Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Dashboard studenta

Miroslav Štaffa

Vedoucí práce: Ing. Aleš Fišer

11. ledna 2016

Poděkování

Chtěl bych poděkovat Ing. Aleši Fišerovi za vedení této práce, jeho rady a pomoc. Dále bych chtěl poděkovat Oddělení pro rozvoj, zejména Bc. Zdeňku Balákovi za pomoc a zpětnou vazbu k této práci. Také bych chtěl poděkovat rodině za podporu a pochopení při psaní mé práce.

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 11. ledna 2016 .....

Odkaz na tuto práci

Štaffa, Miroslav. *Dashboard studenta.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá analýzou uživatelských potřeb běžného studenta ČVUT. Dále se zabývá návrhem a implementací KPI (klíčových ukazatelů výkonnosti) určených pro studentský dashboard. Dashboard by měl být v budoucnu součástí nového informačního systému Fakulty informačních technologií. KPI budou implementovány jako portlety pro podnikový portál Liferay.

Klíčová slova dashboard, KPI, portlet, Liferay, portál, KOSapi

Abstract

This thesis deals with analysis of user needs of common student of CTU. Thesis also deals with design and implementation of KPI (key performance indicators) intended for a student dashboard. Dashboard should be future part of the new information system of Faculty of Information Technology. KPIs will be implemented as portlets for enterprise portal Liferay.

Keywords dashboard, KPI, portlet, Liferay, portal, KOSapi

Obsah

Seznam obrázků

Úvod

Na Fakultě informačních technologií ČVUT v současnosti vzniká nový informační systém, který by měl přinést dlouho očekávanou změnu ve stavu informačních systémů na fakultě. Nyní již běží v pilotním provozu, ve kterém sdružuje portál Spolupráce s průmyslem a portál pro správu závěrečných prací. Systém je vyvíjen oddělením rozvoje na FIT ve spolupráci s ICT oddělením FIT, VIC ČVUT a s Fakultou elektrotechnickou. Připravuje se nový vzhled tohoto informačního systému a zároveň jsou vyvíjeny nové součásti.

Jednou z těchto součástí by měl být i dashboard určený studentům. Dashboardy jsou v dnešní hektické době populární součástí informačních systémů moderních společností, protože umožňují uživateli se rychle a přehledně seznámit s důležitými daty. Hlavními prvky většiny dashboardů jsou tzv. klíčové ukazatele výkonnosti.

Tato práce v první fázi seznamuje čtenáře se zásadami tvorby kvalitního dashboardu. Popíše jednotlivé typy dashboardů, jejich přednosti a využití. Dále zkoumá dashboardy v použití v univerzitním prostředí v České republice i v zahraničí.

Dále se práce zabývá analýzou uživatelských potřeb studenta ČVUT. Výstupy této analýzy můžou posloužit k návrhu nových součástí informačního systému.

V návrhové a implementační části se práce zabývá právě tvorbou klíčových ukazatelů výkonnosti. Ty jsou implementovány jako portlety pro portál Liferay. Tím pádem je možné jejich využití v současném informačním systému fakulty. Nakonec jsou tyto komponenty otestovány a nasazeny na tento portál.

Tuto práci si autor vybral proto, že se mu líbila příležitost vytvořit něco pro rozvoj fakulty. Existuje pak šance, že vytvořené dílo bude opravdu sloužit k prospěchu studentů i školy.

Kapitola 1

Cíl práce

Tato práce si klade za své cíle zpracovat dva hlavní úkoly. Prvním z nich je analyzovat potřeby studenta naší fakulty s ohledem na jejich studium. Je nutné zjistit, jakým způsobem tyto potřeby studenti současně naplňují, a zda by nešlo jednotlivé nástroje zpřístupnit a sjednotit pod nově vznikající informační systém.

Druhým cílem bude navrhnout a implementovat komponenty, které by mohli být součástí studentského dashboardu. Mělo by se jednat převážně o komponenty zprotředkující klíčové ukazatele výkonnosti. Vzniklé komponenty budou následně otestovány, aby mohli být nasazeny do informačního systému. Nasazené komponenty by pak měli mít za následek větší motivaci studentů fakulty a jejich informovanost nutnou k dosažení důležitých studijních úspěchů.

Tato bakalářská práce tedy není čistě implementační, ale stojí někde na pomezí mezi implementační a návrhovou prací, protože se snaží nejen o implementaci součásti nového informačního systému, ale i o důkladnou analýzu potřeb studenta fakulty.

Kapitola 2

Rešerše existujících řešení

Tato kapitola se věnuje tomu, jakým způsobem lze přistupovat k problematice studentských dashboardů a dashboardů obecně. Dále je zaměřená na využívání dashboardů na jiných univerzitách.

2.1 Dashboard

Dashboard je nástroj, většinou používaný v korporátním prostředí, ke zlepšení a zvýšení efektivity zaměstnanců. Obsahuje sadu KPI (Klíčových ukazatelů výkonu) a další důležité informace. Jeho účelem je zprostředkovat dané informace uživateli a to co nejlépe a nejsrozumitelněji. Měl by šetřit čas uživatelům i organizaci v níž je nasazen. Dashboard by měl vždy být ušitý na míru konkrétní společnosti. Dále by měl zobrazovat danému uživateli pouze informace, které se ho bezprostředně týkají. Existuje několik zásad[1], které by měl správný dashboard splňovat, aby mohl fungovat tak, jak je od něj požadováno:

- Informace předává jasně a výstižně.
- Neobsahuje prvky, které by mohly uživatele rozptylovat, může to vést ke zmatení uživatele.
- Vhodně organizuje zobrazované informace, aby podpořil jejich význam a použitelnost.
- Využívá znalosti lidského vizuálního vnímání k co nejlepší prezentaci informací.
- Dobře vypadá.

Existuje několik typů dashboardů podle jejich použití. Jedná se o strategické, analytické a operační dashboardy.

2.1.1 Strategický dashboard

Strategický dashboard se používá k rychlému přehledu informací, které jsou důležité hlavně z dlouhodobého hlediska. To znamená, že není potřeba, aby data tímto dashboardem zprostředkovaná, byla tzv. real-time, neboli aktuální. Většinou se používá právě ve společnostech s rozvětvenou manažerskou hierarchií, kde s jejich pomocí manažeři hodnotí plnění dlouhodobých cílů této společnosti.[1] Tento typ tedy není vhodný pro naše účely.

2.1.2 Analytický dashboard

Analytický dashboard, jak už je z názvu patrné, se zaměřuje na analýzu dat. Takovýto dashboard zprostředkovává interakci s daty pro analytika, který s jeho pomocí může zjišťovat příčiny problémů nebo úspěchů. Může pak upravit chování společnosti takovým způsobem, aby se problémům předešlo a úspěchů bylo dosaženo.[1] Tato filozofie se již částečně blíží požadavkům na studentský dashboard, student by mohl pomocí analytického dashboardu a na základě vhodných dat hodnotit své výsledky.

2.1.3 Operační dashboard

Za operační dashboard se označuje takový typ dashboardu, který pracuje s aktuálními daty a monitoruje okamžitě každou změnu, která nastane. Měl by být praktický, mít jednoduché grafické rozhraní a neměl by obsahovat přehnané množství statistik a analytických prvků.[1] Takovýto dashboard vyhovuje našemu pojetí nejlépe, protože student potřebuje mít vždy aktuální data, ať už se jedná o výsledky zkoušek, testů, hodnocení prací nebo třeba různých změn v rozvrhu nebo harmonogramu.

2.2 Přístup k problematice

2.2.1 Ve firemním prostředí

Dashboardy jsou nejčastěji využívány právě v prostředí převážně větších, ale v některých případech i menších firem. Zavedení dashboardu ve společnosti může přinést mnoho výhod, pro každou společnost jsou ovšem důležitější jiné cíle. Mezi některé z možných přínosů může (dle [2]) patřit:

- Schopnost uživatelů získat a sjednotit data z jednoho či více zdrojů.
- Zorganizovat a prezentovat informace tak, aby mohli být ihned vstřebány uživatelem.
- Efektivně monitorovat business operace.
- Efektivně sdílet informace v rámci společnosti.

- Zvýšit kvalitu rozhodování o dalším směřování společnosti.

Je vidět, že účelem těchto metrik je zlepšení úspěchu společnosti, nikoliv jednotlivce. Implementace dashboardu může také přinést (dle [2]) mnohá úskalí, většinou způsobená špatnou analýzou před zavedením dashboardu ve společnosti:

- Nepřesná data - můžou podrývat důvěryhodnost dashboardu a tím smazat jeho přínosy.
- Nevhodně cílená data - vždy by měla být přínosem pro uživatele, kterým jsou zobrazována.
- Nevhodná vizualizace.

2.2.2 Ve školním prostředí

Během výzkumu této práce byla snaha najít existující řešení dashboardu na jiných univerzitách v České republice. Byly zkoumány informační systémy několika českých veřejných škol a hledáno již existující řešení dashboardu určeného pro studenty. Dále byly vyhledávány příklady ze zahraničí.

2.2.2.1 Studijní informační systém

Studijní informační systém[1] je informační systém Univerzity Karlovy. Při výzkumu byla možnost nahlédnout na některé jeho části a zkoumat, zda se v něm vyskytují nějaké prvky dashboardu pro studenty. Nejblíže k prvkům KPI mělo zobrazování výsledků zkoušek, které je však standardní součástí informačních systémů na univerzitách. Jiné části informačního systému, které by se dali považovat za dashboard se nepodařilo nalézt. Proto nebylo možné čerpat inspiraci při návrhu vlastních komponent pro dashboard.

2.2.2.2 Integrovaný studijní informační systém

Integrovaný studijní informační[2] systém je moderní informační systém v provozu na Vysoké škole ekonomické v Praze. Oproti IS UK poskytuje na první pohled daleko více funkcí. Nejdůležitější součástí pro studenty je tzv. Portál studenta. Tento portál obsahuje jak statistické údaje, tak i prvky, které bychom mohli za součást dashboardu považovat. Jedná se například o následující:

- Srovnávání úspěšnosti termínů zkoušek.
- Kontrolu studijního plánu (skupiny předmětů, kredity).

[1] `https://is.cuni.cz/studium/`
[2] `http://isis.vse.cz/`

- Docházku a hodnocení cvičení.

Kvůli nedostatečnému přístupu k tomuto systému se nepodařilo sehnat doplňující obrázky, ale v dokumentaci k systému je možný náhled na tyto prvky.[3] Z této dokumentace byly čerpány výše zmíněné informace.

2.2.2.3 Zahraniční univerzity

V rámci průzkumu bylo také navštíveno několik informačních systému zahraničních univerzit. Většina dashboardů však z pochopitelných důvodů vyžaduje autorizaci, takže se povedlo získat jen omezené informace o těchto dashboardech.

Například dashboard University of Sunderland[3] poskytuje kalendář z důležitými událostmi školy, poslední novinky, náhled na aktivity lektorů i studentů fakulty, ale také třeba i spojení hromadné dopravy ke škole.

Dále zkoumaný dashboard univerzity Nottingham Trent sice není veřejnosti přístupný, ale má volně dostupnou dokumentaci[4]. Z ní je patrné, že dashboard stojí především na studentské četnosti používání školních systémů, docházky do školy, využívání knihovny a odevzdávání úkolů a podle této četnosti následně přiděluje stupeň závazku studenta. Ten je pak v porovnání z ostatními studenty předložen studentovi, který na jeho základě může stanovit závěry o svém přístupu ke studiu a motivovat studenta ke zlepšení své studentské aktivity.

[3] `https://my.sunderland.ac.uk/dashboard.action`

[4] `https://www.ntu.ac.uk/current_students/document_uploads/179129.pdf`

Kapitola 3

Analýza

V analýze bude nejprve zhodnocen současný stav a možnosti pro každodenní potřeby studenta. Bude zaměřená na to, zda jsou v současné době řešeny případné nedostatky a pokud ano, tak jakým způsobem.

3.1 Současný stav

3.1.1 Kalendář

Současně používaný rozvrh hodin Timetable[5] není v mnoha ohledech vyhovující pro všechny studentské požadavky. Nejsou v něm ani zdaleka všechny školní aktivity studenta, chybí i předměty na jiných fakultách a to včetně předmětů na ÚTVS, které jsou povinné. Nepodporuje výjimky ve výuce, jednorázové akce, zkoušky. Jedním z velkých přínosů pro studenta by tedy byl kalendář, který by slučoval jak rozvrh výuky, tak i termíny zkoušek, jednorázové akce a další školní aktivity. Tento kalendář by také měl umět exportovat data v jednom ze standardních formátů (iCal, XML) kvůli možnému použití na dalších platformách. Jeden projekt, který by mohl situaci řešit, je již v rámci fakultního rozvoje vyvíjen pod názvem Sirius.

3.1.1.1 Sirius

Co tedy přinese projekt Sirius a co by v něm mohlo chybět? Tento projekt by měl (dle [4]) přinést následující:

- Rozvrh hodin, který bude odpovídat skutečnému průběhu semestru
- Podporu zobrazení předmětů z jiných fakult včetně tělocviku na ÚTVS
- Podporu exportu kalendáře pomocí standardního formátu iCal

[5] `timetable.fit.cvut.cz/`

- Plánuje se i navazující komponenta pro přesun studentů mezi paralelkami v průběhu semestru

Projekt by tedy měl vyřešit mnoho výše zmíněných problémů. Za zlepšení by se dala považovat integrace zobrazení zkouškových termínů.

3.1.2 Ukazatele výkonnosti

Mezi hlavní nedostatky současného stavu patří minimální možnost měření a srovnávání výkonu studenta. Toho je možné dosáhnout použitím takzvaných Klíčových ukazatelů výkonosti neboli KPI (z anglického Key Performance Indicators). KPI má zajistit seznámení uživatele, v našem případě studenta, s jeho výkonem v rámci studia, porovnáním s jinými studenty či srovnávacími nároky (např. kreditní limity pro úspěšné ukončení ročníku, bodové limity v rámci předmětu atd.). V současné době je možné porovnávat výkon studenta s ostatními třeba v rámci systému Progtest, kde se objevuje v kontextu srovnávání bodů získaných z jednotlivých úloh. Návrh a implementace konkrétních komponent obsahující ukazatele výkonnosti bude hlavní složka návrhové a implementační části mé práce.

3.1.3 Novinky

Další nutností pro studenta je být seznámen s aktuálními událostmi týkající se školy a fakulty. V současné době jsou tyto informace dostupné jak z oficiálních webových stránek FIT, tak i ČVUT. Stránky FIT podporují syndikaci novinek pomocí technologie RSS. Do budoucna by studenti mohli ocenit tagování článků metadaty a následná personalizace obsahu novinkových feedů studenta. Tomuto by se mohla věnovat některá z dalších závěrečných prací, je zde mnoho prostoru pro kreativní řešení.

3.1.4 Chat

Pro většinu studentů je důležitá komunikace s ostatními studenty případně i lektory. Komunikace mezi studenty je v současné době zprostředkována hlavně pomocí sociálních sítí, především Facebooku, kde vznikla spousta fakultních skupin, kde studenti mohou probírat různá témata, od domácích úloh až po domlouvání se na společném studiu. Komunikace s lektory zůstává výsadou hlavně elektronické pošty. Ovšem oba tyto současně nejvyužívanější způsoby komunikace jsou přece jenom poněkud neinteraktivní, minimálně z toho hlediska, že diskutující člověk může na reakci čekat opravdu dlouho, případně se reakce vůbec nedočkat. Tento problém se snaží řešit další část nového informačního systému a to je LIMS Messenger for Liferay.

3.1.4.1 LIMS

LIMS je instant messenging program pro portál Liferay. Verze LIMS, běžící na fakultě, umožňuje uživateli fakultního informačního systému komunikovat se všemi ostatními uživateli. Dále je možnost vytvořit konverzaci s více uživateli najednou, čehož by se dalo využívat například v komunikaci lektorů s jednotlivými paralelkami, případně diskuse studentů nad studijními záležitostmi v rámci jedné paralelky. LIMS však podle autora[5] obsahuje i další vymoženosti, v budoucnu se tak můžeme dočkat některé z dalších funkcí jako například slučování kontaktů do skupin.

3.1.5 Studijní materiály

Další potřebou studenta je přístup ke studijním materiálům. Většina jich je dnes v systému Edux, někdy se však objevují i na externích webových stránkách (například předmět BI-SI2). V rámci diplomové práce Daniela Heglase vznikla nová součást IS fakulty, kterým je portlet zabývající se přístupem ke studijním materiálům. Ty by v budoucnu měli být uloženy v systému pro správu dat Alfresco a měli by být řádně označeny a kategorizovány. Tento portlet pak využívá API Alfresca a umožňuje přístup k těmto uloženým studijním materiálům. Aplikace ještě není zcela dokončená a je tedy spíše takovým základem budoucího řešení.[6] Další vývoj této aplikace by mohl být zajímavým tématem pro závěrečnou práci.

3.2 Analýza požadavků

Dále se již tato práce bude věnovat převážně analýze, návrhu a implementaci komponent, které budou v jejím rámci vytvořeny. V této části budou vypsány požadavky na implementované součásti dashboardu.

3.2.1 Funkční požadavky

1. Student bude moci kontrolovat a srovnávat s ostatními svůj vážený průměr.

2. Student bude mít možnost kontrolovat a srovnávat s ostatními studijní výsledky z jednotlivých zapsaných předmětů.

3. Studentovi bude umožněno kontrolovat stav své závěrečné práce a bude mít možnost ji srovnat z rozložením stavů ostatních závěrečných prací.

4. Student bude moci kontrolovat svůj průchod studiem pomocí vizualizace počtu kreditů a limitů na pokračování ve studiu.

5. Student bude mít náhled na výkon v probíhajícím zkouškovém období.

3.2.2 Nefunkční požadavky

1. Uživatel bude moci k portletům přistupovat v rámci nového Informačního systému fakulty.
2. Portlety budou vytvořeny pro podnikový portál Liferay ve verzi 6.1.2 v jazyce Java.
3. Pro vytvoření frontendu bude použit JSP.
4. Pro vyhodnocování závislostí a build projektu bude využíván Apache Maven.
5. Aplikace bude podporovat běžně používané webové prohlížeče.

Kapitola 4

Návrh

Všechny komponenty, které budou v rámci této práce implementovány, budou představovat KPI nebo bude KPI jejich součástí. Vychází se přitom ze zadání, kde je specifikováno, že je nutné se zaměřit na každodenní potřeby studenta a KPI. Jelikož součástí KPI jsou zahrnuty i každodenní potřeby studenta, ať už se jedná o sledování prospěchu, kontrola termínů zkoušek, dojde v tomto ohledu ke splnění zadání. U jednotlivých komponent dojde k návrhu chování samotných komponent, uživatelského rozhraní a použitých datových zdrojů. Na konci kapitoly budou představeny technologie použité na implementaci těchto komponent.

4.1 Návrh komponent

Tato sekce je zaměřená na to, co bude od jednotlivých komponent požadováno.

4.1.1 KPI - Porovnání průměru

Toto KPI by mělo studentovi poskytnout náhled na aktuální stav jeho váženého průměru a umožnit ho porovnat s průměry ostatních studentů fakulty. V současnosti je sice student schopen sledovat svůj průměr i v rámci informačního systému KOS, ale jelikož je studijní průměr jedna z hlavních metrik výkonu studenta při studiu, tak se domnívám, že je to jeden z nejdůležitějších ukazatelů. Porovnání s ostatními studenty bude poté studenta motivovat ke zlepšování svých známek a tím pádem i svého váženého průměru.

4.1.2 KPI - Porovnání výkonu v předmětu

Účelem tohoto KPI bude srovnání bodového hodnocení studenta v jednotlivých předmětech zapsaných během semestru. Podobně jako v případě porovnávání průměru by student měl být více motivován k lepším výkonům, nyní však v konkrétních předmětech. Toto KPI navíc sloučí stav bodů v předmě-

tech na jedno místo. V současné době však použití skutečných dat při provozu tohoto KPI není příliš pravděpodobné, více se o tomto problému bude pojednávat v části o zdrojích dat.

4.1.3 KPI - Závěrečné práce

Dalším KPI bude komponenta zobrazující stav závěrečných prací všech studentů posledního ročníku příslušného studijního plánu. To by opět mělo studenta motivovat k včasnému zvolení tématu bakalářské a diplomové práce, případně i k včasnému odevzdání dané práce. Mohlo by to však býti i rizikem, pokud by větší množství studentů oddalovalo jednotlivé kroky k odevzdání práce, vedlo by to k posílení pocitu dostatku času u ostatních studentů. Tím pádem by mohlo dojít k zahlcení jednotlivých aktérů v procesu schvalování závěrečných prací v době před koncem termínu. Nicméně tento stav je v současné době obvyklý, takže by s větší pravděpodobností došlo spíše k vylepšení současného stavu.

4.1.4 KPI - Počet kreditů

Toto KPI umožní studentovi zjistit počet kreditů, které nasbíral během studia, a počet kreditů v současném semestru. Mělo by také ukazovat požadované limity k pokračování ve studiu. Student pak může posoudit, zda mu vychází kredity pro dokončení studijního plánu, případně kredity pro postup do dalšího ročníku nebo semestru.

4.1.5 KPI - Aktuální semestr

Poslední KPI, které bude v rámci této práce vznikat, je zaměřeno na zobrazení výkonu v aktuálním semestru. Student bude mít možnost porovnat vážený průměr v současném semestru s ostatními studenty, bude mít možnost zobrazit si zbývající termíny předmětu a také by mu toto KPI mělo napovídat, jaké známky by měl obdržet, aby si udržel nebo zlepšil průměr na určitou hodnotu. To by mělo studenta motivovat k lepší práci a výsledkům hlavně během zkouškového období. Dále by měla komponenta vhodnou vizualizací umožnit studentovi srovnat svůj průměr s nároky na prospěchové stipendium a tím pádem ho i motivovat ke zlepšení studijních výsledků.

4.1.6 Neimplementované KPI

Následující KPI jsou pouze pro zamyšlení a jako návrh pro budoucí vývoj. V této práci již nebudou tyto KPI implementovány, ať už z časových důvodů nebo z důvodu současně nedostupných datových zdrojů nutných pro funkčnost těchto komponent.

4.1.6.1 KPI - Vyhodnocení pravděpodobnosti pokračování ve studiu na základě dosavadních výsledků

KPI, které by dokázalo vyhodnotit pravděpodobnost studenta pokračovat ve studiu na základě jeho dosavadních výsledků, by mohla být z pohledu studentů velmi zajímavá. Bylo by však nutné vymyslet, jakým způsobem by se z dosavadních výsledků pravděpodobnost získávala. Bylo by možné zohlednit počet kreditů, známky, počet pokusů u zkoušky, včasné odevzdávání úloh i mnohé další aspekty. Na to však v současnosti není připravená infrastruktura dat. V budoucnu by ale mohlo jít o zajímavý projekt.

4.1.6.2 KPI - Vyhodnocení pravděpodobnosti úspěšného složení předmětu

KPI podobné předchozímu, avšak úzce zaměřené na předmět. Opět by bylo nutné vymyslet způsob, jakým zhodnotit studentovu pravděpodobnost na složení předmětu. Tentokrát by měla být sledována kritéria jako chybějící body do zápočtu a včasné odevzdávání úloh. Musely by se také zvážit podmínky a pravidla jednotlivých předmětů, jako třeba možnost opravného testu, počet jednotlivých úloh a testů. Tato data by musela být zanesena v nějakém jednotném systému, který by předměty spravoval, což je opět důvod, proč takovéto KPI v současné době nepřichází do úvahy. Nicméně jde opět o možnost zamyslet se nad podobným měřením a zda by to vůbec mělo přínos, který by vyvážil investovaný čas, nutný pro zprovoznění podobného KPI.

4.2 Zdroje dat

Vznikající komponenty mají poměrně rozsáhlé nároky na datové zdroje, které budou využívat. U jednotlivých komponent bude napsáno, která data budou zapotřebí a jakým způsobem jsou v současné době dostupná. Pokud data nejsou nyní v rozumné formě dostupná, bude nutné specifikovat požadavky a v rámci testování vzniknou data simulovaná.

4.2.1 KPI - Porovnání průměru

Pro výpočet váženého průměru studenta bude používán datový sklad, který v současnosti využívá i Portál pro spolupráci s průmyslem k výpočtu garantovaného hodnocení studenta. Tento datový sklad není v žádném případě živý, tzn. data nejsou aktuální v reálném čase. Datový sklad tedy poslouží spíše jako proof of concept a v budoucnosti by bylo výhodné mít datový zdroj s živými daty. Ten by se dal navíc využít na další potřeby rozvoje fakulty, nejen jako zdroj pro toto KPI.

4.2.2 KPI - Porovnání výkonu v předmětu

U tohoto KPI bude největší slabina právě v nedostatečných zdrojích dat. V současné době jsou výsledky studentů v předmětech rozmístěny přes několik různých systémů, vždy záleží na konkrétním předmětu. A tak jsou výsledky studenta v některých předmětech v systému Edux, v dalších v systému Progtest nebo Moodle, někdy výsledky nejsou v datové podobě dostupné vůbec. Proto zde bude nutné specifikovat požadavky na data a na simulovaných umělých datech otestovat vzniklé KPI. Co se týče sjednocení dat z jednotlivých předmětů, tak by v budoucnu fakultě velmi prospělo mít na všechny předměty jeden systém, kam by bylo hodnocení zaznamenáváno. Mohlo by to být vhodné téma pro některou z dalších závěrečných prací.

4.2.3 KPI - Závěrečné práce

V rámci nového informačního systému jsou dostupné i data týkající se závěrečných prací. Tato komponenta vyžaduje specifické údaje o závěrečných pracích, hlavně tedy rozložení jednotlivých studentů napříč fázemi závěrečné práce. V rámci práce budou definované také požadavky na tyto data a tak bude možné datový zdroj napojit do tohoto KPI.

4.2.4 KPI - Počet kreditů

Této komponentě bude v plném rozsahu postačovat rozhraní projektu KOSapi.

4.2.4.1 KOSapi

KOSapi je školní projekt, který poskytuje rozhraní napojené na KOS databázi a umožňuje tím vývoj aplikací, které potřebují přístup k datům souvisejícím s výukou.[7] Data jsou poskytovány pomocí RESTových služeb v ATOM formátu.

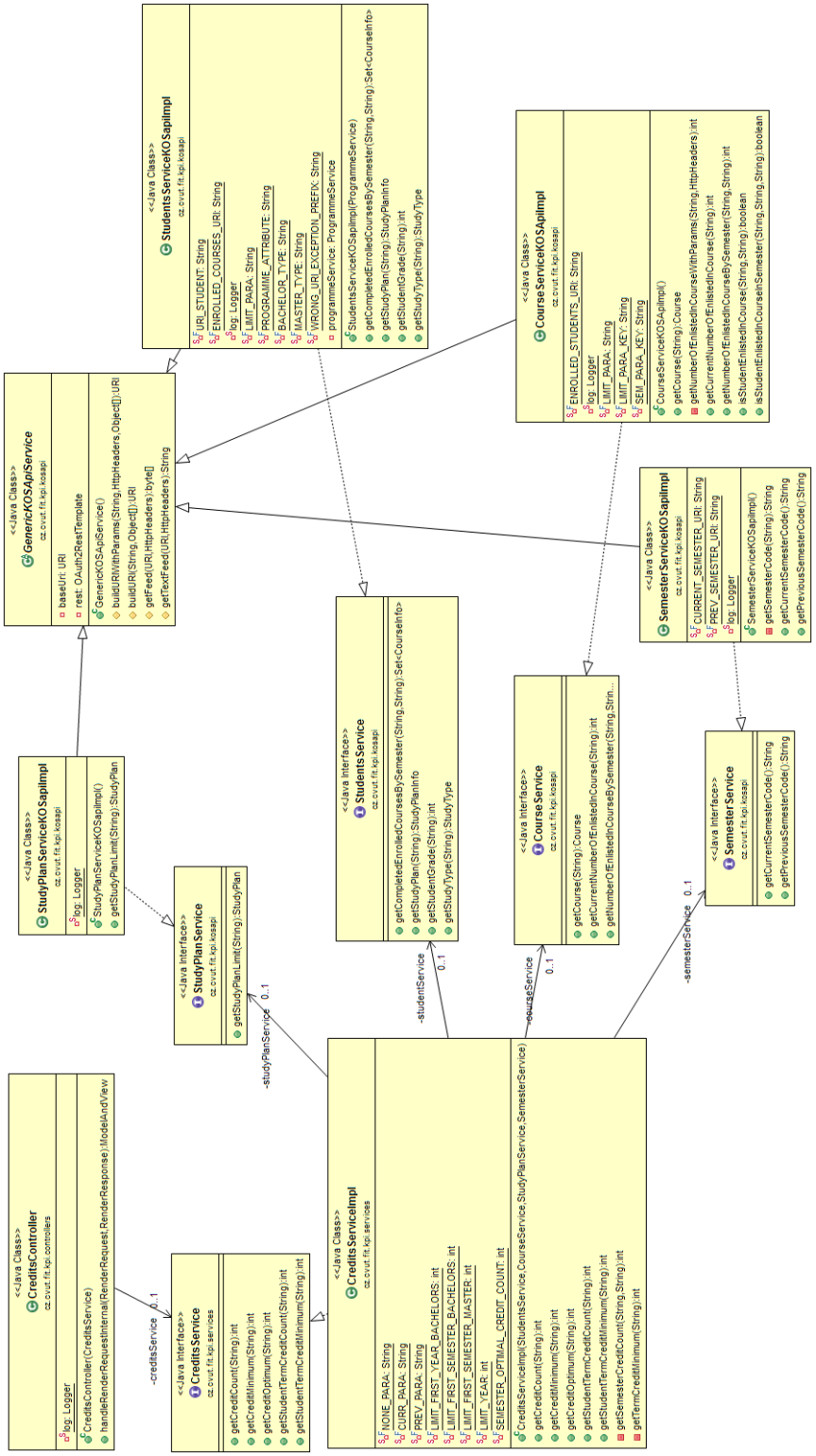
Z projektu KOSapi budeme využívat převážně zdroje pro zjištění zapsaných předmětů studenta a informace o předmětech.

4.2.5 KPI -Aktuální semestr

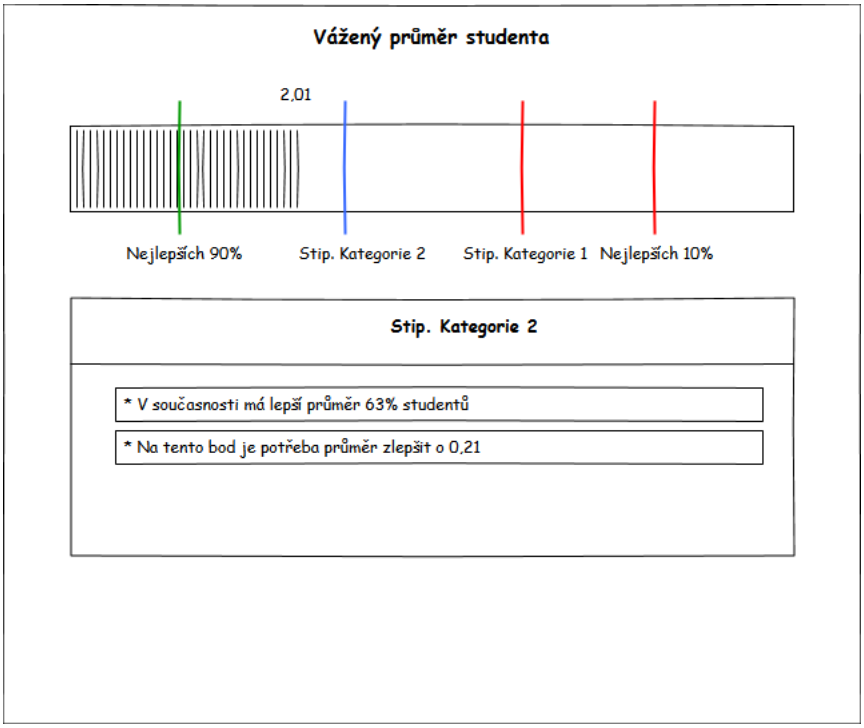
Tato komponenta bude využívat jak data z datového skladu jako u KPI Porovnání průměru, tak i data z rozhraní KOSapi. KOSapi bude nutné pro získání termínů zkoušek, které by se měly studentovi zobrazovat.

4.3 Návrh tříd

Jelikož většina komponent je ze strany backendu poměrně triviální, nebudou zde ukázány všechny třídní diagramy, ale bude vybrán některý ze zajímavějších.



Obrázek 4.1: Počet kreditů - Class diagram



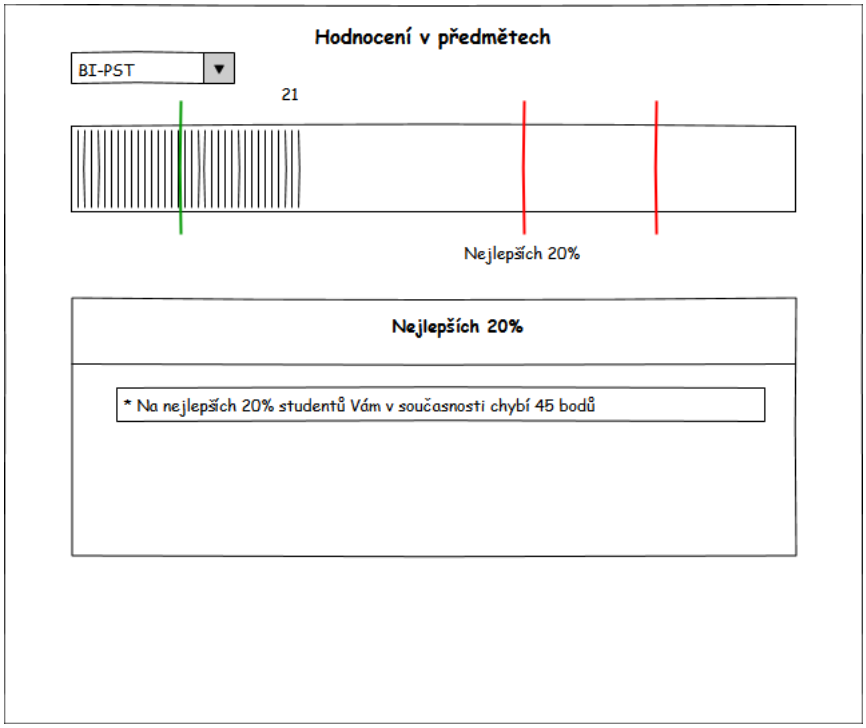
Obrázek 4.2: Porovnání průměru - Wireframe

4.4 Návrh uživatelského rozhraní

Jelikož komponenty zobrazované v dashboardu musí zobrazovat data studentům co nejsrozumitelněji a nejpřehledněji, tak je návrh uživatelského rozhraní velmi důležitou součástí návrhové složky práce.

4.4.1 KPI - Porovnání průměru

Hlavní složka GUI tohoto KPI (4.2) bude položený sloupcový graf, sledující průměr studenta v rozmezí od 4 do 1. Dále bude v grafu umístěno několik zaklikávacích milníků, které po rozkliknutí zobrazí informace ke splnění daného milníku. Příkladem může být milník 10% nejlepších studentů. Po jeho zakliknutí by se student měl dozvědět, o kolik by bylo třeba zlepšit průměr na dosažení daného milníku. Dále budou milníky barevně rozlišené podle toho, zda na ně student dosáhl či nikoliv.

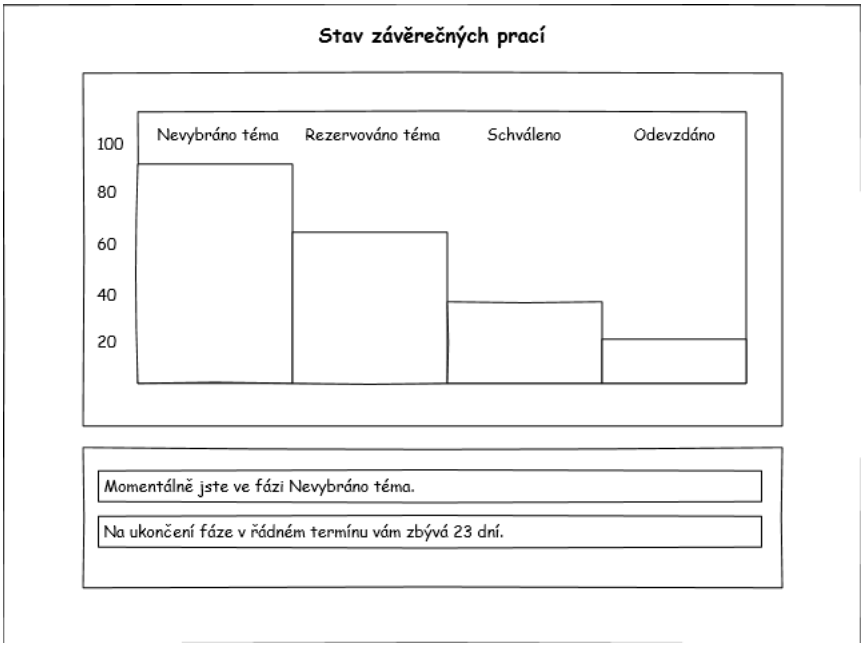


Obrázek 4.3: Porovnání výkonu v předmětu - Wireframe

4.4.2 KPI - Porovnání výkonu v předmětu

V tomto KPI (4.3) bude hlavní složkou histogram z rozložením studentů mezi intervaly bodů z daného předmětu. Bude označeno, ve kterém se nachází student a kolik má bodů. Po zakliknutí jednotlivých milníků známek, bude zobrazeno kolik chybí studentovi bodů na případné zlepšení nejlepší dosažitelné známky.

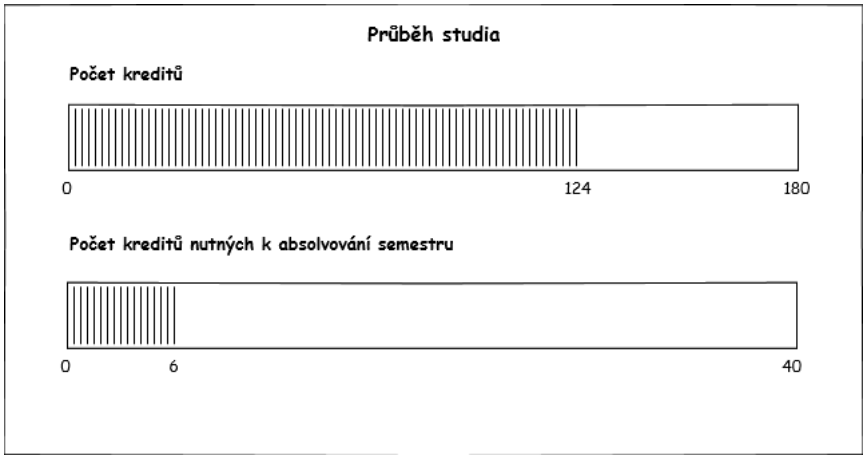
Combobox bude umožňovat překlikávat mezi jednotlivými předměty.



Obrázek 4.4: Závěrečné práce - Wireframe

4.4.3 KPI - Závěrečné práce

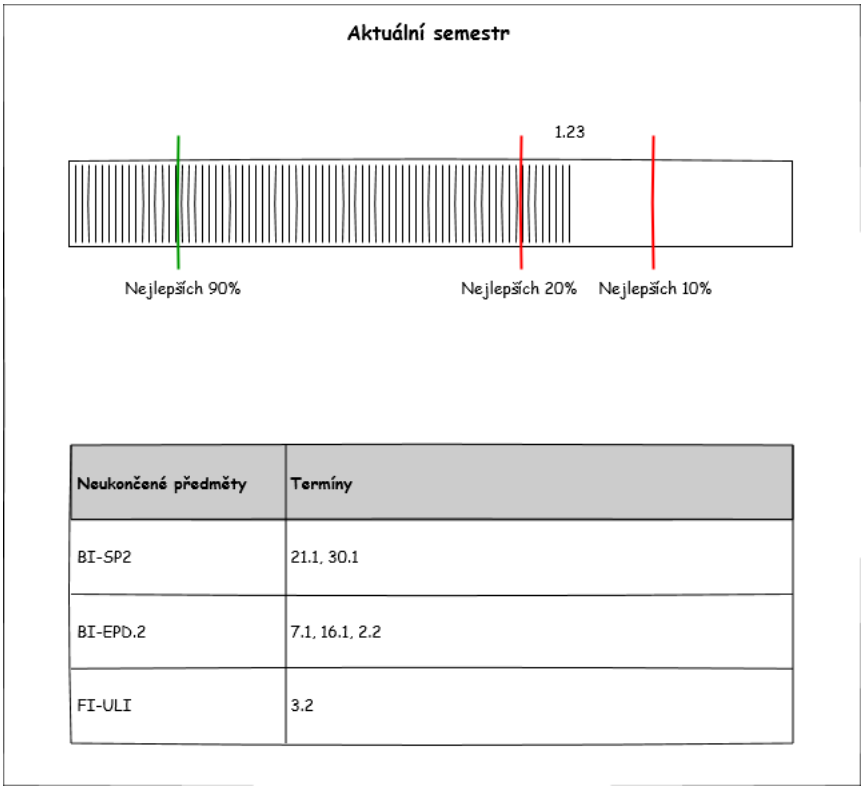
Data tohoto portletu (4.4) budou vizualizována pomocí histogramu. Ten bude mít za úkol zobrazit rozložení studentů mezi jednotlivými fázemi závěrečné práce. Pod histogramem by se měla vyskytovat informace o tom, v jakém stavu se momentálně nachází student.



Obrázek 4.5: Počet kreditů - Wireframe

4.4.4 KPI - Počet kreditů

Tento portlet (4.5) by měl ve dvou progress barech vizualizovat jak postup v absolvování studia pomocí poměru získaných kreditů k celkovému počtu, tak i postup v probíhajícím semestru.



Obrázek 4.6: Aktuální semestr - Wireframe

4.4.5 KPI - Aktuální semestr

Toto KPI (4.6) by mělo zobrazit srovnání studentova průměru z daného zkouškového období s ostatními studenty. Dále by mělo zobrazit tabulku, která bude vizualizovat počet zbývajících termínu u předmětů zapsaných v tomto semestru.

4.5 Použité technologie

Jelikož jsou implementovány komponenty pro již existující řešení, je tato práce značně omezena ve výběru technologií, které budou použity. V následující části budou tyto technologie popsány, se zaměřením na to, z jakého důvodu je nutné použití právě těchto technologií a jaké jsou jejich přednosti a slabiny.

4.5.1 Liferay Portal

Liferay Portal je jedním ze zástupců podnikových portálů. Je vydaný jak pod komerční licencí, tak i pod GNU LGPL[8]. Podnikové portály jsou platformy, které poskytují organizaci vyvinout vlastní specifickou implementaci portálu. Jejich prostřednictvím pak nabízejí svým uživatelům přístup k relevantním datům, procesům a aplikacím.[9]

Nedílnou součástí portálů jsou tzv. portlety. Portlet je webová komponenta, založená na technologii Java, která má za úkol vytvářet obsah pro portál. Tento obsah se nazývá fragment. Uživatel komunikuje s portletem modelem žádost-odezva. Obsah generovaný portletem může být odlišný v závislosti na uživateli nebo uživatelském nastavení. Portlety jsou spouštěné v tzv. portletovém kontejneru.[10] V technologii Liferay je již plně integrovaný portletový kontejner.[11] Liferay již také obsahuje velké množství předinstalovaných portletů. Jako příklad si můžeme uvést portlety pro navigaci, instant messaging, výstražné zprávy, ale celkově jich je v balíku přes 60.[12]

Nový fakultní informační systém v dnešní době běží na Liferay 6.1.2, proto bude nutné vyvíjet komponenty právě pro tuto technologii.

4.5.2 Java

Java je všeobecně použitelný, objektově orientovaný programovací jazyk. Je vysokoúrovňový[13], zdrojový kód je překládán do tzv. bajtkódu. Bajtkód je kód generovaný javovským kompilátorem. Při překládání a spouštění javovského programu nejprve dojde k vygenerování bajtkódu, ten je následně interpretován virtuálním strojem JVM (Java Virtual Machine).[14] Java Virtual Machine označuje jak specifikaci popisující architekturu virtuálního počítače, tak i vlastní program provádějící implementaci bajtkódu.[15]

Pro vytváření portletů je nejčastěji používána právě Java, proto i naše implementace bude používat technologii Java. Můžeme tak využít Spring a jeho Portlet MVC Framework.

4.5.3 Spring Framework

Spring Framework je platforma pro Javu, která zjednodušuje a poskytuje služby pro vývoj Java EE aplikací. Spring se skládá z asi 20 modulů, které se seskupují do několika skupin. Jedná se o Core Container, Data Access/Inte-

gration, Web, AOP (Aspektově Orientované Programování), Instrumentation, Messaging a Test.

Mezi nejdůležitější služby Spring frameworku patří Vkládání Závislostí (Dependency Injection, dříve také pod pojmem Inversion of Control). Tento koncept umožňuje správu závislostí mezi komponentami aplikace.[16] Komponenty aplikace využívající Dependency Injection pak nemusí vytvářet instance konkrétní implementace jiné komponenty, na které je závislá, ale závislost je vyřešena frameworkem, který dodá potřebnou instanci. Všechno je nastavitelné pomocí konfiguračních souborů, je tedy možné jednotlivé implementace komponenty měnit bez zásahu do kódu.[17] Taková aplikace je pak robustnější a modulárnější než aplikace, která nevyužívá tento koncept. Pro vývoj portletů budeme kromě Dependency Injection používat modul zaměřený na vývoj portletů s názvem Portlet MVC Framework.

4.5.3.1 Portlet MVC Framework

Portlet MVC Framework modul je úzce spjatý s Web MVC framework modulem. MVC v názvu značí, že pomocí frameworku bude aplikace rozdělená jednoznačně na tři části, na model (model), view (pohled) a controller (řadič).

Controller v Portlet MVC je pouze jednoduchý interface, který předepisuje dvě metody handleActionRequest a handleRenderRequest. Portlet workflow má tři fáze a to fázi akční, renderovací a resource fázi. K akční fázi dochází jenom jednou a dochází při ní ke změnám na backendu. Fáze renderovací naopak může probíhat vícekrát a jejím účelem je zobrazení obsahu uživateli. Resource fáze umožňuje portletu generovat zdroje.[18] Spring framework, na rozdíl od jiných frameworků, jednoznačně odlišuje akční a renderovací fázi a tím nám umožňuje kontrolovat obě fáze odděleně ve výše zmíněném controlleru.[16]

Java Server Pages Pro zobrazování obsahu je nutné použít jednu z renderovacích technologií. Bude použita Java Server Pages neboli JSP. JSP je technologie pro generování dynamického webového obsahu. Tato technologie nám umožňuje do HTML kódu vkládat kód Javy. Pomocí tzv. JSP stránky je definován způsob, jak bude obsah generován. Je možné generovat klasický HTML dokument, ale JSP podporuje také XML i další značkovací jazyky. Dále umožňuje použití CSS a Javascriptu.[19] Koncept portletu v rámci Spring Frameworku tedy funguje tak, že renderovací fáze naplní datový model, který je předán rendereru. Ten za použití definované JSP stránky a předaného modelu vytvoří dokument, který je prezentován uživateli.

4.5.4 Highcharts

Pro vykreslování grafů v portletech budeme používat javascript knihovnu Highcharts[6]. Ta umožňuje vizualizovat data za pomocí předdefinovaných grafů.

[6] `http://www.highcharts.com/`

Kapitola 5

Realizace

Tato kapitola je zaměřená na popis implementace komponent. Dále jsou nastíněny některé zajímavé problémy, které museli být při realizaci řešeny. Nakonec je srovnáno výsledné řešení s původním návrhem.

5.1 Popis balíčků projektu

V této sekci je popsáno rozčlenění projektu do jednotlivých balíčků.

- **cz.cvut.fit.kpi.controllers** - Obsahuje controllery pro jednotlivé portlety, které obsluhují prezentační vrstvu.
- **cz.cvut.fit.kpi.dao** - Obsahuje Data Access Objecty poskytující rozhraní pro přístup do databáze.
- **cz.cvut.fit.kpi.exceptions** - Obsahuje výjimky definované pro aplikaci.
- **cz.cvut.fit.kpi.kosapi** - Obsahuje rozhraní a jejich implementace pro komunikaci s KOSapi.
- **cz.cvut.fit.kpi.mappers** - Obsahuje mappery pro obsluhování dat z databáze.
- **cz.cvut.fit.kpi.model** - Obsahuje třídy reprezentující datový model aplikace.
- **cz.cvut.fit.kpi.services** - Obsahuje rozhraní a jejich implementace poskytující služby pro controllery.
- **cz.cvut.fit.kpi.utils** - Obsahuje tzv. utility třídy, například pro parsování Atom formátu nebo zaokrouhlování.

5.2 Architektura

Jak již bylo nastíněno v návrhu, aplikace vznikala za použití architektury MVC neboli Model-View-Controller. Tato architektura předepisuje tři základní aplikační vrstvy.[20] Na vrstvu datovou (Model), která je v případě této aplikace řešena prostřednictvím model tříd, které jsou vytvářeny a plněny daty z tříd obsluhujících databázi a KOSapi. Dále na pohledovou vrstvu (View), která je definována prostřednictvím JSP stránek. A nakonec na vrstvu řadičovou (Controller), která zajišťuje plnění pohledové vrstvy daty z vrstvy datové. Takto je zajištěno oddělení jednotlivých vrstev a tím pádem i snadněji udržovatelného kódu aplikace.

5.3 Cachování

Jelikož aplikace používá spoustu volání KOSapi pro získání vhodných dat, bylo přistoupeno k použití cachování. Cachování umožňilo rychlejší vracení návratové hodnoty u metod, které se používají poměrně často a u kterých se data často nemění. To je například při zjišťování počtu kreditů za splnění předmětu.

Byla použita cache Ehcache[7]. Nejprve bylo nutné nakonfigurovat Ehcache ve Spring konfiguraci.

```
<bean id="cacheManager"
    class="org.springframework.cache.ehcache.EhCacheCacheManager"
p:cache-manager-ref="ehcache" />
<bean id="ehcache"
class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean"
p:config-location="classpath:ehcache.xml" />
```

Poté je v souboru ehcache.xml nutné nastavit samotnou cache. Je možné konfigurovat dobu expirace cache, maximální počet prvků v paměti, zda se mají nadbytečné prvky zapisovat na disk, nebo zda se má cache ukládat.

```
<cache name="getCourse" maxElementsInMemory="500" eternal="false"
timeToIdleSeconds="3600" timeToLiveSeconds="3600"
    overflowToDisk="false"
diskPersistent="false" memoryStoreEvictionPolicy="LRU" />
```

Nakonec je potřeba u metody použít anotaci Cacheable s názvem cache nastavené z konfigurace. Takto je možné nakonfigurovat více cache pro více metod.

[7] http://www.ehcache.org/

5.4 Autorizace - OAuth 2.0

Pro přístup ke zdrojům KOSapi bylo nutné implementovat klientské části autorizace proti systému Zuul OAAS, což je fakultní autorizační server implementující protokol OAuth 2.0.[21]

Nejprve bylo nutné aplikaci zaregistrovat v aplikaci AppsManager[8]. AppsManagerem byl vygenerován Client ID a Client Secret, který je použit k autorizaci.

Řešení bylo inspirováno návodem[21]. Framework Spring značně zjednodušuje postup k úspěšnému zprovoznění klientské části autorizace, prakticky stačí vhodná konfigurace v souboru *webapp/WEB-INF/applicationContext.xml*

```
<oauth:resource id="rest"
type="${oauth.client_type}"
client-id="${oauth.client_id}"
client-secret="${oauth.client_secret}"
access-token-uri="${oauth.access_token_uri}"
scope="${oauth.scope}"
authentication-scheme="query"
token-name="${oauth.token_name}"/>

<oauth:rest-template id="oauthRestTemplate"
   resource="rest" />
```

Spring Framework se již postará o získání tokenu a používání získaného tokenu při posílání requestu na KOSapi. Jako schéma autentikace bylo použito query, takže dochází vkládání tokenu přímo do URL ke zdroji.

5.5 Parsování Atom Syndication Formátu

KOSapi poskytuje data ve formátu Atom, proto bylo nutné vymyslet, jakým způsobem bude formát parsován. Nakonec byl použit parser projektu Apache Abdera[9]. Využívané zdroje jsou poskytovány buď jako feed nebo jako samostatný entry, a to v závislosti na vyžádaném zdroji. Pro parsování získaného zdroje byla implementována třída *AtomParser*, která obsahuje metody *parseEntry* a *parseFeed*.

[8] https://auth.fit.cvut.cz/manager/index.jsf
[9] https://abdera.apache.org/

```
@Service
public final class AtomParser {

private AtomParser() {
}

   public static Map<String, Element> parseEntry(byte[] feedIs) {
      InputStream stream = new ByteArrayInputStream(feedIs);
      Parser parser = new Abdera().getParser();

      Document<Entry> doc = parser.parse(stream);
      Entry entry = doc.getRoot();

      Map<String, Element> contentMap = new HashMap<String,
          Element>();

      for(Element element: entry.getContentElement().getElements()) {
         contentMap.put(element.getQName().getLocalPart(), element);
      }

      return contentMap;
   }

   public static List<Map<String, Element>> parseFeed(byte[] feedIs) {
      InputStream stream = new ByteArrayInputStream(feedIs);
      Parser parser = new Abdera().getParser();

      Document<Feed> doc = parser.parse(stream);
      Feed feed = doc.getRoot();

      ArrayList<Map<String, Element>> mapList = new
          ArrayList<Map<String, Element>>();

      for(Entry entry: feed.getEntries()) {
         Map<String, Element> contentMap = new HashMap<String,
             Element>();
         for(Element element: entry.getContentElement().getElements())
             {
            contentMap.put(element.getQName().getLocalPart(), element);
         }
         mapList.add(contentMap);
      }

      return mapList;
   }
}
```

5.6 Výsledné komponenty

Oproti původnímu návrhu bylo nutné některé věci pozměnit a upravit zejména v oblasti vzhledu. Zde budou zdokumentovány provedené změny na jednotlivých komponentách. Výsledné snímky aplikace najdete v příloze Obrázky aplikace

5.6.1 Dashboard

Dashboard s KPI byl implementován pomocí dlaždic, kdy jednotlivé dlaždice obsahují jednotlivá KPI. Jednotlivé KPI je možné podmínečně zobrazovat a schovávat, například Závěrečné práce se studentovi zobrazí jenom v případě, že v současné době pracuje na své Závěrečné práci. Náhled na dashboard je k vidění na obrázku B.1.

5.6.2 KPI - Porovnání průměru

Tato komponenta přibližně odpovídá navrženému stavu. Došlo pouze k přidání milníku pro absolvování s vyznamenáním. Dále byly rozděleny milníky do kategorií, mezi kterými může student přepínat. Výsledný vzhled je na obrázku B.2.

5.6.3 KPI - Porovnání výkonu v předmětu

Toto KPI se prakticky shoduje s návrhem. V současnosti stojí pouze na simulovaných datech, takže pokud by mělo dojít k jeho nasazení, bude nutné vybudovat datový zdroj, který by poskytoval data z jednotlivých předmětů.

V oblasti vzhledu je většina věcí v souladu s návrhem. Výsledek je opět k nahlédnutí na obrázku B.3.

5.6.4 KPI - Závěrečné práce

KPI pro závěrečné práce je v současné době také testováno pouze na simulovaných datech, nicméně datové zdroje obsahující data pro jeho funkčnost již existují, takže by nemělo být problémem KPI v nejbližší době nasadit. Požadavky na data byly v rámci této práce vytvořeny, a tak je pouze otázkou časových možností týmu rozvoje fakulty, než budou patřičně napojeny.

Dále byly přidány informace ke studentově závěrečné práci, jako jsou jméno jeho práce, jméno vedoucího a oponenta. Výsledný vzhled komponenty je na obrázku B.4.

5.6.5 KPI - Počet kreditů

U této komponenty byla přidána hranice symbolizující minimální počet kreditů, které by student měl v daném semestu nabýt, aby splnil požadavky dané

studijním řádem. Graf zobrazující počet kreditů v aktuálním semestru také zobrazuje celkový počet kreditů, které by student měl v optimálním případě na konci semestru mít, aby stihl dokončit studium v běžné době. Náhled možný v příloze na obrázku B.5.

5.6.6 KPI - Aktuální semestr

U této komponenty nedošlo k realizaci plánovaného napovídání známek. Dále byly rozděleny milníky do kategorií, mezi kterými může student přepínat. I poslední komponenta je k vidění v příloze na obrázku B.6.

Kapitola 6

Testování

Testování je nedílná součást procesu vývoje software. V této kapitole budou představeny druhy testování, které byly nad aplikací provedeny, a také se popíše nasazení této aplikace do pilotního provozu.

6.1 Jednotkové testy

Jednotkové testy jsou hlavním nástrojem vývojáře přímo při vývoji aplikace. Jednotkový test má za úkol otestovat jednotku. Takovou jednotkou se rozumí ucelená část aplikace, kterou může být jedna nebo více úzce spjatých tříd[22]. Tyto testy by měli být snadno opakovatelné a rychlé, aby se mohli provádět často. Měli by být také izolované od ostatních jednotek.

Pro jednotkové testování této aplikace jsou používány frameworky JUnit[10] a Mockito[11]. Jednotkové testy aplikace jsou obsaženy v adresáři *src/test/java* v balíčku odpovídající názvem umístění testované jednotky.

6.1.1 JUnit

JUnit je framework určený k psaní jednotkových testů. Testy jsou napsány jako metody třídy označené anotací. Samotný test většinou probíhá tak, že se v testovací metodě zavolá testovaná metoda, případně více testovaných metod. Výstup tohoto volání je pak porovnán s očekávaným výstupem v takzvaném assert volání, které v případě selhání vyhodí výjimku. Pokud nedojde k vyhození výjimky, test je považován za úspěšný. Samotný framework nabízí samozřejmě daleko více možností jak testovat, toto je však nejběžnější postup jednoduchého testu.

Samotné testy jsou provedeny automaticky díky sestavovacímu nástroji maven, který se spouští při sestavování aplikace. Tak je dosaženo snadného znovuopakování testů a kontrole jejich výsledku při každé změně v aplikaci.

[10] `http://junit.org/`
[11] `http://mockito.org/`

6.1.2 Mockito

Pro testování jednotek, které potřebují interakci s webovými službami nebo databází, je zapotřebí se této potřeby nějakým způsobem zbavit. K tomu je využíván nástroj Mockito, který umožní předdefinovat takzvané mock nebo stub objekty. Tyto objekty slouží k zastoupení jiného objektu tak, že se předdefinuje chování (u mock objektů) nebo přímo definují návratové hodnoty pro specifická volání, která se volají v testech (stub)[23]. V případě jednotkových testů této aplikace se bude využívat převážně stubů.

6.2 Integrační testy

V rámci testování byly dále provedeny také integrační testy. Ty měly za úkol otestovat spolupráci více jednotek dohromady zároveň s databází a webovými službami.

K nahrazení databáze používané v provozu byla použita databáze H2[12]. Tato databáze je použitelná tzv. in-memory neboli v paměti. Je tak ideální pro integrační testování, protože je přenositelná zároveň s celým projektem. Byly napsány jednoduché sql skripty, které připraví databázi pro testování.

Pro integrační testování komponent, které používají KOSapi, došlo k použití mock rest serveru, který je dostupný v rámci Spring frameworku. Ten odchytává volání na rozhraní KOSapi a vrací předem připravené odpovědi. Dále také sleduje, zda nedošlo k více voláním, než by v rámci testu bylo očekáváno, nebo zda aplikace posílá správné http hlavičky. Integrační testy jsou stejně jako jednotkové testy umístěny v adresáři *src/test/java* v balíčku odpovídající názvem umístění testované jednotky.

6.3 Testování v prohlížečích

Webová aplikace by měla být možná zobrazit, a to bez chyb, v běžně používaných prohlížečích. Proto se běžně provádí ověření, že tomu tak skutečně je. Testování bylo provedeno na následujících webových prohlížečích.

- Mozilla Firefox 43
- Google Chrome 47
- Internet Explorer 11

V těchto prohlížečích se aplikace ukázala bez problémů, proto je možné ji prohlásit za kompatibilní s těmito prohlížeči. V případě Mozilla Firefox a Google Chrome lze předpokládat i kompatibilita z verzemi staršími, nicméně nebylo to otestováno.

[12] `http://www.h2database.com/`

6.4 Testy s uživateli

Došlo také na uživatelské testování v rámci testů s uživateli. Aplikace byla spuštěna několika studentům, kteří měli možnost aplikaci projít a vyjádřit se k ní. Byl kladen důraz na to, zda student dokáže zjistit přínosná data. Každý student byl nejprve dotazován na rozvržení komponent po dashboardu. Dále došlo po vyzkoušení každé komponenty ke zjištění, které poznatky testující student z dané komponenty získal, a zda to odpovídá požadavkům, které jsou na ně kladeny.

V rámci výstupu z testování byly opraveny zejména grafické chyby, došlo však také na předělání jednoho z grafů komponenty kredity, který byl pro uživatele nesrozumitelný. Dále byly rozdělené milníky do kategorií, mezi kterými je možné přepínat. Také se rozšířily popisky u jednotlivých grafů a milníků některých grafů.

6.5 Nasazení

V rámci testování došlo také k nasazení aplikace na fakultní vývojářský Liferay portál.

V první fázi byl použitý systém Jenkins[13], kterým je zajišťována průběžná integrace. To znamená, že při každé změně kódu v repozitáří, který v případě této aplikace spravuje GitLab[14], dojde k automatickému sestavení, spuštění testů a nasazení aplikace. Jenkins lze také nastavit, aby informoval v případě chyby v jakékoliv fázi nasazení, čehož bylo v rámci vývoje této aplikace využito.

V druhé fázi došlo k zpřístupnění aplikace na url na vývojářském Liferay portálu. Byla vytvořena nová stránka, kde byl nastaven Liferay theme IS FIT. Aplikace je tak nyní dostupná na testovací url.

[13] `https://jenkins-ci.org/`
[14] `https://about.gitlab.com/`

Závěr

V této práci byl obecně charakterizován dashboard a vyznačeny zásady jejich tvorby. Byly rozlišeny jednotlivé typy dashboardu, ukázány jejich klady a zápory a naznačeny případy, při kterých je použití dashboardu daného typu vhodné. Bylo ukázáno, že operační dashboard je nejvhodnějším typem pro potřeby tvorby studentského dashboardu. Dále byly zkoumány různé dashboardy v univerzitním prostředí v České republice i v zahraničí. Nebyl však nalezen žádný, který by se přibližoval požadavkům, jež byly na dashboard v této práci kladeny.

Dále byla provedena analýza uživatelských potřeb studenta ČVUT se zaměřením převážně na studenta Fakulty informačních technologií. Bylo představeno, co dané potřeby zahrnují a zda jsou tyto potřeby v současné době uspokojovány. V případě, že ano, je zhodnocen stávající způsob řešení a zmíněny jeho klady a zápory. Pokud ještě potřeba není žádným způsobem řešena, je navržen způsob, kterým ji uspokojit lze.

Z této analýzy vzešla implementační část práce. Ta se zaměřila na jeden z hlavních současných nedostatků. Tím je nejednotný náhled na studijní výsledky a úspěchy studenta a s tím související chybějící motivační prvky. Došlo proto k návrhu několika klíčových ukazatelů výkonnosti, z nichž bylo pět vybráno pro rozšířený návrh a implementaci. Implementace probíhala podle standardních postupů softwarového inženýrství.

Jelikož je kvalita klíčových ukazatelů výkonnosti závislá na vhodných datových zdrojích, bylo nutné se zamyslet, zda jsou takové datové zdroje dostupné. Pokud ano, bylo těchto dat využito. V případě, že vhodná data zatím dostupná nebyla, došlo k vytvořením dat simulovaných. Aplikace byla následně otestována běžnými postupy testování softwarového projektu. Nakonec byla nasazena na vývojářský Liferay portál fakulty.

Některé komponenty aplikace jsou tak připravené k použití, u jiných bude potřeba ještě připravit datové zdroje. Poté bude možné aplikaci nasadit také do produkčního prostředí, kde již poslouží zvýšení motivace studentů a jejich případným lepším studijním výsledkům při studiu na této fakultě.

Literatura

[1] Hetherington, V.: The Dashboard Demystified: What is a Dashboard? Dostupné z: http://www.dashboardinsight.com/articles/digital-dashboards/fundamentals/the-dashboard-demystified.aspx

[2] Gonzalez, R.: The Top Benefits and Implementation Pitfalls of Business Intelligence Dashboards. Dostupné z: http://www.claraview.com/?q=blog/2013-05-22/top-benefits-and-implementation-pitfalls-business-intelligence-dashboards

[3] Šedá, J.; Tyllich, M.: *Integrovaný studijní informační systém.* Dostupné z: http://isis.vse.cz/dok_server/dokumenty_cteni.pl?id=51&dok=274

[4] Jirůtka, J.: Sirius. Dostupné z: https://rozvoj.fit.cvut.cz/Main/sirius

[5] Mika, M.: Messenger for Liferay - Features. Dostupné z: http://marcelmika.com/lims/features

[6] Heglas, D.: Portlety pro přístup ke studijním materiálům.

[7] Jirůtka, J.: KOSapi - Projekt KOSapi. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki

[8] Cheung, B.: Liferay Adopting the LGPL License. Dostupné z: http://www.liferay.com/web/bryan.cheung/blog/-/blogs/liferay-adopting-the-lgpl-license

[9] Rigsby, J.: Gartner's Magic Quadrant For Horizontal Portals: Oracle, IBM, Microsoft, SAP, Liferay Top Dogs. Dostupné z: http://www.cmswire.com/cms/customer-experience/gartners-magic-quadrant-for-horizontal-portals-oracle-ibm-microsoft-sap-liferay-top-dogs-017717.php

[10] Hepper, S.: *JSR 286: Portlet Specification 2.0*. IBM corp. Dostupné z: https://www.jcp.org/en/jsr/detail?id=286

[11] Gothe, D.: OpenPortal Portlet Container in Liferay. Dostupné z: https://www.liferay.com/web/deepak/blog/-/blogs/openportal-portlet-container-in-liferay

[12] Ozawa, H.; Ferrer, J.; Trinchán, J. M.; aj.: Portlets. Dostupné z: http://www.liferay.com/community/wiki/-/wiki/Main/Portlets

[13] Gosling, J.; Joy, B.; Steele, G.; aj.: *The Java Language Specification Java SE 8 Edition*. Oracle America. Dostupné z: https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf

[14] Haggar, P.: Java bytecode. Dostupné z: http://www.ibm.com/developerworks/library/it-haggar_bytecode/

[15] Daněček, J.: BI-APJ - Přednáška č.4 - JVM.

[16] Johnson, R.; Hoeller, J.; Donald, K.; aj.: *Spring Framework Reference Documentation*. Dostupné z: http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/pdf/spring-framework-reference.pdf

[17] Fowler, M.: Inversion of Control Containers and the Dependency Injection pattern. Dostupné z: http://www.martinfowler.com/articles/injection.html

[18] Islam, H.: JSR 286 Portlet Life Cycle. Dostupné z: http://proliferay.com/jsr-286-portlet-life-cycle/

[19] Roth, M.; Pelegrí-Llopart, E.: *JavaServer Pages Specification*. Dostupné z: http://download.oracle.com/otndocs/jcp/jsp-2.0-fr-oth-JSpec/

[20] Dalling, T.: Model View Controller Explained. Dostupné z: http://www.tomdalling.com/blog/software-design/model-view-controller-explained/

[21] Jirůtka, J.: OAuth 2.0. Dostupné z: https://rozvoj.fit.cvut.cz/Main/oauth2

[22] Fowler, M.: UnitTest. Dostupné z: http://martinfowler.com/bliki/UnitTest.html

[23] Fowler, M.: Mocks Aren't Stubs. Dostupné z: http://martinfowler.com/articles/mocksArentStubs.html

Příloha A

Seznam použitých zkratek

KPI Key performance indicator

IS Informační systém

UK Univeryita Karlova

GNU Lesser General Public License

JVM Java Virtual Machine

MVC Model-view-controller

EE Enterprise Edition

AOP Aspect-Oriented Programming

JSP JavaServer Pages

HTML HyperText Markup Language

XML Extensible markup language

CSS Cascading Style Sheets

ÚTVS Ústav tělesné výchovy a sportu

FIT Fakulta informačních technologií

ČVUT České vysoké učení technické

RSS Rich Site Summary

API Application program interface

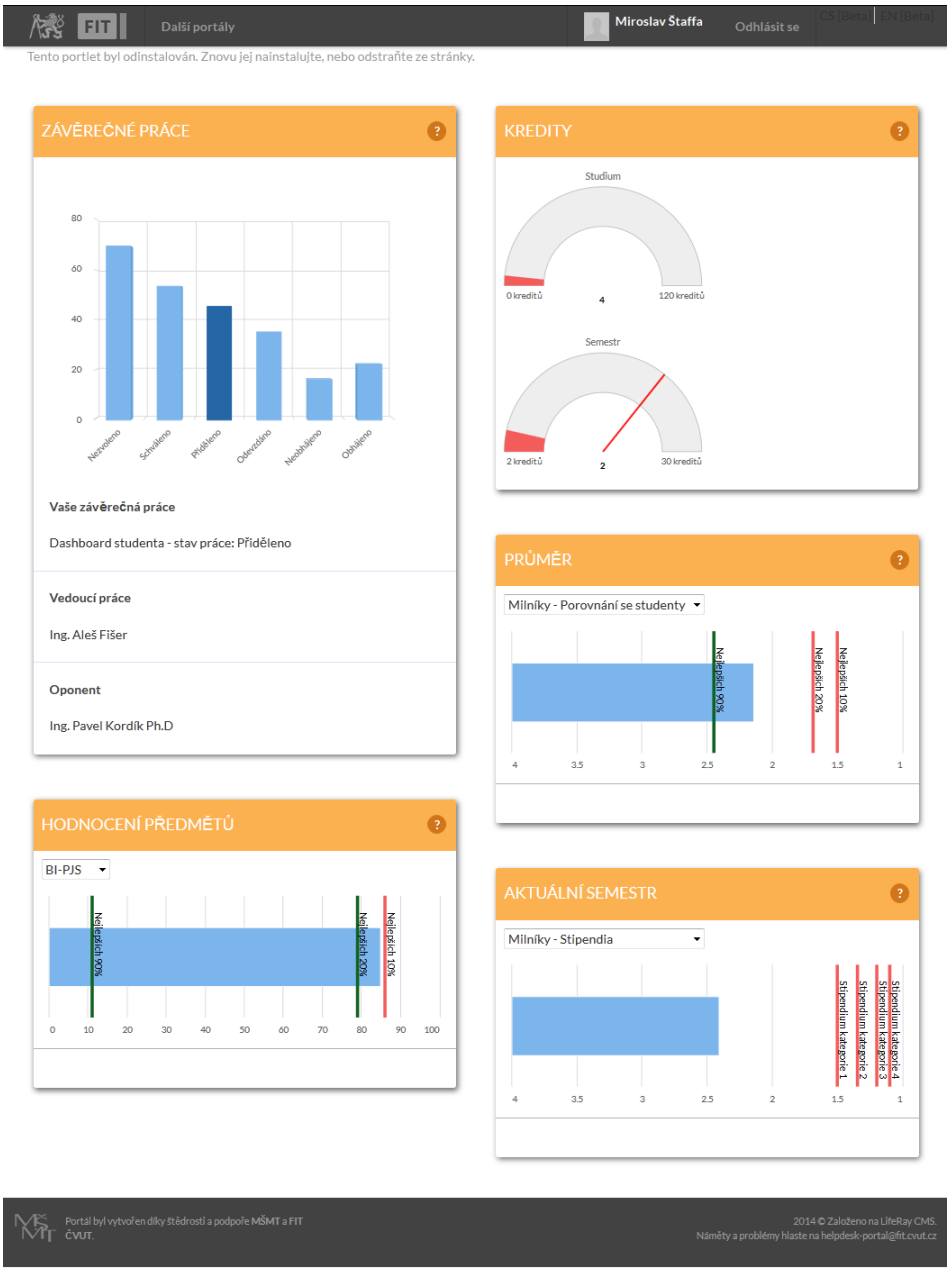
KOS Komponenta studium

URL Uniform resource locator

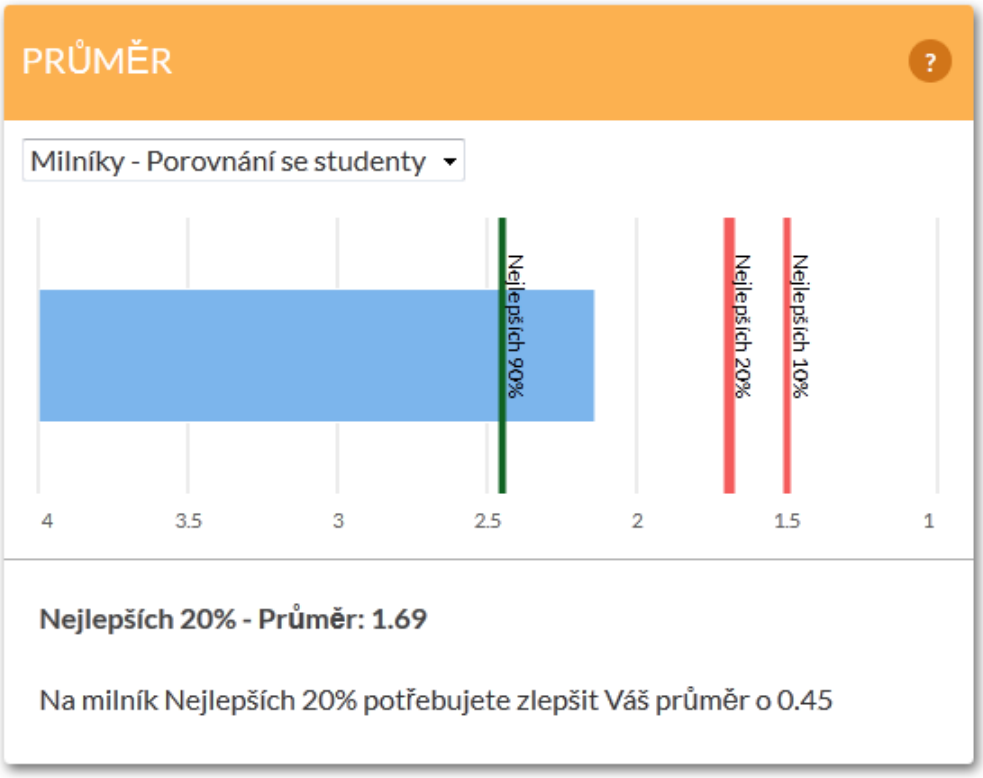
GUI Graphical user interface

Příloha B

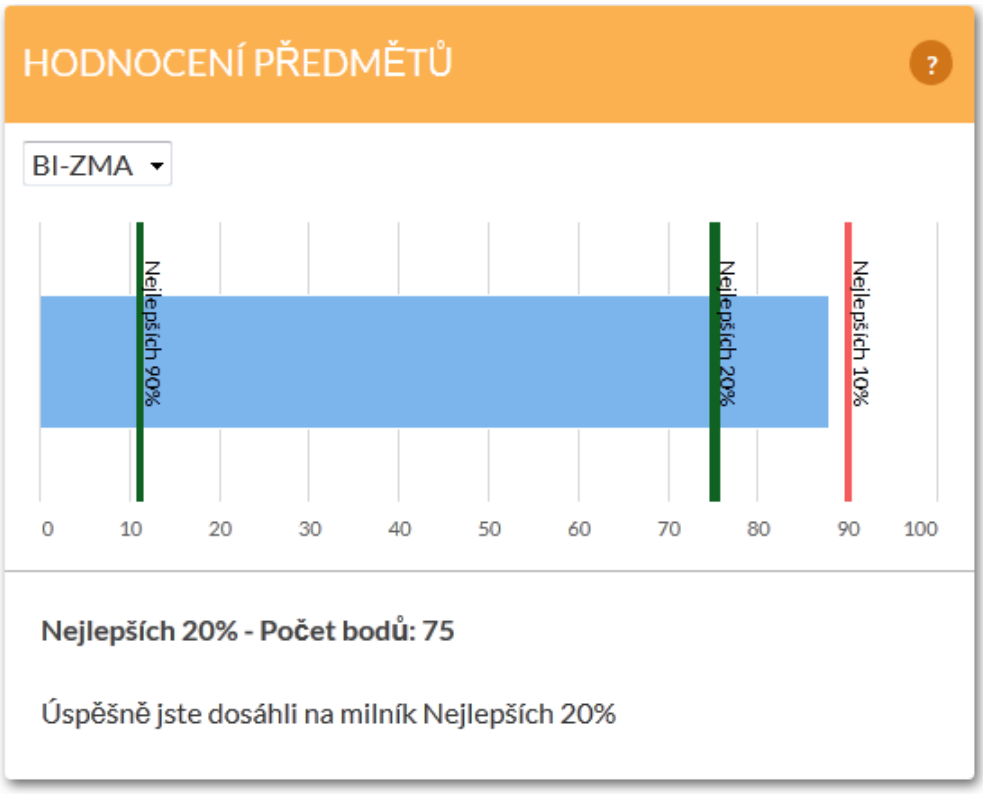
Obrázky aplikace



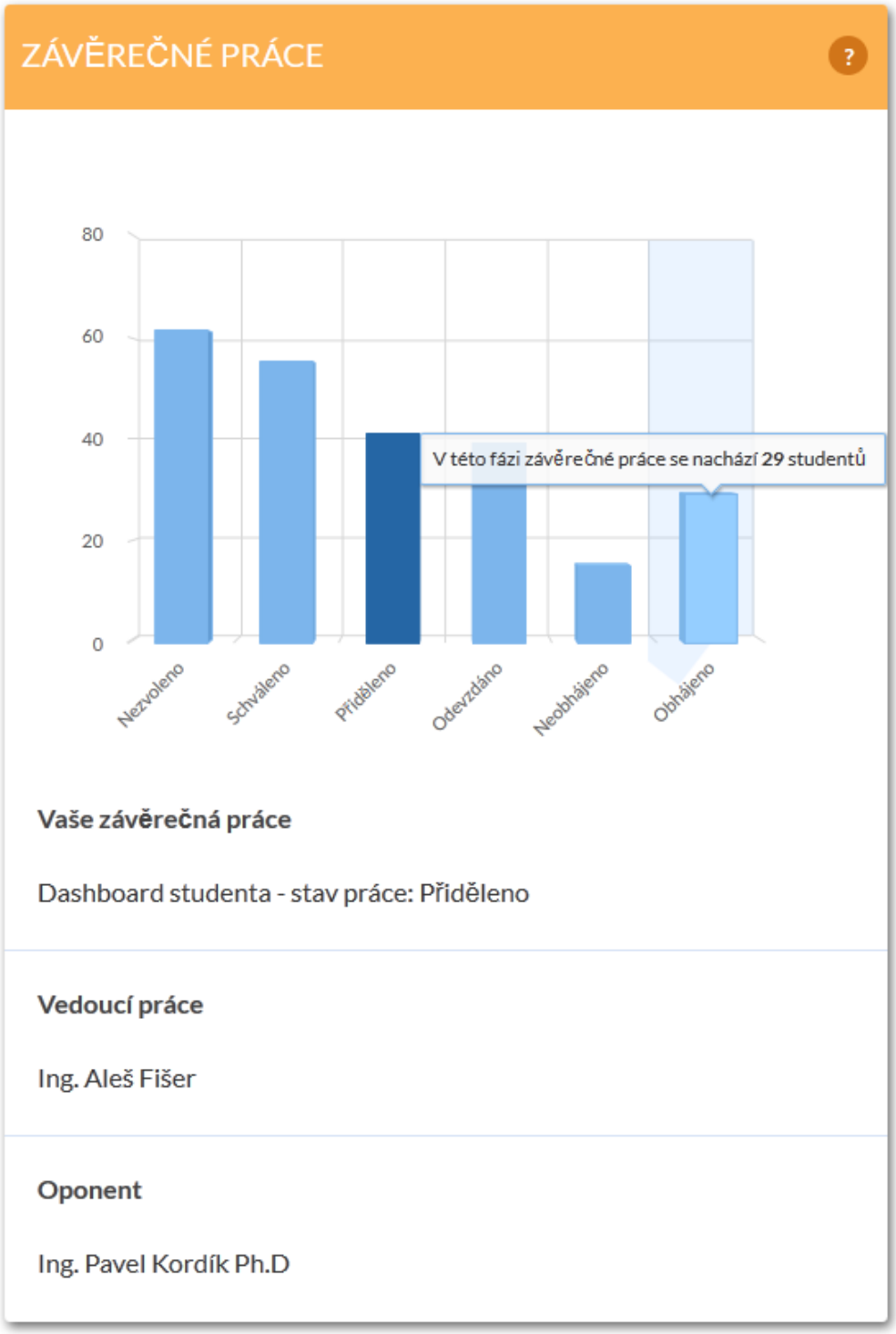
Obrázek B.1: Dashboard - obrázek 1



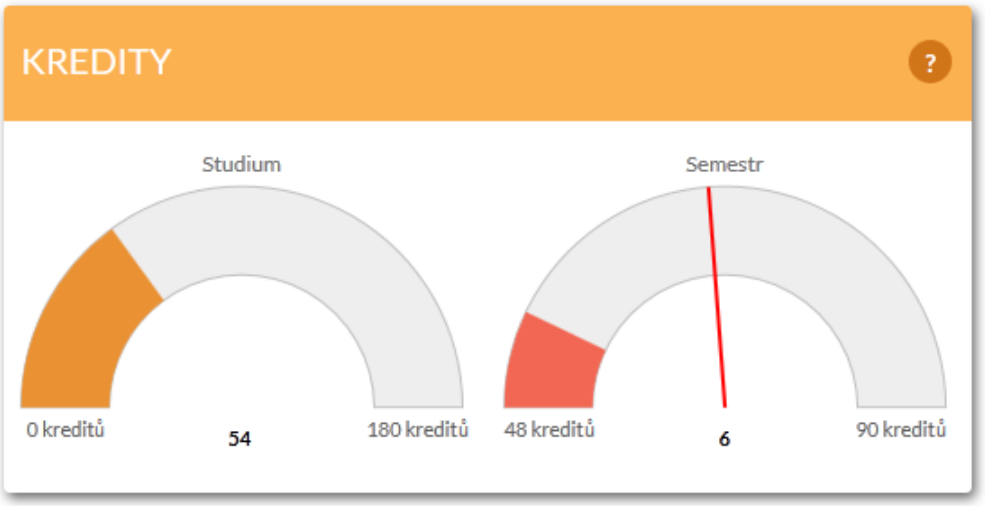
Obrázek B.2: KPI - Průměr



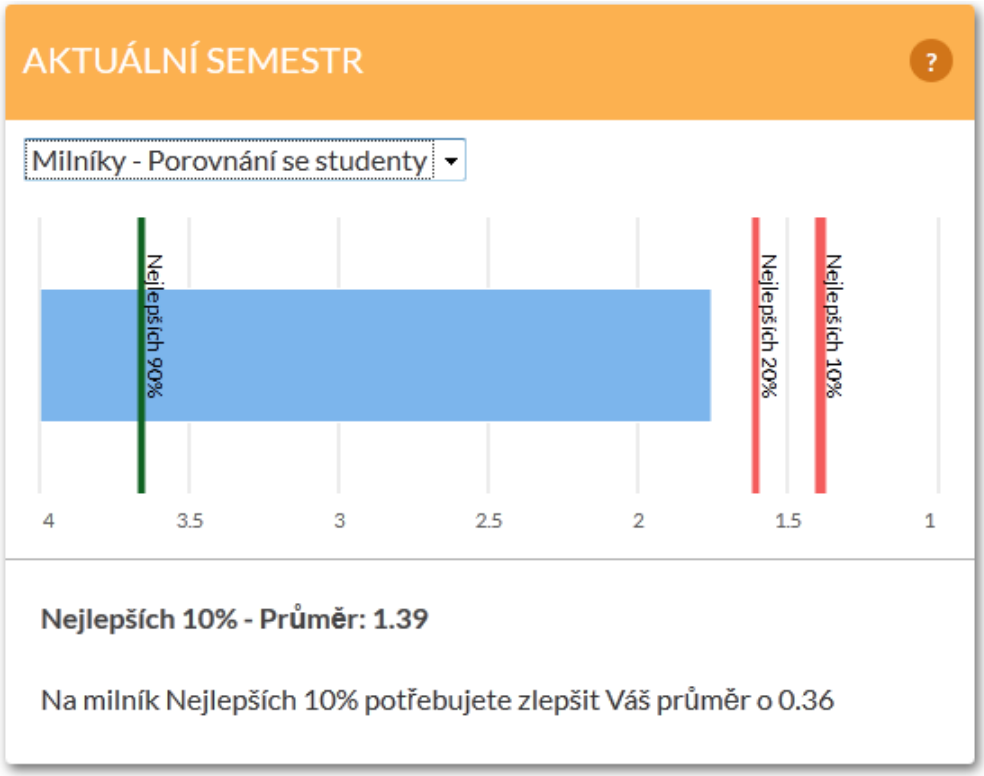
Obrázek B.3: KPI - Hodnocení v předmětech



Obrázek B.4: KPI - Závěrečné práce



Obrázek B.5: KPI - Kredity



Obrázek B.6: KPI - Aktuální semestr

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `war` ..... adresář s archivem obsahující spustitelnou formu aplikace
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `impl` ..... zdrojové kódy implementace
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF