UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2025 Jan Chyška

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Systém pro správu závěrečných prací

Bakalářská práce

2025 Jan Chyška

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Chyška**
Osobní číslo:	**I22102**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Systém pro správu závěrečných prací**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je navrhnout a implementovat systém pro zadávání a správu závěrečných prací, který umožní efektivní správu diplomových a bakalářských prací v akademickém prostředí. Systém bude vyvinut v programovacím jazyce C# a ASP.NET Core MVC. Bude umožňovat zadávání nových prací, spravovat jejich stav a přidávat potřebné dokumenty. Systém umožní přidávat komentáře k jednotlivým pracím a zahrne také funkcionalitu pro verzování diplomových prací, což umožní uchovávat historické verze a sledovat změny. Dále bude obsahovat funkci pro sledování požadavků, která poskytne efektivní přehled nedořešených aspektů rozpracovaných prací. V teoretické části student popíše alternativní produkty na trhu a použité technologie, v praktické části pak zdokumentuje analýzu, návrh a implementaci výsledného systému.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

GRIFFITHS, Ian. Programming C# 12: Build Cloud, Web, and Desktop Applications. O'Reilly Media, 2024. ISBN 978-1098158361.
SKEET, Jon. C# in depth. Fourth edition. Shelter Island, NY: Manning Publications Co., [2019]. ISBN 978-1-61729-453-2.
PECINOVSKÝ, Rudolf. Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky. 2., aktualiz. a rozš. vyd. Myslíme v.. Praha: Grada, 2009. ISBN 978-80-247-2653-3.

Vedoucí bakalářské práce: **Ing. Jan Merta, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2025

Prohlašuji:

Práci s názvem Systém pro zprávu závěrečných prací jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 22.8.2025

Jan Chyška

PODĚKOVÁNÍ

Rád bych vyjádřil své upřímné poděkování panu Ing. Janu Mertovi, Ph.D., za vedení mé bakalářské práce, cenné rady, odborné konzultace a podporu, kterou mi během zpracování práce věnoval. Jeho trpělivost, ochota a vstřícný přístup mi velmi pomohly při řešení odborných i praktických otázek a významně přispěly k dokončení této práce.

ANOTACE

Tato bakalářská práce se zabývá návrhem a implementací informačního systému pro správu závěrečných prací. Cílem je vytvořit webovou aplikaci umožňující evidenci bakalářských a diplomových prací, jejich správu vedoucími a studenty a zpřístupnění výsledných dokumentů. Systém je realizován s využitím platformy .NET 9, architektury ASP.NET Core MVC, databázového serveru Microsoft SQL Server a moderních webových technologií (HTML, CSS, Bootstrap, JavaScript, jQuery, JSON).

KLÍČOVÁ SLOVA

webové aplikace, informační systémy, MVC, SQL Server, závěrečné práce, ASP.NET

TITLE

Information system for final thesis management

ANNOTATION

This bachelor thesis focuses on the design and implementation of an information system for managing final theses. The aim is to develop a web application that enables the registration of bachelor's and master's theses, their administration by supervisors and students, and the publication of the final documents. The system is implemented using the .NET 9 platform, ASP.NET Core MVC architecture, Microsoft SQL Server database, and modern web technologies (HTML, CSS, Bootstrap, JavaScript, jQuery, JSON).

KEYWORDS

web applications, information systems, MVC, SQL Server, final theses, ASP.NET

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

API – Application Programming Interface

ASP.NET Core – framework pro tvorbu webových aplikací v .NET

CSS – Cascading Style Sheets

CSRF – Cross-Site Request Forgery

DB – Databáze

Diff – algoritmus pro porovnání rozdílů mezi texty

DOCX – formát dokumentů Microsoft Word (Office Open XML)

EF Core – Entity Framework Core

GDPR – General Data Protection Regulation

HTML – HyperText Markup Language

HTTP/HTTPS – HyperText Transfer Protocol (Secure)

IIS – Internet Information Services (webový server Microsoftu)

IS – Informační systém

IS/STAG – Informační systém studijní agendy (na českých univerzitách)

JSON – JavaScript Object Notation

LINQ – Language Integrated Query

LMS – Learning Management System

MVC – Model–View–Controller

ORM – Object-Relational Mapping

SDK – Software Development Kit

SQL – Structured Query Language

UI – User Interface

UML – Unified Modeling Language

XSS – Cross-Site Scripting

TERMINOLOGIE

API (Application Programming Interface) – rozhraní, které umožňuje komunikaci mezi různými softwarovými komponentami.

ASP.NET Core MVC – webový framework postavený na architektonickém vzoru Model–View–Controller, který odděluje datovou, logickou a prezentační vrstvu aplikace.

ASP.NET Core Identity – framework v ASP.NET Core pro správu uživatelů, rolí, registraci a přihlašování.

Azure App Service – cloudová služba společnosti Microsoft určená pro hostování webových aplikací.

Bootstrap – CSS framework pro tvorbu responzivních a vizuálně konzistentních webových aplikací.

C# – objektově orientovaný programovací jazyk vyvinutý firmou Microsoft, používaný v rámci .NET platformy.

CSS (Cascading Style Sheets) – jazyk pro popis vzhledu a formátování webových stránek (barvy, rozložení, fonty).

Databáze – organizovaný soubor dat uložený v elektronické podobě, ke kterému lze přistupovat pomocí databázového systému.

Diff algoritmus – algoritmus pro porovnávání dvou verzí textu a zvýraznění rozdílů.

Digitalizace – převod analogových nebo papírových dat do digitální podoby.

Docker – nástroj pro virtualizaci a kontejnerizaci aplikací.

Entity Framework Core (EF Core) – objektově-relační mapovač (ORM) pro .NET, který mapuje databázové tabulky na objekty v C#.

Framework – softwarová kostra nebo platforma, která poskytuje základní komponenty a pravidla pro tvorbu aplikací.

GDPR (General Data Protection Regulation) – obecné nařízení EU o ochraně osobních údajů.

HTML (HyperText Markup Language) – značkovací jazyk pro strukturování obsahu webových stránek.

IIS (Internet Information Services) – webový server vyvinutý firmou Microsoft.

Information System (Informační systém) – komplexní propojení lidí, procesů a technologií za účelem sběru, ukládání a správy dat.

IS/STAG – univerzitní informační systém pro správu studijní agendy, využívaný na řadě českých vysokých škol.

JSON (JavaScript Object Notation) – lehký textový formát pro výměnu dat mezi aplikacemi.

Kubernetes – platforma pro orchestraci kontejnerů a automatizaci nasazování aplikací.

LINQ (Language Integrated Query) – dotazovací jazyk v C#, který umožňuje pracovat s daty přímo v kódu pomocí integrované syntaxe.

LMS (Learning Management System) – systém pro podporu výuky a vzdělávání (např. Moodle).

.NET 9 – vývojová platforma společnosti Microsoft, která umožňuje tvorbu multiplatformních aplikací.

Notifikace (Notification) – upozornění uživateli na důležitou událost (nový komentář, změna stavu práce).

OAuth / OpenID Connect – standardy pro správu identity a jednotné přihlášení uživatelů.

Open XML SDK / DocX – knihovny pro práci s dokumenty ve formátu DOCX.

ORM (Object-Relational Mapping) – přístup, kdy se databázové tabulky mapují na objekty v programovacím jazyce.

Razor Pages / Razor View – šablonovací technologie v ASP.NET pro dynamické generování HTML.

Relational Database (Relační databáze) – databázový systém založený na tabulkách a jejich vztazích.

SignalR – knihovna v ASP.NET Core pro real-time komunikaci mezi serverem a klientem.

SQL (Structured Query Language) – dotazovací jazyk pro práci s databázemi.

SQL Server (Microsoft SQL Server) – relační databázový systém od Microsoftu.

Task (Úkol, To-Do item) – entita systému, která reprezentuje požadavek k řešení v rámci tématu.

Thesis (Téma / práce) – entita systému reprezentující závěrečnou práci (bakalářskou nebo diplomovou).

To-Do seznam – seznam úkolů, kterými se řídí postup studenta na závěrečné práci.

UML (Unified Modeling Language) – standardizovaný jazyk pro modelování softwarových systémů.

User (Uživatel) – entita systému reprezentující studenta, učitele nebo administrátora.

Verzování dokumentů – uchovávání více verzí stejného dokumentu.

View – část architektury MVC, která zobrazuje data uživateli.

Workflow – proces řízení změn stavů (např. koncept → odevzdáno → schváleno / zamítnuto).

XSS (Cross-Site Scripting) – bezpečnostní útok spočívající ve vložení škodlivého skriptu do webové stránky.

CSRF (Cross-Site Request Forgery) – útok, při němž útočník zneužije přihlášení uživatele k nechtěným akcím.

ÚVOD

Závěrečné práce jsou nedílnou součástí vysokoškolského studia. Každá vysoká škola musí zajistit procesy, které studentům umožňují volbu témat, koordinaci s vedoucími prací a následné hodnocení oponenty. Nezbytnou součástí je rovněž archivace výsledných dokumentů a jejich dlouhodobé zpřístupnění akademické půdě i veřejnosti (Jelínek, 2018).

Historicky se správa závěrečných prací odehrávala převážně v papírové podobě, což přinášelo značnou administrativní zátěž, zvýšené riziko chyb v evidenci a omezené možnosti zpětného vyhledávání. V současnosti se do popředí dostává digitalizace, která umožňuje velkou část tohoto procesu realizovat prostřednictvím informačních systémů. Tyto systémy přinášejí vyšší efektivitu, usnadňují komunikaci a zvyšují transparentnost hodnocení (Laudon a Laudon, 2022). Jejich význam spočívá zejména v:

- zefektivnění administrativních procesů – snížení manuální zátěže a eliminace chyb,
- zajištění přehledné evidence – sledování stavu prací a jejich verzí,
- podpoře komunikace – mezi studenty, vedoucími a administrátory,
- transparentnosti hodnocení – jednoznačném přiřazení výsledků a komentářů,
- archivaci a zpřístupnění – bezpečném uložení a snadném vyhledávání prací i po delším časovém období.

Přestože již existují zavedená řešení, jako například univerzitní IS/STAG nebo systémy LMS typu Moodle, často se potýkají s omezeními v oblasti uživatelské přívětivosti, integrace a flexibility. To vytváří prostor pro návrh moderních řešení reflektujících aktuální potřeby akademického prostředí.

Cílem této bakalářské práce je proto navrhnout a implementovat demonstrační webovou aplikaci pro správu závěrečných prací, která bude vycházet z moderních technologií a současně reagovat na nedostatky stávajících systémů. Navrhovaný systém umožní evidenci témat, správu verzí dokumentů, přidávání komentářů a notifikací, a především integrovanou komunikaci mezi studenty a vedoucími přímo v prostředí aplikace. Vedle samotné aplikace je cílem práce také ukázat, jak lze vhodnou kombinací architektury MVC, databázového systému Microsoft SQL Server, technologie Entity Framework Core a webových standardů vytvořit udržitelné a rozšiřitelné řešení.

Práce je rozdělena do dvou hlavních částí. Teoretická část se věnuje obecnému vymezení informačních systémů, jejich charakteristikám, životnímu cyklu a bezpečnostním aspektům. Dále jsou popsány webové technologie a databázové přístupy využité při návrhu aplikace a provedena analýza existujících univerzitních řešení správy závěrečných prací. Praktická část je zaměřena na vlastní návrh a implementaci systému – od analýzy požadavků a návrhu architektury přes realizaci a testování až po možnosti budoucího rozvoje.

Tímto způsobem úvod poskytuje čtenáři nejen rámec problematiky a význam digitalizace závěrečných prací, ale také jasně definuje cíle práce a nastíní strukturu dalších kapitol.

1. TEORETICKÁ ČÁST

1.1 Informační systémy

Informační systém není pouze soubor technických prostředků či softwarových nástrojů. Jedná se o komplexně propojený celek, jehož úlohou je sběr, zpracování, ukládání a distribuce informací s cílem podpořit rozhodovací procesy a efektivitu činností uživatelů (Stair a Reynolds, 2020). V tomto smyslu zahrnuje informační systém nejen hardware a software, ale také uživatele, organizační procesy a zdroje dat, bez nichž by systém postrádal svůj smysl.

1.1.1 Definice a charakteristiky informačních systémů

Laudon a Laudon (2022) definují informační systém jako propojení lidí, technologií a procesů, které společně umožňují sběr, ukládání a analýzu informací. Mezi klíčové charakteristiky informačních systémů patří:

- integrace dat – propojení informací z různých zdrojů,
- automatizace procesů – snižování chybovosti a omezení manuální práce,
- podpora rozhodování – poskytování aktuálních a přesných informací,
- uživatelská přívětivost – intuitivní rozhraní a snadná ovladatelnost,
- bezpečnost – ochrana dat před ztrátou a neoprávněným přístupem.

1.1.2 Životní cyklus informačního systému

Životní cyklus informačního systému (System Development Life Cycle – SDLC) zahrnuje několik vzájemně navazujících fází:

- analýza požadavků – identifikace potřeb uživatelů a organizace,
- návrh systému – specifikace architektury a použitých technologií,
- implementace – vývoj a programování systému,
- testování – ověřování funkčnosti a bezpečnosti,
- provoz a údržba – nasazení systému do praxe a jeho dlouhodobá správa (Satzinger, Jackson a Burd, 2015).

V současné praxi jsou často využívány iterativní metodiky, například Agile nebo Scrum, které kladou důraz na flexibilitu a postupné nasazování jednotlivých částí systému (Cohn, 2010).

1.1.3 Informační systémy ve vzdělávání

Specifickou oblastí jsou informační systémy používané ve vzdělávání. Patří sem klasické Learning Management Systems (LMS), jako jsou Moodle nebo Blackboard, stejně jako systémy určené pro evidenci studentů a správu závěrečných prací. Tyto systémy poskytují například:

- evidenci studentů a jejich studijních výsledků,
- správu zadání a odevzdání prací,
- komunikaci mezi studenty a vyučujícími,
- archivaci závěrečných dokumentů,
- podporu hodnocení a kontrolu plagiátorství (Siemens a Long, 2011).

1.1.4 Přínosy a rizika informačních systémů

Implementace informačních systémů přináší řadu benefitů:

- rychlý a snadný přístup k informacím,
- snížení administrativní zátěže,
- vyšší přesnost a konzistence dat,
- dlouhodobá archivace a efektivní vyhledávání dokumentů.

Je však třeba reflektovat i možná rizika:

- závislost na technologiích,
- potřeba pravidelné údržby a aktualizací,
- potenciální hrozby v oblasti kybernetické bezpečnosti,
- náročnost zaškolení uživatelů při implementaci nových systémů (Stair a Reynolds, 2020).

Závěrem lze konstatovat, že informační systémy představují nezbytný nástroj moderní organizace, avšak jejich využívání vyžaduje odpovídající strategii a péči.

1.2 Analýza konkurenčních řešení

V současné době je správa závěrečných prací na českých vysokých školách zajišťována převážně prostřednictvím univerzitních informačních systémů nebo specializovaných

fakultních aplikací. Rozsah, technické provedení a uživatelská přívětivost jednotlivých řešení se výrazně liší. V následujícím přehledu jsou shrnuty nejvýznamnější systémy používané v akademickém prostředí, včetně analýzy jejich klíčových výhod a nevýhod. Součástí této analýzy bylo také oslovení vybraných univerzit s cílem zjistit, jakým způsobem u nich probíhá komunikace mezi studenty a vedoucími v rámci správy závěrečných prací. Výsledky ukázaly, že žádný ze zkoumaných systémů nenabízí integrovaný komunikační modul; veškerá komunikace probíhá mimo systém, zpravidla prostřednictvím e-mailu nebo osobních konzultací.

1.2.1 IS/STAG

Informační systém STAG patří mezi nejrozšířenější univerzitní informační systémy v České republice a využívá jej řada veřejných vysokých škol. Umožňuje komplexní správu studijních agend, včetně evidence předmětů, studijních plánů, docházky, hodnocení a závěrečných prací. Modul pro správu závěrečných prací zahrnuje možnost výběru a registrace témat studenty, schvalování zadání vedoucími a zadávání hodnocení.

Předností systému STAG je jeho robustnost a vysoká míra integrace s dalšími univerzitními procesy. Nevýhodou je naopak složitá orientace, značná administrativní zátěž a absence integrovaného komunikačního nástroje – veškerá interakce mezi studenty a vedoucími probíhá mimo systém. Při návrhu předkládaného prototypu byl STAG využit jako hlavní inspirační zdroj, nicméně některé funkce byly záměrně vynechány ve prospěch vyšší přehlednosti a uživatelské vstřícnosti.

1.2.2 Moodle (LMS)

Moodle představuje celosvětově rozšířený open-source systém pro podporu výuky (Learning Management System). Díky modulární architektuře umožňuje správu úkolů, odevzdávání souborů, testování a poskytování zpětné vazby mezi studenty a pedagogy. V univerzitním prostředí je často využíván jako doplněk k hlavním informačním systémům, zejména pro organizaci výukových aktivit.

Pro správu závěrečných prací není Moodle ideálním řešením – chybí mu například funkce pro registraci témat, přidělování vedoucích, workflow schvalování nebo sledování verzí dokumentů. Lze jej využít jako podpůrný nástroj, zejména pro sdílení souborů či zadávání průběžných úkolů, avšak i zde komunikace mezi studenty a vedoucími probíhá mimo systém, nejčastěji e-mailem.

1.2.3 IS MU a další univerzitní informační systémy

Příkladem komplexního univerzitního informačního systému je IS Masarykovy univerzity, který nabízí propracovaný modul pro evidenci závěrečných prací. Tento modul pokrývá celý životní cyklus práce – od zadání tématu přes výběr studentem až po odevzdání a archivaci, včetně kontroly originality prostřednictvím plagiátorských databází. Srovnatelné funkcionality nabízí i další velké univerzitní systémy, například UIS ZČU.

Hlavní výhodou těchto řešení je komplexnost a provázanost s ostatními agendami. Nevýhodou je omezená možnost přizpůsobení potřebám menších fakult či kateder a absence integrovaného komunikačního modulu – komunikace probíhá individuálně, mimo systém.

1.2.4 Fakultní a katedrové nástroje

Vedle univerzitních systémů existují i lokální aplikace vyvíjené na úrovni jednotlivých fakult nebo kateder. Tyto nástroje bývají zpravidla jednodušší, šité na míru konkrétním potřebám a vyvíjené interními IT odděleními. Umožňují například evidenci témat, přiřazení studentů nebo nahrávání souborů.

Jejich nevýhodou je omezený rozsah, problematická dlouhodobá udržitelnost, absence standardizace i integrace s ostatními univerzitními procesy a většinou také chybějící funkce pro integrovanou komunikaci. Stejně jako u robustnějších univerzitních systémů zůstává komunikace mezi studentem a vedoucím oddělena a řeší se převážně e-mailem či osobními konzultacemi.

1.2.5 Shrnutí

Z provedené analýzy vyplývá, že žádný z dostupných systémů nenabízí plně integrovanou komunikaci mezi studenty a vedoucími přímo v rámci správy závěrečných prací. Systémy typu IS/STAG a IS MU jsou robustní a funkčně bohaté, avšak složité a administrativně náročné. Moodle je vhodný pro výuku, ale z hlediska správy závěrečných prací neposkytuje potřebné funkce. Lokální fakultní a katedrové nástroje jsou sice flexibilní, ale postrádají dlouhodobou udržitelnost a komplexnost.

Na základě těchto poznatků byl navržen demonstrační prototyp, který záměrně neimplementuje veškeré funkce robustních systémů, ale klade důraz na zjednodušení, a především na integraci komunikačních možností přímo do prostředí správy závěrečných prací. Cílem je ukázat, že současnou praxi, kdy komunikace probíhá odděleně, lze nahradit efektivnějším a uživatelsky přívětivějším systémem.

1.3 Webové technologie

Webové technologie představují zásadní pilíř současného vývoje informačních systémů. Jsou klíčem k efektivní komunikaci mezi uživateli a serverem, umožňují prezentaci dat i interakci s uživatelským rozhraním. Pro tvorbu systému na správu závěrečných prací je nezbytné rozumět principům fungování webových aplikací, jejich architektuře i používaným technologiím.

1.3.1 Architektura klient–server

Většina webových aplikací stojí na architektuře klient–server. Uživatel pracuje s klientem, obvykle webovým prohlížečem, který odesílá požadavky na server. Server následně zpracuje požadavek, komunikuje s databází a vrací odpovědi ve formě webových stránek nebo dat. Toto oddělení prezentační a datové vrstvy je zásadní pro škálovatelnost a správu rozsáhlejších systémů (Tanenbaum a van Steen, 2017).

Klíčovou roli zde sehrávají protokoly:

- HTTP (HyperText Transfer Protocol) – základní protokol pro přenos dat na webu,
- HTTPS (HTTP Secure) – rozšíření HTTP, které díky SSL/TLS přináší šifrování a zajišťuje důvěrnost i integritu komunikace (Rescorla, 2018).

1.3.2 Frontendové technologie

Frontend označuje uživatelské rozhraní aplikace, které vzniká propojením několika zásadních technologií:

- HTML (HyperText Markup Language) – jazyk pro strukturování obsahu webových stránek (W3C, 2023a),
- CSS (Cascading Style Sheets) – nástroj pro vizuální stylování a formátování HTML prvků (W3C, 2023b),
- JavaScript – skriptovací jazyk, jenž umožňuje dynamické prvky a klientskou logiku webu (Flanagan, 2020),
- jQuery – JavaScriptová knihovna, která usnadňuje práci s DOM a AJAXem (Duckett, 2014),

- Bootstrap – CSS framework poskytující sadu grafických komponent a responzivní mřížkový systém (Otto a Thornton, 2021).

1.3.3 Backendové technologie

Na straně serveru probíhá zpracování logiky aplikace, přístup k databázi a řízení oprávnění. Backend typicky:

- přijímá požadavky od klienta,
- zpracovává aplikační logiku,
- komunikuje s databázovými systémy,
- vrací odpovědi ve formátu HTML nebo dat (nejčastěji JSON či XML).

V současnosti je rozšířená architektura REST API (Representational State Transfer), která definuje standardizovanou komunikaci mezi klientem a serverem prostřednictvím HTTP metod (GET, POST, PUT, DELETE). Přenášená data jsou typicky ve formátu JSON, což zajišťuje jejich jednoduchost, čitelnost i širokou podporu napříč programovacími jazyky (Bray, 2017).

1.3.4 Význam webových technologií pro informační systémy

Webové technologie přinášejí informačním systémům několik významných výhod:

- dostupnost odkudkoli – minimálními požadavky jsou webový prohlížeč a připojení k internetu,
- nezávislost na platformě – funkčnost na Windows, Linux i mobilních zařízeních,
- snadná údržba – změny a aktualizace probíhají na serveru a jsou okamžitě dostupné všem uživatelům,
- integrace – jednoduché propojení s dalšími systémy prostřednictvím API.

Souhrnně lze říci, že webové technologie představují vhodný základ pro vývoj aplikací na správu závěrečných prací, jelikož splňují požadavky na dostupnost, bezpečnost i možnost škálování.

1.4 Programovací technologie použité v práci

Při vývoji informačního systému pro správu závěrečných prací byl kladen důraz na využití moderních a osvědčených technologií, které zajistí dlouhodobou udržitelnost, stabilitu

i škálovatelnost řešení. Klíčovou roli v této architektuře zaujímá platforma .NET 9, framework ASP.NET Core MVC a programovací jazyk C#. Uživatelské rozhraní je realizováno pomocí standardních webových technologií.

1.4.1 Platforma .NET 9

Platforma .NET představuje robustní open-source vývojové prostředí společnosti Microsoft, umožňující vytváření multiplatformních aplikací napříč systémy Windows, Linux, macOS a mobilními zařízeními. Nejnovější verze .NET 9 (2024) přináší výrazné zlepšení výkonu, zvýšenou bezpečnost a rozšířenou podporu cloudových aplikací (Microsoft, 2025a).

Mezi hlavní přednosti platformy .NET 9 patří:

- multiplatformnost, která umožňuje nasazení na rozličných operačních systémech,
- vysoký výkon díky optimalizovanému běhovému prostředí CLR,
- podpora moderních standardů (REST API, gRPC, GraphQL),
- rozsáhlý ekosystém knihoven (NuGet),
- zpětná kompatibilita s předchozími verzemi .NET Core a .NET 5/6/7/8.

1.4.2 ASP.NET Core

Framework ASP.NET Core rozšiřuje základní možnosti .NET, zejména v oblasti vývoje webových aplikací. Architektonický vzor Model–View–Controller (MVC) představuje oddělení aplikační logiky, uživatelského rozhraní a řízení toku dat, což výrazně usnadňuje údržbu a rozšiřitelnost systému.

Mezi hlavní přínosy tohoto přístupu patří:

- jasné oddělení vrstev,
- jednodušší testování jednotlivých komponent,
- flexibilita při rozšiřování funkcionality,
- integrované nástroje pro autentizaci a autorizaci.

ASP.NET Core MVC podporuje také moderní technologie jako Razor Pages pro efektivní tvorbu dynamických webových stránek a poskytuje middleware komponenty pro správu požadavků (Microsoft, 2025b).

1.4.3 Programovací jazyk C#

Jazyk C# je objektově orientovaný programovací jazyk společnosti Microsoft, který od svého uvedení v roce 2000 získal významné postavení mezi vývojáři webových, desktopových a mobilních aplikací. Typické vlastnosti jazyka zahrnují:

- silnou typovou kontrolu, která snižuje chybovost,
- podporu objektově orientovaného přístupu (třídy, dědičnost, rozhraní),
- moderní syntaxi (lambdy, asynchronní programování pomocí async/await),
- těsnou integraci s platformou .NET,
- širokou komunitu a dostupnost podpůrných zdrojů.

C# se v praxi osvědčil jako vhodná volba pro robustní informační systémy, které vyžadují spolehlivost a dlouhodobou podporu (Albahari a Albahari, 2022).

1.4.4 Webové standardy v projektu

Na úrovni klientské části systému byly využity standardní webové technologie:

- HTML5 pro strukturování obsahu,
- CSS3 pro tvorbu vzhledu a responzivního designu,
- Bootstrap pro efektivní návrh uživatelského rozhraní,
- JavaScript a jQuery pro implementaci dynamických funkcí a práci s DOM,
- JSON pro přenos dat mezi klientem a serverem.

Tyto technologie představují základní technologický stack, na kterém je celý navržený informační systém postaven.

1.5 Databázové systémy

Základním pilířem každého informačního systému je schopnost efektivně pracovat s daty, zejména jejich ukládání, zpracování a využití. Ukládání, správa, aktualizace a bezpečný přístup k údajům jsou naprostým základem. K tomu lze využít především databázové systémy – poskytují potřebné nástroje pro organizaci a uložení dat, řízení přístupu a zajištění integrity informací.

1.5.1 Relační databázové systémy

Relační databázové systémy (RDBMS) jsou dnes nejrozšířenějším typem databází. Data jsou zde uložena v tabulkách propojených pomocí primárních a cizích klíčů. Relační model, popsaný poprvé Coddem v roce 1970, si získal oblibu zejména díky své jednoduchosti, univerzálnosti a možnostem normalizace.

Mezi hlavní přednosti relačních databází patří:

- přehledná struktura tabulek a jejich vzájemných vztahů,
- využití standardizovaného dotazovacího jazyka SQL,
- podpora transakcí a integritních omezení,
- možnost optimalizace výkonu prostřednictvím indexů.

1.5.2 SQL – Structured Query Language

SQL (Structured Query Language) je standardní jazyk určený pro práci s relačními databázemi. Umožňuje definici databázové struktury, manipulaci s daty i správu uživatelských přístupů. Obvyklé členění příkazů zahrnuje:

- DDL (Data Definition Language): CREATE, ALTER, DROP,
- DML (Data Manipulation Language): SELECT, INSERT, UPDATE, DELETE,
- DCL (Data Control Language): GRANT, REVOKE.
- SQL je dnes průmyslovým standardem a podporují jej všechny významné databázové platformy.

1.5.3 Microsoft SQL Server

V rámci této práce byl využit Microsoft SQL Server, což je robustní relační databázový systém vyvíjený společností Microsoft. Nabízí širokou škálu nástrojů pro správu dat, vysokou úroveň zabezpečení a snadnou integraci s platformou .NET.

Mezi hlavní vlastnosti SQL Serveru patří:

- Transact-SQL (T-SQL) – rozšířená implementace jazyka SQL,
- podpora uložených procedur a triggerů,
- integrované bezpečnostní mechanismy (autentizace, šifrování, role),

- škálovatelnost pro různé velikosti aplikací,
- nástroje pro správu a monitoring (SQL Server Management Studio – SSMS).

SQL Server je často využíván v akademickém prostředí díky provázanosti s ostatními technologiemi Microsoftu, dostupnosti výukových verzí a možnosti integrace s cloudovou platformou Azure SQL Database.

1.5.4 Objektově-relační mapování (ORM)

Moderní vývoj aplikací často spoléhá na objektově-relační mapování (ORM, Object-Relational Mapping), které zjednodušuje práci s databází převodem databázových tabulek na objekty programovacího jazyka. V prostředí .NET je nejčastěji využíván Entity Framework Core, který umožňuje:

- automatickou generaci databázového schématu podle tříd (Code First přístup),
- práci s daty pomocí LINQ dotazů,
- snadnou migraci databáze,
- zabezpečení proti SQL injection útokům.

Použití ORM významně urychluje vývoj a snižuje riziko chyb vznikajících při ručním psaní SQL dotazů.

1.6 Architektura aplikace MVC

Vývoj moderních webových aplikací se opírá o architektonické vzory, jejichž cílem je zajistit přehlednost, udržovatelnost a možnost dalšího rozšiřování kódu. Jedním z nejčastěji využívaných vzorů je Model–View–Controller (MVC), jenž se stal standardem v mnoha programovacích prostředích, včetně ASP.NET Core.

1.6.1 Princip MVC architektury

Architektura MVC rozděluje aplikaci do tří základních vrstev:

Model – reprezentuje datovou vrstvu. Obsahuje třídy a objekty odpovědné za uchovávání a zpracování dat. Model obvykle komunikuje s databázovými systémy nebo jinými úložišti dat.

View – představuje prezentační vrstvu. Zajišťuje vizualizaci dat uživateli, nejčastěji prostřednictvím HTML šablon a kaskádových stylů (CSS). View by neměla obsahovat aplikační logiku, pouze prezentaci dat.

Controller – slouží jako řídicí vrstva. Zpracovává požadavky od uživatelů, zajišťuje komunikaci mezi modelem a view a rozhoduje, která data a jakým způsobem budou zobrazena (Gamma et al., 1995).

Tato architektura umožňuje jasné oddělení odpovědností mezi jednotlivými částmi systému, což zjednodušuje údržbu a testování aplikace.

1.6.2 Výhody MVC

Implementace architektury MVC přináší několik klíčových výhod:

- Oddělení vrstev – změny v uživatelském rozhraní nevyžadují zásah do logiky aplikace a naopak,
- Snazší testování – jednotlivé části systému je možné testovat nezávisle,
- Rozšiřitelnost – architektura podporuje snadné přidávání nových funkcionalit bez zásadních zásahů do stávajícího kódu,
- Podpora týmové práce – více vývojářů může současně pracovat na různých vrstvách aplikace,
- Opětovná použitelnost kódu – modely a komponenty lze využít v různých částech projektu.

1.6.3 MVC v prostředí ASP.NET Core

ASP.NET Core MVC představuje implementaci architektury MVC v rámci .NET ekosystému. Nabízí například:

- Razor Pages – šablonovací systém umožňující kombinaci HTML a C#,
- Model Binding a Validation – automatické mapování dat z formulářů na objekty a jejich validaci.
- Middleware – komponenty pro zpracování HTTP požadavků v rámci aplikační pipeline,

- Dependency Injection – nástroj pro správu závislostí, který zvyšuje modularitu a testovatelnost systému,
- Routing – mechanismus směrování URL adres na příslušné controllery a akce (Microsoft, 2025b).

Tato architektura je vhodná například pro systém správy závěrečných prací, neboť poskytuje jasnou strukturu, podporuje škálování a umožňuje snadnou integraci s databázemi i moderními frontendovými technologiemi.

1.6.4 Alternativy k MVC

Kromě MVC existují i další architektonické vzory, například MVVM (Model–View–ViewModel), často využívané u desktopových aplikací, nebo MVP (Model–View–Presenter). Tyto varianty se zaměřují na odlišné aspekty interakce mezi uživatelským rozhraním a aplikační logikou. MVC však zůstává nejrozšířenějším přístupem pro webové aplikace, a to díky své jednoduchosti a univerzálnosti (Fowler, 2003).

1.7 Bezpečnost webových aplikací

Bezpečnost webových aplikací představuje zásadní aspekt jejich návrhu i provozu, obzvlášť s ohledem na ochranu citlivých údajů. Moderní informační systémy, například systémy pro správu závěrečných prací, typicky pracují s osobními daty studentů, akademických pracovníků či posudky oponentů. Zajištění bezpečnosti zde není pouze technickou nutností, ale i právní povinností, a to zejména v souvislosti s požadavky GDPR.

1.7.1 Klíčové hrozby webových aplikací

Projekt OWASP každoročně identifikuje nejčastější hrozby v oblasti webových aplikací. Mezi nejvýznamnější patří:

- SQL Injection – vložení škodlivého SQL kódu do vstupních polí aplikace, což může vést k neoprávněnému přístupu k databázi.
- Cross-Site Scripting (XSS) – injekce škodlivého JavaScript kódu, který je následně spuštěn v prohlížeči uživatele, což umožňuje například krádež dat.
- CSRF (Cross-Site Request Forgery) – situace, kdy je uživatel donucen neúmyslně provést akci v aplikaci, do níž je přihlášen.
- Nedostatečná autentizace a správa relací – umožňuje útočníkovi získat přístup k účtu jiného uživatele.

- Exponované citlivé údaje – například nešifrovaná hesla či osobní data.
- Chyby v konfiguraci – typicky špatně nastavená oprávnění nebo výchozí hesla.

1.7.2 Autentizace a autorizace

Základním bezpečnostním mechanismem aplikace je autentizace uživatele a následné určení jeho oprávnění (autorizace). Nejčastěji je autentizace realizována kombinací uživatelského jména a hesla, případně vícefaktorovou autentizací (MFA). Bezpečné ukládání hesel se zajišťuje pomocí hashovacích algoritmů, jako jsou bcrypt či Argon2.

V prostředí ASP.NET Core je autorizace často založena na rolích (např. student, vyučující, administrátor) nebo prostřednictvím tzv. policy-based authorization, která umožňuje přesně definovat pravidla přístupu k jednotlivým částem systému.

1.7.3 Bezpečný přenos dat

Veškerá komunikace mezi klientem a serverem by měla být chráněna pomocí protokolu HTTPS, který využívá šifrování prostřednictvím SSL/TLS. Tím je zajištěna:

- důvěrnost dat (ochrana před odposlechem),
- integrita dat (ochrana před modifikací během přenosu),
- ověření identity serveru pomocí certifikátu.

Používání HTTPS je zejména u aplikací pracujících s osobními údaji naprostou nezbytností.

1.7.4 Ochrana proti útokům a bezpečnostní opatření

Pro zvýšení bezpečnosti webové aplikace je vhodné implementovat následující opatření:

- Validace vstupů – důsledná kontrola dat zadaných uživateli,
- Omezení práv uživatelů – aplikace principu nejmenších oprávnění,
- Pravidelné aktualizace – systémových komponent, frameworků i knihoven,
- Monitoring a logování – sledování činností a detekce podezřelých aktivit,
- Ochrana proti útokům hrubou silou – omezení počtu pokusů o přihlášení,
- Content Security Policy (CSP) – omezení zdrojů externího obsahu.

1.7.5 Legislativa a GDPR

Systémy pracující s osobními údaji musejí striktně dodržovat legislativu na ochranu osobních údajů. V Evropské unii je základním předpisem Obecné nařízení o ochraně osobních údajů (GDPR). To stanovuje pravidla pro shromažďování, zpracování a ukládání dat, včetně povinnosti informovat uživatele o účelu zpracování a umožnit jim nad svými údaji kontrolu.

Bezpečnost webových aplikací je tedy komplexní téma, které vyžaduje systematický přístup nejen z technického, ale i z právního hlediska.

1.8 Shrnutí teoretické části

V teoretické části práce byly nejprve definovány základní pojmy, principy a technologie, které jsou klíčové pro návrh a realizaci informačního systému určeného ke správě závěrečných prací. Bez těchto základů bychom se v této oblasti neobešli.

Úvodní kapitola se věnuje samotné problematice správy závěrečných prací, a zároveň poukazuje na význam digitalizace v akademickém prostředí. Důraz byl kladen na potřebu efektivních nástrojů pro evidenci, hodnocení a archivaci těchto prací. Zmíněny jsou rovněž přínosy informačních systémů pro vzdělávání jako takové – jejich nasazení totiž přináší vyšší úroveň organizace a transparentnosti.

Následující kapitoly poskytují přehled o informačních systémech obecně – jejich charakteristice, životním cyklu, výhodách i možných rizicích. Vysvětluji zde také úlohu webových technologií a princip architektury klient–server, které představují technologický základ moderních aplikací.

Podrobně jsou také popsány programovací technologie využité v rámci tohoto projektu, zejména platformu .NET 9, framework ASP.NET Core MVC, jazyk C# a také doplňkové webové technologie jako HTML, CSS, JavaScript, jQuery a Bootstrap. Samostatná pozornost byla věnována databázovým systémům, s důrazem na relační databáze a konkrétně Microsoft SQL Server, využívaný v kombinaci s Entity Framework Core.

Nejdůležitější část teoretického rámce tvořila analýza architektury MVC, která umožňuje oddělení jednotlivých vrstev aplikace a podporuje udržitelnost i rozšiřitelnost systému. Jedna z kapitol se zaměřila také na bezpečnost webových aplikací – popsány byly nejčastější hrozby, principy bezpečného návrhu a legislativní požadavky, které vyplývají z GDPR.

Výsledkem této teoretické části je ucelený souhrn znalostí a technologií, na kterých je možné stavět praktickou část práce. Ta se následně zaměří na konkrétní návrh, implementaci a testování systému pro správu závěrečných prací, a to na základě principů a poznatků shrnutých v této kapitole.

2. PRAKTICKÁ ČÁST

2.1 Úvod do praktické části

Cílem praktické části bylo navrhnout a implementovat prototyp informačního systému pro správu závěrečných prací, který bude odpovídat potřebám akademického prostředí. Tento systém je zaměřen na zefektivnění správy diplomových a bakalářských prací, zajištění transparentní komunikace mezi studenty a vedoucími a vytvoření centrálního prostoru pro ukládání verzí dokumentů.

Při návrhu systému bylo vycházeno z funkcionalit informačního systému STAG, který je v českém prostředí nejrozšířenější. Řada funkcí však byla záměrně vypuštěna, aby vynikl demonstrační charakter prototypu. Cílem bylo ukázat, že i zjednodušená verze systému může přinést zefektivnění práce, a především usnadnění komunikace mezi studentem a vedoucím, což současné robustní univerzitní systémy většinou postrádají.

Na rozdíl od teoretické části, která se zabývala analýzou dostupných řešení a technologií, je praktická část zaměřena na vlastní realizaci. Postupně jsou popsány fáze vývoje od analýzy požadavků, přes návrh architektury a databázového modelu až po implementaci a testování. Součástí kapitoly je i popis možností nasazení a budoucího rozvoje systému.

V průběhu vývoje byla použita moderní technologie ASP.NET Core MVC ve verzi .NET 9 v kombinaci s databázovým systémem Microsoft SQL Server a knihovnou Entity Framework Core. Pro autentizaci a autorizaci uživatelů byl využit framework Identity, který je integrován přímo do platformy. Důraz byl kladen na modulární architekturu, která umožňuje rozšiřitelnost do budoucna, a na využití real-time komunikace prostřednictvím knihovny SignalR.

Výsledný prototyp zahrnuje podporu pro zadávání a schvalování témat, správu verzí dokumentů, jejich porovnávání, přidávání komentářů a úkolů a doručování notifikací. Implementováno je rovněž administrátorské rozhraní, které poskytuje kontrolu nad uživateli a celým obsahem systému.

Praktická část je strukturována do několika podkapitol, které odpovídají jednotlivým fázím životního cyklu softwarového vývoje. Každá kapitola je doplněna ilustrativními příklady v podobě zdrojového kódu, diagramů nebo ukázek z webového rozhraní, které dokumentují funkčnost systému.

2.2 Použité technologie a knihovny

Pro realizaci prototypu byly zvoleny moderní technologie, které se osvědčily v akademickém i komerčním prostředí a zároveň poskytují dostatečnou flexibilitu pro budoucí rozšíření. Následující přehled shrnuje klíčové technologie a jejich roli v systému.

2.2.1 ASP.NET Core MVC (.NET 9)

ASP.NET Core MVC je základním stavebním kamenem aplikace. Poskytuje architektonický vzor Model–View–Controller, který zajišťuje oddělení aplikační logiky, datového modelu a prezentační vrstvy. Díky tomu je možné systém snadno udržovat a rozšiřovat.

2.2.2 Entity Framework Core

Pro práci s databází byl zvolen Entity Framework Core, což je objektově-relační mapovací nástroj (ORM). Umožňuje snadné mapování databázových tabulek na objekty v jazyce C# a práci s daty pomocí LINQ dotazů. EF Core navíc podporuje migrace, které zajišťují konzistentní správu schématu databáze.

```
using Microsoft.EntityFrameworkCore;
using Bakalarska_prace.Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

namespace Bakalarska_prace.Data
{
    Počet odkazů: 29
    public class AppDbContext : IdentityDbContext<ApplicationUser>
    {
        Počet odkazů: 0
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
        {
        }
        Počet odkazů: 10
        public DbSet<Thesis> Tema { get; set; }
        Počet odkazů: 2
        public DbSet<Document> Dokument { get; set; }
        Počet odkazů: 1
        public DbSet<Coment> Komentar { get; set; }
        Počet odkazů: 4
        public DbSet<TaskItem> TodoItems { get; set; }
        Počet odkazů: 1
        public DbSet<Workstation> Pracoviste { get; set; }
        Počet odkazů: 5
        public DbSet<Notification> Notifications { get; set; }
```

Obrázek 1 - Ukázka kódu Entity Framework Core

2.2.3 Microsoft SQL Server

Jako databázový systém byl zvolen Microsoft SQL Server, který poskytuje spolehlivé úložiště pro data, podporuje rozsáhlé transakce a je úzce propojen s platformou .NET. SQL Server je robustním řešením vhodným i pro nasazení na univerzitní úrovni.

2.2.4 ASP.NET Core Identity

Pro autentizaci a autorizaci uživatelů byl použit framework Identity, který umožňuje správu účtů, rolí a přihlášení. Identity podporuje také rozšíření o externí poskytovatele (OAuth, OpenID Connect), což je důležité pro budoucí integraci se systémem jednotného přihlášení univerzity.

```
builder.Services.AddDefaultIdentity<ApplicationUser>(options =>
{
    options.Password.RequiredLength = 6;
    options.SignIn.RequireConfirmedAccount = true;
})
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<AppDbContext>();
```

Obrázek 2 - Zdrojový kód nastavení identity uživatele

2.2.5 SignalR

Pro zajištění real-time komunikace a notifikací byl implementován SignalR. Tento framework umožňuje okamžité doručování zpráv mezi serverem a klientem bez nutnosti opakovaného načítání stránky. SignalR je využíván například při přidání komentáře, změně stavu tématu nebo vytvoření úkolu.

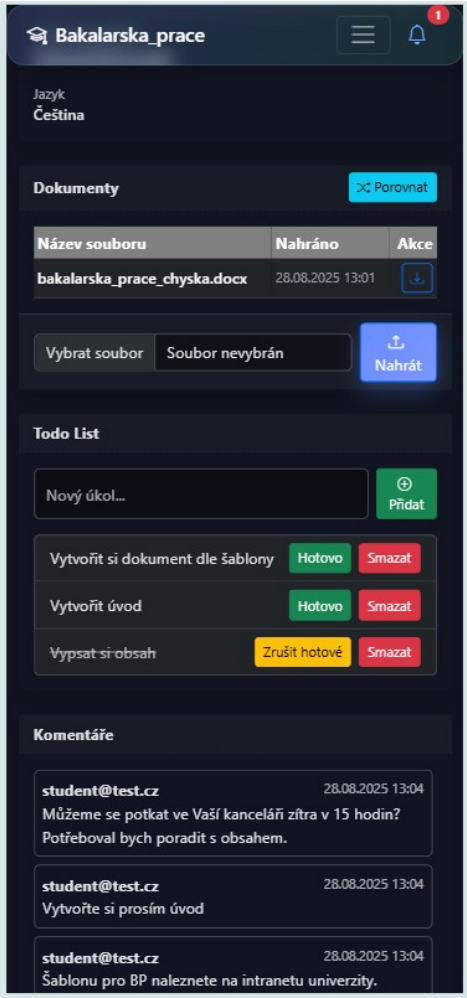
Obrázek 3 - Snímek notifikačního okna

2.2.6 Open XML SDK / DocX

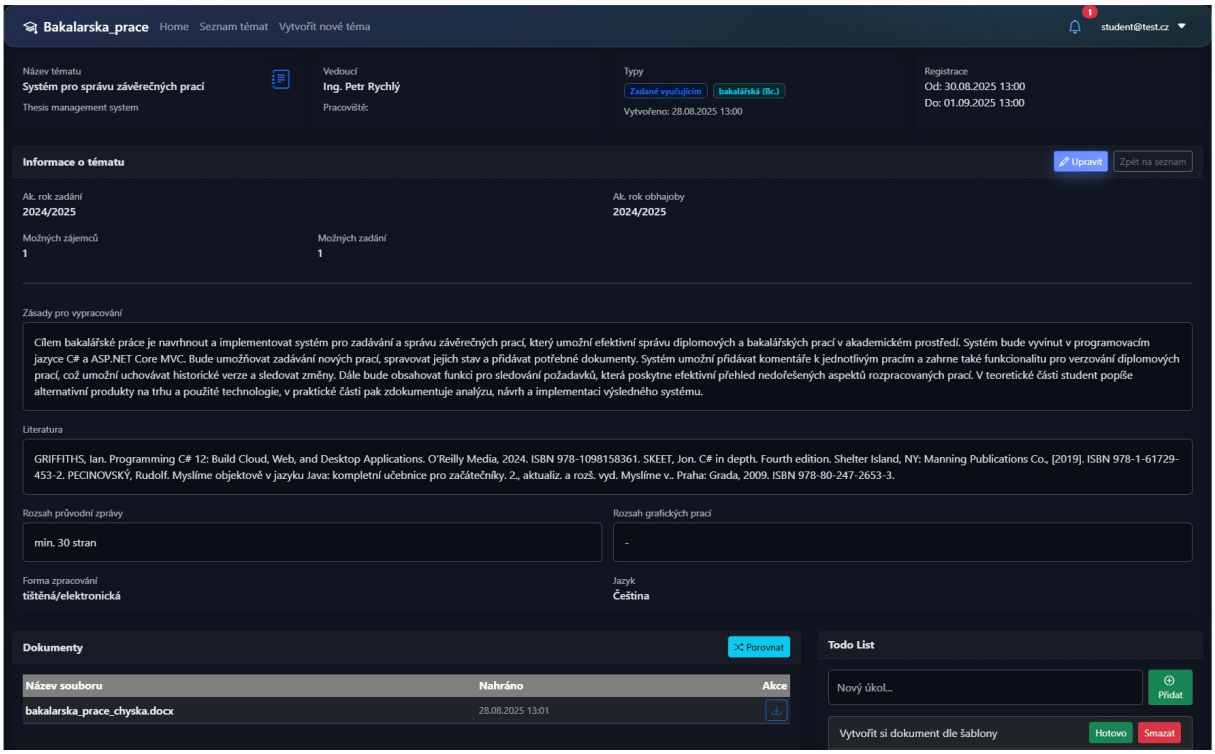
Knihovny Open XML SDK a DocX byly použity pro práci s dokumenty ve formátu DOCX. Umožňují extrakci textu a jeho porovnávání mezi verzemi, což je zásadní pro sledování změn v průběhu psaní závěrečné práce.

2.2.7 Bootstrap a CSS

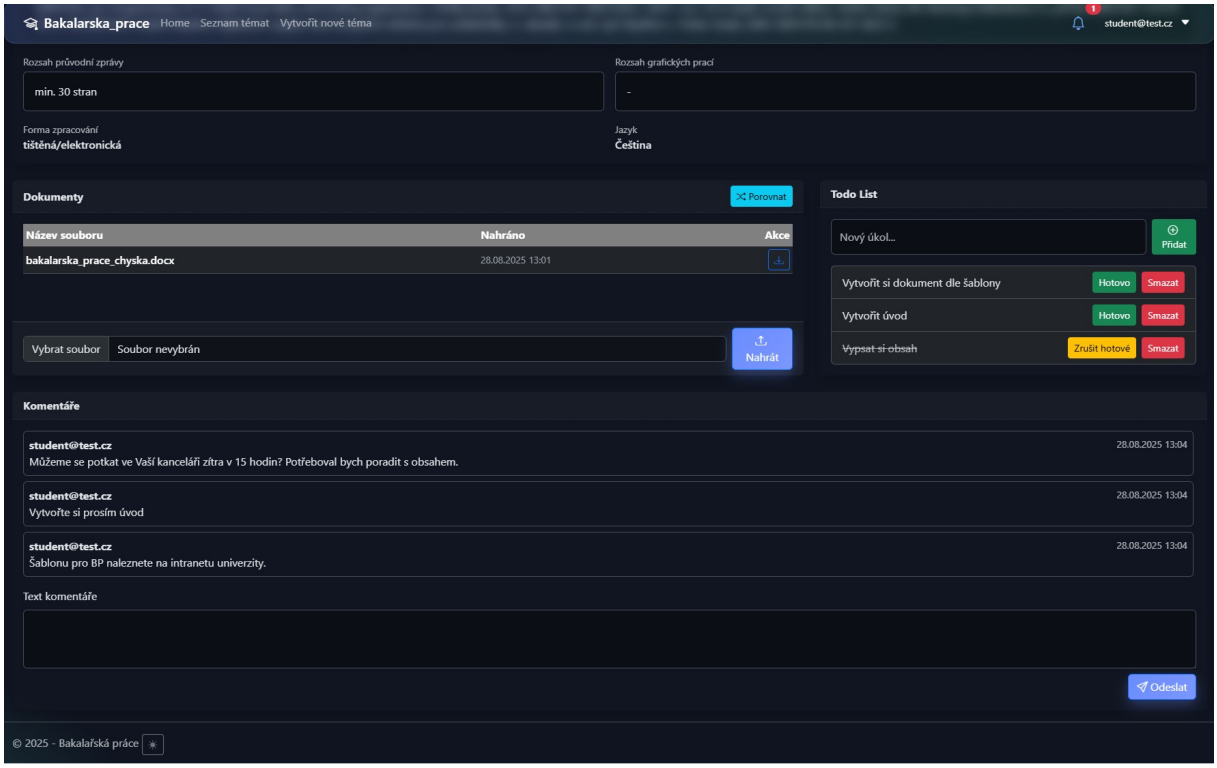
Pro návrh uživatelského rozhraní byl využit framework Bootstrap 5 doplněný o vlastní kaskádové styly. Díky tomu je aplikace responzivní a dobře použitelná jak na stolních počítačích, tak na mobilních zařízeních, které postupně dominují v žebříčku zobrazovacích zařízení.



Obrázek 4 - Responzivní zobrazení detailu témata



Obrázek 5 - Zobrazení detailu tématu na monitoru se standardním rozlišením 1



Obrázek 6 - Zobrazení detailu tématu na monitoru se standardním rozlišením 2

2.3 Analýza požadavků

Analýza požadavků je klíčovým krokem při vývoji systému. Jejím cílem je popsat, co systém musí umět, kdo jej bude používat a jaké má mít kvalitativní vlastnosti. Na základě analýzy vznikají funkční i nefunkční požadavky a scénáře použití.

2.3.1 Stakeholdeři a role uživatelů

Systém je určen pro tři základní skupiny uživatelů:

- **Student** – odevzdává témata závěrečných prací, nahrává verze dokumentů, reaguje na komentáře a sleduje přiřazené úkoly.
- **Teacher (vedoucí práce)** – vytváří témata, přiřazuje řešitele, přidává komentáře, zadává úkoly a hodnotí průběh práce.
- **Administrator** – spravuje uživatelské účty a role, má přístup k administrátorskému rozhraní, dohlíží na správu dat a konzistenci systému.

2.3.2 Funkční požadavky

Hlavní funkce systému zahrnují:

1. Zadání nové práce (téma).
2. Úpravu stávající práce podle role.
3. Přidávání komentářů k tématům a dokumentům.
4. Nahrávání a verzování dokumentů ve formátu DOCX.
5. Porovnání dvou verzí dokumentu a zvýraznění rozdílů.
6. To-Do seznam úkolů se stavem, prioritou a termínem.
7. Notifikace uživatelů při důležitých změnách.
8. Administrátorské rozhraní pro správu uživatelů a dat.
9. Registrace a přihlášení pomocí ASP.NET Core Identity.

2.3.3 Nefunkční požadavky

Kromě funkčnosti byly definovány i požadavky na kvalitu:

- **Bezpečnost** – role-based přístup, ochrana proti útokům (XSS, CSRF).
- **Rozšiřitelnost** – modulární návrh s možností integrace (např. OAuth).

- **Použitelnost** – přehledné a responzivní uživatelské rozhraní.
- **Výkon** – rychlé načítání seznamů a práce s větším objemem dokumentů.
- **Auditovatelnost** – evidence historie změn a verzí dokumentů.

2.3.4 Případové scénáře

Byly definovány následující modelové scénáře:

- **Teacher vytvoří téma** → vyplní formulář → uloží
- **Teacher může přiřadit studenta k tématu** → student dostane notifikaci o výsledku.
- **Student nahraje novou verzi dokumentu** → systém ji očísluje a uloží → vedoucí obdrží upozornění.
- **Teacher porovná dvě verze dokumentu** → systém zvýrazní rozdíly.
- **Teacher přidá úkol do To-Do seznamu** → student vidí termín a prioritu → po splnění označí jako hotové.

2.4 Návrh architektury

Návrh architektury určuje, jak budou jednotlivé části systému vzájemně propojeny a jakým způsobem bude realizována komunikace mezi uživateli, databází a aplikační logikou. Architektura byla navržena s důrazem na modularitu, bezpečnost a možnost budoucího rozšíření.

2.4.1 Vrstvy systému

Aplikace je založena na architektonickém vzoru Model–View–Controller (MVC):

- **Model** – definuje datové entity a jejich vazby, reprezentuje strukturu databáze.
- **View** – tvoří prezentační vrstvu pomocí Razor šablon a zobrazuje data uživateli.
- **Controller** – zpracovává požadavky uživatele, volá logiku servisních tříd a vrací odpovídající pohledy.

Nad MVC je doplněna **servisní vrstva**, která obsahuje logiku pro práci s dokumenty, verzováním, notifikacemi, schvalovacím workflow a To-Do seznamem.

2.4.2 Servisní vrstva

Servisní vrstva odděluje komplexní logiku od controllerů. Obsahuje služby jako:

- DocumentService (uložení, verzování, metadata),
- DiffService (porovnávání dokumentů),
- NotificationService (SignalR + databáze),
- WorkflowService (přechody stavů tématu),
- TaskService (správa úkolů).

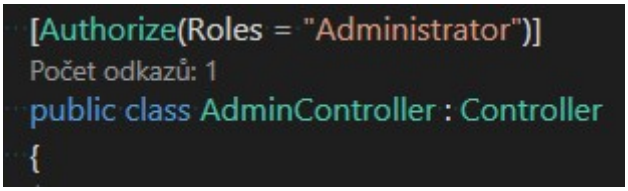
Tento přístup zvyšuje přehlednost a usnadňuje testování.

2.4.3 Autentizace a autorizace

Autentizace je realizována pomocí ASP.NET Core Identity. Uživatelé mají přiřazené role (Student, Teacher, Administrator), které určují jejich oprávnění. Autorizační pravidla jsou implementována na úrovni controllerů a metod.

Například:

- Student může okomentovat a vidět soubory pouze u svého tématu.
- Teacher může přiřadit studenty k tématu.
- Administrator má plný přístup ke správě uživatelů a dat.



Obrázek 7 - Zdrojový kód pro omezení přístupu dle uživatelské role, konkrétně role Administrator

2.4.4 Architektura databáze

Databáze je navržena v Microsoft SQL Serveru s využitím EF Core migrací. Hlavní tabulky jsou:

- **Users** – rozšíření IdentityUser, obsahuje uživatele systému.
- **Theses** – témata závěrečných prací.
- **Documents** – nahrané dokumenty k tématům.
- **Comments** – komentáře k tématům i dokumentům.
- **Tasks** – To-Do položky.

- **Notifications** – uživatelské notifikace.
- **Departments** – číselník pracovišť.

2.4.5 Vazby mezi entitami

- Uživatel – Téma (1:N) – uživatel může být autorem nebo vedoucím více témat.
- Téma – Dokument (1:N) – jedno téma obsahuje více dokumentů.
- Dokument – Komentář (1:N) – jeden dokument může mít více komentářů.
- Téma – Úkol (1:N) – k jednomu tématu může existovat více úkolů.
- Uživatel – Notifikace (1:N) – uživatel může mít více notifikací.

Tento model pokrývá všechny potřebné procesy tohoto prototypu.

2.4.6 Bezpečnost architektury

Architektura je navržena s ohledem na bezpečnost:

- role-based přístup ke controllerům,
- validace vstupů na více úrovních,
- ochrana proti CSRF útokům,
- možnost integrace SSO (OAuth, OpenID Connect).

2.5 Datový model a databázové schéma

Datový model je klíčovým prvkem systému, protože definuje způsob ukládání a organizace informací o závěrečných pracích. Návrh modelu vychází z analýzy požadavků a reflektuje entity a vazby, které odpovídají reálným procesům akademického prostředí.

2.5.1 Hlavní entity

- **User (ApplicationUser)** – rozšíření IdentityUser, obsahuje osobní údaje a role (Student, Teacher, Administrator).
- **Department** – číselník pracovišť, ke kterému je přiřazeno téma.
- **Thesis (Téma)** – klíčová entita reprezentující závěrečnou práci. Obsahuje název, anotaci, klíčová slova, stav a odkazy na autora i vedoucího.
- **Document** – nahraný dokument ve formátu DOCX, který patří k tématu. Uchovává metadata (název, cesta, číslo verze, datum nahrání, autor).

- **Comment** – komentář navázaný na téma nebo dokument, slouží pro zpětnou vazbu a komunikaci.
- **Task (To-Do item)** – úkol související s tématem, obsahuje popis, prioritu, stav a termín splnění.
- **Notification** – upozornění doručované uživatelům. Obsahuje text, typ, datum vytvoření a příznak přečtení.

2.5.2 Vazby mezi entitami

- **User – Thesis** – uživatel může být autorem (studentem) nebo vedoucím (teacherem) více témat.
- **Thesis – Document** – k jednomu tématu lze nahrát více dokumentů.
- **Document – Comment** – dokument může mít více komentářů.
- **Thesis – Task** – k jednomu tématu lze přiřadit více úkolů.
- **User – Notification** – každý uživatel může obdržet více notifikací.

Tento model odpovídá reálným vztahům a umožňuje efektivní správu verzí i zpětné vazby.

2.5.3 Integritní omezení

Databázový model zahrnuje i omezení pro zajištění konzistence dat:

- Číslo verze dokumentu je unikátní v rámci jednoho tématu.
- Komentář musí mít vždy přiřazeného autora.
- Úkol je vždy navázán na konkrétní téma.
- Notifikace musí být přiřazena konkrétnímu uživateli.

2.5.4 Optimalizace výkonu

Pro časté operace byly navrženy indexy (např. stav tématu, datum nahrání dokumentu, uživatel). Datový model také využívá lazy loading a explicitní načítání podle potřeby, aby nedocházelo k nadbytečnému zatížení databáze.

2.5.5 Migrace a inicializace dat

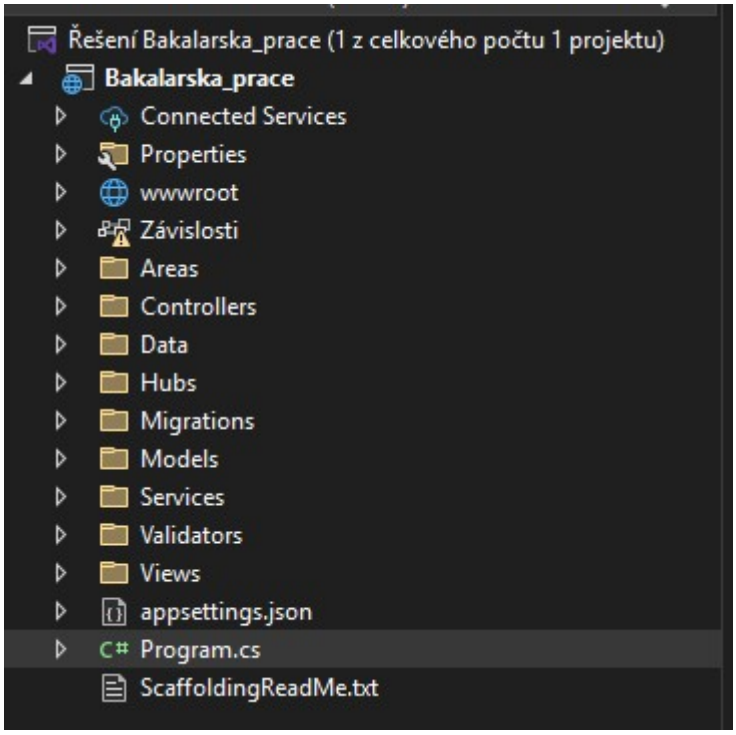
Entity Framework Core poskytuje mechanismus migrací, který umožňuje postupné vytváření a úpravy databázového schématu. Součástí projektu je i inicializace základních dat: vytvoření rolí (Student, Teacher, Administrator) a administrátorského účtu.

2.6 Implementace

Implementace představuje klíčovou fázi vývoje, ve které se návrh architektury a datového modelu převádí do funkční podoby aplikace. Tato kapitola popisuje postupné kroky realizace systému, od inicializace projektu až po vytvoření uživatelského rozhraní.

2.6.1 Inicializace projektu a konfigurace prostředí

Projekt byl vytvořen ve Visual Studiu jako ASP.NET Core MVC aplikace s využitím .NET 9. Do projektu byly přidány balíčky pro Entity Framework Core, Identity a SignalR.



Obrázek 8 - Struktura vytvořeného projektu v programu Visual Studio 2022

Konfigurace proběhla v souboru Program.cs, kde byly registrovány služby: Identity, DbContext, SignalR hub a aplikační služby.

```
// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddRazorPages();
builder.Services.AddControllersWithViews()
    .AddDataAnnotationsLocalization()
    .AddViewLocalization();
builder.Services.AddScoped<NotificationService>();
builder.Services.AddSignalR();
```

Obrázek 9- Zdrojový kód registrace services

2.6.2 Datová vrstva

Pro práci s databází byl vytvořen kontext ApplicationDbContext, který zahrnuje všechny entity (ApplicationUser, Thesis, Document, Comment, TaskItem, Notification, DocumentDiff, Workstation).

```
Počet odkazů: 10
public DbSet<Thesis> Tema { get; set; }
Počet odkazů: 2
public DbSet<Document> Dokument { get; set; }
Počet odkazů: 1
public DbSet<Coment> Komentar { get; set; }
Počet odkazů: 4
public DbSet<TaskItem> TodoItems { get; set; }
Počet odkazů: 1
public DbSet<Workstation> Pracoviste { get; set; }
Počet odkazů: 5
public DbSet<Notification> Notifications { get; set; }
```

Obrázek 10 - Zdrojový kód vytvoření databázových objektů dle modelů v projektu

Dotazy do databáze jsou realizovány pomocí LINQ, což umožňuje jednoduché filtrování, řazení a projekci dat.

```
[HttpGet]
Počet odkazů: 0
public async Task<IActionResult> StudentsByPracoviste(int pracId)
{
    var studentsInRole = await _userManager.GetUsersInRoleAsync("Student");

    var users = studentsInRole
        .Where(u => u.PracovisteId == pracId)
        .OrderBy(u => u.UserName ?? u.Email)
        .Select(u => new
        {
            id = u.Id,
            text = u.UserName ?? u.Email
        })
        .ToList();

    return Json(users);
}
```

Obrázek 11 - Zdrojový kód využívající knihovnou LINQ pro výběr dat z databáze bez nutnosti použití SQL dotazů

2.6.3 Servisní vrstva

Pro oddělení aplikační logiky byla vytvořena servisní vrstva. Obsahuje třídy:

- DocumentService – správa ukládání souborů, verzování a metadat.
- DiffService – porovnání dvou verzí dokumentů.
- NotificationService – generování a odesílání notifikací (SignalR + databáze).
- WorkflowService – implementace stavového automatu pro téma.
- TaskService – správa To-Do seznamu.

```
/// <summary>
/// Vytvori notifikaci pro konkretniho uzivatele a odesle ji pres SignalR
/// </summary>
Počet odkazů: 1
public async Task CreateNotificationAsync(int temaId, string recipientUserId, NotificationType type, string message)
{
    var notif = new Notification
    {
        TemaId = temaId,
        UserId = recipientUserId,
        Type = type,
        Message = message,
        CreatedAt = DateTime.Now,
        IsRead = false
    };

    _db.Notifications.Add(notif);
    await _db.SaveChangesAsync();

    await _hub.Clients.User(recipientUserId).SendAsync("notificationReceived", new
    {
        id = notif.Id,
        temaId = temaId,
        type = type.ToString(),
        message,
        createdAt = notif.CreatedAt
    });
}
```

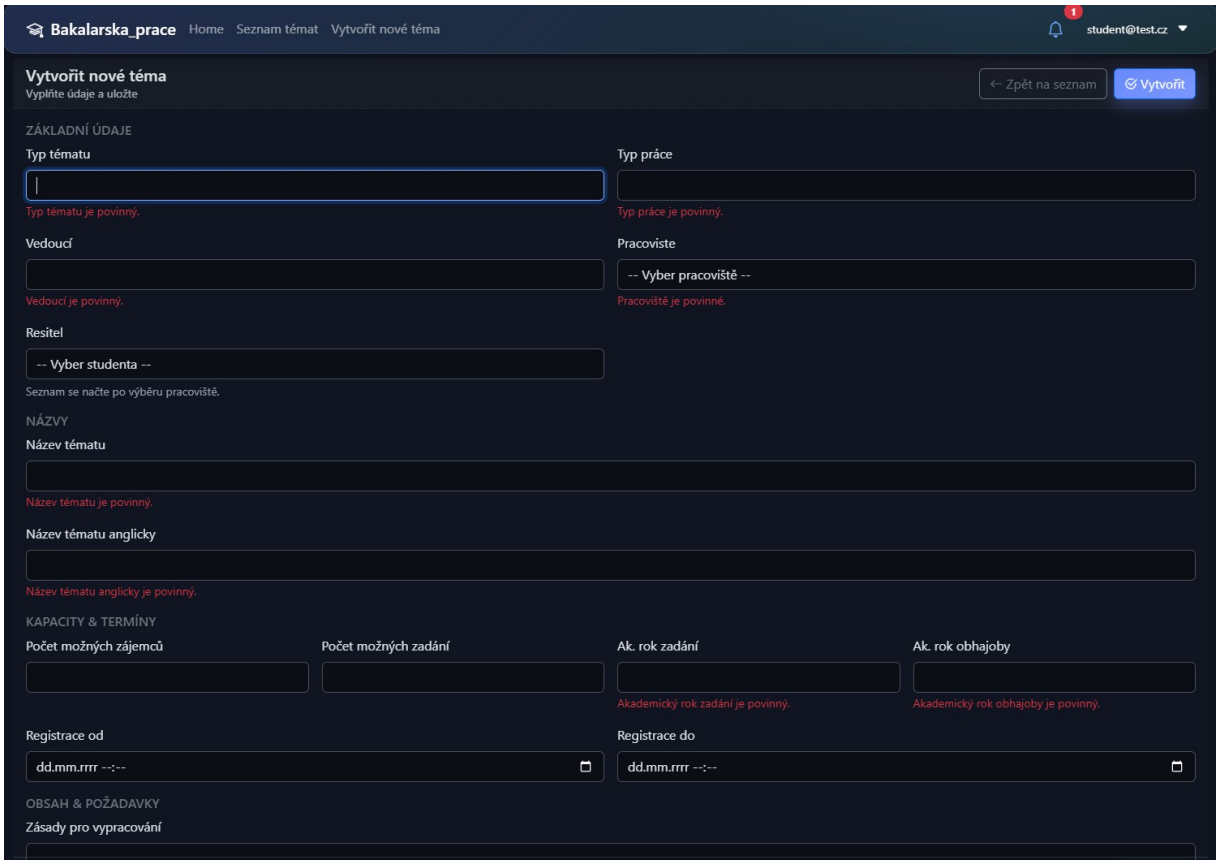
Obrázek 12 - Zdrojový kód metody CreateNotificationAsync, která zajišťuje vytvoření a odeslání notifikace konkrétnímu adresátovi

2.6.4 Uživatelské rozhraní a controllery

Uživatelské rozhraní je realizováno pomocí Razor šablon a Bootstrapu. Pro správu jednotlivých částí systému byly vytvořeny controllery:

- ThesisController – správa témat a dokumentů.

- TaskController – správa To-Do seznamu.
- NotificationController – práce s notifikacemi.
- AdminController – administrátorské prostředí.



Obrázek 13 - Snímek obrazovky formuláře pro nové téma s využitím validace

2.6.5 Vyhledávání v seznamu témat

Uživatel má možnost full textově vyhledávat ve výpisu témat, a to dle názvu témata, vedoucího tématu, pracoviště atd.

2.6.6 Ukládání a verzování dokumentů

Dokumenty jsou ukládány do adresářů na serveru podle identifikátoru tématu a čísla verze. V databázi se ukládají pouze metadata.

Obrázek 14 - Detail témata s verzováním nahraných dokumentů

2.6.7 Porovnání verzí dokumentů

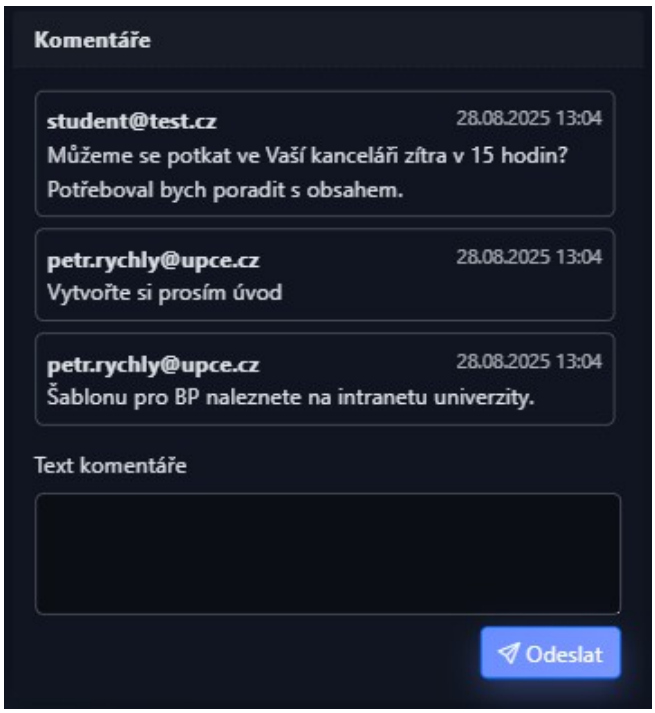
Pro porovnání verzí se využívá extrakce textu pomocí OpenXML SDK a algoritmu diff. Rozdíly jsou zobrazeny barevně přímo v uživatelském rozhraní.

```
[HttpGet]
Počet odkazů: 0
public async Task<IActionResult> CompareDocumentsPreview(int temaId, int firstDocId, int secondDocId)
{
    var doc1 = await _context.Dokument.FindAsync(firstDocId);
    var doc2 = await _context.Dokument.FindAsync(secondDocId);
    if (doc1 == null || doc2 == null) return NotFound();

    var path1 = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", doc1.CestaKSouboru.TrimStart('/'));
    var path2 = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", doc2.CestaKSouboru.TrimStart('/'));

    var text1 = ExtractTextFromWord(path1);
    var text2 = ExtractTextFromWord(path2);

    var differ = new Differ();
    var inlineBuilder = new InlineDiffBuilder(differ);
    DiffPaneModel diffModel = inlineBuilder.BuildDiffModel(text1, text2);

    var diffHtml = GenerateDiffHtml(diffModel);

    var model = new DocumentDiff
    {
        FirstDocumentName = doc1.NazevSouboru,
        SecondDocumentName = doc2.NazevSouboru,
        DiffHtml = diffHtml
    };

    return PartialView("_DocumentComparison", model);
}
```

Obrázek 15 - Zdrojový kód metody CompareDocumentsPreview, která zajišťuje vytvoření modelu obsahující rozdíl dvou dokumentů

2.6.8 Komentáře a spolupráce

Komentáře mohou být přidány k tématu i k dokumentu. Vytvářejí diskusní vlákna, která zajišťují komunikaci mezi studentem a vedoucím. V rámci každého komentáře je evidován odesílatel, text komentáře a datum včetně času odeslání.



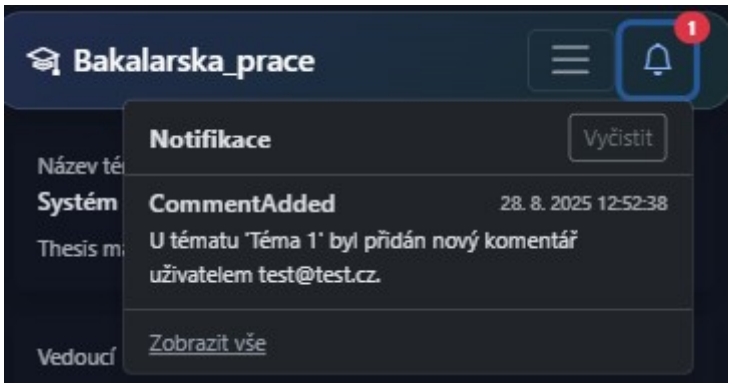
Obrázek 16 - Snímek obrazovky komentářů v detailu témata

2.6.9 Notifikace

Notifikace jsou ukládány v databázi, aby byly k dispozici i v případě, že uživatel nebyl přihlášen a zároveň doručovány v reálném čase pomocí SignalR. Uživatel je o nové notifikaci informován pomocí ikony a počtu nových notifikací v hlavičce stránky. Uživatel má možnost označit jednu či všechny notifikace jako přečtené.

```
/// <summary>
/// Pomocna metoda: pro dane tema rozesle notifikaci vsem adresatum krome odesilatele
/// </summary>
Počet odkazů: 1
public async Task NotifyTemaParticipantsExceptActorAsync(int temaId, string actorUserId, NotificationType type, string message)
{
    var tema = await _db.Tema.AsNoTracking().FirstOrDefaultAsync(t => t.Id == temaId);
    if (tema == null) return;

    var recipientIds = new List<string?>();
    if (!string.IsNullOrEmpty(tema.VedouciUserId)) recipientIds.Add(tema.VedouciUserId);
    if (!string.IsNullOrEmpty(tema.ResitelUserId)) recipientIds.Add(tema.ResitelUserId);

    var finalRecipients = recipientIds
        .Where(id => !string.IsNullOrEmpty(id) && id != actorUserId)
        .Distinct()
        .ToList();

    foreach (var uid in finalRecipients)
    {
        await CreateNotificationAsync(temaId, uid!, type, message);
    }
}
```

Obrázek 17 - Zdrojový kód metody odesílající notifikaci všem odpovídajícím adresátům



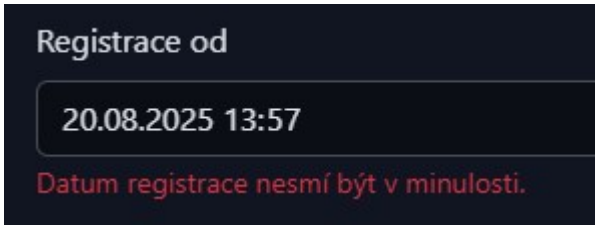
Obrázek 18 - Snímek obrazovky seznamu notifikací v hlavičce

2.6.10 Validace a bezpečnost

Všechna pole formulářů podléhají validaci. Kromě základních atributů jsou implementovány i specifické validace (např. zákaz úkolu s termínem v minulosti).

```
[Display(Name = "Registrace od")]
[PastDateTimeValidator(ErrorMessage = "Datum registrace nesmí být v minulosti.")]
Počet odkazů: 6
public DateTime? RegistraceOd { get; set; }
```

Obrázek 19 - Zdrojový kód zajišťující, že nastavený čas nesmí být v minulosti



Obrázek 20 - Snímek obrazovky chybně zadaného pole ve formuláři a následné validace

2.6.11 Administrace

Administrátorské rozhraní umožňuje správu uživatelů, rolí a číselníků.

2.6.12 Shrnutí implementace

Implementace ukazuje, že navržená architektura je flexibilní a připravená na budoucí rozšíření. Díky servisní vrstvě a modulárnímu přístupu lze jednotlivé části snadno testovat a rozšiřovat.

2.7 Specifikace demonstrační verze

Vyvinutý systém byl navržen především jako demonstrační prototyp, jehož cílem je ukázat možnosti zefektivnění a zpřehlednění procesu správy závěrečných prací. Z tohoto důvodu byla implementace záměrně zjednodušena a některé funkce běžně obsažené v univerzitních informačních systémech nebyly do prototypu zahrnuty.

2.7.1 Registrace a přihlášení pouze přes Identity

V aktuální verzi je autentizace uživatelů realizována prostřednictvím frameworku ASP.NET Core Identity. Tento přístup je vhodný pro demonstrační účely, neboť umožňuje rychlou implementaci základní správy uživatelů. Do budoucna by však bylo možné systém rozšířit o integraci s univerzitním informačním systémem (například IS/STAG) a zajistit jednotné přihlašování (Single Sign-On).

2.7.2 Absence schvalovacího procesu

V demonstrační verzi není implementován proces schvalování témat závěrečných prací. Standardně by tento proces zahrnoval několik kroků – návrh tématu studentem, jeho kontrolu

a schválení vedoucím práce, případně následné potvrzení fakultou. Vynechání tohoto prvku umožnilo soustředit se na jádrovou funkcionalitu systému a zároveň ukázat, jakým způsobem lze proces správy závěrečných prací zjednodušit.

2.7.3 Omezení oproti plně funkčnímu IS STAG

Prototyp byl původně inspirován informačním systémem STAG, nicméně řada funkcí byla vypuštěna. Tím je zdůrazněno, že i zjednodušená podoba systému může nabídnout významné přínosy – především snížení administrativní zátěže a zvýšení uživatelské přívětivosti. Zjednodušením byly vytvořeny předpoklady pro ukázku toho, jak lze stávající praxi inovovat a usnadnit.

Celkově tak demonstrační verze nepředstavuje plnohodnotný informační systém určený k okamžitému nasazení, ale ukázkový model, na jehož základě lze dále stavět a doplňovat další funkcionalitu podle konkrétních požadavků akademického prostředí.

V produkční verzi by navíc bylo možné realizovat přímé přihlášení prostřednictvím univerzitního systému STAG. Tím by bylo zajištěno jednotné přihlášení (Single Sign-On) a odstranila by se nutnost duplicitní registrace uživatelů. Uživatel by se tak po přihlášení doSTAGu mohl automaticky dostat do prostředí systému pro správu závěrečných prací.

Nejvýznamnějším rozdílem oproti existujícím univerzitním systémům je důraz na **integrovanou komunikaci** mezi studentem a vedoucím přímo v prostředí aplikace. Dosavadní praxe, kdy komunikace probíhá mimo systém (nejčastěji prostřednictvím e-mailu), je v prototypu nahrazena komentáři, notifikacemi a real-time přenosem zpráv prostřednictvím technologie SignalR. Tím je celý proces zjednodušen a zpřehledněn.

2.8 Testování a ověření funkčnosti

Testování představuje zásadní fázi vývoje, neboť umožňuje ověřit, zda aplikace skutečně naplňuje stanovené požadavky a vykazuje očekávanou konzistenci v chování. V rámci vypracovaného prototypu byly realizovány jednotkové testy, integrační testy i uživatelské testování.

2.8.1 Jednotkové testy

Jednotkové testy se zaměřily na klíčové aspekty logiky systému:

- Workflow schvalování témat – bylo ověřeno, že změna stavu práce je možná pouze v rámci povolených přechodů.

- Validace dat – proběhla kontrola povinných polí a bylo zajištěno, že nelze zadat úkol s termínem v minulosti.
- Správa úkolů – testovaly se změny stavů úkolů a generování notifikací.
- Porovnání verzí dokumentů – bylo ověřeno, že diff správně identifikuje veškeré změny.

2.8.2 Integrační testy

Integrační testy ověřily správnou spolupráci jednotlivých modulů systému:

- Vytvoření tématu vedoucím a uložení do databáze.
- Nahrání nové verze dokumentu se správným očíslováním.
- Porovnání verzí dokumentů s vizualizací rozdílů v uživatelském rozhraní.
- Přidání komentáře a doručení notifikace autorovi.

2.8.3 Uživatelské testování

Testování použitelnosti bylo realizováno simulací typických uživatelských scénářů:

- Student nahrál dokument a reagoval na komentář.
- Učitel založil téma, přidal řešitele, komentář a zadal úkol.
- Administrátor přidal nového uživatele a přiřadil mu příslušnou roli.

Výsledky prokázaly, že uživatelské rozhraní je intuitivní a jednotlivé funkce jsou dostupné v logické struktuře. Uživatelé doporučili zejména zvýraznění změn stavu a zlepšení odlišení verzí dokumentů.

2.8.4 Výsledky testování

Testování potvrdilo, že prototyp splňuje veškeré funkční i nefunkční požadavky. Jednotkové a integrační testy byly úspěšně dokončeny a uživatelské testování prokázalo vysokou míru spokojenosti.

2.9 Nasazení a budoucí rozvoj

Po úspěšném dokončení testování aplikace následuje její implementace do ostrého provozu. Tím ovšem proces nekončí, již od tohoto bodu je nezbytné uvažovat o možnostech budoucího rozšíření systému, aby zůstal dlouhodobě relevantní a adaptovatelný na proměňující se potřeby univerzity.

2.9.1 Možnosti nasazení

Aplikaci lze implementovat několika způsoby:

- Lokální nasazení na IIS – v tomto scénáři je aplikace hostována na univerzitním serveru s Windows Serverem a IIS. Tento přístup je vhodný zejména pro pilotní provoz nebo menší katedry, kde není vyžadována vysoká škálovatelnost.
- Nasazení do cloudu (Azure App Service) – aplikace běží ve formě webové služby, která umožňuje automatické škálování a zálohování. Mezi hlavní výhody patří snadná správa a vysoká dostupnost služby.
- Kontejnerizace pomocí Dockeru – aplikace je zabalena do Docker image, což zjednodušuje její nasazení v různých prostředích a umožňuje využití orchestrátorů, jako je například Kubernetes.

2.9.2 Bezpečnostní opatření při nasazení

V produkčním prostředí je nutné důsledně dbát na bezpečnost. Mezi klíčová opatření patří:

- Využití HTTPS protokolu a důvěryhodných certifikátů.
- Zavedení silné politiky hesel a dvoufaktorové autentizace.
- Omezení typů a velikostí nahrávaných souborů.
- Pravidelné aktualizace všech frameworků a knihoven.
- Zajištění logování a monitoringu přístupů.

```
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}
app.UseHttpsRedirection();
```

Obrázek 21 - Zdrojový kód zajišťující použití HTTPS v aplikaci

2.9.3 Možnosti budoucího rozšíření

Systém je koncipován modulárně, což usnadňuje další rozvoj. Perspektivní směry rozšíření zahrnují:

- Integraci s fakultním informačním systémem, která umožní automatizované načítání seznamů studentů, předmětů a vypsaných témat.
- Integrace kompletních schvalovacích procesů
- Zavedení jednotného přihlášení (SSO, OAuth) pro zvýšení bezpečnosti a uživatelského komfortu.
- Podpora e-mailových a mobilních notifikací jako doplněk k notifikacím v aplikaci.
- Možnost exportu dat a generování reportů pro potřeby vedení fakulty.

2.9.4 Přínosy rozšíření

Rozšířením systému dojde ke zvýšení uživatelského komfortu a efektivity procesu vedení závěrečných prací. Integrace s existujícími fakultními nástroji povede ke snížení administrativní zátěže a analytické funkce poskytnou relevantní data pro strategické řízení fakulty.

2.9.5 Nasazení aplikace na Windows Server

Tato podkapitola popisuje kompletní postup nasazení výsledné aplikace na webový server IIS ve Windows. Tento způsob je vhodný pro provoz v prostředí školy nebo menší firmy, kde je k dispozici vlastní Windows Server.

2.9.5.1 Požadavky

- **Windows Server 2019/2022** (nebo Windows 10/11 Pro pro test).
- Nainstalovaná role **Web Server (IIS)** včetně **IIS Management Console**.
- **.NET 9 Hosting Bundle (ASP.NET Core Hosting Bundle)**

2.9.5.2 Publikace aplikace

- Na vývojářském stroji nebo přímo na serveru se v kořeni projektu spustí příkaz:
- dotnet publish -c Release -o publish
- V adresáři `publish` vznikne spustitelný soubor **Bakalarska_prace.exe**, všechny knihovny a konfigurační soubor `web.config`.

- Obsah složky `publish` se zkopíruje do cílového umístění na serveru, např. `C:\inetpub\Bakalarska_prace`.

Vytvoření Application Poolu

- Otevřít IIS Manager → Application Pools → Add Application Pool…
- Název: `Bakalarska_pracePool`
- .NET CLR version: *No Managed Code* (ASP.NET Core běží mimo CLR IIS).
- Pipeline mode: Integrated.
- Volitelně v Advanced Settings nastavit *Start Mode = AlwaysRunning* a vypnout *Idle Time-out*, aby se aplikace neuspávala.

Vytvoření webu (Site)

- V IIS Manager → Sites → Add Website…:
- Site name: `Bakalarska_prace`
- Application pool: `Bakalarska_pracePool`
- Physical path: `C:\inetpub\Bakalarska_prace`
- Binding:
    - Type: `http`
    - IP address: konkrétní IP serveru (nebo *All Unassigned*)
    - Port: `80` (nebo jiný)
    - Host name: prázdné, pokud není potřeba DNS.
- Potvrdit OK a web spustit.
- Pro zabezpečení je vhodné přidat také HTTPS binding s platným certifikátem.

NTFS oprávnění

Aplikace musí mít právo zápisu do své složky (např. logy, uploady). Nastavení:

- Pravý klik na `C:\inetpub\Bakalarska_prace` → Properties → Security → Edit → Add.

- Přidat uživatele **`IIS AppPool\Bakalarska_pracePool`**.
- Udělit mu oprávnění Modify.

Testování a diagnostika

- Web je dostupný na `http://<IP_serveru>/`.
- Při chybách se kontroluje **Event Viewer → Application Logs** a **ASP.NET Core Logs**.
- Nejčastější problémy:
- **HTTP 500.31** – chybí ASP.NET Core Hosting Bundle.
- **HTTP 500.30** – chyba při startu aplikace (connection string, chybějící práva).

2.9.5.3 Databázový server – SQL Server a správa

Protože aplikace využívá Entity Framework Core a relační databázi, je nutné, aby uživatel měl připravený databázový server. V této práci je použita platforma Microsoft SQL Server.

Požadavky

- SQL Server 2019/2022 (stačí edice Express nebo Developer).
- SQL Server Management Studio (SSMS) – doporučeno pro správu databáze, loginů a kontrolu dat.

Instalace SQL Serveru

- Spustit instalátor SQL Serveru → New SQL Server stand-alone installation.
- Vybrat komponentu Database Engine Services.
- Nastavit režim ověřování Mixed Mode (Windows i SQL loginy).
- Zadat silné heslo pro účet `sa`.

Instalace SSMS

- Stáhnout a nainstalovat SQL Server Management Studio.
- Tento nástroj se používá pro vytváření databází, správu uživatelů a spouštění SQL dotazů.

Vytvoření databáze

- V SSMS se připojit na instanci serveru (např. `localhost\SQLEXPRESS`).
- V Object Explorer → Databases → New Database…
- Název: `Bakalarska_praceDB`.
- Potvrdit OK.

Vytvoření uživatele a oprávnění

- V SSMS → Security → Logins → New Login…
- Login name: `bakalarska_user`.
- Authentication: SQL Server authentication, zadat heslo.
- V záložce User Mapping vybrat databázi `Bakalarska_praceDB` a role:
- `db_datareader` – čtení
- `db_datawriter` – zápis
- `db_ddladmin` – pokud má aplikace provádět migrace EF Core (změny schématu).

Nastavení connection stringu

V konfiguračním souboru `appsettings.Production.json` se connection string nastaví např. takto:

```
"ConnectionStrings": {
  "DefaultConnection":
"Server=localhost\\SQLEXPRESS;Database=Bakalarska_praceDB;User
Id=bakalarska_user;Password=MojeSilneHeslo;TrustServerCertificate=True;"
}
```

Migrace databáze

Pokud je použit Code-First přístup, je nutné provést migrace:

- příkazem `dotnet ef database update` při nasazení

2.10 Shrnutí praktické části

Praktická část práce popsala návrh, implementaci, testování a možnosti nasazení systému pro správu závěrečných prací. Výsledkem je funkční prototyp, který umožňuje efektivní spolupráci mezi studenty, vedoucími a administrátory.

Byly definovány funkční i nefunkční požadavky, navržena architektura založená na vzoru Model–View–Controller a vytvořen datový model reflektující reálné procesy akademického prostředí. Implementace zahrnovala všechny klíčové funkce: schvalovací workflow, správu verzí dokumentů, porovnání dvou verzí, komentáře, To-Do seznam a notifikační systém.

Testování potvrdilo, že prototyp splňuje zadání a poskytuje spolehlivé prostředí pro správu závěrečných prací. Možnosti nasazení ukázaly flexibilitu systému, ať už jde o lokální instalaci, cloudové prostředí nebo kontejnerizaci pomocí Dockeru. V neposlední řadě byla nastíněna i budoucí rozšíření, která by systém ještě více přiblížila skutečným potřebám univerzity.

Díky modularitě, použitým technologiím a důrazu na bezpečnost je prototyp připraven k dalšímu vývoji a nasazení v akademické praxi.

ZÁVĚR

Navržený a implementovaný systém pro správu závěrečných prací ukazuje, že moderní webové technologie mohou významně přispět k efektivitě a přehlednosti procesů spojených s vedením a hodnocením studentských prací. Výsledný prototyp představuje demonstrační aplikaci, která zahrnuje klíčové funkcionality, jako jsou evidence témat, správa verzí dokumentů, jejich porovnávání, komentáře, úkoly a systém notifikací. Díky tomu poskytuje uživatelům nejen nástroje pro samotnou práci s dokumenty, ale také prostor pro efektivní komunikaci a přehlednou organizaci celého procesu.

Cílem řešení nebylo vytvořit plnohodnotný nástroj připravený k okamžitému nasazení, ale ukázat možnosti zjednodušení a rozvoje oproti současným robustním systémům. Proto byla řada funkcí záměrně vynechána nebo zjednodušena. Například registrace a přihlášení jsou realizovány výhradně prostřednictvím ASP.NET Core Identity, přičemž v budoucnu lze předpokládat integraci jednotného přihlášení (Single Sign-On) nebo napojení na univerzitní informační systémy. Podobně by plná implementace schvalovacího procesu umožnila pokrýt celý životní cyklus závěrečné práce od zadání přes hodnocení až po archivaci.

Význam prototypu spočívá především v tom, že ukazuje možnosti, jak lze usnadnit komunikaci mezi studentem a vedoucím, zefektivnit práci s dokumenty a poskytnout nástroj podporující přehledné sledování postupu prací. Využití architektury ASP.NET Core MVC a databázového systému SQL Server zároveň potvrzuje, že moderní platformy umožňují vytvořit flexibilní a rozšiřitelný základ pro budoucí vývoj, který lze snadno doplnit o další funkcionality, jako je automatická kontrola plagiátorství, pokročilé workflow nebo integrace s cloudovými úložišti.

Přínos práce tedy spočívá nejen v samotném prototypu, ale i v ukázce metodiky, jak k návrhu podobného systému přistupovat – od analýzy požadavků a volby technologií až po implementaci a testování. Výsledné řešení je možné vnímat jako inspiraci pro další rozvoj univerzitních informačních systémů a jako důkaz, že i relativně jednoduchý prototyp může představovat významný krok k digitalizaci a zefektivnění procesů v akademickém prostředí.

Do budoucna lze očekávat, že obdobné systémy budou stále více využívat integraci s externími službami, důraz na uživatelský komfort a pokročilé bezpečnostní mechanismy. Zkušenosti z této práce tak mohou posloužit jako základ pro návrh plně funkčního nástroje, který by mohl být nasazen v praxi a dlouhodobě podporovat studenty, vyučující i administrátory vysokých škol.

POUŽITÁ LITERATURA

Tištěné monografie a knihy

ALBAHARI, Joseph a Ben ALBAHARI, 2022. C# 10 in a Nutshell. Sebastopol: O'Reilly Media. ISBN 978-1098121959.

COHN, Mike, 2010. Succeeding with Agile: Software Development Using Scrum. Boston: Addison-Wesley. ISBN 978-0321579362.

DUCKETT, Jon, 2014. HTML & CSS: Design and Build Websites. Indianapolis: Wiley. ISBN 978-1118008188.

FOWLER, Martin, 2003. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley. ISBN 978-0321127426.

FLANAGAN, David, 2020. JavaScript: The Definitive Guide. 7th ed. Sebastopol: O'Reilly Media. ISBN 978-1491952023.

GAMMA, Erich, Richard HELM, Ralph JOHNSON a John VLISSIDES, 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley. ISBN 978-0201633610.

JELÍNEK, Václav, 2018. Informační systémy. Praha: Grada. ISBN 978-8027106634.

LAUDON, Kenneth C. a Jane P. LAUDON, 2022. Management Information Systems. 16th ed. Harlow: Pearson. ISBN 978-1292403282.

RESCORLA, Eric, 2018. SSL and TLS: Designing and Building Secure Systems. Boston: Addison-Wesley. ISBN 978-0201615982.

SATZINGER, John W., Robert B. JACKSON a Stephen D. BURD, 2015. Systems Analysis and Design in a Changing World. 7th ed. Boston: Cengage. ISBN 978-1305117204.

STAIR, Ralph a George REYNOLDS, 2020. Principles of Information Systems. 13th ed. Boston: Cengage. ISBN 978-0357112438.

TANENBAUM, Andrew S. a Maarten VAN STEEN, 2017. Distributed Systems. 3rd ed. Harlow: Pearson. ISBN 978-0133591620.

Články v časopisech

SIEMENS, George a Phil LONG, 2011. Penetrating the Fog: Analytics in Learning and Education. EDUCAUSE Review. 46(5), s. 30–32. ISSN 1527-6619.

Normy, standardy a specifikace

BRAY, Tim, 2017. The JavaScript Object Notation (JSON) Data Interchange Format. IETF RFC 8259. ISSN 2070-1721.

Elektronické zdroje

MICROSOFT, 2025a. ASP.NET Core Documentation [online]. [cit. 2025-08-29]. Dostupné z: https://learn.microsoft.com/

MICROSOFT, 2025b. Entity Framework Core Documentation [online]. [cit. 2025-08-29]. Dostupné z: https://learn.microsoft.com/

OTTO, Mark a Jacob THORNTON, 2021. Bootstrap Documentation [online]. [cit. 2025-08-29]. Dostupné z: https://getbootstrap.com/

W3C, 2023a. HTML Standard [online]. [cit. 2025-08-29]. Dostupné z: https://www.w3.org/

W3C, 2023b. CSS Standard [online]. [cit. 2025-08-29]. Dostupné z: https://www.w3.org/

SEZNAM PŘÍLOH