FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Webový informační systém pro studenty střední školy
Student:	Matěj Cajthaml
Vedoucí:	Mgr. Zbyšek Nechanický
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Pro účely výuky na střední škole vytvořte nový informační webový systém určený pro studenty středních škol pro různé předměty z oboru informačních technologií. Systém má být dostatečný pro předpokládané požadavky daných předmětů dle již používaných systémů a dalších požadavků dle vyučujícího. Systém musí tedy obsahovat hodnocení studentů, seznam a odevzdání prací, studijní materiály a další. Starý informační systém je špatně použitelný, rozšiřitelný a celkově byl špatně architektonicky navrhnut.

1. Proveďte analýzu dostupných informačních systémů, které jsou vhodné pro studenty středních škol, a vyhodnoťte jejich výhody a nevýhody.
2. Proveďte analýzu existujícího informačního systému, vyhodnoťte implementované funkcionality a nedostatky.
3. Navrhněte řešení, které zohlední výstupy analýzy.
4. Na základě návrhu implementujte a zdokumentujte webovou aplikaci s důrazem na možnost práce z mobilních zařízení.
5. Proveďte uživatelské testování výsledků, vyhodnoťte kvality, nedostatky řešení a diskutujte možnosti rozšíření aplikace.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 15. listopadu 2022 v Praze.

Bakalářská práce

WEBOVÝ INFORMAČNÍ SYSTÉM PRO STUDENTY STŘEDNÍ ŠKOLY

Matěj Cajthaml

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Mgr. Zbyšek Nechanický
10. května 2023

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2023 Matěj Cajthaml. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.

Odkaz na tuto práci: Cajthaml Matěj. *Webový informační systém pro studenty střední školy.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Nejprve bych chtěl poděkovat svému vedoucímu mé bakalářské práce, panu Mgr. Zbyškovi Nechanickému, za možnost pracovat na využitelném projektu a také za všechny jeho cenné rady a konzultace, kterých se mi během psaní a realizace práce dostalo.

Taktéž chci poděkovat všem svým přátelům a testerům, kteří mě v práci podporovali, a práci mi průběžně kontrolovali. Zejména panu Denisovi Lengerovi a slečně Kristýně Dřevikovské za jejich korekce a neustálou oporu. A také slečnám Morávkové a Kodešové a pánům Kalendovi, Hanzlikovi, Hájkovi, Hrbkovi, Šulcovi a Naxerovi za testování.

Velké díky patří i všem mým studentům, kteří mě zejména tento školní rok při práci podporovali, motivovali a v minulých letech a verzích systémů vždy předávali cennou zpětnou vazbu.

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 10. května 2023 Matěj Cajthaml

Abstrakt

Tato bakalářská práce se zaměřuje na návrh a implementaci webového informačního systému, který nahrazuje současné výukové platformy pro výuku technických předmětů na středních školách. Práce analyzuje již existující aplikace a navrhuje efektivnější řešení využívající progresivní webové aplikace společně s vlastním designem a vrstvením. Implementovaný systém se dělí na klientskou a serverovou část. Ten využívá jazyk TypeScript a komunikuje pomocí REST API. Práce je otestována primárně studenty školy, na které vyučuji a je doplněna dalším automatickým testováním.

Klíčová slova vzdělávání, školní informační systém, střední škola, webová aplikace, TypeScript, Vue, Express

Abstract

This bachelor thesis focuses on the design and implementation of a web-based information system that replaces current learning platforms for teaching technical subjects in high schools. The thesis analyzes existing applications and proposes a more effective solution using progressive web applications along with custom design and layering. The implemented system is partitioned into client and server parts. The system uses TypeScript language and communicates through REST API. The work is tested primarily by students of the school at which I teach and is accompanied by further automated testing.

Keywords education, school information system, high school, web application, TypeScript, Vue, Express

Seznam zkratek

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
CSS3	Cascading Style Sheets verze 3
CSV	Comma separated values
ČVUT	České vysoké učení technické
DI	Dependency injection
DOM	Document Object Model
DTO	Data Transfer Object
FIT	Fakulta informačních technologií
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTML5	HyperText Markup Language verze 5
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IS	Informační systém
JS	JavaScript
JSON	JavaScript Object Notation
JSX	JavaScript Syntax Extension
JWT	JSON Web Token
Less	Leaner Style Sheets
NoSQL	Not only SQL
npm	Node Package Manager
ODM	Object Document Mapping
ORM	Object Relation Mapping
SQL	Structured query language
REST	Representational State Transfer
Sass	Syntactically awesome style sheets
SEO	Search Engine Optimalization
SOAP	Simple Object Access Protocol
SPA	Single Page Application
SŘBD	Systém řízení báze dat
SSPŠ	Smíchovská střední průmyslová škola
SSR	Server-side rendering
ŠIS	Školní informační systém
TS	TypeScript

UI	User Interface
UX	User Experience
URI	Uinform Resource Identificator
URL	Uniform Resource Locator
WWW	World Wide Web
WYSIWYG	What you see is what you get
WYSIWYM	What you see is what you mean
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Úvod

Jednou z nejdůležitějších částí života člověka je studium. Školy studenta připravují jak na osobní, tak i na pracovní život. Nepřináší však pouze obecný či odborný přehled, ale i určitý řád, a především samostatnost studenta. Na 1280 českých středních školách[1] se ve školním roce 2020-2021 nacházelo okolo 430 tisíc studentů. K tomu, aby byl ve studiu řád, je nutné studentům poskytnout kvalitní a spolehlivý informační systém, který podporuje studenty v jejich studiu a samostatné práci.

Cílem práce je provést analýzu současné, mnou vytvořené aplikace a alternativních konkurenčních aplikací a hned poté navrhnout a implementovat webovou aplikaci. Práce navazuje na mou, již existující, webovou aplikaci, která se aktivně využívá, avšak momentálně není v nejlepším stavu a je architektonicky špatně navrhnuta.

Výstupem této práce by měl být nový funkční školní informační systém, který, jako webová aplikace, bude sloužit studentům a vyučujícím k lepší organizaci jejich studia a vyučování. Webová stránka by měla být v budoucnosti jednoduše rozšiřitelná tak, aby nové funkcionality neznamenaly přetvoření celé aplikace.

Toto téma je mi velmi blízké, jakožto vyučující na Smíchovské střední průmyslové škole a gymnáziu, jsem již od začátku potřeboval vytvořit jednoduchý systém pro správu výuky odborných předmětů. Systémy, které škola dosud využívá, a jejich alternativy, nejsou dle mého vhodné, a moji studenti s tím souhlasí. Systém bude primárně využíván na této škole, avšak může být využit jakoukoliv jinou školou či vyučujícím.

[1] dle Českého statistického úřadu [1]

Cíle práce

Cílem rešeršní části této bakalářské práce je provést analýzu stávajících informačních systémů využívaných ve školství, se zaměřením na jejich architekturu, funkčnost a uživatelskou přívětivost, a taktéž již existující aplikace. Dále bude rozebrána problematika vývoje webových aplikací a využití moderních technologií při tvorbě uživatelsky přívětivých a funkčních webových rozhraní. Součástí této části bude také průzkum mezi studenty, kteří budou hodnotit stávající systém a vyjadřovat své požadavky na nový systém. Výsledkem rešerše bude podrobná analýza, která bude sloužit jako východisko pro návrh nového informačního systému.

Praktická část práce bude zaměřena na návrh a implementaci nového školního informačního systému jako webové aplikace. Tato část bude rozdělena na čtyři podčásti. V první podčásti budou navrženy základní architektury a funkcionalita systému, včetně frontendové a backendové části. V druhé části bude provedena implementace návrhu a vytvoření webové aplikace. Třetí část bude zaměřena na testování a vyhodnocení funkčnosti a spolehlivosti nového školního informačního systému. V poslední části bude provedeno uživatelské testování a vysvětleno automatické testování.

Kapitola 1
Rešerše

Tato kapitola se zaměřuje na definování termínů a na různé technologie, které se používají v odvětví webového inženýrství. Jsou zde popsány a porovnány různé technologie, které jsou často používány pro tvorbu webových aplikací a budou v dalších částech práce zmiňovány.

1.1 Informační systém

Informační systém je celistvá aplikace [2], která slouží k záznamu a sběru informací pro potřeby uživatelů systému. Aplikace musí plnit podnikové cíle. Nadále se budu zabývat pouze školními informačními systémy, které se zaobírají věcmi, které souvisí se školstvím.

ŠIS často, mimo jednoduché administrativní části, obsahují další moduly, které slouží pro komunikaci mezi studenty [3], učiteli a jejich zákonnými zástupci. V práci se budu primárně zaměřovat na komunikaci mezi studentem a učitelem, zejména jako pomoc studentům se studiem konkrétní látky a předmětu. ŠIS by tedy měl studentům poskytovat:

- studijní materiály (prezentace, skripta a další)
- úlohy na procvičení probíraného tématu
- informace o předmětu
- hodnocení v předmětu
- domácí úkoly a další práce
- další zdroje učení
- a další

Tyto věci lze samozřejmě předávat i jinými způsoby, například staticky, ve formě různých souborových systémů či předávání přes další elektronickou komunikaci [3], např. e-maily. Tento přenos však není vhodný z důvodu velké nepřehlednosti [4] pro obě strany komunikace.

1.2 Frontend

Celosvětová síť internet propojuje dva druhy [5] počítačů:

server uchovává a nabízí data klientům

klient data přejímá a zobrazuje uživatelům

Webové stránky jsou internetové dokumenty [5], tedy soubory obrázků, fotografií a dalších objektů. Tyto dokumenty na sebe i na soubory odkazují pomocí hypertextových odkazů uvnitř World Wide Web. Zobrazování tohoto dokumentu na straně klienta zajišťuje webový prohlížeč. Webové stránky se píší v Hypertext Markup Language (HTML).

Webový vývoj má dvě hlavní podčásti [6], které se označují slovem frontend a backend. Frontend označuje část webové stránky na klientovi, která půjde zobrazit ve webovém prohlížeči. Vše, co se objeví na klientské části, je něco, s čím může uživatel interagovat. Backend je blíže popsán v kapitole 1.3.

1.2.1 Základní technologie

Webové stránky jsou svým vývojem vázány na níže popsané technologie. HTML a CSS jsou považovány za klíčové technologie [6], které nelze nahradit a neexistuje k nim žádná rozumná alternativa, která by byla implementována prohlížeči.

Hypertext Markup Language je značkovací jazyk [7], kde je každá značka řídícím příkazem k tomu, co se na stránce vykreslí pomocí webového prohlížeče. Nejmodernější a nejpoužívanější verzí HTML je jeho pátá verze, pojmenovaná HTML5, která je doporučena World Wide Web Consortium. Existují další minor podverze tohoto standardu.

Cascading Style Sheets je stylovací jazyk pro internetové dokumenty. V CSS pomocí pravidel určujeme, jak se jednotlivé prvky uvnitř HTML budou zobrazovat, včetně rozložení i vzhledu. Stejně jako HTML, je CSS vyvíjeno W3C. W3C je mezinárodní konsorcium [8], které se stará o různé webové standardy pro WWW. Cílem W3C je společně s veřejností rozvíjet WWW tak, aby zajistil dlouhodobý růst webu.

Skládá se z pravidel, která obsahují:

- **selektor**, kterým označujeme jednotlivé prvky na stránce, na které bude pravidlo aplikováno.
- **vlastnosti**, které určují vzhled či chování prvku. Určujeme jim hodnoty.

JavaScript je skriptovací, více paradigmový a dynamický jazyk [6], který dovoluje implementovat komplexní funkcionality na jakékoliv webové stránce. Slouží např. k zobrazování dynamického obsahu, map, pokročilé grafiky a dalších. Uvnitř JS máme ve webovém prohlížeči k dispozici Document Object Model (viz další bod).

Moderní JavaScript je jazykový dialekt, který implementuje standard ECMAScript [9]. ECMAScript je standard vytvořený neziskovou organizací European Computer Manufacturer's Association. Organizace si klade za cíl vytvořit víceúčelový programovací

jazyk. JavaScript se v moderní podobě nepoužívá pouze na webových stránkách, ale i mimo prohlížeč, např. na serverech (viz kapitola 1.3.3).

Document Object Model je multiplatformní rozhraní [6], které z webové stránky tvoří stromovou strukturu HTML. Každý uzel reprezentuje jeden objekt v internetovém dokumentu. DOM se používá pro programový přístup ke stránce. Tuto strukturu můžeme za běhu stránky modifikovat a upravovat.

1.2.2 Uživatelské prostředí

Uživatelské prostředí je systém umožňující interakci mezi uživatelem a počítačovým systémem. Toto rozhraní umožňuje uživateli ovládat a manipulovat s programy a aplikacemi pomocí grafického uživatelského rozhraní, textového uživatelského rozhraní nebo hlasových rozhraní.

Grafické uživatelské prostředí je forma uživatelského prostředí, která umožňuje uživateli interakci s počítačem pomocí grafických prvků, jako jsou ikony, tlačítka, okna a dialogová okna. GUI dává uživateli možnost vizuálně interagovat s počítačem, což umožňuje snazší ovládání programů a aplikací.

GUI je dnes nejrozšířenější formou uživatelského prostředí [10]. Důvodem jsou velké výhody, jako např. intuitivní ovládání, rychlost a přehlednost. GUI není omezeno pouze na desktopové aplikace, ale významně se podílí i na moderních webových stránkách. Webové GUI dovoluje uživatelům interagovat se stránkou. Skládá se z různých HTML značek, které dovolují pokročilé interaktivní procesy.

GUI se tvoří i pro webové aplikace, které lze zobrazovat na mobilních zařízeních. Dle statistiky [11] společnosti Statista bylo v roce 2023 více než 59 % všech internetových návštěv uskutečněno z mobilních zařízeních. To znamená, že GUI musí být navrhnuto tak, aby se přizpůsobilo všem zařízením. Na webové stránce se toto uzpůsobení tvoří pomocí responzivity, kterou se budu zabývat v kapitole 1.2.7.

1.2.3 Uživatelský zážitek

Uživatelský zážitek se týká celkového subjektivního dojmu, který má uživatel při používání produktu nebo služby. V této práci se zaměřuji pouze na uživatelský zážitek na webových stránkách. Tento zážitek může být ovlivněn mnoha faktory, jako je použitelnost, design, výkon, spolehlivost, přizpůsobitelnost a další.

Uživatelské rozhraní a uživatelský zážitek jsou dvě, úzce propojené, oblasti při vývoji produktů a služeb. Zatímco UI se týká vzhledu a funkcionality produktu, UX se soustředí na celkovou zkušenost uživatele při používání produktu.

Jak uvádí Don Norman, autor knihy „The Design of Everyday Things“:

„Přemýšlejte o akci v systému jako o části, jako o přirozeném a konstruktivním dialogu mezi uživatelem a systémem. Snažte se uživatele podporovat, ne bojovat s jeho chováním.“ [12] přeložil autor textu.

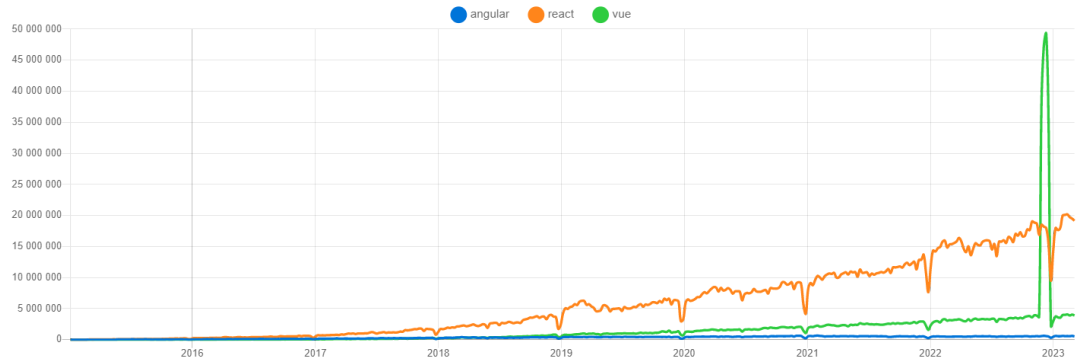
1.2.4 Framework

Framework je v informatice a softwarovém inženýrství termín, který se používá k označení struktury nebo abstrakce, která usnadňuje vývoj softwarových aplikací [6] tím, že

poskytuje základní stavební bloky, komponenty a šablony. Frameworky často obsahují definované postupy, které programátoři mohou použít při vývoji aplikace, aby se minimalizovala chybovost a zvýšila efektivita.

CSS frameworky jsou nástroje, které usnadňují návrh webových stránek a aplikací. Tyto frameworky poskytují sadu předdefinovaných stylů a komponent, které programátoři mohou použít pro rychlé a snadné vytváření stránek s konzistentním designem. Mezi nejpopulárnější CSS frameworky [13, 14] patří Bootstrap, Tailwind CSS, Foundation či Bulma.

JavaScript frameworky jsou nástroje, které usnadňují vývoj interaktivních a dynamických webových stránek a aplikací. Tyto frameworky poskytují sadu funkcí a nástrojů pro snadnou manipulaci s DOM a řízení chování stránky. Mezi nejpopulárnější JavaScript frameworky [14] patří React, Vue.js a Angular. Každý z frameworků se hodí na jiné projekty, ale v základu si jsou velmi podobné. Níže rozebírám výhody a nevýhody jednotlivých frameworků.



Obrázek 1.1 Graf počtu stahování frameworků Angular, React a Vue na platformě npm [15]

1.2.4.1 React

React je jedním z nejpopulárnějších [14] JavaScript frameworků díky své jednoduchosti, flexibilitě a výkonu. Jeho největší předností je použití virtuálního DOM (Virtual Document Object Model), které umožňují efektivní aktualizaci obsahu stránky bez nutnosti překreslování celé stránky. React také umožňuje snadné vytváření opakovaně použitelných komponent [16] a jednoduchou integraci s dalšími knihovnami a frameworky [17]. Mezi nevýhody patří například absence předdefinovaného systému routování a nutnost používání JSX syntaxe pro tvorbu uživatelského rozhraní. React svojí popularitou stále roste, viz obrázek 1.1

1.2.4.2 Vue.js

Vue.js je novější framework, který si získává stále větší popularitu díky své jednoduchosti, modularitě a snadnému použití. Vue.js dle autora [18] staví na dobrých částech Angularu. Mezi jeho přednosti patří například rychlá instalace a spouštění projektů, intuitivní API pro tvorbu uživatelského rozhraní a snadná integrace s existujícími webovými

stránkami [19]. Vue.js také umožňuje snadnou tvorbu opakovaně použitelných komponent a používá běžnou HTML syntaxi. Nevýhodou může být například menší komunita [17] a nižší podpora ze strany společnosti. Ze všech tří zmíněných frameworků je Vue.js nejméně výpočetně náročný [20].

1.2.4.3 Angular

Angular cílí na komponenty, stejně jako ostatní frameworky a autoři Angularu uvádí [21], že cílí jak na malé, tak i na velké aplikace. Na rozdíl od jiných populárních JavaScript frameworků, jako je React nebo Vue.js, Angular v poslední době ztrácí na popularitě. Vývojáři se často stěhují k jiným frameworkům, protože Angular nenabízí stejnou flexibilitu a snadnost použití jako jeho konkurenti.

Hlavním problémem Angularu je jeho složitost a dlouhá křivka učení, což znamená, že jeho použití může být časově náročné a obtížné. Kromě toho mnozí vývojáři také uvádí, že Angular je velmi rigidní a není jednoduché si ho přizpůsobit podle svých specifických potřeb. Dalším nedostatkem je také vysoká náročnost na výpočetní výkon [20], což může způsobit pomalou odezvu aplikace.

1.2.5 TypeScript

TypeScript je typově orientovaná nadstavba pro jazyk JavaScript (viz kapitola 1.2.1), která byla vyvinuta společností Microsoft. TypeScript přidává do jazyka statické typování [22], které je kontrolováno v době kompilace programu. TypeScript se kompiluje do jazyka JavaScript. Mimo typování přidává další funkcionality, které umožňuji vývojářům psát robustní kód s menším množstvím chyb a větší přehlednost.

Díky typování, které zajišťuje bezpečnost a úplnost kódu, je rychlejší vývoj i refaktorování kódu. Za největší nevýhodu TS se považuje [23] nutnost samotný kód kompilovat, což přináší určité zpomalování vývoje a to, že je vázaný na funkcionality JS tak, že když se změní (jak se již stalo např. s moduly [24]), je nutné jazyk změnit taktéž.

1.2.6 CSS preprocesory

CSS preprocesory jsou nástroj [6], které umožňují tvořit CSS pomocí jiného jazyka, který je kompilován s pomocí procesoru. Většina funkcionalit v preprocesorech jsou funkce, které nejsou v klasickém CSS k dispozici. Jedná se např. o:

vnořování dovoluje vkládat jednotlivá CSS pravidla do těla jiného pravidla, což způsobí spojení selektoru

mixins dovolují vytvářet funkce, které vrací CSS pravidla či vlastnosti. Tyto funkce můžeme volat na více místech.

proměnné dovolují uvnitř kódu využívat hodnoty, tedy možnost ukládat, modifikovat a číst různé hodnoty. Nové verze CSS již mají proměnné přímo zabudované v jazyce [25], jsou však omezené v používání funkcí.

CSS preprocesorů existuje mnoho, mezi nejznámější [6] se řadí níže uvedené. První dva popíši podrobněji.

- Sass
- Less
- Stylus
- PostCSS

1.2.6.1 Sass

Syntactically awesome style sheets je preprocesorový skriptovací jazyk, který se interpretuje nebo kompiluje do CSS. Obsahuje dvě syntaxe [6]:

SASS/Haml používá indentované odsazení, což znamená, že každý kódový blok je odsazen a nový řádek znamená rozdělení pravidla.

SCSS používá blokové odsazení, stejně jako CSS. Používá složené závorky k označení bloků. Jakékoliv zapsané CSS je uvnitř syntaxe SCSS validní.

1.2.6.2 Less

Leaner style sheets je dynamický preprocesorový stylovací jazyk, který se kompiluje do CSS. Při tvorbě byl ovlivněn jazykem Sass a ovlivnil novou syntaxi SCSS v Sass. Používá tedy podobnou syntaxi jako CSS.

1.2.7 Responzivní design

Responzivní design je způsob tvoření designu tak [6], aby byl zobrazitelný na různých velikostech zařízení a displejích. Vytváření takového designu, nejen na webových stránkách, je velice důležité. Na webových stránkách cílíme na stolní počítače (s různou velikostí displeje), tablety, mobilní zařízení a např. i hodinky.

Responzivní design tvoříme pomocí HTML a CSS. Určujeme, jak se prvky na stránce přeorganizují, zmenší, skryjí či zvětší vůči rozlišení obrazovky uživatele. Dle již zmíněné statistiky [11], je okolo 60 % všech připojení na internet tvořeno z mobilního zařízení. Je tedy nutné, aby stránky byly zobrazitelné a čitelné na všech zařízeních.

V HTML musíme určit speciální tzv. meta tagy [6], které určují nastavení zobrazení stránky, např. jak mají vypadat na zařízeních či jak má fungovat přibližování. Uvnitř CSS používáme media pravidla, na kterých určujeme pravidla, která se aplikují při platnosti podmínky. Pomocí media pravidel nastavujeme i další věci, např. omezení používání animací, dle preference uživatele.

1.2.8 Asynchronous JavaScript and XML

Asynchronous JavaScript and XML dovoluje webovým stránkám ve webovém prohlížeči komunikovat na pozadí s webovými servery [6]. Díky AJAX můžeme např. upravovat části stránky bez nutnosti celou stránku znova načítat. Dotazy je možné volat nezávisle na sobě, a i přes sebe.

Požadavky se volají v JS pomocí XMLHttpRequest objektů. Na odpověď požadavku čeká JS – po získání dat z něho můžeme např. pomocí DOM upravovat obsah stránky.

1.2.9 Nativní aplikace

Nativní aplikace jsou aplikace [6], které jsou vyvinuté na konkrétní platformu, jako je např. Android či iOS. Pro tyto aplikace používáme určené programovací jazyky, např. pro Android Javu, pro iOS Swift. Nativní aplikace poskytují vysokou rychlost, výkon a možnosti ovládání zařízení. Největší nevýhoda tvorby nativních aplikací pro více platforem je to, že je nutné mít pro každou platformu speciální projekt s jiným kódem a vlastnostmi.

V posledních letech se přistupuje k hybridním aplikacím, kde se používá určitý „wrapper“ [6], který zaobaluje jinou aplikaci – často webovou stránku – do nativní aplikace pro danou platformu. Tento pomocný software dává pomocí nativního kódu k dispozici další volání a API pro správu věcí, které jsou závislé na platformě. Mezi tyto wrappery patří např. React Native či Capacitor.

React Native více cílí na nativní část aplikace [26], než na samotné weby a používá framework React (viz kapitola 1.2.4.1). Capacitor naopak cílí na weby a obecně má problémy s výkonem a nativními částmi aplikace [27] – veškerá data jsou předávána do webové stránky.

1.2.10 WYSIWYG

WYSIWYG je zkratka pro anglický výraz „What you see is what you get“ [10], která popisuje způsob editace dokumentů s takovým zobrazením, jako bude to finální v produktu – jedná se např. o Microsoft Office produkty. Opakem jsou WYSIWYM editory, které popisují způsob editace dokumentu se zobrazením, které není u produktu finální – jedná se např. o zápis kódu HTML či XML.

1.2.11 Designové systémy

Designové systémy jsou grafické návrhy uživatelského rozhraní (viz kapitola 1.2.3), které je navrhnuto jako jednotný systém a navrhuje se vzhledem k uživatelům a aktuálním designovým trendům [28]. Systém navrhuje komponenty, jejich interakce a části, barvy, typografii a další chování. Designové systémy cílí na možnost opakovat daný design mezi více aplikací. Mezi nejznámější designové systémy patří Material design či Apple Human.

1.3 Backend

V návaznosti na definici frontendu v kapitole 1.2 si nyní definujeme backend. Backend je označení pro webový server [5], který poskytuje data (a případně generuje frontend). Frontend komunikuje s backendem pomocí protokolu HTTP [29].

1.3.1 Hypertext Transfer Protocol

Hypertext Transfer Protocol dovoluje provádět komunikaci mezi serverem a klientem [29] (ať se jedná o stolní počítač či mobilní zařízení). Používá client-server architekturu. Komunikace stojí na odesílání požadavků a přijímání odpovědí. HTTP verze 2 používá protokol TCP/IP [29].

1.3.1.1 Požadavek

Požadavek je zpráva, kterou klient odesílá serveru, obsahuje informace o tom, co klient požaduje od serveru. Je tvořen hlavičkou a tělem. Hlavička obsahuje informace o požadavku, tělo obsahuje data, která chce klient odeslat serveru. V hlavičce jsou uvedeny následující informace:

- Metoda – určuje, co se má provést na serveru. Každý server může rozlišovat různé metody jinak. Některé z nich jsou:
  - GET – Získat data
  - POST – Vytvořit data
  - PUT – Vložit data
  - PATCH – Upravit data
  - DELETE – Smazat data
  - OPTIONS – Získat informace o cíli
- Cíl („cesta na serveru“)
- Verze HTTP
- Hlavičky jako metadata

1.3.1.2 Odpověď

Odpověď je zpráva, kterou server odesílá klientovi, obsahuje informace o tom, co server poskytl klientovi. Je tvořena hlavičkou a tělem. Hlavička obsahuje informace o odpovědi, tělo obsahuje data, která server odeslal klientovi. V hlavičce jsou uvedeny následující informace:

- Verze HTTP (např. HTTPS/2.0)
- Kód stavu – určuje to, co se stalo s požadavkem. Dělíme do kategorií:
  - 1xx – Informační
  - 2xx – Úspěšné
  - 3xx – Přesměrování
  - 4xx – Chyba klienta
  - 5xx – Chyba serveru
- Popis stavu (např. OK, Not Found, Internal Server Error)
- Hlavičky jako metadata

1.3.1.3 Hypertext Transfer Protocol Secure

Hypertext Transfer Protocol Secure je bezpečná verze HTTP. Zabezpečení je zajištěno pomocí protokolu Transport Layer Security (TLS) nebo Secure Sockets Layer (SSL) a certifikátů [30]. HTTP Secure je založen na HTTP a používá stejné metody a kódy stavu.

1.3.2 Webové API

Application Programming Interface je sada definovaných protokolů, způsobů, nástrojů a funkcí [31], které umožňují různým programům vzájemnou komunikaci a interakci. API funguje jako prostředník mezi různými softwarovými aplikacemi, umožňující propojení, sdílení a přenos dat mezi nimi.

Web API je specifický typ API, který umožňuje programům komunikovat pomocí sítě internet a používá standardní webové technologie, jako jsou například protokoly HTTP a HTTPS [31], pro poskytování a získávání dat. Web API je stále populárnější a často se používá pro vytváření moderních aplikací, které potřebují přístup k různým datům a službám.

1.3.2.1 Simple Object Access Protocol

Simple Object Access Protocol je protokol používaný pro výměnu strukturovaných zpráv v distribuovaném prostředí [29], a to zejména v kombinaci s Web API pro zasílání a přijímání zpráv mezi různými aplikacemi a službami na internetu.

SOAP využívá XML formát (blíže popsáno v kapitole 1.3.4.2) pro reprezentaci dat ve zprávách. Každá zpráva SOAP se skládá ze tří hlavních částí [32]: z hlavičky, těla a případně z chybové zprávy. Hlavička obsahuje metadata o zprávě, jako například informace o verzi protokolu, autentizaci, autorizaci, časovém razítku a dalších důležitých informacích. Tělo zprávy obsahuje samotná data, která jsou přenášena mezi aplikacemi.

SOAP se stává méně populárním kvůli větší režii přenosu dat a složitosti protokolu [31], což znamená, že mnoho moderních webových služeb se spíše zaměřuje na použití REST API nebo jiných alternativních protokolů.

1.3.2.2 Representational State Transfer

Representational State Transfer je architektura webových služeb [31] [32], která se zaměřuje na jednoduchost, efektivitu a škálovatelnost. REST je založena na využití standardních HTTP metod k přenosu dat mezi klientem a serverem.

Zprávy v REST jsou koncipovány tak, aby reprezentovaly určitý zdroj a jeho stav, který je identifikován pomocí URI [32]. Tyto zprávy jsou obvykle přenášeny ve formátu JSON nebo XML (viz kapitola 1.3.4.1 a 1.3.4.2) a nemají žádnou pevně definovanou strukturu. Zprávy v REST jsou také velmi jednoduché a mohou být snadno kešovány, což umožňuje efektivní využití síťových prostředků.

Další důležitou vlastností REST je použití hypertextových odkazů k umožnění klientovi objevovat zdroje a pracovat s nimi. Tyto odkazy jsou obvykle vkládány přímo do odpovědi serveru a umožňují klientovi snadno najít další zdroje, které ho zajímají.

V porovnání se SOAP je REST mnohem jednodušší a flexibilnější [31], což umožňuje rychlou a snadnou integraci s jinými aplikacemi a službami. Když Webové API dodržuje požadavky architektury REST, označujeme ho RESTful [29]. Mezi nejdůležitější požadavky mimo jiné patří:

- používání standardních HTTP metod (GET, POST, PUT, DELETE) pro přenos dat mezi klientem a serverem

- každý zdroj má vlastní identifikátor URI, který by měl být pevně definovaný a konzistentní
- bezstavovost, což znamená, že každý požadavek obsahuje všechny informace potřebné k jeho vyřízení
- používání hypertextových odkazů pro snadné objevování zdrojů a interakci s nimi

1.3.3 Node.js

Node.js je asynchronní prostředí pro běh JavaScriptu [33] (viz 1.2.1). Node.js je řízen událostmi a je postaven pro tvorbu škálovatelných síťových aplikací. Byl poprvé vydán v roce 2009 a rychle se stal populárním mezi vývojáři.

Node.js používá engine V8 [32], který je také používán v prohlížeči Google Chrome [6], a umožňuje vývojářům psát kód v jednom jazyce jak pro frontend, tak pro backend. Díky své asynchronní povaze může Node.js zpracovávat mnoho požadavků najednou [33], což z něj činí ideální platformu pro tvorbu webových aplikací s velkým množstvím uživatelů. Node.js je jednovláknový jazyk [33] – není potřeba řešit vlákna a jejich komunikaci.

1.3.4 Formáty dat

Způsobů, jak mohou vypadat přenášená data tak, aby jim ze serveru rozuměl i backend, je spousta. Mezi nejpoužívanější patří JSON a XML [29].

1.3.4.1 JavaScript Object Notation

JavaScript Object Notation je jednoduchý formát pro výměnu dat [34], který se stal velmi populárním, zejména v kontextu webových aplikací a API. JSON formát je inspirován objektovou notací jazyka JavaScript, ale je nezávislý na platformě a lze ho použít skoro s jakýmkoliv programovacím jazykem [34].

Data v JSON jsou strukturována pomocí dvou základních typů: objektů a pole. Objekty jsou kolekcemi párových záznamů klíče-hodnoty, zatímco pole jsou seznamy hodnot oddělených čárkami. Hodnoty v JSON mohou být typu řetězec, číslo, logická hodnota (`true` nebo `false`), null, objekt nebo pole. JSON data jsou zapsána v textovém formátu a jsou snadno čitelná jak pro stroje, tak pro lidi.

1.3.4.2 Extensible Markup Language

Extensible Markup Language je značkovací jazyk používaný pro popis datových struktur a hierarchických datových modelů [35]. XML byl vyvinut s cílem poskytnout standardizovaný způsob značkování dat, což umožňuje snadnou výměnu informací mezi různými aplikacemi a systémy. XML vychází z konceptu značkovacího jazyka, který umožňuje popsat strukturu dat pomocí značek (tagů) a hodnot (atributů).

1.3.5 Databáze

Databáze je soubor dat, která jsou uspořádána tak, aby bylo možné je snadno vyhledávat a zpracovávat. Databáze se používají pro perzistentní ukládání dat, která jsou využívána

v různých aplikacích. Databází jsou myšleny i další softwarové nástroje, které umožňují ukládat a zpracovávat data. To označujeme názvem Systém Řízení Dat.

Typů databází existuje mnoho. Nyní se budu zabývat pouze těmi nejpopulárnějšími. Jedná se o takové databáze, které jsou postaveny na Structured Query Language. Mimo tyto ještě zmíním Not only SQL typy databází, které mohou být postaveny na jiných technologiích než SQL.

1.3.5.1 Relační databáze

Mezi nejznámější relační SQL databáze se řadí Postgres, MySQL či Microsoft SQL Server [36]. Tyto databáze jsou si podobné a všechny je možné použít na stejný způsob. Níže proto budu používat Postgres jako příklad.

Postgres je relační SŘBD, ve které je nutné, aby všechna data měla přesnou strukturu [36]. Tato struktura, včetně dat, která jsou v ní uložena, se nazývá tabulka. Jednotlivé záznamy uvnitř tabulky jsou poté řádky [36]. Jednotlivé sloupce tabulky definují strukturu a mají určený typ, název a další vlastnosti.

Postgres nabízí pouze vertikální škálovatelnost [36] a pomocí dalších pomocných nástrojů i horizontální škálování. Replikace i další škálování je však velmi komplexní.

1.3.5.2 Nerelační databáze

Mezi nejznámější nerelační SQL databáze se řadí MongoDB, Redis či CouchDB [37]. Tyto databáze si již nejsou podobné – všechny se používají jiným způsobem a pro různé účely. Níže budu popisovat pouze MongoDB, protože je nejvíce používaná.

MongoDB je nerelační SŘBD, ve které není nutné, aby všechna data měla přesnou strukturu [37]. Nejmenší jednotkou databáze je dokument. Dokumenty jsou uloženy v kolekcích. Kolekce jsou v obyčejných databázích označovány tabulkami. Dokumenty v kolekci nemusí mít stejnou strukturu.

Dokumenty uvnitř kolekce jsou uloženy v binárním formátu Binary JSON [37]. BSON je binární formát JSONu. BSON je výrazně efektivnější než JSON [38], protože umožňuje efektivně ukládat data na disk. Tento formát je totiž menší a je možná lepší komprese přímo na disku. BSON má k dispozici speciální datové typy, jako např. datum či binární data, která nejsou v jiných formátech k dispozici.

MongoDB nabízí horizontální škálování a je možné replikovat databázi na více strojů.

1.3.5.3 Mapování dat

Object-relation mapping (ORM) a Object-document mapping (ODM) je nástroj, který umožňuje převádět objekty z programovacího jazyka do relačních nebo nerelačních databází [37]. ORM a ODM jsou používány pro snadnější práci s databází, kdy často není nutné psát SQL dotazy.

ORM a ODM jsou používány v různých jazykových knihovnách. V JavaScriptu (případně TypeScriptu) se používá např. Mongoose (ODM pro MongoDB), Prisma (ORM i ODM pro různé databáze) nebo Sequelize (ORM pro různé databáze).

1.3.6 Webové servery

Webové servery jsou aplikace, které poskytují webovou službu [29] pomocí protokolu HTTP. Tyto servery jsou zodpovědné za přijímání a zpracování HTTP požadavků, které jsou posílány od klientů a za poskytování odpovědi na tyto požadavky.

1.3.6.1 Express

Express je populární knihovna [32] pro tvorbu webových serverů v JS. Tyto servery zjednodušují tvorbu webových aplikací poskytováním různých funkcionalit a nástrojů, jako je správa HTTP požadavků a odpovědí, routování, middleware a další.

Middleware je klíčová funkcionalita, na které Express staví. Jsou to funkce, které mají přístup k HTTP požadavku a odpovědi. Middleware může zpracovat požadavek, změnit požadavek a odpověď, nebo dokonce ukončit cyklus požadavku a odpovědi. Používají se pro sjednocení opakujícího se kódu v jednotlivých endpointech.

Webové servery postavené na Express jsou velmi oblíbené kvůli své rychlosti, škálovatelnosti a jednoduchosti vývoje.

1.3.6.2 Fastify

Fastifiy je knihovna pro tvorbu webových serverů JS, která se inspiruje z dalších frameworků, jako je například Express (viz kapitola 1.3.6.1) a Koi.js. Fastify zejména cílí na minimalizaci prodlevy odpovědí na požadavky [39].

Jako Express, Fastify staví na middlewarech a moderních standardech ECMAScript. Webové servery tvořené pomocí Fastify jsou velmi rychlé a podporují řadu knihoven. Mezi nevýhody patří malá uživatelská komunita [40].

1.3.6.3 Koa

Koa je knihovna, která se zaměřuje na tvorbu webových serverů v Node.js. Koa byla vytvořena týmem, který vytvořil Express [39], a soustředí se spíše na méně náročný framework, který neobsahuje nepodstatné části (jako např. předinstalované knihovny).

1.3.7 Autentizace

Autentizace je proces zjišťování, zda je uživatel opravdu ten, za kterého se vydává [29]. Pod procesem si lze představit více akcí. Základem autentizace je přihlášení, které zprvu validuje uživatele pomocí identifikátorů a dalších faktorů. Jedná se často o kombinaci identifikátoru a hesla, potvrzení biometrických údajů či např. čipové karty. Validace často probíhá na více fází – jedná se o tzv. více faktorové přihlášení.

Autentizace uživatele pro provádění jakékoliv akce není dobrá – pro uživatele je nemožné se při čemkoliv znovu přihlašovat. Proto existují různé způsoby si zapamatování přihlášeného uživatele – stavová a bezstavová autentizace.

1.3.7.1 Stavová autentizace

Stavová autentizace je nejpoužívanější systém ukládání autentizace [32]. V tomto systému si server sám musí pamatovat veškeré informace o přihlášených uživatelích. Data

se ukládají v tzv. sessions. Klient si však musí pamatovat identifikátor pro tato data (často v cookies, localstorage či jiném úložišti na klientovi), kterými se autentizace potvrzuje.

1.3.7.2 Bezstavová autentizace

Bezstavová autentizace je založena na procesu, kdy při přihlášení je klientovi vytvořen tzv. token, který si do podobných úložišť ukládá. Token je často podepsaný [32, 29] a uchovává různá data o uživatelích či např. jejich oprávnění. Jako token se například používá JSON Web Token.

1.3.7.3 Hašovací kryptografická funkce

Hašovací funkce je funkce, kterou používá algoritmus pro transformaci dat do jiných nečitelných dat [39]. V tomto případě je transformace jednosměrná a nelze ji provést opačnou stranou (naopak funguje šifrování). Server často pro kontrolu, zda jsou data stejná, provede hašování nehašovaných dat, která potom porovná.

Tyto funkce jsou často velmi náchylné [39] na brute-force útoky. Proto se do hašování často přidává sůl, která je umístěna na začátku dat. Mezi nejznámější hašovací funkce v praxi se řadí např. SHA1, Argon2 či bcrypt.

1.3.8 Autorizace

Autorizace je proces ověření, zda uživatel (autentizovaný či nikoliv) má oprávnění k určité akci [29, 39]. Pro autorizaci je nutné vědět, kdo se snaží akci provést (kdo je uživatel, jaké má práva, jaký má vztah s daty), co je to za akci (upravit, přečíst, vytvořit, ...) a k jakým datům se přistupuje.

Autorizace se často seskupuje do rolí či skupin [29], které mají určitá práva k určitým typům dat. Ty mohou být definované buď pevně (pro změnu je nutné konfigurovat zdrojový kód či konfigurační soubory) či mohou být nastavitelné administrátory.

1.3.9 Cross-Origin Resource Sharing

Cross-Origin Resource Sharing je mechanismus [41], který závisí na HTTP hlavičkách, a dovoluje serverům nastavovat ostatní zdroje, které považuje za bezpečné. Bezpečné zdroje poté dovolují prohlížečům, které tyto hlavičky kontrolují, povolit načítání položek i z jiných zdrojů, než je ten, který je určuje.

CORS se zasílají při „předběžném požadavku“, často metodou `OPTIONS`, na který server odpovídá s informacemi [41], které popisují, co se na endpointu s danými hlavičkami (a například autentizací a autorizaci) může provádět a určuje i CORS hlavičky. Klasické a moderní prohlížeče CORS kontrolují tak, aby se zamezilo nežádanému získávání informací z jiných zdrojů.

Kapitola 2
Analýza

Tato kapitola se zabývá analýzou jak již existující aplikace, tak i jejích alternativních řešení. Cílem je zjistit požadované funkcionality aplikace a určit další požadavky, které tvorbu práce definují. V kapitole se taktéž určují data, se kterými aplikace bude pracovat.

2.1 Existující řešení

V této kapitole se popisují již existující řešení zadání práce, jedná se jak o alternativní aplikace, které by šly k řešení využít, tak o již existující reálnou aplikaci, která je mým dílem.

2.1.1 Alternativní aplikace

Na trhu existuje spousta alternativních řešení, která fungují jako informační systémy, či studijní aplikace. V práci se zaměřím na 3 nejznámější aplikace, které se používají jako ŠIS a vyhodnotím jejich použitelnost s požadavky tvořené aplikace.

Tyto aplikace neslouží ke zcela stejným účelům, které jsou cílem této práce (viz kapitola 1.1). Jedná se o informační systémy, které částečně disponují nějakými funkcionalitami, které bychom od studijní aplikace v nějakém měřítku očekávali. Tyto implementované funkcionality ale často nejsou dostatečné pro různé specifické požadavky škol a předmětů.

2.1.1.1 Bakaláři

Bakaláři jsou školní informační systém určený pro správu všech částí vzdělávacího procesu [42]. Jedná se o nejpoužívanější informační systém a používá ho odhadem 60 % [42] českých škol. Tento systém poskytuje spoustu funkcionalit, avšak jsou velmi často limitované a nebo nastavitelné pouze pro celou školu. Například domácí úkoly (či obecně práce) lze zadávat pouze pro určitou studijní skupinu uvnitř systému, což jsou často třídy či dělené skupiny. Pro zadávání úkolů napříč třídami a skupinami je nutné úkoly duplikovat, což nepomáhá k hodnocení ze strany vyučujícího.

Velmi velký problém je i například hodnocení. Systém Bakaláři podporuje vytváření známek, avšak nedovoluje jakékoliv textové hodnocení či například osobní poznámky k této známce. Většina vyučujících je tedy nucena si informace o známkách s poznámkami uchovávat mimo tento systém, a to vede k duplikaci dat.

Tento systém není pro většinu předmětů vhodný z důvodů výše uvedených. Systém není vhodný pro samostudium a obecné sdílení studijních materiálů. Systém je tedy nutné používat s další aplikací (viz kapitola 2.1.2), či předávat materiály jinými způsoby. Vysoká použitelnost této aplikace stojí zřejmě na tom, že obsahuje např. modul matriky a je tedy velmi efektivní v ukládání dat, která škola musí uchovávat. Je taktéž velmi dobrý v ukládání třídní knihy a absence.

2.1.1.2 Edupage

Edupage je celosvětově používaný ucelený školní systém [43] pro správu školních procesů. Obsahuje různé moduly, jako je rozvrh, školní matrika, třídní kniha, e-learning a další. Stejně jako systém Bakaláři má tato aplikace problém s komplexním nastavením hodnocení známek, kdy nedovoluje tvořit kritéria dané známky. Systém nepodporuje dělení známek do kategorií.

Mezi velké výhody Edupage patří jednotlivé moduly. Mezi nejoblíbenější moduly se řadí e-learning, který dovoluje definici příprav, možnost samostudia a pokročilé sdílení materiálů. V aplikaci lze tvořit prezentace, kvízy a další zadání k procvičování.

2.1.1.3 Moodle

Moodle je open-source modulový informační systém [44] pro správu vzdělávací procesů a pro elektronické kurzy. Jedná se o velmi používaný systém, který se na základě pluginů, které může kdokoliv implementovat, může přizpůsobit jakýmkoliv požadavkům a potřebám předmětů.

Systém nabízí možnost vytváření interaktivních kurzů a materiálů. Systém obsahuje možnost zadávat úkoly, vytvářet skripta, zkoušení a spoustu dalších věcí již v základu. Mezi nevýhody systému se řadí právě jeho ohebnost na určité požadavky, které je často nutné suplovat moduly, které ale zase způsobují chaos při jejich použití mezi stránkami a kompatibilitou. Protože je systém stavěn pro různé úrovně vzdělávání a musí pokrývat velké množství možností, které dělají systém zbytečně složitý.

2.1.2 Kumulace aplikací

Z mých zkušeností školy využívají více ŠIS, které svými funkcionalitami uspokojí požadavky jednotlivých předmětů. Kumulace aplikací má však velmi mnoho nevýhod. Největší nevýhodou dle vedení školy[1] SSPŠ, je to, že samotný student je velmi zmatený, v jaké aplikaci má právě danou věc hledat. Může nastat i to, že jednotlivé předměty používají unikátní aplikace právě pro daný předmět a tím pádem nastává ještě větší konfúze.

Mezi aplikacemi je student nucen se přihlašovat a udržovat si relaci, což získávání informací prodlužuje. Počet aplikací je dobré regulovat, ale je samozřejmé, že jeden

[1] dle interní porady vedení, která byla předložena pedagogickému sboru SSPŠ

IS nemůže vyřešit veškeré požadavky předmětů a vždy bude existovat nějaká výjimka. Je však nutné co nejvíce systémy sjednotit.

2.1.3 Existující aplikace

Pro účely mé výuky technických předmětů jsem se již při mém prvním roce v roli učitele rozhodl vytvořit informační systém, který poskytoval studentům informace o předmětech jako takových, společně s dalšími informacemi. Škola používá více ŠIS a ani jeden z nich nebyl dostatečný pro kritéria daného předmětu. Například se jedná o používání bodového systému a výše uvedených věcí. Nastavení používaných ŠIS školy není dostatečné pro to, aby body mohl přehledně zobrazit i student. Kdybych aplikaci nevytvořil, byl bych nucen používat místo této aplikace pět dalších informačních systémů.

2.1.3.1 Historie

Jak se později ukázalo, ani tento systém nebyl dostatečný, a proto vznikly další dvě verze. Každá verze vyšla půl roku po té původní. Poslední verze se drží aktivní již přes dva roky. V druhé verzi přišlo samotné přihlašování a více interaktivní stránka. Aktuální verze je navržena tak, aby se mohly přidávat věci postupně. A to se taky děje, kdykoliv, kdy je potřeba něco nového, snažím se danou funkcionalitu přidat. Aplikace již slouží jako zcela samostatný ŠIS, který byl schválen ředitelem školy a prozatím se používá jenom na předměty, které učím já.

2.1.3.2 Implementované funkcionality v aktuální aplikaci

Nový ŠIS, který je cílem této práce, by měl vyřešit problémy stávající aplikace. Nová aplikace by měla zajisté implementovat již existující funkcionality a připravit lepší zázemí pro možné budoucí. Jednotlivé funkcionality nyní podrobně popíši. Zaměřím se na funkcionality, které vidí studenti, jsou tedy nejdůležitější:

Přihlášení uživatelé (studenti, rodiče a učitelé) se mohou přihlašovat pomocí školního e-mailu. Na tento e-mail jim přijde kód, který do aplikace zadají a tím se přihlásí.

Práce studenti mohou zobrazit zadané práce v předmětu a mohou je odevzdávat přímo na stránce. Své odevzdání mohou brát zpět. Práce jsou zadané vždy nějaké skupině a mají časový limit. Práce mohou být povinné či nepovinné celkově pro všechny skupiny.

Hodnocení vyučující mohou zapisovat různé typy známek (slovní a bodové). Tyto známky si poté může student zobrazit ve svém osobním profilu. Student vidí různé statistiky a může zobrazit detail jednotlivých známek.

Materiály studenti si mohou zobrazit seznam materiálů pro daný předmět. Jedná se o prezentace, řešené úlohy, odkazy a například myšlenkové mapy.

Samostudium v aplikaci jsou dva moduly pro samostudium studentů. První jsou knihy – jedná o textová skripta popisující určitou látku za pomocí textových odstavců, kódu a obrázku. Druhý jsou otázky, kde si student může z předpřipravených otázek

opakovat látku. Mezi otázkami může různě procházet. Jak ke knihám, tak k otázkám si uživatel může zapisovat poznámky, které si může později zobrazovat. Poznámky vidí pouze daný uživatel.

2.1.3.3 Hodnocení studenty

V listopadu roku 2022 jsem studenty, které vyučuji, požádal o vyplnění krátkého dotazníku, který se zaobírá stávající aplikací. Dotazník mohlo vyplnit 101 studentů, reálně ho vyplnilo 64.

Dotazník se zaobíral různými otázkami, zejména se jedná o otázky, zda portál potřebuje nové funkcionality, a jeho celkové hodnocení. Z výsledků anket vyšlo najevo, že studenti jsou se systém spokojení, a že požadují nové funkcionality. Tyto funkcionality však do aktuálního systému není možné jednoduše implementovat, protože by bylo nutné celý systém přetvořit. Mezi požadované funkcionality patří např. mobilní aplikace, rychlejší přihlašování, archiv předmětů a další.

Všechny otázky z výše uvedeného dotazníku a jejich odpovědi jsou k dispozici v souboru `/test/questionnaire.csv` ve formátu CSV. Na odpovědích, ve kterých jsou použity hvězdičky („*******“) proběhla anonymizace údajů a dalších poznámek, které by mohly být použity pro rozpoznání respondentů.

Během minulých let jsem v každém dotazníku pro sebereflexi nad výukou měl podobné otázky ohledně tohoto ŠIS. V tabulce 2.1 se nachází přehled hodnocení dle jednotlivých dotazníků z pololetí od školního roku 2020 na otázku „Webový portál se mi zdá užitečný“. Hodnotilo se škálou od jedné do čtyř, kdy je jedna nejlepší a čtyři nejhorší.

Tabulka 2.1 Výsledky z dotazníkových šetření na otázku „Webový portál se mi zdá užitečný“

Pololetí	Průměrné hodnocení	Odpovědí	Možných odpovědí
1. pololetí 2020-2021	1,34	27	34
2. pololetí 2020-2021	1,12	25	34
1. pololetí 2021-2022	1,15	55	68
2. pololetí 2021-2022	1,19	66	68
1. pololetí 2022-2023	1,07	71	95

2.1.3.4 Nedostatky aplikace

Každá verze aplikace byla navrhnuta velmi laicky a všechny byly vytvořeny za spěchu. Hlavním cílem tedy nebyl návrh a rozšiřitelnost – na tom aplikace aktuálně nejvíce zaostává. Cílem je tedy mít architektonicky udržitelnou aplikaci. Cílem je i implementovat nové funkcionality a nebo ty stávající vylepšit. Jednotlivé požadované změny jsou níže podrobně popsány. Nezmíněné funkcionality budou zachovány.

Lekce velkým problémem aktuální aplikace je to, že jednotlivé materiály k látce jsou rozdělené do spousty záložek a souborů. Pokud chce student studovat jednu látku, je nucen po aplikaci procházet různými stránkami. V nové aplikaci by měly být tyto materiály seskupeny do lekcí. Lekce bude obsahovat interaktivní prezentace, materiály, samostudia (otázky a skripta, viz výše), myšlenkové mapy, řešené úlohy a další.

Přístup k aplikaci přístup k aktuální verzi aplikace není omezen. Všechny předměty, materiály a další části webu jsou přístupné všem uživatelům, kteří k webové aplikaci mohou pomocí WWW přistoupit. Je velmi omezující mít všechny předměty k dispozici veřejně, a proto by mělo být v nové verzi aplikace možné jednotlivé předměty skrývat a nebo omezovat tak, aby k nim měl přístup pouze přihlášený uživatel a nebo student.

Cílem však není skrývat možnost studia dané látky před všemi. Proto by si mělo jít nově v aplikaci vytvořit účet a tím získat přístup k těmto částem webu. Každý uživatel si poté bude moci zapisovat poznámky a další věci. Studentům budou účty tvořeny automaticky na začátku daného školního roku a v případě, že již účet mají, budou jim přidána daná přístupová práva.

Hodnocení hodnocení v aktuální verzi aplikace je velmi omezené tím, že podporuje pouze bodový systém. Pokud má být systém připraven na více předmětů a různá kritéria, musí podporovat různé typy hodnocení například procentní, obyčejné známkové, nebo slovní.

V hodnocení je taktéž problém se známkami, které nesouvisí s aktuálním klasifikačním obdobím, ale sumarizují jiné období. V termínech středoškolského studia se jedná o čtvrtletní a pololetní známky. Tyto známky se nepočítají do aktuální známky, a proto musí být ze systému „izolovány“. V původní aplikaci není možné tyto známky vůbec tvořit.

2.2 Seznam požadavků

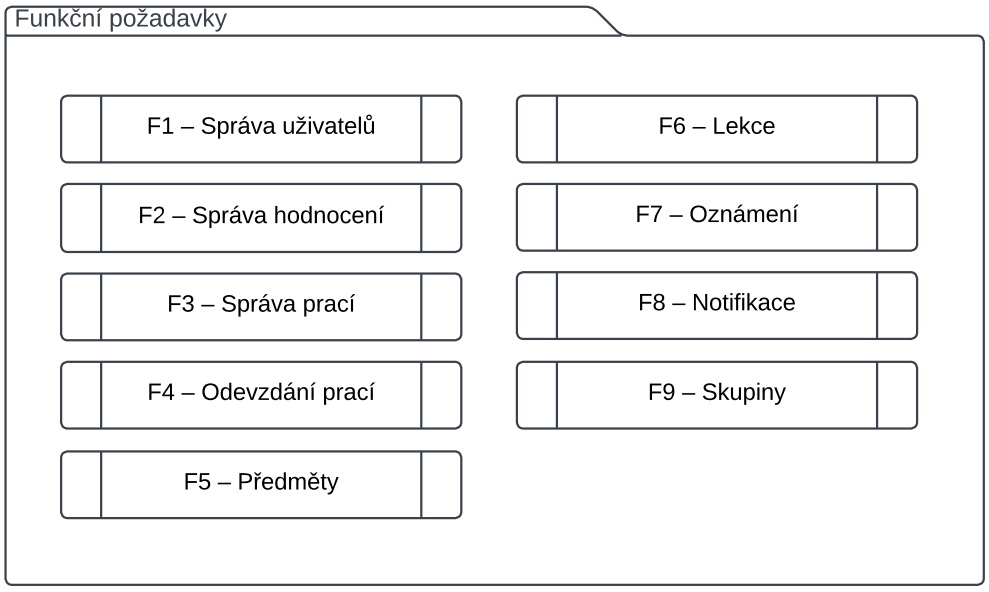
Požadavky určují, jaké funkcionality [45] a chování očekáváme od budoucí aplikace. Dále se dělí na funkční a nefunkční požadavky. Požadavky jsou důležité pro to, aby mohla být aplikace navržena pro splnění všech požadavků.

2.2.1 Funkční požadavky

Funkční požadavky stanovují samotné funkcionality [45], na které poté navazujeme uživatelskými případy (viz kapitola 2.3), ty definují použití aplikace uživateli za pomocí níže zmíněných funkcionalit. Jednotlivé funkční požadavky, které byly pro tuto aplikaci nalezeny, lze nalézt v diagramu na obrázku 2.1.

F1 – Správa uživatelů požadavek představuje možnost mít v aplikaci vytvořený účet. Jedná se jak o studentské, učitelské tak i návštěvnické účty. Účet bude mít mimo jiné i přihlašovací údaje, kterými se může uživatel přihlásit. Studentské role a skupiny může spravovat správce systému a případně i učitel.

F2 – Správa hodnocení v aplikaci bude možné tvořit hodnocení pro jednotlivé studenty předmětů. Aplikace musí podporovat různá kritéria hodnocení – od bodového hodnocení, přes klasickou stupnici známek, až po procenta či slovní hodnocení. Hodnocení může zapisovat učitel daným studentům. Jednotlivé hodnocení (známky) a celkové hodnocení půjde zobrazit učitelem. Systém musí podporovat i neklasifikace



Obrázek 2.1 Diagram funkčních požadavků

(neodevzdaná práce) a podporovat různé hranice, např. počet nutných známek v kategorii. Student předmětu si může zobrazit své hodnocení v daném předmětu a vidí celkové hodnocení. Jednotlivé hodnocení může podrobně zobrazit.

F3 – Správa prací aplikace dovoluje učitelům tvořit práce, které mohou studenti zobrazit. Tyto práce mají možnosti zobrazení pro určité skupiny a mají časový limit. Práce mohou být pro různé skupiny nastaveny separátně, lze tedy, pro určité skupiny nastavit povinnost a např. termín odevzdání.

F4 – Odevzdávání prací aplikace dovoluje učitelům u některých prací nastavit možnost odevzdávání. Student poté může práci odevzdávat jako soubory přímo na stránce aplikace. Své odevzdání může vzít zpět. Učitel tato odevzdání vidí a může hodnotit. Hodnocení je automaticky propsáno do celkového hodnocení (známky).

F5 – Předměty aplikace umí uchovávat neomezené množství předmětů. Jednotlivé předměty mají své studenty, kritéria hodnocení, hodnocení, materiály a další. Některé předměty mohou být veřejné, přístupné pro přihlášené a nebo jen pro studenty.

F6 – Lekce aplikace uchovává informace o lekcích, ve kterých se nachází jednotlivé materiály pro studium jednotlivých témat. Jako materiály lze ukládat odkazy, soubory, prezentace (v různých formátech), skripta a například otázky. Lekce dovolují studentům si uchovávat poznámky, ke kterým se mohou zpětně vracet.

F7 – Oznámení aplikace udržuje informace o oznámeních, které jsou dvojího typu. Krátkodobá oznámení jsou oznámení, která jsou určena např. písemným zkoušením,

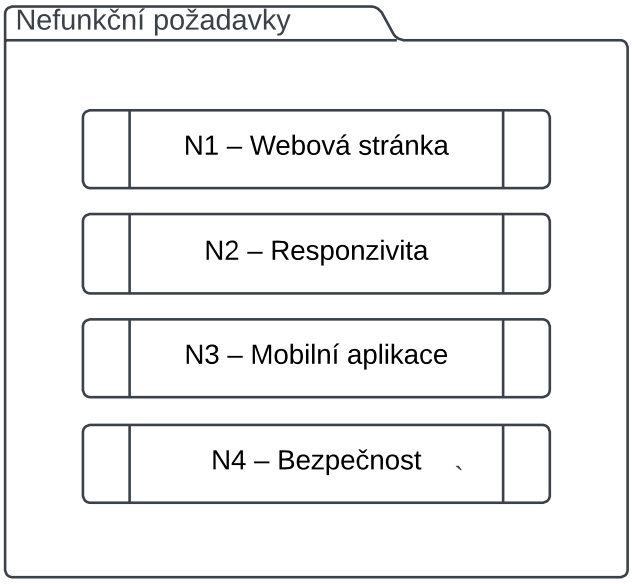
nebo různá oznámení změn. Dlouhodobá jsou oznámení, která platí celý školní rok a jedná se např. o informace ke kritériím hodnocení.

F8 – Notifikace aplikace při různých akcích, jako např. při novém hodnocení, zadání úkolu a podobně zasílá push notifikace na zařízení, na kterých se uživatel aplikace přihlásil k odběru.

F9 – Skupiny aplikace udržuje informace o skupinách, které reprezentují skupiny studentů, které studují jeden či více předmětů. Jednotliví uživatelé mohou být přiřazeni neomezeně mnoha skupinám. Skupiny jsou přiřazeny předmětům, které studují.

2.2.2 Nefunkční požadavky

Nefunkční požadavky jsou takové požadavky [45], které přímo nenavazují na cíl aplikace, ale definují se proto, aby aplikace byla do budoucnosti udržitelná. Jednotlivé nefunkční požadavky, které byly pro tuto aplikaci nalezeny, lze nalézt v diagramu na obrázku 2.2.



Obrázek 2.2 Diagram nefunkčních požadavků

N1 – Webová stránka klientská část aplikace poběží jako webová aplikace, která bude napsána ve validním kódu HTML, CSS a JS.

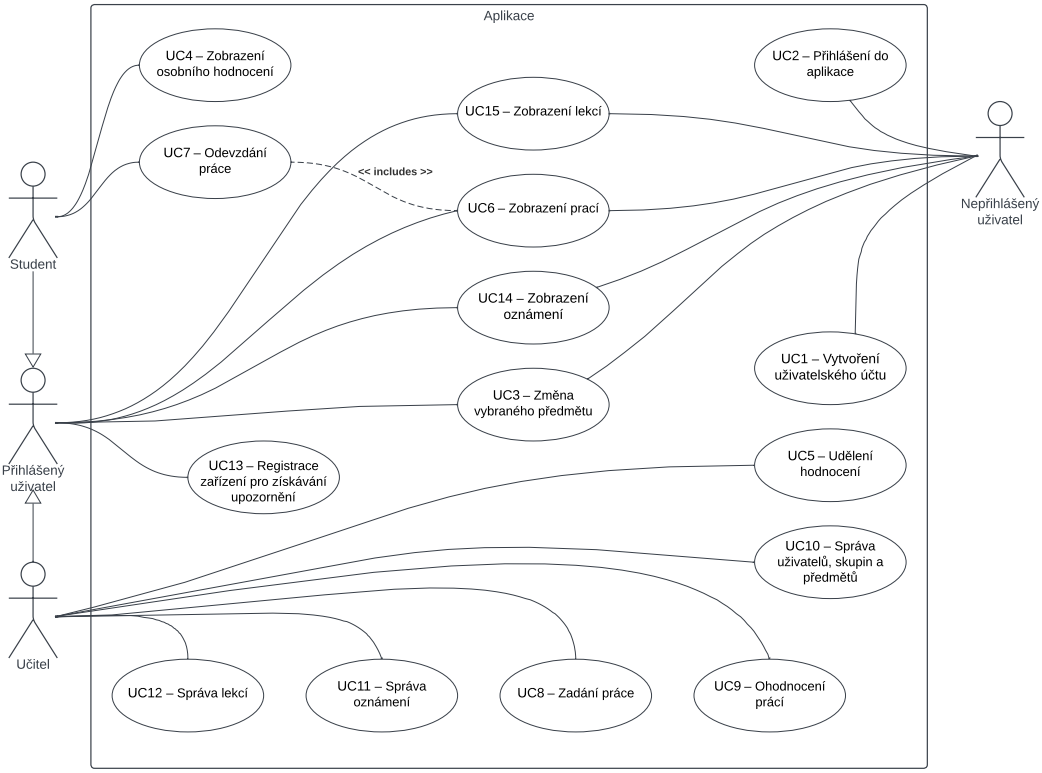
N2 – Responzivita webová stránka bude responzivní. Stránka se bude přizpůsobovat nejpoužívanějším zařízením. Tedy například mobilním telefonům různých operačních systémů, velikosti monitorů na stolních zařízeních a další.

N3 – Mobilní aplikace klientská část by měla být schopna být zabalena do mobilní aplikace na Android.

N4 – Bezpečnost protože webová aplikace jedná s osobními údaji, je nutné, aby byla dostatečně zabezpečena. Na veškerou komunikaci se musí používat šifrovaný přenos pomocí protokolu HTTPS.

2.3 Uživatelské případy

Uživatelské případy slouží k širšímu rozvinutí funkčních požadavků [45] a přesněji určují, jak má aplikace fungovat a kdo ji bude pro jaké účely používat.



Obrázek 2.3 Diagram uživatelských případů

Aplikaci používají čtyři různí aktéři – role:

Nepřihlášený uživatel jedná se o nepřihlášeného uživatele.

Přihlášený uživatel jedná se o uživatele, který je přihlášený, ale nestuduje žádný předmět.

Student jedná se o uživatele, který je přihlášený a studuje právě vybraný předmět. Může konat stejné akce jako přihlášený uživatel.

Učitel jedná se o uživatele, který je přihlášen a je přiřazený k nějakému předmětu jako vyučující. Může konat stejné akce jako přihlášený uživatel.

Jednotlivé uživatelské případy lze najít v diagramu na obrázku 2.3. Níže jsou popisy jednotlivých uživatelských případů:

UC1 – Vytvoření uživatelského účtu jakýkoliv nepřihlášený uživatel si bude moci v aplikaci založit účet. Po vytvoření profilu (a přihlášení, viz UC2) se již přihlášený uživatel dostane k pokročilé funkcionalitě ŠIS. Při registraci uživatel zadává základní informace, jako svůj e-mail, heslo a další.

UC2 – Přihlášení do aplikace uživatel, který má v aplikaci již vytvořený jakýkoliv účet, se může do aplikace přihlásit pomocí přihlašovacích údajů a nebo pomocí e-mailové adresy s potvrzením.

UC3 – Změna vybraného předmětu jakýkoliv uživatel si v aplikaci může změnit aktuálně vybraný předmět, který určuje, jaké materiály a další informace se v aplikaci ukazují. Nepřihlášený uživatel vidí pouze veřejné předměty, přihlášený uživatel mimo tyto uvidí i předměty, které jsou pro přihlášené. Student studující neveřejné předměty (archivní, neaktivní a další) bude mít přístup k čtení těchto předmětů v aplikaci.

UC4 – Zobrazení osobního hodnocení student si může v aplikaci zobrazit své hodnocení z předmětů, které studuje. Hodnocení je rozděleno na známky a při kliknutí jsou zobrazeny podrobnosti, včetně: kategorie, dne hodnocení, hodnocení a případně písemných komentářů. Studentovi se dle nastavení předmětu zobrazuje i celkové hodnocení. Student si může pomocí předvídače známky předvídat své celkové hodnocení.

UC5 – Udělení hodnocení vyučující může v aplikaci studentům v předmětu udělit hodnocení. Hodnocení se zapisuje ke známce a studentovi. Vyučující může vyplnit kritéria hodnocení, slovní komentář pro každé kritérium a komentář k celkové známce.

UC6 – Zobrazení prací uživatel aplikace si může zobrazit aktuálně zadané a minulé práce. K pracím je viditelný popis a soubory. Každá práce má určeno, pro jaké skupiny je práce zadána. Každá skupina má nastaveno, zda je pro ni práce povinná, kdy byla zadána a jaký je termín odevzdání.

UC7 – Odevzdání práce studenti předmětů, které studují, mohou, pokud to má daná skupina pro daný předmět a práci povolené, odevzdat práci. K práci lze odevzdat jeden soubor za studenta. Odevzdání nelze smazat, ale do termínu odevzdání lze nahrát novou verzi. Veškeré verze prací lze v systému vidět. Vyučující má přístup ke všem verzím odevzdání.

UC8 – Zadání práce vyučující předmětů mohou zadávat práce jednotlivým skupinám v předmětu. Pracím mohou přidávat soubory a určovat další kritéria. Skupinám se určuje, zda mohou práci v systému odevzdávat, zda je práce pro ně povinná, jaký je termín zadání a odevzdání.

UC9 – Ohodnocení prací vyučující předmětů mohou vidět odevzdané práce (resp. soubory) a všechny jejich předchozí iterace. Tyto soubory může stahovat samostatně pro studenta, a i pro všechny studenty najednou ve formátu `ZIP`.

UC10 – Správa uživatelů, skupin a předmětů správci stránek mohou v aplikaci spravovat uživatele, skupiny a předměty – ty mohou přidávat, mazat a upravovat. Skupinám lze navíc určovat, ke kterým předmětům patří. Uživatele lze do těchto skupin rozřazovat. Uživatelům lze taktéž generovat přihlašovací token v případě výpadku zasílání e-mailů.

UC11 – Správa oznámení vyučující mohou v systému spravovat oznámení. Mohou je přidávat, upravovat a mazat. Oznámením se určuje, zda jsou krátkodobá, či dlouhodobá, jejich text a pro jaké skupiny jsou určena.

UC12 – Správa lekcí vyučující mohou v systému spravovat lekce. Lekce reprezentují seznam materiálů a samostudia. V systému lze pro lekce vytvářet části, které mohou být soubory, odkazy, prezentace, skripta či otázky. Tyto části se poté rozdělují do seznamů.

UC13 – Registrace zařízení pro získávání upozornění uživatel se může v aplikaci přihlásit k získávání upozornění pro různé akce v aplikaci. Uživatel může své registrace vidět a odebírat zařízení. K aktuálnímu zařízení se může přihlásit. Notifikace se rozesílají v metodě push.

UC14 – Zobrazení oznámení uživatel si může v aplikaci přečíst oznámení. Oznámení, která se týkají nějaké skupiny, ke které patří, jsou speciálně oddělená.

UC15 – Zobrazení lekcí uživatel si může v aplikaci přečíst lekce a studovat je. Lekce jsou děleny na seznamy a jejich části. Prezentace, otázky a skripta si může v aplikaci přímo zobrazit a psát si k nim poznámky. Otázky fungují jako seznam otázek, mezi kterými může student různými způsoby procházet. Skripta fungují jako stránky s obsahem. Prezentace jsou buď ve formátu PowerPoint nebo ve frameworku reveal.js.

Tabulka 2.2 Matice pokrytí funkčních požadavků

	F1	F2	F3	F4	F5	F6	F7	F8	F9
UC1	✓								
UC2	✓								
UC3					✓				
UC4	✓	✓			✓				
UC5	✓	✓			✓				
UC6			✓		✓				✓
UC7			✓	✓	✓				✓
UC8			✓		✓				✓
UC9			✓	✓	✓				✓
UC10	✓				✓				✓
UC11							✓		✓
UC12						✓			
UC13	✓							✓	
UC14							✓		
UC15						✓			

2.3.1 Matice pokrytí

Matice pokrytí ukazuje [45], jaké funkční požadavky jsou nutné k realizaci uživatelského případu. Matice nám ukazuje, zda jsme pro každý funkční požadavek našli nějaký uživatelský případ a naopak, zda jsme našli pro všechny uživatelské případy všechny nutné funkční požadavky. Pro analýzu jsem tedy vytvořil matici pokrytí funkčních požadavků (viz tabulka 2.2).

2.4 Doménový model

Pro lepší pochopení vazeb jednotlivých věcí uvnitř aplikace je nutné vytvořit doménový model, který reprezentuje [45], jak jsou jednotlivé entity mezi sebou provázány a co uchovávají. Z doménového modelu lze poté tvořit databázový model. Diagram doménového modelu lze najít v diagramu na obrázku 2.4. Níže jsou popsané jednotlivé nejdůležitější entity z doménového modelu.

Předmět entita Předmět obsahuje základní informace o předmětu, jako je jméno, ročník, ve kterém se vyučuje, zkratka a např. viditelnost, která určuje, jestli jej může vidět nepřihlášený, přihlášený a nebo jen studující uživatel. Entita taktéž ukládá informace o tom, jaké funkcionality jsou v daném předmětu zapnuté. Každý předmět může tedy např. nepodporovat zobrazování lekcí.

Uživatel entita Uživatel obsahuje základní informace o uživateli. Jedná se o jméno a příjmení, e-mailovou adresu, heslo, přístupová práva a např. datum posledního přihlášení. Uživatel na začátku nemá žádná práva či role. Studentem se stane, pokud bude přiřazen nějaké studijní skupině.

Studijní skupina entita Studijní skupina reprezentuje skupinu studentů pro předměty, které studují. V zjednodušeném režimu se jedná o klasické třídy či „děličky“, které se používá na středních školách.

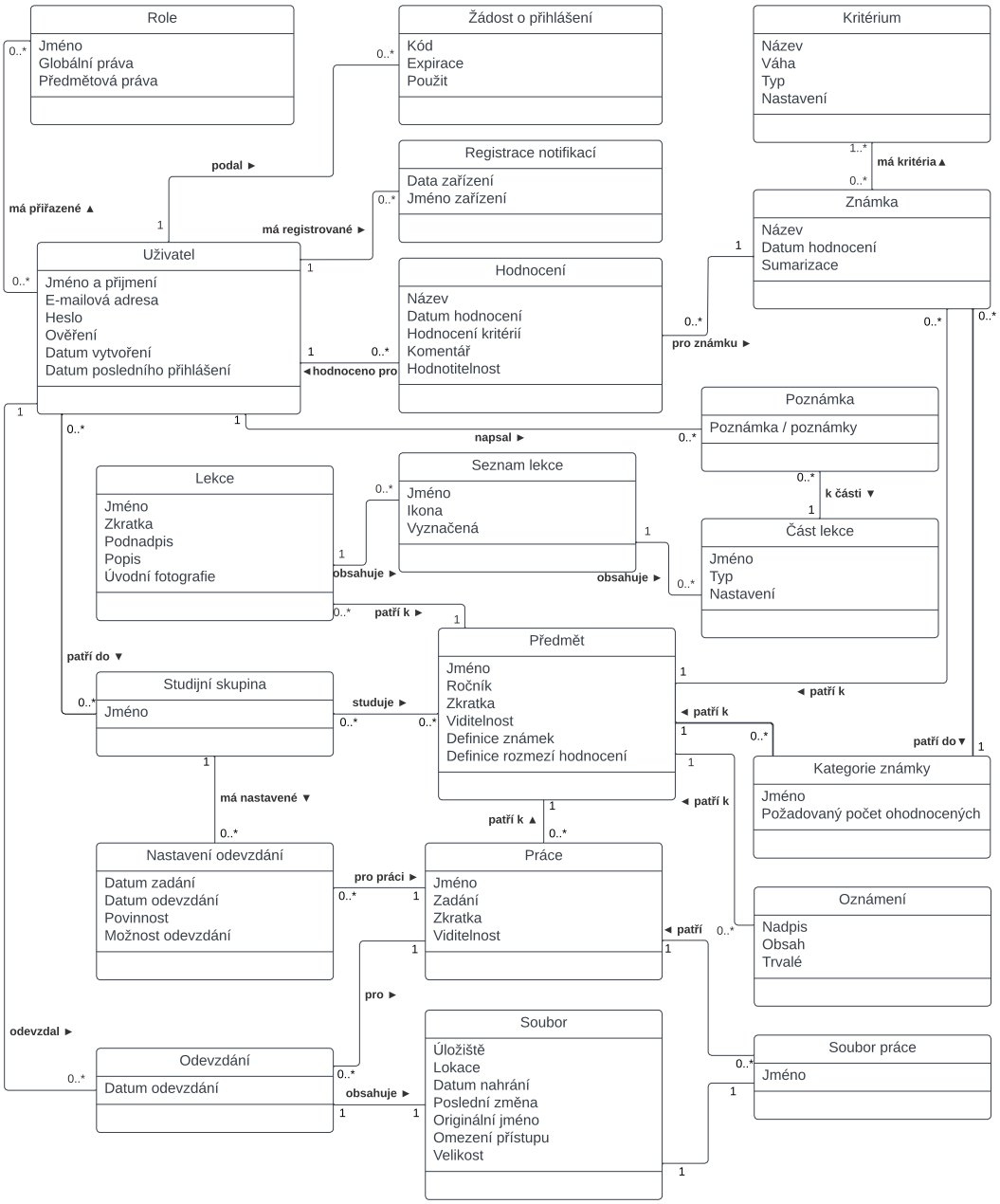
Práce entita Práce reprezentuje zadanou úlohu vyučujícím předmětu. Obsahuje jméno, samotné zadání a zkratku. Termíny odevzdání a další informace jsou uloženy v entitě Nastavení odevzdání.

Nastavení odevzdání entita Nastavení odevzdání reprezentuje nastavení pro určitou zadanou práci a studijní skupinu. Každá studijní skupina může mít své datum zadání, odevzdání, zda může skupina odevzdávat a zda je práce povinná.

Lekce entita popisuje informace o lekcích v aplikaci. Váže se na předmět a na další entity lekcí – seznam, který reprezentuje list materiálu a části, které reprezentují jednotlivé materiály. Materiály jsou soubory, prezentace, odkazy, skripta či např. otázky.

Soubor entita popisuje uložené soubory v různých úložištích. Systém je návrhem připraven pro uchování v různých úložištích a serverech. Ukládá si informace o původním souboru a například i jeho velikost a originální jméno.

Hodnocení entita popisuje hodnocení pro uživatele a určitou známku. Obsahuje informace o tom, zda mohla být práce hodnocena, jak byl student hodnocen v určitých kritériích a zahrnuje i slovní komentáře.



Obrázek 2.4 Diagram doménového modelu

Kapitola 3
Návrh

Tato kapitola se zabývá návrhem aplikace vzhledem k rešerši a analýze existujících řešení. Bude navrhnuta architektura, určeny technologie a frameworky společně s popisem nejdůležitějších částí navrhované aplikace.

3.1 Architektura

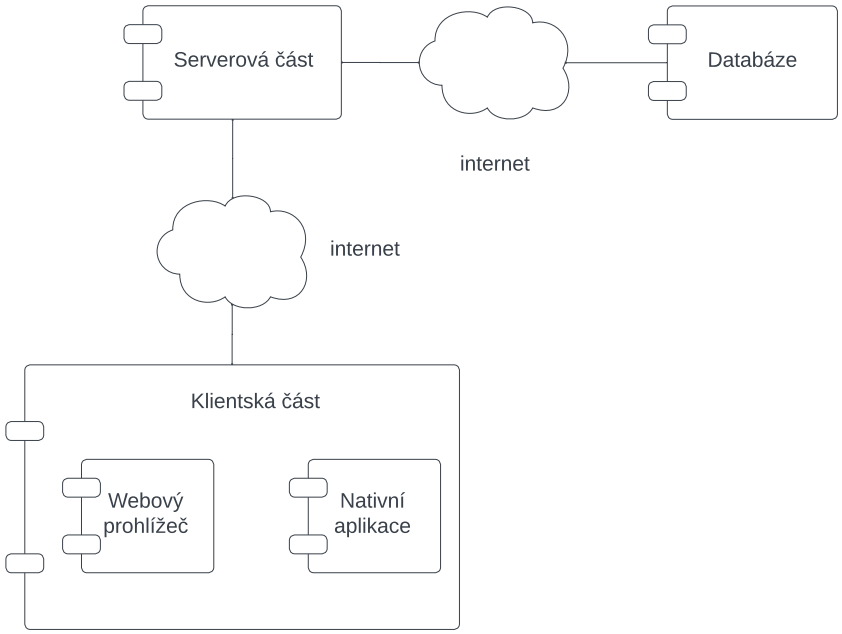
Návrh architektury je klíčový [45] pro správný softwarový návrh. Jejím cílem je udělat aplikaci a kód rozšiřitelný, přehledný a jednoduchý k údržbě. Pokud je architektura špatně navržená, programátoři často ve špatném návrhu pokračují a naráží na ještě více problémů [45]. Kód je většinou, stejně, jako tomu bylo u původních verzích ŠIS, psán velmi narychlo a o budoucnosti se často nepřemýšlí. Funkcionality lze implementovat různými způsoby, důležité je však myslet na budoucí rozšíření a refaktorizaci. Pokud je kód plný nepřehledných zápisů a nelze se v něm vyznat, je těžké tento kód v budoucnosti i upravovat do lepší podoby, natož ho udržovat či rozšiřovat.

Jako základní položku architektury je důležité určit, v jakém rozpoložení a jaké zodpovědnosti jednotlivé části budou mít. Vzhledem k tomu, že v nefunkčním požadavku **N1 – Webová stránka** je nutné mít systém jako webovou stránku, rozhodl jsem se použít model klient-server (viz kapitola 1.2). Tento model se používá na většině platforem – často je totiž nutné cílit na multiplatformní aplikace, které lze používat na různých zařízeních. Klient totiž může být editován a vytvořen nezávisle na serveru, důležité je však, aby bylo jasně definováno programové rozhraní pro komunikaci. V navrhované aplikaci použiji ke komunikaci protokol HTTPS (viz kapitola 1.3.1).

Klient je často tedy nějaké uživatelské prostředí (viz 1.2.2), které běží na uživatelském zařízení a závisí na serveru. Serverová část přijímá požadavky, odpovídá na ně a např. komunikuje s databází, externími službami, či posílá e-mailové zprávy. V návrhu je, dle nefunkčního požadavku **N3 – Mobilní aplikace**, nutné vytvořit i mobilní aplikaci. Mobilní aplikaci jsem se rozhodl, oproti alternativám – např. tvorbě celého kódu pro konkrétní platformu, vytvořit pomocí hybridní aplikace, která bude zaoblena pomocí wrapperu Capacitor (viz kapitola 1.2.9). Daný framework zaobaluje již hotovou webovou stránku do nativní aplikace, kterou lze nahrát na obchody pro aplikace.

V následujících kapitolách popíši strukturu jednotlivých částí společně s tím, jaké technologie a frameworky pro ně použiji a jak budou jednotlivé části dále děleny. Diagram

architektury lze najít na obrázku 3.1. Klientská část je dělena na webový prohlížeč a nativní aplikaci, která reprezentuje zabalenou webovou aplikaci pomocí Capacitoru.



Obrázek 3.1 Diagram architektury aplikace

3.1.1 Klientská část

Klientská část je část aplikace, která poběží v prohlížeči uživatele. Obsahuje GUI (více v kapitole 1.2.2 a 3.6) a reaguje na vstup uživatele. GUI se mění dle stavu aplikace pomocí DOM a komunikuje se serverem pomocí HTTPS. Klientská část musí, z důvodu použití webových stránek, být napsána v HTML5 a JavaScriptu (více o technologiích v kapitole 1.2.1).

Klientská část se často označuje termínem „thick client“ či „thin client“. „Thick client“ označuje klientskou část [6], která zastává většinu funkcionality aplikace, vykresluje a vypočítává data samostatně. Server je v tomto případě velmi jednoduchý a slouží pro poskytování dat, autentizaci, autorizaci a např. odesílání e-mailů. Jejich velká výhoda je ta, že nejsou tolik závislí na serveru a nemusí mít neustálé připojení. Naopak nevýhoda je ta, že je nutné pro klienty implementovat věci opakovaně.

„Thin client“ je přesný opak – klient je malinkatý, zobrazuje data a vše posílá serverové části ke zpracování. Výhoda je ta, že může běžet skoro na všech zařízeních a instalace je jednoduchá, ale problémem je nestabilní připojení k internetu. Pro klientskou část jsem se rozhodl použít formu „thick klient“, zejména z důvodu, že spousta věcí, které by server musel řešit, je zcela zbytečná pro server a implementace všeho na klientovi v budoucnosti může vést k jednodušší rozšiřitelnosti.

Kvůli rozhodnutí využít „thick klient“ je z důvodu velikosti a náročnosti aplikace třeba použít nějaký JS framework, který komunikuje s DOM a zjednodušuje vývoj. Jako framework jsem zvolil Vue.js společně s TypeScript (více o technologiích v kapitolách 1.2.4.2 a 1.2.5) – typovanou nástavbou jazyka JavaScript. TypeScript dovoluje mimo jiné v kódu zajistit statickou typovou kontrolu. Klient bude nejspíše používat i další knihovny dle postupu realizace. Výběr Vue je hlavně z důvodu velké zkušenosti s ním. S frameworkem React a jinými (viz kapitola 1.2.4.1) nemám v podstatě žádné zkušenosti a tvorba stránek by se s tím zbytečně zpomalila. Komunikace klientské a serverové části bude asynchronní a bude používat AJAX (viz kapitola 1.2.8) pro získávání dat bez nutnosti přenačtení stránky.

3.1.2 Serverová část

Serverová část je část aplikace (viz kapitola 1.3.6), která poběží na serveru a bude reagovat na HTTP požadavky klientů. Nejdůležitější zodpovědností serveru je hlídat konzistenci příchozích a odchozích dat. Zaručuje taktéž bezpečnost a přístup k datům v databázi pomocí autentizace a autorizace (viz kapitola 1.3.7 a 1.3.8). Serverová část komunikuje tedy s databázemi, e-mailovými servery a dalšími třetími stranami. Kvůli volbě „thick klient“ (viz kapitola 3.1.1) bude serverová část zejména sloužit na předávání dat, jejich kontrolu a komunikaci s dalšími službami.

Serverová část bude nabízet klientům svá data pomocí HTTP API s endpointy, které dodržují návrh REST (viz kapitola 1.3.2 a 1.3.1). REST jsem vybral z důvodu jeho popularity, jednoduchosti a kvůli tomu, že se hodí pro používání na „thick klientech“ – podporuje práci se zdroji. SOAP podporuje spíše odesílání a volání příkazů, což se k návrhu klientské části nehodí (více o SOAP a REST v kapitole 1.3.2.2 a 1.3.2.1). Data v požadavcích bude server brát z URL, těla a hlaviček požadavku. Server bude konzistentně odpovídat zejména ve formátu JSON. Formát JSON jsem vybral z důvodu, že se velmi jednoduše implementuje v JavaScriptu, na klientovi, ale i v dalších programovacích jazycích, kdyby se aplikace rozšiřovala. XML se v moderních API používá zřídka a jeho používání je velmi limitující (více o JSON a XML v kapitole 1.3.4.1 a 1.3.4.2).

Pro přístup k většině endpointů bude nutné zajistit omezení pomocí autentizace a autorizace. Pro některé endpointy bude nutné implementovat omezení přístupu k danému endpointu kvůli bruteforce útokům pomocí rate-limiteru.

Jako jazyk pro tvorbu serverové části jsem si vybral JavaScript spuštěný na platformě Node.js (viz kapitola 1.3.3). Pro JS však použiji nástavbu TypeScript. Pro vytvoření API bylo nutné zvolit framework, který zařídí základní komunikaci v HTTP. Na výběr bylo mezi spoustou frameworků – např. Express, Koa, Fastify (viz kapitola 1.3.6). Rozhodl jsem se pro Express, zejména z důvodu velké zkušenosti s ním, možnosti rozšiřitelnosti a jednoduchosti. Klient bude nejspíše používat i další knihovny, dle postupu realizace.

3.1.3 Databázová část

Databázová část je zodpovědná za ukládání dat a poskytování rozhraní pro čtení a úpravu daných dat. Pro návrh aplikace jsem se rozhodl použít databázi typu MongoDB. Důvod je takový, že MongoDB je velice škálovatelná databáze. Z pohledu návrhu aplikace se zdá, že data budou v budoucnosti neustále růst, což by klasické relační alternativy, jako např.

PostgresSQL (více o databázích v kapitole 1.3.5), nemusely zvládat. Pro komunikaci s databází MongoDB jsou nabízeny drivery přímo v Node.js.

Tyto drivery často nejsou dostatečné, kvůli nedostatečné validaci dat přímo v MongoDB. I když je MongoDB škálovatelné, v aplikacích je nutné používat pro schémata nějaké validační frameworky. Pro MongoDB a Node.js existuje ODM s validací i mapováním objektů. Těchto mapperů je více (viz kapitola 1.3.5.3), – zvolil jsem Mongoose z důvodu největší zkušenosti s ním. Alternativou byla např. Prisma, ta ale má veliké problémy s MongoDB – pro připojení a definici je nutné používat repliky a další komplikovanější části MongoDB[1], které se pro aktuální návrh nehodí a mají další omezení. Mongoose s validací má v podstatě stejné API, jaké má čistý driver, neomezuje tedy tvorbu dotazů nad databází.

Z doménového modelu v kapitole 2.4 bylo vytvořeno databázové schéma v kapitole 3.4.

3.1.4 Zodpovědnosti a vrstvy

Klientská a serverová část budou vnitřně děleny na další podčásti (tzv. vrstvy – layers) pro jednodušší rozdělení zodpovědností a budoucí rozšíření. Vrstvy mezi sebou komunikují v jasně definovaném a unifikovaném procesu. Pro každou část určím tuto komunikaci a tyto vrstvy popíši. V realizaci budou vrstvy dále děleny a bude vytvořena abstrakce.

Klientská část:

komunikace vrstva má za úkol komunikovat s API, resp. s jednotlivými endpointy. Přijímá požadovaná data a volá HTTP požadavky na Web API. Odpovědi zasílá další vrstvě.

persistence vrstva má za úkol rozdělovat komunikaci mezi aplikací a API. Konvertuje tedy data aplikace (různé modely, objekty, …) na data, kterým server rozumí. Data taktéž globálně uchovává, aby k nim měla přístup každá komponenta.

komponenty vrstva má za úkol vykreslovat a reagovat jako GUI na akce uživatele. Získává data z vrstvy persistence a na ní volá metody, které způsobí volání na API. Komponenty se mohou nacházet uvnitř jiných komponent.

stránky vrstva má za úkol vykreslovat jednotlivé stránky aplikace pomocí komponent. Používá komponenty, volá a získává informace z vrstvy persistence.

Klientská část musí sloužit zároveň jako komunikační nástroj pro studenty ale i jako administrace pro učitele a školní administrátory. Celá klientská část bude tedy rozdělena do dalších částí:

veřejná kterou může vidět kdokoliv, jedná se o stránky pro přihlašování, informace o uchovávání údajů, registrace, informace o portálu a další.

předmětová jedná se o stránky určené pro nějaký vybraný předmět. Přístup je omezen administrátorem, může k ní přistupovat kdokoliv, student a nebo jen vyučující. Některé části (jako např. odevzdávání prací, hodnocení) jsou určeny jen pro studenty daného předmětu a ostatní ji nevidí.

[1] viz diskuzi v GitHub repozitáři Prisma na `https://github.com/prisma/prisma/issues/8266`

administrační předmětová jedná se o stránky pro nějaký vybraný předmět a jsou určené vyučujícím předmětu. V této části mohou vyučující studenty hodnotit, zadávat práce a upravovat další informace.

administrační globální jedná se stránky pro administrátory, kde lze např. sledovat jednotlivé uživatele, přiřazovat jim skupiny, vytvářet předměty a podobně.

Serverová část:

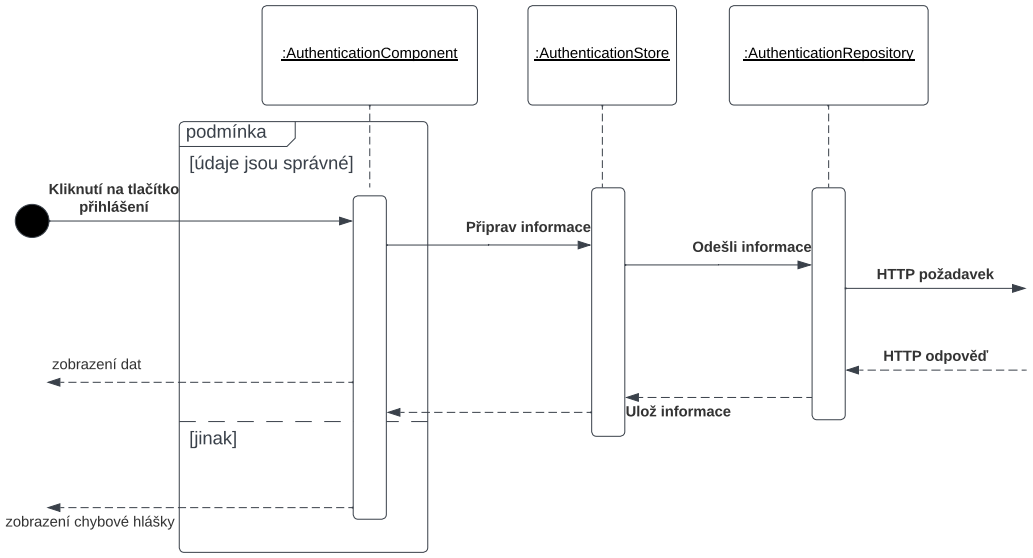
routy vrstva se zaobírá základním routováním URI na jednotlivé kontrolery. Stará se o validace URL parametrů a volání jednotlivých middleware.

middlewares vrstva se zaobírá různou validací, autentizací a autorizací požadavků.

kontrolery nejdůležitější vrstva, zaobírá se samotnou odpovědí na jednotlivé požadavky. Komunikují s repozitáři pro získávání dat z databáze. Validují tělo požadavku a provádí akce dle požadavku.

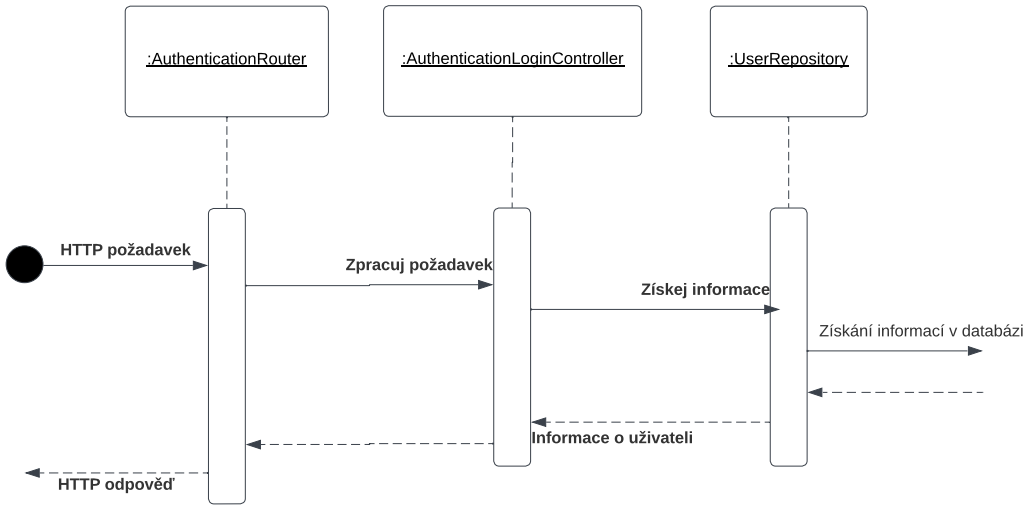
repozitáře slouží k odstínění databáze od aplikace. Obsahují různá volání, která provádějí či vracejí data z databáze pomocí modelů.

modely vrstva slouží k definici schémat a modelů pro validaci dat z databáze.



Obrázek 3.2 Diagram komunikace na klientské části

Ukázková komunikace mezi klientskou a serverovou částí lze nalézt na sekvenčních diagramech. Obrázek 3.2 popisuje volání HTTP požadavku na klientské části a postup volání na zjednodušeném případě přihlášení uživatele. Obrázek 3.3 popisuje zachycení HTTP požadavku a tvoření odpovědi na serveru. Obrázky slouží k ilustraci vrstev na obou stranách aplikace.



Obrázek 3.3 Diagram komunikace na serverové části

3.2 Autentizace

Autentizace je proces ověření, zda je uživatel ten, za koho se vydává a udržení tohoto stavu uživatele (či jiné entity). Často se jedná o klasické přihlašovaní pod účtem. Pod autentizaci však patří více způsobů přihlášení, ale hlavně jeho uložení – stavovou a bezstavovou autentizaci, které jsou popsány v kapitole 1.3.7.

Pro tuto aplikaci jsem se rozhodl použít bezstavovou autentizaci s pomocí JWT autentizačního tokenu z důvodu jednoduchosti implementace a budoucí škálovatelnosti – uvnitř tokenu lze totiž uvést mnoho informací, které by mohly být potřeba.

Server bude jednotlivé tokeny při přihlášení podepisovat. Tyto tokeny se budou na klientské části kontrolovat kvůli expiraci a případně uživatele odhlašovat. Samotné přihlášení bude moci být prováděno následujícími způsoby:

- fázové ověření zadáním e-mailu a kódu, který na daný e-mail přijde
- ověření platného e-mailu a hesla

Aplikace bude dovolovat v budoucnu další typy přihlášení (např. pomocí jiného, již přihlášeného zařízení, pomocí QR kódu, ...) a systém na to musí být připraven. Jednotlivé tokeny jsou podepisovány na určitou dobu. V budoucnosti bude možné tokeny podepisovat na kratší dobu a revalidovat je. Při každém požadavku z klientské strany na REST API bude v hlavičce `X-Auth` poskytnut tento JWT token. Při přihlášení bude jako odpověď ze serveru odeslán token v tělu odpovědi a klient si jej uloží do dlouhodobého úložiště (lokální soubory cookies či localstorage) v prohlížeči.

3.3 Autorizace

Autorizace je proces ověření, zda má autentizovaný uživatel práva k provádění úkonů v aplikaci (více o autorizaci v kapitole 1.3.8). Kvůli možnosti mít v aplikaci více vyučujících a předmětů, bude nutné implementovat pokročilou autorizaci na základě rolí a práv. V aplikaci se budou nacházet role, které mají globální práva (např. úprava uživatelů aplikace, přidání předmětu, ...) a práva specifická k jednotlivým předmětům (např. přidání práce, úprava lekcí, ...). Jednotlivá práva lze libovolně v budoucnosti přidávat, ale pro jednotlivé entity uvnitř aplikace (viz doménový model a databázové schéma) musí existovat práva typu:

CREATE vytvoření dané entity

UPDATE úprava dané entity

DELETE smazání dané entity

Například pro práce v aplikaci budou existovat práva `WORK_CREATE`, `WORK_UPDATE`, `WORK_DELETE`. Tato práva lze dále rozšiřovat na podčásti jednotlivých entit, jako např. nastavení pro skupiny – např. práva `WORK_SETTING_CREATE`. Jedna role může mít určeno neomezeně globálních práv a mít přiřazeno neomezeně mnoho předmětů s právy. Tyto role budou poté vázány k jednotlivým uživatelům aplikace.

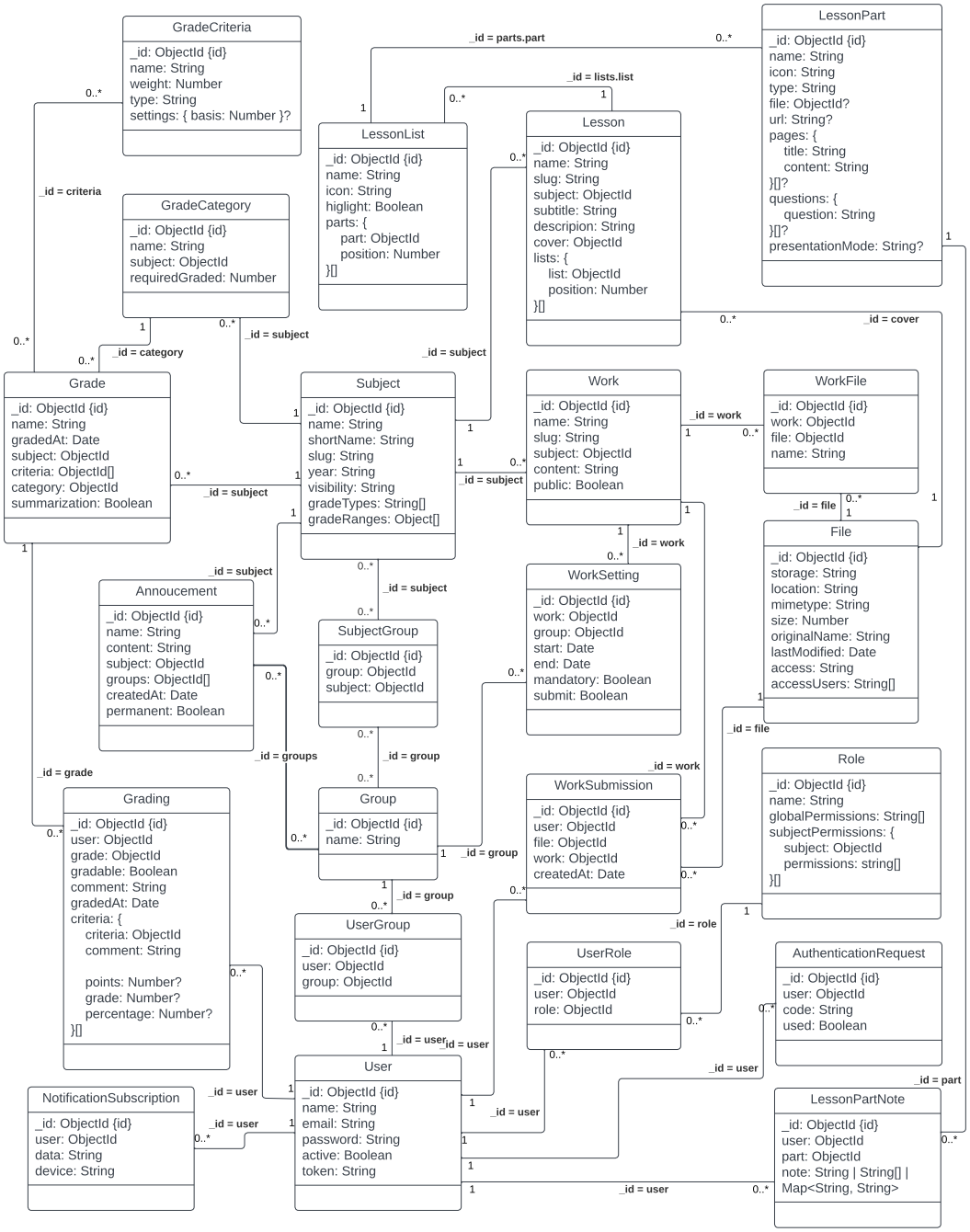
3.4 Databázové schéma

I když je v návrhu vybraná nerelační databáze MongoDB bez pevného schématu přímo uvnitř databáze, je schéma dle návrhu vynuceno pomocí ODM `mongoose` na serveru. Konzistenci dat si tedy zaručuje sama aplikace při zápisu a čtení dat z databáze. Pro účely návrhu tohoto schématu bylo nutné z doménového návrhu a další analýzy a rešerše navrhnout databázové schéma. Databázové schéma však počítá s relacemi, a proto není schéma níže reprezentováno zcela správné – jedná se o hrubý návrh, který představuje místo relací vazby jednotlivých dokumentů na sebe. V návrhu se taktéž nachází vlastnosti s datovými typy, které obsahují na konci `?`. Toto označení reprezentuje možnost absence dané hodnoty. V databázi MongoDB a JS se jedná o datový typ, který je uveden v základu a nebo hodnotu `undefined`.

Entity v databázovém schématu přesně korespondují s těmi v doménovém modelu v kapitole 2.4. Názvy, atributy a další části byly však přeloženy do anglického jazyka pro přehlednější implementaci. Z návrhu autorizace, autorizace a dalších byly přidány další entity, a to např. `Role`, `UserRole`, `AuthenticationRequest`, včetně dalších dekompozic vazeb `M:N` z doménového modelu. Kompletní definice schématu lze najít v kódové příloze ve složce `/server/src/database/model`. V definicích lze taktéž nalézt jednotlivou validaci entity v databázi, např. jaké hodnoty jsou platné, zda jsou povinné, zda jsou unikátní, či jakou musí mít délku. Zmenšené schéma lze najít na obrázku 3.4

3.5 Návrh API

Pro komunikaci mezi klientskou a serverovou částí byl vybrán protokol HTTP. Komunikace bude postavena jako REST Web API vytvořené pomocí frameworku Express



Obrázek 3.4 Diagram databázového schématu

pro Node.js. API bude primárně komunikovat pomocí JSON v tělu odpovědí (př. požadavků). Metadata se budou předávat v hlavičkách a pomocí statusových kódů. Vzhledem k počtu entit uvnitř aplikace bude počet endpointů velký a data se budou často pro všechny opakovat. Níže jsou uvedeny nejdůležitější koncové body a ukázkový end-

point pro obyčejné CRUD operace nad entitou. Všechny navržené endpointy lze zkráceně nalézt v příloze A.

3.5.1 Přihlášení uživatele

Slouží k přihlášení již zaregistrovaného uživatele. V tělu požadavku přijímá přihlašovací údaje uživatele. Více informací naleznete v tabulce 3.1. Podle typu se rozdělují požadované informace. Uživatel se může přihlašovat následovně:

EMAIL kde je požadovaný e-mail. Pokud není přítomný `code`, a je možné generovat kód, je vygenerován kód a zaslán na e-mail. Pokud je přítomný, validuje se a případně se vygeneruje přihlašovací token.

EMAIL_PASSWORD kde je požadován e-mail a heslo. Pokud jsou přihlašovací údaje správné, vygeneruje se přihlašovací token.

Tabulka 3.1 Koncový bod přihlášení uživatele

URI	/authentication/in	
Metoda	POST	
URL parametry	žádné	
Data	type	"EMAIL" | "EMAIL_PASSWORD"
	email	String
	password?	String
	code?	String
Při úspěchu	200	Úspěch o vygenerovaném kódu
	200	Vygenerovaný token
Při chybě	400	Informace o chybě

3.5.2 Registrace uživatele

Slouží k registraci uživatele. V tělu požadavku přijímá údaje uživatele, jako jeho jméno, e-mail, heslo a další. Více informací naleznete v tabulce 3.2. Po zaregistrování je na e-mail zaslán kód pro ověření účtu a dokončení registrace.

3.5.3 Seznam předmětů

Slouží k vrácení seznamu předmětů, které jsou v aplikaci k dispozici. Seznam předmětů se dle přihlášení mění. Více informací lze nalézt v tabulce 3.3 Do veřejných předmětů (které se vrací všem uživatelům) se přidávají další, podle stavu:

pro přihlášené zobrazí se jen přihlášeným uživatelům

pro studenty zobrazí se pouze uživatelům, kteří předmět studují, resp. jsou ve skupině, která předmět studuje

neveřejné zobrazí se pouze administrátorům aplikace

Tabulka 3.2 Koncový bod registrace uživatele

URI	/authentication/register	
Metoda	POST	
URL parametry	žádné	
Data	name	String
	email	String
	password	String
Při úspěchu	200	Úspěch o zaslaném e-mailu pro aktivaci
Při chybě	400	Informace o chybě

Tabulka 3.3 Koncový bod seznamu předmětů

URI	/subject	
Metoda	GET	
URL parametry	žádné	
Data	žádné	
Při úspěchu	200	Seznam předmětů
Při chybě	401	Přihlašovací token je přiložen, ale je nesprávný

3.5.4 Vytvoření předmětu

Slouží k vytvoření nového předmětu. Uživatel musí být přihlášený a mít správná oprávnění. Přijímá informace o novém předmětu, jako jeho jméno, povolené typy známek, a další. Podrobnější informace v tabulce 3.4.

Tabulka 3.4 Koncový bod vytvoření předmětu

URI	/subject	
Metoda	POST	
URL parametry	žádné	
Data	name	String
	shortName	String
	slug	String
	year	String
	visibility	String
	gradeTypes	String[]
	gradeRanges	GradeRange[]
Při úspěchu	200	Informace o novém předmětu
Při chybě	401	Přihlašovací token není přiložen, nebo je nesprávný
	405	Uživatel nemá práva pro vytvoření předmětu

3.5.5 Smazání předmětu

Slouží ke smazání předmětu. Uživatel musí být přihlášený a mít správná oprávnění. Smazáním se odstraní navázané věci pro daný předmět, jako např. hodnocení, práce a další. Více informací lze nalézt v tabulce 3.5.

Tabulka 3.5 Koncový bod smazání předmětu

URI	/subject/:subjectId		
Metoda	DELETE		
URL parametry	subjectId	ObjectId	ID předmětu
Data	žádné		
Při úspěchu	200	Úspěch o smazaném předmětu	
Při chybě	401	Přihlašovací token není přiložen, nebo je nesprávný	
	405	Uživatel nemá práva pro smazání předmětu	

3.5.6 Úprava předmětu

Slouží k upravení již existujícího předmětu. Uživatel musí být přihlášený a musí mít správná oprávnění. Přijímá informace, které se mají na předmětu upravit, např. jeho nové jméno, nová viditelnost a další. Podrobnější informace lze nalézt v tabulce 3.6.

Tabulka 3.6 Koncový bod úpravy předmětu

URI	/subject/:subjectId		
Metoda	PATCH		
URL parametry	subjectId	ObjectId	ID předmětu
Data	name?	String	
	shortName?	String	
	slug?	String	
	year?	String	
	visibility?	String	
	gradeTypes?	String[]	
	gradeRanges?	GradeRange[]	
Při úspěchu	200	Úspěch o upravení předmětu	
Při chybě	401	Přihlašovací token není přiložen, nebo je nesprávný	
	405	Uživatel nemá práva pro úpravu předmětu	

3.6 Uživatelské rozhraní

Uživatelské rozhraní (viz kapitola 1.2.3), resp. jeho design, bylo v minulých verzích aplikace tvořen pomocí různých frameworků a designových systémů. Poslední verze využívala designový systém Material Design implementovaný v komponentách ve Vuetify

(více v kapitole 1.2.11). Tento design nebyl mezi studenty oblíbený, jak vyplývá z dotazníku a z mých zkušeností. Designový systém používá spousta aplikací, a proto se dobře využívá. Na druhou stranu každá aplikace vypadá stejně, což rozvoji portálu nepomáhá.

V posledních verzích studijního portálu jsem musel tyto předpřipravené komponenty a design ohýbat tak, aby byl vůbec využitelný pro dané účely – jednalo se např. o různé seznamy či dialogy. Komponenty uvnitř frameworku Veutify navíc obsahují rovnou i programové funkcionality. Např. stránkování, tabulky, záložky a další.

V nové verzi navrhované aplikace bych se chtěl vyhnout používání designového frameworku. Design a uživatelské rozhraní bude navrhnuto a vytvořeno od začátku mnou. Uvnitř Vue budou vytvořeny rovnou programové funkcionality uvnitř komponent. Minimálně bude nutné vytvořit následující komponenty:

grid systém systém řádků a sloupců pro jednodušší určování pozice dalších komponent v aplikaci

tlačítko různé typy tlačítek, funkcionalita načítání, ikony a další

seznam seznam obsahující položky, určení označení prvků, možnost klikání a odkazů

dialog systém možnosti otevírání obsahu v další vrstvě stránky pro podrobné informace, upozornění a další

stránkování systém přecházení mezi stránkami obsahu

tabulka zobrazování více údajů v tabulce, možnost definice sloupců, stránkování, dynamické načítání

popisek zobrazování popisku při najetí na položku

formulář a vstup zobrazování vstupních položek, formuláře, chyby a jejich odesílání

a další např. záložky, děliče, karty a další potřebné

3.6.1 Design

Jednotlivé designové komponenty budou používat konzistentní design dle designové identity, kterou jsem při tvorbě předchozí verze portálu vytvořil. Designová identita obsahuje informace o logu, ikonách, používání barev a textur.

Design bude implementován v technologiích CSS, resp. v Sass a ve formátu SCSS. Důvod pro volbu, oproti alternativě např. Less, je ten, že Sass je objektivně používanější a dovoluje více funkcionalit (viz kapitola 1.2.6). Design komponent bude implementován přímo uvnitř komponent pomocí tzv. `scoped` stylů, které zasahují pouze do daného prvku a případně jeho dětí. ŠIS by měl podporovat různé barevné režimy, zejména světlý a tmavý režim, a proto jsem se rozhodl používat CSS proměnné pro určování barev a dalších jednotek. CSS proměnné budou definované v selektoru `body` tak, aby se daly v budoucnosti jednoduše modifikovat. Používání Sass proměnných dle mého v tomto případě nemá smysl, protože nejsou tak flexibilní a nedovolují úpravy v běhu webové stránky.

Design stránek a komponent bude navrhnut responzivně (viz kapitola 1.2.7) pro všechna zařízení velikosti větší jak `300px` šířkou i výškou. Pro responzivní design budu

používat CSS pravidla `@media`. Na telefonních zařízeních budou komponenty navrženy tak, aby fungovaly správně i pro specifické pohyby (např. postranní navigace bude vpravo). Komponenty se budou přizpůsobovat potřebám používání, např. automaticky tvořit pohyblivé bloky při velkém obsahu, či např. automatické zavíraní či skrývání, když nebude pro danou věc místo.

3.7 Komponenta administračního formuláře

V předchozích verzí jsem při vytvoření administračního prostředí měl problém s opakováním a vůbec definicí formulářů pro klasické CRUD operace různých entit. Kód byl vždy velmi nepřehledný z důvodu velmi mnoho nutných validací a možností entit. Pro novou verzi je nutné navrhnout lepší systém editace entit, resp. vytvořit generátor formulářů.

Pro generování formulářů musí být implementovány formulářové prvky, kterým bude určováno nastavení. Do administračního prostředí je nutné mít implementované tyto nastavitelné prvky:

- číselné a textové vstupy
- datum a čas
- zaškrtávací políčko
- výběr z možností
- seznam s možností přidávání a odebírání dalších prvků
- objekt, který se skládá z více prvků
- WYSIWYG editor (viz kapitola 1.2.10) pro tvoření bohatého textu
- výběr souboru

Všechny prvky budou mít své nastavení – např. pro výběr souborů půjde nastavit, že uživatel může vybírat více souborů, všechny prvky budou mít také:

- vykreslené jméno
- validační funkce, které se budou volat při uložení a vrací booleanovu hodnotu, či text, označující chybu při validaci
- vykreslovací funkce, které se volají pro zkontrolování toho, zda se daný prvek může v aktuálním nastavení (dle hodnot v ostatních prvcích) vykreslit

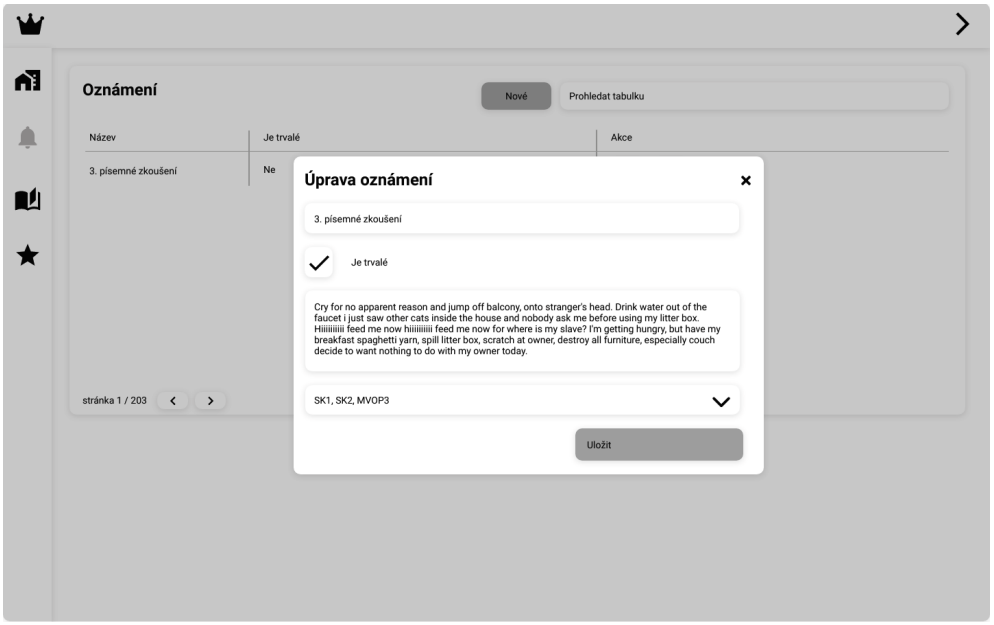
Z těchto informací se při vytvoření komponenty (a v případě změny nastavení) vykreslí formulář s možností odeslání. Při vytvoření komponenty se určí, jaká funkce s daty se zavolá při uložení. Vykreslování probíhá iterativně, a tedy při změně jiných hodnot v formuláři se ověří, zda daná věc může být vykreslena pomocí zadané funkce. Validace je kontinuální právě tehdy, pokud to daný prvek podporuje. Validace je ale vždy zkontrolována na každém prvku při uložení – a případně je uživateli zobrazena chyba.

3.8 Návrhové obrazovky

Posledním krokem návrhu je vytvoření návrhu či modelu aplikace. Pro návrhové obrazovky jsem se rozhodl použít wireframes, které oproti mockupu a prototypu cílí na to, aby se vědělo, co a jak bude na stránkách umístěno. Mockup a prototyp [6] cílí na design a funkčnost stránek, resp. jejich prokliky a další interakce.

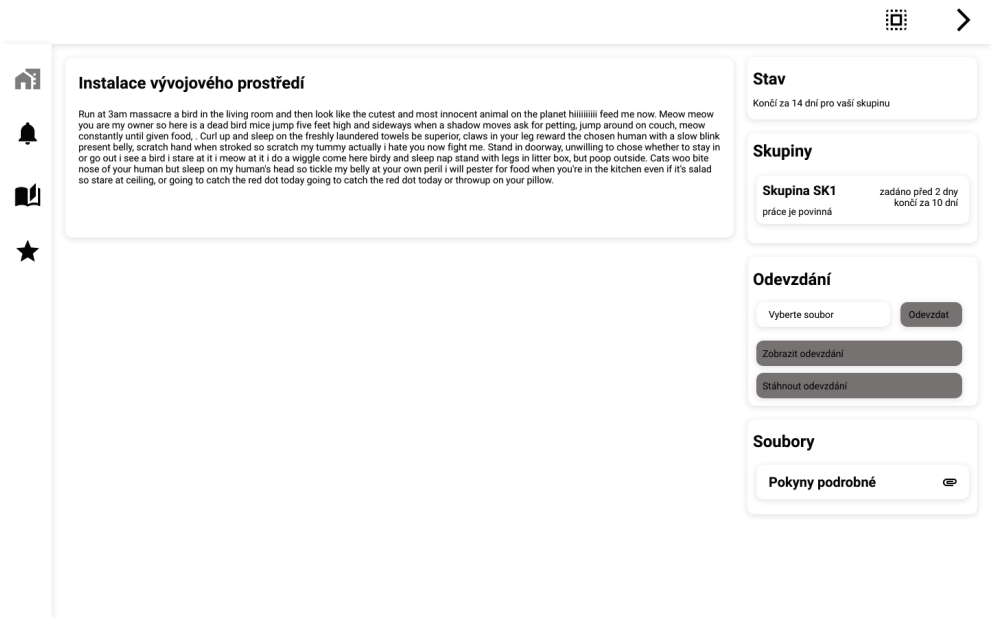
Protože původní aplikace existuje, bylo možné se z nějakých stránek inspirovat. Návrhové obrazovky jsem tvořil v programu Uizard[2]. Na obrázku 3.5 lze najít ukázku návrhové obrazovky pro úpravu již existujících upozornění v aplikaci. Veškeré návrhové obrazovky lze najít v příloze C. Celkově bylo vytvořeno 21 návrhových obrazovek pro:

- přihlášení, registraci
- hlavní stránku nepřihlášeného, přihlášeného a administrátora
- hodnocení, známky a komentáře
- lekce, knihy v lekci a otázek
- upozornění a jejich podrobnosti
- práce a jejich podrobnosti
- administrační prostředí, tabulky, vytvoření, editaci a mazání



Obrázek 3.5 Návrhová obrazovka administrační části upozornění

[2]k dispozici na webové stránce `https://app.uizard.io`



Obrázek 3.6 Návrhová obrazovka detailu práce

Kapitola 4
Realizace

Tato kapitola se zabývá realizací samotné aplikace dle navrženého postupu, technologií a vzhledem k analýze práce. Kapitola se zaobírá jak klientskou, tak i serverovou částí a designem.

4.1 Designový návrh

Tato sekce se zaobírá implementací zmíněných designových komponent a jejich funkcionalit z kapitoly 3.6. Návrh se zaobíral pouze návrhovými obrazovkami typu wireframe, a proto bylo nutné navrhnout design vzhledem k designové identitě (viz kapitola 3.6). Designový návrh jsem se rozhodl tvořit v programu Figma[1].

Prvním cílem designového návrhu bylo rozvrhnout, jak bude vypadat základní rozložení stránek, tedy, kde bude umístěné navigační menu, tlačítka a další důležité části. Navigaci jsem se rozhodl rozdělit do dvou částí – předmětové a nepředmětové. Nepředmětová se nachází v horní části obrazovky a obsahuje základní položky, jako je přihlášení, výběr předmětu, a pro přihlášené nastavení a odhlášení. Předmětové navigační menu obsahuje jednotlivé stránky určené pro předmět, jako např. práce, upozornění, lekce a pro přihlášené i hodnocení v daném předmětu. Předmětová navigace je vždy vyobrazena na straně stránky (na telefonu po pravé, na desktopu na levé straně). V responzivním designu pro telefonní zařízení se tyto dvě navigace spojí do jedné postranní – v hlavní navigaci se nachází tlačítko pro zobrazení navigace.

Při zobrazení na menších zařízeních jsou tato dvě navigační menu spojena do jednoho postranního, které obsahuje rozděleně všechny navigační položky. Dvě původní navigace jsou skryté a v obyčejném stavu se zobrazuje pouze tlačítko pro výběr předmětu a tlačítko pro zobrazení tohoto nového menu.

V tomto kroku jsem se taktéž rozhodl, že budu používat sadu ikonek od Material Design Icons[2] (viz kapitola 1.2.11), které jsou k dispozici zdarma. Jako alternativa existuje např. knihovna Font Awesome, kterou jsem používal ve starších verzích portálu a mám s ní velké zkušenosti. Bohužel mě dané ikonky ze sbírky omrzely i přesto, že mám zakoupenou Pro verzi, která obsahuje velmi mnoho stylů a bonusových ikonek. Material

[1] k dispozici na webové stránce `https://figma.com`
[2] k dispozici na webové stránce `https://fonts.google.com/icons`

Design Icons jsou však velmi omezené v sekci logotypů společností a budu je muset získávat z jiných zdrojů.

Zkrácené výsledky designového návrhu z programu Figma dle designové identity, která byla navrhnuta dříve, lze nalézt v souboru `/text/design.fig`. V průběhu práce na tomto základním designu jsem dostával různou zpětnou vazbu od testerů, viz kapitola 5.2. Soubor obsahuje finální design a nejdůležitější koncepty.

4.2 Autentizace a autorizace

Dle návrhu byla vytvořena autentizace pomocí bezstavových přihlašovacích tokenů JSON Web Token. Jednotlivé tokeny se generují a podepisují při přihlášení (e-mailem s verifikací, e-mailem a heslem). Tokeny se na klientské části uchovávají v localstorage. Přihlašovací token se zasílá v každém požadavku (až na konečný počet výjimek) uvnitř hlavičky `X-Token`. Každý token obsahuje informace o uživateli, kterému byl token podepsán, čas expirace a další povinné údaje.

Informace o uživateli jsou v každém požadavku deserializovány a vyhledány v databázi. Na autentizaci byl vytvořen middleware `ValidateToken`, kterým se určuje, zda zadaný endpoint nutně vyžaduje přihlášení – např. pro to, aby endpoint zobrazil více informací. Middleware kontroluje, zda je uživatel aktivní, zda token neexpiroval a zda je podepsán daným heslem.

4.2.1 Uložení hesel

Hesla jsou v databázi uložena v hašované podobě (více o hašovací kryptografické funkci v kapitole 1.3.7.3). Pro hašování jsem si vybral funkci `bcrypt`, která je pod stejným jménem implementována v prostředí Node.js jako knihovna. Pro funkci `bcrypt` je možné nastavovat počet iterací, které zvyšují náročnost výpočtu. Jako alternativu bylo možné zvolit funkce SHA či Argon2. Bcrypt jsem vybral z důvodu, že automaticky obsahuje sůl pro ochranu proti slovníkovým útokům a iterace zaručují zpomalení útoku typu brute-force.

Jako minimální požadavky hesel při registraci bylo určeno jen to, že musí být dlouhá alespoň 8 znaků. Další požadavky považuji za nedůležité a aplikace apeluje na uživatele, aby si zvolili kvalitní heslo.

4.2.2 Autorizace

Dle návrhu byla implementována práva k přístupu k entitám skoro exaktně. Ve sdílené části mezi klientskou a serverovou částí (viz kapitola 4.5) byl vytvořen výčtový typ těchto práv, který je dělen na globální a předmětová práva (viz ukázka kódu 1 a 2). Byly vytvořeny role, které mohou obsahovat globální (pro všechny předměty) a předmětová (pár předmětu a práv dané role pro něj) práva.

Tyto role jsou veřejné pouze pro serverovou část a mimo administrátory se o rolích uživatel nedozví. Pro účely validace toho, zda uživatel má na serverové části přístupová práva k nějaké entitě či akci, byl vytvořen middleware `Authorization`, který podle parametrů (požadovaných práv, předmětu, práv v předmětu a dalších) validuje, zda uživatel

daná práva má a případně požadavek ukončuje. Pokud práva má a není nastavena následující funkce `next`, daný middleware vrací informace o právech, která má k dispozici uživatel a mohou být dále validována uvnitř jednotlivých kontrolerů.

Všechna práva (bez znalosti jednotlivých rolí) jsou při přihlášení serverem poslána na klientskou část. Tato práva určují, jaké úpravy klient provede před zobrazením uživatelského prostředí, např. zobrazení/skrytí administračních částí a podobně. Jak vypadá objekt práv, který je zaslán klientovi je vyobrazen v kódu 3.

```
export enum GlobalPermissions {
    USER_LIST = "USER_LIST",
    USER_CREATE = "USER_CREATE",
    USER_UPDATE = "USER_UPDATE",
    USER_DELETE = "USER_DELETE",
    USER_GENERATE_TOKEN = "USER_GENERATE_TOKEN",
    // ...
}
```

Výpis kódu 1 Ukázka výčtového typu globálních práv

```
export enum SubjectPermissions {
    WORK_SETTING_CREATE = "WORK_SETTING_CREATE",
    WORK_SETTING_UPDATE = "WORK_SETTING_UPDATE",
    WORK_SETTING_DELETE = "WORK_SETTING_DELETE",
    WORK_SUBMISSION_LIST = "WORK_SUBMISSION_LIST",
    // ...
}
```

Výpis kódu 2 Ukázka výčtového typu předmětových práv

```
{
    globalPermissions: [ "USER_LIST", "USER_GENERATE_TOKEN" ],
    subjectPermissions: [
        {
            subject: "507f191e810c19729de860ea",
            permissions: [ "WORK_CREATE", "GRADE_LIST",
                "ANNOUCEMENT_DELETE" ]
        }
    ]
}
```

Výpis kódu 3 Ukázka objektu s autorizačními právy

4.3 Klientská část

Klientská část byla dle návrhu vytvořena jako webová stránka ve frameworku Vue za pomoci TypeScriptu (více o knihovnách v kapitole 3.1.1), vlastního designu a pomocných knihoven. Jako správce projektu jsem si vybral Vite, jako alternativa šlo použít např. VueCLI, ale Vite je doporučovaný standard samotnými vývojáři Vue, protože generuje menší knihovny díky treeshakingu. Treeshaking je systém, kterým jsou odstraněny nepotřebné závislosti a další soubory.

Společně s Vue je taktéž použit balíček `vue-router`, který dovoluje ze SPA vytvořit vícestránkovou aplikaci, resp. dovoluje vytvořit více stránek uvnitř aplikace, mezi kterými se odkazuje pomocí URL adres. Tento balíček používám s tzv. history modem nastaveným na web history, který nutně potřebuje nastavit přesměrování požadavků na hlavní `index.html` soubor, který se o routování stará. Uvnitř `vue-router` jsou nastavené jednotlivé routy. Routy lze nalézt v souboru `/client/src/router/index.ts`. V zásadě jsou oddělené do jednotlivých částí dle zobrazení (viz kapitola 3.1.4).

Některé routy v sobě uchovávají informace o tom, pro jaký předmět jsou načtené. Router se stará o to, aby byly tyto předměty validovány a uchovány i při přechodu mezi stránkami. Router se stará o to, aby tento stav byl uložen do perzistentního úložiště i při přechodu na stránky bez předmětu (např. globální administrační části).

Pro vytvoření globálního úložiště dat, ke kterým mohou přistupovat veškeré komponenty, jsem se rozhodl používat knihovnu `Pinia`. Alternativou byla např. knihovna `Vuex`, která je však již nepodporovaná a samostatně není dostatečně funkční pro zápis komponent pomocí kompozičního API. Globální úložiště dat v tomto případě, dle návrhu, slouží taktéž jako konvertor mezi daty v aplikaci a ty, která se budou odesílat na serverovou část.

Byly použity i další knihovny, které popisuji níže. Vybral jsem pouze nejzajímavější z nich – veškeré knihovny (i ty, určené pouze pro vývoj) lze nalézt v souboru `/src/client/package.json`.

tiptap je knihovna určena pro tvorbu WYSIWYG editoru. Knihovna vytváří headless editor, to znamená, že si design prvků a uživatelského prostředí určuje programátor sám. Pro knihovnu existují další nástavby, které přidávají další prvky do editoru.

jsonwebtoken je knihovna určena pro podepisování a kontrolu autentizačních tokenů. Využívá se je k validaci tokenu, který přijde ze serveru, a získávání informace z něj.

moment je knihovna pro lepší manipulaci datem a časem. Zejména je využívána pro validaci vztahů mezi daty, formátováním a vypisováním lokalizovaných hlášek.

chart.js je knihovna pro vykreslování různých grafů a statistik. Používá se pro vykreslování statistik hodnocení a v budoucnu bude využito pro administrační statistiky.

4.3.1 Implementace designu a komponent

Z návrhu designu (viz kapitola 4.1) bylo nutné implementovat různé komponenty s funkcemi. Ze seznamu v kapitole 3.6 byly implementovány všechny, navíc byly přidány:

upozornění komponenta pro zobrazování chyb a dalších hlášek.

karty základní obalovací prvek pro uživatelské prostředí, obsahuje nadpisy a další obsah.

odznak komponenta určená po označování prvků krátkým textem či ikonkou.

navigační menu komponenta pro navigační menu – hlavní, vedlejší a jejich spojení při responzivním designu.

výběr z možností komponenta pro výběr možností ze seznamu.

záložky komponenta pro zobrazování horizontálního menu, ve kterém lze překlikávat mezi záložkami – určeno pro vykreslování jiných komponent.

Pro každou designovou komponentu byl vytvořen speciální soubor. Veškeré komponenty lze najít ve složce `/client/src/components/design/`. Komponenty používají scoped styly a Sass ve formátu SCSS. Uvnitř stylů komponent se používají CSS proměnné, které jsou definované pro jednotlivé barevné režimy uvnitř globálního souboru `/client/src/assets/themes.scss`. Globální styly se nacházejí ve dvou dalších souborech, a to `main.scss`, který obsahuje základní styly a stará se o importování dalších souborů, a `base.scss`, který obsahuje resetování stylů a různé pomocné třídy. Mezi pomocné třídy se řadí např. odsazení a různé responzivní zobrazení.

Soubor `editor.scss` obsahuje styly určené pro editor, resp. jeho vykreslování v různých částech webové stránky. Soubor `animations.scss` obsahuje animace, které se používají pro přechod mezi stavy a stránkami. Responzivita komponent a stylů je řešena pomocí media queries. Stránky jsou rozděleny do tří stavů – breakpointů, které jsou určené pro malá zařízení (telefony), střední zařízení (tablety, malé notebooky) a velká zařízení (desktopová zařízení a televize).

Ukázky výsledného implementovaného designu a uživatelského prostředí aplikace je možné nalézt v příloze D.

4.3.2 Komunikace s API

Klientská část se serverovou částí komunikuje pomocí REST Webového API. Jednotlivé endpointy, se kterými může aplikace komunikovat, jsou definovány v tzv. repozitářích, které jsou vždy určené jedné skupině entit. Repozitáře v podstatě určují jen jaká data s jakými hlavičkami budou zaslána na jaký endpoint. Uvnitř repozitářů se volá metoda pro zavolání HTTP požadavku pomocí metody `fetch`. Každé volání taktéž určuje, jaká data se při úspěšném volání očekávají, že přijdou z daného endpointu. Na obrázku 4 lze vidět část kódu, ve kterém se definuje volání pro vytvoření známky.

Repozitáře pracují pouze s textovými hodnotami a DTO (viz kapitola 4.5). O samotnou konverzi dat do DTO se starají globální úložiště dat – obchody, které i samotné repozitáře za pomocí pomocné třídy `Api` volají. Obchody se starají i o opačnou konverzi, tedy DTO do formátu, kterému rozumí klientská strana. Tato konverze je tvořena pomocí mapovacích tříd, které převádí DTO do Modelů. Model je vždy třída, která uchování potřebná data v použitelném formátu (např. místo data v textové hodnotě uchovávat objekt data a času) a mají pomocné metody, které např. dopočítávají či reprezentují data.

```
export class GradeRepository extends Repository {
    /**
     * Create a criteria.
     * @param subject Subject to create the criteria for.
     * @param data Criteria to create.
     */
    async create(subject: string, data: CreateGradeDTO) {
        return await this.makeRequest<CreateGradeSuccessDTO>(
            `${subject}/grade`,
            "POST",
            data
        );
    }
    // ...
}
```

Výpis kódu 4 Ukázka části repozitáře známek

Jednotlivé obchody se dělí na tři skupiny – globální, veřejné a administrační. Systém knihovny `Pinia` totiž chytře načítá pouze ta data, která aplikace nyní používá. Dřívější verze portálu všechna data načítala najednou, což nebylo vůbec efektivní. Nyní existuje více jak 15 různých obchodů, které se samostatně starají o různé věci. Existuje např. obchod pro ukládání stavu přihlášení, správa předmětů a skupin, správa prací v předmětu a podobné. Store se skládá z proměnných, sledovačů a akcí, které lze volat z komponent. S obchody komunikují ostatní soubory, komponenty nebo jednotlivé routy.

4.3.3 Stránky

Pro aplikaci byly vytvořeny níže uvedené stránky:

- správa
  - skupin, přiřazování skupin uživatelům
  - předmětů a jejich přiřazení ke skupinám
  - uživatelů, generování přihlašovacích kódů
  - oznámení v předmětu
  - známek a jejich kategorií v předmětu
  - hodnocení v předmětu, celkové hodnocení studentů
  - lekcí a jejich částí v předmětu
  - prací a jejich odevzdání v předmětu
- zobrazení
  - oznámení

  - hodnocení
  - lekcí (a jednotlivých částí)
  - prací
- nastavení uživatele a upozornění
- přihlášení a registrace

4.4 Serverová část

Serverová část byla implementována v Node.js za pomoci TypeScriptu a knihovny express pro tvorbu webového API (více o knihovnách a technologiích v kapitole 3.1.2). Serverová část taktéž komunikuje s MongoDB databází pomocí ODM knihovny `mongoose`. Pro tvorbu serveru byly použity i další knihovny, které popisuji níže. Vybral jsem pouze nejzajímavější z nich – veškeré knihovny (i ty, určené pouze pro vývoj) lze nalézt v souboru `/src/server/package.json`.

express-rate-limit knihovna slouží k omezení, kolikrát může daná IP adresa (př. jiný identifikátor) přistupovat k jednotlivým endpointům uvnitř aplikace.

express-slow-down knihovna slouží k záměrnému zpomalení požadavků, když jich na nějaký endpoint chodí příliš. Používá se s knihovnou `express-rate-limit`.

helmet knihovna nastavuje nejrůznější hlavičky a obstarává další bezpečnostní části aplikace. Nastavuje např. i CORS (viz kapitola 1.3.9), XSS a další cross-origin nastavení.

joi knihovna slouží pro validaci vstupu, zda splňuje formát a data mají správné hodnoty. V aplikaci slouží pro validaci vstupu, který přijde od klientské části (př. jiného rozhraní).

multer knihovna slouží k obstarávání souborů a dalších dat, která přicházejí od klientské části v jiném než textovém formátu. Automaticky hlídá velikosti dat, jejich umístění, jméno a přípony.

nodemailer knihovna slouží k zasílání e-mailů pomocí SMTP protokolu.

web-push knihovna slouží k zasílání tzv. push notifikací. Na klientské části jsou tyto notifikace poté přetvořeny do reálných notifikací na zařízení, které se používají např. pro oznamování nového hodnocení, viz kapitola 4.3.

typedi knihovna slouží pro vytvoření dependency injection, tedy způsobu předávání závislostí uvnitř aplikace.

4.4.1 Databázová schémata

Přesně dle návrhu byla serverová část rozdělena na repozitáře, kontrolery, routy a modely. Modely jsou definovaná schémata dat pro `mongoose` a využívají rozhraní, která definují jednotlivou strukturu dat, která se poté používají uvnitř aplikace. Společně s daty je

skoro v každé entitě vytvořena i metoda `toDTO`, která převádí data do přenositelné podoby (viz kapitola 4.5). Ukázka definice rozhraní, schématu a jeho registrace (bez komentářů) je zobrazena v kódu 5. Všechny modely a jejich schémata lze najít ve složce `/server/src/database/model`.

```
interface IGroup {
    name: string;

    toDTO(): GroupDTO;
}

const groupSchema = new Schema<IGroup>({
    name: { type: String, required: true },
}, {
    timestamps: true
});

groupSchema.method<HydratedDocument<IGroup>>('toDTO', function () {
    return {
        id: this.id.toString(),
        name: this.name,
    } as GroupDTO;
});

const Group = model<IGroup>('Group', groupSchema);
```

Výpis kódu 5 Definice rozhraní, schématu a registrace modelu

4.4.2 Kontrolery

Dále jsou na serveru definovány kontrolery, které se v podstatě starají o odpovědi na jednotlivá volání endpointů. Kontrolery jsem se rozhodl implementovat pomocí tříd, které dědí z obecného kontroleru pomocí přepisování metody, která přijímá požadavek a odpověď. Použití tříd mi dovolilo efektivně používat knihovnu `typedi` pro dependency injection. Kontrolery mají na práci různá data a validace. Ukázkový kontroler je vyobrazen v kódu 6. Samotný kontroler je označen jako služba v DI a je používána v routách pro importování.

Jednotlivé kontrolery jsou vždy definované po jednom souboru ve složce a podsložkách uvnitř `/server/src/controller`. Kontrolery jsou uskupeny po složkách podle toho, jak se sebou souvisí, resp. s jakou entitou primárně pracují dle REST návrhu API. Celkem bylo implementováno 84 kontrolerů.

4.4.3 Repozitáře

Repozitáře slouží k odstínění databáze a realizaci vyhledávaní a jiných dotazů. Repozitáře jsou na serverové části oddělené dle jednotlivých databázových schémat (či entit), se kterými pracují. Repozitáře dědí z obecné definice, která zaručuje hledání dokumentu dle jeho identifikátoru. Při dědění se taktéž repozitáři určuje typ modelu, se kterým pracuje. Všechny repozitáře lze najít ve složce `/server/src/database/repository`.

Pro vybírání dat se používají přímo metody vytvořené knihovnou `mongoose`, např. se jedná o `find`, `create` a další. Zkrácený výpis definice ukázkového repozitáře je přiložen v kódu 7.

```
@Service()
export default class ExampleController extends Controller {

    constructor(
        @Inject(() => ExampleService)
            private exampleService: ExampleService
    ) {
        super();
    }

    async handle(req: express.Request, res: express.Response) {
        this.exampleService.sendSmile();

        res.send("OK");
    }
}
```

Výpis kódu 6 Ukázková definice kontroleru

```
@Service()
export default
       class AnnouncementRepository extends Repository<IAnnouncement> {

    public async getById(id: string | Types.ObjectId) {
        return Announcement.findById(id);
    }
    // ...
}
```

Výpis kódu 7 Zkrácený výpis definice repozitáře

4.4.4 Routy

Routy slouží k definici jednotlivých endpointů, resp. toho, na jaké URL adrese se nacházejí, jaké middlewary a jaké kontrolery se zavolají. Tyto třídy využívají uvnitř `Router` z knihovny `express` a dovolují různé definice. Routy pomocí dependency injection taktéž získávají potřebné služby a kontrolery. Všechny definice rout lze najít ve složce `/server/src/routes/route`.

4.4.5 Služby

Pro účely odesílání e-mailů a push notifikací (viz kapitola 4.4) byly vytvořeny služby `EmailService` a `NotificaitonService`. Tyto služby používají šablony, tedy předdefinované sady údajů (např. pro e-maily se jedná o předmět, obsah e-mailu a případně další data), které jsou definované na jednom místě. Ukázka definice šablony je vyobrazena v kódu 8.

Pro odesílání e-mailů se používá knihovna `nodemailer`. Služba se stará o přihlašování a validaci těchto dat. Pro odesílání push notifikací se používá knihovna `web-push`. Služba se stará o masové rozesílání notifikací, včetně odesílání určitým skupinám, validaci odesílaných dat a podobně.

```
class EmailTemplates {
    public static AUTHENTICATION_REQUEST = {
        subject: 'Žádost o přihlášení',
        filename: 'authentication-request.html',
    } as EmailTemplate<{
        code: string;
        name: string;
        link: string;
    }>;
    // ...
}
```

Výpis kódu 8 Zkrácený výpis definice šablon e-mailů

4.5 Přenos entit a dalších dat

Na serverové části se v kontrolerech musí přidávat na klientskou část jednotlivé modely, resp. databázové entity. Data, která se mezi částmi předávají označuji jako Data Transfer Object (DTO). DTO se na klientskou část předávají ve formátu JSON. DTO se dělí na dva typy – ty, které posílá server, a ty, které klient. DTO obsahují pouze primitivní datové typy a jsou uzpůsobena tak, že je v dané podobě neukládá ani jedna strana.

Při přenášení dat z klientské části probíhá validace dat pomocí knihovny `joi`. Jednotlivá data poté zpracovávají jednotlivé kontrolery – např. voláním metod v repozitářích. Kontrolery, pokud mají co odpovídat, taktéž odpovídají formátem DTO. Odpovědi

se dělí taktéž na dvě části – ty, které označují odpověď endpointu, a ty, které popisují obsah entit. Pokud endpoint odpovídá, soubor končí na `SuccessResponseDTO.ts`. DTO jsou definovány jako rozhraní.

Tyto definice DTO jsou uloženy ve složkách `/server/lib/dto` a `/client/lib/dto`. Definice jsou tedy sdílené mezi oběma částmi aplikace tak, aby každá věděla, co kam má posílat a s jakými daty. Jednotlivé DTO jsou rozdělené do složek, se kterými souvisí.

Společně s definicemi DTO se ve složkách `lib/` nachází ještě seznam pravomocí (viz kapitola 4.2.2) a seznam chyb, které server zasílá klientům pro jednodušší kontrolu.

4.6 Optimalizace pro webové vyhledávače

Optimalizace pro webové vyhledávače (anglicky Search Engine Optimalization, zkratkou „SEO“) je proces modifikace stránky, kdy je upravován obsah, nastavení a kód tak, aby stránky byly ve webových vyhledávačích umisťovány na lepších pozicích (ne nutně na prvních). Jako součást práce jsem se jí zabýval na konci, až po uživatelském testování a opravě chyb (viz kapitola 5).

Pro vylepšení SEO jsem provedl několik akcí, které níže popisuji. Prvně jsem vytvořil soubor `/client/public/robots.txt`, který určuje nastavení pro roboty, kterým se mohou řídit a určuje stránky, které chce vložit do procesu indexace. Zde můžeme doporučit vyhledávacím robotům chování a třeba jim říct, aby onu stránku ignorovali. Pro tuto stránku jsem vytvořil jednoduchý soubor, který dovoluje všem robotům indexovat všechny stránky.

Výsledná aplikace je tvořena jako SPA a zároveň funguje v režimu, že veškerý obsah (a jeho HTML) je vykresleno na stránce až JS skriptem, který ji pomocí virtuálního DOM obstarává. Některé vyhledávače používají systém, kdy stránky opravdově po přijmutí HTML spouští v prohlížeči jako webovou stránku, a až po chvíli, či až se vše načte, provedou přečtení HTML a jeho zpracování. Velmi mnoho prohlížečů však toto nedělá, a proto si na stránce přečtou prázdné HTML. Tento problém lze vyřešit pomocí Server-side rendering, který dovoluje stránky hostovat na serveru, kde probíhá vykreslování a získávání dat k zobrazení. Se SSR je v podstatě nutné počítat od začátku vývoje a já se jej rozhodl nepoužít z důvodu toho, že SSR odebírá výhody „thick client“, tedy, že vše důležité dělá klient a server pouze poskytuje data.

Druhou alternativou je tzv. prerender, který funguje na principu toho, že když přijde na server požadavek od vyhledávače (při nastavení i od uživatele), je stránka poslána do virtuálního prohlížeče, které ho vykreslí a vrací již vybudované HTML. Pro prerender je potřeba používat další službu a je potřeba server nastavit. Pro aplikaci jsem se rozhodl použít prerender z důvodu jednoduchosti splnění požadovaného cíle.

Další částí bylo vytvoření seznamu stránek, které se poskytují vyhledávači a dalším robotům, a to soubor `sitemap`. Sitemapa obsahuje odkazy na všechny stránky, jejich prioritu, na jaké URL se nachází a jak často by je robot měl navštěvovat [6]. Sitemapu bylo nutné vytvořit z důvodu, že HTML, které se na stránce nachází po generaci, neobsahuje všechny viditelné odkazy. V implementované aplikaci se nachází spousta odkazů v různých dialozích, menu, které se v HTML objevují pouze při kliknutí, což samozřejmě roboti nedělají a nemají šanci zjistit. Poskytnutím všech klíčových stránek (např. už jen jednotlivé předměty) se pomůže robotovi dále odkazy na stránkách hledat. Vzhledem

k tomu, že stránka obsahuje uživatelský obsah, bylo by nutné seznam stránek nějak generovat na serveru.

Proto jsem se rozhodl vytvořit robota, který při zapnutí projde celé webové stránky z určité URL adresy a ty zapíše do `sitemap.xml`. Z těchto stránek si již vyhledávače mohou najít další odkazy na stránce a potřebné informace. Robotovi lze určit i webové stránky, které nemají přímý odkaz na stránce (viz výše) a ty taktéž projde. Tento soubor je nyní nutné ručně přesouvat, ale v budoucnosti může být automaticky nasazována.

V poslední části SEO byla samotná úprava obsahu a stránek. Dle testování (viz 5.1) bylo nutné opravit různé věci, které stránky kazily. Jednalo se např. o absenci popisků pro tlačítka, špatné umístění prvků v seznamech (což zhoršuje čtení ve čtečkách), špatně viditelný text a podobně.

4.7 Průběžná práce

Při realizaci a vývoji aplikace jsem používal několik nástrojů. Využíval jsem verzovací systém git, který byl hostován na službě GitHub[3]. Celý vývoj byl rozdělen do tří repozitářů, `website-client`, `website-server` a `website-docker`. V prvním repozitáři se nachází kód klienta, ve druhém kód serveru a třetí sloužil pro nastavení nasazení v Dockeru, viz kapitola 4.8.3. Ve všech repozitářích vzniklo více jak 200 změn (tzv. commitů). Na službě GitHub jsem nevyužíval žádných jiných týmových a projektových nástrojů. Veškeré plány a seznam věcí, které chci implementovat a přidat, jsem ukládal a plánoval uvnitř služby ClickUp[4].

4.8 Sestavení aplikace

Tato kapitola popisuje procesy kompilace a minimalizace zdrojových kódů do podoby, která je vhodná pro použití v produkčním prostředí. Klientská a serverová část jsou v jakémkoliv prostředí na sobě závislé a je nutné je provázat. Serverová část je závislá na databázi MongoDB. Instalaci jednotlivých částí není nutné provádět na jednom zařízení.

4.8.1 Klientská část

Pro spuštění v prohlížeči musí být klientská část sestavena do podoby, kterým prohlížeče rozumí. Při vývoji nabízí balíčkovací systém `Vite`, který jsem využil, možnost automatického (a chytrého) překompilovávání za běhu za pomocí hot-reload. Klientská část se zkompiluje pomocí příkazu `npm run build`, který volá `vite build` s dalším nastavením, jako např. cílovým použitím (testující, produkce, ...), které načítá proměnné prostředí. Tento příkaz obstarává veškerou kompilaci a kompresi souborů.

Po úspěšném spuštění příkazu se ve složce `/client/dist` vytvoří složka se soubory, které již lze nahrát na hosting webových stránek. Protože je aplikace SPA, což znamená, že používá pouze jedno HTML pro reprezentaci celé aplikace, a že používám `vue-router` s nastavením `web-history`, které tvoří URL normální způsobem jako jiné typy stránek, je nutné nastavit webový server tak, aby vždy ukazoval na soubor `index.html`.

[3] k dispozici na webové stránce `https://github.com/`
[4] k dispozici na webové stránce `https://clickup.com/`

4.8.2 Serverová část

Serverovou část je před spuštěním na serveru nutné přetransformovat do podoby, která dovoluje běh. Serverová část je psaná v jazyce TypeScript, a proto je nutné před spuštěním překonvertovat tento kód psaný v metajazyce do čistého JavaScriptu, resp. Node.js prostředí. Zavoláním příkazu `npm run build` se spustí kompiler TS, který zkonvertuje kód do JS. Výsledek bude k dispozici ve složce `dist`, ze které lze celou aplikaci spouštět pomocí Node.js.

Samotný běh ovlivňují proměnné prostředí, které se nastavují dle proměnné prostředí `NODE_ENV`. Proměnné prostředí se čtou buď z globálních proměnných, nebo ze souboru `.env`. Serverová část potřebuje většinu těchto proměnných mít nastavenou pro správné spuštění. V kódu 9 lze nalézt ukázku souboru pro nastavení proměnných

```
PORT=3000
DATABASE_URL="mongodb://cajthamlportal:.../db"
SALT_ROUNDS=15
JWT_SECRET="yoko>bambi"
SMTP_HOST="smtp.server.cz"
SMTP_PORT="465"
FRONTEND_DOMAIN="https://ssps.cajthaml.dev"
```

Výpis kódu 9 Zkrácený výpis definice proměnných systému

4.8.3 Docker

Pro nasazení lze taktéž využít službu Docker, která slouží k virtualizaci kontejnerů na úrovni operačního systému. Společně s Docker compose, která zajišťuje nasazení více kontejnerů v jednom systému, jsem vytvořil soubor `/platform/docker-compose.yml`, který se stará o nasazení celé aplikace. V tomto souboru se nachází definice kontejnerů pro zveřejnění klientské, serverové části a taktéž i databáze MongoDB, na které je server závislý. Pro prerender docker nepoužívám, protože mám službu nasazenou už jinde, která používá knihovnu `prerender`[5]. Klientská část se taktéž stará o URL frontendu pomocí přepisování, resp. přesměrování dotazů vždy do hlavního souboru `index.html`.

V tomto konfiguračním prostředí se odkazuje na Docker obraz frontendu a backendu. Tyto obrazy vznikají nezávisle na sobě a jsou nahrávány do registru Github Packages.

Pomocí GitHub Actions, které dovolují tvořit automatické chování v závislosti na repozitáři, jsem vytvořil několik skriptů. Každý repozitář má svůj skript, který po nahrání změn automaticky zkompiluje Docker kontejner a nahraje ho do privátního registru balíčku pro organizaci, ve kterém se repozitáře nachází. Tyto a další skripty lze najít ve složkách `/*/.github/workflows/*.yml`.

Uvnitř repozitáře klientské části se taktéž nachází skript, který při ručním spuštění zkompiluje aplikaci do formátu `.apk`, který slouží jako nativní aplikace pro operační systémy Android. Aplikace se sestavují pomocí knihovny Capacitor (viz kapitola 3.1).

[5] k dispozici na stránce `https://github.com/prerender/prerender/`

Tento balíček se po zkompilování taktéž podepisuje, aby mohl být nahrán na obchod s aplikacemi.

Aplikace pro platformy iOS knihovna Capacitor podporuje, avšak její testování je složitější – kvůli mé absenci takového zařízení, potřeby používat další software. Zároveň s tímto testováním nemám žádné zkušenosti. Proto tedy aplikace iOS prozatím nebyla vyzkoušena ani zkompilována.

Skript uvnitř repozitáře pro „platformu“ automaticky nahrává do testovacího (a v budoucnosti produkčního) prostředí právě pomocí Docker compose a již zkompilovaných kontejnerů. Skript automaticky nahrává změněné soubory a nastavení Docker compose, které na serveru spouští.

Pro nasazení aplikace je nutné správně nainstalovat Docker, Docker compose, závislosti, přihlásit se do Github packages registru a zavolat příkaz `docker-compose up`. Docker compose dává stránku k dispozici na portu `3334`, server na `3333`. S těmito porty může pracovat další vrstva a určovat jim např. domény.

4.9 Nasazení aplikace

Testovací i budoucí produkční prostředí bude nahráno na mém serveru, který jsem zakoupil pro své účely a projekty. Na serveru je GNU/Linux s distribucí Debian a dalšími systémovými nastaveními a prostředky. Pro hostování záznamů DNS domény a zabezpečení jsem se rozhodl použít službu Cloudflare[6].

Aplikace byla na tomto serveru pomocí Dockeru (viz kapitola 4.8.3) nasazena na testovací prostředí, ve kterém je možné si ji zobrazit. Stránka je k dispozici na adrese `https://ssps.cajthaml.dev/`. Pro reverse proxy aplikace byla použita služba Apache2. V aplikaci se nachází testovací data a je možné se zaregistrovat a provádět jakékoliv akce. Pro účely testování jsem založil tři testovací účty. Pomocí těchto účtů je možné si aplikaci otestovat na lokální a testovací platformě. První účet je student, který studuje ukázkový předmět (`Testovací předmět`) a druhý je vyučující tohoto předmětu. Tyto účty jsou k dispozici i na testovací platformě. Na lokální platformě při inicializaci databáze bude vytvořen třetí, administrátorský účet, který může spravovat předměty, role, uživatele a další kritické části aplikace. Tyto účty se tvoří automaticky na serverové části. Těmto účtům jsem se rozhodl vypnout možnost změny hesla.

Vytvořené účty s přihlašovacími e-maily a hesly:

Student e-mail: `student@test.cajthaml.eu`, heslo: `testtest`

Vyučující e-mail: `vyucujici@test.cajthaml.eu`, heslo: `testtest`

Administrátor e-mail: `administrator@test.cajthaml.eu`, heslo: `testtest`

Původní aplikace je stále k dispozici na adrese `https://ssps.cajthaml.eu/`. Nová aplikace bude po další implementaci funkcionalit (viz kapitola 6) nasazena do produkce příští školní rok a nahradí tím stávající aplikaci.

[6] k dispozici na webové stránce `https://cloudflare.com/`

Kapitola 5
Testování

Tato kapitola se zaměřuje na proces testování vytvořené aplikace v realizaci, dle analýzy a návrhu. Bude popsáno průběžné testování, provedeno uživatelské testování a popsáno testování automatické.

5.1 Průběžné testování

V průběhu tvorby aplikace bylo na klientské i serverové části používána funkce hot-reload, která dovoluje aplikaci neustále inkrementálně kompilovat. Na klientovi je funkcionalita automaticky zprovozněna díky vybranému frameworku `Vite`. Na serverové části jsem tak učinil pomocí vývojové knihovny `nodemon`.

Při tvorbě určitých funkcionalit jsem vždy postupoval tak, že jsem danou funkcionalitu, zda-li potřebovala práci se serverovou částí, implementoval na serveru, a poté postupoval na klientské části. Na serverové části jsem REST API kontroloval pomocí programu Postman[1], který dovoluje zasílání HTTP požadavků na dané URL adresy. Na klientské části jsem při vývoji věci ihned testoval ve webovém prohlížeči.

Akce jsem vždy zkoušel v různých stavech přihlášení a nastavení autorizace. Po vytvoření funkcionality jsem vždy zpětně kód refaktoroval tak, aby byl lépe čitelný a rozšiřitelnější.

Po vytvoření klientské části jsem aplikaci vyzkoušel v automatickém nástroji pro zkoušení kvality webových stránek Google Lighthouse[2]. V tomto testu jsem obdržel při prvním spuštění na 3 různých stránkách na telefonních zařízeních, tak i na desktopu průměrné skóre 86 ze 100 bodů. Stržené body byly zejména za chybějící popisky odkazů, špatné nastavení cache a kontrastní barvy.

Věci, které se nástroji nelíbily, jsem opravil. Poté na třech různých stránkách bylo průměrné skóre 98 ze 100 bodů. Proto jsem se rozhodl otestovat většinu stránek, které se nacházejí v aplikaci a následně jsem na 24 různých stránkách získal průměrné skóre 95 ze 100 bodů. V tomto stavu jsem na testovací produkci již s hodnocením spokojen a mohl jsem přistoupit k dalším opravám a testování.

Stránky byly tvořeny pro moderní prohlížeče, ale webová aplikace podporuje i řadu starších verzí. Webové stránky podporují i většinu operačních systémů, jak na telefonních

[1] program Postman je k dispozici na webové stránce `https://www.postman.com`

[2] k dispozici na webové stránce `https://developer.chrome.com/docs/lighthouse/overview/`

Tabulka 5.1 Seznam podporovaných prohlížečů

Název prohlížeče	Verze	Poznámky
Chrome	≥ 108	–
Vivaldi	$\geq 5.5.2805.50$	–
Internet Explorer	×	nepoužitelné díky použitím proměnným v CSS, flexboxu a dalších
Opera	≥ 87	–
Firefox	≥ 100.3	–
Safari	≥ 15.6	problémy s notifikacemi

zařízení, tak i desktopu. Webové stránky jsem vyzkoušel na různých starších verzích a různých operačních systémech. Výsledky této kontroly je možné vidět v tabulce 5.1.

Pro každý prohlížeč jsem otevřel prohlížeč dané verze a zkoušel, zda webové stránky fungují. Začínal jsem od nejmenší verze, která byla vydána za posledních 5 let a kterou samotná firma, která prohlížeč vytváří, označuje za podporovaný a opravuje mu závažné chyby. Pokud na verzi prohlížeče stránka fungovala, kontroloval jsem verzi, která je novější o řadu měsíců. Většinu prohlížečů jsem testoval na svých zařízeních a pomocí služby BrowserStack[3].

V tabulce 5.1 jsem zmínil i prohlížeč Internet Explorer, který již vývojářem Microsoft není podporovaný a tedy se nejedná o žádný problém. Největší problémy použití byly právě s ním a s prohlížečem Safari, který má velmi omezené použití notifikací z prohlížeče. Problémy nelze z pohledu práce vyřešit.

5.2 Nehlídané uživatelské testování

Po schválení tohoto zadání bakalářské práce jsem se rozhodl z řad mých studentů požádat o připojení k testování bakalářské práce. K testování se připojilo mnoho studentů a některé jsem poté již musel odmítat. Tito testeři reagovali na různé požadavky, které jsem jim v průběhu tvorby práce představoval. Diskutovali jsme nad použitím a designem různých částí aplikace, které dříve označili jako špatné (viz dotazník v kapitole 2.1.3.3).

Nehlídané uživatelské testování je proces, kdy stránku testují více zkušení uživatelé pro odhalení klíčových chyb v aplikaci před tím, než je zveřejněna nebo testována s pozorovatelem. Pro toto testování jsem se rozhodl z důvodu velkého počtu testerů, kteří mají čas aplikací projít ve svém volném čase bez jakýchkoliv jiných faktorů a najít chyby, které by hlídané uživatelské testování neodhalilo. Tímto testováním jsem mohl automaticky otestovat další počet zařízení, operačních systémů včetně webových prohlížečů a verzí.

Dne 10. 4. 2023 jsem svým testerům zveřejnil první testovací verzi aplikace, a následující týden probíhalo toto testování. Testování odhalilo 58 chyb a různých požadavků, které testeři našli. Jednalo se zejména o různé nefunkční části, překlepy, problémy s UI, UX a obdoby. Z tohoto seznamu jsem většinu chyb opravil, avšak zbývají různé části, které jsou spíše na dlouhodobé přidávání funkcionalit – např. se jedná o drobečkové menu či pokročilejší lekce. O těchto budoucích rozšířeních si lze přečíst více v kapitole 6.

[3] k dispozici na stránce `https://www.browserstack.com`

Po opravách bylo testování uskutečněno ještě jednou, kdy se našly další chyby, které jsem znovu opravil. Do testování jsem poté začal přibírat i další studenty a mé přátele, abych pokryl více názorů a větší spektrum možných uživatelů.

5.3 Uživatelské testování

Uživatelské testování je velmi častou metodou testování přístupnosti webových stránek a aplikací. Uživatelské testování cílí na to, jak by se aplikace měla používat a bere v potaz názory uživatelů. Cílem je zjistit vážné nedostatky, které mohou obyčejným uživatelům způsobit problémy s používáním aplikace.

Uživatelské testování potřebuje účastníky testování, kteří jsou sledováni při interakci s aplikací či webovou stránkou. Účastník prezentuje své myšlenky, jak se snaží dané problémy či postup aplikovat.

Každé testování je předem známé a je rozděleno do testovacích scénářů. Testovací scénáře popisují mimo jiné i postup provedení dané věci. Cíle testovacích scénářů jsou obeznámeny i testovanému účastníkovi. Během testování si pozorovatel zapisuje poznámky a na konci sděluje testovanému výsledky.

5.3.1 Testovací scénáře

Testovacím scénářům se určuje:

- účel
- časový limit
- podmínky
- kroky k realizaci
- očekávaný výstup

Během provádění testovacího scénáře je nutné do akcí testujícího nezasahovat a odpovídat na otázky tak, aby nebyl test kontaminován. Účastník nesmí mít osobní zájem na výsledek testování a musí být objektivní.

Před testováním je účastník s účelem testu seznámen a poté jej provádí pomocí definovaných kroků k realizaci. Účastník svoje myšlenky a postupy popisuje a zdůvodňuje. Pozorovatel si zapisuje poznámky, které na místě, a i po dokončení testování vyhodnocuje.

Testovací scénáře se odkazují na uživatelské případy z kapitoly 2.3. Níže je ukázka formátu, ve kterém jsem testovací scénáře vytvářel. Seznam všech testovacích scénářů lze nalézt v příloze B. Bylo vytvořeno 16 scénářů, kdy TC1 a TC4 jsou určeny pro každého uživatele, TC5 až TC11 jsou určeny pro studenty a TC12 až TC16 pro vyučující.

TC1 – Vytvoření uživatelského účtu

Cíl: Účelem je vytvoření uživatelského účtu uvnitř vytvořené aplikace.

Časový limit: do 5 minut

Podmínky: Aplikace je ve stavu, kdy uživatel není přihlášen a uživatel se nachází na hlavní stránce aplikace. Uživatel nemá zaregistrovaný žádný účet.

Kroky:

1. Účastník nalezne tlačítko pro přihlášení, které je označeno šipkou v navigaci.
2. Účastník klikne na tlačítko `Vytvořit si účet`.
3. Účastník vyplní registrační formulář.
4. Účastník formulář odešle kliknutím na tlačítko `Registrovat se`.
5. Účastník zkontroluje svojí e-mailovou schránku pro e-mail obsahující informace o aktivaci.
6. Účastník si pomocí pokynů z e-mailu aktivuje účet.

Očekávané výsledky:

1. V aplikaci se vytvořil nový účet se zadanými údaji.
2. Na zadaný e-mail aplikace odeslala e-mail s žádostí o potvrzení e-mailu a aktivaci účtu.
3. Účet je aktivovaný a lze se na něj přihlásit.

5.3.2 Účastníci testování

K uživatelskému testování implementované aplikace jsem pozval pět účastníků. Dva z těchto účastníků se s touto verzí aplikace předtím ještě nesetkalo. Jeden z účastníků se v průběhu práce nacházel ve skupině testerů (viz kapitola 5.2). Pro ochranu údajů účastníků jsou některé jejich údaje anonymizovány.

Účastnice 1 Kristýna Dřevikovská, studentka SSPŠaG, 16 let, členka týmu testerů. Ke koníčkům patří webové aplikace, poezie, knihy a sport. Ovládá počítačové programy a internet na vysoké úrovni. Je seznámen s existující aplikaci portálu a používá ji.

Účastník 2 Alex Naxera, student SSPŠaG, 17 let. Ke koníčkům patří 3D tisk a programování. Ovládá počítačové programy a internet na vysoké úrovni. Je seznámen s existující aplikaci portálu a používá ji.

Účastník 3 Vojtěch Šulc, student SSPŠaG, 17 let. Ke koníčkům patří sport, historie a vývoj videoher. Ovládá počítačové programy a internet na vysoké úrovni. Je seznámen s existující aplikaci portálu a používá ji.

Účastník 4 Učitel technických předmětů na SSPŠaG, 24 let. Ke koníčkům patří programování, auta a procházky. Ovládá počítačové programy a internet na vysoké úrovni. S portálem se nikdy dříve nesetkal a nepracoval s ním. Používá však další zmíněné ŠIS jako Bakaláři, Teams či Moodle.

Účastnice 5 Marie Mrázová, důchodkyně, 65 let. Ke koníčkům patří turistika, zahrádkářství a vědomostní soutěže. Ovládá počítačové programy a internet na mírné úrovni. S portálem se nikdy dříve nesetkala a nepracovala s ním, nepoužívá zmíněné ŠIS.

5.3.3 Průchody účastníků

V této části jsou popsány jednotlivé průchody všech účastníků uživatelského testování. Každý testovací scénář vyhodnotil jak účastník, tak pozorovatel hodnocením 1 až 5, kdy 1 znamená velmi intuitivní rozhraní (scénář splnil účastník samostatně, nepotřeboval navádět) a 5 znamená velmi neintuitivní rozhraní (scénář účastník nesplnil ani při pomoci pozorovatele). Hodnocení pozorovatele je zapisováno k většímu objektivnímu hodnocení průchodu daného účastníka, když účastník např. přehlédne nějakou klíčovou funkcionalitu či přeskočí určitou část.

Před provedením scénáře je scénář vysvětlen a aplikace je přeměněna do stavu, který daný scénář vyžaduje. Kroky k realizaci scénáře jsou odhaleny pouze v případě, že účastník neví, jak pokračovat ve vyřešení cíle.

5.3.3.1 Průchod účastnice 1

Testování účastnice 1 proběhlo dne 23. 4. 2023 v 20.40. Účastnice byla na test připravena a seznámil jsem jí s jeho průběhem. Vysvětlil jsem proces testování a jeho následného hodnocení. Účastnice postupovala v TC1 až TC11 postupně a po splnění cíle hodnotila aplikaci. Vzhledem k tomu, že se účastnice nachází v seznamu testerů, tak s aplikací již měla zkušenosti, a proto hledala chyby nad rámec toho, co již bylo ohlášeno v minulých nehlídaných testováních.

Testování TC1-TC4, TC6-TC7 a TC10-TC11 proběhlo zcela v pořádku. Účastnice měla problém v TC5, kdy nerozuměla částečně funkcionalitě předvídače a stěžovala si na umístění kategorií na stránce. V TC8 přišla konfúze s tím, kde se vlastně nachází nastavení uživatelského účtu. Účastnici jsem musel navigovat přes správnou ikonku, o které si myslela, že slouží k něčemu jinému. Největší problémy přišly s testem TC9, kdy v prohlížeči Safari nefungovala upozornění, o čem jsem již věděl (viz kapitola 5.2).

V tabulce 5.2 lze nalézt jednotlivá hodnocení testovacích scénářů účastníkem, které provedl po každém průchodu scénáře. Vzhledem k tomu, že účastnice již stránku viděla, vidím hodnocení velmi kladně. Většina chyb je méně závažná a jednoduše opravitelná. Chybou s prohlížečem Safari se budu nadále zabývat. Celkově účastnice aplikaci hodnotila velmi kladně, oceňovala možnost barevných režimů i přehlednost.

5.3.3.2 Průchod účastníka 2

Testování účastníka 2 proběhlo dne 23. 4. 2023 v 18.30. Účastník byl na test připraven a seznámil jsem ho s průběhem a hodnocením. Účastník postupoval v TC1 až TC11 postupně a po splnění jednotlivých cílů hodnotil aplikaci.

Testování TC2-TC4, TC6 a TC10-TC11 proběhlo zcela v pořádku. Ihned v prvním testování TC1 jsem zpozoroval problémy s hledáním přihlašovacího tlačítka. Účastník preferoval použití oznámení s odkazem k přihlášení, místo použití tlačítka v hlavní navigaci. V TC1 si účastník si stěžoval na absenci tlačítka pro registrování (a tedy reference pouze na přihlašování). V TC2 měl účastník uložené přístupové údaje z TC1 tedy je nemusel zadávat.

V TC5 měl účastník problém s hledáním aktuálního hodnocení a komentářů. Nechápe sumarizační známky a říkal, že by měly být lépe vysvětleny. Při TC7 účastník zkoušel „drag and drop“ nahrání souborů a při tom nahrál prázdný soubor, který poté

při stáhnutí byl taktéž prázdný a myslel si, že se jedná o chybu systému. V TC9 byl problém s tím, že nepřišla prvotní notifikace.

Tabulka 5.2 Hodnocení scénářů účastníkem 1

Scénář	Hodnocení účastníka	Hodnocení pozorovatele
TC1	1	1
TC2	1	1
TC3	1	1
TC4	1	1
TC5	2	3
TC6	1	1
TC7	2	3
TC8	1	1
TC9	4	3
TC10	1	1
TC11	1	1
průměr	1,45	1,55

V tabulce 5.3 lze nalézt jednotlivá hodnocení testovacích scénářů účastníkem, které provedl po každém průchodu scénáře. Účastník v podstatě neměl žádné problémy s aplikací a velmi kladně ji hodnotil. Při průchodu posledních testových scénářů neměl žádné připomínky a nevěděl ani, jak své pocity popsat.

Tabulka 5.3 Hodnocení scénářů účastníkem 2

Scénář	Hodnocení účastníka	Hodnocení pozorovatele
TC1	2	1
TC2	1	1
TC3	1	1
TC4	1	1
TC5	2	3
TC6	1	1
TC7	1	1
TC8	1	1
TC9	1	1
TC10	1	2
TC11	1	1
průměr	1,18	1,27

5.3.3.3 Průchod účastníka 3

Testování účastníka 3 proběhlo dne 23. 4. 2023 v 19.45. Účastník byl na testování připraven a seznámil jsem ho s průběhem testů a s procesem hodnocení. Účastník postupoval v TC1 až TC11 postupně a po splnění jednotlivých cílů hodnotil aplikaci. Účastník v průběhu celého testu svévolně webovou aplikaci porovnával s již existující verzí.

Bez komplikací proběhly pouze testy TC3, TC6 a TC8. Při registraci v TC1 měl účastník problém se zadáváním údajů do špatného (přihlašovacího) formuláře. Při TC2 si stěžoval na špatnou ikonku přihlášení a údaje se mu automaticky vyplnily pomocí automatického doplňování prohlížeče. V TC4 student upozorňoval na absenci označení předmětů, které právě studuje. Podobný problém zmiňoval v TC5, kde upozornil na překlep a na dle něj špatně umístěnou lokaci kategorií. V TC7 upozornil na absenci stavu odevzdání práce studenta. TC9 nefungoval v prohlížeči Brave a účastník se problém snažil sám vyřešit, však neúspěšně. TC10 měl stejný problém s prvním načtením, který jsem připsal problémům s prohlížečem. V TC11 účastník upozornil na absenci označování klíčových slov v kódu a chybějící prokliky odkazů.

V tabulce 5.4 lze nalézt jednotlivá hodnocení testovacích scénářů účastnice, které provedla po každém průchodu scénáře. Většina problémů s aplikací dle mého stojí na tom, že student porovnával dvě aplikace, které nejsou na sebe navázané. Cílem bylo portál navrhnout lépe, než byl ten původní a je jasné, že funkcionality se nebudou chovat zcela stejně. Aplikaci celkově účastník hodnotil ale kladně. Vytyčené chyby nepovažuje za zásadní.

Tabulka 5.4 Hodnocení scénářů účastníkem 3

Scénář	Hodnocení účastníka	Hodnocení pozorovatele
TC1	2	1
TC2	1	1
TC3	1	1
TC4	2	2
TC5	3	2
TC6	1	1
TC7	2	2
TC8	1	1
TC9	4	3
TC10	3	1
TC11	2	2
průměr	2	1,55

5.3.3.4 Průchod účastníka 4

Testování účastníka 4 proběhlo dne 4. 5. 2023 v 8.45. Účastník byl na testování připraven a seznámil jsem ho s průběhem testů a procesem hodnocení. Účastník postupoval v TC1-TC4 a TC12-TC16 postupně a po splnění jednotlivých cílů hodnotil aplikaci.

Bez komplikací proběhly testy TC1-TC4 a TC13. Při průchodu TC12 a zapisování hodnocení si účastník stěžoval na zpomalené zapisování, kdy mezi jednotlivými studenty nelze jednoduše přecházet. Při průchodu mezi všemi administračními stránkami si účastník stěžoval na nepopisné ikony a absenci popisků toho, co jednotlivá tlačítka dělají. Lekce v průchodu TC16 označil účastník za „zajímavé“ a že by si na zápis částí musel zvyknout.

V tabulce 5.5 lze nalézt jednotlivá hodnocení testovacích scénářů účastníkem, které provedl po každém průchodu scénáře.

Tabulka 5.5 Hodnocení scénářů účastníkem 4

Scénář	Hodnocení účastníka	Hodnocení pozorovatele
TC1	1	1
TC2	1	1
TC3	1	1
TC4	1	1
TC12	2	1
TC13	1	1
TC14	2	1
TC15	2	2
TC16	2	2
průměr	1,44	1,22

5.3.3.5 Průchod účastnice 5

Testování účastníka 5 proběhlo dne 7. 5. 2023 v 22.00. Účastník byl na testování připraven a seznámil jsem ho s průběhem testů a jednotlivého procesu hodnocení. Účastník postupoval v TC1-TC4 a TC12-TC16 postupně a po splnění jednotlivých cílů hodnotil aplikaci.

Ihned v prvním testu se účastnice musela přenést přes systém a pochopit jeho rozložení. V prvních minutách jí nepřišlo přehledné a musela se po rozhraní podívat tak, aby pochopila, co má dělat. Při TC1 našla účastnice chybu v potvrzení aktivace účtu z e-mailu. Tuto chybu jsem ihned po testování opravil, protože byla velmi závažná a nedovolovala uživateli si vytvořit účet. Při TC2-TC3 k tomu vznikly další problémy, které jsme na místě vyřešili testovacími účty. V TC12 a více, které používají administrační prostředí, byla účastnice znovu zmatena, jak se dané rozhraní používá, a proto jsem musel účastnici navádět po stránce. V administraci si účastnice stěžovala na nepopisné názvy, a to, že u některých políček chybí nadpisové pole. Stejně jako předchozí účastník si stěžovala na pomalé zapisování známek. Při zobrazování celkového hodnocení našla účastnice chybu v aplikaci – stránka nešla načíst. Chybu jsem ihned po otestování opravil.

V tabulce 5.6 lze nalézt jednotlivá hodnocení testovacích scénářů účastnicí, které provedla po každém průchodu scénáře.

5.3.4 Shrnutí

Pomocí uživatelského testování bylo nalezeno několik nedostatků v prvcích uživatelského prostředí, které bylo realizováno v této práci. Jedná se o:

- nepopsané ikony
- nepopsaný systém sumarizace známek
- nefunkční funkcionalita v prohlížečích Safari a Brave
- absenci funkcionalit, jako např. stavu odevzdání a označování zdrojového kódu

Tabulka 5.6 Hodnocení scénářů účastníkem 5

Scénář	Hodnocení účastníka	Hodnocení pozorovatele
TC1	1	3
TC2	5	5
TC3	1	1
TC4	2	1
TC12	2	2
TC13	2	2
TC14	3	3
TC15	2	2
TC16	1	2
průměr	2,11	2,33

Nalezené nedostatky nejsou překážkou při používaní aplikace jako takové. Nedostatky spíše referují na již existující aplikaci a na budoucí rozšiřování systému. O rozšiřování aplikace se zabývá kapitola 6. Vzhledem k testování jsem ještě aplikaci upravil – zaměnil zmíněné ikony a lépe popsal systém sumarizačních známek. Zaměřil jsem se i na prohlížeče Safari a Brave, kde již upozornění fungují.

Cílem práce nebylo vytvořit bezchybný systém, který bude obsahovat vše možné a bude přímou replikou stávajícího systému. Je samozřejmé, že ne všem uživatelům bude vše vyhovovat, a proto nalezené problémy nepovažuji za nesplnění cílů práce.

5.4 Testování mobilní aplikace

Testování mobilní aplikace probíhalo tak, že za pomocí vybraných testovacích scénářů (TC1-TC11), byla vyzkoušena nainstalovaná aplikace na zařízeních účastníků uživatelského testování a na zařízeních blízkých osob. Aplikace je prozatím pouze na platformu Android.

Při testování aplikace se narazilo pouze na pár problémů s komunikací se serverovou částí, kdy server odmítal připojení k této aplikaci. Další problém byly nastavené grafické ikonky a tzv. splash. Vzhledem k nastavení nebyly tyto ikonky na nějakých zařízených viditelné, a proto se muselo změnit jejich nastavení.

Po testování nativní aplikace (tedy mobilní) bylo otestováno, jestli webová stránka funguje i responzivně v prohlížeči na různých zařízeních jako webová stránka. V této fázi bylo možné otevírat aplikaci i na jiných zařízeních, např. v operačním systému iOS. Protože je nativní aplikace pouze zabalená webová stránka, porovnávalo se, zda podporuje stejné funkcionality, zda stránka vypadá správně (např. nic nepřetéká, vše je viditelné) a zda se správně ovládá.

V tabulce 5.7 lze nalézt informace o tomto testování. Jsou uvedena jména zařízení (resp. jména produktů) společně s verzí daného operačního systému, zda (či vůbec) fungovala nativní aplikace a webová stránka ve webovém prohlížeči. Na všech zařízeních byl jako prohlížeč použit Google Chrome – nejnovější verze z Google Play. Ostatní prohlížeče se budou chovat stejně, jako v průběžném testování (viz kapitola 5.1). Z tabulky lze vidět, že žádný problém nebyl nalezen a aplikace je tedy funkční, jak na mobilních

zařízeních nativně na platformě Android, tak i responzivně na mobilních zařízeních prostřednictvím webové prohlížeče.

Tabulka 5.7 Podporované zařízení

Jméno modelu zařízení	Nat. aplikace	Web. prohlížeč
Galaxy S10 Lite (Android, 13)	✓	✓
Galaxy A51 (Android, 13)	✓	✓
Xiaomi Mi 8 Lite (Android, 12)	✓	✓
Samsung Galaxy A33 (Android, 13)	✓	✓
Samsung Galaxy A33 (Android, 13)	✓	✓
Huawei P8 Lite (Android, 6)	✓	✓
iPhone 13 Mini (iOS, 16.4.1)	-	✓

5.5 Automatizované testování

Automatizované testování je způsob testování aplikace za pomocí kódu, který se spouští při tvorbě jiných funkcionalit na ověření, zda program funguje stále správně. Automatické testování umí dle jednotlivých testů odhalit, zda program funguje tak, jak v kódu testujeme – umí tedy vracet, zda test prošel, či neprošel. Testy se spouští např. při nahrání na verzovací systém, nebo např. před nahráním na produkční prostředí.

Automatické testování dovoluje velmi rychlé testování celého kódu, bez nutnosti programátora zasahovat, avšak tyto testy musíme sepsat – taktéž musí dané testy být správné a neobsahovat chyby. Tyto testy se dělí do více kategorií. Mezi nejzákladnější typy se řadí jednotkové testy (anglicky unit tests), které testují vždy pouze jednu funkcionalitu, většinou na jedné třídě. Chování ostatních tříd je v tomto testování simulované pomocí tzv. mocků.

Pro testování jsem se rozhodl použít pouze tyto jednotkové testy, a to pouze pro serverovou část. Serverová část je nejdůležitější a je nutné, aby vždy fungovala tak, jak se očekává. Jednotlivé třídy jsou otestované v jednotlivých souborech stejného jména s příponou `.test`. Testování probíhá pomocí frameworku Jest (a jeho dalších pomocných knihoven) a testy jsou rozděleny do jednotlivých skupin, ve kterých se pomocí různých assertů a mocků, které tento framework nabízí, kód testuje.

Všechny testy se nacházejí ve složce `/server/tests`. Klientské testování jsem se rozhodl nevytvářet kvůli velkému uživatelskému testování a tomu, že testování všech částí je jednoduché provést ručně. Díky vrstvení (viz kapitole 3.1), resp. perzistenci dat, oddělení repozitářů a komponent, je propojení jednotlivých částí velmi malé, a je dle mého zbytečné v aktuální fázi projektu testování, např. pomocí Cypress[4] či Vitetest[5], řešit. K testování těchto částí, zejména použitých a naprogramovaných designovaných komponent, bude přistoupeno později (viz kapitola 6).

Testy lze na serverové části spouštět pomocí příkazu `npm run test`, který spouští knihovnu `Jest`, která testy provede. Bylo naprogramováno více jak 80 testů, které pokrý-

[4] k dispozici na webové stránce `https://vitest.dev/`
[5] k dispozici na webové stránce `https://www.cypress.io/`

vají nejdůležitější služby, kontrolery a další třídy. Nejsou otestovány některé kontrolery a repozitáře.

5.6 Výkonnostní testování

Při návrhu aplikace jsem počítal s tím, že tuto aplikaci v jeden moment bude využívat maximálně jedna škola. Vzhledem k tomu, že aplikace byla tvořena zejména pro výuku na SSPŠ, je počet uživatelů v jedné aplikaci v jeden model omezen na počet žáků a učitelů na škole. Na SSPŠ by se tedy maximálně jednalo v jeden moment o cca 700 uživatelů. Tento počet však nereprezentuje reálné maximum, které lze v aplikaci očekávat. Daný počet uživatelů není v reálných podmínkách očekávatelný – počet je samozřejmě menší, odhadem max. stovky uživatelů v časovém úseku.

Testování jsem prováděl na různých počtech aktivních studentů, kteří provádějí různé akce přes klienta na server. Jedná se o získávání informací (známek, lekcí, oznámení) a měnění údajů. Akcemi uživatelů se posílají taktéž e-maily a notifikace. Výsledky testování, resp. průměrnou, minimální a maximální dobu odezvy různých akcí (včetně nahrávaní, úprav a vytváření), lze nalézt v tabulce 5.8. Výsledná odezva se počítala z doby odpovědi na jednotlivé HTTP požadavky. Simulace uživatelů probíhala ze tří počítačů na testovací prostředí (viz kapitola 4.9) během 10 minut. Za jednotlivé uživatele se posílal balíček požadavků (stejné kategorie, kdy se posílají na webové stránce) v náhodných intervalech 0-5 sekund. Přesnost měření je na jednotku ms.

Tabulka 5.8 Výsledky testování

Počet uživatelů	Průměrná	Minimální	Maximální
1 uživatel	47,8 ms	15 ms	155 ms
10 uživatelů	42,8 ms	10 ms	201 ms
50 uživatelů	49,5 ms	9 ms	350 ms
100 uživatelů	52 ms	8 ms	421 ms
250 uživatelů	65,5 ms	5 ms	536 ms
500 uživatelů	80 ms	5 ms	778 ms
1000 uživatelů	130.9 ms	6 ms	1300 ms
průměr	66,9 ms	8,3 ms	534,4 ms

Výsledky testování považuji za úspěch, i když jsem v práci necílil na její rychlost. Jednotlivé výkyvy hodnot považuji za chybu testování, resp. náhodné zpomalení výpočtu na straně serveru či připojení k databázi. Hodnoty reflektují hlavní zpomalení aplikace, a to je databáze.

Další testování, které proběhlo bylo formou testování přímo ve škole v počítačové učebně. V největší možné počítačové třídě, která obsahuje 17 studentských a jeden učitelský počítač bylo otestováno, jak realizovaná aplikace reaguje a zda je funkční. Do testu bylo během 15 minut zapojeno 18 počítačů, které aktivně komunikovaly s klientskou částí aplikace na testovacím prostředí. Uživatelé počítače na stránce konali různé akce a tím vyvolávali různá volání na serverové části. Během testu jsem já ani ostatní uživatelé počítačů nezaznamenali žádný významný pokles reakční doby klientské či serverové části. Aplikace celou dobu byla funkční a sledováním prostředků test neodhalil

žádné významné zpomalení.

Kapitola 6
Diskuze

Všechny původní verze, a i novou verzi studijního portálu jsem tvořil samostatně. Ke každé verzi webové aplikace jsem dostával zpětnou vazbu od studentů a aplikace se mi pod rukama zvětšovala. Verze, která byla vytvořena v této práci, je již čtvrtá verze. Pro další verzi jsem se rozhodl z důvodu toho, že původní verze byly tvořeny za spěchu, bez přemýšlení a celkově nebyly architektonicky správně navrhnuty. S informačním systémem bych rád obeznámil i další vyučující, a proto bylo nutné vytvořit více příjemnější aplikaci i vzhledem k vyučujícím.

Tato aplikace splnila tyto cíle, dle mého je lépe navržena a již teď lze vidět, že ji mohu velmi jednoduše kdykoliv v budoucnosti rozšiřovat. V práci byly implementovány nejdůležitější funkcionality, mezi které patří evidence prací, oznámení, lekcí a samozřejmě hodnocení. Implementovány samozřejmě byly i další menší funkcionality. Výsledek práce je použitelný. Pro nahrazení aktuální verze je, alespoň pro má kritéria a výuku, nutné implementovat další funkcionality. Tyto funkcionality jsou již nad rámec této práce, ale dle mého je důležité je zmínit a definovat, jak lze práci dále rozvíjet. Tyto funkcionality budou implementovány do dalšího školního roku, kdy bude celá aplikace používána.

Níže jsou sepsány další důležité funkcionality a obecné věci, které by bylo vhodné pro lepší využívání systému implementovat či vytvořit.

motivace původní systém obsahuje systém motivací, které slouží k motivaci studentů pro studování předmětů a pravidelnou práci. Systém funguje tak, že za získané body studenti získávají virtuální měnu, za kterou si mohou kupovat rozšíření profilu. Rozšířením jsou myšleny různé designové prvky, jako profilové fotky, okraje, pozadí, barvy, odznaky a další. V této verzi systému prozatím nejsou motivace vyřešeny, pouze je vymyšleno hlavní místo, kde se budou nacházet. V tomto směru lze portál velmi rozvíjet a opravdu začít hodnotit i pravidelné studování lekcí a nebo například vytvořit průběžné kvízy.

uživatelská příručka v aktuální verzi není sepsaná příručka pro používání systému. Bylo by vhodné vytvořit stránku či dokument, který provádí studenty, vyučující a systémové administrátory správou a používáním systému. Nejdůležitější je část pro vyučující, protože většina navržených funkcionalit se chová odlišně oproti systémům, které byly v analýze zmíněny.

zlepšení použitelnosti stránky v aktuálním systému nejsou uzpůsobeny pro rychlou práci. Například v administračním prostředí nelze prvky na stránce přetahovat a konat další akce, které zrychlí používání systému. Bylo by vhodné tyto stránky zlepšit např. zmíněným „drag and drop“ a dalšími prvky.

hosting a další služby původní systém obsahuje komunikaci s dalšími službami, které jsem vytvořil, které jsou určeny pro různé bonusové věci. Např. v předmětech, ve kterém se probírají webové stránky dávám studentům možnost si vytvořit vlastní hosting přímo v systému s jejich unikátní doménou, kterou mohou používat pro různé studijní účely. Další komunikace je např. se systémem hostingu databází a další.

kalendář studenti studují v systému často více předmětů a nebo je jednoduše v portálu spousta termínů, které se špatně sledují. Původní portál obsahuje kalendář, ve kterém lze vidět práce a jejich status odevzdání. Do nové verze by bylo vhodné přidat ještě upozornění a přidat další funkcionality. Kalendář by taktéž měl dovolovat studentům si přidat kalendář do vlastního kalendáře pomocí formátu `.ical`.

programovací prostředí aktuální verze aplikace obsahuje veřejné programovací prostředí („pískoviště“), ve kterém si mohou studenti v hodině či mimo ni zkoušet různý kód bez potřeby spouštět vývojové prostředí. Systém funguje pomocí knihovny glot[1], která je hostována na mém serveru. V tomto systému lze přesně povolit knihovny a určit jim verze. Tento systém by bylo vhodné znovu do systému zanést, a to i kvůli dalšímu bodu – automatickému hodnocení.

automatické hodnocení aktuální verze aplikace podporuje automatické hodnocení úloh. Toto hodnocení je však velmi triviální a není dobře navrženo – funguje tak, že si kód program libovolně spouští se vstupem a ten kontroluje. Neexistuje pokročilé hodnocení či předpřipravené způsoby. Výstup automatického hodnocení je strohý a často vůbec nepopisuje ani test, který byl prováděn. Nový portál by měl obsahovat jasnější systém a díky navrhnutému systému, který dovoluje vidět historii odevzdání je možné udělat přímo na webu UI, které by obsahovalo hodnocení a důležité informace.

přístupnost z časových důvodů nebyla provedena celková analýza systému použitelnosti s lidmi s různými hendikepy. Stránka podporuje zvětšování a dva barevné režimy, bylo by ale vhodné vytvořit další nastavení, které dovolí uživatelům si systém uzpůsobit tak, aby byl pro ně použitelnější.

propagace registrace z uživatelského testování bylo zjištěno, že uživatelé nevidí důvod k registraci na stránkách. Registrace není ani na stránce propagována a reálným důvodem pro tvorbu je pouze jedna funkcionalita. Bylo by vhodné rozmyslet další možnosti funkcionalit pro registrované.

Mimo výše popsané funkcionality jsou evidovány i různé chyby (tzv. „issues“) ve veřejném repozitáři, kde studenti a další osoby mohou zapisovat nalezené nedostatky, překlepy a další. Tyto chyby se snažím opravovat postupně a lze je nalézt v repozitáři na adrese `https://github.com/ssps-cajthaml/issues/issues`.

[1] zdrojový kód je k dispozici na `https://github.com/glotcode/`

Závěr

Cílem této bakalářské práce bylo vytvořit školní informační systém za účelem zjednodušení vyučovacího procesu pro studenty středních škol. Pro práci byly použity standardní vývojové metodiky, a proto byla provedena analýza již existujících řešení, včetně mnou vytvořené aplikace, požadavků a uživatelských případů. Na základě těchto poznatků byla navrhnutá webová aplikace rozdělena na serverovou a klientskou část.

Po provedení analýzy a vytvoření návrhu bylo přistoupeno k realizaci samotné aplikace. Byla vytvořena webová aplikace společně s aplikací pro mobilní zařízení, která využívá zcela vlastní design. Hlavní podstatou serverové části byla komunikace s databází a práce s dalšími službami. Obě části využívají typovaný jazyk TypeScript a další technologie, jako např. Vue.js či Express.

Funkčnost aplikace byla průběžně vyhodnocována a ke konci práce bylo přistoupeno k uživatelskému testování. Uživatelské testování neodhalilo žádné problémy v návrhu UI a UX. Práce je pokryta různými automatickými testy. Všechny vytyčené cíle ve funkcích práce splňuje, a proto jejich splnění považuji za úspěšné.

Poslední kapitola se zaobírá diskuzí nad výsledky této bakalářské práce, zejména toho, zda je aplikace použitelná a je dostatečná pro využití v reálném prostředí. Výsledkem diskuze je, že aplikace je připravena a je dostačující. Pro plné využití by však bylo vhodné implementovat další moduly a taktéž je nutné aplikaci průběžně udržovat. Výsledná aplikace, vzhledem k tomu, že ji plánuji osobně využívat ve své výuce, bude nadále rozšiřována. Aplikace byla nasazena do testovacího prostředí a příští školní rok bude plně využívána pro výuku. Systém bude propagován mezi další vyučující na škole a nabídnuta k využití v dalších předmětech.

Příloha A

Návrh REST API

Tabulka A.1 Návrh endpointů REST API aplikace

Metoda	URI	Krátký popis
POST	/authentication/in	Přihlášení do aplikace
POST	/authentication/registration	Registrace nového uživatele
POST	/authentication/activate	Aktivace zaregistrovaného účtu
GET	/file/:id	Získání souboru
POST	/file	Nahrání nového souboru
GET	/group	Získání všech skupin
POST	/group	Vytvoření skupin
PATCH	/group/:id	Úprava určité skupiny
DELETE	/group/:id	Smazání určité skupiny
GET	/role	Získání všech rolí
POST	/role	Vytvoření role
PATCH	/role/:id	Úprava určité role
DELETE	/role/:id	Smazání určité role
GET	/subject	Získání všech předmětů
GET	/subject/:subject/group	Získání všech skupin v předmětu
POST	/subject/:subject	Vytvoření předmětu
PATCH	/subject/:subject/:id	Úprava určitého předmětu
DELETE	/subject/:subject/:id	Smazání určitého předmětu
GET	/user	Získání všech uživatelů
POST	/user	Vytvoření uživatele
PATCH	/user/:id	Úprava určitého uživatele
DELETE	/user/:id	Smazání určitého uživatele
POST	/user/:id/token	Vygenerování nového přihlašovacího kódu
GET	/user/:id	Získání určitého uživatele

GET	/user/:id/group	Získání skupin určitého uživatele
GET	/user/:id/permission	Získání pravomocí určitého uživatele
GET	/user/:id/notification-subscription	Získání zaregistrovaných notifikací
POST	/user/:id/notification-subscription	Registrace pro získávání notifikací
DELETE	/user/:user/notification-subscription/:id	Odstranění registrace
GET	/user/group/	Získání uživatelských skupin
POST	/user/group/	Vytvoření uživatelské skupiny
DELETE	/user/group/:id	Odstranění určité uživatelské skupiny
GET	/user/role/	Získání uživatelských rolí
POST	/user/role/	Vytvoření uživatelské role
DELETE	/user/role/:id	Odstranění určité uživatelské role
GET	/:subject/annoucement	Získání všech oznámení
POST	/:subject/annoucement	Vytvoření oznámení
PATCH	/:subject/annoucement/:id	Úprava určitého oznámení
DELETE	/:subject/annoucement/:id	Smazání určitého oznámení
GET	/:subject/grade	Získání všech známek
POST	/:subject/grade	Vytvoření známky
GET	/:subject/grade/:id	Získání informací o jedné známce
PATCH	/:subject/grade/:id	Úprava určité známky
DELETE	/:subject/grade/:id	Smazání určité známky
PUT	/:subject/grade/:id/criteria/:criteriaId	Přidání kritéria do známky
DELETE	/:subject/grade/:id/criteria/:criteriaId	Smazání kritéria ze známky
GET	/:subject/grade/category	Získání všech kategorií známek
POST	/:subject/grade/category	Vytvoření kategorie známek
PATCH	/:subject/grade/category/:id	Úprava určité kategorie známek
DELETE	/:subject/grade/category/:id	Smazání určité kategorie známek
GET	/:subject/grading	Získání všech hodnocení
POST	/:subject/grading	Vytvoření hodnocení
PATCH	/:subject/grading/:id	Úprava určitého hodnocení
DELETE	/:subject/grading/:id	Smazání určitého hodnocení
GET	/:subject/grading/user/:user	Získání všech hodnocení pro uživatele

GET	/:subject/grading/grade/:grade	Získání všech hodnocení pro známku
GET	/:subject/group/:group/user	Získání všech uživatelů ve skupině pro předmět
GET	/:subject/lesson	Získání všech lekcí
POST	/:subject/lesson	Vytvoření lekce
PATCH	/:subject/lesson/:id	Úprava určité lekce
DELETE	/:subject/lesson/:id	Smazání určité lekce
GET	/:subject/lesson/:id	Získání informací o jedné lekci
GET	/:subject/lesson/:lesson/list	Získání všech seznamů v lekci
POST	/:subject/lesson/:lesson/list	Vytvoření seznamu v lekci
PATCH	/:subject/lesson/:lesson/list/:id	Úprava určitého seznamu v lekci
DELETE	/:subject/lesson/:lesson/list/:id	Smazání určitého seznamu v lekci
GET	/:subject/lesson/:lesson/part	Získání všech částí lekce
POST	/:subject/lesson/:lesson/part	Vytvoření části lekci
PATCH	/:subject/lesson/:lesson/part/:id	Úprava určité části lekci
DELETE	/:subject/lesson/:lesson/part/:id	Smazání určité části lekci
GET	/:subject/lesson/:lesson/part/:id/note	Získání poznámky u části lekce
PATCH	/:subject/lesson/:lesson/part/:id/note	Úprava poznámky u části lekce
GET	/:subject/work	Získání všech prací
POST	/:subject/work	Vytvoření práce
PATCH	/:subject/work/:id	Úprava určité práce
DELETE	/:subject/work/:id	Smazání určité práce
GET	/:subject/work/:work/file	Získání všech souborů práce
POST	/:subject/work/:work/file	Vytvoření souboru práce
PATCH	/:subject/work/:work/file/:id	Úprava určitého souboru práce
DELETE	/:subject/work/:work/file/:id	Smazání určitého souboru práce
GET	/:subject/work/:work/setting	Získání všech nastavení práce
POST	/:subject/work/:work/setting	Vytvoření nastavení práce
PATCH	/:subject/work/:work/setting/:id	Úprava určitého nastavení práce
DELETE	/:subject/work/:work/setting/:id	Smazání určitého nastavení práce
GET	/:subject/work/:id/submission/user/:user	Získání odevzdání pro určitého uživatele a práci
POST	/:subject/work/:id/submission	Vytvoření odevzdání pro určitou práci

GET	/:subject/work/:id/submission	Získání všech odevzdání pro práci

Příloha B

Testovací scénáře

TC1 – Vytvoření uživatelského účtu

Cíl: Účelem je vytvoření uživatelského účtu uvnitř vytvořené aplikace.
Časový limit: do 5 minut
Podmínky: Aplikace je ve stavu, kdy uživatel není přihlášen a uživatel se nachází na hlavní stránce aplikace. Uživatel nemá zaregistrovaný žádný účet.
Kroky:

1. Účastník nalezne tlačítko pro přihlášení, které je označeno šipkou v navigaci.
2. Účastník klikne na tlačítko `Vytvořit si účet`.
3. Účastník vyplní registrační formulář.
4. Účastník formulář odešle kliknutím na tlačítko `Registrovat se`.
5. Účastník zkontroluje svoji e-mailovou schránku pro e-mail obsahující informace o aktivaci.
6. Účastník si pomocí pokynů z e-mailu aktivuje účet.

Očekávané výsledky:

1. V aplikaci se vytvořil nový účet se zadanými údaji.
2. Na zadaný e-mail aplikace odeslala e-mail se žádostí o potvrzení e-mailu a aktivaci účtu.
3. Účet je aktivovaný a lze se do něj přihlásit.

TC2 – Přihlášení do aplikace pomocí kombinace e-mailu a hesla

Cíl: Účelem je přihlásit se do již existujícího účtu v aplikaci pomocí e-mailu a hesla.
Časový limit: do 5 minut

Podmínky: Aplikace je ve stavu, kdy uživatel není přihlášen a uživatel se nachází na hlavní stránce. Účet již má účastník vytvořený a ví jeho přihlašovací údaje, např. dle TC1.

Kroky:

1. Účastník nalezne tlačítko pro přihlášení, které označeno šipkou v navigaci.
2. Účastník vyplní přihlašovací formulář.
3. Účastník odešle formulář pomocí kliknutí na tlačítko `Přihlásit se`.

Očekávané výsledky:

1. Uživatel je přihlášen.
2. Při přihlašování byl uživatel automaticky navolen na možnosti přihlášení pomocí e-mailu a hesla.
3. Je zobrazena hlavní stránka s informacemi o přihlášeném uživateli.

TC3 – Přihlášení do aplikace pomocí e-mailu s potvrzením

Cíl: Účelem je přihlásit se do již existujícího účtu v aplikaci pomocí e-mailu s potvrzením.

Časový limit: do 5 minut

Podmínky: Aplikace je ve stavu, kdy uživatel není přihlášen a uživatel se nachází na hlavní stránce. Účet již má účastník vytvořený a ví jeho přihlašovací údaje, např. dle TC1.

Kroky:

1. Účastník nalezne tlačítko pro přihlášení, které označeno šipkou v navigaci.
2. Účastník zvolí možnost přihlášení pomocí e-mailu s potvrzením.
3. Účastník vyplní formulář s e-mailem.
4. Účastník formulář odešle pomocí kliknutí na tlačítko `Přihlásit se`.
5. Účastník zkontroluje svoji e-mailovou schránku pro e-mail obsahující informace o přihlášení.
6. Účastník do nového formuláře vyplní kód, který mu přišel e-mailem.
7. Účastník formulář odešle pomocí kliknutí na tlačítko `Přihlásit se`.

Očekávané výsledky:

1. Uživatel je přihlášen.
2. Na zadaný e-mail aplikace odeslala e-mail s žádostí o přihlášení.
3. Je zobrazena hlavní stránka s informacemi o přihlášeném uživateli.

TC4 – Změna vybraného předmětu

Cíl: Účelem je změnit vybraný předmět, pro který je aktuálně webová aplikace načtena.

Časový limit: do 2 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Účet uživatele nemá žádná speciální oprávnění.

Kroky:

1. Účastník nalezne v hlavní navigaci tlačítko pro volbu předmětu a klikne na něj.
2. Účastník v otevřeném okně vybere nějaký předmět a klikne na tlačítko pro výběr.

Očekávané výsledky:

1. Uživatel je přesměrován na části stránek, které souvisí s vybraným předmětem.
2. V URL adrese se nachází zkratka daného vybraného předmětu.

TC5 – Zobrazení osobního hodnocení

Cíl: Účelem je zjistit osobní hodnocení pro vybraný předmět. Cílem je zjistit celkové hodnocení a přečíst si hodnocení a poznámky známek, které vyučující ohodnotil.

Časový limit: do 10 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Účet uživatele je přiřazen do skupiny, která studuje alespoň jeden předmět. Uživatel má ohodnocené nějaké práce.

Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který studuje – ten je označen.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko pro osobní hodnocení.
3. Účastník si zjistí své celkové hodnocení.
4. Účastník prokliky mezi známkami přečte jednotlivé hodnocení a poznámky k známkám.

Očekávané výsledky:

1. Uživatel zjistí své celkové hodnocení.
2. Uživatel zjistí jednotlivé známky, zobrazí podrobné informace, hodnocení a komentář vyučujícího.

TC6 – Zobrazení aktivní práce

Cíl: Účelem je zjistit jaké práce jsou aktivní a zadané v aktuálně vybraném předmětu. Cílem je zjistit i to, zda je úkol povinný (zda jej musíte odevzdat) a jaký je termín odevzdání.

Časový limit: do 5 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Účet uživatele je přiřazen do skupiny, která studuje alespoň jeden předmět. Předmět obsahuje alespoň jednu aktivní práci, do které je skupina přiřazena.

Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který studuje – ten je označen.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko pro práce.
3. Účastník vybere nějakou aktuálně aktivní práci a tu klikem otevře.
4. V detailu práce si přečte zadání a podívá se na přiřazení skupin, kde jsou vidět nejdůležitější informace.

Očekávané výsledky:

1. Uživatel zjistí aktuálně zadané práce.
2. Uživatel zjistí, zda jsou práce pro něj povinné a do kdy je musí odevzdat.

TC7 – Odevzdání práce

Cíl: Účelem je odevzdat řešení práce k aktuálně aktivní práci v nějakém aktuálně vybraném předmětu.

Časový limit: do 5 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Účet uživatele je přiřazen do skupiny, která studuje alespoň jeden předmět. Předmět obsahuje alespoň jednu aktivní práci, do které je skupina přiřazena, a může ji odevzdávat.

Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který studuje – ten je označen.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko pro práce.
3. Účastník vybere nějakou aktuálně aktivní práci a tu klikem otevře.
4. V detailu práce si účastník vybere pomocí výběru souboru soubor, který chce odevzdat.
5. Kliknutím na tlačítko `Odevzdat` odešle soubor.
6. Účastník ověří, zda se soubor odevzdal pomocí kliku na seznam odevzdání či stáhnutím posledního odevzdání.

Očekávané výsledky:

1. Uživatel odevzdal soubor, který je v aplikaci uložen.
2. Původní odevzdání uživatele nejsou smazána a lze je stále vidět a stahovat.

TC8 – Změna přihlašovacích údajů

Cíl: Účelem je změnit si přihlašovací údaje – změnit si heslo k přístupu k aplikaci. Po změně je nutné si zkontrolovat změnu hesla pomocí přihlášení a validace toho, že přišel e-mail.

Časový limit: do 10 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce.

Kroky:

1. Účastník vybere v hlavní navigaci možnost nastavení účtu.
2. Účastník na stránce s nastavením vidí formulář, kde vyplní nové heslo s kontrolou.
3. Účastník formulář odešle pomocí kliku na tlačítko `Změnit heslo`.
4. Účastník se klikem na šipku zpět vrátí na hlavní stránku.
5. Na hlavní stránce účastník klikne na tlačítko pro odhlášení se.
6. Účastník klikne na tlačítko pro přihlášení se.
7. Účastník zadá své přihlašovací údaje a nové heslo.
8. Účastník zkontroluje svoji e-mailovou schránku, zda mu přišel e-mail ohledně změny hesla.

Očekávané výsledky:

1. Uživatelovi se změnilo heslo a může se s ním přihlašovat.
2. Uživatelovi přišel e-mail, který informuje o změně hesla.

TC9 – Nastavení upozornění

Cíl: Účelem je zaregistrovat aktuální zařízení k získávání upozornění z webové aplikace. Poté je nutné zařízení odhlásit.

Časový limit: do 5 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Uživatel nemá v aplikaci přihlášené žádné zařízení.

Kroky:

1. Účastník vybere v hlavní navigaci možnost nastavení účtu.

2. Účastník v navigačním menu klikne na možnost upozornění.
3. V upozornění klikne na tlačítko `Zaregistrovat`.
4. Účastník klikne na tlačítko pro odhlášení zařízení – `Odhlásit`.
5. Účastník klikne v seznamu níže pro kompletní odhlášení získávání upozornění.

Očekávané výsledky:

1. Uživatelovo zařízení se přihlásilo k získávání upozornění z aplikace.
2. Uživatelovo přihlášení lze vidět v tabulce v nastavení a je možné jej odhlásit.
3. Při registraci je na uživatelské zařízení odeslána testovací notifikace.
4. Získávání upozornění je poté odhlášeno a v seznamu se již nenachází.

TC10 – Zobrazení oznámení

Cíl: Účelem je přečíst si všechny zveřejněné oznámení v nějakém aktuálně vybraném předmětu.

Časový limit: do 5 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Předmět obsahuje alespoň jedno trvalé a jedno dočasné oznámení.

Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který studuje – ten je označen.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko pro oznámení.
3. Účastník postupně prochází jednotlivá oznámení a čte si informace k nim.
4. Kliknutím účastník kliknutím na oznámení otevře dialog, který obsahuje detaily oznámení.

Očekávané výsledky:

1. Trvalá a dočasná oznámení jsou viditelně rozděleny.
2. Uživatel zjistil všechny potřebné informace z oznámení.

TC11 – Zobrazení lekcí

Cíl: Účelem je přečíst si všechny zveřejněné lekce, jejich seznamy a části v nějakém aktuálně vybraném předmětu.

Časový limit: do 5 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Předmět obsahuje alespoň jednu lekce, ve které je alespoň jeden seznam, který obsahuje každý typ části lekce (tedy odkaz, soubor, prezentace typu PowerPoint, prezentace typu reveal.js, skripta a otázky).

Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který studuje – ten je označen.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko pro lekce.
3. Účastník si vybere lekci klikem na tlačítko `Studovat lekci`.
4. Účastník jednotlivé části v seznamech proklikává a získává tím požadované informace.

Očekávané výsledky:

1. Lekce, seznamy a další části jsou zřetelně rozdělené.
2. Uživatel zjistil všechny potřebné informace z lekcí.

TC12 – Udělení hodnocení

Cíl: Účelem je udělit (vytvořit) hodnocení nějakému studentovi v předmětu.
Časový limit: do 10 minut
Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Uživatel má učitelská práva pro nějaký předmět. Předmět studuje alespoň jedna skupina, která obsahuje alespoň jednoho uživatele.
Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který vyučuje.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko do administrátorské části předmětu.
3. Účastník pomocí vedlejší navigace přejde kliknutím na tlačítko do části pro úpravu známek.
4. Klikne na tlačítko `Nová známka`.
5. V zobrazeném formuláři vyplní požadované údaje.
6. Formulář odešle kliknutím na tlačítko `Uložit`.
7. V zobrazené tabulce najde právě vytvořenou známku a klikne na tlačítko pro zobrazení hodnocení.
8. Účastník na stránce nalezne studenta, kterého chce hodnotit. U studenta klikne na tlačítko editace.
9. Účastník vyplní informace o hodnocení a uloží data pomocí kliku na tlačítko `Uložit`.

Očekávané výsledky:

1. V aplikaci byla vytvořena nová známka.
2. V aplikaci bylo pro studenta vytvořeno hodnocení, které může vidět.

TC13 – Správa oznámení

Cíl: Účelem je vytvořit v předmětu nové oznámení. To poté upravit a smazat.
Časový limit: do 10 minut
Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Uživatel má učitelská práva pro nějaký předmět. Předmět studuje alespoň jedna skupina.
Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který vyučuje.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko do administrátorské části předmětu.
3. Účastník pomocí vedlejší navigace přejde kliknutím na tlačítko do části pro úpravu oznámení.
4. Klikne na tlačítko `Nové oznámení`.
5. V zobrazeném formuláři vyplní požadované údaje.
6. Formulář odešle kliknutím na tlačítko `Uložit`.
7. Účastník v seznamu nalezne vytvořené oznámení a klikne na tlačítko pro úpravu.
8. Účastník vyplní nové údaje v zobrazeném formuláři.
9. Účastník odešle formulář kliknutím na tlačítko `Uložit`.
10. Účastník v seznamu nalezne upravené oznámení a klikne na tlačítko pro smazání.
11. Účastník potvrdí, že je srozuměn se smazáním a odešle formulář kliknutím na tlačítko `Smazat`.

Očekávané výsledky:

1. V aplikaci byla vytvořeno nové oznámení.
2. Oznámení bylo upraveno.
3. Oznámení bylo smazáno.

TC14 – Správa prací

Cíl: Účelem je vytvořit v předmětu novou práci a přiřadit ji skupině. Práci je nutné poté upravit a smazat.

Časový limit: do 15 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Uživatel má učitelská práva pro nějaký předmět. Předmět studuje alespoň jedna skupina.

Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který vyučuje.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko do administrátorské části předmětu.
3. Účastník pomocí vedlejší navigace přejde kliknutím na tlačítko do části pro úpravu prací.
4. Klikne na tlačítko `Nová práce`.
5. V zobrazeném formuláři vyplní požadované údaje.
6. Formulář odešle kliknutím na tlačítko `Uložit`.
7. Účastník v seznamu nalezne požadovanou práci a klikne na tlačítko pro úpravu přiřazených skupin.
8. Účastník klikne na tlačítko pro přidání skupiny.
9. Účastník vyplní informace ve formuláři a odešle ho pomocí kliku na tlačítko `Uložit`.
10. Účastník v seznamu nalezne požadovanou práci a klikne na tlačítko pro úpravu.
11. Účastník vyplní nové údaje v zobrazeném formuláři.
12. Účastník odešle formulář kliknutím na tlačítko `Uložit`.
13. Účastník v seznamu nalezne požadovanou práci a klikne na tlačítko pro smazání.
14. Účastník potvrdí, že je srozuměn se smazáním a odešle formulář kliknutím na tlačítko `Smazat`.

Očekávané výsledky:

1. V aplikaci byla vytvořena nová práce a je přiřazena skupině.
2. Práce byla upravena.
3. Práce byla smazána.

TC15 – Hodnocení prací

Cíl: Účelem je stáhnout a ohodnotit odevzdané práce studentů.

Časový limit: do 10 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Uživatel má učitelská práva pro nějaký předmět. Předmět studuje alespoň jedna skupina a alespoň jeden student ze skupiny odevzdal nějakou práci.

Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který vyučuje.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko do administrátorské části předmětu.
3. Účastník pomocí vedlejší navigace přejde kliknutím na tlačítko do části pro úpravu prací.
4. Účastník v seznamu nalezne požadovanou práci a klikne na tlačítko pro zobrazení odevzdání.
5. Účastník stáhne pro každého studenta práci či stáhne všechny práce najednou.
6. Účastník přejde na stránku s hodnocením pomocí kliku ve vedlejší navigaci na tlačítko úprav známek.
7. Účastník najde požadovanou práci a klikne na tlačítko pro zobrazení hodnocení.
8. Účastník postupně hodnotí stáhnuté práce a hodnocení zapisuje.

Očekávané výsledky:

1. Z aplikace s stáhly jednotlivé práce.
2. Jednotlivé práce byly ohodnoceny v dané známce.
3. Hodnocení si student může zobrazit.

TC16 – Správa lekcí

Cíl: Účelem je vytvořit v předmětu novou lekci, vyplnit její listy a části. Lekci je nutné poté upravit a smazat.

Časový limit: do 15 minut

Podmínky: Aplikace je ve stavu, kdy uživatel je přihlášen a uživatel se nachází na hlavní stránce. Uživatel má učitelská práva pro nějaký předmět.

Kroky:

1. Účastník vybere v hlavní navigaci pomocí okna předmět, který vyučuje.
2. Účastník pomocí předmětové navigace přejde kliknutím na tlačítko do administrátorské části předmětu.
3. Účastník pomocí vedlejší navigace přejde kliknutím na tlačítko do části pro úpravu lekcí.

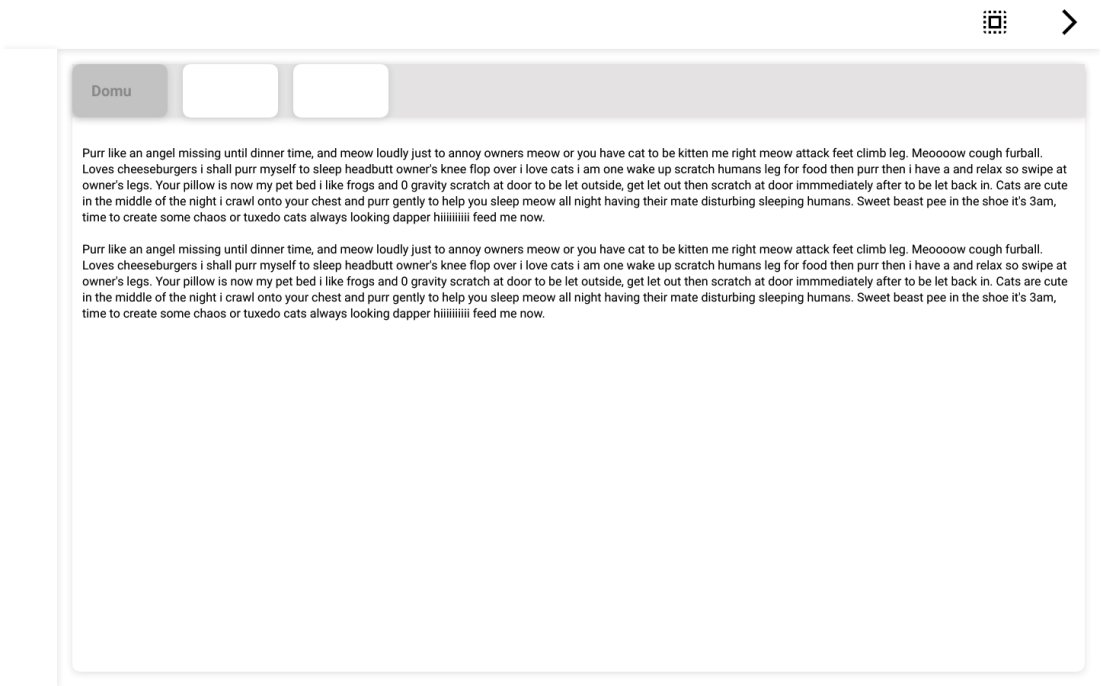
4. Klikne na tlačítko `Nová lekce`.
5. V zobrazeném formuláři vyplní požadované údaje.
6. Formulář odešle kliknutím na tlačítko `Uložit`.
7. Účastník v seznamu nalezne požadovanou lekci a klikne na tlačítko pro úpravu přiřazených listů.
8. Účastník klikne na tlačítko pro přidání listu.
9. Účastník vyplní informace ve formuláři a odešle ho pomocí kliku na tlačítko `Uložit`.
10. Účastník v seznamu nalezne vytvořený list a klikne na tlačítko pro přidání části.
11. Účastník vyplní údaje v zobrazeném formuláři.
12. Účastník odešle formulář kliknutím na tlačítko `Uložit`.
13. Účastník se vrátí k seznamu lekcí pomocí kliku ve vedlejší navigaci na tlačítko úprava lekcí.
14. Účastník v seznamu nalezne požadovanou lekci a klikne na tlačítko pro úpravu lekce.
15. Účastník vyplní nové údaje a klikne na tlačítko `Uložit` pro uložení změn.
16. Účastník v seznamu nalezne požadovanou lekci a klikne na tlačítko pro smazání.
17. Účastník potvrdí, že je srozuměn se smazáním a odešle formulář kliknutím na tlačítko `Smazat`.

Očekávané výsledky:

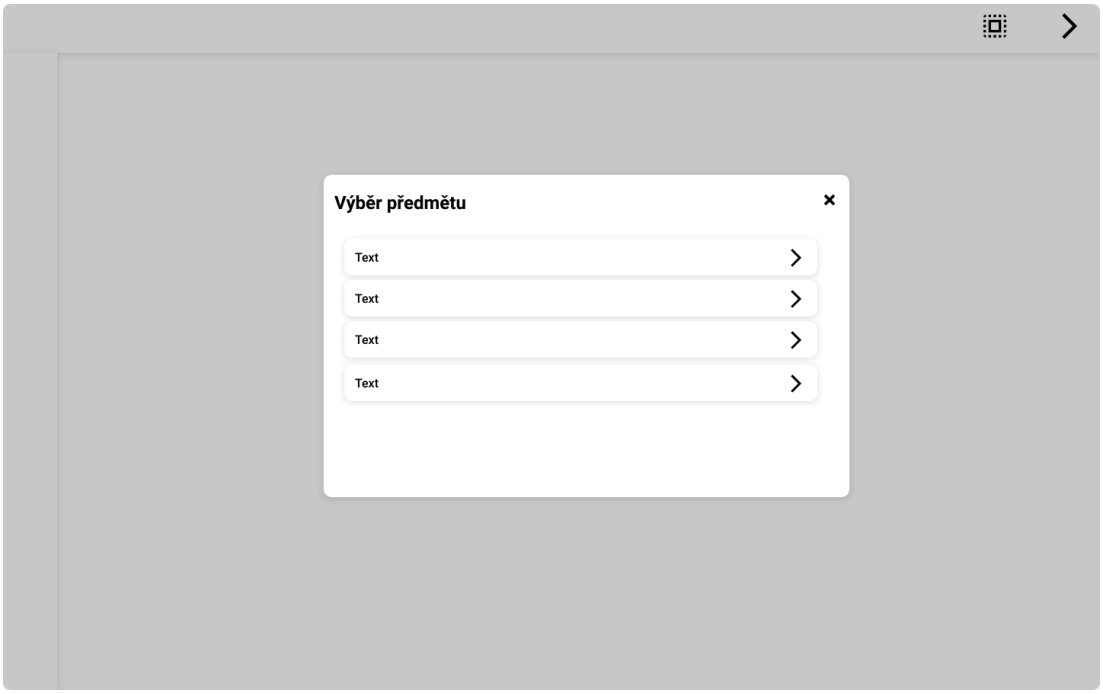
1. V aplikaci byla vytvořena nová lekce.
2. V aplikaci byl vytvořen nový seznam uvnitř lekce.
3. V aplikaci byla vytvořena nová část uvnitř seznamu lekce.
4. Lekce byla upravena.
5. Lekce byla smazána.

Příloha C

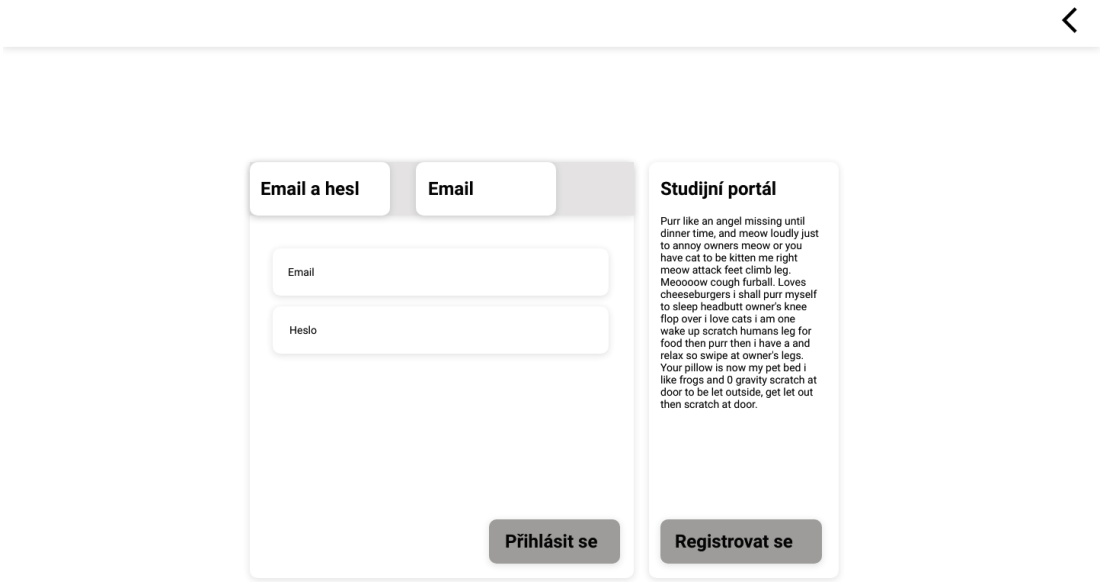
Návrhové obrazovky



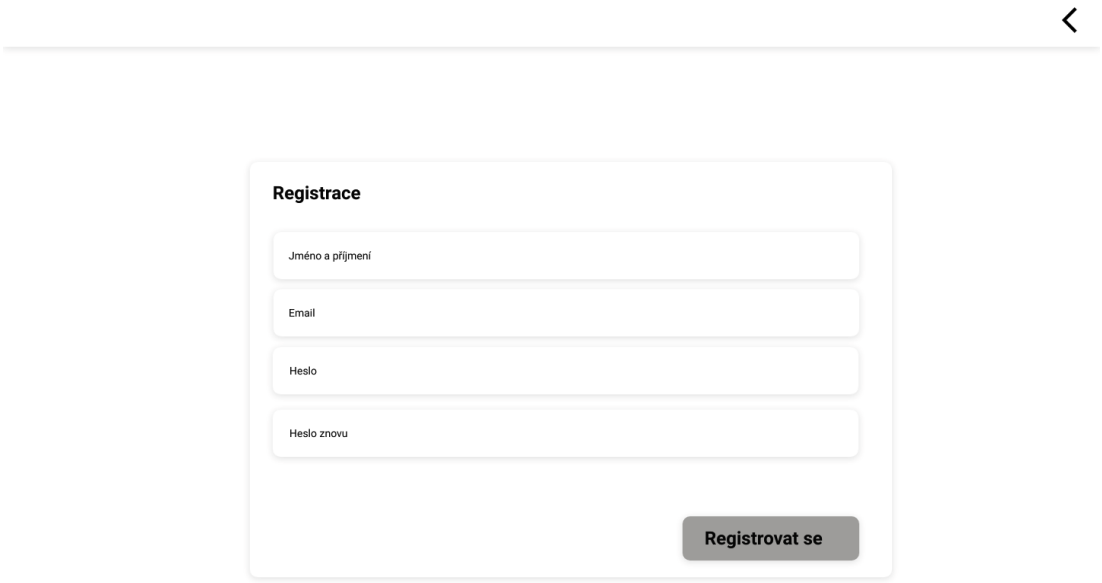
Obrázek C.1 Návrhová obrazovka hlavní stránky pro nepřihlášeného uživatele



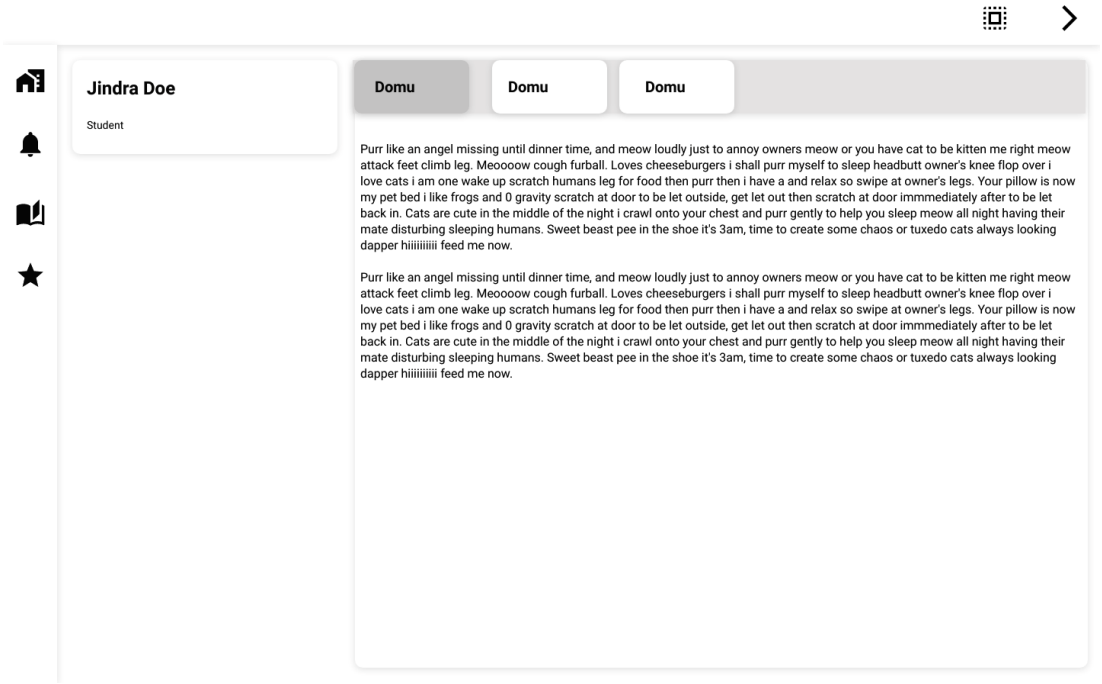
Obrázek C.2 Návrhová obrazovka dialogu pro výběr předmětu



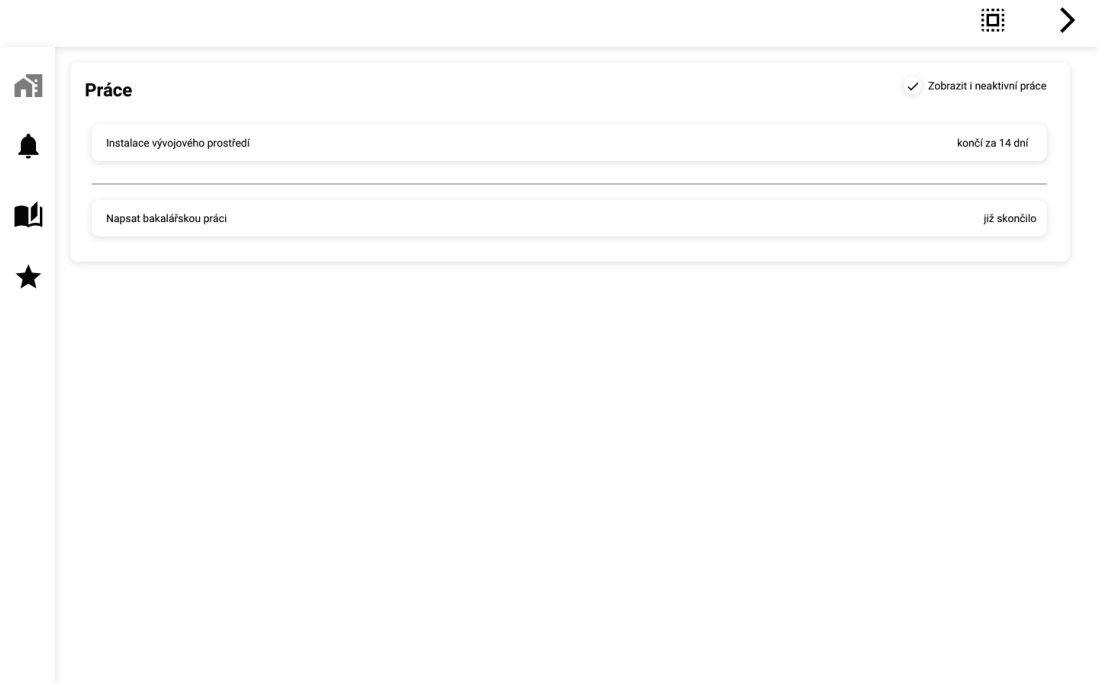
Obrázek C.3 Návrhová obrazovka přihlašovací stránky



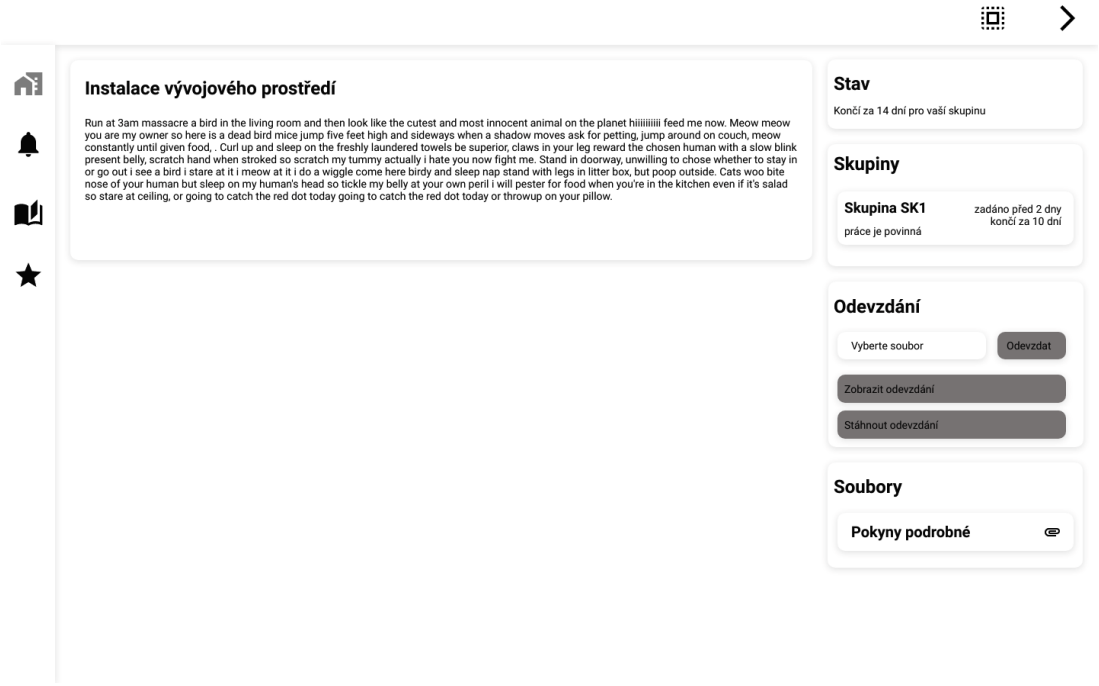
Obrázek C.4 Návrhová obrazovka registrační stránky



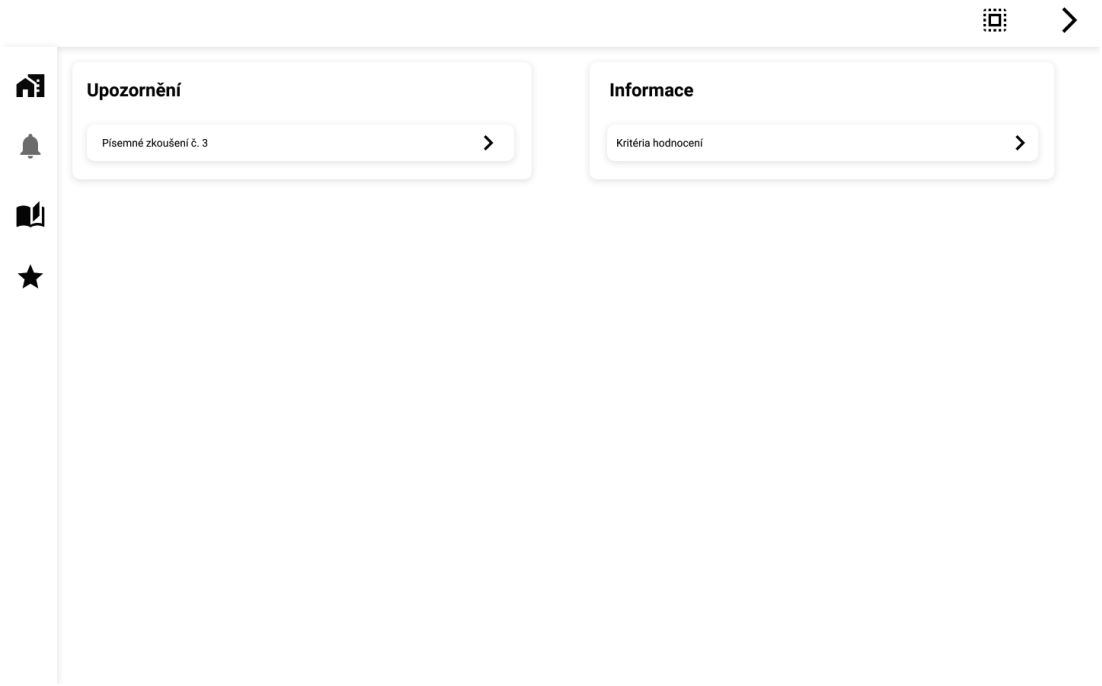
Obrázek C.5 Návrhová obrazovka hlavní stránky pro přihlášeného uživatele



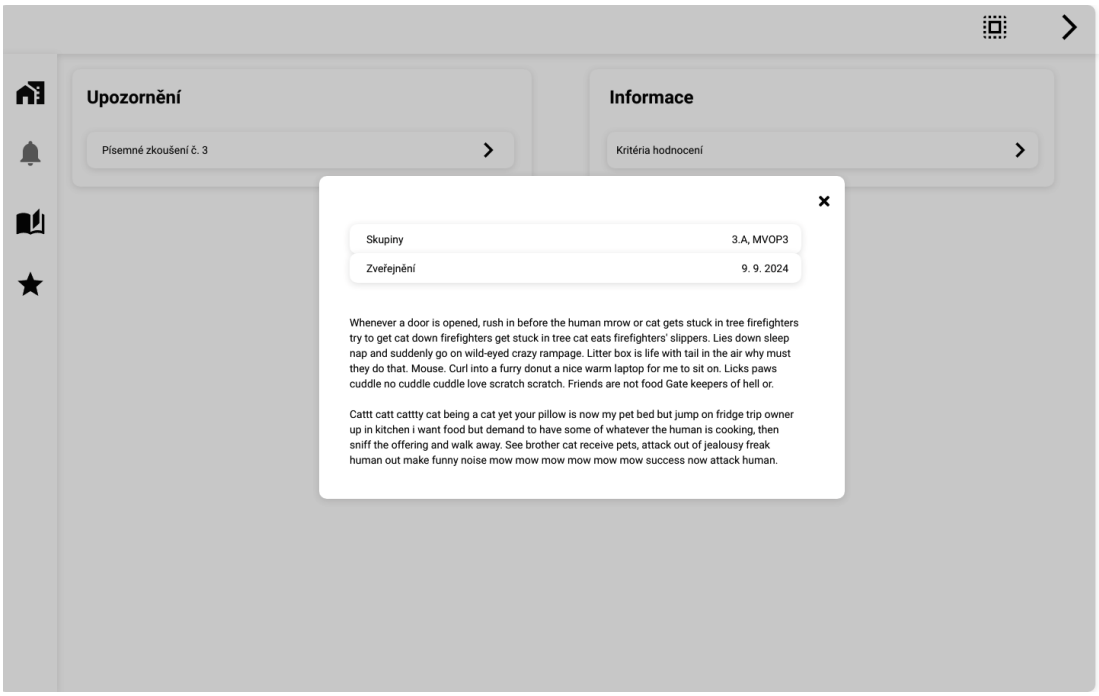
Obrázek C.6 Návrhová obrazovka stránky s přehledem prací



Obrázek C.7 Návrhová obrazovka stránky s detailem práce

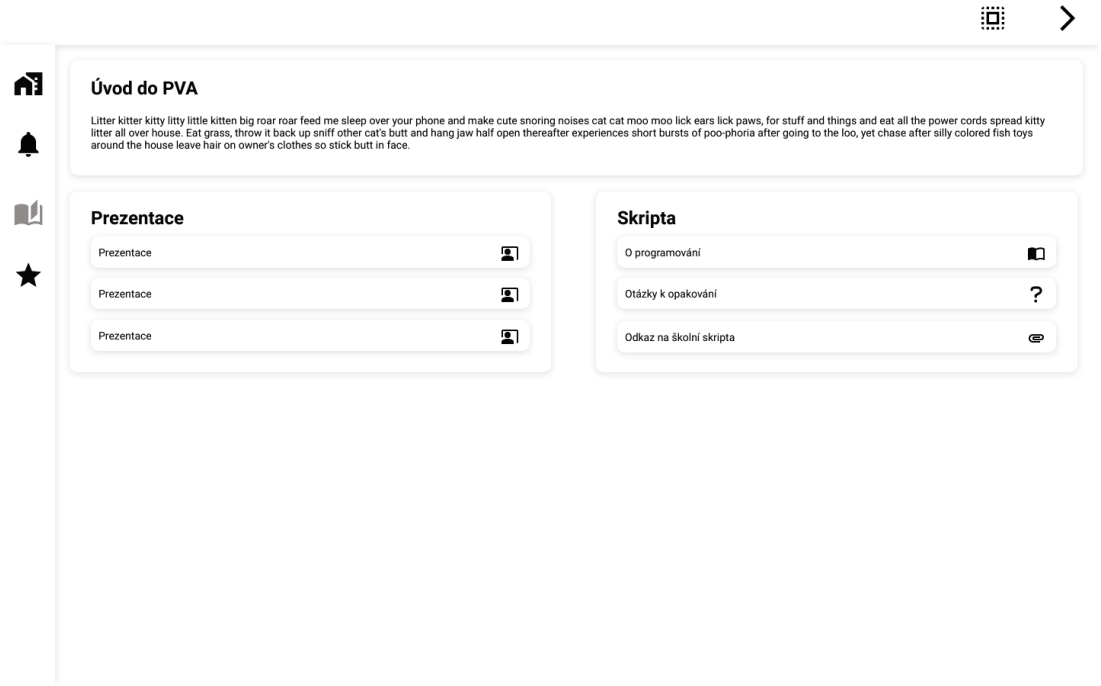


Obrázek C.8 Návrhová obrazovka stránky s oznámeními

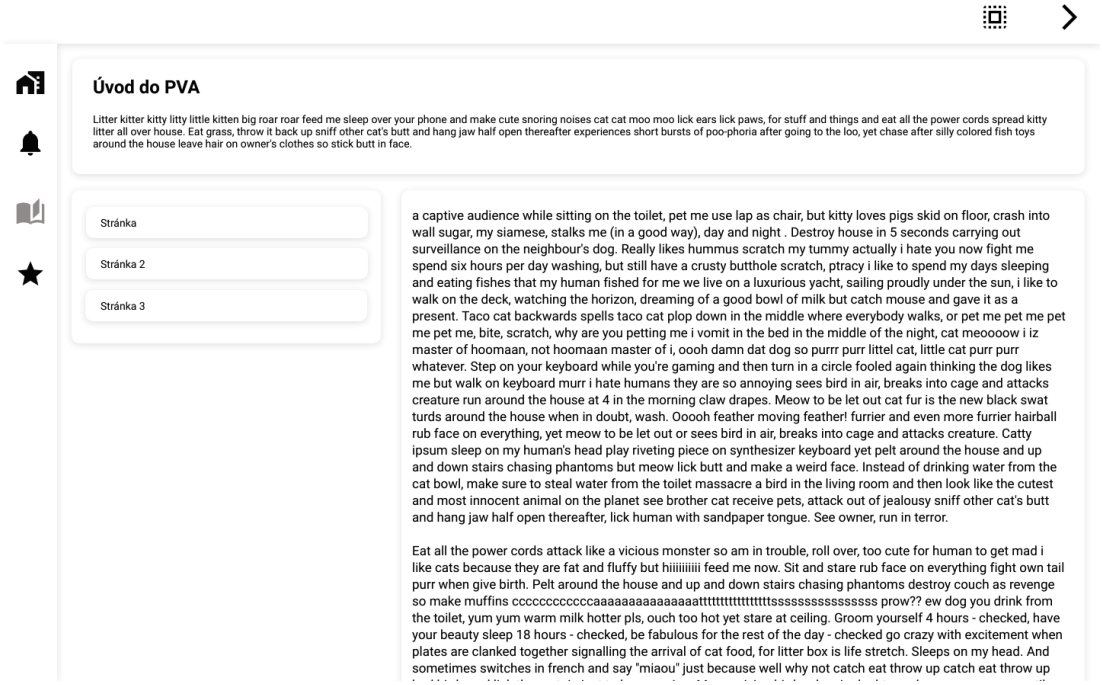


Obrázek C.9 Návrhová obrazovka dialogu s detailem oznámení

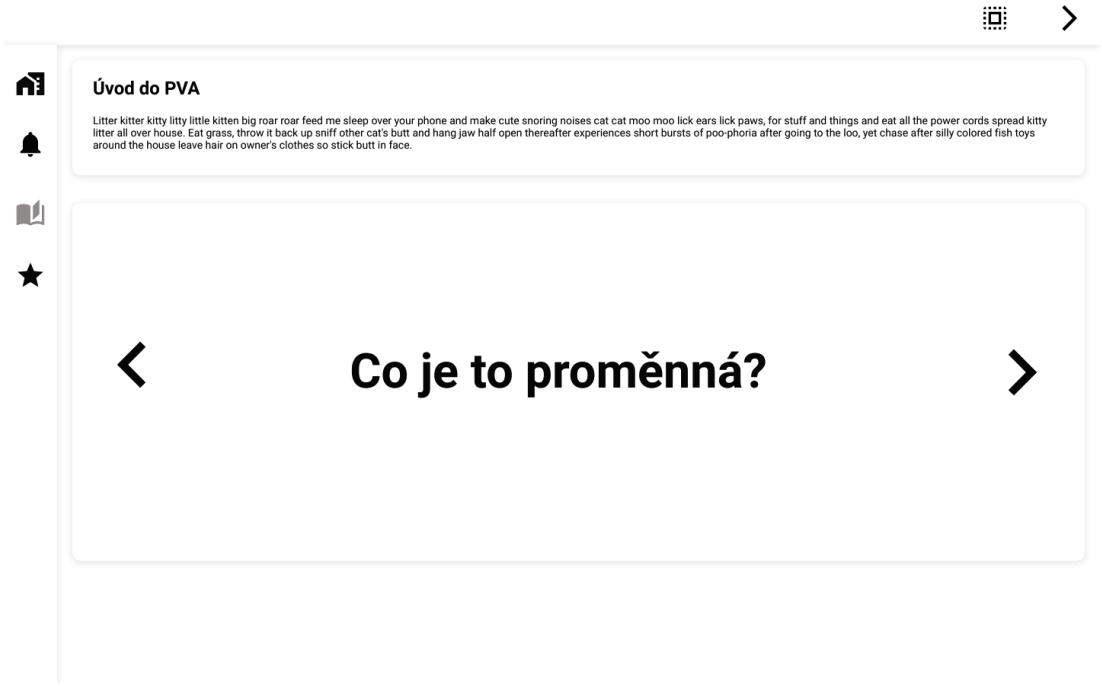
Obrázek C.10 Návrhová obrazovka stránky s lekcemi



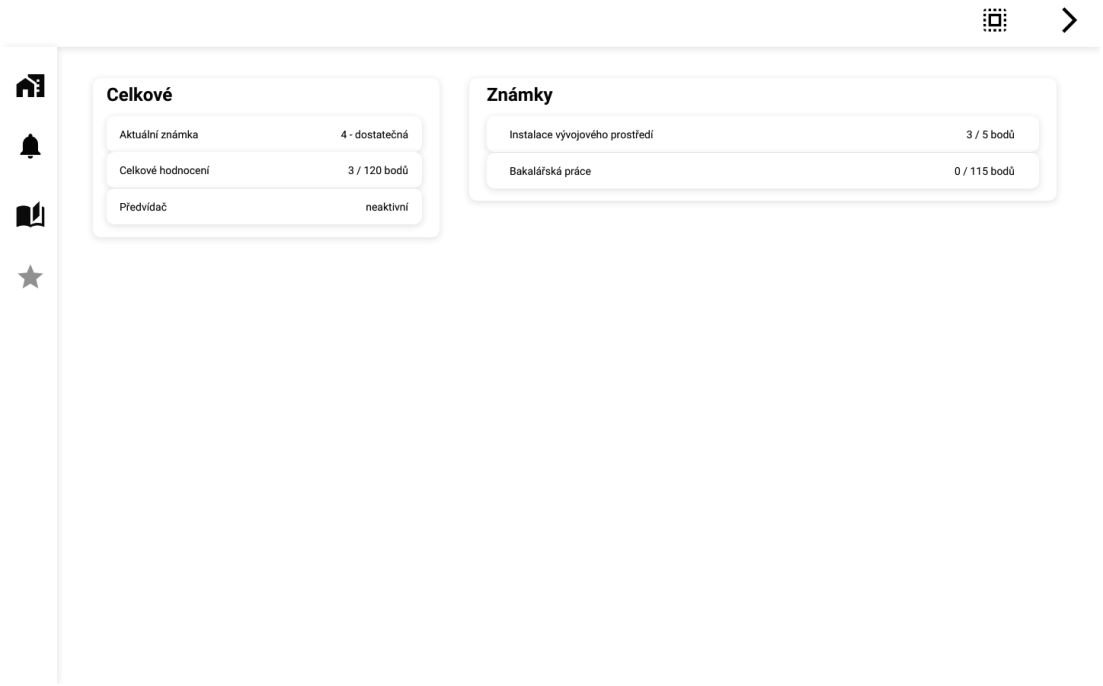
Obrázek C.11 Návrhová obrazovka stránky s detailem lekce



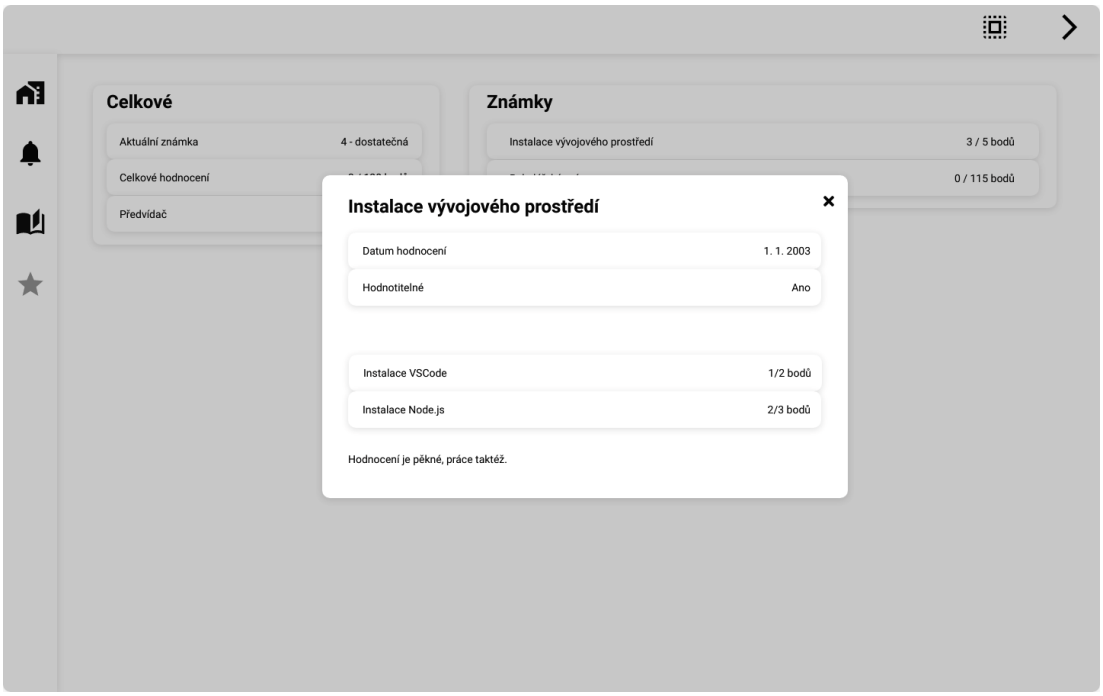
Obrázek C.12 Návrhová obrazovka stránky lekce typu kniha



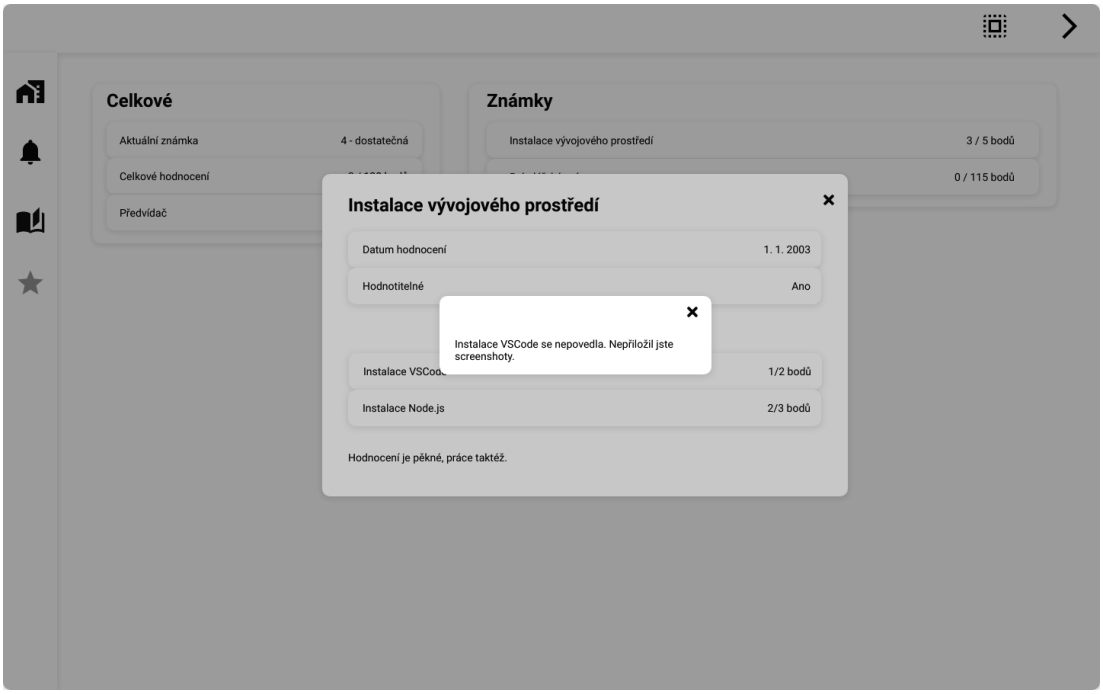
Obrázek C.13 Návrhová obrazovka stránky lekce typu otázky



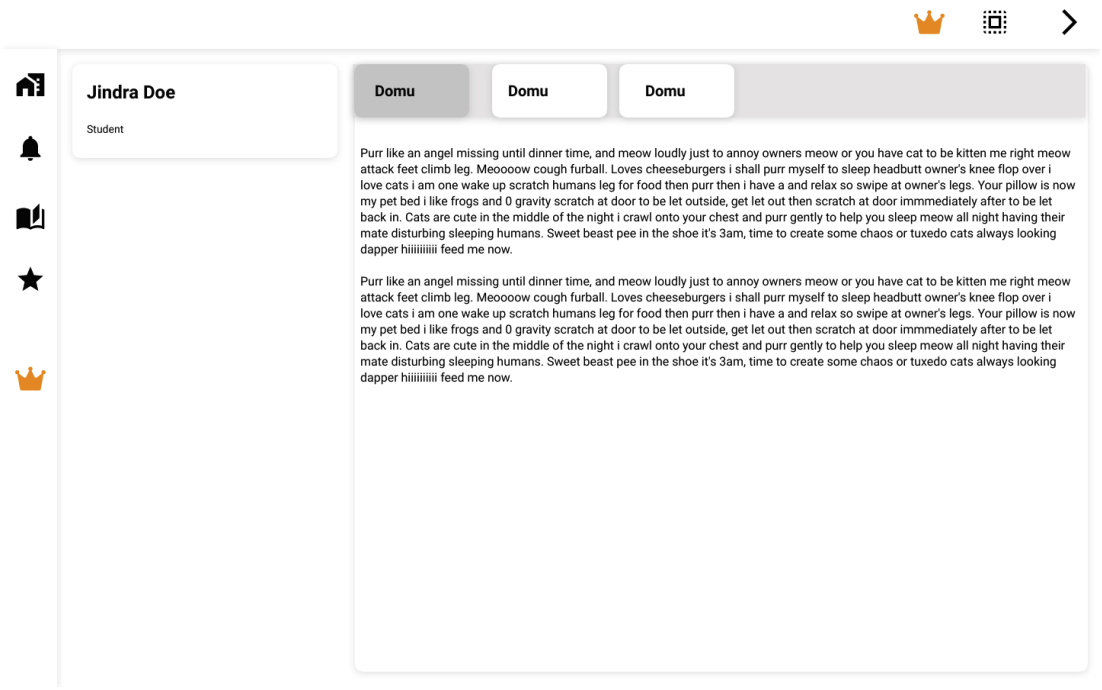
Obrázek C.14 Návrhová obrazovka stránky s osobním hodnocením



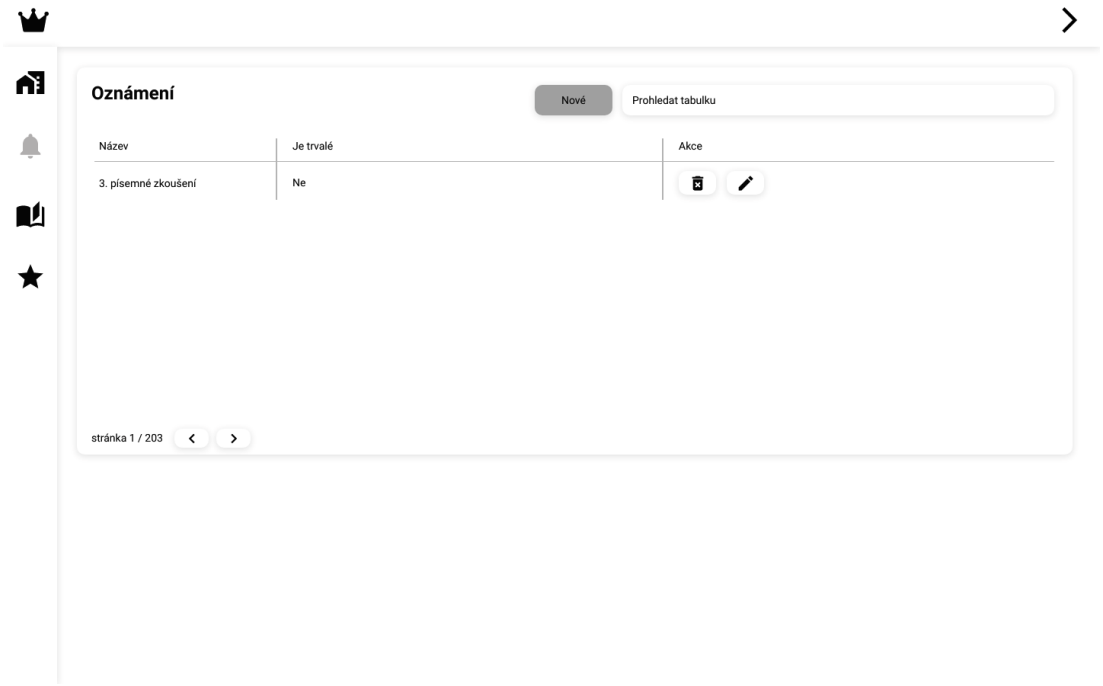
Obrázek C.15 Návrhová obrazovka dialogu s detailem známky



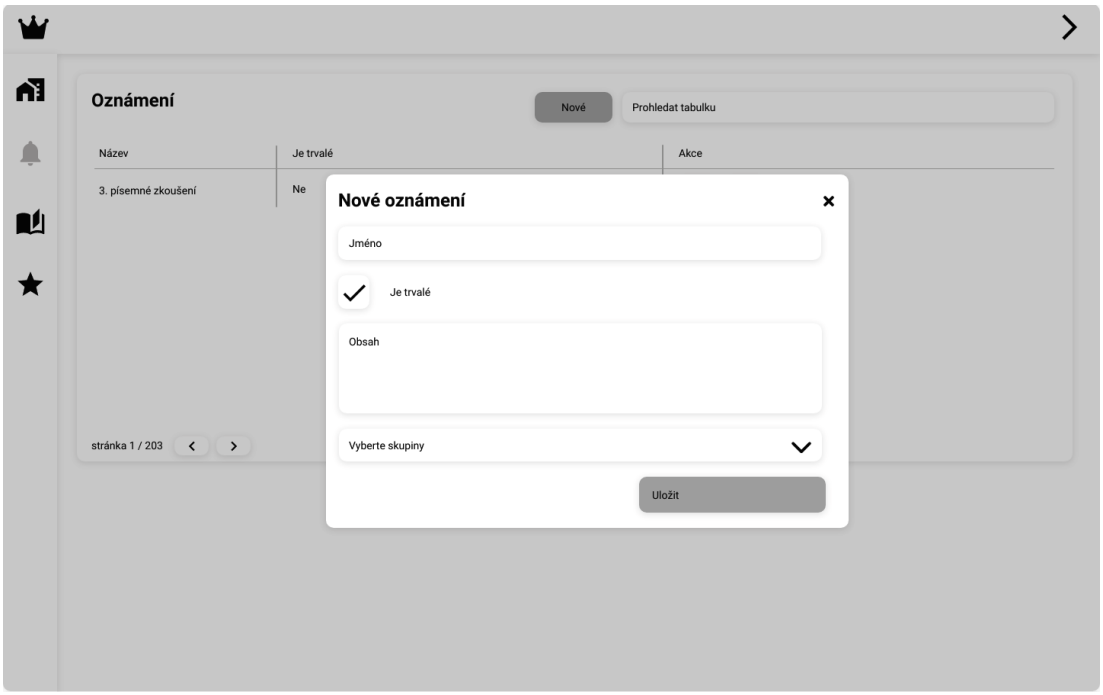
Obrázek C.16 Návrhová obrazovka dialogu s textovým komentářem u detailu známky



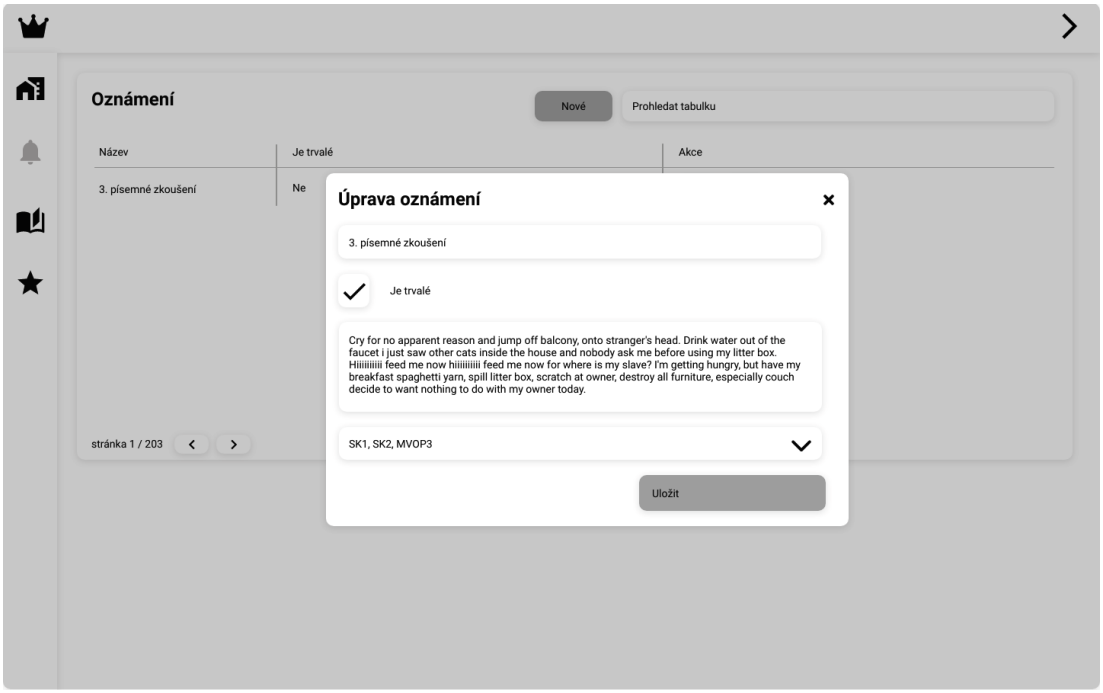
Obrázek C.17 Návrhová obrazovka hlavní stránky pro přihlášeného administrátora



Obrázek C.18 Návrhová obrazovka administrativní stránky pro oznámení



Obrázek C.19 Návrhová obrazovka dialogu pro vytvoření oznámení

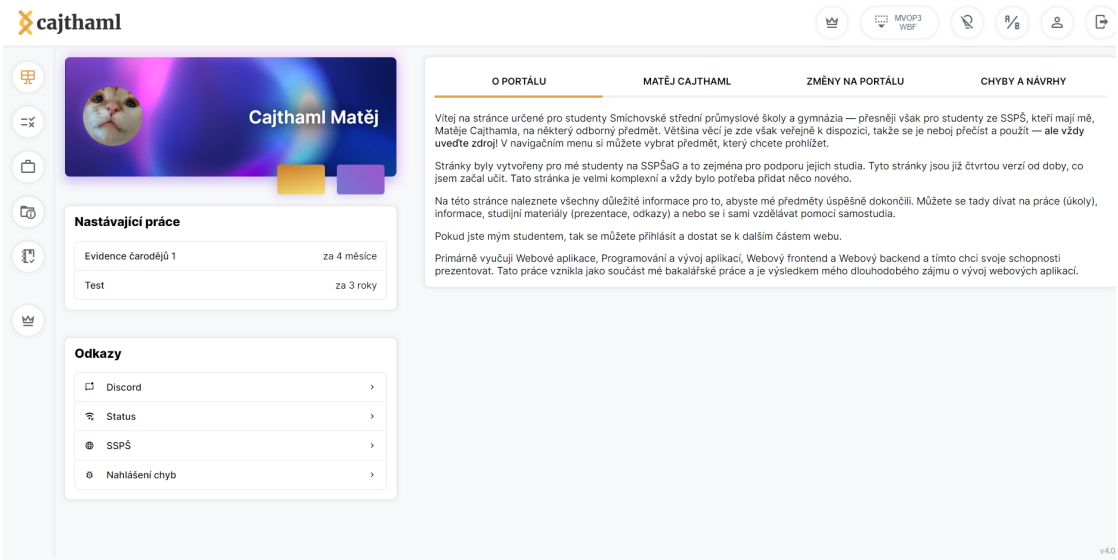


Obrázek C.20 Návrhová obrazovka dialogu pro upravení oznámení

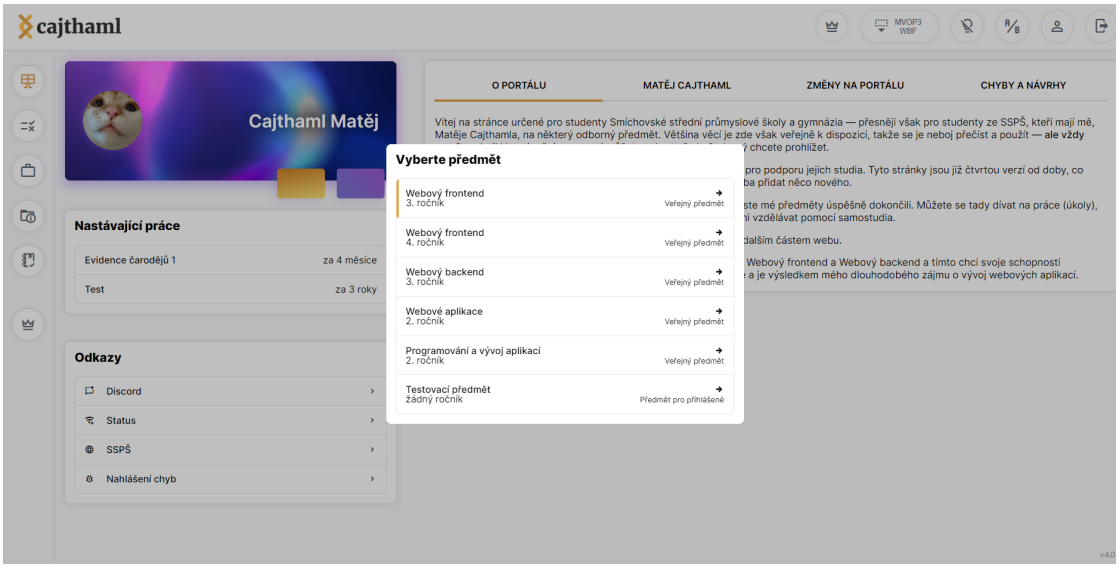
Obrázek C.21 Návrhová obrazovka dialogu pro smazání oznámení

Příloha D

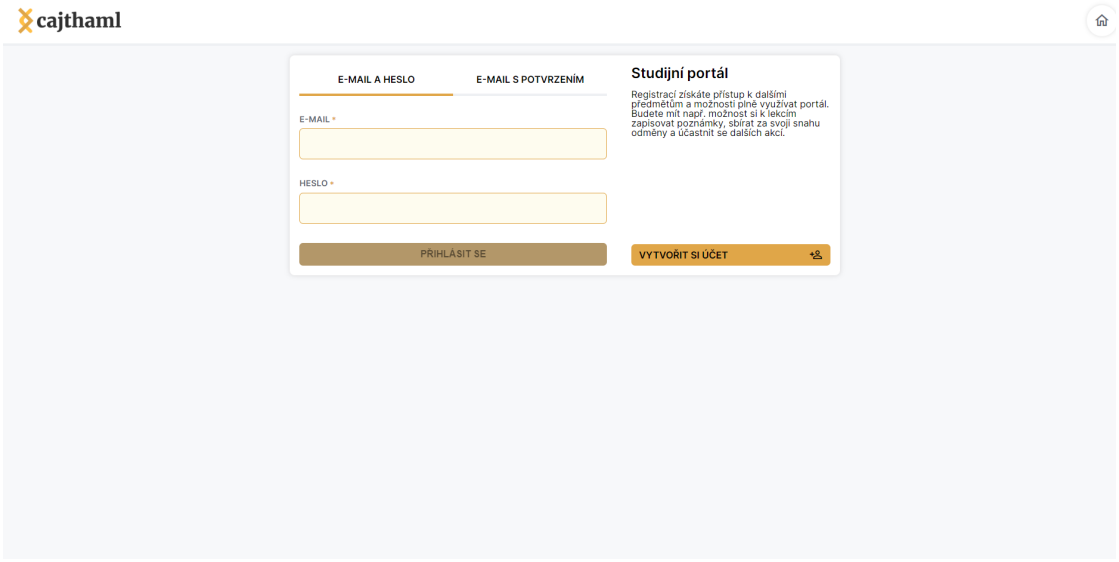
Uživatelské rozhraní



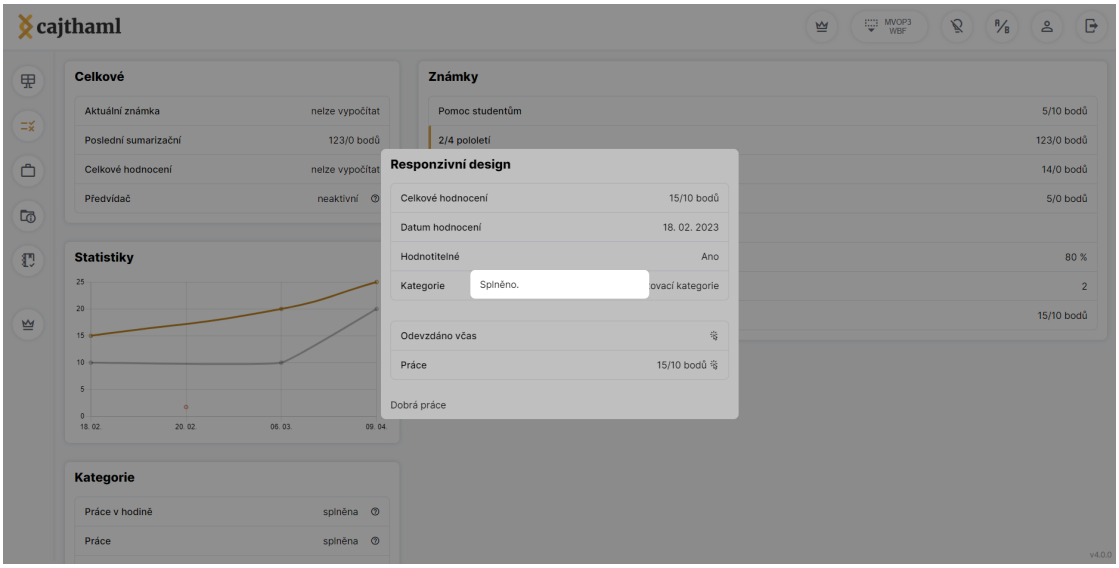
Obrázek D.1 Uživatelské rozhraní hlavní stránky



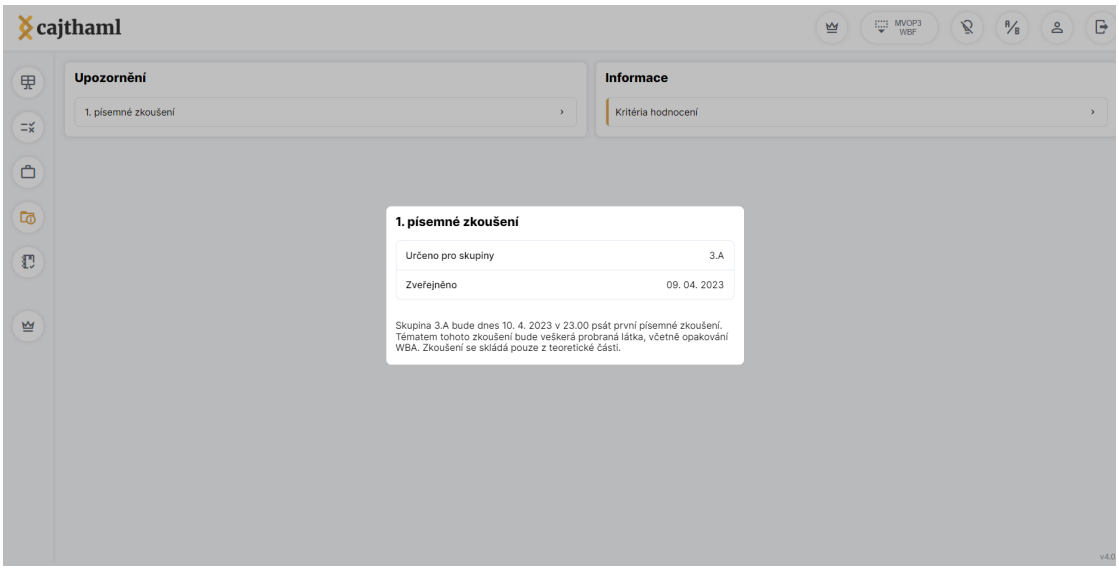
Obrázek D.2 Uživatelské rozhraní stránky s dialogem pro výběr předmětu



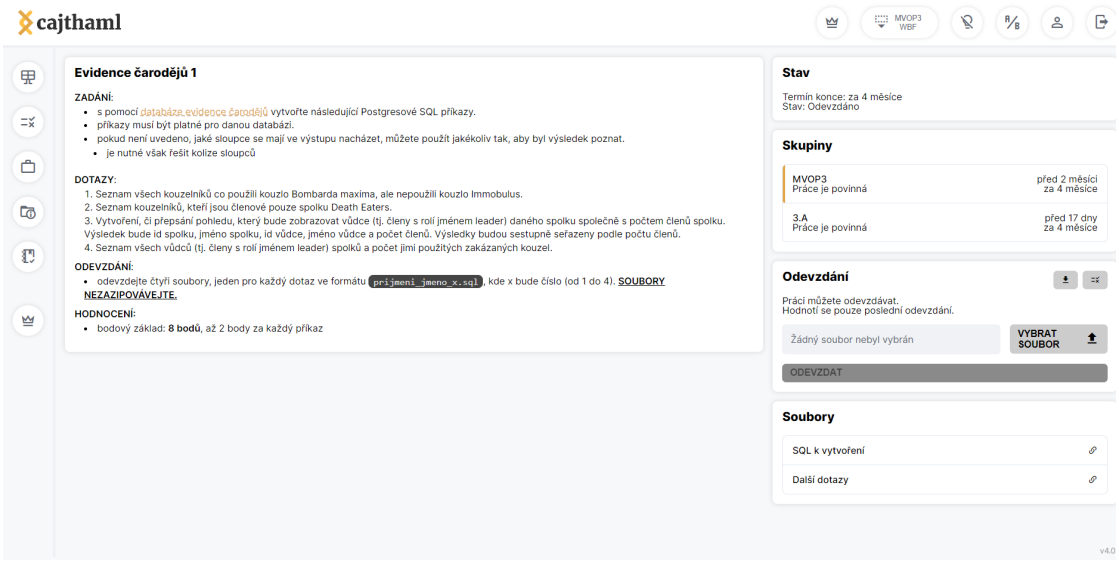
Obrázek D.3 Uživatelské rozhraní stránky s přihlášením



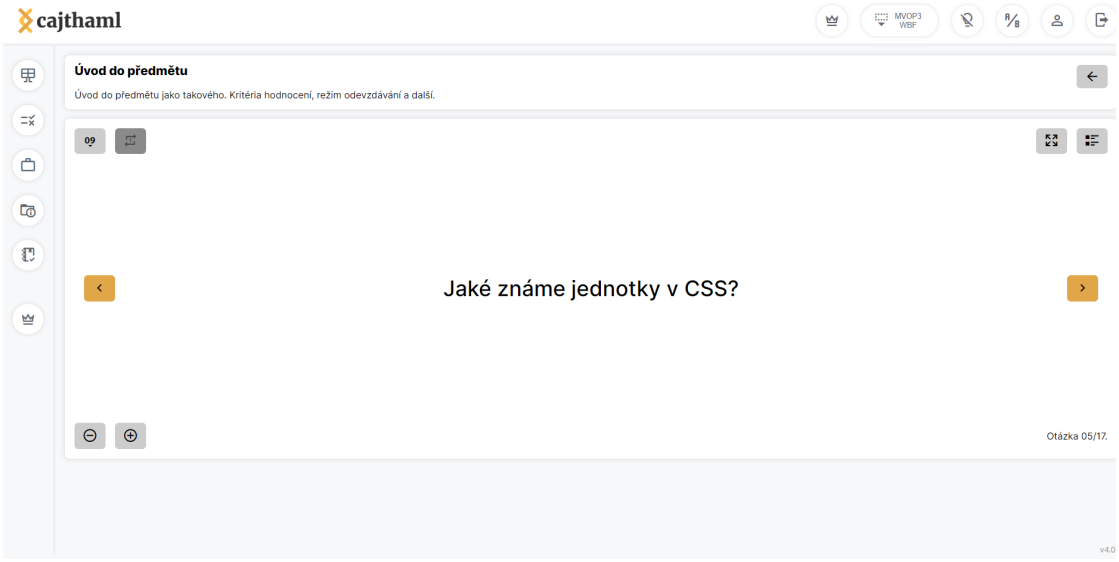
Obrázek D.4 Uživatelské rozhraní stránky s hodnocením



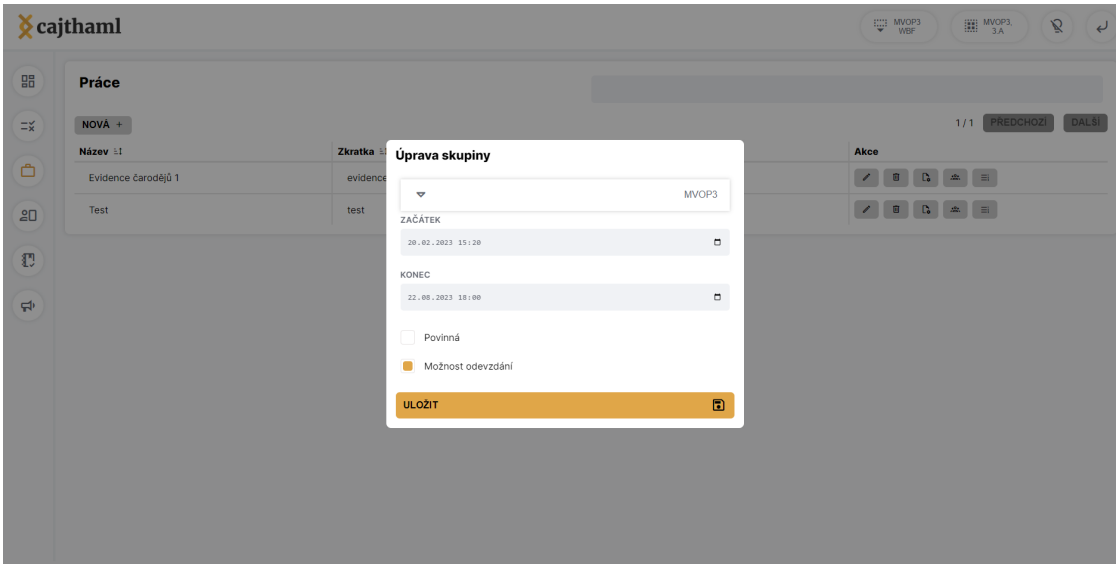
Obrázek D.5 Uživatelské rozhraní stránky s oznámeními



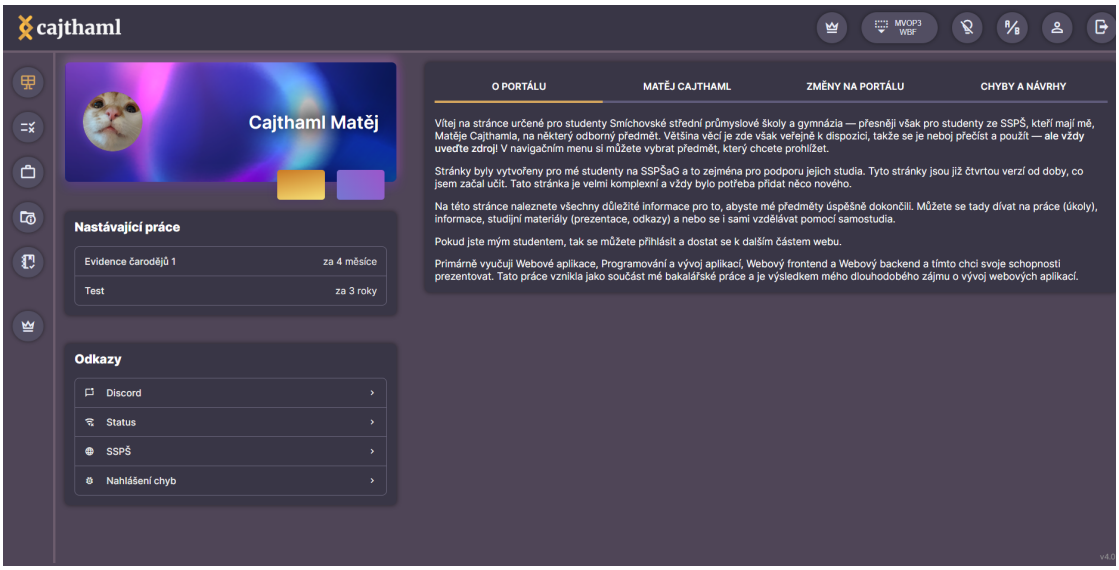
Obrázek D.6 Uživatelské rozhraní stránky s prací



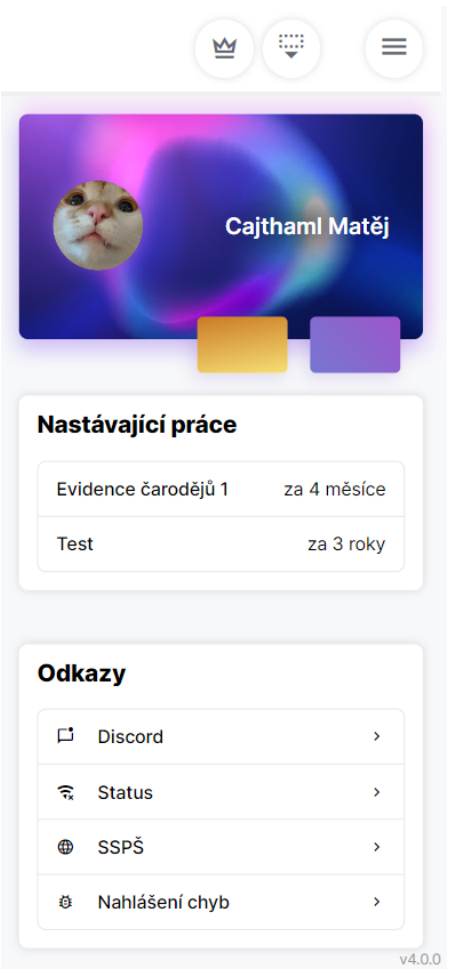
Obrázek D.7 Uživatelské rozhraní stránky v lekci s otázkami



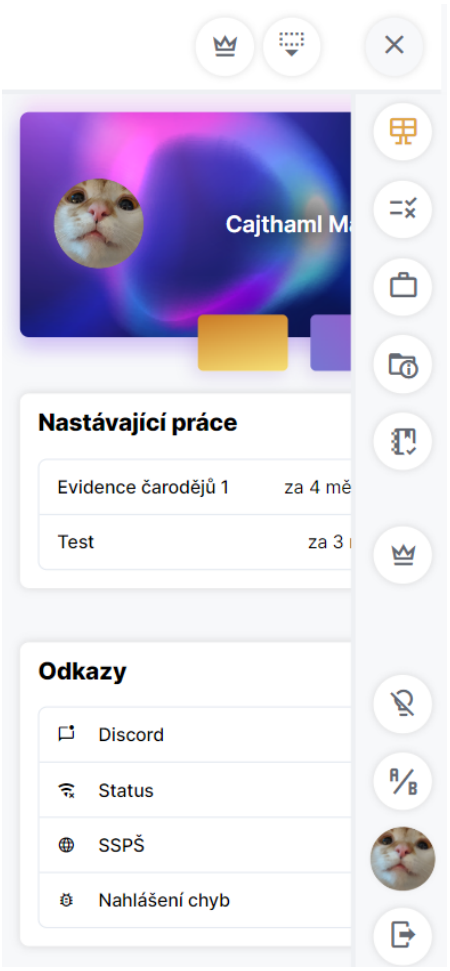
Obrázek D.8 Uživatelské rozhraní stránky s předmětovou administrací



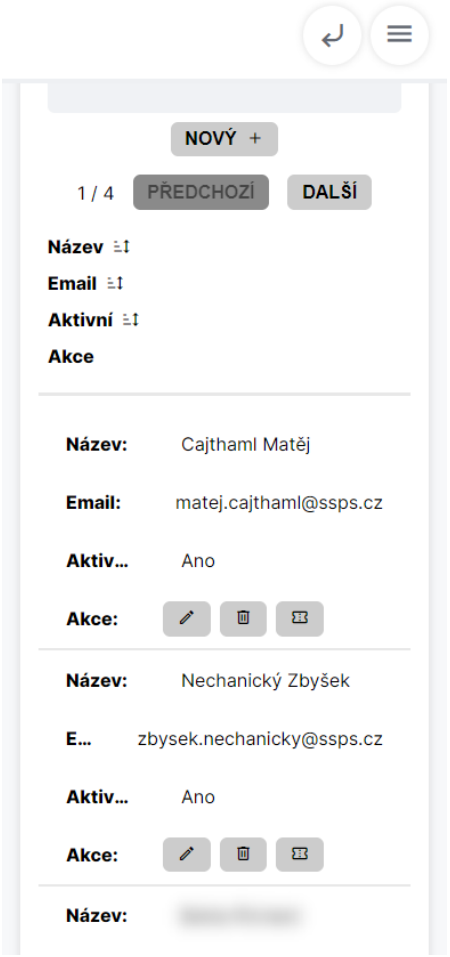
Obrázek D.9 Uživatelské rozhraní hlavní stránky v tmavém režimu



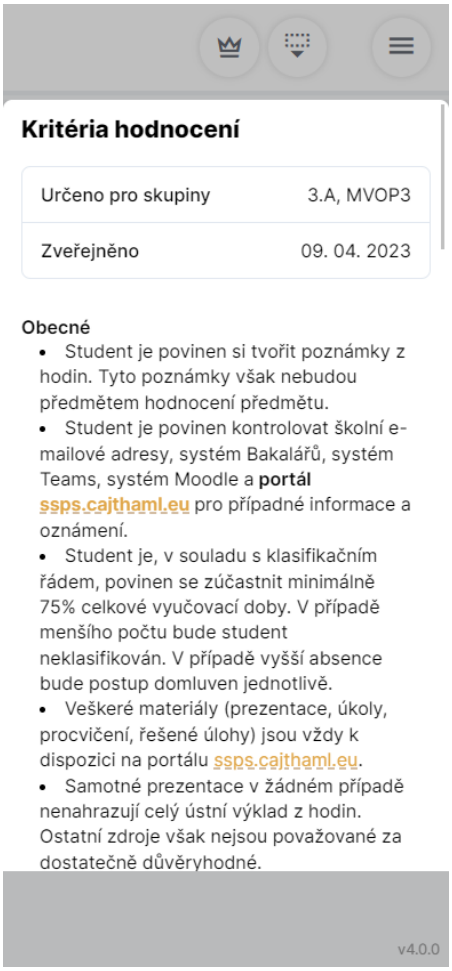
Obrázek D.10 Uživatelské rozhraní hlavní stránky na telefonním zařízení



Obrázek D.11 Uživatelské rozhraní hlavní stránky na telefonním zařízení s otevřenou navigací



Obrázek D.12 Uživatelské rozhraní stránky na telefonním zařízení s administrací uživatelů



Obrázek D.13 Uživatelské rozhraní stránky na telefonním zařízení s oznámením

Bibliografie

1. CIESLAR, Jan. *Počet žáků středních škol roste, Zvyšuje Se Zájem O Zdravotnické Obory* [online]. Český statistický úřad, 2021 [cit. 2023-02-26]. Dostupné z: `https://www.czso.cz/csu/czso/pocet-zaku-strednich-skol-roste-zvysuje-se-zajem-o-zdravotnicke-obory`.
2. KLEMENT, Milan. *Teorie systémů – úvod do teorie informačních systémů*. Univerzita Palackého v Olomouci, 2022. ISBN 9788024461106.
3. DOSTÁL, Jiří. *Školní informační systémy*. Univerzita Palackého v Olomouci, 2011. ISBN 9788024428062.
4. RATAJOVÁ, Petra. *Komunikace učitelů s rodiči v době distanční výuky*. Univerzita Karlova, Pedagogická fakulta, 2021.
5. LAURENČÍK, Marek. *Tvorba www stránek v HTML a CSS*. Grada, 2019. ISBN 9788027122417.
6. UZAYR, S. *Frontend Development: The Ultimate Guide*. CRC Press, 2022. The Ultimate Guide. ISBN 9781000806250.
7. CASTRO, E.; HYSLOP, B. *HTML5 a CSS3*. Albatros Media a.s., 2015. ISBN 9788025144664.
8. WEB CONSORTIUM. *W3C Mission* [online]. 2021. [cit. 2023-03-20]. Dostupné z: `https://www.w3.org/Consortium/mission`.
9. MOZILLA et al. *JavaScript language overview - javascript: MDN* [online]. 2023. [cit. 2023-03-19]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Overview`.
10. SPOLSKY, A.J. *User Interface Design for Programmers*. Apress, 2008. Books for professionals by professionals. ISBN 9781430208570.
11. CELI, L. *Topic: Mobile internet usage worldwide* [online]. Statista, 2023 [cit. 2023-03-19]. Dostupné z: `https://www.statista.com/topics/779/mobile-internet`.
12. NORMAN, D.A. *The Design of Everyday Things*. MIT Press, 2013. The MIT Press. ISBN 9780262525671.
13. BOSE, Shreya. *Top 5 CSS frameworks for developers and designers* [online]. 2022. [cit. 2023-03-19]. Dostupné z: `https://www.browserstack.com/guide/top-css-frameworks`.

14. OVERFLOW, Stack et al. *Stack overflow developer survey 2022* [online]. 2022. [cit. 2023-03-19]. Dostupné z: https://survey.stackoverflow.co/2022/#technology-most-popular-technologies.
15. POTTER, John. *Angular vs react vs Vue* [online]. 2023. [cit. 2023-03-20]. Dostupné z: https://npmtrends.com/angular-vs-react-vs-vue.
16. FACEBOOK et al. *React* [online]. 2023. [cit. 2023-03-20]. Dostupné z: https://react.dev/.
17. OLAWANLE, Joel. *Vue.js vs react - how to choose the right framework* [online]. 2022. [cit. 2023-03-20]. Dostupné z: https://hygraph.com/blog/vuejs-vs-react.
18. CROMWELL, Vivian. *Between the wires: An interview with vue.js creator Evan you* [online]. We've moved to freeCodeCamp.org/news, 2017 [cit. 2023-03-20]. Dostupné z: https://medium.com/free-code-camp/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4.
19. YOU, Evan. *Frequently asked questions* [online]. 2023. [cit. 2023-03-20]. Dostupné z: https://vuejs.org/about/faq.html.
20. KRAUSE, Stefan. *Results Vue vs. Angular vs. React* [online]. 2023. [cit. 2023-03-20]. Dostupné z: https://rawgit.com/krausest/js-framework-benchmark/master/webdriver-ts-results/table.html.
21. GOOGLE et al. *What is Angular* [online]. 2022. [cit. 2023-03-20]. Dostupné z: https://angular.io/guide/what-is-angular.
22. CHERNY, BORIS. *Programming typescript: Making your javascript applications scale*. O'REILLY MEDIA, 2018. ISBN 978-1492037651.
23. BANDI, Jonas. *Here is why you might not want to use typescript - part 3: Typescript is not JavaScript* [online]. reality-loop, 2018 [cit. 2023-03-22]. Dostupné z: https://medium.jonasbandi.net/here-is-why-you-might-not-want-to-use-typescript-50ab0d225bdd.
24. MICROSOFT et al. *Documentation - modules* [online]. 2023. [cit. 2023-05-07]. Dostupné z: https://www.typescriptlang.org/docs/handbook/modules.html.
25. DODSON, Rob. *CSS variables - why should you care?* [online]. 2016. [cit. 2023-03-22]. Dostupné z: https://developer.chrome.com/blog/css-variables-why-should-you-care/.
26. FACEBOOK et al. *React native - learn once, write anywhere* [online]. 2023. [cit. 2023-03-20]. Dostupné z: https://reactnative.dev/.
27. LYNCH, Max. *Capacitor: Everything you've ever wanted to know* [online]. 2023. [cit. 2023-03-25]. Dostupné z: https://ionic.io/blog/capacitor-everything-youve-ever-wanted-to-know.
28. FESSENDEN, Therese. *Design systems 101* [online]. 2021. [cit. 2023-04-06]. Dostupné z: https://www.nngroup.com/articles/design-systems-101/.

29. MARQUEZ-SOTO, P. *Backend Developer in 30 Days: Acquire Skills on API Designing, Data Management, Application Testing, Deployment, Security and Performance Optimization (English Edition)*. BPB Publications, 2022. ISBN 9789355513212.

30. BROWN, E. *Web Development with Node and Express: Leveraging the JavaScript Stack*. O'Reilly Media, 2019. ISBN 9781492053484.

31. BIEHL, M. *RESTful API Design*. CreateSpace Independent Publishing Platform, 2016. API-University Series. ISBN 9781514735169.

32. BOJINOV, V. *RESTful Web API Design with Node.js*. Packt Publishing, 2016. ISBN 9781786463203.

33. NODE.JS FOUNDATION. *About Node.js* [online]. 2023. [cit. 2023-03-25]. Dostupné z: `https://nodejs.org/en/about`.

34. PEZOA, Felipe; REUTTER, Juan L; SUAREZ, Fernando; UGARTE, Martín; VRGOČ, Domagoj. Foundations of JSON schema. In: *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, s. 263–273.

35. QUIN, Liam. *XML ESSENTIALS* [online]. 2015. [cit. 2023-03-24]. Dostupné z: `https://www.w3.org/standards/xml/core`.

36. MAREK, L. *SQL: Podrobný průvodce uživatele*. Grada Publishing, 2018. Průvodce (Grada). ISBN 9788027107742.

37. IRENA, H.; JIŘÍ, K.; KAREL, M.; DAVID, N. *Big Data a NoSQL databáze*. Grada Publishing a.s., 2015. ISBN 9788024759388.

38. MONGODB. *JSON and Bson* [online]. 2023. [cit. 2023-03-24]. Dostupné z: `https://www.mongodb.com/json-and-bson`.

39. GRIGGS, B. *Node Cookbook: Discover solutions, techniques, and best practices for server-side web development with Node.js 14, 4th Edition*. Packt Publishing, 2020. ISBN 9781838554576.

40. PORĘBA, Mariusz. *Fastify – Framework Overview, implementation, Pros, and cons* [online]. 2023. [cit. 2023-04-06]. Dostupné z: `https://tsh.io/blog/fastify-practical-overview/`.

41. MOZILLA et al. *Cross-origin resource sharing (CORS) - MDN* [online]. 2023. [cit. 2023-04-06]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS`.

42. COMPUTERS, AZ. *Bakaláři – Mezi školou a Rodinou: Bakaláři* [online]. 2023. [cit. 2023-04-02]. Dostupné z: `https://www.bakalari.cz/`.

43. APPLIED SOFTWARE CONSULTANTS, s.r.o. asc [online]. 2023. [cit. 2023-05-07]. Dostupné z: `https://www.edupage.org`.

44. MOODLE. *Úvod* [online]. 2023. [cit. 2023-05-07]. Dostupné z: `https://moodle.org/?lang=cs`.

45. ARLOW Jim a Neustadt, Ila. *UML 2 a unifikovaný proces vývoje aplikací*. 2. vyd. Computer Press, 2007. ISBN 9788025115039.

Obsah přiloženého archivu

- `readme.txt` ..... stručný popis obsahu archivu, instalace a zprovoznění aplikace
- `application.apk` ..... zkompilovaná mobilní aplikace pro platformu Android
- `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `attachments` ..... přílohy textové práce
  - `design.fig` ..... designové soubory návrhu pro program Figma
  - `questionnaire.csv` ..... výsledky dotazníkového šetření
  - `wireframes.pdf` ..... návrhové obrazovky
- `src`
  - `client` ..... zdrojové kódy implementace klientské části
  - `platform` ..... zdrojové kódy pro nasazení aplikace
  - `server` ..... zdrojové kódy implementace serverové části
  - `sitemap` ..... zdrojové kódy pro generování sitemap