FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	NURIS - Informační systém pro MI-NUR
Student:	Bc. David Jirout
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Realizujte podpůrný informační systém pro předmět MI-NUR vyučovaný v rámci magisterského studia na ČVUT FIT.

Analyzujte výukový proces předmětu MI-NUR se zaměřením na tvorbu týmů, semestrální práce a jejich hodnocení.
Na základě analýzy a návrhu obdobného systému vytvořeného studenty MI-NUR navrhněte vlastní řešení.
Zvolte vhodnou implementační platformu a podpůrné nástroje s ohledem na snadnou údržbu.
Systém implementujte a řádně otestujte. Vhodné testy sám navrhněte (uživatelské testy, unit testy, atp.).
Systém nasaďte na ČVUT FIT a zhodnoťte jeho přínosy.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 16. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

NURIS - Informační systém pro MI-NUR

Bc. David Jirout

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

10. ledna 2019

Poděkování

Chtěl bych především poděkovat vedoucímu této práce Ing. Jiřímu Hunkovi za možnost zvolit si toto téma diplomové práce. Dále bych chtěl poděkovat Bc. Pavlu Kovářovi za výraznou pomoc při instalaci a konfiguraci fakultního serveru. V neposlední řadě bych chtěl poděkovat svým nejbližším za jejich trpělivost a podporu, bez které by tato práce nemohla vzniknout.

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, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Říčanech dne 10. ledna 2019 .....

Odkaz na tuto práci

Jirout, David. *NURIS - Informační systém pro MI-NUR*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019. Dostupný také z WWW: ⟨`http://minur.fit.cvut.cz`⟩.

Abstrakt

Tato diplomová práce se zabývá analýzou, návrhem a realizací webové aplikace pro podporu předmětu MI-NUR: Návrh uživatelského prostředí vyučovaného na Fakultě informačních technologí ČVUT v Praze. Cílem práce je kompletní vývoj podpůrné aplikace od základů. To zahrnuje provedení analýzy potřeb studentů i vyučujících tohoto předmětu, navržení samotné aplikace a tu následný vývoj a nasazení pro použití na fakultním serveru. Velký důraz je kladen na návrh uživatelského rozhraní aplikace a následné uživatelské testování.

Klíčová slova MI-NUR, Návrh uživatelského rozhraní, webová aplikace, analýza, návrh, implementace, testování

Abstract

The subject of this diploma thesis is analysis, design and implementation of web application supporting subejct MIE-NUR: User interface design, which is being taught at the Faculty of Information Technology at CTU in Prague. The aim of this thesis is a complete development of the support application right from the basics. That involves analyzing needs of students as well as teachers of this subject, designing the application and then development and

deployment on the faculty server. One of the main focuses of the thesis is the user interface design and user testing.

Keywords MI-NUR, UI, web application, analysis, design, implementation, testing

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Vzhledem k počtu studentů a zároveň také počtu vyučovaných předmětů roste potřeba využití sotfwarových systému pro správu výuky. Jinak tomu není ani u předmětů vyučovaných na katedře softwarového inženýrství Fakulty informačních technologií Českého vysokého učení technického v Praze, konkrétně u předmětu Návrh uživatelského rozhraní (MI-NUR).

V rámci předmětu Návrh uživatelského rozhraní zpracovávají studenti v týmech semestrální práci. Nejprve utvoří menší týmy o počtu tří až pěti studentů. Následně vymyslí téma projektu, kterým by se chtěli v průběhu semestru zabývat, a dále pro daný projekt vytváří postupně dva prototypy a provádí uživatelské testování.

Vzhledem k postupnému zániku fakultní aplikace pro podporu výuky EDUX (`https://edux.fit.cvut.cz/`), který dosud sloužil k odevzdávání semestrálních prací předmětu MI-NUR, vzniká potřeba tento systém nahradit. Tento fakt dal vzniknout požadavku na realizaci systému NURIS - informačnímu systému na správu předmětu MI-NUR.

Systém NURIS by měl narozdíl od aplikace EDUX vyhovovat všem potřebám studentů předmětu MI-NUR, nejen sloužit jako místo pro odevzdávání semestrální práce. Měl by také podporovat vytváření týmů studentů. Zároveň by pro vyučující měl nabídnout pohodlný způsob, jak odevzdané práce hodnotit.

Toto téma bylo autorovi navrženo vedoucím práce Ing. Jiřím Hunkou. NURIS byl již během vývoje v zimním semestru akademického roku 2018/2019 nasazen v produkčním prostředí na fakultním serveru https:
minur.fit.cvut.cz a byl při výuce aktivně využíván. Prvotními testery byli tedy v tomto semestru samotní studenti předmětu MI-NUR spolu se svým cvičícím a vedoucím této práce, Ing. Jiřím Hunkou.

Kapitola 1

Cíl práce

Primárním cílem této práce je navrhnout, naimplementovat, nasadit a otestovat NURIS – informační systém pro správu předmětu MI-NUR: Návrh uživatelského rozhraní. Jelikož tento předmět dosud využíval pouze nyní již zanikající aplikaci EDUX pro nahrávání semestrálních prací, vznikal celý systém od nuly. Je tedy nejprve nutné analyzovat potřeby studentů a vyučujících, následně všechny části systému (databázi, backend i frontend) navrhnout a naimplementovat. Dále provést konfiguraci a nasazení na fakultním serveru a v neposlední řadě také testování jak s nezávislými testujícími subjekty, tak formou zpětné vazby se samotnými studenty předmětu MI-NUR: Návrh uživatelského rozhaní. Design frontendu vychází z projektu NUR Expert, který byl vypracován Martinem Pavelkem, Igorem Kuľkou, Jakubem Šillerem a Michalem Popkem v rámci předmětu MI-NUR v zimním semestru 2017/2018.

Kapitola 2

Struktura práce

V kapitole Analýza a návrh je čtenáři předložen průběh celého procesu analýzy a návrhu systému. Jsou zde uvedeny jak funkční požadavky a samotný návrh všech základních částí, tj. databáze, backendu a samotného uživatelské rozhraní, tak i přehled použitých technologií a důvody, proč byly pro tuto práci právě tyto technologie vybrány jako ty nejvhodnější.

Kapitola Realizace popisuje, jakým způsobem byl návrh jednotlivých částí systému pomocí použitých technologií realizován. Zabývá se nejprve datovým modelem aplikace, architekturou backendové části včetně formátu přenášených dat, popisu rozhraní API a zabezpečení celého systému a následně také architekturou frontendu a designem UI. Dále je zde čtenář seznámen s tím, jak byl projekt nasazen na fakultním serveru. V poslední řadě je zde uvedeno, jaké změny byly v systému NURIS provedeny na základě požadavků v průběhu semestru.

V kapitole Vzhled aplikace je čtenář seznámen s vizuální podobou aplikace po implementaci všech funkčních požadavků.

Kapitola Testování obsahuje informace o tom, jaká byla při testování použita metodika a s jakými testovacími subjekty byly provedeny uživatelské testy. Zároveň jsou zde zveřejněny výsledky všech testů včetně následných úprav systému.

Kapitola 3

Analýza a návrh

3.1 Proces analýzy

V případě analýzy požadavků na systém NURIS nebylo přistoupeno ke klasickému postupu rešerše obdobných (konkurenčních) aplikací. Požadavky ze strany studentů přímo vyplývají z popisu předmětu Návrh uživatelského rozhraní[1]. Prvotním zadavatelem a subjektem průběžně poskytujícím požadavky ze strany vyučujících tohoto předmětu byl Ing. Jiří Hunka, který v době vývoje NURIS v zimním semestru 2018/2019 vedl všechna cvičení tohoto předmětu na Fakultě informačních technologií.

3.2 Průběh předmětu MI-NUR: Návrh uživatelského rozhraní

V rámci předmětu MI-NUR: Návrh uživatelského rozhraní, pro který je systém NURIS vyvíjen, studenti nejprve na začátku semestru vytvoří zpravidla tří až pětičlenné týmy. Dále vymyslí téma semestrálního projektu, na kterém by chtěli v rámci svého týmu pracovat. Poté postupně na cvičeních následují 3 iterace dílčích odevzdání semestrálního projektu:

1. Rozvaha + Mock-up – studenti definují svůj zamýšlený projekt a vytvoří lo-fi prototyp aplikace,
2. Rešerše + Hi-fi – studenti provedou rešerši konkurenčních aplikací a vyhotoví hi-fi prototyp aplikace,
3. Testování – studenti mají za úkol provést heuristickou analýzu své aplikace a uživatelské testování.

Za každé z dílčích odevzdání jsou studenti odměněni body. Další mohou studenti získat za účast na testování až u dvou projektů ostatních studentů a také za dřívější odevzdání poslední iterace semestrální práce. Za včasné odevzdání

všech dílčích iterací, kde z každé musí studenti obdržet minimálně polovinu z možných bodů, získají studenti právo na udělení zápočtu.

3.3 Funkční požadavky

Základní funkční požadavky lze shrnout v následujících několika bodech:

- jednoduchá, minimalistická aplikace pro správu předmětu Návrh uživatelského rozhraní
- co možná nejjednodušší design pro intuitivní ovládání
- aplikace umožní studentům na začátku semestru zformovat týmy
- studenti budou moci pomocí aplikace odevzdávat vypracovaná řešení semestrálních úloh
- vyučující může v aplikaci provádět hodnocení odevzdaných řešení

3.4 Use cases

Ze spektra možností, jak postupovat při návrhu systému NURIS, byla jako vhodná shledána metoda use cases. Ta je zaměřena na otázky, jaké subjekty (actors) budou aplikaci používat a jaké úkony (use cases) bude daný uživatel v systému potřebovat provést. Z pohledu actors můžeme use cases rozdělit do dvou kategorií – studenty a vyučující.

3.4.1 Use cases z pohledu studenta

- použití fakultního přihlašování pro přístup do aplikace
- možnost založit tým (projekt) pro semestrální práci
- možnost připojit se k již existujícímu týmu (projektu)
- správa týmu, resp. jeho členů (možnost odebrat neaktivního člena apod.)
- zadání aktuální iterace semestrální práce včetně informací o současném stavu a termínu odevzdání
- možnost nahrávat vypracované odevzdání do systému, případně připojit poznámku
- mít přehled o počtu dosažených bodů v semestru

3.4.2 Use cases z pohledu vyučujícího

- použití fakultního přihlašování pro přístup do aplikace
- přehledné zobrazení seznamu všech týmů včetně aktuálního stavu
- možnost kontrolovat utváření týmů studentů
- průběžná kontrola odevzdaní jednotlivých týmů
- systém hodnocení, přidělování bodů týmům
- možnost 'předhodnocení' odevzdání
- možnost schválení či zamítnutí odevzdání
- přehled o počtu dosažených bodů jednotlivých studentů

3.5 Návrh uživatelského rozhraní

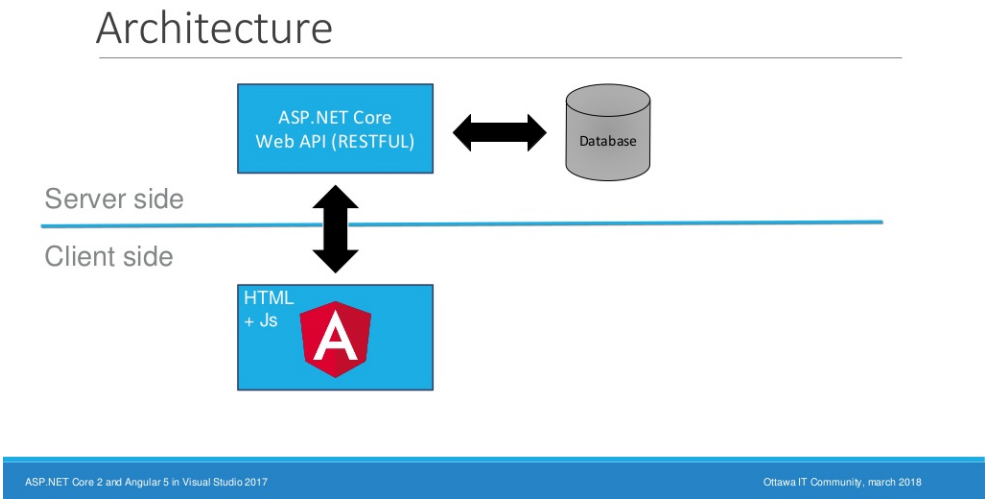
Hlavním požadavkem na moderní aplikaci je z pohledu klienta jednoduché intuitivní ovládání a minimalistický responzivní design. Z tohoto důvodu bylo přistoupeno k vytvoření Single Page Application (SPA). SPA je typ webové aplikace, kdy je uživateli načtena a zobrazena jediná HTML stránka, která na základě přihlášeného uživatele a jeho interakce mění svůj obsah.[2] V případě systému NURIS bylo rozhodnuto, že z důvodu oddělení zodpovědností (angl. separation of concerns, SoC) je vhodné, aby takové stránky existovaly právě dvě – první pro seznam všech existujících projektů a druhá pro detail projektu.

Jak bylo zmíněno již v úvodu této práce, samotné GUI a barevné schéma klientské aplikace (frontendu) jsou silně inspirovány projektem NUR Expert, který byl vypracován Martinem Pavelkem, Igorem Kuľkou, Jakubem Šillerem a Michalem Popkem v rámci předmětu MI-NUR v zimním semestru 2017/2018.

3.6 Návrh architektury systému

Jedním z hlavních cílů této práce je návrh architektury systému. Z funkčních požadavků a zamýšleného užití aplikace je více než zřejmé, že se nutně musí jednat o webovou aplikaci. Tím je automaticky zaručeno, že aplikace bude (z uživatelského úhlu pohledu) plně multiplatformní. Na druhou stranu se mohou objevit problémy s uživatelským rozhraním, neboť rozdílné webové prohlížeče mohou některé prvky aplikace zobrazovat jiným způsobem, v horším případě by mohla být v závislosti na použitém prohlížeči ovlivněna i celková funkcionalita systému.

V souladu s aktuálními programátorskými zvyklostmi a vzhledem k povaze funkčních požadavků a uvažovaných use cases se nabízí, aby byl pro návrh celkové architektury aplikace využit návrhový vzor Client-Server. Pro samotnou



Obrázek 3.1: Diagram architektury Client-Server

[3]

serverovou část se pak nabízí architektura Model-View-Controller (dále jen MVC).

3.6.1 Client-Server

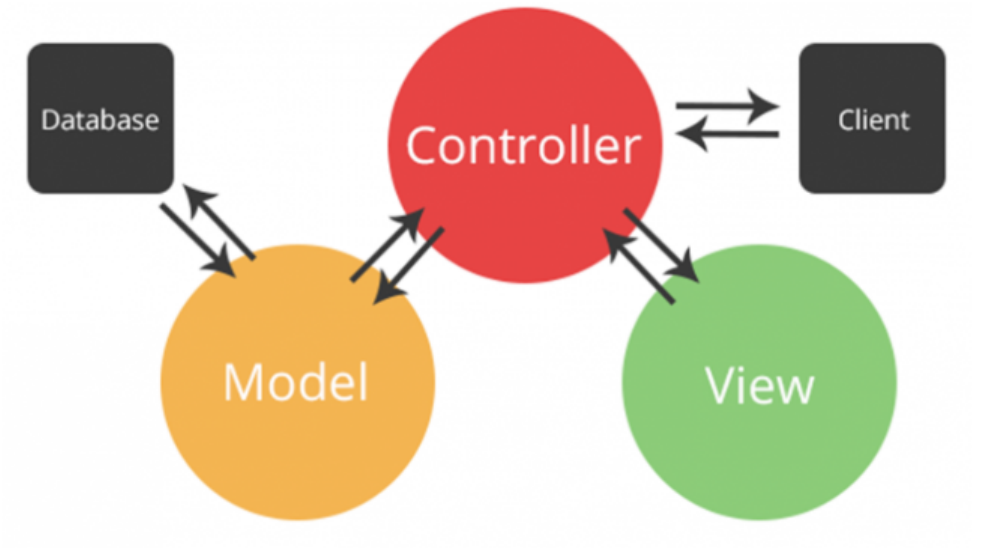
Jak již vyplývá z využití návrhového vzoru Client-Server (viz Obrázek 3.1), je aplikace rozdělena na dvě samostatné části – klientskou aplikaci (client) a server.

3.6.1.1 Server

Část aplikace označovaná jako serverová je ta, která je pro všechny instance klientské aplikace (v obecném případě i pro více různých klientských aplikací) sdílená. Zpravidla ji tvoří 3 vrstvy:

1. datová vrstva, která slouží jako adaptér pro přístup ke zdrojům dat,
2. business vrstva, která zajišťuje překlad surových dat z datové vrstvy na data srozumitelná pro zbytek aplikace a veškerou výpočetní logiku,
3. rozhraní pro zprostředkování (API) dat klientským aplikacím.

U aplikací, u kterých se nepředpokládá připojení na více datových zdrojů, mohou datová a business vrstva splynout v jednu. Stejně tak u serverových aplikací, které nemají za úkol žádné komplexnější výpočty, může business



Obrázek 3.2: Diagram návrhového vzoru MVC

[4]

vrstva splynout s rozhraním pro zprostředkování dat. Serverové aplikace s menší složitostí tak mohou být tvořeny jen jedinou vrstvou, neboť více vrstev by namísto rozčlenění aplikace na menší logické celky naopak učinilo systém zbytečně složitějším.

3.6.1.2 Client

Klientská aplikace se narozdíl od serverové vyznačuje tím, že pro každého připojeného klienta existuje v samostatné instanci. Každý klient tak u sebe v prohlížeči pracuje s vlastním lokálním stavem aplikace.

3.6.2 Model-View-Controller

Model-View-Controller (MVC) je, jak již název napovídá, návrhový vzor, který tvoří 3 navzájem komunikující části – Model, View a Controller. To je ilustrováno na obrázku 3.2.

3.6.2.1 Model

Model obsahuje veškerou logiku aplikace. Je zodpovědný za formu a uložení dat, jejich konzistenci, přístup k datovým zdrojům (lokálním i vzdáleným) i souborovému systému.

3.6.2.2 View

View (angl. "pohled") je část aplikace, která obstarává uživatelské rozhraní a prezentuje data klientovi.

3.6.2.3 Controller

Controller propojuje v návrhovém vzoru MVC model a view. Klientská aplikace komunikuje výhradně s controllerem, který na základě interakce uživatele povede (resp. deleguje) požadované změny v modelu a vrátí upravené view.

3.7 Datové úložiště

Možností, jak uchovávat data, existuje v dnešní době celá řada. Od primitivního ukládání serializovaných dat do souborů, přes klasické relační databáze, až po modernější možnosti, jako jsou například NoSQL či grafové databáze.

3.7.1 Ukládání do souborů

Metoda ukládání dat do souborů je obecně z hlediska bezpečnosti považována za nevhodnou.[5] Pro účely této práce se očekává, že data budou konzistentní, k čemuž tato metoda uložení není příliš vhodná.

3.7.2 Relační databáze

Tento druh databáze je v dnešní době zdaleka nejpopulárnějším na trhu.[6] Tato databáze je vhodná pro data relačního typu a zajišťuje vlastnosti ACID (Atomicity, Consistency, Isolation, Durability), čímž zajišťuje konzistenci uložení dat. Lze předpokládat, že data systému NURIS budou relačního charakteru, a proto se zdá být relační databáze zajišťující konzistenci dat vhodným řešením datového úložiště.

3.7.3 NoSQL databáze

NoSQL databáze je charakteristická tím, že je každý záznam tvořen dokumentem (většinou ve formátu JSON), který není svázán žádnou pevnou a předem definovanou strukturou. Jedná se tedy o dokumentovou resp. souborovou databázi. Z tohoto důvodu není, co se týče uložení dat pro systém NURIS, nejvhodnější.

3.7.4 Grafové databáze

Tento typ databáze reprezentuje entity a vztahy mezi nimi jako vrcholy a hrany v orientovaném grafu. Zpravidla však neposkytují indexy na všech entitách (vrcholech) a k těm pak nelze přistupovat přímo pouze na základě

atributů.[7] Z tohoto důvodu není ani tento typ databáze pro sytém NURIS vyhovující.

3.7.5 Shrnutí a zhodnocení

Pro účely NURIS lze předpokládat, že ukládaná data nebudou jakkoli komplexního charakteru, tudíž nebude třeba využívat možností, které nabízí NoSQL databáze. Nejlepším řešením tak zůstává uložení dat v klasické relační databázi, která zajišťuje jak konzistenci dat, tak jejich bezproblémovou dostupnost.

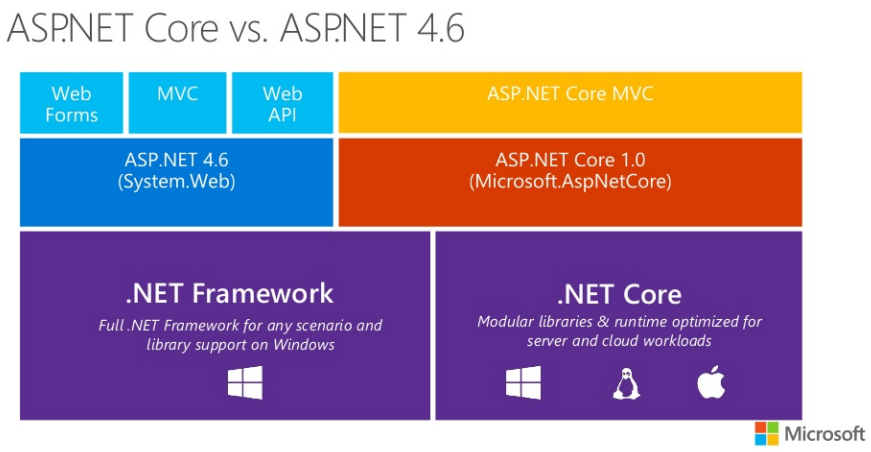
3.8 Použité technologie

V této sekci práce je popsáno, jaké technologie byly nakonec pro realizaci systému NURIS zvoleny a z jakého důvodu. Je třeba si uvědomit, že výběr technologie nemůže být založen čistě na teoretické vhodnosti pro řešení daného problému. Je třeba zohlednit i zkušenosti vývojáře (v tomto případě autora této práce) s danou technologií.

3.8.1 Microsoft ASP.NET Core

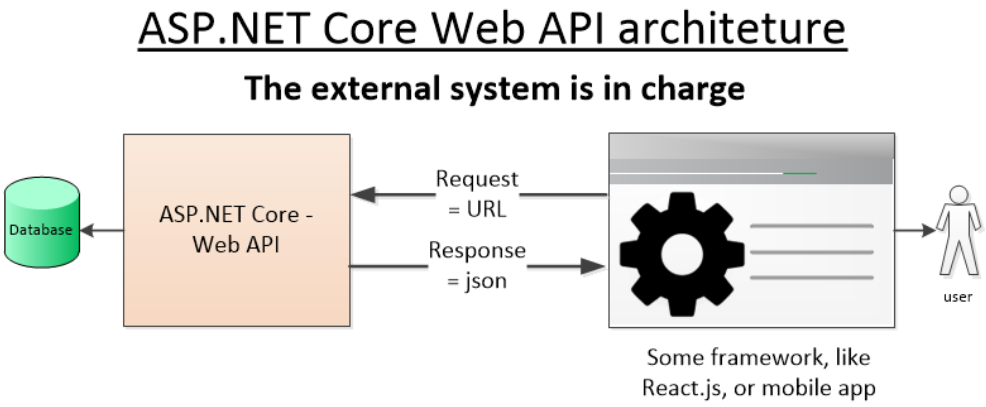
Pro implementaci serverové (backendové) části systému NURIS byla především z důvodu zkušeností autora této práce zvolena technologie ASP.NET Core od společnosti Microsoft. O té lze otevřeně říci, že je ze strany tohoto známého softwarového gigantu velmi průkopnická. To je dáno především tím, že je narozdíl od standardního .NET frameworku čistě multiplatformní. Schéma porovnávající klasický ASP.NET framework s ASP.NET Core je zobrazeno na obrázku 3.3.

Technologie ASP.NET Core nativně podporuje architekturu MVC[9], přičemž generování a renderování view (zobrazení) pro klienta lze přenechat čistě na klientské aplikaci a není tak nutné generovat pohledy přímo z prostředí .NET jako tomu je při použití klasického ASP.NET frameworku. Jak tato architektura funguje je graficky znázorněno na obrázku 3.4.



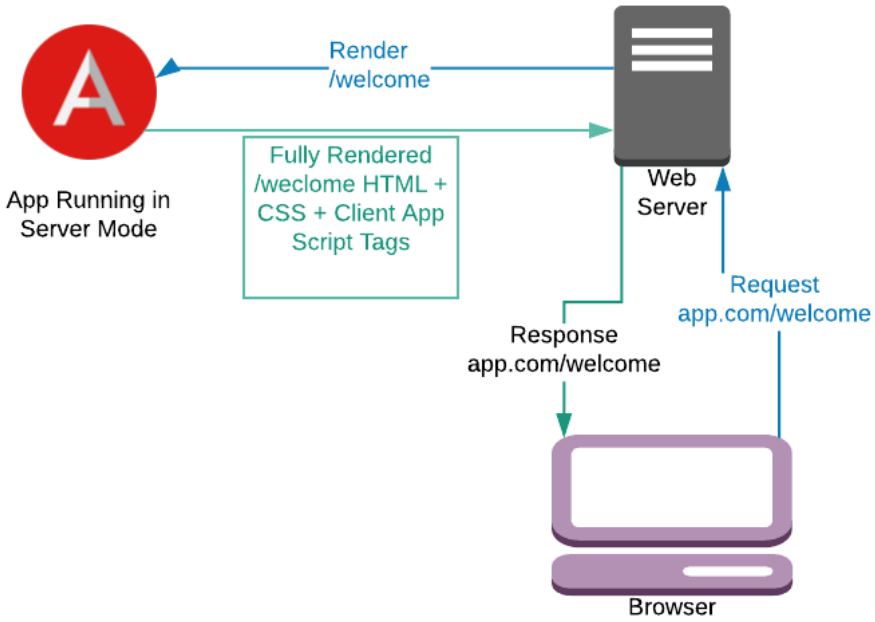
Obrázek 3.3: Porovnání ASP.NET a ASP.NET Core

[8]



Obrázek 3.4: Architektura webové aplikace založené na technologii ASP.NET Core

[10]



Obrázek 3.5: Schéma fungování webové aplikace s Angular Universal

[11]

3.8.2 Angular Universal

AngularJS je v dnešní době jedním z nejrozšířenějších JavaScript frameworků pro tvorbu single-page klientských aplikací (SPA). Je vytvořen v TypeScriptu, což je skriptovací programovací jazyk od firmy Microsoft, který klasický JavaScript rozšiřuje o standardní prvky objektově orientovaných programovacích jazyků. Přidává tak nejen třídy (classes), rozhraní (interfaces) a dědičnost (inheritance), ale také např. typovou kontrolu (type checking). Navíc nativně podporuje architekturu MVC. Díky těmto vlastnostem je AngularJS velmi šikovný a vývojářsky přívětivý nástroj. Stejně jako ostatní JavaScriptové frameworky běží v prohlížeči klienta.

Angular Universal je technologie, která umožňuje spouštět AngularJS na straně serveru. Zatímco klasická AngularJS aplikace běží v prohlížeči klienta, kde renderuje jednotlivé HTML stránky v objektovém modelu dokumentu (DOM) na základě interakce klienta, Angular Universal vytváří statické stránky aplikace už na serveru pomocí procesu zvaného server-side rendering (SSR). HTML stránky mohou navíc být na straně serveru předgenerované pro pozdější použití. Klient tak po odeslání požadavku na server obdrží zpět do

prohlížeče hotovou stránku bez nutnosti cokoli sám renderovat (viz obrázek 3.5). Aplikace je tak z pohledu uživatele rychlejší a responzivnější.

3.8.3 PostgreSQL

V sekci 3.7 bylo zdůvodněno, proč byla relační databáze vybrána jako vhodné řešení pro uložení dat systému NURIS. Nyní zbývá rozhodnout, jaký databázový stroj zvolit. Z běžně používaných se nabízí MySQL, Oracle, MSSQL nebo PostgreSQL. První dva zmíněné stroje byly z výběru vyřazeny takřka okamžitě. MySQL databáze je sice jednoduše použitelná, nicméně mívá problémy se stabilitou, je z hlediska funkcionality poněkud omezená a její možnosti často závisí na doplňcích (addons) třetích stran[12]. Databáze Oracle se v dnešní době těší značné oblíbenosti, nicméně podléhá placené licenci a tudíž je pro tento fakultní projekt nevhodná. MSSQL je v edici Express sice bezplatná a pro tento projekt dostačující, nicméně nakonec bylo rozhodnuto o použití databáze PostgreSQL, která nabízí kompletní funkcionalitu relačního databázového stroje, je open-source a dá se poměrně snadno spravovat z webového prohlížeče.

3.8.4 NGINX

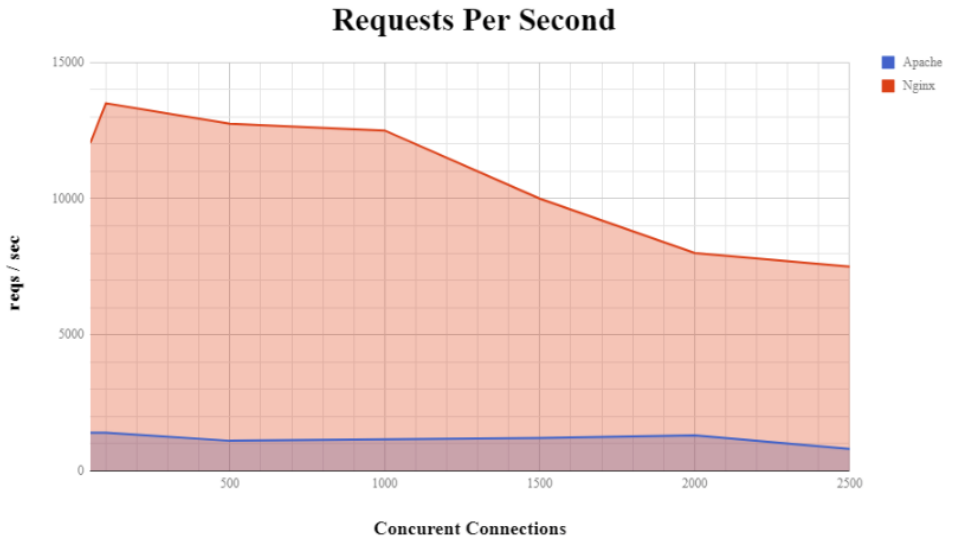
Vzhledem k tomu, že na fakultním serveru bude dle zvyklostí nasazen operační systém s unixovým jádrem, nelze kvůli svázanosti s OS Windows použít webový server. S tím jediným měl autor této práce a vývojář systému NURIS předešlé zkušenosti. Při volbě, jaký webový server tedy zvolit, připadají v úvahu dva největší hráči na trhu – Apache a NGINX.[13]

V obou případech se jedná o HTTP servery podporující reverzní proxy. Obě možnosti zároveň podporují funkce generického TCP/UDP proxy serveru a jsou tak pro nasazení systému NURIS obě vyhovující. Aby bylo možné učinit kvalifikované rozhodnutí, je třeba zaměřit se na porovnání výkonnosti obou webových serverů.

Ačkoli lze u systému NURIS předpokládat jen malý počet současně připojených klientů (počet studentů MI-NUR v zimním semestru 2018/2019 nikdy nepřesáhl číslo 75), je z grafů 3.6 3.7 zřejmé, že webový server NGINX dokáže obsloužit výrazně větší počet simultánních požadavků při mnohem nižším využití paměti než server Apache. Z tohoto důvodu bylo nakonec přistoupeno k nasazení systému NURIS na webový server NGINX.

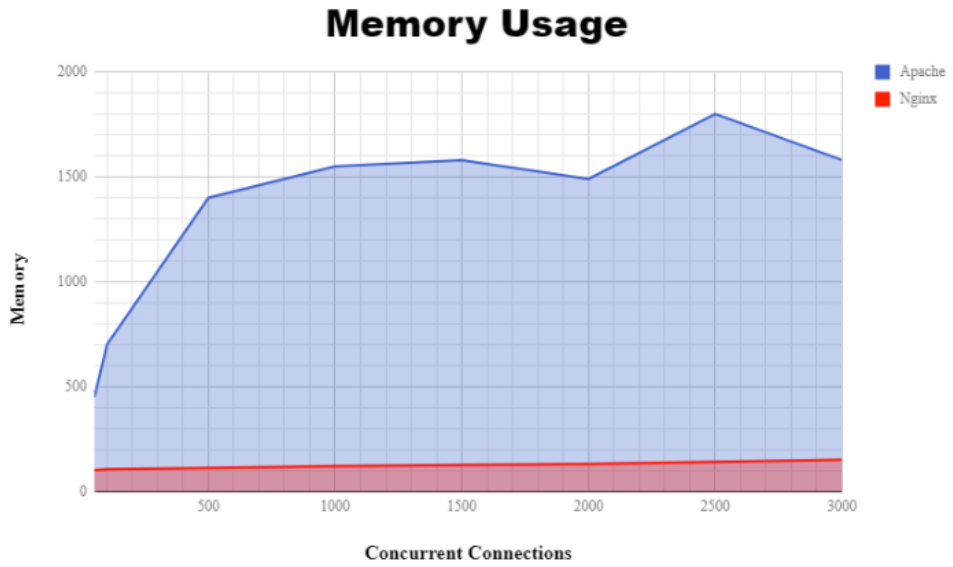
3.8.5 GitLab

Pro vývoj webové aplikace pro fakultní použití je vhodné využít možností verzovacího systému. Ten má za úkol zaznamenávat veškeré změny a úpravy ve zdrojových kódech vyvíjené aplikace. Pomocí verzovacího systému je možné prohlížet starší verze kódu aplikace a v případě nutnosti se k některé ze starších



Obrázek 3.6: Porovnání výkonnosti webových serverů – počet obsloužených požadavků za sekundu

[14]



Obrázek 3.7: Porovnání výkonnosti webových serverů – použití paměti

[14]

verzí vrátit. Stejně tak lze přehledně vidět, jaké úpravy byly na aplikaci provedeny v průběhu času. Tyto vlastnosti jsou o to přínosnější v okamžiku, kdy se k vyvíjené aplikaci připojuje další vyvojář a nebo dochází k předání vývoje či údržby novému vývojáři.

Podobně jako v jiných oblastech software existuje i v dnešní době v oblasti verzovacích systémů několik možností. Nejznámější z těchto možností jsou git a subversion (zkráceně svn). Jelikož ale Fakulta informačních technologií ČVUT pro svou výuku i mimo ni aktivně využívá GitLab na adrese `https://gitlab.fit.cvut.cz/`, nebylo takřka o čem rozhodovat. Využití této fakultní služby bylo jasnou a přímočarou volbou.

Kapitola 4

Realizace

V této kapitole je popsáno, jakým způsobem probíhala realizace systému NURIS na základně již provedené analýzy a návrhu. Popis fungování některých stěžejních částí systému je zde dokumentován nejen slovně, ale i útržky zdrojového kódu.

4.1 Vývojové prostředí

Vývoj celého systému, tedy jak klientské aplikace, tak serverové části, probíhal na platformě Windows ve vývojovém prostředí Visual Studio Community 2017. Tento software od společnosti Microsoft je nejpokročilejším vývojovým nástrojem pro celou platformu .NET. Pro vývoj klientské aplikace (frontendu) umožňuje použití jak klasických ASP.NET Web Forms využívajících .NET Framework, tak také vývoj pomocí moderních JavaScriptových (resp. TypeScriptových) frameworků jako jsou Angular a React. Obě rozdílné části systému lze tedy vyvíjet v jediném programátorsky přívětivém vývojovém prostředí (IDE).

4.2 Datový model

Při vývoji projektu jako je systém NURIS je nutné nejprve navrhnout strukturu databáze. Ta v tomto případě nebude nikterak složitá. Je však žádoucí, aby entity v databázi odpovídaly datovým třídám backendu. K tomuto účelu a pro následný přístup k datům je použit Microsoft Entity Framework Core (dále jen EF Core).

4.2.1 Entity Framework Core

Tato technologie od společnosti Microsoft je lightweight, rozšiřitelná a multiplatformní verze populárního Entity Frameworku umožňujícího přístup ke zdrojům dat. EF Core může sloužit jako objektově-relační mapovač (ORM),

což umožňuje vývojářům .NET aplikací pracovat s databází užitím .NET objektů a tím tak eliminuje většinu kódu pro přístup k datům, který by jinak bylo potřeba psát ručně.

Pro samotný návrh datového modelu máme při použití EF Core dvě možnosti – metodu "database-first"a metodu "code-first". První zmíněná metoda označuje situaci, kdy jsou vývojářem jako první vymodelovány tabulky v databázi. Z těch jsou poté s použitím EF Core vygenerovány datové třídy backendu. Druhá ze zmíněných metod značí přesně opačnou situaci. Jako první jsou tedy vytvořeny datové třídy v backendu aplikace a z nich jsou použitím EF Core vytvořeny tabulky a vztahy v relační databázi. Toto bývá pro nové projekty častěji využívaná možnost, neboť umožňuje vygenerovat celou databázi včetně všech tabulek a relací automaticky pouze použitím EF Core příkazů bez nutnosti vytváření jakýchkoli SQL queries.

Pro vývoj datového modelu systému NURIS byla tedy vybrána varianta "code-first". Ta funguje skrze vytváření tzv. "migrací".

4.2.2 Migrace

Po vytvoření nebo změně datových tříd reprezentujících datový model v backendu aplikace dochází k desynchronizaci modelu s databází. Je možné existující databázi smazat a nechat EF Core vygenerovat zcela novou, nicméně to má za následek ztrátu všech dat. Funkce migrace v EF Core nabízí možnost, jak inkrementálně aktualizovat databázové schéma a zároveň zachovat všechna existující data. Pomocí příkazu

```
Add-Migration název-migrace
```

v PowerShell či Package Manager konzoli Visual Studia je možné vygenerovat migraci s daným názvem. Ta se skládá ze samotného souboru migrace a z obrazu stávající databáze, který slouží k rozpoznávání, zda byla v datovém modelu provedena nějaká změna. Jakmile je migrace vytvořena, je možné použít příkaz

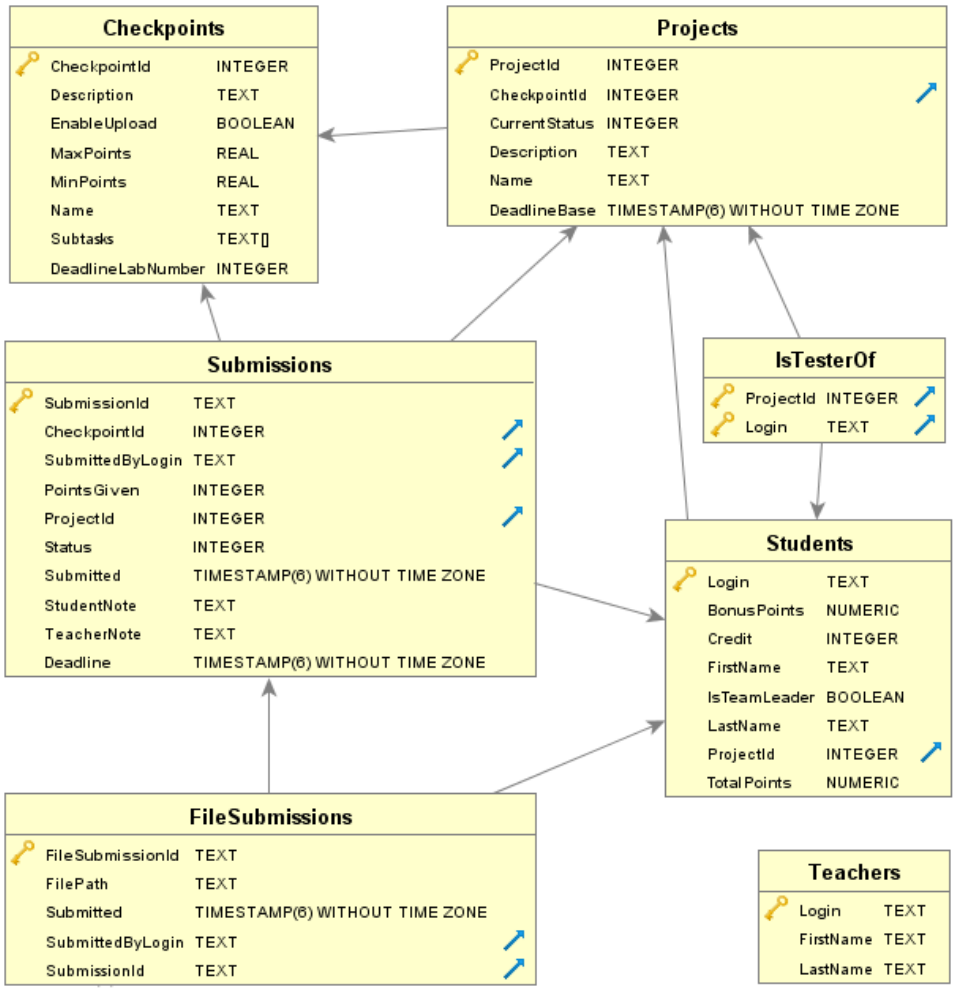
```
Update-Database [název-migrace]
```

k aktualizaci databáze buď po migraci, jejíž název je volitelným parametrem příkazu, nebo na nejnovější dostupnou verzi, pokud je příkaz použit bez parametru.

4.2.3 Výsledné schéma databáze

Po vytvoření všech potřebných datových tříd a vygenerování relační databáze pomocí migrací EF Core vypadá výsledné schéma databáze tak, jak je vidět na obrázku 4.1.

schema.png



Obrázek 4.1: Schéma databáze systému NURIS

4.3 Architektura backendu

Jak bylo zmíněno v sekci 3.6.2, architektura serverové části (backendu) aplikace odpovídá architektuře MVC 3.2. V této kapitole jsou rozebrány a zdokumentovány stěžejní části backendu, od formátu přenášených dat přes popis aplikačního rozhraní po zabezpečení celé aplikace.

4.3.1 JSON

Jako formát pro přenos dat mezi serverovou a klientskou částí aplikace byl zvolen JavaScript Object Notation, zkráceně JSON. Jedná se o formát umožňující snadný zápis i čitelnost nejen pro výpočetní techniku, ale také pro člověka. Je plně podporován a nativně používán ve všech aplikacích postavených na technologii ASP.NET Core, což zajišťuje snadné mapování objektů z i do formátu JSON. Pro ilustraci tohoto formátu je na následujících řádcích uveden příklad serializovaného projektu ze systému NURIS.


```
{
 "Id": 56,
 "Name": "Testovací projekt",
 "Description": "Tento projekt slouží výhradně k účelu testování.",
 "Students": [
 {
  "FirstName": "jméno",
  "LastName": "příjmení",
  "Login": "login123",
  "TestedProjectIds": [],
  "TestedProjectNames": [],
  "BonusPoints": 0
 }
 ],
 "Testers": [],
 "Submissions": [
 {
  "SubmissionId": "a74f9059-5968-4acb-97da-cbb07d8c5960",
  "CheckpointId": 1,
  "Checkpoint": {
   "CheckpointId": 1,
   "Name": "Najděte si tým a nebo si založte vlastní",
   "Description": "",
   "Subtasks": null,
   "MinPoints": 0.0,
   "MaxPoints": 0.0,
   "DeadlineLabNumber": 3,
   "EnableUpload": false
```

```
  },
  "FileSubmissions": [],
  "Submitted": null,
  "SubmittedBy": null,
  "Points": 0,
  "MaxPoints": 0.0,
  "Status": 0,
  "StatusName": "Čeká na zhotovení",
  "StudentNote": null,
  "TeacherNote": null,
  "Deadline": "2018-11-07T10:45:00"
 }
 ],
 "Checkpoint": {
  "CheckpointId": 1,
  "Name": "Najděte si tým a nebo si založte vlastní",
  "Description": "",
  "Subtasks": null,
  "MinPoints": 0.0,
  "MaxPoints": 0.0,
  "DeadlineLabNumber": 3,
  "EnableUpload": false
 },
 "CurrentStatus": 0
}
```


4.3.2 REST API

Representational State Transfer (REST) API je datově orientované aplikační rozhraní používající protokol HTTP/HTTPS ke 4 základním operacím pro práci s daty – GET (čtení), POST (zápis), PUT (změna) a DELETE (mazání). Všechna data, která jsou v tomto kontextu označována jako tzv. resources, mají přidělenu svou vlastní URI adresu. Tyto adresy dohromady tvoří hierarchickou strukturu. Výčet resources dostupných pomocí REST API systému NURIS je uveden v tabulce 4.1.

Jednou ze základních vlastností REST služeb je bezstavovost (statelessness). To znamená, že každý HTTP požadavek je ze strany REST API obsloužen v naprosto izolovaném prostředí. Každý požadavek tak musí obsahovat veškerá data nutná pro jeho obsloužení, nikdy nemůže být nijak závislý na předchozích požadavcích. Systém NURIS se tak narozdíl od samotných projektů nepohybuje mezi žádnými stavy.

Tabulka 4.1: NURIS REST API resources

URI	Metoda	Popis
`/api/projects`	GET	seznam všech projektů
`/api/projects/{projectId}`	GET	detail projektu
`~/{projectId}`	POST	vytvoří nový projekt
`~/{projectId}`	DELETE	smaže projekt
`~/{projectId}/students`	POST	přidá uživatele k projektu
`~/{projectId}/students/{login}`	DELETE	odebere uživatele
`~/{projectId}/testers`	POST	přidá testera k projektu
`~/{projectId}/testers/{login}`	DELETE	odebere testera z projektu
`~/{projectId}/checkpoints`	PUT	posune projekt do další fáze
`~/{projectId}/submissions/{subId}`	DELETE	smaže soubor odevzdání
`~/submissions`	POST	odešle soubor odevzdání
`~/submissions/confirm`	PUT	potvrdí odevzdání
`~/submissions/cancel`	PUT	zruší potvrzení odevzdání
`~/submissions/evaluate`	PUT	odešle hodnocení odevzdání
`~/purge`	DELETE	smaže všechny projekty

4.3.3 ProjectsController

Všechny REST API resources uvedené v tabulce 4.1 jsou definované v třídě ProjectsController.cs. Ta tvoří jádro celého backendu, jelikož vzhledem k nízké složitosti jsou v případě systému NURIS všechny tři vrstvy aplikační logiky popsané v sekci 3.6.1.1, (tedy rozhraní, business a datová vrstva) spojeny v jednu. Veškerou logiku aplikace tedy spravuje již zmíněný ProjectsController.

Při přístupu klienta na některou z resource adres je provedeno volání metody přidružené k dané adrese. V rámci metody je provedeno vyhledání odpovídajících dat v databázi a výsledek je zaslán klientovi jako odpověď ve formátu JSON (viz sekce 4.3.1). Pro ilustraci je níže uveden příklad metody pro přidání uživatele resp. studenta k projektu. V tomto případě je požadavek HTTP POST směřovaný na URI adresu api/projects/projectId/students převeden na volání metody JoinProject(...).

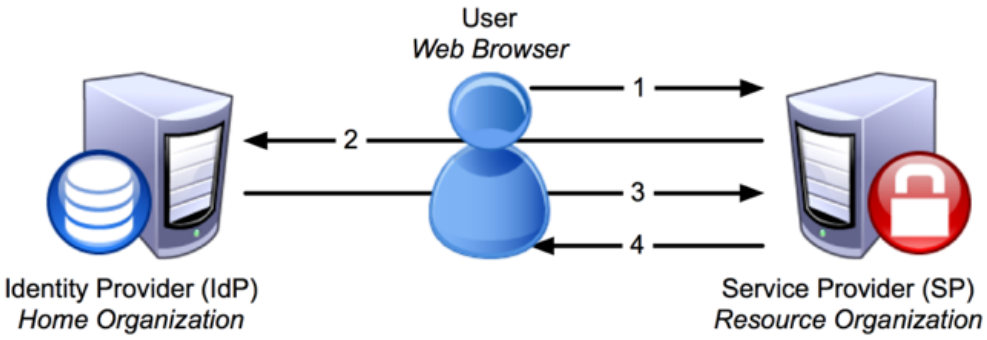
```
namespace NURIS.Controllers
{
  [Authorize]
  [Produces("application/json")]
  [Route("api/projects")]
  public class ProjectsController : Controller
  {
    [...]
    [HttpPost("{projectId}/students")]
    public IActionResult JoinProject([FromRoute] int projectId)
    {
      var student = _nurisContext.Students.Find(User.Identity.Name);
      student.ProjectId = projectId;
      student.Project = _nurisContext.Projects.Find(projectId);

      _nurisContext.SaveChanges();

      _logger.LogInformation(
        @"User {User.Identity.Name} joined project
                id:{projectId} - {projectName}"
      );

      return Json(student, settings);
    }
    [...]
  }
}
```

4.3.4 Zabezpečení systému NURIS

V této sekci je zdokumentováno, jakým způsobem je serverová část systému NURIS zabezpečena proti neoprávněné manipulaci s daty.

Jelikož systém NURIS pracuje jak s osobními, tak s uživatelskými daty studentů a jejich vyučujících, je třeba pečlivě dbát na bezpečnost spravovaných dat. Pokud by se případnému útočníkovi podařilo proniknout do systému přes špatně zabezpečené REST API, mohlo by to studentům MI-NUR: Návrh uživatelského rozhraní znemožnit splnění tohoto předmětu a tím ovlivnit celkový průběh studia.

Při každém přístupu na REST API resource je proto nutné provést kontrolu identity uživatele. Zároveň je třeba zabránit tomu, aby se nemohl útočník v systému NURIS vydávat za někoho jiného (krádež identity). Autentizace uživatelů, na konci které je uživateli vystaven autentizační token pro přístup do aplikace, se proto skládá hned z několika fází. První z nich je ověření totožnosti



Obrázek 4.2: Schéma autentizace uživatele pomocí ShibbolethSSO

[16]

uživatele využitím celofakultního Single Sign-On (SSO) přihlašování pomocí ShibbolethSSO díky registraci systému ve federaci cvutID.

4.3.4.1 ShibbolethSSO

„Shibboleth je Single Sign-on (SSO) řešení pro webové aplikace vybudované s využitím technologie SAML. Systém tvoří dva druhy komponent - poskytovatel identity (Identity Provider, IdP) a poskytovatelé služeb (Service Provider, SP, zjednodušeně aplikace). IdP funguje jako registr uživatelů a zajišťuje veškeré nutné kroky k jejich autentizaci (např. pomocí uživatelského jména a hesla). IdP a na něj napojené SP tvoří federaci. Federace provozovaná na ČVUT má název cvutID.“[15]

Systém NURIS v rámci tohoto schématu zaujímá roli SP zatímco IdP je autentizační server pod správou ČVUT v Praze. Samotný proces autentizace uživatele (viz Obrázek 4.2) probíhá ve 4 krocích:

1. SP detekuje, že se uživatel snaží přistoupit k chráněným datům.
2. SP vygeneruje autentizační požadavek, který spolu s uživatelem odešle IdP.
3. IdP provede autentizaci uživatele a odešle autentizační odpověď spolu s uživatelem zpět SP.
4. SP ověří autentizační odpověď ze strany IdP a povolí uživateli přístup k požadovaným chráněným datům.

V těle autentizační odpovědi, kterou systém NURIS obdrží od IdP v kroku 3, je uvedena identita uživatele. Autentizace uživatele přes fakultní přihlášení však ještě neznamená, že je uživatel oprávněn využívat systém NURIS. Takové právo připadá pouze studentům, kteří mají v daném semestru

zapsaný předmět MI-NUR: Návrh uživatelského rozhraní a samozřejmě také vyučujícím tohoto předmětu. Z tohoto důvodu byl implementován KOS Data Downloader.

4.3.4.2 KOS Data Downloader

KOSDataDownloader.cs je třída ve vedlejším projektu KOSDataService. V okamžiku, kdy se daný uživatel (student nebo vyučující) pokouší poprvé vstoupit do systému NURIS, jsou pomocí aplikační logiky této třídy aktualizována data z KOSu. Jak je zmíněno výše, právo na přístup náleží pouze studentům, kteří mají v daném semestru zapsán předmět MI-NUR, a vyučujícím tohoto předmětu. Pokud je uživatel oprávněn využívat systém NURIS, jsou jeho data uložena do databáze, aby jeho příští přihlášení již nemuselo být ověřováno pomocí KOSu (což se v některých případech ukázalo jako velmi pomalé, v řádu jednotek a někdy i desítek sekund). Pro uživatele, kteří nejsou oprávněni užívat systém NURIS, nebudou i přes autentizaci pomocí ShibbolethSSO načtena žádná aplikační data.

4.3.4.3 Udělení autorizačního tokenu

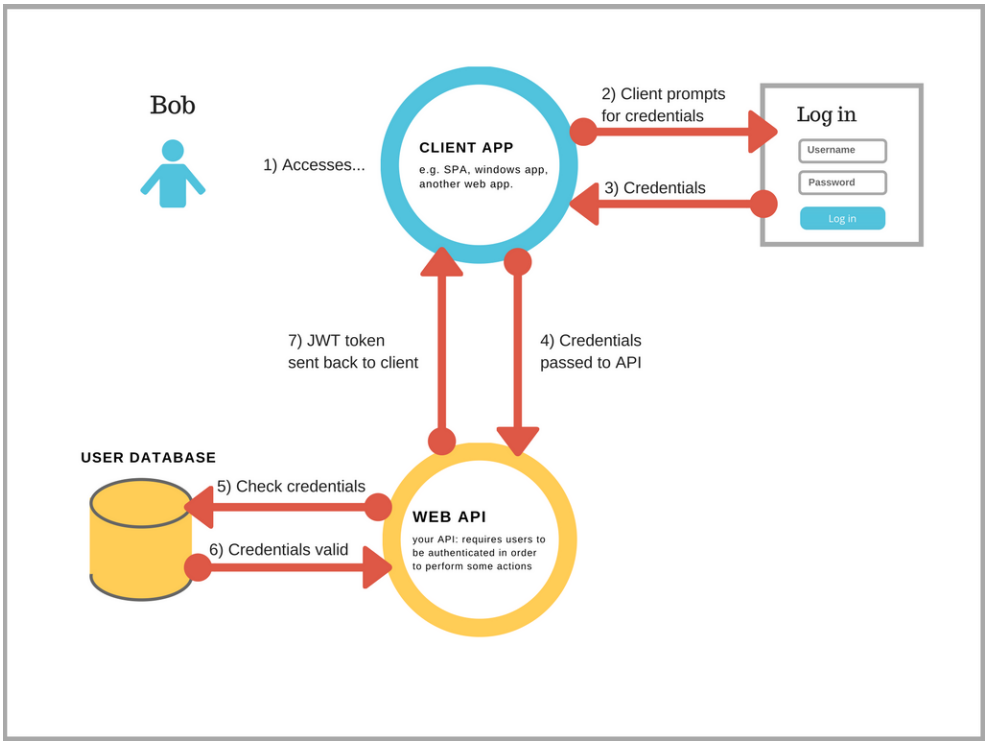
V případě, že je uživatel ověřen pomocí ShibbolethSSO i přes KOS, dojde k přesměrování na adresu /api/login, která je obsluhována třídou LoginController.cs. Zde je uživateli vystaven autentizační token, v kterém je zakódována identita uživatele. S použitím tohoto tokenu je uživatel oprávněn využívat odpovídající REST API resources. Celý proces autentizace a autorizace je ilustrován na obrázku 4.3. V kontextu tohoto obrázku zastává sekci „Log In“ přihlášení pomocí SihholethSSO. V případě, že uživatel není v databázi nalezen, je kontaktován systém KOS.

4.4 Architektura frontendu

Aplikace vyvíjené pomocí technologie Angular Universal se po vzoru jiných objektově-orientovaných jazyků vyznačují vnitřním hierarchickým uspořádáním. Celou aplikaci tvoří tzv. moduly, které jsou složeny z tzv. komponent. Každá komponenta je zpravidla tvořena třemi soubory:

- souborem html, kde je definován vzhled komponenty,
- souborem css, kde jsou definovány kaskádové styly komponenty,
- souborem ts (TypeScript), kde se nachází aplikační logika komponenty.

Frontend systému NURIS je tvořen jediným modulem app.module složeným z několika komponent, z nichž dvě lze označit za zcela stěžejní.



Obrázek 4.3: Schéma autentizace a autorizace systému NURIS [17]

4.4.1 Hlavní komponenty

Jak bylo zmíněno výše i v sekci 3.5, jádro klientské části aplikace tvoří dvě angular komponenty, konkrétně projectlist.component a project.component.

4.4.1.1 Komponenta Projectlist

První zmíněná komponenta zajišťuje zobrazení seznamu projektů a akce s ním spojené. Při vstupu do aplikace je automaticky zobrazena právě tato komponenta. Všem uživatelům umožňuje ze seznamu, který lze filtrovat, vybrat projekt, jehož detail chce uživatel zobrazit. Studentům navíc poskytuje možnost založit nový projekt, ale pouze v případě, že již nejsou členy nějakého projektu. Pokud jsou, je jejich projekt zvýrazněn a zobrazen v seznamu jako první. Vyučujícím nabízí možnost smazat veškerá aktuální data a založit nový semestr.

Při zakládání nového semestru jsou použita data uložená v konfiguračním souboru *appsettings.json*.

4.4.1.2 Komponenta Project

Druhá komponenta (project.component) zajišťuje zobrazení a akce spojené s detailem konkrétního projektu. Nabízí možnost spravovat členy týmu, nahrávat či mazat soubory odevzdání a potvrzovat splnění odevzdání. Student z jiného projektu má možnost se přidat (a následně odebrat) jako člen i jako tester. Zakládající student má možnost projekt smazat. Vyučující má možnost „předhodnocovat“ částečné odevzdání a udělovat konečné hodnocení.

4.4.2 Stav projektu

Akce, které jsou uživateli dostupné, jsou závislé na aktuálním stavu projektu. Každý projekt se vždy nachází v jednom ze čtyř následujících stavů:

- „Čeká na zhotovení“ – Projekt čeká na zhotovení odevzdání ze strany studentů. Ti mohou nahrávat a mazat svá řešení, potvrdit odevzdání. Potvrzením přechází projekt do stavu „Čeká na schválení“. Vyučující může provádět „předhodnocení“.

- „Čeká na schválení“ – Projekt čeká na hodnocení ze strany vyučujícího. Vyučující může přidělit body a schválit odevzdání, čímž projekt přesune do stavu „Schváleno“. Druhou možností vyučujícího je odevzdání zamítnout a tím projekt přesunout do stavu „Vráceno k přepracování“.

- „Schváleno“ – Jakmile je projektu přidělen status „Schváleno“, je automaticky posunut do další fáze (iterace) a zpět do stavu „Čeká na zhotovení“.

- „Vráceno k přepracování“ – Opravňuje všechny uživatele ke stejným akcím jako stav „Čeká na zhotovení“.

Pohybem mezi těmito stavy je zajištěn workflow celého projektu.

4.5 Parametry serveru a nasazení

Pro nasazení systému NURIS přidělilo oddělení ICT FIT ČVUT autorovi této práce virtuální server s následujícími parametry:

- procesor: Intel Xeon E5-2630 @2.30Ghz, přiděleny 2 jádra
- operační paměť: 4GiB DIMM DRAM EDO
- pevný disk: 50GiB SCSI virtual HDD
- síťová karta: 82545EM Gigabit Ethernet Controller (Copper)

Jako operační systém byla po konzultaci s kolegou Bc. Pavlem Kovářem zvolena distribuce `GNU/Linux Debian 4.9.110-3+deb9u2 x86_64`. Po konfiguraci síťového připojení, instalaci všech potřebných knihoven, instalaci webserveru NGINX a konfiguraci ShibbolethSSO byl systém NURIS konečně nasazen v produkčním prostředí.

4.6 Prvotní problémy s autentizací ShibbolethSSO

Ukázalo se ale, že po úspěšné autentizaci uživatelů pomocí ShibbolethSSO (viz sekce 4.3.4.1) dojde v systému NURIS k chybě. Po několika dnech ladění bylo zjištěno, že virtuální server Kestrel, který ASP.NET Core aplikace využívají k běhu, není schopný přijmout HTTP odpověď s hlavičkami, které obsahují diakritiku. Ukázalo se, že se tato chyba navíc objevuje pouze v případě, že je aplikace hostována na linuxovém serveru, tudíž ji nebylo možné při ladění na lokální vývojové stanici s Windows najít. Následující úryvek ukazuje, jak vypadá formát HTTP odpovědi (resp. jejích hlaviček), která je po úspěšné autentizaci pomocí ShibbolethSSO zaslána LoginControlleru (viz sekce 4.3.4.3) systému NURIS. Osobní data byla v tomto úryvku anonymizována.

```
"GET /api/Login HTTP/1.1
Connection: keep-alive
Host: minur.fit.cvut.cz
X-Forwarded-For: xxx.xxx.xxx.xxx
X-Forwarded-Proto: https
accept:
↪ text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
origin: https://idp2.civ.cvut.cz
referer: https://idp2.civ.cvut.cz/idp/profile/SAML2/Redirect/SSO
accept-encoding: br, gzip, deflate
user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14)
↪ AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0
↪ Safari/605.1.15
accept-language: cs-cz
cookie: _shibsession_xxx=xxx
AUTH_TYPE: shibboleth
REMOTE_USER: xxx
Shib-Application-ID: default
Shib-Authentication-Instant: 2018-10-11T18:33:51.545Z
Shib-Authentication-Method:
↪ urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
Shib-AuthnContext-Class:
↪ urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
Shib-Handler: https://minur.fit.cvut.cz/Shibboleth.sso
Shib-Identity-Provider: https://idp2.civ.cvut.cz/idp/shibboleth
```

```
21 Shib-Session-ID: xxx
22 Shib-Session-Index: xxx
23 affiliation: student@cvut.cz;member@cvut.cz
24 cn: xxx
25 commonNameASCII: xxx
26 e-mailAddress: xxx@fit.cvut.cz
27 entitlement: urn:mace:terena.org:tcs:escience-user;
   ↪ urn:mace:dir:entitlement:common-lib-terms;
   ↪ urn:mace:terena.org:tcs:personal-user
28 eppn: xxx@cvut.cz
29 givenName: xxx
30 mail: xxx@fit.cvut.cz
31 nickname: xxx
32 o: České vysoké učení technické v Praze
33 org-dn: České vysoké učení technické v Praze
34 orgunit-dn: Fakulta informačních technologií
35 ou: Fakulta informačních technologií
36 [...]
```

Z úryvku výše je zřetelně vidět, že HTTP hlavička bude vždy obsahovat diakritiku minimálně v hodnotách klíčů „o“, „org-dn“, „orgunit-dn“ a „ou“, tedy v názvech univerzity a fakulty. Na problém, proč virutální server Kestrel není schopný v linuxovém prostředí zpracovat diakritiku v HTTP hlavičkách, se nepodařilo nalézt přímé řešení. Jediným řešením bylo nakonec změnit konfiguraci ShibbolethSSO tak, aby jediným zasílaným atributem byl „RemoteUser“, jehož hodnota obsahuje fakultní login uživatele. Ostatní uživatelská data, jako jsou jméno a příjmení, bylo nutné na základě získaného loginu zajistit pomocí KOS Data Downloader (viz sekce 4.3.4.2).

4.7 Změny systému v průběhu semestru

Jak bylo zmíněno již v úvodu této práce, systém NURIS byl již od okamžiku nasazení první verze v říjnu 2018 při výuce aktivně využíván. Následkem toho přicházely ze strany studentů, ale především ze strany vyučujícího a zároveň vedoucího této práce Ing. Jiřího Hunky, požadavky na změnu či rozšíření funkcionality. Oproti první nasazené verzi byly postupně provedeny následující úpravy:

- odevzdání více souborů – Původní návrh uvažoval odevzdání vždy jen jednoho souboru, s tím, že v případě potřeby odevzdat více souborů by bylo nutné soubory zkomprimovat do archivu. Na základě požadavku byla tedy provedena změna v systému tak, aby mohli všichni členové projektu nahrávat svá (částečná) odevzdání.

- potvrzení odevzdání – V okamžiku, kdy bylo umožněno odevzdání více souborů, bylo nutné přijít se způsobem, jak označit, že jsou studenti s odevzdanými soubory již spokojeni a chtějí si své odevzdání nechat ohodnotit. Proto vznikl koncept potvrzování odevzdání. Jako čas odevzdání se počítá čas potvrzení odevzdání.

- komunikace pomocí poznámek – Studentům a vyučujícím bylo umožněno komunikovat pomocí krátkých poznámek. Studenti mohou zaznamenat poznámku při potvrzování odevzdání. Vyučující mohou poznámku zaznamenat nejen při udílení hodnocení, ale i při „předhodnocování".

- pevné termíny odevzdání – V původním návrhu bylo studentům umožněno odevzdání i po termínu proto, aby měl hodnotící vyučující možnost uznat i řešení, která nebyla odevzdána včas. Po konzultaci s vedoucím práce byly pak termíny nastaveny jako pevné, tudíž pozdní odevzdání již nebylo možné.

- logování – Veškeré interakce všech uživatelů systému jsou na straně serveru logovány.

- GUI – Z původního, graficky strohého a minimalistického grafického designu byla aplikace převedena do modernějšího, svěžejšího designu, který vychází z návrhu týmu NUR Expert ze zimního semestru 2017/2018.

- řazení projektů v seznamu – Projekty byly původně řazeny jen podle názvu. Nově jsou projekty primárně sdruženy a seřazeny podle aktuálního termínu odevzdání a až sekundárně podle názvu. Pro studenty se navíc projekt, jehož je daný student členem, zobrazuje jako první a navíc zvýrazněný.

- volné termíny odevzdání – Praxe ukázala, že „volné" termíny odevzdání je třeba opět umožnit. V případě, že vyučující zamítl odevzdání studentů po uplynutí termínu, studenti už neměli možnost nahrát do aplikace přepracovanou verzi. V současném stavu je tedy pozdní odevzdání i potvrzení odevzdání možné, včasnost je barevně vyznačena.

- filtrování projektů – Na úvodní obrazovku bylo přidáno pole pro filtrování projektů. Toho využijí zejména vyučující, ale také studenti, kteří hledají tým k testování.

- zpětná dostupnost starších dat – V původním návrhu se v detailu projektu již nadále nezobrazovaly žádné informace k uplynulým iteracím. V současném stavu jsou všechny iterace tzv. „rozbalovací", přičemž ve výchozím zobrazení je rozbalena pouze aktuální iterace. Lze tedy prohlížet i starší iterace včetně odpovídajících odevzdání.

- projekt identifikován URL adresou – Původně bylo ID prohlíženého projektu uloženo v localstorage v prohlížeči uživatele. Pro pohodlnější práci vyučujících, kteří často potřebují prohlížet a hodnotit více projektů najednou (na různých záložkách v prohlížeči), bylo nutné toto chování změnit. Při více otevřených projektech se totiž záznamy v localshotage přepisovaly a nebylo tak reálně možné pracovat s více projekty zároveň. Toto bylo vyřešeno přesunutím ID projektu přímo do URL adresy.
- potvrzovací dialogy – Pro zmenšení rizika „ukliknutí“ byly ke všem akcím přidány potvrzovací dialogy.
- zrušení potvrzení odevzdání – Bylo přidáno tlačítko umožňující studentům zrušit potvrzení odevzdání. Studenti pak mohou nahradit stávající či nahrát nové soubory odevzdání. Touto akcí se však anuluje původní čas odevzdání, ten je opět zaznamenán až při dalším potvrzení.
- bodování studentů s odkazy na testované projekty – Postranní panel s bodováním studentů nyní zobrazuje bonusové body za účast na testování jiného projektu jako odkaz na daný projekt. Tato vlastnost usnadňuje používání aplikace především pro vyučující.
- „předhodnocování“ v jakémkoliv stavu projektu – Vyučujícím bylo umožněno předpřipravit a uložit hodnocení aktuální iterace nehledě na stav, v jakém se projekt aktuálně nachází. Hodnocení není pro studenty viditelné, připojená poznámka ano.
- změna hodnocení uplynulých iterací – Vyučujícím bylo umožněno zpětně měnit bodové hodnocení uplynulých iterací.

Kapitola 5

Vzhled aplikace

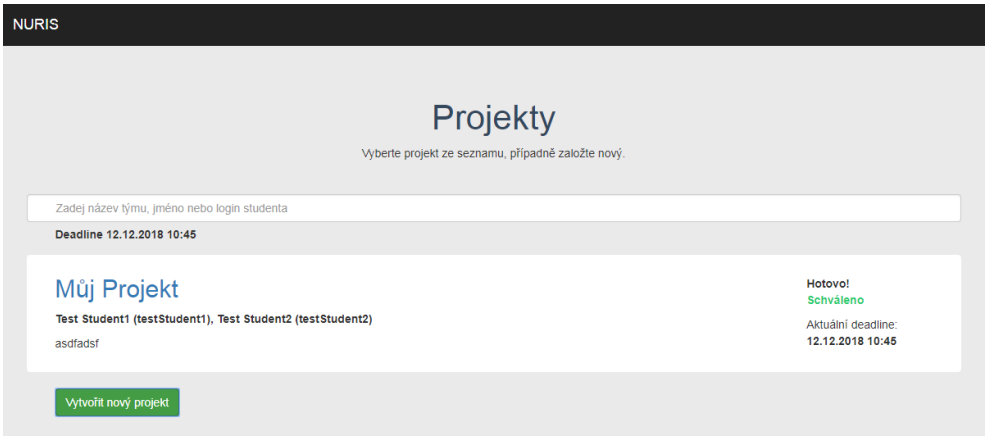
Tuto kapitolu tvoří obrazová dokumentace mapující vzhled aplikace. Je rozdělena na dvě části podle dvou hlavních obrazovek resp. komponent (viz sekce 4.4.1).

Výsledné GUI obrazovky seznamu projektů:

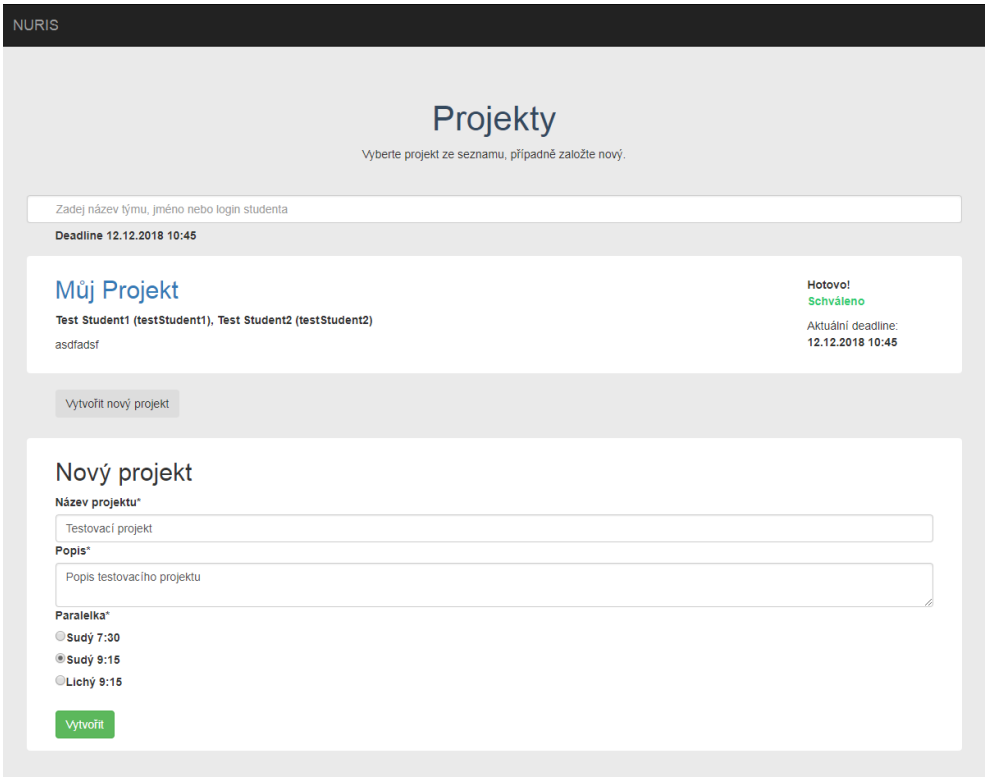
- z pohledu studenta bez projektu (viz Obrázek 5.1),
- z pohledu studenta při vytváření projektu (viz Obrázek 5.2),
- z pohledu studenta náležícího do projektu (viz Obrázek 5.3),
- z pohledu vyučujícího (viz Obrázek 5.4).

Výsledné GUI obrazovky detailu projektu:

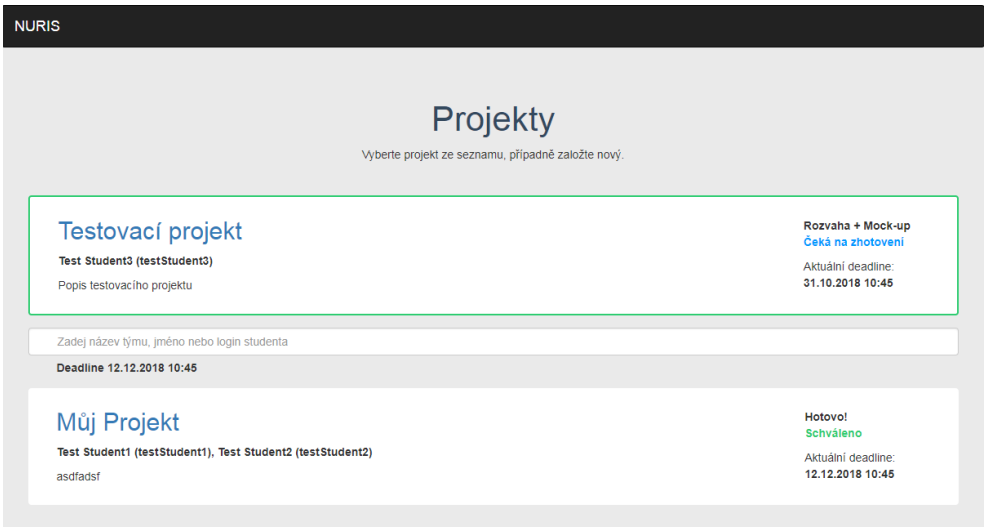
- z pohledu studenta z jiného projektu (viz Obrázek 5.5),
- z pohledu studenta bez projektu (viz Obrázek 5.6),
- z pohledu studenta náležícího do projektu (viz Obrázek 5.7),
- z pohledu studenta náležícího do projektu, potvrzujícího odevzdání (viz Obrázek 5.8),
- z pohledu studenta náležícího do projektu, po potvrzení odevzdání (viz Obrázek 5.9),
- z pohledu vyučujícího (viz Obrázek 5.10),
- z pohledu vyučujícího po potvrzení odevzdání ze strany studentů (viz Obrázek 5.11).



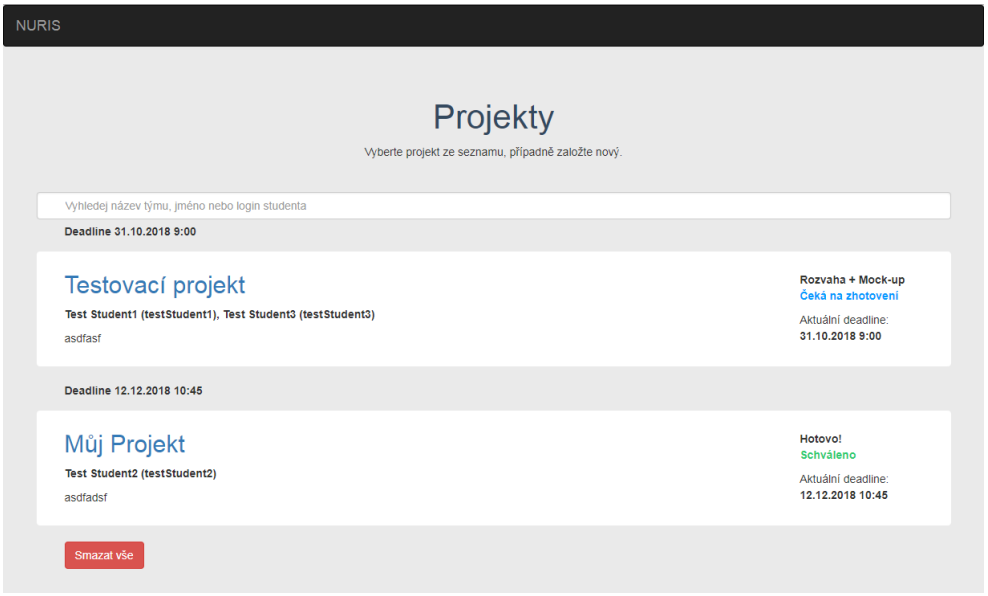
Obrázek 5.1: NURIS GUI: seznam projektů z pohledu studenta bez projektu



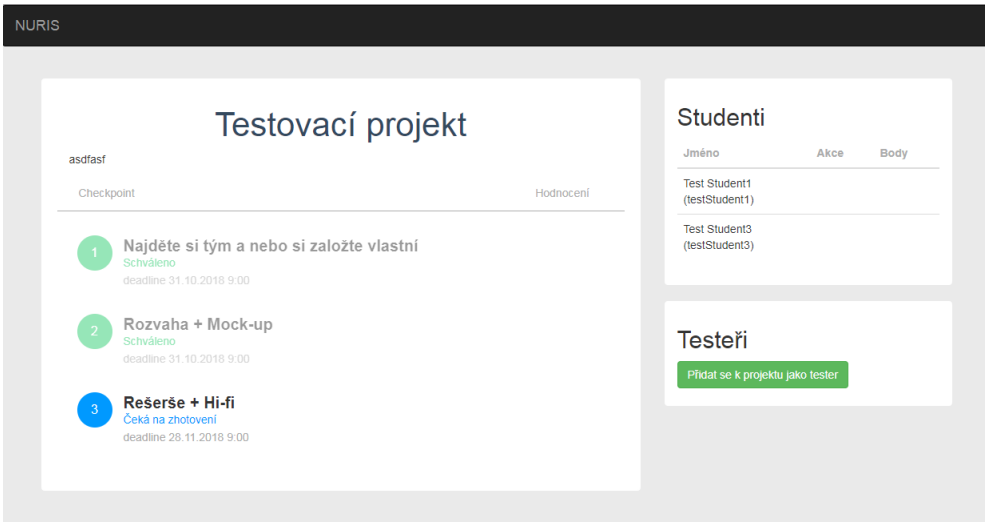
Obrázek 5.2: NURIS GUI: seznam projektů z pohledu studenta vytvářejícího projekt



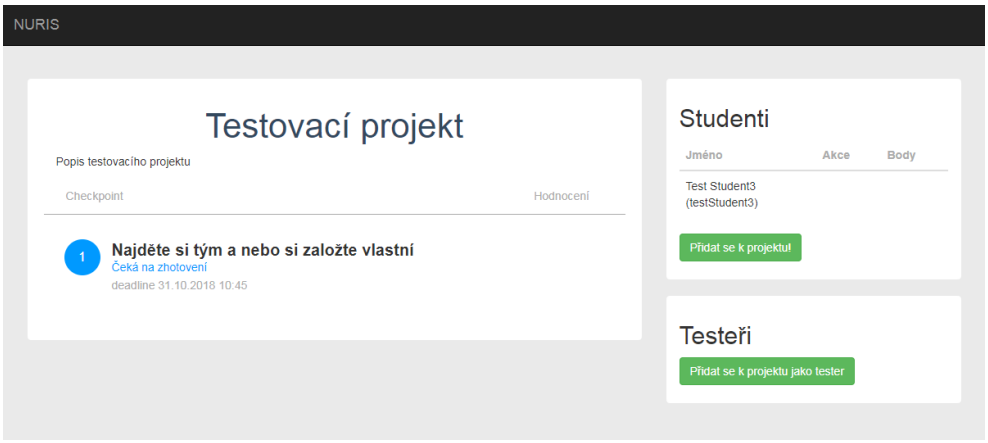
Obrázek 5.3: NURIS GUI: seznam projektů z pohledu studenta s projektem



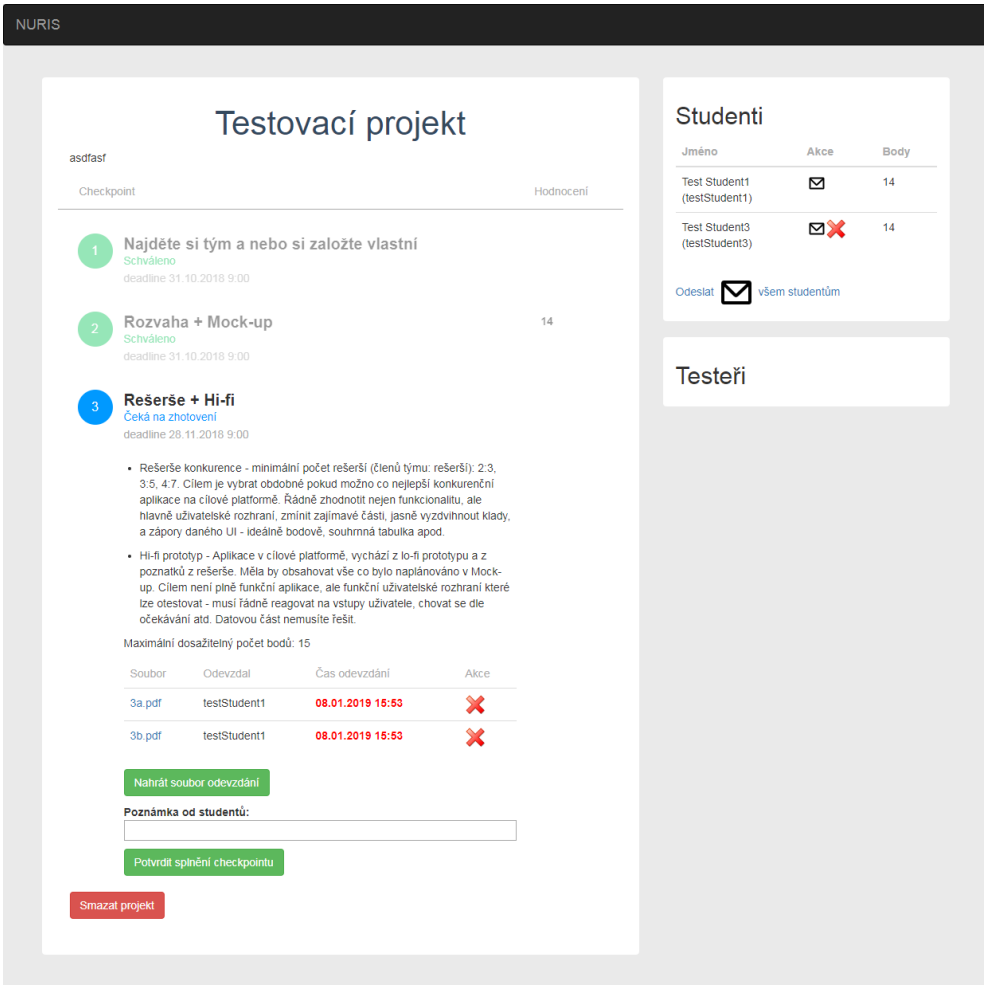
Obrázek 5.4: NURIS GUI: seznam projektů z pohledu vyučujícího



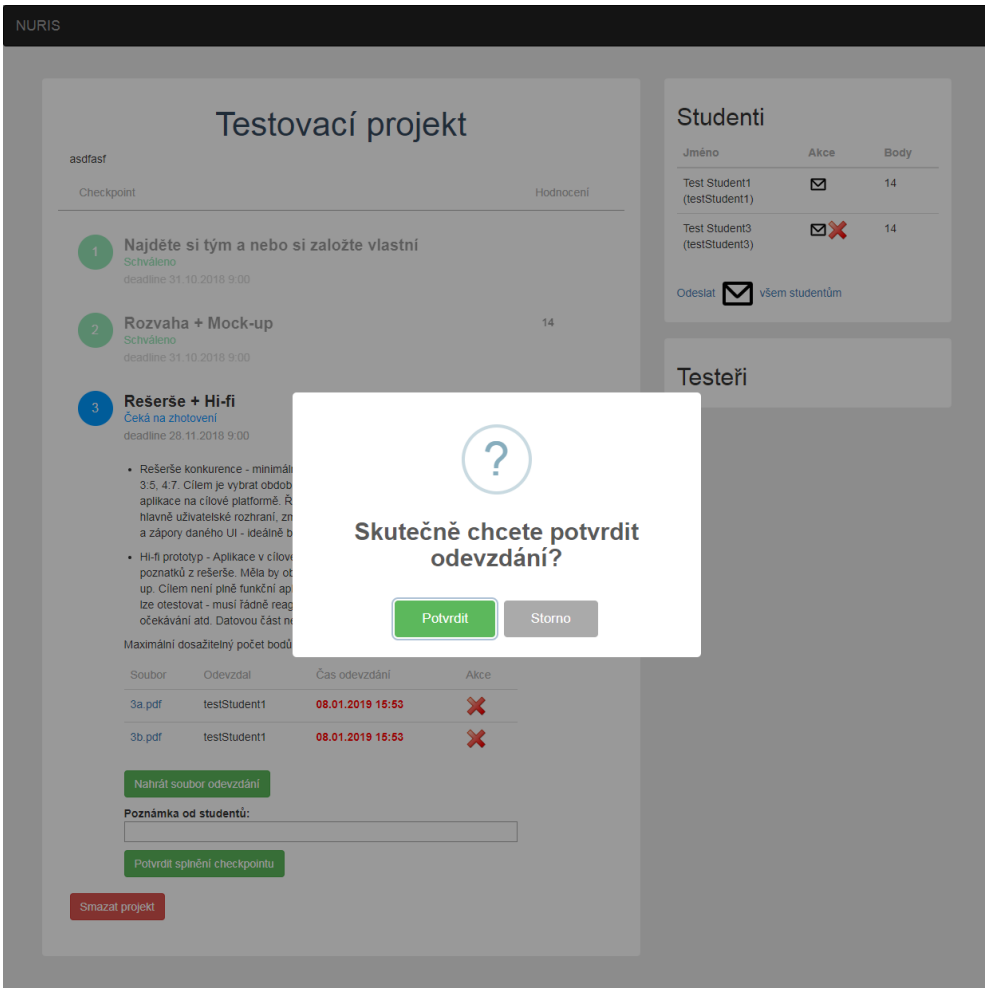
Obrázek 5.5: NURIS GUI: detail projektu z pohledu studenta z cizího projektu



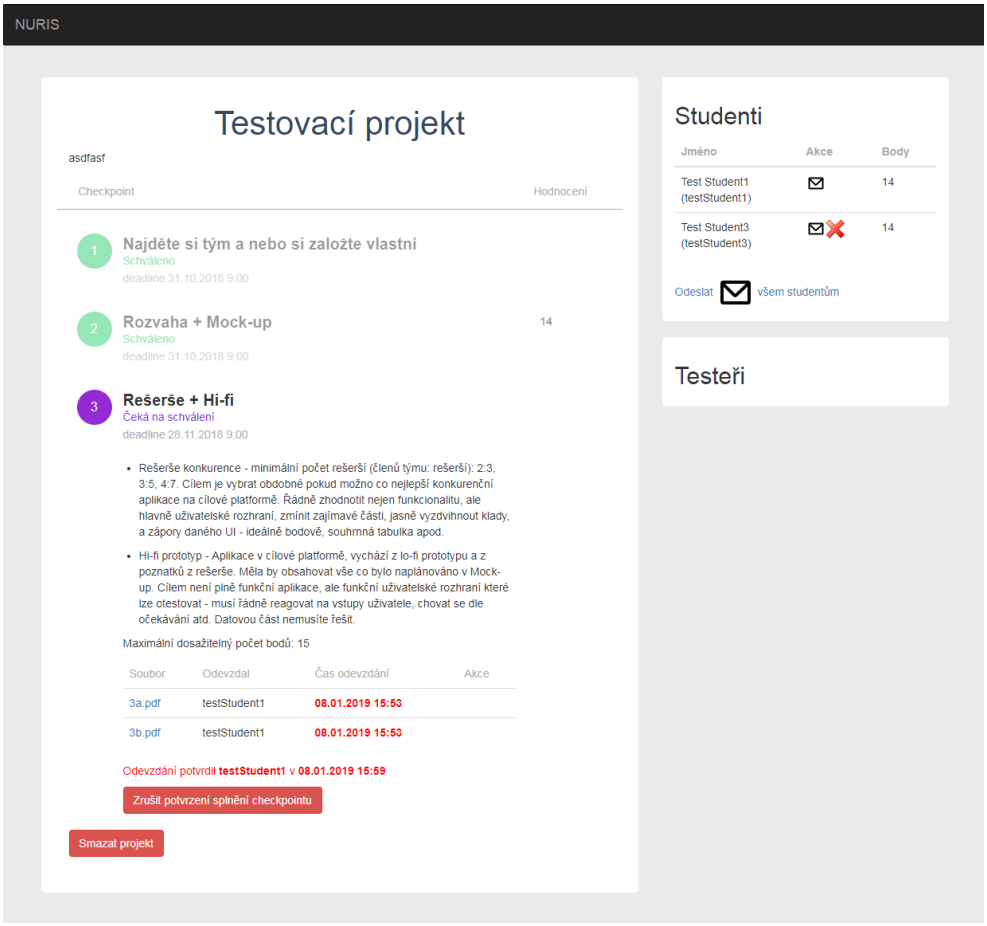
Obrázek 5.6: NURIS GUI: detail projektu z pohledu studenta bez projektu



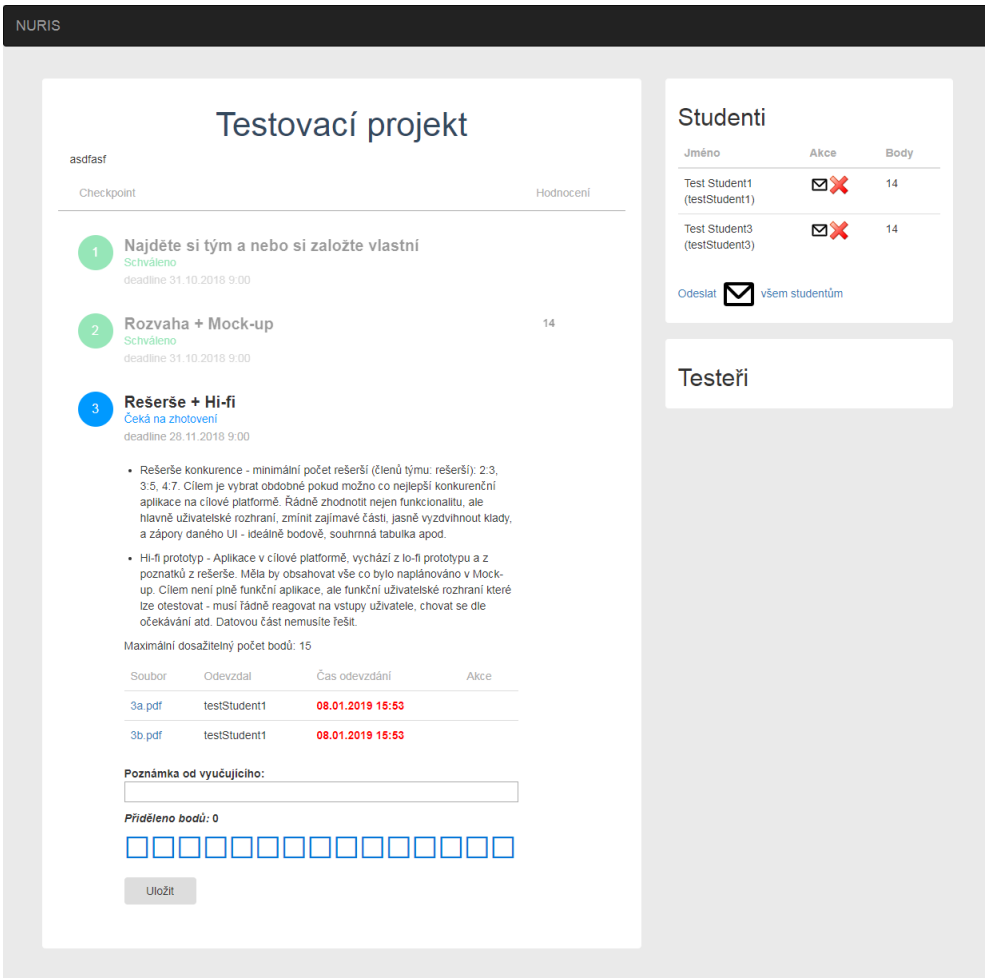
Obrázek 5.7: NURIS GUI: detail projektu z pohledu studenta patřícího do projektu



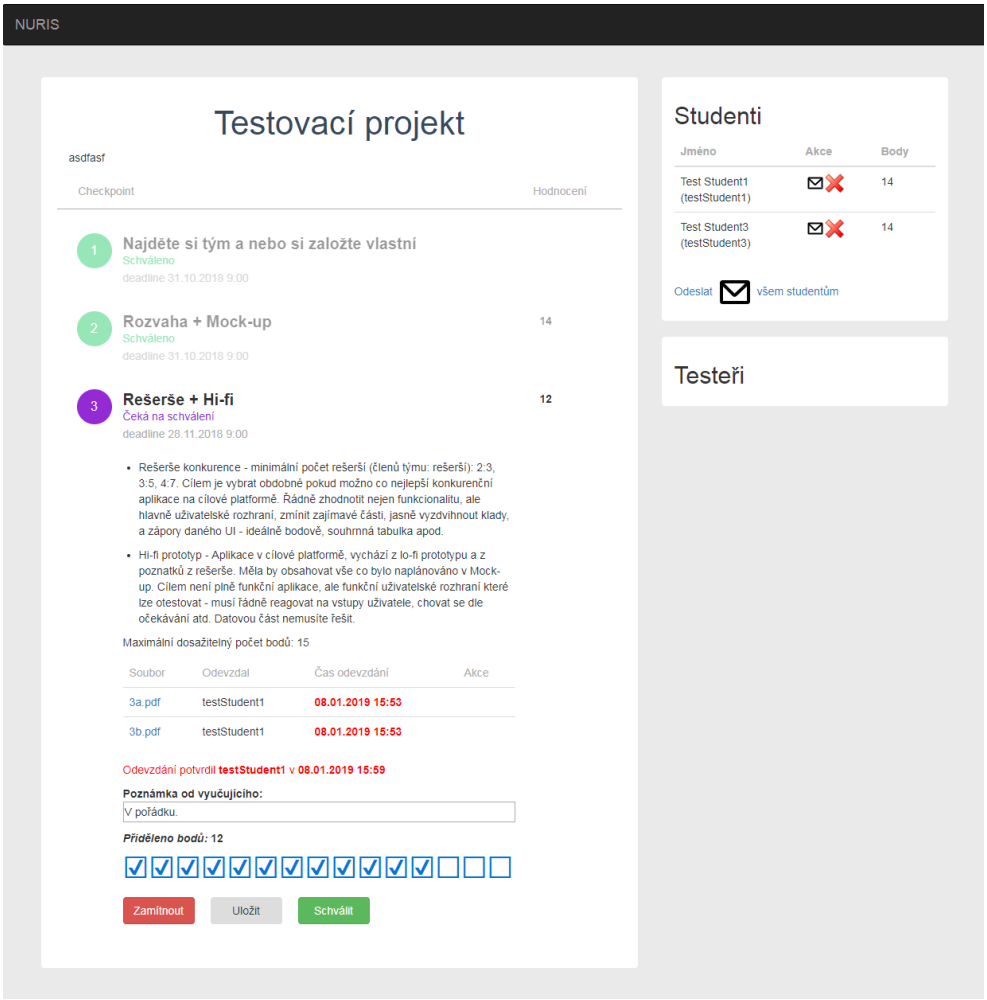
Obrázek 5.8: NURIS GUI: detail projektu z pohledu studenta patřícího do projektu, potvrzujícího odevzdání



Obrázek 5.9: NURIS GUI: detail projektu z pohledu studenta patřícího do projektu, po potvrzení odevzdání



Obrázek 5.10: NURIS GUI: detail projektu z pohledu vyučujícího bez potvrzeného odevzdání



Obrázek 5.11: NURIS GUI: detail projektu z pohledu vyučujícího s potvrzeným odevzdáním

Kapitola 6

Testování

Tato kapitola je věnována testování výsledné aplikace, především pak samotného uživatelského rozhraní.

6.1 Heuristická analýza

Ke zhodnocení uživatelského rozhraní bude jako první použita Nielsenova heuristická analýza. Ta se skládá z následujících pravidel.

6.1.1 Deset pravidel Nielsenovy heuristické analýzy[18]

1. **Visibility of system status** – The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2. **Match between system and the real world** – The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. **User control and freedom** – Users often choose system functions by mistake and will need a clearly marked "emergency exit"to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. **Consistency and standards** – Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. **Error prevention** – Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

6. **Recognition rather than recall** – Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. **Flexibility and efficiency of use** – Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. **Aesthetic and minimalist design** – Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. **Help users recognize, diagnose, and recover from errors** – Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. **Help and documentation** – Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

6.1.2 Vyhodnocení

Uživatelské rozhraní systému NURIS bude zhodnoceno v každém z 10 bodů popsaných v sekci 6.1.1. U každého nálezu bude zhodnocena závažnost daného nedostatku. Rozlišeny budou tři stupně závažnosti:

1. nízká závažnost – problém nemá vliv na použitelnost aplikace

2. střední závažnost – problém může mít vliv na použitelnost aplikace a měl by být odstraněn

3. vysoká závažnost – problém má kritický dopad na použitelnost aplikace a musí být neprodleně odstraněn

Tabulka 6.1: Vyhodnocení heuristické analýzy

Pravidlo	Závažnost	Poznámka
1	střední	Aktuální stav projektu zobrazen v seznamu i v detailu. Při nahrávání větších souborů odevzdání chybí indikace, že upload skutečně probíhá.
2	ok	Hierarchie systému odpovídá realitě.
3	střední	Z detailu projektu vede jediná cesta zpět na seznam projektů přes „home“ tlačítko NURIS vlevo nahoře. Možná by bylo na místě doplnit stránku o tlačítko „zpět na seznam projektů“. Operace mazání jsou ze své podstaty nevratné. Tento fakt je zmírněn použitím potvrzovacích dialogů.
4	ok	Nejsou použity žádné nestandardní ovládací prvky
5	ok	Jediné uživatelské vstupy, kde může dojít k zadání nevalidních dat, jsou při zakládání projektu. To je ošetřeno validací.
6	ok	Veškerá relevantní data jsou vždy viditelná.
7	ok	Ovládacích prvků je minimum, seznam projektů je možné filtrovat
8	ok	Všechny dialogy jsou stručné a výstižné.
9	ok	Zatím nebyl objeven žádný uživatelem způsobený chybový stav.
10	nízká	Systém je natolik jednoduchý, že by jeho ovládání mělo být velice snadné. Jedinou dokumentací je zatím tato práce.

Pomocí Nielsenovy heuristiky bylo dosaženo následujících zjištění:

- Bylo by vhodné přidat nějakou formu indikace, že je odevzdání nahráváno.
- Možná by bylo vhodné přidat na detail projektu tlačítko pro návrat na seznam projektů. Je třeba si ale uvědomit, že při intuitivním umístění tlačítka nad obsahem stránky se zmenší rozsah zobrazených dat pro uživatele s nižším rozlišením zobrazovacího zařízení.
- Možná by bylo vhodné umožnit uživateli vrátit zpět smazaná data. Zde však vyvstává otázka, jaká pravidla by povolení takové operace zahrnovalo a zda by se nedalo nějakým způsobem zneužít.
- Možná by měla být přímou součástí systému NURIS nějaká forma nápovědy.

6.2 Uživatelské testování

Tato sekce je zaměřena na výběr způsobu uživatelského testování a určení optimálního počtu testujících uživatelů.

6.2.1 Způsoby uživatelského testování

Způsobů, jak provádět uživatelské testování, je známo hned několik [19]. Pro účely této práce jsou relevantní dotazníky, pozorování a průchody systémem.

6.2.1.1 Dotazníky

Dotazníky jsou zřejmě nejjednodušší formou uživatelského testování. S jejich pomocí lze od uživatele získat kvalitní zpětnou vazbu. Ta se nemusí nutně skládat jen z hodnocení používání systému, ale může cílit také na další věci, jako například náměty na přidání či změnu funkcionality.

Míra zpětné vazby dotazníku závisí především na kvalitě otázek a na jejich pokrytí veškeré funkcionality aplikace.

6.2.1.2 Pozorování

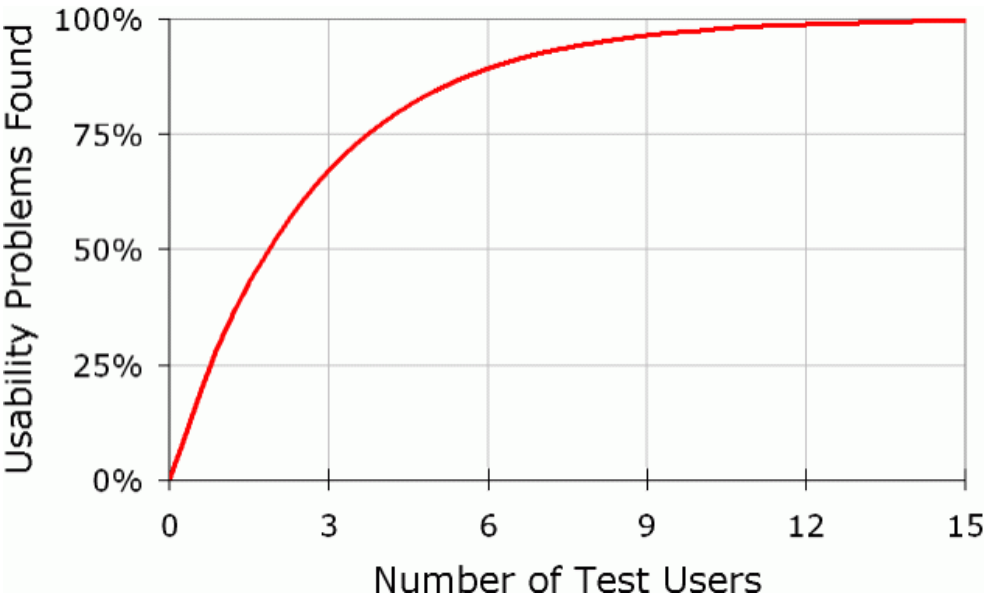
Tento druh uživatelského testování zahrnuje pozorování uživatele při interakci s aplikací. Data obdržená z dotazníku nemusí plně korespondovat s tím, jak se uživatel při práci s aplikací reálně chová. Nejlepší zpětné vazby lze dosáhnout při pozorování uživatele v jeho přirozeném prostředí. Buď je tedy potřeba uživatele pozorovat přímo (osobně), a nebo lze využít techniky a pořizovat audiovizuální záznam testování. Při osobní účasti na testování hrozí ovlivnění uživatele přítomností dohlížející osoby (nervozita) a navíc je osobní účast časově náročná. Přenášení a nahrávání audiovizuálního záznamu je zase poměrně náročné technicky.

6.2.1.3 Průchody systémem

Řízený či neřízený průchod systémem je druh uživatelského testování, při kterém je zaznamenáván obraz monitoru včetně pohybu myši a vstupu z klávesnice. Při řízeném průchodu dostane uživatel předem vytvořený scénář, který následuje. Ten by měl pokrývat všechny běžně užívané úlohy, které aplikace nabízí. Tento způsob testování lze bez problémů provést i vzdáleně přes internet.

6.2.1.4 Zhodnocení a výběr metod testování

S ohledem na časovou a technickou složitost zmíněných metod testování byly pro účely této práce nakonec vybrány metody dotazníku a průchodu systémem.



Obrázek 6.1: Vztah počtu nalezených chyb na počtu testujících uživatelů

6.2.2 Postačující počet testovaných subjektů

Podle Jakoba Nielsena[20] platí pro vztah počtu testujících uživatelů a procentu odhalených chyb rovnice:

$$N = (1 - (1 - L)^n)$$

kde N značí procento nalezených problémů, L udává v procentech počet problémů nalezených během jednoho testování a n je počet testujících uživatelů. Podle J.Nielsena je hodnota L průměrně 31%. Po dosazení je získán graf 6.1. Efektivitu testování lze navíc posílit rozdělením testování na více vln. Při 7 testujících uživatelích, 3 v první vlně a 4 ve druhé, vychází po dosazení do rovnice:

$$(1 - (1 - 0,31)^3) = 67,15\%$$

$$(1 - (1 - 0,31)^4) = 77,33\%$$

$$67,15 + 77,33 * (100 - 67,15) = 92,55\%$$

6.3 První vlna uživatelského testování

Jak bylo zmíněno v sekci 6.2.2, první vlny uživatelského testování se zúčastnili 3 testující subjekty. Subjektům byl poskytnut archiv s pěti testovacími soubory (1, 2a, 2b, 3a, 3b).

6.3.1 Výběr testujících subjektů

Pro první vlnu testování byli testující subjekty vybrány z řad studentů Fakulty informačních technologií a Fakulty elektrotechnické ČVUT v Praze. Byli vybráni studenti, kteří zatím nestudovali předmět MI-NUR: Návrh uživatelského rozhraní, a to z důvodu, aby nebyli ovlivněni jakoukoli předchozí znalostí či zkušeností z tohoto předmětu. Testujícím subjektům byl poskytnul stručný a výstižný popis předmětu a především průběhu semestrální práce.

6.3.2 Scénář

Pro účely uživatelského testování byl vytvořen následující scénář:

1. Tvorba týmů
   a) Založte si nový projekt (název, popis a paralelku si vymyslete).
   b) O Váš projekt není zájem, smažte ho a přidejte se k projektu TestProjectX.
   c) Se složením týmu jste spokojen(a), potvrďte ho.
2. Vyčkejte na reakci vyučujícího.
3. Rozvaha + Mock-up
   a) S týmem jste zpracovali aktuální checkpoint, nahrajte soubor odevzdání 1 a to posléze potvrďte.
4. Vyčkejte na reakci vyučujícího.
   a) S týmem jste zpracovali aktuální checkpoint, nahrajte soubor odevzdání 2a a to posléze potvrďte
   b) Zapomněli jste s týmem připojit poznámku, že své "řešení, pokud bude potřeba, vysvětlíte osobně". Napravte to.
   c) Vyčkejte na reakci vyučujícího.
   d) Provedli jste s týmem požadovanou opravu. Nahraďte starý soubor odevzdání 2a novým souborem 2b a potvrďte s poznámkou "opraveno".
5. Vyčkejte na reakci vyučujícího.
6. Domluvil(a) jste se, že budete týmu YYY dělat testera. Zaznamenejte to do systému.
   a) Do Vašeho projektu TestProjectX nahrajte částečné řešení 3a.
   b) Vyčkejte na reakci vyučujícího.
   c) Nahrajte částečné řešení 3b.

   d) Takhle jste již spokojeni, nechte si odevzdání ohodnotit

7. Vyčkejte na reakci vyučujícího.

8. Zjistěte, kolik jste celkem za semestr získal(a) bodů.

6.3.3 Výsledky

Po splnění scénáře byl s testujícími subjekty ještě proveden krátký rozhovor. Dotazy směřovaly především na to, jak se subjektům systém NURIS používal, jestli bylo vše dostatečně přehledné a zdali mají nějaký námět na zlepšení. Z tohoto rozhovoru spolu s pozorováním testujících při plnění daného scénáře byly vyvozeny následující výsledky:

1. První testující subjekt

   - subjekt hledal tlačítko „nový projekt“ na obrazovce se seznamem projektů někde nahoře
   - subjektu připadaly barvy tlačítek u potvrzovacích dialogů lehce matoucí

2. Druhý testující subjekt

   - subjekt vyjádřil připomínku umístit tlačítko „Vytvořit nový projekt“ v seznamu projektů někde nahoře
   - subjekt zmínil, že design ikonek (mail a delete) v detailu projektu je trochu nejednotný
   - subjekt si všiml, že chybí potvrzovací dialog při přidávání k týmu jako člen i jako tester
   - v některých případech byla barva tlačítek potvrzovacího dialogu pro subjekt matoucí, především u zrušení potvrzení odevzdání
   - tlačítko „Smazat projekt“ připadalo subjektu příliš blízko k ostatním běžně používaným ovládacím prvkům
   - subjekt si nevšiml, že jsou uplynulé iterace rozbalovací

3. Třetí testující subjekt

   - subjekt očekával tlačítko „nový projekt“ někde nahoře, zmínil, že se dá špatně najít
   - subjekt navrhl vytvořit tabulku s archivem poznámek u iterací
   - text ve vyhledávacím poli přišel subjektu lehce matoucí

6.3.4 Úpravy systému po první vlně testování

Po první vlně testování byly v aplikaci provedeny následující změny:

- Tlačítko „Vytvořit nový projekt“ bylo zvětšeno a umístěno nad seznam projektů.
- Potvrzovací dialog byl doplněn k přidávání se k týmu jako člen i jako tester.
- Barvy tlačítek potvrzovacích dialogů byly změněny. Potvrzovací tlačítko má nyní zelenou resp. červenou barvu podle toho, zda se jedná o kladnou (přidání uživatele nebo testera, nahrávání, potvrzení) či zápornou (mazání, zrušení odevzdání) volbu. Tlačítko „Zrušit“ je vždy šedé.
- Byl upraven text ve vyhledávácím poli (filtru).
- Rozbalitelnost uplynulých iterací na obrazovce detailu projektu je nyní indikována změnou kurzoru.
- Tlačítko „Smazat projekt“ bylo zvětšeno a umístěno nad seznam projektů.

6.4 Druhá vlna uživatelského testování

Druhé vlny uživatelského testování se zúčastnili 4 testující subjekty, tedy o jeden subjekt více než v první vlně.

6.4.1 Výběr testujících subjektů

Do druhé vlny byli taktéž vybráni studenti technických oborů ČVUT, kteří nemají s předmětem MI-NUR: Návrh uživatelského rozhraní žádné zkušenosti. Vyběr testujících subjektů tedy probíhal stejným způsobem jako v první vlně.

6.4.2 Scénář

Vzhledem k tomu, že scénář z první vlny vcelku důkladně pokrývá veškeré běžné uživatelské úlohy a situace, s nimiž se student MI-NUR během semestru setká, byl scénář z první vlny zachován i pro druhou vlnu uživatelského testování.

6.4.3 Výsledky

Z druhé vlny testování byly jakožto zpětná vazba získány následující poznámky:

1. První testující subjekt

- subjekt měl problémy najít přechod z detailu projektu zpět na seznam
- vzhledem k velmi špatnému internetovému připojení byla odezva systému na nahrání souboru dlouhá, což bylo pro subjekt matoucí

2. Druhý testující subjekt

   - subjekt měl problémy najít tlačítko pro přechod zpátky na seznam projektů
   - subjekt zmínil jako námět zobrazovat v tabulce souborů odevzdání i již smazané soubory

3. Třetí testující subjekt

   - subjekt očekával nějaké potvrzení po úspěšném nahrání souboru
   - na konci testování však zmínil, že po odevzdání prvního souboru a pochopení, jak systém funguje, mu v tomto ohledu vše přijde v pořádku

4. Čtvrtý testující subjekt

   - subjekt se automaticky pokoušel odevzdat první soubor metodou drag&drop

6.5 Dotazník

Za účelem získání zpětné vazby od studentů, kteří již systém NURIS v zimním semestru 2018/2019 při výuce používali, byl vytvořen dotazník. Ten byl na konci semestru, tj. po dokončení semestrálních projektů, rozeslán studentům MI-NUR. Vzhledem k tomu, že se systém NURIS během semestru stále vyvíjel, byla součástí dotazníku obrazová dokumentace aktuální verze aplikace.

Sbírání dat pomocí dotazníku probíhalo souběžně s uživatelským testováním aplikace.

6.5.1 Zadání

Až na výjimky byly otázky kladeny tak, aby měl student v případě negativní odpovědi možnost uvést důvod své nespokojenosti. Dotazník měl následující podobu:

1. Zakládal(a) jste projekt, nebo jste se připojil(a) k již existujícímu projektu?

   a) Pro zakládající: Bylo pro Vás zakládání projektu snadné a intuitivní?

   b) Pro připojující se: Bylo pro Vás nalezení projektu a následné připojení se k němu snadné a intuitivní?

2. Připadá vám základní obrazovka se seznamem projektů přehledná?
3. Připadá vám obrazovka s detailem projektu přehledná?
4. Bylo na první pohled zřetelné, v jakém stavu se aktuálně projekt nachází a zda je od Vás aktuálně vyžadována nějaká interakce?
5. Bylo na první pohled vidět, kolik máte aktuálně bodů?
6. Bylo na první pohled vidět, kolik máte času na vyhotovení aktuálního odevzdání (kdy je další deadline)?
7. Poskytl Vám systém ke každému checkpointu dostatek informací k vyhotovení odevzdání?
8. Byla pro Vás tabulka se soubory odevzdání dostatečně přehledná?
9. Připadal Vám koncept potvrzování odevzdání jasný a srozumitelný?
10. Stalo se Vám, že jste potřebovali zrušit potvrzení odevzdání?
11. Připadala Vám komunikace s vyučujícím pomocí poznámek při potvrzování odevzdání dostatečná?
12. Všiml(a) jste si, že již zhotovené a ukončené checkpointy lze na obrazovce detailu projektu 'rozbalit' a zpětně prohlédnout?
13. Bylo pro Vás snadné najít v systému tým, který jste chtěl(a) testovat?
14. Bylo pro Vás snadné připojit se k nalezenému týmu jako tester?
15. Je pro Vás obrazovka zakončující životní cyklus projektu jasná a pochopitelná? Je z této obrazovky jasné, že z Vaší strany není již nadále vyžadována žádná interakce?
16. Pokud máte nějaký další námět nebo připomínku, co Vám v systému chybělo či chybí, zde máte možnost se vyjádřit...

6.5.2 Zpětná vazba dotazníku

Na následujících řádcích jsou čtenáři představeny poznámky, které byly získány jako zpětná vazba z dotazníku. Získaná data byla rozdělena podle toho, zda se jedná o nedostatek či o námět na změnu či přidání funkcionality. Nahlášené nedostatky:

1. Založení nového projektu: „Klikl jsem na vytvorit projekt a vubec jsem si nevsiml ze mi nekde na spodu stranky vyskocil formular. Zaroven jsem to posilal prednasejicimu jak priklad spatneho UI pro bonusove body.“
    - Již opraveno. Tlačítko „Vztvořit nový projekt“ bylo již přesunuto nad seznam projektů a tím byl problém vyřešen.
2. Obrazovka se seznamem projektů: „Mix vlastního projektu s ostatními, které mě v 99% případů nebudou vůbec zajímat.“
    - Již opraveno. Aktivní projekt je vždy zobrazuje zvýrazněný a na prvním místě.
3. Obrazovka s detailem projektu: „Dlouhé texty s popisem zadání. Když jdu odevzdat úkol, už mám zadání asi splněné.“
4. Obrazovka s detailem projektu: „Slovní komentář vyučujícího by měl být výraznější. Opět jsem si ho nevšiml. Jinak OK.“
    - Již opraveno. Komentáře byly zvýrazněny.
5. Obrazovka s detailem projektu: „Šedý text odevzdaných částí blbě čitelný. Mohl by dostat black barvu po rozkliknutí“
6. Workflow: „Nezobrazuje se progress bar při nahrávání souboru“

Náměty na změnu či rozšíření funkcionality:

1. Zakládání projektu: „Cool by bylo načíst z KOSu, do jaké paralelky patřím, a tu předvybrat.“
2. Workflow: „Odkaz na vzorovou práci by určitě neuškodil. Takhle jsem to musel dohledávat na eduxu.“
3. Workflow: „Bylo dobré mít ukázku u vyhotovení“
4. Workflow: „Input Box na poznámky bych zvětšil na Text Area.. když člověk píše delší poznámku, tak tam prakticky nic nevidí“
5. Ostatní náměty: „Emailové notifikace, logo, sjednocení názvu systému a URL, na které běží.“
6. Ostatní náměty: „Odstranění studenta, který se z týmu odpojil.“
    - Odstranit neaktivního studenta může jakýkoli jiný člen týmu nebo vyučující.

Po sečtení všech odpovědí ohledně spokojenosti s jednotlivými uživatelskými akcemi vychází průměrná spokojenost studentů (čistě na základě dotazníku) na 89,8%. Je však třeba znovu připomenout, že byl systém NURIS během celého semestru ve vývoji a studenti tak neprováděli hodnocení nejnovější verze systému.

6.6 Vyhodnocení druhé vlny testování a dotazníku

V druhé vlně byl objeven jediný znatelnější nedostatek. Tím se ukázal být přechod z obrazovky detailu projektu zpět na seznam projektů. Tento nedostatek byl již zaznamenán a posouzen v rámci vyhodnocení heuristické analýzy (viz sekce 6.1.2). Přidání tlačítka pro návrat na seznam projektů (mimo již existujícího loga NURIS s funkcionalitou „domů“ vlevo nahoře) je však například v přímém rozporu s nedostatkem č.3 z dotazníku.

Řada drobných nedostatků zaznamenaných v dotazníku byla již dříve vyřešena (viz poznámky přímo v dotazníku). Jediný nedostatek či námět, na kterém se shodlo více studentů, je zahrnutí ukázky vyhotovení do popisu aktuální iterace. Aktuální ukázkové řešení je umístěno v zanikající aplikaci EDUX (`https://edux.fit.cvut.cz/`) v osobním prostoru jednoho ze studentů. Vzhledem k tomu, že cílem této práce je pro výuku předmětu MI-NUR zcela nahradit EDUX, bude pro přesun a publikování souborů v systému NURIS potřeba zajistit souhlas autora/autorů.

Závěr

Cílem této diplomové práce bylo vyvinout NURIS – informační systém pro správu předmětu MI-NUR: Návrh uživatelského rozhraní. To obnášelo analyzovat potřeby studentů i vyučujících předmětu MI-NUR a na základě této analýzy navrhnout, naimplementovat a nasadit výslednou aplikaci na fakultní server. Tento cíl byl splněn.

Systém NURIS byl navíc od okamžiku prvního nasazení v říjnu 2018 při výuce aktivně využíván. V průběhu zimního semestru 2018/2019 se tak postupně objevovaly další funkční požadavky, ať už ze strany studentů, tak především ze strany vyučujícího a zároveň vedoucího této práce Ing. Jiřího Hunky. Tyto požadavky byly průběžně zpracovávány a systém NURIS byl pravidelně aktualizován.

Na konci zimního semestru bylo provedeno uživatelské testování aplikace s nezávislými testujícími subjekty z řad studentů technických oborů ČVUT v Praze. Další testování uživatelského rozhraní bylo provedeno se studenty předmětu MI-NUR pomocí dotazníku. Závažnější nedostatky, které byly objeveny při testování, byly stejně jako většina drobných odstraněny. Zbytek poznámek a námětů získaných testování je ponechán k diskuzi a otevírá prostor dalšímu vývoji aplikace.

Literatura

[1] Ing. Hunka, J.: MI-NUR: Návrh uživatelského rozhraní. [online], říjen 2018, [cit. 2019-01-03]. Dostupné z: https://courses.fit.cvut.cz/MI-NUR/classification/index.html

[2] Wasson, M.: ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET. [online], [cit. 2019-01-03]. Dostupné z: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

[3] Duveau, L.: ASP.NET Core 2.1 and Angular 5 in Visual Studio 2017. [online], [cit. 2019-01-03]. Dostupné z: https://www.slideshare.net/lduveau/aspnet-core-21-and-angular-5-in-visual-studio-2017

[4] Levkovsky, M.: Thinking in Redux (when all you've known is MVC). [online], [cit. 2019-01-03]. Dostupné z: https://hackernoon.com/thinking-in-redux-when-all-youve-known-is-mvc-c78a74d35133

[5] Sulaiman, A.: File System vs. Database. [online], [cit. 2019-01-03]. Dostupné z: https://dzone.com/articles/which-is-better-saving-files-in-database-or-in-fil

[6] neznámý, A.: DB-Engines Ranking. [online], [cit. 2019-01-03]. Dostupné z: https://db-engines.com/en/ranking

[7] neznámý, A.: Graph DBMS. [online], [cit. 2019-01-03]. Dostupné z: https://db-engines.com/en/article/Graph+DBMS

[8] Ögr. Gör. Erkan HÜRNALI: ASP.NET vs ASP.NET Core. [online], [cit. 2019-01-05]. Dostupné z: https://www.slideshare.net/ehurnali/aspnet-vs-aspnet-core

[9] Nick Anderson, M. W.: Tutorial: Create a web API with ASP.NET Core MVC. [online], [cit. 2019-01-05]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/tutorials/first-web-api?view=aspnetcore-2.2&tabs=visual-studio

[10] Smith, J. P.: How to write good, testable ASP.NET Core Web API code quickly. [online], [cit. 2019-01-05]. Dostupné z: https://www.thereformedprogrammer.net/how-to-write-good-testable-asp-net-core-web-api-code-quickly/

[11] Rutnik, A.: Bringing SEO to Angular Applications. [online], [cit. 2019-01-05]. Dostupné z: https://medium.com/slalom-engineering/bringing-seo-to-angular-applications-9de14995dc67

[12] Mack, J.: Five Advantages & Disadvantages Of MySQL. [online], [cit. 2019-01-05]. Dostupné z: https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/

[13] neznámý, A.: Comparison of the usage of Apache vs. Nginx vs. Microsoft-IIS for websites. [online], [cit. 2019-01-05]. Dostupné z: https://w3techs.com/technologies/comparison/ws-apache,ws-microsoftiis,ws-nginx

[14] neznámý, A.: Web server performance comparison. [online], [cit. 2019-01-05]. Dostupné z: https://help.dreamhost.com/hc/en-us/articles/215945987-Web-server-performance-comparison

[15] oddělení ICT FIT ČVUT: Úvod do Shibboleth SSO. [online], [cit. 2019-01-05]. Dostupné z: https://ict.fit.cvut.cz/~web/current/web/identity/shibboleth-sp/

[16] Cantor, S.: Shibboleth Concepts: Home. [online], [cit. 2019-01-05]. Dostupné z: https://wiki.shibboleth.net/confluence/display/CONCEPT/Home

[17] Hilton, J.: Secure your ASP.NET Core 2.0 API (part 1 - issuing a JWT). [online], [cit. 2019-01-05]. Dostupné z: https://jonhilton.net/2017/10/11/secure-your-asp.net-core-2.0-api-part-1---issuing-a-jwt/

[18] Nielsen, J.: 10 Usability Heuristics for User Interface Design. [online], [cit. 2019-01-07]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/

[19] Ing. Pavel Žikovský, P.: Návrh uživatelského rozhraní, 4. přednáška - Usability, Special testing, Personas. [online], [cit. 2019-01-07]. Dostupné z: https://gitlab.fit.cvut.cz/MI-NUR/mi-nur/blob/master/media/x04-Usability__Special_Testing__Personas.pdf

[20] Nielsen, J.: Why You Only Need to Test with 5 Users. [online], [cit. 2019-01-07]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

Příloha A

Seznam použitých zkratek

MVC Model-View-Controller

API Application Programming Interface

IIS Internet Information Service

DOM Document Object Model

SSR Server-Side Rendering

IDE Integrated Development Environment

ORM Object-Relational Mapper

SQL Structured Query Language

JSON JavaScript Object Notation

REST Representational State Transfer

SSO Single Sign-On

GUI Graphical user interface

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `dll` ..... adresář se zkompilovanou verzí
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF
- `test` ..... adresář se záznamy testování