FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webové uživatelské rozhraní pro projekt Grades
Student:	Marek Mouček
Vedoucí:	Ing. Štěpán Plachý
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

- Analyzujte hlavní klíčové procesy v projektu Grades. Prozkoumejte a zhodnoťte stávající řešení.
- Na základě získaných informací navrhněte nový webový frontend, který zjednoduší tyto procesy.
- Zásadní části systému pro studenty jsou:
  1. Úvodní stránka s přehledem hodnocení zapsaných předmětů.
  2. Stránky hodnocení jednotlivých předmětů.
- Zásadní části systému pro vyučující jsou:
  1. Stránky pro zápis hromadného hodnocení.
  2. Stránky pro zápis hodnocení konkrétnímu studentovi.
- Soustřeďte se na uživatelské rozhraní a modulární implementaci tak, aby bylo možné jednoduše přidat další subsystémy.
- Uživatelské rozhraní by mělo být dostatečně responzivní pro pohodlné zobrazení a ovládání na počítači i mobilním telefonu.
- Při implementaci pro mobilní telefon se zaměřte na podporu offline funkcionality pomocí dostupných technologií.
- Uživatelské rozhraní řádně otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 29. ledna 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Webové uživatelské rozhraní pro projekt Grades

Marek Mouček

Katedra softwarového inženýrství
Vedoucí práce: Ing. Štěpán Plachý

14. května 2019

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 14. května 2019 ……

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Mouček, Marek. *Webové uživatelské rozhraní pro projekt Grades*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato práce se zabývá redesignem a přepsáním webové aplikace (frontend) pro projekt Grades do nejnovějších technologií s podporou offline funkcionality a optimalizací pro obrazovky mobilních zařízení. Nová webová aplikace komunikuje se stávajícím Grades API. Umožňuje studentům získávat informace o své klasifikaci z jednotlivých předmětů. Učitelům umožňuje pohodlné hodnocení studentů. Aplikace je naprogramována ve frameworku Angular a architektura je zvolena tak, aby byla co nejvíce modulární a udržovatelná.

Klíčová slova webová aplikace, Klasifikace FIT ČVUT, Angular, service workers, REST, progresivní webová aplikace

Abstract

This bachelor thesis deals with redisign and rewrite of the Grades web application (frontend) into latest technologies with support of offline functionality and optimalizations for mobile devices screen. The new web application communicates with the existing Grades API. It allows students to view all classifications of their subjects. It also allows teachers to assess students. The

application is programmed in the Angular framework and the architecture is chosen to be as modular and maintainable as possible.

Keywords web application, Klasifikace FIT ČVUT, Angular, service workers, REST, progressive web application

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Elektronická klasifikace FIT ČVUT v Praze (známá pod názvem Grades) je jedním z oficiálních řešení pro studentskou klasifikaci na naší fakultě. Studentům nabízí přehledné zobrazení bodů a známek z jednotlivých předmětů, učitelům velkou flexibilitu v definici pravidel pro hodnocení předmětů a možnost jednoduchého hodnocení studentů.

Její webové uživatelské rozhraní (webová aplikace) používá zastaralé technologie, proto je obtížné přidávat další funkcionality a udržovat její kód. Mimo tyto technické nedostatky jsou některé její části pro studenty nepřehledné a neodráží jejich potřeby. Webová aplikace v současné době působí jako technicky zaměřená, což snižuje uživatelskou přívětivost.

Vývojářský tým Grades musí kromě webové aplikace udržovat i mobilní aplikace. Nové řešení by mělo díky fungování bez připojení k internetu, responzivitě a dodržování principů progresivních webových aplikací nahradit mobilní aplikace. Nativní aplikace budou sice vždy o něco lepší, ale na druhou stranu je nutné místo několika implementací udržovat pouze jednu, což usnadňuje vývoj v malém týmu.

Cílem této bakalářské práce je zlepšit použitelnost a vzhled webové aplikace, udržitelnost kódu samotného tvorbou úplně nové webové aplikace a také využít nejmodernější technologie a postupy k tvorbě offline funkcionality. Součástí práce je dále revize současného REST API pro potřeby nového frontendu s případným doporučením vhodných budoucích vylepšení.

Motivací vylepšení webového rozhraní je snaha podpořit naši interní aplikaci a tímto způsobem přispět fakultě a také využít možností vývoje offline webových aplikací a progresivních webových aplikací, které v některých případech dokáží nahradit nativní mobilní aplikaci.

Práce je rozdělena do několika kapitol. Nejprve jsou zkoumány možnosti jednotlivých technologií a popsány principy, které jsou dále dodrženy. Poté je analyzováno stávající řešení, jeho nedostatky a následně jsou navržena vhodná vylepšení a na jejich základě nová aplikace. Následně je popsána architektura

nového řešení a také jsou navržena vhodná vylepšení stávajícího REST API. Poslední kapitole se zabývá testováním použitelnosti nové webové aplikace a automatickými testy.

Kapitola 1

Cíl práce

Cílem práce je analyzovat klíčové procesy v projektu klasifikace FIT ČVUT v Praze (známou pod názvem Grades), prozkoumat a zhodnotit stávající řešení. Na základě těchto informací navrhnout nový webový frontend, který zjednoduší tyto procesy.

Při analýze a implementaci se práce zaměřuje na některé klíčové stránky tohoto projektu. Pro studenty je zásadní úvodní stránka s přehledem a hodnocením zapsaných předmětů a stránky s hodnocením jednotlivých předmětů. Pro učitele jsou zásadní stránky pro zápis hromadných hodnocení předmětů a stránky pro zápis hodnocení konkrétních studentů.

Dále je cílem vytvořit a implementovat kvalitní a responzivní uživatelské rozhraní (pro pohodlné zobrazení na mobilních telefonech) a také modulární implementace pro snadnou rozšiřitelnost systému. Při implementaci bude využita technologie service workers pro offline funkcionality. Nakonec bude vše řádně otestováno.

Kapitola 2

Rešerše

V této kapitole stručně představím technologie a standardy, které budou v rámci práce využívány. Popíši také fungování frameworku Angular a především technologie service workers, která je klíčová pro požadovanou funkcionalitu.

2.1 REST

REST (REpresentational State Transfer) je sada principů (není to protokol) pro návrh webového API (application programming interface) založeném na HTTP protokolu. Pro získání nebo úpravy částí dat se volá konkrétní URL endpoint (cesta ke zdroji). [1]

Volání endpointu se nazývá request a zpětně zaslaná data se nazývají response. Názvosloví je tedy shodné s HTTP požadavky. [1]

2.1.1 Práce se zdroji

REST je postaven na HTTP protokolu a pracuje se zdroji pomocí requestů (které obsahují HTTP metodu). [2]

Metody a jejich použití:

- GET – využívá se pouze k získání (čtení) zdroje, metoda je bezpečná (nemodifikuje zdroj).
- POST – slouží k tvorbě nového zdroje.
- DELETE – využívá se k odstranění zdroje.
- PUT – slouží ke kompletní aktualizaci (změna všech parametrů) nebo nahrazení již vytvořeného zdroje.
- PATCH – používá se k částečným úpravám zdroje (např. jednoho parametru). [2]

Poté se vrací response s návratovým kódem a případně i daty. Návratové kódy jsou závislé na metodě a na úspěšnosti této operace. [2]

2.1.2 Principy

REST API staví na následujících principech:

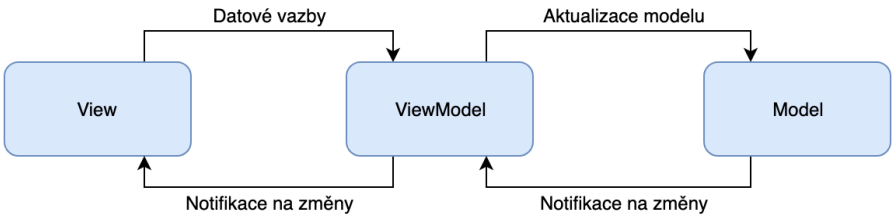
- Jednotné rozhraní – každý zdroj by měl mít pouze jednu adresu. Všechny adresy v API by měly mít logickou strukturu a dodržovat stejná pravidla pojmenování.
- Architektura klient/server – serverová a klientská část aplikace by na sobě neměly být závislé, což umožňuje nezávislý vývoj těchto částí.
- Bezstavovost – jednotlivé interakce klient/server na sobě nesmí být závislé. Každá interakce obsahuje všechna data a je nezávislá na předchozích requestech.
- Kešovatelnost – kešování vhodných requestů. Zlepšuje odezvu a škálovatelnost celého systému.
- Vrstvený systém – REST umožnuje použití vrstvených systémů (například je možné na jednom serveru data ukládat a na dalším ověřovat autentizaci). [3]

2.2 Angular

Angular je často používaný framework, který se využívá pro tvorbu single-page webových aplikací v HTML a TypeScriptu. Je to novější verze frameworku AngularJS, kde bylo množství změn tak vysoké, že se Google rozhodl framework AngularJS ve verzi 2 přejmenovat na Angular. Nové verze frameworku Angular jsou oproti frameworku AngularJS rychlejší, spolehlivější, zabírají méně místa a jsou pro vývojáře přívětivější. [4]

Architektura angularu do jisté míry využívá softwarový návrhový vzor MVVM (model-view-viewmodel). Zvolil jsem tento framework, protože je robustní a obsahuje všechny potřebné nástroje pro vývoj škálovatelných webových aplikací, zároveň má kolem sebe velkou a aktivní komunitu a je napsán v jazyku TypeScript, který je snadné v rámci frameworku Angular používat.

TypeScript je nástavba nad jazykem JavaScript a jeho výhodami jsou především statické typy, interface, enum a další (to vede k zachycení některých chyb již při kompilaci). Kompiluje se do JavaScriptu, takže jej lze bezmezně využívat na všech webových aplikacích ke zlepšení škálovatelnosti kódu.



Obrázek 2.1: Návrhový vzor MVVM

2.2.1 Návrhový vzor MVVM

Návrhový vzor MVVM, znázorněn na obrázku č. 2.1, staví na těchto třech hlavních částech:

- View – jeho cílem je vytvořit strukturu a formát zobrazení dat na obrazovku. Obsahuje logiku potřebnou pouze ke správnému zobrazení dat a formátování. Je propojený s ViewModel pomocí datových vazeb.

- ViewModel – implementuje atributy a hodnoty, na které se lze ve View navázat (anglicky data binding) a notifikuje View na změny v datech. Jeho hlavní účel je starat se o business logiku aplikace. ViewModel zároveň může aktualizovat Model a stará se o jeho zpřístupnění pro View, může také data z Modelů kombinovat pro snazší zobrazení ve View.

- Model – modely jsou nevizuální prvky aplikace, které reprezentují data této aplikace (například DTO objekty). Modely se často používají v kombinaci se službami a můžou obsahovat validace. [5]

2.2.2 Moduly (modules)

Moduly jsou funkční celky aplikace, které ji rozdělují na jednotlivé větší části a přispívají tak ke škálovatelnosti aplikace. Z velké části jsou na sobě nezávislé, ale samozřejmě umožňují mít mezi sebou jistou provázanost. Jsou to uzavřené celky, které definují komponenty, které pak lze použít v rámci tohoto modulu, dále také můžou importovat jiné moduly nebo exportovat komponenty či moduly. Exportované komponenty či moduly lze použít v modulu, který importuje tento původní modul. Speciální typ modulu je routovací modul, který se využívá pro navigaci uvnitř modulu. [6]

2.2.3 Komponenty (components)

Komponenty jsou menší celky modulů. Každá komponenta obsahuje aplikační logiku (ViewModel) a její HTML šablonu (View) společně s CSS styly. Jsou to obvykle části stránek nebo složité prvky UI, které se kvůli složitosti vyplatí vložit do vlastní komponenty. Komponenta může mít vstupní parametry a výstupní eventy, které využívá pro komunikaci se svou nadřazenou komponentou. Komponenty do sebe lze neomezeně vkládat. [6]

Komponenta se vkládá do HTML šablony pomocí speciálního HTML tagu, který definuje ve své konfiguraci. Na tento HTML tag se poté také navazují vstupní parametry a případně využívají její eventy. V HTML šabloně komponenty pak lze provádět s daty mnoho základních operací, jako například podmínky, cykly nebo využívat pipe operátor. [6]

2.2.4 Služby (services)

Každá služba má pevně daný a specifický účel. Komponenty delegují jisté činnosti na služby (například práce s daty, volání API apod.), které tak jsou do jisté míry další znovupoužitelné prvky aplikace. Služby jsou v komponentech dostupné přes dependency injection. [6]

2.2.5 Routing

Router je jeden ze základních prvků Angularu, který umožňuje navigaci z jednoho View do jiného. Router interpretuje URL v prohlížeči a pomocí pravidel ji váže na konkrétní komponentu. Každý modul tedy může obsahovat konfiguraci routeru, která váže URL (s možnými parametry v cestě) na konkrétní komponentu a zároveň může určovat minimální podmínky nutné pro zobrazení těchto cest (například uživatelské role). [6]

2.2.6 Pipe operátor

Důležitou součástí Angularu jsou pipe operátory, které se používají v šablonách HTML pro pohodlnou transformaci dat (například pipe operátor `date`, který se používá pro konverzi data do lidského formátu). Kromě využívání předdefinovaných pipe operátorů je velkou výhodou především možnost vytvářet vlastní pipe operátory, které pak lze v šablonách používat. [6]

2.3 Reaktivní programování

Reaktivní programování je paradigma, zabývající se proudy dat a šíření asynchronních změn v aplikaci. Část kódu vysílá události, zatímco se jiné části kódu můžou přihlásit k odběru těchto událostí. Tento přístup je výhodný především při volání API requestů, které díky asynchronním událostem nezdržují hlavní vlákno a tedy načítání zbytku aplikace. [7]

Implementace tohoto paradigmatu závisí na několika klíčových prvcích. Prvním z nich je „pozorovaný“ (anglicky observable nebo také subject), který podle svého nastavení vysílá data. Vysílání těchto dat může probíhat s určitou periodou, pouze jednou za životní cyklus (vhodné u API requestů) nebo jakkoliv jinak podle potřeby objektu. Dalším je „pozorovatel“ (observer), který se přihlásí k odběru dat u konkrétního pozorovaného a tímto způsobem je schopen registrovat změny dat. Posledním důležitým prvkem je „plánovač“ (scheduler), který je potřebný právě pro asynchronní část procesu a rozhoduje na kterých vláknech poběží pozorovaný a pozorovatel. [7]

Při implementaci tohoto principu je také důležité myslet na práci s více objekty zároveň a jejich interakci. Programátor totiž může chtít vytvářet složitější závislosti těchto objektů (například může požadovat aby se před voláním nějaké metody vždy dokončilo několik API volání). Může také chtít data určitým způsobem zpracovávat nebo vyvolat jisté chování v závislosti na dokončování událostí. [7]

2.3.1 RxJS

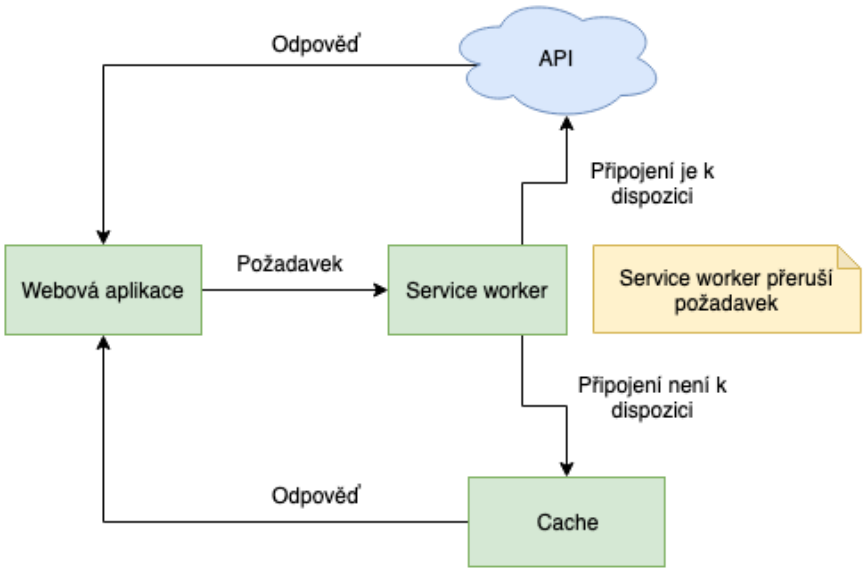
Reaktivní programování v poslední době nabývá na důležitosti a vzniká čím dál více knihoven, které jej podporují. Jednou z nich je knihovna RxJS napsaná v TypeScriptu. V Angularu se využívá především pro asynchronní API requesty, ale její použití není žádným způsobem omezeno pouze na práci s API. Pomocí speciálních funkcí umožňuje kombinaci a spojování událostí do jedné a také transformace dat, což se hodí nejen u složitější API requestů.

Programátor ji samozřejmě může využívat i mimo Angular a nejen na API requesty. Kromě klasického objektu observer obsahuje také několik dalších objektů typu subject, které umožňují jednoduché vytváření vlastních událostí. Pomocí metody objektu je pak možné změnit data v objektu subject, který se pak postará o předání této změny do všech na něm závislých pozorovatelů.

Díky tomuto principu může být některá část GUI závislá na určitých asynchronních datech, které jsou zase závislá na jiných asynchronních datech a změna v jakékoliv části se promítne do GUI, což automatizuje mnohé jinak složité procesy. Tento přístup zjednodušuje kód a zvyšuje jeho přehlednost.

2.4 Offline funkcionalita

Offline funkcionalita slouží pro podporu alespoň částečného fungování webové aplikace bez internetového připojení. To se může hodit jak na počítači, tak především v mobilních zařízeních. Podle nejnovějších standardů je možné řešit offline funkcionalitu webové aplikace pomocí technologie service workers. Starším způsobem byla aplikační cache, která mizí z webového standardu, a proto zdroje důrazně doporučují vyhnout se jejímu použití a místo ní použít technologii service workers, především protože nabízí mnohem širší možnosti práce s keší. [8]



Obrázek 2.2: Zjednodušená ukázka fungování offline funkcionality

2.4.1 Service workers

Service workers fungují jako virtuální proxy mezi prohlížečem a sítí. Příklad takového fungování je znázorněn na obrázku č. 2.2. V prohlížeči běží na odděleném vlákně a nemají přístup k DOM, mohou ovšem komunikovat se zbytkem kódu a lze jim tedy předat práci a následně získat výsledek této operace pomocí promise. Kromě toho se mohou také starat o PUSH notifikace. [9]

Používají se především pro podporu offline funkcionality a řeší problém kešování (tedy zrychlují aplikaci i pokud je uživatel online). Před načtením aplikace se do prohlížeče nainstaluje service worker, který může čekat na volání z jiné části kódu nebo kešovat API požadavky a další zdrojové soubory aplikace. [9]

2.4.1.1 Implementace v Angularu

Angular má vlastní implementaci service workers, která pomáhá vývojářům s jejich nastavením, díky čemuž není nutné programovat s nízkoúrovňovým API. Po nainstalování modulu do Angularu je potřeba nastavit funkcionalitu service workers. [6]

Aplikace se ukládá do keše jen jako celek (takže jsou všechny zdrojové soubory vždy načítány společně). Běžící aplikace se nezačne najednou aktualizovat, čeká se na nové načtení. Při novém načtení se zobrazí stará verze

aplikace a na pozadí se stáhnou nové zdroje, které se zobrazí až při dalším načtení aplikace. Toto chování zajišťuje nejvyšší možnou rychlost načítání. Jednotlivé zdroje jsou stahovány pouze v případě změn. [6]

Dále se do keše ukládají API requesty na základě definovaných pravidel. Existují dvě strategie pro ukládání requestů. První je založená na nejvyšší čerstvosti dat a pokud je dostupné připojení k internetu, tak se zdroj vždy načte nový. Druhá je založená na zrychlení aplikace a umožňuje na definovaný čas ukládat request, což může u velkých requestů přinést výrazné zrychlení. Je nutné pro jednotlivé endpointy vhodně zvolit strategii, ale ve většině případů je nejvhodnější zvolit strategii pro nejvyšší čerstvost zdroje. [6]

2.5 Progresivní webové aplikace

Na offline funkcionalitu navazuje další téma, a to progresivní webové aplikace (PWA). Takové aplikace fungují v prohlížečích, ale zároveň umožňují využívat funkcí mobilních telefonů a nabízí podobné výhody jako nativní mobilní aplikace. Pomocí technologie service wokers fungují i bez připojení k internetu a pomocí souboru web manifest říkají mobilním zařízením (a prohlížečům) údaje o aplikaci pro možné přidání na plochu zařízení a zobrazení splash screen.

Jednou z velkých nevýhod progresivních webových aplikací je slabá podpora na iOS zařízeních ze strany společnosti Apple. Technologie service workers sice funguje, ale instalace aplikace na plochu a její následné používání není tak snadné jako na platformě Android. Technickou nevýhodou je především nefunkčnost push nofitikací. [10]

2.5.1 Výhody

Progresivní webové aplikace mají následující výhody:

- Vyhledatelnost – měly by být vyhledatelné ve vyhledávačích a obsahovat soubor web manifest (který definuje název, ikonu, barvy a další).
- Instalovatelnost – uživatelé by měly mít možnost nainstalovat si aplikaci na plochu (tuto funkcionalitu opět zajišťuje na podporovaných platformách web manifest).
- Linkovatelnost – zachování možnosti orientovat se na webu pomocí odkazů.
- Nezávislost na připojení – měly by částečně fungovat (například v minulosti zobrazené zdroje by se měly opět zobrazit), přestože je síť pomalá, anebo když není zařízení připojené k internetu (k tomuto účelu slouží především technologie service workers).

- Progresivnost – využívání všech moderních funkcionalit webu s částečnou podporou starších prohlížečů.
- Responzivita – aplikace by se měla správně zobrazovat na různých velikostech obrazovky (a tedy různých zařízeních).
- Re-angažování uživatelů – uživatel by měl dostávat nový obsah i když zrovna nemá webovou aplikaci otevřenou (to lze zajistit například pomocí notification API).
- Bezpečnost – je důležité využívat HTTPS, ale také je nutné při vývoji aplikace myslet na její zabezpečení. [10]

Kapitola 3

Analýza

Tato kapitola se zabývá popisem stávajícího řešení (včetně klíčových procesů) a zejména jeho nedostatky. Navrhuji zde vhodná vylepšení a následně popisuji formální požadavky, případy užití a postup tvorby grafického návrhu pro novou webovou aplikaci.

3.1 Uživatelské rozhraní

Uživatelské rozhraní webové aplikace Grades z velké části není uživatelsky přívětivé a rozhodně má co zlepšovat. Dále se zaměřím a popíši problémy na jednotlivých klíčových stránkách. Základní principy, ze kterých vycházím jsou popsány v [11].

3.1.1 Navigace a horní lišta

Jedním z největších problému je navigace na levé straně. Uživatel má totiž kromě předmětů, které využívají klasifikaci, zobrazené také předměty bez klasifikace. Toto znepřehledňuje navigaci a uživateli to nepřináší žádné výhody. V navigaci má také zobrazenou dokumentaci, která ale většinu uživatelů nezajímá a tedy nemusí být v bočním menu. V navigaci by také neměly být položky „upozornění“ a „moje nastavení“, které je vhodné umístit do horní lišty, jak jsou uživatelé zvyklí z jiných webů.

Obě menu také místy obsahují drobné problémy se vzdáleností ikon a rozmístěním. Většina grafických prvků je příliš velká a webová aplikace tak zabírá velké množství prostoru a ztrácí na přehlednosti.

3.1.2 Domovská stránka

Během psaní práce ještě neexistovala nová podoba domovské stránky, která zobrazuje všechny předměty se známkami a celkovým počtem bodů. Nyní již

existuje, ale stále se zde zobrazují předměty s nedefinovaným hodnocením. Studenti by zde navíc ocenili zobrazení posledních ohodnocených testů.

3.1.3 Detail předmětu

Stránky s detailem jednotlivých předmětů pro studenty se také během psaní práce velmi změnily, ukázka této stránky během psaní mé práce je na obrázku č. 3.1. Dnes už jsou poměrně vyladěné a přehledné, vždy je ale co zlepšovat. V předchozím stavu obsahovaly četné chyby odsazení a další detaily, které ve výsledku zhoršovaly přehlednost zobrazených dat. Ve stávajícím řešení bych upustil od škálování barev v hodnocení, které nevypadá dobře.

Kromě toho bych do horní části obrazovky umístil pro studenty důležité informace jako známku z předmětu, celkový počet bodů, stav zápočtu a nejlépe i souhrnné body ze semestru a ze zkoušky. Vše by mohlo být graficky odlišené tak, aby byly informace přehledné a na první pohled viditelné.

3.1.4 Hromadné hodnocení předmětu

Dále se dostáváme k části určené pro učitele. V tuto chvíli vynechávám v práci definice klasifikace předmětu, na které se mé řešení nezaměřuje. V učitelském podmenu u jednotlivých předmětů je mnoho položek, které by bylo možné eliminovat. Všechny tyto položky lze elegantně skrýt do jednoho celku, což zjednoduší orientaci v aplikaci a zároveň zjednoduší navigaci. Takovým řešením by bylo například na začátku navigovat učitele do sekce pro hodnocení skupin studentů, kde si mohou pomocí jednoduchého filtru zobrazit pouze konkrétní položku hodnocení (pokud chtějí hodnotit pouze ji), anebo při kliknutí na studenta hodnotit tohoto studenta, místo několika stránek přístupných z levého menu.

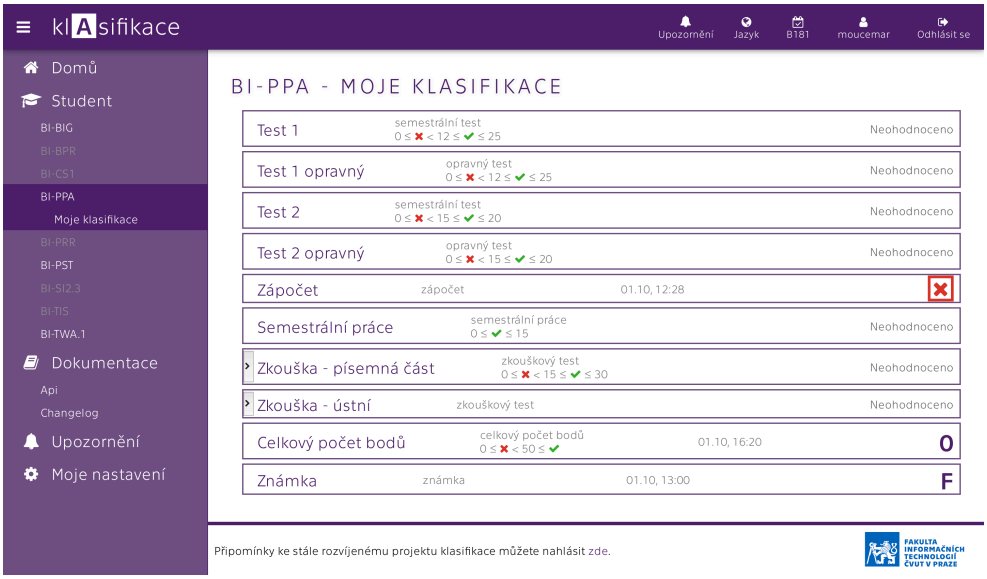
Všechny tabulky hodnocení jsou u stávající řešení příliš velké. Na notebooku není velká část tabulky zobrazená a uživatelé musí značně scrollovat, což zvyšuje jejich nepohodlí. Tabulka by měla být o něco menší a přehlednější.

3.2 Technologie

Z technického hlediska je se současným řešením problém především se starými technologiemi, kód je špatně udržovatelný a není příliš přehledný, což mimo jiné způsobuje problémy s přidáváním nových funkcí.

Řešení je napsáno ve frameworku AngularJS a tedy využívá standardní JavaScript místo alternativního jazyka TypeScript.

Další nedostatek je způsob tvorby položek v levém menu. O výstavbu celého menu se stará API endpoint a to včetně URL adres, ikon a kompletních textů, všechna tato data se získají během jediného volání. Vzhledem k obsahu menu toto rozhodně není správná strategie a snižuje udržitelnost celé aplikace.



Obrázek 3.1: Stránka s hodnocením předmětu v současném řešení

3.3 Procesy

V této sekci se zabývám vylepšením dvou nejdůležitějších procesů ve webové aplikaci Grades, které jsou hodnocení předmětu a zobrazení hodnocení.

3.3.1 Proces hodnocení předmětu

V této části se zaměřím na proces ohodnocení předmětu, který lze efektivně vylepšit využitím offline funkcionality.

3.3.1.1 Stávající řešení

Učitel přejde na stránku hodnocení předmětu, kde si vybere skupinu, kterou se chystá hodnotit. Následně zadává hodnocení studentů do tabulky a poté má možnost tyto změny uložit. Pokud se tak rozhodne, odesílá se požadavek na server s novým aktualizovaným hodnocením studentů.

Pokud během hodnocení vypadne připojení k internetu, nebo dojde k jiným problémům, učitel ztrácí svá data. Je potřeba brát v úvahu, že takových dat může být velké množství a jejich ztráta přidá učitelům práci navíc. Učitel navíc nemá přehled o tom, jaké změny v tabulce udělal, protože tabulka zvýrazňuje i položky, kterým byla navrácena původní hodnota.

3.3.1.2 Nové řešení

Proces začíná vstupem na stránku hodnocení předmětu, kde si učitel vybere skupinu, kterou chce hodnotit. Pokud učitel v minulé relaci studenty hodnotil,

ale spojení bylo přerušeno nebo odešel bez uložení či zrušení, tak se mu načtou jeho předchozí hodnocení s upozorněním ,že se tato data načetla a zároveň jsou tato hodnocení v tabulce graficky odlišena.

Po načtení skupiny či dat může učitel změny buď zrušit, nebo uložit. Pokud mu v této chvíli (nebo kdykoli během hodnocení studentů) vypadne připojení k internetu, aplikace bude dále fungovat, pouze ho při pokusu odeslat dat upozorní na skutečnost, že pro odeslání dat musí být připojen k internetu. Jak již bylo řečeno, data se automaticky ukládají, takže v této chvíli neztratí svá zadaná hodnocení.

Pokud je připojení k internetu aktivní, tak se po kliknutí na tlačítko uložit odesílají aktualizovaná data na server, kde se zpracují.

O fungování aplikace v offline režimu se stará technologie service workers, které lze z funkčního/uživatelského hlediska přirovnat ke cache. Service workers ukládají zdrojové soubory aplikace a podle nastavení také ukládají HTTP požadavky.

3.3.2 Proces zobrazení hodnocení

Proces zobrazení hodnocení provádí student, kterého zajímá známka nebo zpravidla nejnovější hodnocení z předmětu.

3.3.2.1 Současný stav

Student vybere z menu předmět, ke kterému ho zajímá hodnocení, případně klikne na novou notifikaci. Předmět se mu otevře a student musí v nepřehledných hodnoceních hledat důležité údaje (známka, celkový počet bodů, ...) a pokud ho zajímá nejnovější hodnocení, musí ho pracně manuálně hledat. Pokud není student připojen k internetu, aplikace se ani neotevře.

3.3.2.2 Nové řešení

Jakmile student otevře webovou aplikaci Grades, vidí zde poslední hodnocení ze všech navštěvovaných/studovaných předmětů (nemusí tedy vůbec chodit na jednotlivé stránky předmětů), zároveň hned vidí důležité údaje jako je například známka, celkový počet bodů a další. Pokud studenta zajímá nějaké jiné hodnocení, může přejít na stránku předmětu podobně jako u stávajícího řešení. Pokud není student připojen k internetu, zobrazí se mu hodnoty načtené z poslední relace.

3.4 REST API

Problémem současného řešení (ať už pro webovou nebo mobilní aplikaci) je také REST API, které pro některé obrazovky nemá vhodné endpointy. Tudíž

je na některých obrazovkách nutné volat několik endpointů a v některých případech (například na obrazovce s hodnocením studentů) i data transformovat. REST API navíc obsahuje konceptuální chyby.

Při vhodném návrhu API by připojené aplikace měly provádět co nejméně požadavků (samozřejmě se zachováním přehlednosti cest a logiky) a neměly by být nuceny data zpracovávat. V části realizace jsou navrženy vhodná vylepšení.

3.5 Formální požadavky

Po analýze stávajícího řešení byly identifikovány následující funkční a nefunkční požadavky (některé se částečně prolínají se stávajícím řešením).

3.5.1 Funkční požadavky

1. Autorizace uživatelů – portál musí umožňovat přihlášení přes autorizační server FIT ČVUT.
2. Změna semestru – uživatel má možnost zvolit si semestr, ze kterého se zobrazují data. V této části bude mít také k dispozici zkratky pro rychlé přepnutí na minulý, aktuální či příští semestr.
3. Jazykové mutace – webová aplikace bude dostupná ve více jazykových variantách (čeština, angličtina) s možností jednoduché a rychlé změny jazyka.
4. Notifikace (upozornění) – uživatel bude na webu v reálném čase upozorněn na nové hodnocení v předmětu. To ale bude fungovat pouze pokud má otevřený webový portál. Může si také zobrazit svá poslední upozornění a při kliknutí na upozornění se toto upozornění změní na přečtené. Uživatel má možnost označit všechna upozornění jako přečtená.
5. Ztlumení předmětu – v nastavení má uživatel možnost ztlumit upozornění jednotlivých předmětů nebo vypnout zasílání emailových upozornění.
6. Přehled – stránka s prezentací studijních výsledků pro předmětů ve zvoleném semestru. U každého předmětu se zobrazí známka, zápočet a celkový počet bodů. Zároveň se u něj budou zobrazovat poslední hodnocení pro rychlou orientaci v předmětech.
7. Stránka předmětu – uživatel má možnost přehledně vidět jednotlivá hodnocení v předmětu. Pro lepší orientaci má zde odlišně zobrazený celkový počet bodů, známku, zápočet a body ze semestru.
8. Stránka hodnocení předmětu – učitelé zde mají možnost hodnotit studenty v tabulce. Po zvolení skupiny studentů může učitel měnit hodnocení studentům (změněná hodnocení jsou pro přehlednost graficky

odlišená), které se průběžně ukládají. Může také vyfiltrovat pouze jedno konkrétní hodnocení. Učitel následně změny buď uloží, nebo zruší. V případě opuštění stránky se změny uloží a v příštím otevření se zase načtou. Po kliknutí na studenta se zobrazí hodnocení konkrétního studenta.

9. Stránka hodnocení studenta – na této stránce učitel hodnotí pouze konkrétního studenta.

3.5.2 Nefunkční požadavky

1. Podpora prohlížečů – aplikace bude podporovat nejnovější verze prohlížečů Chrome, Firefox, Safari a Edge.
2. Framework Angular – aplikace bude napsána ve frameworku Angular 7 a bude dodržovat jeho standardy. Angular je implementován v jazyce TypeScript, který bude samozřejmě využit i při vývoji aplikace.
3. Fungování bez připojení k internetu – v případě výpadku či absence připojení k internetu bude aplikace částečně fungovat (za předpokladu, že ji uživatel v minulosti spustil a aplikace měla možnost se alespoň jednou načíst). Možnosti offline využívání aplikace budou samozřejmě patřičně omezené.
4. PWA – aplikace bude progresivní a bude ji možné přidat na plochu a spouštět podobně jako nativní aplikaci v podporovaných OS (v současnosti nabízí plnou podporu pouze OS Android).
5. Responzivita – webová aplikace bude plně responzivní a bude tedy podporovat zobrazení jak na počítači, tak na mobilních zařízeních.

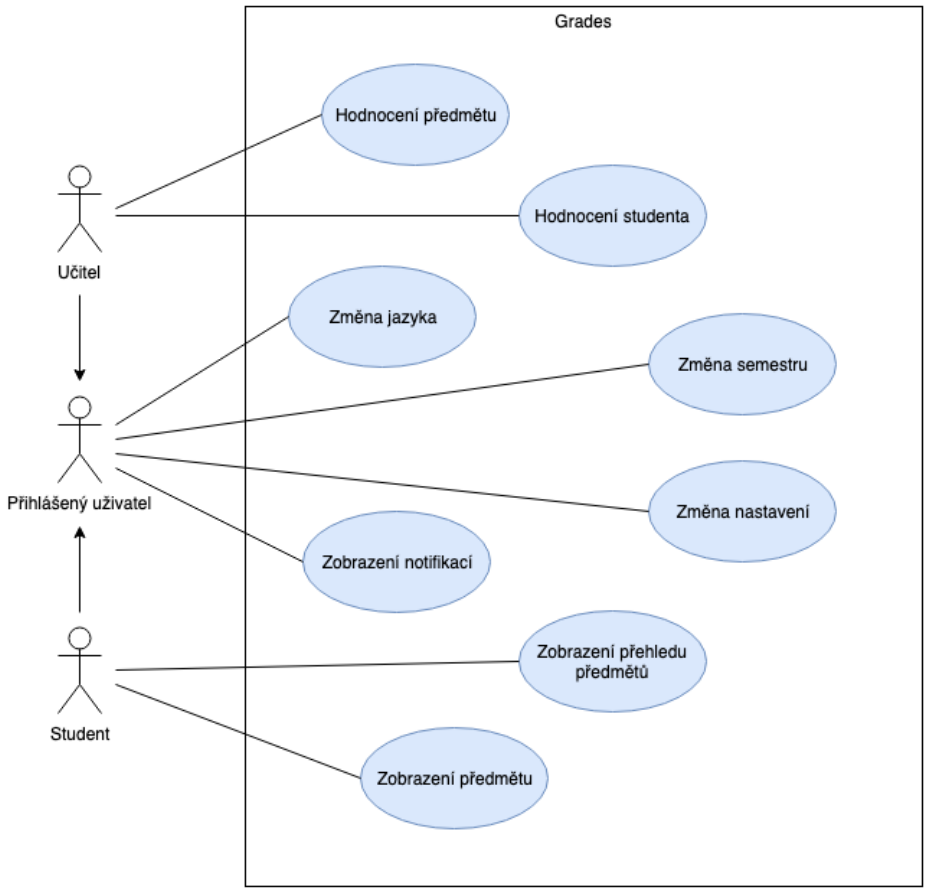
3.6 Případy užití

3.6.1 Seznam účastníků

Účastníci mohou nabývat následujících rolí, přičemž každý účastník má přístup pouze k určitým sekcím v aplikaci:

- Přihlášený uživatel,
- student,
- učitel.

Je nutno poznamenat, že role student a učitel jsou role pouze ve vztahu k předmětům, nikoli v rámci celé aplikace. Ve stávající verzi Grades existují navíc ještě role editor, garant, vedoucí katedry, katederní rozvrhář a admin. Těmito rolemi se nebudu dále zabývat, protože jimi přístupné sekce jsou nad rámec zadání této práce.



Obrázek 3.2: Diagram případů užití

3.6.2 Případy užití

Obrázek 3.2 popisuje interakce uživatelů s Grades. Interakce jsou následující:

- Změna nastavení – přihlášený uživatel si může změnit nastavení notifikací pro aktuální semestr. Může si ztlumit notifikace pro jednotlivé předměty (jako učitel i student) a může si odhlásit odběr emailů.

- Změna semestru – přihlášený uživatel si může změnit semestr, ze kterého se zobrazují data. V případě, že si uživatel toto nastavení nezmění, automaticky se mu zobrazuje aktuální semestr.

- Změna jazyka – přihlášený uživatel má také možnost změnit si jazyk, pokud mu nevyhovuje automatické nastavení jazyka z prohlížeče.

- Zobrazení notifikací – přihlášený uživatel si může zobrazit poslední nebo všechny notifikace. Po kliknutí na notifikaci se mu zobrazí předmět, kterého se notifikace týká. Má také možnost označit všechny notifikace jako přečtené.

- Zobrazení přehledu předmětů – studentovi se při vstupu do klasifikace zobrazí přehled, kde bude mít zobrazené předměty, které aktuálně studuje. U nich uvidí známku, zápočet, celkový počet bodů a poslední hodnocení předmětu, díky čemuž nebudou studenti nuceni přecházet do stránek předmětu. Je zde samozřejmě také možnost přejít na tento předmět.

- Zobrazení předmětu – po výběru předmětu v menu (nebo z odkazu na úvodu) si student zobrazí klasifikace k předmětu. Položky jako známka, celkový počet bodů, zápočet a počet bodů ze semestru jsou na první pohled viditelné a odlišené. Dále má zde student možnost procházet všechna hodnocení a případně si k nim zobrazit dodatečné informace jako například datum zhodnocení, rozsah a poznámku (pokud existují).

- Hodnocení předmětu – po výběru učitelského předmětu v menu si učitel zobrazí hodnocení předmětu. Následně si vybere skupinu, kterou chce hodnotit a zobrazí se mu tabulka pro hodnocení této skupiny. Může také hodnotit pouze jeden test při výběru z menu nebo vyhledat konkrétního studenta. Tohoto studenta se může rozhodnout hodnotit a přejít do hodnocení studenta. Učitel na první pohled vidí položky, které editoval a při ztrátě nebo opuštění stránky se může pohodlně vrátit do této sekce bez ztráty dat.

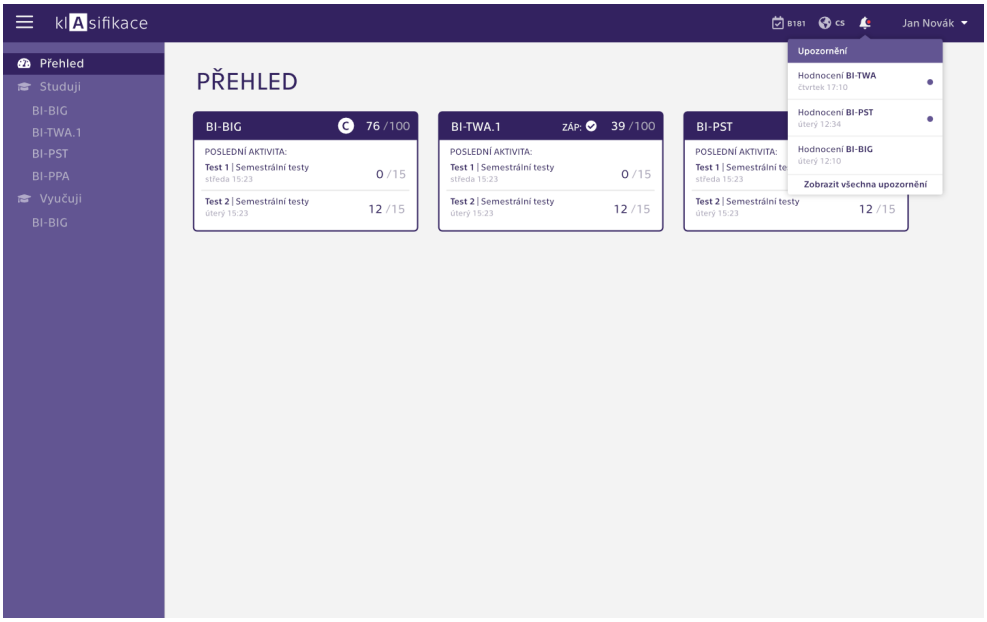
- Hodnocení studenta – učitel si přehledně zobrazí položky hodnocení pro konkrétní studenta i s graficky zobrazeným rozsahem bodování. Může také k jednotlivým hodnocením přidávat poznámku.

3.7 Návrh uživatelského rozhraní

Na základě procesů, požadavků a případů užití jsem navrhl vzhled uživatelského rozhraní. Vzhledem k již existující verzi Grades jsem při návrhu rozhraní postupoval agilně. Byl vytvořen mockup pro tři zásadní obrazovky, který určuje celkový vizuální styl aplikace, a další obrazovky byly vytvořeny přímo v prohlížeči ([12]).

Výhodou tohoto postupu je především možnost prezentovat vzhled rozhraní v plně funkční podobě. Následné drobné úpravy vzhledu lze velmi snadno implementovat a poté konzultovat s týmem.

Oproti mockupu je v implementované aplikaci výraznější změna především v barevnosti celé aplikace. Nakonec jsem se rozhodl zvolit modrou barvu, která Grades lépe váže s celkovým vizuálním stylem ČVUT [13]. Předchozí barva



Obrázek 3.3: Návrh stránky přehledu

neměla dobré opodstatnění. Dále v textu budou popsány a vysvětleny klíčové obrazovky.

3.7.1 Navigace a horní lišta

V horním liště má student možnost změnit semestr, jazyk, zobrazit si notifikace a po kliknutí na své jméno přejít do nastavení nebo se odhlásit.

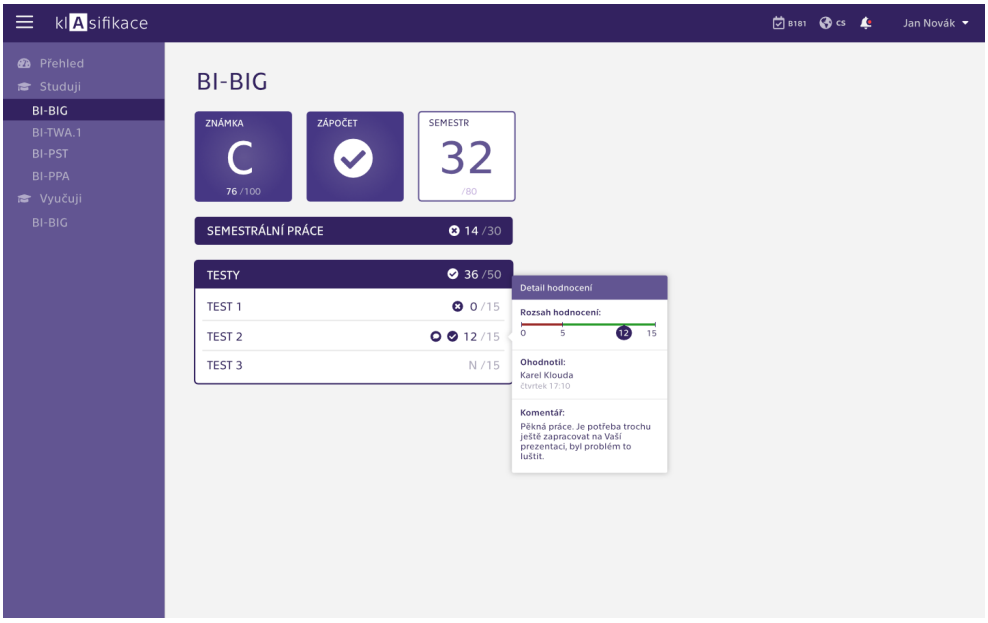
Hlavní navigace na levé straně obsahuje odkaz na přehled, studentské předměty a učitelské předměty.

3.7.2 Přehled

Stránka přehledu obsahuje přehled studentských předmětů. U každého předmětu se zobrazuje známka, zápočet, celkový počet bodů, 2 poslední položky hodnocení (díky čemuž se student nemusí proklikávat na stránku předmětu) a odkaz na předmět. Návrh této obrazovky je na obrázku č. 3.3.

3.7.3 Detail předmětu

Detail předmětu studentům výrazně zobrazí známku, celkový počet bodů, zápočet a body ze semestru. Dále obsahuje jednotlivé položky hodnocení, které lze rozbalit a zobrazovat podpoložky. U položek se zobrazují různé údaje podle typu hodnocení. Po najetí na položku se zobrazí detail hodnocení, kde je

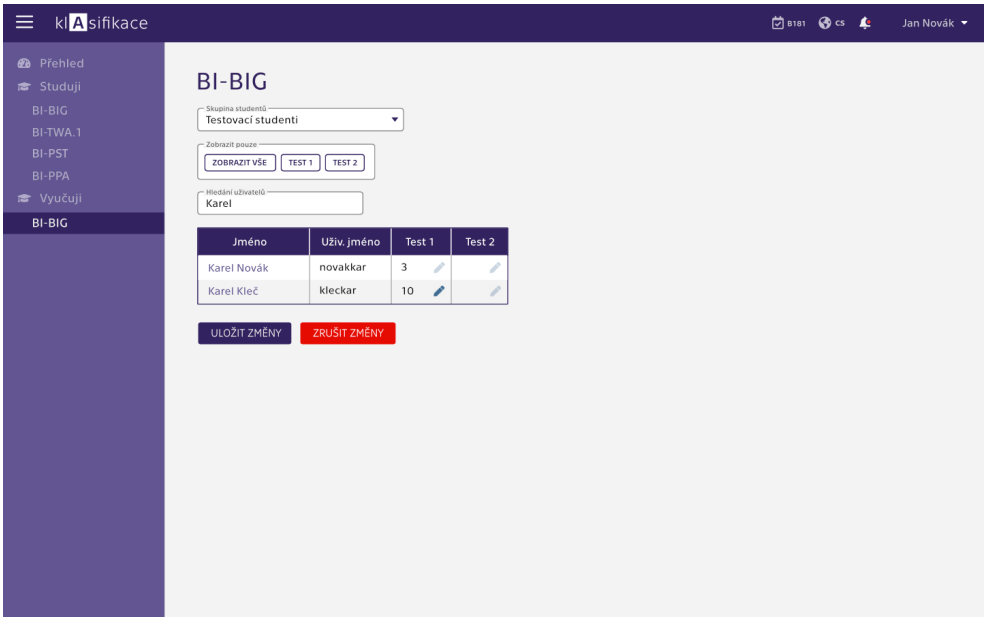


Obrázek 3.4: Návrh stránky detailu předmětu

graficky zobrazen rozsah bodování, datum hodnocení a poznámka (zobrazují se pouze dostupné údaje). Návrh této obrazovky je na obrázku č. 3.4.

3.7.4 Hromadné hodnocení předmětu

V horní části má učitel možnost vybrat si skupinu studentů, kterou chce hodnotit. Dále má k dispozici jednoduchou filtraci, která mu umožní hodnotit pouze jednu položku hodnocení. Poté je zde zobrazena tabulka (ve které lze studenty vyhledávat) s grafickým znázorněním editovaných položek. Jméno studenta a hlavička tabulky jsou vždy viditelné. Dole jsou k dispozici tlačítka pro uložení a zrušení změn. Návrh této obrazovky je na obrázku č. 3.5.



Obrázek 3.5: Návrh stránky hodnocení předmětu

Kapitola 4

Realizace

4.1 Architektura

Při návrhu architektury jsem vycházel z doporučení na stránkách frameworku Angular a také využívám celky popsané v kapitole rešerše. Celá architektura staví na modulech a její hlavní části jsou zobrazené na obrázku č. 4.1. Vycházím z následujících typů modulů, které jsou dopodrobna popsány dále v textu:

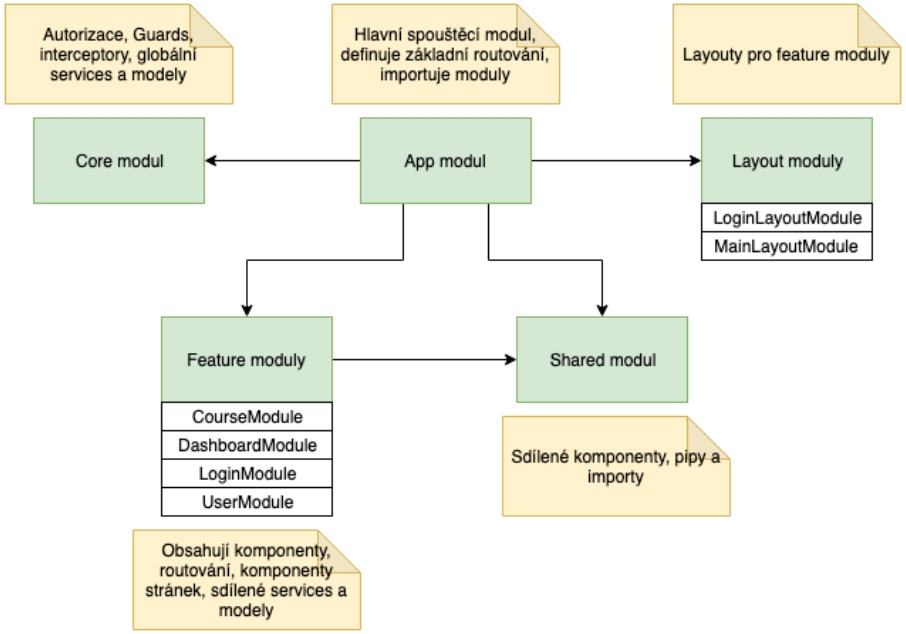
- App modul,
- Core modul,
- Shared modul,
- Layout moduly,
- Feature moduly.

4.1.1 App modul

App modul je základní modul v frameworku Angular, který se v aplikaci zavádí jako první. Importuje externí moduly důležité pro správné fungování aplikace (například modul pro dynamické překlady nebo toast zprávy), layout moduly, feature moduly (pouze pokud není využíván lazy loading) a shared modul. Obsahuje a importuje také hlavní router modul, který přiřazuje layout moduly a hlavní cesty jednotlivým modulům, ze kterých pak načítá jejich routovací moduly.

4.1.2 Core modul

Core modul se stará o autentizaci a autorizaci uživatele, definuje interceptory (v našem případě pouze `AppErrorHandler`, který zpracovává všechny chyby



Obrázek 4.1: Architektura aplikace

z aplikace, především pak HTTP chyby) a guardy pro použití v routerech. Dále obsahuje globální services a modely, které nelze přiřadit do žádného konkrétního feature modulu.

4.1.3 Shared modul

Shared modul je využíván pro definici sdílených prvků pro celou aplikaci. Definuje a exportuje sdílené komponenty a operátor pipe, které jsou pak využívány ve feature modulech. Může také definovat sdílené třídy, které nelze přiřadit do konkrétního feature modulu.

4.1.4 Layout moduly

Layout moduly jsou skupina modulů, které definují layout pro určitou část aplikace, kde se pak dále v určité části zobrazují stránky z feature modulů. Layout moduly jsou importovány a využívány v app modulu. Aplikace obsahuje dva layout moduly a to:

- `LoginLayoutModule` – layout pro obrazovku přihlášení,
- `MainLayoutModule` – layout pro obrazovky v hlavní části aplikace.

4.1.5 Feature moduly

Feature moduly jsou skupina modulů, které vytvářejí určité funkční části aplikace. Definují především komponenty, které slouží pro zobrazení jednotlivých stránek a používají se v routeru, pomocné komponenty v rámci toho modulu a také sdílené services a modely. Obsahují router modul, který slouží pro navigaci v rámci tohoto feature modulu. V aplikaci existují následující feature moduly:

- `CourseModule` – modul zajišťující funkci klasifikace pro studenty a také editaci klasifikace předmětu a studenta pro učitele,
- `DashboardModule` – modul obsahující funkci přehledu,
- `LoginModule` – modul zajišťující funkci přihlášení,
- `UserModule` – modul obsahující funkci pro uživatelské nastavení a funkce spojené s uživateli (nastavení, notifikace).

4.2 Offline funkcionalita

Framework Angular nabízí balíček pro zjednodušenou práci s technologií service workers, který je ale nutné vhodně nakonfigurovat. Pro všechny endpointy jsem zvolil strategii nejvyšší čerstvosti, protože za normálních okolností by měla být všechna data nejnovější.

Balíček neřeší requesty, které se odesílají na server, protože odeslat data po připojení k internetu často nemusí být vhodné chování. V případě nové aplikace Grades toto není požadované chování, protože se hodnocení můžou mezitím změnit. Tato situace, ale musí být nějakým způsobem řešena, aby uživatel neztratil svá data.

V nové aplikaci Grades je tato situace vyřešena implementací automatického ukládání rozepsaných hodnocení. Po připojení k internetu má učitel možnost rozhodnout se, zda chce data uložená z předchozí relace poslat na server nebo změny kvůli neaktuálnosti zrušit.

4.3 Doporučené úpravy REST API

V návrhové sekci jsem naznačil nutnost úprav REST API pro zrychlení frontendu a pro zjednodušení práce s daty. Navíc byly v REST API nalezeny koncepční chyby, které zde popíši a následně navrhnu vhodná řešení.

V případě situací nevhodných endpointů byla na frontendu, za účelem vytvoření funkčního prototypu, vytvořena snadno odstranitelná dočasná řešení.

4.3.1 Chybové zprávy

V současné době nejsou v REST API vhodně vyřešeny chybové zprávy. Při chybách se na serveru nastavuje speciální hlavnička `message-type` na konkrétní hodnotu a v response se poté vrací pouze ID k překladu. Standardní postup pro API by bylo posílat json s informacemi o chybě doplněnými o ID k překladu [14]. Zároveň jsem našel zvláštní případ použití HTTP kódu 400 (bad request). Při nedostupnosti KOS API posílá Grades API kód 400 společně s ID k překladu jako při ostatních chybách. V tu chvíli bych ovšem očekával kód 500 (internal server error), protože v tomto případě není chyba v požadavku u klienta, ale na serveru.

4.3.2 Autorizace uživatelů

REST API porušuje bezstavovost, protože se o celý proces autorizace přes OAuth 2 stará pouze server. Klient (frontend) pouze přesměruje uživatele na adresu backendu, který následně uživatele přesměruje na bránu OAuth, se kterou komunikuje přímo backend. Backend si OAuth token udržuje v session u jednotlivých uživatelů.

Po vypršení této session neodpovídá server na další requesty HTTP kódem 403 (forbidden), ale pouze přesměrováním. Pro kontrolu stavu přihlášení je na serveru vyhrazen endpoint, který ale není vhodné periodicky volat, navíc je to nestandardní postup. V novém frontendu bude tato situace patřičně vyřešena po implementaci OAuth na backendu.

Doporučeným řešení je aby si klient sám obstaral OAuth token od autorizačního serveru a poté tento token přikládal k jednotlivým voláním endpointů. Po vypršení platnosti tokenu by měly další neautorizované requesty selhat chybovým kódem HTTP 403.

4.3.3 Menu

V současném frontendu se vytváří menu z API jako celek. To znamená, že se pošle celá struktura menu, včetně ikon, adresy odkazů a všech podpoložek. Některé základní položky v menu (jako například přehled) není nutné posílat z API vůbec, stejně tak není nutné posílat odkazy ani ikony. Toto není vhodné řešení a vzhledem k úpravám menu v nové aplikaci, bylo nutné toto obejít a dočasně vytvářet menu z několika endpointů.

Nový frontend zobrazuje pouze studentské předměty s definovanou klasifikací (vynechává předměty, které má student zapsané a zároveň nemají v Grades definovanou klasifikaci). Studentské předměty s definovanou klasifikací se v novém menu získávají z tohoto endpointu pro celé menu. Učitelské předměty se získávají z endpointu pro uživatelské role, který sice obsahuje i studentské předměty, ale bohužel pouze všechny zapsané předměty. Toto je pouze dočasné řešení a na REST API bude stačit menší úprava – přidat do tohoto endpointu studentské předměty pouze s definovanou klasifikací.

4.3.4 Úvodní obrazovka

Během psaní práce neexistoval žádný endpoint pro vytvoření obrazovky přehledu. V současné době už existuje, ale zobrazuje i předměty, které nemají definice a navíc neobsahuje položky nejnovějšího hodnocení, které se v novém přehledu zobrazují.

Jako dočasné řešení je nejprve nutné získat předměty s definovanou klasifikací pomocí získávání dat popsaným výše v části menu a poté volat endpoint pro každý předmět, kde se získají všechna hodnocení, která se následně řadí podle data a pak se z nich získají dvě nejnovější hodnocení. Je zjevné, že vytvořením vhodného endpointu, který by u každého předmětu posílal celkový počet bodů, známku, zápočet a určitý počet nejnovějších hodnocení, by bylo možné na úvodní stránce ušetřit několik requestů a čas strávený zpracováním dat a tím výrazně zrychlit její načítání.

4.3.5 Stránka předmětu

Současný endpoint není pro potřeby této nové stránky vhodný, protože by měl navíc vracet známku, zápočet, celkový počet bodů a body ze semestru. Server navíc zasílá pouze kolekci všech hodnocení a frontend je musí seřadit a vytvořit celou vnořenou strukturu u položek hodnocení, což je koncepčně špatně. Vhodným řešením by bylo zasílat z API již vytvořenou strukturu a tím u klienta ušetřit čas strávený zpracováním dat a tvorbou struktury.

Kapitola 5

Testování uživatelského rozhraní

V této kapitole se zaměřím na testování uživatelského rozhraní pomocí metody testování použitelnosti (testování s reálnými uživateli) a automatického testování.

5.1 Testování použitelnosti

Cílem testování použitelnosti je hledání problémových částí v uživatelském rozhraní aplikace s reálnými uživateli, které by jiným způsobem nemusely být odhaleny.

Základem testovaní použitelnosti je kvalitní příprava. Nejprve je nutné připravit nejdůležitější testovací scénáře, kterými budou uživatelé podrobeni. Následně je potřeba identifikovat cílové skupiny uživatelů a z každé skupiny jich několik vybrat pro testování. Testovaným osobám stačí pouze notebook s aplikací a testovací scénáře, které mají vykonat.

Po započetí testování by neměl moderátor do jeho průběhu nijak zasahovat, měl by pouze sledovat osoby (výraz, pohyby, pauzy) a případně si psát poznámky. Po dokončení testování by měl moderátor aplikaci projít s uživateli a zapsat si výsledek testování a případně také subjektivní pocity uživatelů.

Pro účely testování použitelnosti jsem připravil testovací protokol (na obrázku 5.1), společně s několika testovacími scénáři. Vybral jsem 6 klíčových uživatelů v následujících skupinách:

- Noví uživatelé – 2 uživatelé, kteří nikdy nepoužívali Grades a nejsou tedy z naší fakulty, ale jsou studenti jiných škol.
- Studenti – 2 uživatelé, kteří používají Grades v roli studenta na naší fakultě.
- Učitelé – 2 uživatelé, kteří používají Grades v roli učitele na naší fakultě.

5.1.1 Testovací scénáře

Pro účely uživatelského testování jsem připravil 6 testovacích scénářů. Přestože jsou scénáře fakticky vázané na role v aplikaci Grades, tak všichni uživatelé budou provádět všechny scénáře. Každý uživatel je totiž v rámci aplikace důležitý a může přispět svou odezvou i na role, které dosud nebyly v jeho kompetenci. Pro uživatele starší verze aplikace totiž mohou být některé scénáře snazší, protože aplikaci znají. Všechny tyto scénáře jsou popsané v tabulce č. 5.1.

Scénáře č. 1 a č. 2 jsou zaměřeny na obecné ovládaní aplikace a jejího nastavení. Scénáře č. 3 a č. 4 jsou zaměřeny na studenty – tedy především na zobrazení hodnocení a využívání nové funkce přehledu. Scénáře č. 5 a č. 6 jsou zaměřeny na učitele a zabývají se tedy hodnocením skupiny a studentů.

Tabulka 5.1: Testovací scénáře

Číslo scénáře	Název scénáře	Popis scénáře
1	Změna nastavení	Zobrazte si Vaše uživatelské nastavení a ztlumte si libovolný předmět.
2	Změna semestru	Změňte semestr na minulý semestr.
3	Zobrazení hodnocení posledního testu	Zjistěte hodnocení posledního testu z libovolného předmětu. Pokuste se ho zjistit co nejjednodušší cestou.
4	Zobrazení předmětu	Zobrazte si libovolný předmět a u libovolného testu zjistěte minimální počet bodů pro splnění tohoto testu (najděte test, který má stanovený minimální počet bodů).
5	Hodnocení skupiny	Vyberte si libovolný předmět, zvolte skupinu testovací studenti, zobrazte si pouze jeden konkrétní sloupec, vyplňte alespoň 3 hodnocení a změny uložte.
6	Hodnocení studenta	Vyberte si libovolný předmět a následně libovolného studenta ohodnoťte v podrobném hodnocení. Přidejte mu alespoň jedno hodnocení s libovolnou poznámkou.

5.1.2 Testovací protokol

Po dokončení testovacího scénáře je celý průběh vyhodnocen moderátorem a zároveň je uživatel požádán o odezvu. U každého scénáře se hodnotí zda byl

splněn podle očekávání, jak dlouho trvalo jeho splnění a případné poznámky moderátora či uživatele.

V další části má uživatel prostor pro subjektivní hodnocení a obecné odezvy nad rámec testovacích scénářů. Uživatel si aplikaci podle libosti může projít i nad rámec testovacích scénářů a může do této části zapsat svůj názor. Pokud chce může také aplikaci porovnat s aktuální verzí Grades. Protokol připravený pro vyplnění je k vidění na obrázku č. 5.1.

5.1.3 Průběh testování

V této části popíši průběh jednotlivých scénářů a nalezené problémové oblasti u kterých lze také uvažovat o budoucím zlepšení.

5.1.3.1 Scénář č. 1 – změna nastavení

Tento scénář proběhl u všech uživatelů aplikace bez problémů. Uživatelé ho prováděli velmi rychle, průměrný čas byl 12 sekund.

5.1.3.2 Scénáře č. 2 – změna semestru

Scénář zvládli všichni uživatelé velmi rychle. Jeden uživatel, který nikdy Grades nepoužíval, hledal semestr v nastavení, později to zdůvodnil tím, že na jeho škole používají jiné značení semestru. Průměrný čas dokončení byl 7 sekund.

5.1.3.3 Scénáře č. 3 – zobrazení hodnocení posledního testu

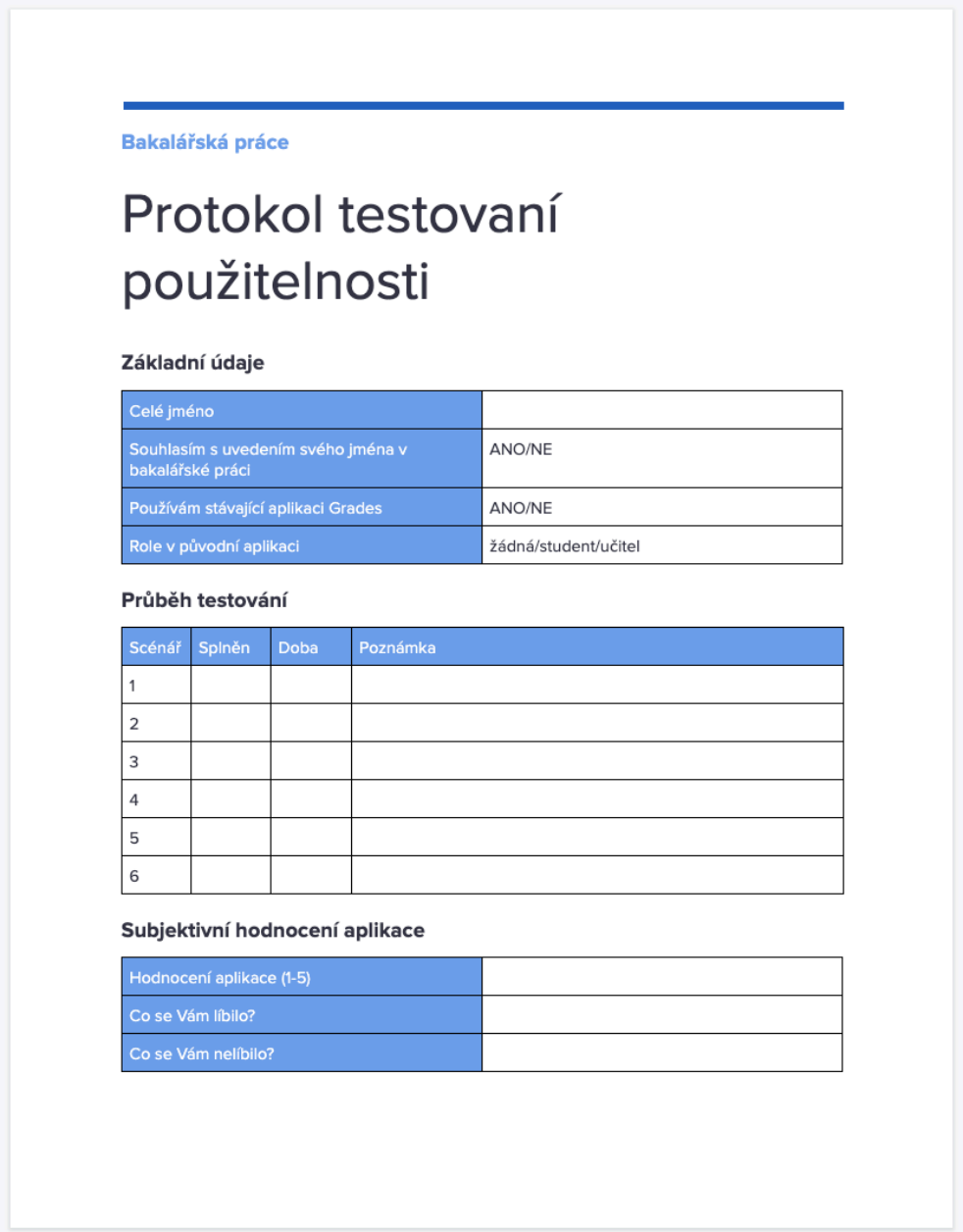
Tento scénář zvládali uživatelé poměrně rychle. Pouze jeden učitel hledal hodnocení posledního testu u předmětu, místo aby ji hledal přímo na přehledu. Průměrný čas dokončení byl 19 sekund.

5.1.3.4 Scénáře č. 4 – zobrazení předmětu

Většina uživatelů zvládla tento scénář bez problémů. Dva uživatele chvíli hledali, kde je možné zobrazit detail hodnocení testu. Průměrný čas dokončení byl 15 sekund.

5.1.3.5 Scénáře č. 5 – hodnocení skupiny

Scénář obecně probíhal dobře. Polovina uživatelů najela kurzorem nejprve na tlačítko zrušit změny. Jeden učitel se dostal do hodnocení studenta a zkoušel studenta ohodnotit klikáním na grafiku rozsahu. Jeden uživatel zkoušel uložit výsledky pomocí tlačítka enter. Průměrný čas dokončení byl 36 sekund.



Bakalářská práce

Protokol testovaní použitelnosti

Základní údaje

Celé jméno	
Souhlasím s uvedením svého jména v bakalářské práci	ANO/NE
Používám stávající aplikaci Grades	ANO/NE
Role v původní aplikaci	žádná/student/učitel

Průběh testování

Scénář	Splněn	Doba	Poznámka
1			
2			
3			
4			
5			
6			

Subjektivní hodnocení aplikace

Hodnocení aplikace (1-5)	
Co se Vám líbilo?	
Co se Vám nelíbilo?	

Obrázek 5.1: Protokol testování použitelnosti

5.1.3.6 Scénáře č. 6 – hodnocení studenta

Scénář probíhal dobře, ale mnoho uživatelů klikalo v tabulce hodnocení předmětu na ikonu tužky. Později to zdůvodnili tím, že zde očekávali možnost přidání poznámky. I přesto byl průměrný čas dokončení pouze 36 sekund.

5.1.4 Shrnutí testování použitelnosti

Aplikace byla uživateli přijata velmi dobře a všechny scénáře byly splněny v rozumném čase. Přesto byly nalezeny určité problémové oblasti k budoucímu zlepšení.

Bylo vidět, že všichni studenti byly o něco rychlejší ve scénářích pro učitele. To si zdůvodňuji tím, že učitelé jsou zvyklí na starou verzi aplikace.

Uživatelé hodnotili aplikaci velmi dobře, většinou nejlepším možným hodnocením. Uživatelům se líbil vzhled aplikace, nové zobrazení předmětu s důležitými informacemi, úvodní stránka přehledu, bezproblémová funkčnost a obecně uživatelský průchod aplikací. Někteří uživatelé navrhovali drobná vylepšení pro tabulku hodnocení, která sám vnímám jako vhodná.

Odezva z testování použitelnosti bude použita k budoucímu zlepšování webové aplikace a celé testování tedy vnímám jako velmi přínosné.

5.2 Automatické testování

Automatické testování je pro vývojáře klíčový nástroj, který přináší značné úspory prostředků a času. Automatizovat lze téměř všechny typy testů, nejefektivnější je však testování neměnných částí kódů. Cílem těchto testů je časová úspora při hledání chyb – díky automatickým testům mohou být některé chyby velmi rychle zřejmé. Z principu se nejvíce hodí pro části aplikace, ve kterých jsou prováděny složité operace. [15]

Při vytváření automatických testů je nutné pečlivě zvážit výhody a nevýhody. Zjevnou výhodou je časová úspora při automatické spouštění a vyhodnocení testů – využití těchto automatických testů minimalizuje budoucí chyby programátora při úpravách kódu. Nevýhodou těchto testů je nutnost jejich údržby v případě větších změn v aplikaci. [15]

V Angularu je pro unit testování možné využít Jasmine test framework. Testy se pak přes konzoli spouští v nástroji Karma, která testy spouští přímo v prohlížeči. Spouštění testů je jednoduché a díky nástroji Karma lze velmi rychle identifikovat příčiny problémů. [6]

Ve webové aplikace se nenachází příliš mnoho části, které provádí složité operace. Pokud nějaká část provádí složitější operace, je to ve většině případů kvůli stávajícímu API. Po konzultaci s vývojovým týmem Grades jsem se vzhledem k budoucím změnám na API rozhodl vytvořit pouze několik unit testů, které budou sloužit jako vzor pro vytváření dalších testů. V budoucnu je společně se změnami API plánováno lepší pokrytí kódů testy.

Závěr

Tato bakalářská práce se zabývala analýzou webové aplikace Grades a poté návrhem, implementací a testováním nové vylepšené webové aplikace. Nové webové uživatelské rozhraní je plně responzivní, funguje bez připojení k internetu a zlepšuje klíčové procesy a používání celé aplikace.

Procesy hodnocení předmětu a zobrazení předmětu byly vylepšeny pomocí offline funkcionality a dalších drobných vylepšení. Po nalezení a identifikaci těchto problémových oblastí, bylo navrženo nové řešení společně s vylepšeným grafickým návrhem.

Řešení je implementováno za pomocí nejnovějších principů a technologií. Implementace je modulární, což zlepšuje udržitelnost a možnosti rozvoje celého řešení. Využívá framework Angular, který usnadňuje vývoj webových aplikací. V současném REST API byly nalezeny koncepční nedostatky a chyběly v něm některé vhodné endpointy pro nové řešení, proto byly navrženy vhodné úpravy a pro účely vytvoření funkčního prototypu byla implementována snadno odstranitelná dočasná řešení. Požadavek na offline funkcionalitu byl naplněn pomocí technologie service workers. Aplikace dodržuje principy progresivních webových aplikací a je tedy možné ji snadno nainstalovat do mobilních zařízení, které tuto funkcionalitu podporují.

Celá webová aplikace byla řádně uživatelsky otestována. Při testování byli využiti jak uživatelé, kteří vidí aplikaci poprvé, tak i studenti a učitelé naší fakulty. Testování se ukázalo jako úspěšné a aplikace získala kladné ohlasy, společně s cennou odezvou.

V budoucnu bude možné rozšířit aplikaci o další stránky (jako například stránku pro definice hodnocení k předmětu), implementovat push notifikace a na základě výsledků testování implementovat drobné úpravy aplikace. Po dokončení úprav bude možné touto aplikací nahradit stávající webovou a mobilní aplikaci.

Literatura

[1] Liew, Z.: Understanding And Using REST APIs. *Smashing magazine [online]*, leden 2018, [cit. 2019-02-20]. Dostupné z: https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/

[2] HTTP Methods. *REST API Tutorial [online]*, [cit. 2019-02-20]. Dostupné z: https://restfulapi.net/http-methods/

[3] REST Architectural Constraints. *REST API Tutorial [online]*, [cit. 2019-02-20]. Dostupné z: https://restfulapi.net/rest-architectural-constraints/

[4] Training, J.: Difference Between AngularJs vs. Angular 2 vs. Angular 4 vs. Angular 5 vs. Angular 6. *JanBask Training [online]*, říjen 2018, [cit. 2019-03-02]. Dostupné z: https://www.janbasktraining.com/blog/angularjs-vs-angular/

[5] The Model-View-ViewModel Pattern. *Microsoft docs [online]*, srpen 2017, [cit. 2019-02-22]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

[6] Google: *Angular docs [online]*. [cit. 2019-02-22]. Dostupné z: https://angular.io/

[7] Patel, K.: What is Reactive Programming? *Medium [online]*, prosinec 2016, [cit. 2019-02-22]. Dostupné z: https://medium.com/@kevalpatel2106/what-is-reactive-programming-da37c1611382

[8] Using the application cache. *MDN web docs [online]*, 2019, [cit. 2019-02-23]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache

[9] Making PWAs work offline with Service workers. *MDN web docs [online]*, 2019, [cit. 2019-02-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Offline_Service_workers

[10] Progressive web apps. *MDN web docs [online]*, 2019, [cit. 2019-02-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

[11] Porter, J.: Principles of User Interface Design. *Bokardo [online]*, [cit. 2019-03-01]. Dostupné z: http://bokardo.com/principles-of-user-interface-design/

[12] Roberts, H.: Designing in the browser leads to better quality builds. *CSS Wizardry [online]*, říjen 2010, [cit. 2019-03-05]. Dostupné z: https://csswizardry.com/2010/10/designing-in-the-browser-leads-to-better-quality-builds/

[13] ČVUT v Praze: *Grafický manuál identity ČVUT [online]*. [cit. 2019-03-10]. Dostupné z: https://www.cvut.cz/sites/default/files/content/e254fb38-e72d-463b-8c9f-cb0435416f29/cs/20161215-graficky-manual-identity-cvut-v-praze.pdf

[14] Levin, G.: REST API Error Codes 101. *RestCase [online]*, prosinec 2015, [cit. 2019-03-20]. Dostupné z: https://blog.restcase.com/rest-api-error-codes-101/

[15] Automatizované testování. *Testování softwaru [online]*, [cit. 2019-03-03]. Dostupné z: http://testovanisoftwaru.cz/automatizovane-testovani/

Příloha A

Instalační příručka

Všechny adresáře jsou zde uváděny relativně k adresáři zdrojových kódů práce. Instalace práce vyžaduje spuštěné Grades REST API na libovolné adrese a portu, je pouze nutné nastavit adresu REST API a sock js v souboru `environments/environment.prod.ts`, následně se zdrojové kódy zkompilují příkazem `ng build --prod`. Výsledkem kompilace jsou statické soubory, spustitelná verze práce se nachází v adresáři `dist`.

Pro spuštění zkompilované práce lze využít libovolný webový server, který bude nejdříve všechny requesty přesměrovávat do souboru `index.html`. Pro snadné spuštění práce jsem připravil příkaz `npm run start-live-server`, který vyžaduje instalaci balíčku `live-server` a spuštěné Grades REST API na adrese `http://localhost:8088` (příkaz využívá proxy, která zabraňuje vzniku cors chyby).

Příloha B

Seznam použitých zkratek

API Application programming interface

REST Representational state transfer

PWA Progressive web application

DTO Data transfer object

DOM Document object model

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `user-testing` ..... testovací protokoly z testování použitelnosti
- `dist` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS