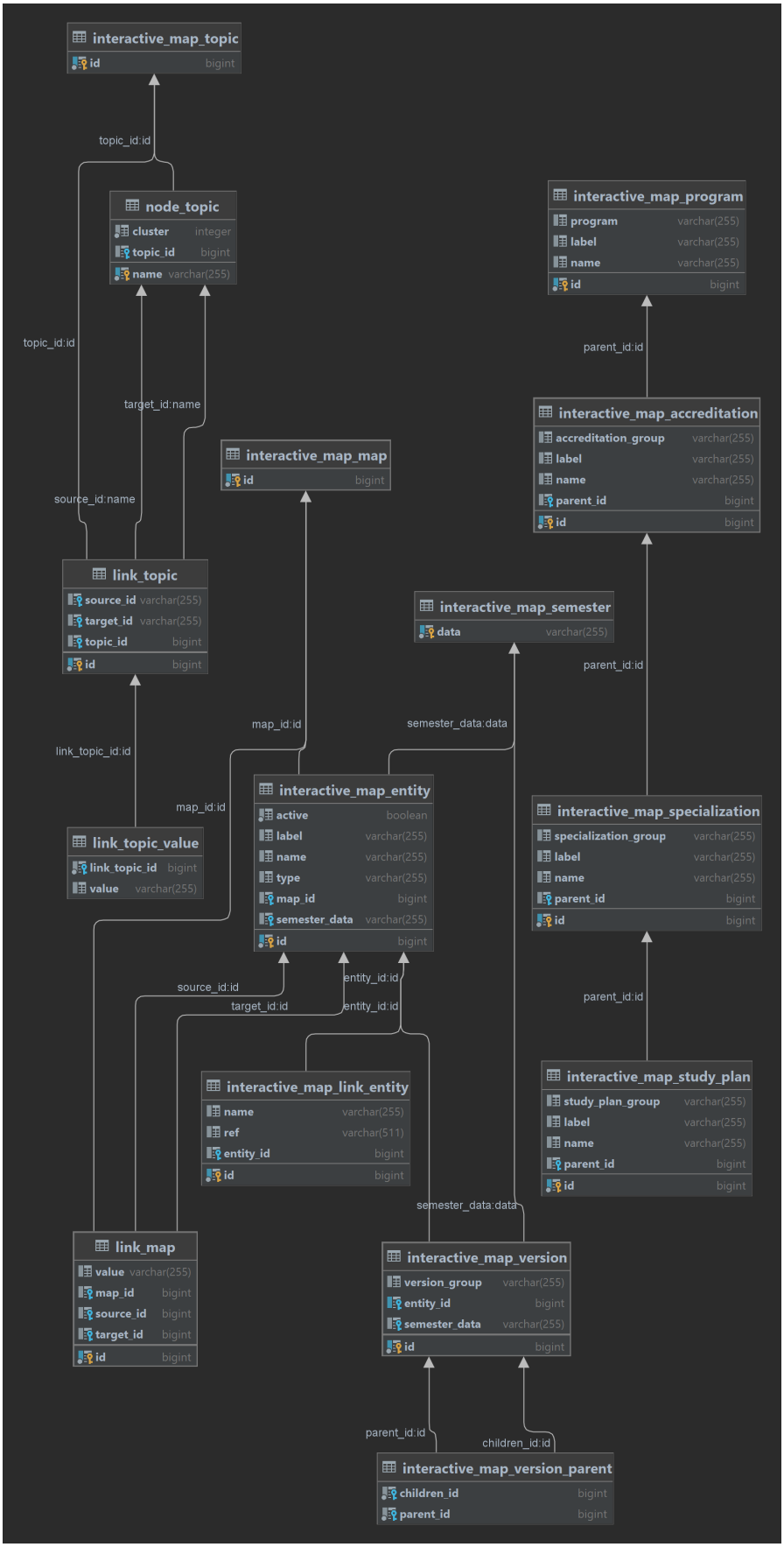
Detaily práce s naplněním databáze jsem popsal v sekci 4.2.3. Každá z těchto anotací přispívá k přesnému a jasnému definování struktury databáze a vztahů, což je důležité pro správné navrhování a optimalizaci výkonu aplikace.

3.2.3.1 Schéma Databáze

Pro dokumentaci schématu databáze byly využity nástroje poskytované integrovaným vývojovým prostředím *IntelliJ IDEA*. To mi umožnilo automaticky generovat a prezentovat schéma, zobrazené na obrázku 3.5

Automatická generace databáze je sice pohodlná, ale ne vždy funguje dokonale. Například při praktickém používání databáze jsem narazil na problém s nedostatečnou délkou pole ref, které bylo původně omezeno na 255 znaků. Toto omezení vedlo k ořezávání URL adres, čímž byla narušena funkčnost odkazů. Pro řešení tohoto problému bylo rozhodnuto zvýšit délku pole na 511 znaků.

V první iteraci vývoje backendu mé aplikace byly objekty „Znalostní mapy“ implementovány jako jednotná třída „InstitutionalEntity“. Pro rozlišení různých typů entit, jako jsou předměty, učitelé, katedry, studijní plány a témata, bylo zavedeno pole *type*. Tento přístup jsem zvolil, protože umožňuje zjednodušit model dat a urychlit počáteční fázi vývoje. V dalších iteracích, pro vytvoření přesnějšího představení rozdílů mezi typy entit a zlepšení škálovatelnosti systému, lze přejít k použití polymorfismu.



Obrázek 3.5 Schéma databáze interaktivní mapy.

Kapitola 4
Implementace

4.1 Použité technologie

4.1.1 Spring Boot

Vybral jsem Spring Boot pro vývoj back-endu webu, protože již mám zkušenosti s touto technologií a rád bych si rozšířil své znalosti a prohloubil porozumění jejím možnostem. Spring Boot nabízí bohatou funkcionalitu a podporu, což výrazně zjednodušuje integraci s různými systémy a službami potřebnými pro vzdělávací zařízení. Například umožňuje snadno pracovat se systémy pro správu vzdělávání, databázemi studentů a integrovat se s dalšími vzdělávacími platformami.

Díky velké a aktivní komunitě snadno nalézám řešení na vznikající otázky a získávám technickou podporu. Kromě toho přítomnost hotových řešení pro zabezpečení a autentizaci v Spring Boot je kritická pro ochranu osobních údajů studentů, což je prioritou pro jakoukoli vzdělávací platformu. Tímto způsobem mi výběr Spring Boot umožní nejen realizovat stanovené úkoly, ale také poskytne možnost flexibilně rozšiřovat backend v dalších iteracích projektu.

4.1.2 PostgreSQL

V průběhu vývoje mého backendového projektu na platformě JVM jsem čelil potřebě vybrat vhodný systém pro správu databází. Rozhodl jsem se pro PostgreSQL. Toto rozhodnutí bylo motivováno několika klíčovými aspekty, které považuji za kriticky důležité pro úspěch mého projektu : kompatibilita s JVM, rozšířené možnosti SQL, spolehlivost a bezpečnost. Podrobněji popsáno v sekci 2.2.2

4.1.3 Spring Data JPA

Bylo také důležité vybrat efektivní technologii pro práci s databázemi, která by podporovala jednoduchost vývoje a integraci se stávajícími Spring aplikacemi. Rozhodnutí bylo učiněno ve prospěch Spring Data JPA z několika klíčových důvodů: integrace se Spring Frameworkem, automaticky generované dotazy a snížení šablonového kódu. Podrobněji popsáno v sekci 2.2.3.1.

4.1.4 Spring Security

Pro ochranu aplikace jsem zvolil framework Spring Security. Podrobněji o technologii jsem napsal v sekci 2.2.3.2. Také v sekci 4.3 popisuji, jak přesně je Spring Security používán v projektu pro autentizaci a autorizaci.

4.1.5 Liquibase

Liquibase je řešení pro správu změn schématu databáze, které umožňuje rychleji a bezpečněji provádět změny v databázích ve všech fázích, od vývoje po produkci. Chcete-li začít s Liquibase snadno a jednoduše, můžete použít SQL pro psaní migračních skriptů. Pokud chcete využít možnosti abstrakce databáze, které umožňují provádět změny jednou a nasazovat je na různé databázové platformy, můžete specifikovat změny nezávislé na konkrétní databázi ve formátech XML, JSON nebo YAML [20].

V rámci mého projektu jsem použil Liquibase, aby jsem zjednodušil proces spuštění aplikace s automatickým naplněním databáze. Toto řešení mi umožnilo automatizovat migrace a zajistit počáteční naplnění dat při spuštění programu.

4.2 Práce s daty

Abych demonstroval funkčnost databáze a její důležitost v architektuře aplikace, rozhodl jsem se ji naplnit aktuálními informacemi. To mi umožnilo nejen ověřit funkčnost systému, ale také posoudit jeho efektivitu a rychlost zpracování dotazů. V této části podrobně popisuji proces hledání a vkládání dat.

4.2.1 Naplnění databáze Studium

Pro vyplnění informací o „Studium“ (podrobněji v sekci 2.1.2) jsem použil aktuální data z webové stránky BK.

4.2.2 Naplnění databáze Tématické okruhy

Pro vyplnění informací o „Tematicke okruhy“ (podrobněji v sekci 2.1.4) jsem se rozhodl použít mockové data, která již byla napsána ve frontendu. Toto rozhodnutí bylo učiněno za účelem zjednodušení procesu vývoje a testování systému.

4.2.3 Naplnění databáze Znalostní Mapa

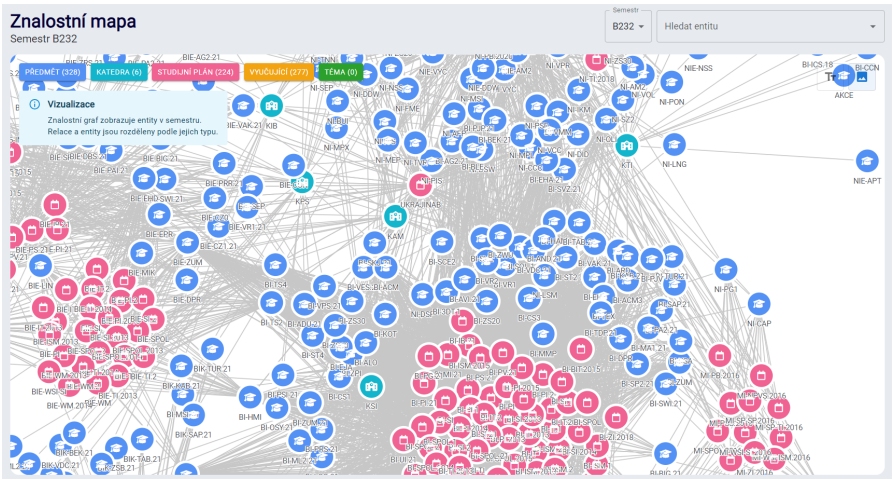
Rozhodl jsem se naplnit „Znalostní Mapu“ (podrobněji v sekci 2.1.3) informacemi pro dva semestry: letní a zimní. Přitom jsem využil různé přístupy k naplnění, aby bylo možné ukázat některé omezení a slabé stránky ve frontendové implementaci.

4.2.3.1 Letní semestr

Pro rychlé a úplné naplnění databáze informacemi o letním semestru jsem se spojil s panem Ing. Michalem Valentou, Ph.D., který mi poskytl potřebné údaje. Od něj jsem obdržel kompletní seznam studijních plánů, předmětů, vyučujících a kateder, stejně jako vztahů mezi těmito entitami. Všechna data byla prezentována ve snadno zpracovatelném formátu souboru .csv, což výrazně zjednodušilo proces jejich integrace do databáze. Výsledek zobrazení informací o letním semestru na webu je zobrazen na obrázku 4.1.

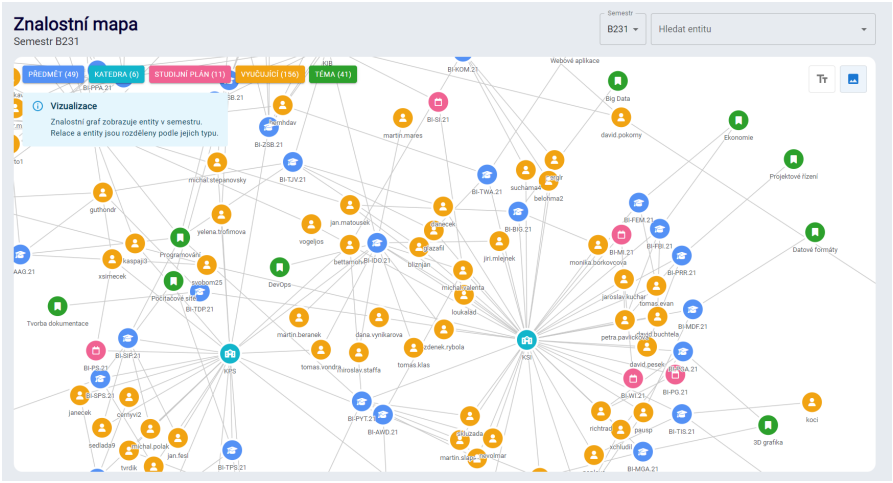
4.2.3.2 Zimní semestr

Na rozdíl od letního semestru jsem se rozhodl pro zimní semestr přijmout jiný přístup k naplnění databáze. Základní informace jsem získal z bakalářské práce Martiny Chomyšinové [21], která obsahovala podrobné údaje o předmětech bakalářského programu a s nimi spojených kompetencích. To mi umožnilo přidat entity `téma`. Kromě toho jsem pro popis vztahů mezi vyučujícími



Obrázek 4.1 Zobrazení informací o letním semestru

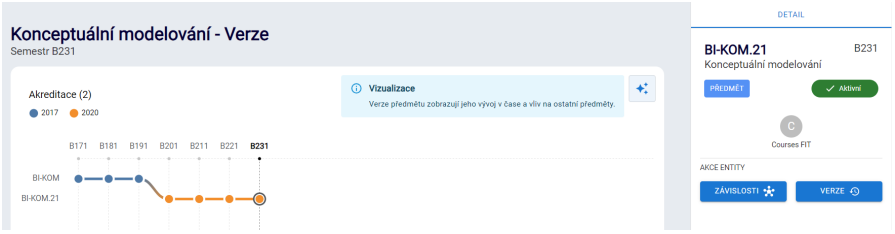
a předměty využil informace z webu KOS, kde jsou uvedeny role vyučujících v rámci předmětů. Pro získání úplných jmen vyučujících a jejich příslušnosti k katedrám jsem využil web Usermap. Výsledek zobrazení informací o zimním semestru na webu je zobrazen na obrázku 4.2.



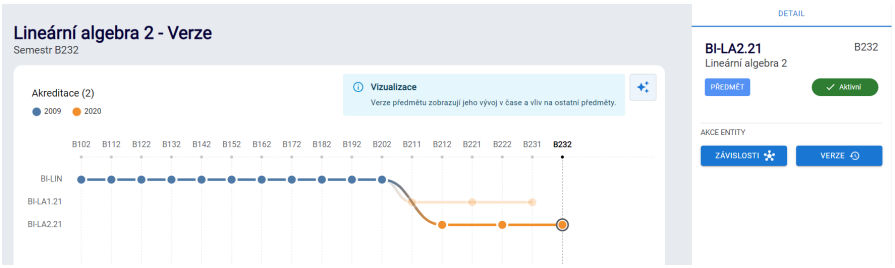
Obrázek 4.2 Zobrazení informací o zimním semestru

4.2.4 Naplnění databáze Verze

Pro vyplnění informací o „Verze“ jsem prezentoval verze předmětů BI-KOM.21 (obrázek 4.3) a BI-LA2.21 (obrázek 4.4). Tyto předměty byly vybrány k demonstraci rozdílů mezi verzemi. Například předmět BI-KOM.21 v rámci akreditace oboru „Informatika“ platné do roku 2024 byl dříve označován pouze jako BI-KOM. Dále jsem použil příklad s lineární algebrou, protože v rámci téže akreditace to byl původně jediný předmět BI-LIN, který je v současné akreditaci rozdělen na dva samostatné předměty: BI-LA1.21 a BI-LA2.21.



Obrázek 4.3 Zobrazení verzi předmětu BI-KOM



Obrázek 4.4 Zobrazení verzi předmětu BI-LA2.21

4.3 Autentizace a autorizace

V mém projektu server poskytuje uživatelům možnost vytvářet, měnit a mazat data. Pro zajištění bezpečnosti a ochranu dat před neoprávněným přístupem jsem do systému integroval framework Spring Security. V rámci této konfigurace byla přidána role administrátora, která poskytuje plný přístup ke všem funkcím `CRUD`.

Základem pro implementaci autentizačního a autorizačního systému v mém projektu byl článek publikovaný na webu *Medium*. Tento zdroj poskytl podrobný průvodce používáním Spring Boot 3 a Spring Security 6 pro vytvoření mechanismů JWT autentizace a autorizace. Odkaz na článek: .

4.3.1 Realizace autentizace správce

V průběhu vývoje mého projektu jsem čelil potřebě zajistit bezpečný přístup k administrativnímu rozhraní. Abych zjednodušil vývojový proces, rozhodl jsem se použít metodu ukládání přihlašovacích údajů správce v paměti. Tento způsob mi umožnil rychle implementovat autentizační funkce.

Pro implementaci autentizace jsem použil následující přístup v konfiguraci Spring Security:

Výpis kódu 4.1 Konfigurace UserDetailsService pro autentizaci správce

```
@Bean
public UserDetailsService userDetailsService() {
    var user = User.withUsername("admin")
            .password("adminpassword")
            .roles("ADMIN")
            .build();
    return new InMemoryUserDetailsManager(user);
}
```

- **Vytvoření uživatele:** `User.withUsername("admin")` nastavuje uživatelské jméno, které bude použito pro přihlášení.

- **Nastavení hesla:** `password("adminpassword")` nastavuje heslo.
- **Přiřazení role:** `roles("ADMIN")` určuje, že uživatel má práva správce, což mu dává přístup ke všem funkcím.

Vybral jsem `InMemoryUserDetailsManager` pro správu uživatelských dat, protože toto řešení je ideální pro počáteční fázi vývoje a testování.

Avšak tento způsob implementace není ideální, a v dalších fázích vývoje serveru je nutné vytvořit ukládání informací o účtech v databázi a rozšířit seznam rolí pro studenty a učitele.

4.3.2 Nastavení přístupu

V rámci projektu byla věnována zvláštní pozornost nastavení autorizace pro různé typy HTTP požadavků.

Pro nastavení přístupových práv byla použita následující konfigurace v Spring Security:

Výpis kódu 4.2 Konfigurace pravidel autorizace v Spring Security

```
.authorizeHttpRequests(request ->
        request
                .requestMatchers(HttpMethod.GET).permitAll()
                .requestMatchers(HttpMethod.POST, "/auth").permitAll()
                .anyRequest().hasRole("ADMIN")
                )
```

Výpis kódu 4.2 odráží následující aspekty bezpečnosti:

- **Dostupnost `GET` požadavků**: Všechny `GET` požadavky jsou dostupné bez omezení.
- **Otevřený přístup k autentizaci**: `POST` požadavky na cestě `/auth` jsou také otevřené pro všechny.
- **Omezený přístup pro administrativní funkce**: Všechny ostatní požadavky vyžadují, aby uživatel měl roli `ADMIN`.

4.4 Integrace s front-end

Ve webových aplikacích je správná integrace mezi frontendem a backendem klíčovým aspektem. Tato část práce je věnována podrobnému popisu interakce mezi uživatelským rozhraním a serverovou částí projektu, označující, jak jsou data přenášena a zpracovávána.

4.4.1 Analýza a příprava dat

Pro pochopení struktury dat, kterou požaduje frontend, jsem začal analýzou mockových dat, která již byla zdokumentována v projektu. Následně, abych zajistil, že serverová část správně předává informace, pro každou třídu jsem vytvořil odpovídající DTO a Mapper třídy. V těchto třídách byly implementovány metody pro efektivní konverzi dat z modelů do DTO a zpět, což zaručovalo přesnost a integritu přenosu dat mezi serverem a klientem.

Ve vývoji mého projektu jsem věnoval zvláštní pozornost typům dat, které mohou přijímat pouze určité hodnoty. Například, pro objekty na stránce „Znalostní Mapa“, pole `type` v třídě `Entity` přijímá pouze předem stanovené hodnoty, jako jsou `course`, `teacher`, `plan`, `institute`, a `topic`. To zajišťuje přísnost a shodu dat mezi frontendem a backendem. Prozkoumal jsem, jak jsou v různých třídách definovány proměnné, které mohou přijímat pouze určité hodnoty, aby byla zajištěna správnost zpracování dat a usnadněna validace. To pomáhá předejít chybám a zajišťuje správné zobrazení informací na straně frontendu.

4.4.2 Integrace a Zpracování Požadavků

Poté jsem přešel k úpravě obslužných rutin (handlers), aby frontend začal posílat požadavky přímo serverové části, vzdávaje se používání mock dat. Abych úspěšně přepsal obslužné rutiny, prostudoval jsem možnosti knihovny msw (Mock Service Worker), která umožňuje řídit síťové chování při testování a vývoji. Našel jsem příklady použití msw pro nastavení zpracování `GET` požadavků, což mi pomohlo správně nastavit zachytávání požadavků a jejich zpracování. Informace a příklady jsem našel na oficiálním webu msw, dostupné na: `https://mswjs.io/docs/network-behavior/rest`.

Ve výpisu kódu 4.3 jsem ukázal metodu `GET`, která se používá pro dotaz na data mapy podle semestru přes REST API. Kód se obrací na server na `http://localhost:8080/map/${semester}`, kde `semester` je parametr předaný v URL požadavku. Po získání dat od serveru ve formátu JSON, kód vrátí tato data klientovi se stavem 200, což značí úspěšné provedení požadavku. Tento příklad ilustruje, jak je možné integrovat frontend s backendem, zpracovávat požadavky asynchron.

Výpis kódu 4.3 Zachycení požadavku pro získání dat mapy podle semestru.

```
rest.get('/api/map/:semester', async (req, res, ctx) => {
    const { semester } = req.params;
    const backendResponse =
        await fetch(`http://localhost:8080/map/${semester}`);
    const map = await backendResponse.json();
    return res(ctx.status(200), ctx.json(map));
})
```

4.5 Testování

V této části popíši proces testování backendových komponent aplikace vyvinuté na platformě Spring Boot. Testování zahrnuje jak automatické, tak manuální metody ověřování funkčnosti a výkonnosti systému.

4.5.1 Automatické testování

4.5.1.1 Testování kontrolerů

Pro kontrolery byly napsány testy, které ověřují správnost funkcí metod `GET`, `POST`, `PUT` a `DELETE`. Pro ověření požadavků dostupných pouze pro administrátory jsem použil anotaci `@WithMockUser`. To umožnilo emulovat požadavky jménem administrátora, což je důležité pro ověření funkčnosti dostupné pouze uživatelům s administrátorskými právy.

4.5.1.2 Testování mapperů

Zvláštní pozornost byla věnována testování mapperů, které zajišťují převod dat mezi formátem přijatým v DTO (Data Transfer Object) a formátem entit (Entity). Automatické testy byly vyvinuty pro ověření, že mapování z DTO do Entity a z Entity do DTO probíhá správně. To je důležité pro zajištění správné integrace s frontendem, protože chyby v převodu dat mohou vést k nesprávnému zobrazení informací nebo k chybám v logice aplikace.

4.5.2 Ruční testování

Také jsem provedl ruční testování po integraci backendu s frontendem. Hlavní pozornost byla věnována správnému zobrazení informací, protože v této fázi frontend podporuje pouze požadavky typu `GET`.

4.5.2.1 Testování výkonu

V rámci mé práce jsem věnoval zvláštní pozornost testování `GET` požadavků. Na rozdíl od `POST`, `PUT` a `DELETE` požadavků, které interagují s jedním datovým prvkem, `GET` požadavky často vracejí značné množství dat. To je důležité pro analýzu výkonu systému, protože rychlost zpracování těchto požadavků přímo ovlivňuje zobrazení informací uživatelům. Pro hodnocení výkonu v reálných podmínkách byly provedeny testy s naplněnou databází, s využitím webu ReqBin pro generování `GET` požadavků a měření doby jejich zpracování.

Před provedením testů byly dočasně zakomentovány anotace, které jsou nezbytné pro integraci backendu s frontendem, `@CrossOrigin(origins = "http://localhost:3000")`.

Pro hodnocení výkonu systému byly vybrány požadavky, které reprezentují různé aspekty fungování aplikace:

- `localhost:8080/program` — požadavek vrací informace o studijních programech, akreditacích, specializacích a studijních plánech.
- `localhost:8080/map/b232` — testování požadavku na „Znalostní mapu“ za semestr b232, která obsahuje největší množství objektů (obrázek 4.1).
- `localhost:8080/version/1020` — požadavek na získání dat o nejobsáhlejší větvi verzí předmětu „lineární algebra“.
- `localhost:8080/topics` — požadavek na získání dat o tematických okruzích.
- `localhost:8080/entity` — požadavek na získání všech dostupných entit, testuje schopnost systému rychle zpracovávat požadavky na základní datové prvky.

Tabulka 4.1 Výsledky testování výkonu `GET` požadavků

URL	Stav	Doba odpovědi	Velikost odpovědi
localhost:8080/program	Stav: 200	309 ms	5.05 kb
localhost:8080/map/b232	Stav: 200	834 ms	445.56 kb
localhost:8080/entity	Stav: 200	163 ms	161.60 kb
localhost:8080/topics	Stav: 200	865 ms	123.43 kb
localhost:8080/version/1020	Stav: 200	36 ms	1.78 kb

4.5.2.2 Problémy s frontendem

Výsledky testování odhalily chyby ve fungování frontendu. Podívejme se na ně podrobněji.

- **Zobrazení závislostí entity:** Problém nastává při pokusu o zobrazení závislostí entity. Při kliknutí na tlačítko „Zavislost“, měli být uživatelé přesměrováni z aktuální stránky, například `/map/b231` (kde b231 je číslo semestru), na stránku `/map/b231/entity/{id}`. Kvůli chybě ve frontendu byl požadavek však směrován na nesprávnou adresu `/map/b101/entity/{id}`, kde b101 je nesprávné číslo semestru. To vede k tomu, že server nemůže správně zpracovat požadavek, protože obdrží nesprávné číslo semestru a tudíž nenajde požadovanou entitu.
- **Obnovení stránky:** Při výběru jiného semestru ze seznamu na stránce “Znalostni Mapa” frontend neodesílá nový požadavek na server, a proto je nutné stránku obnovit, aby se zobrazily informace o vybraném semestru.

4.6 Dokumentace

Veškerá dokumentace k mému projektu je napsána v angličtině. Pro dokumentaci serverové části byl použit nástroj Swagger, který umožňuje interaktivní prohlížení a testování API endpointů přímo ve webovém prohlížeči. Swagger automaticky generuje srozumitelnou a snadno navigovatelnou dokumentaci na základě anotací v kódu, což výrazně usnadňuje orientaci ve funkcích a struktuře API. Tato dokumentace je dostupná jak lokálně, tak na serveru, což zlepšuje její dostupnost a užitečnost pro vývojáře a uživatele systému. Podrobné informace o implementaci a použití Swaggeru v projektu jsou dostupné v dokumentu `README.md`.

Kapitola 5
Závěr

Bakalářská práce je věnována vytvoření serverové části pro webovou aplikaci „interaktivní mapa průchodů FIT ČVUT“. Cílem práce bylo analyzovat prototyp webové frontend aplikace, navrhnout a realizovat backend aplikace.

Práce je rozdělena na dvě části - teoretickou a praktickou. První část práce je věnována seznámení se strukturou a daty prezentovanými ve webové aplikaci. A také analýze a výběru technologií vybraných pro psaní back-endu (praktické části).

V analýze frontendu byly identifikovány hlavní datové objekty, které serverová část přijímá, například znalostní mapa a další objekty související s touto mapou, jako například prvky mapy a relace mezi nimi. Také seznámení s logikou aplikace, což pomohlo při psaní tříd a jejich vazeb navzájem.

Pro vývoj backendu byla provedena analýza a srovnání technologií pro psaní back-endu na základě platformy JVM. V důsledku toho byly zvoleny technologie Spring Boot pro rychlou a efektivní tvorbu REST API a PostgreSQL jako spolehlivá a výkonná relační databáze.

Praktická část zahrnovala navrhování databáze, realizaci serverové části s integrací do již existujícího systému. Byly úspěšně realizovány všechny předpokládané funkce, například jako zobrazení informací, autorizace administrátorského účtu s možností manipulace s daty. Také bylo dosaženo kompatibility s frontendem.

Data aplikace byla aktualizována prostřednictvím integrace s webovým zdrojem Bílá kniha, což zajistilo uživatelům přístup k nejaktuálnějším informacím. Testování bylo prováděno jak v izolovaných modulech, tak v kontextu plné integrace backendu a frontendu. V průběhu naplnění databáze a ručního testování byly odhaleny slabosti a chyby ve fungování frontendu, jako je například nesprávné vytváření URL požadavků.

Takto byly úspěšně dosaženy všechny cíle, které byly stanoveny na začátku práce, například takové, jako je kompletně zaplněná databáze a funkční integrace s frontendem. Vyvinutý projekt nejenom že usnadňuje navigaci pro studenty a učitele fakulty FIT ČVUT, ale také přispívá k lepšímu plánování akademické kariéry, motivuje studenty k efektivnímu využívání zdrojů univerzity. Výsledky této práce mohou být využity v budoucích výzkumech a vývoji, které jsou zaměřeny na zlepšení vzdělávacího procesu a vzdělávacího prostředí.

Bibliografie

1. MATOUŠKOVÁ, Klára. *Interaktivní mapa průchodu studiem na FIT ČVUT*. Praha, Česká republika, 2023. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
2. ČVUT. *Bílá kniha ČVUT* [online]. [B.r.]. [cit. 2024-05-05]. Dostupné z: `https://bilakniha.cvut.cz/`.
3. *Harmonogram bakalářského a magisterského studia 2023/2024* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://fit.cvut.cz/studenti/bakalar-magistr/harmonogram/harmonogram-bsp-msp-2023-2024.pdf`.
4. *Průvodce prváka bakalář* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://fit.cvut.cz/cs/studium/pruvodce-studiem/bakalarske-a-magisterske-studium/pruvodce-prvaka-bakalar`.
5. *KOS* [online]. České vysoké učení technické v Praze, [b.r.] [cit. 2024-04-25]. Dostupné z: `https://www.kos.cvut.cz/`.
6. *FIT CTU Courses* [online]. Fakulta informačních technologií, České vysoké učení technické v Praze, [b.r.] [cit. 2024-04-25]. Dostupné z: `https://courses.fit.cvut.cz/`.
7. ČVUT-VIC, Ing. Petr Karel. *Vyhledávání osob na ČVUT* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://usermap.cvut.cz/`.
8. *Spring Boot* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://spring.io/projects/spring-boot/`.
9. *Micronaut Guide* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://docs.micronaut.io/4.4.3/guide/`.
10. *About Quarkus* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://quarkus.io/about/`.
11. *Maven Tooling Guide* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://quarkus.io/guides/maven-tooling#dev-mode`.
12. *Quarkus Continuum* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://quarkus.io/continuum/`.
13. *Introduction to Vert.x and Reactive* [online]. [B.r.]. [cit. 2024-04-26]. Dostupné z: `https://vertx.io/introduction-to-vertx-and-reactive/`.
14. *About PostgreSQL* [online]. [B.r.]. [cit. 2024-04-25]. Dostupné z: `https://www.postgresql.org/about/`.
15. *Spring Data JPA* [online]. [B.r.]. [cit. 2024-05-06]. Dostupné z: `https://spring.io/projects/spring-data-jpa`.

16. *What is Spring Data JPA and Why Should You Use It?* [online]. [B.r.]. [cit. 2024-05-06]. Dostupné z: https://thorben-janssen.com/what-is-spring-data-jpa-and-why-should-you-use-it/.
17. *Spring Security* [online]. [B.r.]. [cit. 2024-04-26]. Dostupné z: https://spring.io/projects/spring-security.
18. *Spring Security Reference* [online]. [B.r.]. [cit. 2024-04-26]. Dostupné z: https://docs.spring.io/spring-security/site/docs/3.2.5.RELEASE/reference/htmlsingle/#what-is-acegi-security.
19. *Three-Tier Architecture* [online]. [B.r.]. [cit. 2024-05-02]. Dostupné z: https://www.ibm.com/topics/three-tier-architecture.
20. *Introduction to Liquibase* [online]. [B.r.]. [cit. 2024-05-06]. Dostupné z: https://docs.liquibase.com/concepts/introduction-to-liquibase.html.
21. CHOMYŠINOVÁ, Martina. *Konceptualizace kompetencí studia na FIT ČVUT*. Praha, Česká republika, 2024.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `frontend` ..... zdrojové kódy implementace frontend
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF