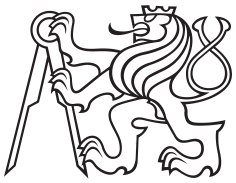
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Webový klient pro správu mobilních výukových úloh systému EduARd

Anastasia Surikova

Vedoucí: Ing. Ivo Malý, Ph.D.
Obor: Software
Studijní program: Otevřená informatika
Květen 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Surikova** Jméno: **Anastasia** Osobní číslo: **466389**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webový klient pro správu mobilních výukových úloh systému EduARd

Název bakalářské práce anglicky:

Pokyny pro vypracování:

Analyzujte strukturu systému EduARd, zejména komunikaci serverové části (backend) a jednotlivých klientských aplikací. Dále nastudujte mechanizmus správy výukových úloh a jednotlivé role uživatelů systému. Na základě analýzy doplňte a upravte základní návrh uživatelského rozhraní webového klienta. Hlavní funkce klienta budou umožňovat správu uživatelů systému, tvorbu jednotlivých výukových úloh a integraci dalších specializovaných editorů, které jsou vyvíjené dalšími členy vývojového týmu. Při vývoji postupujte v souladu s iterativní metodikou User-Centered Design (UCD). Prototyp implementujte pomocí vhodného frameworku, např. Angular nebo React. Aplikaci otestujte pomocí testů software i pomocí testů s uživateli.

Seznam doporučené literatury:

[1] K. Boogaart, You, I, and ReactiveUI, 2018.
[2] A. Hussain, Angular 5: From Theory To Practice: Build the web applications of tomorrow using the new Angular web framework from Google, CodeCraft, 2017.
[3] T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ivo Malý, Ph.D., katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. Ivo Malý, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studentky

Poděkování

Děkuji vedoucímu mé práce Ing. Ivovi Malému, Ph.D. za velmi cenné rady a zpětnou vazbu. Děkuji svojí matce za možnost studovat v Česku, za její lásku a podporu. Taky bych chtěla poděkovat svým přátelům za pomoc při studiu a svým kolegům za otestování aplikace.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškerou použitou literaturu.

V Praze, 25. května 2019

Abstrakt

Cílem této bakalářské práce je vytvoření webové aplikace pro správu uživatelů a mobilních výukových úloh systému EduARd. Aplikace komunikuje se serverovou částí systému EduARd přes poskytnuté REST API, umožňuje registrovaným správcům škol spravovat učitele a studenty a učitelům spravovat výukové úlohy. Hlavním zaměřením vývoje byla uživatelská zkušenost z použití aplikace. Důraz je kladen na vše, co tuto zkušenost může vylepšit: návrh, intuitivní použití uživatelského rozhraní, možnost náhledu a pravidelné testování s cílovou skupinou uživatelů.

Klíčová slova: webová aplikace, design zaměřený na uživatele, React.js, komunikace s REST API, systém na správu obsahu a uživatelů

Vedoucí: Ing. Ivo Malý, Ph.D.
katedra počítačové grafiky a interakce FEL

Abstract

The aim of this bachelor thesis is to create a web application for user and mobile educational tasks management of the EduARd system. The application communicates with the server side of the EduARd system via the provided REST API, allowing registered school administrators to manage teachers and students and teachers to manage educational tasks. The main focus of the development was user experience. Emphasis is placed on everything that can improve this experience: design, intuitive user interface, preview and regular testing with target audience.

Keywords: web application, user-centered design, React.js, REST API communication, content and user management system

Title translation: Web application for EduARd mobile learning tasks management

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

1.1 Motivace

V dnešní době se již těžko obejdeme bez počítačů a mobilních zařízení. Poskytují nám zajímavé a propracované aplikace, které nám usnadňují život. Chystáte se ráno ven a nevíte jak se obléci? Zapnete si aplikaci na předpověď počasí. Potřebujete mít v práci přehled o splněných a nadcházejících projektech? Použijete aplikaci na sledování úkolů. Plánujete se jít po práci podívat na film? Zjistíte přes webovou aplikaci aktuální program kina. Večer se vrátíte domů a pustíte si aplikaci na sledování spánku, která vás ráno probudí.

Technologie se rozvijí a významně ovlivňují všechny oblasti našeho života. Není výjimkou i oblast vzdělávání. Základní a střední školy za podpory státu a komerčních firem[2] kupují velké množství elektronických pomůcek pro podporu výuky. Jsou to počítače do každé učebny, tablety a mobilní telefony pro žáky a učitelé. Často si ale školy nevědí rady s obsahem, kterým by mohly tato zařízení naplnit.

Projekt EduARd, který se realizuje na Fakultě elektrotechnické Českého vysokého učení technického v Praze, si klade za cíl vytvořit systém pro podporu výuky humanitních a přírodních předmětů na základních a středních školách. Systém vytvořený v rámci projektu bude sloužit nejen pro tvorbu a sdílení učebních materiálů, ale také umožní přesunout výuku vybraných témat z lavic ven. Pomocí systému EduARd si učitelé budou moct vytvářet vlastní výukové úlohy, vázat je na GPS[1] a propojovat s 3D scénami virtuální reality. Systém umožní následně tyto materiály sdílet mezi učitele a studenty. Studenti si budou moct takovou úlohu stáhnout do svého chytrého telefonu nebo tabletu a na zadané lokaci si ji prohlédnout. V rámci výukové úlohy studenti najdou 3D scény dostupné k prohlížení v reálném čase přes kameru a také úkoly spojené s danou lokací.

Výsledkem projektu EduARd bude kvalitní software, který naplní zakoupená mobilní zařízení na základních a středních školách zajímavým obsahem a tím udělá výuku mnoha předmětů zajímavější a interaktivnější.

[1] Globální polohový systém

1.2 Cíle

Cílem této bakalářské práce je navrhnout a implementovat webovou část systému EduARd, která bude sloužit pro správu uživatelů a výukových úloh v rámci jedné školy.

V textu práce se nejdříve budeme věnovat systému EduARd jako celku, vysvětlíme si z jakých komponent se skládá a kteří uživatelé ho budou používat. Poté vytvoříme požadavky na výslednou webovou aplikaci pro správu uživatelů a výukových úloh a přejdeme k popisu její implementaci. Implementace bude vycházet ze základního návrhu aplikace poskytnutého zadavatelem, který doplníme o některé obrazovky. Výslednou aplikaci otestujeme pomocí programátorských testů a testů s uživateli. Na závěr vyhodnotíme výsledek, kterého se nám povedlo dosáhnout.

Kapitola 2
Analýza

Tato kapitola se zabývá analýzou systému, pro který tvoříme naši aplikaci, a určením požadavků na výsledek naší práce.

Systém EduARd se vyvíjí na Fakultě Elektrotechnické Českého vysokého učení technického v Praze v několikačlenném vývojovém týmu. Každý v tomto týmu má na starosti vývoj určité části systému. Tato práce se věnuje vývoji komponenty na správu mobilních výukových úloh, proto ostatní komponenty popíšeme pouze stručně.

Komponentám systému EduARd budeme nadále říkat **aplikace**. Vzdělávacím jednotkám, pro které je systém vyvíjen, budeme říkat **školy** nebo **instituce**.

2.1 Systém EduARd

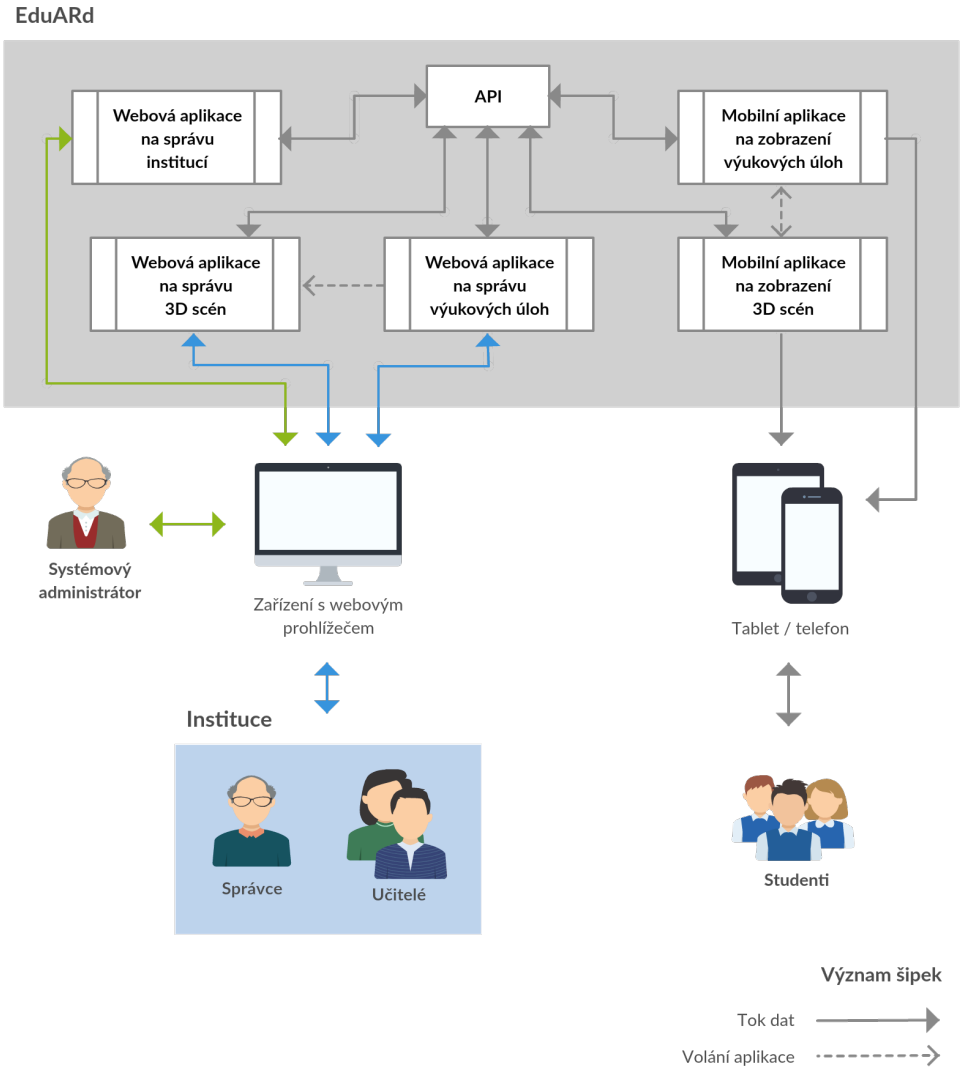
Systém EduARd se skládá ze dvou hlavních částí: serverové části a klientských aplikací.

- **Serverová část** se skládá pouze z jedné aplikace:
  - **REST API** — Obsahuje třídy a funkce, které obstarávají požadavky klientských aplikací. Podrobněji si o této části povíme v sekci 3.1
- **Klientské aplikace** využívají dvě platformy:
  - **Web**
    - **Aplikace na správu institucí** — Aplikace pro správu institucí a jejich správců.
    - **Aplikace na správu výukových úloh** — Aplikace na správu výukových úloh a uživatelů v rámci jedné instituce. Zaměření této bakalářské práce.
    - **Aplikace na správu 3D scén** — Aplikace na tvorbu 3D obsahu.

- **Mobil**
  - **Aplikace na zobrazení výukových úloh** — Mobilní aplikace umožňující prohlížení 2D obsahu.
  - **Aplikace na zobrazení 3D scén** — Mobilní aplikace umožňující prohlížení 3D obsahu.

2.2 Uživatelské role

Hlavními aktéry, kteří se systémem EduARd komunikují, je systémový administrátor, správce instituce, učitel a student. Každý z nich využívá svou platformu: administrátor, správce a učitel - web, studenti - mobilní zařízení. Pro lepší pochopení je tato myšlenka spolu s částmi systému EduARd znázorněna na obrázku 2.1.



Obrázek 2.1: Komunikace uživatelů s částmi systému EduARd.

Každý aktér má v systému EduARd svou uživatelskou roli:

- **Systémový administrátor** — Má přístup do webové aplikace pro správu institucí. V systému EduARd se jeho roli říká `Sysadmin`.
- **Správce instituce** — Má přístup do webové aplikace pro správu uživatelů v rámci jedné instituce. Má přehled o tom jací učitelé a studenti systém EduARd používají. V systému EduARd se jeho roli říká `InstitutionAdmin`.
- **Učitel** — Má přístup do webové aplikace pro správu výukových úloh a do rozhraní pro správu 3D scén. V systému EduARd se jeho roli říká `Editor`.
- **Student** — Má přístup do mobilních aplikací pro stahování a prohlížení výukových úloh a 3D scén. V systému EduARd se jeho roli říká `User`.

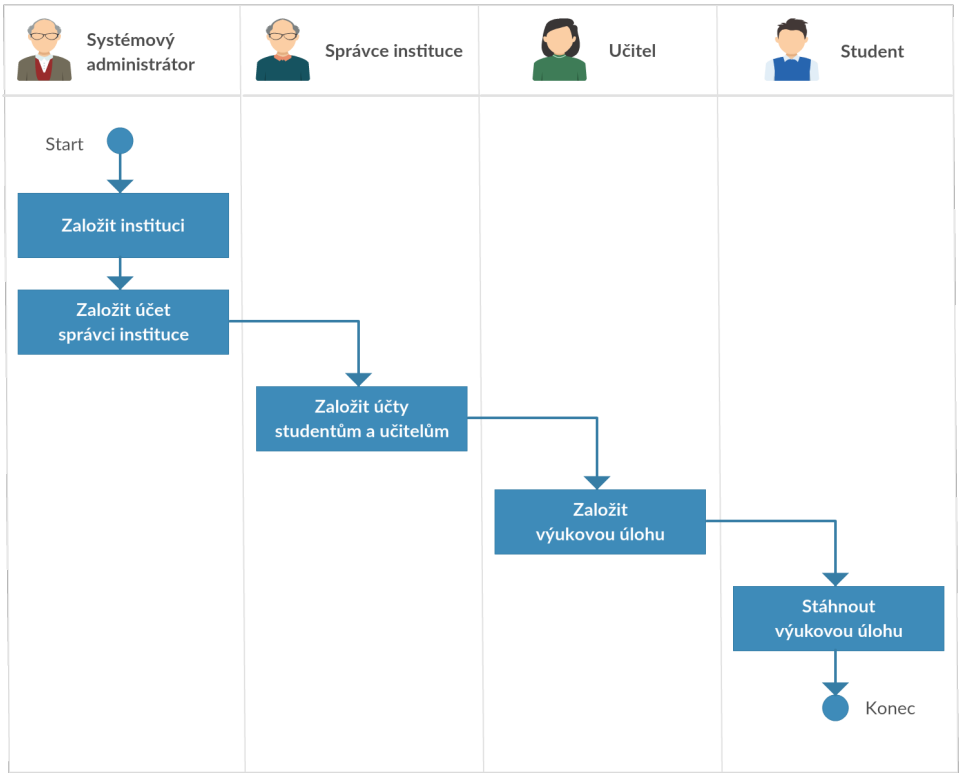
Oprávnění jednotlivých uživatelských rolí je také znázorněno v tabulce 2.1. Tato práce se zabývá vývojem webové aplikace pro správu uživatelů a výukových úloh (v tabulce znázorněno tučně), proto nás budou zajímat pouze 2 z uvedených rolí: `InstitutionAdmin` a `Editor`.

Role	Webová aplikace				Mobilní aplikace	
	správa institucí	**správa uživatelů**	**správa úloh**	správa 3D scén	prohlížení úloh	prohlížení 3D scén
Sysadmin	•					
InstitutionAdmin		•				
Editor			•	•		
User					•	•

Tabulka 2.1: Role a oprávnění uživatelů.

Použití systému EduARd začíná v okamžiku, kdy systémový administrátor založí novou instituci a poskytne jejímu správci přihlašovací údaje do aplikace na správu uživatelů. Tam správce instituce založí účty učitelům a studentům. Učitelé se následně přihlásí do aplikace na správu obsahu a mohou přidávat výukové úlohy. Studenti si je po přihlášení do mobilní aplikace mohou stáhnout k prohlížení. Tento proces je znázorněn na obrázku 2.2.

Lze si všimnout, že správa uživatelů a výukových úloh je v této práci spojená do jedné aplikace. Má to být systém, který budou používat školní zaměstnanci, a nemá smysl ho rozdělovat do dvou nezávislých částí.

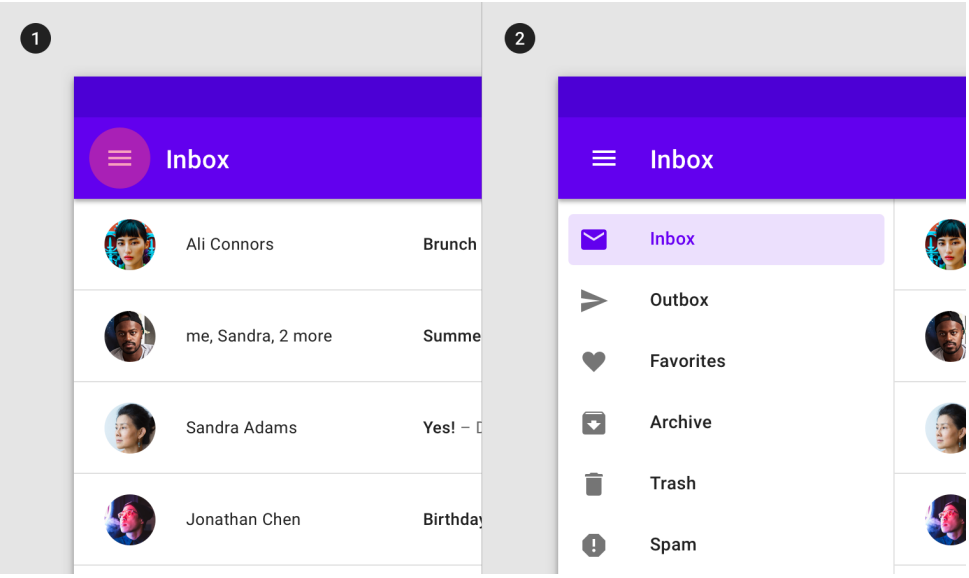


Obrázek 2.2: Digram aktivit znázorňující použití systému EduARd

2.3 Cílová skupina

Z předchozí sekce vyplývá, že cílovou uživatelskou skupinou naší aplikace jsou zaměstnanci školy, konkrétně její správce a učitelé. Věková kategorie se nedá přesně určit, proto musíme brát ohled jak na starší, tak i na mladší uživatelé. To znamená, že nebudeme upřednostňovat nové technologie v oblasti uživatelských rozhraní, ale spolehneme se na něco, co bude známo i starším generacím.

Například, jedním z nových trendů je schování hlavní nabídky webové stránky za tlačítko se třemi horizontálními čárami. Kliknutí na takové tlačítko rozbalí hlavní nabídku. Toto chování je znázorněno na obrázku 2.3. Dříve se toto řešení používalo pouze v mobilních verzích webových stránek, ale v dnešní době se to implementuje i do verzí, které uživatel vidí, když stránku otevře na počítači. Příklad implementace takového řešení můžeme nalézt na stránce Google Dokumentů[3]. Designeři a vývojáři se tak snaží přiblížit vzhled webové stránky ke vzhledu mobilní aplikace. Uživatelé, kteří aktivně mobilní aplikace používají, se hned zorientují a nabídku bez problémů najdou. Starší generace nemusí být v použití podobných technologií natolik zběhlá a může mít problém hlavní nabídku najít. Proto při implementaci naší aplikace dáme přednost použitelnosti uživatelského rozhraní před novými trendy v této oblasti.

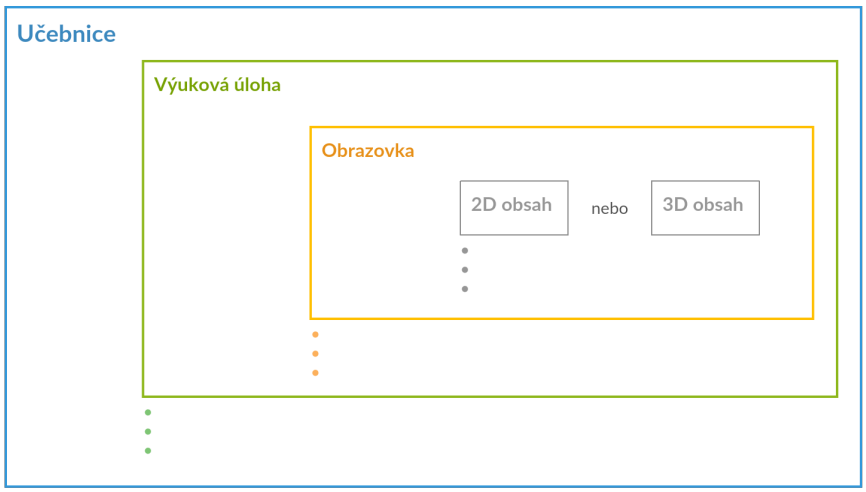


Obrázek 2.3: Ukázka zobrazení schované hlavní nabídky[1].

2.4 Struktura výukových úloh

Jak tedy vypadají jednotlivé výukové úlohy, jejichž správu máme implementovat?

Výukové úlohy se seskupují do učebnic a skládají se z obrazovek. Jednotlivé obrazovky reprezentují doslova obrazovku, kterou aktuálně vidí uživatel na svém zařízení, a může obsahovat buď 2D anebo 3D obsah. 3D obsahem se rozumí 3D scéna, jejíž tvorba a správa probíhá v jiné aplikaci systému EduARd. Součástí 2D obsahu může být text, obrázek a seznam otázek. Správa 2D obsahu bude probíhat v naší aplikaci. To, jak jednotlivé části zapadají do sebe, je znázorněno na obrázku 2.4.



Obrázek 2.4: Učebnice a její součásti

Správa výukových úloh tedy znamená nejen její přidání, odstranění a editaci, ale taky i správu učebnic, které úlohy seskupují, a obrazovek, které tvoří obsah jednotlivých úloh. Popíšeme tedy podrobněji jednotlivé části.

2.4.1 Učebnice

Seskupuje několik výukových úloh dohromady. Má následující vlastnosti:

- Maximální počet pokusů pro splnění úloh
- Rozložení úloh. Rozložení může být jedním ze tři druhů:
  - Zobrazení úloh na mapě
  - Zobrazení úloh v seznamu
  - Zobrazení seznamu a mapy zároveň

2.4.2 Výuková úloha

Reprezentuje soubor vzdělávacích materiálů navázaný na lokaci podle GPS souřadnic. Vlastnosti takového souboru jsou:

- Název
- Zeměpisná šířka
- Zeměpisná délka

V aplikaci a nadále v textu práce tuto část učebnice budeme nazývat **úkol**.

2.4.3 Obrazovka

Dílčí část úkolu. Obsah obrazovky má vzdělávací charakter a může to být 2D nebo 3D obsah. Oba druhy obrazovek mají společnou jednu vlastnost:

- Název

Obrazovka s 3D obsahem kromě názvu v sobě nese pouze jednu informaci:

- Označení připojené 3D scény

Obrazovka s 2D obsahem má následující informace:

- Popis
- Obrázek
- Pozice obrázku. Může nabývat následujících hodnot:
  - Výchozí pozice
  - Obrázek je napravo od textu
  - Obrázek je nalevo od textu
  - Obrázek je pod textem
  - Obrázek je nad textem
- Seznam otázek

2.4.4 Otázka

Otázka tvoří další komplexnější prvek s vlastními charakteristiky. Charakteristiky se liší u jednotlivých druhů otázek, kterých může být několik:

- Otázka s textovou odpovědí
- Otázka s číselnou odpovědí
- Otázka s jednou správnou odpovědí
- Otázka s více správnými odpověďmi
- Seřazovácí otázka
- Otázka na interval
- Přiřazení tvrzení k jedné z dostupných opcí

Všechny otázky mají společné následující vlastnosti:

- Typ otázky
- Text otázky
- Správná odpověď
- Nutnost zamíchat možné odpovědi
- Nutnost vyhodnotit uživatelskou odpověď

U otázek s textovou odpovědí se přidává další vlastnost:

- Nutnost rozlišovat velká a malá písmena u uživatelské odpovědi

Otázka na přiřazení tvrzení k dostupným opcím nese v sobe navíc:

- Seznam opcí

Otázka na interval navíc obsahuje:

- Rozsah intervalu
- Validní interval
- Krok intervalu

2.4.5 Shrnutí

Analýza struktury výukové úlohy nám rozdělila její správu na dílčí součásti:

- Správa učebnic
- Správa úkolů
- Správa obrazovek
- Správa otázek
- Správa obrázků učebnice

Z těchto dílčích součástí budou vyplývat funkční požadavky na naši aplikaci.

2.5 Funkční požadavky

Jedním z hlavních požadavků na naši aplikaci je to, že aplikace musí umožnit jak správu uživatelů, tak i správu výukových úloh. Rozdělíme tedy funkční požadavky do dvou odpovídajících částí.

2.5.1 Správa uživatelů

Přidání uživatele do systému EduARd funguje na principu pozvánek. Správce zadá e-mailovou adresu nového uživatele, kam se mu odešle pozvánka s odkazem pro nastavení hesla. Po proběhlém nastavení hesla je uživatel úspěšně přidán do systému. Naše aplikace musí umožnit správci odesílat pozvánky, mazat pozvánky a spravovat aktivní uživatele. Mechanismus odeslání pozvánky na e-mail a nastavení hesla řeší jiná aplikace systému EduARd.

Definujeme tedy funkční požadavky na naši aplikaci, které se týkají správy uživatelů:

- **FR01 - Přidání uživatelů** — Správce bude moct odeslat pozvánky k připojení do systému učitelům a studentům.
- **FR02 - Zrušení pozvánky** — Správce bude moct zrušit odeslanou pozvánku.
- **FR03 - Odstranění uživatele** — Správce bude moct odstranit učitele či studenta ze systému.
- **FR04 - Editace role uživatele** — Správce bude moct upravit uživatelskou roli učitelovi nebo studentovi.

2.5.2 Správa výukových úloh

V sekci 2.4 jsme si řekli, že správa výukových úloh se nám rozdělila na dílčí části. Na základě tohoto rozdělení definujeme funkční požadavky na správu výukových úloh:

- Správa učebnic
  - **FR05 - Přidání učebnice** — Učitel bude moct založit novou učebnici.
  - **FR06 - Odstranění učebnice** — Učitel bude moct odstranit existující učebnici.
  - **FR07 - Editace učebnice** — Učitel bude moct editovat vlastnosti existující učebnice.
- Správa úkolů
  - **FR08 - Přidání úkolu** — Učitel bude moct přidat úkol do zvolené učebnice.

  - **FR09 - Odstranění úkolu** — Učitel bude moct odstranit úkol z učebnice.
  - **FR10 - Editace úkolu** — Učitel bude moct editovat vlastnosti úkolu a jeho pořadí v seznamu úkolů.
- Správa obrazovek
  - **FR11 - Přidání obrazovky** — Učitel bude moct přidat obrazovku do zvoleného úkolu.
  - **FR12 - Odstranění obrazovky** — Učitel bude moct odstranit obrazovku z úkolu.
  - **FR13 - Editace obrazovky** — Učitel bude moct editovat vlastnosti obrazovky a její pořadí v seznamu obrazovek.
- Správa otázek
  - **FR14 - Přidání otázky** — Učitel bude moct přidat otázku do zvolené obrazovky.
  - **FR15 - Odstranění otázky** — Učitel bude moct odstranit otázku z obrazovky.
  - **FR16 - Editace otázky** — Učitel bude moct editovat vlastnosti otázky a její pořadí v seznamu otázek.
- Správa obrázků učebnice
  - **FR17 - Nahrání obrázku** — Učitel bude moct nahrát obrázek do zvolené učebnice.
  - **FR18 - Vložení obrázku** — Učitel bude moct vložit jeden z dostupných obrázků učebnice na zvolenou obrazovku.

Další funkční požadavky byly definovány zadavatelem:

- Správa úvodních obrazovek
  - **FR19 - Přidání úvodní obrazovky učebnice** — Učitel bude moct přidat úvodní obrazovku do zvolené učebnice.
  - **FR20 - Odstranění úvodní obrazovky učebnice** — Učitel bude moct odstranit úvodní obrazovku z učebnice.
  - **FR21 - Editace úvodní obrazovky učebnice** — Učitel bude moct editovat vlastnosti úvodní obrazovky a její pořadí v seznamu úvodních obrazovek.
- Integrace 3D obsahu
  - **FR22 - Přechod do aplikace pro správu 3D obsahu** — Učitel bude moct z aplikace přejít do jiné aplikace na správu 3D obsahu.
  - **FR23 - Vložení 3D obsahu** — Učitel bude moct vložit 3D scénu do zvolené obrazovky.

2.6 Kvalitativní požadavky

Od zadavatelů víme, co by od aplikace požadovali. Tedy jaké vlastnosti by měla naše aplikace mít.

- **QR01 - Intuitivní v použití** — Aplikaci budou používat zaměstnanci školy, proto by její použití mělo být intuitivní jak pro mladé, tak i pro starší uživatele.
- **QR02 - Možnost náhledu** — Bylo by dobré, aby uživatel viděl, jak budou vypadat informace, které právě do aplikace zadal.
- **QR03 - Multijazyčnost** — Aplikace by měla umožnit překlad popisku a nápověd i do jiných jazyků než čeština. V budoucnu by mělo být možné nasadit systém EduARd i do zahraničních škol.

2.7 Existující řešení

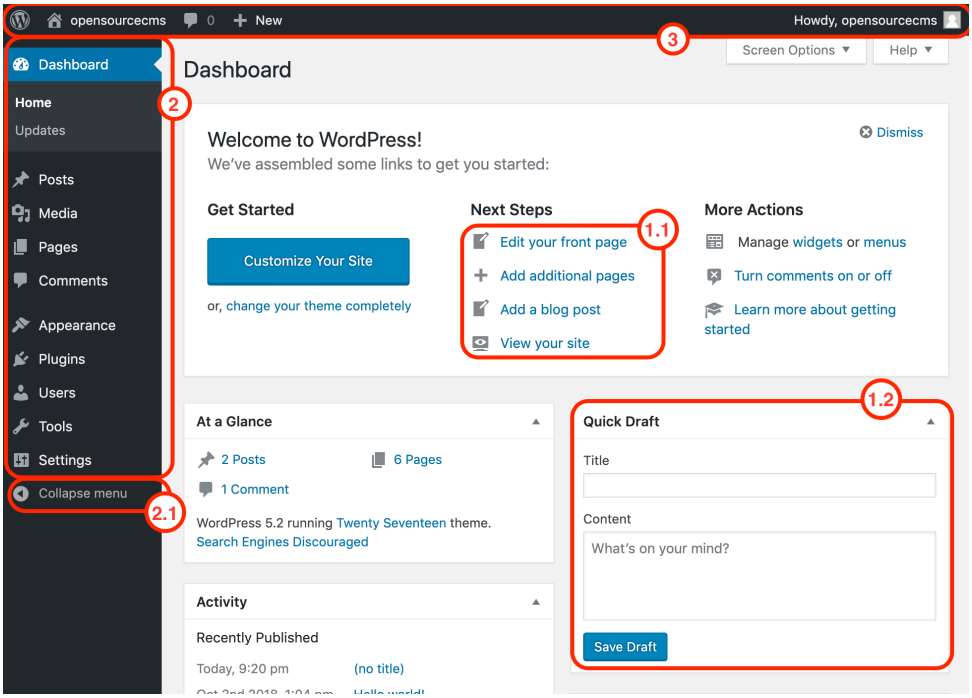
Webové řešení, kterým se zabývá tato bakalářská práce je pro systém EduARd šité na míru. Proto na trhu neexistuje jiná aplikace, která by splňovala všechny požadavky a uměla komunikovat s ostatními částmi systému EduARd. Podíváme se však na webové platformy, které nabízí správu obsahu webových stránek, tzv. CMS[1]. I když nejsou pro systém EduARd vhodné, mají podobný účel a mohli bychom v nich najít inspiraci pro naši aplikaci. Dvěma nejpopulárnějšími[4] platformami jsou Wordpress a Joomla!.

2.7.1 Porovnání podobných aplikací

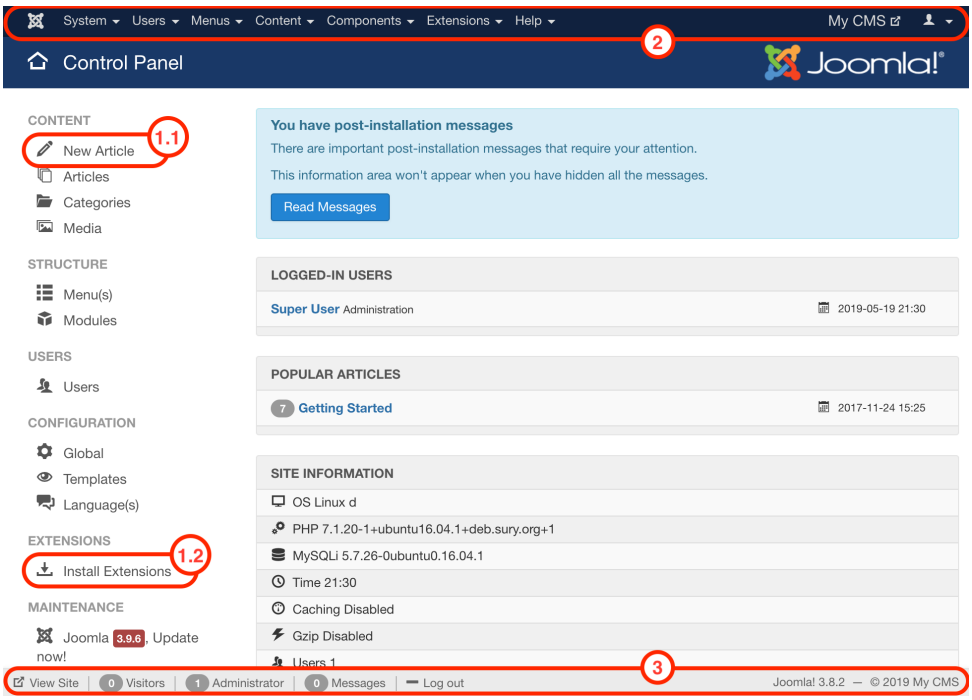
Webové aplikace pro správu obsahu Wordpress a Joomla! jsou obě napsány v jazyce PHP a využívají SQL databázi. Nás ale bude především zajímat uživatelský požitek z použití obou systémů než jejích implementace a funkcionalita.

Ovládací panel. První věc, kterou vidí uživatel po přihlášení, je tzv. ovládací panel - stránka, na které se nachází odkazy na rychlé akce (např. přidání článku či nabídky) a přehled aktivit. Lze si všimnout, že obě CMS mají ovládací panel a duplikují na něm některé položky hlavní nabídky. Uživateli také umožňují již z ovládacího panelu provést určité akce (viz označení 1.2 a 1.2 na obrázcích 2.5 a 2.6). Jediný vzhledový rozdíl je, že ovládací panel sytému Wordpress využívá blokové zobrazení, a ovládací panel systému Joomla! řádkové. Využití ovládacího panelu se osvědčilo v obou nejpopulárnějších systémech pro správu obsahu, proto je dobré ho implementovat i do naší webové aplikace.

[1] Content Management System - systém pro správu obsahu



Obrázek 2.5: CMS Wordpress



Obrázek 2.6: CMS Joomla!

Navigace. Jedním z největších rozdílů systému Wordpress a Joomla! je pozice navigace. Na obrázku 2.5 je číslem 2 označená hlavní nabídka systému Wordpress, která je vertikální a je umístěna vlevo od obsahu. Na obrázku 2.6

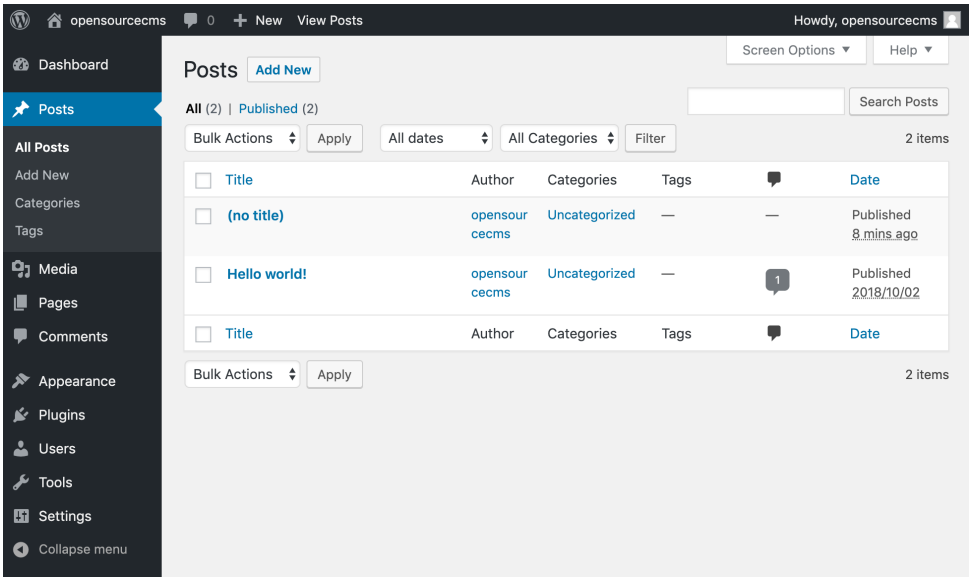
číslo 2 znázorňuje horizontální nabídku systému Joomla! umístěnou nahoře. Prostor v levé části stránky se v systému Joomla využívá pro boční navigaci, viz označení číslem 1 na obrázku 2.8.
Na první pohled se zdá, že řešení systému Wordpress zabírá větší prostor a ochuzuje uživatele o pracovní plochu, ale není tomu tak. Nabídka se dá zmenšit do užší podoby pomocí tlačítka „Collapse menu“, označeného na obrázku 2.5 číslem 2.1. Pak vidíme pouze ikony položek nabídky, jednotlivé popisky jsou po zmenšení schovány. Toto řešení se zdá dynamičtější, ale těžko se posuzuje, zda je lepší než horizontální nabídka nahoře, u které jsou popisky stále viditelné.

Možnost odhlášení. Oba systémy mají k dispozici stavový řádek, který udává počet uživatelů nebo aktualizací pluginů a umožňuje uživateli se z editačního rozhraní odhlásit (viz označení číslem 3 na obrázcích 2.5 a 2.6). V systému Worpress je tento řádek umístěn nahoře, v systému Joomla! je stavový řádek dole. Náš systém nebude tak komplexní, proto bych možnost odhlášení implementovala do stejného prostoru jako hlavní nabídku.

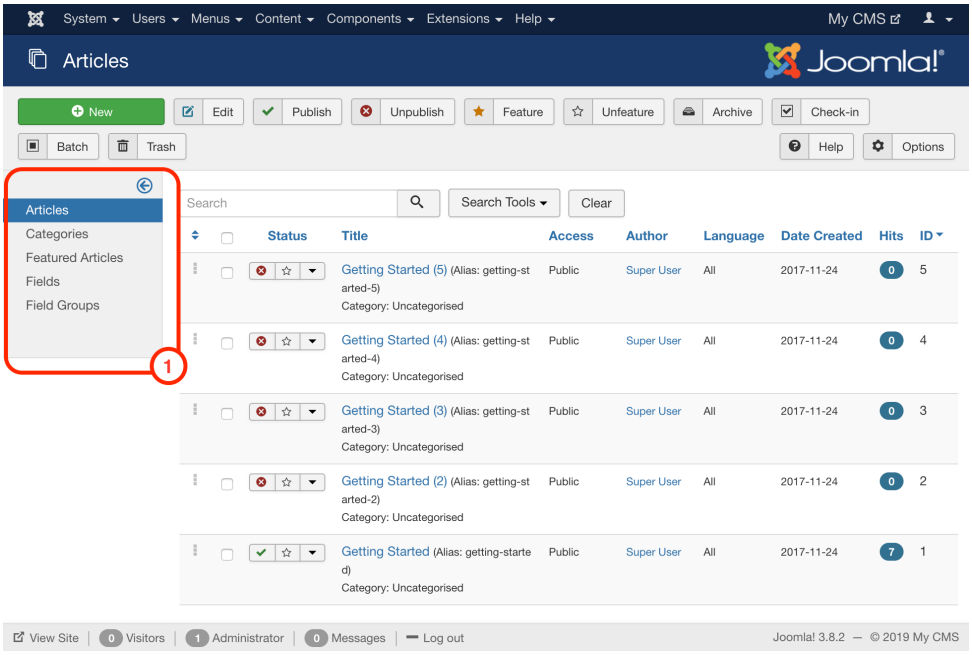
Barvy. Barevné řešení obou systémů jsou si podobná. Mají světlejší barvy pro hlavní pracovní plochu a tmavší barvy pro navigaci. Oba se drží barev, díky kterým aplikace vypadá profesionálně a důvěryhodně. Jedná se především o odstíny šedé a modré. Výraznější barvy mají pouze akční tlačítka, což je úplně v pořádku. Podobná barevná kombinace bude implementována i v naší aplikaci.

Zobrazení seznamu položek. Výpis položek v obou systémech vypadá podobně viz obrázek 2.7 a obrázek 2.8. V obou případech se jedná o zobrazení ve tvaru tabulky. Po kliknutí na název položky uživatel se dostane na její editaci. Vzhledem k tomu, že naše aplikace má být intuitivní pro různé věkové kategorie, přidala bych navíc i akční tlačítko, které uživatele přesměruje na editaci položky. Ne všechny uživatele může napadnout, že název nabídky je klikací, proto bych při implementaci zpracovala obě možnosti navigace.

Editace položky. Stránka pro editaci textového obsahu vypadá opět v obou systémech skoro stejně viz obrázky 2.9 a 2.10. Plusem je WYSIWYG editor (na obrázcích má označení číslem 1) umožňující uživateli ihned vidět výsledek formátování textu. Zkratka WYSIWYG je akronym věty „What You See Is What You Get“, česky „co vidíš, to dostaneš“. Tuto funkcionalitu by mohla mít i naše aplikace. Jediným rozdílem editoru systému Joomla! je nefunkčnost položek hlavní nabídky, což znamená, že uživatel nemůže z editační stránky odejít aniž by změny uložil či zamítnul kliknutím na jedno z akčních tlačítek. V systému Wordpress odkazy hlavní nabídky jsou na editační stránce aktivní, ale při pokusu o navigaci aplikace upozorní uživatele, že opuštěním stránky přijde o svoje změny. Toto řešení se mi zdá vhodnější, jelikož je dobré upozornit uživatele na to, co se stane, než mu pouze znemožnit navigaci bez jakéhokoliv vysvětlení.

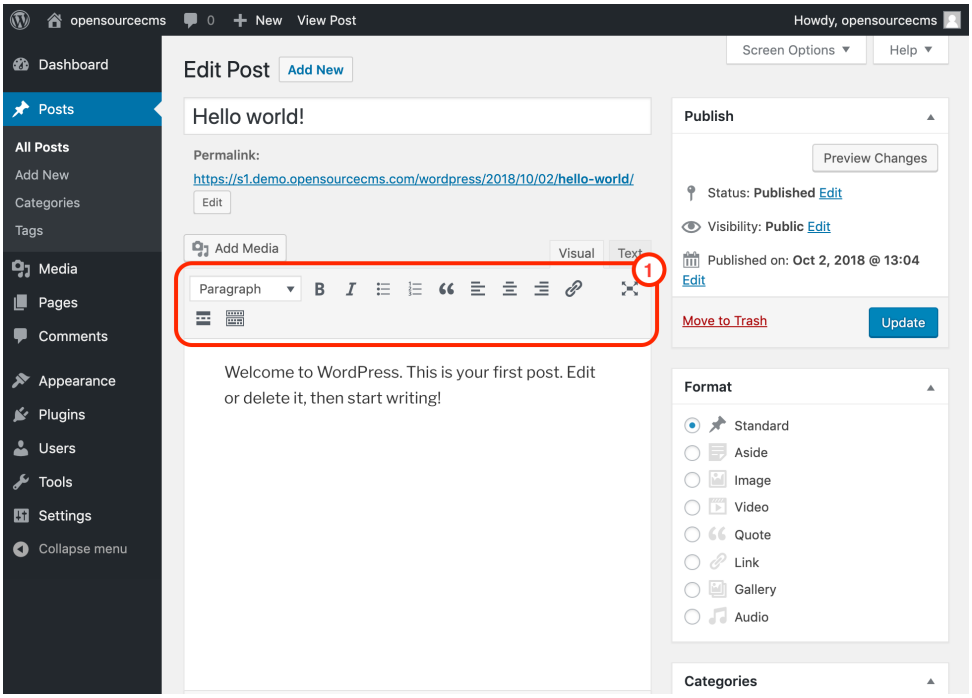


Obrázek 2.7: Zobrazení seznamu položek v CMS Wordpress

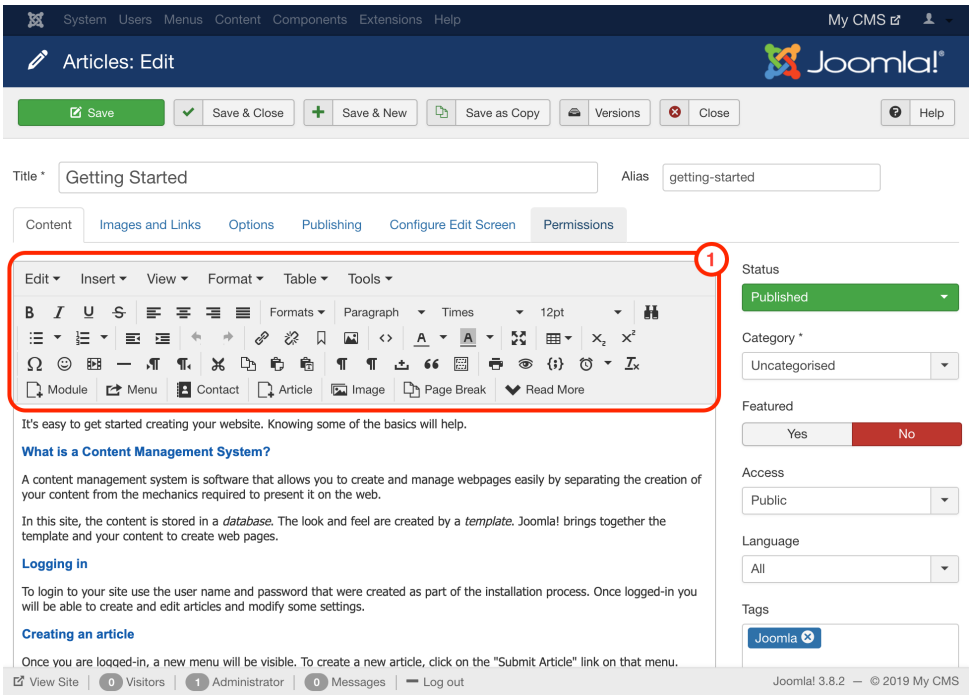


Obrázek 2.8: Zobrazení seznamu položek v CMS Joomla!

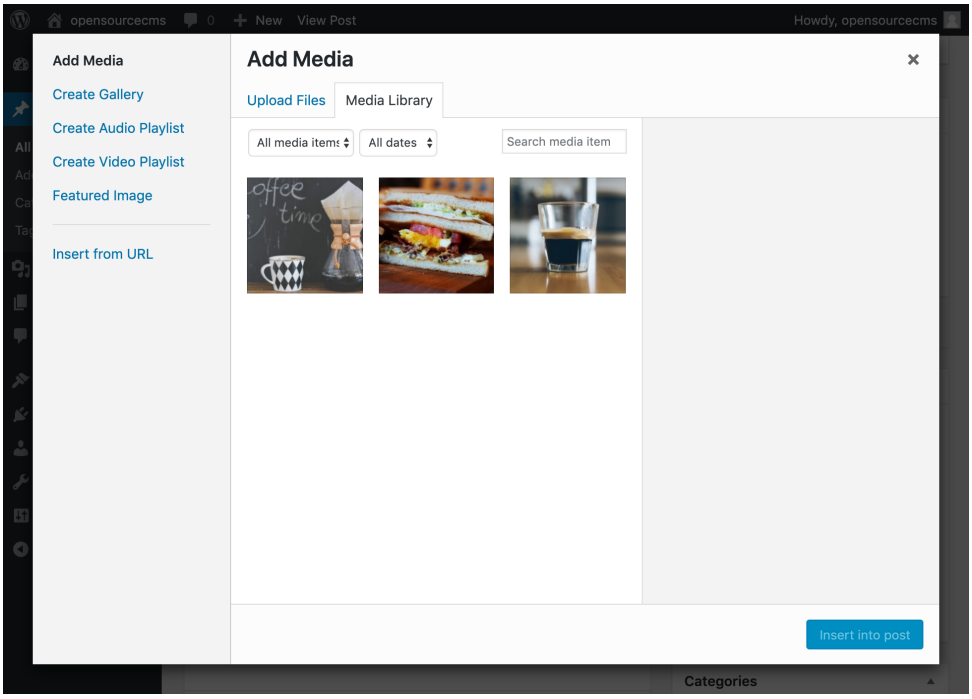
Galerie obrázků. Pro správu obrázků se v obou systémech používá Galerie. Díky ní uživatel nemusí opětovně nahrávat stejný obrázek, pokud ho již do systému nahrál dříve. V tomto případě existující obrázek může byt vybrán z galerie. Pokud uživatel chce do systému přidat nový obrázek, galerie nabízí možnost procházet lokální soubory a nahrát obrázek z počítače. Oba systémy otevírají galerii v modálním oknu při editaci článku, viz obrázky 2.11 a 2.12, ale je ji možné otevřít i samostatně nezávisle na článku.



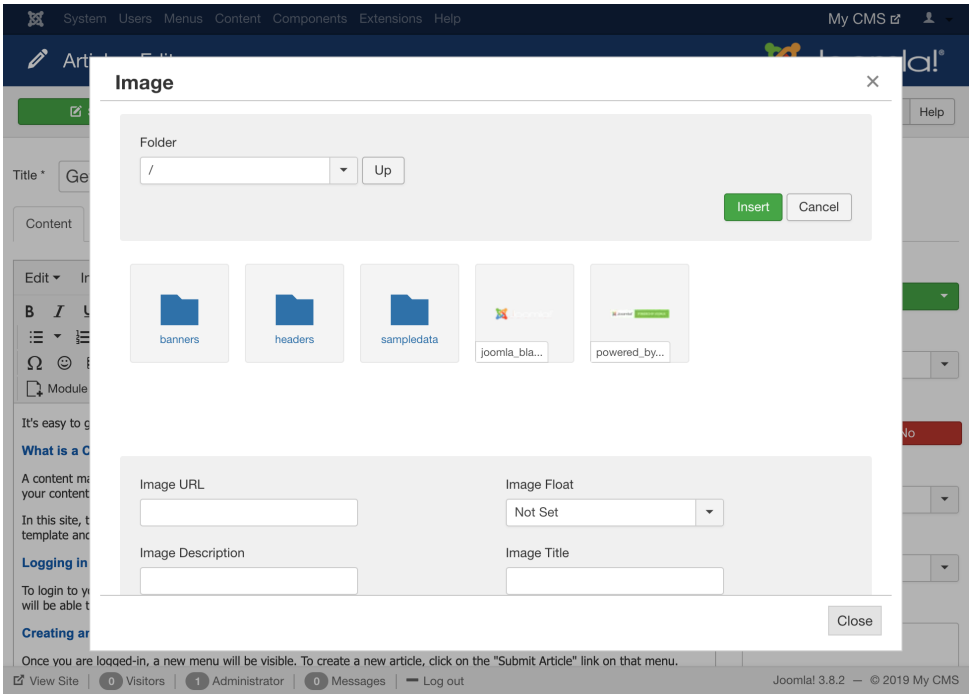
Obrázek 2.9: Editace položky v CMS Wordpress



Obrázek 2.10: Editace položky v CMS Joomla!



Obrázek 2.11: Galerie obrázků v CMS Wordpress



Obrázek 2.12: Galerie obrázků v CMS Joomla!

2.7.2 Shrnutí

Analýza podobných řešení nám přiblížila představu o vzhledu webové aplikace, kterou budeme implementovat. Shrneme odvozené závěry:

- **Ovládací panel** — Po přihlášení se uživateli zobrazí ovládací panel s nabídkou hlavních akcí.
- **Navigace** — Vzhledem k tomu, že naše aplikace nebude moc náročná na navigaci, zvolíme dynamičtější verzi vertikální nabídky. Navíc do ní přidáme možnost se z aplikace odhlásit.
- **Barvy** — Zvolíme tmavší barvy pro nabídku a světlejší pro pozadí hlavní plochy. Použijeme odstíny modré a šedé. Výraznější barvy budou pouze u akčních tlačítek.
- **Zobrazení seznamu položek** — Seznam položek zobrazíme ve tvaru tabulky. Na editaci položky se uživatel dostane jak kliknutím na název položky, tak i akčním tlačítkem.
- **Editace položky** — Aplikace upozorní uživatele při pokusu o navigaci z editační obrazovky.
- **Galerie obrázků** — Aplikace při požadavku na vložení obrázku zobrazí uživateli galerii v modálním oknu. Galerie bude taky dostupná k prohlížení mimo editaci položky.

Kapitola 3
Vývoj

3.1 REST API systému EduARd

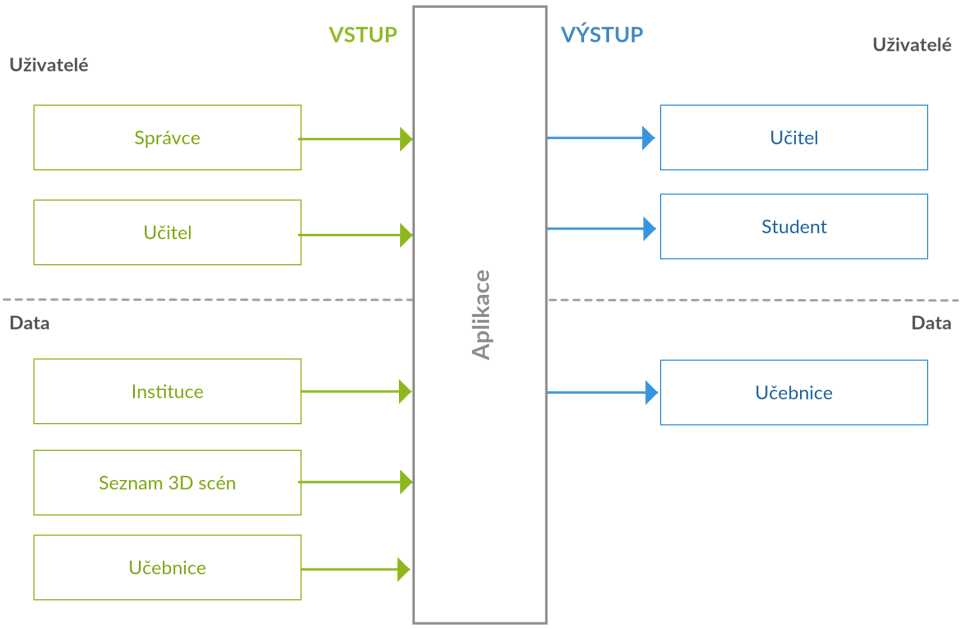
Klientské aplikace komunikují s serverovou částí pomocí již zmíněného REST API. REST je zkratkou pro Representational State Transfer[5]. Je to koncept komunikace mezi klientem a serverem, kdy se klient dotazuje na server pomocí předem definovaných identifikátorů zdrojů dat. V informatice se takové identifikátory označují zkratkou URI (anglicky „Uniform Resource Identifier“, česky „Jednotný Identifikátor Zdroje“)[6]. Každý identifikátor slouží pro adresaci zdroje dat a umožňuje provádět nad nimi jednu nebo více z CRUD (CREATE, READ, UPDATE, DELETE) operací. Server tak může od klienta přijmout požadavek na vložení, čtení, aktualizaci nebo mazání dat. Pro komunikaci se využívá HTTP protokol, a každé CRUD operaci odpovídá jedna z HTTP metod, viz tabulka 3.1.

CRUD operace	HTTP metoda
CREATE	POST
READ	GET
UPDATE	PUT
DELETE	DELETE

Tabulka 3.1: CRUD operace a odpovídající HTTP metody

Zkratka API znamená Application Programming Interface[7] a označuje soubor metod a tříd sloužících pro vykonání klientských požadavků. Může se jednat o vykonání CRUD operací nad databází anebo o funkční požadavek, např. odeslání e-mailů.

Naše aplikace bude využívat REST API systému EduARd jak pro čtení, tak i pro zápis (zápisem rozumíme vložení, aktualizace a mazání) dat. Pro lepší pochopení toho, jaká data bude naše aplikace číst a jaká zapisovat, slouží obrázek 3.1. Jelikož v aplikaci implementujeme 2 funkcionality: správu uživatelů a správu dat, jsou na obrázku vstupní a výstupní data rozdělená do dvou odpovídajících kategorií. Zda jsou data vstupní (určena ke čtení) nebo výstupní (určena pro zápis), je dáno směrem šipky.



Obrázek 3.1: Vstupní a výstupní data aplikace

REST API systému EduARd nabízí celou řadu identifikátorů pro přístup k datům. Naši webovou aplikaci pro správu uživatelů a výukových úloh budou zajímat jen některé z nich.

Autorizace. Prvním a základním požadavkem na naši webovou aplikaci je možnost se do ní přihlásit. Proto slouží identifikátory, spojené s uživatelskou autorizací.

`POST /api/Auth/SignIn` — Přihlášení do aplikace s parametry `username` (přihlašovací jméno) a `password` (heslo). Po přihlášení se uživateli přidělí tzv. `accessToken`, který se musí posílat s každým požadavkem na REST API. To zajistí bezpečnou komunikaci mezi klientem a serverem. Kromě `accessToken` uživatel dostane `refreshToken`, o kterém si povíme v dalším odstavci.

`POST /api/Auth/Refresh` — Přidělený `accessToken` má omezenou platnost, která po určité stanovené době vyprší. Pro obnovení tokenu se musí uživatel odhlásit a znovu přihlásit, což není při použití webové aplikace příjemné. Abychom se tomuto vyhnuli, budeme token obnovovat odesláním požadavku POST na uvedený identifikátor s parametry `accessToken` a `refreshToken`.

Správa uživatelů instituce. Požadavky na tyto identifikátory bude vykonávat uživatel, který má roli `InstitutionAdmin`, tj. správce instituce.

`GET /api/Institutions/current` — Vrací informace o instituci, do níž spadá aktuálně přihlášený správce. Server ve své odpovědi vrátí mimo jiné hodnoty `members`, který reprezentuje seznam uživatelů instituce, a `invitations`, který udává seznam odeslaných pozvánek pro nové uživatele. Každý uživatel je definován e-mailem, který je zároveň jeho unikátním identifikátorem a slouží jako přihlašovací jméno do příslušné klientské aplikace, a rolí. Pozvánka

má svůj unikátní identifikátor, e-mail, na který byla odeslána, a roli, která má být pozvanému uživateli přidělena.

`POST /api/Institutions/Invite` — Slouží k založení účtů pro nové uživatelé v rámci instituce. Jako parametry přijímá hodnoty `email` (e-mailovou adresu uživatele) a `roles` (seznam rolí, které mu mají být přiděleny).

`POST /api/Institutions/Manage` — Využijeme pro editaci uživatelských rolí. Jako parametry odešleme hodnoty `username` (přihlašovací jméno uživatele, tudíž jeho e-mailovou adresu) a `roles` (role, které mu mají být přiděleny)

`DELETE /api/Institutions/{email}` — Slouží pro odstranění uživatele z instituce. Uživatel je jednoznačně identifikován svým uživatelským jménem. Uživatelské jméno uživatele, jehož chceme z instituce odstranit, pošleme v parametru `email`.

`DELETE /api/Institutions/invite/{id}` — Pokud odešleme chybnou pozvánku a chceme ji zrušit, odešleme požadavek DELETE na uvedený identifikátor, kde `id` značí unikátní identifikátor pozvánky, jíž chceme odstranit.

Správa výukových úloh. V této části webové aplikace budeme přistupovat k více zdrojům REST API, než ve správě uživatelů. Bude nás zajímat způsob odeslání požadavků na správu učebnic, stahování a nahrávání obrázků a získání informací o dostupných 3D scénách. Provádění těchto akcí bude umožněno uživateli, který bude mít roli `Editor`.

`GET /api/Books` — Vrací seznam všech učebnic, dostupných v rámci dané instituce.

`GET /api/Books/{id}` — Vrací jednu učebnici, jejíž identifikátor odpovídá zadanému parametru `id`.

`POST /api/Books` — Slouží pro přidání nové učebnice. Vyžaduje parametr `name` - název učebnice.

`POST /api/Books/Upload` — Slouží pro nahrání příloh pro danou učebnici. Může se jednat jak o obrázek, tak i o soubor s popisem výukových úloh, o kterém si povíme podrobněji v sekci 3.5.

`PUT /api/Books/{id}` — Využijeme pro editaci učebnice s identifikátorem odpovídajícímu zadanému parametru `id`. Přijímá parametry `name` - název učebnice, `description` - popisek učebnice, `numberOfAttempts` - počet pokusů na splnění výukových úloh, `numberOfTasks` - počet výukových úloh.

`DELETE /api/Books/{id}` — Slouží pro smazání učebnice, jejíž identifikátor odpovídá zadanému parametru `id`.

`GET /api/Scenarios` — Vrací seznam 3D scén dostupných pro danou instituci. V naší aplikaci budeme potřebovat pouze názvy těchto scén.

`GET /api/Assets/{bookId}` — Vrací seznam všech příloh dostupných pro učebnici, jejíž identifikátor odpovídá zadanému parametru `bookId`.

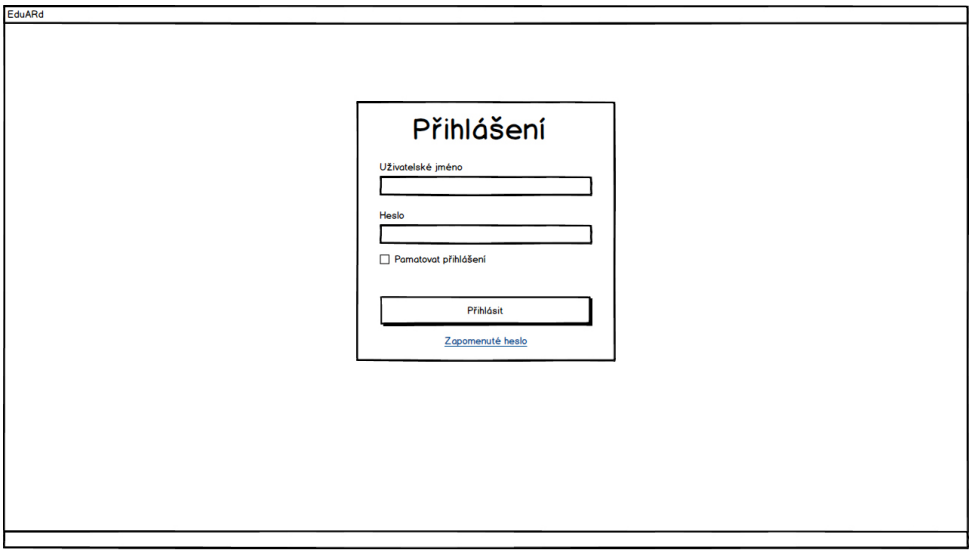
`GET /api/Assets/Download/{fileName}` — Umožní stáhnout přílohu, jejíž unikátní název odpovídá zadanému parametru `fileName`.

3.2 Návrh uživatelského rozhraní

3.2.1 Základní návrh

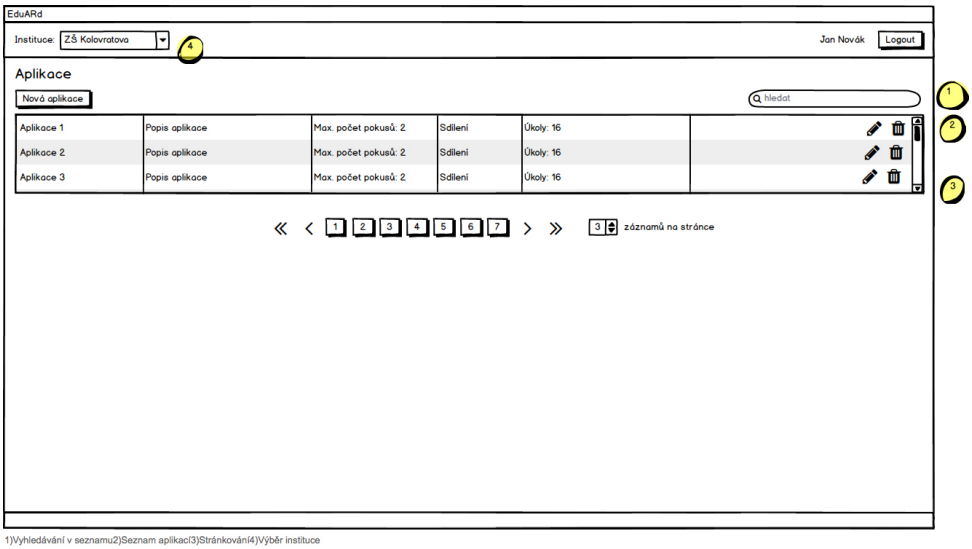
Od zadavatelů byl poskytnut základní návrh aplikace, který zachycoval vzhled 4 stránek:

- **Přihlašovací stránka** — Viz obrázek 3.2.



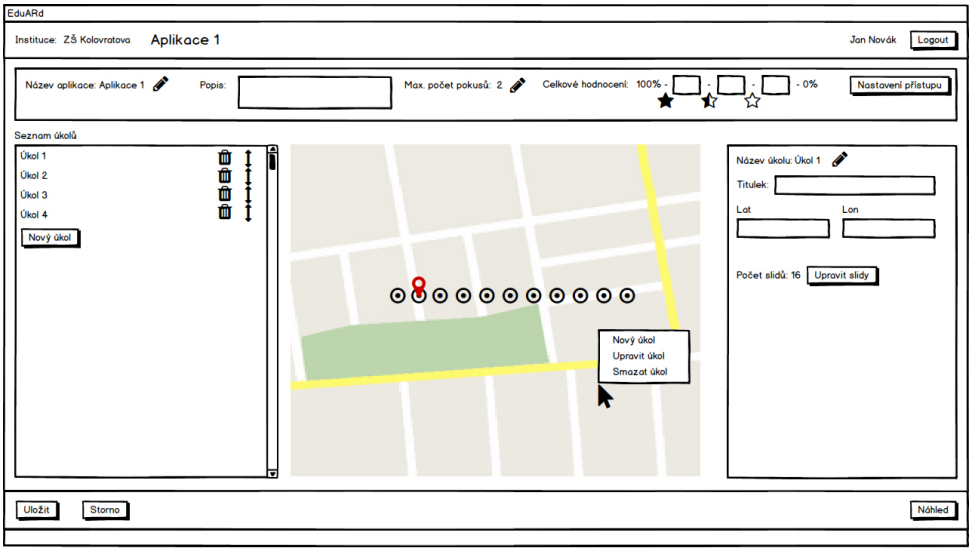
Obrázek 3.2: Základní návrh - Přihlašovací stránka

- **Seznam učebnic** — Viz obrázek 3.3.



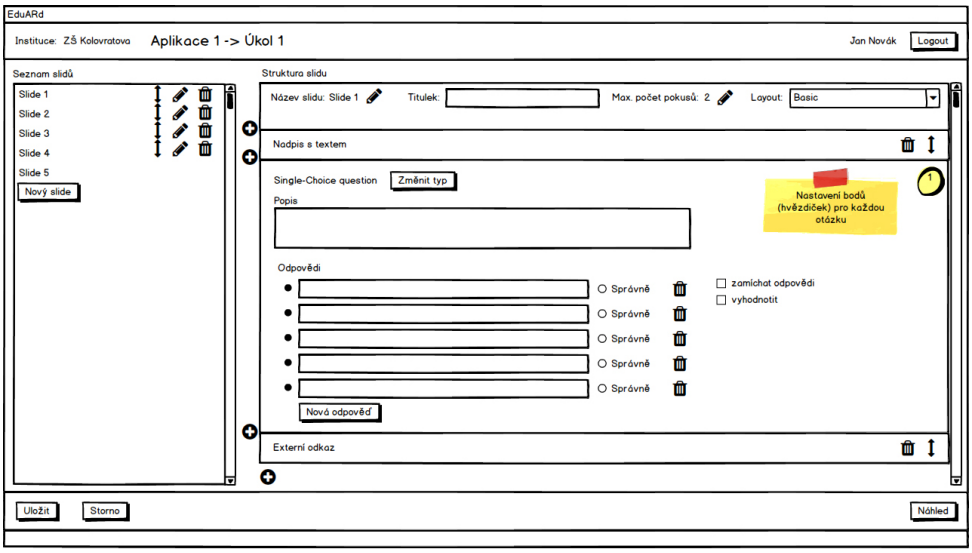
Obrázek 3.3: Základní návrh - Seznam učebnic

- **Detail učebnice** — Obsahuje informace o učebnici a seznam úkolů. Viz obrázek 3.4.



Obrázek 3.4: Základní návrh - Detail učebnice

- **Seznam obrazovek úkolu** — Obsahuje seznam obrazovek ve vybraném úkolu. Viz obrázek 3.5.



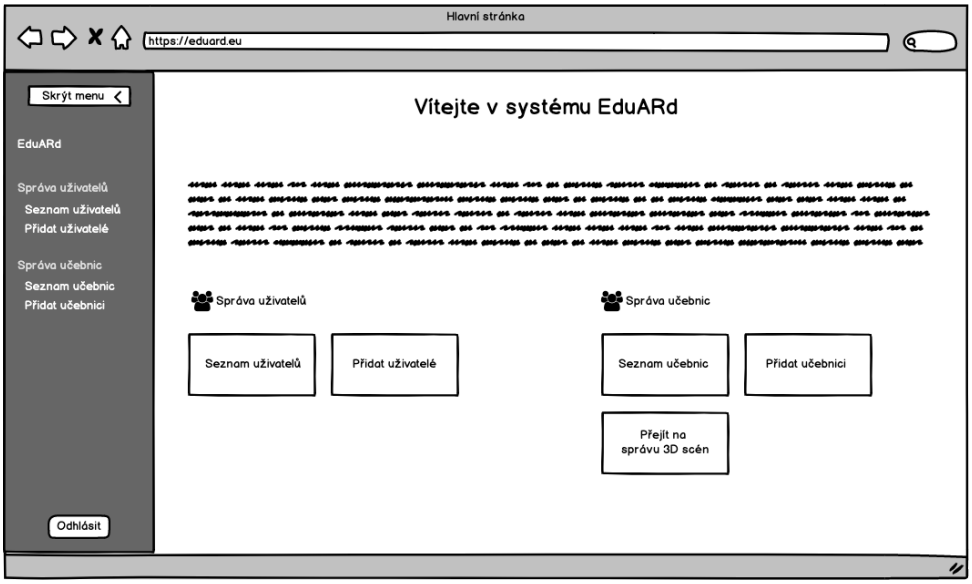
Obrázek 3.5: Základní návrh - Seznam obrazovek úkolu

Je nutné si poznamenat, že v základním návrhu se používá jiné názvosloví. Aby se předešlo zmatku, uvedeme, jak se jednotlivé názvy změnily:

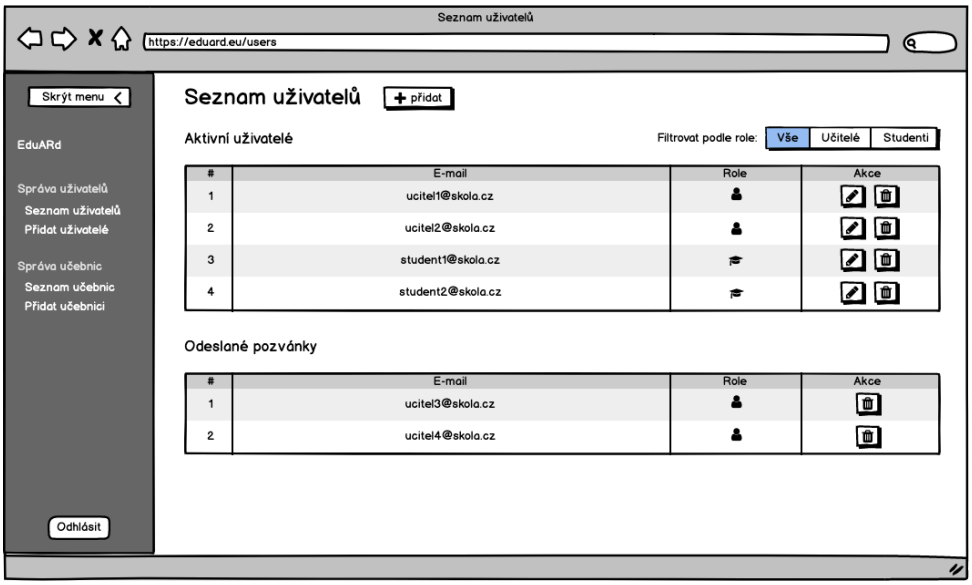
- **Aplikace** — Nově **učebnice**.
- **Slide** — Nově **obrazovka**.

3.2.2 Doplnění návrhu

Jedním z bodů zadání této práce bylo doplnit základní návrh aplikace, a to o stránku zachycující správu uživatelů. Návrh se implementoval v nástroji Balsamiq Cloud[8], výsledek lze vidět na obrázku 3.7. Jelikož naše aplikace bude plnit 2 účely, smysl má přidat také i hlavní stránku aplikace, která by uživatele uvítala v systému a sloužila mu jako rozcestník. Návrh hlavní stránky je znázorněn na obrázku 3.6. Do obou návrhu byla přidána i hlavní nabídka aplikace pro rychlou navigaci mezi části systému.



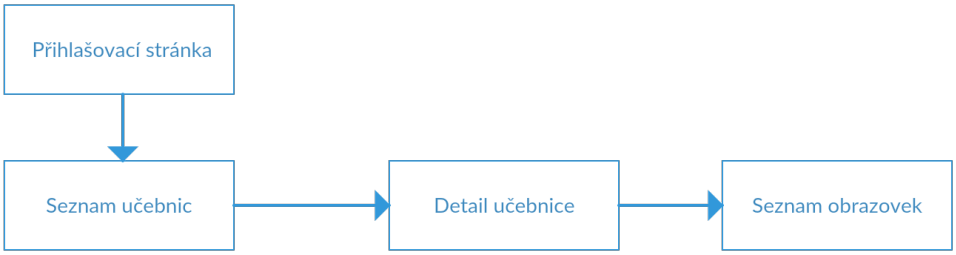
Obrázek 3.6: Doplnění návrhu - Hlavní stránka aplikace a navigace



Obrázek 3.7: Doplnění návrhu - Seznam uživatelů a navigace

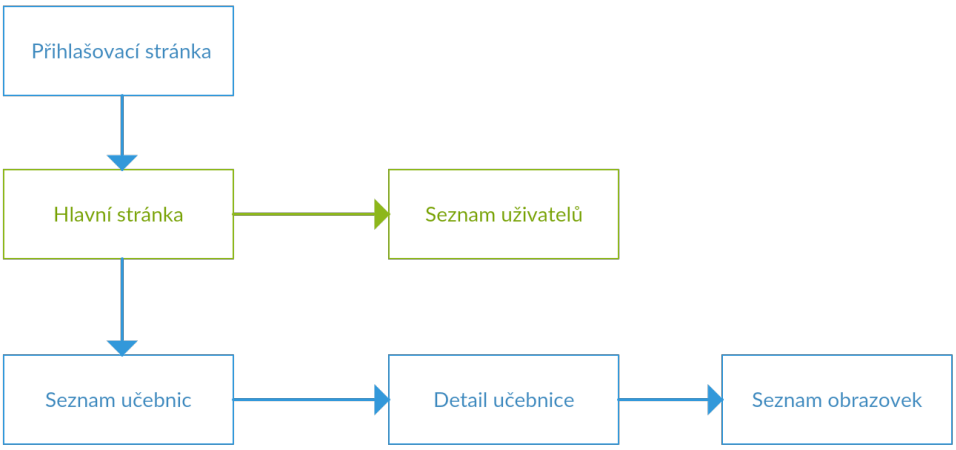
3.2.3 Struktura aplikace

Ze základních návrhů vyplývala základní struktura aplikace, která je znázorněna na obrázku 3.8.



Obrázek 3.8: Základní struktura aplikace podle poskytnutých návrhů

Po doplnění základního návrhu se struktura obohatila o další dva prvky, viz obrázek 3.9. Podle této struktury a doplněných návrhů můžeme začít implementovat aplikaci.



Obrázek 3.9: Finální struktura aplikace

3.3 Použité technologie a knihovny

3.3.1 React.js

Jedná se o knihovnu vytvořenou společnosti Facebook[9] v jazyce Javascript[10]. Tento jazyk se používá převážně na webových stránkách[10], kde slouží hlavně pro manipulaci se její strukturou, neboli s uživatelským DOM[11]. V naší aplikaci se používá verze Javascriptu z roku 2015 zvaná ECMAScript 6 (ve zkratce ES6). Knihovna React.js je určena pro tvorbu uživatelských rozhraní, v nichž se data neustále mění. Paradigma React.js spočívá v rozdělení webové stránky na komponenty, které na základě svého **stavu** generují příslušný **DOM**. V roce 2013, kdy společnost Facebook zveřejnila knihovnu pro volné použití, odezva na React.js byla velmi špatná. Bylo tomu tak právě kvůli

míchání DOM a programování dohromady. Avšak časem začala knihovna získávat čím dál větší oblibu a dneska je jedním z nejpoužívanějších nástrojů pro tvorbu uživatelských rozhraní webových stránek. Knihovna má velké množství přispěvatelů, kteří tvoří různé předpřipravené komponenty k volnému použití. Tato skutečnost se stala jedním z mnoha důvodů, proč pro naši aplikaci byl zvolen právě React.js a ne jeho alternativa Angular od společnosti Google[12].

3.3.2 SASS

Vzhled webové stránky je definován nejen její strukturou, ale taky tím, jak jednotlivé prvky té struktury vypadají. Pro přizpůsobení vzhledu DOM prvků se využívá kaskádových stylů CSS[13]. V naší aplikaci využíváme preprocesor CSS zvaný SASS[14]. Jedná se o kompilovaný jazyk, rozšiřující syntaxi CSS o proměnné, funkce, podmínky a další užitečné věcí, které vývojáři ušetří čas při stylizaci webové stránky.

3.3.3 React Bootstrap

Jedná se o verzi knihovny Boostrap[15] adaptovanou speciálně pro použití v React.js. Knihovna Bootstrap se vyvinula na webovém portálu Twitter.com a sloužila pro sjednocení vzhledu jednotlivých komponent napříč různými systémy. Bootstrap obsahuje sadu kaskádových stylů upravujících typografii, barvy a velikosti často používaných prvků webových stránek. Její verze pro React.js obsahuje navíc předpřipravené komponenty pro tlačítka, modální okna, formuláře a další. Pomoci knihovny Bootstrap je také velmi jednoduché přizpůsobit webovou stránku pro různě velká rozlišení monitorů. Nastavení knihovny (výchozí barvy, písma, pixelové zalamování kontejneru stránky a jiné) nastavíme v naší aplikaci pomocí SASS proměnných.

3.3.4 Yarn

V aplikaci chceme využívat různé předpřipravené komponenty a knihovny třetích stran. Stahovat a instalovat tyto komponenty a knihovny do naší aplikace nám umožní systém Yarn[16]. Tento systém spravuje knihovny třetích stran ve formě balíčků, o kterých pak máme v aplikaci přehled a můžeme je lehce aktualizovat či odstranit. Yarn nahrazuje svého staršího předchůdce npm[17], který byl značně pomalejší a poskytoval menší zabezpečení.

3.3.5 Webpack

Technologii Webpack využijeme v naší aplikaci pro její přípravu ke spuštění v prohlížeči. Musíme totiž importované balíčky, stejně jako kód psaný v nové verzi Javascriptu a styly psané v SASS, zkompilovat i s ohledem na to, že některé uživatelé můžou používat starší verze prohlížeče. Webpack zkompiluje zdrojové soubory naší aplikaci do několika souborů s koncovkami `.js` a `.css`, případně k ním přidá použité obrázky. Tyto soubory už jsou „srozumitelné“ pro všechny verze prohlížečů a stránka je připravená k zobrazení.

3.3.6 Font Awesome

Font Awesome[18] je řešení pro vložení vektorových ikon do webové stránky. Jak napovídá název, ikony jsou vkládány jako písmo. Lze u nich libovolně měnit velikost a barvy stejně jako u jakéhokoliv textu a budou vypadat dobře i při velkém rozlišení monitoru. V naší aplikaci používáme verzi knihovny upravenou speciálně pro React.js. To znamená, že každá ikona se vkládá jako komponenta.

3.3.7 Mapy.cz API

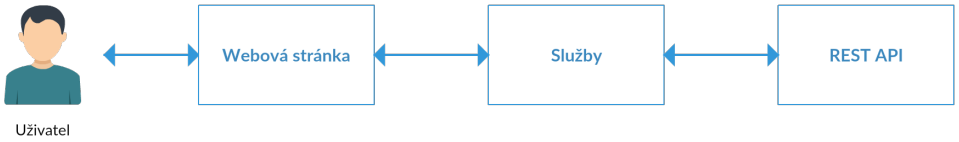
Jednotlivé výukové úlohy, jak již bylo popsáno v kapitolách výše, se vážou na GPS. Tato skutečnost a taky základní návrh napovídají, že v naší aplikaci musíme použít nějaké dostupné řešení pro zadání pozice na mapě. Nejpoužívanější mapovou službou je dneska služba Google Maps[19], která se ale od 16. července 2018 stala placenou[20]. I když nabízí určitý počet načtení mapy zdarma, požaduje údaje platební karty pro vystavení účtu za případné překročení limitu. V rámci bakalářské práce se nám víc hodí použit službu Mapy.cz API[21], která nabízí stejnou funkcionalitu a je zdarma i pro komerční účely.

3.4 Komunikace se serverem

Funkčnost naší aplikace je podmíněná komunikací se serverovou částí systému EduARd, tj. s REST API, která byla popsána v sekci 3.1. Se serverovou částí se komunikuje pomocí požadavků na předem definované unikátní identifikátory zdrojů. Odesílat požadavky a přijímat odpovědi ze serveru nám umožní funkce `fetch`[22]. Je vhodné implementovat jednotlivé dotazy na REST API mimo React komponenty generující prezentační vrstvu aplikace a mít je setříděné podle účelu použití. K tomu v aplikaci zavedeme následující služby:

- **AuthService.js** — Odesílá požadavky na zdroje, jejichž identifikátor začíná `/api/Auth/`. Odeslané požadavky řeší přihlášení uživatelů.
- **UserService.js** — Odesílá požadavky na zdroje, jejichž identifikátor začíná `/api/Institutions/`. Odeslané požadavky řeší správu uživatelů instituce.
- **BookService.js** — Odesílá požadavky na zdroje, jejichž identifikátor začíná `/api/Books/`. Odeslané požadavky řeší správu učebnic instituce.
- **AssetsService.js** — Odesílá požadavky na zdroje, jejichž identifikátor začíná `/api/Assets/`. Odeslané požadavky řeší správu obrázků učebnice.
- **ScenarioService.js** — Odesílá požadavky na zdroje, jejichž identifikátor začíná `/api/Scenarios/`. V aplikaci odešleme pouze jeden požadavek, a to na seznam dostupných 3D scén.

Komunikace mezi prezentační vrstvou a REST API pak probíhá přes tyto služby a je znázorněna na obrázku 3.10.



Obrázek 3.10: Komunikace mezi prezentační vrstvou a REST API přes služby

3.5 Parsování učebnice

V sekci 2.4 jsme uvedli, že se výukové úlohy seskupují do učebnic a obsahují obrazovky buď s 2D nebo 3D obsahem. Ze sekce 3.1 víme, že se REST API můžeme dotázat na učebnici. Výukové úlohy a jejích obsah je pak v každé učebnici reprezentován ve formátu XML. Zkratka XML znamená eXtensible Markup Language, v překladu do češtiny rozšiřitelný značkovací jazyk. Jedná se o standardní formát pro výměnu dat. Formát XML byl zvolen tvůrci systému EduARd, proto se nebudeme zabývat tím, jak formát vznikl, jaké jsou jeho přednosti a jeho syntaxí. Rozebereme pouze to, jak s ním v naší aplikaci budeme zacházet.

Formát XML není moc vhodný pro použití v aplikaci, která svá data neustále mění. Modifikují se nejen vlastnosti úloh a jejích obrazovek, ale taky pořadí těchto prvků v příslušných seznamech. Generovat nový XML soubor pokaždé, když uživatel provede nějakou změnu, by bylo náročné na výpočetní výkon a velmi neefektivní. Proto je potřeba tato data převést na **objekty**, které se při programování v Javascriptu běžně používají.

Definici objektu nalezneme v dokumentaci jazyku Javascript[23]. Ta definuje objekt jako kolekci vlastností, ve které každá vlastnost je reprezentovaná asociací `"klíč": "hodnota"`.

3.5.1 Načtení učebnice

Pro načtení učebnice se využívá příslušná metoda REST API popsána v sekci 3.1. Učebnice, kterou z REST API dostaneme, je reprezentovaná objektem, který obsahuje některé vlastnosti učebnice:

- `id` — Unikátní identifikátor.
- `name` — Název učebnice.
- `numberOfAttempts` —Počet pokusů na splnění úkolů.
- `numberOfTasks` — Počet úkolů.
- `description` — Popis učebnice.

- `resourcePath` — Cesta ke stažení XML reprezentace učebnice.

Příklad objektu učebnice získaného z REST API:

```
{
    "id": 1,
    "name": "Biology",
    "numberOfAttempts": 3,
    "numberOfTasks": 3,
    "description": "The book describes plants and animals.",
    "resourcePath": "/api/books/download/XML_book_id_1.xml"
}
```

Další vlastností učebnice, jako je například rozložení úkolů, jsou k nalezení v XML souboru, který můžeme stáhnout z cesty uvedené v `resourcePath`.

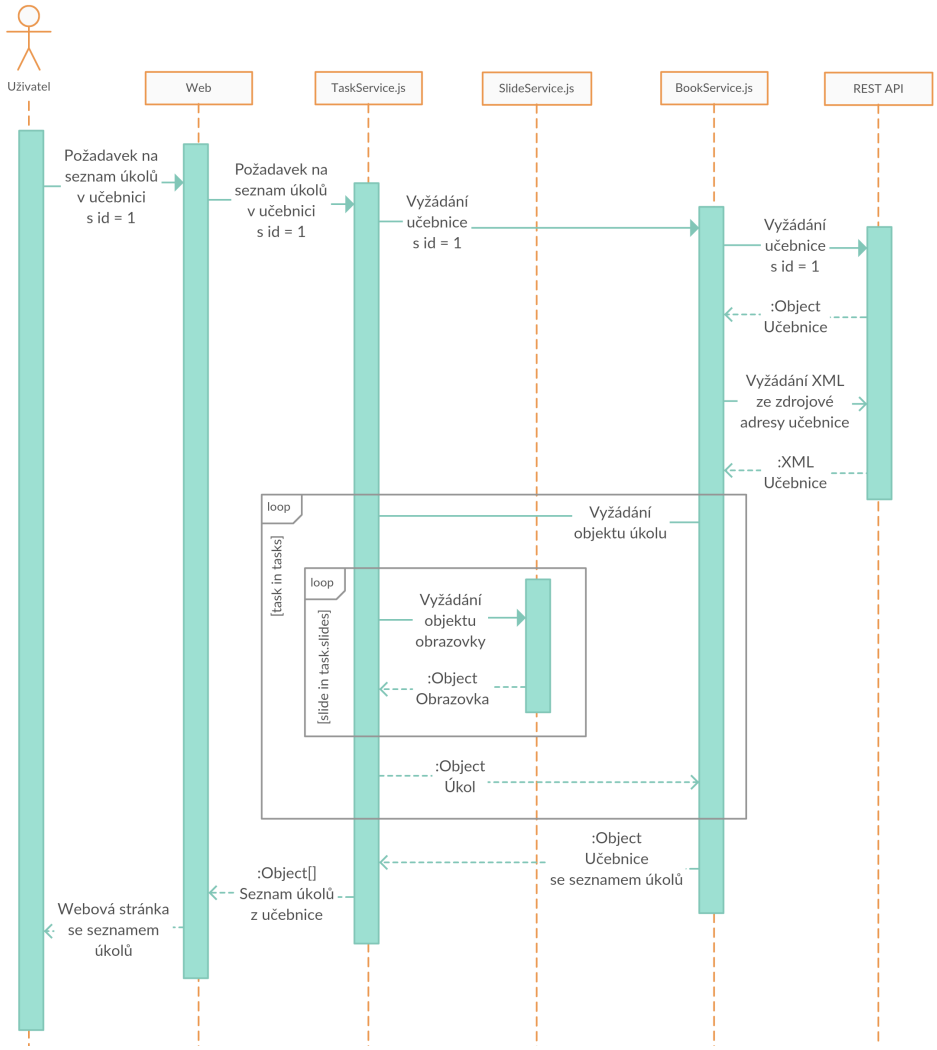
Řešíme tedy problém, který nastane ve chvíli, kdy se uživatel dotáže na seznam úkolů pro vybranou učebnici. Znamená to, že aplikace musí nejdřív získat objekt učebnice a následně stáhnout její XML. Z tohoto XML se musí vytáhnout seznam úkolů a pro každý úkol sestavit seznam obrazovek včetně jejích obsahu. Pro vytvoření objektu z XML zavedeme další 3 služby:

- **TaskService.js** — Poskytuje metody pro získání seznamu úkolů vybrané učebnice a konverzi XML reprezentaci úkolu na objekt.
- **SlideService.js** — Poskytuje metody pro získání seznamu obrazovek vybraného úkolu a konverzi XML reprezentaci obrazovky na objekt.
- **QuestionService.js** — Poskytuje metody pro konverzi XML reprezentaci otázky na objekt.

Všimneme si, že je potřeba zadefinovat objekty reprezentující úkol, obrazovku a otázku. Využijeme možnost zadefinovat třídy, která je nově dostupna v Javascript ES6. Třídy budou mít atributy odpovídající vlastnostem popsaným v sekci 2.4.

Uživatel zašle požadavek na seznam úkolů webové stránce, tedy na prezentační vrstvu aplikace, která tento požadavek přepošle na službu **TaskService.js**. Ta se dotáže služby **BookService.js** na učebnici. Služba BookService.js získá z REST API požadovanou učebnici s odkazem na XML soubor popisující výukové úlohy a následně tento soubor stáhne. Ve staženém souboru služba BookService.js najde tag označující seznam úkolů učebnice a pro každou XML reprezentaci úkolu se dotáže služby TaskService.js na konverzi tohoto úkoli na objekt. Při konverzi úkolu služba TaskService.js volá službu **SlideService.js**, která zajistí převod XML reprezentaci každé obrazovky úkolu na objekt. Po dokončení konverzi TaskService.js vrátí službě BookService.js seznam úkolů včetně jejích obrazovek, který se přiřadí do

objektu učebnice získaného z REST API. Tento objekt je vrácen zpět službě TaskService.js, která dřív odeslala požadavek na získání učebnice. Z objektu učebnice se vytáhne seznam úkolů, který je vrácen do prezentační vrstvy a následně zobrazen uživateli. Implementace tohoto řešení je znázorněna na obrázku 3.11.



Obrázek 3.11: Sekvenční diagram zobrazení seznamu úkolů učebnice

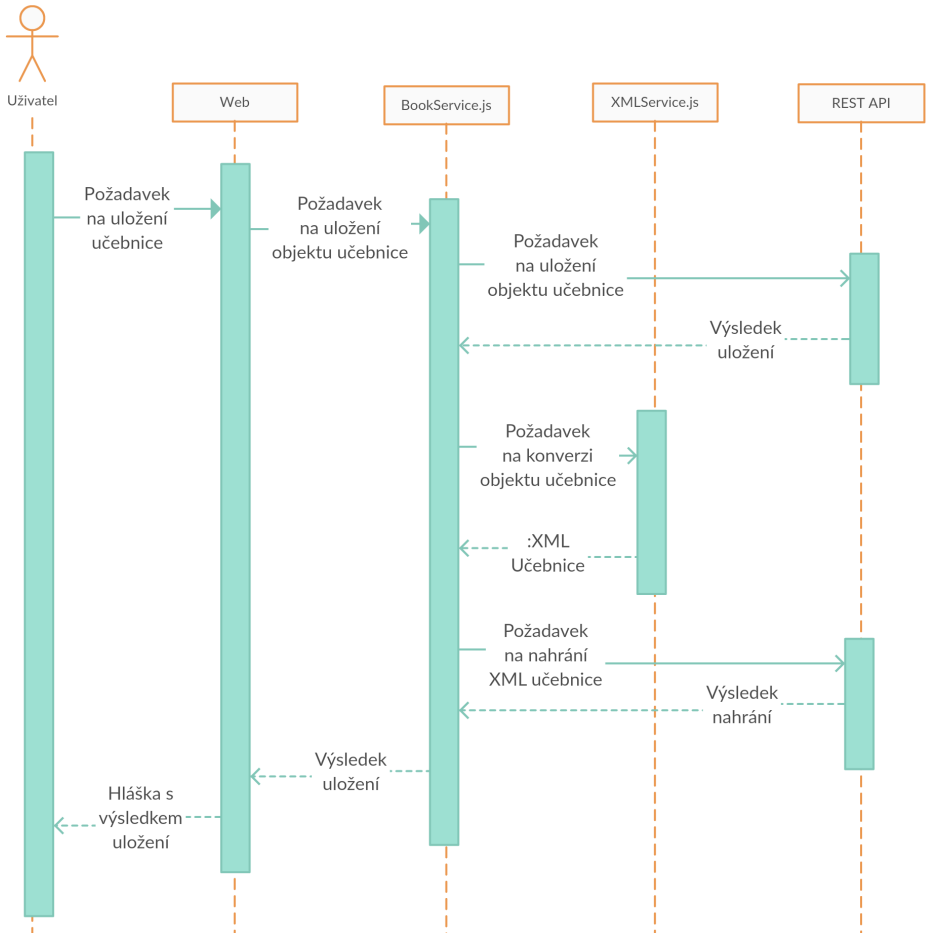
Podobným způsobem se postupuje například při požadavku na seznam obrazovek konkrétního úkolu. Využívají se stejné služby, akorát v jiném pořadí.

3.5.2 Uložení učebnice

Proces uložení učebnice je v systému rozdělen na dvě části:

- Uložení objektu učebnice.
- Nahrání aktualizované XML reprezentace učebnice.

V každé části se vykonává jeden asynchronní požadavek na REST API, což znamená, že ukládání objektu a nahrání XML běží paralelně. Na dokončení obou procesů v aplikaci vyčkáme pomocí funkce `Promise.all()`. V části, která zajišťuje nahrání XML reprezentace učebnice musíme nejdřív XML soubor sestavit. K tomu nám v aplikaci slouží služba **XMLService**, které předáme objekt učebnice obsahující seznam úkolů včetně obrazovek a jejích obsahu. Služba XMLService nám vrátí XML reprezentaci učebnice, kterou můžeme do REST API nahrát. Tento proces je znázorněn na obrázku 3.12.

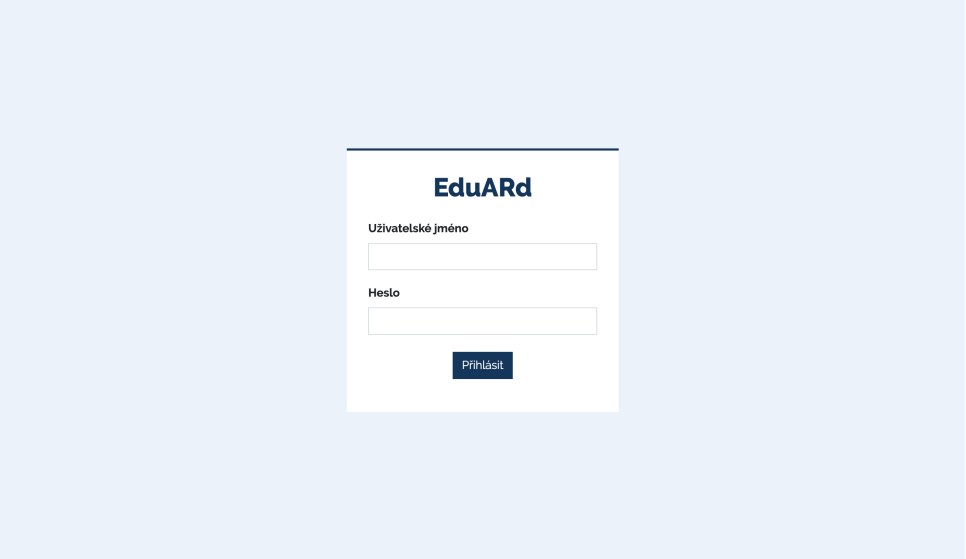


Obrázek 3.12: Sekvenční diagram uložení učebnice

3.6 Implementace uživatelského rozhraní

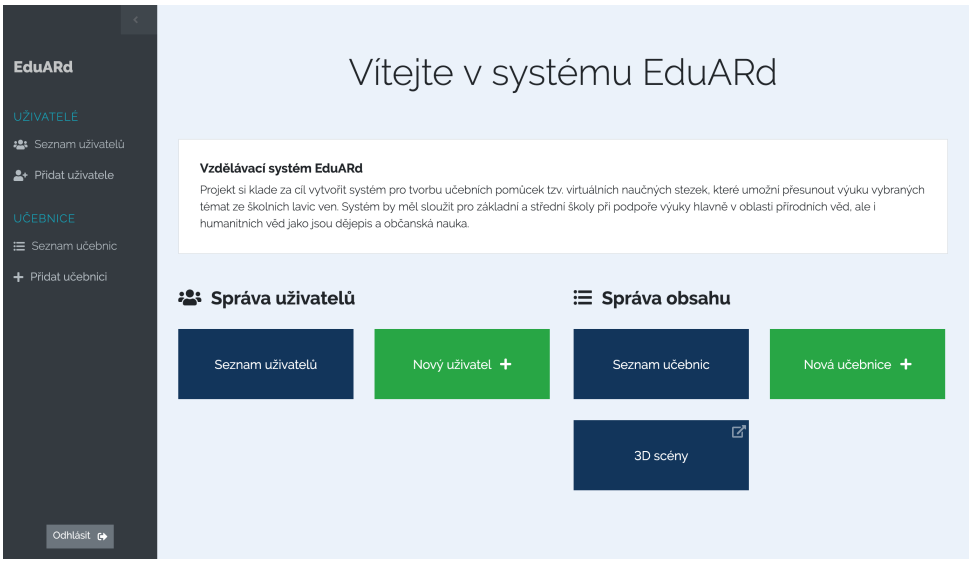
V sekci 3.2 jsme si definovali vzhled jednotlivých stránek naší webové aplikace. Podle těchto návrhů a za použití technologií popsaných v sekci 3.3 byly jednotlivé stránky implementovány. Výsledná aplikace má 6 stránek:

1. **Přihlašovací stránka** — Viz obrázek 3.13.



Obrázek 3.13: Přihlašovací stránka

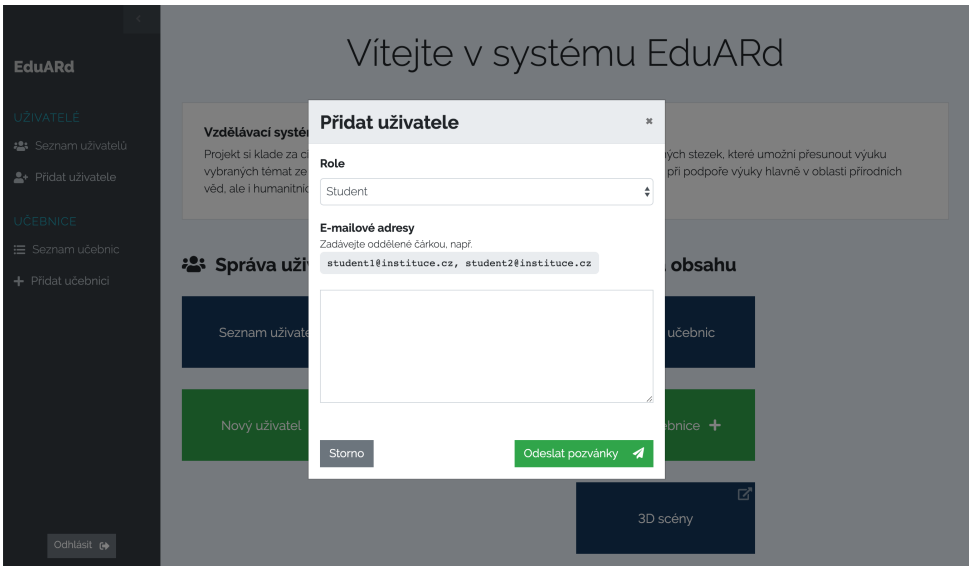
2. **Ovládací panel** — Viz obrázek 3.14. Tato stránka se uživateli zobrazí hned po přihlášení.



Obrázek 3.14: Ovládací panel

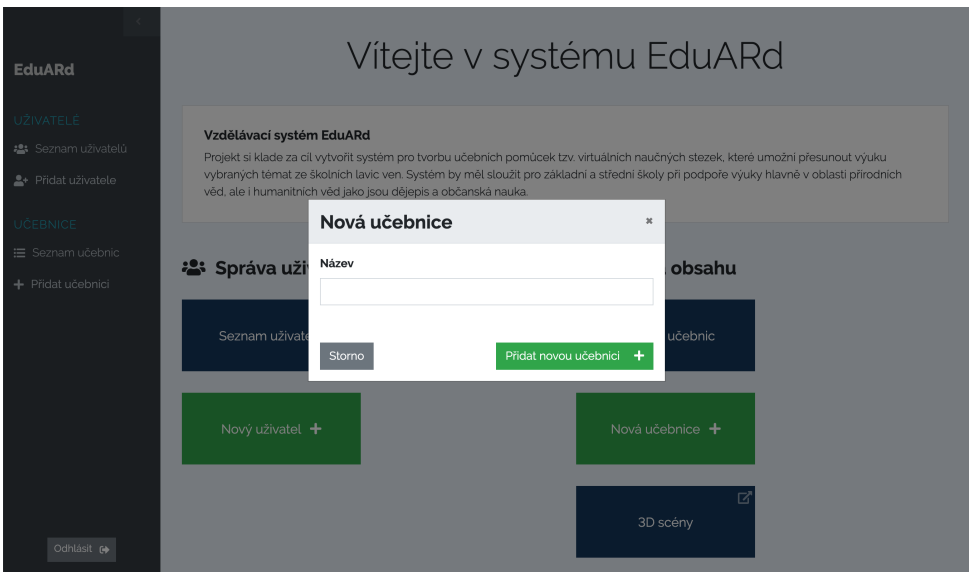
Na ovládacím panelu kromě odkazu na editor 3D obsahu jsou uživateli k dispozici tlačítka pro rychlé akce:

- **Přidání nových uživatelů** — Formulář na přidání se otevře v modálním oknu. Viz obrázek 3.15.



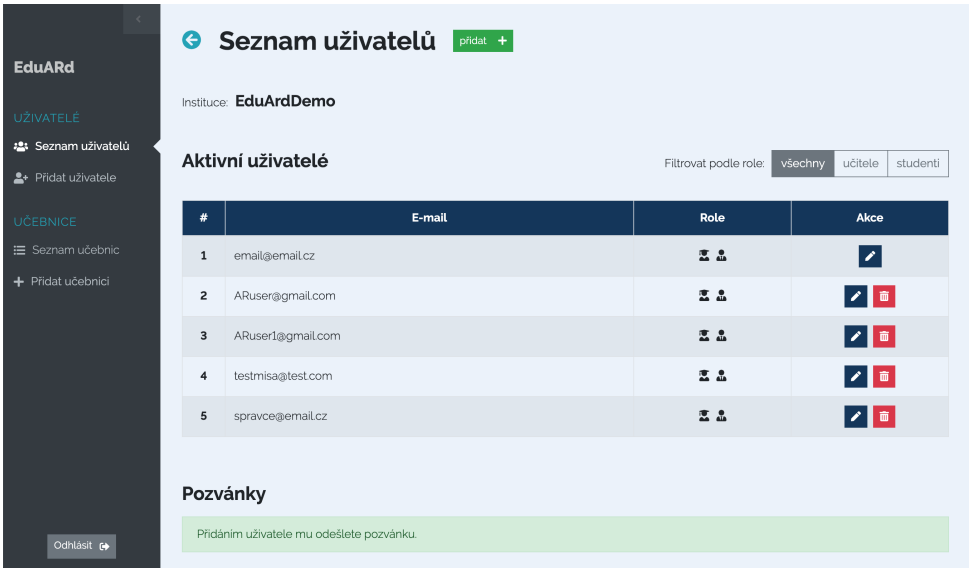
Obrázek 3.15: Modální okno pro přidání nových uživatelů

- **Přidání nové učebnice** — Formulář na přidání se otevře v modálním oknu. Viz obrázek 3.16.



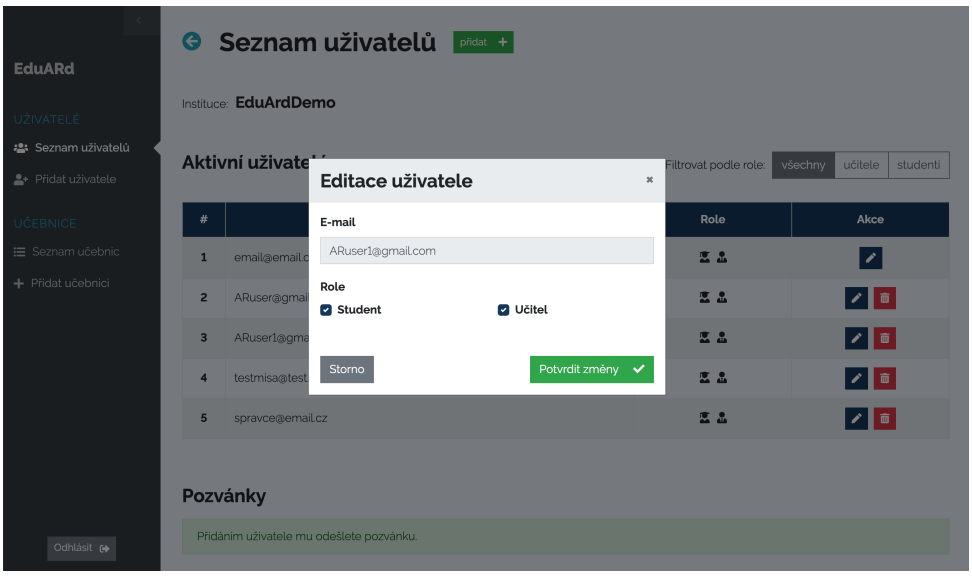
Obrázek 3.16: Modální okno pro přidání nové učebnice

3. **Seznam uživatelů** — Viz obrázek 3.17. Otevření modálního okna na přidání nových uživatelů je dostupné po kliknutí na tlačítko „přidat“.



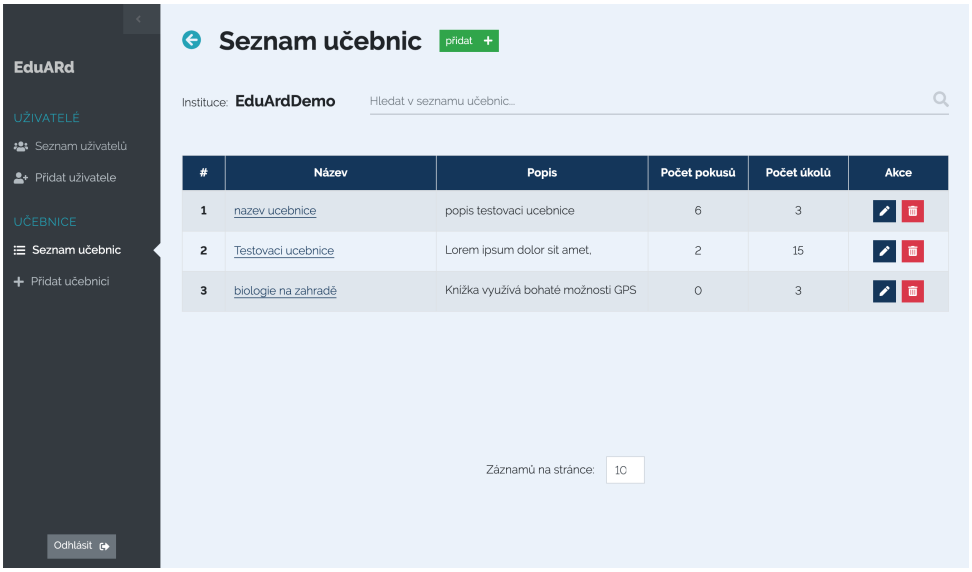
Obrázek 3.17: Seznam uživatelů

Kliknutím na tlačítko s ikonou péra se otevře modální okno pro editaci rolí uživatele, viz obrázek 3.18.



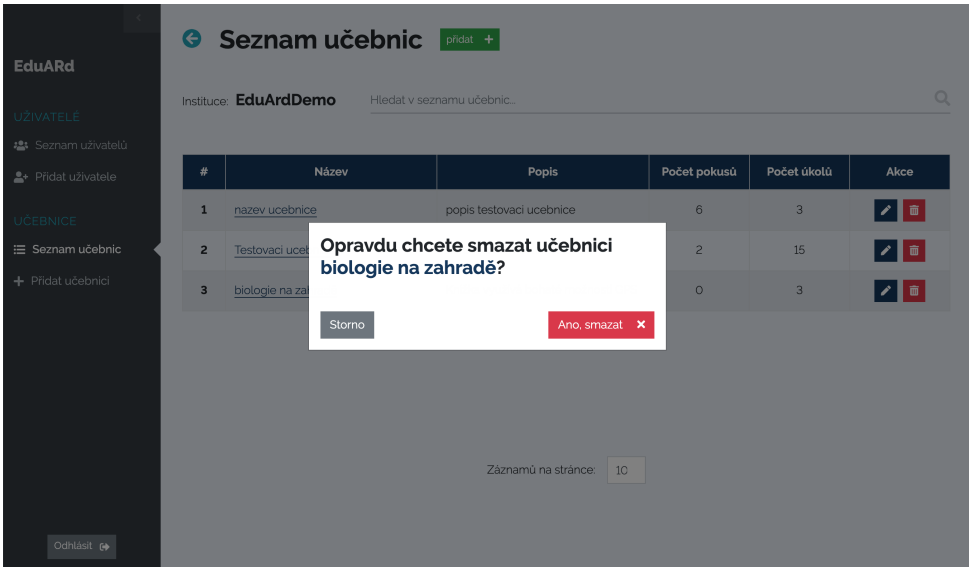
Obrázek 3.18: Seznam uživatelů

4. **Seznam učebnic** — Viz obrázek 3.19. Otevření modálního okna na přidání nové učebnice je dostupné po kliknutí na tlačítko „přidat“. Kliknutím na tlačítko s ikonou péra je uživatel přesměrován na detail učebnice.



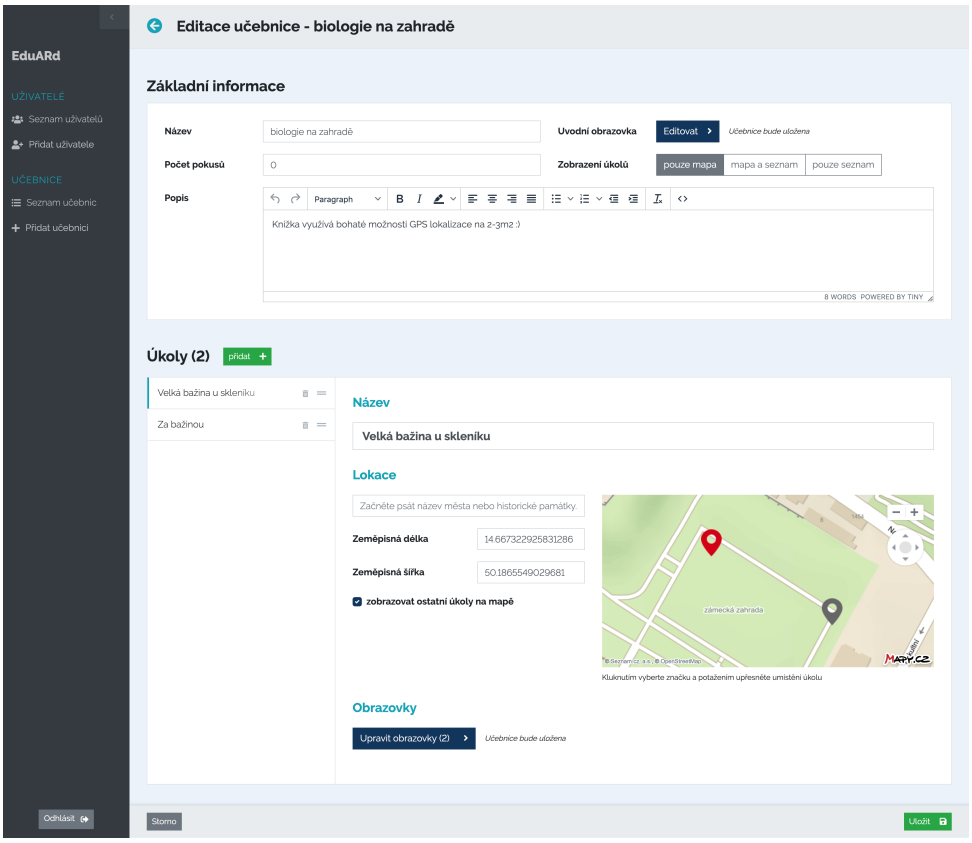
Obrázek 3.19: Seznam učebnic

Při požadavku na smazání učebnice je vyžádáno potvrzení uživatele, viz obrázek 3.20. Obdobné chování je implementováno při mazání uživatele, úkolu a obrazovky.



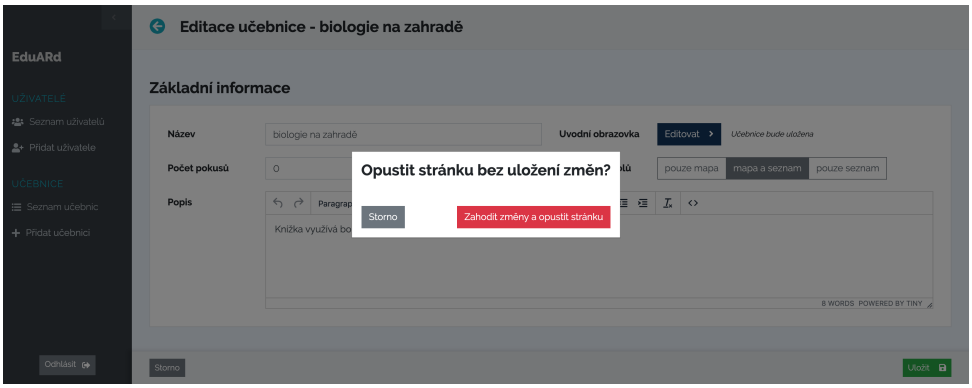
Obrázek 3.20: Potvrzení prováděné akce

5. **Detail učebnice** — Viz obrázek 3.21. Tato stránka obsahuje informací o učebnici a seznam jejích úkolů. Z každého úkolů lze pomocí tlačítka „Upravit obrazovky“ navigovat na stránku se seznamem obrazovek úkolu.



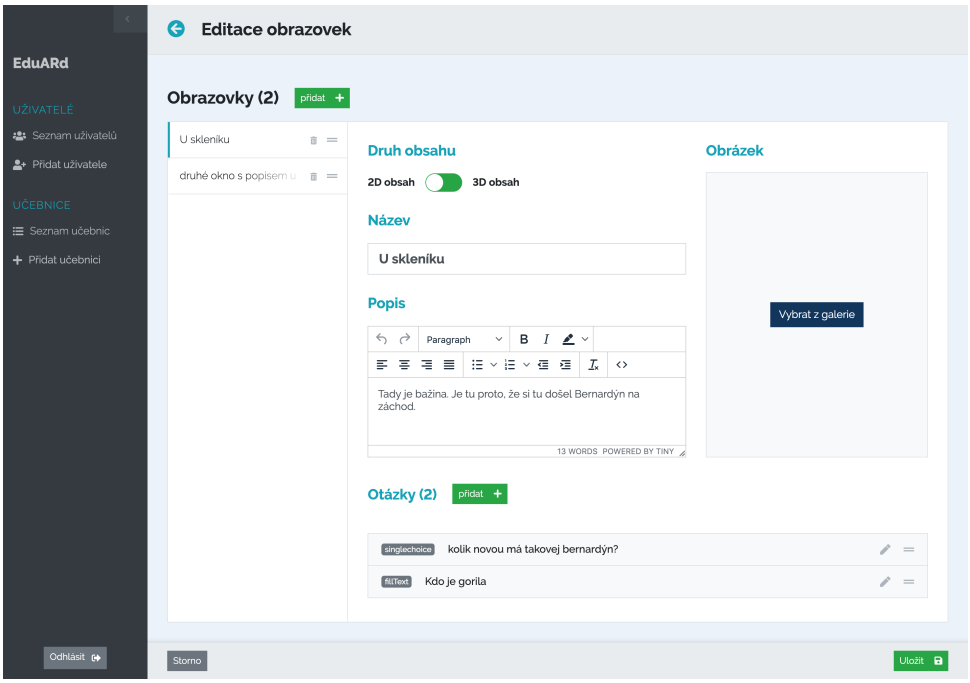
Obrázek 3.21: Detail učebnice se seznamem úkolů

- **Upozornění na neuložená data** — Obrázek 3.22 ukazuje příklad upozornění uživatele při opuštění stránky bez řádného uložení změn.



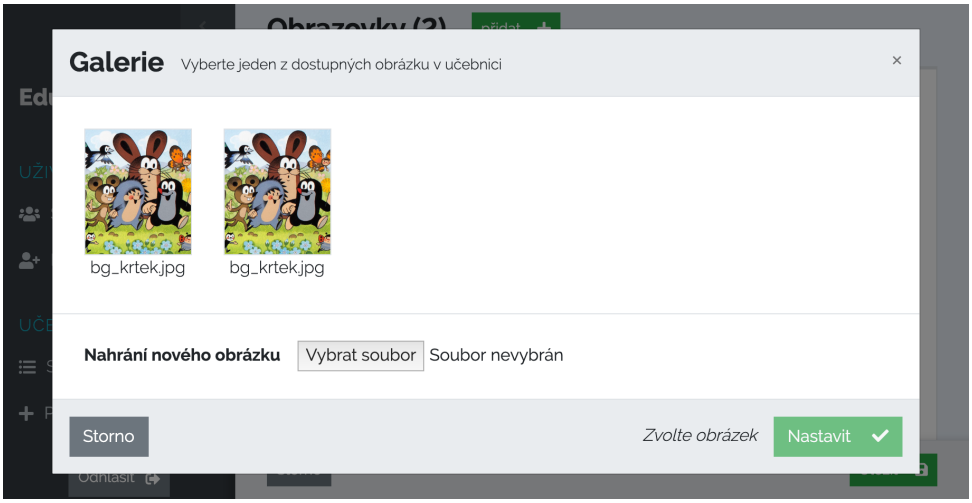
Obrázek 3.22: Upozornění na neuložená data

6. **Seznam obrazovek úkolu** — Viz obrázek 3.23. Při kliknutí na obrazovku v seznamu se otevře její detail, kde můžeme zvolit druh obsahu (2D nebo 3D), otevřít v modálním oknu galerii a spravovat otázky.



Obrázek 3.23: Seznam obrazovek úkolu

- **Procházení galerie** — Na obrázku 3.24 je znázorněno procházení galerie, například při vložení obrázku na obrazovku.

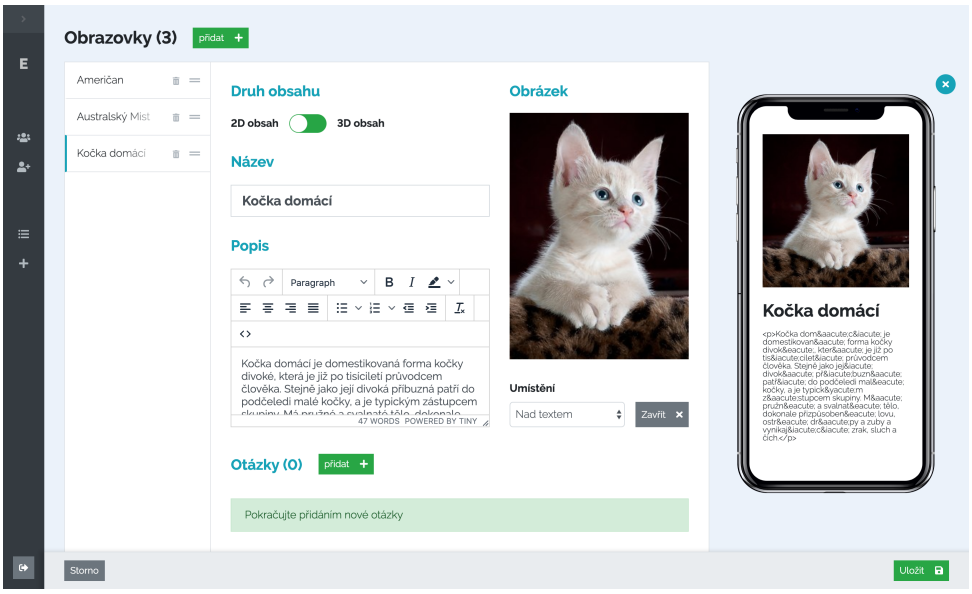


Obrázek 3.24: Galerie obrázků

3.7 Splnění kvalitativních požadavků

3.7.1 Náhledy

Kromě funkčních požadavků na aplikaci jsme si v sekci 2.6 definovali požadavky kvalitativní. Jedním z nich byl požadavek na umožnění náhledu zadávaných informací. Tento požadavek byl splněn implementací náhledu umístění obrázku v detailu obrazovky. Po zvolení obrázku z galerie či nahrání nového souboru je uživateli nabídnuta možnost zvolit umístění obrázku vůči textu. Možné pozice obrázku byly popsány v definici obrazovky v sekci 2.4.3. Po kliknutí na tlačítko „Náhled“ se uživateli z pravé strany webové stránky zobrazí náhled konečného výsledku na obrazovce chytrého telefonu. Toto chování je znázorněno na obrázku 3.25. Zobrazení chytrého telefonu je dosaženo pouhým použitím kaskádových stylů, které příslušnému DOM prvku nastaví do pozadí obrázek chytrého telefonu. Texty a obrázek pak tvoří obsah tohoto DOM prvku.



Obrázek 3.25: Náhled umístění obrázku vůči textu

3.7.2 Multijazyčnost

Jedním z kvalitativních požadavků na aplikaci byla možnost v budoucnu nasadit systém i do zahraničních škol. Znamená to, že aplikace musí umožňovat programátorsky přepnout jazyk popisků a zobrazovaných hlášek i do jiných jazyků než čeština. Knihovna React.js nabízí několik modulů pro překlad obsahu webové stránky. Všechny tyto moduly nabízí širokou škálu možností a musí se instalovat jako další knihovna do aplikace. Jako lepší řešení se zdá být vytvoření konfiguračního souboru pro každý jazyk a referencování popisků podle předdefinovaných konstant. Toto řešení je odlehčenější, jelikož se používá pouze jeden soubor pro každý jazyk, ne celá knihovna obsahující

složitou logiku pro přepnutí jazyku. Řešení je také dost jednoduché v použití, jelikož místo stávajících českých popisků se vloží pouze příslušná konstanta, popisky se nemusí obalovat dalším kontejnerem, jak je tomu u modulů třetích stran. Pro lokalizaci celého systému do dalšího jazyka stačí udělat dva kroky:

1. Vytvořit nový soubor v adresáři `/src/js/_config/languages` s popisky v příslušném jazyce
2. V souboru `/src/js/_config/_config.js` změnit hodnotu konstanty `LANG` podle příslušného jazyka

Editory jako např. WebStorm nebo InteliJ IDEA v druhém kroku samy nabídnou import příslušné konstanty ze souboru s popisky.

Příklad použití. Uvedeme příklad lokalizaci systému v anglickém jazyce.

1. Vytvoříme nový soubor `/src/js/_config/languages/_en.js`. Obsah souboru bude obsahovat konstantu `EN` (název není striktně předepsán), která bude reprezentovat objekt popisků v následujícím tvaru:

   `"předdefinovaný klíč": "lokalizovaný popisek".`

   Názvy předdefinovaných klíčů se měnit nesmějí, používají se v aplikaci při odkazu na lokalizovaný popisek.

   ```
   export const EN = {

       Welcome: 'Welcome in EduARd',
       WelcomeTitle: 'Educational system EduARd',

       ...
   }
   ```

2. V souboru `/src/js/_config/_config.js` nastavíme hodnotu konstanty `LANG` na název naší jazykové konstanty:

   ```
   import {EN} from "./languages"

   export const LANG = EN;
   ```

V aplikaci je každý popisek referencován následujícím způsobem:

```
<div className='Homepage-Title'>
    <span>{LANG.Welcome}</span>
    <h3>{LANG.WelcomeTitle}</h3>
</div>
```

Veškerá jazyková nastavení jsou tedy provedeny pouhým přidáním nového jazykového souboru `_en.js` a nastavením konstanty `LANG` na hodnotu `EN`.

Kapitola 4
Testování

4.1 Testování komponent knihovny React.js

Je dobrým zvykem před testováním aplikace s reálnými uživateli otestovat její funkčnost pomoci programátorských testů. Jedněmi z mnoha nástrojů pro testování aplikací napsaných v knihovně React.js jsou Jest od společnosti Facebook[24] a Enzyme od společnosti Airbnb[25]. Pomoci nich se dá otestovat správné načtení jednotlivých komponent webové stránky a jejích chování při provedení uživatelské akce (např. při kliknutí na tlačítko). Testy prováděné v naší aplikaci jsou k naleznutí v příloze této bakalářské práce jako součást implementace. Uvedeme pouze funkce, které se v těchto testech testovali:

- Správné načtení aplikace
- Přesměrování nepřihlášeného uživatele na přihlašovací stránku
- Zobrazení a skrytí modálních oken:
  - pro přidání uživatelů
  - pro editaci uživatelských rolí
  - pro přidání učebnice
  - pro správu obrázků v galerii učebnice
- Načtení komponenty pro zobrazení detailu obrazovky
- Přidání otázky na obrazovku

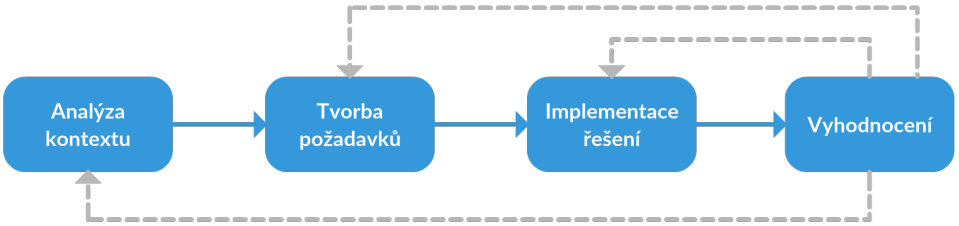
4.2 Testování během vývoje

Aplikace se vyvíjela v souladu s principy User-Centered Design (česky: Design orientovaný na uživatele). Jedná se o přístup k vývoji uživatelských rozhraní, kdy se uživatel neustále vývoje účastní a případně vznáší požadavky na změny[26]. Vývoj pak probíhá v několika fázích:

- Analýza kontextu použití aplikace

- Tvorba uživatelských požadavků
- Implementace řešení
- Vyhodnocení řešení oproti požadavkům

V každé z těchto fází se vývojář soustředí hlavně na uživatele a na jeho požadavky. Po poslední fázi vyhodnocení se vývoj vrací zpět k analýze, tvorbě požadavků a implementaci řešení s ohledem na nově získané informace. Tento proces je znázorněn na obrázku 4.1.



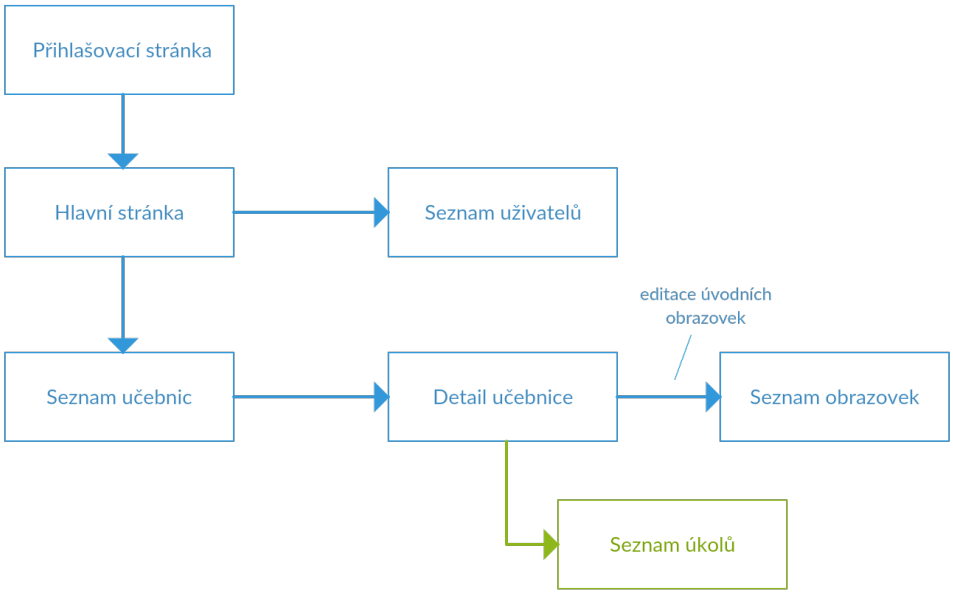
Obrázek 4.1: Fáze vývoje v souladu s principy User-Centered Design

Naše aplikace se vyvíjela jako jedna z komponent systému EduARd. Aplikace tvořila vstupní data pro ostatní části systému, proto ji během vývoje aktivně používali jiní členové vývojového týmu. Toto poskytlo možnost odladit vzhled a chování aplikace ještě před testováním s cílovou skupinou uživatelů. Jednou z hlavních změn uživatelského rozhraní aplikace bylo přesunutí seznamu úkolů učebnice na samostatnou stránku. Tato změna reagovala na požadavek k omezení nutnosti posouvat stránku v případě editace úkolů. Na obrázku 4.2 je znázorněná změněná struktura aplikace po provedené změně. Vazba na seznam obrazovek z detailu učebnice zůstala pro případ editace úvodních obrazovek učebnice.

4.3 1. kolo testování

Pro testování hotové aplikace byla zvolena skupina lidí, jejíchž věk odpovídá cílové uživatelské skupině. To znamená, že se věk účastníků testování pohyboval v rozmezí 30-45 let. To zajistilo možnost otestovat aplikaci jak s mladší, tak i se starší generací uživatelů. I když věkový rozdíl není na první pohled tak razantní, během testování se projevil rozdílný způsob použití aplikace u mladších a straších uživatelů. Mladší uživatelé měli tendenci se v aplikaci pohybovat rychleji, nezkoumali všechny informace, spíš hledali CTA[1] tlačítka, která by je posunula v úkolech dál. Vyzkoušeli taky různé zajímavé řešení systému: sbalovací menu a náhled umístění obrázku. Starší uživatelé takové zajímavosti zcela přehlíželi, vykonání nějaké akce si vždy rozmýšleli a každou provedenou změnu uložili, aniž by k tomu byli vyzváni. Toto pozorování potvrdilo splnění jednoho z kvalitativních požadavků na aplikaci: intuitivní v použití pro různé věkové kategorie uživatelů.

[1] Call To Action



Obrázek 4.2: Struktura aplikace po přesunutí úkolů na samostatnou stránku

Účastníci testování měli za úkol vyzkoušet jak roli správce instituce, tak i roli učitele. K obou účtům dostali přístupy a odkaz na online verzi aplikace. Úkoly byly následující:

- Správce
  1. Přidání uživatelů — Přidat 3 nové učitelé.
  2. Zrušení pozvánky — Zrušit pozvánku jednomu z přidaných učitelů.
  3. Přidání uživatele — Přidat nového studenta.
  4. Editace uživatele — Přidat roli studenta existujícímu učiteli v systému.
- Učitel
  1. Přidání učebnice — Založit novou učebnici s daným názvem.
  2. Editace učebnice — Přidat k založené učebnici popis a počet pokusů na splnění úkolů.
  3. Přidání úvodní obrazovky — Přidat 2 úvodní obrazovky. Ka každé obrazovce nahrát obrázek.
  4. Přidání výukové úlohy — Založit v učebnici úkoly, které by napovídaly trasu. Každý úkol měl daný název a lokaci.
  5. Editace výukové úlohy — V každém úkolů založit 1 nebo 2 obrazovky obsahující různé typy otázek nebo 3D scénu.
  6. Editace obrazovek — Změnit pořadí obrazovek v daném úkolu. Změnit obrázek na jedné z úvodních obrazovek učebnice.
  7. Odstranění obrazovek — Odstranit danou obrazovku v daném úkolu.

8. Odstranění výukové úlohy — Odstranit úkol v učebnici.

Celé znění testovacích úkolů je k naleznutí v příloze B.

4.3.1 Výsledky

- Správce
  1. Přidání uživatelů
     - Bez problémů u všech účastníků.
  2. Zrušení pozvánky
     - Bez problémů u všech účastníků.
  3. Přidání uživatele
     - Bez problémů u všech účastníků.
  4. Editace uživatele
     - Bez problémů u všech účastníků.
- Učitel
  1. Přidání učebnice
     - Bez problémů u všech účastníků.
  2. Editace učebnice
     - Účastníci nedokázali hned alokovat tlačítko pro uložení.
     - Účastníci si nevšimli hlášky o proběhlém uložení učebnice.
  3. Přidání úvodní obrazovky
     - Bez problémů u všech účastníků.
  4. Přidání výukové úlohy
     - Účastníci hledali způsob se dostat k editaci úkolů z editace úvodních obrazovek.
     - Všichni účastníci se vrátili zpět na seznam učebnic a po tom přešli do detailu učebnice. Odtud se dostali na editaci úkolů.
  5. Editace výukové úlohy
     - Všichni účastníci se bez problémů dostali na editaci obrazovek aktivního úkolu. Ve chvíli, kdy po editaci obrazovek měli pokračovat v přidávání úkolů, byli zmatení. V tuto chvíli nevěděli rozdíl mezi úkolem a obrazovkou, co z toho je rodičovským prvkem a co potomkem.
     - Všem účastníkům trvala navigace mezi seznamem obrazovek jednoho úkolů a seznamem obrazovek druhého úkolů. Navíc se ve struktuře systému ztráceli.
     - Účastníci měli problém se zadáním otázky na interval. Nevěděli čím se liší `rozsah intervalu` a `validní interval odpovědi`.
  6. Editace obrazovek

- Starší uživatelé nenapadlo, že po najetí myší na stávající obrázek úkolů ho budou moct změnit či odstranit.

7. Odstranění obrazovek
   - Bez problémů u všech účastníků.
8. Odstranění výukové úlohy
   - Bez problémů u všech účastníků.

Výsledky prvního kola testování ukázaly, že největším problémem byla navigace mezi úkoly a obrazovky v rámci jedné učebnice. Uživatelé, kteří systém testovali během vývoje byli se strukturou učebnice dobře obeznámení, proto podobné potíže nezažívali. Potenciální koneční uživatelé však struktura systému zmátla a způsobila to, že navigace mezi jednotlivými stránkami aplikace trvala mnohem déle, než by bylo uživateli příjemné.

Na základě výsledků testování byly vytvořeny požadavky pro zlepšení uživatelského požitku z použití aplikace a navrženy možné způsoby řešení:

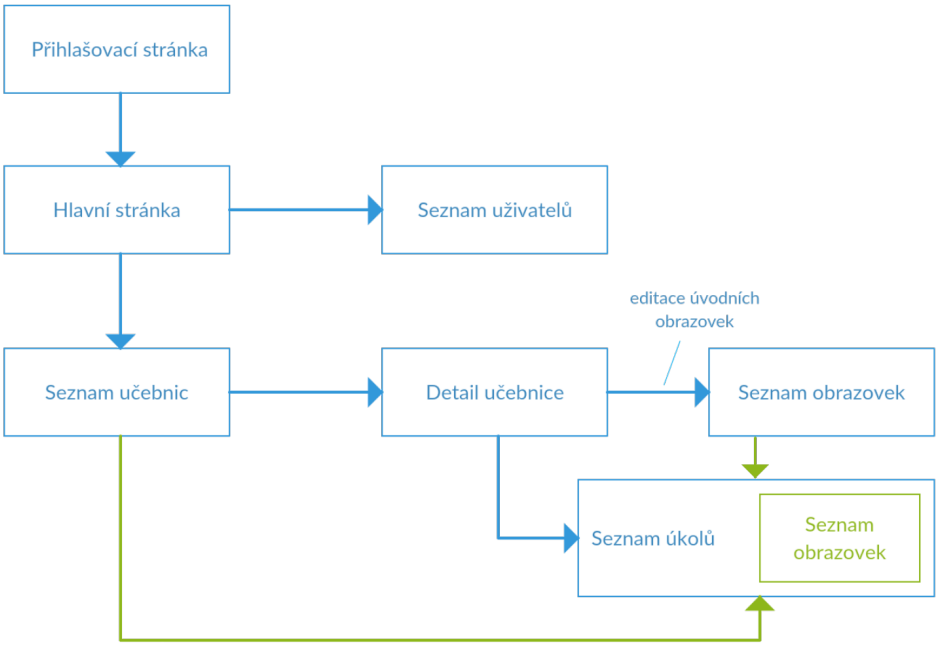
1. **Požadavek** - Zvýraznit tlačítko na uložení.
   - **Řešení** - Posunout tlačítko k názvu sekci, což je viditelnější pozice, a zvětšit.
2. **Požadavek** - Zvýraznit hlášku o uložení proběhlých změn.
   - **Řešení** - Zobrazovat hlášku o uložení v modálním oknu.
3. **Požadavek** - Přidat možnost navigace z úvodních obrazovek na editaci úkolů učebnice.
   - **Řešení** - Přidat tlačítko `Editovat úkoly učebnice` do detailu každé úvodní obrazovky.
4. **Požadavek** - Vylepšit navigaci mezi detailem učebnice a jejími součástmi.
   - **Řešení 1** - Přidat možnost přejít na editaci úkolů učebnice ze seznamu učebnic.
   - **Řešení 2** - Přidat drobečkovou navigaci.
5. **Požadavek** - Znázornit vztah mezi úkolem a jeho obrazovkami.
   - **Řešení 1** - Spojit seznam úkolů a obrazovek do jednoho dvouúrovňového seznamu. To znamená, že seznam obrazovek bude vnořen do seznamu úkolů a úkoly, které obsahují obrazovky, budou rozbalovácí.
   - **Řešení 2** - Přidat možnost přidávat nové obrazovky úkolu hned při jeho editaci.
   - **Řešení 3** - Přidat možnost přidat nový úkol při editaci obrazovky.
6. **Požadavek** - Znázornit možnost editovat obrázek obrazovky.

- **Řešení** - Přidat ikonu označující editaci (např. ozubené kolo). To donutí uživatele přejet myší přes obrázek, čímž se mu zobrazí nabídka editace obrázku.

7. **Požadavek** - Vymyslet uživatelsky příjemnější způsob zadání otázky na interval.

   - **Řešení** - Řešení necháváme otevřené k diskuzi se zadavatelem. Hodilo by se buď implementovat nějakou názornou ukázku toho, jak se interval bude používat v klientských aplikacích, anebo zjednodušit strukturu otázky. Vyřešení tohoto požadavku necháme na vývoj mimo bakalářskou práci.

Pro znázornění provedených změn ve struktuře aplikace slouží obrázek 4.3.



Obrázek 4.3: Struktura aplikace po přesunutí úkolů na samostatnou stránku

4.4 2. kolo testování

Po zpracování změn z předchozího kola testování provedeme druhé kolo testování, tentokrát s jinými uživateli než v předchozím kole. Zajímá nás, zdá se zrychlila navigace v systému a zda je pro uživatele názornější vztah mezi úkolem a obrazovkou. Během druhého kola testování uživatelé aktivně používali drobečkovou navigaci a přidaná navigační tlačítka. Potíže měli pouze u přidání posledního úkolů, který obsahovat pouze jednu obrazovku s 3D scénou. V tuto chvíli si uživatele zase zaměnili pojmy „úkol“ a „obrazovka“. Tento problém ale vznikl hlavně kvůli tomu, že data, která se zadávala,

vymýšlel někdo jiný, a taky chybělo základní školení uživatelů. Kdyby tento úkol zadával učitel, který před tím tuto 3D scénu sám vytvořil, tak by jistě věděl, že ji vkládá do obrazovky nějakého úkolů.

4.4.1 Výsledky

Změny provedené po druhém kole testování byly spíše kosmetické, anebo k diskuzi se zadavatelem:

1. Posunutí seznamu odeslaných pozvánek uživatelům nad seznam aktivních uživatelů. Pozvánky reprezentují proměnlivou část a po přijetí všech pozvánek se táto část stává prázdnou a ze stránek mizí. Je to první věc, která zajímá správce.

   - Zpracováno.

2. Zvýraznění možnosti skrýt hlavní nabídku.

   - Zpracováno.

3. Přejmenování částí učebnic: místo `Úkoly` používat slovo `Lokace`, místo `Obrazovky - Stanoviště`.

   - K diskuzi se zadavatelem.

4.5 Vzhled aplikace po testování

Proběhlá testování s uživateli měla velký dopad na vzhled a strukturu aplikace. Počet stránek aplikace se nezměnil, ale změnil se obsah některých z nich. Vzhled aplikace **před** testováním je znázorněn na obrázcích v sekci 3.6. To, jak jednotlivé stránky naší aplikace vypadají **po** testování, popíšeme v následujících odstavcích.

1. **Přihlašovací stránka**

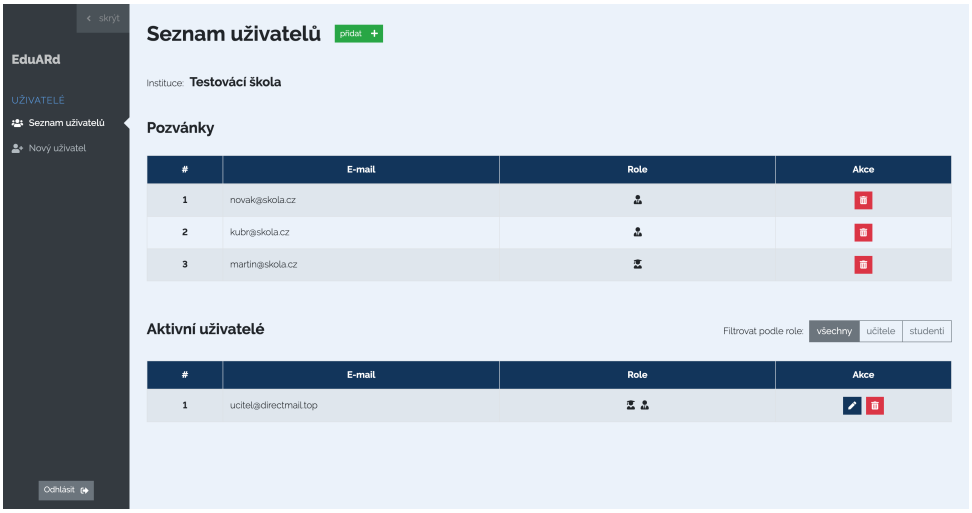
   Stránka zůstala beze změn. Její podobu je možné vidět na obrázku 3.13.

2. **Ovládací panel**

   Stránka zůstala beze změn. Její podobu je možné vidět na obrázku 3.14.

3. **Seznam uživatelů**

   Na stránce se seznamem uživatelů proběhla pouze jedna změna, a to vyměnění pozic seznamu pozvánek a seznamu aktivních uživatelů. Nový vzhled stránky je znázorněn na obrázku 4.4.

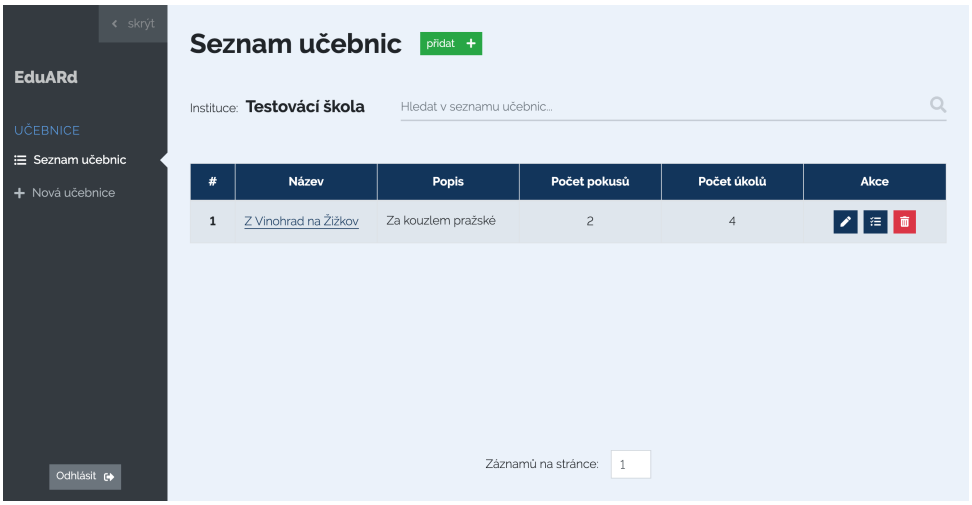


Obrázek 4.4: Nový vzhled stránky se seznamem uživatelů

Na obrázku 4.4 si lze všimnout, že tlačítko pro skrytí hlavní nabídky získalo popisek a stalo se výraznější. Hlavní nabídka aplikace zobrazuje pouze možnost správy uživatelů kvůli tomu, že právě přihlášený uživatel má jedinou roli správce.

4. Seznam učebnic

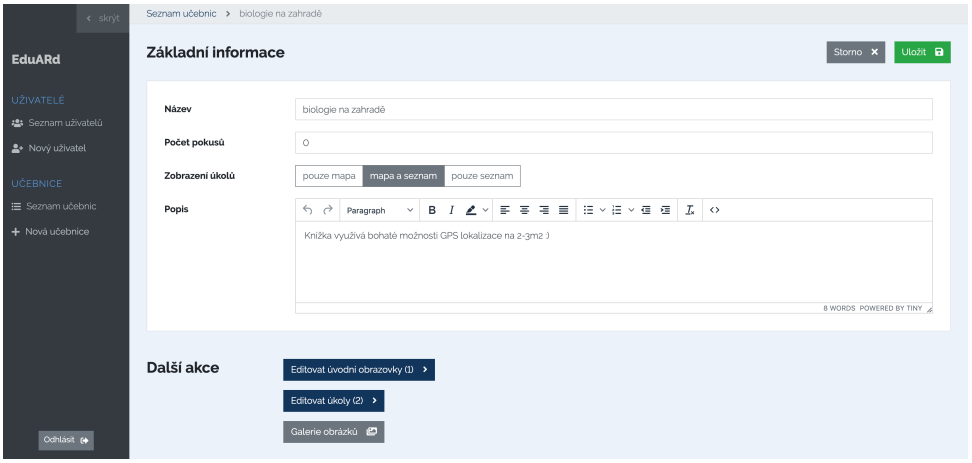
Každá položka seznamu dostala navíc tlačítko pro editaci úkolů. Tím byla umožněna navigace na editaci úkolů učebnice aniž by se musel zobrazit její detail. Tento stav je vidět na obrázku 4.5. Hlavní nabídka zde ukazuje pouze možnost správy učebnic, jelikož přihlášený uživatel má jedinou roli učitele.



Obrázek 4.5: Nové tlačítko pro editaci úkolů v položce seznamu

5. Detail učebnice

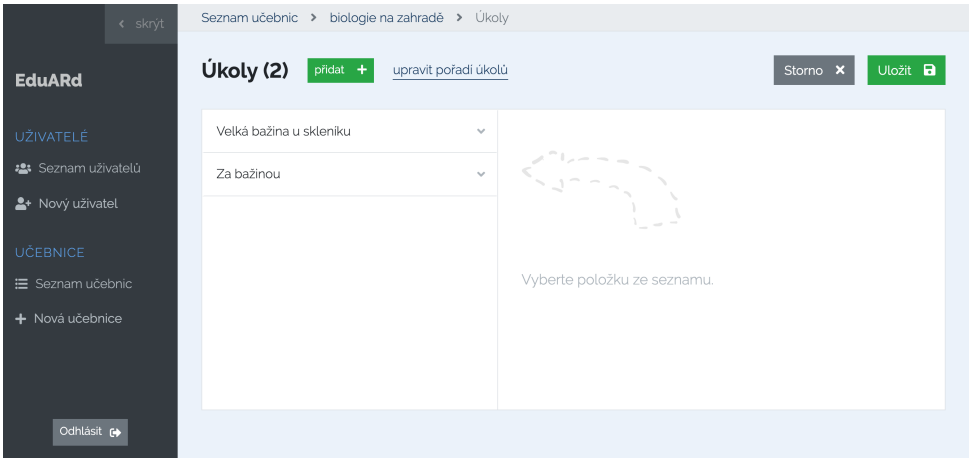
Největší změnou je přesunutí seznamu úkolů z detailu učebnice na samostatnou stránku. Místo něj se objevila tlačítka pro navigaci na editaci úvodních obrazovek, úkolů a otevření galerie. Další změnou je umístění tlačítek pro uložení a storno. Nyní jsou umístěny nahoře u názvu sekce a mají větší velikost. Z obrázku 4.6 znázorňujícího nový vzhled detailu učebnice, je taky patrná drobečková navigace. Díky ní se uživatel může pohybovat zpětně do každého předchozího kroku editace učebnice.



Obrázek 4.6: Nový vzhled detailu učebnice

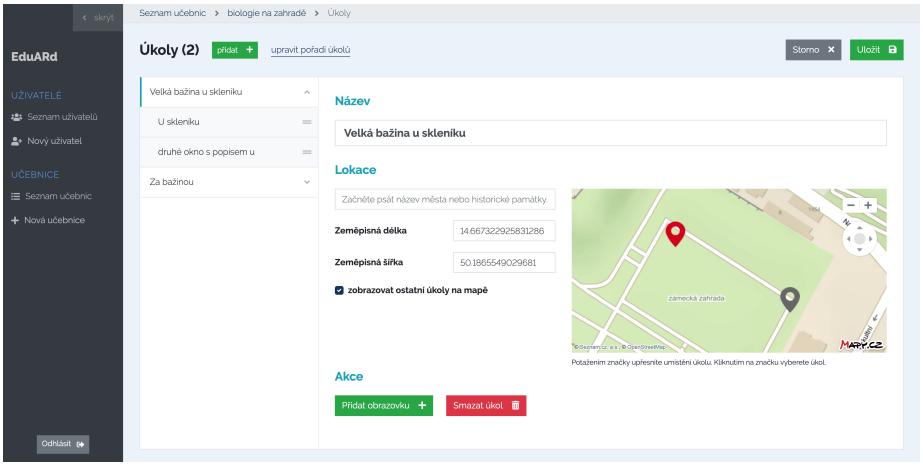
6. Seznam úkolů a obrazovek

Nově přidána stránka. Vznikla v důsledku přesunutí seznamu úkolů na samostatnou stránku. Po navigaci na stránku je uživatel vyzván ke zvolení úkolu, viz obrázek 4.7.



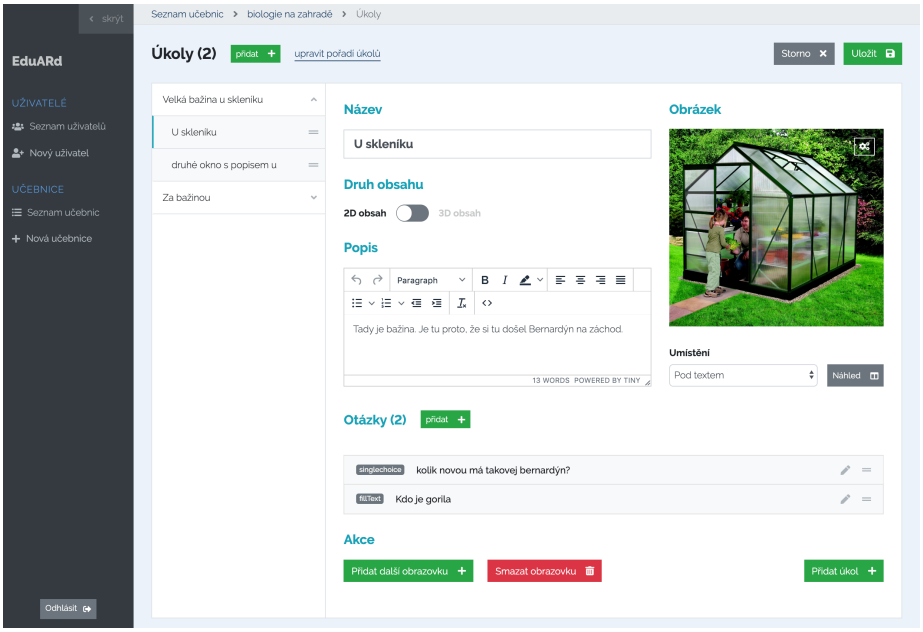
Obrázek 4.7: Vzhled seznamu úkolů po navigaci na stránku

Později se na tuto stránku přidal i vnořený seznam obrazovek každého úkolu. Po kliknutí na položku úkolu se v pravé straně zobrazí jeho detail a ve výpisu se rozbalí seznam podřazených obrazovek. Tato situace je znázorněna na obrázku 4.8. V detailu úkolu přibyla tlačítka pro přidání nové obrazovky úkolu a odstranění aktivního úkolu.



Obrázek 4.8: Vzhled seznamu úkolů s aktivním detailem úkolu

Kliknutím na položku rozbaleného menu se nám pravá strana vymění za detail aktivní obrazovky, viz obrázek 4.9. Z detailu obrazovky jsou patrná nová tlačítka pro přidání další obrazovky úkolu, smazání aktivní obrazovky a přidání nového úkolu. Tato změna výrazně urychlila proces editace učebnice.



Obrázek 4.9: Vzhled seznamu úkolů s aktivním detailem obrazovky

Kapitola 5

Závěr

Cílem této bakalářské práce bylo implementovat webovou aplikaci pro správu mobilních výukových úloh, která by navíc umožnila správu uživatelů jedné instituce. Aplikace měla být intuitivní pro všechny věkové kategorie a měla umožnit uživateli zobrazit náhled zadaných dat. Kromě toho, mělo být možné celou aplikaci jednoduše přepnout do jiného jazyka. Tyto požadavky byly podrobněji definovány v sekcích 2.5 a 2.6.

Během implementace se mi povedlo všechny definované požadavky splnit. Velkou roli a největší dopad na finální podobu aplikace mělo závěrečné testování s potenciálními uživateli, které bylo popsáno v sekci 4.2. Ve vzhledu a struktuře aplikace po testování proběhlo několik změn, které jsou vidět na obrázcích v sekci 4.5.

5.1 Budoucí vývoj

Doufám, že v budoucnu budu na vývoji aplikace pokračovat. Chtěla bych ji rozšířit o další funkcionality, jako je například sdílení učebnic mezi institucemi, uložení učebnice do seznamu oblíbených učebnic uživatele a vyhodnocení úkolů, které studenti na svých zařízeních splnili. Většina těchto funkcionalit vyžaduje další práci na systému EduARd jako celku, proto je momentálně není možné implementovat odděleně v rámci webové aplikace.

Pro budoucí vývoj bych doporučila rozšířit aplikaci o další možnosti náhledu zadávaných dat. Hodilo by se mít náhled v každém kroku editace učebnice.

Zvětšení pokrytí komponent testy považuji za nezbytný bod dalšího vývoje aplikace. Na programátorské testy často při vývoji aplikací nezbývá čas, není to ale část, která by se měla zanedbat. Programátorské testy pomáhají předejít nečekanému chování systému, například v důsledku reakce na uživatelskou akci. U naší aplikaci bych ráda těmito testy pokryla všechny komponenty systému. K tomu je potřeba implementovat simulaci lokálního REST API. Tato část testování byla z rozsahu bakalářské práce vynechána, jelikož je časově náročná a spoléhá na konečnou verzi REST API, která v tuto chvíli vývoje není jistě dána.

Nesmí při dalším vývoji chybět i testování s uživateli, které by se mělo provádět pravidelně a s cílovou skupinou uživatelů.

Příloha A

Literatura

[1] Google, Inc., "Material design - navigation drawer - visibility." [Online]. Available: https://material.io/design/components/navigation-drawer.html#standard-drawer

[2] Knesplová, Jana, MBA, "Tablety ve školách pomáhají," *Řízení školy*, 5 2015, rubrika: ICT ve škole.

[3] Google LLC. [Online]. Available: https://docs.google.com/

[4] W3Techs.com, "8 guidelines for exceptional web design, usability, and user experience." [Online]. Available: https://blog.hubspot.com/blog/tabid/6307/bid/30557/6-guidelines-for-exceptional-website-design-and-usability.aspx

[5] Roy Thomas Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[6] Berners-Lee, et al., "Uniform resource identifier (uri): Generic syntax." [Online]. Available: https://tools.ietf.org/html/rfc3986

[7] MuleSoft, LLC, "What is an api?" [Online]. Available: https://www.mulesoft.com/resources/api/what-is-an-api

[8] Balsamiq Studios, LLC. [Online]. Available: https://balsamiq.cloud

[9] Facebook Inc., "React.js." [Online]. Available: https://reactjs.org/

[10] MDN web docs, Mozilla and individual contributors, "Javascript." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[11] MDN web docs, Mozilla and individual contributors, "Document object model (dom)." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

[12] Google, Inc., "Angular." [Online]. Available: https://angular.io/

[13] MDN web docs, Mozilla and individual contributors, “Cascading style sheets (css).” [Online]. Available: https://developer.mozilla.org/cs/docs/Web/CSS

[14] Hampton Catlin, Natalie Weizenbaum, Chris Eppstein, Jina Anne, and numerous contributors, “Sass: Syntactically awesome style sheet.” [Online]. Available: https://sass-lang.com/

[15] Mark Otto, Jacob Thornton, and contributors, “Bootstrap.” [Online]. Available: https://getbootstrap.com/

[16] Yarn Contributors., “Yarn.” [Online]. Available: https://yarnpkg.com/lang/en/

[17] npm, Inc., “npm.” [Online]. Available: https://www.npmjs.com/

[18] Fonticons, Inc., “Font awesome.” [Online]. Available: https://fontawesome.com/

[19] Google, Inc., “Google maps.” [Online]. Available: https://maps.google.com/

[20] Google, Inc., “Pricing and plans. google maps platform.” [Online]. Available: https://cloud.google.com/maps-platform/pricing/

[21] Seznam.cz, a.s., “Mapy api.” [Online]. Available: https://api.mapy.cz/

[22] MDN web docs, Mozilla and individual contributors, “Using fetch.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

[23] MDN web docs, Mozilla and individual contributors, “Working with objects.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects

[24] Facebook Inc., “Jest. delightful javascript testing.” [Online]. Available: https://jestjs.io/en/

[25] Airbnb, Inc., “Enzyme.” [Online]. Available: https://airbnb.io/enzyme/

[26] T. Lowdermilk, *User-centered design: a developer’s guide to building user-friendly applications.* "O’Reilly Media, Inc.", 2013.

Příloha B

Úkoly k testování s uživateli

B.1 Správce

B.1.1 Správa uživatelů

1. Přidat nové učitelé: novak@skola.cz, kubr@skola.cz a martin@skola.cz.

2. Zrušit pozvánku uživateli martin@skola.cz, jelikož není učitel, ale student.

3. Přidat nového studenta: martin@skola.cz.

4. Učitel ucitel@email-wizard.com chce testovat i studentskou část aplikace. Přiřadit mu také roli studenta.

B.2 Učitel

B.2.1 Přidání učebnice

Dostal se vám do ruky materiál o pražské čtvrti Žižkov a chcete vzít děti na procházku po jeho nejzajímavějších místech. K tomu potřebujete založit učebnici, která by vám napovídala trasu, zastávky a úkoly spojené s každým místem.

Učebnice

Učebnice, kterou založíte, se bude jmenovat „Z Vinohrad na Žižkov“. Její popis bude „Za kouzlem pražské bohémy“ a studenti budou mít 2 pokusy na splnění úkolů.

Úvodní obrazovky

Název	Popis	Obrázek
Historické centrum	Legendární Žižkov je – mimo historické centrum – snad nejslavnější pražskou čtvrtí. V dobách minulých jeho převážně dělnické obyvatelstvo, veselý život místních hospůdek a kabaretů i rozmanitá, kopcovitá krajina ve stínu vrchu Vítkova společně vytvořily neodolatelnou auru, které propadl kdekterý aspirující umělec.	
Žižkovští Jardové	Žižkov byl i domovem dvou Jaroslavů – Haška, autora světoznámého humoristického románu Osudy dobrého vojáka Švejka za světové války, a básníka Seiferta, jediného českého nositele Nobelovy ceny za literaturu. Dnešní Žižkov se rychle mění, ale jeho srdce zůstává bohémské stále.	

Úkoly (trasa)

1. Náměstí Míru

Název	Náměstí Míru
Lokace	Náměstí Míru

1. obrazovka

Název	Trhy	
Popis	O tom, že je svažité náměstí Míru centrálním bodem Vinohrad, není pochyb.	
Otázka s jednou správnou odpovědí	Text	Kdy se na náměstí konají trhy?
	Odpovědi (správná odpověď je zvýrazněná tučně)	každý den **v období svátků** pouze o Vánocích pouze o víkendech

2. obrazovka

Název	Kostel sv. Ludmily	
Popis	Ve středu náměstí stojí působivý novogotický kostel sv. Ludmily, jehož vysoké věže dominují širokému okolí.	
Otázka s číselnou odpovědí	Text	Jak jsou věže vysoké (v metrech)?
	Odpověď	60

2. Divadlo na Vinohradech

Název	Divadlo na Vinohradech
Lokace	Divadlo na Vinohradech

1. obrazovka

Název	Budova divadla	
Popis	V kontrastu ke strohým liniím kostela se po levé straně náměstí rýsuje secesní Divadlo na Vinohradech s bohatě zdobenou fasádou.	
Otázka na interval	Text	V jakých letech probíhala stavba budovy?
	Správný interval	1905–1907

2. obrazovka

Název	Slavnostní otevření	
Popis	Divadlo bylo slavnostně otevřeno 24. listopadu 1907. Prvním představením byla hra Jaroslava Vrchlického.	
Otázka s textovou odpovědí	Text	Jak se jmenovala táto hra?
	Odpověď	Godiva

3. Pavilon

Název	Pavilon
Lokace	Pavilon

1. obrazovka

Název	Exteriér
3D scéna	Scenario

B.2.2 Editace učebnice

1. Je potřeba odstranit obrázek úvodní obrazovky „Historické centrum“.
2. Potřebujete vyměnit foto Jaroslava Haška za foto Jaroslava Seiferta na úvodní obrazovce „Žižkovští Jardové“. Nové foto:

3. Je potřeba smazat úkol „Pavilion“.
4. Je potřeba prohodit obrazovky „Trhy“ a „Kostel sv. Ludmily“ v úkolu „Náměstí Míru“.

Příloha C

Instalační návod

Implementace využívá balíčkovací systém Yarn. Návod na jeho instalaci je k naleznutí na stránkách oficiální dokumentace[16].

Po rozbalení zdrojových souborů implementace je potřeba vykonat následující příkazy v Terminálu (Linux / MacOS) nebo Příkazovém řádku (Windows):

- Nainstalovat závislosti projektu pomocí příkazu:

```
yarn
```

- Spustit aplikaci lokálně na adrese `http://localhost:3000`:

```
yarn start
```

Nepovinné:

- Spustit testy:

```
yarn test
```

- Sestavit aplikaci pro produkci (výsledek sestavení je k naleznutí ve složce `/build`):

```
yarn build
```

Je možné použit balíčkovací systém npm, který je předchůdcem systému Yarn. Příkazy vypadají následovně (pořadí odpovídá pořadí příkazů systému Yarn):

```
npm install
npm start
npm test
npm run build
```