Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Agenda - Informační systém

Oddělení správy a vývoje APV důchodových agend

2025

Michal Chadim

Vedoucí práce:
Mgr. Radek Janoštík, Ph.D.

Studijní program: Informační technologie,
kombinovaná forma

Bibliografické údaje

Autor:	Michal Chadim
Název práce:	Agenda - Informační systém (Oddělení správy a vývoje APV důchodových agend)
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2025
Studijní program:	Informační technologie, kombinovaná forma
Vedoucí práce:	Mgr. Radek Janoštík, Ph.D.
Počet stran:	50
Přílohy:	elektronická data v systému katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Michal Chadim
Title:	Agenda - Information System (Department of Administration and Development of Pension Agenda Software)
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2025
Study program:	Information Technologies, combined form
Supervisor:	Mgr. Radek Janoštík, Ph.D.
Page count:	50
Supplements:	electronic data in system of department of computer science
Thesis language:	Czech

Anotace

Tato bakalářská práce se zabývá vývojem informačního systému Agenda, který usnadňuje každodenní pracovní činnosti zaměstnanců Oddělení správy a vývoje APV důchodových agend. Aplikace je vyvinuta jako webová aplikace postavená na architektuře MVC v technologii ASP.NET s využitím SQLite jako databázového řešení. Klientská část využívá AJAX pro asynchronní komunikaci se serverem a dynamickou aktualizaci obsahu. Práce je rozdělena do dvou částí – první se věnuje použitým technologiím a jejich přínosu pro vývoj moderních informačních systémů, zatímco druhá část se zaměřuje na programovou dokumentaci a popis implementace systému.

Synopsis

This bachelor's thesis deals with the development of the Agenda information system, which facilitates the daily work activities of employees in the Department of Administration and Development of APV pension agendas. The application is developed as a web application built on the MVC architecture using ASP.NET technology and SQLite as the database solution. The client side uses AJAX for asynchronous communication with the server and dynamic content updates. The thesis is divided into two parts – the first focuses on the technologies used and their contribution to the development of modern information systems, while the second part is devoted to the program documentation and the description of the system's implementation.

Klíčová slova: ASP.NET; MVC; AJAX; informační systém; SQLite; webová aplikace

Keywords: ASP.NET; MVC; AJAX; information system; SQLite; web application

Tímto bych chtěl poděkovat vedoucímu bakalářské práce panu Mgr. Radkovi Janoštíkovi, Ph.D. za odborné vedení při vytváření této práce a za cenné rady, které mi pomohly při návrhu a vývoji informačního systému Agenda.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

1 Úvod **8**
1.1 Současná situace . . . 8
1.2 Motivace . . . 9
1.3 Cíl práce . . . 10
1.4 Ochrana citlivých údajů a práce s testovacími daty . . . 10
1.5 Varianty nového řešení . . . 10
1.6 Právní rámec pro IS ve státní správě . . . 11
1.7 Principy 3E v informačních systémech . . . 12

2 Návrh architektury a implementace IS **13**
2.1 Návrh architektury . . . 14
2.2 Databázová vrstva . . . 15
2.2.1 Architektura databáze . . . 15
2.2.2 Moderní trendy v databázích . . . 16
2.2.3 Výběr řešení a implementace SQLite . . . 17
2.3 Backend technologie . . . 18
2.3.1 ASP.NET Core . . . 18
2.3.2 ASP.NET Identity . . . 18
2.3.3 Entity Framework Core . . . 20
2.4 Frontend technologie . . . 20
2.4.1 Značkovací a stylovací jazyk HTML a CSS . . . 21
2.4.2 Skriptovací jazyk JavaScript a jeho využití . . . 21
2.4.3 Asynchronní komunikace pomocí AJAX a REST API . . . 21
2.4.4 CSS frameworky (Bootstrap a alternativy) . . . 23
2.4.5 JavaScriptová knihovna jQuery . . . 23
2.4.6 JavaScriptová knihovna FullCalendar . . . 23
2.4.7 WYSIWYG editor CKEditor 5 pro práci s obsahem . . . 24
2.4.8 Knihovna SweetAlert2 pro modální dialogy . . . 27
2.4.9 Responzivita a přizpůsobení zobrazení . . . 27
2.5 Externí skripty pro správu systému . . . 28
2.6 Adresářová struktura projektu . . . 28

3 Moduly IS Agenda **29**
3.1 iÚkoly – Interní úkoly . . . 30
3.1.1 Požadavky . . . 31
3.1.2 Úkoly . . . 31
3.1.3 Dokumentace . . . 31
3.1.4 Práce . . . 32
3.1.5 Komerční řešení a moderní trendy . . . 32
3.2 iNávody – Výuka pro nové pracovníky oddělení . . . 33
3.3 iKalendář – Plánování úloh . . . 33
3.4 iDocházka – Přítomnost zaměstnanců . . . 34
3.5 iUživatel – Administrace . . . 36

3.5.1 Detail uživatele 36
3.5.2 Seznam uživatelů 37
3.5.3 Role uživatelů 37
3.6 Logy 38
3.7 O aplikaci 38
3.8 E-mailové notifikace uživatelům 39

4 Budoucí vývoj IS **40**

5 Zpětná vazba od uživatelů a testování **42**

Závěr **44**

Conclusions **45**

A Obsah elektronických dat **46**

Seznam zkratek **47**

Literatura **48**

Seznam obrázků

1 Vizualizace principů 3E převzata z [8] 13
2 Zjednodušený návrh architektury informačního systému *Agenda* . 15
3 Přehled hlavních entit databáze IS Agenda (zjednodušený model, bez vazeb) 16
4 Hlavní výhody databázového systému SQLite 17
5 Přihlašovací obrazovka systému *Agenda* s využitím knihovny `ASP.NET Identity` 19
6 Diagram průběhu přidání uživatele pomocí AJAX a MVC 22
7 Komunikace knihovny FullCalendar s backendem systému *Agenda* 24
8 ER vazba mezi tabulkami `Chapters` a `Pages` 25
9 Proces nahrávání obrázku a úklidu nepoužitých souborů 26
10 Ukázka dialogového okna vytvořeného pomocí SweetAlert2 27
11 Zjednodušená adresářová struktura projektu *Agenda* 29
12 Hlavní obrazovka systému *Agenda* 30
13 Hlavní obrazovka části Dokumentace v rámci modulu *iÚkoly* . . . 32
14 Tématická sada Mainframe v modulu *iNávody* 33
15 Zobrazení plánovaných událostí v modulu *iKalendář* pomocí knihovny *FullCalendar* 34
16 Zobrazení nepřítomnosti zaměstnanců v dynamickém kalendáři modulu *iDocházka* 35
17 Legenda emotikonů nepřítomnosti zaměstnanců v rámci modulu *iDocházka* 35
18 Administrace uživatelů – přístupná pouze oprávněným osobám . . 36
19 Zobrazení historie přihlášení uživatelů v modulu *Logy* 38
20 Ukázka e-mailové notifikace při založení události v iDocházce . . . 40
21 Ukázka e-mailové notifikace po změně přihlašovacího hesla 40
22 Ukázka administračního rozhraní pro správu kapitol v modulu *iNávody* 42

Seznam tabulek

1 Přehled současných nástrojů používaných v oddělení APV 9
2 Přehled hlavních technologií použitých při vývoji systému 14

1 Úvod

Informační systémy se staly nedílnou součástí moderní veřejné správy, kde hrají klíčovou roli při zpracování dat, správě agend a zefektivňování administrativních procesů. Správně navržený Informační systém (IS) může výrazně přispět ke zvýšení transparentnosti, dostupnosti služeb i efektivitě práce úředníků [1].

Ministerstvo práce a sociálních věcí (MPSV) prostřednictvím svých vnitřních oddělení vyvíjí a spravuje různé softwarové nástroje, které podporují výkon agend v oblasti důchodového pojištění. V rámci oddělení správy a vývoje aplikačního programového vybavení důchodových agend vzniká řada nástrojů, jež jsou primárně určeny pro interní použití pracovníků České správy sociálního zabezpečení (ČSSZ) [2].

V současnosti však dochází k určité roztříštěnosti těchto řešení – využívány jsou samostatné Windows Presentation Foundation (WPF) aplikace, tabulky v prostředí Microsoft Excel či dokumenty vytvořené v Microsoft Word. Tato heterogenní struktura nejen komplikuje údržbu a rozvoj systémů, ale přináší i rizika duplicit, nesprávné práce s daty a zvyšuje nároky na zaškolování uživatelů [3].

Cílem této bakalářské práce je analyzovat současný stav, navrhnout jednotný IS pro podporu klíčových činností oddělení a následně realizovat funkční webovou aplikaci postavenou na moderních technologiích. Součástí práce je i posouzení legislativního rámce, který se vztahuje na IS ve veřejné správě, a uplatnění principů tzv. 3E (efektivnost, ekonomičnost, účelnost), které jsou dnes nedílnou součástí odpovědného hospodaření ve veřejném sektoru. Jejich praktická aplikace je podrobně vysvětlena v metodické příručce Ministerstva financí České republiky z roku 2022, která konkretizuje povinnost jejich dodržování při řízení organizací veřejné správy [4].

1.1 Současná situace

Oddělení správy a vývoje aplikačního programového vybavení důchodových agend při MPSV se dlouhodobě podílí na vývoji a údržbě nástrojů, které zajišťují interní agendy spojené s výpočtem, revizí a evidencí důchodových dávek. Pracovní prostředí je však do značné míry roztříštěné – jednotlivé procesy jsou zajišťovány prostřednictvím několika nesourodých systémů a nástrojů.

V současnosti se pro různé účely využívají samostatné WPF aplikace, dokumenty ve formátu Microsoft Word (zejména pro dokumentaci k programům) a tabulky v prostředí Microsoft Excel (například pro evidenci nepřítomnosti), jejich přehled shrnuje tabulka 1. Tyto nástroje byly často vytvářeny ad hoc pro konkrétní potřebu nebo specifickou agendu. V důsledku toho dochází k neefektivnímu předávání informací mezi uživateli, k duplicitnímu zadávání dat a k časté potřebě manuálních úprav.

Dalším výrazným problémem je nejednotná technická úroveň používaných nástrojů a jejich nízká interoperabilita. Například tabulky v prostředí Microsoft Excel slouží pro plánování nepřítomnosti zaměstnanců, přičemž přístup k nim má

obvykle pouze jedna osoba. Tento centralizovaný přístup není efektivní a znesnadňuje koordinaci v týmu, zejména při zástupu či změnách plánů. Interní úkoly jsou spravovány pomocí samostatné WPF aplikace, která však s ohledem na svůj věk a komplexitu vyžaduje značné úsilí při jakýchkoli úpravách a její rozšiřování je již velmi obtížné. Dokumentace k jednotlivým programům je uložena v prostředí Microsoft Word na sdíleném cloudovém úložišti, které je značně nepřehledné a nepodporuje snadnou správu verzí ani rychlou orientaci v obsahu. Neexistence jednotného systému pro správu těchto informací dále komplikuje nejen běžnou práci, ale i analytické činnosti, reporting a předávání agend při personálních změnách.

Na základě této analýzy vznikla potřeba vytvořit jednotný IS, který by sjednotil nejčastěji používané moduly (např. evidenci úkolů, interní analýzy, kalendář, docházku či uživatelskou správu) do jediné webové aplikace dostupné z interní sítě. Ta bude poskytovat jednotné prostředí pro správu dat, zvýší přehlednost a zároveň umožní snadnější údržbu a další rozvoj systému.

Tabulka 1: Přehled současných nástrojů používaných v oddělení APV

Nástroj	**Použití**	**Identifikované problémy**
WPF aplikace	Správa interních úkolů	Obtížná rozšiřitelnost, zastaralá technologie, složitá údržba
Microsoft Word	Dokumentace programů	Nepřehledná struktura, chybí správa verzí, obtížná orientace
Microsoft Excel	Evidence docházky / plánování nepřítomnosti	Přístup jen pro jednu osobu, riziko chyb, nemožnost sdílené editace

1.2 Motivace

Odstranit popsanou roztříštěnost nástrojů není samoúčelné. Projekt sjednoceného informačního systému sleduje tři klíčové cíle:

1. **Strategický soulad.** Digitalizační strategie MPSV i principy 3E vyžadují efektivní, bezpečné a transparentní zpracování agend. Bez centralizovaného IS toho nelze dosáhnout.

2. **Provozní efektivita.** Jednotný systém odstraní duplicitní zadávání dat, zkrátí interní schvalovací procesy a zjednoduší údržbu – předpoklad úspory pracovního času v řádu desítek hodin měsíčně.

3. **Kvalita služeb a uživatelský komfort.** Přehledné webové rozhraní, jednotné přihlašování a automatizované reporty zvýší spokojenost zaměstnanců, usnadní tzv. onboarding, tedy proces začlenění nových kolegů do pracovního prostředí, a minimalizují riziko chyb.

1.3 Cíl práce

Hlavním cílem této bakalářské práce je sjednotit vybrané interní procesy oddělení správy a vývoje aplikačního programového vybavení důchodových agend (MPSV) prostřednictvím plnohodnotné webové aplikace *Agenda*.

Pro naplnění tohoto záměru byly stanoveny následující dílčí cíle:

1. **Procesní analýza a požadavky** Provést analýzu stávajících nástrojů a pracovních postupů, a na jejím základě formulovat funkční i nefunkční požadavky na sjednocený informační systém.

2. **Návrh architektury** Navrhnout vhodnou datovou strukturu a logickou i technologickou architekturu s ohledem na provoz v interní síti MPSV.

3. **Implementace aplikace** Vytvořit webovou aplikaci *Agenda* s využitím architektury Active Server Pages .NET (ASP.NET) MVC, databázového systému Structured Query Language – Lightweight Embedded (SQLite) a asynchronního uživatelského rozhraní. Pro tento účel je využita technologie Asynchronous JavaScript and XML (AJAX), která umožňuje dynamickou interakci mezi klientem a serverem bez nutnosti obnovy celé stránky.

4. **Ověření funkčnosti a přínosů** Nasadit webovou aplikaci v pilotním režimu, demonstrovat sjednocení procesů a kvantifikovat časové i organizační úspory.

1.4 Ochrana citlivých údajů a práce s testovacími daty

Vzhledem k tomu, že tato práce se zaměřuje na informační systém využívaný ve státní správě, bylo nutné při její tvorbě zohlednit požadavky na ochranu citlivých a osobních údajů. Veškeré ukázky uživatelského rozhraní, testovacích scénářů i příruček byly připraveny výhradně s využitím fiktivních nebo anonymizovaných dat.

Záměrně nebyly uváděny žádné skutečné osobní údaje, názvy pracovišť, vnitřní identifikátory ani další informace, které by mohly podléhat ochraně dle vnitřních předpisů či legislativy, zejména nařízení GDPR [5] a zákona o kybernetické bezpečnosti [6]. Tato opatření byla provedena v souladu s platnými právními předpisy.

Přesto byla zachována věcná správnost a funkčnost prezentovaných příkladů tak, aby bylo možné posoudit návrh a implementaci systému bez porušení bezpečnostních či etických zásad.

1.5 Varianty nového řešení

Před zahájením vývoje systému Agenda bylo zvažováno několik možných přístupů, jak řešit identifikované problémy s roztříštěností nástrojů a neefektivním řízením interních agend.

První možností bylo ponechání stávajícího stavu a dílčí vylepšení jednotlivých nástrojů. Tato varianta by však neřešila zásadní problémy s jejich propojením, správou verzí a duplicitním zadáváním dat. Navíc některé nástroje, zejména starší WPF aplikace, jsou technologicky zastaralé a jejich rozvoj je obtížný kvůli nedostatečné dokumentaci a složité údržbě.

Další variantou bylo využití některého z existujících komerčních nebo open-source nástrojů pro správu úkolů a dokumentace. Tyto nástroje však nejsou navrženy s ohledem na specifika agend důchodového pojištění a interní procesy oddělení, a jejich přizpůsobení by vyžadovalo rozsáhlou konfiguraci nebo vývoj nadstaveb, což by vedlo k dodatečným nákladům.

Jako nejvhodnější se ukázalo řešení formou vývoje vlastního webového informačního systému dostupného z interní sítě. Tato varianta umožňuje přesně reflektovat specifické potřeby oddělení a zároveň zajistit moderní technické zázemí, dobrou rozšiřitelnost do budoucna a snadnější správu dat v jednotném prostředí.

1.6 Právní rámec pro IS ve státní správě

Vývoj a provoz IS ve veřejné správě podléhá celé řadě legislativních norem a metodických pokynů. Základní rámec stanovuje zákon č. 365/2000 Sb., o informačních systémech veřejné správy, který definuje požadavky na správu, provoz i bezpečnost těchto systémů [7]. Tento zákon klade důraz mimo jiné na integritu, dostupnost a důvěryhodnost spravovaných dat.

Z hlediska ochrany osobních údajů je nutné řídit se nařízením Evropského parlamentu a Rady (EU) 2016/679, známým jako General Data Protection Regulation (GDPR), které ukládá správcům a zpracovatelům údajů povinnosti při práci s osobními daty [5]. I v rámci interních systémů, které nejsou veřejně přístupné, je nutné zajistit adekvátní úroveň zabezpečení a transparentnost zpracování.

Relevantním předpisem z pohledu kybernetické bezpečnosti je také zákon, č. 181/2014 Sb. o kybernetické bezpečnosti [6]. Uplatní se zejména tehdy, kdyby systém *Agenda* spravoval či zpracovával data kritická pro chod veřejné instituce.

Doporučení a směr dalšího rozvoje digitálních služeb ve státní správě stanovuje strategický dokument vlády ČR „Cesta k evropské digitální dekádě“ [3].Ten mimo jiné klade důraz na využití moderních technologií, interoperabilitu systémů, zajištění kybernetické bezpečnosti a rovný přístup ke službám občanům.

Vývoj systému *Agenda* proto reflektuje nejen technické požadavky dané interními potřebami, ale i výše uvedené právní a strategické rámce, aby výsledné řešení bylo v souladu s aktuálními standardy a očekáváními kladenými na digitální služby ve veřejné správě.

1.7 Principy 3E v informačních systémech

Ve veřejné správě je nezbytné při plánování, realizaci i provozu projektů důsledně zohledňovat principy tzv. 3E – **efektivnosti (effectiveness)**, **ekonomičnosti (economy)** a **účelnosti (efficiency)**. Tyto zásady jsou zakotveny v metodických dokumentech MF ČR, které upozorňují na povinnost jejich aplikace zejména při rozhodování o alokaci veřejných prostředků [4].

Navrhovaný informační systém **Agenda** je koncipován tak, aby systematicky naplňoval všechny tři uvedené principy:

- **Efektivnost** je zajištěna tím, že systém přímo reaguje na konkrétní potřeby oddělení a řeší reálné problémy spojené s roztříštěnou agendou. Sjednocení používaných nástrojů do jednoho celku přispívá k lepší organizaci práce, snižuje chybovost a zvyšuje dostupnost informací.

- **Ekonomičnost** spočívá ve využití otevřených technologií a interních vývojových kapacit, čímž dochází k minimalizaci nákladů na licence a externí dodavatele. Díky jednoduché architektuře a použití databázového systému SQLite lze navíc systém provozovat bez nákladné infrastruktury.

- **Účelnost** je naplněna tím, že systém je navržen s ohledem na specifika daného organizačního prostředí. Obsahuje funkční moduly pro interní úkoly, docházku, dokumentaci, analýzy či správu uživatelů, a reflektuje konkrétní procesy oddělení. Nejedná se tedy o univerzální řešení, ale o nástroj přizpůsobený skutečným potřebám.

Vzájemné propojení těchto tří principů vytváří předpoklady pro dosažení tzv. **optima** – stavu, kdy informační systém nejen splňuje jednotlivé požadavky z hlediska efektivnosti, ekonomičnosti a účelnosti, ale zároveň poskytuje maximální hodnotu při vynaložení minimálních prostředků. Tento synergický efekt je klíčovým cílem při návrhu moderních informačních systémů ve veřejné správě.

Vizualizaci principů 3E a jejich průniku, který vede k optimu, ilustruje obrázek 1.

Obrázek 1: Vizualizace principů 3E převzata z [8]

Systém **Agenda** tak lze považovat za vhodný příklad aplikace principů 3E v praxi vývoje informačních systémů ve veřejné správě.

2 Návrh architektury a implementace IS

Tato kapitola se zaměřuje na technologickou stránku vývoje informačního systému *Agenda*. Cílem bylo vytvořit přehledný, bezpečný a snadno rozšiřitelný systém, který bude provozovatelný v interním prostředí bez nároků na složitou infrastrukturu.

Systém byl navržen jako vícevrstvá webová aplikace, která je rozdělena na tři základní části: databázovou vrstvu, serverovou (backendovou) logiku a klientskou (frontendovou) prezentaci. Každá z těchto vrstev je popsána samostatně v následujících částech.

Při výběru technologií byl kladen důraz na moderní přístupy, otevřenost řešení, minimální provozní náklady a možnost snadného nasazení v prostředí veřejné správy. Výsledná architektura kombinuje technologii ASP.NET pro backend, SQLite jako jednoduché databázové řešení a standardní webové technologie (HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, AJAX) pro frontendovou část systému.

Výsledný návrh systému *Agenda* staví na osvědčených a moderních technologiích, které zajišťují jeho funkčnost, rozšiřitelnost a snadnou údržbu. Přehled použitých technologií je uveden v tabulce 2.

Tabulka 2: Přehled hlavních technologií použitých při vývoji systému

Technologie	**Použití**
ASP.NET Core	Backend, zpracování požadavků
Entity Framework Core	Práce s databází přes ORM
SQLite	Databázová vrstva, ukládání dat
HTML5 / CSS3	Struktura a styl webových stránek
JavaScript	Interaktivita, validace formulářů
AJAX	Asynchronní komunikace se serverem
Bootstrap	Stylování a responzivní rozvržení
CKEditor 5	Rich-text editor
FullCalendar	Zobrazení kalendáře a událostí

Kombinace těchto technologií umožňuje vytvořit multiplatformní, bezpečný a zároveň uživatelsky přívětivý systém, vhodný pro prostředí veřejné správy i menších organizací.

2.1 Návrh architektury

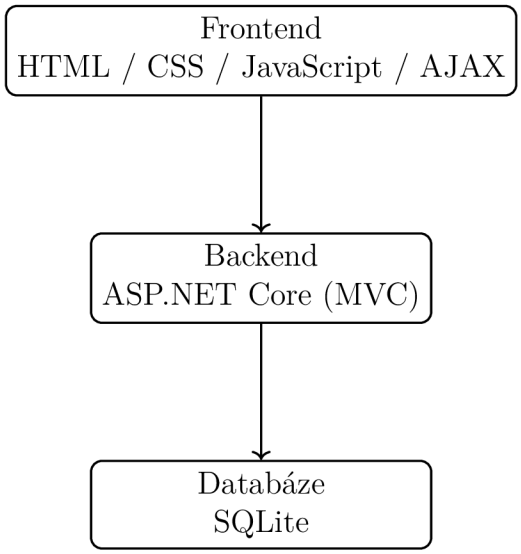
Informační systém *Agenda* byl navržen jako webová aplikace s oddělenými vrstvami podle principu vícevrstvé architektury. Tento přístup zajišťuje lepší čitelnost kódu, snadnější údržbu, vyšší bezpečnost a možnost budoucího rozšiřování.

Architektura je rozdělena do tří hlavních vrstev:

- **Databázová vrstva** – slouží pro trvalé uložení dat. Využívá databázový systém SQLite, který umožňuje jednoduché nasazení bez potřeby databázového serveru.
- **Serverová (backendová) vrstva** – zajišťuje logiku systému, zpracování požadavků od uživatele, práci s daty a komunikaci s databází. Využívá technologii ASP.NET a architekturu Model–View–Controller (MVC).
- **Klientská (frontendová) vrstva** – poskytuje uživatelské rozhraní přístupné z webového prohlížeče. Je tvořena pomocí technologií HTML, CSS a JavaScriptu, přičemž pro dynamické prvky je využit AJAX.

Tato vícevrstvá architektura zajišťuje oddělení jednotlivých zodpovědností a umožňuje snadnější ladění, testování a budoucí modernizaci jednotlivých částí systému. Díky použitým technologiím lze aplikaci provozovat zcela nezávisle na platformě a databázovém serveru, což je výhodné zejména v prostředí veřejné správy s omezenými možnostmi pro komplexní infrastrukturu.

Architektura systému je znázorněna na obrázku 2.



Obrázek 2: Zjednodušený návrh architektury informačního systému *Agenda*

2.2 Databázová vrstva

Tato podkapitola se zaměřuje na návrh, strukturu a technologické pozadí databázové vrstvy informačního systému Agenda. Popisuje architekturu databáze, moderní trendy v oblasti databázových technologií a důvody, proč bylo pro realizaci zvoleno řešení pomocí databáze SQLite.

2.2.1 Architektura databáze

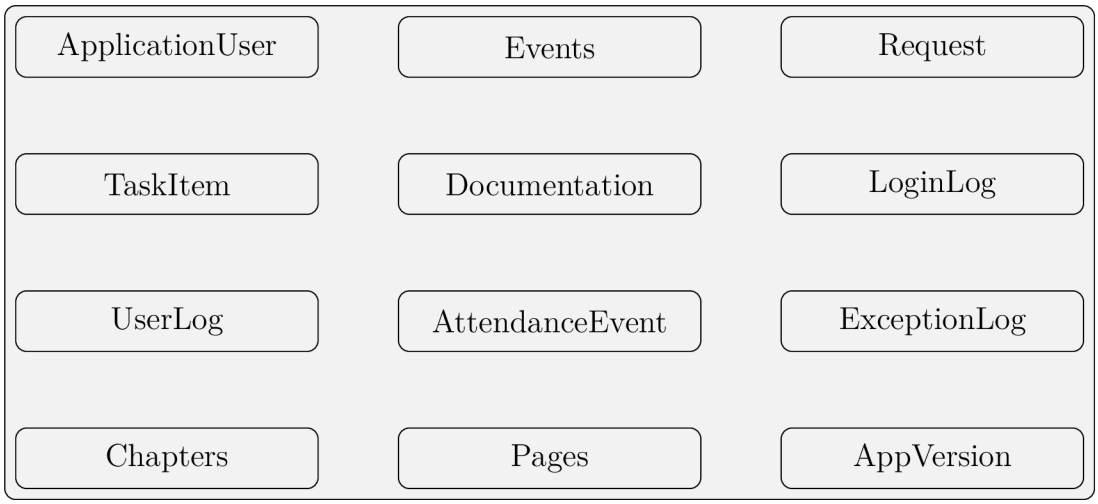
Databázová vrstva tvoří základní stavební kámen každého moderního informačního systému, jelikož zajišťuje trvalé uchování a integritu dat. V případě systému *Agenda* je databáze navržena s důrazem na jednoduchost, přehlednost a snadnou rozšiřitelnost. Její struktura odpovídá specifickým potřebám systému, zejména v oblasti správy uživatelů, úkolů, dokumentací a jejich vzájemných vazeb.

Databáze je relačního typu, což znamená, že data jsou ukládána do tabulek propojených pomocí cizích klíčů. Tento přístup je vhodný pro systémy, kde je klíčová konzistence dat a jednoznačné vztahy mezi entitami.

Mezi hlavní entity patří například tabulky `Chapters` a `Pages`, které slouží pro správu dynamického obsahu systému. Kalendářové události jsou evidovány v tabulkách `Events` a `EventTypes`. Funkčnost systému je dále podpořena tabulkami jako `Requests` (požadavky), `Tasks` (úkoly) a `Documentations` (dokumentace). Uživatelé systému jsou spravováni pomocí ASP.NET Identity, přičemž k tomu slouží tabulky jako `AspNetUsers`, `AspNetRoles` a související.

Pro účely sledování provozu a ladění systému jsou využívány tabulky `LoginLogs`, `UserLogs` a `ExceptionLogs`, které zaznamenávají přihlášení, uživatelské akce a případné výjimky v aplikaci. Nechybí ani podpora verzování systému pomocí tabulek `AppVersions` a `AppVersionChanges`, které evidují jednotlivé aktualizace a jejich obsah.

Všechny tabulky jsou navrženy tak, aby byly ve standardní normalizované podobě, což zajišťuje odstranění redundance a podporuje integritu dat při práci s databází [9].



Obrázek 3: Přehled hlavních entit databáze IS Agenda (zjednodušený model, bez vazeb)

2.2.2 Moderní trendy v databázích

Databázové technologie se neustále vyvíjejí v reakci na rostoucí nároky na výkon, škálovatelnost a flexibilitu. Moderní informační systémy často využívají nejen klasické relační databáze, ale i nové přístupy, které lépe vyhovují specifickým potřebám aplikací.

Mezi klíčové trendy posledních let patří:

- **NoSQL databáze** – určené pro práci s nestrukturovanými nebo polostrukturovanými daty, často využívané v prostředí s vysokou zátěží a potřebou horizontálního škálování (např. MongoDB, Cassandra) [10].
- **In-memory databáze** – ukládají data přímo v operační paměti, čímž výrazně zvyšují rychlost zpracování dat (např. Redis, SAP HANA) [11].
- **Distribuované databáze** – umožňují ukládat data napříč více servery nebo datacentry, čímž zajišťují vysokou dostupnost a odolnost proti výpadkům (např. Google Spanner, CockroachDB) [12].
- **Cloudová řešení** – databázové služby poskytované formou Database-as-a-Service (DBaaS), jako jsou Amazon RDS, Azure SQL Database nebo Firebase, které snižují nároky na správu a údržbu databází [13].

Ačkoliv systém *Agenda* využívá tradiční relační přístup, je jeho návrh kompatibilní s případným přechodem na některé z modernějších řešení, pokud to bude vyžadovat budoucí rozšíření nebo vyšší nároky na škálovatelnost.

2.2.3 Výběr řešení a implementace SQLite

Při výběru databázového systému pro informační systém *Agenda* byly klíčové následující požadavky:

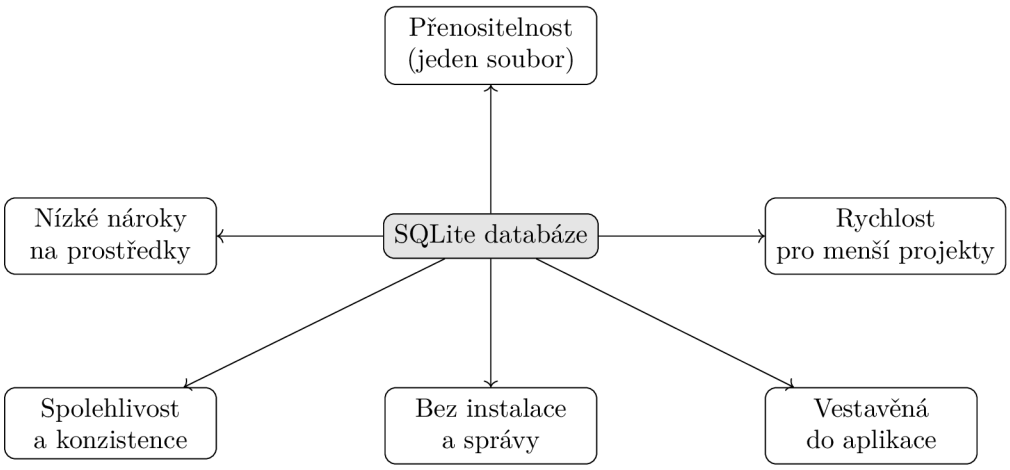
- snadná instalace a správa bez nutnosti centrálního databázového serveru,
- nízké systémové nároky a možnost nasazení na různých platformách,
- podpora relačního modelu a jazyk Structured Query Language (SQL),
- kompatibilita s vývojovým prostředím a knihovnami pro *ASP.NET*.

Na základě těchto kritérií bylo zvoleno řešení SQLite, což je lehká, vestavěná relační databáze, která nevyžaduje samostatný server. SQLite ukládá veškerá data do jednoho souboru, což výrazně zjednodušuje správu a zálohování systému. Jedná se o open-source projekt široce využívaný i v komerčním prostředí – například v mobilních aplikacích, prohlížečích nebo vestavěných systémech [14].

Další výhodou SQLite je snadná integrace s Entity Framework Core (EF Core), který výrazně urychluje vývoj díky mapování objektů na databázové tabulky. Využití principu objektově-relačního mapování navíc usnadňuje přechod na jiný databázový systém (například Oracle), aniž by bylo nutné výrazně měnit aplikační kód.

Použití SQLite tak přináší ideální kompromis mezi jednoduchostí, funkčností a výkonností, zejména pro menší aplikace nebo pilotní nasazení v rámci veřejné správy, kde není k dispozici komplexní infrastruktura.

Hlavní výhody databázového systému SQLite shrnuje obrázek 4, který názorně ukazuje jeho přednosti jako přenositelnost, nízké systémové nároky nebo jednoduché nasazení bez konfigurace.



Obrázek 4: Hlavní výhody databázového systému SQLite

2.3 Backend technologie

Backendová vrstva systému *Agenda* je postavena na moderní, otevřené platformě ASP.NET Core, která zajišťuje vysoký výkon, bezpečnost a přenositelnost aplikace napříč operačními systémy. Pro správu uživatelů a autentizaci je využita knihovna ASP.NET Identity (Identity), která poskytuje rozšířitelné schéma pro uložení identitních údajů. Přístup k datům je realizován pomocí objektově-relačního mapování Entity Framework Core, jež usnadňuje práci s relační databází SQLite pomocí známých objektových konceptů.

2.3.1 ASP.NET Core

ASP.NET Core je moderní, multiplatformní open-source webový framework vyvíjený společností Microsoft, určený pro vývoj webových aplikací, Representational State Transfer (REST) Application Programming Interface (API) a cloudových služeb. Tento framework vychází z původního systému ASP.NET, avšak je přepracován do modulární a výkonné architektury vhodné i pro běh mimo prostředí Windows [15].

Mezi hlavní výhody ASP.NET Core patří podpora běhu na různých operačních systémech (Windows, Linux, macOS), integrovaná podpora *dependency injection*, výkonný *middleware pipeline* a výborná integrace s dalšími technologiemi, jako jsou EF Core a Identity. Aplikace postavené na tomto frameworku mohou být snadno nasazeny v cloudových prostředích jako Microsoft Azure nebo jako kontejnerizované služby pomocí Dockeru [16].

Součástí ASP.NET Core je i vestavěná podpora pro vývoj REST API pomocí rozhraní MVC nebo tzv. minimal APIs, které zjednodušují vývoj menších aplikací. V rámci této práce je využita architektura MVC, která přispívá k lepší separaci logiky, snadnější testovatelnosti a udržitelnosti systému.

Bezpečnostní mechanismy, mezi které patří ochrana proti Cross-Site Request Forgery (CSRF), autorizace pomocí rolí a možnost snadné integrace autentizačních schémat (například OAuth 2.0 nebo OpenID Connect), činí ASP.NET Core vhodným řešením pro vývoj podnikových i veřejně přístupných aplikací.

2.3.2 ASP.NET Identity

Správa uživatelů, přihlášení a autorizace jsou klíčové komponenty každého moderního informačního systému. Pro tyto účely je v systému *Agenda* použita knihovna Identity, která poskytuje rozšiřitelnou infrastrukturu pro správu uživatelů, rolí, autentizaci a autorizaci.

Knihovna je součástí frameworku ASP.NET a je navržena tak, aby podporovala přístup přes *cookie-based* autentizaci, OAuth 2.0, OpenID Connect i dvoufaktorové ověření. Umožňuje vývojářům využít předdefinované struktury (například `IdentityUser`) nebo definovat vlastní modely s rozšířenými atributy podle specifických požadavků systému [17].

V systému *Agenda* je `Identity` integrována s SQLite databází prostřednictvím Entity Framework Core, přičemž schéma je přizpůsobeno tak, aby obsahovalo dodatečné informace jako telefon, pracoviště nebo administrátorský status. Dále je systém rozšířen o správu uživatelů přes webové rozhraní, včetně možnosti měnit role a přístupová práva.

Součástí Identity systému je také základní autentizační mechanismus, který zajišťuje přihlášení a odhlášení uživatelů pomocí jednoduchého formuláře s validací vstupů. Po úspěšném přihlášení je uživatel přesměrován na úvodní stránku systému, kde je podle své role oprávněn k přístupu k jednotlivým modulům. V pravé části hlavičky aplikace je viditelné tlačítko pro odhlášení, které uživateli umožní bezpečně ukončit relaci.

Ukázka přihlašovací obrazovky systému *Agenda* je zobrazena na obr. 5.

Obrázek 5: Přihlašovací obrazovka systému *Agenda* s využitím knihovny `ASP.NET Identity`

Mezi hlavní výhody `Identity` patří [17]:

- hotová implementace bezpečnostních standardů (včetně hashování hesel pomocí algoritmu PBKDF2),
- možnost škálování (například přechod na externí poskytovatele jako Azure Active Directory),
- snadná integrace s autorizací pomocí atributů jako `[Authorize]` v rámci MVC a Web API,
- rychlé nasazení základních procesů jako je přihlášení, odhlášení a obnova hesla.

2.3.3 Entity Framework Core

EF Core je (Object-Relational Mapping (ORM)) vyvíjený společností Microsoft, který výrazně zjednodušuje práci s daty v aplikacích postavených na frameworku ASP.NET [16]. Místo ručního psaní SQL dotazů umožňuje **pracovat s daty prostřednictvím běžných C# tříd** – tzv. entit – a využívat nad nimi jazyk Language Integrated Query (LINQ). Tím se zvyšuje typová bezpečnost kódu a snižuje se riziko chyb způsobených nesouladem mezi databází a programem [17].

- **Code First & Migrations** – doménový model se definuje v C#, na jehož základě EF Core generuje databázové schéma. Případné změny modelu se propagují pomocí *migrací*, které udržují verze schématu pod verzovacím systémem.
- **Podpora více databází** – kromě SQLite, použitého v této práci, lze jediným konfiguračním krokem přejít na PostgreSQL, MySQL či Oracle, což usnadňuje budoucí škálování [18].
- **Lazy a eager loading** – vývojář volí, kdy mají být související entity načteny, čímž optimalizuje výkon a šetří přenesená data.
- **Bezpečnost** – parametry jsou při generování SQL dotazů automaticky escapovány, což minimalizuje riziko SQL injection útoků.

V systému *Agenda* EF Core:

1. mapuje doménové třídy (např. `Task`, `Event`, `User`) na tabulky v databázi SQLite;
2. spravuje schéma databáze pomocí automatických migrací, které se spouštějí při každém nasazení;
3. poskytuje repozitářovou vrstvu, již spotřebovávají servisní třídy a kontrolery MVC.

Díky tomu je datová vrstva výrazně jednodušší, lépe testovatelná (podporuje *in-memory* poskytovatele) a připravená na budoucí rozšíření systému o další moduly či přechod na výkonnější relační databázi [9].

2.4 Frontend technologie

Frontendová vrstva systému *Agenda* tvoří uživatelské rozhraní, které je přístupné prostřednictvím webového prohlížeče. Jejím hlavním cílem je zajistit přehlednou, intuitivní a responzivní interakci mezi uživatelem a systémem. K tomu využívá kombinaci technologií HTML, CSS a JavaScript, které dnes tvoří základ webového vývoje [19].

Struktura stránky je tvořena pomocí jazyka **HTML5**, vizuální styl zajišťuje **CSS3** a interaktivita je realizována pomocí **JavaScriptu** [20]. Pro zjednodušení

návrhu rozhraní a urychlení vývoje je využit CSS framework **Bootstrap**, který poskytuje předdefinované komponenty a responzivní mřížku pro různé velikosti obrazovek [21]. Interaktivní funkce, jako validace formulářů nebo notifikace, jsou řešeny pomocí JavaScriptu a AJAXu [20].

Frontend byl navržen s důrazem na přístupnost, srozumitelnost a rychlou odezvu uživatelského rozhraní, což jsou klíčové požadavky zejména v prostředí veřejné správy, kde aplikace využívá široké spektrum uživatelů s různou technickou úrovní.

2.4.1 Značkovací a stylovací jazyk HTML a CSS

Základním stavebním kamenem frontendové vrstvy systému *Agenda* jsou značkovací a stylovací jazyky **HTML5** a **CSS3**. Jazyk HTML slouží pro strukturování obsahu webových stránek – definuje prvky jako nadpisy, odstavce, formuláře nebo tabulky. Moderní verze HTML5 rozšiřuje možnosti o nové semantické značky (např. `<article>`, `<section>`, `<nav>`), které přispívají ke zpřehlednění kódu a lepší dostupnosti webu [22].

Pro vizuální prezentaci webových prvků je použit jazyk CSS. Pomocí kaskádových stylů lze oddělit vzhled od struktury a zajistit tak snadnější údržbu i konzistenci designu napříč celým systémem. CSS3 přináší nové možnosti, jako jsou přechody, animace, flexibilní rozvržení (flexbox) nebo media queries, které umožňují vytvářet responzivní design [23].

2.4.2 Skriptovací jazyk JavaScript a jeho využití

Pro dynamické ovládání uživatelského rozhraní a realizaci interaktivních prvků je v systému *Agenda* využit skriptovací jazyk **JavaScript**. Tento jazyk umožňuje reagovat na uživatelské akce (např. kliknutí na tlačítko), měnit obsah stránky bez nutnosti jejího znovunačtení a komunikovat asynchronně se serverem [20].

JavaScript je dnes považován za základní technologii moderního webového vývoje spolu s HTML a CSS. Mezi jeho výhody patří podpora ve všech běžných prohlížečích, rozsáhlý ekosystém knihoven a frameworků (např. jQuery, React) a snadná integrace s dalšími webovými technologiemi [24].

2.4.3 Asynchronní komunikace pomocí AJAX a REST API

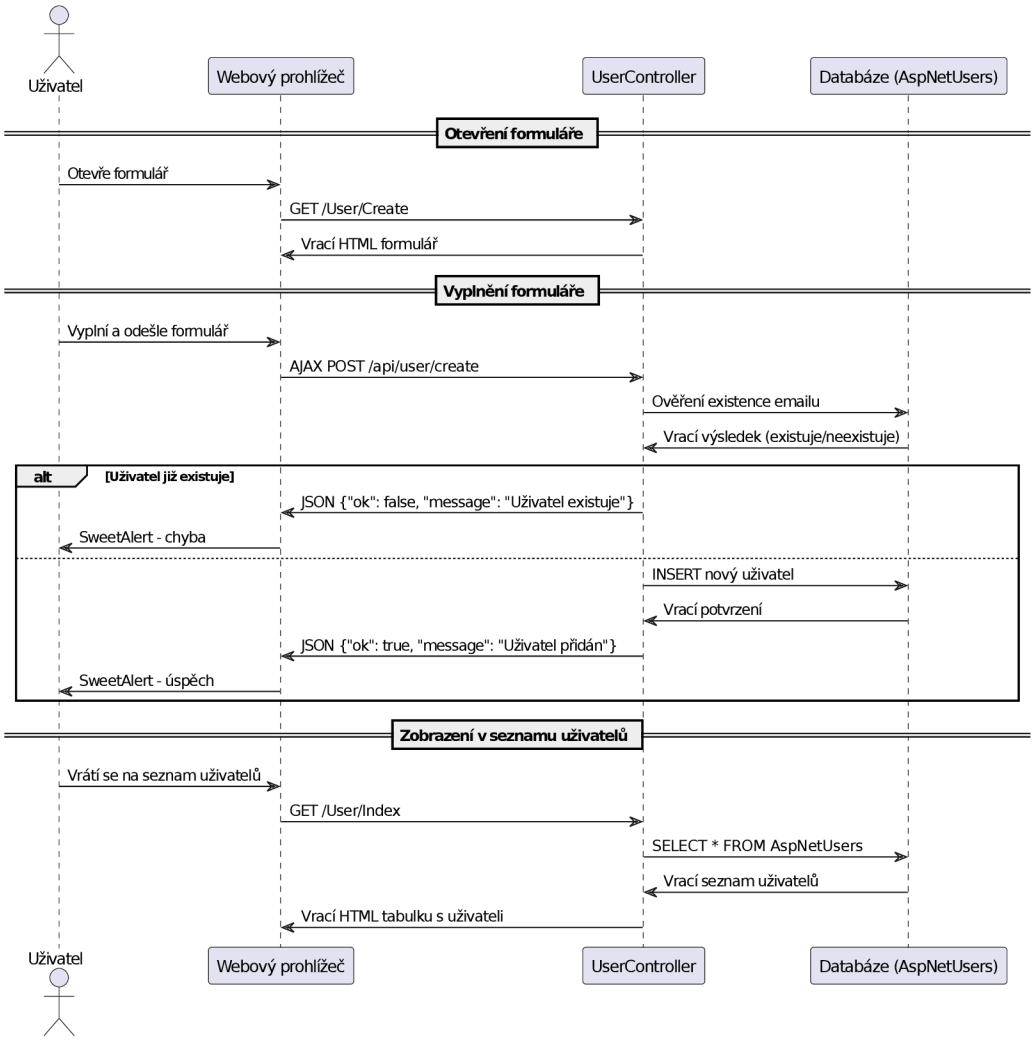
AJAX je technika, která umožňuje webovým aplikacím komunikovat se serverem na pozadí bez nutnosti znovunačítání celé stránky. V systému *Agenda* je AJAX využit zejména pro operace jako je přihlašování, odesílání formulářů, mazání položek nebo načítání dynamického obsahu, čímž se zvyšuje plynulost a interaktivita aplikace [25].

AJAX se v kombinaci s JavaScriptem a REST API stává standardním nástrojem pro realizaci moderních webových aplikací. API rozhraní systému Agenda poskytuje jednotlivé funkce (např. Create, Read, Update, Delete (CRUD) operace nad databázovými záznamy) a AJAX zajišťuje jejich vyvolání z klientské

části. Tento přístup výrazně snižuje latenci aplikace a zároveň umožňuje lepší oddělení klientské a serverové logiky [25].

Přínosem AJAXu je také možnost implementace notifikací, zobrazení validací či změn v reálném čase, což je v administrativních systémech uživatelsky přívětivé a efektivní. Veřejná správa, kde často dochází k práci s formuláři a větším objemem dat, z tohoto asynchronního přístupu výrazně těží.

Na obrázku 6 je znázorněna základní komunikace knihovny FullCalendar s backendem aplikace. JavaScriptová komponenta na klientské straně odesílá pomocí AJAXového požadavku dotaz na specifický API endpoint, který zpracovává kontroler na straně serveru. Ten následně získá potřebná data z databáze a vrací je zpět ve formátu JSON, čímž umožňuje plynulé a asynchronní zobrazení událostí v kalendáři.



Obrázek 6: Diagram průběhu přidání uživatele pomocí AJAX a MVC

2.4.4 CSS frameworky (Bootstrap a alternativy)

Pro rychlý návrh moderního a responzivního uživatelského rozhraní je v systému *Agenda* využit framework Bootstrap [21]. Ten poskytuje rozsáhlou knihovnu předdefinovaných tříd a komponent pro tvorbu mřížkového rozložení, formulářů, tlačítek a dalších vizuálních prvků. Využitím těchto nástrojů lze výrazně urychlit vývoj frontendové části a zároveň zajistit jednotný vzhled celé aplikace [19].

Bootstrap je postaven na technologii CSS a využívá i JavaScript pro některé interaktivní komponenty. Díky responzivní mřížce (grid systému) lze snadno navrhovat rozhraní, které se přizpůsobí různým rozlišením a zařízením. Jeho použití je zvláště výhodné v prostředí veřejné správy, kde je žádoucí udržet vizuální konzistenci a přístupnost systému bez potřeby složitých návrhů na míru.

V projektu byly zvažovány i alternativy, jako například Bulma, Tailwind CSS nebo Materialize, nicméně Bootstrap byl zvolen pro svou širokou dokumentaci, stabilní komunitu a snadnou integraci do ASP.NET aplikací [21].

2.4.5 JavaScriptová knihovna jQuery

Knihovna jQuery je široce používaná open-source knihovna jazyka JavaScript, která byla navržena s cílem zjednodušit manipulaci s dokumenty ve formátu HTML, správou událostí, animacemi a komunikací pomocí technologie AJAX. Umožňuje psát méně kódu pro běžné úkoly a zajišťuje kompatibilitu napříč různými webovými prohlížeči [26].

Pro zajištění interaktivních funkcí na straně klienta využívá systém *Agenda* knihovnu jQuery. Ta se uplatňuje například při přetahování a změně pořadí kapitol prostřednictvím metody `sortable()` z rozšíření jQuery UI. Díky této funkcionalitě je možné spravovat obsah bez opětovného načítání stránky, což přispívá k vyšší plynulosti a komfortu při práci s aplikací. Dále jQuery slouží ke skriptování častých úkonů, jako je ovládání viditelnosti prvků nebo dynamická manipulace se strukturou Document Object Model (DOM).

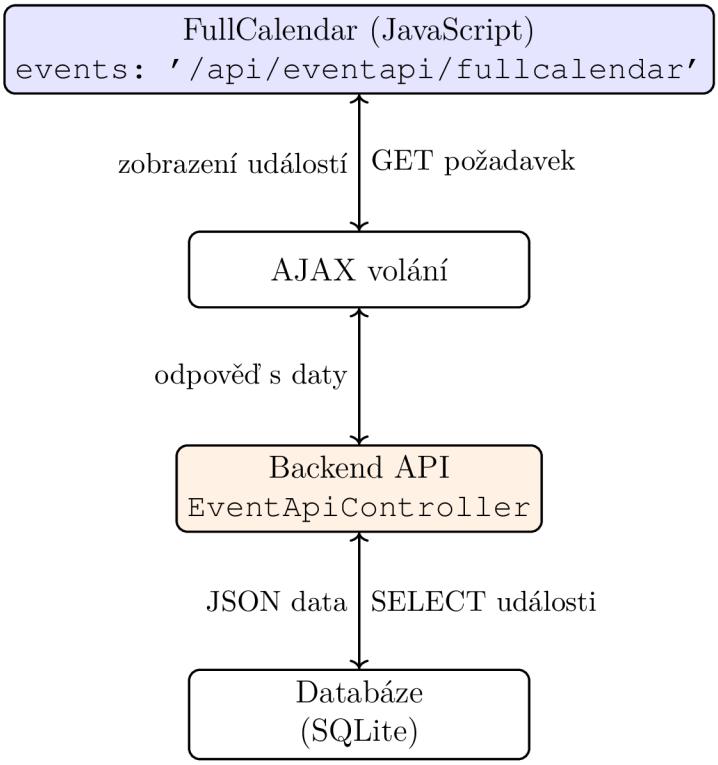
2.4.6 JavaScriptová knihovna FullCalendar

Pro vizualizaci událostí a plánování aktivit je v systému *Agenda* využívána open-source JavaScriptová komponenta **FullCalendar**. Tento nástroj umožňuje zobrazit kalendář s podporou denního, týdenního i měsíčního přehledu a nabízí také možnost drag-and-drop manipulace s událostmi.

Knihovna komunikuje s backendem pomocí AJAX volání, díky čemuž lze události dynamicky načítat i aktualizovat bez potřeby opětovného načtení stránky. Modul `iKalendář` systému *Agenda* využívá knihovnu FullCalendar pro přehledné zobrazení spuštěných programů jednotlivých oddělení.

Výhodou tohoto řešení je především vysoká přizpůsobitelnost, přehlednost zobrazení a snadná integrace do aplikací postavených na ASP.NET MVC [20].

Na obrázku 7 je znázorněna základní komunikace knihovny FullCalendar s backendem aplikace. JavaScriptová komponenta na klientské straně odesílá pomocí AJAX požadavku dotaz na specifický API endpoint, který zpracovává kontroler na straně serveru. Ten následně získá potřebná data z databáze a vrací je zpět ve formátu JSON, čímž umožňuje plynulé a asynchronní zobrazení událostí v kalendáři.



Obrázek 7: Komunikace knihovny FullCalendar s backendem systému *Agenda*

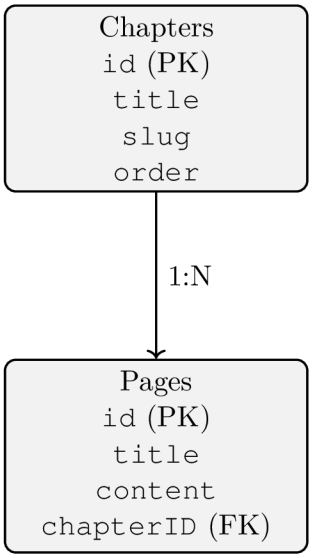
2.4.7 WYSIWYG editor CKEditor 5 pro práci s obsahem

Pro editaci a ukládání formátovaného textového obsahu do databáze je v systému *Agenda* nasazen moderní What You See Is What You Get (WYSIWYG) editor **CKEditor 5**. Tento editor poskytuje uživatelsky přívětivé prostředí pro formátování textu, vkládání odkazů, seznamů a dalších prvků bez nutnosti znalosti HTML.

Editor CKEditor 5 je využíván ve více částech systému. V modulu `iNávody` slouží k vytváření a ukládání výukových materiálů týkajících se programování v jazyce Common Business-Oriented Language (COBOL), práce na *mainframe* a činností oddělení. Dále je implementován v modulu programové dokumentace a také v elektronickém systému analýz, kde uživatelé prostřednictvím editace formátovaného textu připravují technické popisy řešení.

Ve všech případech je výsledný HTML obsah bezpečně uložen do databáze a při zobrazení vykreslen ve veřejném rozhraní aplikace. Editor je v systému

Agenda napojen na dvě databázové tabulky: `chapters` a `pages`, které slouží ke správě výukových kapitol a jejich podstránek. Vztah mezi těmito entitami je znázorněn na obrázku 8.



Obrázek 8: ER vazba mezi tabulkami `Chapters` a `Pages`

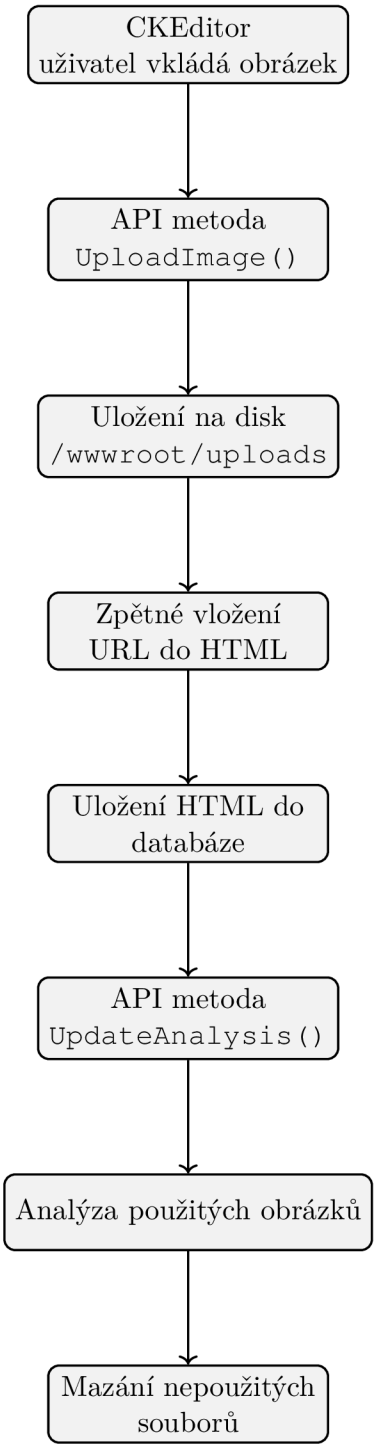
Při implementaci bylo nutné řešit nahrávání obrázků vkládaných do obsahu. Z důvodu omezené funkčnosti neplacené verze CKEditoru bylo vytvořeno vlastní řešení, které umožňuje nahrát obrázek na server, automaticky jej pojmenovat podle dokumentace a vložit do editoru. Soubory jsou pro účely této práce ukládány do složky `wwwroot`, snadno dostupné při vývoji a testování. V produkčním prostředí však bude jejich správa řešena odděleně, mimo veřejně přístupné prostředky, s využitím samostatného úložiště s řízeným přístupem.

Neplacená verze CKEditoru 5 poskytuje pouze základní funkcionalitu pro práci s textem. Chybí například možnost měnit velikost a typ písma, barvu textu či jeho pozadí. Dále není dostupná funkce podtržení, pokročilé styly nadpisů ani rozšířená práce s obrázky – například zarovnání, obtékání textu nebo nastavení alternativního popisu (atributu `alt`). Rozšíření editoru o tyto funkce je možné pouze prostřednictvím placených doplňků nebo komerční licence.

Zvolený přístup, který kombinuje základní editor s vlastní logikou zpracování nahraných souborů, představuje kompromis mezi funkčností a technickou nezávislostí na komerčních modulech. Nepoužívané soubory jsou průběžně identifikovány a automaticky odstraňovány ze serveru. Tento mechanismus přispívá k udržení přehledného úložiště a zároveň eliminuje závislost na placených rozšířeních editoru.

S ohledem na další rozšiřování systému a rostoucí požadavky uživatelů by však do budoucna stálo za zvážení nasazení alternativního WYSIWYG editoru s otevřenou architekturou a širší sadou funkcionalit bez licenčních omezení.

Proces bezpečného nahrávání souborů a následného odstraňování nepoužitých dat je schematicky znázorněn na obrázku 9.



Obrázek 9: Proces nahrávání obrázku a úklidu nepoužitých souborů

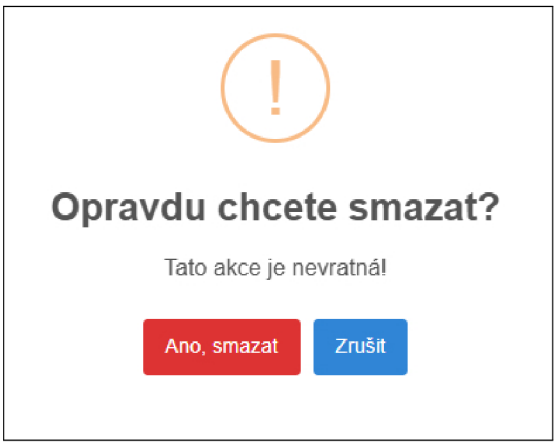
Tato technologie výrazně zvyšuje flexibilitu a použitelnost systému při práci s textem, přičemž nabízí výbornou rozšiřitelnost a integraci do moderních webových aplikací [27].

2.4.8 Knihovna SweetAlert2 pro modální dialogy

SweetAlert2 je moderní JavaScriptová knihovna sloužící k zobrazování stylizovaných modálních dialogových oken ve webových aplikacích. Oproti nativním funkcím jazyka JavaScript, jako jsou `alert()`, `confirm()` nebo `prompt()`, nabízí podstatně širší možnosti vzhledu a chování. Uživatelé mohou prostřednictvím této knihovny zobrazovat informační, chybová či potvrzovací hlášení v graficky konzistentní podobě, která odpovídá celkovému vizuálnímu stylu aplikace.

Knihovna podporuje plně responzivní zobrazení, vlastní ikony, tlačítka, vstupní pole a různé animace. Důležitou vlastností je i možnost následného programového zpracování reakce uživatele, například potvrzení nebo zrušení akce. V systému *Agenda* je SweetAlert2 využívána zejména pro zobrazování úspěšných či chybových hlášení, potvrzení mazání záznamů a další interaktivní prvky spojené s odezvou systému vůči uživatelským akcím. [28]

Na obrázku 10 je ukázka modálního dialogového okna vytvořeného pomocí knihovny SweetAlert2, zobrazujícího potvrzení odstranění záznamu.



Obrázek 10: Ukázka dialogového okna vytvořeného pomocí SweetAlert2

2.4.9 Responzivita a přizpůsobení zobrazení

Při návrhu uživatelského rozhraní systému *Agenda* byly zohledněny moderní trendy webového vývoje, včetně podpory responsivního designu. Přestože knihovna `Bootstrap`, použitá v projektu, nabízí vestavěné mechanismy pro tvorbu responzivního rozvržení (např. grid systém, media queries), cílové zařízení systému bylo od počátku jasně definováno.

Systém je určen pro použití na standardních pracovních stanicích s rozlišením **Full HD** (1920 × 1080 px) a vyšším. Vzhledem k tomuto omezenému rozsahu

zařízení nebyla zohledňována podpora pro mobilní telefony či tablety. Uživatelské rozhraní je optimalizováno tak, aby efektivně využívalo dostupný prostor na větších obrazovkách a zobrazovalo co nejvíce informací bez nutnosti častého přepínání či rolování.

2.5 Externí skripty pro správu systému

V rámci systému jsou využívány externí skripty napsané v jazyce *PowerShell* pro správu databáze a datových úložišť.

PowerShell je výkonný příkazový řádkový interpret a skriptovací jazyk vyvinutý společností *Microsoft*. Jeho hlavním cílem je automatizace administrativních úloh a efektivní správa systémových prostředků. PowerShell je postaven na .NET platformě, což umožňuje přístup k širokému spektru funkcionalit operačního systému, včetně práce se souborovým systémem, správou procesů, registrů, sítí či služeb. Díky integraci objektového modelu může PowerShell předávat mezi příkazy složité datové struktury, nikoli pouze textové výstupy, čímž poskytuje vyšší flexibilitu než tradiční příkazové interprety [29].

První skript provádí zálohu databázového souboru SQLite. Tento soubor je zkopírován, zabalen do archivu a uložen do předem určené archivní složky na lokálním disku. Zároveň je vytvořena kopie archivu ve složce synchronizované se službou *OneDrive*, odkud je následně automaticky zálohována do cloudového úložiště prostřednictvím systému *Windows*.

Druhý skript slouží k údržbě databáze. Odstraňuje staré logy a také provádí mazání starších kopií odeslaných e-mailových zpráv, které jsou systémem ukládány pro účely archivace.

Oba skripty jsou automaticky spouštěny v nočních hodinách mezi 2:30 a 2:45, tedy mimo běžnou pracovní dobu. Díky tomu nedochází k žádné souběžné zátěži databáze a je minimalizováno riziko narušení provozu systému.

Použití těchto skriptů přispívá k dlouhodobé udržitelnosti systému, zajištění bezpečnosti dat a efektivní správě úložného prostoru.

2.6 Adresářová struktura projektu

Projekt *Agenda* je vytvořen jako ASP.NET MVC aplikace se standardní strukturou rozšířenou o několik vlastních složek. Zdrojové soubory jsou logicky rozděleny do samostatných adresářů podle své odpovědnosti, což zajišťuje přehlednost, snadnou údržbu i rozšiřitelnost systému.

Zjednodušené schéma hlavních složek projektu je znázorněno na obrázku 11.

- `Controllers` – kontrolery zajišťující řízení aplikační logiky,
- `Models`, `Enums`, `Options` – datové modely, výčtové typy a konfigurační objekty,
- `Services`, `Contracts` – aplikační logika a rozhraní pro práci s e-mailem a dalšími funkcemi,

- `Middleware`, `Extensions` – rozšíření a zpracování požadavků,
- `ViewComponent`, `Views` – komponenty a šablony pro uživatelské rozhraní,
- `Data`, `Migrations` – datová vrstva, kontext databáze a migrace,
- `Program.cs`, `appsettings.json` – vstupní bod aplikace a konfigurační soubor,
- `agenda.db` – lokální databáze ve formátu SQLite.

Součástí kódu je také systematické používání inline dokumentace pomocí XML summary komentářů. Tento přístup umožňuje nejen automatickou generaci technické dokumentace, ale také efektivnější zapojení více vývojářů do budoucího rozvoje systému díky jednotné a dohledatelné popisnosti kódu.

```
AgendaF/
|-- Controllers/
|-- Models/
|-- Enums/
|-- Options/
|-- Services/
|-- Contracts/
|-- Data/
|-- Middleware/
|-- Extensions/
|-- ViewComponent/
|-- Views/
|-- Migrations/
|-- Program.cs
|-- appsettings.json
\-- agenda.db
```

Obrázek 11: Zjednodušená adresářová struktura projektu *Agenda*

3 Moduly IS Agenda

Tato kapitola se věnuje popisu jednotlivých funkčních modulů systému *Agenda*, které tvoří základ jeho aplikační logiky a rozhraní pro uživatele. Každý modul je navržen jako samostatná komponenta, která řeší specifickou oblast interních procesů oddělení správy a vývoje aplikačního programového vybavení důchodových agend.

Na obrázku 12 je zobrazena úvodní obrazovka systému *Agenda*, která poskytuje přehledné rozhraní pro přístup k jednotlivým modulům a funkcím aplikace.

Modulární architektura umožňuje nejen lepší přehlednost kódu, ale i snadnější rozšiřitelnost systému v budoucnosti. Jednotlivé části aplikace jsou navzájem propojeny prostřednictvím společných modelů a sdílených služeb, přičemž dodržují principy oddělení zodpovědnosti a opakovatelného návrhu.

Moduly byly navrženy na základě analýzy aktuálních potřeb oddělení, přičemž důraz byl kladen na jednoduchost ovládání, srozumitelnost a možnost údržby. V této kapitole jsou podrobně popsány funkce, uživatelské rozhraní i specifické požadavky jednotlivých částí systému. Součástí jsou také poznámky k návrhu databáze, vazbám mezi entitami a případné technické poznámky k implementaci.

Cílem této části práce je přiblížit logiku systému jak z hlediska uživatele, tak z pohledu vývojáře, a poskytnout úplný přehled o implementovaných funkcionalitách systému *Agenda*.

Obrázek 12: Hlavní obrazovka systému *Agenda*

3.1 iÚkoly – Interní úkoly

Modul `iÚkoly` představuje základní součást informačního systému *Agenda* a zajišťuje kompletní workflow interních vývojových požadavků oddělení. Skládá se ze čtyř navzájem provázaných částí: **Požadavky**, **Úkoly**, **Dokumentace** a **Práce**. Každá část představuje jednu fázi zpracování požadavku a všechny části spolu úzce souvisejí.

Základní datový model modulu je postaven na následujícím vztahu: jeden požadavek může mít maximálně jeden interní úkol (`IU`), přičemž tento úkol může obsahovat více dokumentací. Závěrečná část **Práce** slouží k zobrazení aktuálně

přiřazených úloh jednotlivých analytiků a programátorů.

3.1.1 Požadavky

Sekce Požadavky zajišťuje evidenci a správu všech požadavků pro oddělení. Uživatelé s oprávněním (vedoucí, zástupce, asistent) zde mohou zakládat nové požadavky. U každého požadavku jsou evidovány následující údaje: název, číslo jednací, popis, zadavatel a datum obdržení.

Po schválení požadavku má oprávněný uživatel možnost vygenerovat interní úkol. Administrace požadavků zahrnuje možnosti úprav, detailního náhledu, uzavření, znovuotevření nebo smazání požadavku. Některé akce jsou dostupné pouze v určitých stavech požadavku a při splnění oprávnění – např. nelze smazat požadavek, který má přiřazený interní úkol. Z důvodu bezpečnosti jsou administrační akce zobrazeny až po stisknutí tlačítka „Zobrazit admin akce“.

3.1.2 Úkoly

Tato sekce zobrazuje všechny vytvořené interní úkoly. Každý úkol je rozšířen o informativní údaj *datum očekávaného splnění*, který v budoucnu umožní generování statistik a notifikací. Interní úkoly vznikají výhradně v sekci Požadavky prostřednictvím tlačítka „Generovat IU“.

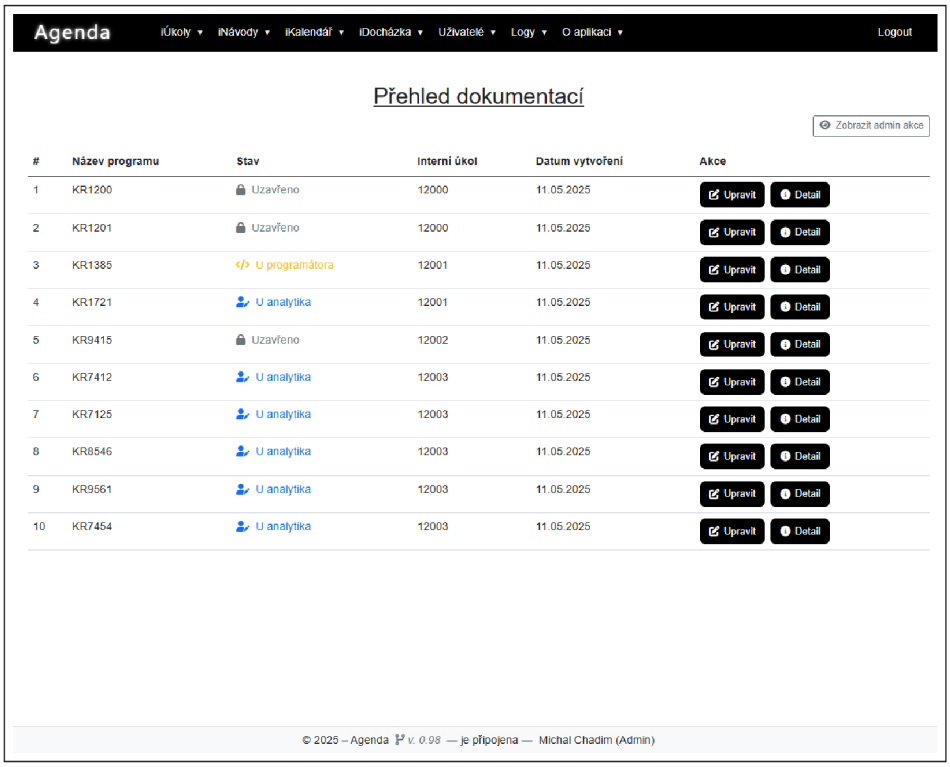
I zde jsou k dispozici administrační akce (detail, úprava, uzavření, znovuotevření, smazání), jejichž dostupnost závisí na stavu úkolu. Např. úkol s aktivní dokumentací nelze smazat. V této sekci se rovněž zakládají jednotlivé dokumentace, které jsou svázané s daným interním úkolem v poměru 1:N.

3.1.3 Dokumentace

Zakládání dokumentací probíhá v předchozí sekci pomocí moderního formuláře. Po vytvoření nové dokumentace jsou automaticky odeslány notifikační e-maily: analytik je informován o nutnosti vyplnit analýzu a programátor je upozorněn na očekávanou práci. Samotná analýza se vyplňuje prostřednictvím WYSIWYG editoru a je spolu s dalšími metadaty uložena do databáze.

Při dokončení analýzy může analytik programátora informovat o připraveném zadání. Po dokončení programátor dokumentaci vrací analytikovi ke kontrole. Pokud je vše v pořádku, může být dokumentace uzavřena, čímž je informován vedoucí pracovník. Všechny tyto akce jsou dostupné dle role uživatele.

Ukázka hlavní obrazovky části Dokumentace, je zobrazena na obr. 13.



Obrázek 13: Hlavní obrazovka části Dokumentace v rámci modulu *iÚkoly*

3.1.4 Práce

Závěrečná část **Práce** zobrazuje jednotlivým uživatelům (analytikům i programátorům) jejich aktivní dokumentace. Díky tomuto přehledu není nutné tyto dokumentace vyhledávat v celkovém seznamu. Pomocí odkazu „Detail“ se uživatel dostane do stejného administračního rozhraní jako v sekci Dokumentace.

3.1.5 Komerční řešení a moderní trendy

Na trhu existuje celá řada placených systémů pro správu interních úkolů a týmové spolupráce, mezi které patří například *Jira*, *Asana*, *ClickUp* nebo *Trello*. Tyto nástroje nabízejí pokročilé možnosti plánování, integraci s dalšími systémy, vizualizaci úkolů pomocí Kanban nebo Gantt diagramů, notifikace, reporting i mobilní aplikace.

Moderním trendem je důraz na **uživatelskou přívětivost**, **automatizaci procesů** a možnost **vzdálené týmové spolupráce**. Některé systémy využívají také Artificial Intelligence (AI) pro prediktivní plánování nebo vyhodnocování efektivity. Například studie publikovaná v časopise *Applied Sciences* uvádí, že AI techniky, jako je strojové učení a hybridní modely, mohou výrazně zlepšit oblasti plánování, měření a zvládání nejistot v projektovém řízení tím, že poskytují slibné schopnosti prognózování a rozhodování [30].

3.2 iNávody – Výuka pro nové pracovníky oddělení

Modul *iNávody* slouží jako interní výukový nástroj pro nové pracovníky oddělení správy a vývoje důchodových agend. Cílem modulu je efektivně předat základní znalosti a principy práce s interními systémy, používanými technologiemi a specifiky prostředí **Mainframe**.

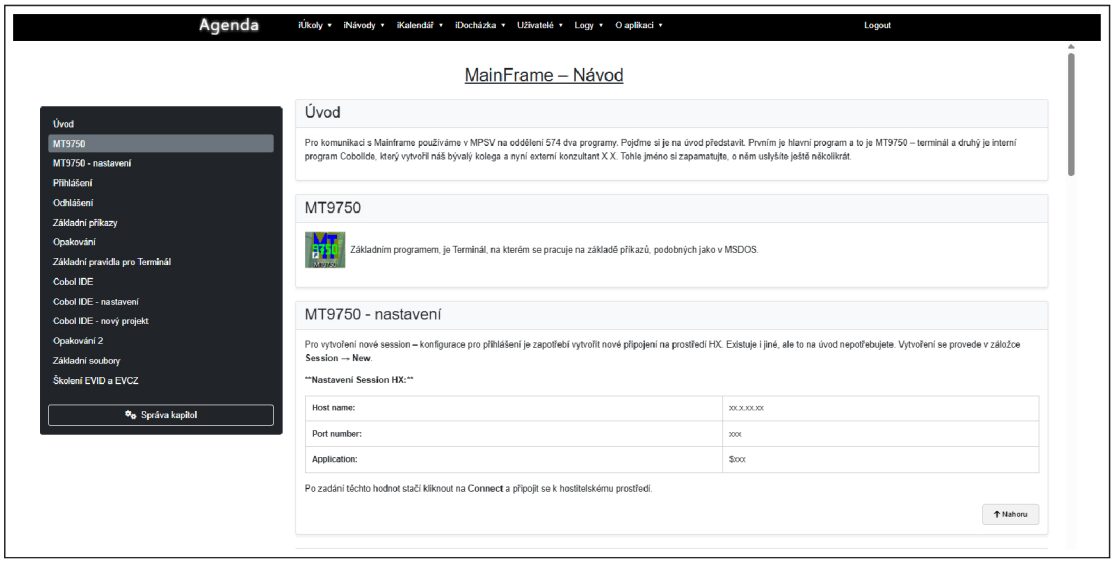
Systém *iNávody* zahrnuje několik tematických sad, které pokrývají:

- základy práce v prostředí **Mainframe** včetně příkazů,
- výuku programovacího jazyka **COBOL**,
- popis běžných činností a agend řešených na oddělení.

Uživatelské rozhraní modulu je plně dynamické – jednotlivé kapitoly a jejich obsah jsou generovány na základě dat uložených v databázi. K editaci a ukládání formátovaného HTML obsahu slouží vizuální editor **CKEditor 5**.

Obsah je prezentován prostřednictvím Razor stránky, která využívá model ve formátu `IEnumerable<Chapter>`. Navigační menu i jednotlivé sekce se generují automaticky na základě struktury uložené v databázi.

Ukázka tématické sady Mainframe v rámci modulu *iNávody* je zobrazena na obr. 14.



Obrázek 14: Tématická sada Mainframe v modulu *iNávody*

3.3 iKalendář – Plánování úloh

Modul *iKalendář* slouží k zobrazení a správě plánovaných aktivit oddělení, především pravidelně spouštěných programů. Původní způsob evidence pomocí tabulky v aplikaci Microsoft Excel byl nahrazen moderním webovým řešením, které umožňuje rychlý a přehledný přístup k informacím odkudkoli v rámci interní sítě.

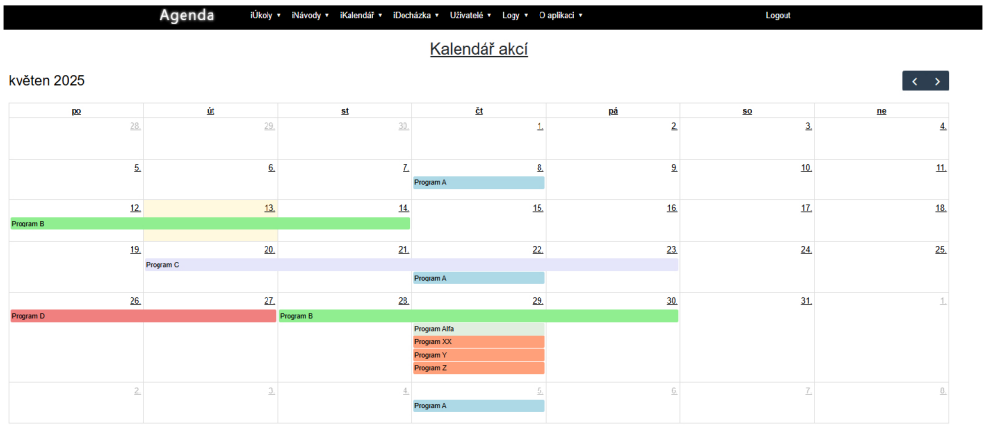
Zobrazení plánovaných událostí je realizováno pomocí knihovny **FullCalendar**, která umožňuje interaktivní zobrazení kalendáře s podporou barevného odlišení jednotlivých typů událostí. Kalendář získává data z databáze pomocí asynchronního API volání. V databázi jsou jednotlivé záznamy spravovány přes administrační rozhraní, kde oprávnění uživatelé mohou přidávat, upravovat nebo mazat události.

Každá událost je popsána sadou parametrů, mezi které patří:

- název události (`Title`),
- datum a čas začátku (`StartDate`),
- datum a čas konce (`EndDate`),
- typ události (`EventType`).

Typ události je reprezentován samostatnou entitou, která je ukládána v databázi a umožňuje dynamickou správu bez nutnosti zásahu do zdrojového kódu. Každému typu je přiřazen název a barva (např. světle modrá nebo světle zelená), která určuje způsob jeho zobrazení v kalendáři. Aktuálně je definováno pět typů událostí, ale struktura systému umožňuje jejich snadné rozšiřování a přizpůsobení vzhledu podle aktuálních potřeb.

Ukázka zobrazení kalendáře s plánovanými aktivitami pomocí knihovny *FullCalendar* je na obr. 15.



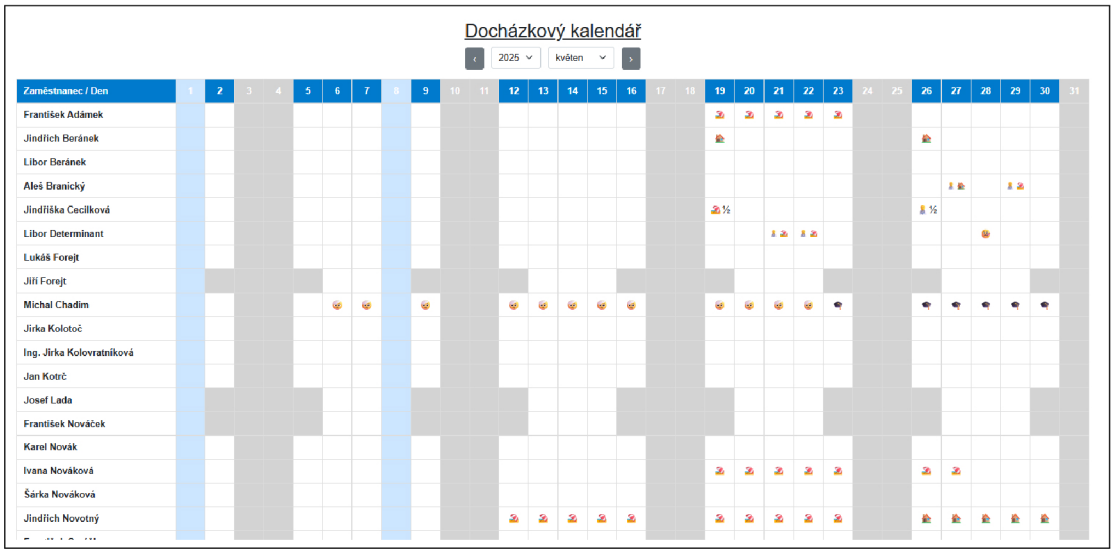
Obrázek 15: Zobrazení plánovaných událostí v modulu *iKalendář* pomocí knihovny *FullCalendar*

3.4 iDocházka – Přítomnost zaměstnanců

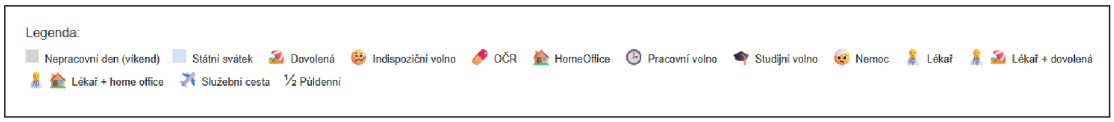
Modul *iDocházka* je navržen jako interní nástroj pro správu a přehledné zobrazení nepřítomnosti zaměstnanců oddělení. Jeho cílem je nahradit původní řešení na bázi tabulek v Excelu moderním systémem, který je přístupný online a nabízí aktuální informace o přítomnosti jednotlivých členů týmu.

Základ modulu tvoří webová stránka s přehledným měsíčním kalendářem, ve kterém jsou vizuálně zobrazeny nepřítomnosti jednotlivých zaměstnanců. Na rozdíl od modulu *iKalendář*, zde nebyl použit nástroj *FullCalendar*, neboť ve své bezplatné verzi nenabízí požadovaný typ zobrazení. Místo toho bylo zvoleno vlastní řešení postavené na čistém JavaScriptu, které získává data přes dvě samostatná API volání – první vrací seznam zaměstnanců oddělení a druhé poskytuje záznamy o jejich nepřítomnostech. Tato data jsou následně vykreslena do dynamické tabulky s barevným zvýrazněním.

Ukázka vykreslovacího kalendáře je zobrazena na obr. 16, legenda použitých emotikonů pak na obr. 17.



Obrázek 16: Zobrazení nepřítomnosti zaměstnanců v dynamickém kalendáři modulu *iDocházka*



Obrázek 17: Legenda emotikonů nepřítomnosti zaměstnanců v rámci modulu *iDocházka*

Modul je rozdělen do dvou hlavních částí:

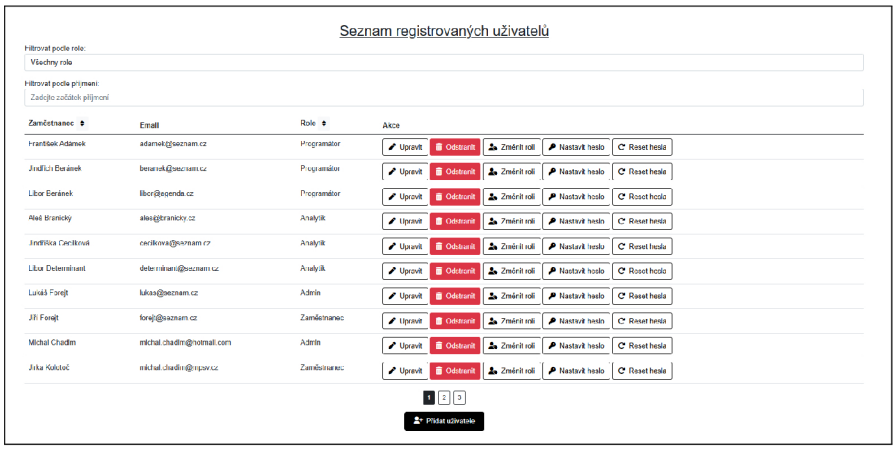
- **Zobrazení nepřítomnosti** – přehledný vizuální kalendář pro všechny zaměstnance oddělení, sloužící jako okamžitý zdroj informací o plánovaných dovolených, školeních, neschopenkách nebo jiných typech absence.
- **Správa docházky** – rozhraní dostupné přihlášeným uživatelům, které umožňuje vytvářet, upravovat a mazat záznamy o nepřítomnosti. Uživatelé mohou zadávat své vlastní plánované nepřítomnosti, zatímco vedoucí pracovníci mají možnost schvalovat nebo zamítat tyto požadavky.

Tímto způsobem je dosaženo vyšší transparentnosti v týmu, usnadnění plánování pracovních činností i rychlejší koordinaci v případě zástupů. Modul zároveň plní roli jednoduchého workflow systému s oprávněními na základě rolí, a je tak důležitou součástí vnitřní agendy oddělení.

3.5 iUživatel – Administrace

Modul *iUživatel* slouží k administraci uživatelů informačního systému *Agenda*. Je přístupný pouze oprávněným osobám, mezi které patří administrátoři, vedoucí oddělení a jejich zástupci. Jeho účelem je zajištění kompletní správy uživatelských účtů, přiřazování rolí a dohledu nad bezpečnostními aspekty přístupu do systému.

Ukázka administrační obrazovky pro oprávněné osoby je zobrazena na obr. 18.



Obrázek 18: Administrace uživatelů – přístupná pouze oprávněným osobám

Modul umožňuje vytvářet nové uživatele, upravovat jejich údaje, nastavovat nebo měnit jejich role (např. běžný uživatel, vedoucí, administrátor), případně účty deaktivovat. Možnost mazání uživatelů je z bezpečnostních důvodů ve výchozím nastavení deaktivována, avšak může být jednoduše opětovně aktivována v kódu aplikace v případě potřeby.

Správa uživatelů je implementována nad systémem Identity, což zajišťuje vysokou míru bezpečnosti při autentizaci a autorizaci, včetně podpory hashování hesel, správy relací a ověřování přístupových práv. Pro přehlednost je rozhraní modulu navrženo s důrazem na uživatelskou přívětivost a jednoduché ovládání.

3.5.1 Detail uživatele

Každý přihlášený uživatel systému má přístup ke svému profilu prostřednictvím sekce *Detail uživatele*. Tato část slouží ke správě osobních údajů a k údržbě přihlašovacích údajů. Uživatel zde může zobrazit a upravit informace jako jméno, příjmení, emailová adresa, telefonní číslo, pracoviště nebo popis činnosti.

Součástí tohoto rozhraní je také možnost změny hesla. Funkcionalita změny hesla je realizována v souladu s bezpečnostními doporučeními a využívá mechanismy knihovny Identity, včetně ověření původního hesla a validace nového hesla dle nastavených pravidel.

Uživatelská editace je důležitým prvkem systému nejen z hlediska bezpečnosti, ale také z pohledu personalizace prostředí a zajištění aktuálnosti informací. Data zadaná uživatelem jsou validována na straně klienta i serveru, aby byla zajištěna jejich integrita a správnost.

3.5.2 Seznam uživatelů

Administrátorská sekce *Seznam uživatelů* poskytuje oprávněným osobám (administrátorům, vedoucím a jejich zástupcům) rozšířený přístup ke správě všech registrovaných uživatelů v systému. Tento modul umožňuje provádět základní operace, jako je úprava osobních údajů, změna hesla, deaktivace nebo reset hesla, a dále také rozšířenou správu přístupových práv.

Vedoucí má v systému oprávnění resetovat heslo uživateli, což znamená vygenerování nového dočasného hesla, zatímco administrátor může heslo přímo změnit. Dále mohou uživatelům měnit jejich roli v systému, což ovlivňuje jejich oprávnění v rámci jednotlivých modulů aplikace.

Pro lepší přehlednost a efektivní práci obsahuje tato sekce nástroje pro řazení a filtrování. Uživatelé mohou být seřazeni podle příjmení (abecedně) nebo podle přiřazené role. Kromě toho je možné využít textové vyhledávání, které pracuje s počátečními znaky příjmení a umožňuje tak rychlé vyhledání konkrétního zaměstnance.

Díky těmto funkcím je modul *Seznam uživatelů* důležitým nástrojem pro správu přístupových práv a udržování aktuálních informací v systému.

3.5.3 Role uživatelů

Tato sekce slouží ke správě uživatelských rolí v systému *Agenda*. Role určují oprávnění jednotlivých uživatelů a jejich přístup k jednotlivým modulům a funkcím systému. Modul *Role uživatelů* je přístupný výhradně uživatelům s rolí administrátora.

V současné verzi systému je z bezpečnostních důvodů povoleno pouze přidávání nových rolí. Tato omezená funkcionalita zajišťuje konzistenci systému a minimalizuje riziko nechtěných změn v nastavení oprávnění.

Ostatní operace, jako je úprava nebo mazání existujících rolí, jsou v rámci uživatelského rozhraní standardně skryty. Tyto funkce jsou však technicky implementovány a mohou být v případě potřeby aktivovány úpravou zobrazení ve vrstvě *View*. Toto řešení umožňuje zachovat flexibilitu systému, aniž by byla ohrožena jeho stabilita či bezpečnost.

Role tvoří důležitý prvek řízení přístupu a zajišťují, aby jednotlivé části systému byly dostupné pouze oprávněným uživatelům v souladu s jejich pracovním zařazením.

3.6 Logy

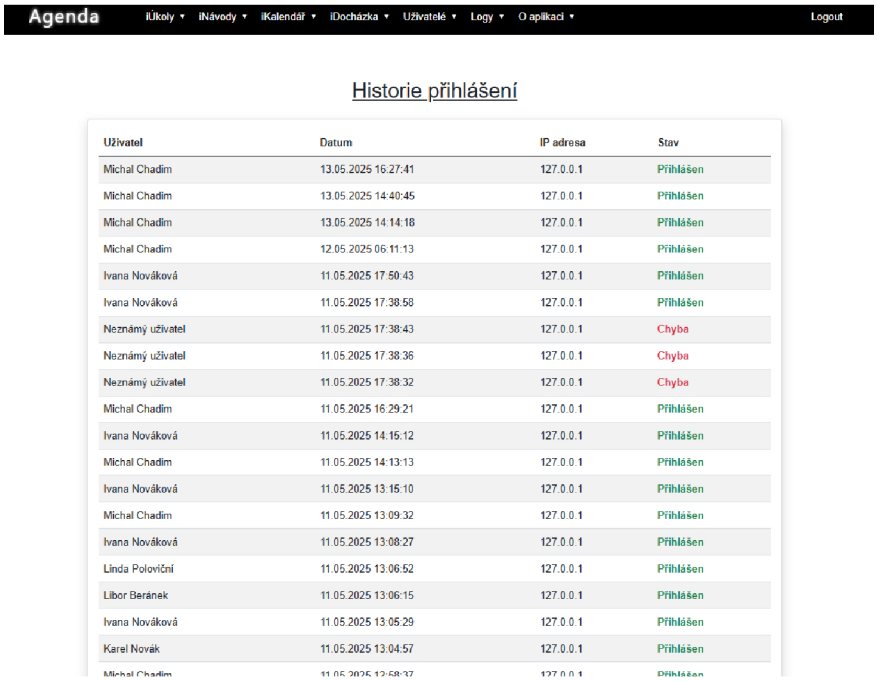
Modul **Logy** slouží výhradně administrátorům systému a umožňuje detailní náhled do interních záznamů o chodu aplikace *Agenda*. Jeho cílem je poskytnout přehled o aktivitách uživatelů a případných problémech systému, které mohou sloužit jako podklad pro audit, ladění nebo odhalení bezpečnostních incidentů.

Aplikace zaznamenává tři hlavní typy událostí do samostatných databázových tabulek:

- `LoginLogs` – obsahuje informace o úspěšných i neúspěšných pokusech o přihlášení,
- `UserLog` – sleduje vytvoření a odstranění uživatelských účtů,
- `ExceptionsLogs` – eviduje systémové výjimky, pokusy o neoprávněný přístup a další chyby aplikace.

Z bezpečnostních důvodů je modul navržen jako čistě pasivní – umožňuje pouze zobrazení logovaných záznamů bez jakékoli možnosti úprav, mazání nebo manipulace ze strany uživatele. Tím je zajištěna integrita logovaných informací a zabráněno neoprávněným zásahům do historie událostí.

Ukázka historie přihlášení uživatelů je zobrazena na obr. 19.



Uživatel	Datum	IP adresa	Stav
Michal Chadim	13.05.2025 16:27:41	127.0.0.1	Přihlášen
Michal Chadim	13.05.2025 14:40:45	127.0.0.1	Přihlášen
Michal Chadim	13.05.2025 14:14:18	127.0.0.1	Přihlášen
Michal Chadim	12.05.2025 06:11:13	127.0.0.1	Přihlášen
Ivana Nováková	11.05.2025 17:50:43	127.0.0.1	Přihlášen
Ivana Nováková	11.05.2025 17:38:58	127.0.0.1	Přihlášen
Neznámý uživatel	11.05.2025 17:38:43	127.0.0.1	Chyba
Neznámý uživatel	11.05.2025 17:38:36	127.0.0.1	Chyba
Neznámý uživatel	11.05.2025 17:38:32	127.0.0.1	Chyba
Michal Chadim	11.05.2025 16:29:21	127.0.0.1	Přihlášen
Ivana Nováková	11.05.2025 14:15:12	127.0.0.1	Přihlášen
Michal Chadim	11.05.2025 14:13:13	127.0.0.1	Přihlášen
Ivana Nováková	11.05.2025 13:15:10	127.0.0.1	Přihlášen
Michal Chadim	11.05.2025 13:09:32	127.0.0.1	Přihlášen
Ivana Nováková	11.05.2025 13:08:27	127.0.0.1	Přihlášen
Linda Poloviční	11.05.2025 13:06:52	127.0.0.1	Přihlášen
Libor Beránek	11.05.2025 13:06:15	127.0.0.1	Přihlášen
Ivana Nováková	11.05.2025 13:05:29	127.0.0.1	Přihlášen
Karel Novák	11.05.2025 13:04:57	127.0.0.1	Přihlášen
Michal Chadim	11.05.2025 12:58:37	127.0.0.1	Přihlášen

Obrázek 19: Zobrazení historie přihlášení uživatelů v modulu *Logy*

3.7 O aplikaci

Závěrečná část systému *Agenda* slouží jako interní dokumentační rozhraní, které poskytuje dvě klíčové informace: popis aplikace a historii jejích verzí. Obě tyto

sekce jsou generovány dynamicky na základě dat uložených v databázi a poskytují programátorům a uživatelům snadno přístupný přehled o vývoji systému a jeho funkcionalitě.

Dokumentace aplikace je realizována stejným způsobem jako modul *iNávody*. Administrátoři mohou definovat jednotlivé kapitoly a obsah prostřednictvím integrovaného editoru **CKEditor 5**.

Verzovací systém přehledně zachycuje vývoj jednotlivých verzí systému *Agenda*. Každá verze je uložena jako samostatný záznam s datem vydání, číslem verze, kategorií (např. stabilní, testovací) a seznamem změn. Změny jsou reprezentovány sadou textových popisů a vizuálních ikon, které jsou rovněž ukládány do databáze.

Komponenta `Accordion` z frameworku Bootstrap je využita pro uživatelské rozhraní zobrazující historii verzí. Tím je dosaženo přehledné a responzivní prezentace verzí, kde jsou detailní informace o každé verzi skryty v rozbalitelných sekcích. Nejnovější verze je vizuálně zvýrazněna pro rychlou orientaci.

Administrátoři mohou nové verze a změny přidávat prostřednictvím jednoduchého formuláře, čímž je zajištěna plná kontrola nad obsahem a aktuálností informací.

Díky tomuto modulu je možné zpětně dohledat, jaké změny byly v systému provedeny, kdy a proč – což zvyšuje transparentnost vývoje a zjednodušuje správu verzí zejména v prostředí veřejné správy, kde je potřeba udržovat detailní záznamy o úpravách informačních systémů.

3.8 E-mailové notifikace uživatelům

Systém *Agenda* obsahuje komponentu pro automatické e-mailové notifikace, která slouží k informování uživatelů o důležitých událostech. Notifikace jsou generovány např. při vytvoření nového účtu, požadavku na reset hesla, schvalování žádosti o nepřítomnost (v případě změny stavu) a také při přidělení dokumentace konkrétnímu uživateli.

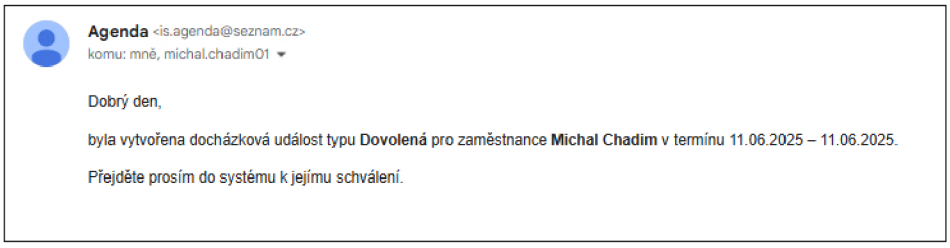
Z technického hlediska je odesílání zpráv realizováno pomocí asynchronní fronty, do níž jsou zprávy vkládány z jednotlivých částí systému. Zpracování fronty probíhá odděleně od hlavního běhového vlákna webové aplikace, čímž je zajištěno, že odesílání e-mailů neblokuje reakční čas serveru. Tímto způsobem je systém lépe škálovatelný a odolnější vůči dočasným výpadkům poštovního serveru.

Odesílání samotných e-mailů je implementováno s využitím třídy `SmtpClient` v prostředí NET, přičemž přihlašovací údaje a parametry připojení (např. adresa serveru, port, zabezpečené připojení) jsou uloženy v konfiguračním souboru `appsettings`. Citlivé informace, jako jsou e-mailová adresa a heslo odesílatele, jsou odděleny do chráněné části konfigurace a nejsou veřejně dostupné v běžné distribuci systému.

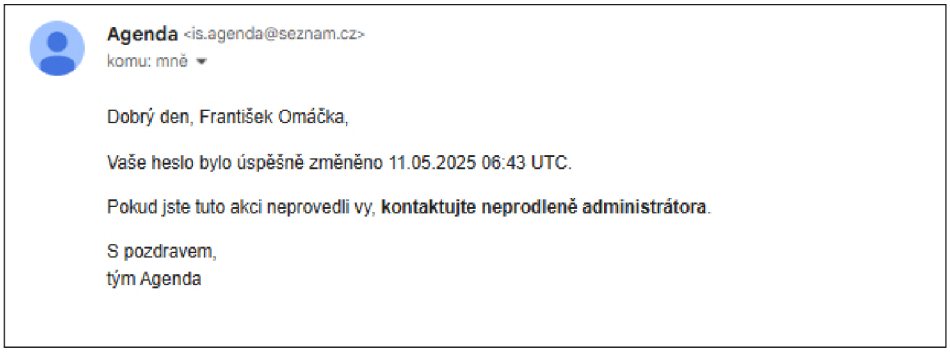
Notifikace jsou generovány na základě šablon, které mohou obsahovat proměnné jako jméno uživatele, název dokumentace či přímý odkaz do systému.

Celý mechanismus tak přispívá k efektivní komunikaci mezi systémem a uživateli, zvyšuje přehlednost a snižuje potřebu manuální kontroly změn.

Ukázky odeslaných notifikačních e-mailů jsou uvedeny na obr. 20 a 21.



Obrázek 20: Ukázka e-mailové notifikace při založení události v iDocházce



Obrázek 21: Ukázka e-mailové notifikace po změně přihlašovacího hesla

4 Budoucí vývoj IS

I když informační systém *Agenda* již v aktuální podobě pokrývá řadu klíčových interních agend oddělení, jeho architektura a implementace byly navrženy s důrazem na rozšiřitelnost a budoucí rozvoj. V rámci dalšího vývoje lze zvažovat přidání nových funkcionalit, vylepšení stávajících modulů nebo integraci s dalšími interními systémy.

Následující body představují nejvýznamnější směry, kterými by se mohl vývoj systému *Agenda* ubírat:

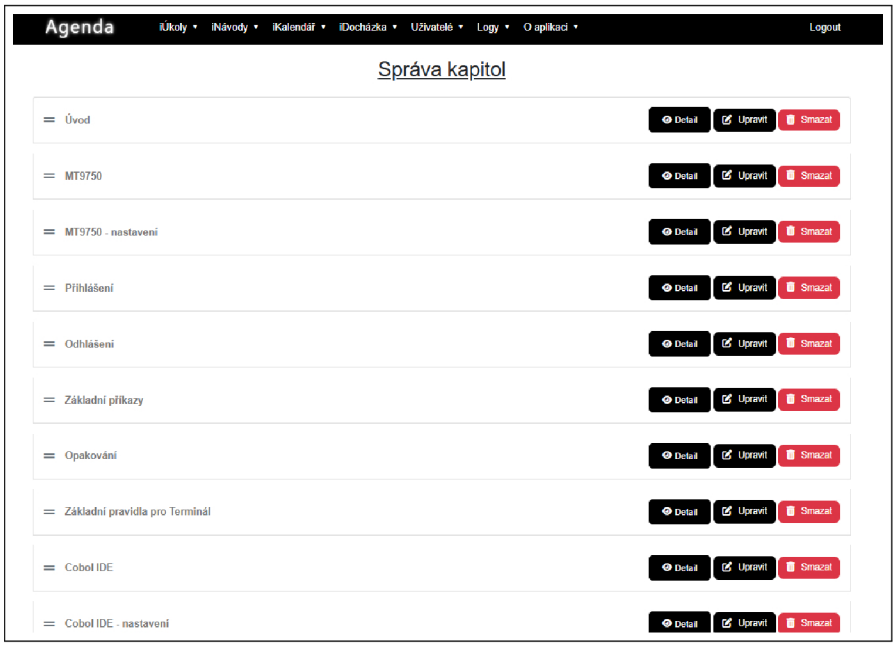
- rozšíření modulu **interních úkolů** o priority, stavové workflow, termíny a možnost přiřazení více uživatelů,
- vylepšení **správy oprávnění** – např. detailnější členění přístupových práv nebo možnost vytvářet vlastní role,
- rozšíření modulu **docházky** o exportní funkce a možnosti integrace s externími personálními systémy,

- rozšíření modulu **iKalendář** o exportní nástroj umožňující přímý výstup do formátu kompatibilního s *Microsoft Excel* pro další zpracování,
- přidání funkcionality pro zpracování a implementaci požadavků zaslaných prostřednictvím **interního helpdesku**, včetně možnosti kategorizace, evidence stavu a automatizovaného přiřazení,
- vylepšení modulu **iNávody** o interaktivní testy znalostí,
- rozšíření vlastního uploaderu o možnosti nastavení maximální velikosti obrázku, generování názvu souboru podle vstupního názvu a další parametry nahrávání,
- rozšíření externích skriptů o možnost automatického **zasílání notifikačních e-mailů** v případě chyby nebo selhání, ideálně prostřednictvím interního API aplikace,
- vytvoření samostatného rozhraní pro **administraci klíčových oblastí**, odděleného od hlavní části aplikace, včetně vlastního administračního menu a specifických oprávnění.

Je však důležité dodat, že samostatné administrační rozhraní zatím nebylo implementováno záměrně. Aktuálně není rozsah administračních funkcí natolik rozsáhlý, aby vyžadoval oddělenou sekci, jak je běžné u komplexnějších informačních systémů. Správa jednotlivých oblastí proto probíhá přímo v rámci odpovídajících modulů – například přidávání a úprava návodů je součástí modulu *iNávody*, správa událostí v modulu *iKalendář*, docházková evidence v modulu *iDocházka* a práce s dokumentací v příslušných sekcích.

Tento přístup je v současné fázi systému efektivní a přehledný. Vytvoření samostatné administrační části se bude zvažovat až v případě, kdy se správa systému stane rozsáhlejší a bude vyžadovat centralizovanější přístup.

Ukázka administrace kapitol v rámci modulu *iNávody* je uvedena na obr. 22.



Obrázek 22: Ukázka administračního rozhraní pro správu kapitol v modulu *iNávody*

Zásadní oblastí pro další vývoj bude také optimalizace práce s narůstajícím množstvím dat. Již nyní eviduje systém desítky interních úkolů a další záznamy budou postupně přibývat. To klade důraz na efektivní předávání dat mezi serverem a klientem, například pomocí stránkování, filtrování nebo asynchronního načítání dat (AJAX). Uživatelé již nyní využívají moderní komponenty s možností dynamického řazení, hledání nebo filtrování údajů v uživatelských seznamech – tyto principy budou dále rozšiřovány i na další části systému podle aktuálních potřeb.

Konečné řešení pro práci s velkými datovými objemy bude záviset na dalším vývoji systému a zpětné vazbě od uživatelů. Architektura aplikace je však nastavena tak, aby bylo možné flexibilně reagovat a případně nasadit pokročilé techniky jako je lazy loading, server-side filtering nebo stránkování pomocí REST API.

Tyto návrhy reflektují nejen aktuální potřeby oddělení, ale i dlouhodobé trendy v oblasti vývoje informačních systémů. Díky otevřené architektuře a modulárnímu řešení lze nové funkcionality doplňovat postupně a cíleně, bez nutnosti zásahů do stávající infrastruktury.

5 Zpětná vazba od uživatelů a testování

V průběhu vývoje informačního systému *Agenda* byl kladen důraz nejen na technickou stránku implementace, ale také na praktické využití systému v rámci reálného provozu oddělení. Aplikace byla po dokončení základní funkcionality

nasazena v testovacím režimu v interním prostředí organizace, kde ji měli možnost otestovat zaměstnanci oddělení správy a vývoje důchodových agend.

Získaná zpětná vazba od kolegů sehrála klíčovou roli při dolaďování detailů a vylepšení použitelnosti aplikace. Na základě jejich připomínek byly upraveny následující oblasti:

- **Zobrazování titulů uživatelů** – na základě podnětů byly do systému doplněny akademické tituly, které se nyní zobrazují u uživatelů v seznamu i v detailu,
- **Řazení dat** – v administraci uživatelů bylo doplněno vícekriteriální řazení podle abecedy, rolí nebo počátečního znaku příjmení, což zlepšilo orientaci v seznamu,
- **Funkcionalita modulu iDocházka** – byla upravena logika načítání dat a jejich zobrazení v kalendáři podle aktuálních potřeb týmu.

Testovací nasazení zároveň umožnilo odhalit drobné technické problémy, například konflikty při validaci formulářů nebo špatně formulované chybové hlášky, které byly následně opraveny. Praktická spolupráce se skutečnými uživateli tak významně přispěla k vyšší kvalitě systému a jeho lepšímu přijetí v budoucím ostrém provozu.

Závěr

Tato bakalářská práce se zabývala návrhem a implementací informačního systému *Agenda*, který je určen pro interní potřeby oddělení správy a vývoje APV důchodových agend. Cílem bylo vytvořit centralizovaný systém, jenž by nahradil roztříštěné a neefektivní systémy využívané na oddělení. V rámci práce byla provedena analýza současného stavu, návrh vícevrstvé architektury a následná implementace webové aplikace.

Systém byl vytvořen s využitím moderních technologií, konkrétně ASP.NET Core, Entity Framework Core a databázového systému SQLite. Klientská část byla realizována pomocí HTML, CSS, JavaScriptu a AJAXu s důrazem na jednoduché, přehledné a responzivní uživatelské rozhraní. Aplikace obsahuje několik funkčních modulů, mimo jiné pro správu interních úkolů, dokumentace, plánování, docházky či administrace uživatelů.

Práce zároveň reflektuje právní rámec vývoje IS ve veřejné správě a principy 3E (efektivnost, ekonomičnost, účelnost), které byly při návrhu systému systematicky uplatněny. Projekt byl otestován v rámci interního prostředí, přičemž zpětná vazba od uživatelů vedla k dalšímu vylepšení použitelnosti a funkcionalit.

Výsledkem je funkční aplikace informačního systému, který má potenciál zefektivnit každodenní činnost zaměstnanců, zlepšit organizaci interních procesů a poskytnout udržitelný základ pro další rozvoj.

Conclusions

This bachelor's thesis focused on the design and implementation of the information system named Agenda, developed for the internal needs of the department responsible for the administration and development of pension agenda software. The aim was to create a centralized tool that replaces fragmented and inefficient systems currently used within the department. The thesis included an analysis of the current state, the design of multi-layered architecture followed by the implementation of the web application.

The system was developed using modern technologies, specifically ASP.NET Core, Entity Framework Core, and SQLite. The frontend was implemented using HTML, CSS, JavaScript, and AJAX, with an emphasis on a simple, clear and responsive user interface. The application includes several functional modules, including task management, documentation, planning, attendance, and user administration.

The thesis also addresses the legal framework for information systems in public administration and integrates the principles of 3E (effectiveness, economy, and efficiency), which were systematically applied in the system's design. The system was tested in an internal environment, and feedback from users led to further improvements in usability and functionality.

The outcome is a functional prototype of information system that has the potential to streamline the daily operations of employees, improve the organization of internal processes, and provide a sustainable foundation for future development.

A Obsah elektronických dat

Veškerá elektronická data k této bakalářské práci byla odevzdána prostřednictvím individuálního adresáře vytvořeného na cloudovém úložišti Katedry informatiky Univerzity Palackého v Olomouci, provozovaném na platformě NextCloud.

Struktura adresáře je v souladu s oficiálními pokyny katedry a obsahuje následující položky:

- **bin/** – Složka obsahuje archiv se zkompilovanou verzí webové aplikace připravenou pro spuštění.
- **doc/** – Tato složka obsahuje finální verzi textu bakalářské práce ve formátu PDF a archiv se zdrojovými soubory LaTeX dokumentu.
- **install/** – Obsahuje potřebné instalační soubory pro zprovoznění aplikace na čistém počítači.
- **src/** – Obsahuje kompletní zdrojové kódy webové aplikace v archivu ZIP, včetně všech projektových, konfiguračních a podpůrných souborů.
- **readme.txt** – Textový soubor s pokyny pro spuštění webové aplikace, popisující základní kroky nutné pro instalaci a provoz systému v testovacím prostředí.
- **AgendaF.xml** – XML dokumentace vygenerována Visual Studiem 2022 při kompilaci projektu

Seznam zkratek

AI Artificial Intelligence

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

ASP.NET Active Server Pages .NET

COBOL Common Business-Oriented Language

CRUD Create, Read, Update, Delete

CSS Cascading Style Sheets

DBaaS Database-as-a-Service

DOM Document Object Model

EF Core Entity Framework Core

GDPR General Data Protection Regulation

HTML HyperText Markup Language

Identity ASP.NET Identity

IS Informační systém

LINQ Language Integrated Query

MF ČR Ministerstvo financí České republiky

MPSV Ministerstvo práce a sociálních věcí

MVC Model–View–Controller

ORM Object-Relational Mapping

REST Representational State Transfer

SQL Structured Query Language

SQLite Structured Query Language – Lightweight Embedded

WPF Windows Presentation Foundation

WYSIWYG What You See Is What You Get

Literatura

[1] Kopecký, Martin. *Informační systémy ve veřejné správě*. 2020. ISBN 978-80-271-0737-6.

[2] Havlíček, Karel; kolektiv. *Moderní řízení ve veřejné správě*. Druhé vyd. 2019. ISBN 978-80-7400-740-2.

[3] Úřad vlády ČR. *Cesta k evropské digitální dekádě: Strategický plán digitalizace Česka do roku 2030*. 2023. Online; přístup 2025-03-28. Dostupný z: ⟨https://digitalnicesko.gov.cz/media/files/Cesta_k_Evropsk%C3%A9_digit%C3%A1ln%C3%AD_dek%C3%A1d%C4%9B_strategick%C3%BD_pl%C3%A1n_digitalizace_%C4%8Ceska_do_roku_2030_2icFk2m.pdf⟩.

[4] Ministerstvo financí ČR. *Povinnost aplikace principu 3E při hospodaření územních samosprávných celků*. 2022. Online; přístup 2025-03-28. Dostupný z: ⟨https://www.mfcr.cz/assets/attachments/2022-09-26_CHJ-MP-23-Povinnost-aplikace-principu-3E.pdf⟩.

[5] Evropská unie. *Nařízení Evropského parlamentu a Rady (EU) 2016/679 (GDPR) – obecné nařízení o ochraně osobních údajů*. 2016. Online; přístup 2025-03-28. Dostupný také z: ⟨https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=CELEX%3A32016R0679⟩.

[6] Parlament České republiky. *Zákon č. 181/2014 Sb., o kybernetické bezpečnosti*. 2014. Online; přístup 2025-03-28; ve znění pozdějších předpisů. Dostupný také z: ⟨https://www.zakonyprolidi.cz/cs/2014-181⟩.

[7] Parlament České republiky. *Zákon č. 365/2000 Sb., o informačních systémech veřejné správy*. 2000. Online; přístup 2025-03-28; ve znění pozdějších předpisů. Dostupný také z: ⟨https://www.zakonyprolidi.cz/cs/2000-365⟩.

[8] Ministerstvo financí ČR. *Metodický pokyn CHJ č. 3 – Metodika veřejného nakupování: Naplňování principů 3E v praxi veřejného zadávání*. 2024. Online; přístup 2025-03-28; verze 2.0 ze dne 2.12.2024. Dostupný z: ⟨https://www.mfcr.cz/cs/kontrola-a-regulace/rizeni-a-kontrola-verejnych-financi/metodicka-podpora-chj/2016/metodicky-pokyn-chj-c-3--metodika-verejn-25582⟩.

[9] Elmasri, Ramez; Navathe, Shamkant B. *Fundamentals of Database Systems*. 7. vyd. 2015. ISBN 9780133970777.

[10] Han, Jing; E, Haihong; Le, Guan; Du, Jian. A survey of NoSQL databases. *Proceedings of the 6th International Conference on Pervasive Computing and Applications*. 2011, s. 363–366.

[11] Garcia-Molina, Hector; Salem, Kenneth aj. The future of databases. *Communications of the ACM*. 2011, roč. 54, č. 6, s. 44–53.

[12] Corbett, James C.; Dean, Jeffrey; Epstein, Michael aj. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*. 2013, roč. 31, č. 3, s. 1–22.

[13] Rath, Srikanta Patnaik; Mahapatra, Durga Prasad. Cloud Database Management Systems: A Survey. *International Journal of Cloud Applications and Computing (IJCAC)*. 2017, roč. 7, č. 3, s. 1–19.

[14] Owen, Mike. *The Definitive Guide to SQLite*. 2010.

[15] Esposito, Dino. *Modern Web Development with ASP.NET Core 3*. 2020. ISBN 978-0135752073.

[16] Galloway, Jon aj. *Professional ASP.NET Core 2*. 2018. ISBN 978-1119449270.

[17] Esposito, Dino. *Modern Web Development with ASP.NET Core 8*. 2024. ISBN 9780138200051.

[18] Documentation, SQLite. *SQLite: Features and Usage*. 2024. Online; přístup 2024-02-10. Dostupný z: ⟨`https://www.sqlite.org/docs.html`⟩.

[19] Duckett, Jon. *HTML5 a CSS3 pro webové designéry*. 2014. ISBN 978-80-251-4295-9.

[20] Powell, Thomas A. *JavaScript a Ajax: Moderní programování webových aplikací*. 2011. ISBN 978-80-251-3362-9.

[21] Documentation, Bootstrap. *Bootstrap CSS Framework*. 2024. Online; přístup 2024-02-10. Dostupný z: ⟨`https://getbootstrap.com/docs/5.3/getting-started/introduction/`⟩.

[22] Duckett, Jon. *HTML and CSS: Design and Build Websites*. 2011. ISBN 978-1118008188.

[23] Freeman, Elisabeth; Freeman, Eric. *Head First HTML and CSS*. 2012. ISBN 978-0596159900.

[24] Flanagan, David. *JavaScript: The Definitive Guide*. 7. vyd. 2020. ISBN 9781491952023.

[25] Mozilla Developer Network. *AJAX a asynchronní JavaScript*. 2024. Online; přístup 2024-02-10. Dostupný z: ⟨`https://developer.mozilla.org/cs/docs/Web/Guide/AJAX`⟩.

[26] jQuery Foundation. *jQuery Documentation*. 2025. Online; přístup 2025-05-12. Dostupný z: ⟨https://api.jquery.com/⟩.

[27] CKSource. *CKEditor 5 Documentation*. 2024. Online; přístup 2025-04-21. Dostupný z: ⟨https://ckeditor.com/docs/ckeditor5/latest/index.html⟩.

[28] Team, SweetAlert2. *SweetAlert2 — A beautiful, responsive, customizable replacement for JavaScript's popup boxes*. 2025. Online; přístup 2025-05-12. Dostupný z: ⟨https://sweetalert2.github.io/⟩.

[29] Microsoft. *Windows PowerShell Documentation*. 2024. Online; přístup 2025-05-12. Dostupný z: ⟨https://learn.microsoft.com/en-us/powershell/⟩.

[30] Taboada, Iria; Daneshpajouh, Amir; Toledo, Nicolás; Vass, Tiago de. Artificial Intelligence Enabled Project Management: A Systematic Literature Review. *Applied Sciences*. 2023, roč. 13, č. 8, s. 5014. Dostupný také z: ⟨https://doi.org/10.3390/app13085014⟩.

[31] Team, FullCalendar. *FullCalendar JavaScript Event Calendar*. 2024. Online; přístup 2025-04-21. Dostupný z: ⟨https://fullcalendar.io/docs⟩.