VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

DATABÁZOVÝ SYSTÉM SOUTĚŽE STŘÍBRNÝ PÍST SMC

DATABASE SYSTEM FOR SILVER PISTON SMC COMPETITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Vojtěch Lenhart
AUTHOR

VEDOUCÍ PRÁCE — Ing. Ladislav Dobrovský
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Vojtěch Lenhart**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Ladislav Dobrovský**
Akademický rok:	2023/24

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Databázový systém soutěže Stříbrný píst SMC

Stručná charakteristika problematiky úkolu:

Každým rokem společnost SMC Industrial Automation CZ, s.r.o. pořádá odbornou soutěž pro studenty středních škol s názvem Stříbrný píst. Letošního ročníku (2023) se zúčastnilo rekordních 66 týmů z Česka i Slovenska, které mezi sebou soutěžily v řešení rozmanitých úloh a odnesly si tak spoustu praktických dovedností a zážitků. S rostoucím počtem účastníků ale také roste náročnost organizace soutěže, především spojená se zápisem výsledků, s účty a daty jednotlivých soutěžících a týmů, učitelů a organizátorů.

Cíle diplomové práce:

Seznámení se soutěží Stříbrný píst SMC. Seznámení se stávající databází soutěže – struktura, potřebná data (soutěžící, učitelé, organizátoři, bodovací systém), funkcionalita. Popis struktury.
Databázová online aplikace pro soutěž Stříbrný píst SMC. Návrh databázové struktury SP, struktura a správa účtů: učitel, student, organizátor – nastavení přístupových práv, potřebná data, správa, login. ER diagram databázového schematu.
Struktura a správa soutěžních úloh: bodování, časová synchronizace, zabezpečení stránek a účtů.
Implementace výsledků jednotlivých soutěží a celkové pořadí týmů v závislosti na termínu konání soutěže (interaktivní tabulka).
Vytvoření mobilní aplikace pro uživatele jako průvodce soutěží (informovat, připomínat, navigovat).

Seznam doporučené literatury:

WATT, Adrienne a Nelson ENG, 2014. Database Design [online]. 2nd Edition. BCcampus [cit. 2023-10-19]. Dostupné z: https://opentextbc.ca/dbdesign01/

PECINOVSKÝ, Rudolf, 2021. Python: kompletní příručka jazyka pro verzi 3.10. Praha: Grada Publishing. Knihovna programátora (Grada). ISBN 978-80-271-3442-7.

PRATA, Stephen. Mistrovství v C++. 4., aktualiz. vyd. Přeložil Boris SOKOL. Brno: Computer Press, 2013. Bestseller (Computer Press). ISBN isbn.978-80-2513-828-1.

Flask: web development, one drop at a time [online]. [cit. 2023-10-19]. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/

Qt for Android [online]. [cit. 2023-10-20]. Dostupné z: https://doc.qt.io/qt-6/android.html

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2023/24

V Brně, dne

L. S.

Ing. Pavel Heriban, Ph.D. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

ABSTRAKT

Tato diplomová práce se zaměřuje na návrh a realizaci databázového systému pro soutěž Stříbrný píst SMC. Práce zahrnuje seznámení se se soutěží, analýzu stávající databáze a technické aspekty vývoje, včetně bezpečnostních opatření. Analyzuje potřeby a požadavky soutěže a navrhuje vhodné řešení pro správu soutěžních úloh, bodování a správu uživatelských účtů. Implementace aplikace obsahuje vytvoření databázové struktury, implementaci bezpečnostních opatření a vývoj uživatelského rozhraní s responzivním designem pro různá zařízení. Výsledkem je komplexní databázový systém splňující požadavky soutěže a poskytující uživatelsky přívětivé prostředí pro správu soutěže Stříbrný píst SMC. Dosažené výsledky jsou zhodnoceny a jsou zde pak zmíněny možnosti pro další rozvoj a zlepšení systému.

ABSTRACT

This master's thesis focuses on the design and implementation of a database system for the SMC Silver Piston competition. The work includes an introduction to the competition, an analysis of the existing database and technical aspects of development, including security measures. It analyzes the needs and requirements of the competition and proposes a suitable solution for managing competition tasks, scoring and user account management. The application implementation involves creating a database structure, implementing security measures and developing a user interface with responsive design for various devices. The result is a complex database system that meets the requirements of the SMC Silver Piston competition and provides a user-friendly environment for managing the competition. The achieved results are evaluated and possibilities for further development and improvement of the system are mentioned.

KLÍČOVÁ SLOVA

databázový systém, webová aplikace, hodnocení soutěže, Flask, ochrana údajů, hash

KEYWORDS

database system, web application, competition rating, Flask, data protection, hash

ÚSTAV AUTOMATIZACE A INFORMATIKY

2024

BIBLIOGRAFICKÁ CITACE

LENHART, Vojtěch. *Databázový systém soutěže Stříbrný píst SMC*. Brno, 2024. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/157927. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Ladislav Dobrovský.

PODĚKOVÁNÍ

Tímto bych rád poděkoval vedoucímu mé diplomové práce Ing. Ladislavovi Dobrovskému za všestrannou pomoc, množství cenných a inspirativních rad a připomínek při konzultacích této práce. Taktéž bych chtěl poděkovat své rodině, přítelkyni a všem, kteří mě během studia podporovali.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 24. 5. 2024 ……

Vojtěch Lenhart

OBSAH

1 ÚVOD 15
2 SOUTĚŽ STŘÍBRNÝ PÍST SMC 17
2.1 Popis soutěže 17
2.1.1 Registrace 17
2.1.2 Termíny 18
2.1.3 Úlohy 18
2.1.4 Hodnocení 18
2.1.5 Vyhlášení vítězů 18
2.2 Současný systém 19
2.3 Požadavky na aplikaci 19
3 POUŽITÉ TECHNOLOGIE A DALŠÍ MOŽNOSTI 21
3.1 Programovací jazyky 21
3.1.1 Backend 21
3.1.2 Frontend 22
3.2 Webové frameworky 22
3.3 Relační databáze 23
3.3.1 MySQL 23
3.3.2 SQLite 24
3.3.3 PostgreSQL 24
3.3.4 MariaDB 24
3.4 Použité knihovny 24
3.4.1 Flask-Login 24
3.4.2 Werkzeug 25
3.4.3 SQLAlchemy 25
3.4.4 Flask-Migrate 26
3.4.5 Flask-Admin 26
3.4.6 Flask-Babel 26
3.4.7 HTML knihovny 27
3.4.8 WTForms 27
3.4.9 Openpyxl 28
3.4.10 BytesIO 28
3.5 Ochrana údajů 28
4 VLASTNÍ NÁVRH APLIKACE 29
4.1 Flask aplikace 29
4.1.1 Inicializace 30
4.2 Návrh databáze 30
4.2.1 SQLite 32
4.2.2 MariaDB 33

4.2.3 Struktura databáze 33
4.2.4 ER diagram 38
4.3 Stránky aplikace 38
4.3.1 Dle role 39
4.3.2 Základní šablona 39
4.3.3 Styly 41
4.3.4 Registrace 42
4.3.5 Přihlášení 44
4.3.6 Odhlášení 45
4.3.7 Domovská stránka 45
4.3.8 Profil 46
4.3.9 Týmy 47
4.3.10 Hodnocení 50
4.3.11 Výsledky 54
4.3.12 Ovládací panel 56
4.3.13 Seznam účastníků 60
4.3.14 Správa 61
4.4 Funkce aplikace 63
4.4.1 Ochrana hesel 63
4.4.2 Administrátorské rozhraní 64
4.4.3 Lokalizace 69
4.4.4 Export dat 70
4.4.5 Nový ročník soutěže 72
5 UVEDENÍ DO PROVOZU 73
6 ZÁVĚR 75
SEZNAM POUŽITÉ LITERATURY 77
SEZNAM OBRÁZKŮ 79
SEZNAM ZKRATEK A SYMBOLŮ 81

1 ÚVOD

V dnešním moderním světě se neustále zvětšuje význam počítačových aplikací a systémů. Díky velkému zájmu o využívání internetu a stále se rozvíjejícím technologiím množství zpracovávaných dat rapidně roste. Efektivita systémů se tedy musí stále zlepšovat a zároveň je potřeba klást velký důraz na zabezpečení při práci s daty uživatelů. Systémy je třeba navrhnout tak, aby byly pro uživatele jednoduše přístupné, intuitivní a aby poskytovaly spolehlivý výkon bez ohledu na narůstající množství dat. Současně by měly umožňovat snadnou správu a integraci s dalšími technologiemi, což zvyšuje jejich flexibilitu a užitnou hodnotu v různých aplikacích a prostředích.

Tato práce se zaměřuje na návrh a realizaci databázového systému pro soutěž Stříbrný píst SMC. Cílem je vytvořit aplikaci, která nejenže zajistí bezpečné ukládání a správu dat, ale také poskytne intuitivní a efektivní rozhraní pro uživatele.

Úvodní kapitola popisuje problém správy soutěže a potřebu efektivního databázového řešení. Zabývá se také bezpečnostními opatřeními a technickými aspekty vývoje, včetně požadavků na uživatelskou přívětivost a integraci s dalšími systémy. Následující část zkoumá dostupné technologie a osvědčené postupy v oblasti vývoje webových aplikací a jejich využití při návrhu databázového systému. V další kapitole jsou popsány specifické metody a techniky použité při vývoji systému, včetně návrhu databázového schématu, implementace bezpečnostních opatření a vývoje uživatelského rozhraní.

Výsledkem práce je komplexní databázový systém, který splňuje všechny stanovené požadavky a poskytuje spolehlivé a uživatelsky přívětivé řešení pro správu soutěže Stříbrný píst SMC. V závěru práce jsou zhodnoceny výsledky implementace a jsou diskutovány možnosti pro případný rozvoj a zlepšení systému v rámci další spolupráce.

2 SOUTĚŽ STŘÍBRNÝ PÍST SMC

Firma SMC Corporation je jedním z předních světových hráčů v oblasti průmyslové automatizace. Společnost byla v roce 1959 založena v Tokiu v Japonsku. Dnes má společnost SMC více než 23 000 zaměstnanců v 80 zemích světa. Jejich hlavním cílem je poskytovat komplexní řešení pro průmyslovou automatizaci, které zahrnují pneumatické zařízení a elektropneumatické systémy. SMC nabízí širokou škálu produktů, které posilují konkurenceschopnost firem po celém světě. Zásadním pilířem jejich úspěchu jsou technické znalosti, které podporuje síť 5 technických center, včetně dvou v Evropě. [1]

Mezinárodní soutěž „Stříbrný píst SMC“ byla vytvořena společností SMC Industrial Automation CZ s.r.o. Jedná se o každoroční akci, která se zaměřuje na podporu vzdělávání studentů 2. až 4. ročníků středních technických škol z České republiky a Slovenska. Koná se již od ročníku 2015/2016 a má za sebou již sedmý úspěšný ročník. Hlavním cílem soutěže je rozšíření podpory výuky průmyslové automatizace a motivace studentů k dalšímu vzdělávání v tomto oboru. Soutěž probíhá formou šesti disciplín během tří soutěžních dnů. Týmy se mohou skládat ze dvou až tří studentů registrované školy. V rámci soutěže budou sbírat body v disciplínách v oblastech pneumatických a elektropneumatických systémů a průmyslové automatizace. [2]

Soutěž je rozdělena do třech termínů (tzv. české, moravské a slovenské kolo), ve kterých týmy sbírají v šesti disciplínách body, jejichž součet generuje vítěze jednotlivých termínů, a nakonec vítěze celkového.

2.1 Popis soutěže

2.1.1 Registrace

Před samotnou registrací probíhá komunikace mezi pořadateli a školami, zda se škola bude chtít zúčastnit. V rámci předběžných informací sdělí učitel plánovaný počet týmů za danou školu a preferovaný termín účasti. Daný vedoucí (učitel) může vést i více týmů. Tato skutečnost je omezena pořadatelem a je závislá na předběžných informacích dle účasti (maximálně 72 týmů z obou zemí na ročník).

Registrace do soutěže se vždy otevírá po určitém datu definované pořadateli soutěže. Počínaje tímto dnem se můžou registrovat týmy pod vedením učitele. Každý tým má vždy 2-3 soutěžící, kteří soutěží za danou školu. Při registraci týmu se uvádí termín, na který se dostaví a zda se chce účastnit i doplňkové soutěže (programování). Pro dokončení registrace pak pořadatelé musí týmu přidělit číslo týmu a skupinu. Číslo týmu se přiděluje dle termínu a číslo skupiny pak dle časového harmonogramu, který se sestaví pro každý termín.

2.1.2 Termíny

Soutěž probíhá standardně ve 3 termínech (2 pro Českou republika a 1 pro Slovenskou republiku). Jeden termín je navíc jako záložní pro případ náhlého zrušení nebo jiné ojedinělé situace. Každý termín má danou adresu, kde se koná, tj. stát, město a konkrétní adresu. Každý soutěžní termín má stejná pravidla a podmínky pro soutěžící.

2.1.3 Úlohy

Hlavní soutěž

Všechny registrované týmy se automaticky účastní hlavní soutěže. Hlavní soutěž obsahuje soubor předem popsaných úloh. V každé úloze je možné získat určitý počet bodů. V posledním ročníku bylo obsahem 6 disciplín [2]:

- P1 + P2 – pneumatické a elektropneumatické úlohy (SMC Pneutrainer)
- ASIM – pneumatické a elektropneumatické úlohy (SMC AutoSIM)
- PO – identifikace reálných elektro-pneumatických komponent
- TEST – test všeobecných znalostí – fyzika, mechanika, elektrotechnika
- FMS – hledání závad na jednoúčelovém stroji

Doplňková disciplína

Každý tým se může volitelně přihlásit do této vedlejší soutěže při registraci.

Pro účast v doplňkové disciplíně se zasílají vzorky dané škole, aby měl tým čas na přípravu a seznámení s programovatelným automatem a elektrickou osou. Zapůjčené vzorky pak soutěžící na termínu odevzdávají. Vyhodnocení doplňkové disciplíny probíhá zvlášť mimo hlavní soutěž. V posledním ročníku bylo doplňkovou disciplínou [2]:

- Programování – programování jednoúčelového stroje (2D manipulátor)

2.1.4 Hodnocení

Hodnocení jednotlivých úloh provádí SMC zaměstnanci (porotci) v průběhu dne. Každý porotce má v kompetenci hodnotit jednu či více úloh. Konkrétní počet bodů pro daný tým a konkrétní úlohy zadá porotce do systému a automaticky se tyto body započítávají do celkového součtu bodů. Tabulka průběžných výsledků se mění v závislosti počtu bodů (seřazena sestupně).

2.1.5 Vyhlášení vítězů

Po hodnocení všech úloh je oznámen vítěz daného termínu (kola) a týmy na 2. a 3. místě. Po skončení posledního termínu se pak vyhlásí celkové výsledky a první 3 týmy jsou odměněny trofejí a dalšími cenami. Vyhlášen je také „Národní vítěz“ pro každý stát.

2.2 Současný systém

Současný bodovací systém je plně závislý na administrátorovi, který do něj ručně zapisuje soutěžící, učitele i kolegy, kteří se starají o bodování soutěže. Veškeré informace o zmíněných uživatelích (telefonní čísla, adresy, hesla apod.) jsou doplňovány právě správcem (adminem). Každý ročník soutěže pak znamená znovu ručně zadat veškerá data a průběžně stále komunikovat s kolegy, kteří se starají o evidenci soutěžících, škol, učitelů a výměnu e-mailových zpráv se zodpovědnými osobami. Takto navržený systém s sebou tedy nese značnou časovou náročnost a vyšší chybovost.

Systém má 3 typy uživatelů v rámci jednotlivých termínů soutěže s pevně nastavenými logovacími údaji (login, heslo) pro soutěžící a pedagogy. Uživatelé si mohou pouze v rámci svých termínů kontrolovat průběžný stav bodování, časový harmonogram soutěže daného termínu, nahlédnout do seznamu všech soutěžících a hodnotících zadaných úloh včetně kontaktů.

Dalším typem uživatele je správce, který může editovat seznamy soutěžících a učitelů, hodnotitelé jednotlivých soutěžních disciplín, kteří mohou hodnotit pouze zadané disciplíny. Poslední role je admin, který má přístup do všech editovatelných i výpisových formulářů. Admin má také možnost měnit na základě časových parametrů soutěže čas zobrazení celkových výsledků soutěže pro soutěžící a učitele.

Současný systém je psaný v kombinaci PHP, HTML a JavaScript jazyce, databázový systém a FTP je uložen na serverech společnosti WEDOS.cz.

2.3 Požadavky na aplikaci

Na soutěž se každým rokem přihlašuje stále více týmů a škol. S tím roste také organizační náročnost pro pořadatele se současným systémem. Z toho důvodu vznikl požadavek na nový návrh databáze a aplikace. V rámci pořádání soutěže vznikly požadavky na nové funkcionality a vylepšení systému.

Registrace nového uživatele (student, učitel) by měla probíhat samostatně mimo nutnost zásahu admina ručně. V rámci registrace je pak třeba každého registrovaného uživatele ověřit, aby se mohl přihlásit do systému. Správa porotců a správců zůstane v kompetenci admina. Při registraci se informace uloží do databáze. Heslo uživatele musí být uloženo pomocí vytvořeného hash klíče a kryptografické soli (salted hash), což znamená, že heslo je transformováno na hash hodnotu hesla, aby byla zajištěna lepší ochrana a bezpečnost uživatelských dat.

Po registraci a ověření učitele by měl uživatel zadat předběžné informace o účasti v soutěži (počet týmů, termín a účast v doplňkové disciplíně). Tyto předběžné informace pak budou sloužit pro stanovení limitu počtu týmů v každé škole a také pro případné zaslání vzorků do dané školy. Škola pak má parametry, zda má vzorky, případně jaké.

Z důvodu konání soutěže ve dvou zemích byla navrhnuta možnost mít aplikaci v obou jazycích zemí. Výsledky by měly jazyk dle místa konání a budou rozděleny na část hlavní soutěže a doplňkové disciplíny. Díky přechodu na nový systém se bude

měnit i bodování úloh. V současném systému se body násobily a následně sečetly, což nebylo přehledné. V návrhu bude mít každá úloha maximální počet bodů a tyto body se budou už jen sčítat do celkového počtu.

Od nové role správce se očekává stejných práv jako administrátor (mimo hodnocení soutěže). Správci budou moct upravovat data, nahlížet do databáze a exportovat data.

Vzhled webové aplikace by měl obsahovat určitou škálu barev dle současných webových stránek firmy SMC. To samé platí pak i pro použití fontů a velikosti písma atd.

3 POUŽITÉ TECHNOLOGIE A DALŠÍ MOŽNOSTI

3.1 Programovací jazyky

V dnešní době je tvorba webových aplikací nezbytnou součástí digitálního světa. Při vývoji těchto projektů je klíčové správně vybrat vhodný programovací jazyk. Existuje spousta různých jazyků, které lze použít jak pro backend, tak i pro frontend vývoj. Při výběru je důležité zvážit jejich vlastnosti, komunitu, dostupné nástroje a požadavky pro konkrétní projekt.

3.1.1 Backend

Backend webových aplikací je zodpovědný za zpracování dat, interakci s databází a poskytování informací pro frontend. Mezi nejpoužívanější programovací jazyky patří:

- Python,
- JavaScript (s využitím Node.js),
- Ruby,
- PHP,
- Java.

Python se vyznačuje jednoduchou syntaxí a silnou komunitou, což z něj činí oblíbenou volbu pro vývojáře. JavaScript s Node.js umožňuje použití stejného jazyka jak pro backend, tak i pro frontend, což usnadňuje synchronizaci a spolupráci mezi oběma částmi aplikace. Ruby a jeho framework Ruby on Rails jsou známé pro svou produktivitu a rychlost vývoje. PHP a Java jsou tradiční volbou pro větší a komplexnější projekty. [3]

Python

Pro tuto aplikaci byl pro backend zvolen programovací jazyk Python díky jednoduchosti implementace a předchozích zkušeností.

Python je moderní programovací jazyk s jednoduchou syntaxí, který umožňuje snadné navrhování jak jednoduchých, tak i rozsáhlejších programů. Jeho popularita neustále roste, a stává se klíčovým jazykem v různých oblastech jako výuka, výzkum, statistika, umělá inteligence, skriptování a webové programování. Python je ideální pro ty, kteří chtějí rychle a efektivně vytvářet programy k řešení svých problémů, a zároveň poskytuje dostatečné možnosti pro vývoj rozsáhlejších aplikací. [4]

3.1.2 Frontend

Frontend je část webové aplikace, která se zabývá tvorbou uživatelského rozhraní webových aplikací. Zahrnuje vytváření a design webových stránek, které uživatelé vidí a interagují s nimi ve svém prohlížeči. Kód frontendu je spouštěn přímo ve webovém prohlížeči uživatele. Základní jazyky pro frontend z textu jsou: [3]

- HTML/CSS – používají se k vytváření struktury a stylizaci webových stránek,
- JavaScript – dynamický jazyk používaný k interaktivitě a funkcionalitě webových stránek,
- TypeScript – nadmnožina jazyka JavaScript s typovou kontrolou, která zlepšuje jeho spolehlivost a strukturu.

V dnešních moderních prohlížečích je implementován nízkoúrovňový (assembly-like) jazyk WebAssembly, díky kterému je možné spouštět frontendový kód napsaný v různých jazycích. WebAssembly kompiluje tyto jazyky do formátu, který běží efektivně v prohlížeči s téměř nativním výkonem. To umožňuje rychlejší a plynulejší provádění výpočetně náročných operací. [5]

3.2 Webové frameworky

Python frameworky jsou nástroje, které usnadňují vývoj webových aplikací v jazyce Python. Tyto frameworky poskytují strukturu a sadu nástrojů, které vývojáři mohou využít k vytváření robustních a efektivních aplikací. Mezi populární Python frameworky patří: [6]

- Flask – tzv. mikroframework, který se vyznačuje svou lehkostí a modularitou. Je vhodný pro implementaci menších projektů a umožňuje snadné a rychlé přizpůsobení se různým potřebám. Je využíván společnostmi jako LinkedIn a Pinterest,
- Django – open source webový framework, který je vhodný pro vývojáře všech úrovní. Je známý díky svému ORM systému, automatickému generování administrativního panelu nebo integrovaným webovým serverem pro ověření aplikace během vývoje. Je využíván např. pro Instagram nebo Pinterest,
- Pyramid – populární mezi vývojáři díky své flexibilitě a transparentnosti. Umožňuje mapování URL adres a generování HTML struktury. Používají ho např. společnosti Mozilla a Dropbox,
- Web2Py – full-stack open-source framework, který umožňuje rychlé, škálovatelné, bezpečné a přenosné webové aplikace. Je vybaven integrovanou podporou pro HTTP odpovědi, cookies a relace
- TurboGears, CherryPy, Bottle, Tornado (Facebook) a další.

Pro tuto aplikaci je používán framework Flask díky jeho širokým možnostem rozšíření a podpoře mnoha knihoven.

3.3 Relační databáze

Základem pro práci s daty na souborové úrovni je přístup k souborům, který je realizován pomocí systémů řízení souborů. Tyto systémy umožňují ukládání dat formou záznamů na vnějších pamětích a používají různé techniky indexace k rychlému přístupu k datům. Nicméně zpracování dat na této úrovni má své nevýhody:

- redundance a nekonzistence dat (pokud jsou data ukládána různými aplikacemi zvlášť ve svých souborech, vznikají duplicitní data),
- obtížný přístup k datům pro uživatele,
- izolace dat v různých souborech,
- problémy s ochranou a integritou dat.

Hlavní nevýhodou je omezenost možností vytváření vazeb mezi záznamy, což může vést k technicky složitým částem kódu a prodlužování doby tvorby programu. Proto je v takových případech vhodné použití relační databáze. [7]

Databáze jsou jako digitální úložiště, která mohou zahrnovat od jednoduchých seznamů dat po složité systémy. Tyto systémy modelují část reálného světa nazývanou prostor problému a pomáhají organizovat data a vztahy mezi nimi. Datový model popisuje entity, jejich atributy a vztahy, zatímco databázové schéma definuje fyzické uspořádání dat. Když systému řekneme, jak chceme, aby data vypadala, vytvoří fyzické objekty pro jejich uložení a správu. Databáze zahrnují tabulky, pohledy, dotazy a uložené procedury, které pomáhají organizovat a chránit data. [8]

V dnešní době se organizace stále více stávají datově řízenými. Výběr správné databáze může výrazně zefektivnit tok dat a přinést konkurenční výhodu. Existuje mnoho typů databází např.: [9]

- MySQL – open-source relační databázový systém (podpora PHP),
- PostgreSQL – open-source objektově-relační databázový systém (volně dostupný, SQL kompatibilita),
- MongoDB – open-source dokumentově orientovaná databáze,
- Oracle database – široce používaný relační databázový systém (vysoký výkon),
- SQLite – malá a rychlá open-source databáze (nevyžaduje instalaci ani server, ideální pro mobilní vývoj),
- Redis – open-source in-memory key-value databáze (často používán pro cache management),
- MariaDB – relační databázový systém kompatibilní s MySQL protokolem a klienty.

3.3.1 MySQL

Je to nejpopulárnější open source databáze na světě a používá se ve významných webových aplikacích, jako jsou Facebook, Twitter, YouTube a další. MySQL nabízí několik produktů a cloudových služeb pro správu dat, včetně modulu MySQL HeatWave pro analýzu v reálném čase a strojové učení. [10]

3.3.2 SQLite

SQLite je databázový systém, který implementuje samostatný a transakční SQL databázový engine, který nevyužívá server. Je snadno použitelný a vhodný pro vestavěné aplikace a jednoduché projekty. Má výhody, jako je rychlý výkon, spolehlivost a přenositelnost. Mezi jeho nevýhody patří omezená konkurence a chybějící správa uživatelů. [11] Tento typ databáze byl využit pro počáteční vývoj aplikace, poté jej nahradila databáze MariaDB.

3.3.3 PostgreSQL

PostgreSQL je pokročilý open-source relační databázový systém. Podporuje jak SQL (relační) dotazování, tak i JSON (ne-relační) dotazování. PostgreSQL je díky komunitě velmi stabilní databáze s více než 20 lety vývoje. Je používán jako primární databáze pro mnoho webových aplikací, mobilních aplikací a analytických aplikací (Apple, Cisco, Red Hat a další). [12] Tato databáze byla díky vlastnímu nástroji pgAdmin pro správu a vývoj databáze hned druhou možností pro použití ve vlastním návrhu aplikace.

3.3.4 MariaDB

MariaDB je open-source relační databázový systém, který vznikl jako odnož (fork) původního MySQL. Systém zvládá obrovské množství dat se zachováním vysoké rychlosti a spolehlivému zabezpečení. Zachovává si vysokou kompatibilitu s MySQL díky tomu, že na jeho vývoji pracují původní tvůrci MySQL. [13] Jedním z původních zakladatelů MySQL je Michael Widenius, který později inicioval vývoj MariaDB skrz obavy z akvizice MySQL společností Oracle [14].

Databáze MariaDB je používána pro ukládání veškerých dat webové aplikace.

3.4 Použité knihovny

3.4.1 Flask-Login

Flask-Login je knihovna poskytující správu uživatelských relací pro framework Flask. Zvládá běžné úkoly přihlašování, odhlašování a pamatování si relací uživatelů po delší dobu. Tato knihovna umožňuje:

- ukládání ID aktivního uživatele (v rámci Flask relace), řešení funkcionality „remember-me“,
- snadné přihlašování a odhlašování,
- omezení zobrazení stránek pro přihlášené nebo odhlášené uživatele (dekorátor `@login_required`),
- ochranu relací uživatelů před krádeží souborů cookie.

Tato knihovna již neřeší způsoby ověření uživatele (Werkzeug), ukládání uživatelů, přidání nových uživatel či rozšířená práva (role). Knihovna se konfiguruje pomocí nástroje `LoginManager`. [15]

3.4.2 Werkzeug

Werkzeug je komplexní knihovna pro webové aplikace WSGI [16]. Pro tuto aplikaci je použita pouze část knihovny `werkzeug.security`, která se věnuje bezpečnosti. Konkrétně jsou zde využity dvě funkce:

- `generate_password_hash`,
- `check_password_hash`.

První funkce je pro bezpečné generování hash hodnoty zadaného hesla. Tato zašifrovaná hodnota se pak může ukládat do databáze. Výchozí metodou pro hash hesla je pro tuto funkci metoda scrypt (konkrétní parametry jsou `scrypt:32768:8:1`, tento řetězec je obsažen ve všech vygenerovaných hash hodnotách hesel).

Druhou funkcí je `check_password_hash`, kterou se kontroluje, zda heslo zadané odpovídá hash hodnotě uložené v databázi. Je použita například pro přihlášení.

3.4.3 SQLAlchemy

SQLAlchemy je nástroj a ORM (Object Relational Mapper) pro Python, který umožňuje vývojářům efektivně pracovat s databázemi. Poskytuje sadu vzorů pro ukládání dat a přístup k nim, což usnadňuje psaní databázového kódu. Mezi hlavní funkce SQLAlchemy patří:

- výkonný ORM – umožňuje jednoduchou správu databázových objektů (vkládání, aktualizace a mazání dat),
- vlastní dotazovací systém – umožňuje vytvářet složité SQL dotazy pomocí Pythonu (např. spojování tabulek, poddotazů a dalších SQL operací).
- načítání dat – flexibilní systém pro načítání souvisejících dat a kolekcí, které mohou být načteny při prvním přístupu nebo najednou,
- Core systém – samostatná vrstva pro práci s SQL, která zahrnuje vytváření SQL příkazů, správu připojení a typy dat.

SQLAlchemy umožňuje plnou kontrolu nad strukturou dotazů a návrhem databázového schématu, aniž by generovala špatné dotazy. Umožňuje také introspekci databází a generování SQL příkazů z Python struktury (API založené na method chaining). ORM se používá pro mapování komplexních objektů nebo pokud je preferován Python kód místo SQL dotazů. Naopak se nepoužívá v případech potřeby vyššího výkonu, kontroly generovaných SQL dotazů nebo např. pro jednodušší projekty, kde by bylo použití ORM nadbytečné. Knihovna podporuje správu transakcí, kde změny nejsou uloženy, dokud není zavolána funkce `commit()`. Tím se zajišťuje bezpečnost a efektivita práce s databázemi. [17]

V rámci aplikace je dodatečně zahrnuta i knihovna Flask-SQLAlchemy, která poskytuje integraci s knihovnou SQLAlchemy. Toto rozšíření zjednodušuje práci s databázemi v aplikacích Flask tím, že poskytuje jednoduchý způsob definice databázových modelů a manipulace s nimi.

V aplikaci se využívá funkce chaining (např. v obr. 17), která umožňuje vytvářet komplexní dotazy přidáváním více metod za sebe, což vede k čitelnějšímu a strukturovanějšímu kódu. Tento způsob je obzvláště užitečný pro vytváření složitých SQL dotazů. Umožňuje postupné budování dotazu, kde každá metoda modifikuje a rozšiřuje předchozí část.

3.4.4 Flask-Migrate

Knihovna Flask-Migrate je rozšíření pro Flask, které zajišťuje migrace databází SQLAlchemy pomocí nástroje Alembic. Mezi hlavní výhodu patří integrace Flasku s Flask-SQLAlchemy, což usnadňuje nastavení a používání migrací v rámci Flask aplikací. Migrace databáze se provádí přes příkazový řádek Flasku. Nejprve se migrace inicializuje příkazem `flask db init`, pak je příkazem `flask db migrate` vytvořen skript pro migraci databáze (v jazyku Python) a následně je vyvolán pomocí příkazu `flask db upgrade`. Jakmile je takto databáze vytvořena, tak se pro případné změny v modelu databáze opět musí vytvořit nový migrační skript a poté je potřeba jej vyvolat příkazem `flask db upgrade`. [18]

3.4.5 Flask-Admin

Pro implementaci administrátorského rozhraní bylo třeba zvážit různé možnosti pro danou aplikaci a definovat požadavky na toto prostředí (funkční a vzhledová stránka).

Pokud je potřeba mít vlastní vzhled a funkčnosti rozhraní, bylo by lepší volbou ruční vytvoření všech požadovaných stránek, formulářů a jiných funkcionalit dle potřeb. Nabízí se také možnost použití frontendové administrace s knihovnami jako React, Vue nebo Angular. Pak jsou tu knihovny Flask-Admin nebo Django-Admin, které jsou omezeny na použití aplikace Flask nebo Django. Webová aplikace používá framework Flask, a proto je voleno rozšíření Flask-Admin. Tato volba splňuje všechny požadavky na toto rozhraní.

Flask-Admin umožňuje vytvářet složitá rozhraní tím, že seskupuje jednotlivé pohledy do tříd. Každá webová stránka v rozhraní odpovídá pohledovým modelům (`ModelView`). Pohledový model je třída, která definuje, jak jsou data zobrazena a editována ve Flask-Admin rozhraní. Tyto pohledové modely jsou pak spojeny s konkrétními databázovými modely, protože umožňují seskupit veškerou logiku pro vytváření, čtení, aktualizaci a mazání záznamů do jedné samostatné třídy pro každý model. [19]

3.4.6 Flask-Babel

Tato knihovna je použita pouze pro překlad stránky administrátorského rozhraní. Jinou možností byl ruční překlad všech stránek, nicméně Flask-Babel poskytnul jednoduchou implementaci pro tento případ. V definici této knihovny je uveden výchozí jazyk, který se pak používá pro celé rozhraní knihovny Flask-Admin. Původní knihovna se jmenovala dříve Flask-Babelex, ale s novou verzí a díky širšímu použití knihovny se tento název změnil na Flask-Babel.

3.4.7 HTML knihovny

Cílem této práce je navíc vytvoření mobilní aplikace, což je realizováno díky responzivnímu designu webových stránek. Při zobrazení na mobilním zařízení se stránky přizpůsobují danému rozlišení zařízení a uživatel je takto schopen ovládat aplikaci i ze svého mobilního telefonu.

Pro realizaci takto přizpůsobitelných stránek je použito open-source frameworku **Bootstrap**. Díky rozsáhlé knihovně předpřipravených komponentů, jako jsou tlačítka, formuláře, navigační panely, modální okna a další, umožňuje Bootstrap rychlý vývoj webových stránek bez nutnosti psaní vlastního kódu. Navíc poskytuje podporu pro všechny hlavní prohlížeče, což zajišťuje konzistentní zobrazení a funkčnost na různých platformách. [20]

Pro vykreslení ikon k odkazům v navigační liště je použita knihovna ikon **FontAwesome**. Knihovna obsahuje širokou škálu různých ikon pro veškeré kategorie aplikací. Ikony lze snadno integrovat do webového prostředí, vykreslují se jako vektorový obrázek a jsou responzivní pro zobrazení na různých zařízeních.

V administrátorském prostředí jsou pomocí dynamických grafů vykresleny data z databáze. Grafy jsou sestrojeny pomocí knihovny **ApexCharts**, která poskytuje širokou škálu typů grafů (sloupcové, čárové, plošné, koláčové, radarové, scatter ploty a další). Import knihovny probíhá přidáním odkazu v souboru HTML.

Pro stránky je integrována JavaScript knihovna **jQuery**, která usnadňuje manipulaci s HTML šablonami. Pro odesílání nebo načtení dat slouží pak AJAX (Asynchronous JavaScript and XML) požadavky, které lze v rámci knihovny využívat.

3.4.8 WTForms

Pro specifické požadavky jsou pole definována pomocí vlastností polí z knihovny WTForms. Používají se tyto vlastnosti polí [21]:

- DataRequired – validátor, který zajišťuje, že pole nemůže být prázdné,
- StringField – pole pro textové vstupy, jako jsou jména nebo emailové adresy,
- PasswordField – pole pro heslo, které skryje zadaný text, aby nebyl viditelný,
- SelectField – rozbalovací seznam pro výběr jedné možnosti z předdefinovaného seznamu,
- QuerySelectField – rozbalovací seznam, který je doplněn daty na základě dotazu na databázi, umožňuje výběr jedné možnosti,
- QuerySelectMultipleField - rozbalovací seznam, který je doplněn daty na základě dotazu na databázi, umožňuje výběr více možností.

3.4.9 Openpyxl

Pro export dat je v aplikaci implementována knihovna **Openpyxl**, která slouží k čtení a zápisu souborů Microsoft Excel s příponou `.xlsx` (formát OpenXML). Je široce využívána pro svou schopnost manipulovat s Excel soubory s použitím skriptu (vytváření nových sešitů, přidávání listů, zápis dat do buněk, formátování buněk, vytváření grafů a mnoho dalších funkcí). V programu se vytváří sešit, do něj se pak doplní data do buněk zvlášť do listů a následně se soubor uloží.

3.4.10 BytesIO

BytesIO je třída z modulu `io` v Pythonu, který pracuje s různými typy vstupů a výstupů. Existují tři hlavní typy I/O: textové I/O, binární I/O a surové I/O. **BytesIO** se používá pro práci s binárními daty, jako jsou soubory ve formátu JPEG, PDF nebo Excel, které mohou být zpracovány a přenášeny v paměti bez nutnosti zápisu na disk. Tato třída je zvláště užitečná v aplikacích, které vyžadují dočasné uchovávání dat v binární podobě, jako jsou webové aplikace nebo služby API. [22]

V této aplikaci je třída BytesIO použita pro vytvoření dočasného úložiště pro Excel soubor, který je generován a odesílán uživateli jako příloha. Toto řešení umožňuje přenos souboru přímo v odpovědi HTTP, aniž by bylo nutné ukládat soubor na disk.

3.5 Ochrana údajů

Otázka ochrany soukromí a bezpečnosti uživatelů je ústřední pro problematiku zabezpečení dat. S rozvojem technologií a stále širším sdílením dat se rozsah úniků soukromých informací může stát velmi rozsáhlým. Identifikace odpovědných stran za úniky soukromí je náročná, zejména kvůli využívání různých datových kombinací, které ztěžují určení odpovědnosti. Jednotné datové formáty mohou problém ještě zhoršit, protože zranitelnosti v těchto formátech mohou být zneužity napříč mnoha platformami, což činí úniky dat závažnějšími a rozšířenějšími. Řešení těchto problémů s ochranou soukromí a bezpečností je proto klíčové pro zajištění důvěry uživatelů v nové technologie. [23]

4 VLASTNÍ NÁVRH APLIKACE

Nejprve bylo potřebné ustanovení požadavků a cílů pro vlastní návrh aplikace. Tato fáze zahrnovala určení všech funkcí aplikace, potřeb uživatelů a technických požadavků, které musí být splněny. Následně byla navržena architektura aplikace, která zahrnuje rozhodnutí o použití Flask frameworku, volbu databáze a dalších technologií a knihoven.

4.1 Flask aplikace

Po zvolení Flasku je framework nainstalován spolu s dalšími potřebnými závislostmi. Poté je vytvořena adresářová struktura projektu, která organizuje kód aplikace a umožňuje snadnou správu souborů. Hlavní skript aplikace `__init__.py` inicializuje Flask aplikaci. Samotná aplikace se spouští souborem `app.py` v hlavní složce aplikace. Pro postupnou tvorbu a testování aplikace je zapnut mód `debug`, který restartuje aplikaci při jakékoliv změně kódu. Další skripty a potřebné soubory jsou v podsložce `website`.

Při spouštění aplikace se v Pythonu rozlišuje situace, jak byla aplikace spuštěna. Toto rozpoznání režimu je klíčové pro správné chování aplikace. Pokud je modul spuštěn jako samostatný skript, stává se hlavním modulem a je označen jako `main`. To znamená, že je vhodné v kódu definovat podmíněný příkaz, který určuje, co se má provést při přímém spuštění daného modulu (obr. 1). Tím se zajistí správné fungování aplikace v různých situacích. Když by se modul importoval pomocí `import`, je modul spuštěn v interaktivním režimu a jeho jméno odpovídá jménu zdrojového souboru. [4]

```
app = create_app()

if __name__ == '__main__':
    app.run(debug = True)
```

Obr. 1: Podmíněný příkaz pro přímé spuštění modulu

Různé části aplikace jsou odděleny s použitím tříd `Blueprint`. Jsou zde dva skripty, které obsahují URL adresy a cesty pro určitou část aplikace. Prvním souborem je `auth.py`, ve kterém probíhají všechny akce související s autorizací uživatele (přihlášení, odhlášení, registrace a sekce profilu). Ve druhém souboru `views.py` jsou pak definovány všechny ostatní funkce aplikace. Pro vykreslování stránek na webu je použito šablon HTML. Šablony jsou vytvořeny pomocí Jinja2 templating enginu a slouží k vykreslování dynamických stránek (na serveru).

4.1.1 Inicializace

V rámci inicializace proběhne nejprve import všech knihoven, inicializuje se databáze pomocí SQLAlchemy. Dále pokračuje funkce `create_app` pro vytvoření Flask aplikace. V rámci funkce pak probíhá:

- konfigurace aplikace (tajný klíč pro zabezpečení, přístup k databázi),
- import dalších skriptů pro různé části aplikace (`Blueprint`),
- import všech tabulek z databáze a jejich pohledů pro administrátorské rozhraní,
- konfigurace pro administrátorské rozhraní (výchozí jazyk a vzhled) a pak zobrazení potřebných tabulek a odkazů v rozhraní,
- samotná inicializace aplikace a aktualizace databáze, probíhá kontrola záznamů rolí a existence administrátora v tabulkách uživatelů,
- definice nástroje `LoginManager` knihovny Flask-Login (nastavení výchozí stránky pro nepřihlášeného uživatele). Funkce `load_user` pak vrací ID přihlášeného uživatele,
- jsou definovány dvě funkce `inject_terminy a inject_celkove_terminy`, které v rámci dekorátoru `@app.context_processor` vrací výstup, který je přístupný ve všech šablonách aplikace. Tímto způsobem se na frontend posílají data o všech termínech a stav funkce zobrazení celkových výsledků na všechny stránky.

4.2 Návrh databáze

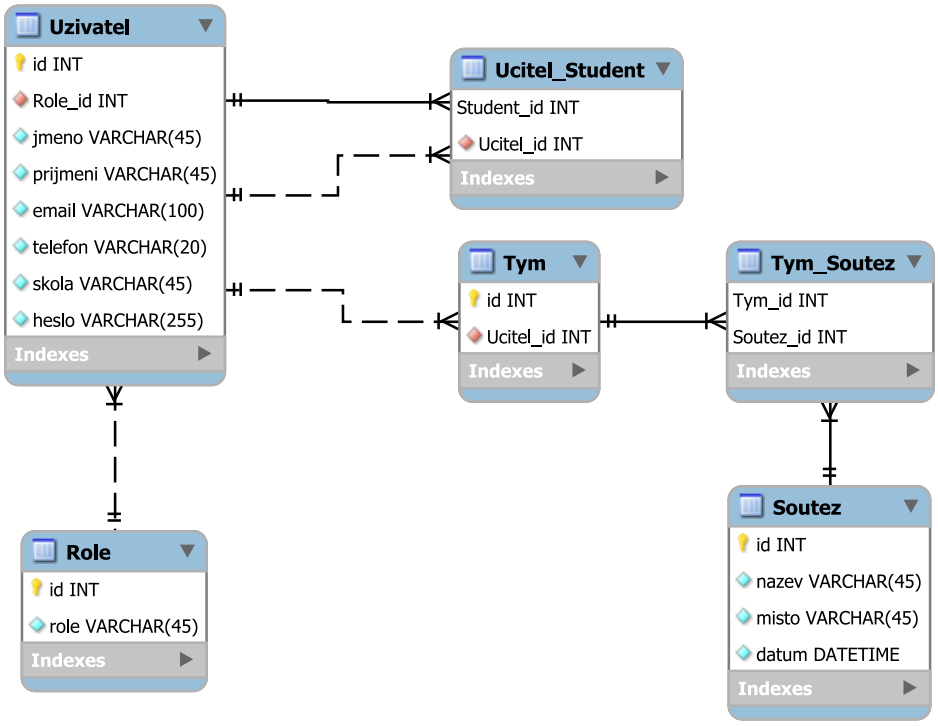
Z prvních informací o daném systému a dle zadaných požadavků byl systematicky vypracován návrh databáze. Prvním krokem bylo vytvoření tabulky s názvem `User`, kde jsou veškeré informace o daných uživatelích. Hlavními sloupci v této tabulce jsou:

- `id` (integer) – primární klíč,
- `role_id` (integer) – cizí klíč odkazující na tabulku role (`Role`),
- `jmeno`, `prijmeni`, `email`, `telefon`, `skola` a `heslo` (string).

Dále byla doplněna databáze o tabulku role (`Role`), která odkazuje na již zmíněnou tabulku uživatele (`User`) pomocí primárního klíče – `id`. Hlavním parametrem je pak její samotný název. Zde bude uveden přímo název dané role (Student, Učitel, Porotce, Admin).

V dalším kroku bylo navrhnuto tabulky tým (`Tym`), vazba učitel-student (`Ucitel_Student`), vazba tým-soutěž (`Tym_Soutez`) a samotná soutěž (`Soutez`). Studenti jsou začleněni do týmů prostřednictvím pomocné tabulky `Ucitel_Student`, což umožňuje spojení s učitelem, který má svůj tým. Týmy jsou poté registrovány do určitých soutěží skrze další pomocnou tabulku `Tym_Soutez`. Tento návrh tabulek a vazeb mezi nimi vycházel z prvotních informací k danému systému a sloužil jako výchozí bod pro testování vkládání dat do databáze a následnou manipulaci s nimi.

Pro každý návrh byl sestrojen diagram pojmenovaný jako ERD (Entity Relationship Diagram). Diagram znázorňuje strukturu databáze pomocí vykreslení tabulek a vztahů mezi jednotlivými sloupci.



Obr. 2: ER diagram prvního návrhu databáze

Tento návrh databáze (obr. 2) byl sestrojen pomocí aplikace MySQL Workbench. Tato aplikace byla zvolena z důvodu předešlé výuky právě v této aplikaci. MySQL Workbench umožňuje snadno vytvářet návrhy databází v podobě diagramů a následně exportovat tyto návrhy do kódu SQL. Tato funkce byla využita a následně byl kód SQL zpracováván do vlastního návrhu aplikace.

Při studování možností importu kódu databáze do Flask aplikace se nabízela alternativa v podobě knihovny SQLAlchemy. Díky této knihovně se dá v Pythonu navrhovat databáze za běhu aplikace, je možnost ji jednoduše měnit a kód je čitelný. Volil jsem tedy tuto variantu a dosavadní návrh modelu byl přepsán pomocí dané knihovny do samostatného souboru `models.py`. V tabulkách databáze jsou pro sloupce definovány tyto typy dat:

- integer – data jsou ve formátu celočíselné hodnoty,
- string – data jsou ve formátu textového řetězce s proměnnou nebo pevnou délkou,
- boolean – data označující stavy pravda/nepravda (True/False),
- date – data obsahující datumové hodnoty.

Uvedený SQL kód v obr. 3 vybere všechny sloupce z tabulky `Uloha`, provádí spojení s tabulkou `Porotce_Uloha` na základě podmínky `Uloha.id = Porotce_Uloha.uloha_id` a následně filtrování výsledků tak, aby `Porotce_Uloha.porotce_id` = `current_user.id`. Za `current_user.id` se dosazuje hodnota ID aktuálně přihlášeného uživatele.

```
SELECT *
FROM Uloha
JOIN Porotce_Uloha ON Uloha.id = Porotce_Uloha.uloha_id
WHERE Porotce_Uloha.porotce_id = 'current_user.id'
```

Obr. 3: Kód SQL

Další kód z obr. 4 je ekvivalentní původnímu dotazu SQL a je vytvořen pomocí knihovny SQLAlchemy (je využita funkce chaining). Rozdíl je v tom, že SQLAlchemy vytváří parametrizovaný dotaz, což je bezpečnější z hlediska prevence SQL injection a umožňuje snadněji pracovat s proměnnými.

```
ulohy_porotce = Uloha.query\
    .join(Porotce_Uloha, Uloha.id == Porotce_Uloha.uloha_id)\
    .filter(Porotce_Uloha.porotce_id == current_user.id).all()
```

Obr. 4: Kód SQLAlchemy

Když by se vypsal dotaz SQLAlchemy pomocí klasického SQL, je zde uvedena hledaná hodnota `?` místo `current_user.id`. To je způsob, jak SQLAlchemy značí parametr, který bude nahrazen reálnou hodnotou, jako je například `current_user.id`.

4.2.1 SQLite

Model databáze je nezbytné propojit s Flask aplikací v inicializačním kódu pomocí konfiguračního skriptu. Nejprve se definuje použití SQLAlchemy a poté specifikuje cesta k databázi. Následně je pak potřeba databázi vytvořit pomocí příkazu `db.create_all()`.

S tímto krokem je taky potřeba zvolit konkrétní typ databáze. S ohledem na jednoduchost použití byla pro prvotní testování zvolena SQLite databáze. Tento typ databáze ukládá všechna data do jediného souboru na disku, což pak usnadňuje její použití a přenositelnost. Cesta k tomuto souboru je uvedena v konfiguračním skriptu. Aplikace mohla být následně spuštěna.

Během testování byly pak pomocí příkazů knihovny SQLAlchemy přidány do databáze záznamy jak uživatele, tak i základní role. Poté se model rozšiřoval dle původního návrhu databáze. Model byl postupně komplexnější a při provádění operací nastaly problémy ohledně vazeb mezi tabulkami. Tyto obtíže souvisely s typem databáze SQLite a bylo nutné typ databáze změnit.

Nabízely se možnosti databází MySQL, PostgreSQL nebo MariaDB. na základě doporučení vedoucího práce byla vybrána varianta databáze MariaDB.

4.2.2 MariaDB

Po instalaci databázového serveru MariaDB bylo potřeba server nastavit a definovat pro následné použití na lokálním serveru (adresy, porty, uživatelské jméno, heslo). V kontextu inicializace aplikace a konfiguračního skriptu bylo upraveno napojení na databázi běžící na pozadí lokálního počítače (localhost).

V rámci instalace byl pak ještě instalován program HeidiSQL, který byl doporučen vedoucím práce a současně se instaloval jako doplněk k MariaDB. Po nastavení programu a napojení na databázi lze jednoduše procházet všechny tabulky databáze, včetně všech dat uložených v nich. Data i databázi lze jakkoliv upravit jak ručně, tak i s pomocí SQL příkazů. Postupně se tak mohlo kontrolovat, zda je model vytvořen správně a odpovídá záměrům funkce aplikace.

Díky tomuto programu lze do databáze importovat větší množství dat, které byly pro testování aplikace nezbytné. Pro tyto účely se využilo nástroje Data Generator [24], kde se vygenerovaly náhodná jména a příjmení uživatelů, následně v Excelu přiřadily další vlastnosti dle struktury tabulky uživatelů (`User`). Data se pak importovaly pomocí HeidiSQL do databáze. Pro názornou ukázku a testovaní stačilo mít těchto záznamů 100, další se pak přidávaly už v rámci samotné aplikace (registrace, administrátor).

Tabulka rolí (`Role`) má určitý počet záznamů a pro zachování funkčnosti aplikace je potřeba, aby se tyto role nepřepsaly, či nějak nezměnily. Pro zajištění této skutečnosti je v rámci inicializace aplikace spuštěna funkce pro kontrolu rolí. V této funkci se definují role dle ID a názvu. Když by nastala nějaká chyba či změna, tak po inicializaci aplikace se vždy tabulka rolí obnoví do původního stavu definovaného v této funkci.

S podobným záměrem je zde i funkce pro hlídání, zda je administrátor v seznamu jako uživatel. Pokud by nějakým způsobem došlo k vymazání jeho účtu, opět se touto funkcí zajistí, aby byl učet zachován v tabulce uživatelů (s výchozími údaji).

Pro další úpravy a vylepšování verzí databáze se začala používat knihovna Flask-Migrate. Pomocí této knihovny se vždy po nějaké úpravě modelu databáze používá příkaz `flask db migrate`. Pak se úprava následně pomocí `flask db upgrade` nahraje a vytvoří nová verze databáze.

Program HeidiSQL byl používán pro zobrazení všech tabulek databáze. Všechna data lze kontrolovat a případně upravovat. Úpravu lze provádět manuálně nebo pomocí SQL kódu. V rámci aplikace lze provést export i import do databáze.

4.2.3 Struktura databáze

Dle dalších požadavků byl model databáze postupně upravován. Vznikl nový návrh databáze (obr. 7), kde jsou tabulky: uživatelé (`User`), role (`Role`), školy (`Skola`), vzorky (`Vzorky`), týmy (`Tym`), vazby týmů a studentů (`Tym_Student`), termíny (`Termin`), úlohy (`Uloha`), vazby porotců a úloh (`Porotce_Uloha`), hodnocení (`Hodnoceni`), funkce (`Funkce`). [25]

Uživatel (User)

Třída `User` se dědí od třídy `UserMixin`, která vychází z knihovny Flask-Login, a je zde kvůli správné funkci uživatelského ověřování při přihlášení atd. Obsahuje sloupce:

- `id` (integer) – primární klíč,
- `jmeno` (string) – jméno uživatele,
- `prijmeni` (string) – příjmení uživatele,
- `email` (string) – email uživatele, používá se pro přihlášení,
- `telefon` (string) – telefon uživatele (nepovinný),
- `skola_id` (integer) – cizí klíč, u studentů a učitelů odkazuje na školu (`Skola`), ostatní mají hodnotu NULL (žádnou),
- `heslo` (string) – obsahuje hash hesla uživatele,
- `role_id` (integer) – cizí klíč, odkazuje na roli (`Role`),
- `max_tymu` (integer) – maximální počet týmů učitele, ostatní mají hodnotu NULL,
- `overeny` (boolean) – hodnota, která určuje, zda se může uživatel přihlásit.

Pro přístup k datům z jiných tabulek jsou definovány vztahy `skola`, `role`, `tym`, `vedeni`:

- `skola` – umožňuje přístup na školu (`Skola`) studenta nebo učitele,
- `role` – umožňuje přístup na roli (`Role`) uživatele,
- `tym` – slouží pro studenty a odkazuje na tým (`Tym`), ve kterém je student členem. Pomocí parametru `back_populates=clenove` lze pak přistoupit ke všem členům jakéhokoliv týmu,
- `vedeni` – slouží naopak pro učitele a odkazuje na všechny týmy, které učitel vede. Tento vztah obsahuje parametr `back_populates=ucitel`, který umožňuje přístup k učiteli (`User`) z objektu tým (`Tym`).

Role (Role)

Obsahuje sloupce:

- `id` (integer) – primární klíč,
- `nazev` (string) – název role (Student, Učitel, Porotce, Správce, Admin).

```
class Role(db.Model):
    __tablename__ = 'role'
    id = db.Column(db.Integer, primary_key=True) # 5 rolí
    nazev = db.Column(db.String(255), unique=True, nullable=False)
    def __repr__(self):
        return self.nazev
```

Obr. 5: Kód pro tabulku rolí (`Role`)

Škola (Skola)
V tabulce budou školy, které se můžou do soutěže registrovat. Student a učitel pak při registraci vyberou ze seznamu škol, pod kterou školou se do soutěže hlásí. Záznamy spravuje administrátor a správci. Obsahuje sloupce:

- `id` (integer) – primární klíč,
- `nazev` (string) – název školy,
- `adresa` (string) – adresa školy,
- `vzorky_id` (integer) – cizí klíč odkazující na tabulku vzorků (`Vzorky`),
- `pujcene_vzorky` (string) – popis vzorků, které má škola zapůjčeny,
- `stat` (string) – definuje, ze kterého státu škola a uživatelé s ní spojení pochází,
- `info_pocet_tymu` (integer) – předběžný počet soutěžních týmů,
- `info_poslat_vzorky` (boolean) – zda chce škola poslat vzorky,
- `info_termin_id` (integer) – předběžný termín účasti školy (cizí klíč).

Původním záměrem bylo mít stát u každého uživatele, ale nakonec se tento parametr přesunul do tabulky školy (`Skola`). Dále je tady doplněn vztah `vzorky` pro přístup k datům v tabulce vzorky (`Vzorky`) a vztah `info_termin` pro přístup k odpovídajícímu termínu (`Termin`).

Vzorky (Vzorky)
Tabulka je neměnná a slouží k předání parametru `nazev` při výpisu informací o školách. Obsahuje sloupce:

- `id` (integer) – primární klíč,
- `nazev` (string) – obsahuje řetězce: Nezaslány, Zapůjčeny, Vráceny.

Tým (Tym)
Obsahuje sloupce:

- `id` (integer) – primární klíč,
- `ucitel_id` (integer) – cizí klíč odkazující na učitele (`User`),
- `ucitel_tym` (integer) – číslo určující kolikátý tým je to pro učitele,
- `hlavni_soutez` (boolean) – určuje, zda se tým účastní hlavní soutěže,
- `vedlejsi_soutez` (boolean) – určuje, zda se tým účastní vedlejší soutěže,
- `termin_id` (integer) – cizí klíč odkazující na termín (`Termin`), kterého se tým účastní,
- `cislo_tymu` (integer) – soutěžní číslo týmu dle termínu, přidělí admin/správci,
- `skupina` (integer) – číslo skupiny dle harmonogramu, přidělí admin/správci.

Pro přístup k datům z jiných tabulek a omezení mazání propojených záznamů jsou definovány vztahy `termin`, `ucitel`, `clenove`, `tym_student`, `hodnoceni`:

- `termin` – umožňuje přístup na termín (`Termin`), kterého se tým účastní,
- `ucitel` – umožňuje přístup na učitele (`User`), který vede daný tým,
- `clenove` – umožňuje přístup na všechny členy (`User`) týmu,

- `tym_student` – vazba s parametrem `cascade=all, delete-orphan`, díky kterému se smažou i vazby tým-student (`Tym-Student`) odkazující na tým (`Tym`), který byl smazán,
- `hodnoceni` – vazba s parametrem `cascade all, delete-orphan`, díky kterému se smažou i hodnocení (`Hodnoceni`) týmu (`Tym`), který byl smazán.

Vazby tým-student (Tym-Student)
Jde o spojovací tabulku, která definuje spojení mezi týmem (`Tym`) a studenty – členy týmu (`User`). Obsahuje sloupce:

- `tym_id` (integer) – cizí klíč odkazující na tým (`Tym`),
- `student_id` (integer) – cizí klíč odkazující na studenta (`User`).

Oba sloupce jsou zároveň primárními klíči, aby byla zajištěna unikátnost těchto dvojic. Není tedy možná stejná kombinace v tabulce.

Termín (Termin)
Obsahuje sloupce:

- `id` (integer) – primární klíč,
- `datum` (date) – datum konání termínu (unikátní a neprázdné),
- `mesto` (string) – město konání termínu,
- `adresa` (string) – přesná adresa konání termínu,
- `stat` (string) – stát, ve kterém se termín koná (CZ/SK),
- `vysledky` (boolean) – určuje, zda se zobrazí výsledky studentům a učitelům.

Vlastnost `stat` je zde i z informačního důvodu, ale především k zobrazení stránky výsledků daného termínu dle státu v daném jazyce (čeština, slovenština).

Úloha (Uloha)
Obsahuje sloupce:

- `id` (integer) – primární klíč,
- `nazev` (string) – zkratka či popis úlohy,
- `soutez` (integer) – číslo udávající kategorii soutěže (1 – hlavní, 2 – vedlejší),
- `max_pocet` (integer) – číslo udávající maximální počet bodů, které může tým v dané úloze získat.

Dále je tady zaveden vztah `porotci`, která odkazuje na porotce (`User`), kteří jsou spojeni skrz spojovací tabulku vazby porotce-uloha (`Porotce-Uloha`) na danou úlohu (`Uloha`).

Vazby porotce-úloha (Porotce-Úloha)
Jde o spojovací tabulku, která definuje spojení mezi porotcem (`User`) a úlohami (`Úloha`). Označuje schopnost porotce hodnotit danou úlohu. Obsahuje sloupce:

- `porotce_id` (integer) – cizí klíč odkazující na porotce (`User`),
- `uloha_id` (integer) – cizí klíč odkazující na úlohu (`Uloha`).

Oba sloupce jsou opět primárními klíči, aby byla zajištěna unikátnost těchto dvojic – stejná kombinace v tabulce je vyloučena.

Hodnocení (Hodnoceni)

Tabulka, která má vyjadřovat hodnocení týmu (`Tym`) daným porotcem (`User`) v dané úloze (`Uloha`), obsahuje sloupce:

- `id` (integer) – primární klíč,
- `tym_id` (integer) – cizí klíč odkazující na tým (`Tym`),
- `porotce_id` (integer) – cizí klíč odkazující na porotce (`User`),
- `uloha_id` (integer) – cizí klíč odkazující na úlohu (`Uloha`),
- `skore` (integer) – číslo vyjadřující počet bodů, které tým získal v dané úloze.

Pro přístup k datům z jiných tabulek jsou definovány vztahy `tym`, `porotce`, `uloha`:

- `tym` – umožňuje přístup na tým (`Tym`), který je hodnocen,
- `porotce` – umožňuje přístup na porotce (`User`), který vytvořil dané hodnocení,
- `uloha` – umožňuje přístup na úlohu (`Uloha`), která byla hodnocena.

Tyto vztahy jsou vytvořeny z důvodu nutnosti použití v administrátorském rozhraní Flask-Admin.

```
class Hodnoceni(db.Model):
    __tablename__ = 'hodnoceni'
    id = db.Column(db.Integer, primary_key=True)
    tym_id = db.Column(db.Integer, db.ForeignKey('tym.id'), nullable=False)
    porotce_id = db.Column(db.Integer, db.ForeignKey('user.id'),
                           nullable=False)
    uloha_id = db.Column(db.Integer, db.ForeignKey('uloha.id'),
                         nullable=False)
    skore = db.Column(db.Integer, nullable=False)
    porotce = db.relationship('User')
    uloha=db.relationship('Uloha')
```

Obr. 6: Kód pro tabulku hodnocení (`Hodnoceni`)

Funkce (Funkce)

Tato tabulka reprezentuje key-value tabulku uživatelských funkcí. Key-value tabulka v databázi slouží k ukládání a vyhledávání dat na základě jednoduchých klíčů a jejich přidružených hodnot. Nemá žádné vazby na ostatní tabulky. Obsahuje sloupce:

- `id` (integer) – primární klíč,
- `nazev` (string) – název uživatelské funkce,
- `stav` (boolean) – stav uživatelské funkce určující, zda je vypnutá či zapnutá.

Ovládání stavu uživatelských funkcí je v kompetenci administrátora a správců.

4.2.4 ER diagram

Obr. 7: ER diagram navrhnuté databáze [25]

4.3 Stránky aplikace

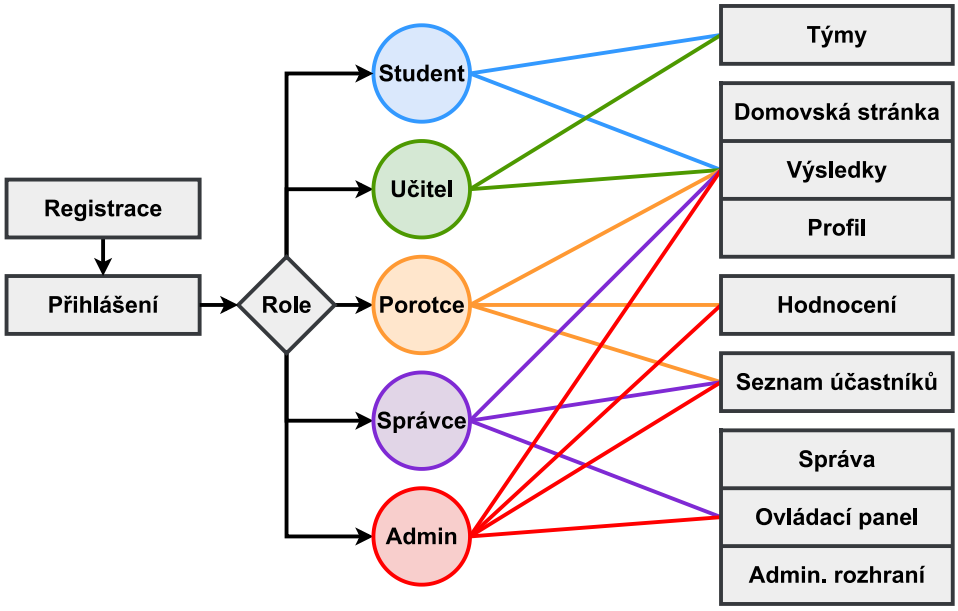
Stránky aplikace jsou definovány pomocí cest, které odpovídají různým URL adresám. Tyto cesty jsou zpracovávány funkcemi, které generují odpovědi na základě požadavků. Pro většinu stránek jsou vytvořeny šablony HTML stránek s Jinja2 syntaxí. Šablony umožňují vložení dat do HTML kódu pro zobrazení. Pro správné zobrazení stránek jsou ve speciálním adresáři umístěny statické soubory (CSS styly, obrázky).

Zabezpečení stránek je implementováno pomocí dekorátoru `@login_required`, který dá přístup na stránky jen přihlášeným uživatelům. V rámci spouštěných funkcí se pak ještě kontrolují role uživatele na určitých stránkách a v případě neoprávněného vstupu na stránku, uživatele přesměruje na jinou stránku (např. domovskou).

4.3.1 Dle role

Pro každou roli byl sepsán manuál do PDF souboru. Na navigační liště je odkaz na stáhnutí tohoto manuálu. Na backendu se zkontroluje, jakou roli má přihlášený uživatel a na základě toho se uživateli stáhne daný manuál. V manuálu je vysvětlena každá funkcionalita, kterou může daný uživatel provést. Každá role má přístup na určité stránky (obr. 8):

- student – domovská stránka, tým, výsledky a profil,
- učitel – domovská stránka, týmy, výsledky a profil,
- porotce – domovská stránka, seznam účastníků, hodnocení, výsledky a profil,
- správce – domovská stránka, ovládací panel, administrátorské rozhraní, seznam účastníků, správa, výsledky a profil,
- administrátor - domovská stránka, ovládací panel, administrátorské rozhraní, seznam účastníků, správa, hodnocení, výsledky a profil.



Obr. 8: Přístup na stránky dle role uživatele

4.3.2 Základní šablona

Tato šablona obsahuje základní strukturu pro všechny stránky webové aplikace. Obsahuje:

- nastavení HTML dokumentu – metadata (kódování, zobrazení dle rozlišení zařízení),
- odkazy na externí styly a skripty použité pro vykreslování nebo funkce stránek (Bootstrap, Font Awesome, jQuery),
- odkaz na vlastní globální CSS styly ze souboru `styles.css`,
- nastavení titulku stránky, který se bude specifikovat v potomcích této šablony,
- hlavičku stránky, která se vykresluje vždy s firemním logem a logem soutěže,

- navigační lišta, která se mění například dle role přihlášeného uživatele. Poskytuje snadný přístup k různým částem aplikace,
- nastavení pro vyskakovací upozornění, která jsou zobrazeny pod navigační lištou. Mají za úkol informovat uživatele o různých událostech. Zelené upozornění značí například úspěšné přihlášení a červené upozornění pak například chybu při špatně zadaném heslu,
- samotný obsah stránky definovaný v potomcích této šablony.

Navigační lišta

Tento flexibilní a dynamický prvek je v základní šabloně zobrazován hned pod hlavičkou stránky s nápisem „Mezinárodní studentská soutěž Stříbrný píst SMC“. Hlavička se skryje navigační lištou při posunu stránky směrem dolů.

Pro zařízení, kde by se odkazy nemohly vypisovat vedle sebe v liště, se zobrazí lišta s rozbalovacím tlačítkem, které po stisknutí vyvolá seznam se všemi odkazy pod sebou.

Navigační lišta (obr. 9) obsahuje odkazy na různé stránky aplikace, jako je domovská stránka, stránka týmu, seznam účastníků atd. Odkazy jsou reprezentovány ikonami z knihovny Font Awesome a popiskem.

Obr. 9: Navigační lišta pro porotce

Vzhled a odkazy v liště se mění v závislosti na přihlášeném uživateli a jeho roli v aplikaci. Například, pokud je uživatel přihlášen a má roli učitele, zobrazí se mu odkaz na stránku týmů, zatímco uživatel s rolí administrátora nebo porotce má k dispozici odkaz na stránku hodnocení.

Pro některé odkazy je použito rozbalovací menu, které umožňuje zobrazení dalších možností. Například pro uživatele s rolí admina nebo správce je k dispozici menu „Správa“ s možnostmi pro správu týmů, termínů a škol. To stejné platí pro výsledky, které jdou zobrazit pro každý termín nebo celkové výsledky. Tyto možnosti se zobrazují dle dalších podmínek, zda je přihlášený uživatel student nebo učitel a je zapnuté toto zobrazení výsledků pro daný termín.

V pravém straně lišty jsou dva odkazy na harmonogram soutěže a na manuál pro danou roli uživatele. Na kraji se pak zobrazuje informace o přihlášeném uživateli (včetně emailové adresy). Pokud uživatel přihlášen není, zobrazí se v navigačním liště zpráva o tom, že uživatel není přihlášen.

4.3.3 Styly

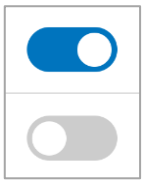
V souboru `styles.css` jsou definovány vlastní styly pro všechny stránky. Jako první jsou definovány specifické barvy používané firmou. Tyto barvy se pak dodržují pro všechny prvky na stránkách. Dle požadavků se používá primárně font Helvetica a veškerý text má specifickou černou barvu (80% Black) a pozadí bude primárně bílé. V souboru jsou pak definovány určité styly:

- nadpisů, textů, tabulek,
- ikon v liště,
- velikost hlavičky a prvky v ní,
- specifické barvy SMC pro tlačítka,
- velikost zaškrtávacích políček,
- specifické styly polí a tabulek pro každou stránku atd.

Základem stránek je vždy podklad s minimální výškou a zarovnáním na střed. Obsah stránek je pak v ohraničeném obdélníkovém těle s různými rozměry. Šířka je dána pevně pro každou stránku zvlášť a výška se přizpůsobuje obsahu. Jsou zde i styly pro dynamickou změnu vlastností při určité akci:

- při přesunu kurzoru na tlačítko se tlačítko zvýrazní (změna velikosti, barva),
- panely týmů se interaktivně zvýrazňují dle zobrazení týmu (animace),
- řádky v tabulce se zvýrazní při přejetí kurzorem (změna velikosti, barva),
- nadpisy pro výběr prvků na stránce – např. výsledky (změna velikosti, podtržení),
- pole na ovládacím panelu (změna velikosti, barvy, animace),

Pro ovládání funkcí v ovládacím panelu pro správce nebo administrátora byl vytvořen ovládací prvek (obr. 10). Toto ovládání reprezentuje zaškrtávací políčko, které nelze vidět. Místo tohoto políčka je zobrazen přepínač, který mění svůj vzhled na základě toho, zda je aktivovaný (zaškrtnutý) nebo ne. Při aktivaci se provede animace, díky které je znázorněn stav přepínače.



Obr. 10: Znázorněné přepínače při zapnutém a vypnutém stavu

4.3.4 Registrace

Stránka registrace je přístupná kdykoliv a uživatel se dostane přes odkaz při vstupu do aplikace, pokud není již přihlášen. Jazyk stránky (čeština/slovenština) je dán výchozím nastavením prohlížeče.

Backend

Backend přijímá GET i POST požadavky ze serveru. Při GET požadavku (mimo odeslání formuláře) se provede:

- kontrola, zda je registrace studentů nebo učitelů zapnutá,
- pokud je zapnutá registrace, zapíše se role jako možnost k registraci,
- uloží se všechny školy, které jsou zapsány v databázi,
- vykreslí se šablona `signup.html` a na frontend se posílají vybraná data (role k registraci, školy k registraci, slovník překladu stránky).

Pokud uživatel potvrdí registrační formulář v aplikaci (POST požadavek) nastane:

- načtení všech hodnot z odeslaného formuláře (jméno, příjmení, role, email, telefon, škola, heslo a potvrzení hesla),
- kontroluje se, zda již není zadaný email registrován. Pokud ano, je zobrazena chybová hláška a uživatel je vrácen zpět na začátek registrace,
- pokud se registruje učitel (`role_id = 2`), nastavíme mu maximální počet týmů (výchozí = 1),
- kontroluje se, zda není řetězec telefon prázdný (nepovinný údaj), případně se nastaví na `None`,
- pak probíhá kontrola všech zadaných údajů skrze jejich délku (musí mít určitý počet znaků),
- poslední kontrolní podmínkou je, aby byla obě zadaná hesla shodná,
- když jsou splněny všechny podmínky vytvoří se nový uživatel. Všechny zadané údaje se přiřadí k uživateli, ze zadaného hesla se vygeneruje hash hodnota a uloží se,
- nový uživatel se přidá do databáze a následně se změny zapíší. Zobrazí se hláška o úspěšné registraci a uživatel je přesměrován na přihlášení.

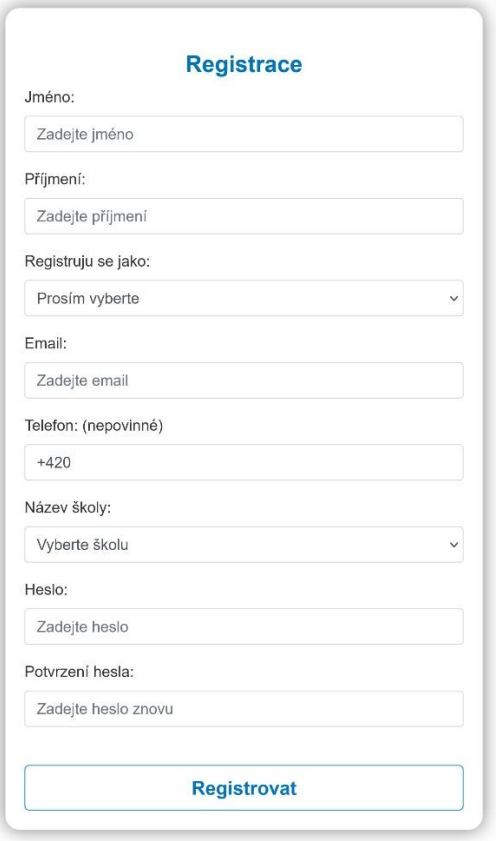
```
if len(jmeno) < 3:
    flash(text["mess_zadane_jmeno"], category='error')
elif len(prijmeni) < 3:
    flash(text["mess_zadane_prijmeni"], category='error')
elif len(email) < 5:
    flash(text["mess_zadany_email"], category='error')
elif len(heslo1) < 5:
    flash(text["mess_zadane_heslo"], category='error')
elif heslo1 != heslo2: # shoda hesel
    flash(text["mess_neshodne_hesla"], category='error')
else: # vše v pořádku, úspěšná registrace
    new_user = User(jmeno=jmeno,
                    prijmeni=prijmeni,
                    role_id=role_id,
                    email=email,
                    telefon=telefon,
                    skola_id=skola_id,
                    heslo=generate_password_hash(heslo1),
                    max_tymu = max_tymu,
                    overeny=False)
    db.session.add(new_user)
    db.session.commit()
    flash(text["mess_uzivatel_registrovan"], category='success')
    return redirect(url_for('auth.login'))
```

Obr. 11: Kontrola údajů a přidání uživatele

Frontend

Nejprve se kontroluje, zda jsou nějaké role povoleny k registraci. Pokud nejsou žádné, tak se uživatel nemůže registrovat (registrace je uzavřená). Když je alespoň jedna role k registraci otevřena, zobrazí se registrační formulář, viz obr. 12.

Nabídka rolí se vybírá z přijatých dat z backendu (student, učitel). Taktéž nabídka škol je ze seznamu z databáze. Po stisknutí tlačítka „Registrovat“ se formulář odešle a POST požadavek se zpracuje na pozadí serveru.



Obr. 12: Registrační formulář

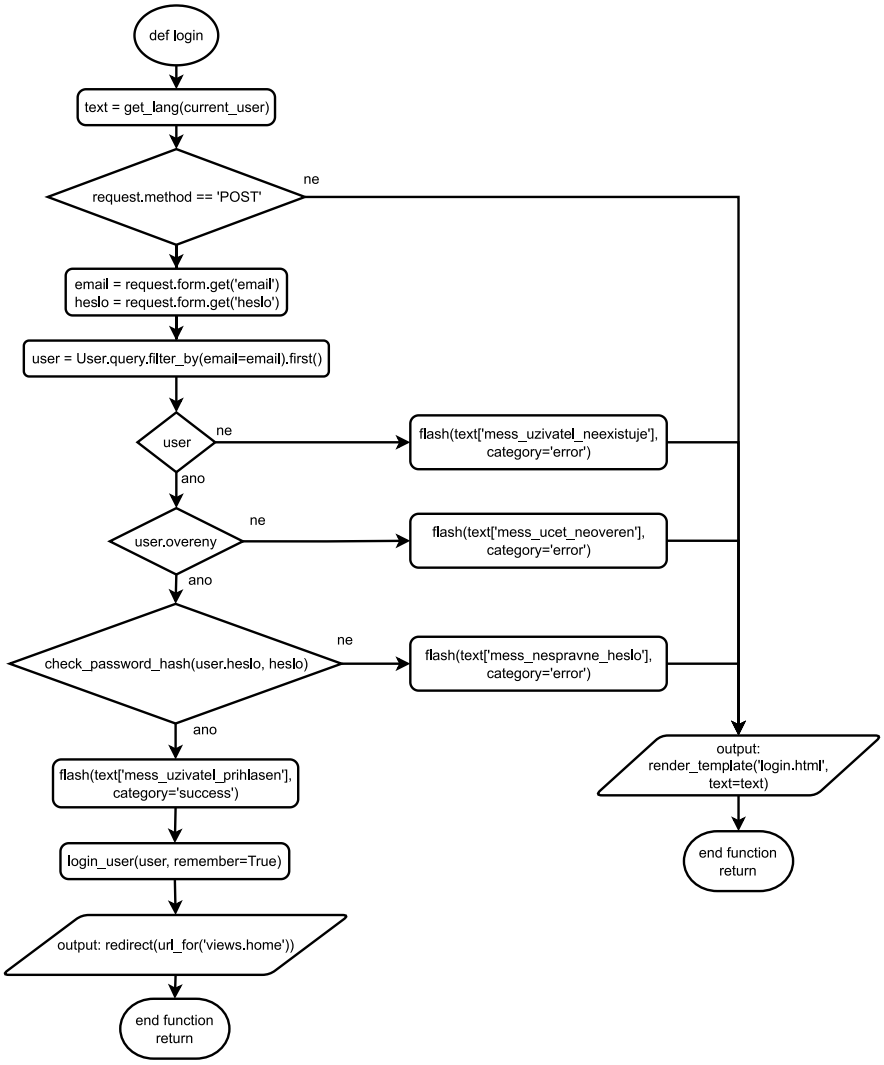
4.3.5 Přihlášení

Na stránku přihlášení je uživatel odkázán pokaždé, když není nikdo přihlášen, případně po úspěšné registraci. Překlad stránky je opět nastaven dle preferencí prohlížeče.

Backend

Backend (obr. 13) zpracovává GET i POST požadavky. Nejprve se nastaví jazyk stránky, vykreslí se šablona `login.html` a posílá se slovník vybraného jazyka.

Při POST požadavku se načtou data z přihlašovacího formuláře. Následně se dle emailu vyhledá přihlašovaná osoba z databáze. Pokud jej nelze najít, stránka se obnoví s upozorněním na neexistující účet. Dále se kontroluje ověření a heslo uživatele (pomocí kontroly `check_password_hash`). Při úspěšném přihlášení probíhá přesměrování na domovskou stránku.



Obr. 13: Vývojový diagram přihlašování

Frontend

V přihlašovacím formuláři se nachází dvě pole pro zadání emailu a hesla. V dolní části je pak tlačítko Přihlásit pro odeslání formuláře (přihlášení).

4.3.6 Odhlášení

Každý přihlášený uživatel má vpravo na liště možnost odhlášení svého účtu. Po stisknutí tohoto tlačítka proběhne na pozadí serveru skript pro odhlášení uživatele a pak je uživatel přesměrován na přihlašovací stránku.

4.3.7 Domovská stránka

Po přihlášení uživatele se zobrazí vždy tato stránka. Obsahuje základní informace k soutěži a výběry z minulých ročníků (výhry, žebříčky, fotky). Obsah spravuje administrátor dle potřeby.

4.3.8 Profil

Stránka je přístupná pro všechny přihlášené uživatele a slouží pro přehled o svém účtu. V rámci svého účtu si může uživatel měnit údaje (heslo, případně i kontaktní údaje při zapnuté funkci `zmena_udaju`. Pod svými údaji je pak ještě tabulka, kde jsou uvedeni všichni porotci s jejich kontaktními údaji a úlohami, které hodnotí. Překlad stránky se řídí dle státu školy uživatele, respektive nastavení prohlížeče.

Backend

Backend zpracovává GET i POST požadavky. Při GET požadavku se na pozadí provádí:

- načtení stavu funkce pro změnu údajů uživatelů,
- načtení všech porotců z databáze,
- pro každého porotce jsou vyhledány záznamy o úlohách, které má na starosti. Tyto úlohy jsou vypsány za sebou v řetězci,
- informace o porotcích jsou spojeny do jedné proměnné (array),
- vykreslí se šablona `profile.html` a na frontend se posílají vybraná data (informace o uživateli, informace o porotcích, slovník překladu stránky).

Pokud server přijme POST požadavek načítá se z formuláře `zmena`, která definuje, která akce byla provedena. Při změně profilu:

- se získají všechna data z formuláře (jméno, příjmení, email a telefon) a kontrolují se stejně jako při registraci (kapitola 4.3.4),
- pokud dojde ke změně údajů, vypíše se při úspěšné změně, které údaje byly změněny.

Při změně hesla:

- se získají z polí současné heslo, nové heslo a potvrzení hesla,
- proběhne kontrola, zda současné heslo odpovídá heslu zadanému,
- nové heslo se pak kontroluje stejně jako v registraci (kapitola 4.3.4),
- pokud nové heslo splňuje podmínky, zabezpečí se pomocí `generate_password_hash` a uloží se do databáze.

Frontend

Na stránce (obr. 14) se nejprve zobrazují v šedém obdélníku informace o uživateli (jméno, příjmení, email a telefon). Pod čarou pak vypisuje školu a roli. Pro učitele se zobrazuje navíc informace, zda má škola půjčeny vzorky. Ve spodní části obdélníku jsou pak tlačítka pro změnu údajů a hesla. Změna údajů může být vypnutá od správců/administrátora, a v tom případě se tlačítko pro tuto funkci nezobrazí.

Pro změnu údajů se vyvolá modální okno, kde se v polích zobrazí stávající informace a uživatel je může změnit. Tlačítkem „Uložit změny“ se data posílají na backend.

Pro změnu hesla se taky otevírá modální okno. Tentokrát s 3 prázdnými poli, kde uživatel musí zadat současné heslo k účtu a pak dvakrát nové heslo. Tlačítkem „Změnit heslo“ se obsah formuláře odešle.

Pod šedým obdélníkem se pak zobrazuje seznam porotců pro soutěžní úlohy. Tabulka obsahuje všechny porotce a administrátora, který může hodnotit všechny úlohy. U každého záznamu je uvedeno jméno, příjmení, kontaktní email a pak výpis úloh, které může daný porotce hodnotit.

Obr. 14: Profil administrátora

4.3.9 Týmy

Na stránku týmy mají přístup pouze studenti a učitelé. Pokud se někdo mimo tyto role pokusí o zobrazení této stránky, je uživatel vrácen na domovskou stránku a upozorněn, že zde nemá přístup. Překlad se řídí dle státu školy uživatele případně nastavením prohlížeče.

Backend

Server přijímá opět GET i POST požadavky. Při zobrazení stránky (GET požadavku) se dle role nastavují proměnné pro frontend. Pro roli učitele se z databáze načítají studenti, kteří mají stejnou školu jako učitel a zároveň nejsou ještě v žádném týmu. Seznam se seřazuje dle příjmení studentů. Pak se načtou všechny týmy daného učitele a případná volná čísla týmů. Z databáze se pak ještě pro učitele na frontend odesílají stavy funkcí (registrace a mazání týmů) z key-value tabulky Funkce. Pro studenty se všechny proměnné posílají prázdné (nejsou potřebné).

S rolí učitele je pak možnost odeslat data na backend (POST požadavek). Ve formuláři se odesílá parametr změna, který identifikuje, zda jde o registraci týmu či vyplnění předběžných informací školy. Při změně údajů školy jsou načteny 3 pole, které uživatel vyplnil (předběžný počet týmů, termín a zda chtějí poslat vzorky). Následně se tyto hodnoty zapíšou ke škole učitele do databáze.

Při registraci nového týmu pak data obsahují číslo nového týmu pro učitele, seznam studentů do týmu, účast v hlavní a vedlejší soutěži a vybraný termín, kterého se bude tým účastnit. Program dále pokračuje:

- načtení a kontrola dat, zda nejsou prázdné,
- potom se hledá tým dle čísla, jestli už neexistuje,
- v opačném případě se nový tým vytvoří a zapíše do databáze,
- dále je potřeba zapsat každého studenta do vytvořeného týmu do tabulky vazeb tým-student (Tým-Student). Každý záznam je zapsán do databáze,
- učitel je informován o úspěšném vytvoření týmu.

Učitel má možnost na stránce týmy při výběru týmů daný tým smazat, což zachycuje funkce pro smazání týmu, kdy je definována cesta `/delete_team`, která akceptuje pouze POST požadavky. Nejdříve se získává ID týmu (`tym_id`) z formuláře a poté se použije k získání konkrétního týmu z databáze pomocí funkce `Tym.query.get(tym_id)`.

Pokud tým s daným ID existuje, je odstraněn z databáze pomocí `db.session.delete(tym)`. Následně je uživatel přesměrován opět na stránku týmů a je informován o úspěšném smazání týmu.

Frontend

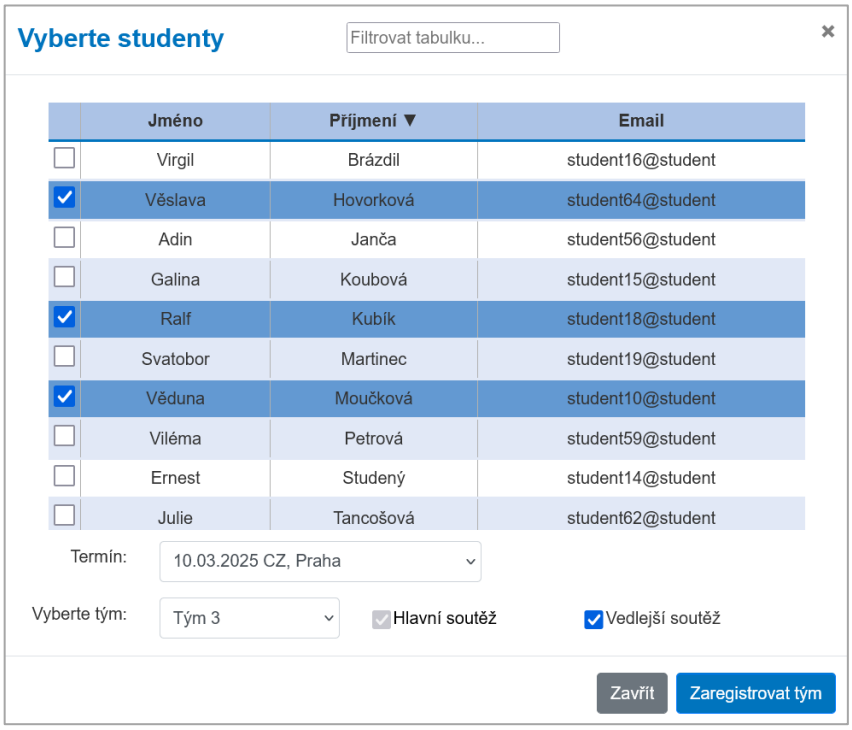
Obsahově je stránka pro každou roli jiná. Pro učitele jsou nad hlavním tělem stránky tlačítka pro vyvolání modálního okna. Jedno tlačítko je pro zadání předběžných informací pro danou školu. Učitel zde zadává, kolik bude mít škola předběžně soutěžních týmů, jaký termín má v plánu navštívit a zda bude škola chtít vzorky pro doplňkovou soutěž. Toto tlačítko se zobrazuje, dokud není zapnutá registrace týmů.

Druhé tlačítko je pro registraci týmů. Toto tlačítko se zobrazí pouze pokud je funkce povolena správci/administrátorem. Učitel zde registruje nové týmy pod jeho vedením. Otevře se modální okno (obr. 15), kde je jako první seznam všech studentů se stejnou školou jako má přihlášený učitel. Je zde možnost označení studentů pomocí políčka vlevo v řádku nebo pak pomocí skriptu JavaScript. Ten zajišťuje, že při kliku kdekoliv na řádek se tento řádek označí v políčku a zároveň se zvýrazní jako vybraný. Učitel zde musí vybrat 2 nebo 3 studenty do jednoho týmu. Tabulku si může seřadit klikem na libovolnou hlavičku sloupce pomocí dalšího skriptu pro řazení tabulky. Další funkcionalitou je zde filtrování tabulky, kdy po zadání textu do pole a potvrzení klávesou Enter se tabulka filtruje.

Po vybrání studentů se pak pro tým vybírá termín. Další položkou je číslo týmu pro učitele, kde se zobrazují jen zbývající volné čísla. Výchozím nastavením je jeden tým pro učitele, takže si bude učitel bude moct registrovat pouze tým s číslem 1. Pokud má povoleno větší množství týmů, může vybírat z takových, které ještě nejsou registrované.

Vedle rolovací nabídky jsou pak dvě zaškrtávací políčka pro registraci do hlavní/vedlejší soutěže. Políčko pro hlavní soutěž je deaktivované, protože každý registrovaný tým se musí účastnit hlavní soutěže a vedlejší soutěž je pak volitelná.

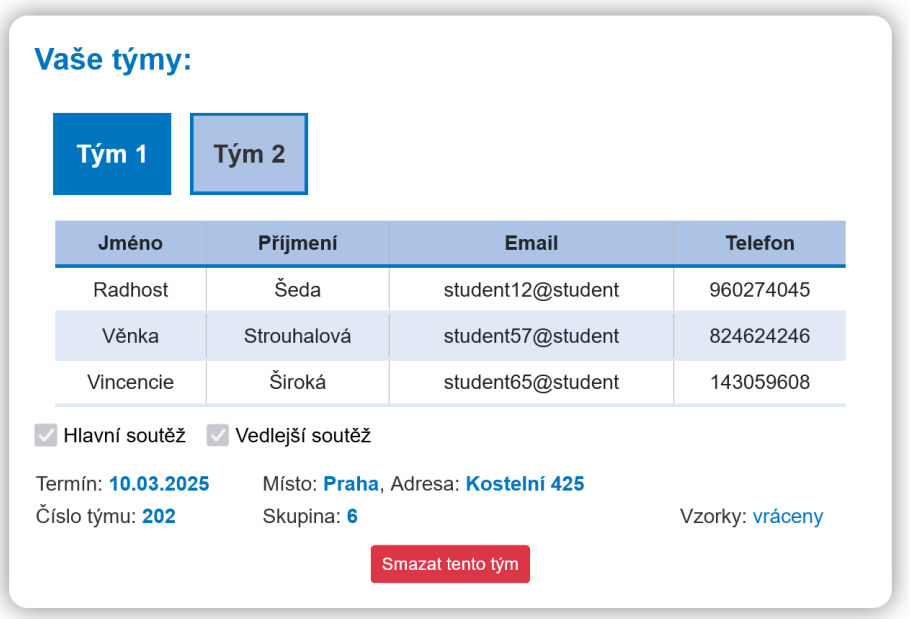
Potvrzení registrace probíhá pomocí JavaScript funkce `zapsat_tym`, která se spustí se zmáčknutím potvrzovacího tlačítka „Zaregistrovat tým“. Tato funkce je vytvořena, aby získala všechna zaškrtnutá políčka studentů a tím získala jejich ID. Dále se kontroluje, kolik je studentů vybraných, a pokud výběr podmínce nevyhovuje, je učitel upozorněn, aby vybral 2 až 3 studenty. Všechny pole jsou pak nakopírovány do pomocného formuláře, který se nakonec odešle.



Obr. 15: Registrace týmu

V hlavním těle stránky pro učitele je pak výpis všech týmů (obr. 16), které jsou registrovány. Každý tým má tlačítko, které aktivuje zobrazení informací o daném týmu. Zobrazí se tabulka registrovaných studentů a pod ní pak zaškrtávací políčka znázorňující účast v hlavní/vedlejší soutěži. Dále se zde zobrazuje termín (datum, místo, adresa), číslo týmu a skupina (od správců dle termínu) a pak informace, zda má škola vypůjčeny vzorky.

Pod těmito informacemi je zde ještě formulář s tlačítkem, které slouží ke smazání vybraného týmu. Toto tlačítko se zobrazuje pouze, pokud je tato funkce zapnuta. Tlačítko odešle formulář, který stránky přesměruje na `/delete_team` a zároveň spustí JavaScript funkci `potvrdit_smazani_tymu`. Funkce vyvolá potvrzovací dialog, zda chce učitel opravdu smazat daný tým. Po potvrzení se odešle formulář s POST požadavkem a ID vybraného týmu.



Obr. 16: Zobrazení týmů učitele

Stránka tým je potom pro studenta už pouze informativní. Vidí zde svůj tým, vedení týmu, své kolegy a poté stejné informace jako učitel (hlavní/vedlejší soutěž, termín, číslo týmu, skupina). Pokud student nemá žádný tým, je o tom informován.

4.3.10 Hodnocení

Stránka slouží pro hodnocení jednotlivých úloh, kterých se týmy účastnili. Přístup zde mají pouze porotci a administrátor. Opět je zavedena podmínka pro případný vstup neoprávněného uživatele. Překlad stránky se řídí primárně nastavením prohlížeče.

Backend

Pro požadavek GET je nejprve předán identifikátor termínu (`termin_id`) z URL adresy pomocí metody `request.args.get('termin_id')`. Pokud žádný takový termín dle identifikátoru neexistuje, je uživatel odkázán na domovskou stránku. Následně získáme všechny hlavní, vedlejší úlohy a také úlohy, které má přihlášený uživatel možnost hodnotit skrz spojovací tabulku porotců a úloh (Porotce_Uloha).

Další část kódu (obr. 17) provede shromáždění informací (ID, číslo týmu, název školy, stát školy, skupina týmu a účast v doplňkové soutěži) o všech týmech daného termínu pomocí spojení více tabulek, filtrů a seřazení dle čísla týmu. V tomto dotazu je využita funkce chaining, kdy se řetězí metody `join`, `outerjoin`, `filter`, `group_by` a `order_by`.

```
tymy_info = db.session.query(Tym.id,
                             Tym.cislo_tymu,
                             Skola.nazev,
                             Skola.stat,
                             Tym.skupina,
                             Tym.vedlejsi_soutez,
    ).join(User, Tym.ucitel_id == User.id)\
    .join(Termin, Tym.termin_id ==Termin.id)\
    .outerjoin(Skola, User.skola_id == Skola.id)\
    .filter(Termin.id == termin_id)\
    .group_by(Tym.id, Tym.cislo_tymu, Skola.nazev, Skola.stat,
              Tym.skupina, Tym.vedlejsi_soutez)\
    .order_by(Tym.cislo_tymu).all()
```

Obr. 17: Kód pro výpis informací o týmech

Nyní budou doplněny informace o týmech dalšími informacemi (obr. 18). Když někde data chybí, vyplní se pouze pomlčka. Nejprve se hledají hodnocení hlavních úloh pro každý tým. Pokud je hodnocení naleznuto, uloží se hodnota k ostatním hodnocením. Probíhá zde i kontrola, pokud přihlášený porotce danou úlohu hodnotí, a ještě nemá žádné hodnocení, označí se porotci tým jako nehodnocený.

To samé pak platí pro hodnocení doplňkové soutěže. Všechny načtené hodnocení se následně přidají k dosavadní tabulce s informacemi týmů. Pak již skript končí a vrací šablonu `rating.html` společně s daty pro výpis a úpravu hodnocení.

Dále stránka hodnocení přijímá POST požadavek společně s daty pro úpravu hodnocení. Načte se tým, který porotce hodnotí, pak se hodnoty hodnocení dle úloh rozdělí a pokud došlo ke změně hodnoty, tak se zapíšou do databáze. Když ještě žádné hodnocení neexistuje, vytvoří se nové. Při manipulaci s daným hodnocením se vždy do databáze ukládá i uživatel, který změnu provedl.

```
for t in tymy_info:
    tym_id, tym_cislo, skola, stat, skupina, soutez = t
    if tym_cislo == None: tym_cislo = "-"
    if skupina == None:
        skupina ='-'
    for uloha_id in ulohy_hlavni_ids:
        hod = Hodnoceni.query.filter_by(tym_id=tym_id,
                                        uloha_id=uloha_id).first()
        skore_hlavni.append(str(hod.skore) if hod else '-')
        if not hod and tym_cislo not in tymy_nehod_hlavni and uloha_id in
                       [uloha.id for uloha in ulohy_porotce_hlavni]:
            tymy_nehod_hlavni.append(tym_cislo)
    hodnoceni_hlavni.append((tym_id,tym_cislo, skola, stat,
                             skupina, *skore_hlavni))
    if soutez:
        for uloha_id in ulohy_vedlejsi_ids:
            hod = Hodnoceni.query.filter_by(tym_id=tym_id,
                                            uloha_id=uloha_id).first()
            skore_vedlejsi.append(str(hod.skore) if hod else '-')
            if not hod and tym_cislo not in tymy_nehod_vedlejsi and uloha_id
                         in [uloha.id for uloha in ulohy_porotce_vedlejsi]:
                tymy_nehod_vedlejsi.append(tym_cislo)
        hodnoceni_vedlejsi.append((tym_id, tym_cislo, skola, stat,
                                   skupina, *skore_vedlejsi))
```

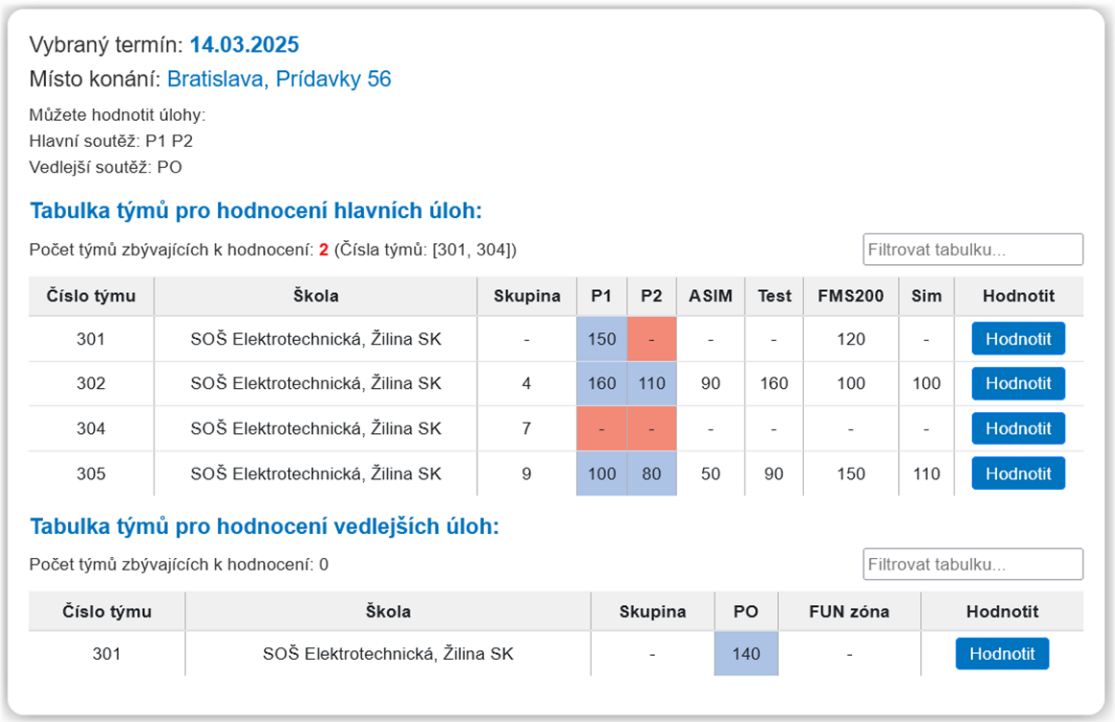
Obr. 18: Kód pro výpis týmů s hodnocením ve všech úlohách

Frontend

Pro vstup na stránku hodnocení si uživatel vybírá, který termín bude hodnotit. Následně se načte stránka se všemi týmy pro vybraný termín a jejich hodnocením (obr. 19). na stránce se zobrazí informace o termínu a výpis úloh, které porotce hodnotí a následně dvě tabulky s hlavní a vedlejší soutěží.

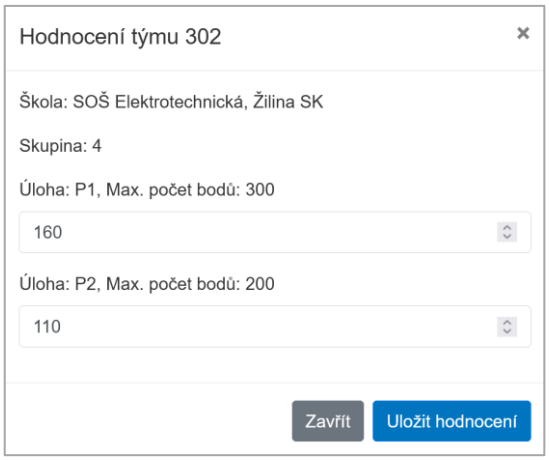
V první tabulce jsou všechny týmy daného termínu seřazené dle čísla týmu. Tabulka obsahuje název školy, skupinu týmu a pak všechny hodnoty získaného skóre ve všech hlavních úlohách. V posledním sloupci se nachází tlačítko pro hodnocení daného týmu.

Ve druhé tabulce jsou uvedeny jen týmy, které byly registrovány do doplňkové soutěže. Opět obsahuje stejné informace a každý řádek označuje hodnocení daného týmu tentokrát v doplňkové soutěži. Obě tabulky je možno filtrovat pomocí vstupního pole nad tabulkou. Tabulka se filtruje po zadání řetězce a stisknutí klávesy Enter.



Obr. 19: Hodnocení týmů porotcem

Pro zadání (úpravu) hodnocení je vytvořeno tlačítko „Hodnotit“, které vyvolá modální okno (obr. 20) pro zvolený tým. V okně se vypíšou stejné informace o týmu jako v tabulce. Ve spodní části pak jsou umístěny vstupní pole pro hodnocení daných úloh. Pole pro úpravu se zobrazí jen pro úlohy, které má daný porotce možnost hodnotit. Pole jsou omezena na číselné hodnoty a jsou limitovány maximálním počtem bodů úlohy.



Obr. 20: Modální okno pro hodnocení úloh

V tabulkách je červeným podbarvením buněk zvýrazněno hodnocení, které dosud nebylo vytvořeno a zároveň má být přihlášeným uživatelem hodnoceno (případně má právo hodnotu měnit). Týmy, kterým takto chybí hodnocení od porotce, jsou vyznačeny nad tabulkou. Tato informace je vyznačena pro porotce, aby se snížila šance na chybějící hodnocení.

4.3.11 Výsledky

Tato stránka slouží k zobrazení výsledkové listiny z obou soutěží a všech termínů. Přístup na tyto stránky mají všichni. Zobrazení se týká všech termínů a pak celkových výsledků. Pro studenty a učitele je přístup omezen funkcí zobrazení výsledků. Když je funkce vypnutá, uživatel (student nebo učitel) se na výsledky nedostane a bude přesměrován na domovskou stránku. Pokud je zapnutá, hlídá se možnost zobrazení pouze termínu, kterého se daný student (učitel) účastní.

Backend

Nejprve je předán identifikátor termínu (`termin_id`) z URL adresy. Pokud zadaný termín neexistuje, je uživatel odkázán na domovskou stránku. Poté se kontrolují role uživatele a zobrazení výsledků, případně probíhá přesměrování na domovskou stránku. Aby se přihlášený uživatel mohl na stránku výsledků termínu dostat, musí se jeho tým účastnit daného termínu. Případně pokud má více týmů (učitel), tak zkontrolovat všechny termíny týmů.

```
if current_user.role_id == 1:
    if not current_user.tym \
        or current_user.tym[0].termin_id != termin_show.id:
        # pokud není zadaný termín u jeho týmu, vyhodit
        flash(text["mess_vysledky_nepovoleny"], category='error')
        return redirect(url_for('views.home'))
elif current_user.role_id == 2:
    if current_user.vedeni:
        allow = False
        for tym in current_user.vedeni:
            if tym.termin_id == termin_show.id:
                allow = True
        if allow == False:
            flash(text["mess_vysledky_nepovoleny"], category='error')
            return redirect(url_for('views.home'))
    else: # pokud nevede tým, nemá přístup na jakékoliv výsledky
        flash(text["mess_vysledky_nepovoleny"], category='error')
        return redirect(url_for('views.home'))
```

Obr. 21: Kód pro kontrolu přístupu na výsledky termínu

Dalším krokem je nastavení překladu stránky dle státu termínu díky pomocné funkci `get_lang(current_user, termin_show)`. Program pak pokračuje:

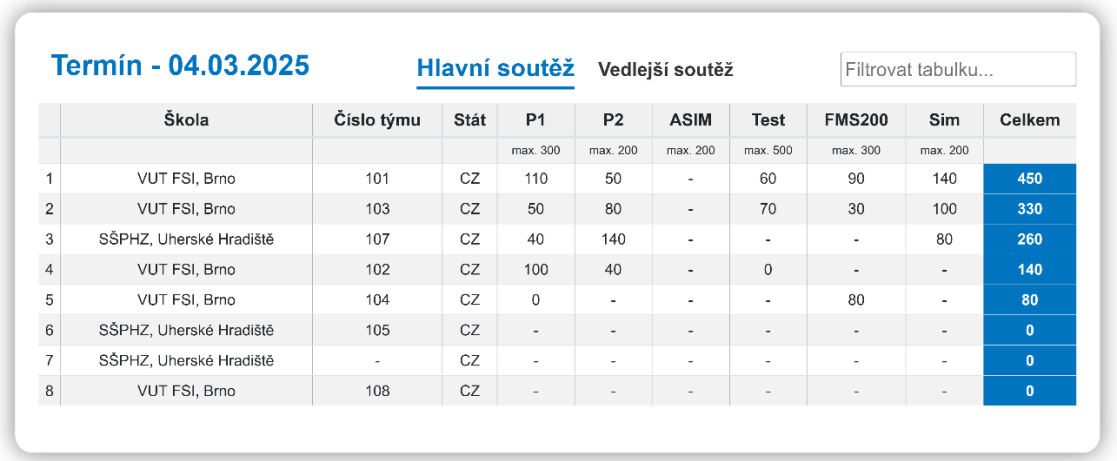
- načtením informací o všech týmech z termínu (stejný jako u hodnocení viz obr. 17, kromě seřazení dle čísla týmu),
- pokud není získán žádný termín z URL (pouze `/vysledky`), pak se budou zobrazovat celkové výsledky. Načtou se informace všech týmů,
- pro každý tým je pak hledáno každé hodnocení jako u stránky hodnocení (kapitola 4.3.10). Při každé úloze se skóre hodnocení přitom sčítá do celkového skóre týmu,

- vzniká tak další atribut týmu celkového počtu bodů zvlášť v hlavní a vedlejší soutěži. Tento atribut se zařadí do výsledné tabulky ještě před samotné hodnocení všech úloh z důvodu jednodušší iterace hodnocení úloh na frontendu,
- výsledková tabulka se nakonec ještě seřadí dle celkového počtu bodů týmů. První budou týmy s nejvyšším počtem bodů. Přidává se navíc index pořadí týmu,
- nakonec je vyvolána šablona `results.html` a výsledková tabulka pro obě soutěže je posílána na frontend.

Frontend

Výsledková stránka je koncipována tak, aby pojmula co nejvíce týmů. Bylo zmenšeno písmo v tabulkách a nad tabulkami jsou pak ovládací prvky pro přepínání výsledků hlavní nebo vedlejší soutěže.

Při načtení stránky se v prvním řádku zobrazí datum termínu, respektive označení celkových výsledků. Vedle se pak nachází 2 ovládací tlačítka pro výběr soutěže. Jako první se automaticky aktivuje tlačítko pro zobrazení výsledků hlavní soutěže. Aktivované tlačítko je definováno stylem CSS a je při výběru podtrženo a zvětšeno. Vpravo na řádku se pak nachází pole pro filtrování tabulky výsledků.



	Škola	Číslo týmu	Stát	P1	P2	ASIM	Test	FMS200	Sim	Celkem
				max. 300	max. 200	max. 200	max. 500	max. 300	max. 200	
1	VUT FSI, Brno	101	CZ	110	50	-	60	90	140	450
2	VUT FSI, Brno	103	CZ	50	80	-	70	30	100	330
3	SŠPHZ, Uherské Hradiště	107	CZ	40	140	-	-	-	80	260
4	VUT FSI, Brno	102	CZ	100	40	-	0	-	-	140
5	VUT FSI, Brno	104	CZ	0	-	-	-	80	-	80
6	SŠPHZ, Uherské Hradiště	105	CZ	-	-	-	-	-	-	0
7	SŠPHZ, Uherské Hradiště	-	CZ	-	-	-	-	-	-	0
8	VUT FSI, Brno	108	CZ	-	-	-	-	-	-	0

Obr. 22: Zobrazení výsledků pro určitý termín

Hlavním prvkem je na stránce samotná tabulka výsledků (obr. 22). Pro hlavní soutěž jsou zobrazena data o soutěžních týmech (pořadí, škola, číslo týmu, stát, úlohy hlavní soutěže a pak celkový počet bodů). Součástí hlavičky tabulky je druhý řádek popisující maximální počet bodů v každé úloze. Týmy jsou seřazeny sestupně dle celkového získaného počtu bodů v hlavní soutěži.

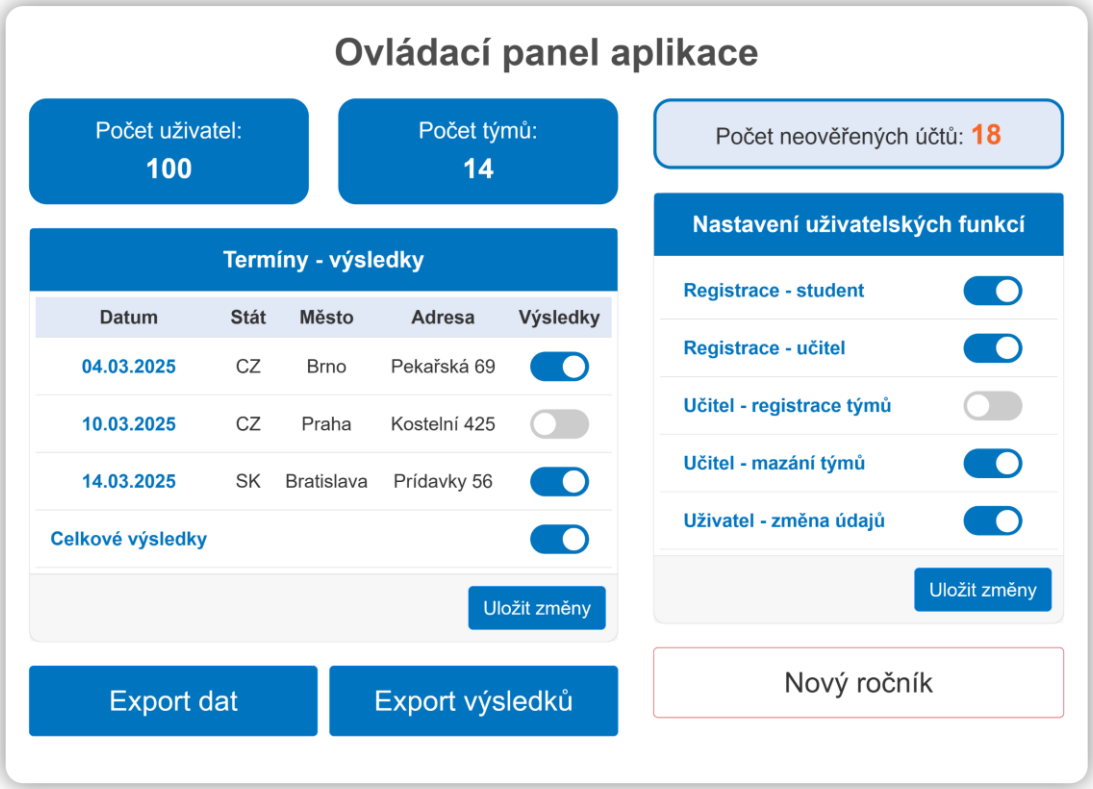
Při výběru vedlejší soutěže se zobrazuje podobná tabulka (pořadí, číslo týmu, stát, úlohy a celkový počet bodů) jen s týmy, které vedlejší soutěž absolvovaly. Tabulka je opět stejně seřazená sestupně dle počtu bodů.

Pro přepínání mezi tabulkami slouží JavaScript, který dle aktivovaného tlačítka zobrazí či schová danou tabulku.

4.3.12 Ovládací panel

Pro jednoduché ovládání základních funkcí byl vytvořen ovládací panel (obr. 23). Původním záměrem bylo, aby měl na tuto stránku přístup pouze administrátor. Později se dle požadavků rozšířila možnost použití i pro roli správce.

Na stránku se tedy dostane jen uživatel s daným oprávněním. Bez oprávnění je uživatel přesměrován na domovskou stránku. Překlad stránky se řídí dle nastavení prohlížeče.



Obr. 23: Ovládací panel aplikace

Backend

Zahájením skriptu se získá preference jazyka prohlížeče pro získání slovníku. Proběhne kontrola oprávnění dle role uživatele. Dále se získávají data pro zobrazení na webu:

- počet uživatelů (celkem, studenti, učitelé, porotci a správci),
- počet ověřených a neověřených účtů,
- načtení všech neověřených uživatelů a jejich dat,
- počet registrovaných týmů pro daný ročník (celkem, pro každý termín a bez termínu viz obr. 24),
- získání všech stavů funkcí pro aplikaci (registrace studentů, registrace učitelů, registrace týmů, mazání týmů a změna údajů).

```
for termin in Termin.query.all():
    pocet_tymu = Tym.query.filter_by(termin_id=termin.id).count()
    tymy_data.append({"name": f"Termín {termin.id}", "pocet": pocet_tymu})

tymy_data.append({"name": "Bez termínu",
                  "pocet": Tym.query.filter_by(termin_id=None).count()})
```

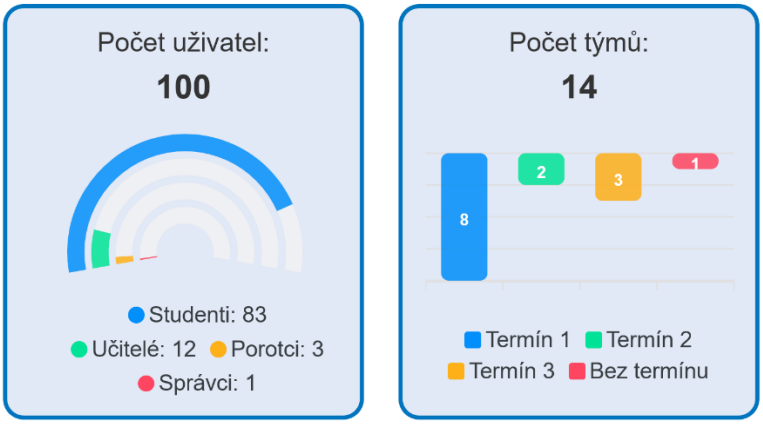
Obr. 24: Kód pro získání počtů týmu dle termínu

Skript pak vrací na web šablonu `dashboard.html` a všechna načtená data.

Frontend

Celá stránka je koncipována tak, aby byla přehledná a zároveň sloužila k základnímu ovládání webové aplikace. Tělo stránky je rozděleno na dva hlavní sloupce. V prvním sloupci se nachází dva obdélníkové panely, které obsahují číselné údaje o soutěži (počet uživatelů a počet týmů).

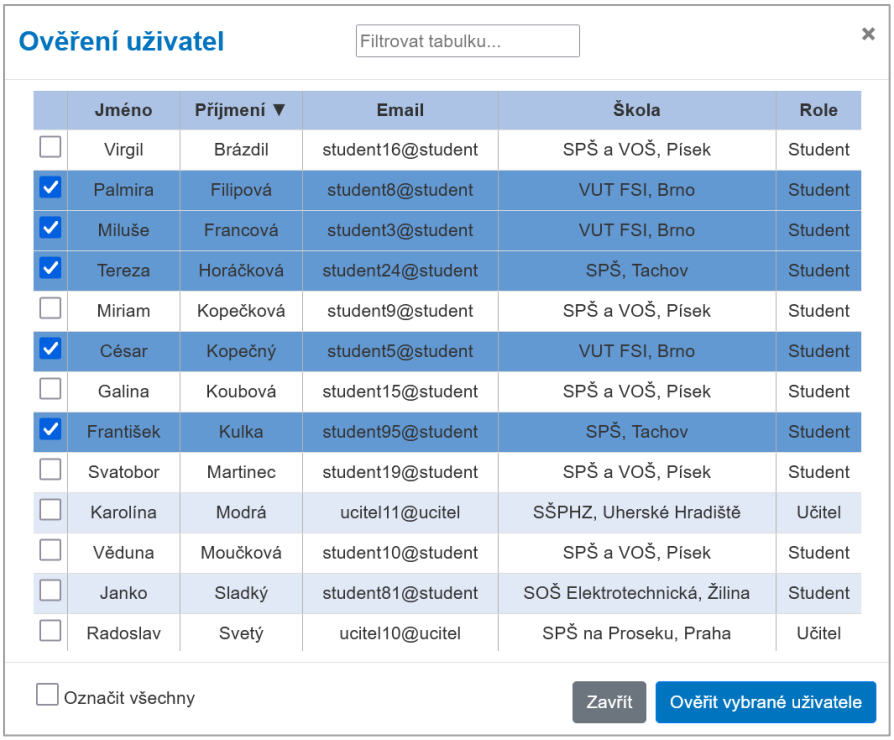
Oba panely jsou interaktivní a lze je pomocí kliknutí aktivovat (obr. 25). Panely se svisle roztáhnou a zároveň posunou obsah pod nimi. Při aktivaci prvního panelu se zobrazí graf, který doplňuje informace o počtech uživatelů dle rolí. Graf je sestrojen pomocí JavaScriptu z knihovny ApexCharts. Data pro graf jsou načtena pomocí proměnné `users` z backendu (počet studentů, učitelů, porotců a správců). Druhý panel po aktivaci zobrazí také graf knihovny ApexCharts, tentokrát klasický sloupcový. Graf znázorňuje počty týmů registrovaných k určitému termínu. Data (array) jsou předána z backendu a následně musela být pomocí JavaScript převedena do formátu JSON a pak teprve zpracována a načtena jako vstup pro graf.



Obr. 25: Panely pro zobrazení počtu uživatelů a týmů

Další panel je umístěn do pravého sloupce stránky a upozorňuje na neověřené účty. Po aktivaci tohoto panelu je panel opět roztažen směrem dolů a doplní se zde graf o procentuálním složení ověřených a neověřených účtů s dodatečným tlačítkem sloužícím pro ověření uživatelů.

Pro ověření uživatelů je otevřeno modální okno (obr. 26) se seznamem neověřených účtů v databázi. Každý řádek obsahuje políčko k zaškrtnutí, které slouží pro označení uživatelů, které chce přihlášený ověřit. Pro označování řádků byl vytvořen skript, který označí políčko i v případě kliknutí kdekoliv na řádek záznamu. Označený řádek se navíc vyznačí modrou barvou pro přehledný výběr uživatelů.



Obr. 26: Modální okno pro ověření uživatelů

Tabulku je možné filtrovat pomocí vstupního pole nad tabulkou. Pro jednoduché hledání v tabulce byl přidán skript pro řazení tabulky dle vybraných sloupců. Po kliknutí na hlavičku sloupce, skript seřadí tabulku dle daného sloupce dle abecedy. Po opětovném kliknutí se řadí záznamy naopak od konce abecedy.

Pro potřebu ověření všech uživatelů je uvedeno dole pod tabulkou políčko pro označení všech řádků. Skript opět označí nebo případně odznačí všechny políčka v tabulce.

Po výběru ověřovaných uživatelů se pak pomocí potvrzovacího tlačítka spustí skript pro vypsání všech ID vybraných uživatelů. Data se pak zapíší do formuláře a posílají se s POST požadavkem na dodatečnou stránku pro změnu informací.

V prvním sloupci pod panely je okno s termíny soutěže, kde má uživatel možnost vypnout či zapnout zobrazení výsledků pro daný termín. Každý termín má uvedeno datum, stát, město a adresu konání termínu. Na konci řádku je pak přepínač (obr. 10), který řídí zobrazování výsledků pro studenty a učitele. Poslední řádek obsahuje přepínač pro zobrazování celkových výsledků. Pro změnu zobrazení je nutno nastavit přepínače dle potřeby a následně uložit změny pomocí tlačítka, které se odkazuje opět na dodatečnou stránku pro příjem dat.

Pod panelem jsou pak umístěna dvě tlačítka pro export dat nebo výsledků do souboru Excel. Tato tlačítka odkazují na stránku backendu `/export` s různými argumenty pro rozlišení exportu dat nebo výsledků.

V pravé části stránky je pak umístěn ovládací panel pro všechny funkce potřebné v aplikaci (registrace studentů, registrace učitelů, registrace týmů, mazání týmů a změna údajů). Každá funkce je uvedena na řádku spolu s přepínačem, který ovládá jejich zapnutí či vypnutí. Pro změnu nastavení je potřeba všechny přepínače nastavit dle požadovaného stavu a pak změny potvrdit tlačítkem pod tabulkou. Stavy se posílají pomocí formuláře opět na dodatečnou stránku, která změny zpracuje.

Dalším tlačítkem pod panelem je pak pouze pro administrátora tlačítko pro smazání dat z aktuálního ročníku. Odkazuje na stránku backendu, kde probíhají akce pro export, zálohu a smazání dat z aktuálního ročníku soutěže a příprava pro nový ročník.

Příjem dat

Pro přijmutí dat slouží dodatečná stránka `/zmena_dashboard`. Stránka přijímá POST požadavky ze serveru společně s daty z formuláře. Na konci skriptu proběhne přesměrování na stránku s ovládacím panelem. Pro případ ověřování uživatelů:

- se načítá seznam ID vybraných uživatelů,
- každý uživatel nalezen dle ID,
- pomocí `user.overeny = True` je uživatel označen jako ověřený,
- každá změna je zapsána do databáze a uložena,
- proběhne přesměrování na stránku ovládacího panelu.

Pro případ změny zobrazení výsledků:

- načtení všech termínů,
- pro každý termín získán stav z formuláře,
- pokud nastala změna oproti záznamu v databázi, je změna uložena,
- načtení stavu pro zobrazení celkových výsledků,
- porovnání s aktuálním stavem,
- v případě změny uložení nového stavu do databáze,
- proběhne přesměrování na stránku ovládacího panelu.

Pro případ změny nastavení funkcí:

- načtení všech funkcí,
- porovnání aktuálního stavu s novým,
- změny jsou zapsány a uloženy,
- proběhne přesměrování na stránku ovládacího panelu.

4.3.13 Seznam účastníků

Stránka seznamů účastníků je přístupná všem kromě studentů a učitelů. Má informativní charakter a prezentuje informace o studentech, učitelích, týmech a školách. Překlad stránky se řídí nastavením prohlížeče.

Backend

První se kontroluje role uživatele a případně uživatele vrátí na domovskou stránku. Následně se začnou shromažďovat informace o všech registrovaných studentech:

- jméno a příjmení,
- email,
- telefon,
- stát školy,
- škola studenta,
- případné číslo týmu, pokud jej má.

Dále se načítají informace o všech registrovaných učitelích. Má stejné vlastnosti jako student, akorát se kontroluje, zda učitel vede více týmů a případně čísla jednotlivých týmů složí do jednoho řetězce. Stejně se načítají všechny týmy s informacemi:

- číslo týmu, pokud jej má,
- škola vedoucího učitele,
- registrace do vedlejší soutěže,
- datum termínu účasti, pokud jej má,
- číslo skupiny, pokud jej má.

Dále se do proměnné načtou informace o školách:

- název školy,
- stát, adresa školy,
- vzorky, případně jaké má škola půjčeny,
- předběžné informace o počtu týmů, termínu účasti a poslání vzorků.

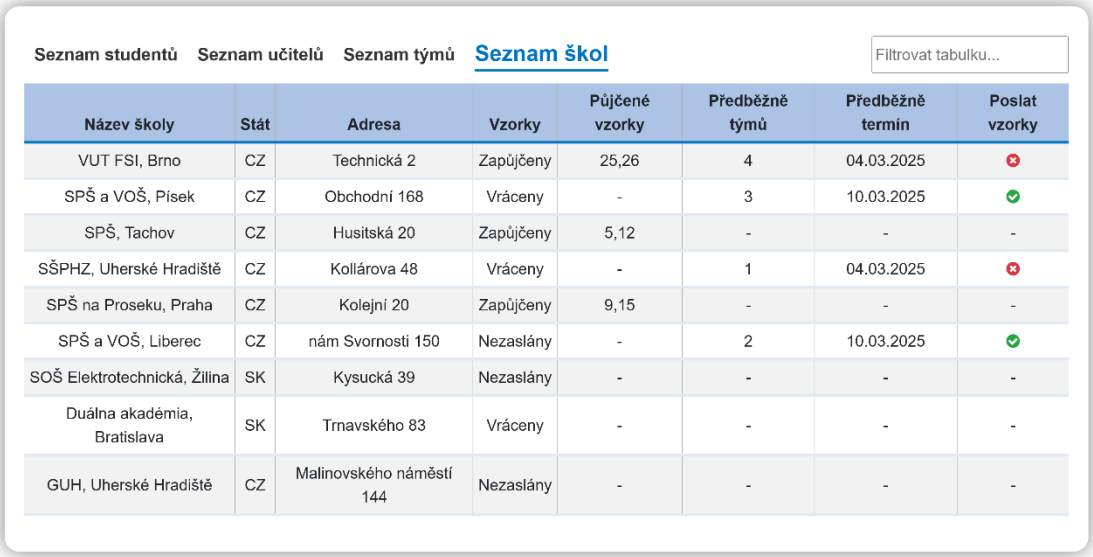
Nakonec se vykresluje šablona list.html a posílají se všechna zmíněná data.

Frontend

Na stránce se zobrazí v prvním řádku volby pro zobrazení seznamů. Uživatel si vybírá z:

- seznam studentů (zobrazen při vstupu na stránku),
- seznam učitelů,
- seznam týmů,
- seznam škol.

Vedle výběru vpravo na řádku je pak vstupní pole pro filtr tabulek. Při výběru seznamu se spustí JavaScript, který aktivuje zmáčknuté tlačítko, deaktivuje ostatní, zobrazí odpovídající tabulku a schová ostatní tabulky. To samé platí pro filtry tabulek. V tabulkách se vypisují všechny informace přijaté z backendu (obr. 27).



Název školy	Stát	Adresa	Vzorky	Půjčené vzorky	Předběžně týmů	Předběžně termín	Poslat vzorky
VUT FSI, Brno	CZ	Technická 2	Zapůjčeny	25,26	4	04.03.2025	✗
SPŠ a VOŠ, Písek	CZ	Obchodní 168	Vráceny	-	3	10.03.2025	✓
SPŠ, Tachov	CZ	Husitská 20	Zapůjčeny	5,12	-	-	-
SŠPHZ, Uherské Hradiště	CZ	Kollárova 48	Vráceny	-	1	04.03.2025	✗
SPŠ na Proseku, Praha	CZ	Kolejní 20	Zapůjčeny	9,15	-	-	-
SPŠ a VOŠ, Liberec	CZ	nám Svornosti 150	Nezaslány	-	2	10.03.2025	✓
SOŠ Elektrotechnická, Žilina	SK	Kysucká 39	Nezaslány	-	-	-	-
Duálna akadémia, Bratislava	SK	Trnavského 83	Vráceny	-	-	-	-
GUH, Uherské Hradiště	CZ	Malinovského náměstí 144	Nezaslány	-	-	-	-

Obr. 27: Zobrazení seznamu všech škol

4.3.14 Správa

Správa je rozdělena na více stránek. Pro každou stránku se rozlišují GET požadavky (získání údajů) a POST požadavky (změna údajů). Na stránky mají přístup pouze správci nebo administrátor. Uživatel se na správu dostane, když rozklikne v navigační liště správu a pak si vybere z nabídky. Překlad stránek se řídí dle nastavení prohlížeče.

Číslo týmu	Stát	Škola	Vedlejší soutěž	Termín	Skupina	Změna
-	CZ	SŠPHZ, Uherské Hradiště	✓	04.03.2025	-	Upravit
101	CZ	VUT FSI, Brno	✗	04.03.2025	-	Upravit
201	CZ	SPŠ a VOŠ, Písek	✗	10.03.2025	5	Upravit
301	SK	SOŠ Elektrotechnická, Žilina	✓	14.03.2025	-	Upravit

Obr. 28: Tabulka pro správu týmů

Správa týmů

Po odeslání všech týmů z backendu a zobrazení šablony `team_management.html` se na stránce (obr. 28) objevuje tabulka s registrovanými týmy z databáze s parametry (číslo týmu, škola, stát, registrace do doplňkové soutěže, datum termínu účasti, skupina). Každý záznam v tabulce lze upravit pomocí tlačítka „Upravit“ na konci řádku každého záznamu.

V tabulce je přidán styl pro vybarvení buňky, která nemá žádnou hodnotu a je potřeba ji doplnit správcem. Toto vybarvení se týká primárně čísla týmu, skupiny a popřípadě termínu.

Pokud chce uživatel změnit parametry týmu otevře se modální okno pro úpravu dat. V okně se vypíšou navíc informace o učiteli a studentech daného týmu. Dále jsou dostupná pole pro úpravu a následně i tlačítko pro uložení změn. Tlačítko má funkci odeslání formuláře s POST požadavkem, který pak zpracovává backend aplikace.

Dle ID se nejprve najde daný tým a pak se kontrolují zadané informace a případné změny se zapíší. Databáze se nakonec uloží. Pokud nastane nějaká chyba (IntegrityError), databáze se pomocí příkazu `db.session.rollback()` vrátí do původního stavu.

Správa termínů

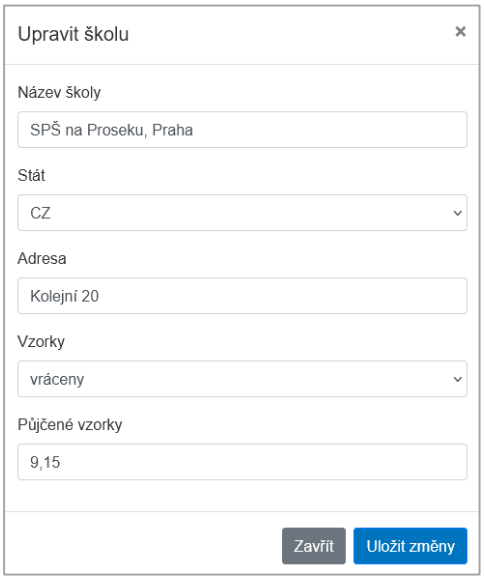
Při vstupu na stránku správu termínů se nemusí posílat data dodatečně. Termíny jsou dostupné z jakékoliv stránky pomocí skriptu v inicializačním skriptu. Stránka využívá šablonu `term_management.html`, která vykresluje obdobně tabulku se všemi termíny a jejich parametry (ID, datum termínu, stát, město, adresa konání a funkce pro povolení zobrazení výsledků pro daný termín).

Při úpravě dat termínu je otevřeno modální okno s poli pro úpravu. Pro uložení změn stejným způsobem pomocí tlačítka odešle formulář na backend aplikace. Najde se příslušný termín a mění se údaje. Při chybové hlášce se databáze vrátí do původního stavu, jinak se databáze na konci uloží a stránka je opět načtena.

Správa škol

Po vykreslení šablony `school_management.html` a odeslání všech škol do databáze je na stránce tabulka se školami a jejich údaji (název školy, stát, adresa, vzorky, půjčené vzorky, předběžný počet týmů, termín a zda chce poslat škola vzorky). Každý záznam v tabulce lze opět upravit pomocí tlačítka „Upravit“ na konci řádku každého záznamu. Nad tabulkou je ještě tlačítko pro vytvoření nové školy.

Při úpravě nebo vytvoření se vyvolá modální okno (obr. 29), ve kterém jsou pole pro úpravu informací o škole. Při potvrzení změn se formulář odesílá s POST požadavkem na backend, kde se změny zpracovávají.



Obr. 29: Modální okno pro změnu dat školy

Na backendu se přijmou data z formuláře, kontroluje se ID školy a pokud není žádné, vytvoří se nová škola se zadanými informacemi. Pokud je ID validní, vyhledá se daná škola a porovnávají se změněné informace. Když je nějaká změna, informace se změní a zapíší do databáze.

Po zapsání změn se databáze uloží a uživatel je o změně informován. V případě neúspěšného pokusu o změnu, se databáze vrátí do původního stavu. Nabídka rolí se vybírá z přijatých dat z backendu (student, učitel). Taktéž nabídka škol je ze seznamu z databáze. Po stisknutí tlačítka „Registrovat“ se formulář odešle a POST požadavek se zpracuje na pozadí serveru.

4.4 Funkce aplikace

Při vývoji aplikace bylo nutno vytvořit několik funkcí, které zajišťují bezproblémový chod aplikace a zároveň splňují požadavky na návrh aplikace.

4.4.1 Ochrana hesel

Pro jednoduchost použití by se heslo mohlo ukládat přímo do databáze v čitelné formě. Pro přihlášení se pak kontroluje jen shodnost řetězce. Nicméně takové řešení není vhodné z hlediska zabezpečení. V případě odcizení by pak měl daný útočník veškerá hesla a tím i přístup všude. Hesla by si navíc mohl zobrazit i administrátor, což by pak bylo ve sporu s ochranou osobních údajů. Hesla se proto ukládají již zašifrovaná jako hashová hodnota. [26]

V aplikaci je použita knihovna Werkzeug (kapitola 3.4.2) pro zabezpečení hesel pomocí hashovací funkce. Konkrétně je zde využita metoda `generate_password_hash` pro vytvoření hash hodnoty reprezentující heslo (obr. 30) a metoda `check_password_hash` pro ověření hesla při přihlášení (obr. 31).

```
from werkzeug.security import generate_password_hash

# uživatel zadá heslo jako zadane_heslo
zadane_heslo = "tajne_heslo_uzivatele"

# následně se generuje hash zadávaného hesla
hashed_zadane_heslo = generate_password_hash(zadane_heslo)
```

Obr. 30: Kód pro vytvoření hash hesla

Nastavení knihovny Werkzeug pro hashování hesel má výchozí hodnoty, které jsou obvykle vhodné pro většinu aplikací. Pro tuto aplikaci jsou tedy zvoleny parametry výchozí:

- hashovací algoritmus scrypt,
- automatické generování kryptografické soli (salted hash) pro zvýšení bezpečnosti,
- výchozí počet iterací.

Pokud je potřeba použití jiných parametrů hashování hesel, lze je upravit v příkazu generate_password_hash. Příkaz umožňuje specifikovat konkrétní parametry hashování, jako je například počet iterací. Obecně platí, že zvyšování počtu iterací zpomalí proces hashování, což ztěžuje útoky typu „brute-force“, ale může zároveň zvýšit čas potřebný pro vygenerování hesla.

Výstupem funkce je pak řetězec, který se ukládá do databáze k určitému záznamu uživatele.

Pro ověření zadaného hesla (obr. 31) se používá metoda `check_password_hash`. Tato funkce vyžaduje dva vstupní argumenty. Prvním je uložený řetězec v databázi u uživatele jako heslo a druhým pak řetězec ze vstupu, který chceme porovnat s uloženým heslem.

```
from werkzeug.security import check_password_hash
# uživatel zadá heslo jako zadane_heslo
# heslo uložené v databázi jako hash je ulozene_heslo
# ověření hesla zadaného
if check_password_hash(ulozene_heslo, zadane_heslo):
    print("Heslo je správné")
else:
    print("Heslo je nesprávné")
```

Obr. 31: Kód pro ověření zadaného hesla

4.4.2 Administrátorské rozhraní

Pro jednoduchý přístup k datům v databázi je využito knihovny Flask-Admin. Pomocí této knihovny je vytvořeno administrátorské rozhraní (obr. 36). V inicializačním skriptu jsou definovány přístupy k tabulkám databáze (uživatelé, školy, týmy, vazby tým-student, termíny, úlohy, hodnocení). Pro přístup k datům se musí definovat, co všechno se bude zobrazovat z daných tabulek, případně měnit atd. V administrátorském rozhraní je na navigační liště odkaz na každou definovanou tabulku a pak vpravo se nachází dva odkazy. První odkaz je pro návrat do aplikace, vrátí uživatele na domovskou stránku. Druhý odkaz je přímo pro odhlášení uživatele, poté se zobrazí stránka přihlášení.

Administrátorské rozhraní je přístupné jak pro administrátora, tak i pro každého správce dle požadavků firmy. Na tuto stránku se uživatel dostane skrz odkaz „Administrace“ v navigační liště a odkazuje na URL `/admin`. Tento odkaz je přístupný pouze pro oprávněné uživatele.

Modelové pohledy

Zobrazení tabulek vychází z třídy `AdminModelView`, která je rozšířena ze základní třídy `ModelView` z knihovny Flask-Admin. Tato rozšířená třída je použita pro vytvoření modelových pohledů pro správu dat. Zajišťuje, že tyto pohledy jsou přístupné pouze pro přihlášené uživatele s rolí administrátora nebo správce. Pokud uživatel nemá přístup, je přesměrován na domovskou stránku aplikace.

Uživatelé (User_admin)

Modelový pohled `User_admin` pro správu uživatel je definován pro zobrazení základních údajů v tabulce v administrátorském rozhraní (ID, role, email, jméno, příjmení, telefon, škola, max. počet týmů a parametr ověřeného účtu). Pohled je definován pomocí:

- `column_list` – seznam sloupců, které se mají zobrazit v pohledu,
- `column_labels` – popisky jsou tímto parametrem přiřazeny ke sloupcům,
- `column_filters` – definice sloupců v tabulce, které je možné filtrovat,
- `column_searchable_list` – sloupce, dle kterých může uživatel vyhledávat,
- `column_sortable_list` – sloupce, dle kterých lze záznamy v tabulce seřazovat.

Pak je přidána další vlastnost pro tvorbu nebo úpravu pomocí modálního okna. V rámci tohoto okna lze pak definovat parametry (`form_columns`), které se budou zobrazovat a lze je upravit ve formuláři. Některým polím jsou pak pomocí `form_extra_fields` přidány další vlastnosti:

- jméno, příjmení, email – vstupní pole akceptuje řetězec, který nesmí být prázdný (`DataRequired`),
- heslo – vstupní pole je definováno jako vstup pro heslo (černé tečky místo znaků) a je opět nutné, aby nebylo pole prázdné (`DataRequired`).

Pro zadání nebo úpravu již stávajícího hesla je potřeba, aby se řetězec ze zadaného pole neuložil přímo do databáze, ale aby z něj byla vygenerována hash hodnota, která bude uložena. Zde je doplněna tedy funkce `on_model_change` (obr. 32), která se spustí při úpravě nebo vytvoření nového záznamu.

```
def on_model_change(self, form, model, is_created):
    # při tvorbě uživatele, generován hash hesla
    if is_created:
        model.heslo = generate_password_hash(form.heslo.data)
    else:
        # kontrola změny hesla, generován hash hesla
        old_password = form.heslo.object_data
        if not old_password == model.heslo:
            model.heslo = generate_password_hash(form.heslo.data)
```

Obr. 32: Funkce pro ukládání hesla při změně v administrátorském rozhraní

Školy (Skola_admin)

Modelový pohled `Skola_admin` je definován pro správu informací o školách (ID, stát, název školy, adresa, půjčení vzorků, půjčené vzorky, předběžný počet týmů, termín účasti a zda chce poslat škola vzorky). Parametry jsou použity stejně jako pro uživatele, je umožněno záznamy filtrovat, vyhledávat a seřazovat. Úprava probíhá opět v modálním okně. Upraveno je zde pole pro stát, které v modálním okně nabízí dvě možnosti k výběru (CZ/SK) a je povinné.

Týmy (Tym_admin)

Modelový pohled `Tym_admin` je definován pro správu informací o týmech (ID, číslo týmu, informace o učiteli, škola, účast ve vedlejší soutěži, vybraný termín a skupina). Funkce filtrování, vyhledávání a seřazování záznamů jsou opět zachovány.

Informace o učiteli jsou spojeny do jedné buňky díky pomocné funkci a následnému použití příkazu `column_formatters` pro formátování dat do buňky. Informace jsou spojeny do jednoho řetězce ve formátu „jméno příjmení, email“ učitele.

Úpravy a vytvoření týmu je realizováno skrz modální okno. Pro pole učitel je upraven vstup pro výběr možností (obr. 33) díky pomocné funkci `ucitel_query` pro získání všech učitelů z databáze a následně pomocí formátovací funkce `format_ucitel` jsou informace o učitelích shromážděny do jednoho řetězce.

```
def ucitel_query():
    return User.query.filter_by(role_id=2)

def format_ucitel(ucitel):
    if ucitel:
        return f"{ucitel.jmeno} {ucitel.prijmeni},
                {ucitel.email}, {ucitel.skola.nazev}"
    return ''
```

Obr. 33: Funkce pro získání učitelů a pro formátování informací

Navíc je v modálním okně pole pro vypsání členů týmu. Opět je zde nutné upravit vstup pro toto pole a jsou načteni všichni studenti a následně se informace o nich stejně formátují jako u učitelů.

Obě pole jsou pomocí příkazu `form_extra_fields` (obr. 34) definována pro vstupní informace o učitelích a studentech. Díky formátovacím funkcím jsou pak data připravena pro výpis v seznamu. Učitel lze vybrat pouze jeden, ale členů je možno vybrat více, a proto je pro pole členů umožněn vícenásobný výběr (QuerySelectMultipleField).

```
form_extra_fields = {
    'ucitel': QuerySelectField(
        label='Učitel',
        query_factory=ucitel_query,
        allow_blank=True,
        get_label=format_ucitel
    ),
    'clenove': QuerySelectMultipleField(
        label='Studenti',
        query_factory=student_query,
        allow_blank=True,
        get_label=format_student
    )}
```

Obr. 34: Definice pro vstup dat pro pole učitele a členů týmu

Vazby Tým-Student (Tym_Student_admin)

Modelový pohled `Tym_Student_admin` je definován pro správu vazeb mezi týmy a studenty (tým, škola, student). Pro tyto sloupce jsou definovány formátovací funkce (obr. 35), tak aby zobrazily číslo týmu, školu učitele (název, adresa, stát) a informace o studentovi (ID, jméno, příjmení, email).

```
def _tym_formatter(view, context, model, name):
    return f"{model.tym.cislo_tymu}"

def _skola_formatter(view, context, model, name):
    return f" {model.tym.ucitel.skola.nazev},
              {model.tym.ucitel.skola.adresa},
              {model.tym.ucitel.skola.stat}"

def _student_formatter(view, context, model, name):
    return f"ID: {model.student.id}, {model.student.jmeno}
                 {model.student.prijmeni} {model.student.email}"

column_formatters = {'tym': _tym_formatter,
                     'skola': _skola_formatter,
                     'student': _student_formatter}
```

Obr. 35: Funkce pro formátování do tabulek

Úpravy a vytvoření týmu je realizováno opět skrz modální okno. Pro pole tým je upraven vstup pro výběr možností opět podobnou funkcí pro získání všech týmů z databáze a následně pomocí formátovací funkce jsou data upravena pro výpis ze seznamu. Pole student využívá již definovanou funkci pro získání studentů a funkci pro následné formátování. Obě pole jsou definována ve `form_extra_fields` jako výběr ze seznamu (QuerySelectField).

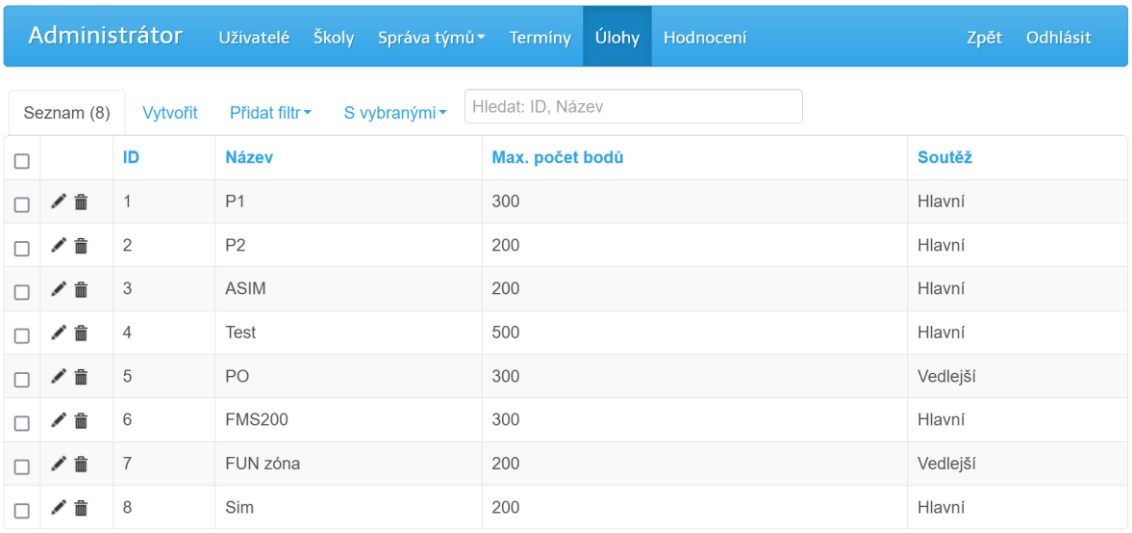
Termíny (Termin_admin)

Modelový pohled `Termin_admin` je definován pro správu informací o termínech (ID, datum, stát, město, adresa konání, funkce zobrazení výsledků). Funkce filtrování, vyhledávání a seřazování záznamů jsou pro určité sloupce zachovány.

Pro úpravu nebo vytvoření záznamu je opět otevřeno modální okno se vstupními poli pro úpravu informací o termínu. Všechna pole jsou povinná a stát se vybírá z definovaného seznamu (CZ/SK). Zobrazování výsledků pro daný termín se ovládá pomocí zaškrtávacího políčka dle potřeby.

Úlohy (Uloha_admin)

Modelový pohled `Uloha_admin` je definován pro správu informací o soutěžních úlohách (ID, název, maximální počet bodů a označení hlavní nebo vedlejší soutěže). Funkce filtrování, vyhledávání a seřazování záznamů v tabulce jsou pro určité sloupce zachovány.



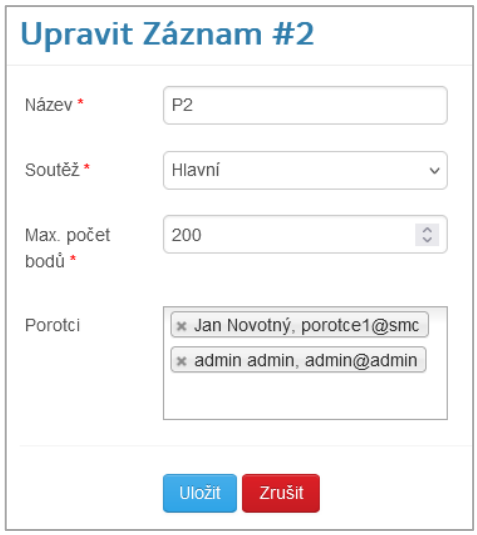
Obr. 36: Administrační rozhraní pro úlohy

Pro sloupec označení hlavní nebo vedlejší soutěže je definována formátovací funkce (obr. 37), která vrací do tabulky „Hlavní“ nebo „Vedlejší“ dle čísla v parametru označení soutěže.

```
def _soutez_formatter(view, context, model, name):
    if model.soutez == 1:
        return 'Hlavní'
    elif model.soutez == 2:
        return 'Vedlejší'
    else:
        return 'Neznámá'
```

Obr. 37: Funkce pro formátování označení typu soutěže

Pro úpravu nebo vytvoření úloh je opět otevřeno modální okno (obr. 38). Pole pro úpravu jsou všechna povinná až na poslední přidané pole. Toto pole načítá všechny porotce, kteří mohou danou úlohu hodnotit. Pole je definováno jako možnost vícenásobného výběru (QuerySelectMultipleField) a poskytuje k výběru všechny porotce, popřípadě administrátora z databáze pomocí přídavné funkce a následně i formátovací.



Obr. 38: Modální okno pro úpravu úlohy

Hodnocení (Hodnoceni_admin)

Modelový pohled `Hodnoceni_admin` je definován pro správu jednotlivých hodnocení týmů ke všem úlohám (ID, tým, hodnotící porotce, úloha a získané skóre týmu k dané úloze). Tabulku lze filtrovat dle týmu, porotce či úlohy a seřazení tabulky je možné dle sloupce skóre.

Informace o týmu (číslo, škola), porotci (jméno, příjmení, email) a úlohách (název) jsou díky formátovacím funkcím shromážděny do samostatných sloupců.

Pro úpravu nebo vytvoření záznamu slouží opět modální okno. V tomto okně jsou 4 pole, kde je možná úprava. Pole pro tým, porotce a úlohy jsou definována pro možnost výběru ze seznamu (QuerySelectField). Data pro seznam jsou získána z databáze díky funkcím `query` a pak formátováním. Poslední pole skóre týmu je povinné pro zadání čísla.

4.4.3 Lokalizace

Pro první testování byla stránka administrátorského rozhraní v angličtině (popisky, filtry, tlačítka). Je předpokládáno, že správci nebo admin budou mít toto rozhraní v českém jazyku. Proto bylo využito knihovny Flask-Babel, která se definuje v inicializačním skriptu s výchozím jazykem češtinou. Celé administrátorské rozhraní je tedy v českém jazyku.

Mimo administrátorské rozhraní bylo potřebné se zaměřit i na všechny ostatní stránky aplikace. Uživatelé této aplikace budou z České republiky a Slovenska. Dle požadavků je tedy potřebné mít každou stránku v jazyku uživatele.

Pro tuto aplikaci byl vytvořen jednoduchý slovník v samostatném souboru `translations.py`. V tomto souboru jsou definovány oba jazyky (CZ, SK). Pro oba jazyky jsou pomocí proměnných definovány určitá slovní spojení nebo věty pro výpis na všech stránkách. Každý záznam je tedy jak v českém jazyce, tak i ve slovenském.

Pro získání slovních spojení dle jazyka se při každém načtení stránky spouští funkce `get_lang` (obr. 39). Tato funkce přijímá vstupní argumenty o přihlášeném uživateli, případně je možno jako vstup použít určitý termín. Funkce probíhá takto:

- nejprve se získá seznam preferovaných jazyků prohlížeče,
- pokud je ve vstupních argumentech termín, řídí se výběr jazyka dle státu termínu,
- jinak pokud má přihlášený uživatel školu a ta stát, tak se řídí výběr jazyka dle daného státu školy,
- nebo pokud se nachází v preferovaných jazycích prohlížeče čeština nebo slovenština, bere se jazyk překladu jako první nalezený v seznamu,
- když nevyhovuje žádná podmínka je výchozím jazykem nastavena čeština,
- nakonec je výstupem funkce část slovníku ze souboru vybraná dle získaného jazyka.

```
def get_lang(current_user, termin=[]):
    browser_languages = request.accept_languages
    preferred_languages = [language for language, _ in browser_languages]
    if termin and (termin.stat == "CZ" or termin.stat == "SK"):
        lang = termin.stat
    elif current_user.is_authenticated and \
         hasattr(current_user.skola, 'stat') and \
        (current_user.skola.stat =="CZ" or current_user.skola.stat =="SK"):
        lang = current_user.skola.stat
    elif 'cs' in preferred_languages or "sk" in preferred_languages:
        for pref in preferred_languages:
            if pref == "cs": lang = "CZ"; \
                break
            elif pref == "sk": lang = "SK"; \
                break
    else: lang = "CZ"
    text = translations.get(lang, translations["CZ"]) # výchozí čeština
    return text
```

Obr. 39: Funkce pro získání slovníku dle jazyka

Z backendu se slovník pak vždy posílá jako proměnná pro frontend. Každá stránka se pak odkazuje na danou část slovníku při výpisu slovních spojení nebo vět.

4.4.4 Export dat

V rámci pozdějšího zpracování dat je dle požadavků v aplikaci vytvořena možnost exportovat data do tabulek programu Excel. Tuto možnost mají uživatelé s rolí správce nebo administrátor. Na stránce ovládacího panelu jsou ve spodní části tlačítka sloužící pro export dat. Export do souboru je rozdělen na dva typy. Prvním typem exportu je uložení dat z databáze z tabulek uživatelů, škol, týmů, úloh a termínů. Dalším souborem pro export jsou tabulky výsledků. Typ exportu se rozpoznává skrz argument při přístupu na stránku `/export`.

Export je realizován s pomocí OpenPyXL knihovny a také třídy BytesIO, díky které lze vytvořený soubor uložit do paměti RAM a pak jej stáhnout.

Skript začne kontrolou role uživatele a v případě neoprávněného přístupu vyvolá přesměrování na domovskou stránku. Následně se identifikuje typ exportu a připraví se nový sešit Excel. Pro export dat je vytvořeno několik funkcí. Každá funkce přijímá data získané z databáze a následně je zpracovává do samostatného listu souboru.

```
def create_users_sheet(wb, users):
    ws_users = wb.active
    ws_users.title = "Uživatelé"
    columns = ['ID', 'Role', 'Jméno', 'Příjmení', 'Email', 'Telefon',
               'Škola', 'Ověřený', 'Max Týmů']
    for idx, column in enumerate(columns, start=1):
        ws_users.cell(row=1, column=idx, value=column)

    for row_idx, user in enumerate(users, start=2):
        ws_users.cell(row=row_idx, column=1, value=user.id)
        ws_users.cell(row=row_idx, column=2,
                      value=user.role.nazev if user.role else '')
        ws_users.cell(row=row_idx, column=3, value=user.jmeno)
        ws_users.cell(row=row_idx, column=4, value=user.prijmeni)
        ws_users.cell(row=row_idx, column=5, value=user.email)
        ws_users.cell(row=row_idx, column=6, value=user.telefon)
        ws_users.cell(row=row_idx, column=7,
                      value=user.skola.nazev if user.skola else '')
        ws_users.cell(row=row_idx, column=8,
                      value='Ano' if user.overeny else 'Ne')
        ws_users.cell(row=row_idx, column=9, value=user.max_tymu)

    adjust_column_widths(ws_users)
```

Obr. 40: Funkce pro vytvoření listu uživatelů

Prvním listem je seznam uživatel, který je vytvořen pomocí funkce `create_users_sheet` (obr. 40). Funkce již přijímá načtený seznam všech uživatelů. Jsou definovány popisky sloupců a následně k nim vyhrazeny i sloupce pro výpis informací o uživatelích. Automaticky se pak díky další funkci `adjust_column_widths` upravuje šířka sloupců, tak aby odpovídala nejdelšímu záznamu ve sloupci.

Tímto způsobem se vytvoří list pro každou kategorii dat (všichni uživatelé, studenti, učitelé, SMC – porotci, správci a administrátor, školy, týmy, úlohy, termíny). Automaticky se nastaví i výchozí název souboru na `SP_data_datum.xlsx` (datum zastupuje aktuální datum exportu).

Při exportu výsledků se používají dvě funkce. První je pro vytvoření dvou listů, kde budou celkové výsledky pro hlavní, respektive vedlejší soutěž. Další funkcí je vytvoření takového počtu listů, aby byl pro každý termín samostatný list pro hlavní a vedlejší soutěž. Takový list pak znázorňuje hodnocení týmů, kteří v daný termín soutěžili. Soubor

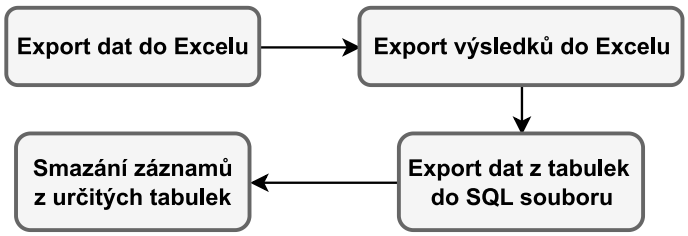
vyjadřuje stejné informace jako stránka aplikace „výsledky“ (kapitola 4.3.11). Týmy jsou z databáze získány stejným způsobem a jejich hodnocení také. Pro každou tabulku pak platí, že jsou týmy seřazeny dle získaného celkového počtu bodů a v prvním sloupci je jejich pořadí.

Pro soubor s výsledky je tentokrát definován výchozí název `SP_vysledky_datum.xlsx`. Skript nakonec vytvořený sešit uloží prostřednictvím knihovny BytesIO do paměti RAM, odkud se následně posílá klientovi pro stáhnutí jako příloha.

4.4.5 Nový ročník soutěže

Po skončení soutěže v daném ročníku je potřeba všechna data uložit a následně databázi připravit pro další ročník. Tuto funkci lze v případě potřeby zavolat pomocí tlačítka v ovládacím panelu, které je přístupné pouze pro administrátora. Když se tato funkce spustí kontroluje se nejprve role přihlášeného uživatele. Pokud nejde o administrátora, skript končí a uživatel je vrácen na domovskou stránku. Následně kód pokračuje:

- kontrolou, zda je vytvořena složka `backup` v adresáři aplikace,
- exportem dat a výsledků do souborů Excel. Soubory jsou uloženy do složky `backup` pro zpětné nahlédnutí dat z ukončeného ročníku,
- definicí tabulek (dat) k exportu do kódu SQL,
- založení souboru `backup_datum.sql`,
- výpis kódu jazyka SQL pro případný import do databáze,
- mazání všech záznamů v tabulkách hodnocení (`Hodnoceni`), vazeb tým-student (`Tym-Student`), týmů (`Tym`) a všech studentů a učitelů z tabulky uživatel (`User`).



Obr. 41: Postup přípravy aplikace pro další ročník

Výsledný soubor s kódem SQL je připraven k importu do již definované databáze. Současné data z tabulek vymaže a nahradí je záznamy, které byly uloženy v době, kdy se data z databáze smazala (příprava na nový ročník). Pro takový import lze použít program HeidiSQL, který má přístup k celé databázi.

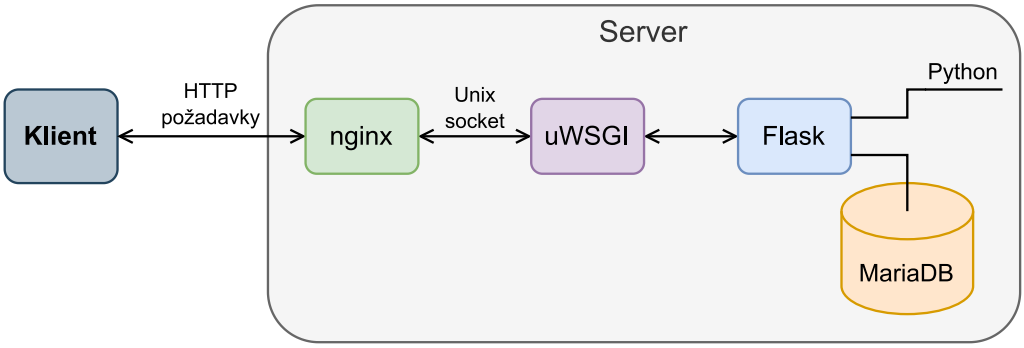
Po skončení skriptu se obnoví stránka ovládacího panelu. Díky informačním panelům o počtech uživatelů a týmů lze ověřit nachystání aplikace a databáze pro nový ročník soutěže.

5 UVEDENÍ DO PROVOZU

Celkový vývoj probíhal na osobním počítači, kde byla spuštěna služba databáze MariaDB a zároveň Flask aplikace na lokálním serveru. Pro postupné doplňování funkcí byla Flask aplikace spuštěna v `debug` módu. V rámci tohoto režimu se mohly zdrojové soubory postupně ladit a aplikace se po každé změně restartovala. V rámci prohlížeče pak stačilo obnovit stránku, změny se projevily nebo se případně vypsal chybný kód při špatné konfiguraci či jiné chybě. Nicméně se nesmí tento režim používat v produkčním prostředí skrz bezpečnostní riziko.

Pro nasazení aplikace na server přístupný veřejnosti byl zřízen vedoucím práce server pro běh této aplikace. Server byl poskytnut v rámci Ústavu automatizace a informatiky (ÚAI). Pro spuštění aplikace na serveru bylo využito těchto nástrojů:

- **uWSGI** – server WSGI (Web Server Gateway Interface), který přijímá HTTP požadavky a předává je do Python aplikace (aplikace je obsluhována tímto serverem),
- **nginx** – webový server a reverzní proxy server, který přijímá http požadavky od klientů a přeposílá do uWSGI aplikace, případně zpět klientovi,
- **uWSGI Emperor** – správce uWSGI procesů. Spouští a zastavuje uWSGI instance dle konfiguračních souborů. Umožňuje snadné spuštění více aplikací na jednom serveru.



Obr. 42: Komunikace serveru a klienta [27]

Na server byly přesunuty všechny soubory potřebné pro spuštění aplikace a byla nainstalována MariaDB databáze. Následně byla testovací data importována do databáze na serveru pomocí SQL kódu. Poté byly upraveny všechny konfigurační soubory pro spuštění aplikace. Nakonec je restartem nástrojů spuštěna aplikace, která je přístupná odkudkoliv pomocí konkrétní URL adresy.

6 ZÁVĚR

V této práci bylo řešeno téma návrhu a realizace databázového systému pro soutěž Stříbrný píst SMC s cílem zajistit bezpečné ukládání a správu dat. Při návrhu systému byl kladen důraz na uživatelskou přívětivost a integraci s dalšími systémy. Práce obsahuje seznámení se se soutěží a analýzu stávající databáze včetně její struktury a potřebných dat, jako jsou informace o soutěžících, organizátorech nebo bodovacím systému.

Práce popisuje moderní technologie používané v oblasti vývoje webových aplikací. Následně jsou zvoleny vhodné postupy pro vlastní návrh databázového systému. Implementace zahrnuje použití robustních a škálovatelných řešení, která zajistila spolehlivý běh systému.

Cílem této práce bylo navrhnout vlastní databázovou strukturu, která zahrnuje správu všech účtů (učitelé, studenti, organizátoři) s nastavením přístupových práv. Pro návrh databázové struktury byl vytvořen ER diagram, vyjadřující vztahy mezi tabulkami. Dále byla implementována správa soutěžních úloh, bodování, ovládací funkce systému a zabezpečení stránek a účtů. Aplikace zahrnuje interaktivní tabulky, které zobrazují výsledky jednotlivých soutěží a celkové pořadí týmů v závislosti na termínu konání soutěže. Webová aplikace je navrhnuta s responzivním designem stránek pro zachování všech funkcí jak na PC, tak i na mobilním zařízení.

Výsledkem práce je komplexní databázový systém, který splňuje všechny stanovené požadavky a poskytuje uživatelsky přívětivé prostředí. Systém byl navržen tak, aby byl snadno rozšiřitelný a přizpůsobitelný různým potřebám soutěže.

V průběhu řešení se objevilo několik zajímavých myšlenek, které by mohly být předmětem dalšího projednání s firmou a případné implementace. Patří sem například možnost rozšíření systému pro více států, automatického exportu vizitek soutěžích pro následný tisk, průběžného ukládání dat z databáze nebo přihlášení administrátora na jiný účet. V rámci těchto možností rozšíření a následné realizaci této webové aplikace na serveru firmy je potvrzena další spolupráce.

Celkově lze konstatovat, že autor dosáhl významného přínosu v oblasti návrhu a realizace databázových systémů pro soutěže. Práce přináší hodnotný příspěvek k efektivní správě soutěží a poskytuje solidní základ pro další rozvoj a zlepšování.

SEZNAM POUŽITÉ LITERATURY

[1] SMC CZ. *O společnosti*. Online. 2024. Dostupné z: https://www.smc.eu/cs-cz/o_spolecnosti. [cit. 2024-05-06].

[2] SMC CZ. *Stříbrný píst SMC*. Online. Dostupné z: https://www.smcengineering.cz/stribrnypist/. [cit. 2024-05-06].

[3] DESIGNVELOPER. *The 10 Best Web App Languages in 2024*. Online. 2024. Dostupné z: https://www.designveloper.com/blog/web-app-languages/. [cit. 2024-02-16].

[4] PECINOVSKÝ, Rudolf. *Python: kompletní příručka jazyka pro verzi 3.11*. Knihovna programátora (Grada). Praha: Grada Publishing, 2023. ISBN 978-80-271-3891-3.

[5] MDN. *WebAssembly*. Online. 2024. Dostupné z: https://developer.mozilla.org/en-US/docs/WebAssembly. [cit. 2024-05-23].

[6] CHOJNOWSKA, Maria. *10 Best Python Web Frameworks*. Online. 2023. Dostupné z: https://sunscrapers.com/blog/10-best-python-web-frameworks/. [cit. 2024-03-04].

[7] POKORNÝ, Jaroslav a VALENTA, Michal. *Databázové systémy*. 2. přepracované vydání. Praha: Česká technika - nakladatelství ČVUT, 2020. ISBN 978-80-01-06696-6.

[8] RIORDAN, Rebecca M. *Vytváříme relační databázové aplikace*. Databáze. Praha: Computer Press, 2000. ISBN 80-7226-360-9.

[9] KUMAR, Dharmendra. *12 Best Databases To Use In 2024: A Comprehensive Guide*. Online. 2023. Dostupné z: https://hevodata.com/learn/best-database/. [cit. 2024-03-07].

[10] ORACLE. *MySQL*. Online. 2024. Dostupné z: https://www.mysql.com/. [cit. 2024-03-07].

[11] *About SQLite*. Online. 2023. Dostupné z: https://www.sqlite.org/about.html. [cit. 2024-03-07].

[12] POSTGRESQL TUTORIAL. *What is PostgreSQL?* Online. 2024. Dostupné z: https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/. [cit. 2024-03-07].

[13] MARIADB KNOWLEDGE BASE. *About MariaDB Software*. Online. 2010, 2019. Dostupné z: https://mariadb.com/kb/en/about-mariadb-software/. [cit. 2024-03-07].

[14] STITCH. *MySQL vs. MariaDB: drop-in or diverging?* Online. 2024. Dostupné z: https://www.stitchdata.com/resources/mysql-vs-mariadb/. [cit. 2024-05-22].

[15] *Flask-Login 0.7.0 documentation*. Online. Dostupné z: https://flask-login.readthedocs.io/en/latest/. [cit. 2024-03-06].

[16] EBY, Phillip J. *PEP 3333 – Python Web Server Gateway Interface v1.0.1*. Online. 2010, 2023. Dostupné z: https://peps.python.org/pep-3333/. [cit. 2024-05-22].

[17] *PyPi: SQLAlchemy 2.0.30*. Online. 2024. Dostupné z: https://pypi.org/project/SQLAlchemy/. [cit. 2024-03-05].

[18] *Flask-Migrate documentation*. Online. Dostupné z: https://flask-migrate.readthedocs.io/en/latest/. [cit. 2024-03-07].

[19] *Flask-Admin 1.6.0 documentation*. Online. Dostupné z: https://flask-admin.readthedocs.io/en/latest/. [cit. 2024-03-16].

[20] GEEKSFORGEEKS. *Bootstrap Tutorial*. Online. 2024. Dostupné z: https://www.geeksforgeeks.org/bootstrap/. [cit. 2024-04-01].

[21] WTFORMS. *Fields - WTForms Documentation (3.0.x)*. Online. Dostupné z: https://wtforms.readthedocs.io/en/3.0.x/fields/. [cit. 2024-03-17].

[22] *Python documentation: io — Core tools for working with streams*. Online. Dostupné z: https://docs.python.org/3/library/io.html. [cit. 2024-05-06].

[23] CHENG, Shenghui. Application Challenges of Web 3.0. Online. In: *Web 3.0: Concept, Content and Context*. Springer Singapore, 2024, s. 179–193. ISBN 978-981-99-6319-5. Dostupné z: https://doi.org/10.1007/978-981-99-6319-5_8. [cit. 2024-05-15].

[24] *RNDGen: Data Generator*. Online. Dostupné z: https://www.rndgen.com/data-generator. [cit. 2024-04-15].

[25] WATT, Adrienne a ENG, Nelson. *Database Design*. Online. 2nd Edition. BCcampus, 2014. Dostupné z: https://opentextbc.ca/dbdesign01/. [cit. 2024-03-15].

[26] MIKUĽÁK, Martin. *Programujeme WWW stránky pro úplné začátečníky*. Brno: Computer Press, 2011. ISBN 978-80-251-3252-4.

[27] NGUYEN, Hien D. *Setup Django behind uWSGI and NGINX on CentOS 7*. Online. 2017. Dostupné z: https://www.vndeveloper.com/django-behind-uwsgi-nginx-centos-7/. [cit. 2024-05-22].

SEZNAM OBRÁZKŮ

Obr. 1: Podmíněný příkaz pro přímé spuštění modulu 29
Obr. 2: ER diagram prvního návrhu databáze 31
Obr. 3: Kód SQL 32
Obr. 4: Kód SQLAlchemy 32
Obr. 5: Kód pro tabulku rolí (`Role`) 34
Obr. 6: Kód pro tabulku hodnocení (`Hodnoceni`) 37
Obr. 7: ER diagram navrhnuté databáze [25] 38
Obr. 8: Přístup na stránky dle role uživatele 39
Obr. 9: Navigační lišta pro porotce 40
Obr. 10: Znázorněné přepínače při zapnutém a vypnutém stavu 41
Obr. 11: Kontrola údajů a přidání uživatele 43
Obr. 12: Registrační formulář 44
Obr. 13: Vývojový diagram přihlašování 45
Obr. 14: Profil administrátora 47
Obr. 15: Registrace týmu 49
Obr. 16: Zobrazení týmů učitele 50
Obr. 17: Kód pro výpis informací o týmech 51
Obr. 18: Kód pro výpis týmů s hodnocením ve všech úlohách 52
Obr. 19: Hodnocení týmů porotcem 53
Obr. 20: Modální okno pro hodnocení úloh 53
Obr. 21: Kód pro kontrolu přístupu na výsledky termínu 54
Obr. 22: Zobrazení výsledků pro určitý termín 55
Obr. 23: Ovládací panel aplikace 56
Obr. 24: Kód pro získání počtů týmu dle termínu 57
Obr. 25: Panely pro zobrazení počtu uživatelů a týmů 57
Obr. 26: Modální okno pro ověření uživatelů 58
Obr. 27: Zobrazení seznamu všech škol 61
Obr. 28: Tabulka pro správu týmů 61
Obr. 29: Modální okno pro změnu dat školy 62
Obr. 30: Kód pro vytvoření hash hesla 63
Obr. 31: Kód pro ověření zadaného hesla 64
Obr. 32: Funkce pro ukládání hesla při změně v administrátorském rozhraní 65
Obr. 33: Funkce pro získání učitelů a pro formátování informací 66
Obr. 34: Definice pro vstup dat pro pole učitele a členů týmu 66
Obr. 35: Funkce pro formátování do tabulek 67
Obr. 36: Administrační rozhraní pro úlohy 68
Obr. 37: Funkce pro formátování označení typu soutěže 68
Obr. 38: Modální okno pro úpravu úlohy 69
Obr. 39: Funkce pro získání slovníku dle jazyka 70

Obr. 40: Funkce pro vytvoření listu uživatelů.....71
Obr. 41: Postup přípravy aplikace pro další ročník.....72
Obr. 42: Komunikace serveru a klienta [27].....73

SEZNAM ZKRATEK

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CSS Cascading Style Sheets

CZ Česká republika

ERD Entity-Relationship Diagram

FTP File Transfer Protocol

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

ID Identifikátor

JPEG Joint Photographic Experts Group

JSON JavaScript Object Notation

ORM Object-Relational Mapper

PC Personal Computer

PDF Portable Document Format

PHP PHP: Hypertext Preprocessor

RAM Random-Access Memory

SK Slovenská republika

SQL Structured Query Language

URL Uniform Resource Locator

WSGI Web Server Gateway Interface

XML Extensible Markup Language