FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Webová aplikace pro odevzdávání maturitních projektů
Student:	Ondřej Cach
Vedoucí:	Ing. Jiří Dostál, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

V rámci maturitní zkoušky na Střední průmyslové škole elektrotechnické a Vyšší odborné škole Pardubice žáci vytváří maturitní projekt. Pro potřeby školy je žádoucí mít informace o projektech uloženy v jednom informačním systému, který ulehčí správu projektů i jejich odevzdání a hodnocení.
Cílem této práce je implementovat webovou aplikaci pro odevzdávání maturitních projektů s ohledem na bezpečnost a kryptografické ověření odevzdání práce.

1) Proveďte sběr a analýzu požadavků budoucích uživatelů aplikace.
2) Proveďte rešerši existujících řešení s přihlédnutím k potřebám střední školy.
3) Navrhněte bezpečnostní model pro nezpochybnitelné odevzdání souborů maturitního projektu.
4) Navrhněte webovou aplikaci pro správu projektů s ohledem na hlavní zjištěné požadavky.
5) Implementujte webovou aplikaci podle vykonané analýzy.
6) Při vývoji aplikace se řiďte bezpečnostními doporučeními projektu OWASP.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 20. ledna 2023 v Praze.

Bakalářská práce

WEBOVÁ APLIKACE PRO ODEVZDÁVÁNÍ MATURITNÍCH PROJEKTŮ

Ondřej Cach

Fakulta informačních technologií
Katedra počítačových systémů
Vedoucí: Ing. Jiří Dostál, Ph.D.
11. května 2023

Odkaz na tuto práci: Cach Ondřej. *Webová aplikace pro odevzdávání maturitních projektů*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Poděkování vii

Prohlášení viii

Abstrakt ix

Seznam zkratek x

Úvod 1

1 Analýza 3
- 1.1 Současná správa projektů . . . 3
- 1.2 Rešerše existujících řešení . . . 3
- 1.3 Analýza a sběr požadavků uživatelů . . . 5
  - 1.3.1 Funkční požadavky . . . 5
  - 1.3.2 Nefunkční požadavky . . . 6
- 1.4 Model případů užití . . . 7
  - 1.4.1 Aktéři . . . 7
  - 1.4.2 Případy užití . . . 7

2 Návrh 13
- 2.1 Návrh architektury webové aplikace . . . 13
  - 2.1.1 Frontend . . . 13
  - 2.1.2 Backend . . . 14
- 2.2 Návrh uživatelského rozhraní . . . 15
  - 2.2.1 Návrh rozvržení . . . 15
  - 2.2.2 Návrh některých komponent . . . 19
- 2.3 Návrh bezpečnostního modelu . . . 19
  - 2.3.1 Prostředky . . . 19
  - 2.3.2 Bezpečnostní požadavky . . . 21
  - 2.3.3 Identifikace hrozeb . . . 25

3 Implementace 29
- 3.1 Zvolené technologie . . . 29
  - 3.1.1 Použité technologie na frontendu . . . 29
  - 3.1.2 Adresářová struktura frontendu . . . 32
  - 3.1.3 Použité technologie na backendu . . . 33
  - 3.1.4 Adresářová struktura backendu . . . 35
  - 3.1.5 Použité technologie pro ukládání dat . . . 35
  - 3.1.6 Databázový model . . . 35
- 3.2 Snímky obrazovky implementované aplikace . . . 36
- 3.3 Nahrávání souborů . . . 38
- 3.4 Autentizace . . . 39
- 3.5 Autorizace . . . 39

3.6 Práce s časovým razítkem . . . 40
3.7 Kešování časového razítka . . . 40
3.8 Problém N+1 dotazů . . . 42

4 Testování **45**
4.1 Uživatelské testování původního prototypu . . . 45
4.2 Testování nahrávání souborů po kusech . . . 46
4.3 Testování funkčnosti časového razítka . . . 46
4.4 Testování vůči OWASP TOP 10 zranitelnostem . . . 46
4.4.1 A01:2021-Broken Access Control . . . 46
4.4.2 A02:2021-Cryptographic Failures . . . 47
4.4.3 A03:2021-Injection . . . 47
4.4.4 A04:2021-Insecure Design . . . 47
4.4.5 A05:2021-Security Misconfiguration . . . 47
4.4.6 A06:2021-Vulnerable and Outdated Components . . . 48
4.4.7 A07:2021-Identification and Authentication Failures . . . 48
4.4.8 A08:2021-Software and Data Integrity Failures . . . 48
4.4.9 A09:2021-Security Logging and Monitoring Failures . . . 48
4.4.10 A10:2021-Server-Side Request Forgery . . . 49

5 Závěr **51**

A Instalační příručka – frontend **53**
A.1 Verze nástrojů . . . 53
A.2 Nainstalování závislostí . . . 53
A.3 Nastavení proměnných prostředí . . . 53
A.4 Spuštění aplikace v lokálním vývojovém prostředí . . . 53
A.5 Sestavení aplikace pro produkční nasazení . . . 54

B Instalační příručka – backend **55**
B.1 Verze nástrojů . . . 55
B.2 Požadavky na server . . . 55
B.3 Nainstalování závislostí . . . 56
B.4 Nastavení proměnných prostředí . . . 56
B.4.1 Vygenerování klíče . . . 56
B.5 Vytvoření databázových tabulek . . . 56
B.6 Bezpečnostní upozornění . . . 57
B.7 Optimalizace pro nasazení na produkční server . . . 57
B.8 Spuštění aplikace . . . 57

C Snímky obrazovky implementované aplikace **59**

Obsah přiloženého archivu **69**

Seznam obrázků

1.1 Účastníci v modelu případů užití 8
1.2 Model případu užití 11

2.1 Architektura webové aplikace 13
2.2 Wireframe pro layout přihlašovací stránky 15
2.3 Přihlašovací stránka 16
2.4 Wireframe starého návrhu nástěnky s harmonikou 17
2.5 Wireframe pro nástěnku se seznamem projektů (auth layout) 17
2.6 Wireframe pro obrazovku s konkrétním projektem 18
2.7 Wireframe pro rozvržení administace 18
2.8 Wireframe pro modální dialogové okno 20
2.9 Wireframe pro notifikační zprávy 20
2.10 Vytvoření podepsaného časového razítka 24
2.11 Ověření podepsaného časového razítka 24
2.12 Případ užití – nahrání souboru na backend 25

3.1 Návrhové vzory – Active Record vs. Data Mapper 34
3.2 Databázový model 37
3.3 Komunikace se serverem při nahrávání souboru po kusech 39
3.4 Informování uživatele o výsledku ověření časového razítka u souboru 40

C.1 Administrace uživatelů 59
C.2 Administrace projektů 60
C.3 Administrace školního roku 60
C.4 Administrace školního roku – import uživatelů 61
C.5 Úprava zadání maturitního projektu a nahrávání souborů 61
C.6 Nahrávání souborů k projektu – překročení deadline 62
C.7 Nahrávání souborů k projektu – schválení uživatelem 62
C.8 Nahrávání souboru na server 63
C.9 Nahrané soubory u projektu 63
C.10 Nástěnka se seznamem projektů a přepínáním rolí 64
C.11 Stránka pro obnovu zapomenutého hesla 64

Seznam tabulek

2.1 Přehled autorizace rolí pro daný projekt 22

3.1 Přehled GitHub repozitářů JS frameworků (k 6. květnu 2023) 29
3.2 Přehled GitHub repozitářů PHP frameworků (k 7. květnu 2023) 33

Seznam výpisů kódu

1 Získání podepsaného časového razítka pomocí OpenSSL 23
2 Ověření podepsaného časového razítka pomocí OpenSSL 23
3 Ukázka React komponenty 30
4 Ošetření HTML vstupu pomocí Dompurify 32
5 Získání podepsaného časového razítka pomocí knihovny PhpAsn1 35
6 Definování pravidla v politice a jeho ověření 41
7 Kešování výsledku ověření časového razítka 42
8 Kód obsahující problém N+1 dotazů 43
9 Problém N+1 dotazů v podobě SQL dotazů 43
10 Kód s eager načítáním, které řeší problém N+1 dotazů 43
11 Odpovídající SQL dotazy, když je použito eager načítání 43

Rád bych poděkoval za vstřícnost, trpělivost, ochotu ke konzultacím a cenné rady a připomínky svému vedoucímu bakalářské práce, Ing. Jiřímu Dostálovi, Ph.D.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2023 .....

Abstrakt

Tato bakalářská práce se zabývá návrhem systému pro odevzdávání maturitních prací na Střední průmyslové škole elektrotechnické a Vyšší odborné škole Pardubice. Cílem práce je implementování webové aplikace pro odevzdání maturitních projektů s ohledem na bezpečnost a kryptografické ověření práce. V řešení jsou použity metody softwarového inženýrství pro analýzu požadavků a případů užití. Dále je vypracován návrh webové aplikace včetně bezpečnostního modelu pro nezpochybnitelné odevzdání souborů maturitních projektů s využitím digitálně podepsaného časového razítka. Na předchozí analýzu a návrh navazuje implementace webové aplikace, která je řešena pomocí Single Page Application, psané v Reactu, na frontendu, interagující s REST API, používajícím framework Laravel, na backendu. Výsledkem této práce je funkční webová aplikace umožňující střední škole efektivně spravovat maturitní práce žáků.

Klíčová slova webová aplikace, odevzdávání maturitních prací, digitálně podpesaná časová razítka, React, Laravel

Abstract

This bachelor's thesis deals with the design of GCSE project submissions system at the Secondary School of Electrical Engineering and Higher Vocational School Pardubice. The objective of the thesis is to implement a web application for the submission of GCSE projects considering the security and cryptographic verification of the work. Software engineering methods are used to analyze requirements and use cases. Furthermore, the design of the web application including a security model for non-repudiable submission of GCSE project files using trusted timestamping is developed. The previous analysis and design are followed by the implementation of the web application, which is resolved using a Single Page Application, written in React, on the frontend, interacting with REST API, using Laravel framework, on the backend. The result of this work is a fully working web application that allows a high school to effectively manage the GCSE projects of students.

Keywords web application, GCSE projects submission, trusted timestamping, React, Laravel

Seznam zkratek

API	Application Programming Interface
CA	Certificate Authority
CI/CD	Continuous Integration/Continuous Deployment
CSRF	Cross Site Request Forgery
CORS	Cross-Origin Resource Sharing
CSP	Content Security Policy
CSS	Cascading Style Sheets
DER	Distinguished Encoding Rules
DoS	Denial of Service
HOTP	HMAC-based One-time Password
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JS	JavaScript
JSON	JavaScript Object Notation
JSX	JavaScript Syntax Extension
LMS	Learning Management System
NIST	National Institute of Standards and Technology
OTP	One-time Password
ORM	Object-Relational Mapping
OWASP	Open Worldwide Application Security Project
PHP	Hypertext Preprocessor
PKI	Public Key Infrastructure
REST	Representational State Transfer
RFC	Request for Comments
SASS	Syntactically Awesome Stylesheet
SCSS	Sassy CSS
SPŠE	Střední průmyslová škola elektrotechnická
SMTPS	Simple Mail Transfer Protocol Secure
SPA	Single Page Application
SSL	Secure Sockets Layer
SSRF	Server-Side Request Forgery
SQL	Structured Query Language
TLS	Transport Layer Security
TOTP	Time-based One-time Password
TSA	Time Stamping Authority
TSP	Time-Stamp Protocol
TSQ	Time Stamp Query
TSR	Time Stamp Response
TTL	Time To Live
UI	User Interface
UUID	Universally Unique IDentifier
UX	User Experience
XSS	Cross Site Scripting
XXE	XML External Entity

Úvod

V rámci maturitní zkoušky na Střední průmyslové škole elektrotechnické a Vyšší odborné škole Pardubice (SPŠE a VOŠ Pardubice) žáci vytváří maturitní projekt. Pro potřeby školy je žádoucí mít informace o projektech uloženy v informačním systému, který sjednotí a ulehčí správu projektů i jejich odevzdání a hodnocení.

Střední škole ze Školského zákona (č. 561/2004 Sb.) a Vyhlášky o bližších podmínkách ukončování vzdělávání ve středních školách maturitní zkouškou (č. 177/2009 Sb.) vyplývají povinnosti ohledně maturitních prací, jako např. kdy je potřeba určit zadání, vedoucího a oponenta maturitní práce žáka, nebo v jakém termínu je třeba předat posudky žákovi a členům zkušební maturitní komise. Také z těchto důvodů je pro školu výhodné mít informační systém, který zastřeší maturitní práce.

Před třemi lety, když autor této bakalářské práce maturoval na SPŠE a VOŠ Pardubice, neexistoval systém, který by umožňoval vedení školy efektivně spravovat maturitní práce.

V současné době stále neexistuje aplikace, která by uspokojila všechny požadavky, které na něj střední škola klade. Proto jedním ze stanovených cílů práce bylo takový systém navrhnout a implementovat v rámci bakalářské práce.

Vyvinutá aplikace umožní vedení střední školy maturitní přehledně projekty spravovat. Také vedoucí a oponenti budou moci sledovat přiřazené projekty, ale i odevzdávat posudky. Studentům aplikace umožní sepsat spolu s vedoucím zadání práce, prostřednictvím aplikace také práci odevzdat a následně si prohlédnout posudky vedoucího a oponenta.

Hlavním cílem této práce je implementovat webovou aplikaci pro odevzdávání maturitních projektů s ohledem na bezpečnost a kryptografické ověření odevzdání práce podle vykonané analýzy.

Cílem teoretické části práce je provedení sběru a analýzy požadavků budoucích uživatelů aplikace, dále pak provedení rešerše existujících řešení s přihlédnutím k potřebám střední školy.

Cílem praktické části práce je navržení bezpečnostního modelu pro nezpochybnitelné odevzdání souborů maturitního projektu a navržení webové aplikace pro správu projektů s ohledem na hlavní zjištěné požadavky. Na tyto cíle navazuje hlavní cíl samotné implementace.

Výsledky práce budou přínosné pro SPŠE a VOŠ Pardubice, která bude aplikaci využívat pro správu a odevzdávání maturitních projektů.

Kapitola 1
Analýza

1.1 Současná správa projektů

Před třemi lety, když na SPŠE a VOŠ Pardubice neexistoval systém pro správu maturitních projektů, probíhala správa projektů následovně. Žáci museli sepsat zadání své práce do textového dokumentu, který poté odevzdali učiteli na předmětu `Maturitní seminář (MS)` nebo `Praxe (PX)`. Tento učitel po sběru zadání v určeném termínu předal dokumenty školnímu koordinátorovi maturitních projektů. Koordinátorovi přišla od učitelů předmětu `MS` a `PX` zadání, která si musel setřídit, aby je mohl zkontrolovat. Dále si také musel udržovat tabulku v kancelářském softwaru, ve které si uchovával informace o žácích, jejich projektech, vedoucích a oponentech. Řádově se jednalo o 160–180 žáků.

Před dvěma roky autor bakalářské práce po domluvě s vedením školy implementoval prototyp aplikace pro správu maturitních projektů, který byl nasazen na školní server a nyní slouží druhým rokem pro účely školy. Ukázal se jako použitelný pro odevzdání zadání maturitní práce a odevzdání souborů, ale chybí mu řádná administrace, ve které by mohlo vedení školy zakládat nový školní rok, importovat do systému uživatele, sledovat odevzdávání souborů projektů, celkově analyzovat a další.

1.2 Rešerše existujících řešení

Existující webovou aplikaci pro odevzdávání maturitních prací, která by splňovala představu, kterou škola má, není jednoduché nalézt. Jedním způsobem, jak takovou aplikaci hledat, je porozhlédnutí se po středních školách stejného zaměření, případně po vysokých školách, kde závěrečné práce mívají podobné náležitosti. Pro nalezení takových škol lze využít web www.atlasskolstvi.cz [1].

Na webech středních škol byla informace o existujícím systému publikována pouze v ojedinělých případech. Jediné aplikace, které se podařilo veřejně dohledat, byly:

- Střední škola informatiky, elektrotechniky a řemesel Rožnov pod Radhoštěm
  - Vlastní systém provozovaný na `https://www.roznovskastredni.cz/matprac/`. [2]
  - „*SYSTÉM PRO SPRÁVU, HODNOCENÍ A EVIDENCI MATURITNÍCH PRACÍ*
    Pro zadávání témat, výběr vedoucího práce, evidenci stavu rozpracování maturitních prací, dílčí i celkové hodnocení, evidenci a archivaci prací slouží školní intranetový systém nazvaný MATPRAC, do kterého mají žáci a vedoucí prací přístup přes webové rozhraní z adresy: https://www.roznovskastredni.cz/matprac.“ [3]

- Střední průmyslová škola strojní a elektrotechnická a Vyšší odborná škola, Liberec
  - Vlastní aplikace provozovaná na `https://prace2.pslib.cloud/`. [4] Tato aplikace slouží pouze pro výběr tématu, odevzdávání probíhá pouze fyzicky.
  - „*Maturitní práce (MP) - L4*
    23.09.2022 - výběr vlastního tématu, které by student chtěl zpracovávat, dohoda s vedoucím práce, příprava cílů a osnovy práce
    30.09.2022 - zadání tématu do systému, rozvaha o finanční náročnosti práce (vedoucí práce) a tisk přihlášky
    (…)
    15.03.2023 - odevzdání práce 1-2 tištěná pare (dle pokynu vedoucího práce) + el. podoba na SD kartě vlepené na zadních deskách práce, všechny praktické výstupy, odevzdání 1 elektronické pare pro SOČ“ [5]
- Střední průmyslová škola a Vyšší odborná škola, Písek
  - Používají LMS (Learning Management System) Moodle provozovaný na `https://moo.sps-pi.cz/`. [6]
  - „*Kompletní práce se odevzdává do do informačního střediska školy v jednom tištěném exempláři, který bude obsahovat elektronický nosič dat (CD, DVD, USB flash disk, SD karta), na kterém bude uvedena kompletně zpracovaná práce včetně příloh. V případě tvorby software, také zdrojový kód navrženého software. V případě projektu, také projektová dokumentace (podrobná technická zpráva, úplná výkresová dokumentace, podrobný rozpočet). Elektronicky na moo.sps-pi.cz; kurz Maturitní práce*“ [7]

Webové stránky vysokých škol poskytovaly veřejně informaci o průběhu odevzdávání závěrečných prací častěji než stránky středních škol. Při této analýze byly nalezeny následující způsoby odevzdání na následujících školách:

- České vysoké učení technické v Praze – Fakulta informačních technologií
  - Vlastní aplikace ProjectsFIT provozovaná na `https://projects.fit.cvut.cz/`. [8]
  - „*TÉMA*
    Student si může procházet nabízená témata v systému ProjectsFIT (https://projects.fit.cvut.cz/).
    (…)
    Vedoucí v ProjectsFIT vytvoří nové zadání a přidělí k němu studenta.
    (…)
    ODEVZDÁVÁNÍ
    (…)
    Odevzdejte práci v ProjectsFIT, včetně příloh.“ [9]
- Univerzita Karlova, Matematicko-fyzikální fakulta
  - Výběr tématu práce a elektronické odevzdávání v rámci informačního systému SIS provozovaným na `https://is.cuni.cz/studium/dipl_st/`. [10]
  - „*Současně se odevzdává elektronická verze bakalářské práce do SIS. Podrobné informace naleznete zde a v Opatření rektora č. 72/2017. Doporučuje se, aby velikost práce nepřesáhla 850 MB.*“ [11]
- Univerzita Hradec Králové – Fakulta informatiky a managementu
  - V rámci systému `eVŠKP` (elektronické odevzdávání vysokoškolských kvalifikačních prací) provozovaným na `https://ris.uhk.cz/evskp/` (nutné přihlášení). [12]
  - „*Před fyzickým odevzdáním jednoho výtisku na katedru musí shodnou elektronickou verzi práce nahrát do systému eVŠKP (elektronická VysokoŠkolská Kvalifikační Práce).*“ [13]

- Vysoké učení technické v Brně
  - Modul `Moje závěrečná práce` v rámci informačního systému IS VUT, dostupným na `https://www.vut.cz/studis/student.phtml?sn=zav_prace_moje`. [14]
  - „*Před odevzdáním listinné formy práce je nutné odevzdat elektronicky v IS VUT v modulu Moje závěrečná práce!*“ [15]

Bylo zjištěno, že školy většinou řeší elektronické odevzdání přes systém, který se zaměřuje výhradně na správu závěrečných prací, případně přes modul v rámci rozsáhlejšího informačního systému. Jelikož se většinou jedná o proprietární systémy pro účely uvedených škol, nebylo možné o těchto aplikacích zjistit více informací.

V jednom případě byl použitý LMS Moodle, který je open source, ovšem pro účely SPŠE a VOŠ Pardubice zbytečně složitý, nespecializuje se na maturitní projekty, ale na výukové kurzy a nesplňuje požadavky školy, které jsou zavedeny v následující podkapitole.

1.3 Analýza a sběr požadavků uživatelů

Sběr a analýza požadavků je prvním a důležitým krokem při vytváření jakékoli aplikace. Požadavky totiž vymezují, co vše by mělo být implementováno v rámci vyvíjené aplikace. Pro potřeby této bakalářské práce bylo použito základní dělení požadavků na funkční a nefunkční.

Funkční požadavky určují, jaké chování bude systém nabízet, zatímco nefunkční požadavky specifikují vlastnosti nebo omezující podmínky dané aplikace. [16]

Všechny následující požadavky vzešly z rozhovorů s vedením SPŠE a VOŠ Pardubice.

1.3.1 Funkční požadavky

- **FP1 – Evidence a správa projektů**
  - Aplikace bude evidovat maturitní projekty. Každý žák má vytvořen maturitní projekt, u kterého může se svým vedoucím upravovat zadání.
  - Priorita: Vysoká
  - Složitost: Vysoká
- **FP2 – Správa uživatelů**
  - Administrátor má možnost spravovat uživatele a jejich role v aplikaci. Několik týdnů před termínem maturitních zkoušek je potřeba do systému přidat předsedy a místopředsedy maturitních komisí, kteří si mohou s dostatečným časovým předstihem projekty prohlédnout.
  - Priorita: Střední
  - Složitost: Střední
- **FP3 – Odevzdání práce, posudků a oskenovaného zadání**
  - Ke každému projektu je potřeba ve vymezeném deadline nahrát soubory. Žák tímto způsobem odevzdává text a data práce. Vedoucí a oponent musí nahrát své posudky. Administrátor nahrává oskenované maturitní zadání podepsané ředitelem školy.
  - Priorita: Vysoká
  - Složitost: Střední
- **FP4 – Import dat do nového školního roku**

- Na začátku nového školního roku je potřeba nastavit mnoho informací, jako jsou třídy, třídní učitelé, žáci a jejich prázdné projekty, atd. Pro zjednodušení a urychlení tohoto procesu je potřeba mít možnost tato data importovat z CSV souboru.
- Priorita: Vysoká
- Složitost: Střední

- **FP5 – Exportování projektů do Microsoft Excel tabulky**
  - Pro další procesy, které souvisí s maturitními pracemi, je potřeba mít možnost data projektů exportovat do tabulkového procesoru. S využitím exportovaných dat se dají ulehčit návazné akce, jako je jmenování vedoucích a oponentů, díky hromadné korespondenci. Dále také importování tématu maturitní práce, jakožto specifikace zkoušky, do informačního systému CERTIS od Cermatu. Zadání témat maturitních prací do systému CERTIS je povinné, jelikož se z něj poté generují maturitní vysvědčení, na kterých je povinnou položkou *Téma maturitního projektu*
  - Priorita: Střední
  - Složitost: Nízká
- **FP6 – Exportování zadání projektů do Microsoft Word dokumentu**
  - Jelikož je nutné zadání maturitních projektů tisknout, aby je ředitel školy mohl podepsat a mohla se archivovat, vzniká nutnost generování dokumentů ze zadání vyplněných v aplikaci.
  - Priorita: Vysoká
  - Složitost: Střední

1.3.2 Nefunkční požadavky

- **NP1 – Dostupnost přes web**
  - Systém bude dostupný formou webové aplikace v internetovém prohlížeči.
- **NP2 – Responzivita webové aplikace**
  - Webová aplikace bude navržena, aby ji bylo možné snadno používat i na mobilních zařízeních alespoň pro prohlížení projektů. Největší návštěvnost aplikace se očekává ze zařízení s větší obrazovkou – PC, notebook.
- **NP3 – Autorizace**
  - Veškeré prostředky, se kterými aplikace pracuje, budou dostupné pouze autentizovaným a zároveň autorizovaným uživatelům. Autorizace bude převážně probíhat pomocí uživatelských rolí v kontextu s daným prostředkem.
- **NP4 – Nezpochybnitelné odevzdání souborů**
  - Při odevzdání souborů je potřeba kryptograficky zaručit, že konkrétní data souboru existovala v určitém čase nahrání. Důležité vlastnosti tedy jsou integrita souboru a časová nepopiratelnost.
- **NP5 – Množství a kapacita dat**
  - Předpokládá se evidování 150–200 projektů za školní rok. Ke každému projektu je povoleno nahrát soubory o velikosti 10,5 GiB celkem (10 GiB data projektu a $5 \times 100 = 500$ MiB dalších souborů – posudky, text práce, sken zadání). Pro každý školní rok je potřeba mít volnou kapacitu na úložišti souborů alespoň $10,5 \times 200 = 2100$ GiB $= 2,1$ TiB.

- Pozn.: Toto je horní odhad na potřebnou kapacitu, z předchozích dvou roků používání prototypu se ukázalo, že kapacita nahrávaných souborů dosahuje horní hranice pouze ve výjimečných případech. Všechny nahrané soubory z daného školního roku zabíraly v součtu pouze 100 GiB disku.

1.4 Model případů užití

Modelování případů užití je dalším krokem, který následuje po sběru a analýze požadavků na aplikaci. Při tomto modelování je potřeba vymezit aktéry aplikace, nalézt případy užití a sepsat jejich scénáře.

1.4.1 Aktéři

Aktéři jsou entity, které s aplikací pracují. Převážně to jsou uživatelé se specifickými rolemi, ale mohou to být i např. další systémy.

- **Žák**
  - Uživatel, který studuje na škole v maturitním ročníku. Tento uživatel má vytvořený maturitní projekt, u kterého bude se svým vedoucím upravovat zadání a později k projektu bude nahrávat soubory.
- **Třídní učitel**
  - Uživatel, který má práva zobrazení projektů žáků, kteří jsou přiřazeni do třídy tohoto učitele.
- **Oponent maturitní práce**
  - Uživatel, který má přístup k projektům, u kterých je přiřazen jako oponent. Tyto projekty potřebuje prostudovat, aby k nim mohl napsat a odevzdat posudky.
- **Vedoucí maturitní práce**
  - Uživatel, který má přístup k projektům, které vede. S žákem daného projektu upravuje zadání maturitní práce a poté bude psát a odevzdávat posudky k těmto projektům.
- **Předseda a místopředseda**
  - Předseda, resp. místopředseda maturitní komise jsou uživatelé, kteří budou mít přístup k projektům tříd, kterým budou předsedat u maturitní zkoušky.
- **Administrátor**
  - Administrátoři jsou uživatelé, kteří spravují celý systém a kontrolují průběh práce na projektech. Mohou zobrazit a upravovat všechny projekty, dále např. nastavují deadline pro odevzdání souborů.

1.4.2 Případy užití

- **UC1 – Zobrazení seznamu projektů**

1. Uživatel si zobrazí stránku `nástěnka`, která obsahuje seznam projektů.
2. Aplikace zobrazí seznam projektů pro zvolenou roli.

3. Pokud má uživatel více rolí, může si zobrazit seznam pro jinou roli stisknutím příslušného tlačítka.
4. Případ užití může pokračovat bodem č. 2.

UC2 – Zobrazení konkrétního projektu

1. Uživatel si zobrazí stránku s konkrétním projektem. Žák je na tuto stránku automaticky přesměrován po úspěšném přihlášení do aplikace. Uživatelé ostatních rolí stisknou řádek projektu v tabulce seznamu projektu.
2. Aplikace zobrazí stránku se zobrazením projektu.

UC3 – Nahrání souborů projektu

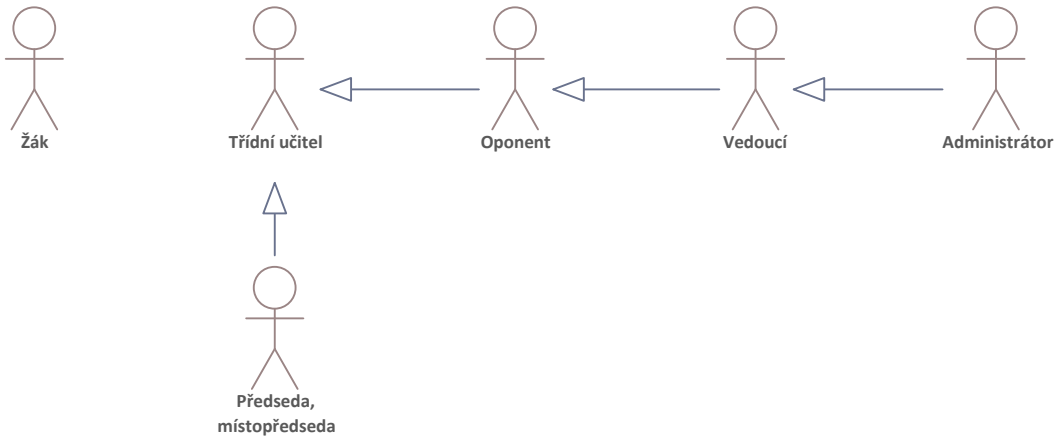
1. Uživatel přejde na stránku s konkrétním projektem (viz UC2 výše).
2. Pokud má roli autorizovanou k nahrání nějakého z definovaných typů souborů a zároveň je současné datum v rozmezí aktuálního deadline, aplikace zobrazí nahrávací formulář pro soubor.
3. Uživatel přetáhne na formulář soubor, který chce nahrát. Případně soubor vybere po kliknutí na tlačítko „Vybrat soubor“.
4. Aplikace soubor odešle na backend a požádá uživatele o potvrzení metadat nahrání.
5. Uživatel potvrdí, že nahraný soubor a jeho metadata odpovídají.

UC4 – Úprava zadání konkrétního projektu

1. Uživatel přejde na stránku s konkrétním projektem (viz UC2 výše).
2. Pokud je autorizovaný k úpravě projektu (žák, vedoucí nebo administrátor) a zároveň projekt není zamknutý pro úpravy, aplikace zobrazí aktivní formulářová pole a tlačítko „Uložit“.
3. Uživatel vyplní nebo upraví formulář a stiskne tlačítko „Uložit“.
4. Aplikace odešle data formuláře na backend a uživateli zobrazí hlášku o výsledku operace.

UC5 – Uzamčení projektu pro úpravy

1. Uživatel přejde na stránku s konkrétním projektem (viz UC2 výše).



Obrázek 1.1 Účastníci v modelu případů užití

2. Pokud je autorizovaný k uzamčení projektu (vedoucí nebo administrátor), aplikace zobrazí tlačítko „Zamknout projekt“, resp. „Zamknout projekt jako administrátor“.
3. Uživatel stiskne tlačítko pro uzamčení.
4. Aplikace odešle žádost na backend a informuje uživatele o výsledku této operace.

- **UC6 – Nastavení globálního deadline**

1. Uživatel s rolí administrátor přejde na stránku `Nastavení deadline` v administraci.
2. Aplikace zobrazí dva formuláře s časovými poli pro začátek a konec deadline. První formulář je pro deadline odevzdání souborů uživatelem. Druhý je pro odevzdání posudků vedoucích a oponentů.
3. Administrátor vyplní formulář a stiskne tlačítko „Uložit“.
4. Aplikace odešle formulář na backend a informuje uživatele o výsledku této operace.

- **UC7 – Prodloužení odevzdání projektu**

1. Uživatel s rolí administrátor si zobrazí seznam všech projektů v administraci a u projektu, kterému chce prodloužit deadline pro odevzdání souborů, stiskne tlačítko „Prodloužit odevzdání“.
2. Aplikace zobrazí vyskakovací okno, ve kterém zobrazí formulář s polem pro prodloužené datum konce deadline žáka a druhým polem pro vedoucího a oponenta.
3. Administrátor vyplní nová data do formuláře a formulář odešle stisknutím tlačítka „Uložit“.
4. Aplikace odešle data formuláře na backend a uživateli zobrazí hlášku o výsledku operace.

- **UC8 – Export zadání**

1. Uživatel s rolí administrátor si zobrazí seznam všech projektů (viz UC1 výše).
2. Aplikace mu zobrazí tlačítka „Exportovat všechna zadání do Excelu“ a „Exportovat všechna zadání do Wordu“.
3. Pokud uživatel stiskne tlačítko „Exportovat všechna zadání do Excelu“:
   a. Aplikace odešle požadavek na backend a vrátí administrátorovi XLSX, případně CSV soubor ke stažení.
4. Pokud uživatel stiskne tlačítko „Exportovat všechna zadání do Wordu“:
   a. Aplikace odešle požadavek na backend a vrátí administrátorovi DOCX soubor ke stažení.

- **UC9 – Import dat**

1. Uživatel s rolí administrátor na stránce `Školní rok` klikne na tlačítko „Založit nový školní rok“.
2. Aplikace zobrazí vyskakovací okno s formulářem obsahujícím pole pro školní rok a pro CSV data.
3. Administrátor zadá aktuální školní rok a klikne na tlačítko „Pokračovat“.
4. Administrátor přetáhne CSV soubor na obrazovku, případně zkopíruje data z CSV souboru do textového pole.
5. Aplikace zobrazí, jaká data bude z CSV souboru importovat.
6. Pokud administrátor souhlasí se zobrazenými daty k importování, stiskne tlačítko „Importovat“, jinak má možnost nahrát jiný soubor, tj. scénář pokračuje od bodu č. 5.

7. Aplikace po stisknutí tlačítka odešle data na backend a informuje uživatele o stavu importování.

UC10 – Administrace prostředků

1. Uživatel s rolí administrátor vstoupí do administrace.
2. Aplikace zobrazí rozhraní administrace s navigačním menu.
3. Administrátor v menu klikne na prostředek, se kterým chce pracovat (např. uživatelé, projekty, ...)
4. Aplikace zobrazí tabulku se záznamy daného prostředku. Dále zobrazí tlačítka pro vytvoření nového záznamu, upravení stávajícího záznamu, smazání záznamu, případně obnovení smazaného záznamu[1].
5. Pokud administrátor stiskne tlačítko „Přidat nový“, případně „Upravit“:
   a. Aplikace zobrazí vyskakovací okno s formulářem obsahujícím pole, která patří k danému prostředku. Pokud se jedná o nový záznam, pole budou prázdná. V případě úpravy stávajícího záznamu se pole předvyplní původními hodnotami.
   b. Administrátor vyplní formulář a klikne na tlačítko „Vytvořit“, resp. „Uložit“.
   c. Aplikace odešle data formuláře na backend a uživateli zobrazí hlášku o provedení této operace.
6. V případě, že administrátor klikne na tlačítko „Smazat“, případně „Obnovit“:
   a. Aplikace zobrazí vyskakovací okno, které požaduje schválení od uživatele.
   b. Pokud administrátor klikne na tlačítko „Potvrdit“:
      i. Aplikace odešle požadavek na backend a zobrazí uživateli příslušnou hlášku.
   c. Pokud administrátor klikne na tlačítko „Zrušit“:
      i. Aplikace skryje vyskakovací okno a akce se neprovede.

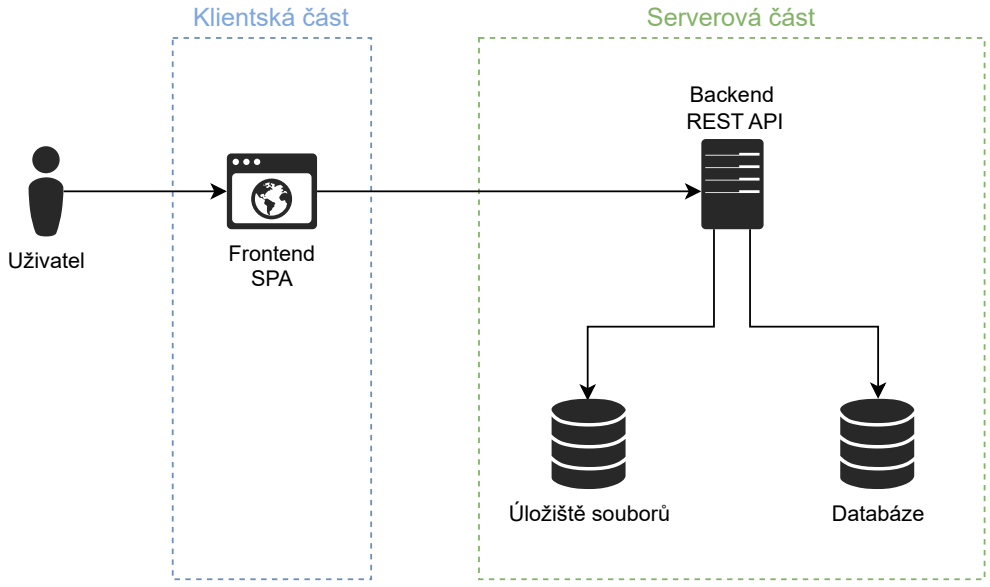
[1] Pozn.: ne každý prostředek poskytuje všechny uvedené operace.

Obrázek 1.2 Model případu užití

Kapitola 2
Návrh

2.1 Návrh architektury webové aplikace

Webová aplikace je rozdělena na dva samostatné celky, které spolu interagují. Těmito celky jsou frontend a backend a detailněji jsou popsány níže. Toto rozdělení je výhodné pro rozšiřitelnost. Kdyby v budoucnu např. nastala potřeba změnit uživatelské rozhraní, může se přepsat pouze frontendová část. Nový frontend by pracoval se stejnými daty, které poskytuje API (Application Programming Interface) na backendu, a proto by do backendové části nemusel proběhnout žádný zásah. Toto rozdělení zobrazuje následující diagram 2.1.



Obrázek 2.1 Architektura webové aplikace

2.1.1 Frontend

Frontend je část aplikace, kterou uživatel přímo vidí a se kterou pracuje, jelikož právě tato část, narozdíl od backendu, poskytuje uživatelské rozhraní (User Interface – UI).

V této bakalářské práci je frontend realizován formou Single-page application (SPA), což je typ webové aplikace, který nahraje jedinou webovou stránku a poté její obsah mění v závislosti na interakci s uživatelem a na datech přijatých z API. Narozdíl od Multi-page application zde není potřeba webovou stránku v prohlížeči obnovovat, čímž lze docílit plynulejších přechodů mezi pohledy v aplikaci a lepší uživatelské přívětivosti (User Experience – UX). Mezi nevýhody, se kterými se SPA aplikace většinou potýkají, lze řadit např. horší optimalizaci pro vyhledávače (SEO – Search Engine Optimalization).

Vzhledem k tomu, že navrhovaná aplikace má sloužit pro účely střední školy a její uživatelé k ní dostanou přístup přes odkaz např. na webových stránkách školy nebo e-mailem, není potřeba řešit její indexaci ve vyhledávačích. Tím odpadá nutnost řešit problém se SEO. Naopak popsané výhody SPA aplikace, které vedou k tomu, že se práce s webovou aplikací přibližuje aplikací nativní, zde budou přínosné.

Jaké technologie byly pro implementaci frontendu vybrány, je popsáno v kapitole 3.1.1.

2.1.2 Backend

Backend je část aplikace, která je zodpovědná za poskytování dat pro frontend a také pro provádění akcí a ukládání dat z frontendu. S touto částí aplikace nepracuje přímo uživatel.

Tato část je také zodpovědná za kontrolu aplikační logiky, autentizaci a autorizaci. Frontendová SPA aplikace sice svoji, odlehčenou, aplikační logiku obsahuje také, avšak nesmí být opomenuto, že frontend pracuje na straně klienta a může být upraven. Naopak backend funguje na straně serveru a je jediným důvěryhodným zdrojem.

V této bakalářské práci je backend realizován jako REST (Representational State Transfer) API. Architektura REST [17] se vyznačuje tím, že:

- Klient a server jsou na sobě nezávislí
- Je bezstavová (stateless) – každý požadavek od klienta na server musí obsahovat všechny potřebné informace, nelze spoléhat na „kontext“
- Klíčovými informacemi jsou prostředky (resources)
- Existuje jednotné rozhraní
  - URL identifikuje zdroj
    - např. `/projects/7/file/PROJECT_TEXT` – označuje soubor typu `PROJECT_TEXT` u projektu s ID = 7
  - HTTP metody (verbs) definují operace nad prostředkem, např.:
    - `GET` – získání obsahu prostředku
    - `POST` – vytvoření prostředku
    - `DELETE` – smazání prostředku
    - `PUT/PATCH` – úprava obsahu prostředku
  - Návratový kód označuje výsledek operace
    - `2xx` – úspěch (např. `201 Created`)
    - 4xx – chyba u klienta (např. `403 Forbidden`)
    - 5xx – chyba na straně serveru (např. `500 Internal Server Error`)
- Umožňuje být ve vrstvené architektuře – např. mezi klienta a server lze přidat server pro vyvažování zátěže (load balancing)

Ve většině případů pro přenos dat používá protokol HTTP a formát JSON (JavaScript Object Notation).

Jaké technologie byly pro implementaci backendu zvoleny, je popsáno v kapitole 3.1.3.

2.2 Návrh uživatelského rozhraní

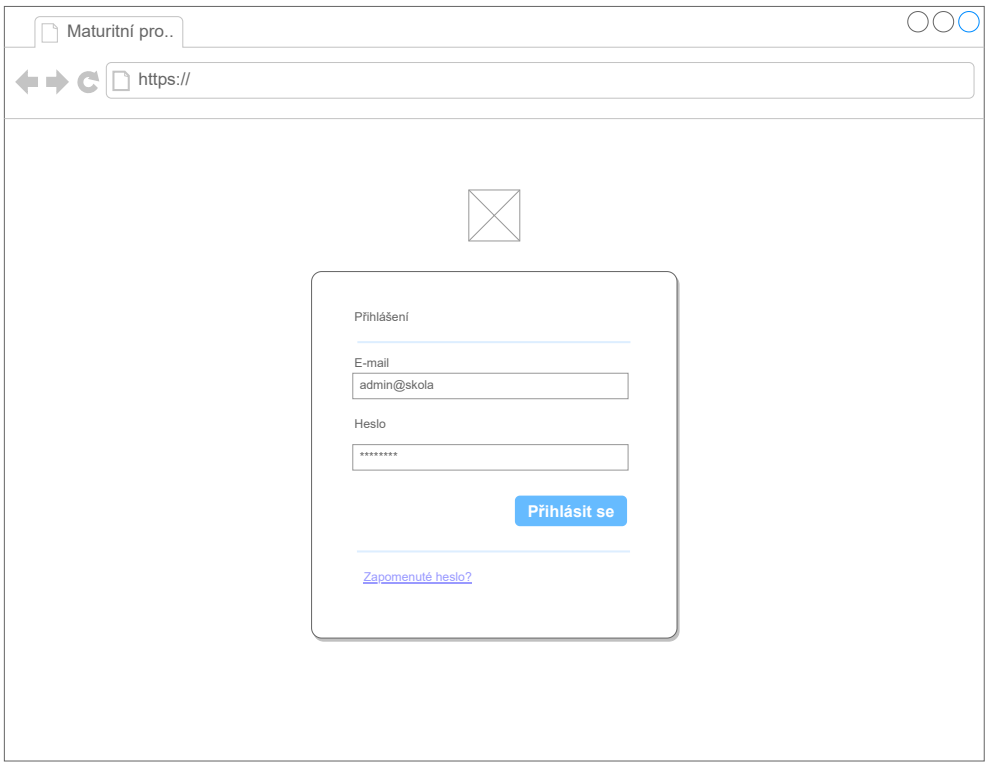
Uživatelské rozhraní bylo navrženo ve třech rozvrženích pro různé situace. Při návrhu se postupovalo přístupem desktop-first, jelikož se z provedené analýzy 1.3.2, podle NP2, očekává nejvyšší návštěvnost právě z desktopových zařízení. Navrženy byly také některé komponenty, které se používají v rámci celé aplikace.

2.2.1 Návrh rozvržení

První rozvržení, označené jako **guest layout**, obsahuje blok zarovnaný na střed obrazovky horizontálně i vertikálně. Nad něj lze umístit logo školy, případně název. Tento blok by měl být dostatečně odlišený od pozadí a měl by většinou obsahovat formulář, případně nějakou hlášku. Toto rozvržení se používá na obrazovkách, kde uživatel není autentizovaný:

- Přihlašovací stránka
- Stránka se zapomenutým heslem
- Stránka s resetováním hesla

Wireframe tohoto rozvržení si lze prohlédnout na obrázku 2.2 a výslednou implementovanou obrazovku na obrázku 2.3.



Obrázek 2.2 Wireframe pro layout přihlašovací stránky

Druhé rozložení, pojmenované jako **auth layout**, obsahuje horizontální menu, ve kterém se nachází logo školy a informace o aktuálně přihlášeném uživateli s rozbalovacím menu, ve kterém se nachází odkazy na práci s jeho profilem (změna hesla a odhlášení), případně ještě odkaz do administrace, pokud má uživatel patřičnou roli. Pod horizontálním menu může být umístěna

Obrázek 2.3 Přihlašovací stránka

další lišta s drobečkovou navigací, ta se používá na obrazovce se zobrazením konkrétního projektu. Tělo stránky pak bude obsahovat stejné bloky jako v guest layoutu. Ty mohou být umístěny při dostatečné šířce okna i vedle sebe. Případně lze použít jeden široký blok. Nakonec lze přidat patičku, která může obsahovat informace nebo tlačítka s akcemi. Toto rozvržení je použito na obrazovkách, kde je uživatel autentizovaný:

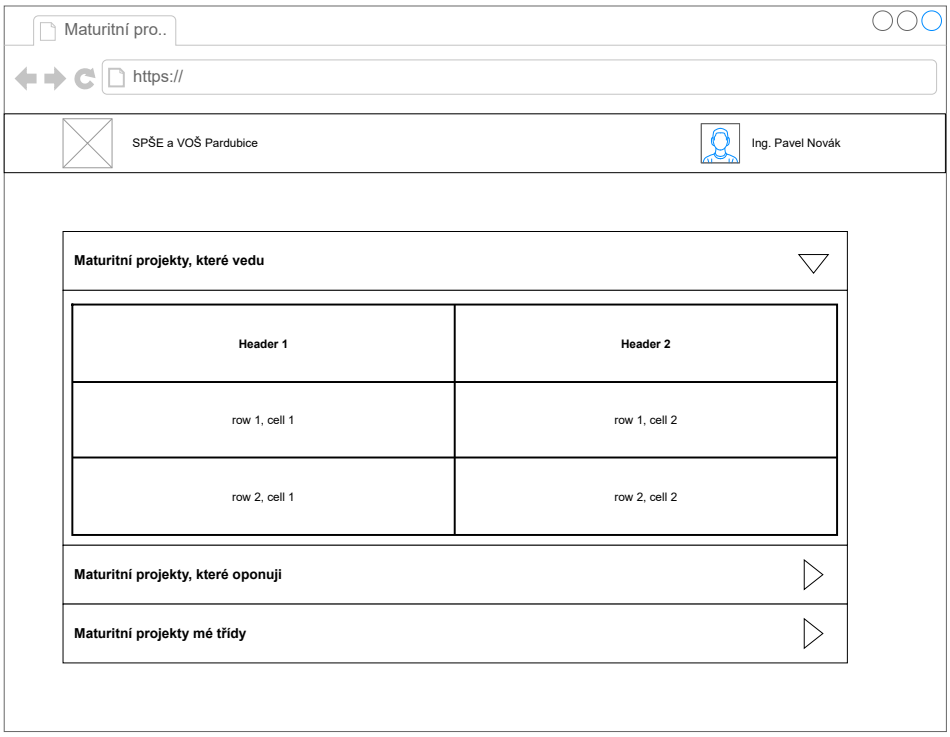
- Nástěnka
- Stránka konkrétního projektu

Nástěnka je stránka, na kterou je uživatel (kromě žáka) přesměrován z přihlašovací obrazovky po úspěšné autentizaci. Na této stránce se mají uživatelům zobrazit seznamy maturitních projektů, ke kterým mají přístup. Jelikož může uživatel nabývat více rolí a z každé role mu vyplývají jiné povinnosti[1], je důležité, aby uživatel věděl, jaký maturitní projekt s jakou rolí prohlíží. V minulém prototypu aplikace, který byl používán, toto bylo řešeno pomocí harmoniky (accordion), která měla rozbalovací sekce s tabulkami projektů pro každou roli (lze vidět na wireframe 2.4). Zde bylo pro dosažení lepšího UX navrženo přepínání rolí pomocí záložek. To zaručí, že uživatel uvidí své přidělené role pohromadě na jednom místě a tabulku s projekty může vždy očekávat na stejném místě. Wireframe nového návrhu lze nalézt na obrázku 2.5.

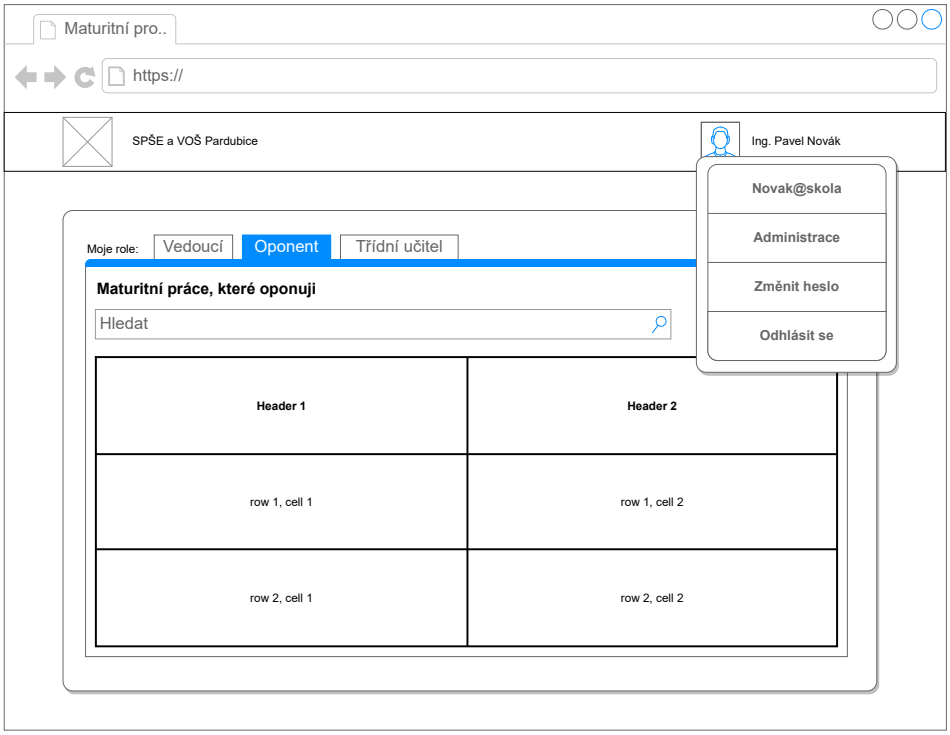
Na stránku konkrétního projektu se uživatel dostane kliknutím na řádek v tabulce projektů. Žákovi se tato stránka zobrazuje místo nástěnky popsané výše. Zde se využije již zmíněná patička s akcemi, která se posouvá při skrolování. Ta je zvolena, aby měl uživatel akci „Uložit“ stále na očích a nezapomněl na ni. Drátěný model lze vidět na obrázku 2.6.

Posledním navrženým rozložením je **admin layout**, který se používá v administraci. Místo horizontálního menu se zde používá postranní panel, který obsahuje logo školy, tlačíko „Zpět“ umožňující rychlou navigaci na nástěnku a vertikální navigační menu, které obsahuje prostředky, se kterými může administrátor pracovat. Vpravo od postranního panelu se opět nachází široký blok známý už z předchozích rozložení. Wireframe tohoto rozložení je zachycen na obrázku 2.7.

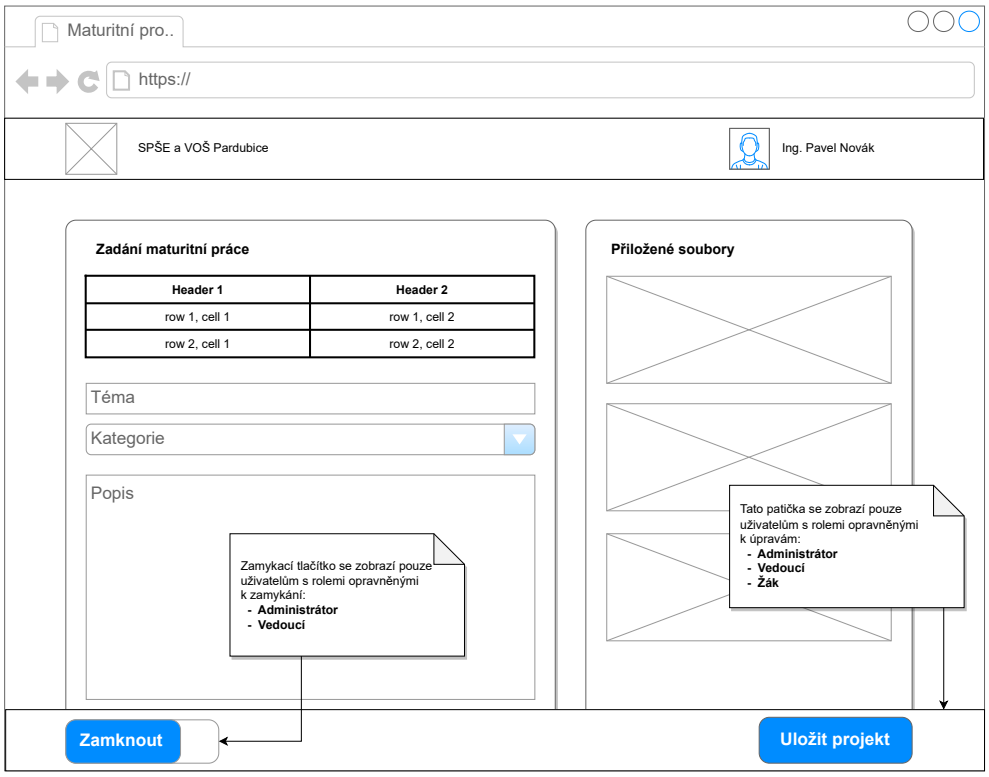
[1] Třídní učitel může být zároveň vedoucím a oponentem některých projektů. Jako vedoucí musí upravit zadání, zamknout projekt a nahrát posudek, zatímco jako oponent pouze nahrává posudek.



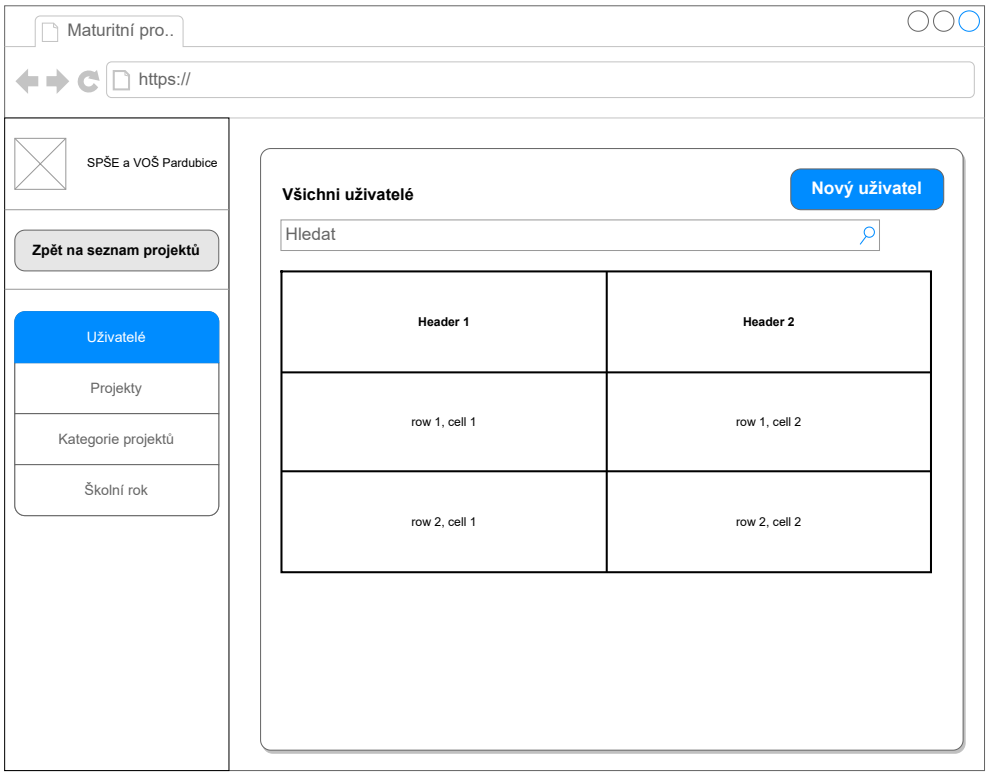
Obrázek 2.4 Wireframe starého návrhu nástěnky s harmonikou



Obrázek 2.5 Wireframe pro nástěnku se seznamem projektů (auth layout)



Obrázek 2.6 Wireframe pro obrazovku s konkrétním projektem



Obrázek 2.7 Wireframe pro rozvržení administace

2.2.2 Návrh některých komponent

2.2.2.1 Modální dialogové okno

Pro rychlé akce, kterými jsou např. úprava jména existujícího uživatele nebo třeba prodloužení deadline pro konkrétní projekt, by nemuselo být příliš vhodné přesměrovávat uživatele na novou stránku a zpět. Pro takové akce se bude v aplikaci používat modální dialogové okno, navržené ve wireframe 2.8.

2.2.2.2 Notifikační zprávy (toasts)

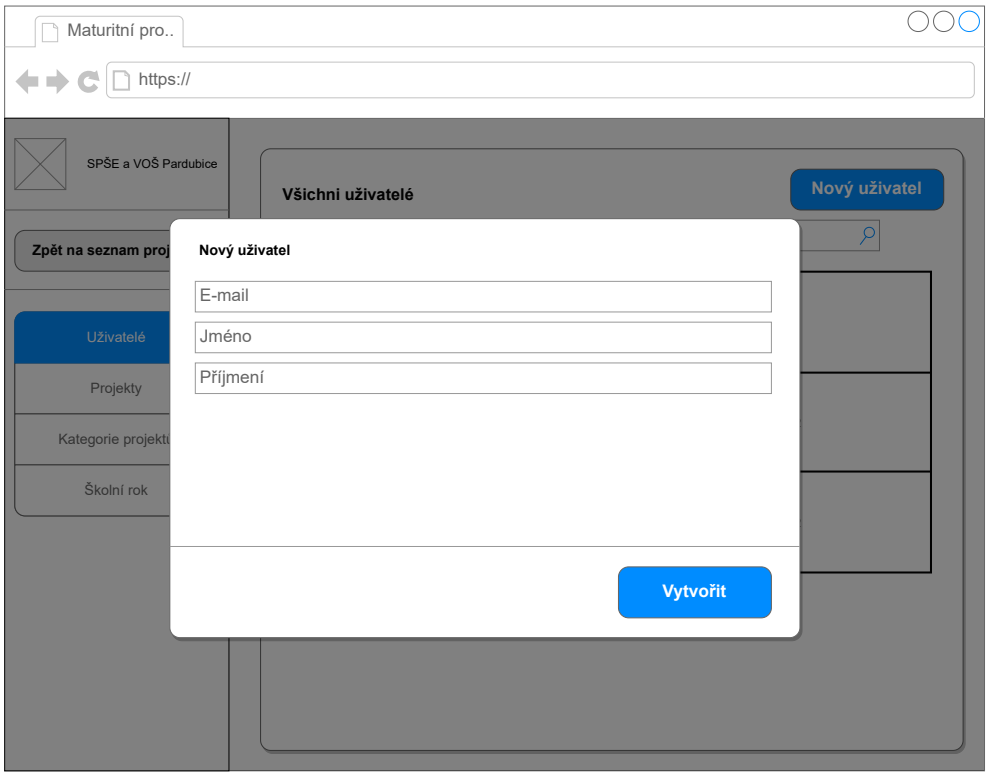
Z navržené architekury, kdy SPA aplikace na frontendu komunikuje s REST API na backendu, je potřeba uživatele informovat o výsledku prováděné operace, případně o dalších informacích, které REST API může poskytnout (např. důvod, proč akce nešla provést). Toho je docíleno zobrazováním notifikačních zpráv v pravém horním rohu. Uživatelé jsou dnes na toto umístění většinou navyklí, případně si rychle zvyknou. Výhodou je, že uživatel může očekávat notifikační hlášku vždy na stejném místě. Hlášky budou barevně odlišené podle výsledku operace, kdy zelená barva značí úspěch, naopak červená nebezpečí, v našem případě nezdar. Navíc každá notifikace bude obsahovat ikonu, která bude zdůrazňovat typ hlášky – fajfka, vykřičník, atd. Wireframe zachycující notifikační zprávy lze nalézt na obrázku 2.9.

2.3 Návrh bezpečnostního modelu

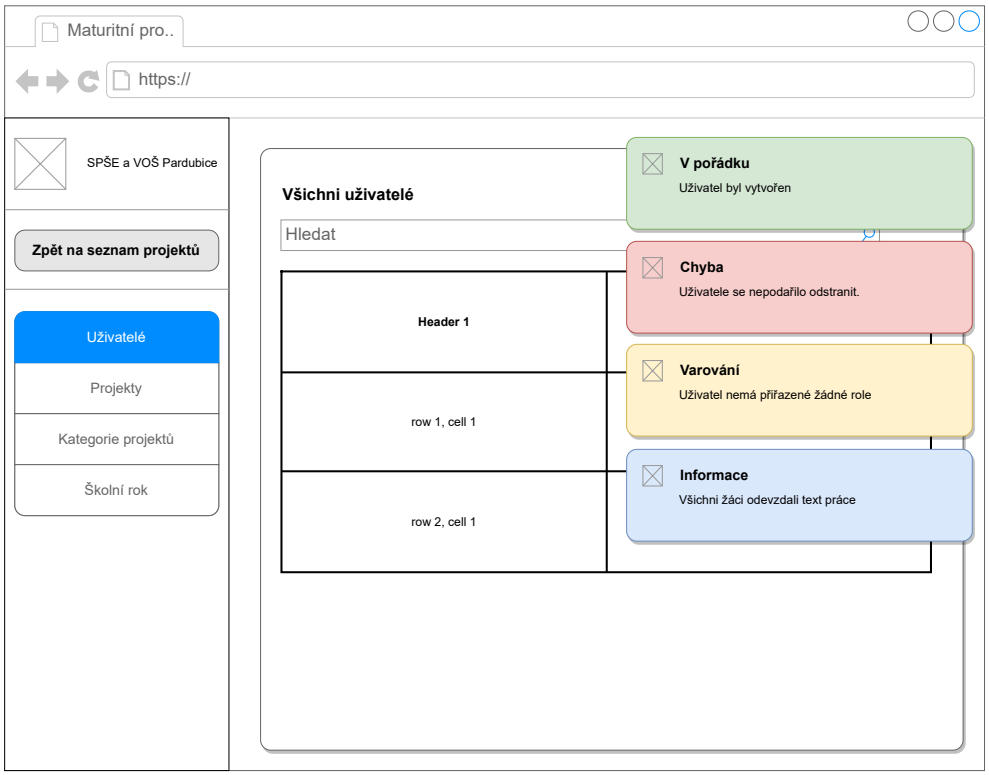
2.3.1 Prostředky

Aplikace pracuje s následujícími prostředky, ze kterých vyplývají bezpečnostní požadavky.

- Projekt
  - Akce, které lze nad projektem provádět:
    - Zobrazení – získání informací ohledně projektu – název, kategorie, vedoucí, ...
    - Úprava – úprava zadání projektu – název, kategorie, popis, specifikace
    - Nahrání souborů
    - Zamknutí pro editaci
- Soubory projektu
  - Typy souborů:
    - Oskenované zadání
    - Text práce
    - Data práce
    - Posudek vedoucího
    - Posudek oponenta
- Deadline
  - Dělení podle:
    - Role – student vs. vedoucí a oponent
    - Rozmezí – globální pro všechny projekty vs. konkrétní pro jeden projekt (prodloužení odevzdání)
- Uživatelé
- Uživatelské role



Obrázek 2.8 Wireframe pro modální dialogové okno



Obrázek 2.9 Wireframe pro notifikační zprávy

2.3.2 Bezpečnostní požadavky

Bezpečnostní požadavky plynou z požadavků, které SPŠE a VOŠ Pardubice na odevzdávací aplikaci má, s přihlédnutím k „Best Practices“ (doporučeným postupům) např. u používání hesel.

1. Autorizace uživatelů pomocí uživatelských rolí (přehled lze nalézt v tabulce 2.1):
   - **Student** může prohlížet, upravovat a nahrávat soubory typu `text projektu` a `data projektu` pouze ke svému maturitnímu projektu.
   - **Třídní učitel** může pouze prohlížet projekty své třídy, tyto projekty nemůže upravovat, ani k nim nahrávat soubory (pokud není zároveň vedoucím či oponentem projektu).
   - **Předseda** a **místopředseda** maturitní komise mohou prohlížet pouze projekty tříd, kde předsedají.
   - **Oponent** může prohlížet a nahrávat soubory typu `posudek oponenta` k maturitní projektům, které oponuje.
   - **Vedoucí** může prohlížet, upravovat a nahrávat soubory typu `posudek vedoucího` k maturitním projektům, které vede. Dále může tento projekt zamknout pro úpravy.
   - **Administrátor** může prohlížet, upravovat a nahrávat soubory typu `oskenované zadání` ke všem maturitním projektům. Tyto projekty může definitivně zamknout pro úpravy.
2. Uživatel nesmí manipulovat (zobrazení, upravení, nahrání souborů) s projektem, ke kterému nemá přístup se svojí uživatelskou rolí, např.:
   - Student A není přiřazen k projektu studenta B, proto nemůže s jeho projektem manipulovat.
   - Třídní učitel třídy 4.C nemůže zobrazit projekt studenta B, který patří do třídy 4.D.
3. Uživatel nesmí upravit projekt, který je zamčený proti úpravám.
   - Př.: Vedoucí zamkne projekt vůči úpravám, jelikož je zadání připravené ke kontrole administrátorem. Student ani vedoucí, kteří by za normálních okolností měli práva projekt upravovat, nyní tak nemohou učinit.
4. Uživatel může k projektu nahrávat soubory pouze v definovaném deadline.
   - Př.: Globální deadline pro odevzdávání studentova textu práce je `2023-05-07 23:59:59` a prodloužení není nastaveno. Pokud by se student snažil nahrát soubor např. 10. 5. 2023, API na backendu mu neumožní soubor nahrát.
5. Deadline pro odevzdávání souborů může měnit pouze administrátor.
6. Nahrání souboru uživatelem bude nezpochybnitelné, tj. bude zaručeno:
   - integrita souboru – lze ověřit, zda se data souboru od nahrání uživatelem nezměnila
   - časová nepopiratelnost – lze ověřit, že data souboru existovala v konkrétním čase
7. Přihlašování pomocí hesla bude vycházet z doporučení NIST SP 800-63B (Digital Identity Guidelines). [18]
   - Minimální délka hesla je 8 znaků.
   - Budou povoleny všechny tisknutelné znaky.
   - Nebude povolena nápověda k heslu, ani bezpečnostní otázky.
   - Bez požadavků na složitost hesla – např. nutnost alespoň 1 speciálního znaku, apod.

- Heslo bude v databázi uložené pouze jako otisk (hash) pomocí algoritmu bcrypt.
- Atd.

8. Veškerá komunikace mezi frontendem a backend API bude probíhat přes šifrovaný protokol HTTPS.

Tabulka 2.1 Přehled autorizace rolí pro daný projekt

Role	Zobrazení	Úprava	Nahrání souborů	Zamknutí
Student	Ano	Ano	Text a data práce	Ne
Třídní učitel	Ano	Ne	Ne	Ne
Předseda/místopředseda	Ano	Ne	Ne	Ne
Oponent	Ano	Ne	Posudek oponenta	Ne
Vedoucí	Ano	Ano	Posudek vedoucího	Ano
Administrátor	Ano	Ano	Oskenované zadání	Ano, definitivně

2.3.2.1 Nezpochybnitelné odevzdání souborů

Pro nezpochybnitelné odevzdání souborů je potřeba zaručit integritu a časovou nepopiratelnost. Tyto vlastnosti pomůže zaručit digitálně podepsané časové razítko. K získání takového razítka je využitý TSA (Time Stamping Authority) server, který používá TSP (Time-Stamp Protocol) popsaný v RFC 3161 [19].

Na TSA server se odešle hash souboru pomocí `openssl-ts` Time Stamping Authority tool [20] jako TSQ soubor (Time Stamp Query). Ze serveru se vrátí odpověď ve formě TSR souboru (Time Stamp Response). Oba soubory TSQ i TSR využívají binární formát DER (Distinguished Encoding Rules), jak je popsáno v RFC 3161 [19]:

```
3.4. Time-Stamp Protocol via HTTP
[...]
Two MIME objects are specified as follows.

Content-Type: application/timestamp-query
   <<the ASN.1 DER-encoded Time-Stamp Request message>>

Content-Type: application/timestamp-reply
   <<the ASN.1 DER-encoded Time-Stamp Response message>>
```

Celý postup manipulace s digitální časovým razítkem pak vypadá následovně:

- Získání podepsaného časového razítka (ukázka odpovídajících příkazů na výpisu kódu 1)
  1. Vygenerování TSQ (Time Stamp Query) pro daný soubor (spočítá hash a uloží do DER formátu)
  2. Odeslání TSQ na TSA server a uloží si odpověď do Time Stamp Response (DER formát)
- Ověření podepsaného časového razítka (ukázka odpovídajících příkazů na výpisu kódu 2)
  1. Stažení certifikačního řetězu CA (certifikační autority)
  2. Ověření podepsaného razítka s využitím certifikačního řetězu

Existuje velké množství TSA serverů, ze kterých lze vybírat. Některé jsou placené, existují však i servery, které službu poskytují zdarma. Jednou možností, kde se dají takové servery hledat, je *Přehled kvalifikovaných poskytovatelů certifikačních služeb a jejich kvalifikovaných služeb* [21],

```
# Krok 1 - Vygenerování žádosti do 'zadost.tsq' pro soubor 'text_prace.pdf'
openssl ts -query -data text_prace.pdf -no_nonce -cert -out zadost.tsq

# Krok 2 - Odeslání žádosti 'zadost.tsq' na TSA server
#   odpověď se uloží do 'podpis.tsr'
curl -H 'Content-Type: application/timestamp-query' \
    --data-binary @'zadost.tsq' \
    https://tsa.cesnet.cz:5817/tsa -o podpis.tsr
```

Výpis kódu 1 Získání podepsaného časového razítka pomocí OpenSSL

```
# Krok 1 - Stažení certifikačního řetězu do 'certifikacni_retez.pem'
curl https://crt.cesnet-ca.cz/CESNET_CA_Root.pem -o CESNET_CA_Root.pem
curl https://crt.cesnet-ca.cz/PersonalSigning2.pem -o PersonalSigning2.pem
cat CESNET_CA_Root.pem PersonalSigning2.pem > certifikacni_retez.pem

# Krok 2 - Ověření razítka 'podpis.tsq' pro soubor 'text_prace.pdf'
#  pomocí 'certifikacni_retez.pem'
openssl ts -verify \
    -data text_prace.pdf \
    -in ./podpis.tsr \
    -CAfile certifikacni_retez.pem

# Na standardní výstup se v případu úspěchu vypíše:
Verification: OK

# Pokud by data byla pozměněná a nepodařilo by se ověřit podpis,
#   dostali bychom následující chybu:
Verification: FAILED
```

Výpis kódu 2 Ověření podepsaného časového razítka pomocí OpenSSL

který podle zákona[2] zveřejňuje Ministerstvo vnitra České republiky. V současné době se na tomto seznamu vyskytují tři poskytovatelé, kteří nabízí službu *Vydávání kvalifikovaných časových razítek*:

1. První certifikační autorita, a. s. (`https://www.ica.cz/`)
2. Česká pošta, s. p. (`http://qca.postsignum.cz/`)
3. eIdentity a. s. (`http://www.eidentity.cz/`)

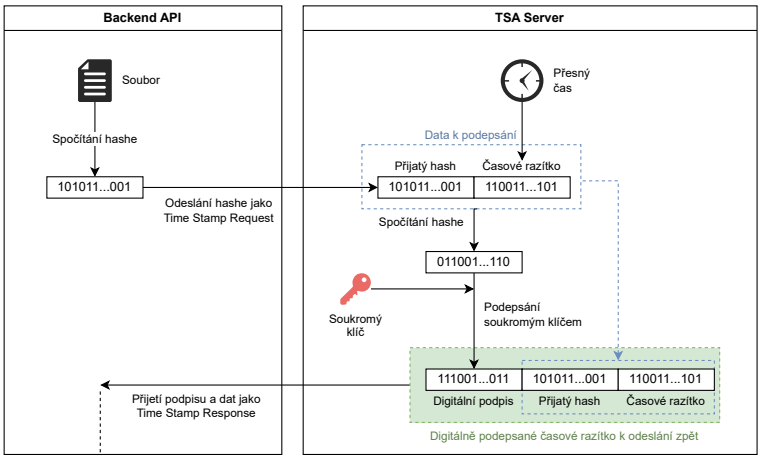
Veřejné servery, které službu poskytují zdarma, jsou např.:

1. CESNET CA (`https://pki.cesnet.cz/cs/tsa-overview`)
2. freeTSA.org (`https://freetsa.org/`)

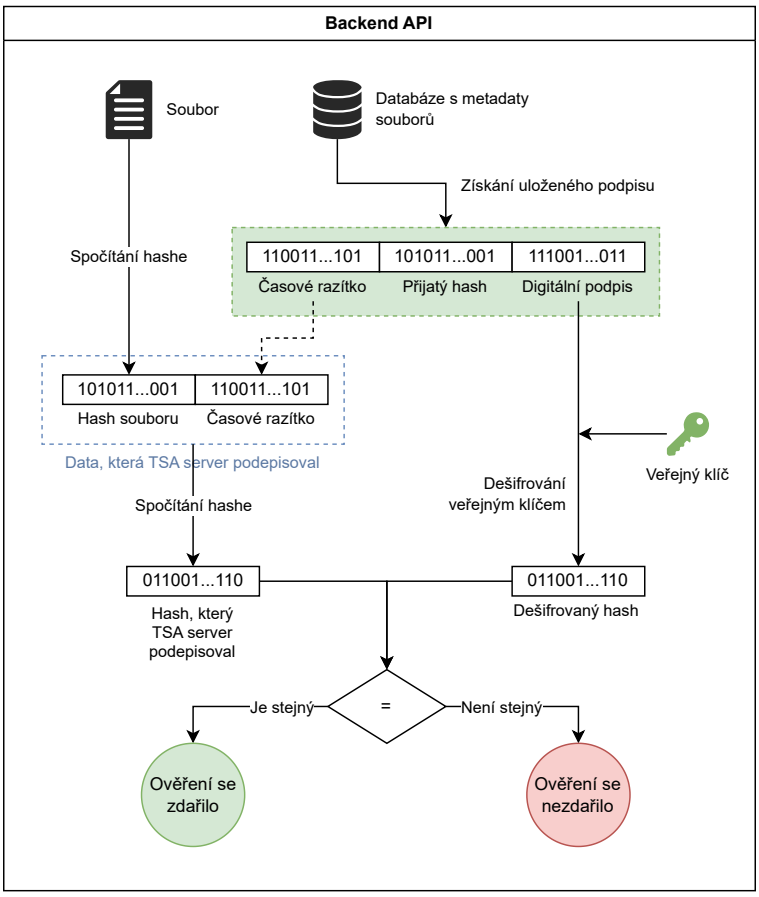
Pro potřeby této bakalářské práce je využit server od CESNET CA. Tuto službu poskytuje sdružení CESNET bezplatně pro výzkumné a vzdělávací instituce. [22]

[2] Ust. § 9 odst. 2, písm. e) zákona č. 227/2000 Sb., o elektronickém podpisu a o změně některých dalších zákonů (zákon o elektronickém podpisu)

Jak TSA server využívá asymetrickou kryptografii pro vytvoření podepsaného časového razítka, je zachyceno na schématu 2.10 a využití kryptografie pro jeho ověření se nachází na schématu 2.11.



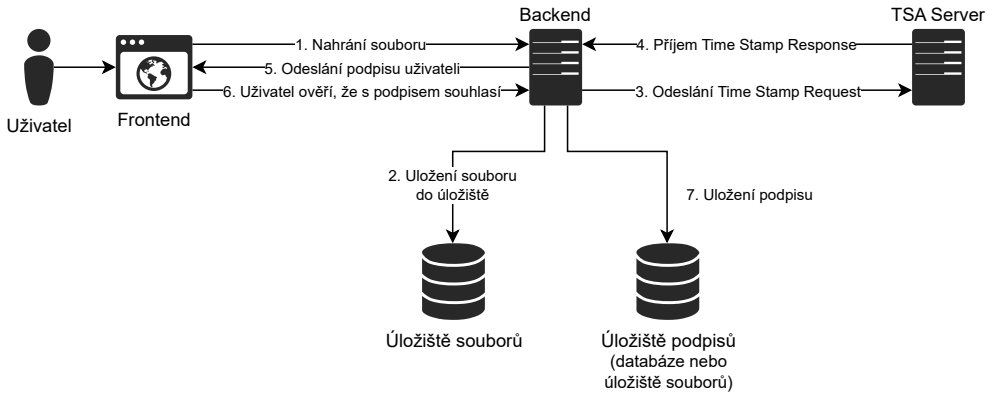
Obrázek 2.10 Vytvoření podepsaného časového razítka



Obrázek 2.11 Ověření podepsaného časového razítka

Celý případ užití nahrání souboru na backend (zakreslený na diagramu 2.12) se získáním časového razítka probíhá následovně:

1. Uživatel nahraje soubor na backend.
2. Backend uloží soubor do úložiště souborů.
3. Backend odešle Time Stamp Request na TSA server.
4. TSA Server vrátí podepsané časové razítko v Time Stamp Response.
5. Backend odešle uživateli na frontend podepsané časové razítko s dalšími metadaty (velikost souboru, hash souboru, datum a čas nahrání).
   - Tj. z důvodu, aby měl i uživatel ve svém držení důkaz, že určitá data existovala v konkrétním čase.
6. Uživatel potvrdí, že souhlasí s podpisem a metadata odpovídají.
7. Backend uloží podpis do úložiště.
   - Úložištěm pro podpis může být databázový server nebo i souborový server.
8. Nahrávání je úspěšně dokončeno.



Obrázek 2.12 Případ užití – nahrání souboru na backend

2.3.3 Identifikace hrozeb

Identifikace hrozeb (Threat Modeling) je důležitým krokem při návrhu bezpečnostního modelu.

Slouží k rozpoznání a pochopení hrozeb a jejich zmírnění v kontextu ochrany cenných dat a prostředků. V podstatě se jedná o pohled na aplikaci a její prostředí optikou bezpečnosti [23].

2.3.3.1 Útoky na autentizační mechanismus – heslo

Vzhledem k tomu, že se k autentizaci bude používat přihlašovací jméno a heslo, musí být zváženy následující hrozby.

1. Brute-force útok
   - zkoušení všech kombinací z celého prostoru hesel

2. Lidský faktor
   - uživatel může heslo prozradit jiné osobě
   - útočník může uhádnout uživatelovo heslo

Pokud by útočník získal nějakým způsobem heslo uživatele, mohl by se autentizovat pod jeho identitou. Autorizovaný by byl podle role uživatele. Například pro roli `student` by útočník mohl: změnit uživatelovo heslo, zobrazit i upravit zadání studentova maturitního projektu, zobrazit soubory nahrané k projektu a nahrát soubory typu `text práce` i `data práce`.

2.3.3.1.1 Mitigace útoku na heslo Za předpokladu, že by útočník zkoušel nejkratší, 8místné kombinace (viz bezpečnostní požadavky: minimální délka hesla) sestavené pouze z malých písmen anglické abecedy (`a-z`, 26 znaků), celý prostor hesel by měl $26^8 \approx 208,8$ miliard prvků. V případě malých a velkých znaků abecedy (`a-zA-Z`, 52 znaků) se prostor rozšiřuje na $52^8 \approx 53\,460$ miliard prvků. Při online[3] brute-force útoku se musí také započítat doba přenosu dat po síti, která na takovém prostoru hesel není zanedbatelná.

Další okolností, která je účinná i při offline útoku, je použití hashovací funkce `bcrypt`, která je navržena, aby výpočet hashe byl pomalejší a ztěžoval tak možný útok.

Dále lze využít Rate Limiting, který například povolí pouze 10 pokusů o přihlášení za 1 minutu. Ten by ovšem při slabém nastavení mohl útočník využít k uzamčení účtu, a tudíž k odepření služby tomuto uživateli.

Zmírnit útok na lidský faktor lze pomocí vícefaktorového ověření, kdy se kromě faktoru znalosti (v tomto případě heslo) vyžaduje navíc faktor vlastnictví, kterým by zde mohl být mobilní telefon s autentizační aplikací (např. Google Authenticator nebo Microsoft Authenticator), která generuje jednorázové kódy pomocí One-time Password algorithm (OTP) typu Time-based (TOTP) a HMAC-based (HOTP) [24]. V této bakalářské práci nebude vícefaktorové ověření implementováno, jelikož by to přesahovalo rozsah bakalářské práce.

2.3.3.2 Útoky na nahrávání souborů

Nahrávání souborů je jednou ze stěžejních části této aplikace. Zároveň však existují různé útoky, které mohou být provedeny, pokud backend není správně naprogramován. Tyto útoky, ale také postup, jak se jim vyvarovat, popisuje OWASP (Open Worldwide Application Security Project), nezisková organizace pracující na zvýšení bezpečnosti software [25], ve své příručce OWASP Cheat Sheet Series, v kapitole File Upload Cheat Sheet [26]. Z útoků, které jsou popsány v této příručce, jsou zde vybrány ty, které mají větší pravděpodobnost, že by mohly nastat.

2.3.3.2.1 Nahrání ZIP bomby ZIP bomba je speciálně vytvořený archiv s velkým kompresním poměrem. Jedním takovým příkladem je archiv, který má velikost 10 MiB, ale při extrahování se rozbalí do velikosti 281 TiB. Dalším příkladem je archiv o velikosti 46 MiB rozbalený do velikosti 4,5 PiB [27].

Pokud by backend nechal extrahovat takový archiv, mohlo by dojít k zaplnění kapacity úložiště souborů, což by mohlo vést k odepření služby (DoS – Denial of Service).

Mitigace nahrání ZIP bomby Vzhledem k tomu, že server soubory pouze ukládá a poté je při žádosti vrací ke stažení a k rozbalování archivu na straně serveru nedojde, aplikaci to neohrozí.

[3] Útočník zkouší hesla přes přihlašovací formulář, nemá přístup k hashům uloženým v databázi. Kdyby měl útočník k dispozici hash hesla, mohl by na něj útočit pomocí offline brute-force útoku.

2.3.3.2.2 Path Traversal Známý také jako Directory Traversal[4] je útok, který se snaží pomocí úpravy názvu souboru donutit server získat soubor z jiné složky na disku, než bylo původně programátorem zamýšleno. Případně do této složky soubor i uložit. Do názvu souboru útočník typicky přidává sekvenci znaků `../` (i několikrát za sebou), která při nesprávném ošetření vede k přesunu do nadřazené složky. Pokud útočník zná, nebo dokáže odhadnout architekturu webové aplikace, mohl by získat např. konfigurační soubory, zdrojové kódy nebo data ostatních uživatelů. Tyto soubory by mohl i přepsat.

Mitigace útoku Path Traversal Při nahrávání souboru aplikace nebere v potaz název souboru poskytnutý uživatelem. Místo toho generuje unikátní identifikátor (UUID), který se připojí k cestě, kam se ukládají soubory. Tím se soubor uloží na serveru správně, nezávisle na vstupu od uživatele.

2.3.3.2.3 Odepření služby (DoS) Velkým počtem požadavků na stažení většího souboru (např. v řádu GiB), může dojít k vyčerpání prostředků serveru a odepření služby.

Při nahrávání souboru na server by ke stejné situaci mohlo dojít při velkém počtu požadavků na nahrání velkého souboru. Případně při nahrání dostatečně velkého souboru, který by zaplnil zbývající volnou kapacitu disku, aby ostatní uživatelé nemohli nahrát své soubory.

Mitigace odepření služby Aby nedošlo ke zahlcení serveru způsobenému velkým počtem požadavků na přenos velkých souborů (směrem ze serveru i směrem na server), můžeme zde aplikovat Rate Limiting, který omezí takové požadavky např. na 5 požadavků za minutu. Samozřejmostí zde je, že se soubory mohou pracovat pouze autentizovaní uživatelé.

Backend kontroluje velikost nahrávaných souborů a v požadavcích jsou určené maximální velikosti souborů podle jejich typu. Pro posudky, oskenované zadání a text práce je povolena velikost 100 MiB. Největším souborem, který může uživatel nahrát, jsou `data projektu`, která nahrává pouze uživatel role student a mohou dosahovat velikosti až 10 GiB. Uživatel může nahrávat soubory neomezeně mnohokrát v rozmezí stanoveného deadline a pouze poslední odevzdání je platné. Proto, aby nedošlo k rychlému zaplnění disku, server předchozí nahrané verze maže.

[4] V překladu: procházení složek.

Kapitola 3

Implementace

3.1 Zvolené technologie

3.1.1 Použité technologie na frontendu

Jak bylo popsáno v kapitole návrhu, pro frontend byl zvolen přístup SPA aplikace. Pro vývoj takových aplikací se dnes často používá některý z dostupných JS frameworků (JavaScript). Vybrané frameworky, které jsou dnes populární, jsou zobrazeny v tabulce 3.1.

Pro vývoj frontendové části byl zvolen framework React, jelikož má přehlednou a obsáhlou dokumentaci, velkou komunitu a navíc v něm autor této bakalářské práce již programoval několik webových aplikací.

Tabulka 3.1 Přehled GitHub repozitářů JS frameworků (k 6. květnu 2023)

Název	GitHub repozitář	Stars	Forks	Watchers	Rep. vytvořen
React	facebook/react	206 935	43 175	6 629	2013-05-24
Vue 3	vuejs/core	37 296	6 826	753	2018-01-12
Vue 2	vuejs/vue	203 458	33 670	5 993	2013-07-29
Angular	angular/angular	87 986	23 528	3 037	2014-07-18
Svelte	sveltejs/svelte	67 429	3 306	868	2016-10-20
Ember.js	emberjs/ember.js	22 452	4 297	876	2011-05-26

Instalační příručku k frontendové aplikaci lze nalézt v příloze A.

3.1.1.1 React

React je open-source knihovna pro tvorbu webových a nativních uživatelských rozhraní. [28] UI by se mělo v Reactu skládat z menších znovupoužitelných částí kódu, které se nazývají „komponenty“.

Pro zápis komponent se zde využívá JSX (JavaScript Syntax Extension) syntaxe, kterou React zpopularizoval. Ta umožňuje psaní kódu podobného HTML (HyperText Markup Language) v rámci JavaScriptu. JSX kód se poté pod pokličkou převádí na JS objekty. [29]

Jak může vypadat komponenta v Reactu, zachycuje ukázka kódu 3.

```
// Export komponenty 'Avatar'
export default function Avatar(props) {
    // Logika komponenty
    const onClick = () => { ... };

    // Zde je využitá syntaxe JSX
    return (
        <>
            <img src={props.image} alt={props.name} />
            <span>{props.name}</span>
        </>
    );
}
```

Výpis kódu 3 Ukázka React komponenty

3.1.1.2 Bootstrap, React-Bootstrap

Bootstrap je rozšířený CSS (Cascading Style Sheets) framework, který obsahuje komponenty jako např. tlačítka, modální dialogové okna, otevírací menu (dropdown) a další. [30]

Všechny tyto komponenty se dají dále přizpůsobovat. Díky tomu, že Bootstrap používá Sass, lze většinu vzhledových úprav komponent provést přes změnu určité proměnné v kódu.

React-Bootstrap je knihovna, která zapouzdřuje Bootstrap komponenty v React komponentách, což usnadňuje jejich použití v React projektu.

Tento framework byl zvolen pro urychlení vývoje uživatelského rozhraní frontendové části aplikace.

3.1.1.3 Sass

Sass (Syntactically Awesome Stylesheet) je CSS preprocesor, který rozšiřuje CSS o proměnné, zanořování (nesting), dělení kódu do samostatných souborů (modulů), matematické operátory nebo např. o mixins (více CSS deklarací sloučeno do jedné[1]). [31] Tento preprocesor umožňuje zápis kódu ve dvou syntaxích – SCSS (Sassy CSS) a Sass, pro potřeby této práce je použita syntaxe SCSS. Jelikož se jedná o preprocesor, je potřeba SCSS kód zkompilovat do CSS. [32] O to se v tomto projektu stará balíček `sass`.

3.1.1.4 Npm

Npm je správce JS balíčků pro platformu Node.js, ale i pro frontendové balíčky. Informace o projektem vyžadovaných balíčcích uchovává v souboru `package.json`, instalované balíčky poté lokálně u projektu ve složce `node_modules`, případně globálně ve stejnojmenné složce umístěné na disku v závislosti na operačním systému a nastavení. [33]

V této práci je využitý pro správu a instalaci frontendových JS balíčků.

3.1.1.5 SWR

SWR je knihovna pro React obsahující hooks (speciální funkce) pro načítání dat z API. Název SWR označuje `stale-while-revalidate`, což je HTTP kešovací strategie. Tato strategie nejprve vrátí data z mezipaměti, poté odešle požadavek na získání dat z API a nakonec vrátí

[1] To umožňuje jednoduché znovupoužití, navíc lze do mixin předávat argumenty, na kterých jsou závislé. Tato vlastnost by se dala přirovnat funkcím v programovacích jazycích.

aktuální data. Díky tomu získávají komponenty konstantně a automaticky data a UI bude rychlé a reaktivní. [34]

Tato knihovna je v projektu využita pro získávání veškerých dat z API, se kterými je potřeba na frontendu pracovat – získání dat o uživateli, maturitním projektu, souborech přiřazených k projektu, atd.

3.1.1.6 Axios

Axios je často využívaný HTTP klient, tzn., že umí odesílat HTTP požadavky a zpracovávat odpovědi na ně. Pro odesílání požadavků používá `XMLHttpRequests`. [35] Mezi jeho vlastnosti, které tato aplikace využívá, patří:

- Podpora Promise API pro asynchronní kód
- Zrušení odeslaných požadavků dříve, než přijde odpověď
- Automatický převod těla požadavku na formáty:
  - JSON (`application/json`)
  - Multipart/FormData (`multipart/form-data`)
  - URL encoded form (`application/x-www-form-urlencoded`)
- Automatické zpracování JSON formátu v odpovědi
- Podpora pro ochranu proti CSRF (Cross Site Request Forgery)

3.1.1.7 Quill, ReactQuill

ReactQuill poskytuje wrapper okolo open-source textového editoru Quill. [36] Ten umožňuje formátování textu, vkládání seznamů, odkazů, obrázků a je dále rozšiřitelný. [37]

V tomto projektu je využitý na následujících místech:

- `SingleProjectPage` – stránka se zobrazením konkrétního projektu, zde se dvakrát předává do komponenty pro generování formuláře `GeneralForm` jako pole:
  - Téma maturitní práce (popis)
  - Hlavní body – specifikace maturitní práce
- `AdminAssignmentText` – stránka v administraci pro úpravu obecného textu zadání, který je pak zobrazen u všech projektů (v komponentě `SingleProjectPage`).

3.1.1.8 Dompurify

Dompurify je knihovna, která pomáhá ošetřit HTML a předchází tak XSS (Cross Site Scripting) útokům.[38]

Přestože React automaticky ošetřuje vstupy a zabraňuje tím XSS útokům ve výchozím stavu, lze ho donutit pomocí vlastnosti `dangerouslySetInnerHTML`, aby předané HTML vyrenderoval. Renderování HTML bylo potřeba docílit v komponentě `AssignmentAdditionalText`, která načítá z API obecný text zadání zobrazený u všech maturitních projektů. Tento text obsahuje nadpisy, seznamy a formátování textu, proto je třeba patřičné HTML tagy zachovat. Obecný text zadání sice může upravit v administraci pouze administrátor, přesto je třeba tento vstup ošetřit pro ochranu všech uživatelů.

```
// Zjednodušená ukázka bez načítacího skeletonu
const {data: text, isLoading} = useFetchAssignmentText();

return (
    <div dangerouslySetInnerHTML={{
        __html: DOMPurify.sanitize(text?.text, {
            ALLOWED_TAGS: ['p', 'strong', 'b', 'em', 'i', 'ul', 'ol', 'li'],
        }),
    }}/>
);
```

Výpis kódu 4 Ošetření HTML vstupu pomocí Dompurify

3.1.1.9 FilePond

FilePond je knihovna, která umožňuje uživatelsky příjemnější nahrávání souborů. Její jádro je napsané v čistém JavaScriptu a jsou poskytnuty wrappery pro různé JS frameworky, včetně Reactu. [39]

Pro tento projekt důležitou funkcionalitou, kterou knihovna poskytuje, je nahrávání souboru po částech. Tento problém je detailněji popsán v kapitole 3.3.

3.1.2 Adresářová struktura frontendu

Frontendová část aplikace obsahuje následující adresáře a soubory. Pouze složky `node_modules`, resp. `build` jsou vytvořeny až po zavolání odpovídajících příkazů nástroje `npm` — `npm install`, resp. `npm run build`.

- `build` ... adresář obsahující vygenerovanou aplikaci pro produkční nasazení
- `node_modules` ... adresář obsahující externí balíčky (vytváří nástroj `npm`)
- `public` ... adresář obsahující soubory, ze kterých se poté generuje `build`
  - `favicon.ico` ... ikona webu
  - `index.html` ... generuje se z něj `build`; lze v něm např. změnit `title`
  - `robots.txt` ... nastavení pravidel pro crawlery (roboty indexující weby)
- `src`
  - `api` ... adresář obsahující skripty pro práci s API
  - `assets` ... adresář obsahující obrázky
  - `components` ... adresář obsahující React komponenty
  - `helpers` ... adresář obsahující pomocné skripty s konstantami a funkcemi
  - `hooks` ... adresář obsahující skripty s definovanými hooks
  - `routes` ... adresář obsahující komponenty se stránkami použitými v routeru
  - `styles` ... adresář obsahující Sass styly
  - `index.js` ... hlavní skript spouštějící webovou aplikaci
  - `reportWebVitals.js`
- `.env` ... soubor s proměnnými prostředí
- `.env.local` ... soubor s proměnnými prostředí pro lokální vývoj
- `license.txt` ... text licence
- `package.json` ... správa závislostí pro nástroj `npm`
- `README.md` ... instalační příručka

3.1.3 Použité technologie na backendu

V kapitole návrhu byl backend navržen jako REST API, které bude poskytovat data. Pro tento úkol lze vybírat z většího množství programovacích jazyků a frameworků, které jsou pro tyto jazyky k dispozici. Nabízí se např. jazyk Java s použitím frameworku Spring Boot, Python a frameworky Flask nebo Django REST. Dále pak JavaScript s použitím Node.js a frameworku Express.js nebo i jazyk Ruby s využitím Ruby on Rails. V neposlední řadě může být zvoleno PHP (Hypertext Preprocessor), které je od svého počátku vyvíjené směrem na webové aplikace, a samozřejmě pro něj také nalezneme frameworky, které ušetří práci s vývojem.

Pro účely této bakalářské práce byl zvolen jazyk PHP s frameworkem Laravel, jelikož s těmito technologiemi má autor nejvíce zkušeností v porovnání s uvedenými možnostmi. Navíc alternativní frameworky nepřináší žádné zásadní výhody, které by Laravel nenabízel. Navíc má Laravel aktivní, širokou komunitu a rozsáhlý ekosystém nástrojů okolo Laravelu. Populární PHP frameworky, ze kterých bylo vybíráno, jsou uvedeny v tabulce 3.2 se statistikami získanými z repozitářů na serveru GitHub.

Tabulka 3.2 Přehled GitHub repozitářů PHP frameworků (k 7. květnu 2023)

Název	GitHub repozitář	Stars	Forks	Watchers	Rep. vytvořen
Laravel	laravel/laravel	73 338	22 858	4 471	2011-06-08
Symfony	symfony/symfony	28 270	8 829	1 140	2010-01-04
Nette	nette/nette	1 486	237	151	2009-08-13
CakePHP	cakephp/cakephp	8 629	3 349	558	2010-05-08
CodeIgniter	codeigniter4/CodeIgniter4	4 673	1 740	273	2015-08-27

Instalační příručku k backendové aplikaci lze nalézt v příloze B.

3.1.3.1 Laravel

Laravel je framework pro webové aplikace s elegantní syntaxí. [40] Při vývoji usnadňuje běžné úkoly, které se řeší ve většině webových projektů, jako např.:

- Směrování požadavků (routing)
- Předávání závislostí (dependency injection)
- Objektově relační mapování databáze (ORM)
- Definování a migrace databázových schémat (database schema migrations)

Laravel je mocným nástrojem, který umožňuje vytvářet velké, robustní aplikace.[41]

Jeho popularitu dokazuje fakt, že z repozitáře pro PHP balíčky Packagist byl stažen více jak 251milionkrát. Pro srovnání: framework Symfony má takových stažení „pouze“ přes 74 milionů.

Při vývoji této aplikace byly využity tři rozšíření z jeho ekosystému:

- Laravel Breeze – kostra pro vytváření nové aplikace
- Laravel Sanctum – autentizační řešení pro SPA aplikace
- Laravel Telescope – ladicí prostředí zobrazující výjimky, prováděné SQL dotazy a další

3.1.3.2 Composer

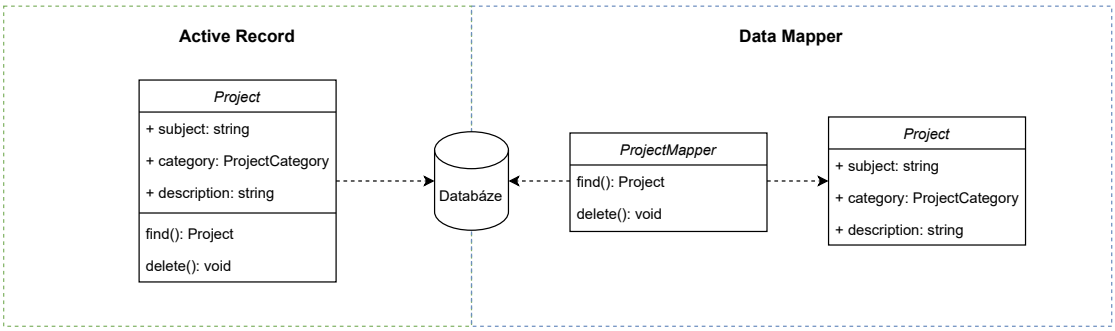
Composer je nástroj pro správu závislostí v PHP. Umožňuje vývojářům definovat (v souboru `composer.json`) knihovny, na kterých vyvíjená aplikace závisí, a composer je nainstaluje, případně aktualizuje. Balíčky, pokud není uvedeno jinak, neinstaluje globálně, ale v rámci projektu, kde je uchovává ve složce `vendor`. [42]

V této práci je Composer použitý pro nainstalování Laravel frameworku a také dalších závislostí.

3.1.3.3 Eloquent ORM

Eloquent je knihovna pro objektově relační mapování, což znamená, že se záznamy z databáze transformují na PHP objekty (nazývané modely) a naopak. [43]

Eloquent používá návrhový vzor Active Record, který má za následek narušení zásady třívrstvé architektury (prezentační, business a datová vrstva), jelikož model kromě uchovávání dat obsahuje také logiku. Druhým návrhovým vzorem, který ovšem respektuje třívrstvou architekturu, je Data Mapper, který nalezneme v ORM knihovně Doctrine, kterou naopak používá framework Symfony. V této bakalářské práci však použití Eloquentu nepřináší žádné znatelné omezení. Porovnání, jak mohou třídy vypadat v uvedených vzorech, zachycuje diagram 3.1.



Obrázek 3.1 Návrhové vzory – Active Record vs. Data Mapper

3.1.3.4 PhpOffice

PhpOffice je balíček, který v sobě obsahuje knihovny PhpSpreadsheet a PhpWord.

PhpSpreadsheet je knihovna napsaná v čistém PHP a nabízí třídy, které umožňují pracovat s různými formáty tabulek z tabulkových procesorů jako Microsoft Excel nebo LibreOffice Calc.[44]

Knihovna PhpWord je obdobou třídy PhpSpreadsheet, avšak pro textové dokumenty. Podporuje formáty Microsoft Office Open XML, Open Document Format, RTF, HTML a PDF. [45]

Tyto knihovny jsou používány pro exportování dat projektů pro Microsoft Excel a generování dokumentů zadání pro Microsoft Word, pro splnění funkčních požadavků FP5 a FP6, které vzešly ze sběru požadavků. To, že jsou často používány, dokazuje opět Packagist, kde PhpSpreadsheet má v době psaní tohoto textu přes 112 milionů stažení a PhpWord přes 12 milionů.

Užitečnou funkcionalitu v knihovně PhpWord nabízí třída TemplateProcessor, která umožňuje nahrazovat zástupná klíčová slova za poskytnuté hodnoty. Toho je využito právě u generování dokumentů se zadáním.

3.1.3.5 PhpAsn1

PhpAsn1 je knihovna, která umožňuje enkódovat dekódovat struktury ve formátu ASN.1 za použití pravidel X.690. Toto kódování je často použité v prostředí s X.509 PKI (Public Key Infrastructure). [46]

Tato knihovna se využívá ve třídě `TrustedTimestampService`, kde se s ní získává čas digitálního podepsání souboru. Ukázku použití zachycuje výpis kódu 5.

```php
$timestampReplyObject = ASNObject::fromBinary($data);
// Je potřeba projít objekt a jeho Sequence a Set pomocí indexů,
// které byly vypozorovány.
return Carbon::instance(
    $timestampReplyObject[1][1]
        ->getContent()[0][4]
        ->getContent()[0]
        ->getContent()[3]
        ->getContent()[1][1][0]
        ->getContent()
);
```

Výpis kódu 5 Získání podepsaného časového razítka pomocí knihovny PhpAsn1

3.1.4 Adresářová struktura backendu

Backendová část aplikace obsahuje následující adresáře a soubory. Složka `vendor` je vytvořena až po zavolání příkazů nástroje `composer` — `composer install`.

```
app ..... adresář obsahující kontrolery, modely, policies, servicies, ...
bootstrap ..... adresář obsahující pomocné skripty spuštěné při startování aplikace
config ..... adresář obsahující skripty pro nastavení serveru
database ..... adresář obsahující factories, migrations a seeders
lang ..... adresář obsahující soubory překladů hlášek
public
resources ..... adresář obsahující pohledy pro odesílání e-mailů
routes ..... adresář obsahující nastavení cest pro REST API
storage ..... adresář obsahující uložené soubory
tests
vendor ..... adresář obsahující externí balíčky (vytváří nástroj composer)
.env.exmample ..... soubor s ukázkou proměnných prostředí
artisan ..... command-line rozhraní pro práci s Laravelem
composer.json ..... správa závislostí pro composer
license.txt ..... text licence
README.md ..... instalační příručka
```

3.1.5 Použité technologie pro ukládání dat

Pro ukládání dat byla navržena relační databáze. Při vývoji byla pro aplikaci použitá databáze MariaDB, ovšem díky použití Eloquent ORM (viz výše) není problém použít jinou relační databázi jako např. MySQL, PostgreSQL, nebo i SQL Server. Na školním serveru, kde se bude aplikace provozovat, se používá MySQL.

3.1.6 Databázový model

Databázový model je navržený s důrazem na aktuální školní rok. Ten představuje tabulka `school_years`, která má boolean sloupec `active`, který určuje aktuálně probíhající školní rok. Na tuto tabulku je vytvořená vazba z tabulky `school_classes`, jelikož každá školní třída existuje pouze v jednom školním roce. Dále z ní vedou vazby na tabulky některých rolí. Výslovně na `class_teachers`, `chairmen`, `opponents` a `supervisors`, jelikož tyto role jsou taktéž platné

pouze v daném školním roce (př.: uživatel může být ve školním roce 2022/2023 třídním učitelem a vedoucím, v dalším školním roce již nebude třídním učitelem a také nemusí být ani vedoucím, ale může třeba některé projekty oponovat). Tabulka `chairmen` je se školními třídami propojená rovnou dvakrát, jelikož první vazba určuje předsedu maturitní komise a druhá místopředsedu.

Tabulka `users` reprezentuje uživatele v aplikaci. Aby mohl uživatel provádět určité akce, musí mít navíc vytvořené tabulky rolí, které zastupuje. Role administrátora není vázaná na školní rok, protože se předpokládá, že administrátor bude spravovat systém více let a v případě potřeby lze tuto roli jednoduše odebrat smazáním záznamu z tabulky a např. ji přiřadit někomu jinému.

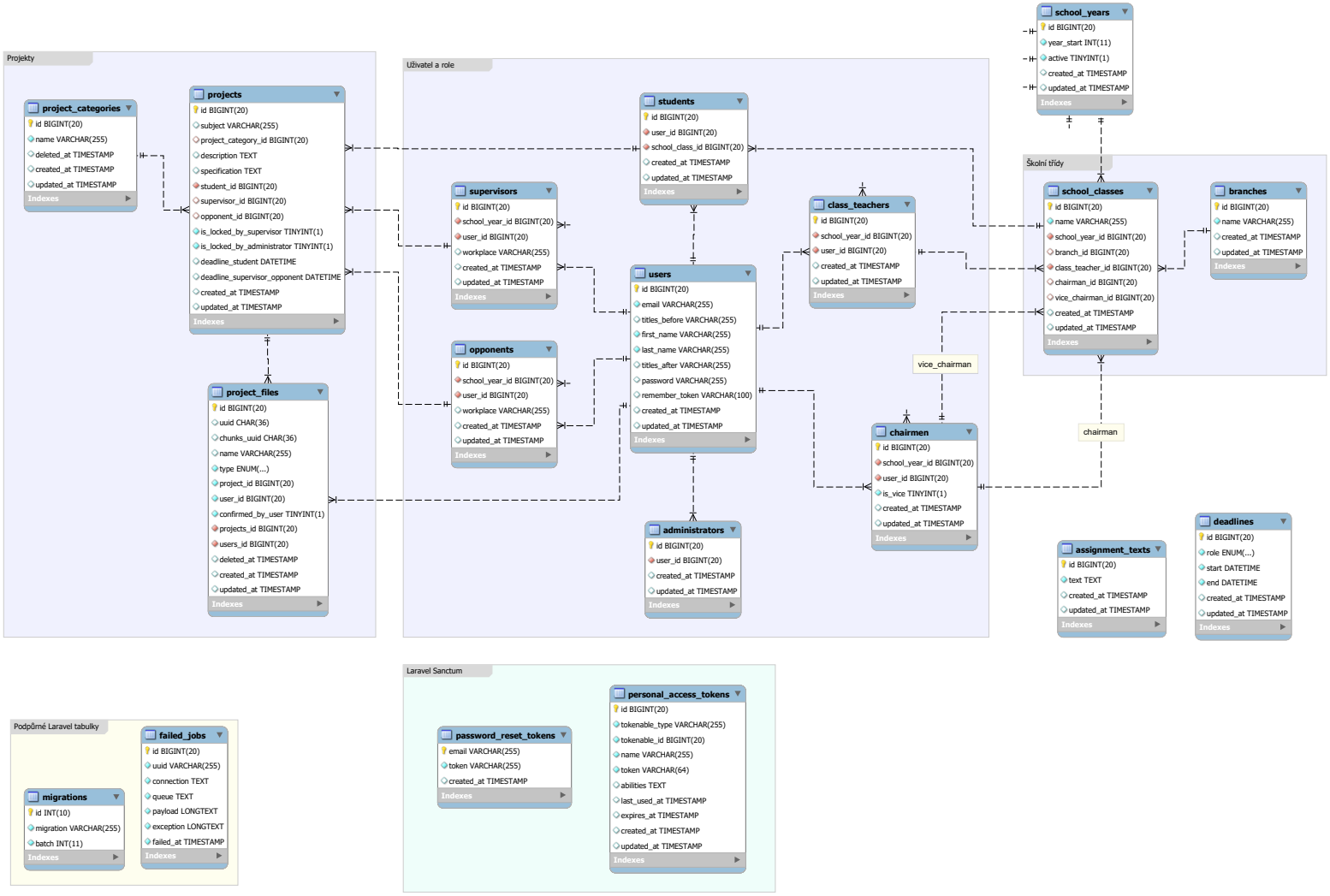
Další skupinou jsou tabulky uchovávající data pro projekty. Hlavní tabulkou pro tento účel je tabulka `projects` s vazbou na `projects_categories`, která se chová jako číselník. Projekt má vazbu na uživatelské role žáka, který ho vlastní (při vytváření žáka se mu vytváří prázdný projekt). Dále pomocí cizích klíčů určuje vazbu na vedoucího a oponenta. Důležitou součástí projektu jsou nahrané soubory, které se uchovávají v tabulce `project_files` a obsahují UUID, přes které lze najít nahraný soubor v úložišti souborů (mimo databázi).

V neposlední řadě můžeme v modelu nalézt pomocné tabulky `deadlines`, pro globální deadliny projektů, a `assignment_texts`, uchovávající obecný text zadání. Dále pak tabulky pro nástroj Laravel Sanctum a podpůrné tabulky pro Laravel s informacemi o databázových migracích a spuštěných úkolech ve frontě.

Databázový model lze blíže prozkoumat na ER diagramu 3.2.

3.2 Snímky obrazovky implementované aplikace

Snímky obrazovky z webové aplikace si lze prohlédnout v příloze C.



Obrázek 3.2 Databázový model

3.3 Nahrávání souborů

Jednou z obtížností, která musela být řešena při vývoji aplikace, bylo nahrávání souborů. To z důvodu povolené velikosti až 10 GiB pro jeden soubor, která je vymezená jako NP5 v 1.3.2. Soubory maturitního projektu dosahují velikosti v řádu gigabajtů, protože tématem mohou být i střih videa, 3D modelování s následným renderingem, grafická díla, apod.

V PHP interpreteru je velikost nahrávaných dat přímo omezena konfiguračními proměnnými `upload_max_filesize`, která definuje maximální velikost nahrávaného souboru, a `post_max_size`, definující maximální velikost odesílaných dat na server. Jelikož se nahrávaný soubor počítá do odesílaných dat, musí platit `post_max_size` $\geq$ `upload_max_filesize`, jinak se nikdy nedosáhne potenciálu hodnoty `upload_max_filesize`.

Dalšími proměnnými, na kterých práce se soubory závisí nepřímo, jsou `max_execution_time` označující maximální délku běhu skriptů, a `max_input_time`, který definuje maximální čas, po který může skript načítat odeslaná data.

Jedním způsobem, jak dosáhnout úspěšného nahrávání velkých souborů, je změnit výše uvedené proměnné v konfiguračním souboru `php.ini`, případně pomocí volání funkce `ini_set`. Nastavení hodnoty `post_max_size` na 10 GiB by však bylo příliš benevolentní pro všechny ostatní požadavky na server, čehož by mohl zneužít útočník a zatěžovat tak server. Navíc, pokud bychom přenášeli velký soubor tímto způsobem, a při nahrávání by došlo k chybě (z jakéhokoli důvodu), musel by se celý proces nahrávání opakovat, nehledě na to, zda bylo odesláno teprve několik megabajtů nebo už skoro celý soubor.

Tyto nedostatky řeší alternativní způsob nahrávání, implementovaný v této práci, a to nahrávání po kusech (tzv. chunk uploading). Soubory lze dělit na kusy dostatečně malé, aby se vešly do omezení `post_max_size` a `upload_max_filesize`. Ty se mohou simultánně a v jakémkoli pořadí přenést na server, kde se poté opět složí do jednoho velkého souboru (tentokrát samozřejmě ve správném pořadí).

Backend musí narozdíl od prvního, klasického způsobu nahrávání souboru, řešit následující problémy:

- Autorizaci – kusy ke konkrétnímu souboru může nahrávat pouze jeden uživatel
- Hlídání maximální velikosti souboru – jelikož se soubory přenáší po kusech, musí se spočítat velikost dosavadních nahraných kusů
- Sloučení souborů
  - Je potřeba vědět, jaké kusy patří ke kterému souboru, a dokázat je sloučit ve správném pořadí (vyřešeno pomocí složky s názvem podle UUID souboru)
  - Server musí vědět, kdy má začít se slučováním kusů

Autorizace je řešená pomocí kontroly záznamu v databázi v tabulce `project_files`. Ta obsahuje sloupce `user_id` s cizím klíčem na tabulku `users` a `chunks_uuid`, což označuje UUID složky, do které se kusy souborů nahrávají. Než backend zapíše kus souboru do složky s daným UUID (které se také přenáší v požadavku), zkontroluje, zda existuje takový záznam v databázi.

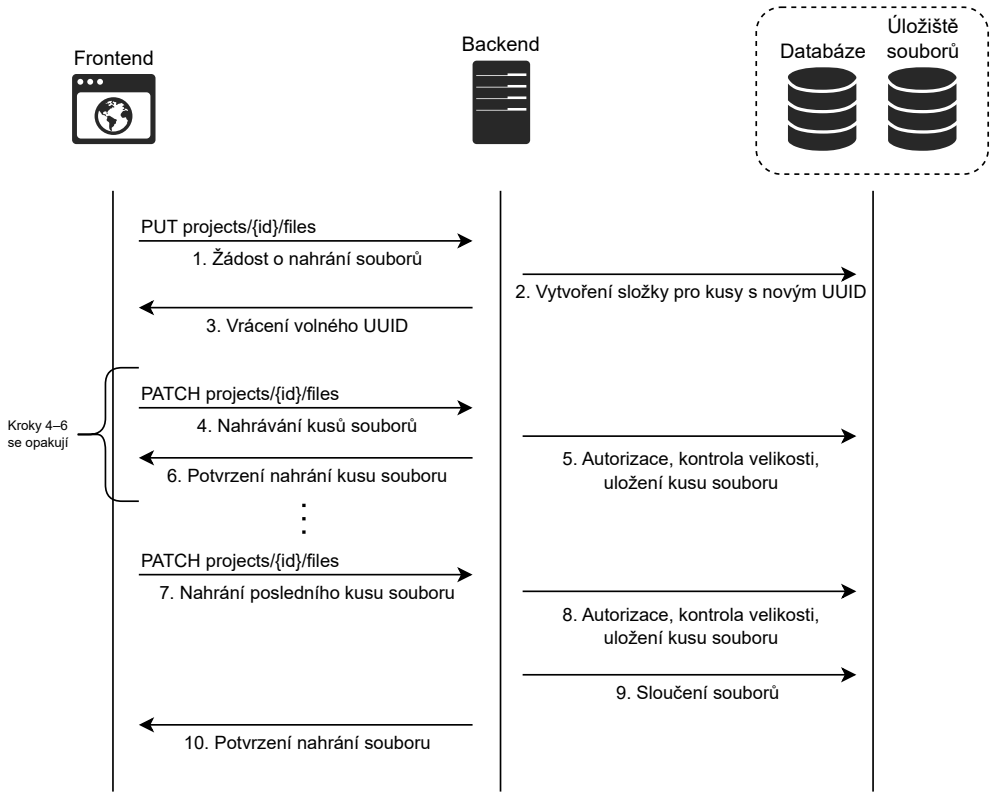
Spoléhání se pouze na náhodně generované UUID by nestačilo. Kdyby útočník dokázal odhadnout nebo odposlechnout UUID právě nahrávaného souboru jiného uživatele, mohl by nahrát kusy souborů místo něho. Že by se nemělo věřit náhodnosti UUID, doporučuje i OWASP ve svých Cheat Sheet Series: „*UUID typu 4 jsou generovány náhodně, [...]. Pokud není známo, že je jejich implementace bezpečná v konkrétním jazyce nebo frameworku, nemělo by se na náhodnost UUID spoléhat.*“ (přeloženo) [47].

Při každém požadavku na nahrání kusu souboru se hlídá nepřekročení maximální velikosti souboru tak, že se spočítá součet velikostí všech kusů ve složce s daným UUID.

Při vývoji bylo uvažováno nad dvěma možnostmi, jak backend může poznat, že se přenesl celý soubor. První možností je speciální HTTP požadavek ze strany frontendu, který tuto skutečnost

oznámí. Při znalosti velikosti celého souboru, kterou lze přenášet v hlavičce HTTP požadavku, se naskytuje druhá možnost, a to kontrola, zda velikost původního souboru odpovídá součtu velikostí dosud nahraných kusů. Tento součet je potřeba počítat, už kvůli předchozí kontrole velikosti popsané výše. Proto byla pro implementaci vybrána tato možnost. Komunikaci frontendu s backendem při nahrávání souboru po kusech zachycuje diagram 3.3. Na tento proces poté navazuje vytvoření digitálně podepsaného časového razítka a ověření dat uživatelem – tento pohled zachycuje obrázek 2.12.

Pro implementaci na frontendu byla zvolena, výše popsaná, knihovna FilePond [39]. Uvažovanou alternativou mohla být knihovna FineUploader [48], se kterou měl autor zkušenosti, jelikož se s ní setkal při práci na jiných projektech v minulosti. Protože byl vývoj této knihovny ukončen na podzim 2018 [49], nebyla nakonec vybrána pro tento projekt.



Obrázek 3.3 Komunikace se serverem při nahrávání souboru po kusech

3.4 Autentizace

Pro řešení autentizace byl zvolen nástroj Laravel Sanctum, jelikož škola, pro kterou je tato aplikace určena, v současné době neprovozuje vlastní autentizační server. Uživatel se v aplikace autentizuje pomocí e-mailu a hesla, které jsou uloženy v databázi. Heslo je uložené v podobě `bcrypt` hashe.

3.5 Autorizace

Díky tomu, že je Laravel komplexním frameworkem, máme k dispozici rovnou dvě řešení pro ulehčení autorizačních kontrol – brány (gates) a politiky (policies). Brány jsou jednoduché k po-

užití a slouží pro definování samostatných pravidel. Politiky umožňují tato pravidla uchovávat organizovaná, což je výhodné při vytváření robustních aplikací. [50] Proto byly pro tuto práci zvoleny politiky, které jsou logicky členěny podle modelů, se kterými pracují.

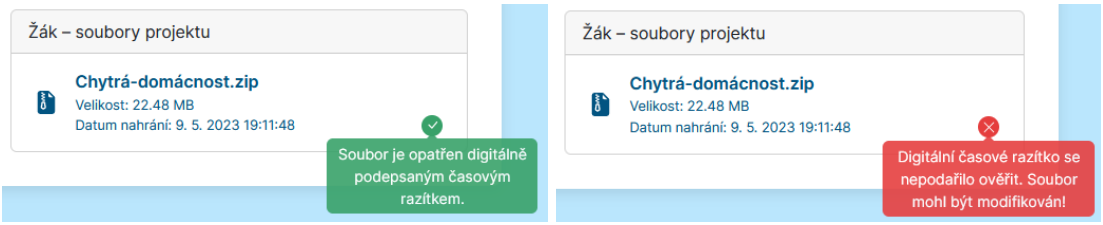
Při použití správného názvosloví Laravel umí tyto politiky sám přiřadit k odpovídajícím modelům, tudíž není potřeba politiky registrovat manuálně. Ověření se poté v kódu provádí přes volání metody `cannot` na uživateli, které se předá název pravidla v politice a instance modelu pro předání dat k ověření. Pokud je pravidlo nezávislé na konkrétní instanci, lze předat pouze název třídy, aby se aplikovala správná politika (např. `Project::class`). Ukázku definici pravidla v politice a následné ověření zachycuje výpis kódu 6.

3.6 Práce s časovým razítkem

Pro práci s digitálně podepsaným časovým razítkem je použitý nástroj OpenSSL s příkazem `ts`. Jedná se o volání command-line příkazů s kontrolou, zda příkaz úspěšně doběhl, případně se čtením standardního výstupu. Tyto příkazy jsou stejné s těmi, které již byly popsány ve výpisech kódu 1 a 2. Volání příkazů probíhá pomocí Process třídy, kterou poskytuje Laravel.

Tento přístup byl zvolen, jelikož PHP sice umí pracovat s knihovnou OpenSSL, ale nabízí z ní pouze některé příkazy. Příkazy pro časová razítka, v době psaní tohoto textu, nejsou v knihovně implementovány. [51]

Dále bylo potřeba vyřešit, jak naložit se získanou informací, zda se podepsané razítko dalo ověřit či nikoli. Neúspěšná verifikace razítka by v optimálním světě neměla nastat, značilo by to, že byl soubor nebo razítko modifikováno. To je ovšem přesně důvod, proč bylo kryptografické ověření práce navrženo a implementováno. Pesimistickým přístupem by bylo informovat uživatele, že se ověření nezdařilo a nedovolit mu takový soubor stáhnout. V práci je zvolen optimističtější přístup, kdy uživatele pouze upozorníme, ale soubor může být přesto stažen. Tím by tento proces každopádně neměl končit. Je potřeba, aby administrátor takovou situaci prověřil a zjistil příčinu nepovedené verifikace. Úspěšné a neúspěšné ověření časového razítka zachycují snímky obrazovky na obrázku 3.4.



Obrázek 3.4 Informování uživatele o výsledku ověření časového razítka u souboru

3.7 Kešování časového razítka

Jedním z problémů, který se při vývoji vyskytl, byla delší odezva od backendu, po nasazení funkcionality, která ověřuje platnost digitálního časového razítka a získává z něj důvěryhodný čas. Zaprvé samotné ověření časové razítka na lokálním serveru při vývoji trvalo v řádu jednotek sekund. Zadruhé se soubory s tímto ověřováním přenášely s ostatními daty projektu, přestože zobrazení těchto informací na frontendu je na sobě nezávislé. Místo toho, aby se na frontendu zobrazila data o projektu, která byla načtena rychle, čekalo se delší dobu na ověření digitálního razítka a po tuto dobu se uživateli na projektové stránce ukazovalo načítání.

Druhý problém byl vyřešen jednoduchým rozdělením cest v REST API. Soubory přiložené k projektu získaly novou cestu `projects/project/files`. Nyní se uživateli načte projekt rychle bez čekání na ověření razítka.

```php
/**
 * Třída pro politiku pro model Project obsahující pravidlo 'show'
 */
class ProjectPolicy
{
    public function show(User $user, Project $project): bool
    {
        return
            // Uživatel může zobrazit, pokud je administrátor, nebo...
            ($user->administrator) ||
            // ... je student, který vlastní tento projekt, nebo ...
            ($user->student &&
                $user->student->id === $project->student->id) ||
            /* ... další kontroly ... */;
    }
}

/**
 * Kontroler obsahující metodu pro zobrazení konkrétního projektu
 */
class ProjectController extends Controller
{
    public function show(Request $request, Project $project): ProjectResource
    {
        // Pokud není podmínka splněna, vrať Error 403 Forbidden
        abort_if(
            // Zkontroluj, zda přihlášený uživatel splňuje pravidlo 'show'
            // definované v politice (Project -> ProjectPolicy)
            // s ověřením konkrétního projektu
            $request->user()->cannot('show', $project),
        403);

        // Zde kód pokračuje, pokud byla podmínka splněna...
    }
}
```

Výpis kódu 6 Definování pravidla v politice a jeho ověření

```
// Definice konstant pro kešování - prefix pro klíč a doba TTL
const CACHE_PREFIX = "uploaded-file-";
const CACHE_TTL = 60 * 60; // 1 hodina

// Pokusí se vrátit data z cache, jinak vrátí nová (cache miss)
return cache()->remember(
    self::CACHE_PREFIX . $this->id, // Klíč
    self::CACHE_TTL, // TTL
    function () { // Callback, který je volán při cache miss
        return [
            'digital_timestamp' => [
                // Časově náročné operace, kvůli kterým se kešuje
                'datetime' =>
                    TrustedTimestampService::
                        getSignedDatetimeFromTimestampReply($this),
                'verified' =>
                    TrustedTimestampService::
                        verifyDigitalTimestamp($this),
            ],

            /* ... další souborová metadata ... */
        ];
});
```

Výpis kódu 7 Kešování výsledku ověření časového razítka

Po rozdělení cest už první problém není tak vážný, při testování vycházelo ověřování razítka souboru okolo 5 sekund. Přesto bylo pro lepší UX rozhodnuto problém vyřešit. V optimálním případě by se nahraný soubor a jeho časové razítko neměly v úložišti souborů změnit. Zároveň razítko slouží jako důkaz, zda je zachována integrita a nepopiratelnost souboru. Z těchto důvodů není potřeba, aby ověření razítka probíhalo vždy v reálném čase. Kdyby nastal akutní případ, že by bylo potřeba mít aktuální důkaz, lze ověření razítka získat na vyžádání. Z těchto důvodů se metadata souboru, včetně výsledků ověření, kešují. Čas, po kterém se cache zneplatní (TTL – Time to Live), byl stanoven na 1 hodinu, což by měla být dostatečně dlouhá doba, aby mohli uživatelé pracovat v systému bez zbytečného zatěžování serveru. Současně to je dostatečně krátká doba, aby data uložená v cache nebyla příliš stará a uživatelé stále měli jistotu, že jsou nahrané soubory v pořádku.

Jak se v Laravelu s kešováním pracuje, zobrazuje výpis kódu 7.

3.8 Problém N+1 dotazů

Dalším problémem, v Laravelové komunitě již dobře známým [52], byl tzv. problém N+1 dotazů. Ten nastává, když se při iterování přes kolekci modelů přistupuje přes definované vztahy k jiným modelům. Následkem je, že se pro každou položku v iterátoru vykoná samostatný SQL dotaz do databáze. Ve výchozím stavu totiž Laravelové modely používají tzv. líné načítání (lazy loading) a modely připojené přes vztahy jsou získány z databáze až v momentu, kdy je k nim opravdu přistoupeno. [52]

Následující PHP kód pro vypsání všech žáků a jejich projektů by obsahoval problém N+1 dotazů.

```
// Model Student má vazbu HasOne na model Project
foreach(Student::all() as $student)
    echo $student->project->subject;
```

Výpis kódu 8 Kód obsahující problém N+1 dotazů

Vykonané SQL dotazy by totiž vypadaly následovně. Pro každého žáka by se načítala data o jeho projektu v samostatném dotazu. To mívá za následek zbytečnou zátěž databáze a zpomalení celé aplikace.

```
-- 1. dotaz - získání všech žáků
SELECT * FROM students;

-- Dalších N dotazů, pro získání N projektů stylem každý zvlášť
SELECT * FROM projects WHERE projects.student_id = 1;
SELECT * FROM projects WHERE projects.student_id = 2;
SELECT * FROM projects WHERE projects.student_id = 3;
...
SELECT * FROM projects WHERE projects.student_id = N;
```

Výpis kódu 9 Problém N+1 dotazů v podobě SQL dotazů

Tento problém lze lehce vyřešit použitím tzv. eager (nedočkavého) načítání, které spočívá v dopředném definování vazeb, se kterými bude kód dále pracovat a Eloquent tyto vazby dokáže načíst pomocí méně SQL dotazů, většinou pomocí jediného. Vazby se dají definovat pomocí metody `with`.

```
// Použití eager loadingu pro opravení problému N+1 dotazů
foreach(Student::with('project')->get() as $student)
    echo $student->project->subject;
```

Výpis kódu 10 Kód s eager načítáním, které řeší problém N+1 dotazů

V tomto případě se SQL dotazy vykonají pouze dva.

```
-- 1. dotaz - získání všech žáků
SELECT * FROM students;

-- Pouze jeden dotaz získavající data o potřebných projektech
SELECT * FROM projects WHERE projects.student_id IN (1, 2, 3, ..., N);
```

Výpis kódu 11 Odpovídající SQL dotazy, když je použito eager načítání

Při implementaci této aplikace bylo s těmito problémy dopředu počítáno. Pokud takový problém stejně nastal, byl nalezen a vyřešen pomocí použití eager načítání.

Kapitola 4
Testování

4.1 Uživatelské testování původního prototypu

Uživatelské testování implementované webové aplikace teprve proběhne před nasazením na nový školní rok. Nová implementace nahrazuje starý prototyp, který je však nyní druhým školním rokem v provozu. Díky tomu se k autorovi dostala zpětná vazba od skutečných uživatelů aplikace – žáků, učitelů i administrátorů.

Jedním z UX problémů, který byl nahlášen, bylo, že uživatelé občas nerozpoznali, co znamená, že je projekt uzamčen. Někteří se domnívali, že se to vztahuje i na nahrávání souborů, které tím ovšem dotčeno není a řídí se globálními deadliny, případně prodlouženými deadliny pro konkrétní projekt. Tyto žáky totiž mátlo, že se v patičce na stránce s projektem (viz wireframe 2.6) místo tlačítka „Uložit“ zobrazila hláška *Projekt je uzamčen pro editaci.* Jednalo se ovšem pouze o pár případů, které by se daly řešit individuálním vysvětlením nebo poskytnutím uživatelské příručky. Přesto byl tento problém vzat v potaz a hláška byla přepsána na *Projekt je uzamčen pro úpravu zadání. Můžete však nahrávat soubory do daného deadline.*

Přihlašování přes uživatelské jméno (login) bylo další vznesenou připomínkou u prototypu aplikace. Ten odpovídá začátku (po zavináč) školní e-mailové adresy, která má na SPŠE a VOŠ Pardubice jeden z následujících tvarů:

- Pro učitele – `prijmeni@spse.cz`
- Pro žáky se jedná o kombinaci až prvních šesti znaků z příjmení, následují dva znaky z křestního jména a za ně se připojí rok nástupu. Navíc pro e-maily žáků je použita subdoména `zaci.spse.cz`.
  - E-mail fiktivního žáka Františka Omáčky by mohl vypadat `omackafr23@zaci.spse.cz`

Uživatelé si nebyli jistí, zda mají zadávat uživatelské jméno nebo e-mailovou adresu – přikláněli se spíše k e-mailové adrese. Navíc předsedové maturitní komise jsou učitelé z jiných škol a ve webové aplikaci se jim jako uživatelské jméno nastavovala celá e-mailová adresa, aby to pro ně bylo intuitivní. Dá se říct, že zde vznikalo takové schizma. Tato připomínka byla také vyřešena, s uživatelským jménem se stejně pracovalo pouze při přihlášení a lze ho vždy vygenerovat z e-mailové adresy, není potřeba ho ukládat navíc v databázi. Pro autentizaci se v nové webové aplikaci používá e-mailová adresa.

Při testování prvotního prototypu bylo ještě nalezeno několik problémů s nesprávnou funkčností způsobenou chybami v kódu, které byly vyřešeny.

4.2 Testování nahrávání souborů po kusech

Při testování funkčnosti nahrávání souborů po kusech (popsáno v podkapitole 3.3) se nenarazilo na žádné problémy. Testováno bylo na lokálním vývojovém prostředí na Windows 10 s PHP 8.1 a Laravelem v10.3.1. Při testech byly nahrávány soubory různých velikostí, od několika megabajtů až po konečnou hranici 10 GiB.

Všechny soubory byly úspěšně nahrány, při nahrávání nenastal žádný problém a bylo ověřeno, že kontrolní součty souboru u klienta a výsledného složeného souboru na backendu si navzájem odpovídají.

4.3 Testování funkčnosti časového razítka

Při nahrávání souborů během testování nahrávání souborů po kusech (předchozí podkapitola) server vytvářel podepsaná časová razítka pro každý soubor. Razítka se podařilo vždy ověřit a backend správně odesílal na frontend informaci, že se verifikace podařila.

Pro otestování případu, kdy se razítko nepovede ověřit, byly soubory uložené na serveru záměrně modifikovány. Ověření bylo aktualizováno až za hodinu, jelikož je výsledek verifikace kešován z důvodů, které jsou popsány v podkapitole 3.7. Server nedokázal razítko ověřit a tentokrát na frontend odesílal výsledek neúspěšné verifikace.

Testování této funkcionality dopadlo úspěšně a veškeré testy dopadly podle očekávání.

4.4 Testování vůči OWASP TOP 10 zranitelnostem

OWASP Top 10 je žebříček deseti největších potencionálních hrozeb, které mohou nastat u webových aplikací. [53] Poslední revize tohoto žebříčku proběhla v roce 2021.

Následující kapitoly popisují jednotlivé hrozby a obsahují diskuzi s případným testováním, zda je webová aplikace, vyvinutá v rámci této bakalářské práce, vůči nim bezpečná.

4.4.1 A01:2021-Broken Access Control

Kontrola přístupu definuje politiku takovou, aby uživatelé nemohli s aplikací pracovat mimo svá vymezená přístupová práva. Chyby v kontrole přístupu typicky vedou k neoprávněnému přístupu k informacím, modifikaci nebo zničení všech dat, či k vykonání logiky, která je za limitem, který má uživatel nastavený. [54]

Všechny akce, které uživatel provádí, jsou ověřovány na backendu pomocí politik popsaných v podkapitole 3.5. Jedinou informací navíc, kterou by uživatel mohl získat pomocí reverzního inženýrství, je vytvoření představy, jak vypadá na frontendu administrační rozhraní a jaké cesty obsahuje na REST API. K těmto cestám nemá samozřejmě běžný uživatel přístup a nastavená politika na backendu by mu vrátila chybu 403 Forbidden. Tuto informaci lze získat kvůli tomu, že je celá frontendová SPA aplikace psána a generována dohromady a obsahuje i komponenty pro administrační rozhraní. Jelikož se jedná o vcelku běžnou administraci, pro systém tato případná znalost nepředstavuje hrozbu. Zabránit by tomu šlo pomocí server-side renderingu, kdy frontendová aplikace získává komponenty ze serveru až po autorizaci.

Všechny REST API cesty, které backend nabízí, byly při testu projity. Prvním testem byly předpokládané funkčnosti akcí, ke kterým jsou uživatelé autorizováni. Druhým testem se zkoumalo, zda uživatelé nemohou provádět akce nad rámec svých přístupových práv. Při tomto testu nebyly nalezeny žádné problémy.

Tato kategorie zahrnuje i slabinu špatně nakonfigurovaných CORS (Cross-Origin Resource Sharing), které by umožňovaly přístup na API z neověřených a nedůvěryhodných zdrojů. Seznam povolených hostů se definuje pomocí klíče `allowed_origins` v souboru `config/cors.php` na

backendu. `Allowed_origins` jsou nastaveny na hodnotu `FRONTEND_URL` získanou z proměnných prostředí.

4.4.2 A02:2021-Cryptographic Failures

V předešlé verzi žebříčku se tato kategorie nazývala Sensitive Data Exposure. Nová kategorie se zaměřuje na chyby v použití kryptografie nebo na chybějící kryptografii v místech, kde by měla být použita. [55]

Při testování bylo zjištěno, že žádná data nejsou přenášena v podobě prostého textu. Pro webové požadavky se používá protokol HTTPS, pro e-mailovou komunikaci s protokolem SMTPS (Simple Mail Transfer Protocol Secure) se používá šifrování TLS (Transport Layout Security).

Pro ukládání hesel v databázi se používá hashovací funkce `bcrypt`. U kryptografického ověření časového razítka práce je nutné důvěřovat použitému TSA serveru a nástroji OpenSSL.

4.4.3 A03:2021-Injection

Do této kategorie spadají slabiny jako Cross Site Scripting (XSS) nebo třeba SQL injection. [56]

Díky použitému frameworku Laravel a stavbě SQL dotazů pouze přes Eloquent bez použití metody `DB::raw` je aplikace dostatečně chráněná proti útoku SQL injection, resp. musíme tomuto frameworku důvěřovat. Což ovšem s klidným svědomím můžeme díky jeho široké komunitě, open-source licenci a léty prověřenému provozu.

Na cestách API byl spuštěn nástroj `sqlmap` pro automatizované otestování této zranitelnosti. Ten však zjistil, že by parametry neměly být náchylné vůči danému útoku.

Použití frameworku React na frontendu by mělo aplikaci dostatečně chránit vůči útoku XSS. Na jednom místě v aplikaci je povoleno vkládání HTML kódu, což tuto ochranu narušuje v tomto konkrétním místě. Tento problém byl ale vyřešen pomocí sanitizace vstupu pomocí Dompurify, o kterém je psáno v podkapitole 3.1.1.8.

Uvedené místo bylo otestováno na vložení vstupů obsahující škodlivý kód ze seznamu známých XSS payloadů z GitHub repozitáře `swisskyrepo/PayloadsAllTheThings`. Díky přísně nastaveným pravidlům pro Dompurify, žádný ze škodlivých kódů nebyl vykonán. Aplikaci by šlo preventivně ještě chránit pomocí nastavení CSP (Content Security Policy) hlaviček, to ovšem už přesahuje rámec této práce.

4.4.4 A04:2021-Insecure Design

Nová kategorie z roku 2021 se zaměřuje na hrozby vztažené k návrhu a architektuře aplikace. Mezi chyby spadající do této kategorie může patřit generování chybových hlášek obsahujících citlivé informace nebo nezabezpečené úložiště přihlašovacích údajů. [57]

Tato kategorie by byla pro autora obtížná k testování, jelikož při návrhu dbal na bezpečnost a tvořil ho s použitím doporučení skupiny OWASP. Proto by při white-box[1] testování došel ke stejným závěrům jako při návrhu.

4.4.5 A05:2021-Security Misconfiguration

Jelikož se dnes aplikace posouvají směrem k vysoce konfigurovatelnému software, není divu, že se tato kategorie posunula na pátou příčku v žebříčku. Hlavními slabinami v této kategorii jsou konfigurace, nedostatečná omezení a XML External Entity Reference (XXE). [58]

Jelikož vyvíjená aplikace nepracuje s daty ve formátu XML, není zde místo (aspoň v kódu psaném autorem), kde by mohl nastat útok XXE. Tato kategorie je opět široká a obtížná na

[1] Testování s plným přístupem ke zdrojovým kódům a návrhům.

testování. Podle ukázkových scénářů útoků, které uvádí OWASP, je vyvíjená aplikace chráněna následovně:

- V aplikaci nejsou ponechány žádné ukázky, které by útočník mohl zneužít.
- Výpis obsahu adresáře na serveru (directory listing) není povolený.
- Aplikace v chybových hláškách neuvádí detailní chybové informace, které by obsahovaly například výpis průběhu kódu (stack trace).

4.4.6 A06:2021-Vulnerable and Outdated Components

Tato kategorie získala umístění na druhé pozici v komunitním dotazníku, ale měla také dostatečně výskytů v nasbíraných datech, aby se umístila v Top 10. [59]

Balíčkovací nástroje pro správu závislostí (frontend – npm, backend – composer) obsahují příkaz `audit`, který umí vypsat nahlášené zranitelnosti u aktuální verze balíčku.

Na frontendu příkaz `npm audit` nalezl zranitelnost střední závažnosti (moderate), a to XSS v balíčku quill. O této chybě byla na serveru GitHub vedena diskuze, zda se nejedná o falešně pozitivní označení zranitelnosti. [60] Autor zkoušel tuto zranitelnost reprodukovat a nepodařilo se mu dosáhnout situace, ve které by aplikace spustila škodlivý kód. Druhou nalezenou zranitelností na frontendu byl balíček `nth-check`, na kterém je závislý balíček `react-scripts`. Zde byla uvedena vysoká závažnost (high) z důvodu Inefficient Regular Expression Complexity. Tento balíček se používá jen ve vývojové verzi `dev`, jak je popsáno v GitHub issue [61], proto aplikaci, vyvinutou v bakalářské práci, případná zranitelnost neohrožuje.

Na backendu nebyly nalezeny žádné zranitelnosti pomocí příkazu `composer audit`.

4.4.7 A07:2021-Identification and Authentication Failures

Tato kategorie zahrnuje slabiny jako nesprávnou autentizaci nebo Session Fixation. [62]

Díky tomu, že je autentizace řešena pomocí open-source balíčku Laravel Sanctum, používají se komunitou ověřené best-practices a většina problémů je vyřešena v základu. Jednou slabinou, které OWASP uvádí, je povolení slabých a velmi známých hesel jako `Password1`. Toto bylo vyřešeno pomocí nastavených validačních pravidel pro heslo.

Jako další slabina v této kategorii je uvedena práce se session. Identifikátor session se nikdy nepřenáší v URL a po přihlášení uživatele se nepoužívá znovu, ale je vygenerován nový. Zneplatnění session při odhlášení nebo při dlouhé neaktivitě také probíhá v pořádku.

4.4.8 A08:2021-Software and Data Integrity Failures

Nová kategorie z roku 2021 se zaměřuje na předpoklady pro update softwaru, kritická data a CI/CD (Continuous Integration/Continuous Deployment) pipeline bez ověřování integrity. Toto zahrnuje slabiny jako stahování kódu bez ověření integrity nebo deserializace neověřených dat. [63]

V tomto projektu se nepracuje s CI/CD pipeline, proto aplikaci není možné otestovat vůči uvedené hrozbě. Deserializace objektů také není prováděna.

4.4.9 A09:2021-Security Logging and Monitoring Failures

Logování a monitorování získalo třetí místo v komunitním žebříčku. Logování může být obtížné pro testování, ale pro detekování a řešení incidentů je kritické. [64]

Logování chyb, případně i dalších informací, probíhá do souboru `laravel.log`. Komplexnější monitorování aplikace (např. detekce automatizovaných nástrojů) není v této práci řešena.

4.4.10 A10:2021-Server-Side Request Forgery

Server-Side Request Forgery (SSRF) nastává, když aplikace nahrává vzdálený prostředek bez validace URL dodané uživatelem. To může útočníkovi umožnit odeslání požadavku na neočekávané cíle, a to i na takové, které jsou uživateli za normálních okolností skryty za firewallem. [65]

Při testování nebyla nalezena žádná volání, která by skládala URL z uživatelských neověřených dat. SSRF tedy nemůže v této verzi aplikace nastat.

Kapitola 5
Závěr

Cílem této práce bylo provést sběr a analýzu požadavků uživatelů, vytvořit rešerši existujících řešení, navrhnout bezpečnostní model pro nezpochybnitelné odevzdání souborů, v neposlední řadě pak navrhnout a implementovat samotnou webovou aplikaci s ohledem na bezpečnost a kryptografické ověření práce.

Při rešerši existujících řešení nebyla nalezena žádná dostupná aplikace, která by uspokojivě řešila všechny požadavky Střední průmyslové školy elektrotechnické a Vyšší odborné školy Pardubice. Proto byl proveden sběr požadavků od vedení školy s následnou analýzou těchto požadavků a navržením případů užití.

Z této analýzy vycházel návrh bezpečnostního modelu, kde byly identifikovány možné hrozby a navrženy jejich mitigace. Při tomto návrhu byla pro nezpochybnitelné odevzdání souborů maturitní práce zvolena digitálně podepisovaná časová razítka. Ta zaručují integritu souborů a časovou nepopiratelnost.

Poté byla v řešení webová aplikace pro správu a odevzdání projektů. Navrhla se její architektura a nakreslily se drátěné modely pro rozvržení některých obrazovek a jejich komponent.

Výsledkem je implementovaná webová aplikace skládající se ze dvou částí, které spolu komunikují – z frontendové Single Page Application a backendového REST API. Ta obsahuje různé stránky a logiku v závislosti na uživatelských rolích. Žák má možnost upravovat zadání svého projektu a odevzdat k němu požadované soubory v určeném deadline. Vedoucí s oponentem v aplikaci nahrávají své posudky k daným projektům. Třídní učitel, předseda a místopředseda maturitní komise dostávají přístup pro čtení k projektům v daných třídách v termínech stanovených zákonem. Administrátor má k dispozici administrační rozhraní, ve kterém sleduje stav práce na projektech, včetně odevzdávání souborů. Dále z něho importuje nový školní rok s novými uživateli a rolemi i již pro existující uživatele v systému. Také může exportovat zadání do připravené šablony k tisku, aby tato zadání mohl podepsat ředitel školy a aby se mohla archivovat. Tato zadání poté nahrává zpět do systému, aby k nim měli žáci přístup.

Při implementaci této aplikace byl brán ohled na bezpečnostní doporučení projektu OWASP. Nejvíce u nahrávání souborů, které obsahuje množství potenciálních hrozeb a slabé implementace. Dále pak např. u funkcionality zapomenutého hesla, specifických nastavení pro framework Laravel nebo při prevenci útoku OS Command Injection.

Webová aplikace se nasadí pro produkční použití na serveru SPŠE a VOŠ Pardubice již v následujícím školním roce a nahradí tím dva roky používaný prototyp, který neobsahoval administrační rozhraní a některé další funkcionality. Před nasazením webové aplikace na školní server se počítá s vykonáním uživatelského testování aplikace.

Do budoucna je aplikaci možno dále rozvíjet. Počítá se s napsáním jednotkových testů, které pomohou detekovat chyby.

Příloha A

Instalační příručka – frontend

Soubory frontendu se nachází ve složce `mp_frontend`. Veškeré operace popisované v této příručce počítají se spouštěním příkazů v této složce.

Tuto příručku lze také nalézt ve formátu Markdown v souboru `README.md`.

A.1 Verze nástrojů

Vývoj probíhal na prostředí s operačním systémem Windows 10 s následujícími nástroji daných verzí:

- `node v14.17.6`
- `npm 6.14.15`

A.2 Nainstalování závislostí

Před spuštěním aplikace, nebo i před spuštěním sestavení aplikace pro nasazení na produkční server je potřeba nainstalovat závislosti definované v souboru `package.json`. Toho lze docílit příkazem pro nástroj `npm`.

```
npm install
```

Balíčky, na kterých je aplikace závislá, se nainstalují do složky `node_modules`.

A.3 Nastavení proměnných prostředí

Dále je potřeba nastavit proměnné prostředí v souborech `.env`, případně `.env.local` pro lokální vývoj. Ukázku tohoto nastavení lze najít v souboru `.env`.

Je potřeba nastavit hodnotu proměnné `REACT_APP_API_URL`, která určuje URL adresu, na které je dostupný backend poskytující REST API pro tuto aplikaci.

A.4 Spuštění aplikace v lokálním vývojovém prostředí

Aplikaci lze spustit pro lokání vývoj pomocí následujícího příkazu:

```
npm run start
```

V příkazovém řádku by se měla objevit informace o sestavení vývojové verze:

```
Compiled successfully!

You can now view mp-frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```

A.5 Sestavení aplikace pro produkční nasazení

Aplikaci lze sestavit pomocí následujícího příkazu:

```
npm run build
```

Sestavenou aplikaci lze nalézt v nově vytvořené složce `build`.

Příloha B

Instalační příručka – backend

Soubory backendu se nachází ve složce `mp_backend`. Veškeré operace popisované v této příručce počítají se spouštěním příkazů v této složce. Tuto příručku lze také nalézt ve formátu Markdown v souboru `README.md`.

B.1 Verze nástrojů

Vývoj probíhal na prostředí s operačním systémem Windows 10 s následujícími verzemi nástrojů:

- `PHP 8.1.12`
- `Composer 2.0.8`
- `Laravel 10.3.1`

B.2 Požadavky na server

Použití frameworku Laravel definuje následující požadavky na server:

- `PHP >= 8.1`
- `Ctype PHP Extension`
- `cURL PHP Extension`
- `DOM PHP Extension`
- `Fileinfo PHP Extension`
- `Filter PHP Extension`
- `Hash PHP Extension`
- `Mbstring PHP Extension`
- `OpenSSL PHP Extension`
- `PCRE PHP Extension`
- `PDO PHP Extension`

- `Session PHP Extension`
- `Tokenizer PHP Extension`
- `XML PHP Extension`

B.3 Nainstalování závislostí

Před spuštěním serveru je potřeba doinstalovat závislosti definované v souboru `composer.json`. To lze provést pomocí nástroje `composer`.

```
composer install
```

Závislosti se nainstalují do složky `vendor`.

B.4 Nastavení proměnných prostředí

Dále je potřeba nastavit proměnné prostředí v souboru `.env`. Ukázku tohoto souboru lze najít v souboru `.env.example`. Tento soubor zkopírujte nebo přejmenujte na `.env`.

Je potřeba nastavit následující proměnné kvůli správnému fungování autentizačních cookies:

- `FRONTEND_URL`
- `SANCTUM_STATEFUL_DOMAINS`
- `SESSION_DOMAIN`

Dále pak proménně s prefixy:

- `APP_`
- `DB_`
- `MAIL_`

A nakonec proměnné pro správné fungování časových razítek:

- `TIMESTAMP_SERVER_URL`
- `PATH_TO_OPENSSL`
  - proměnná může zůstat prázdná, pokud lze spouštět příkaz `openssl` pomocí správně nastavené systémové proměnné PATH

B.4.1 Vygenerování klíče

Klíč pro šifrování dat je potřeba vytvořit pomocí následujícího příkazu.

```
php artisan key:generate
```

B.5 Vytvoření databázových tabulek

Tabulky lze vytvořit pomocí následujícího příkazu:

```
php artisan migrate
```

Pro vložení základních dat do databáze lze spustit seeder:

```
php artisan db:seed
```

B.6 Bezpečnostní upozornění

Z hlediska bezpečnosti je více než důležité, aby HTTP server měl nastavenou kořenovou složku pro VirtualHost na složce `public`. Jinak by byl konfigurační soubor `.env` vystaven veřejně a mohl by být přečten útočníkem, čímž by mohlo dojít ke kompromitaci dat!

B.7 Optimalizace pro nasazení na produkční server

Před nasazením na produkční server by měla proběhnout optimalizace.

Pro optimalizaci autoloaderu, který je generován nástrojem ‘composer‘, lze spustit následující příkaz.

```
composer install --optimize-autoloader --no-dev
```

Pro nakešování konfigurace Laravelu a dalšího lze spustit tyto příkazy.

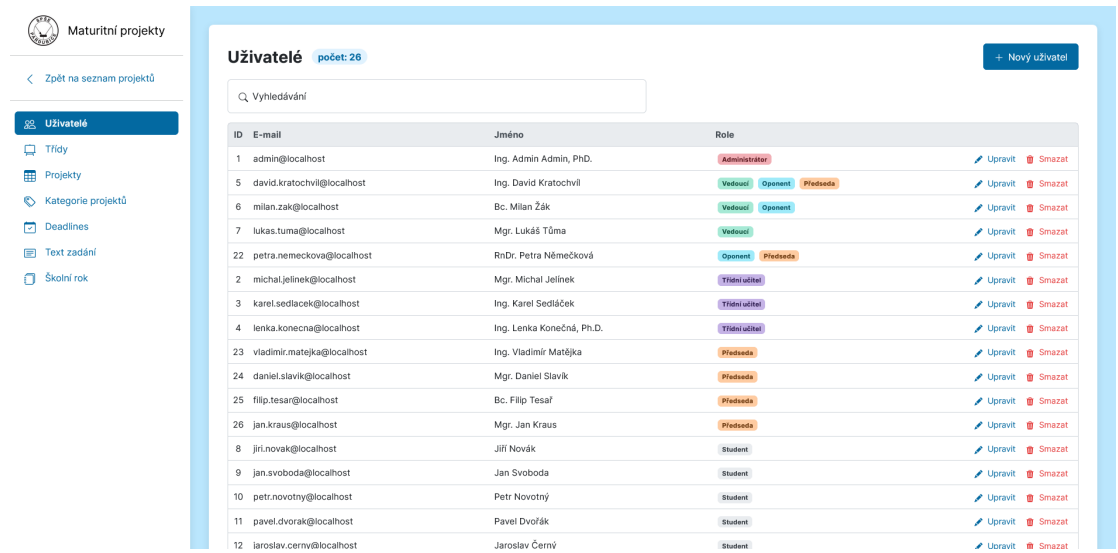
```
php artisan config:cache
php artisan event:cache
php artisan route:cache
php artisan view:cache
```

B.8 Spuštění aplikace

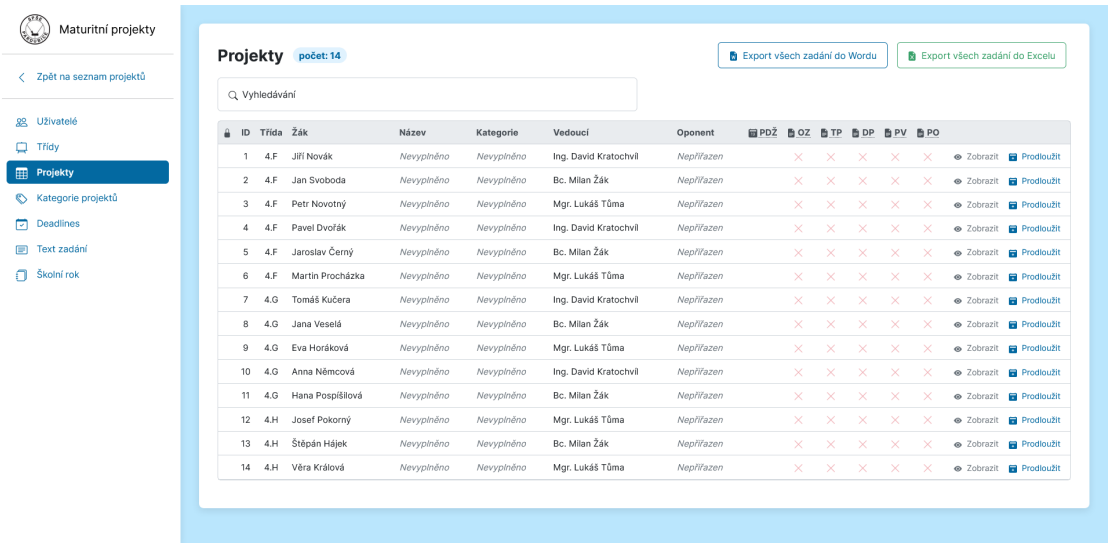
Aplikace je spuštěna ve chvíli, kdy je soubor `public/index.php` vystaven na zapnutém serveru.

Příloha C

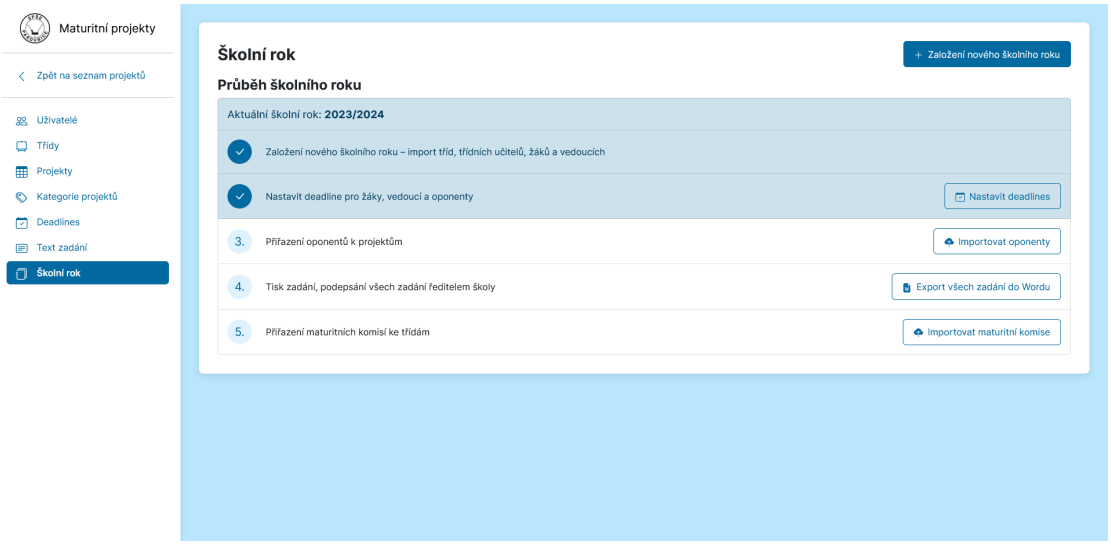
Snímky obrazovky implementované aplikace



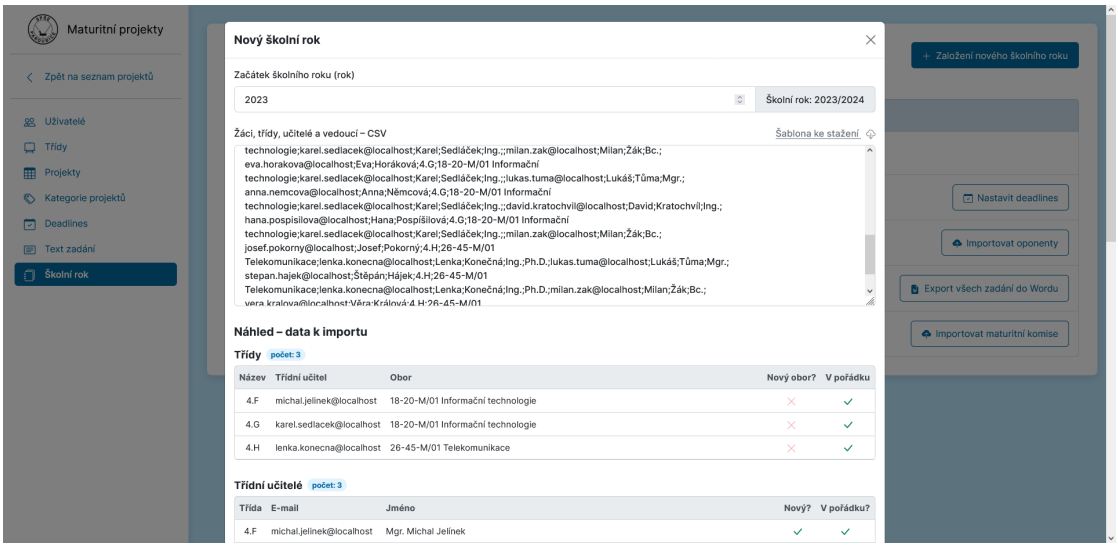
Obrázek C.1 Administrace uživatelů



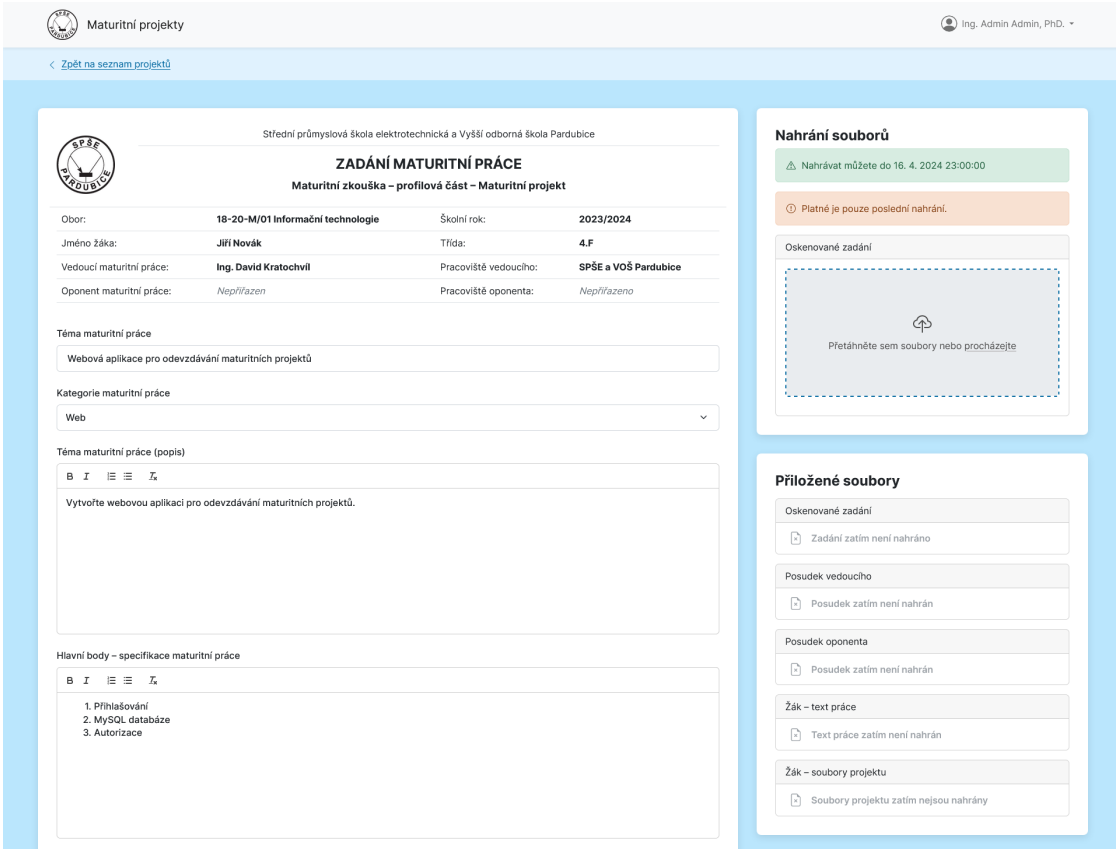
Obrázek C.2 Administrace projektů



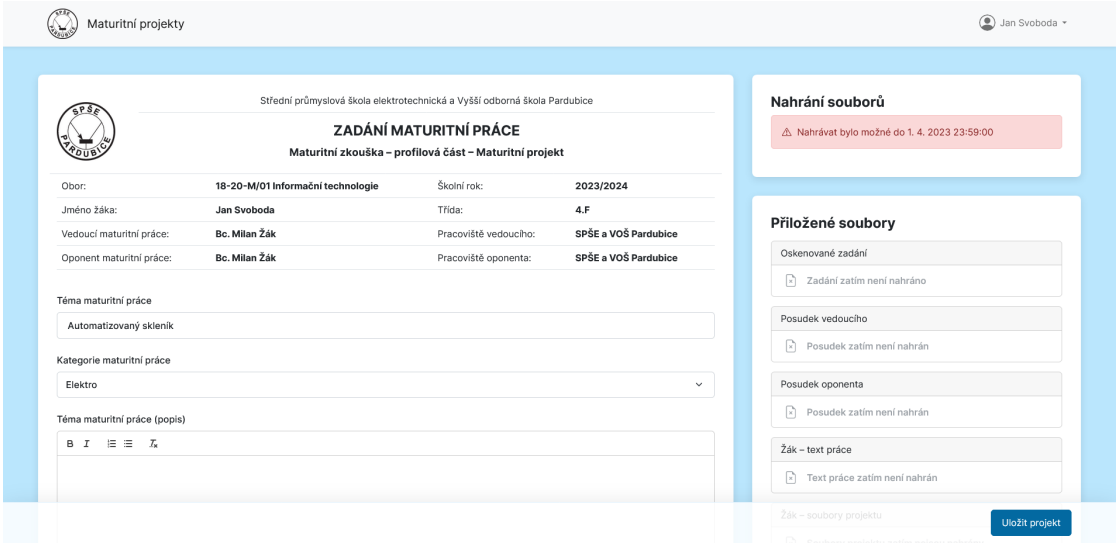
Obrázek C.3 Administrace školního roku



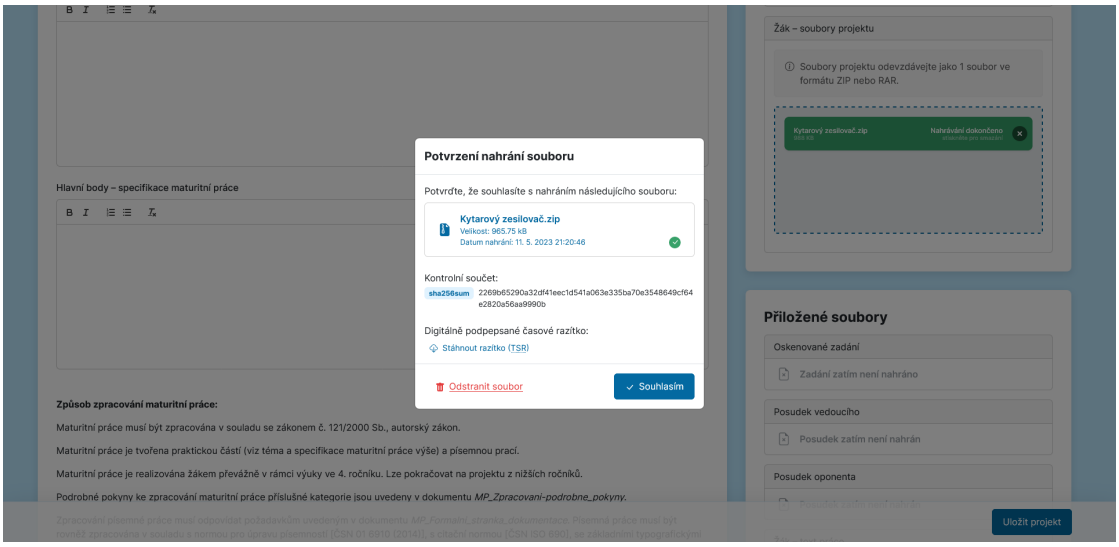
Obrázek C.4 Administrace školního roku – import uživatelů



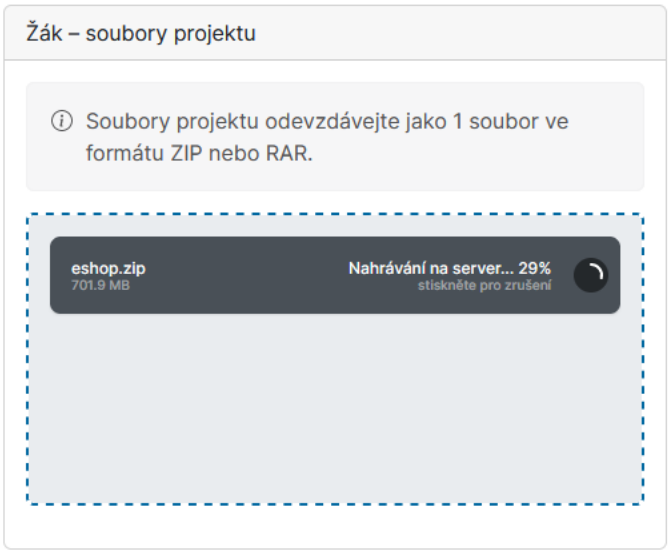
Obrázek C.5 Úprava zadání maturitního projektu a nahrávání souborů



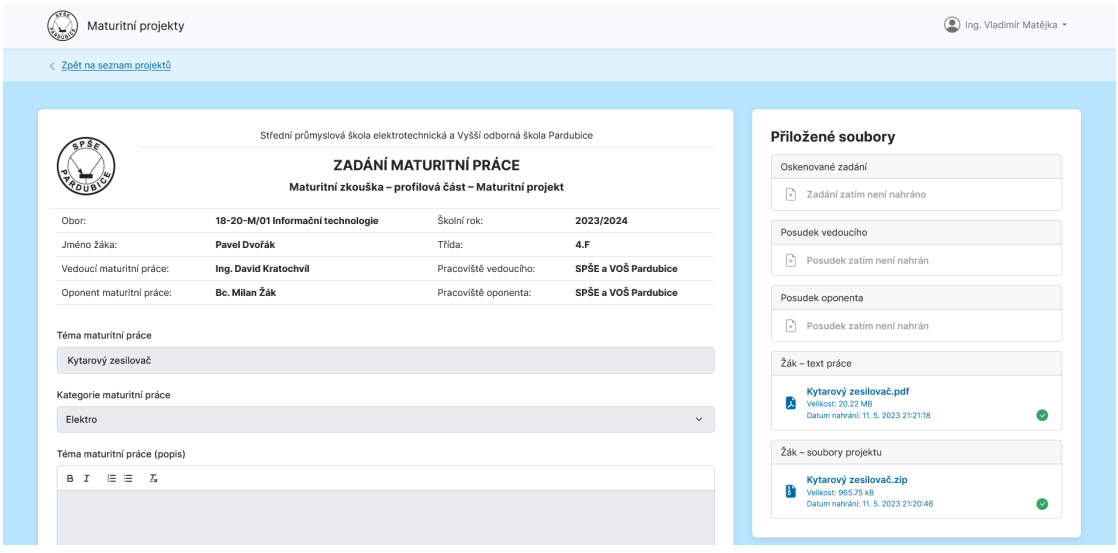
Obrázek C.6 Nahrávání souborů k projektu – překročení deadline



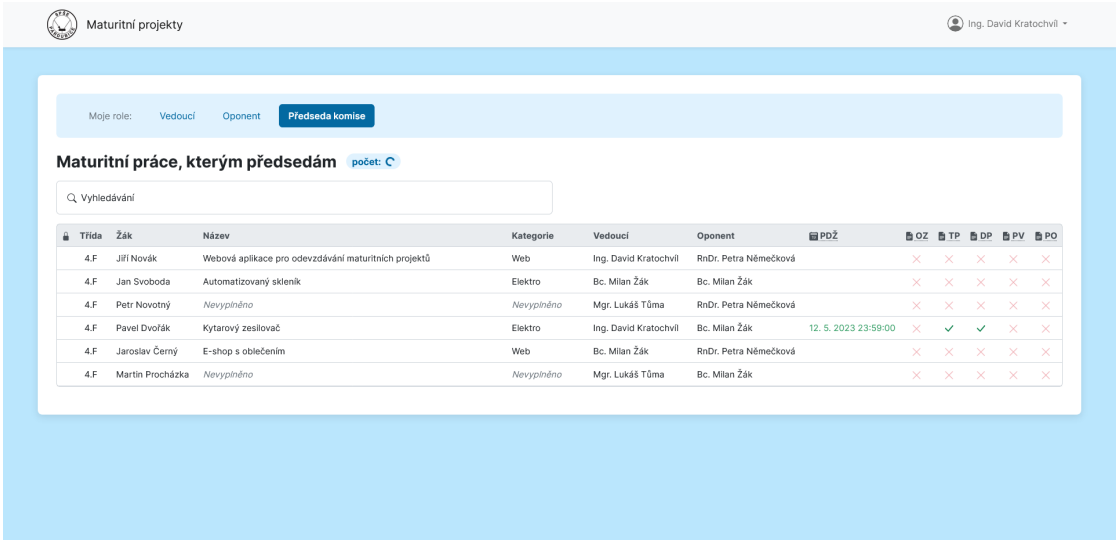
Obrázek C.7 Nahrávání souborů k projektu – schválení uživatelem



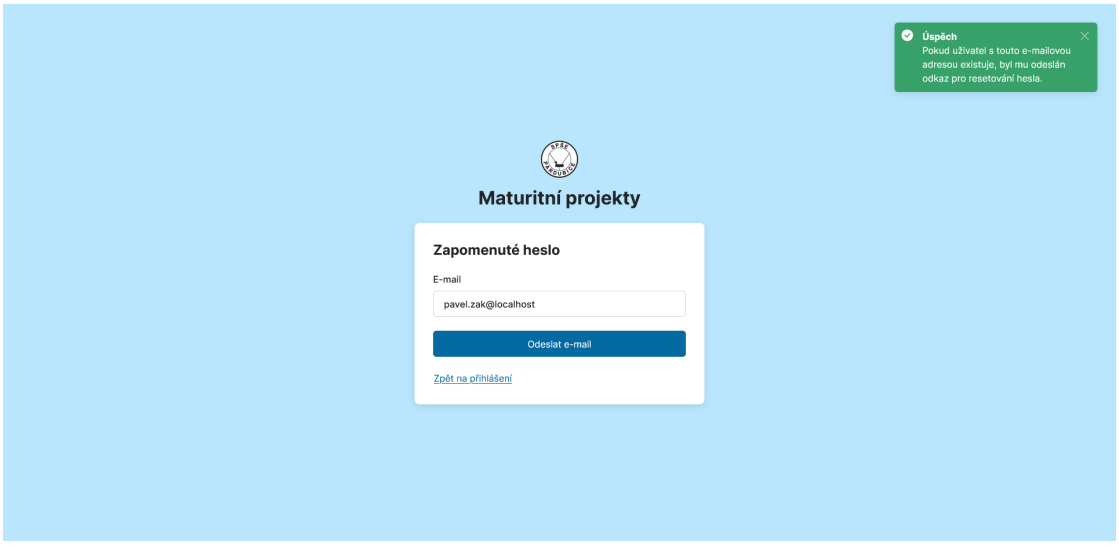
Obrázek C.8 Nahrávání souboru na server



Obrázek C.9 Nahrané soubory u projektu



Obrázek C.10 Nástěnka se seznamem projektů a přepínáním rolí



Obrázek C.11 Stránka pro obnovu zapomenutého hesla

Bibliografie

1. P.F. ART, SPOL. S R. O. *Vysoké školy, střední školy, ZŠ a jazykové školy v celé ČR* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://www.atlasskolstvi.cz/.
2. SŠIEŘ – ROŽNOV POD RADHOŠTĚM. *Správa maturitních prací* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://www.roznovskastredni.cz/matprac/.
3. SŠIEŘ – ROŽNOV POD RADHOŠTĚM. *Maturitní práce - obory IT a ME | RožnovskáStřední.cz* [online]. 2018-10-17. [cit. 2023-05-09]. Dostupné z: https://www.roznovskastredni.cz/maturitni-prace.
4. SPŠSE A VOŠ LIBEREC. *Dlouhodobé práce na SPŠSE a VOŠ Liberec* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://prace2.pslib.cloud/.
5. SPŠSE A VOŠ LIBEREC. *Maturitní a ročníkové práce | SPŠSE a VOŠ Liberec* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://pslib.cz/obory/technicke-lyceum/maturitni-a-rocnikove-prace-.
6. STŘEDNÍ PRŮMYSLOVÁ ŠKOLA A VYŠŠÍ ODBORNÁ ŠKOLA PÍSEK. *Dlouhodobé práce na SPŠSE a VOŠ Liberec* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://moo.sps-pi.cz/.
7. STŘEDNÍ PRŮMYSLOVÁ ŠKOLA A VYŠŠÍ ODBORNÁ ŠKOLA PÍSEK. *Maturitní a ročníkové práce | SPŠSE a VOŠ Liberec* [online]. 2022-04-07. [cit. 2023-05-09]. Dostupné z: http://wiki.sps-pi.cz/index.php/Maturitn%C3%AD_pr%C3%A1ce.
8. FAKULTA INFORMAČNÍCH TECHNOLOGIÍ, ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE. *ProjectsFIT* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://projects.fit.cvut.cz/.
9. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE, FAKULTA INFORMAČNÍCH TECHNOLOGIÍ. *Závěrečná práce • SZZ • FIT ČVUT Course Pages* [online]. 2023-05-05. [cit. 2023-05-09]. Dostupné z: https://courses.fit.cvut.cz/SZZ/prace/index.html.
10. UNIVERZITA KARLOVA. *Studijní informační systém – Témata prací (Výběr práce)* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://is.cuni.cz/studium/dipl%5C_st/.
11. UNIVERZITA KARLOVA, MATEMATICKO-FYZIKÁLNÍ FAKULTA. *Průvodce po bakalářské práci | Matematicko-fyzikální fakulta* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://www.mff.cuni.cz/cs/studenti/bc-a-mgr-studium/bakalarske-a-diplomove-prace/pruvodce-po-bakalarske-praci.
12. UNIVERZITA HRADEC KRÁLOVÉ. *Elektronická VysokoŠkolská Kvalifikační Práce – Univerzita Hradec Králové* [online]. [B.r.]. [cit. 2023-05-09]. Dostupné z: https://ris.uhk.cz/evskp/.

13. UNIVERZITA HRADEC KRÁLOVÉ. *Kvalifikační práce – Univerzita Hradec Králové* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://www.uhk.cz/cs/fakulta-informatiky-a-managementu/studium/statni-zaverecne-zkousky-a-kvalifikacni-prace/kvalifikacni-prace-bakalarska-diplomova.
14. FAKULTA INFORMAČNÍCH TECHNOLOGIÍ, VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. *Informační systém VUT* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://www.vut.cz/studis/student.phtml?sn=zav_prace_moje.
15. FAKULTA INFORMAČNÍCH TECHNOLOGIÍ, VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. *Státnice a bakalářské práce* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://www.fit.vut.cz/study/theses/bachelor-theses/.cs.
16. ARLOW, Jim; NEUSTADT, Ila. *UML 2 a unifikovaný proces vývoje aplikací: Průvodce analýzou a návrhem objektově orientovaného softwaru*. Brno: Computer Press, 2007. ISBN 9788025115039.
17. FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. *DISSERTATION* [online]. 2000 [cit. 2023-05-09]. Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.
18. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST Special Publication 800-63B (Digital Identity Guidelines)* [online]. National Institute of Standards a Technology, 2020-03-02 [cit. 2023-04-13]. Dostupné z: https://pages.nist.gov/800-63-3/sp800-63b.html.
19. ZUCCHERATO, Robert; CAIN, Patrick; ADAMS, Dr. Carlisle; PINKAS, Denis. *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)* [RFC 3161]. RFC Editor, 2001 [cit. 2023-04-13]. Request for Comments, č. 3161. Dostupné z DOI: 10.17487/RFC3161.
20. THE OPENSSL PROJECT AUTHORS. *openssl-ts, ts - Time Stamping Authority tool (client/server)* [online]. 2023. [cit. 2023-04-13]. Dostupné z: https://www.openssl.org/docs/man1.1.1/man1/openssl-ts.html.
21. MINISTERSTVO VNITRA ČESKÉ REPUBLIKY. *Přehled kvalifikovaných poskytovatelů certifikačních služeb a jejich kvalifikovaných služeb* [online]. Ministerstvo vnitra České republiky, 2023 [cit. 2023-04-13]. Dostupné z: https://www.mvcr.cz/clanek/prehled-kvalifikovanych-poskytovatelu-certifikacnich-sluzeb-a-jejich-kvalifikovanych-sluzeb.aspx.
22. CESNET, Z. S. P. O. *Časové značky - Informace pro uživatele* [online]. CESNET, z. s. p. o., 2021-08-03 [cit. 2023-04-13]. Dostupné z: https://pki.cesnet.cz/cs/tsa-overview.
23. DRAKE, Victoria. *Threat Modeling | OWASP Foundation* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://owasp.org/www-community/Threat_Modeling.
24. GOOGLE. *Google Authenticator OpenSource* [online]. 2021. [cit. 2023-04-13]. Dostupné z: https://github.com/google/google-authenticator.
25. OWASP FOUNDATION, INC. *OWASP Foundation, the Open Source Foundation for Application Security* [online]. 2023. [cit. 2023-04-13]. Dostupné z: https://owasp.org/.
26. JIM MANICO, Jakub Maćkowski a kol. *File Upload – OWASP Cheat Sheet Series* [online]. 2021. [cit. 2023-04-13]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html.
27. FIFIELD, David. *A better zip bomb* [online]. 2021-07-29. [cit. 2023-04-13]. Dostupné z: https://www.bamsoftware.com/hacks/zipbomb/.
28. META PLATFORMS, INC. *React* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://react.dev/.
29. META PLATFORMS, INC. *Writing Markup with JSX – React* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://react.dev/learn/writing-markup-with-jsx.

30. BOOTSTRAP TEAM. *Bootstrap · The most popular HTML, CSS, and JS library in the world.* [Online]. 2023. [cit. 2023-05-09]. Dostupné z: https://getbootstrap.com/.
31. SASS TEAM. *Sass: Syntactically Awesome Style Sheets* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://sass-lang.com/.
32. SASS TEAM. *Sass: Sass Basics* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://sass-lang.com/guide.
33. . *npm | npm Docs* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://docs.npmjs.com/cli/v9/commands/npm.
34. VERCEL, INC. *React Hooks for Data Fetching – SWR* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://swr.vercel.app/.
35. ZABRISKIE, Matt. *Getting Started | Axios Docs* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://axios-http.com/docs/intro.
36. ZENOAMARO. *zenoamaro/react-quill: A Quill component for React* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://github.com/zenoamaro/react-quill.
37. SLAB. *Quill – Your powerful rich text editor* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://quilljs.com/.
38. ZENOAMARO. *cure53/DOMPurify: DOMPurify - -a DOM-only, super-fast, uber-tolerant XSS sanitizer for HTML, MathML and SVG.* [Online]. 2023. [cit. 2023-05-09]. Dostupné z: https://github.com/cure53/DOMPurify.
39. SCHENNINK, Rik. *Easy File Uploading With JavaScript | FilePond* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://pqina.nl/filepond/.
40. LARAVEL LLC. *Laravel - The PHP Framework For Web Artisans* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://laravel.com/.
41. OTWELL, Taylor. *The Laravel Framework* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://github.com/laravel.
42. NILS ADERMANN, Jordi Boggiano a komunita. *Introduction – Composer* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://getcomposer.org/doc/00-intro.md.
43. LARAVEL LLC. *Eloquent: Getting Started – Laravel – The PHP Framework For Web Artisans* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://laravel.com/docs/10.x/eloquent.
44. DOMINIK BONSCH, Gabriel Bull a spol. *PHPOffice/PhpSpreadsheet: A pure PHP library for reading and writing spreadsheet files* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://github.com/PHPOffice/PhpSpreadsheet.
45. DOMINIK BONSCH, Gabriel Bull a spol. *PHPOffice/PHPWord: A pure PHP library for reading and writing word processing documents* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://github.com/PHPOffice/PHPWord.
46. *fgrosse/PHPASN1: A PHP library to encode and decode arbitrary ASN.1 structures using ITU-T X.690 encoding rules.* [Online]. 2023. [cit. 2023-05-09]. Dostupné z: https://github.com/fgrosse/PHPASN1.
47. OWASP FOUNDATION, INC. *Cryptographic Storage – OWASP Cheat Sheet Series* [online]. 2021. [cit. 2023-05-09]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html#uuids-and-guids.
48. *Fine Uploader Javascript Upload Library* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://fineuploader.com.
49. *Fine Uploader is shutting down · Issue #2073 · FineUploader/fine-uploader* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://github.com/FineUploader/fine-uploader/issues/2073.

50. LARAVEL LLC. *Authorization - Laravel - The PHP Framework For Web Artisans* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://laravel.com/docs/10.x/authorization.
51. PHP GROUP. *PHP: Introduction - Manual* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://www.php.net/manual/en/intro.openssl.php.
52. LARAVEL LLC. *Eloquent: Relationships – Laravel – The PHP Framework For Web Artisans* [online]. 2023. [cit. 2023-05-09]. Dostupné z: https://laravel.com/docs/10.x/eloquent-relationships#eager-loading.
53. OWASP FOUNDATION, INC. *OWASP Top Ten | OWASP Foundation* [online]. 2023. [cit. 2023-05-10]. Dostupné z: https://owasp.org/www-project-top-ten/.
54. OWASP FOUNDATION, INC. *A01 Broken Access Control - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A01_2021-Broken_Access_Control/.
55. OWASP FOUNDATION, INC. *A02 Cryptographic Failures - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/.
56. OWASP FOUNDATION, INC. *A03 Injection - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A03_2021-Injection/.
57. OWASP FOUNDATION, INC. *A04 Insecure Design - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A04_2021-Insecure_Design/.
58. OWASP FOUNDATION, INC. *A05 Security Misconfiguration - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/.
59. OWASP FOUNDATION, INC. *A06 Vulnerable and Outdated Components - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/.
60. *Security Issue CVE-2021-3163 · Issue #3364 · quilljs/quill* [online]. 2021-05-11. [cit. 2023-05-10]. Dostupné z: https://github.com/quilljs/quill/issues/3364.
61. *Help, 'npm audit' says I have a vulnerability in react-scripts! · Issue #11174 · facebook/create-react-app · GitHub* [online]. 2021-07-02. [cit. 2023-05-10]. Dostupné z: https://github.com/facebook/create-react-app/issues/11174.
62. OWASP FOUNDATION, INC. *A07 Identification and Authentication Failures - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/.
63. OWASP FOUNDATION, INC. *A08 Software and Data Integrity Failures - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/.
64. OWASP FOUNDATION, INC. *A09 Security Logging and Monitoring Failures - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/.
65. OWASP FOUNDATION, INC. *A10 Server Side Request Forgery (SSRF) - OWASP Top 10:2021* [online]. 2021. [cit. 2023-05-10]. Dostupné z: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_(SSRF)/.

Obsah přiloženého archivu

- `mp-backend` ..... adresář se zdrojovými kódy backendu
- `mp-frontend` ..... adresář se zdrojovými kódy frontendu
- `text` ..... text práce
  - `src` ..... zdrojové kódy práce ve formátu LaTeX
  - `thesis.pdf` ..... text práce ve formátu PDF