

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

NOVÝ L.I.S.T.
BAKALÁRSKA PRÁCA

2025
ONDREJ BABINSKÝ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

NOVÝ L.I.S.T.
BAKALÁRSKA PRÁCA

Študijný program: Aplikovaná informatika
Študijný odbor: Informatika
Školiace pracovisko: Katedra aplikovej informatiky
Školiteľ: Mgr. Pavel Petrovič, PhD.

Bratislava, 2025
Ondrej Babinský



ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Ondrej Babinský

Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)

Študijný odbor: informatika

Typ záverečnej práce: bakalárska

Jazyk záverečnej práce: slovenský

Sekundárny jazyk: anglický

Názov: Nový L.I.S.T.
New L.I.S.T.

Anotácia: Na viacerých predmetoch v našom študijnom programe používame systém na zadávanie úloh L.I.S.T., ktorý bol vyvinutý pôvodne ako bakalárska práca, neskôr na ňom autor pokračoval vo svojej diplomovej práci a potom prebiehal ďalší príležitostný vývoj. Hoci systém slúži úspešne viac ako 10 rokov, pomaly dospel do štátia, keď je vhodné vytvoriť nový - ktorý ale využije potenciál tisícov úloh a zadaní, ktoré systém obsahuje.

Ciel: Nový ekvivalent systému L.I.S.T. je nad rámec rozsahu jednej bakalárskej práce, takže sa očakáva, že študent vyvinie sadu modulov, ktoré sa integrujú s inými modulmi vyvinutými ďalšími študentami v iných bakalárskych prácach. Práca bude zahŕňať podrobne zmapovanie existujúcej funkcionality súčasného systému, analýzu požiadaviek nutnej a doplnkovej funkcionality, ktorú treba zachovať a zozbieranie požiadaviek pre nový systém od súčasných používateľov systému L.I.S.T. Na základe tejto analýzy študent navrhne, implementuje a integruje svoj komponent do nového systému L.I.S.T.

Literatúra: Andrej Jursa, Nový dlhodobý viacúčelový sklad úlohy na cvičenia, bakalárska práca, Fakulta matematiky, fyziky a informatiky, 2013.

Andrej Jursa, Realizácia automatického testovania úloh a kontroly plagiátorstva v úlohovom systéme LIST, diplomová práca, Fakulta matematiky, fyziky a informatiky, 2015.

Filip Piták: Systém pre dlhodobú správu zadaní úloh, bakalárska práca, Fakulta matematiky, fyziky a informatiky, 2022.

Full Stack Development with Spring Boot 3 and React - Fourth Edition: Build modern web applications using the power of Java, React, and TypeScript, Packt Publishing, 2023.

Kľúčové

slová: LIST, LMS, automatické testy, programátorské úlohy

Vedúci: Mgr. Pavel Petrovič, PhD.

Katedra: FMFI.KAI - Katedra aplikovanej informatiky

Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 01.03.2024

Dátum schválenia: 31.10.2024

doc. RNDr. Damas Gruska, PhD.




64514349

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie

Čestne vyhlasujem, že celú bakalársku prácu na tému **Nový L.I.S.T.**, vrátane všetkých jej príloh a obrázkov, som vypracoval samostatne, a to s použitím literatúry uvedenej v priloženom zozname a nástrojov umelej inteligencie. Vyhlasujem, že nástroje umelej inteligencie som použil v súlade s príslušnými právnymi predpismi, akademickými právami a slobodami, etickými a morálnymi zásadami za súčasného dodržania akademickej integrity a že ich použitie je v práci vhodným spôsobom označené.

V Babíne, dňa 4.6.2025

.....
Ondrej Babinský

Podakovanie: Ďakujem školiteľovi Mgr. Pavlovi Petrovičovi, PhD. za jeho odborné vedenie, pravidelné konzultácie a cenné rady, ktoré mi výrazne pomohli pri smerovaní tejto práce.

Abstrakt

Cieľom tejto bakalárskej práce je prispieť k návrhu a implementácii nového systému typu LMS (Learning Management System), ktorý má slúžiť ako moderná náhrada za dlhodobo používaný, technologicky zastaraný systém L.I.S.T., ktorý je využívaný pri výučbe na viacerých predmetoch Fakulty matematiky, fyziky a informatiky UK. Nový systém pokrýva celý cyklus práce s úlohami - od ich zadania, cez odovzdávanie riešení a hodnotenie, až po správu používateľov, kurzov a rôznych výstupov pre učiteľov aj študentov.

Systém je navrhnutý ako webová aplikácia s oddeleným frontendom a backendom. Backend je postavený na technológii ASP.NET Core a PostgreSQL, pričom využíva REST API, modulárnu architektúru a bezpečnostné mechanizmy ako JWT autentifikáciu. Frontend je vytvorený v Reacte a TypeScripte s použitím Material UI a komponentového návrhu.

V rámci tímovej spolupráce som sa sústredil najmä na implementáciu systému prihlásenia a používateľov, evidencie kurzov a úloh, ako aj na návrh a tvorbu používateľského rozhrania pre študentov. Výsledkom práce je funkčná, rozšíriteľná a udržiavateľná platforma, ktorá tvorí súčasť pripravovaného nového systému a je vhodná na nasadenie do výučby.

Kľúčové slová: LIST, LMS, automatické testy, programátorské úlohy

Abstract

The goal of this bachelor thesis is to contribute to the design and implementation of a new Learning Management System (LMS) intended as a modern replacement for the long-used but technologically outdated L.I.S.T. system. This system has been utilized in the teaching of several courses at the Faculty of Mathematics, Physics and Informatics at Comenius University. The new solution aims to cover the entire workflow related to assignments - from their creation and submission to evaluation and the management of users, courses, and outputs for both teachers and students.

The system is designed as a web application with a separate frontend and backend. The backend is built using ASP.NET Core and PostgreSQL, following a modular architecture and providing a secure REST API using JWT authentication. The frontend is implemented in React and TypeScript with Material UI and a component-based design.

As part of a collaborative development team, my work focused primarily on implementing the authentication and user management system, managing courses and assignments, and designing and developing the student-facing user interface. The resulting platform is a functional, extensible, and maintainable part of the new system, ready to be deployed in real teaching environments.

Keywords: LIST, LMS, automatic tests, programming tasks

Obsah

| | |
|---|-----------|
| Úvod | 1 |
| 1 Východiská práce | 3 |
| 1.1 Learning Management Systems (LMS) | 3 |
| 1.1.1 Charakteristika a účel LMS | 3 |
| 1.1.2 Generické funkcionality LMS | 4 |
| 1.1.3 Architektúry LMS systémov | 6 |
| 1.2 Systém L.I.S.T. | 7 |
| 1.2.1 História a účel systému | 8 |
| 1.2.2 Používateľské roly a oprávnenia | 10 |
| 1.2.3 Používateľské rozhranie | 10 |
| 1.3 Ďalšie existujúce LMS systémy | 15 |
| 1.3.1 Moodle | 15 |
| 1.3.2 Google Classroom | 16 |
| 1.4 Technológie | 18 |
| 1.4.1 Backendové technológie | 18 |
| 1.4.2 Frontendové technológie | 20 |
| 1.4.3 Autentifikácia a zabezpečenie pomocou JWT | 21 |
| 2 Návrh | 22 |
| 2.1 Definícia dôležitých pojmov | 22 |
| 2.2 Požiadavky | 23 |
| 2.2.1 Používatelia | 23 |
| 2.2.2 Kurzy | 24 |
| 2.2.3 Úlohy | 25 |
| 2.2.4 Študentské používateľské rozhranie | 25 |
| 2.3 Technológie a ich prepojenia | 26 |
| 2.3.1 Backend | 26 |
| 2.3.2 Frontend | 26 |
| 2.4 Bezpečnosť a autentifikácia | 27 |
| 2.4.1 Tokenová autentifikácia (JWT) | 27 |

| | | |
|----------|---|-----------|
| 2.4.2 | Autorizácia a ochrana endpointov | 28 |
| 2.4.3 | Hashovanie hesiel pomocou BCrypt | 29 |
| 2.4.4 | Ochrana pred útokmi - Rate Limiting | 29 |
| 2.4.5 | Záznam prístupov a auditovanie | 30 |
| 2.5 | Modulárna architektúra systému | 30 |
| 2.5.1 | Rozdelenie systému na moduly | 30 |
| 2.5.2 | Komunikácia medzi modulmi | 30 |
| 2.6 | Opis implementovaných častí systému | 31 |
| 2.6.1 | Modul Users | 31 |
| 2.6.2 | Roly | 32 |
| 2.6.3 | Modul Courses | 34 |
| 2.6.4 | Modul Tasks | 35 |
| 2.6.5 | Používateľské rozhranie | 35 |
| 3 | Implementácia | 42 |
| 3.1 | Zvolený implementačný prístup | 42 |
| 3.2 | Štruktúra projektu | 42 |
| 3.2.1 | Frontend | 43 |
| 3.2.2 | Backend | 44 |
| 3.3 | Používatelia a roly | 44 |
| 3.3.1 | Prehľad modulu | 44 |
| 3.3.2 | Dátové modely | 45 |
| 3.3.3 | Databázový kontext | 47 |
| 3.3.4 | API endpointy | 48 |
| 3.3.5 | Správa používateľov - učiteľské rozhranie | 50 |
| 3.4 | Kurzy (Courses) | 53 |
| 3.4.1 | Prehľad modulu | 53 |
| 3.4.2 | Dátové modely | 54 |
| 3.4.3 | Databázový kontext | 56 |
| 3.4.4 | API endpointy | 57 |
| 3.4.5 | Učiteľské rozhranie pre správu kurzov | 59 |
| 3.4.6 | Komunikácia frontendu s backendom | 63 |
| 3.5 | Úlohy (Tasks) | 64 |
| 3.5.1 | Prehľad modulu | 64 |
| 3.5.2 | Dátové modely | 65 |
| 3.5.3 | Databázový kontext | 66 |
| 3.5.4 | API endpointy | 67 |
| 3.5.5 | Učiteľské rozhranie pre správu úloh | 69 |
| 3.5.6 | Komunikácia frontendu s backendom | 72 |

| | |
|--|-----------|
| 3.6 Študentské používateľské rozhranie | 74 |
| Záver | 80 |

Zoznam obrázkov

| | | |
|------|--|----|
| 1.1 | Schéma komunikácie cez API Gateway | 7 |
| 1.2 | REST API komunikácia | 8 |
| 1.3 | Používateľské rozhranie systému LaMSfET | 9 |
| 1.4 | Študentské zobrazenie úloh a hodnotení | 11 |
| 1.5 | Zostavy úloh v pôvodnom systéme | 12 |
| 1.6 | Kategórie úloh v pôvodnom systéme | 13 |
| 1.7 | Zobrazenie komentára v systéme L.I.S.T. | 14 |
| 1.8 | Google Classroom - správa zadania | 18 |
| 1.9 | Swagger - rozhranie API dokumentácie | 20 |
| 2.1 | Sekvenčný diagram autentifikácie pomocou JWT | 28 |
| 2.2 | Diagram rozhodovania pri autorizácii endpointu | 29 |
| 2.3 | UML diagram modulu Users | 32 |
| 2.4 | Používateľské roly a ich oprávnenia | 33 |
| 2.5 | Diagram tried modulu Courses | 34 |
| 2.6 | UML diagram modulu Tasks | 35 |
| 2.7 | Prihlasovacia obrazovka systému | 36 |
| 2.8 | Prehľad kurzov po prihlásení | 37 |
| 2.9 | Detail kurzu - Popis | 38 |
| 2.10 | Zoznam zadania kurzu | 39 |
| 2.11 | Detail zadania s úlohami a možnosťou odovzdania | 40 |
| 2.12 | Prehľad bodov a termínov v kurze | 41 |
| 3.1 | Štruktúra priečinkov frontend a backend aplikácie | 43 |
| 3.2 | Zoznam API endpointov pre modul Users | 48 |
| 3.3 | Používateľské rozhranie pre učiteľa - tabuľka s akciami. | 51 |
| 3.4 | Dialógové okno na úpravu používateľa - meno, e-mail, rola. | 52 |
| 3.5 | Zoznam API endpointov pre kurzy | 58 |
| 3.6 | Prehľad kurzov v učitelskom rozhraní | 60 |
| 3.7 | Editor popisu kurzu v TinyMCE | 61 |
| 3.8 | Náhľad popisu kurzu | 62 |

| | | |
|------|---|----|
| 3.9 | Správa účastníkov kurzu | 63 |
| 3.10 | Zoznam API endpointov pre modul Tasks | 68 |
| 3.11 | Prehľad úloh v učiteľskom rozhraní | 70 |
| 3.12 | Možnosti filtrovania úloh | 71 |
| 3.13 | Editor úlohy v učiteľskom rozhraní | 72 |
| 3.14 | Rozhranie s rozdelením na tvoje a ostatné kurzy | 75 |
| 3.15 | Popis kurzu a navigačný sidebar | 76 |
| 3.16 | Zoznam zadani podľa kategórie a stavu | 77 |
| 3.17 | Grafické zobrazenie progresu a najbližších termínov | 78 |
| 3.18 | Zobrazenie úlohy a možnosť odovzdania riešenia | 79 |

Úvod

Na Fakulte matematiky, fyziky a informatiky Univerzity Komenského je súčasťou viacerých predmetov aktívna práca na programátorských úlohách. Predmety ako Programovanie 1 a 2, Tvorba informačných systémov, Operačné systémy, Programovanie v jazyku Java a mnohé ďalšie si vyžadujú efektívny spôsob správy úloh, komunikácie medzi študentmi a vyučujúcimi, priebežného hodnotenia a sledovania pokroku. V prostredí narastajúceho dôrazu na spätnú väzbu, dostupnosť informácií a automatizované vyhodnocovanie výsledkov je pre vyučovanie nevyhnutné mať nástroj, ktorý tieto požiadavky splňa.

V reakcii na túto potrebu vznikol pôvodný systém s názvom **lamfset**, ktorý však časom zastaral nielen po technologickej stránke, ale aj z pohľadu používateľskej skúsenosti. Na jeho miesto nastúpil systém **L.I.S.T.**, ktorý bol vytvorený v rámci bakalárskej práce Andreja Jursu v roku 2013 [8] ako moderná náhrada za pôvodný zastaraný systém. Tento nový úlohopraktický systém výrazne zjednodušíl prácu s kurzami a zadaniami, čím sa stal prirodzenou súčasťou výučby na viacerých predmetoch na fakulte. V roku 2015 [9] bol systém ďalej rozšírený v jeho diplomovej práci, ktorá priniesla podporu automatického testovania úloh a detekcie plagiátorstva, čím ešte viac zvýšila praktickú hodnotu systému pre každodenné použitie vo výučbe.

Aj napriek tomu, že sa L.I.S.T. stále aktívne používa, postupom času sa jeho limity stali čoraz zreteľnejšie. Pridávanie novej funkcionality sa ukázalo ako zdĺhavé a neefektívne, keďže systém nie je pripravený na jednoduchú rozšíriteľnosť. Mnohé požiadavky, ktoré dnes považujeme za štandard - ako napríklad flexibilné prístupové oprávnenia, moderné používateľské rozhranie či bezpečné a škálovateľné API - nie je možné do systému jednoducho doplniť bez zásadného prepisu.

Preto sa čoraz viac ukazuje potreba vytvoriť nový systém od základov, postavený na moderných technológiách, s dôrazom na čistú architektúru a dobrú udržiavateľnosť. Ako pri obnove starého mosta, ktorý roky slúžil, no už nedokáže uniesť nápor modernej dopravy, aj v prípade L.I.S.T.-u nastal čas postaviť nové základy - pevnejšie, flexibilnejšie a pripravené na budúcnosť.

Takýto systém musí byť nielen schopný nahradíť doterajšie riešenie, ale zároveň vytvárať priestor pre jeho ďalší vývoj - tak, aby mohol reagovať na nové potreby predmetov a vyučujúcich. Nové komponenty systému L.I.S.T., ktorých vývojom táto práca

prispieva, by mali byť pripravené na reálne nasadenie už v nasledujúcom akademickom semestri, poskytovať základnú funkcionality správy kurzov, úloh a používateľov a vytvárať stabilné základy pre budúce rozširovanie.

Cieľom tejto bakalárskej práce je preto prispieť k návrhu a implementácii nového systému, ktorý tieto požiadavky splňa, a realizovať vybrané časti jeho funkcionality. Dôraz kladiem na bezpečnú a prehľadnú správu používateľov a roľí, efektívny manažment kurzov a úloh, ako aj vytvorenie prehľadného študentského rozhrania, ktoré zohľadňuje používateľské potreby. Systém je navrhnutý tak, aby bol technologicky aktuálny, ľahko rozšíriteľný a pripravený na dlhodobé používanie.

Táto práca preto postupne predstavuje východiská a analýzu existujúcich riešení, návrh architektúry a dátových štruktúr, ako aj samotnú implementáciu jednotlivých častí systému, s dôrazom na tie, na ktorých som sa priamo podieľal.

Kapitola 1

Východiská práce

Táto kapitola sa venuje teoretickým a technickým základom, ktoré súvisia s návrhom systému na správu programátorských úloh. Najskôr predstavuje koncept Learning Management Systems (LMS) a ich architektúry, následne analyzuje existujúce systémy podobného typu, doteraz na FMFI UK využívaný systém L.I.S.T., a napokon sa zameriava na prehľad technológií vhodných pre vývoj nového riešenia.

1.1 Learning Management Systems (LMS)

1.1.1 Charakteristika a účel LMS

Definícia LMS

Learning Management System (LMS) je softvérová platforma určená na komplexnú správu vzdelávacích aktivít. V základnej forme zabezpečuje automatizáciu administratívnych procesov, sledovanie pokroku používateľov a generovanie prehľadov o výkone [3].

Primárny účel LMS

Účelom LMS je umožniť organizáciám efektívne riadiť vzdelávanie - od vytvárania kurzov až po hodnotenie a spätno-väzobné vyhodnocovanie. LMS systémy zjednodušujú celý cyklus výučby, čím znižujú administratívnu záťaž a zvyšujú dostupnosť vzdelávacích materiálov.

Oblasti využitia

LMS systémy nachádzajú uplatnenie v širokom spektri oblastí, pričom ich využitie nie je len v školskom prostredí. Vo vzdelávaní slúžia ako nástroj na organizáciu výučby, zadávanie úloh, správu študijných materiálov a hodnotenie študentov. Sú súčasťou

každodenného fungovania základných, stredných aj vysokých škôl, kde zefektívňujú komunikáciu medzi vyučujúcimi a študentmi.

Mimo školstva sa LMS systémy používajú vo firemných a inštitucionálnych prostrediach. Firmy ich využívajú na interné školenia, onboarding nových zamestnancov, periodické preškoľovanie či zabezpečenie odborných certifikácií.

LMS nachádzajú uplatnenie aj v neziskových organizáciách, štátnej správe, združenictve či vo vojenskom sektore, kde sa používajú na školenia v oblasti legislatívy, bezpečnosti, etiky či technických zručností. Všade tam, kde je potrebné zabezpečiť konzistentné a sledovateľné vzdelávanie vo veľkom meradle, ponúkajú LMS efektívne riešenie.[13].

Výhody použitia LMS

Jednou z hlavných výhod je centralizácia vzdelávacích aktivít, vďaka ktorej je možné efektívne riadiť kurzy, študijné materiály aj používateľov na jednom mieste.

LMS systémy pomáhajú organizáciám jednoducho aktualizovať obsah, sledovať pokrok jednotlivcov a zabezpečiť konzistentnosť školení.

Ďalšou výhodou je možnosť rýchleho nasadenia a škálovania systému podľa potrieb konkrétnej organizácie - LMS môže efektívne fungovať v malých tímcach aj vo veľkých inštitúciách s tisícami používateľov. V praxi to potvrdzuje napríklad použitie LMS pri výučbe anglického jazyka na Università degli Studi di Milano, kde systém spracoval viac než 2000 online testov pre viac ako 800 študentov. LMS v tomto prípade výrazne zjednodušil administráciu, poskytol spätnú väzbu a podporil aj interaktivitu a zapojenie študentov, čím preukázal svoju efektivitu pri práci s veľkými skupinami [6].

1.1.2 Generické funkcionality LMS

LMS systémy sa skladajú z množstva funkcionalít, ktoré spoločne umožňujú správu vzdelávania. Medzi najdôležitejšie patrí správa kurzov a používateľov, zadávanie a odovzdávanie úloh, testovanie a hodnotenie, komunikácia medzi účastníkmi, analytika a sledovanie pokroku, ako aj integrácia s inými systémami.[3] Tieto funkcie zabezpečujú, že LMS môže plnohodnotne podporovať proces vzdelávania.

Správa kurzov a obsahu

Jednou z dôležitých funkcionalít LMS je možnosť vytvárať a organizovať kurzy vrátane pridávania vzdelávacích materiálov, ako sú texty, videá či prezentácie. Kurzy sú zvyčajne rozdelené do tematických celkov, čo uľahčuje orientáciu v obsahu a zlepšuje prístupnosť pre študentov. LMS navyše umožňuje opäťovné využitie materiálov a ich jednoduchú aktualizáciu bez zásahu do celej štruktúry kurzu.

Správa používateľov a oprávnení

LMS ponúka nástroje na registráciu a správu používateľov vrátane ich priraďovania do kurzov či skupín. Každý používateľ má pridelenú rolu (napríklad študent, vyučujúci, asistent alebo administrátor), ktorá definuje, akú úroveň prístupu a aké funkcionality bude mať používateľ v systéme. Správa oprávnení umožňuje nastavovať, čo používateelia s jednotlivými rolami môžu alebo nemôžu vykonávať, čím sa zabezpečuje ochrana citlivých údajov.

Zadávanie a odovzdávanie úloh

Systémy LMS umožňujú vyučujúcim zadávať úlohy s presnými požiadavkami, termínmi odovzdania a hodnotiacimi kritériami. Študenti môžu svoje riešenia odovzdávať priamo cez systém vo forme súborov alebo textových vstupov. Systém zároveň zabezpečuje centralizovanú evidenciu odovzdaných úloh.

Testovanie a hodnotenie

Testy, kvízy a iné spôsoby hodnotenia patria medzi základné súčasti moderných vzdelávacích systémov. Softvér LMS umožňuje ich jednoduchú tvorbu, pričom hodnotenie môže prebiehať automaticky alebo manuálne podľa typu úlohy. Študenti dostávajú okamžitú spätnú väzbu a výsledky sa uchovávajú pre ďalšiu analýzu a sledovanie pokroku.

Komunikačné nástroje

Komunikácia medzi vyučujúcimi a študentmi je kľúčová pre úspešný priebeh kurzov. Systémy preto často obsahujú diskusné fóra, chaty a možnosti rôznych notifikácií, ako sú e-mailové upozornenia. Tieto nástroje podporujú interaktivitu, umožňujú pokladať otázky a riešiť problémy bez potreby použitia inej platformy.

Analytika a reportovanie

Súčasťou LMS systémov sú aj analytické nástroje, ktoré spracovávajú údaje o aktivitách používateľov. Učitelia a administrátori majú k dispozícii štatistiky o účasti, pokroku a výsledkoch študentov. Zároveň aj študenti môžu sledovať svoj vlastný pokrok, porovnávať výsledky a získať spätnú väzbu o tom, ako sa im darí. Tieto informácie pomáhajú učiteľom zistiť, aké sú problémové oblasti, a študentom udržiavať prehľad o vlastnom napredovaní.

Integrácie a štandardy

Moderné LMS systémy podporujú rôzne integračné možnosti, ktoré zvyšujú ich flexibilitu a použiteľnosť. Bežne sa prepájajú s HR systémami, externými databázami či nástrojmi na správu identít. V prípade programátorských a akademických predmetov je čoraz dôležitejšia aj integrácia s antiplagiátorskými systémami, ktoré umožňujú kontrolovať originalitu odovzdaných úloh.

1.1.3 Architektúry LMS systémov

Architektúra LMS systému predstavuje spôsob, akým je systém navrhnutý a ako sú jeho jednotlivé komponenty prepojené. Typ architektúry má veľký vplyv na škálovateľnosť, bezpečnosť, údržbu aj flexibilitu systému pri pridávaní nových funkcií. Moderné LMS systémy preto vznikajú na takých architektúrach, ktoré dokážu spoľahlivo obsluhovať veľa používateľov a zároveň umožňujú jednoduché rozšírenie inými systémami a novými funkcionalitami.

Monolitická architektúra

Monolitická architektúra predstavuje prístup k vývoju softvérových systémov, pri ktorom všetky časti aplikácie - používateľské rozhranie, logika aplikácie aj databázová vrstva tvoria jeden celok. V kontexte LMS systémov to znamená, že správa kurzov, úloh, používateľov či analytika sú súčasťou jedného veľkého systému. Výhodou monolitickej architektúry je jednoduchšia implementácia, testovanie a nasadzovanie, čo z nej robí vhodné riešenie pre menšie projekty alebo systémy s menej komplexnými požiadavkami. Nevýhodou je hlavne slabšia flexibilita pri rozširovaní a problémy so škálovaním, keďže akákoľvek zmena v jednej časti aplikácie často znamená prerábanie a zmeny v celom systéme.[1].

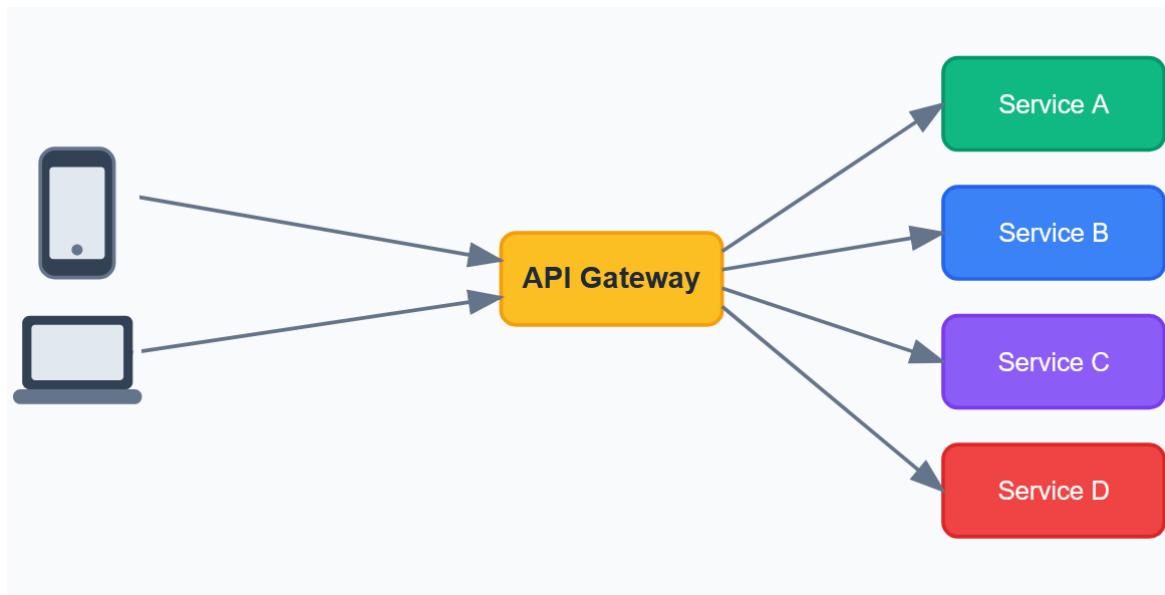
Modulárna architektúra

Modulárna architektúra rozdeľuje systém na samostatné komponenty (moduly), z ktorých každý rieši svoju časť funkcionality. Tento prístup zvyšuje flexibilitu a jednoduchšie rozšírenie systému o nové časti bez zásahu do ostatných modulov. Výhodou je aj lepšia škálovateľnosť, keďže moduly môžu byť nezávisle aktualizované alebo optimalizované. Modulárna architektúra tiež umožňuje opäťovné použitie existujúcich modulov v rôznych aplikáciách [11].

Mikroslužby

Mikroslužbová architektúra je prístup, pri ktorom je celý systém rozdelený na viacero nezávislých služieb. Každá mikroslužba zabezpečuje špecifickú časť funkcionality a s

ostatnými službami komunikuje cez definované rozhrania API (aplikačné programové rozhranie), ktoré umožňujú výmenu dát cez sieť. Tento model umožňuje nasadzovať a škálovať jednotlivé časti systému samostatne, čo zvyšuje flexibilitu a zjednodušíuje údržbu pri veľkých a komplexných aplikáciach. Ďalšou výhodou je možnosť používať rôzne technológie a databázy pre rôzne služby. Na druhej strane, nevýhodou je vyššia celková zložitosť systému - správa siete, komunikácia medzi službami a zabezpečenie konzistencie dát si vyžadujú dôkladné plánovanie [16].



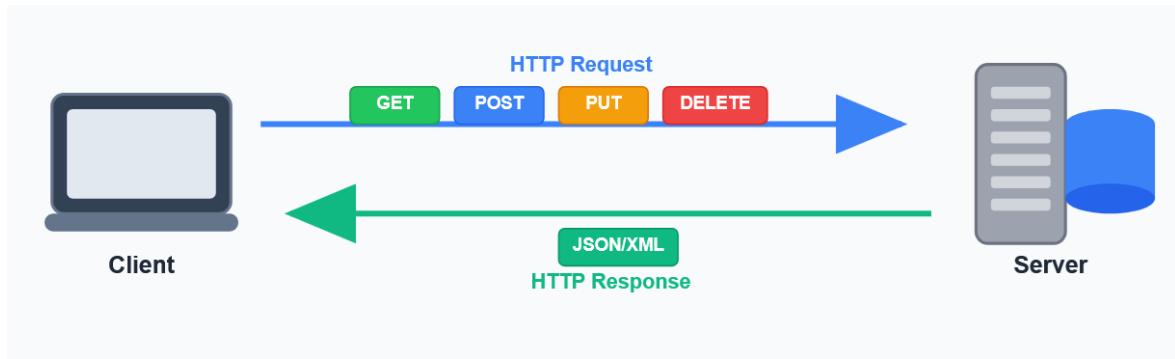
Obr. 1.1: Diagram zobrazuje centrálnu úlohu API Gateway, ktorá sprostredkúva požiadavky od rôznych klientskych zariadení smerom k jednotlivým mikroslužbám v aplikácii.

Komunikačné rozhranie: REST API

Bez ohľadu na použitú architektúru sa LMS systémy často spoliehajú na API (aplikačné programové rozhranie) na komunikáciu medzi vnútornými komponentmi alebo s externými aplikáciami. Najrozšírenejším štýlom na návrh takýchto rozhraní je REST (prenos reprezentácie stavu), ktorý umožňuje jednoduchú a štandardizovanú výmenu dát. Podľa správy Postman State of the API 2023 používa REST až 86% vývojárov, čo z neho robí najbežnejší prístup pri implementácii webových API [17].

1.2 Systém L.I.S.T.

Systém L.I.S.T. predstavuje nástroj na správu študijných úloh, ktorý sa používa pri výučbe programovania na FMFI UK. V tejto kapitole popíšeme jeho vznik, vývoj, súčasné



Obr. 1.2: Ilustrácia REST API komunikácie. Klient odosiela požiadavky pomocou HTTP metód (GET, POST, PUT, DELETE), server na ne odpovedá prostredníctvom HTTP odpovede, často vo formáte JSON alebo XML.

využitie, ako aj jednotlivé časti systému vrátane používateľských rolí a používateľského rozhrania.

1.2.1 História a účel systému

Systém L.I.S.T. (Long-term Internet Storage of Tasks) vznikol ako nástupca staršieho riešenia LaMSfET, ktoré slúžilo na správu a archiváciu cvičných úloh. Pôvodný systém LaMSfET bol vytvorený v roku 2009 v rámci bakalárskej práce a používal sa na doméne fakulty ako základný nástroj na zadávanie úloh vo výučbe programovania. Postupne sa však ukázali jeho technické limity a nízka čitateľnosť kódu, čo viedlo k rozhodnutiu o vývoji nového systému.

The screenshot shows the LaMSfET application interface. At the top, there is a logo and the text "Longterm and Multipurpose Storage for Excercise Tasks". A message box says "otvorená zostava: Para: Úloha 3 (10 úloh)" and "hľadaj úlohu". On the left, there is a sidebar with three main sections: "Úlohy" (Tasks), "Zostavy úloh" (Task compositions), and "Administrácia" (Administration). The "Úlohy" section contains links: "zoznam úloh", "zadať novú úlohu", "import úloh", and "hierarchie tagov". The "Zostavy úloh" section contains "zoznam zostáv" and "vytvoriť novú zostavu". The "Administrácia" section contains "administrácia študentov", "hodnotenie študentov", "administrácia predmetov", "nastavenia systému", "zálohovanie a obnova", and "odhlásiť". The main area is titled "Zoznam úloh" (List of tasks) and displays a table of tasks. The table has columns: #, meno (name), text (text), naposledy (last updated), and akcia (action). There are 9 tasks listed:

| # | meno | text | naposledy | akcia |
|---|-----------------------------|---|-----------|---|
| 1 | Generuj náhodné pole | Napište program, ktorý ako argument dostane prirodzené... | 28.9.2009 | editovať vybrať vymazať |
| 2 | Veľkosť dvojrozmerného poľa | Definujte statickú metódu int pocet(int[][] a), ktorá vráti... | 28.9.2009 | editovať vybrať vymazať |
| 3 | trojuholníkové pole | Definujte statickú metódu int[][] triangle(int N), ktorá pre... | 28.9.2009 | editovať vybrať vymazať |
| 4 | Abecedný zoznam triedy | Dve rovnako dlhé polia String[] meno, a String[] priezvisko... | 28.9.2009 | editovať vybrať vymazať |
| 5 | Polynom | Definujte triedu Polynom, ktorá reprezentuje polynom v... | 4.10.2009 | editovať vybrať vymazať |
| 6 | Dictionary | Definujte triedu Dictionary s konštruktormi... | 4.10.2009 | editovať vybrať vymazať |

Obr. 1.3: Používateľské rozhranie systému LaMSfET

Na základe tejto potreby bol v roku 2013 navrhnutý a implementovaný nový systém s názvom L.I.S.T., ktorý vznikol ako bakalárská práca študenta Andreja Jursu. Jeho cieľom bolo vytvoriť udržiavateľný a rozšíriteľný systém postavený na známych a dobre dokumentovaných open-source technológiách. Tento nový návrh riešil mnohé problémy predchádzajúceho softvéru a zároveň položil základy pre budúci rozvoj.[8]

Následný vývoj systému pokračoval v diplomovej práci, zameranej na rozšírenie možností testovania a hodnotenia študentských riešení. Pôvodný mechanizmus testovania, založený na paralelnom spúštaní, bol nahradený izolovanými kontajnermi, ktoré zaručujú vyššiu bezpečnosť a zabraňujú preťaženiu servera. Zmeny sa dotkli aj hodnotenia - nové riešenie umožňuje bezpečné ukladanie čiastkových výsledkov, presné bodovanie a šifrovanie dát, čím sa predchádza manipulácií zo strany študenta. Dôležitým prínosom bola integrácia nástroja MOSS na porovnanie zdrojového kódu, ktorý pomáha odhalovať plagiáty. Doplnený bol aj vlastný porovnávač Java kódu, čím sa otvorila cesta k ďalším rozšíreniam v oblasti detekcie podobnosti.[9].

V súčasnosti sa systém L.I.S.T. aktívne využíva na Katedre aplikovanej informatiky FMFI UK vo viacerých predmetoch zameraných na programovanie, vývoj softvéru a informatické paradigmy. Ide napríklad o základné kurzy ako Programovanie 1 a Programovanie 2, ako aj pokročilejšie predmety ako Programovanie 4 - Java, Funkcionálne programovanie, Programovacie paradigmy a Tvorba informačných systémov. Systém je taktiež súčasťou výučby ďalších kurzov, ako sú Princípy počítačov - Systémové

programovanie a Princípy počítačov - Operačné systémy. Využíva sa aj v oblastiach zameraných na praktické aplikácie a algoritmy, napríklad v kurzoch Vývoj mobilných aplikácií, Grafy, grafové algoritmy a optimalizácia či Umelá inteligencia.

Systém slúži na správu kurzov, zadávanie úloh, odovzdávanie riešení, automatické testovanie a hodnotenie úloh, čím podporuje efektívnu výučbu a spätnú väzbu medzi študentmi a vyučujúcimi.

1.2.2 Používateľské roly a oprávnenia

Systém L.I.S.T. v súčasnosti rozlišuje dve základné používateľské roly: **študent** a **učiteľ**. Študenti majú prístup k svojim kurzom, môžu si prezerať zadania a odovzdávať riešenia. Učitelia majú plný prístup k správe kurzov, zadávaniu úloh, hodnoteniu, aj k výsledkom študentov. Tento jednoduchý model však spôsobuje určité obmedzenia - napríklad študent, ktorý zároveň pomáha s opravovaním úloh, musí byť nastavený ako učiteľ. V dôsledku toho získa prístup aj ku kurzom, v ktorých je sám študentom, čo je nežiadúce z pohľadu bezpečnosti a prehľadnosti.

Systém tak momentálne nerozlišuje jemnejšie role alebo delegovanie práv, čo komplikuje prácu vo väčších kurzoch alebo pri spolupráci viacerých vyučujúcich. Tento nedostatok poukazuje na potrebu flexibilnejšieho modelu oprávnení, ktorý by umožnil lepšie prispôsobiť práva jednotlivým používateľom podľa ich skutočných úloh v systéme.

1.2.3 Používateľské rozhranie

Jednou z slabín pôvodného systému L.I.S.T. je jeho používateľské rozhranie. Používatelia (najmä študenti a učitelia) sa často stretávajú s komplikovaným ovládaním a vizuálne neatraktívnym prostredím, čo znižuje celkový používateľský zážitok a zvyšuje riziko chýb pri práci so systémom. Hoci je rozhranie technicky responzívne a dá sa používať aj na mobilných zariadeniach, jeho praktická použiteľnosť je obmedzená - niektoré dialógy sú príliš rozsiahle, čo spôsobuje, že na menších obrazovkách musí používateľ manuálne posúvať obsah v oboch smeroch, pretože prvky sa nezobrazujú v prehľadnej vertikálnej štruktúre, ale zostávajú rozmiestnené horizontálne, teda znižuje flexibilitu prístupu z menších obrazoviek.

Na Obr. 1.4 je vidieť náhľad rozhrania z pohľadu študenta, ktorý si prehliada zoznam úloh a výsledky. Farebná schéma, kontrasty aj typografia nezodpovedajú súčasným štandardom UI/UX dizajnu.

| Navigácia | | Úlohy a hodnotenie | | | | |
|---|---------------------|--|----------------------------------|----------------------------------|---------------------|--------------------------|
| Zapisať sa na kurz | | Tvorba informačných systémov / Zimný semester 2024/2025 | | | | |
| Zmeniť svoju skupinu | | <input type="checkbox"/> Zobrazit všetky úlohy. | | | | |
| Popis kurzu | | Názov | Limit odosielania riešení | Body | | |
| Obsah kurzu | | Dochádzka | | | | |
| Úlohy a hodnotenie | | Získal(a) si 2 bodov z 3. | | | | |
| Projekty | | Kvíz - raw | | | | |
| Body za úlohy | | | | | | |
| Typ úlohy | Body | | | | | |
| Dochádzka | 2 / 3 (min 2) | | | | | |
| Kvíz | 28,25 / 35 (min 20) | Neodovzdáva sa. | 32.7 / 34 | nahradny 13.1. Meno Profesora | | |
| Kvíz - raw | 226,4 / 280,5 | Neodovzdáva sa. | - / 25 | Zatial' nehodnotené. | | |
| Midterm | 14 / 15 (min 10) | Neodovzdáva sa. | 36,1 / 37 | Meno Profesora | | |
| Quiz - raw | 0 / 0 | Neodovzdáva sa. | 18 / 22 | Meno Profesora | | |
| Účasť | 3 / 3 (min 2) | Neodovzdáva sa. | 20,7 / 22,5 | Meno Profesora | | |
| Spolu | 42,25 / 50 | Neodovzdáva sa. | 20,5 / 27 | Meno Profesora | | |
| Body za projekty | | | | | | |
| Projekt | Body | | | | | |
| Spolu | 0 / 0 | Neodovzdáva sa. | 20,5 / 25 | Meno Profesora | | |
| Legenda farieb | | | | | | |
| Body sa započítavajú do celkového súčtu bodov | | Neodovzdáva sa. | 21 / 22 | Meno Profesora | | |
| Body sa nezapočítavajú do celkového súčtu bodov | | Neodovzdáva sa. | 14,9 / 15 | Meno Profesora | | |
| Získal(a) si 226,4 bodov z 280,5. | | | | | | |
| Midterm | | | | | | |
| UML | | Bez obmedzenia na odovzdanie riešenia. | 14 / 15 | Meno Profesora | | |
| UML diagrams (cvičný midterm) | | Bez obmedzenia na odovzdanie riešenia. | - / 0 | Zatial' nehodnotené. | | |
| Získal(a) si 14 bodov z 15. | | | | | | |
| Účasť | | | | | | |
| Pozvané prednášky expertov z praxe | | Neodovzdáva sa. | 3 / 3 | Meno Profesora | | |
| Získal(a) si 3 bodov z 3. | | | | | | |
| Získal(a) si 42,25 bodov z 50 za všetky typy úloh. | | | | | | |
| Legenda farieb: | | | | | | |
| Dlh po deadline | Po deadline | Deň pred deadlineom | Dva dni pred deadlineom | Týždeň pred deadlineom | Dlh pred deadlineom | Po deadline s riešeniami |

Obr. 1.4: Zobrazenie úloh a hodnotení v pôvodnom systéme L.I.S.T. z pohľadu študenta, ktoré obsahuje prehľad získaných bodov, termíny a komentáre učiteľov.

Na Obr. 1.5 je zobrazené rozhranie pre nastavovanie zostáv úloh. Problémom je sústredenie veľkého množstva rôznorodých nastavení na jednom mieste bez vizuálneho rozlíšenia ich významu. Chýbajú napríklad nadpisy alebo oddelovacie prvky medzi sekciami (ako napríklad hodnotenie, typy súborov či deadline). Navyše pri viacerých voľbách nie sú dostupné žiadne tooltipy alebo pomocné texty, čo môže sťažovať pochopenie jednotlivých možností najmä novým používateľom.

Zostavy úloh

| | | | |
|--|---|-------|------------|
| O zostave úloh | Dalšie oprávnenia | Úlohy | Inštrukcie |
| Názov: * | K01 - Introduction Upravit jazykové prekyvia | | |
| Kurz: * | Operačné systémy | | |
| Typ zostavy úloh: * | Kvíz - raw | | |
| Zostava úloh iba pre skupinu: | <input type="checkbox"/> | | |
| Je publikovaná?: | <input type="checkbox"/> | | |
| Čas začiatku publikovania: | | | |
| Čas konca odosielania riešení: | | | |
| Zadať počet bodov manuálne?: | <input checked="" type="checkbox"/> | | |
| Počet bodov: | 18 | | |
| Povolení komentáre: | <input type="checkbox"/> | | |
| Nehajte prázdne, aby sa ďalej riešenia odosielali bez časového obmedzenia. | | | |
| Zadat počet bodov manuálne?: | <input checked="" type="checkbox"/> | | |
| Počet bodov: | 18 | | |
| Povolení komentáre: | <input type="checkbox"/> | | |
| Povolené typy súborov: | | | |
| Povolené typy súborov pre odosielanie studentských riešení. Ak študent posle súbor tohto typu, bude zabalený do ZIP archívu. Toto je čiarkou oddelený zoznam typov (prípony súborov). Pozor: Tieto prípony súborov musia byť definované v application/config/mimes.php, inak nebudú fungovať. <input type="checkbox"/> x86 Intel Assembler <input type="checkbox"/> C++ <input type="checkbox"/> Go <input type="checkbox"/> Haskell (GHC) <input type="checkbox"/> Java <input type="checkbox"/> Python <input type="checkbox"/> Text | | | |
| Povolené typy testov: | | | |
| Povolené typy testov: <input checked="" type="checkbox"/> x86 Intel Assembler <input type="checkbox"/> C++ <input type="checkbox"/> Go <input type="checkbox"/> Haskell (GHC) <input type="checkbox"/> Java <input type="checkbox"/> Python <input type="checkbox"/> Text | | | |
| Povolenie hodnotenie automatickými testami: | | | |
| Minimum úloh potrebných na hodnotenie: <input type="checkbox"/> | | | |
| Maximum úloh povolených na hodnotenie: | | | |
| <input type="checkbox"/> Počet úloh, ktorých testy musí študent vykonať, aby sa zohľadnilo riešenie zostavy. Ak je toto číslo väčšie ako počet úloh s hodnotiacimi testami, bude toto číslo korigované počas procesu hodnotenia. <input type="checkbox"/> 0 Maximálny počet úloh uvažovaných do hodnotenia. Toto číslo je počet najlepších výsledkov v testoch, ktoré budú zosumované do bodov. Ak má jedna úloha viacero hodnotiacich testov, ich výsledky sú zosumované do hodnotenia úlohy. <input type="checkbox"/> Normálna <input checked="" type="checkbox"/> Neposielaj žiadne notifikácie. <input type="checkbox"/> Použíť tieto e-mailové adresy spolu s e-mailovými adresami učiteľov v kurze / skupinách. <input type="checkbox"/> Posieľať e-mailové notifikácie iba na tieto e-mailové adresy. | | | |

Obr. 1.5: Rozhranie pre správu a nastavenie zostáv úloh v pôvodnom systéme L.I.S.T., kde sa nastavujú parametre ako povolené typy súborov, automatické hodnotenie a ďalšie možnosti.

Ďalším príkladom je kategorizácia úloh (Obr. 1.6), ktorá je súčasťou funkčnej vizuálnej techniky, ale je technicky zastaraná. Stromová štruktúra kategórií je zobrazovaná veľmi jednoduchým spôsobom, ktorý nepodporuje pohodlnú správu ani dobrú orientáciu v rámci rozsiahlejších dát.

Kategórie

Nová kategória

+ Zobrazit / skryť formulár

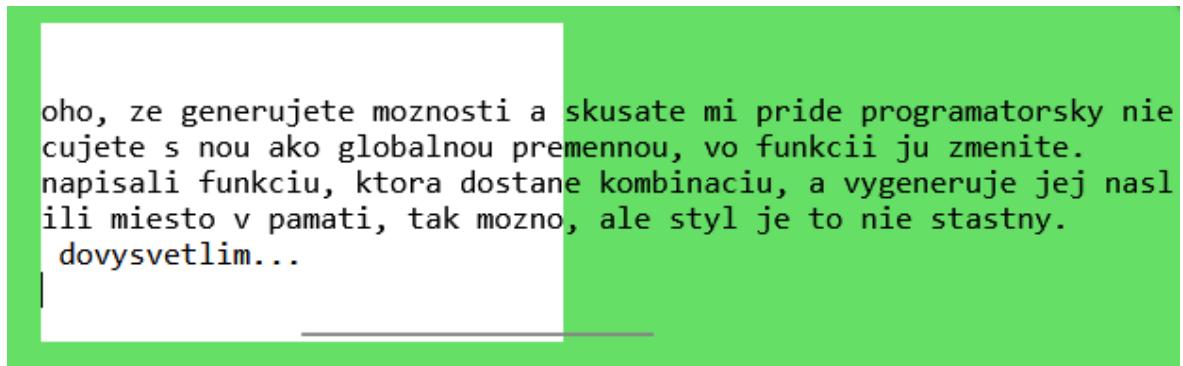
Strom kategórií

- DEBUG--- (6)
- AI (5)
 - časové rady (2)
 - autoregresny model (2)
 - Period-doubling bifurcation (2)
 - vyhľadžovanie (2)  
 - časové rady s neurčitosťou (0)
 - skrytý markovovský model (HMM) (8)
 - evolucne algoritmy (0)
 - geneticke algoritmy (3)
 - hry (1)
 - Markovovsky rozhodovaci proces (2)
 - logika (0)
 - logika prvého rádu (3)
 - výroková logika (2)
 - lokálne optimalizácie (1)
 - horolezecký algoritmus (3)
 - planovanie (3)
 - pravdepodobnosť (4)
 - Bayesovske siete (9)
 - Monte Carlo Tree Search (1)
 - Naivna klasifikacia (5)
 - sampling (1)
 - priame sampling (3)
 - rejection sampling (3)

Obr. 1.6: Rozhranie pre správu kategórií úloh v pôvodnom systéme L.I.S.T., ktoré zobrazuje stromovú štruktúru tém a umožňuje úpravy a správu kategórií.

Problémy v oblasti UI/UX možno pozorovať aj pri konkrétnom komponente urče-

nom na zobrazovanie komentárov (Obr. 1.7). Ak učiteľ zadá dlhší komentár, študent pri jeho prezeraní musí horizontálne posúvať obsah, pretože text sa nezalamuje automaticky a zobrazovací box má fixnú šírku. Navyše, ak text presiahne šírku bieleho pozadia, pokračuje na zelenom podklade mimo hlavného okna, čo pôsobí rušivo a znižuje čitateľnosť. Tieto problémy by sa dali odstrániť implementáciou automatického zalamovania textu (word-wrap: break-word) a nastavením maximálnej šírky boxu podľa šírky rodičovského kontajnera.



Obr. 1.7: Zobrazenie dlhého komentára učiteľa s horizontálnym posúvaním a vybočením mimo bieleho rámu.

Celkovo možno povedať, že používateľské rozhranie pôvodného systému nereflektuje dnešné štandardy UX/UI dizajnu. Nedostatky sa prejavujú najmä:

- **Nedostatočná hierarchia a prehľadnosť** - Na stránke zadávania úloh sú všetky nastavenia zobrazené v jednej sekcií bez logického členenia, čo sťažuje rýchle vyhľadanie konkrétnych funkcií.
- **Vizuálny štýl** - Použitá farebná schéma a typografia súce zabezpečujú čitateľnosť, no pôsobia nekoherentne a rušivo, čím znižujú estetickú kvalitu rozhrania.
- **Neintuitívne formuláre** - Niektoré nastavenia obsahujú možnosti (napr. začiarkavacie polička) bez sprievodného popisu, čo môže viesť k nejasnostiam pri ich používaní.
- **Nedostatočná responzivita** - Niektoré komponenty (napr. komentáre alebo formuláre) si vyžadujú horizontálne posúvanie alebo nezalamujú text, čo výrazne znižuje použiteľnosť na menších obrazovkách a pri práci s dlhšími vstupmi.

Tieto nedostatky naznačujú potrebu zásadnej zmeny používateľského rozhrania s dôrazom na prehľadnosť, jednoduchosť ovládania, responzivitu a moderný dizajn.

1.3 Ďalšie existujúce LMS systémy

Na trhu existuje množstvo rôznych LMS systémov, ktoré sa líšia cieľovou skupinou, funkcionálitou, cenou a spôsobom nasadenia. Medzi najznámejšie patria open-source riešenia aj komerčné produkty, ktoré používajú univerzity a rôzne inštitúcie na celom svete. Tieto systémy poskytujú širokú škálu funkcionálít - od správy kurzov a úloh cez hodnotenie a testovanie až po integrácie s externými službami.

1.3.1 Moodle

Moodle je celosvetovo rozšírený systém typu LMS, ktorý slúži na podporu výučby. Ide o open-source softvér, čo v praxi znamená, že je dostupný zadarmo a môže byť voľne upravený a rozšírený podľa potrieb konkrétnej organizácie. Systém bol pôvodne vyvinutý v roku 2002 s cieľom vytvoriť flexibilnú platformu podporujúcu výučbu.[14]

Moodle ponúka veľké množstvo funkcionálít, ktoré pokrývajú väčšinu potrieb moderného vzdelávania. Medzi jeho hlavné funkcie patria[12]:

- **Správa kurzov a obsahu** - vytváranie kurzov, nahrávanie materiálov, organizácia učebných zdrojov do rôznych kategórií, sekcií alebo tém.
- **Zadávanie úloh a testovanie** - učitelia môžu zadávať úlohy, nastavovať termíny a automatizované testy s rôznymi typmi otázok.
- **Hodnotenie a spätná väzba** - systém obsahuje flexibilný hodnotiaci modul a umožňuje poskytovať spätnú väzbu k študentským prácам.
- **Komunikačné nástroje** - diskusné fóra, správy, chaty a emailové notifikácie uľahčujú komunikáciu medzi účastníkmi kurzov.
- **Analytika a reportovanie** - sledovanie pokroku študentov, štatistiky aktivít a generovanie prehľadov.
- **Rozšíriteľnosť** - dostupné stovky pluginov na prispôsobenie funkcionality podľa konkrétnych potrieb školy alebo univerzity.

Výhodou Moodle je jeho flexibilita, rozsiahla komunita používateľov a vývojárov a možnosť integrácie s inými systémami. Umožňuje tiež multijazyčné prostredie a je vhodný pre rôzne úrovne vzdelávania - od základných škôl až po univerzity a firemné školenia. Medzi slabšie stránky patrí niekedy zložitejšie používateľské rozhranie, ktoré môže byť pre nových používateľov menej intuitívne, a potreba vlastnej technickej správy a údržby pri lokálnej inštalácii.[10]

Moodle na Univerzite Komenského

Univerzita Komenského v Bratislave využíva Moodle ako svoju hlavnú platformu pre e-learning. Systém je spravovaný Centrom informačných technológií UK a je dostupný pre všetkých učiteľov a študentov s univerzitným prihlásovacím menom a heslom. Moodle na UK podporuje tvorbu kurzov, zadávanie úloh, diskusie a ďalšie formy online výučby. Systém je každoročne aktualizovaný na novú verziu, pričom sa prenášajú kurzy z predchádzajúcich rokov bez študentských dát, aby boli pripravené na ďalší akademický rok.[15]

1.3.2 Google Classroom

Google Classroom je cloudová platforma pre správu výučby, ktorú vyvinula spoločnosť Google. bola predstavená v roku 2014 ako súčasť balíka Google Workspace for Education a je určená na zjednodušenie vytvárania, šírenia a hodnotenia úloh vo vzdelávaní. Základná verzia služby je bezplatná a dostupná pre používateľov, ktorí používajú Google účty, pričom existujú aj rozšírené platené verzie s pokročilejšími funkciami [4].

Hlavným cieľom Google Classroom je uľahčiť komunikáciu medzi učiteľmi a študentmi a znížiť množstvo papierovej dokumentácie. Platforma je priamo integrovaná s ďalšími nástrojmi od Google, ako sú Google Disk, Dokumenty, Tabuľky, Prezentácie či Meet, čo umožňuje plynulé zdieľanie a spoluprácu v reálnom čase.

Hlavné funkcionality:

- **Správa kurzov a úloh:** Učitelia môžu jednoducho vytvárať kurzy, zadávať úlohy, nastavovať termíny odovzdania a poskytovať spätnú väzbu.
- **Hodnotenie:** Systém umožňuje priebežné aj záverečné hodnotenie študentov vrátane komentárov a bodového ohodnotenia.
- **Komunikácia:** Učitelia a študenti môžu komunikovať prostredníctvom oznámení, komentárov pod úlohami a súkromných správ.
- **Integrácia:** Úzka integrácia s Google Diskom na správu súborov, Google Meet pre videohovory a ďalšími aplikáciami v rámci Google Workspace.
- **Prístupnosť:** Podpora mobilných aplikácií (Android, iOS), čo umožňuje prístup k materiálom a úlohám kdekoľvek.
- **Automatizácia:** Automatické generovanie priečinkov na Google Disku pre jednotlivé kurzy a úlohy, čím sa zjednoduší organizácia materiálov.

Výhody:

- Jednoduché a intuitívne používateľské rozhranie.

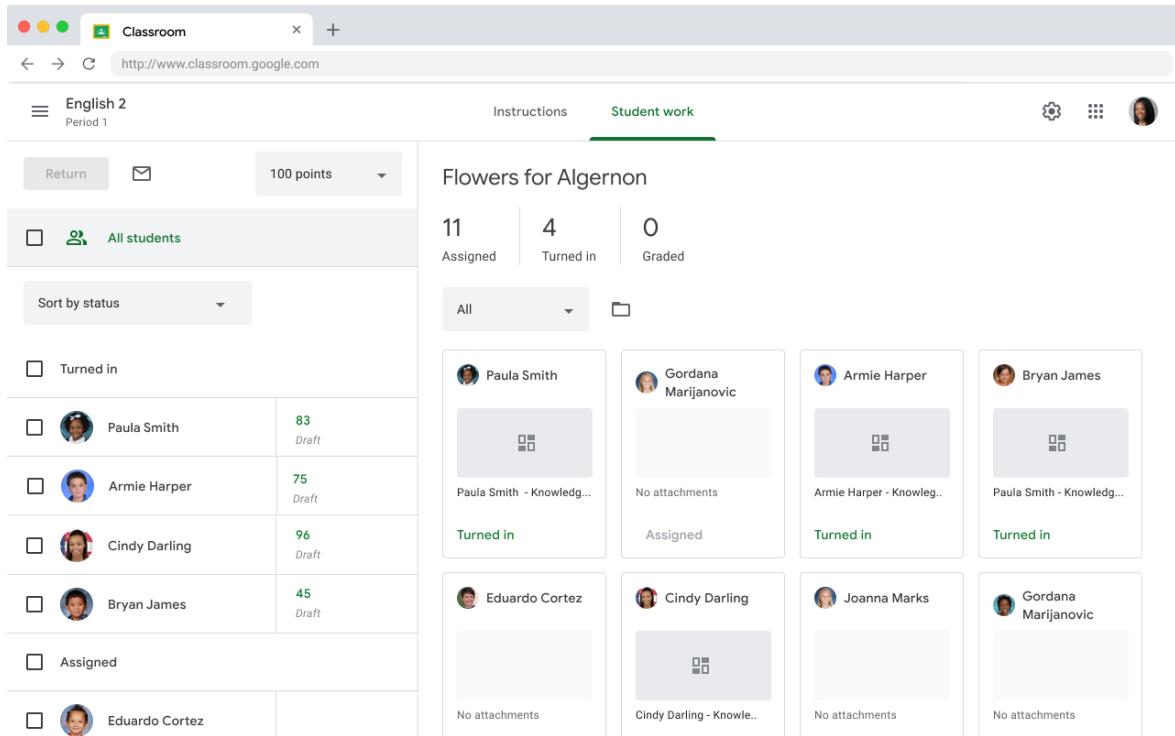
- Silná integrácia s nástrojmi Google.
- Rýchle nasadenie a minimálna potreba správy IT infraštruktúry.
- Bezzplatná základná verzia dostupná pre školy.

Nevýhody:

- Menej pokročilé možnosti prispôsobenia v porovnaní s robustnými LMS systémami ako Moodle.
- Závislosť od Google účtov a clouдовých služieb.
- Obmedzené analytické a reportovacie nástroje v základnej verzii.

Google Classroom je široko využívaný najmä na základných a stredných školách, ale čoraz častejšie aj na univerzitách ako doplnok ku komplexnejším LMS systémom. Podľa údajov z roku 2021 malo Google Classroom približne 150 miliónov používateľov po celom svete, pričom jeho popularita výrazne vzrástla počas pandémie a nadálej rastie vďaka jednoduchému používaniu a bezplatnému prístupu pre školy. Platforma sa v USA stala dominantnou vo vzdelávacom prostredí - až 68% školských obvodov ju pravidelne využíva.[2]

Treba však poznamenať, že spoločnosť Google je známa častými zmenami vo svojom produktovom portfóliu, vrátane náhlych ukončení služieb bez priamej náhrady. Využívanie Google Classroom preto nesie isté riziká z pohľadu dlhodobej udržateľnosti a závislosti od externého komerčného poskytovateľa, čo môže byť problematické najmä pre verejné inštitúcie či školy, ktoré preferujú otvorené alebo dlhodobo stabilné riešenia.



Obr. 1.8: Náhľad prostredia Google Classroom pri správe zadania a odovzdaných riešení študentov. Tento pohľad umožňuje učiteľom rýchlo skontrolovať stav odovzdania jednotlivých úloh a priebežne sledovať aktivity študentov[5].

1.4 Technológie

Táto kapitola predstavuje výber technológií, ktoré sú v súčasnosti často využívané pri vývoji moderných webových aplikácií. Ich charakteristiky zahŕňajú širokú podporu, dostupnú dokumentáciu, stabilitu, bezpečnosť a vhodnosť na tvorbu škálovateľných riešení. Opisované technológie pokrývajú oblasti backendu, frontendu, práce s databázou, autentifikácie a vývoja API.

1.4.1 Backendové technológie

.NET a C#

.NET je robustný framework, ktorý poskytuje výkonné prostredie pre vývoj webových aplikácií vrátane backendových služieb. Programovací jazyk **C#** patrí medzi najpoužívanejšie a je známy dobrou čitateľnosťou, silnou typovou kontrolou a rozsiahlymi možnosťami využitia. Vďaka svojej multiplatformovej podpore umožňuje vývoj aplikácií naprieč rôznymi operačnými systémami a je podporovaný silnou komunitou aj oficiálnou dokumentáciou.[7]

ASP.NET Core Web API

ASP.NET Core Web API je moderný framework určený na tvorbu aplikačných rozhraní, ktoré sprostredkúvajú komunikáciu medzi klientskou aplikáciou (napríklad webovým rozhraním) a serverom. Umožňuje jednoducho spracovať požiadavky, poskytovať údaje a vykonávať operácie ako vytváranie, čítanie, úprava a mazanie záznamov. Medzi jeho výhody patrí dobrý výkon, flexibilita a podpora asynchronného spracovania požiadaviek, čo je dôležité najmä pri aplikáciách s väčším počtom používateľov.

Entity Framework Core

Entity Framework Core je objektovo-relačný mapovací nástroj (ORM), ktorý výrazne uľahčuje prácu s databázou prostredníctvom objektového modelu. Podporuje automatické migrácie a využívanie LINQ dotazov, ktoré umožňujú prácu s dátami priamo v kóde, čo zrýchluje vývoj a znižuje riziko chýb. Vďaka širokej podpore databáz je vhodný pre projekty rôzneho rozsahu.

PostgreSQL

PostgreSQL patrí medzi najpopulárnejšie open-source relačné databázové systémy a je známy svojou stabilitou a vysokou spoľahlivosťou. V kombinácii s ORM nástrojmi dokáže efektívne zabezpečiť ukladanie a správu údajov aj pri komplexných systémoch.

Swagger

Na generovanie a údržbu dokumentácie API sa využíva **Swagger**, ktorý automaticky tvorí prehľadnú a interaktívnu dokumentáciu. Táto dokumentácia výrazne uľahčuje vývoj, testovanie a slúži ako spoľahlivý zdroj informácií pri ďalších úpravách systému.

The screenshot shows the Swagger UI interface for a RESTful API. At the top, there's a header with 'List API' and 'OAS 3.0'. Below it, a link to '/swagger/v1/swagger.json' and a note 'Backend GET/POST/PUT/DELETE'. The main area is organized into sections:

- Assignments** (blue header):
 - GET /api/assignments
 - POST /api/assignments
 - GET /api/assignments/{id}
 - PUT /api/assignments/{id}
 - DELETE /api/assignments/{id}
- AssignmentTaskRel** (green header):
 - POST /api/assignment-task-rel
 - DELETE /api/assignment-task-rel
 - GET /api/assignment-task-rel
 - GET /api/assignment-task-rel/by-assignment/{assignmentId}
 - GET /api/assignment-task-rel/by-task/{taskId}
- Auth** (blue header):
 - POST /api/auth/register
 - POST /api/auth/login

Obr. 1.9: Ukážka používateľského rozhrania nástroja Swagger, ktorý zobrazuje dostupné API rozhrania vrátane podrobností o jednotlivých požiadavkách a odpovediach.

1.4.2 Frontendové technológie

React a TypeScript

React je JavaScriptová knižnica určená na tvorbu používateľských rozhraní. Jej hlavnou výhodou je komponentový prístup, ktorý umožňuje rozdeľovať aplikáciu na malé, znovupoužiteľné časti. Tento prístup uľahčuje správu veľkých projektov a zlepšuje čitateľnosť a údržbu kódu. React je široko využívaný v priemysle a má rozsiahlu komunitu a podporu.

TypeScript rozširuje JavaScript o typovú kontrolu. Zavedenie typov zvyšuje bezpečnosť a spoľahlivosť kódu, keďže umožňuje odhaliť chyby už počas vývoja. TypeScript zároveň zlepšuje čitateľnosť a udržiavateľnosť aplikácie, čo je významné najmä pri väčších projektoch.

Material UI

Material UI je knižnica predpripravených komponentov, ktorá vychádza z dizajnového jazyka Material Design. Ponúka rýchle a konzistentné vytváranie moderného a responzívneho používateľského rozhrania. Komponenty ako tlačidlá, tabuľky či formuláre sú navrhnuté s dôrazom na prístupnosť a kvalitný dizajn, čo pomáha k jednotnému

štýlovaniu aplikácie.

TinyMCE

TinyMCE je editor, ktorý slúži na úpravu a formátovanie textu. Ide o bohatý WYSIWYG (Čo vidíte, to dostanete) editor, ktorý používateľom umožňuje jednoducho vytvárať a formátovať obsah priamo v prehliadači. Je vhodný najmä na zadávanie textových úloh alebo poznámok a podporuje rôzne možnosti formátovania vrátane odkazov, obrázkov a zoznamov.

1.4.3 Autentifikácia a zabezpečenie pomocou JWT

JWT Bearer slúži na zabezpečenie autentifikácie používateľa, ktorá umožňuje overenie a autorizáciu používateľov prostredníctvom šifrovaných tokenov. Tento prístup poskytuje riešenie pre spravovanie prístupu k chráneným zdrojom.

Kapitola 2

Návrh

Vývoj novej verzie systému L.I.S.T. prebieha ako tímový projekt, pričom jednotliví členovia tímu pracujú na samostatných častiach podľa vopred dohodnutého rozdelenia. Táto kapitola sa zameriava primárne na tie časti systému, ktorých majoritu som osobne navrhol a implementoval.

Moja práca sa týka oblastí autentifikácie používateľov, kontroly prístupu pomocou používateľských rolí, manažmentu kurzov a úloh, ako aj návrhu používateľského rozhrania pre študentov. Pri návrhu týchto častí bol kladený dôraz na bezpečnosť, prehľadnosť, škálovateľnosť a dobrú používateľskú skúsenosť.

Podrobnejšie požiadavky na funkcionality týchto komponentov sú uvedené v nasledujúcej časti.

2.1 Definícia dôležitých pojmov

Nezávisle od konkrétnej implementácie je možné definovať niekoľko základných konceptov s ktorými sa stretnete v nasledujúcich kapitolách, ktoré tvoria jadro systému:

- **Používateľ** - Entita, ktorá interahuje so systémom. Môže mať rôzne roly, ktoré ovplyvňujú prístup k funkcionálitám systému.
- **Obdobie** - Časový úsek (napr. semester), v rámci ktorého sa otvárajú kurzy a prebieha výučba.
- **Kurz** - Predstavuje vyučovací predmet vedený jedným alebo viacerými učiteľmi, ku ktorému sú priradené úlohy, zadania a študenti.
- **Zostava úloh (zadanie)** - Súbor úloh priradených ku konkrétnemu kurzu.
- **Úloha (task)** - Jednotlivá programátorská alebo teoretická úloha, ktorá sa nachádza v jednej alebo viacerých zostavách.

- **WYSIWYG editor** - Textový editor, ktorý umožňuje formátovanie zadania úlohy alebo popisu kurzu pomocou grafického rozhrania. Podporuje formátovanie textu, tabuľky, obrázky či vložený kód.
- **Kategória úloh** - Hierarchicky usporiadaná štruktúra tém, do ktorých sú úlohy zaradené.
- **Rola** - Typ používateľa, ktorý definuje jeho oprávnenia.
- **Účastník kurzu** - Každý študent zapísaný v konkrétnom kurze.

2.2 Požiadavky

V tejto časti sú popísané konkrétné funkčné požiadavky na tie komponenty systému, ktorých návrh a implementáciu som riešil. Ide o základ pre implementáciu jednotlivých častí, pričom každá požiadavka odráža buď potrebu používateľov, alebo technickú nevyhnutnosť. Požiadavky sú rozdelené tematicky podľa oblastí, v ktorých sa funkcia nachádza.

2.2.1 Používatelia

- Používateľ sa musí vedieť bezpečne prihlásiť do systému, ak má vytvorený účet.
- Používateľ sa musí vedieť odhlásiť zo systému.
- Používateľ si musí vedieť požiadať o obnovenie hesla v prípade jeho zabudnutia.
- Učiteľ musí mať možnosť vytvoriť nový používateľský účet.
- Učiteľ musí mať možnosť upraviť údaje existujúceho používateľa.
- Učiteľ musí mať možnosť prihlásiť sa do systému ako konkrétny študent (napr. na účely testovania).
- Systém musí umožniť import študentov zo súboru (napr. exportovaného z AISu) a vytvoriť im účty, ak ešte neexistujú.

Používateľské roly

- Každý používateľ musí mať priradenú rolu, ktorá definuje jeho oprávnenia v systéme.
- Systém musí rozlišovať medzi minimálne tromi typmi rolí: študent, učiteľ a asistent.

- Rola „študent“ umožňuje napríklad zápis do kurzu a ich zobrazenie alebo odovzdávanie riešení.
- Rola „učiteľ“ oprávňuje používateľa spravovať kurzy, úlohy, používateľov a hodnotenia.
- Rola „asistent“ má obmedzené učiteľské oprávnenia a prístupy, zároveň má možnosť slúžiť ako študentský účet.
- Učiteľ musí mať možnosť meniť roly existujúcim používateľom.
- Systém musí rešpektovať prístupové oprávnenia definované rolou pri každej akcii používateľa.

2.2.2 Kurzy

- Učiteľ musí mať možnosť vytvoriť nový kurz, upraviť jeho údaje a v prípade potreby ho aj odstrániť.
- Pri vytváraní alebo editovaní kurzu môže učiteľ nastaviť:
 - názov kurzu,
 - obdobie (semester alebo školský rok),
 - kapacitu kurzu (maximálny počet študentov),
 - dátum uzávierky zápisu,
 - viditeľnosť kurzu pre študentov,
 - spôsob zápisu (automaticky / na schválenie učiteľom).
- Ku každému kurzu môže učiteľ pridať popis prostredníctvom WYSIWYG editora, pričom má možnosť prepínať medzi režimom úprav a náhľadom, ktorý zobrazuje popis tak, ako ho vidí študent.
- Kurzy musia byť prepojené so zostavami úloh, pričom učiteľ má možnosť priradiť zostavu úloh k danému kurzu.
- Systém musí umožniť správu účastníkov - učiteľ má možnosť schvaľovať alebo zamietať žiadosti študentov o zápis do kurzu.
- Študenti musia mať možnosť zobraziť si detail kurzu vrátane popisu a základných informácií, zaradených zadaní a prehľadu svojich výsledkov v rámci kurzu.

2.2.3 Úlohy

- Umožniť vytváranie, úpravu a mazanie úloh.
- Pri úlohe je možné nastaviť:
 - názov úlohy,
 - zadanie pomocou WYSIWYG editora (vrátane formátovania, obrázkov a kódu),
 - interný komentár pre učiteľov (neviditeľný pre študentov).
- Používateľ môže prepínať medzi režimom editácie a náhľadom, ktorý zobrazuje úlohu tak, ako ju uvidí študent.
- Každá úloha môže byť priradená ku kategórii.
- Učiteľ môže vytvoriť novú úlohu kopírovaním existujúcej, pričom ju môže efektívne upraviť.

2.2.4 Študentské používateľské rozhranie

- Po prihlásení má študent prístup k zoznamu kurzov, pričom kurzy sú rozdelené na tie, do ktorých je študent zapísaný, a ostatné dostupné kurzy.
- Ak je to povolené konfiguráciou kurzu, študent môže požiadať o zápis do kurzu, do ktorého ešte nie je zapísaný.
- Po kliknutí na konkrétny kurz sa študentovi otvorí detailná stránka kurzu s viačerými sekciami: popis, zadania a prehľad hodnotenia.
- Sekcia **Popis** zobrazuje základné informácie o kurze, ako sú jeho názov, obdobie, učiteľ, kapacita a textový popis.
- Sekcia **Zadania** obsahuje zoznam zostáv úloh priradených ku kurzu, rozdelených podľa typu. Každé zadanie uvádza maximálny počet bodov a počet bodov, ktoré študent získal (ak už bolo ohodnotené).
- Ku každému zadaniu je uvedený deadline, ktorý je vizuálne zvýraznený farebne podľa jeho blízkosti, alebo či už uplynul.
- Po otvorení konkrétneho zadania sa zobrazí jeho detail spolu so zoznamom úloh. Študent má možnosť nahrať súbor ako riešenie zadania.

- Sekcia **Prehľad hodnotenia** zobrazuje aktuálny počet bodov, ktoré študent získal v danom kurze, spolu s maximálnym možným počtom bodov. Ďalej ponúka prehľad bodového zisku podľa typu zadania a zoznam najbližších deadlineov.
- Používateľské rozhranie je optimalizované pre jednoduchosť a prehľadnosť, s dôrazom na rýchly prístup k aktuálnym zadaniám a termínom.

2.3 Technológie a ich prepojenia

Pri vývoji systému boli zvolené technológie, ktoré predstavujú moderný štandard v oblasti webového vývoja, pričom dôraz bol kladený na stabilitu, výkonnosť, rozšírenosť a dlhodobú podporu. Vývojový stack bol navrhnutý tak, aby umožňoval čisté oddelenie backendovej a frontendovej časti systému s možnosťou rozšíriteľnosti v budúcnosti.

2.3.1 Backend

Serverová časť systému je implementovaná pomocou platformy .NET 8, ktorá poskytuje robustné a efektívne prostredie pre vývoj webových aplikácií. Ako hlavný programovací jazyk bol použitý C#, ktorý v kombinácii s ASP.NET Core Web API umožňuje vytvárať RESTful aplikačné rozhrania. Tieto rozhrania tvoria základnú komunikačnú vrstvu medzi frontendom a backendom a zabezpečujú prenos dát medzi klientom a serverom. Pre prácu s databázou bol použitý objektovo-relačný mapovací nástroj Entity Framework Core, ktorý zjednodušuje manipuláciu s dátami prostredníctvom objektového modelu a podporuje generovanie migrácií.

Dátová vrstva je postavená na open-source relačnej databáze PostgreSQL, ktorá je známa svojou výkonnosťou, spoľahlivosťou a rozšírenou podporou v rámci .NET ekosystému. Na dokumentáciu a testovanie API rozhraní bol využitý nástroj Swagger, ktorý automaticky generuje interaktívne rozhranie pre vývojárov a uľahčuje overovanie správania jednotlivých endpointov.

2.3.2 Frontend

Klientská časť systému je implementovaná pomocou knižnice React, ktorá umožňuje tvorbu dynamických a responzívnych používateľských rozhraní založených na komponentovom prístupe. Vývoj vo frameworku prebieha v jazyku TypeScript, ktorý rozširuje JavaScript o statickú typovú kontrolu a zvyšuje spoľahlivosť pri väčších projektoch. Na tvorbu používateľského rozhrania bola použitá knižnica Material UI, ktorá ponúka predpripravené vizuálne komponenty s dôrazom na konzistenciu a mobilnú responzitivitu.

Na formátovanie textového obsahu (napr. zadania úloh či popisy kurzov) bol integrovaný editor TinyMCE. Ide o plnohodnotný WYSIWYG editor, ktorý umožňuje používateľom intuitívne vytvárať štruktúrovaný obsah s podporou formátovania, tabuľiek, obrázkov a ďalších prvkov, pričom výstup je uložený vo forme HTML.

Frontend a backend spolu komunikujú výhradne prostredníctvom REST API, pričom každá interakcia používateľa na klientovi (napr. otvorenie kurzu, odovzdanie riešenia, zmena údajov) vyvoláva požiadavku na konkrétny endpoint na serveri.

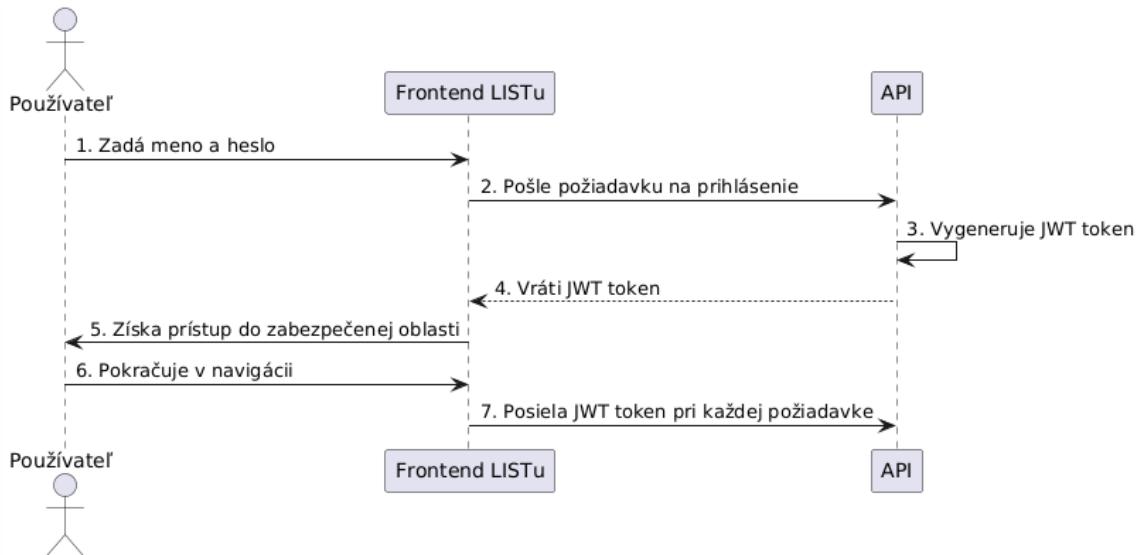
2.4 Bezpečnosť a autentifikácia

Bezpečnosť systému je riešená na viacerých úrovniach - od overovania identity používateľov, cez riadenie prístupu k jednotlivým časťam systému, až po ochranu proti automatizovaným útokom. Vzhľadom na architektúru SPA (Single Page Application), kde frontend a backend komunikujú cez REST API, je nevyhnutné zabezpečiť všetky požiadavky, ktoré smerujú na server.

2.4.1 Tokenová autentifikácia (JWT)

Na overenie identity používateľov bola zvolená bezstavová autentifikácia pomocou JWT (JSON Web Token). Po úspešnom prihlásení je používateľovi serverom vygenerovaný token, ktorý obsahuje základné informácie (identifikátor používateľa, rolu) a je podpísaný tajným kľúčom. Tento token sa následne posiela pri každej požiadavke ako súčasť hlavičky.

Backend pri spracovaní požiadavky token overí, a ak je platný, priradí požiadavke identitu používateľa. Výhodou tohto prístupu je škálovateľnosť a jednoduchšia správa relácií bez potreby udržiavať stav na serveri.



Obr. 2.1: Sekvenčný diagram zachytáva proces prihlásenia používateľa, generovania JWT tokenu na strane API a následného posielania tohto tokenu pri ďalších požiadavkách klienta.

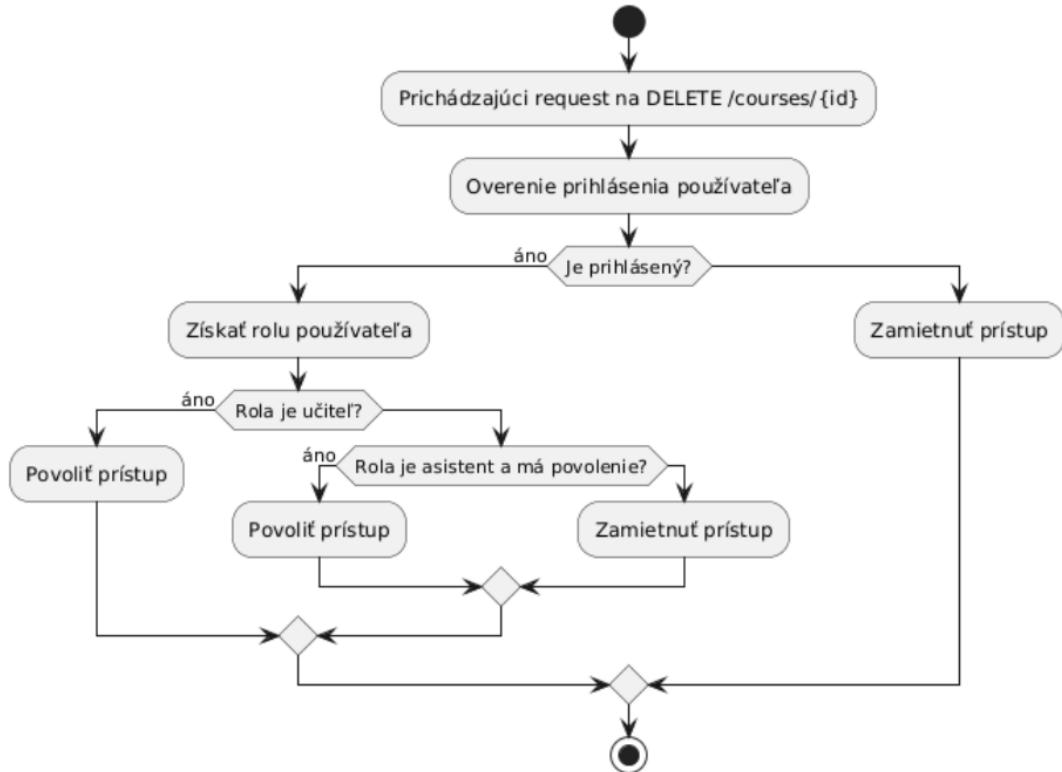
2.4.2 Autorizácia a ochrana endpointov

Autorizácia je realizovaná pomocou vstavaných mechanizmov ASP.NET Core. Prístup k jednotlivým API endpointom je kontrolovaný pomocou atribútov ako `[Authorize]`, ktoré zabezpečujú, že k citlivým operáciám majú prístup len oprávnení používateelia.

Autorizáciu je možné rozšíriť o špecifikáciu rolí či podmienok - napríklad:

- `GET /courses` - dostupné pre každého prihláseného používateľa,
- `DELETE /courses/{id}` - prístupné len pre používateľa s rolou učiteľa alebo asistenta so špecifickým povolením.

Zároveň tento prístup zvyšuje odolnosť systému voči útokom typu **Cross-Site Request Forgery (CSRF)**. Keďže JWT token sa neukladá do cookies, ale je manuálne vkladaný do hlavičky `Authorization`. To znemožňuje útočníkovi vykonať podvrhnutú požiadavku bez priameho zásahu používateľa, čím sa eliminuje väčšina bežných CSRF útokov.



Obr. 2.2: Activity diagram znázorňuje rozhodovací proces pri volaní citlivého endpointu `DELETE /courses/{id}`. Na základe roly používateľa a jeho oprávnení sa rozhoduje, či má k operácii prístup.

Rovnaký princíp sa uplatňuje pri správe úloh, používateľov, hodnotení a ďalších entít.

2.4.3 Hashovanie hesiel pomocou BCrypt

Pri ukladaní hesiel sa nikdy neukladá ich pôvodná (plaintext) verzia. Namiesto toho sa heslá hashujú pomocou knižnice `BCrypt.Net`.

Takto získaný hash sa uloží do databázy. Použitie algoritmu **BCrypt** prináša viačeré výhody z hľadiska bezpečnosti. Vďaka zabudovanému náhodnému saltu je odolný voči rainbow table útokom, keďže pre každé heslo sa generuje unikátny hash. Navyše, samotný algoritmus je navrhnutý tak, aby bol zámerne výpočtovo náročnejší, čo výrazne znižuje efektivitu brute-force útokov pri pokusoch o prelomenie hesiel.

2.4.4 Ochrana pred útokmi - Rate Limiting

Na ochranu systému pred nadmerným množstvom požiadaviek (napr. brute-force útoky, DDoS, skriptované bombardovanie API) je nasadený mechanizmus **rate limiting**. Ten obmedzuje počet požiadaviek z jednej IP adresy počas daného časového intervalu.

Týmto spôsobom sú chránené nielen prihlásovacie požiadavky, ale všetky verejne dostupné API endpointy, čo výrazne znižuje riziko zahľtenia systému alebo zneužitia rozhrania na neautorizované skenovanie.

2.4.5 Záznam prístupov a auditovanie

Na účely späťnej analýzy a zvýšenia bezpečnosti je súčasťou systému aj logovanie vybraných akcií ako je volanie citlivých API operácií, ako je mazanie, zmeny používateľských údajov alebo zásahy do kurzov a hodnotení.

Tieto záznamy môžu slúžiť na detekciu podozrivého správania, spätnú kontrolu incidentov, prípadne na auditovanie systému v rámci správy a údržby.

2.5 Modulárna architektúra systému

2.5.1 Rozdelenie systému na moduly

Architektúra systému je navrhnutá modulárne, pričom každá tematická oblasť (napr. kurzy, úlohy, používateľia) je implementovaná ako samostatný modul, a to konzistentne na úrovni backendu aj frontendu.

Na strane backendu každý modul (napr. List.Users, List.Courses, List.Tasks) predstavuje samostatný projekt v rámci riešenia. Obsahuje vlastný DbContext, dátové modely, kontroléry, služby a štruktúru potrebnú pre nezávislý vývoj a testovanie. Spoločné typy a rozhrania sú zdieľané cez projekt List.Common, ktorý slúži ako centrálné miesto pre definíciu kontraktov a infraštrukturých typov. Moduly sú registrované cez rozhranie IModule, čo umožňuje dynamickú a prehľadnú inicializáciu v hlavnom projekte List.Server.

Frontendová časť systému dodržiava rovnaký princíp: každá funkcia má vlastný priečinok v rámci adresára src/modules, kde sa nachádzajú jej komponenty, stránky a typy. Napríklad modul Courses obsahuje dialógy na vytváranie a úpravu kurzov, ako aj zodpovedajúce zobrazenia. Zdieľané komponenty a služby sú umiestnené v priečinkoch shared a services.

Takéto rozdelenie systému uľahčuje orientáciu v kóde, zvyšuje znovupoužiteľnosť komponentov a umožňuje paralelný vývoj viacerých tímov bez nadmerného prepojenia medzi jednotlivými časťami.

2.5.2 Komunikácia medzi modulmi

Moduly backendu medzi sebou nekomunikujú pomocou explicitného volania cez rozhrania, ale ich závislosti sú riešené prostredníctvom projektových referencií (Project-Reference). Každý modul môže referencovať iný modul, ak potrebuje využiť jeho typy

alebo služby. Napríklad modul `Courses` má referenciu na moduly `Users` a `Logs`, čo mu umožňuje pracovať s používateľskými údajmi alebo zaznamenávať akcie do logov. Takáto komunikácia je jednostranná - modul deklaruje závislosť na inom module, čím sa zachováva jednosmernosť a prehľadnosť závislostí.

Modul `Common` slúži ako spoločná knižnica zdieľaná viacerými časťami systému. Obsahuje všeobecne použiteľné služby (napr. pre ukladanie súborov) alebo pomocné funkcie, ktoré nie sú viazané na konkrétnu doménu. Nie je teda určený na sprostredkovanie komunikácie medzi doménami, ale na centralizáciu opakovane použiteľnej logiky.

Na strane frontendu sú moduly reprezentované ako samostatné priečinky v rámci štruktúry `src/modules`, ktoré obsahujú komponenty, stránky a typy súvisiace s danou doménou (napr. `Courses`, `Tasks`, `Users`). Komponenty sú navrhnuté tak, aby boli zno-vupoužiteľné a ľahko integrované do iných častí aplikácie. Ich zdieľanie medzi modulmi prebieha jednoducho prostredníctvom `import` direktív, čo je štandardný spôsob práce v React ekosystéme. Spoločná logika je sústredená v priečinkoch `services` a `shared`.

Aj keď medzi frontend modulmi neexistuje formálna hranica ako na backende, do-držiava sa tematické rozdelenie komponentov do priečinkov podľa funkcionality.

2.6 Opis implementovaných častí systému

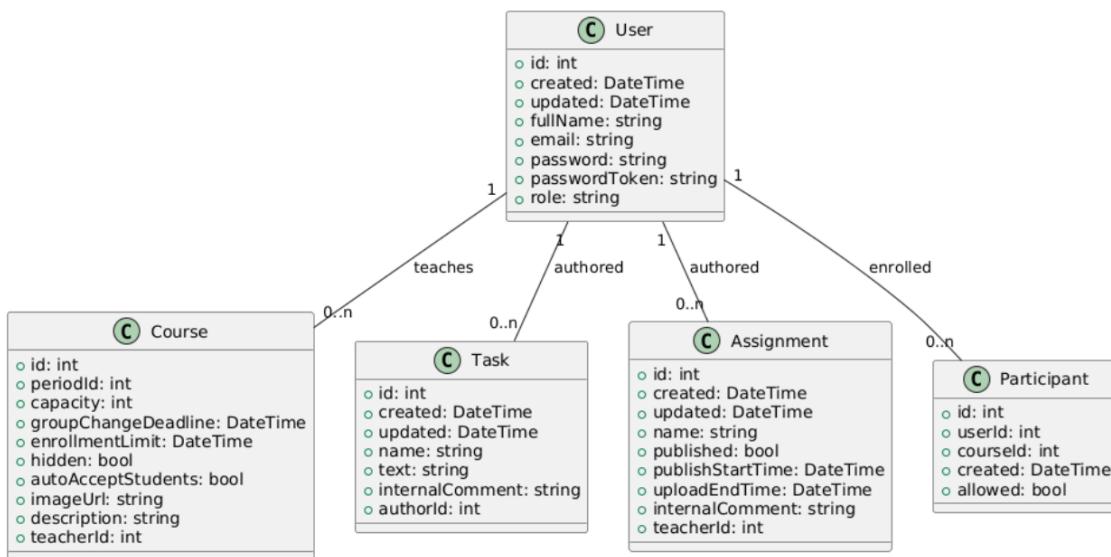
2.6.1 Modul Users

Modul `Users` predstavuje centrálnu súčasť systému, ktorá spravuje všetkých používateľov – študentov, učiteľov, prípadne ďalšie roly. Každý používateľ má okrem identifikačných údajov (meno, e-mail) aj prihlásovacie informácie (heslo, bezpečnostný token) a rolu určujúcu jeho oprávnenia v systéme.

Používatelia sú priamo previazaní s viacerými entitami iných modulov:

- ako autori úloh v module `Tasks`,
- ako tvorcovia zadania v module `Assignments`,
- ako vyučujúci kurzov v module `Courses`,
- a ako študenti zapísaní v kurzoch cez entitu `Participant`.

Vzhľadom na kľúčové postavenie tejto entity sú vzťahy smerované z ostatných modulov na používateľa, čo zaručuje jednoznačnú identifikateľnosť a konzistenciu údajov naprieč systémom. Prístup k rôznym funkciałitám je riadený práve na základe roly používateľa.



Obr. 2.3: UML diagram modulu **Users** zobrazuje entitu **User** ako centrálny prvk systému a jej vzťahy na entity **Course**, **Task**, **Assignment** a **Participant**.

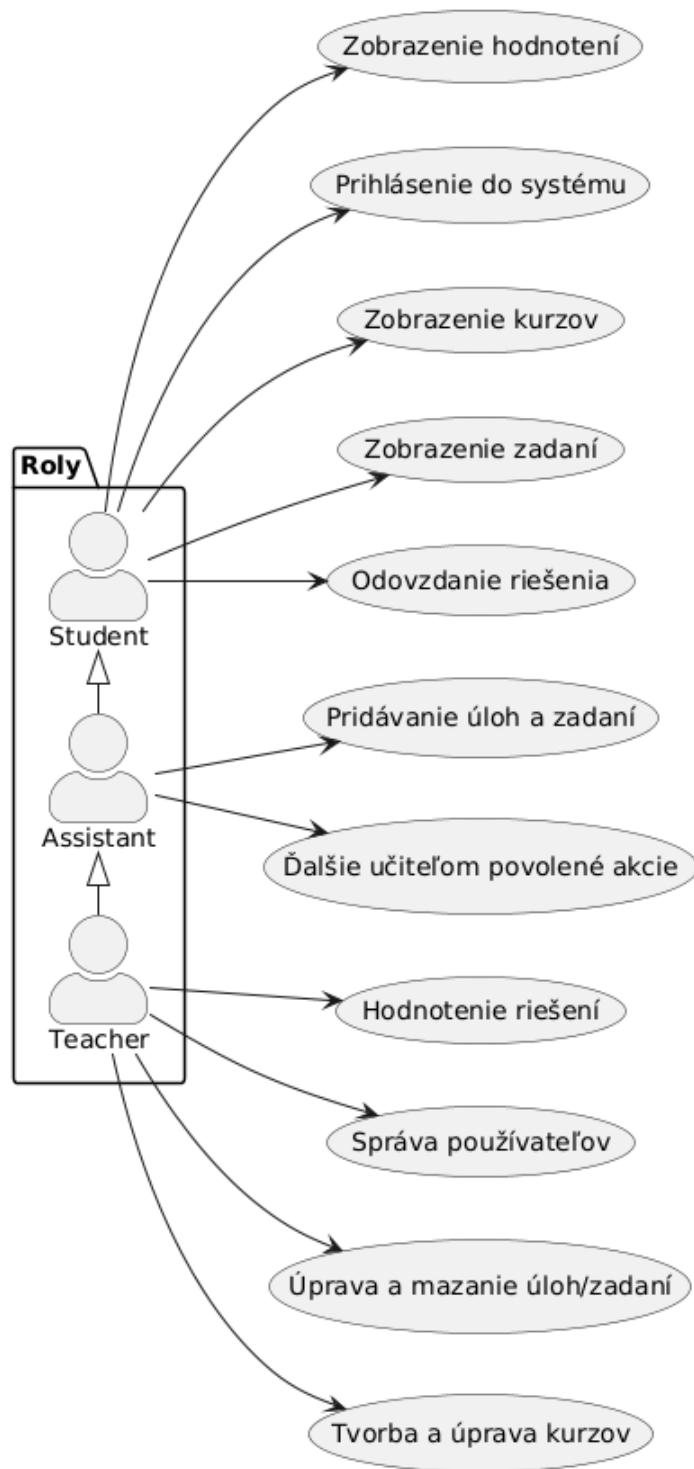
2.6.2 Roly

Prístup k jednotlivým časťam systému je riadený pomocou používateľských rolí. Každý používateľ má jednu z troch hlavných rolí: Student, Assistant alebo Teacher.

Hierarchia rolí je postavená dedičným spôsobom:

- **Assistant** automaticky zahŕňa všetky oprávnenia študenta.
- Rola **Teacher** zahŕňa oprávnenia asistenta a zároveň definuje ďalšie správcovské možnosti.

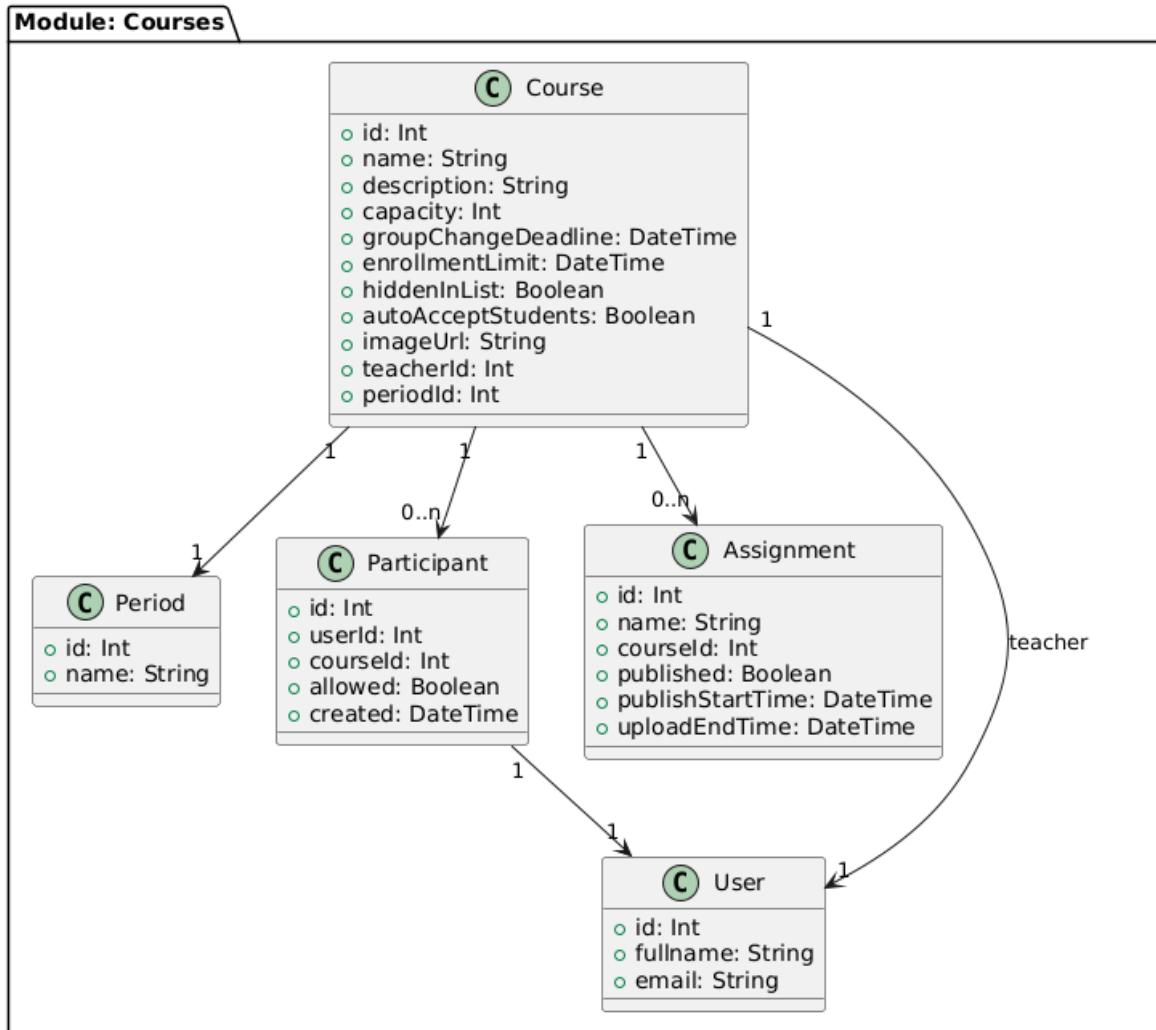
Učiteľ má možnosť **nastaviť**, ktoré učiteľské oprávnenia budú pridelené konkrétnemu asistentovi. Vďaka tomu môže byť rozsah asistentských práv prispôsobený podľa potreby - napr. umožniť len vytváranie úloh, ale nie ich hodnotenie. Asistentské oprávnenia sa pridelujú samostatne pre každý kurz, čo umožňuje jemne vyladiť rozsah práv jednotlivých používateľov v rôznych kurzoch. Jeden používateľ tak môže byť napríklad asistentom v jednom kurze a obyčajným študentom v inom.



Obr. 2.4: Use-case diagram znázorňuje základné oprávnenia jednotlivých rolí a ich hierarchiu. Učiteľ má prístup ku všetkým funkcionálitám systému a zároveň určuje, čo môžu asistenti vykonávať nad rámec študentských akcií.

2.6.3 Modul Courses

Modul **Courses** zabezpečuje správu kurzov, ich základných atribútov a priradovanie k časovým obdobiam (periods). Učiteľ má možnosť vytvárať, upravovať a odstraňovať kurzy, nastavovať ich kapacitu, dátumy pre uzávierku zápisu či viditeľnosť. Súčasťou modulu je aj evidencia účastníkov kurzu pomocou entít **Participant**, ako aj prepojenie na zadania (z modulu **Assignments**), ktoré sú ku kurzom priradené. Každý kurz je asociovaný s hlavným učiteľom predmetu. Tento používateľ je považovaný za vedúceho kurzu, čo však nevylučuje, že na výučbe sa môžu podieľať aj ďalší učitelia alebo asistenti s prístupovými právami.



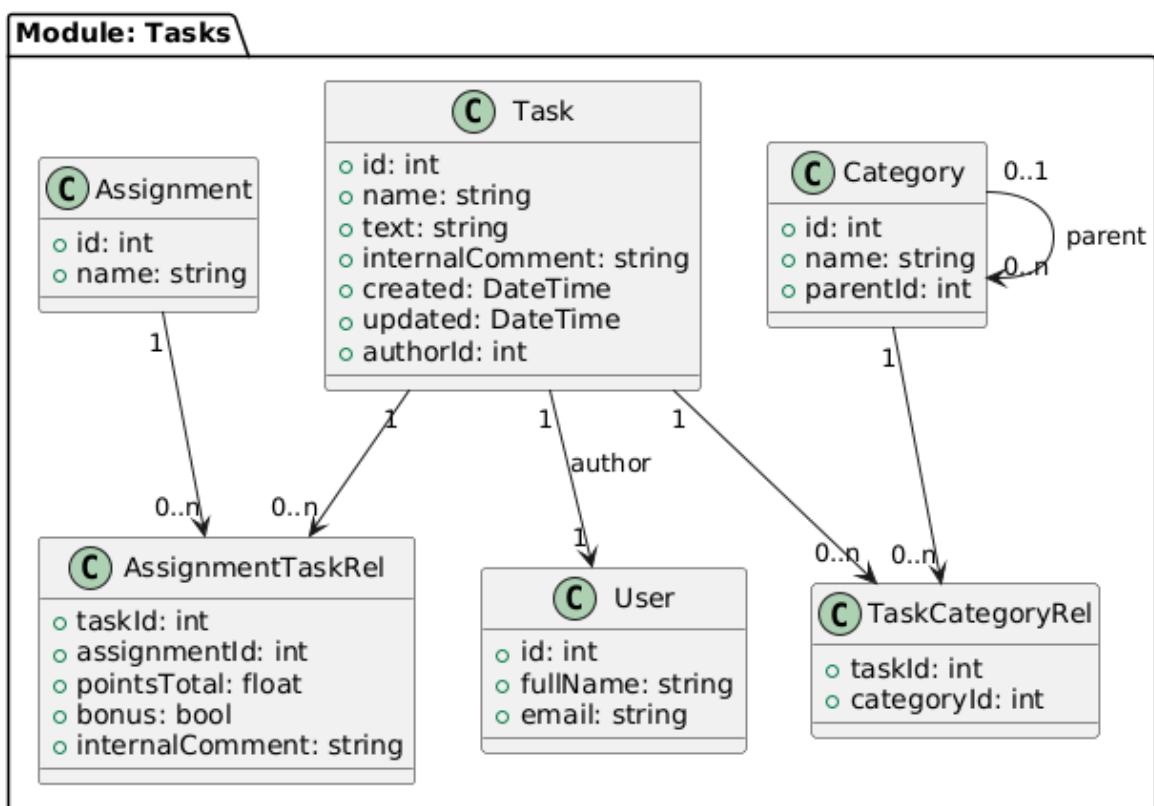
Obr. 2.5: Diagram tried znázorňujúci základné entity modulu Courses, ich atribúty a vzájomné vzťahy. Zahŕňa triedy Course, Period, Participant a externé napojenie na User a Assignment.

2.6.4 Modul Tasks

Modul Tasks je zodpovedný za správu jednotlivých úloh (taskov), ich kategorizáciu a priradenie k zadaniam v kurzoch. Každá úloha obsahuje okrem názvu a textového zadania aj metadáta ako dátum vytvorenia, interný komentár a informáciu o autrovi. Úlohy je možné zoskupiť do kategórií, ktoré sú hierarchicky usporiadane pomocou stromovej štruktúry.

Systém umožňuje, aby jedna úloha patrila do viacerých kategórií, čo je zabezpečené pomocou relácie TaskCategoryRel. Rovnako môže byť úloha použitá vo viacerých zadaniach (Assignment), čo zabezpečuje relácia AssignmentTaskRel, ktorá zároveň definuje pridelené body či bonusy.

Úlohy vytvárajú učitelia, pričom každý task obsahuje referenciu na autora. Prístup k úlohám je riadený na základe používateľskej roly.



Obr. 2.6: UML diagram modulu Tasks znázorňuje entity, väzby na zadania, kategórie a autora úlohy.

2.6.5 Používateľské rozhranie

Používateľské rozhranie systému L.I.S.T. je navrhnuté s dôrazom na jednoduchosť, prehľadnosť a rýchlosť orientáciu. Po úspešnom prihlásení je používateľ automaticky presmerovaný na úvodnú obrazovku, ktorá sa lísi v závislosti od jeho roly. Študen-

tom sa zobrazí študentské rozhranie so zoznamom kurzov, zadaniami a hodnoteniami. Učiteľom sa otvorí prostredie na správu kurzov, úloh a používateľov. Asistent má na výber, či chce vstúpiť do študentského rozhrania alebo do obmedzeného učiteľského rozhrania, ktoré mu poskytuje prístup len k vybraným funkcionálitám.

V tejto práci sa zameriavame najmä na používateľské rozhranie pre študenta, ktoré bolo predmetom návrhu a prototypovania v rámci implementačnej časti.

Na prihlásenie slúži samostatná prihlasovacia obrazovka, kde používateľ zadá e-mail a heslo (obr. 2.7). V prípade aktívneho zaškrtnutia možnosti „Zapamätať si ma“ sa token uloží do lokálneho úložiska pre zachovanie relácie.

The screenshot shows a login form titled "Vitajte v L.I.S.T.". It contains fields for "Email" (with the value "tvoj@email.sk") and "Heslo" (with five dots indicating the password). There is a checkbox labeled "Zapamätaj si ma". Below the form is a button labeled "PRIHLÁSIŤ SA" and a link labeled "Zabudnuté heslo?".

Obr. 2.7: Používateľ zadáva svoje prihlasovacie údaje do systému L.I.S.T. pomocou jednoduchého formulára.

Po prihlásení je používateľ presmerovaný na hlavnú stránku s prehľadom kurzov (obr. 2.8). Táto stránka je rozdelená na dve hlavné sekcie:

- **Tvoje kurzy** - kurzy, do ktorých je študent aktuálne zapísaný,
- **Ostatné kurzy** - dostupné kurzy, do ktorých sa môže študent zapísat, alebo čaká na potvrdenie učiteľom.

V hornej časti sa nachádza vyhľadávacie pole pre filtrovanie kurzov podľa názvu, prípadne vyučujúceho a dropdown na výber aktuálneho obdobia.

Student – Gregor Nezívý

Slovenčina ODHLÁSIŤ SA

Vyhľadaj kurzy

Obdobie

▼

Tvoje kurzy

| | | |
|--|--|---|
| | | |
| Programovanie 4 - Java LS 2025 Meno učiteľa AKTÍVNE | Operačné systémy LS 2024 Meno učiteľa AKTÍVNE | Programovacie paradigmy ZS 2024 Meno učiteľa AKTÍVNE |

Ostatné kurzy

| | | |
|---|---|--|
| | | |
| Programovanie 1 ZS 2025 Meno učiteľa PRIDAŤ SA | Programovanie 2 LS 2025 Meno učiteľa PRIDAŤ SA | Tvorba informačných systémov ZS 2024 Meno učiteľa PRIDAŤ SA |

Obr. 2.8: Po prihlásení sa študentovi zobrazia jeho kurzy a dostupné kurzy, ku ktorým sa môže pridať. Kurzy možno vyhľadávať a filtrovať podľa semestra.

Kliknutím na konkrétny kurz sa otvorí stránka s detailom kurzu, ktorá obsahuje postranný panel (sidebar) pre navigáciu medzi jednotlivými sekciami. Na obrázku 2.9 vidíme sekciu **Popis**, ktorá obsahuje názov kurzu, semester, učiteľa, kapacitu a podrobnejší opis s výstupmi kurzu.

| Student – meno Študenta | | Slovenčina | ODHLÁSIŤ SA |
|-------------------------|--|------------|-------------|
| Kurzy | | | |
| Popis | Názov kurzu / Semester YYYY | | |
| Úlohy | <p style="text-align: center;">Názov kurzu</p> <p>Kurz Názov kurzu poskytuje študentom prehľad rôznych prístupov k programovaniu, ktoré formujú spôsob, akým sa riešia úlohy v softvérovom vývoji. Cieľom je porozumieť výhodám a nevýhodám jednotlivých paradigm a naučiť sa ich aplikovať v praxi.</p> <p>Obsah kurzu:</p> <ul style="list-style-type: none"> • Téma 1 – základné princípy, príkazy, stav programu • Téma 2 – triedy, objekty, dedičnosť, polymorfizmus • Téma 3 – čisté funkcie, vyššie poradie funkcií, immutabilita • Téma 4 – deklaratívny prístup, pravidlá a fakty, Prolog • Téma 5 – výber vhodného prístupu pre konkrétny problém <p>Výstup z kurzu:</p> <ul style="list-style-type: none"> • Rozpoznať a použiť vhodnú paradigmu pre daný typ problému • Napísat jednoduché programy vo viacerých paradigmatických štýloch • Porozumieť výhodám kombinávania viacerých prístupov <p>Kurz je vedený interaktívne formou prednášok a cvičení, kde si študenti vyskúšajú jednotlivé koncepty na praktických príkladoch v rôznych programovacích jazykoch.</p> | | |
| Prehľad | <p>Informácie o kurze</p> <p>Učiteľ: Meno učiteľa Obdobie: Semester YYYY Kapacita: XX</p> | | |
| Projekty | | | |

Obr. 2.9: Stránka s popisom kurzu zobrazuje názov, semester, výstupy a základné informácie o kurze.

Sekcia **Úlohy** (obr. 2.10) poskytuje prehľad všetkých zadanií, ktoré sú rozdelené do typov zadaní zvolených učiteľom. Používateľ môže filtrovať úlohy podľa ich stavu (napr. odovzdané, otvorené, hodnotené). Pre každé zadanie sa zobrazuje jeho deadline, počet bodov a aktuálny stav.

Student - Gregor Nezivý

Slovenčina ODHLÁSIŤ SA

| Kurzy | Zadania | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|---|--------|----------|--|---------|----------|------|--------|----------|---------------------|-------|----------|----------|----------------------|-------|----------|---------|----------|------|--------|--------|----------------------|--------|----------|---------|----------|------|--------|-----------|----------------------|--------|----------|-----------|----------------------|--------|----------|
| Popis | Filtrovať podľa stavu: <input checked="" type="checkbox"/> VŠETKY <input type="checkbox"/> OHODNOTENÉ <input type="checkbox"/> ODOVZDANÉ <input type="checkbox"/> OTVORENÉ <input type="checkbox"/> ODOVZDANÉ NESKORO <input type="checkbox"/> NEODOVZDANÉ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Úlohy | Domáce úlohy 0 / 10 points <table border="1"> <thead> <tr> <th>Zadanie</th> <th>Deadline</th> <th>Body</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>Domáca 2</td> <td>4. 6. 2025 23:59:59</td> <td>- / 5</td> <td>Otvorené</td> </tr> <tr> <td>Domáca 1</td> <td>28. 5. 2025 23:59:59</td> <td>- / 5</td> <td>Otvorené</td> </tr> </tbody> </table> Skuška 0 / 30 points <table border="1"> <thead> <tr> <th>Zadanie</th> <th>Deadline</th> <th>Body</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>Skuška</td> <td>26. 5. 2025 23:59:59</td> <td>- / 30</td> <td>Otvorené</td> </tr> </tbody> </table> Midtermy 0 / 30 points <table border="1"> <thead> <tr> <th>Zadanie</th> <th>Deadline</th> <th>Body</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>Midterm 2</td> <td>21. 5. 2025 23:59:59</td> <td>- / 15</td> <td>Otvorené</td> </tr> <tr> <td>Midterm 1</td> <td>20. 5. 2025 23:59:59</td> <td>- / 15</td> <td>Otvorené</td> </tr> </tbody> </table> | | | | Zadanie | Deadline | Body | Status | Domáca 2 | 4. 6. 2025 23:59:59 | - / 5 | Otvorené | Domáca 1 | 28. 5. 2025 23:59:59 | - / 5 | Otvorené | Zadanie | Deadline | Body | Status | Skuška | 26. 5. 2025 23:59:59 | - / 30 | Otvorené | Zadanie | Deadline | Body | Status | Midterm 2 | 21. 5. 2025 23:59:59 | - / 15 | Otvorené | Midterm 1 | 20. 5. 2025 23:59:59 | - / 15 | Otvorené |
| Zadanie | Deadline | Body | Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Domáca 2 | 4. 6. 2025 23:59:59 | - / 5 | Otvorené | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Domáca 1 | 28. 5. 2025 23:59:59 | - / 5 | Otvorené | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Zadanie | Deadline | Body | Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Skuška | 26. 5. 2025 23:59:59 | - / 30 | Otvorené | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Zadanie | Deadline | Body | Status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Midterm 2 | 21. 5. 2025 23:59:59 | - / 15 | Otvorené | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Midterm 1 | 20. 5. 2025 23:59:59 | - / 15 | Otvorené | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Prehľad | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Projekty | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Obr. 2.10: Študent vidí zoznam zadaní, ich stav, deadline a bodové hodnotenie. Zoznam možno filtrovať podľa stavu úlohy.

Kliknutím na konkrétné zadanie v zozname sa otvorí samostatná stránka zadania (obr. 2.11). Tá obsahuje podrobne inštrukcie, zoznam všetkých úloh patriacich do daného zadania a sekciu na odovzdanie riešenia. Každá úloha má zobrazený svoj názov, text zadania, prípadne ilustráciu alebo očakávaný výstup. V dolnej časti sa nachádza komponent na nahratie súborov, kde študent pridá svoje riešenie a potvrdí jeho odovzdanie. Podporované sú viaceré formáty súborov a odovzdanie prebieha jedným kliknutím.

Student – Štefan Ignáč

Slovenčina ODHLÁSIŤ SA

| | |
|---|---|
| Kurzy Popis Úlohy Prehľad Projekty | <h3>Skúška</h3> <p>Inštrukcie:</p> <p>Vypracujte všetky úlohy podľa zadania. Použite vhodné programovacie jazyky a nástroje. Riešenie odovzdajte vo formáte špecifikovanom v požiadavkách. Doba na vypracovanie je 90 minút.</p> <h4>Správa procesov v operačnom systéme</h4> <p>Vytvorte program, ktorý pomocou systémového volania fork() vytvorí nový proces. Rodičovský proces počká na dokončenie potomka pomocou wait(). Vypíšte PID rodiča aj potomka a demonštrujte, ktorý kód sa vykonáva v ktorom procese.</p> <div style="text-align: center; margin-top: 10px;"> [Diagram procesov - vizualizácia fork() operácie] </div> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <pre>// Ukážka očakávaného výstupu: // Parent PID: 1234, Child PID: 1235 // Child process executing... // Parent waiting for child... // Child process finished</pre> </div> <div style="text-align: right; margin-top: 10px;">Vytvoril: meno Učiteľa</div> <h3>Mutexy</h3> <p>Napište program, ktorý pomocou zámku (napr. mutex) zabezpečí synchronizovaný prístup viacerých vláken k zdieľanej premennej. Spusťte aspoň dve vlákna, ktoré budú zvyšovať hodnotu spoločného čítača. Na konci vypíšte správny výsledok.</p> <div style="text-align: right; margin-top: 10px;">Vytvoril: meno Učiteľa</div> <h3>ODOVZDANIE ZADANIA</h3> <div style="border: 1px dotted #ccc; padding: 10px; text-align: center;"> <p>Nahrajte súbory s riešením (podporované formáty: .zip, .rar, .tar.gz, .c, .cpp, .py, .java)</p> <input type="button" value="Vybrať súbory"/> </div> <div style="text-align: right; margin-top: 10px; background-color: #ccc; padding: 5px;">ODOVZDAŤ RIEŠENIE</div> |
|---|---|

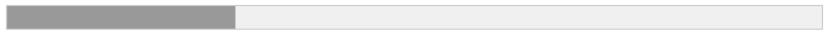
Obr. 2.11: Zadanie zobrazujúce všetky jeho úlohy. V dolnej časti stránky sa nachádza komponent na odovzdanie riešenia.

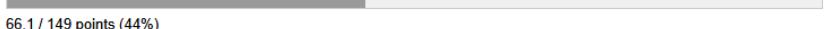
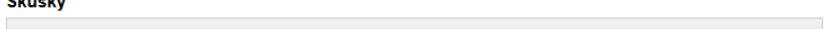
Táto obrazovka tvorí centrálny priestor na prácu so zadaním a umožňuje študentovi rýchlo pochopiť požiadavky, naplánovať riešenie a jednoducho ho nahrať do systému.

V sekcií **Prehľad** (obr. 2.12) má študent k dispozícii vizualizáciu celkového bodového progresu, rozdelenie bodov podľa kategórií, stav odovzdaní (napr. počet neodovzdaných či hodnotených úloh) vo forme koláčového grafu a zoznam najbližších deadline-ov. Táto obrazovka slúži na rýchle zorientovanie sa v tom, čo všetko má ešte študent splniť.

Student – Gregor Nezivý

Slovenčina ODHLÁSIŤ SA

| | | | |
|----------------|--|--|--|
| Kurzy | Prehľad kurzu | | |
| Popis | | | |
| Úlohy | | | |
| Prehľad | Celkový progress 83.1 / 294 points  28% | | |
| Projekty | | | |

| | |
|-----------------------------|--|
| Body podľa kategórií | |
| Domáce úlohy |  66.1 / 149 points (44%) |
| Skúšky |  0 / 60 points (0%) |
| Projekty |  17 / 55 points (31%) |

| | |
|---|--|
| Stav tvojich zadania | |
|  | |
| <input checked="" type="checkbox"/> Ohodnotené <input type="checkbox"/> Odovzdané <input type="checkbox"/> Otvorené <input type="checkbox"/> Odovzdané neskoro <input checked="" type="checkbox"/> Neodovzdané | |

| | |
|--|------------------|
| Prichádzajúce Deadliney | |
| Paging and TLB Due in 3 days | 20 points |
| Swapping Due in 7 days | 29 points |
| Threads and locks Due in 14 days | 18 points |
| Processes Due in 20 days | 15 points |
| Memory Management Due in 24 days | 20 points |

Obr. 2.12: Stránka s prehľadom zobrazuje celkový bodový stav, rozdelenie bodov podľa kategórií, stav zadaní a blížiace sa deadliney.

Toto rozhranie poskytuje študentovi všetky potrebné informácie a funkcie súvisiace s priebehom výučby a odovzdávaním zadaní. Rozloženie sekcií je konzistentné naprieč kurzmi, čím sa zabezpečuje jednoduchá orientácia a efektívne používanie aplikácie.

Kapitola 3

Implementácia

V tejto kapitole popisujeme proces vývoja systému. Zameriavame sa na praktickú stránku riešenia, vrátane spôsobu implementácie jednotlivých funkcií, riešenia problémov, ktoré sa počas vývoja vyskytli, ako aj na konkrétnu rozhodnutia prijaté počas tvorby systému.

3.1 Zvolený implementačný prístup

Na implementáciu systému sme použili štandardizovaný webový technologický stack. Frontend aplikácie je vytvorený v **Reacte**, zatiaľ čo backend je postavený na **ASP.NET Core** s využitím **Entity Framework Core** na prácu s databázou **PostgreSQL**. Klientska a serverová časť medzi sebou komunikujú cez **REST API** pomocou protokolu **HTTP**, pričom v produkčnom prostredí sa predpokladá využitie zabezpečeného prenosu cez **HTTPS**.

3.2 Štruktúra projektu

Implementácia systému L.I.S.T. je rozdelená na dve hlavné časti - frontend (klientská aplikácia v Reacte) a backend (serverová aplikácia v .NET Core). Obe časti sú štruktúrované prehľadne a logicky - backend podľa jednotlivých modulov systému a frontend podľa súvisiacich funkcií a používateľského rozhrania. Takýto prístup podporuje dobrú udržiavateľnosť, zrozumiteľnosť a jednoduchšie rozširovanie systému.

Celková organizačná štruktúra priečinkov oboch častí je znázornená na obrázku 3.1.



Obr. 3.1: Prehľad adresárovej štruktúry frontendu (vľavo) a backendu (vpravo). Každá časť je organizovaná podľa modulov a zodpovedností.

3.2.1 Frontend

Frontendová aplikácia je napísaná v TypeScripte pomocou Reactu a pozostáva zo zložky `src`, ktorá je ďalej členená podľa súvisiacich oblastí funkcionality, ako napríklad `Courses`, `Tasks`, `Assignments`, `Authentication` alebo `Users`.

Každá takáto oblasť obsahuje:

- **Components** - jednotlivé znovupoužiteľné vizuálne prvky ako karty, dialógy či filter bary,
- **Pages** - kompletné zobrazenia stránok (napr. zoznam úloh alebo editor úlohy),

- **Types** - dátové typy a rozhrania špecifické pre danú oblasť.

Okrem toho frontend obsahuje priečinok **services**, kde sa nachádzajú napríklad definície volaní na backend pomocou REST API, a priečinok **shared**, ktorý združuje spoločné komponenty, typy a pomocné funkcie využívané naprieč viacerými časťami aplikácie.

Takto zvolená organizácia prispieva k znovupoužiteľnosti, konzistentnosti a jedno-
duchej orientácii vývojárov v štruktúre projektu.

3.2.2 Backend

Backendová časť systému je implementovaná v jazyku C# pomocou frameworku ASP.NET Core. Backend je rozdelený do modulov podľa funkčnej oblasti. Každý modul (napr. **List.Courses**, **List.Users**, **List.Tasks**) predstavuje samostatný logický celok.

Každý modul zvyčajne obsahuje:

- **Controllers** - REST API endpointy, ktoré spracovávajú HTTP požiadavky,
- **Models** - entitné triedy mapujúce databázové tabuľky,
- **DTOs** - dátové prenosové objekty používané pri komunikácii medzi frontendom a backendom,
- **Data** - kontext databázy pre daný modul,
- **Migrations** - databázové migrácie vytvorené pomocou Entity Frameworku.

Takýto modulárny prístup umožňuje nezávislý vývoj jednotlivých častí systému a zároveň lepšie oddelenie zodpovedností.

3.3 Používateelia a roly

3.3.1 Prehľad modulu

Modul **List.Users** tvorí základnú súčasť systému, ktorá sa stará o správu používateľov, ich autentifikáciu, registráciu a pridelenie rolí. Zabezpečuje všetky kľúčové operácie súvisiace s používateľskými účtami - od vytvárania nových používateľov, cez prihlásование, až po administráciu údajov a import viacerých účtov naraz.

V systéme sú podporované rôzne roly používateľov - najmä **študent** a **učiteľ**, pričom každá z nich definuje oprávnenia k jednotlivým časťam systému. Okrem týchto hlavných rolí existuje aj špeciálna role **asistent**, ktorá kombinuje možnosti oboch spomenutých typov používateľov. Asistent môže vystupovať ako študent (napr. byť

účastníkom kurzov), ale zároveň môže mať zverené vybrané kompetencie podobné učiteľovi - napríklad možnosť spravovať zadania alebo študentov v rámci konkrétnych kurzov.

Rola asistenta je navrhnutá ako flexibilný rozširujúci prvok systému. Do budúcnosti je systém pripravený na pridanie správy oprávnení, kde bude možné presne určiť, ku ktorým kurzom, úlohám alebo časťam systému má daný asistent prístup. Týmto spôsobom možno podporiť zdieľanie vyučovacích povinností medzi viacerými osobami bez potreby plného učiteľského oprávnenia.

Medzi hlavné funkcionality modulu patria:

- registrácia a autentifikácia používateľov pomocou e-mailu a hesla,
- generovanie a validácia JWT tokenov pre zabezpečený prístup k API,
- správa používateľských údajov - vrátane aktualizácie a vymazania účtov,
- pridávanie nových používateľov individuálne alebo hromadne (CSV import),
- pridelovanie a zmena roly používateľa zo strany učiteľa.

Architektúra modulu je navrhnutá tak, aby bola rozšíriteľná ľahko a bez závažných zásahov do ostatných častí systému. Modul využíva samostatný databázový kontext `UsersDbContext`, ktorý obsluhuje entitu používateľa a súvisiace operácie. Modul zároveň poskytuje servisnú vrstvu (`UserService`), ktorá abstrahuje logiku nad databázou a zjednodušuje implementáciu controllerov.

Závislosti modulu sú smerované na `List.Common`, kde sa nachádzajú spoločné funkcionality (napr. nahrávanie súborov na server), a na `List.Logs`, ktorý zabezpečuje zaznamenávanie udalostí. Rovnako ako ostatné moduly, aj `List.Users` je registrovaný v hlavnej aplikácii `List.Server`, kde je integrovaný do celej API vrstvy systému.

3.3.2 Dátové modely

Modul `List.Users` definuje hlavnú entitu `User`, ktorá reprezentuje každého používateľa systému - bez ohľadu na jeho rolu. Všetky informácie potrebné pre autentifikáciu, autorizáciu a základnú identifikáciu používateľa sú uchovávané v tejto triede. Model sa nachádza v priečinku `List.Users.Models` a je mapovaný na databázovú tabuľku `Users` pomocou atribútu `[Table("Users")]`.

User

Trieda `User` obsahuje identifikátor používateľa, časové pečiatky, autentifikačné údaje (e-mail, heslo) a rolu, ktorá určuje oprávnenia používateľa v systéme.

```

1 [Table("Users")]
2 public class User
3 {
4     [Key]
5     public int Id { get; set; }
6     public DateTime Created { get; set; } = DateTime.UtcNow;
7     public DateTime Updated { get; set; } = DateTime.UtcNow;
8     [Required]
9     public string Fullname { get; set; }
10    [Required]
11    [EmailAddress]
12    public string Email { get; set; }
13
14    [Required]
15    public string Password { get; set; }
16    public string? PasswordToken { get; set; }
17
18    [Required]
19    [JsonConverter(typeof(JsonStringEnumConverter))]
20    public UserRole Role { get; set; }
21 }
```

Popis atribútov:

- **Id** - unikátny identifikátor používateľa, primárny kľúč databázovej tabuľky.
- **Created**, **Updated** - dátumy vytvorenia a poslednej aktualizácie záznamu; automaticky nastavované pri vytváraní objektu.
- **Fullname** - celé meno používateľa, povinný údaj.
- **Email** - e-mailová adresa používateľa, unikátny a povinný údaj, validovaný pomocou atribútu **[EmailAddress]**.
- **Password** - hash hesla používateľa, uložený ako povinný reťazec.
- **PasswordToken** - voliteľný token využiteľný pri obnove hesla.
- **Role** - rola používateľa, ktorá je uložená ako enumerácia **UserRole** a serializovaná do textovej podoby v JSON odpovediach.

Vďaka anotácii **JsonStringEnumConverter** sa hodnota roly serializuje ako reťazec (napr. "Teacher") namiesto číselnej hodnoty enumerácie, čo zvyšuje čitateľnosť JSON odpovedí a zjednoduší frontendovú prácu.

UserRole

Typ `UserRole` je enumeračný typ (`enum`), ktorý definuje možné roly používateľa v systéme. Tieto roly sú priamo využívané na riadenie prístupu k jednotlivým časťam aplikácie (napr. úprava kurzov, správa úloh, odovzdávanie riešení).

```

1 public enum UserRole {
2     Teacher,
3     Assistant,
4     Student
5 }
```

Popis rolí:

- **Teacher** - rola vyučujúceho, má plný prístup k správe kurzov, úloh, účastníkov, hodnotení a ďalších
- **Assistant** - rola asistenta, ktorá je navrhnutá ako flexibilná kombinácia vlastností študenta a učiteľa. Umožňuje, aby bol používateľ zároveň účastníkom kurzov aj aktívnym pomocníkom pri ich správe. Hoci aktuálne systém nepodporuje podrobné priradzovanie oprávnení, architektúra je na túto možnosť pripravená a umožňuje jej budúce doplnenie podľa potrieb vyučujúcich.
- **Student** - bežný používateľ, ktorý sa prihlásuje na kurzy, zobrazuje zadania, odovzdáva riešenia a prehliada si svoje výsledky.

Tento dátový model tvorí základ identifikačnej a autentifikačnej vrstvy systému. Je priamo využívaný v procese registrácie, prihlásovania aj pri pridelovaní oprávnení pri jednotlivých REST API volaniah.

3.3.3 Databázový kontext

Modul `List.Users` využíva vlastný databázový kontext `UsersDbContext`, ktorý je zodpovedný za správu perzistentných údajov o používateľoch. Tento kontext dedí od triedy `DbContext` (Entity Framework Core) a definuje jedinú entitu `Users`, ktorá reprezentuje tabuľku používateľov v databáze.

```

1 public class UsersDbContext(DbContextOptions<UsersDbContext> options) : DbContext(options)
2 {
3     public DbSet<User> Users { get; set; }
4
5     protected override void OnModelCreating(ModelBuilder modelBuilder)
6     {
7         base.OnModelCreating(modelBuilder);
```

```

8
9     modelBuilder.Entity<User>()
10        .HasIndex(u => u.Email)
11        .IsUnique();
12    }
13 }

```

Okrem základnej konfigurácie databázovej entity sa v kontexte nachádza aj explícitné nastavenie unikátneho indexu nad stĺpcom `Email`, ktoré zabezpečuje, že v systéme nemôže existovať viacero používateľov s rovnakou e-mailovou adresou. Táto vlastnosť je dôležitá najmä pri registrácii a autentifikácii používateľov.

3.3.4 API endpointy

Modul `List.Users` definuje REST API endpointy, ktoré zabezpečujú prístup k funkcionality správy používateľov a autentifikácie. Tieto endpointy sú rozdelené do dvoch hlavných častí - `AuthController` (autentifikácia) a `UsersController` (administrácia používateľov). Ich prehľad je znázornený na obrázku 3.2.

| Users | |
|--------|---------------------|
| GET | /api/users/teachers |
| GET | /api/users |
| POST | /api/users |
| PUT | /api/users/{id} |
| DELETE | /api/users/{id} |
| POST | /api/users/import |
| Auth | |
| POST | /api/auth/register |
| POST | /api/auth/login |

Obr. 3.2: Zoznam všetkých REST API endpointov definovaných v rámci modulu `Users`.

Autentifikačné endpointy

- POST /api/auth/register - slúži na registráciu nového používateľa. Tento endpoint je zabezpečený a dostupný len pre používateľov s rolou Teacher (vďaka atribútu [Authorize(Roles = "Teacher")]).
- POST /api/auth/login - verejne dostupný endpoint, ktorý autentifikuje používateľa na základe e-mailu a hesla. V prípade úspechu vráti JWT token, ktorý klient následne používa pri autorizovaných požiadavkách.

Administratívne endpointy

- GET /api/users/teachers - vráti zoznam všetkých učiteľov v systéme.
- GET /api/users - stránkovaný výpis všetkých používateľov s možnosťou vyhľadávania.
- POST /api/users - vytvorenie nového používateľa (napr. ako admin).
- PUT /api/users/{id} - aktualizácia existujúceho používateľa podľa jeho ID.
- DELETE /api/users/{id} - odstránenie používateľa z databázy.
- POST /api/users/import - hromadný import viacerých používateľov zo súboru vo formáte CSV.

Všetky vyššie uvedené endpointy sú zabezpečené pomocou atribútu [Authorize(Roles = "Teacher")], čím je zabezpečené, že prístup k nim majú len oprávnení používateľia. Autentifikácia prebieha cez JWT token, ktorý klient odosielá v hlavičke Authorization pri každej požiadavke:

`Authorization: Bearer <token>`

Ukážka implementácie: registrácia používateľa Endpoint POST /api/auth/register slúži na vytvorenie nového používateľského účtu. Najprv sa overuje platnosť vstupných údajov a existencia zadaneho e-mailu v databáze. Ak sú dáta v poriadku, heslo sa bezpečne zahashuje pomocou knižnice BCrypt.Net, čím sa zabezpečí jeho ochrana.

Následne sa vytvorí nový objekt používateľa so zadanými údajmi a predvolenou rolou Student, ak nebola špecifikovaná. Po úspešnom uložení sa vracia odpoveď s ID nového účtu. Prístup k tomuto volaniu majú len používateľia s rolou Teacher, čo zabezpečuje kontrolu nad registráciou.

```

1  [HttpPost("register")]
2  [Authorize(Roles = "Teacher")]
3  public async Task<IActionResult> Register(RegisterRequest request)
4  {

```

```

5     if (!ModelState.IsValid)
6         return BadRequest(ModelState);
7
8     if (await _context.Users.AnyAsync(u => u.Email == request.Email))
9         return BadRequest("Email already in use.");
10
11    var hashedPassword = BCrypt.Net.BCrypt.HashPassword(request.Password);
12
13    var user = new User
14    {
15        Fullname = request.Fullname,
16        Email = request.Email,
17        Password = hashedPassword,
18        Role = request.Role ?? UserRole.Student
19    };
20
21    _context.Users.Add(user);
22    await _context.SaveChangesAsync();
23
24    return Ok(new { message = "User registered", user.Id });
25 }
```

Bezpečné hashovanie hesiel Heslá používateľov sa nikdy neukladajú v čitateľnej podobe. Na ich hashovanie sa používa knižnica BCrypt.Net, ktorá využíva adaptívny algoritmus bcrypt a náhodný salt. Tento prístup výrazne zvyšuje bezpečnosť proti útokom ako brute-force alebo rainbow table.

Pri overovaní hesla pri prihlásovaní sa hash kontroluje pomocou:

```
BCrypt.Net.BCrypt.Verify(request.Password, user.Password)
```

Generovanie JWT tokenu Po úspešnom prihlásení je používateľovi vygenerovaný JWT token, ktorý obsahuje základné údaje o používateľovi (ID, meno, e-mail, rola). Token má definovanú platnosť (napr. 24 hodín, alebo 14 dní keď používateľ označí „zapamätať si“).

Tento systém zabezpečuje plnohodnotnú autentifikáciu a správu používateľov pomocou moderných bezpečnostných štandardov a dobre štruktúrovaného REST API.

3.3.5 Správa používateľov - učiteľské rozhranie

Používatelia s rolou Teacher majú v systéme prístup k administrácii všetkých používateľských účtov. V špeciálnej sekcií frontendovej aplikácie môžu prehliadať zoznam

účtov, filtrovať ich podľa mena alebo e-mailu, upravovať údaje, vymazávať používateľov a importovať nové účty hromadne cez CSV súbor. Taktiež majú možnosť pridávať nových používateľov cez dialógové okno.

| Názov | Email | Role | Akcie |
|------------|------------------------|-----------|-------|
| Asistent 2 | Assistant2@example.com | Assistant | |
| Asistent 3 | Asistent3@example.com | Assistant | |
| Asistent 4 | Asistent4@example.com | Assistant | |
| Asistent 5 | Asistent5@example.com | Assistant | |
| Učitel 1 | Teacher1@example.com | Teacher | |
| Učitel 2 | Teacher2@example.com | Teacher | |
| Student 1 | Student@example.com | Student | |
| Student 2 | Student2@example.com | Student | |
| Student 3 | Student3@example.com | Student | |
| Student 4 | Student4@example.com | Student | |

Rows per page: 10 ▾ 1–10 of 13 < >

Obr. 3.3: Používateľské rozhranie pre učiteľa - tabuľka s akciami.

Každý záznam obsahuje meno používateľa, e-mail a jeho rolu. V stĺpci akcií sú k dispozícii tlačidlá pre úpravu, nastavenie asistentských oprávnení (v prípade, že ide o asistenta) a zmazanie účtu. Aj keď zatiaľ nie sú asistentské oprávnenia plne implementované, dialógové okno pre ich nastavenie je už pripravené a otvorí sa po vytvorení alebo úprave asistenta.

Upraviť používateľa

Meno a priezvisko *

Asistent 2

E-mail *

Assistant2@example.com

Rola

Teacher

Assistant

Student

ZRUŠIŤ ULOŽIŤ ZMENY

Obr. 3.4: Dialógové okno na úpravu používateľa - meno, e-mail, rola.

Po kliknutí na tlačidlo pre úpravu sa zobrazí formulár s predvyplnenými údajmi používateľa. V ňom môže učiteľ upraviť meno, e-mail a rolu. Po potvrdení zmien sa údaje odošlú na backend pomocou HTTP PUT požiadavky.

Ukážka interakcie s API pri úprave používateľa Nasledujúci úryvok ukazuje implementáciu funkcie, ktorá spracuje uloženie zmien z dialógu `EditUserDialog`. Dáta sa odošlú na server, používateľ je upozornený notifikáciou a v prípade asistenta sa otvorí dodatočný dialóg pre správu oprávnení.

```

1  <EditUserDialog
2      isOpen={isEditDialogOpen}
3      user={selectedUser}
4      onSubmit={async (updatedUser) => {
5          try {
6              await api.put(`/users/${updatedUser.id}`, {
7                  fullName: updatedUser.fullname,
8                  email: updatedUser.email,
9                  role: updatedUser.role
10             });
11             showNotification("Zmeny boli uložené", "success");
12
13             if (updatedUser.role === "Assistant") {
14                 setSelectedAssistant(updatedUser);

```

```

15         setScopeDialogOpen(true);
16     }
17
18     reload();
19 } catch (error) {
20     showNotification("Nepodarilo sa uložiť zmeny", "error");
21 }
22 }
23 onClose={() => setEditDialogOpen(false)}
24 >

```

Táto ukážka kódu znázorňuje moderný prístup k správe údajov na strane klienta. Po prijatí aktualizovaných údajov z formulára odošle HTTP PUT požiadavku na server, pričom aktualizuje meno, e-mail a rolu používateľa. Po úspešnom uložení zmien sa zobrazí notifikácia o úspechu a obnoví sa zoznam používateľov prostredníctvom funkcie `reload()`, aby sa zobrazili aktuálne údaje. Ak bola zvolená role `Assistant`, otvorí sa navyše dialóg na správu asistentských oprávnení, ktorý môže byť v budúcnosti rozšírený o konkrétnie priradenia k vybraným kurzom alebo učiteľom.

Komponent využíva vlastný API wrapper pre HTTP volania, centrálnie konfigurovaný systém notifikácií pomocou React Contextu a zároveň reaguje na výber používateľa tým, že si lokálne uchováva jeho stav v `selectedUser`.

3.4 Kurzy (Courses)

3.4.1 Prehľad modulu

Modul `List.Courses` predstavuje serverovú časť systému zodpovednú za správu kurzov a všetkých súvisiacich entít. Jeho hlavnou úlohou je zabezpečiť komplettnú funkcionalitu súvisiacu s evidenciou kurzov vrátane ich vytvárania, aktualizácie, mazania, správy parametrov a priradenia vyučujúcich a študentov.

Medzi kľúčové funkcionality modulu patrí:

- vytváranie nových kurzov učiteľmi,
- zápis a evidencia študentov ako účastníkov kurzov,
- správa parametrov kurzu, ako je popis, kapacita dostupnosť, prihlásovacie obmedzenia a termíny,
- podpora automatického alebo ručného schvaľovania zápisu študentov.

Modul zároveň definuje štruktúru dát pre ukladanie kurzov a účastníkov do databázy, vystavuje REST API pre interakciu s klientskou aplikáciou a zabezpečuje validáciu a spracovanie požiadaviek podľa prístupových práv používateľov.

Modul má závislosť na projektoch `List.Users`, `List.Common` a `List.Logs`, ktoré zabezpečujú napríklad prístup k informáciám o používateľoch, spoločné dátové typy a logovanie akcií. Zároveň je tento modul referencovaný v hlavnom projekte `List.Server`, ktorý integruje všetky moduly systému do jednej spustiteľnej webovej aplikácie.

3.4.2 Dátové modely

Modely sa nachádzajú v priečinku `List.Courses.Models`. Modul `Courses` definuje tri základné entity: `Course`, `Participant` a `Period`, ktoré reprezentujú hlavné dátové objekty súvisiace so správou kurzov.

Course

Trieda `Course` reprezentuje jeden kurz v systéme. Obsahuje množstvo vlastností, ktoré popisujú základné parametre kurzu, jeho správanie, vzťahy k iným entitám a zobrazenie v systéme.

Na začiatku je definovaný identifikátor `Id`, ktorý je označený ako primárny kľúč a slúži na jednoznačné určenie každého kurzu v databáze. Vlastnosť `Name` obsahuje názov kurzu a je povinná, čo zabezpečuje atribút `[Required]`. Tento atribút garantuje, že pri vytváraní alebo aktualizácii kurzu musí byť uvedené jeho meno.

Ďalej nasleduje vlastnosť `Capacity`, ktorá určuje maximálny počet študentov, ktorí sa môžu na daný kurz zapísť. Hodnoty `GroupChangeDeadline` a `EnrollmentLimit` sú voliteľné dátumy. Prvá z nich definuje termín, do ktorého môžu študenti meniť svoju skupinu, a druhá určuje posledný možný dátum zápisu do kurzu.

Logická vlastnosť `HiddenInList` indikuje, či sa kurz zobrazuje študentom v zo-zname dostupných kurzov. Ak je táto hodnota nastavená na `true`, kurz nie je zobrazený bežným používateľom. Vlastnosť `AutoAcceptStudents` určuje režim akceptovania študentov: ak je nastavená na `true`, zápisu sú automaticky schvaľované; ak je `false`, musí zápis potvrdiť vyučujúci.

Atribút `ImageUrl` obsahuje voliteľný odkaz na titulný obrázok kurzu, ktorý sa môže zobraziť v používateľskom rozhraní. Vlastnosť `Description` slúži na textový (HTML-formátovaný) opis kurzu, ktorý je taktiež nepovinný.

Každý kurz má prideleného učiteľa, ktorý ho vytvoril a spravuje. Táto väzba je vyjadrená pomocou vlastnosti `TeacherId`, ktorá uchováva identifikátor vyučujúceho. Navigačná vlastnosť `Teacher` je zároveň označená ako cudzí kľúča cez atribút `[ForeignKey("TeacherId")]`, čo umožňuje prístup k celému objektu typu `User` predstavujúceho učiteľa.

Podobne kurz patrí do jedného časového obdobia. Vlastnosť `PeriodId` uchováva identifikátor obdobia. Vlastnosť `Period` predstavuje navigačnú väzbu na objekt obdobia a je označená ako cudzí kľúč cez `[ForeignKey("PeriodId")]`.

Nakoniec trieda obsahuje kolekciu `Participants`, ktorá reprezentuje všetkých účastníkov zapísaných na daný kurz. Táto väzba zabezpečuje jednoduchý prístup k zoznamu študentov priradených ku konkrétnemu kurzu.

```

1  public class Course
2  {
3      [Key]
4      public int Id { get; set; }
5      [Required]
6      public string Name { get; set; } = string.Empty;
7      [Required]
8      public int Capacity { get; set; }
9      public DateTime? GroupChangeDeadline { get; set; }
10     public DateTime? EnrollmentLimit { get; set; }
11     [Required]
12     public bool HiddenInList { get; set; }
13     [Required]
14     public bool AutoAcceptStudents { get; set; }
15     public string? ImageUrl { get; set; }
16     public string? Description { get; set; }
17
18     public int TeacherId { get; set; }
19     [ForeignKey("TeacherId")]
20     public User Teacher { get; set; } = null!;
21
22     public int? PeriodId { get; set; }
23     [ForeignKey("PeriodId")]
24     public Period Period { get; set; } = null!;
25
26     public ICollection<Participant> Participants { get; set; };
27 }
```

Participant

Trieda `Participant` reprezentuje vzťah medzi používateľom a kurzom. Záznam tejto entity vznikne vtedy, keď sa študent prihlási do kurzu, ale ešte nemusí byť schválený.

Každý záznam má svoj unikátny identifikátor. Obsahuje identifikátor študenta a

kurzu, čím vytvára väzbu medzi týmito dvoma entitami. Navigačné vlastnosti umožňujú priamy prístup k údajom o používateľovi aj o kurze.

Súčasťou záznamu je aj vlastnosť `Created`, ktorá uchováva dátum a čas vytvorenia záznamu. Vlastnosť `Allowed` vyjadruje, či bol študent akceptovaný do kurzu. V predvolenom stave je táto hodnota nastavená na `false`, čo znamená, že študent je v stave „čakajúci na schválenie“.

Period

Trieda `Period` reprezentuje časové obdobie, ku ktorému môže byť kurz priradený. Typickým príkladom takého obdobia je semester alebo školský rok.

Obsahuje identifikátor a názov obdobia. Okrem toho zahŕňa kolekciu všetkých kurzov, ktoré do daného obdobia patria. Táto väzba umožňuje jednoduché filtrovanie a prácu s kurzami podľa časového zaradenia.

3.4.3 Databázový kontext

Modul `Courses` využíva vlastný databázový kontext `CoursesDbContext`, ktorý dedí z triedy `DbContext` v rámci frameworku Entity Framework Core. Tento kontext definiuje prístup k databázovým entitám `Course`, `Participant` a `Period` prostredníctvom vlastností typu `DbSet`, čím umožňuje CRUD operácie nad nimi.

V rámci metódy `OnModelCreating` sú nakonfigurované jednotlivé entity, ich väzby a správanie pri operáciách ako mazanie, či tvorba indexov. Ako príklad uvádzam konfiguráciu entity `Participant`, ktorá reprezentuje prepojenie medzi používateľom a kurzom:

```

1 modelBuilder.Entity<Participant>(entity =>
2 {
3     entity.ToTable("participants");
4     entity.HasKey(e => e.Id);
5
6     entity.HasOne(e => e.User)
7         .WithMany()
8         .HasForeignKey(e => e.UserId)
9         .OnDelete(DeleteBehavior.Cascade);
10
11    entity.HasOne(e => e.Course)
12        .WithMany(c => c.Participants)
13        .HasForeignKey(e => e.CourseId)
14        .OnDelete(DeleteBehavior.Cascade);
15

```

```
16     entity.HasIndex(e => new { e.UserId, e.CourseId }).IsUnique();  
17 };
```

Tento kód definuje:

- prepojenie účastníka s používateľom a kurzom cez cudzie kľúče,
- správanie pri mazaní (**Cascade**) - ak sa vymaže používateľ alebo kurz, automaticky sa vymažú aj príslušné záznamy o účasti užívateľov v kurze,
- unikátny index, ktorý zabezpečuje, že sa ten istý používateľ nemôže zapísť do rovnakého kurzu viackrát.

Podobne sú v tejto metóde nakonfigurované aj ostatné entity, napríklad správanie medzi kurzom a učiteľom, alebo medzi kurzom a obdobím. Všetky tieto pravidlá zabezpečujú integritu dát a správne správanie systému pri operáciách s databázou.

3.4.4 API endpointy

Modul `Courses` definuje REST API endpointy, ktoré umožňujú klientským aplikáciám komunikovať so serverom prostredníctvom protokolu HTTP. Endpointy sú implementované v triedach `CourseController`, `ParticipantController` a `PeriodController`, pričom každá trieda sa stará o konkrétnu entitu systému.

Typický požiadavkový cyklus zahŕňa:

- odoslanie požiadavky z klienta na konkrétny URL adresu (napr. `/api/courses`),
- spracovanie požiadavky na serveri - validácia vstupov, práca s databázou,
- odoslanie odpovede klientovi vo forme JSON objektu s príslušným HTTP statu-
- som.

Zoznam všetkých dostupných endpointov pre kurzové API znázorňuje obrázok 3.5.

| Course | |
|-------------|-------------------------------------|
| GET | /api/courses |
| POST | /api/courses |
| GET | /api/courses/student-visible |
| GET | /api/courses/{id} |
| DELETE | /api/courses/{id} |
| PUT | /api/courses/{id} |
| PUT | /api/courses/{id}/description |
| Participant | |
| GET | /api/participants/mine |
| POST | /api/participants |
| GET | /api/participants/course/{courseId} |
| PATCH | /api/participants/approve |
| DELETE | /api/participants/remove |
| Period | |
| GET | /api/periods |
| POST | /api/periods |
| DELETE | /api/periods/{id} |
| PUT | /api/periods/{id} |

Obr. 3.5: Zoznam všetkých dostupných REST API endpointov v rámci modulu Courses.

- GET /api/courses - vráti zoznam všetkých kurzov vrátane základných informácií o učiteľovi a období.
- POST /api/courses - vytvorí nový kurz.
- GET /api/courses/student-visible - vráti len kurzy viditeľné pre študentov.
- GET /api/courses/{id} - detailné informácie o konkrétnom kurze.
- DELETE /api/courses/{id} - odstráni kurz.
- PUT /api/courses/{id} - aktualizuje vlastnosti kurzu.
- PUT /api/courses/{id}/description - aktualizuje popis kurzu.

Podobne aj pre účastníkov (Participant) a obdobia (Period) existujú špecifické endpointy, ktoré obsluhujú vytváranie, získavanie a mazanie záznamov, ktoré tiež môžeme vidieť na obrázku 3.5.

Ukážka implementácie konkrétneho endpointu

Ako príklad si ukážeme endpoint GET `/api/periods`, ktorý vráti všetky obdobia evidované v systéme spolu s počtom kurzov, ktoré do nich patria. Endpoint je verejne dostupný pre všetkých autentifikovaných používateľov.

Prístup k jednotlivým endpointom je zároveň zabezpečený podľa rolí používateľa. Napríklad študent má prístup iba k čítaniu údajov (napr. GET `/api/courses`), ale nemôže vytvárať, upravovať alebo mazať kurzy. Naopak, učiteľ alebo asistent môže využívať aj požiadavky typu POST, PUT a DELETE pri práci so svojimi kurzami.

```

1  [HttpGet]
2  public async Task<IActionResult> GetAll()
3  {
4      var periods = await _context.Periods
5          .Select(p => new PeriodReadDto
6          {
7              Id = p.Id,
8              Name = p.Name,
9              CourseCount = p.Courses.Count()
10         })
11         .ToListAsync();
12
13     return Ok(periods);
14 }
```

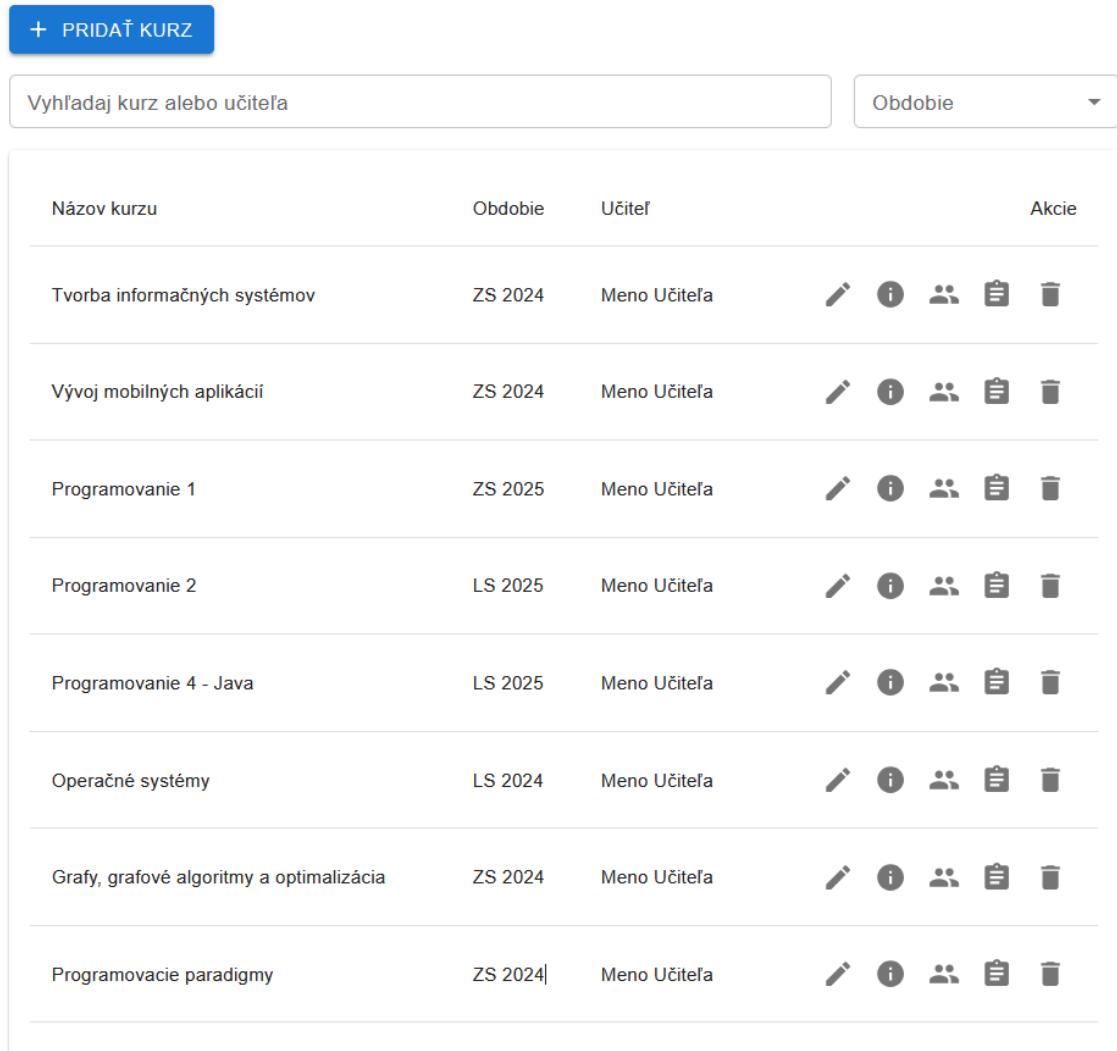
Logika je jednoduchá: cez databázový kontext `_context` sa dotazujeme na všetky záznamy z tabuľky `Periods`, pričom zároveň zistíme počet kurzov pre každé obdobie. Tieto údaje sú premietnuté do DTO triedy `PeriodReadDto` a odoslané klientovi ako odpoveď vo formáte JSON.

Táto štruktúra umožňuje jednoduché a efektívne získavanie agregovaných informácií, ktoré sú priamo využiteľné vo frontende, napríklad pre výpis filtrov alebo štatistiky.

3.4.5 Učiteľské rozhranie pre správu kurzov

Učiteľské rozhranie poskytuje vyučujúcim nástroj na správu ich kurzov. Po prihlásení sa učiteľovi zobrazí prehľad všetkých existujúcich kurzov, pričom má možnosť každý kurz upravovať alebo spravovať jeho súvisiace časti (obrázok 3.6).

Kurzy



The screenshot shows a user interface for managing courses. At the top, there is a blue button labeled '+ PRIDAŤ KURZ'. Below it is a search bar with the placeholder 'Vyhľadaj kurz alebo učiteľa' and a dropdown menu labeled 'Obdobie'. The main area is a table with columns: 'Názov kurzu', 'Obdobie', 'Učiteľ', and 'Akcie'. The 'Akcie' column contains icons for editing, viewing details, adding students, marking as completed, and deleting.

| Názov kurzu | Obdobie | Učiteľ | Akcie |
|--|---------|--------------|-------|
| Tvorba informačných systémov | ZS 2024 | Meno Učiteľa | |
| Vývoj mobilných aplikácií | ZS 2024 | Meno Učiteľa | |
| Programovanie 1 | ZS 2025 | Meno Učiteľa | |
| Programovanie 2 | LS 2025 | Meno Učiteľa | |
| Programovanie 4 - Java | LS 2025 | Meno Učiteľa | |
| Operačné systémy | LS 2024 | Meno Učiteľa | |
| Grafy, grafové algoritmy a optimalizácia | ZS 2024 | Meno Učiteľa | |
| Programovacie paradigmy | ZS 2024 | Meno Učiteľa | |

Obr. 3.6: Prehľad kurzov učiteľa so všetkými dostupnými akciami pre každý kurz.

Používateľské rozhranie ponúka nasledujúce možnosti pre každý kurz:

- **Úprava základných informácií** - možnosť upraviť názov kurzu, maximálny počet študentov alebo priradené obdobie.
- **Editor popisu kurzu** - učiteľ má k dispozícii WYSIWYG editor (TinyMCE) pre pohodlné formátovanie popisu kurzu. Môže vkladať nadpisy, zoznamy, formátovaný text, kódové bloky, obrázky a ďalšie. Tento popis sa zobrazuje aj študentom. Veľkou výhodou je možnosť prepínať medzi režimom úprav a náhľadu, pričom náhľad používa rovnaký komponent ako študentské rozhranie - čím je zaručené, že učiteľ uvidí presne to isté, čo študenti. Ukážka editora a prehľadu je na obrázkoch 3.7 a 3.8.

The screenshot shows the TinyMCE editor interface with the following elements:

- EDITOR** tab is selected.
- PREHĽAD** tab is visible.
- Toolbar:** Includes icons for back, forward, font style (Roboto, Arial, etc.), font size (21px), bold (B), italic (I), underline (U), font color (A), text alignment (text color, align left, center, right, justify), and a more options menu (...).
- Title:** **Programovacie paradigmy**
- Description:** Kurz **Programovacie paradigmy** ponúka študentom prehľad rôznych prístupov k programovaniu, ktoré formujú spôsob, akým sa riešia úlohy v softvériovom vývoji. Cieľom je porozumieť výhodám a nevýhodám jednotlivých paradigm a naučiť sa ich aplikovať v praxi.
- Section:** **Obsah kurzu:**
 - **Imperativne programovanie** – základné princípy, príkazy, stav programu
 - **Objektovo-orientované programovanie (OOP)** – triedy, objekty, dedičnosť, polymorfizmus
 - **Funkcionálne programovanie** – čisté funkcie, vyššie poradie funkcií, imutabilita
 - **Logické programovanie** – deklaratívny prístup, pravidlá a fakty, Prolog
 - **Porovnanie paradigm** – výber vhodného prístupu pre konkrétny problém
- Section:** **Výstupy z kurzu:**
 - Rozpoznať a použiť vhodnú paradigmu pre daný typ problému
 - Napísat jednoduché programy vo viacerých paradigmatických štýloch
 - Porozumieť výhodám kombinovania viacerých prístupov
- Text:** Kurz je vedený interaktívne formou prednášok a cvičení, kde si študenti vyskúšajú jednotlivé koncepty na praktických príkladoch v rôznych programovacích jazykoch.
- Editor status:** h2
- Build with tinyMCE**
- Buttons:** ZRUŠIŤ (Cancel) and ULOŽIŤ (Save).

Obr. 3.7: Úprava HTML popisu kurzu pomocou WYSIWYG editora TinyMCE.

Programovacie paradigmy / ZS 2024**Popis kurzu****Programovacie paradigmy**

Kurz **Programovacie paradigmy** ponúka študentom prehľad rôznych prístupov k programovaniu, ktoré formujú spôsob, akým sa riešia úlohy v softvérovom vývoji. Cieľom je porozumieť výhodám a nevýhodám jednotlivých paradigm a naučiť sa ich aplikovať v praxi.

Obsah kurzu:

- **Imperativné programovanie** – základné princípy, príkazy, stav programu
- **Objektovo-orientované programovanie (OOP)** – triedy, objekty, dedičnosť, polymorfizmus
- **Funkcionálne programovanie** – čisté funkcie, vyššie poradie funkcií, imutabilita
- **Logické programovanie** – deklaratívny prístup, pravidlá a fakty, Prolog
- **Porovnanie paradigm** – výber vhodného prístupu pre konkrétny problém

Výstupy z kurzu:

- Rozpoznať a použiť vhodnú paradigmu pre daný typ problému
- Napísat jednoduché programy vo viacerých paradigmatických štýloch
- Porozumieť výhodám kombinovania viacerých prístupov

Kurz je vedený interaktívne formou prednášok a cvičení, kde si študenti vyskúšajú jednotlivé koncepty na praktických príkladoch v rôznych programovacích jazykoch.

Informácie o kurze

| | |
|-----------|--------------|
| Učiteľ: | Meno Učiteľa |
| Obdobie: | ZS 2024 |
| Kapacita: | 55 |

Obr. 3.8: Režim prehľadu zobrazuje HTML popis kurzu tak, ako ho vidia študenti.

- **Správa účastníkov kurzu** - učiteľ má k dispozícii nástroj na správu účastníkov prihlásených do kurzu. Môže ich filtrovať podľa mena, schvaľovať alebo odstraňovať z kurzu. Každý účastník má viditeľný stav („potvrdený“ alebo „čaká na schválenie“), pričom nové žiadosti je možné jedným kliknutím schváliť.

Účastníci kurzu – Programovacie paradigmy (ZS 2024)

| Vyhľadaj podľa mena | | | |
|---------------------|----------------------|--------------------|-------|
| Meno | Email | Stav | Akcie |
| Student 4 | Student4@example.com | Čaká na schválenie | ✓ ✗ |
| Student 1 | Student@example.com | Potvrdený | ✗ |
| Student 2 | Student2@example.com | Potvrdený | ✗ |
| Student 3 | Student3@example.com | Potvrdený | ✗ |

Obr. 3.9: Zoznam študentov v kurze s možnosťou schválenia alebo odstránenia.

- **Zadania v kurze** - rozhranie umožňuje učiteľovi pridávať k danému kurzu zadania a spravovať ich.
- **Vymazanie kurzu** - ikona koša umožní kurz odstrániť, pričom pred samotným zmazaním sa zobrazí výstražné potvrdenie, aby nedošlo k náhodnej strate údajov.

Toto komplexné rozhranie zefektívnuje správu kurzov pre vyučujúcich a zároveň zaručuje konzistentné zobrazovanie informácií študentom.

3.4.6 Komunikácia frontendu s backendom

Frontendová aplikácia využíva knižnicu `axios` na komunikáciu s backendom prostredníctvom REST API. V rámci projektu je vytvorený centralizovaný `api` objekt, ktorý definuje základné nastavenia pre všetky požiadavky - vrátane základnej URL adresy backendu a predvolených hlavičiek.

Pri každej požiadavke sa automaticky pridáva aj `Authorization` hlavička s JWT tokenom, ktorý identifikuje prihláseného používateľa a umožňuje backendu overiť oprávnenie. Token je získaný z lokálneho úložiska a pridaný prostredníctvom interceptorov, ktoré sú súčasťou konfigurácie `axios`. V prípade chyby 401 Unauthorized sa používateľ automaticky odhlasuje zo systému.

Získavanie údajov z backendu je vo frontende riešené pomocou asynchrónnych funkcií. Nasledujúci príklad ukazuje, ako sa získava zoznam všetkých kurzov zo servera.

Po úspešnom načítaní sa zoznam uloží do stavu komponentu, pričom sa počas čakania zobrazuje indikácia načítavania. V prípade chyby sa používateľovi zobrazí chybové hlásenie pomocou notifikácie.

```

1  const fetchCourses = async () => {
2      setLoading(true);
3      try {
4          const res = await api.get<Course[]>("/courses");
5          setCourses(res.data);
6      } catch (err) {
7          console.error(err);
8          showNotification("Nepodarilo sa načítať kurzy.", "error");
9      } finally {
10         setLoading(false);
11     }
12 };

```

Tento prístup je v systéme štandardizovaný a využíva sa pri všetkých ďalších operáciách ako je vytváranie, aktualizácia alebo mazanie kurzov, účastníkov a ďalších entít. Vďaka centralizovanému nastaveniu API klienta je zabezpečená konzistentnosť požiadaviek a zjednodušenie obsluhy odpovedí zo servera.

3.5 Úlohy (Tasks)

3.5.1 Prehľad modulu

Modul `List.Tasks` predstavuje serverovú časť systému, ktorá zabezpečuje správu úloh a súvisiacich kategórií. Jeho hlavnou úlohou je umožniť učiteľom vytvárať a spravovať úlohy, ktoré sú následne priradené k jednotlivým zadaniám v iných moduloch systému. Študenti majú možnosť tieto úlohy zobrazovať, pričom obsah úlohy tvorí zadanie v textovej podobe s možnosťou formátovania.

Medzi základné funkcionality modulu patrí:

- vytváranie a aktualizácia úloh učiteľom,
- možnosť kategorizovať úlohy pomocou stromovej štruktúry kategórií,
- evidencia autora každej úlohy,
- zadanie úlohy vo formáte HTML (zadávané pomocou editora TinyMCE),
- ukladanie interného komentára - viditeľného iba vyučujúcim,

Modul je navrhnutý tak, aby umožňoval štruktúrované vyhľadávanie úloh podľa priradených kategórií. Využíva rovnaký prístup k validácii a správe údajov ako ostatné moduly systému.

List.Tasks má závislosť na moduloch List.Users, List.Common a List.Logs, ktoré poskytujú informácie o používateľoch, spoločné dátové typy a logovanie udalostí. Zároveň je tento modul referencovaný v hlavnom projekte List.Server, ktorý integruje všetky časti systému do jednej spustiteľnej aplikácie.

3.5.2 Dátové modely

Modely sa nachádzajú v priečinku List.Tasks.Models. Modul Tasks definuje tri základné entity: TaskModel, TaskCategoryRel a CategoryModel, ktoré reprezentujú hlavné dátové objekty súvisiace so správou úloh a ich kategorizáciou.

TaskModel

Trieda TaskModel reprezentuje jednu úlohu vytvorenú učiteľom. Každá úloha obsahuje základné informácie ako názov, zadanie (v HTML formáte), interný komentár, dátumy vytvorenia a poslednej úpravy, ako aj informáciu o autorovi (učiteľovi). Súčasťou je aj kolekcia väzieb na kategórie, do ktorých úloha patrí.

```
1 public class TaskModel
2 {
3     [Key]
4     public int Id { get; set; }
5     [Required]
6     public DateTime Created { get; set; }
7     [Required]
8     public DateTime Updated { get; set; }
9     [Required]
10    public string? Name { get; set; }
11    public string? Text { get; set; }
12    public string? InternalComment { get; set; }
13
14    [Required]
15    public int AuthorId { get; set; }
16    [ForeignKey(nameof(AuthorId))]
17    public User Author { get; set; } = null!;
18
19    public ICollection<TaskCategoryRel> TaskCategories { get; set; }
20 }
```

TaskCategoryRel

Trieda `TaskCategoryRel` reprezentuje väzbu medzi úlohou a kategóriou. Táto entita zabezpečuje many-to-many vzťah, v ktorom môže byť jedna úloha zaradená do viacerých kategórií a zároveň každá kategória môže obsahovať viacero úloh. Záznam obsahuje identifikátory úlohy aj kategórie a umožňuje jednoduché filtrovanie úloh podľa ich zaradenia.

CategoryModel

Trieda `CategoryModel` reprezentuje kategóriu úloh a podporuje hierarchickú (stromovú) štruktúru. Každá kategória môže mať nadradenú kategóriu (rodiča) a zároveň zoznam podradených kategórií. Táto stromová organizácia umožňuje systematické trienie úloh a uľahčuje ich prehľadné zobrazovanie vo frontende.

3.5.3 Databázový kontext

Modul `Tasks` využíva vlastný databázový kontext `TasksDbContext`, ktorý dedí, ako pri module `Courses`, z triedy `DbContext` v rámci frameworku Entity Framework Core. Tento kontext definuje prístup k databázovým entitám `TaskModel`, `CategoryModel` a `TaskCategoryRel` prostredníctvom vlastností typu `DbSet`, čím umožňuje vykonávať CRUD operácie nad nimi.

V rámci metódy `OnModelCreating` sú definované jednotlivé väzby medzi entitami, ich správanie pri mazaní a špecifické obmedzenia - ako napríklad obojsmerné väzby alebo zložené primárne kľúče pre prepojovacie tabuľky. Osobitne sa definuje aj správanie entity `User`, ktorá je sice používaná v module, no je spravovaná iným databázovým kontextom.

Príkladom je konfigurácia entity `CategoryModel`, ktorá podporuje stromovú štruktúru kategórií prostredníctvom väzby na rodičovskú kategóriu:

```

1 modelBuilder.Entity<CategoryModel>()
2   .HasOne(c => c.Parent)
3   .WithMany(c => c.Children)
4   .HasForeignKey(c => c.ParentId)
5   .OnDelete(DeleteBehavior.Cascade);

```

Tento kód definuje:

- hierarchickú väzbu medzi kategóriami, kde každá kategória môže mať nadradenú kategóriu a zároveň viacero podradených,
- správanie pri mazaní (`Cascade`) - ak sa vymaže nadradená kategória, automaticky sa vymažú aj všetky jej podkategórie.

Okrem toho sú v kontexte definované aj väzby medzi úlohami a používateľmi (autor úlohy), ako aj väzby medzi úlohami a kategóriami cez prepojovaciu entitu `TaskCategoryRel`, kde sa používa zložený primárny kľúč `task_id`, `category_id`. Všetky tieto pravidlá spolu zabezpečujú integritu dát a konzistentné správanie systému pri operáciách nad databázou.

3.5.4 API endpointy

Modul `Tasks` definuje REST API endpointy, ktoré zabezpečujú kompletnú funkcionality pre správu úloh, ich kategórií a väzieb medzi nimi. Kompletný zoznam všetkých dostupných endpointov znázorňuje obrázok 3.10.

| Category | |
|-----------------|---|
| GET | /api/category |
| POST | /api/category |
| GET | /api/category/{id} |
| PUT | /api/category/{id} |
| DELETE | /api/category/{id} |
| TaskCategoryRel | |
| POST | /api/tasks/task-category |
| DELETE | /api/tasks/task-category |
| GET | /api/tasks/task-category/by-task/{taskId} |
| GET | /api/tasks/task-category/by-category/{categoryId} |
| Tasks | |
| GET | /api/tasks |
| POST | /api/tasks |
| GET | /api/tasks/{id} |
| PUT | /api/tasks/{id} |
| DELETE | /api/tasks/{id} |
| GET | /api/tasks/filter |

Obr. 3.10: Zoznam všetkých dostupných REST API endpointov v rámci modulu Tasks.

Základné endpointy sa nachádzajú v rámci `/api/tasks` a slúžia na CRUD operácie nad úlohami. Základné operácie zahŕňajú:

- GET `/api/tasks` - vráti zoznam všetkých úloh, vrátane základných údajov o autorovi a dátumoch vytvorenia a úpravy,
- GET `/api/tasks/{id}` - vráti podrobnosti o konkrétnej úlohe podľa jej ID,
- POST `/api/tasks` - vytvorí novú úlohu na základe údajov z formulára,

- PUT /api/tasks/id - aktualizuje existujúcu úlohu,
- DELETE /api/tasks/id - vymaže konkrétnu úlohu,
- GET /api/tasks/filter - umožňuje filtrovať úlohy podľa názvu, autora alebo kategórie.

Okrem hlavnej časti pre úlohy má modul Tasks aj samostatné endpointy pre správu kategórií (/api/category) a ich väzieb na úlohy (/api/tasks/task-category), ktoré sú taktiež znázornené na obrázku 3.10. Tieto poskytujú špecifické operácie ako pridávanie kategórií, hierarchické zaraďovanie a priradzovanie kategórií k jednotlivým úlohám.

3.5.5 Učiteľské rozhranie pre správu úloh

Učiteľské rozhranie poskytuje vyučujúcim nástroje na správu úloh, ktoré môžu byť neskôr použité v zadaniach pre študentov. Základný pohľad na rozhranie zobrazuje zoznam všetkých existujúcich úloh spolu s informáciou o autorovi a interným komentárom, ktorý je viditeľný iba vyučujúcim. Každá úloha má k dispozícii akcie na jej úpravu alebo vymazanie. Tento prehľad je znázornený na obrázku 3.11.

Úlohy

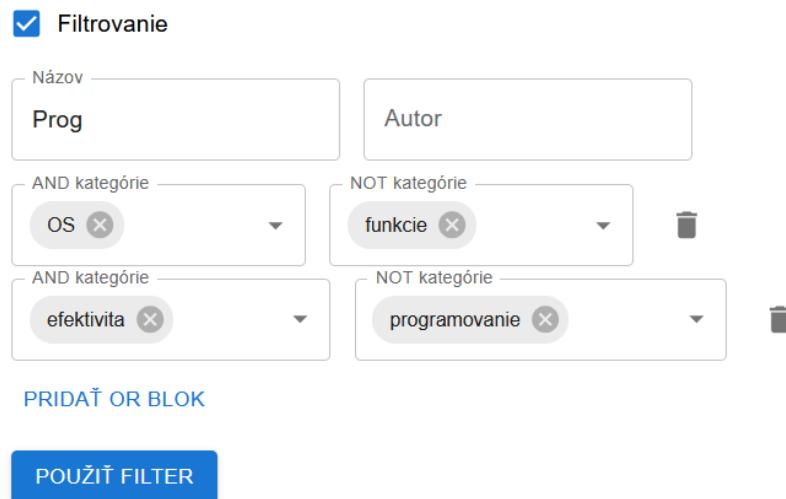
PRIDAŤ ÚLOHU

Filtrovanie

| Názov | Autor | Vnútorný komentár | Akcie |
|--|----------------|------------------------------|-------|
| Správa procesov v operačnom systéme | Meno Učiteľa | Vytvorenie/ukončenie procesu | |
| Mutexy | Meno Učiteľa | Ochrana kritickej sekcie | |
| Interprocesová komunikácia cez rúrky | Meno Učiteľa | | |
| Zdieľaná pamäť a jej bezpečné použitie | Meno Učiteľa 2 | | |
| Správa vlákien a plánovanie CPU | Meno Učiteľa | Plánovanie a prechody stavu | |
| Signály | Meno Učiteľa | Asynchronné prerušovanie | |
| Virtuálna pamäť a správa swapu | Meno Učiteľa | | |
| Thread pool a efektívna správa vlákien | Meno Učiteľa | | |
| Semafory a ich využitie pri synchronizácii | Meno Učiteľa | Počítanie prístupu | |

Obr. 3.11: Zoznam existujúcich úloh spolu s autorom, interným komentárom a akciami na úpravu či vymazanie.

Rozhranie ponúka vyhľadávanie a pokročilé filtrovanie úloh podľa viacerých kritérií. Vyučujúci môže filtrovať úlohy podľa názvu, autora, ako aj podľa priradených kategórií. Kategórie môžu byť filtrované pomocou logiky AND alebo NOT, čo umožňuje vyhľadanie veľmi špecifických úloh. Príklad zobrazenia filtrovacieho formulára možno vidieť na obrázku 3.12.



Obr. 3.12: Rozšírené možnosti filtrovania podľa názvu, autora a kategórií.

Pri vytváraní alebo úprave úlohy sa používa rovnaký editor, ktorý pozostáva z troch záložiek:

- **Editor** - umožňuje vyplniť názov a text úlohy pomocou WYSIWYG editora TinyMCE,
- **Prehľad** - náhľad úlohy tak, ako ju uvidia študenti,
- **Kategórie** - výber kategórií, do ktorých úloha patrí.

Editor umožňuje formátovanie textu úlohy vrátane zvýrazňovania kódu, čo je užitočné najmä pri zadávaní technických alebo programovacích úloh. Interný komentár slúži na doplňujúce poznámky učiteľa a nie je zobrazovaný študentom. Ukážka editora je uvedená na obrázku nižšie.

Obr. 3.13: Vytváranie alebo úprava úlohy pomocou editora TinyMCE s možnosťou priradenia interného komentára.

Týmto spôsobom majú vyučujúci k dispozícii intuitívne a efektívne nástroje na správu úloh, ktoré zaručujú konzistentnosť ich formátovania aj dostupnosť relevantných metadát pre ďalšie spracovanie.

3.5.6 Komunikácia frontendu s backendom

Frontendová aplikácia komunikuje s backendom pomocou centralizovaného objektu `api`, ktorý je postavený na knižnici `axios`. Všetky požiadavky automaticky zahŕňajú `Authorization` hlavičku s JWT tokenom, čo zabezpečuje identifikáciu a autorizáciu používateľa pri volaniach REST API.

Pri práci s úlohami sa v aplikácii bežne používajú jednoduché asynchrónne funkcie, ktoré slúžia ako stavebné bloky pre získavanie, pridávanie a mazanie dát. Nasledujúce ukážky ilustrujú niektoré z typických interakcií medzi klientom a serverom:

```

1 // Vymazanie vybranej úlohy po potvrdení
2 const handleDelete = async () => {

```

```

3  if (!taskToDelete) return;
4  try {
5    await api.delete(`/tasks/${taskToDelete.id}`);
6    showNotification("Úloha bola vymazaná.", "success");
7    fetchTasks();
8  } catch {
9    showNotification("Chyba pri vymazaní úlohy.", "error");
10 } finally {
11  setConfirmOpen(false);
12 }
13 };

```

Mazanie úloh je previazané s potvrdením cez modálne dialógové okno. Po úspešnom odstránení sa zoznam úloh automaticky aktualizuje.

```

1 // Načítanie úloh s podporou filtrov a kategórií
2 const fetchTasks = async () => {
3   setLoading(true);
4   try {
5     const params: Record<string, string | undefined> = {
6       name: filters.name,
7       author: filters.author,
8     };
9
10    if (filterEnabled && filters.categoryBlocks.length > 0) {
11      params.categoryFilter = filters.categoryBlocks
12        .map((b) => {
13          const hasInclude = b.include.length > 0;
14          const hasExclude = b.exclude.length > 0;
15
16          if (!hasInclude && !hasExclude) return null;
17
18          let part = hasInclude ? b.include.join(",") : "";
19          if (hasExclude) part += `;!${b.exclude.join(",")}`;
20          return part;
21        })
22        .filter(Boolean)
23        .join("|");
24    }
25

```

```

26 const response = await api.get("/tasks/filter", { params });
27 setTasks(response.data);
28
29 } catch {
30   showNotification("Nepodarilo sa načítať úlohy.", "error");
31 } finally {
32   setLoading(false);
33 }
34 };

```

Táto funkcia slúži na načítanie úloh z backendu s podporou viacerých filtrov - konkrétnie podľa názvu, autora a logiky výberu kategórií. Ak sú zapnuté kategórie, jednotlivé bloky filtrovania (zahrnuté/vylúčené ID) sa zakódujú do jedného reťazca pomocou separátorov a odošlú ako categoryFilter parameter. Získané úlohy sa následne ukladajú do lokálneho stavu komponentu a používateľ vidí aktuálny zoznam.

```

1 // Pridanie novej úlohy pomocou POST požiadavky
2 await api.post("/tasks", {
3   name,
4   text: updatedText,
5   internalComment: comment,
6 });

```

Tento jednoduchý zápis reprezentuje typický POST request na vytvorenie novej úlohy. Dáta sa odosielajú v tele požiadavky ako objekt so základnými atribútmi - názov, HTML text zadania a interný komentár pre učiteľa. V praxi sa používa vo formulári editora a predstavuje základný príklad komunikácie klienta s REST API pri vytváraní nových záznamov.

Tieto úryvky z kódu predstavujú základnú štruktúru klientskych volaní pre prácu s REST API. Ich jednoduchosť a opakovateľnosť umožňuje efektívne zostavovať komplexnejšie operácie, ako napríklad filtrovanie úloh podľa viacerých kritérií, prácu s kategóriami, editovanie záznamov či nahrávanie obrázkov do textového editora. Všetky tieto funkcionality zdieľajú jednotný prístup ku komunikácii so serverom, čo zjednoduší vývoj a údržbu frontendu.

3.6 Študentské používateľské rozhranie

Frontendová časť určená pre študentov bola implementovaná v jazyku TypeScript pomocou knižnice React. Použité boli preddefinované komponenty knižnice Material UI, ktoré umožňujú konzistentný a responzívny dizajn. Celé rozhranie je ladené do

modernej kombinácie bielej a modrej farby, pričom návrh vychádza z princípov Google Material Design. Komponenty boli navrhnuté ako samostatné a znovupoužiteľné moduly, členené podľa funkčných celkov (napr. zoznam kurzov, detail kurzu, prihlásenie na kurz). Interakčná logika využíva React Hooks a contexty na správu stavu a komunikáciu medzi komponentmi. Cieľom bolo vytvoriť prehľadné, intuitívne a technicky udržateľné rozhranie, ktoré zjednodušuje prácu študenta so systémom.

Po úspešnom prihlásení sa používateľovi zobrazí zoznam kurzov, ktorých je členom (sekcia „Tvoje kurzy“), ako aj ďalších kurzov, do ktorých sa môže prihlásiť. Kurzy sú reprezentované ako kartičky s názvom, obdobím a menom učiteľa. Kliknutím na niektorý z „tvojich“ kurzov sa študent presunie do detailu daného kurzu.

Tvoje kurzy

- Vývoj mobilných aplikácií**
ZS 2024
Meno Učiteľa
- Tvorba informačných systémov**
ZS 2024
Meno Učiteľa
- Programovacie paradigmy**
ZS 2024
Meno Učiteľa

Ostatné kurzy

- Grafy, grafové algoritmy a optimalizácia**
ZS 2024
Meno Učiteľa
KURZ JE PLNÝ
- Operačné systémy**
LS 2024
Meno Učiteľa
PRIDAŤ SA
- Programovanie 4 - Java**
LS 2025
Meno Učiteľa
PRIDAŤ SA
- Programovanie 2**
LS 2025
Meno Učiteľa
PRIDAŤ SA

Obr. 3.14: Rozhranie s rozdelením na tvoje a ostatné kurzy

Po výbere kurzu sa otvorí jeho detailná stránka s textovým popisom, informáciami o učiteľovi, období a kapacite. Na ľavej strane sa nachádza bočný navigačný panel, ktorý umožňuje prepínanie medzi jednotlivými sekciami kurzu - napríklad Úlohy, Prehľad alebo Projekty.

Student – Student 1

Slovenčina ▾ ODHLÁSIŤ SA

Kurzy

Popis

Úlohy

Prehľad

Projekty

Programovacie paradigmá / ZS 2024

Popis kurzu

Programovacie paradigmá

Kurz **Programovacie paradigmá** ponúka študentom prehľad rôznych prístupov k programovaniu, ktoré formujú spôsob, akým sa riešia úlohy v softvérovom využívaní. Cieľom je porozumieť výhodám a nevýhodám jednotlivých paradigiem a naučiť sa ich aplikovať v praxi.

Obsah kurzu:

- **Imperatívne programovanie** – základné princípy, príkazy, stav programu
- **Objektovo-orientované programovanie (OOP)** – triedy, objekty, dedičnosť, polymorfizmus
- **Funkcionálne programovanie** – čisté funkcie, vyššie poradie funkcií, imutabilita
- **Logické programovanie** – deklaratívny prístup, pravidlá a fakty, Prolog
- **Porovnanie paradigiem** – výber vhodného prístupu pre konkrétny problém

Výstupy z kurzu:

- Rozpoznať a použiť vhodnú paradigmu pre daný typ problému
- Napsať jednoduché programy vo viacerých paradigmatických štýloch
- Porozumieť výhodám kombinovania viacerých prístupov

Kurz je vedený interaktívne formou prednášok a cvičení, kde si študenti vyskúšajú jednotlivé koncepty na praktických príkladoch v rôznych programovacích jazykoch.

Informácie o kurze

| | |
|-----------|--------------|
| Učiteľ: | Meno Učiteľa |
| Obdobie: | ZS 2024 |
| Kapacita: | 55 |

Obr. 3.15: Popis kurzu a navigačný sidebar

V záložke „Zadania“ je dostupný zoznam všetkých úloh rozdelený podľa kategórií, ako sú „Domáce úlohy“ alebo „Skúška“. Každé zadanie obsahuje dátum odovzdania, počet bodov a aktuálny status (napr. Odovzdané, Otvorené, Neodovzdané). Zadania možno filtrovať podľa stavu pomocou horného panela s tlačidlami.

Zadania

Filtrovať podľa stavu:

| | | | | | |
|--------|------------|-----------|----------|-------------------|-------------|
| VŠETKY | OHODNOTENÉ | ODOVZDANÉ | OTVORENÉ | ODOVZDANÉ NESKORO | NEODOVZDANÉ |
|--------|------------|-----------|----------|-------------------|-------------|

Domáce úlohy
0 / 10 points

| Zadanie | Deadline | Body | Status |
|----------|----------------------|-------|----------|
| Domaca 2 | 4. 6. 2025 23:59:59 | - / 5 | Otvorené |
| Domaca 1 | 28. 5. 2025 23:59:59 | - / 5 | Otvorené |

Skuska
0 / 30 points

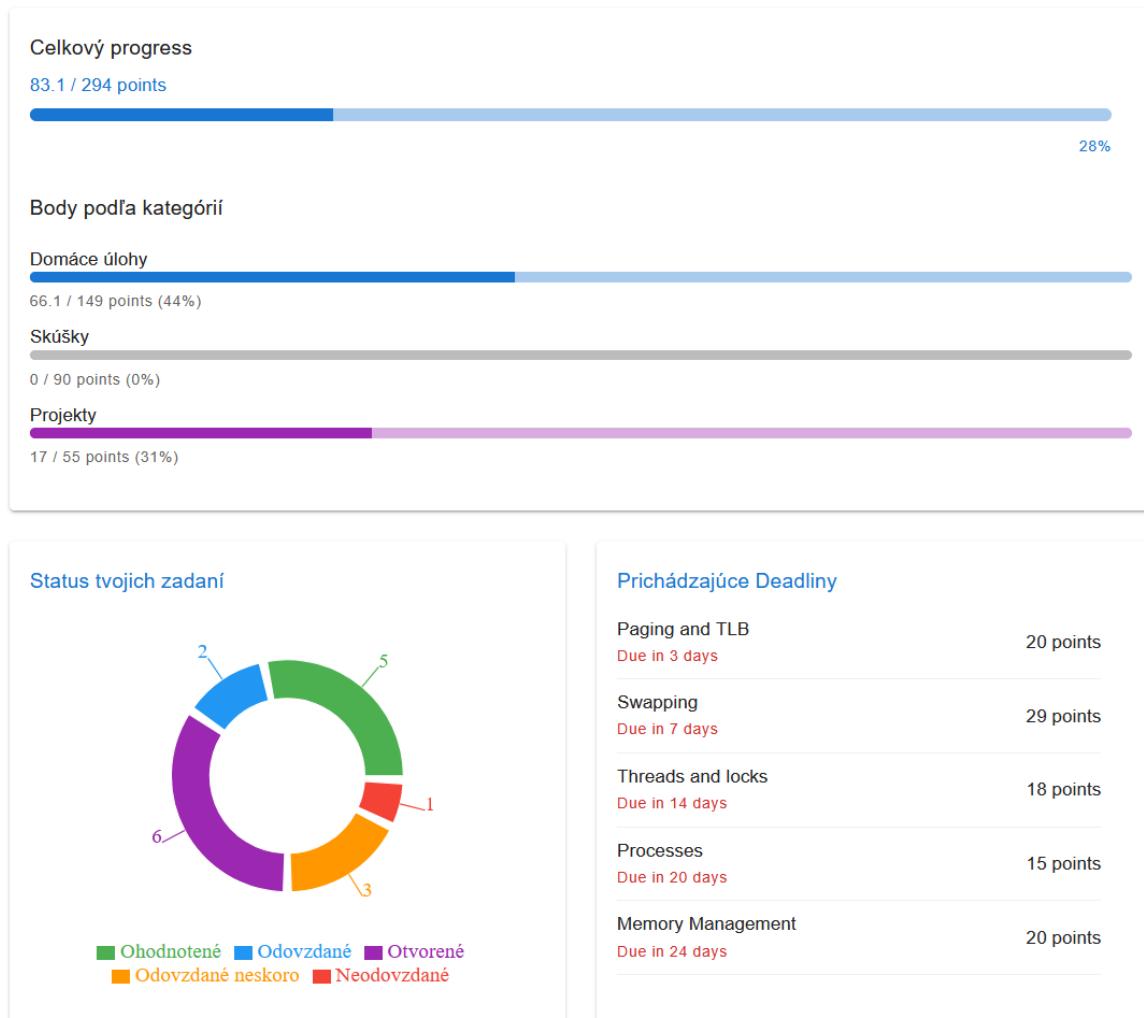
Midtermy
0 / 30 points

| Zadanie | Deadline | Body | Status |
|-----------|----------------------|--------|----------|
| Midterm 2 | 21. 5. 2025 23:59:59 | - / 15 | Otvorené |
| Midterm 1 | 20. 5. 2025 23:59:59 | - / 15 | Otvorené |

Obr. 3.16: Zoznam zadaní podľa kategórie a stavu

Sekcia „Prehľad“ poskytuje študentovi zhrnutie aktuálneho stavu v kurze. Zobrazuje sa tu celkový progres (body získané zo všetkých úloh), bodové rozdelenie podľa kategórií (projekty, skúšky, domáce úlohy), ako aj kruhový graf so statusmi zadaní. V pravom dolnom rohu sú zobrazené aj nadchádzajúce termíny s dátumom a počtom bodov.

Prehľad kurzu



Obr. 3.17: Grafické zobrazenie progresu a najbližších termínov

Kliknutím na konkrétné zadanie sa otvorí jeho detailná obrazovka, kde sa zobrazí názov úlohy, jej text a autor. V dolnej časti stránky sa nachádza komponent pre odozvanie riešenia vo forme súboru - akceptované sú archívy a zdrojové súbory (.zip, .c, .cpp, .py, .java, ...). Po nahratí súboru sa riešenie odošle na server.

Názov zadania**Inštrukcie:**

Toto je inštrukcia k zadaniu ako celku

Zdieľaná pamäť a jej bezpečné použitie**Vymedzené miesto na zadanie úlohy**

Vytvoril: meno Autora

Mutexy**Vymedzené miesto na zadanie úlohy**

Vytvoril: meno Autora

Odovzdanie riešenia

Vyberte súbor...

VYBRAŤ SÚBOR

ODOVZDAŤ RIEŠENIE

Podporované formáty: .zip, .rar, .tar.gz, .c, .cpp, .py, .java

Obr. 3.18: Zobrazenie úlohy a možnosť odovzdania riešenia

Všetky komponenty a ich funkcia boli navrhnuté s dôrazom na jednoduchosť, prehľadnosť a intuitívnosť. Navigácia je rýchla a jednotná pre všetky kurzy. Vďaka predom navrhnutému a premyslenému rozdeleniu stránok má študent okamžitý prístup k najdôležitejším informáciám a úlohám.

Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať nový informačný systém typu Learning Management System - modernú náhradu za existujúci, dnes už technologicky zastaraný systém L.I.S.T. V rámci tímového vývoja sa nám podarilo vytvoriť funkčný základ nového LMS, ktorý je pripravený na nasadenie a každodenné používanie pri výučbe na Fakulte matematiky, fyziky a informatiky.

Nový systém je postavený na technológiách, ktoré zohľadňujú nároky na škálovateľnosť, bezpečnosť, prehľadné používateľské rozhranie a jednoduchú rozšíriteľnosť. Backend je vytvorený v platforme .NET 8 s využitím architektúry REST API, modularizácie jednotlivých funkčných častí a databázy PostgreSQL. Frontend je postavený na Reacte a TypeScripte s využitím Material UI, pričom rešpektuje komponentový prístup a udržiavateľnú štruktúru kódu. Tento vývojový stack nám umožňuje pokračovať vo vývoji jednotlivých častí nezávisle, čo zvyšuje efektivitu tímovej spolupráce.

Moja práca v rámci tíme sa sústredila najmä na návrh a implementáciu komponentov súvisiacich s autentifikáciou používateľov, správou kurzov a správou úloh, ako aj tvorbou používateľského rozhrania pre študentov. Osobitný dôraz som kládal na prehľadnosť dátových modelov a modularitu backendu. Na strane frontendu som pracoval na návrhu intuitívneho rozhrania, ktoré študentovi sprístupňuje zadania, hodnotenia a správu kurzov vo forme prehľadných a responzívnych obrazoviek.

Vytvorený systém poskytuje pevný základ pre ďalší vývoj. Okrem základnej funkcionality, ktorá je plne použiteľná už v aktuálnom stave, sme navrhli architektúru tak, aby bolo jednoduché dopĺňať ďalšie moduly alebo ich rozširovať. Medzi najbližšie kroky patrí migrácia historických údajov a súborov zo starého systému, ako aj rozšírenie oprávnení rolí, ktoré umožní priradovať práv podľa potrieb jednotlivých predmetov.

Celý zdrojový kód systému je verejne dostupný v repozitári na platforme GitHub na adrese: <https://github.com/New-LIST/L.I.S.T>. Projekt je otvorený a pripravený na ďalší vývoj, čo umožňuje jeho budúce rozširovanie aj zapojenie nových vývojárov pri jeho dlhodobej údržbe a zlepšovaní.

Výsledkom našej práce je teda moderný, otvorený a udržiavateľný systém, ktorý reaguje na aktuálne potreby fakulty a zároveň vytvára technologický priestor pre jeho dlhodobé používanie a rozvoj.

Literatúra

- [1] Grzegorz Blinowski, Anna Ojdowska, and Adam Przybyłek. Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE access*, 10:20357–20374, 2022.
- [2] Imed Bouchrika. How google conquered the classroom: The googlification of schools worldwide for 2025, 2025. Research.com, dostupné online: <https://research.com/education/how-google-conquered-the-classroom>, citované dňa 1. apríla 2025.
- [3] Ryann K Ellis. Learning management systems. *Alexandria, VI: American Society for Training & Development (ASTD)*, 2009.
- [4] Google. Google classroom help, 2024. Google, dostupné on-line: <https://support.google.com/edu/classroom>, citované dňa 14. apríla 2025.
- [5] Google Developers. Grader iframe overview, 2025. dostupné on-line: <https://developers.google.com/workspace/classroom/add-ons/get-started/iframes/grader-iframe>, citované dňa 22.5.2025.
- [6] Philip Grew, Ivan Longhi, Fiorella De Cindio, Laura A Ripamonti, et al. Applying an lms to large language classes. In *Proc. IASTED International Conference on Web-Based Education (WBE'04)*, pages 480–484, 2004.
- [7] Anders Hejlsberg, Scott Wiltamuth, and Peter Golde. *C# Language Specification*. Addison-Wesley Longman Publishing Co., Inc., USA, 2003.
- [8] Andrej Jursa. Nový dlhodobý viacúčelový sklad zadaní, 2013. Bакalárska práca, Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky. Dostupné on-line: <https://opac.crzp.sk/?fn=detailBiblioFormChildE1V9QF&sid=E17FC85307AF86FA32C0B9535D4A>, citované dňa 4. apríla 2025.
- [9] Andrej Jursa. Realizácia automatického testovania úloh a kontroly plagiárstva v úlohopom systéme list. Diplomová práca, Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Bratislava, 2015. Diplomová

- práca, Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky. Dostupné on-line: <https://opac.crzp.sk/?fn=detailBiblioForm&sid=06ED14E8665C76DEA6C55894DD38>, citované dňa 3. apríla 2025.
- [10] Jay Melton. The lms moodle: A usability evaluation. *Languages Issues*, 11(12):1, 2006.
 - [11] Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a modular architecture enough? In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 28747–28760. Curran Associates, Inc., 2022.
 - [12] Moodle HQ. Moodle documentation. Dostupné on-line: https://docs.moodle.org/500/en/Main_page, citované dňa 11. apríla 2025, 2024.
 - [13] Viktor Shurygin, Natalya Saenko, Angelina Zekiy, Elena Klochko, and Mikhail Kulapov. Learning management systems in academic and corporate distance education. *International Journal of Emerging Technologies in Learning (iJET)*, 16(11):121–139, 2021.
 - [14] NHS Simanullang and Juniastel Rajagukguk. Learning management system (lms) based on moodle to improve students learning activity. In *Journal of Physics: Conference Series*, volume 1462, page 012067. IOP Publishing, 2020.
 - [15] Univerzita Komenského v Bratislave. Ako funguje elearning cez moodle na uk. Dostupné on-line: https://moodle.uniba.sk/local/staticpage/view.php?page=ako_funguje_elearning_cez_moodle, citované dňa 12. apríla 2025, 2025.
 - [16] Victor Velepucha and Pamela Flores. A survey on microservices architecture: Principles, patterns and migration challenges. *IEEE access*, 11:88339–88358, 2023.
 - [17] Alex Xu. The evolving landscape of api protocols in 2023. Dostupné on-line: <https://blog.postman.com/api-protocols-in-2023/>, citované dňa 11. apríla 2025, 2023.