Sem vložte zadání Vaší práce.

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

Diplomová práce

Panel pro vizualizaci pedagogických výkonů

Bc. Marie Remešová

Vedoucí práce: Ing. Pavel Kordík, Ph.D.

19. května 2015

Poděkování

Velmi ráda bych zde poděkovala svému vedoucímu, Ing. Pavlu Kordíkovi, Ph.D. za trpělivost a cenné rady pro psaní práce, Ing. Michalovi Valentovi, Ph.D. za konzultace, svému manželovi a celé rodině za oporu a trpělivost, kterou po dobu zpracování mé práce se mnou měli.

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 19. května 2015 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2015 Marie Remešová. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Remešová, Marie. *Panel pro vizualizaci pedagogických výkonů.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato práce se zabývá metodami vývoje portletů pro portál Liferay. Věnuje se popisu, získávání a následné vizualizaci dat z fakultou shromažďované databáze úvazků.

V teoretické části se dále věnuje pro úlohu vhodným metodám vizualizace a volbě barev pro tvorbu barevných škál. Podstatnou část práce tvoří zpracování a následná vizualizace dat za využití knihovny D3.js. V rámci práce byla vytvořena sada dashboardů, které zaměstnancům fakulty umožní zorientovat se v úvazcích a díky nově implementovanému mechanismu porovnávání pomůže managerům při řízení chodu fakulty i jednotlivých kateder.

Klíčová slova D3.js, úvazky, Liferay portál, portlet, vizualizace

Abstract

This thesis deals with methods for porlet development for the Liferay portal. It covers the description, acquisition and following visualisation of data from the faculty's employee load database.

The theoretical part examines visualisation methods convenient for this type of application and the choice of colors for different colors scales. The mainstay of the thesis consists of processing and the subsequent visualisation of

data using the D3.js library. During the work on the thesis, a set of dashboards was created to enable the employees of the faculty to better orient in their work loads and thanks to the newly implemented comparison mechanism will help the managers with running the faculty and its departments.

Keywords D3.js, loads, Liferay portal, portlet, visualization

Obsah

Úvod **1**

1 Cíl práce **3**
1.1 Popis řešeného problému 3
1.2 Popis struktury práce ve vztahu k vytyčeným cílům: 4
1.3 Výstup práce 5

2 Analýza a návrh **7**
2.1 Seznámení s daty obsaženými v databázi úvazků 7
2.2 Rešerže aktuálních portálů fakulty a použitých technologií . . . 9
2.3 Analýza a návrh implementace produktu 13
2.4 Vizualizace dat 21
2.5 Podpora pro mobilní platformu 25
2.6 Liferay 27

3 Implementační část **31**
3.1 Návod pro snadný vývoj portletů 33

4 Vytvořené portlety **37**
4.1 Portlet dekompozice úvazků na dílčí aktivity 37
4.2 Portlet pohledu děkana na úvazky celé fakulty 42
4.3 Portlet srovnání úvazků napříč semestry 45
4.4 Portlet porovnání úvazků zaměstnanců napříč katedrou 47
4.5 Tvorba barevných škál pro grafy 49
4.6 Načítání dat pro prezentaci 50
4.7 Řešené problémy s daty uloženými v databázi 51

5 Testování **53**
5.1 Uživatelské testování 53

Závěr **55**
Shrnutí 55
Navazující práce 55
Možnosti navázání na tuto práci 55

Literatura **57**

A Seznam použitých zkratek **59**

B Obsah přiloženého CD **61**

Seznam obrázků

2.1 Metodika výpočtu pedagogického výkonu. Obrázek byl získán z dokumentace úvazkostroje [1] 8
2.2 Uživatelské rozhraní "úvazkostroje" 9
2.3 Model fungování MVC[1] 10
2.4 Ukázka současného řešení 11
2.5 Ukázka současného řešení 12
2.6 Ukázka sloupcového grafu 22
2.7 Ukázka výsečového grafu 22
2.8 Ukázka spojnicového grafu 23

3.1 Hierarchie třídy portletu MVCPortlet. 32

4.1 Screenshot portletu grafu dekompozice úvazků na dílčí aktivity. . . 38
4.2 Class diagram třídy pro datovou reprezentaci koláčového grafu. . . 39
4.3 Screenshot dekompozičního grafu úvazku po zanoření do druhé vrstvy. 41
4.4 Screenshot portletu pohledu děkana s katedrami v první úrovni grafu. 42
4.5 Screenshot portletu pohledu děkana s typy pracovního poměru v první úrovni grafu. 44
4.6 Screenshot srovnávacího portletu napříč semestry s hraniční křivkou normy na plný úvazek. 46
4.7 Screenshot srovnávacího portletu napříč katedrou v neanonymizované podobě s hraniční křivkou normy plného úvazku. 48
4.8 Screenshot srovnávacího portletu napříč katedrou v anonymizované podobě s hraniční křivkou reálného požadavku na úvazek. 49
4.9 Class diagram načítání dat z databáze či datového skladu 50

Úvod

Prvotní motivací pro výběr tohoto tématu byla uživatelská zkušenost s mnohými informačními systémy, které jsem měla možnost během svého studia na FIT ČVUT poznat a využít. Některé systémy podléhají správě univerzity, jiné systémy spravuje přímo fakulta. Zorientovat se mezi všemi prostředími i adresami a najít tu správnou aplikaci, která obsluhuje námi řešený požadavek, bylo často netriviální úlohou. V průběhu mého bakalářského studia na MFF UK jsem přišla jako student do kontaktu s jednotným informačním systémem, který byl schopný obsloužit většinu mých požadavků. Takovým systémem však ČVUT prozatím nedisponuje. Na FIT ČVUT v nedávné době vznikla iniciativa, která si klade za cíl vytvořit jednotný portál, který by obsloužil co možná největší množství požadavků studentů a zaměstnanců v rámci jednoho systému, v jednotném grafickém prostředí, s jedním přihlášením a z jedné URL adresy. Mým vedoucím diplomové práce vypsané téma zahrnovalo návrh a implementaci dílčí komponenty pro tento systém a zároveň mi nabízelo možnost seznámit se s dosud pro mě neznámým prostředím portálu Liferay.

Tato práce je, stejně jako již několik prací předchozích, na něž navazuji, součástí projektu, který má za cíl přispět k vytvoření nového informačního systému FIT ČVUT.

Kapitola 1

Cíl práce

Úkolem mojí práce je seznámit se a následně zmapovat možnosti tvorby interaktivních panelů v portálu Liferay, navrhnout a implementovat postup umožňující snadno vytvářet uživatelské dashboardy zobrazující informace a data z fakultních databází, a tento postup demonstrovat na vybraných úlohách. Jako hlavní úlohu, kterou v rámci práce zpracovávám, je vizualizace pedagogických výkonů v databázi úvazků. Vedle možnosti dekompozice výkonů až na úroveň dílčích aktivit jednotlivých pracovníků si kladu za cíl, aby mnou vytvořená sada panelů umožnila i porovnávání výkonů jednotlivých pracovníků mezi sebou navzájem. A to jak v anonymizované, tak v neanonymizované podobě dle přidělených přístupových práv. Mnou vytvořený systém by měl umožňovat jeho snadnou rozšířitelnost na datový sklad.

1.1 Popis řešeného problému

Naše fakulta shromažďuje každý semestr velké množství dat o svých studentech a zaměstnancích. Tato data je potřeba nejen shromáždit a uložit, ale zároveň i důležité statistiky ze sbíraných dat prezentovat osobám, kteří chod fakulty řídí. Vytvořením snadno obsluhovatelné a přehledné komponenty je možné ušetřit čas a usnadnit práci při řešení přidělených úkolů. Takovýchto komponent, které obsluhují různé úkony, je zapotřebí vyvinout mnoho. V rámci své diplomové práce se budu věnovat získávání a vizualizaci dat věnujícím se zaměstnaneckým úvazkům, jejich ohodnocením a složením. Vzhledem k tomu, že v rámci české akademické obce jsou vědci hodnocení dle "výkonnosti" a vysoké školy jsou pak podle výkonnosti svých vědeckých pracovníku financovány, měla by efektivita vědecké činnosti na jednotlivých vysokých školách (či fakultách) být mapována a díky statistikám snadno monitorována a snáze řízena. Základním požadavkem na mnou vyvíjený systém je jeho jednoduchost a snadná rozšiřitelnost. Systém by měl v ideálním případě umožňovat jednotný přístup všem oprávněným uživatelům v režimu 24/7 za použití jed-

notných přihlašovacích údajů do IS. Měl by umožňovat snadnou customizaci a přidělování přístupových práv různým kategoriím uživatelů.

Těmto požadavkům nejlépe vyhovuje aplikace v podobě webového portálu, který je schopen pod sebou integrovat velké množství odlišných funkcí, tj. diskusní fóra, e-mailové klienty, kalendáře apod. Výše zmíněné vlastnosti webového portálu zjednodušují přístup k informacím přes jednotné rozhraní, umožňují sdílení informací či případnou integrací dat z jiných zdrojů. Díky propracovanému systému zabezpečení (prostřednictvím ověřování přihlašovacích údajů), kterým obvykle portálová řešení disponují, poskytuje bezpečnou platformu pro ukládání informací.

1.2 Popis struktury práce ve vztahu k vytyčeným cílům:

Řešení úlohy tak, jak je specifikována v zadání práce, je rozsáhlejšího charakteru. Pro snadnější orientaci v úloze i v textu jsem svou úlohu rozdělila na dílčí podúlohy, kterým se budu ať už teoreticky, nebo ve fázi implementace, ve své diplomové práci věnovat.

1. Popis dat obsažených v databázi úvazků.
2. Stručná rešerže aktuálně dostupného řešení
3. Sběr uživatelských požadavků na vyvíjené portlety.
4. Nastudování existujících typů grafů a nalezení kritérií pro výběr vhodného typu grafu pro vizualizaci vzhledem ke konkrétním datům.
5. Rešerže technologií používaných v novém IS (Liferay, PostgreSQL).
6. Seznámení s dostupnými a existujícími knihovnami (například D3.js) a prostředky pro vizualizaci a tvorbu grafů.
7. Výběr pohledů a tvorba dotazů pro získání potřebných a zajímavých dat z databáze úvazků.
8. Návrh uživatelského rozhraní, volba vhodných grafů, využívaných technologií.
9. Implementace.
10. Testování.

1.3 Výstup práce

Nabyté vědomosti a nově vytvořenou funkcionalitu systému demonstruji v praktické části své diplomové práce na aktuálních datech z databáze úvazkostroje. K využití bude předána sada portletů.

Kapitola 2

Analýza a návrh

2.1 Seznámení s daty obsaženými v databázi úvazků

Výkony jednotlivých pracovníků fakulty jsou na Fakultě informačních technologií ČVUT evidovány přímo v informačním systému ČVUT [2].

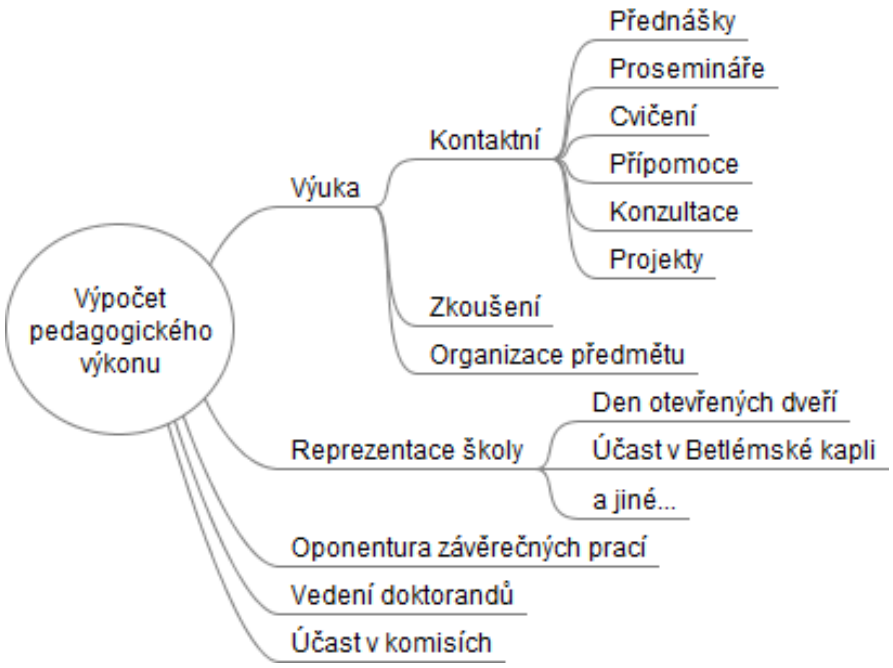
Tento systém byl původně koncipován jako webová aplikace, v níž bylo možno nejen počítat výkony jednotlivých pracovníků, ale také plánovat pedagogické zatížení na další semestry. Možnost plánování však není v informačním systému využívána. - Systém v současné době slouží výhradně k evidenci a vyhodnocování pedagogických výkonů pracovníků v rámci jednotlivých semestrů.

Klíčovým bodem celého návrhu je metodika či výběr parametrů, podle kterých bude efektivita jednotlivých pracovníků měřena. Je tedy nejprve nutné vstupní parametry nějak definovat, vymezit a kvantifikovat.

Tyto parametry můžeme rozdělit na 3 základní okruhy: pedagogický výkon (tj. výuka, komise a vedení prací) / věda / projekty. Co se týče pedagogických výkonů je také nutno nejprve vymezit jejich časové omezení. V akademické sféře se pro časové vymezení činnosti nejčastěji užívá akademický rok, jenž je dále možno rozdělit na semestry, výukové týdny atd. V rámci semestru tedy můžeme sledovat "výkonnost" daného pracovníka - tj. výuka, hodnocení a garance předmětů,počet vypsaných prací, počet vedených prací, počet rezervovaných prací, počet oponentur. Výpočet pedagogického výkonu se provádí za každý semestr zvlášť, nicméně směrodatné je výsledné hodnocení za oba semestry dohromady.

Dále pak jsou v rámci semestru / akademického roku sledovány aktivity pracovníků v oblasti vědy a práce na projektech. V tomto případě je tedy možné jednotlivé pracovníky v tomto směru vzájemně porovnávat či porovná-

[2] Podrobnější informace k hodnocení pracovníků viz [2] Samotná dokumentace metodiky výpočtu je umístěna na samostatné stránce viz [1].



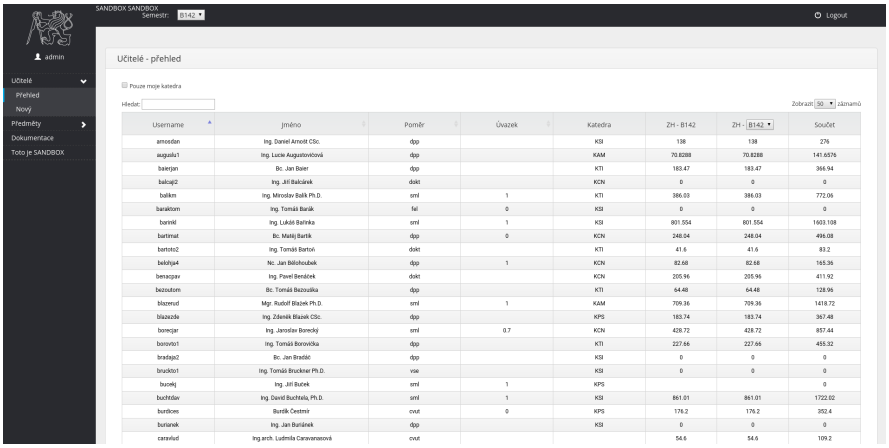
Obrázek 2.1: Metodika výpočtu pedagogického výkonu. Obrázek byl získán z dokumentace úvazkostroje [1]

vat vůči nějaké referenční hodnotě, kterou může být děkanem pevně stanovený počet prací, které by měl každý zaměstnanec v daném akademickém roce splnit.

Dále pak je možno v aplikaci sledovat počet odučených hodin za semestr, počet cvičených předmětů apod. Tato oblast je na FIT ČVUT hodnocena počtem započítatelných hodin - tj. pro každý semestr je dán konkrétní referenční počet započítatelných hodin, jenž odpovídá plnému úvazku.

Podrobněji se metodice výpočtu pedagogického výkonu věnuje bakalářská práce Josefa Němečka[3], v níž demonstruje podíl jednotlivých činností na pedagogickém výkonu, viz obrázek 2.1.

Jak jsem již výše uvedla, jsou uživateli systému jednotliví zaměstnanci fakulty, jejichž výkony jsou v informačním systému evidovány, na straně jedné a pracovníci fakulty, kteří jsou kontrolou a rozvrhováním jednotlivých činností (výkonů) pracovníků pověřeni - tzv. “úvazkáří”, na straně druhé. Tito pracovníci do informačního systému zaznamenávají data potřebná k výpočtu pedagogického výkonu a plánují také vytížení pedagogů v budoucích semestrech. Na obrázku 2.2 je uživatelské rozhraní v současnosti používaného a na portalu fakulty dostupného “úvazkostroje” pro simulaci úvazků.

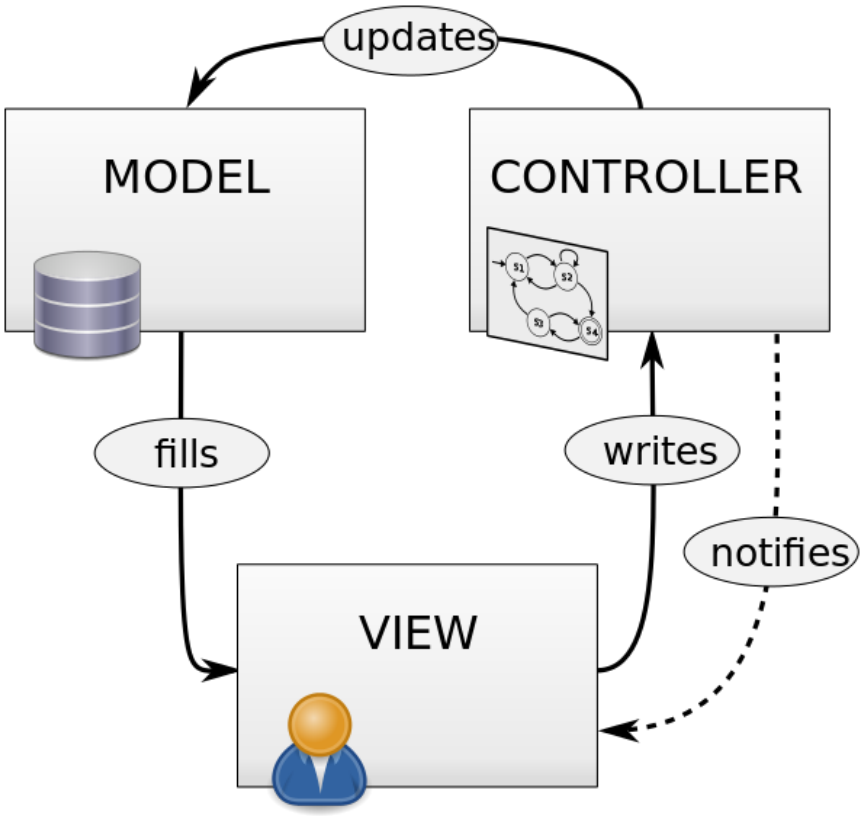


Obrázek 2.2: Uživatelské rozhraní “úvazkostroje”

2.2 Rešerže aktuálních portálů fakulty a použitých technologií

Informační systém ČVUT je jako webová aplikace postavena na třívrstvé MVC (Model-view-controller) architektuře. Tato architektura (často také uváděná jako návrhový vzor) rozděluje datovou vrstvu (model), prezentační vrstvu (uživatelské rozhraní / view) a doménovou vrstvu (řídící logiku / controller) do tří nezávislých komponent tak, že modifikace kterékoliv z nich má jen minimální vliv na ostatní komponenty. V tomto modelu datová vrstva zprostředkovává komunikaci s databází, získává z ní potřebná data a po eventuální úpravě tato data zpět do databáze ukládá. Vrstva doménová data získaná z datové vrstvy vyhodnocuje a poté je zobrazuje v uživatelském rozhraní. Prezentační vrstva ve formě uživatelského rozhraní zajišťuje prezentaci dat uživateli.

MVC architektura představuje způsob, jak zajistit flexibilitu a spolehlivost aplikace, a to i při častých změnách a jejich rychlém vývoji. V MVC architektuře jde zjednodušeně řečeno o vzájemné oddělení aplikační a prezentační vrstvy. Vztah mezi datovou a doménovou vrstvou - tj. vztah směrem od modelu ke controlleru - není architekturou MVC pokryt. Tuto oblast řeší PostgreSQL.



Obrázek 2.3: Model fungování MVC[3]

Princip fungování MVC architektury je následující:

1. Uživatel provede nějakou akci v uživatelském rozhraní

2. Řadič obdrží oznámení o této akci z objektu uživatelského rozhraní.

3. Řadič přistoupí k modelu a v případě potřeby ho zaktualizuje na základě provedené uživatelské akce, případně zpracuje změněná data.

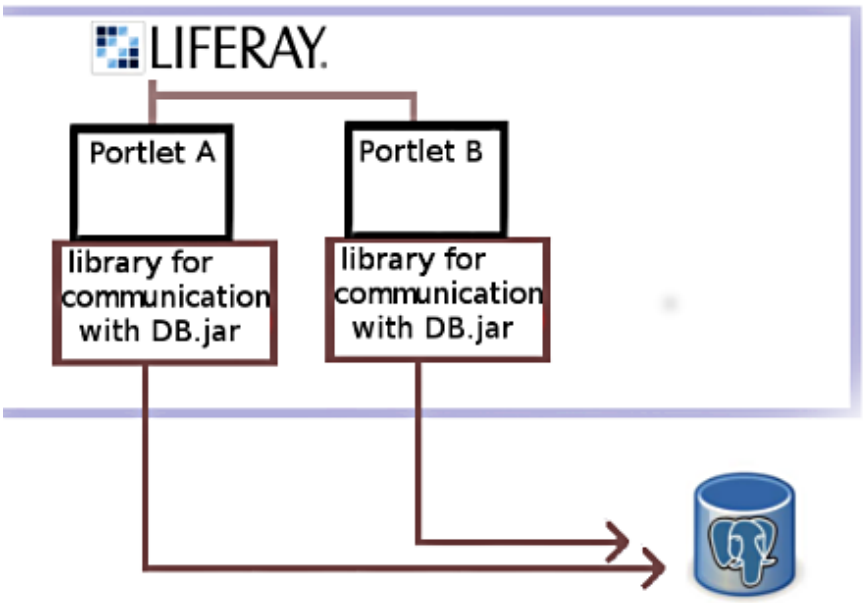
4. Komponenta View použije zaktualizovaný model pro zobrazení zaktualizovaných dat uživateli, přičemž View získává data přímo z modelu, zatímco model nepotřebuje žádné informace o komponentě View (je na ní nezávislý). Controller ale nepředává doménové objekty (model) komponentě View, nicméně jí může poslat příkaz, aby svůj obsah podle modelu zaktualizovala.

[3]Obrázek převzat z Wikipedie: [4]

5. Samotnému konečnému zobrazení výsledku uživateli ještě může u web-aplikací předcházet odpověď ze serveru na klienta, aby si ihned vyžádal obnovení stránky (client side redirect, životnost 0, takže okamžitý): Tím je zaručeno, že při obnovení stránky uživatelem (refresh, F5 v prohlížeči) nevyvolá na serveru požadovanou akci opakovaně, ale že se jedná pouze o obnovení pohledu, nyní už bez požadavku na změnu dat (modelu). Účelem je změna URL a dat http requestu, aby poslední v řadě již nebyl "server-side data-affecting"(ovlivňující model), ale pouze "read-only"(pouhé zobrazení). Celý tento client-refresh (změna URL) se děje automaticky a bez povšimnutí uživatelem.

6. Uživatelské rozhraní čeká na další akci uživatele, která celý cyklus zahájí znovu.

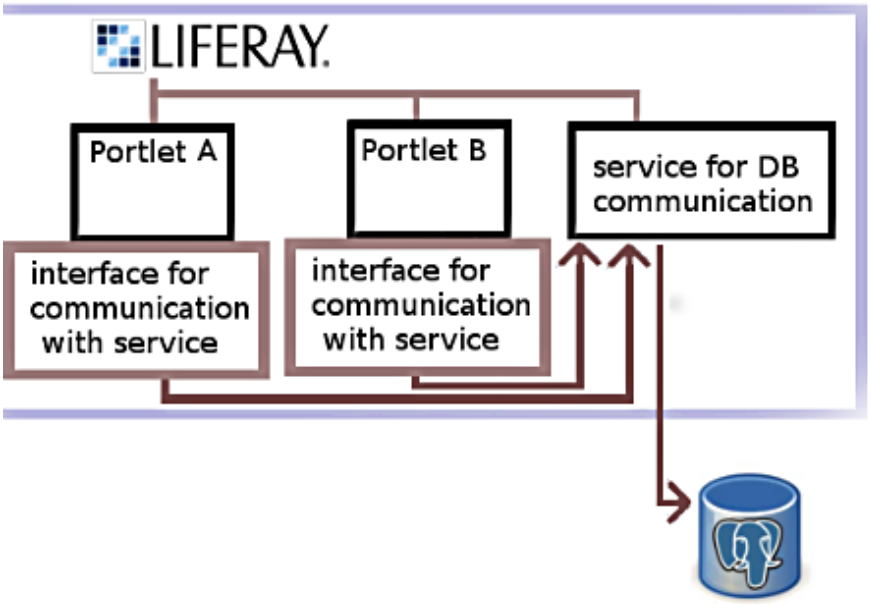
V současné době je v Informačním systému ČVUT zaveden mechanismus pro sdílení služeb přes JavaCode na úrovni Liferay. Aktuální způsob komunikace jednotlivých portletů s databází je takový, že si každý portlet s sebou nese vlastní JAR knihovnu, která vytvoří nové spojení do databáze atd. (viz Obr. 2.4)



Obrázek 2.4: Ukázka současného řešení

Po úspěšném zavedení mechanismu by komunikace měla fungovat tak, že portlety znají pouze rozhraní pro komunikaci se službou (knihovnou) a vlastní implementace služby je ve WAR souboru, který se deployuje na Liferay zvlášť,

přičemž pouze tato služba má spojení do databáze. (viz Obr. 2.5) To znamená, že v tomto typu komunikace jsou 3 moduly - klient (tj. portlet využívající službu), služba samotná (WAR, která obsahuje implementaci dané služby) a interface.



Obrázek 2.5: Ukázka současného řešení

Databáze KOS poskytuje aplikační rozhraní (API) v podobě RESTful webových služeb, které zprostředkovává přístup k vybrané části dat v databázi KOS, čímž odstraňuje nutnost zpracovávání exportů, neustálou duplikaci všech dat a potíže s jejich udržováním. RESTové služby staví na osvědčených konceptech webu jakožto distribuovaného prostředí vzájemně provázaných informací. KOSapi umožňuje a podporuje vznik školních i studentských aplikací, které pro svou činnost vyžadují online aplikační přístup k datům souvisejícím s výukou. Tvoří také jednu ze základních komponent nově vznikajícího Informačního systému ČVUT (IS ČVUT). Systém je zabezpečen autentizací prostřednictvím LDAP, to znamená, že heslo se přenáší přes šifrovaný kanál, nikdo k němu nemá přístup.

Co se týče použitých technologií, na kterých je aplikace KOSapi postavená, je jich několik. Celý systém je postaven na platformě Java EE 7 (OpenJDK). Pro jednoltivé oblasti jsou použity následující frameworky:

- Spring Framework (app platforma, loC kontejner)
- Spring Security (pro autentizaci a autorizaci)

- RESTEasy
- Hibernate ORM

Použitými knihovnami je Eclipse (resp. EclipseLink MOXy), ASpectJ a několik knihoven vyvinutých přímo pro KOSapi. Použitými nástroji pro správu a řízení aplikací jsou Maven 2 a Git. Pro serverové prostředí jsou použity Tomcat 7, PostgreSQL 9 a Gentoo Linux.

2.3 Analýza a návrh implementace produktu

Základním požadavkem na systém je jeho jednoduchost a snadná rozšiřitelnost. Systém by měl v ideálním případě umožňovat jednotný přístup velkému množství uživatelů v režimu 24/7 a použití jednotných přihlašovacích údajů. Měl by umožňovat snadnou customizaci a přidělování přístupových práv různým kategoriím uživatelů.

Těmto požadavkům nejlépe vyhovuje aplikace v podobě webového portálu, který je schopen pod sebou integrovat velké množství odlišných funkcí, tj. diskusní fóra, e-mailové klienty, kalendáře apod. Výše zmíněné vlastnosti webového portálu zjednodušují přístup k informacím přes jednotné rozhraní, umožňují sdílení informací případnou integraci dat z jiných zdrojů. Díky propracovanému systému zabezpečení (prostřednictvím ověřování přihlašovacích údajů), kterým obvykle portálová řešení disponují, poskytuje bezpečnou platformu pro ukládání informací.

Data jsou v portálech obvykle ukládána v podobě portletů, tj. malých Java aplikací.

Portálová řešení si obecně nekladou za cíl do sebe integrovat aplikace jako DMS či CMS, ale díky portletům čerpat informace a ty zobrazovat uživateli na jednom místě. Portál je tedy sada webových stránek,k terá automaticky nabízí uživateli mnoho možností – přihlásit se ke svému účtu, číst obsah z několika různých zdrojů, nastavit si rozložení stránky či vyhledávat. Portál tedy "spravuje" společné úkoly (správu uživatelů a skupin, nastavení grafického vzhledu,...) a obsah samotný dodávají jednotlivé portlety do "okének", které jim portál na stránce vyhradí. Pro zobrazování informací se využívá portletů, což jsou malé grafické komponenty. Zjednodušeně řečeno je portlet další rozhraní pro komunikaci se systémem. Portlety jsou používané jako výměnné komponenty uživatelského rozhraní poskytující prezentační vrstvu pro informační systém. Součinnost portálů a portletů je zajišťována pomocí API, které je definováno v Java Portlet Specification [5]. Díky tomuto API mohou být portlety spuštěny na jakémkoliv portálu založeném na Java EE bez ohledu na jeho implementaci.

Portlety jsou ve své podstatě velmi podobné servletům. Stejně jako servlety jsou portlety řízeny portletovým kontejnerem, který zpracovává požadavky a generuje dynamický obsah. Portletový kontejner je „obal“, který port-

lety spouští, řídí jejich životní cyklus, zajišťuje prostředky potřebné pro běh portletů a poskytuje persistentní úložný prostor pro jejich nastavení. Portlet přijímá a zpracovává požadavky, podle kterých pak mění svůj obsah. Portlety jsou používáné portály jako zásuvné komponenty uživatelského rozhraní. Obsah generovaný portletem, také nazývaný „fragment“, je kus kódu ve značkovacím jazyce (HTML, XHTML, WML). Jednotlivé fragmenty se pak mohou spojovat a dohromady tvořit portálovou stránku. Webový klient komunikuje s portletem pomocí standardního paradigmatu požadavek/odpověď implementovaného portálem a uživatelé tak většinou mohou interagovat s portletem stejným způsobem jako se pohybují po běžných webových stránkách. Obsah generovaný portletem se může lišit v závislosti na uživateli a uživatelském nastavení portletu.

Portlety jsou univerzální aplikace, takže se dají začlenit do jakéhokoliv portálu, pokud splňují standardy. Standardy pro portlety (JSR-168 či JSR-286) definují rozhraní, jaké používají portlety při komunikaci s portálem. Uživatel nikdy nekomunikuje přímo s portletem – každý požadavek (odeslání formuláře, kliknutí na odkaz) si nejprve přebere portál, pošle ho správnému portletu, ten ho zpracuje a vytvoří nějaký výsledek. Výsledek pak portlet pošle zpět portálu, který ho zobrazí uživateli.

2.3.1 Životní cyklus portletu

Jak jsem již uvedla, o portlet se stará a řídí ho portletový kontejner, a to po celou dobu životního cyklu portletu. Každý portlet má implementované čtyři metody řídící jeho životní cyklus. Tyto metody jsou:

- $init(PortletConfigconfig)$
- $processAction(ActionRequestrequest, ActionResponseresponse)$
- $doView(RenderRequestrequest, RenderResponseresponse)$
- $destroy()$

Inicializační metoda $init(PortletConfigconfig)$ je volána ihned po tom, co je instance portálu vytvořena. PortletConfig reprezentuje konfigurační data (pouze pro čtení), která jsou specifikována v portlet.xml souboru. Například zde můžeme najít inicializační parametry. Metoda

$$processAction(ActionRequestrequest, ActionResponseresponse)$$

je volána při odpovědi na podnět od uživatele (například kliknutí myší na odkaz nebo odeslání formuláře). V této metodě mohou být volány i metody business komponent, jako jsou JavaBeans. ActionRequest a ActionResponse jsou pod-rozhraní PortletRequest a PortletResponse. Pomocí této metody může portlet změnit své nastavení nebo perzistentní informace.

Metoda

$$doView(RenderRequestrequest, RenderResponseresponse)$$

v životním cyklu portletu je vyvolána při prvním vykreslení portletu, dále při vykreslení po akcí vyvolaném přesměrování na stránku, které metoda přísluší a dále pak vždy, pokud v některém z ostatních portletů na stránce je vyvolána akce. Obecně se dá říci, že tato metoda generuje obsah založený na aktuálním stavu portletu a vždy zajistí jeho vykreslení do aktuálně platné podoby.

Poslední metodou je metoda *destroy*(). Tato metoda je volána předtím, než je instance portletu „předhozena“ garbage collectoru. V této metodě se tedy provádějí „zametací“ akce, kdy dochází například k uvolňování prostředků a zdrojů.

Portlet koná svou činnost ve dvou fázích, které se vážou ke dvěma druhům požadavků portálu.

Základní je fáze “**render**”, ve které portlet poskytuje portálu nějaký obsah k zobrazení. Render fázi může předcházet fáze “**action**”, ve které portlet provede nějakou akci, na základě níž pak upraví svůj obsah pro následující render fáze.

Navrhovaný systém by měl být rozdělen na 2 části - část serverovou a část klientskou.

2.3.2 Klientská část - Liferay

Klientská část systému bude využívat technologie Liferay a půjde o portletovou aplikaci, která bude přístupná přes webové rozhraní.

Důvodem proč si zvolit právě Liferay portál je fakt, že umožňuje správu dat, aplikací a procesů z jednoho centrálního uživatelského rozhraní, přičemž pro běžného uživatele se několik heterogenních systémů tváří jako jeden systém a portletům lze nastavovat různá práva dle rolí. Podnikový portál Liferay[6], jak jsem již výše uvedla, je založený na jazyce Java. Je dostupný jak v komerční, tak i open-source komunitní verzi, která je šířena pod licencí GNU LGPL a dalšími proprietárními licencemi. Je distribuován ve dvou edicích: CE (Community edition), která je zdarma a je určena pro méně náročné nasazení, a EE (Enterprise edition), která se od CE liší podporou a samozřejmě cenou. Je určena pro reálné nasazení v prostředí internetu či intranetu. Pro možnost bezplatné instalace portálu na vlastním počítači jsem zvolila pro účel vývoje a testování verzi CE.

Data se v Liferay ukládají v relační databázi. Nabízeno je hned několik možností. Jmenujme například Oracle, MySQL, MSSQL či PostgreSQL. Liferay obsahuje i velké množství již integrovaných rozšíření jako chat, forum, mailový klient, kalendář a umožňuje také použití vlastních modulů pro podporu DMS, CMS či běh v clusteru. Samotná instalace Liferay portálu je velmi snadná. Portál se skládá z jednotlivých stránek, do kterých lze vkládat integrované nebo nově vyvinuté aplikace ve formě portletů. Portlety jsou v podstatě

samostatné funkční jednotky jejichž logika a funkcionalita je naprogramovaná v jazyce Java.

Výhodou použití Liferaye je jeho nezávislost na použité platformě, neboť Liferay podporuje všechny známé operační systémy, aplikační servery a databáze. Uživatelské rozhraní Liferaye má i lokalizaci pro češtinu. K jeho dalším výhodám patří to, že mezi základní funkce Liferay portálu patří zabezpečená SSO (Single Sign On) autentizace, nástroje pro správu workflow, snadné uspořádání prvků pomocí drag-and-drop, automatické nahrávání souborů přes WebDAV a tagování a vyhledávání v obsahu portálu.

Liferay CMS je systém pro správu obsahu, pravděpodobně nejdůležitější nástroj portálu. Pomocí něj mohou uživatelé nejen spravovat, ale i vytvářet nový obsah přímo z prohlížeče. Mohou upravovat celkový vzhled portálu volbou různých témat, vkládat a odebírat portlety, vytvářet nové stránky a upravovat jejich obsah. Vestavěný editor umožňuje publikovat webový obsah bez jakékoli znalosti programování. Prostřednictvím CMS má také uživatel přístup k obrázkové galerii a knihovně dokumentů, které mohou být sdíleny v rámci celého podniku, určité skupiny, nebo pouze jednotlivce.

Liferay nabízí podporu pro různá rozšíření, ať už vzhledu nebo funkčnosti portálu. Pro tyto účely existuje několik typů pluginů. Layout Templates – Šablony, jejichž účelem je přizpůsobení rozložení jednotlivých portletů v portálu. Hook plugins – Vhodné pro rozšíření a úpravu základních funkcionalit portálu.

Liferay Portal je založen převážně na Java technologiích, které zahrnují například Apache ServiceMix, Ehcache, jBPM, ICEfaces, Apache Lucene, MuleSource ESB, Spring 3.0 & AOP, Struts & Tiles, Apache a JBoss Seam. Kromě toho využívá i následující technologie: jQuery, JavaScript či PHP. Liferay Portal podporuje standardy JSR-168, JSR-127, JSR-170, JSR-286 a JSF-314.

Pro vývoj portletů a portálových aplikací (více spolupracujících portletů) pro Liferay lze využít většinu rozšířených IDE jako Eclipse (STS), NetBeans či IntelliJ Idea. Jako buildovací nástroje lze využít Ant či Maven. Při dekompozici zadaného úkolu jsem zjistila, že je třeba na navrhovaný systém pohlížet ze dvou uživatelských pohledů:

1. informativní pohled, poskytující informace o jednom uživateli

2. porovnávací pohled, poskytující data ke srovnání více uživatelů nebo napříč jednotlivými kategoriemi

První z nich by měl umožňovat zobrazovat pouze charakteristiky a statistiky jediného uživatele. Druhý v pořadí by měl umožňovat vzájemné srovnání mezi jednotlivými zaměstnanci, srovnání napříč semestry a porovnání podílu věda / výuka / propagace. A to i napříč jednotlivými katedrami. Co se týče uživatelských práv, první přístup by měl být zobrazitelný výhradně "řadovému" zaměstnanci fakulty, zatímco druhý v pořadí, umožňující srovnání, by

měl být přístupný "vedení" fakulty a vedoucím jednotlivých kateder (byť v anonymizované podobě).

Tyto požadavky Liferay jednoznačně splňuje. Velmi snadno je v Liferay řešitelné zobrazování obsahu v závislosti na definovaných rolích a oprávněních. Pro single user je možná personalizace webových stránek. K autentizaci uživatele je možné využít komponenty pro integraci s LDAPem, OpenId nebo Facebookem.

Ve své základní podobě se portlet skládá z hlavní třídy, JSP stránek a tzv. deployment descriptorů (popisné soubory určené pro portál).

JSP (JavaServer Pages) je technologie podobná PHP, ale využívá jazyk Java. JSP stránka obsahuje jednak základní HTML tagy, ale navíc i speciální tagy či přímo kusy Java kódu, které portálový server zpracuje, než pošle stránku klientovi.

Hlavní třída portletu vždy dědí od abstraktní třídy

$$javax.portlet.GenericPortlet$$

u níž zpravidla potřebujeme přepsat tyto metody:

- $init()$ – inicializace portletu, volá se jen jednou, a to při začlenění (deploy) portletu do portálu
- $doView()$ – metoda obsluhující render fázi
- $processAction()$ – metoda obsluhující akční fázi

Předávání parametrů mezi portálem a portletem probíhá pomocí objektů dvou tříd: v akční fázi ActionRequest a ActionResponse, v render fázi RenderRequest a RenderResponse. Zde se návrháři inspirovali komunikací u protokolu HTTP. První objekt obsahuje parametry požadavku a druhý objekt obsahuje odpověď portletu.

Z pohledu JSP stránky jsou součástí požadavku, kdežto odpovědí je vytvořená stránka.

2.3.3 PostgreSQL

Jak jsem již uvedla dříve, Liferay ukládá data do relační databáze. Pro nasazení na úlohu, kterou ve své práci řeším, se jako nevhodnější ukázalo užití PostgreSQL. Škálovatelnost PostgreSQL umožňuje použití i pro masivní OLTP systémy a datové sklady určené pro tisíce uživatelů. PostgreSQL je nabízena v několika produktových řadách, od komunitního open source, po robustní řešení s podporou 24x7. PostgreSQL řeší současný přístup více uživatelů prostřednictvím systému známého jako Multi-Version Concurency Control (MVCC) , který každému uživateli zpřístupňuje "snapshots"databáze a umožňuje provádět změny, aniž by je ostatní uživatelé před potvrzením transakce viděli. To do velké míry eliminuje potřebu zamykání a zajišťuje, že databáze efektivně splňuje ACID principy.

2.3.3.1 Nahrání portletů na server

Po vytvoření portletu je možné v rámci Eclipse stisknutím pravého tlačítka myši a vybráním nabídky Run As –> Run on Server nahrát dosud nenahraný nebo pozměněný projekt. Další volbou je nahrání přes Liferay –> SDK –> Direct Deploy, který nahraje soubory projektu, pokud již existuje dříve nahraná verze projektu na serveru.

Webové aplikace (portlety) lze do portálu přidávat přes výchozí menu nabídku Add –> Applications výběrem portletu ze zobrazených kategorií a pouhým přetažením na stránku, kde chceme mít portlet umístěn.

Význam jednotlivých konfiguračních souborů je k dispozici například zde [7] a bude dále rozebrán později v textu.

Vytvořený portlet pak již stačí nahrát na námi vytvořený server samotným přetažením. Po nahrání se portlet objeví v nabídce dostupných portletů a my ho již můžeme nahrát na naši stránku přes výše zmíněný postup Add –> Applications. Nové portlety jsou zobrazovány pod kategorií Sample (závisí na informaci v liferay-display.xml).

Vytvořený portlet pracuje s architekturou Liferay MVC. Pokud bychom potřebovali realizovat převod na Spring Portlet MVC , provedli bychom to následujícím způsobem:

1. V souboru portlet.xml změníme

   ```
   <portlet-class>
     com.liferay.util.bridges.mvc.MVCPortlet
   </portlet-class>
   ```

   na

   ```
   <portlet-class>
     org.springframework.web.portlet.DispatcherPortlet
   </portlet-class>
   ```

2. Dle specifikace Spring Porlet MVC dohledává DispatcherPortlet pro portlet vlastní kontext v souboru

   ```
   [portlet-name]-portlet.xml
   ```

3. Přidáme potřebné springovské knihovny – po otevření souboru $liferay-plugin-package.properties$ přes nabídku $PortalDependencyjars \rightarrow Add$

Hlavní způsob, jímž se portlet workflow liší od servlet workflow, je, že požadavek na portlet může mít dvě odlišné fáze: akce fáze a fáze vykreslení. Akce fáze se provádí pouze jednou a jen tam, kde dojde ke změně nebo akci "backend", jako je například provádění změn v databázi. Vykreslovací fáze pak vyrábí to, co se uživateli zobrazí pokaždé, když se zobrazení aktualizuje. Pro jeden požadavek se akční fáze provádí pouze jednou, ale výkonná fáze může být provedena několikrát. To poskytuje (a vyžaduje) jasné oddělení mezi činnostmi, které upravují přetrvávající stav systému a činnostmi, které vytvářejí to, co se zobrazí uživateli.

Skutečnou silnou stránkou specifikace JSR-168[8] je dvojí fáze portletového requestu. To znamená, že například dynamické výsledky vyhledávání mohou být aktualizovány pravidelně na displeji, aniž by uživatel explicitně opětovně spouštěl vyhledávání. Většina ostatních portletů MVC Frameworku v sobě skrývá dvě fáze, aby to vypadalo stejně jako tradiční vývoj servletu. Oddělení těchto dvou fází je zachována v celém frameworku Spring Portlet MVC. Hlavním projevem tohoto přístupu je, že pokud servlet verze tříd MVC bude mít jednu metodu, která se zabývá žádostí, portlet verze tříd MVC bude mít dvě metody, které se zabývají žádostí: jeden pro akční fázi a jeden fází vykreslení. Například tam, kde servlet verze $AbstractController$ má metodu $handleRequestInternal(\ldots)$, portlet verze $AbstractController$ má $handleActionRequestvnitř$ní$(\ldots)$ a $handleRenderRequestInternal(\ldots)$ metody.

Framework je navržen jako DispatcherPortlet, který odesílá požadavky s nastavitelnými parametry zobrazení a rozlišení zobrazení, stejně jako DispatcherServlet ve webovém frameworku. Nahrávání souboru je také podporováno stejným způsobem.

Lokální rozlišení a rozlišení tématu nejsou v portletu MVC podporovány - tyto oblasti jsou v gesci portálu / portlet kontejneru a nejsou vhodné na úrovni Springu. Nicméně všechny mechanismy Springu, které jsou závislé na stavu (jako je internacionalizace zpráv), budou i nadále fungovat, protože DispatcherPortlet funguje podobně jako DispatcherServlet.

Výchozí popisovač je stále velmi jednoduchý řadič rozhraní, které nabízejí pouze dvě metody:

- $voidhandleActionRequest(request, response)$
- $ModelAndViewhandleRenderRequest(request, response)$

Framework má stejnou hierarchii jako AbstractController či SimpleFormController. Vázání dat, příkaz použití objektů, manipulaci s daty či zobrazení jsou stejné jako v rámci servletu.

Všechny možnosti vykreslování servletu jsou použity přímo pomocí speciálního servletu s názvem ViewRendererServlet. Při použití tohoto servletu je požadavek portletu převeden na žádost směrem k servletu a může být vyjádřen pomocí celé obvyklé servletové infrastruktury. To znamená, že všechny

existující renderers, jako jsou JSP, rychlost, atd., mohou být použity v portletu.

Spring Portlet MVC podporuje vlákna, jejichž životní cyklus je dán rozsahem aktuálního požadavku nebo HTTP Session. To sice není specifikem samotného Spring Portlet MVC, ale spíše kontejneru WebApplicationContext (y), který Spring Portlet MVC používá. Portlet MVC je request-driven webový MVC rámec, navržený pro portlet, který odesílá požadavky a nabízí i další funkce usnadňující vývoj portletových aplikací. Spring DispatcherPortlet však znamená víc než jen to. Je v něm plně integrován Spring ApplicationContext.

Stejně jako běžné portlety, je DispatcherPortlet deklarován v xml souboru webové aplikace:

```
 <portlet>
    <portlet-name>sample</portlet-name>
    <portlet-class>
      org.springframework.web.portlet.DispatcherPortlet
    </portlet-class>
    <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>view</portlet-mode>
    </supports>
    <portlet-info>
        <title>Sample Portlet</title>
    </portlet-info>
</portlet>
```

V rámci Portlet MVC má každý DispatcherPortlet vlastní WebApplicationContext, který dědí všechny vlastnosti již definované v Root WebApplicationContext. Tyto parametry lze potlačit v rámci portletu a patřičný rozsah definovat lokálně na danou instanci portletu. Sping DispatcherPortlet má několik speciálních vlastností, které používá, aby bylo možné zpracovávat žádosti.

Proces nasazení aplikace Spring Portlet MVC není jiný, než používají některé portletové aplikace JSR-168. Obecně platí, že kontejner (portálu / portletu) běží v jednom webapp v servletovém kontejneru a portlety je nutno spustit v jiném webapp servlet kontejneru. Specifikace JSR-168 přesně neudává, jakým způsobem je portlet volán, každý kontejner má pro volání svůj vlastní mechanismus, který obvykle zahrnuje jakýsi "proces nasazení", který provádí změny v portletu webapp sám a pak zaregistruje portlety v rámci kontejneru.

Minimálně soubor web.xml v portletu webapp je upraven pro distribuci dat do servletu, takže bude volat kontejner. V některých případech jeden servlet bude poskytovat služby pro všechny portlety ve webapps, v ostatních přípa-

dech bude instance servletu pro každý portlet. Některé portletové kontejnery jsou knihovny a / nebo konfigurační soubory do webapp.

2.4 Vizualizace dat

Vzhledem k tomu, že porovnání (pedagogické a vědecké) činnosti jednotlivých zaměstnanců fakulty je velmi složité, neboť každý z pracovníků v akademické sféře se věnuje jiné oblasti vědy, učí různý počet předmětů (s různou časovou dotací), má odlišný počet diplomantů a doktorandů, pracuje na různě náročných projektech atd., je tedy nejprve nutné najít metodu, jak "výkony" pracovníků hodnotit či porovnávat.

Při analýze této úlohy jsem vycházela z obvyklých zvyklostí a z požadavků ze strany hodnotitelů (z řad vedení pracovišť, ústavů či vedení fakulty / univerzity). Jako nejvhodnější se z jejich pohledu jeví vizuální prezentace těchto informací, a to zejména v podobě grafů.

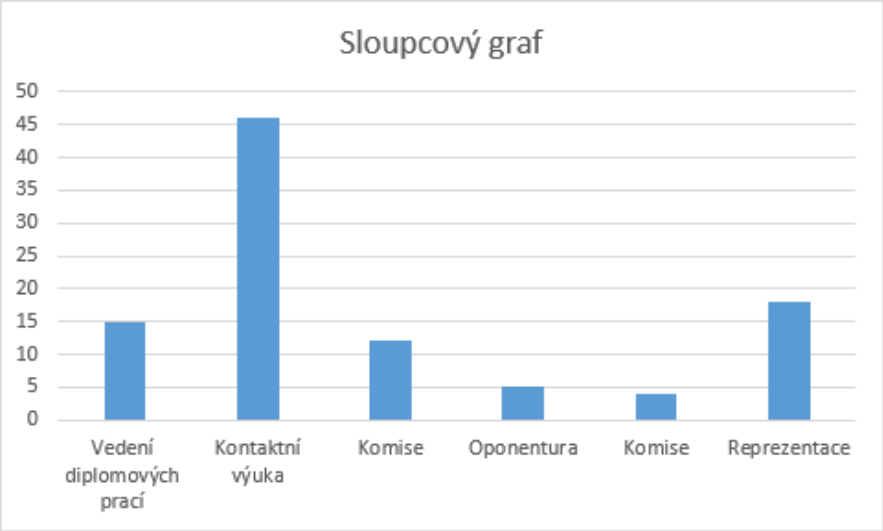
Právě za použití prezentace informací v podobě grafů je možné porovnávat "neporovnatelné". Pro tuto úlohu se tedy výborně hodí metoda vizualizace dat. Zobrazení informací prostřednictvím jejich grafického (vizuální) zobrazení - vizualizace - je prostředkem pro snazší pochopení zobrazovaných informací. Vizualizace pomáhá uspořádat poznatky a najít souvislosti mezi fakty. Výhodou vizualizace je eliminace nebo minimalizace nutnosti numerického porovnávání, při kterém jsou často vyžadovány dodatečné znalosti, například znalost vlastností použitých kvantitativních ukazatelů [9].

Vizualizace dat je jednou z fází dolování dat, proto je kombinací více disciplín jako matematiky (zejména statistiky), dolování dat, grafiky (pro vlastní zobrazení dat) atd. Celý proces návrhu lze rozdělit do několika kroků (od sběru dat, přes jejich filtraci, až po jejich výsledné zobrazení).

Grafy slouží k porovnání vzájemné souvislosti mezi dvěma a více proměnnými, nalezení souvislostí mezi objekty, návazností a toků. Základní typy grafů dělíme dle souvislostí, které graf zobrazuje. Základní dělení je tedy dle níže uvedených charakteristik zobrazovaných dat:

- rozložení (histogram, korelační graf)
- porovnání, a to buď mezi jednotlivými hodnotami (sloupcový graf) nebo v průběhu času (spojnicový graf)
- složení (výsečový/koláčový graf)
- vztahy, ať už mezi dvěma (korelační graf) či více proměnnými (bublinový graf)

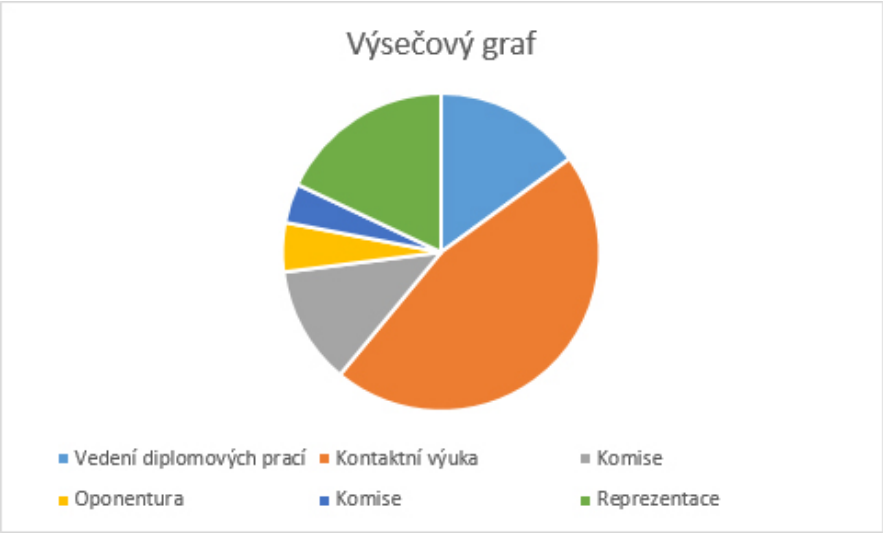
Sloupcový graf (Obr.2.6) znázorňuje změny dat v průběhu časového období nebo znázorňuje porovnání položek. Je vhodný pro zobrazování a porovnávání



Obrázek 2.6: Ukázka sloupcového grafu

hodnot pro různé, obvykle nespojité nominální a ordinální kategorie. Velmi dobře se dají v tomto typu grafu zobrazit data na třech osách.

Výsečový (koláčový) graf (Obr.2.7 slouží k zobrazení datových bodů jako procentní hodnoty z celku. Nevýhodou tohoto grafu je ale to, že pokud je zobrazovaná hodnota příliš malá (pod 5%), je tento graf pro zobrazení dané informace nevhodný.



Obrázek 2.7: Ukázka výsečového grafu

Spojnicový graf (Obr. 2.8) zobrazuje prostřednictvím bodů vztah mezi číselnými hodnotami v několika řadách nebo znázorňuje dvě skupiny čísel jako jednu řadu souřadnic X a Y. Tento typ grafu umožňuje jednoduše vnímat výkyvy v datech z hlediska určitého časového úseku.

Obrázek 2.8: Ukázka spojnicového grafu

Pro prezentaci dat je také možno využít netradiční grafy[4] (jako je třeba prstencový graf, vodopádový graf, Ganttův či Paretův diagram).

Vyjdeme-li z teorie, každý graf se skládá z několika "komponent" [13] - tj. rámec, obsah a popisky. Rámcem v tomto případě rozumíme rozvržení grafu, případně jeho měřítko. Obsahem grafu [5] jsou rozuměny body, sloupce či linie reprezentující samotné kvantitativní údaje v grafu. Popisky grafu určují jeho kontext a jedná se o pojmenování grafu či legendy ke grafu, které slouží ke správném čtení a interpretaci grafu. Jde o to, aby graf byl pochopitelný i pouze sám o sobě. Rozložení těchto prvků je dáno volbou typu grafu.

Při tvorbě grafů, vizualizujících data, je nutné si zvolit několik charakteristik:

- vhodný tvar a typ grafu, který přehledně zobrazí data
- velikost grafu
- podrobnost (tj. jak podrobná data dokáže graf přehledně zobrazit a jaká data je zapotřebí zobrazit již v jiném grafu (podgrafu), na který uživatele odkážeme)

[4] Příklady netradičních grafů viz [10] nebo [11] či [12]
[5] Někteří autoři tento komponent nazývají datovými prvky.

V prvé řadě je nutné stanovení významu grafické reprezentace hodnot. Tj. zda budeme data prezentovat poměrovým způsobem (rozdělení celku na dílčí části dle percentuálního zastoupení, resp. přínosu) či metodou reprezentace nabitých hodnot bez závislosti na oboru hodnot.

Je nutné se zamyslet nad vhodnou volbou barev pro reprezentaci vizualizovaných hodnot. (Předpokládejme bez újmy na obecnosti, že všechny námi vizualizované entity mají omezený obor hodnot.)

Je nutno zvolit vhodnou barevnost grafů [6] (tj. vhodnou barevnu paletu, popřípadě stupnici barev viz např. [15]), která umožní vizuálně oddělit rozdílná data, nebo naopak sjednotit data, jež mají něco společného. Neboť okem vnímáme tři čtvrtiny našich vjemů, je pro přehlednost grafů vhodné používat barvy, neboť prezentace dat za použití pouze černo-bílé barevné škály činí informace nepřehledné a poněkud nudné. Dá se dokonce říct, že barva je centrálním prvkem vizualizace. Byť je vnímání subjektivní, je možno říci, že "význam" některých (základních) barev je u většiny lidí podobný a vychází z intuitivního vnímání barev. - Červená - chyba, něco špatně, nebo má za cíl upoutat pozornost. Základními barvami jsou červená, zelená a modrá, přičemž nejdelší vlnovou délku má barva červená a nejkratší barva modrá. Z těchto 3 základních barev bychom měli vycházet.

Co se týče barvy pozadí, na němž jsou vizualizovaná data zobrazována, ta by měla být neutrální, nedráždivá - nejlépe bílá.

Další otázkou je počet dimenzí grafu - zda zvolit zobrazení ve 2D či ve 3D. Zde platí, že při tvorbě grafů je nutné ctít pravidlo, že používat trojrozměrné grafy je vhodné pouze v případě, že je to nezbytně nutné, tj. v případě, že třetí rozměr poskytuje jakousi další informaci. Pomineme-li tu skutečnost, že 3D grafy (byť jsou to stále pouze dvoudimenzionální data) jsou velmi "náchylné" ke zkreslování informací.

Pro srozumitelnost grafů je velmi důležité umístění popisů grafu (uvnitř / vně / žádné) a také to, zda budou informace v grafu čistě číselné (včetně percentuálních hodnot) nebo zda budou doplněny textovým popisem. Také případné umístění legendy sehrává významnou roli pro snadné pochopení prezentovaných informací.

Pro design informačního systému je také důležité zachování jakéhosi jednotného grafického stylu daného sdělení. V prvé řadě je nutno uvést, že pro snadnou orientaci a přehlednost není vhodné (ba ani žádoucí) kombinace různých fontů (a velikostí písma). V tomto směru je také neméně důležité samotné umístění grafu či hierarchické uspořádání sdělení. Použitý font by měl být "čistý" a bez příkras.

[6] Pro efektivní volbu barev existuje nástroj [14], určený k tvorbě barevných škál, u kterých je zajištěno, že je dokáží přečíst i lidé se všemi typy barvosleposti. Tento nástroj byl vytvořen na základě podrobných výzkumů vnímání barev.

2.5 Podpora pro mobilní platformu

Diskutovanou otázkou také je, zda řešení bude podporovat zobrazování dat přes mobilní zařízení. Dalším dotazem, na který je třeba nalézt odpověď je, pokud by data byla dostupná i pro malá zařízení, jaká data bude možno zobrazovat a jak, případně jaké množství dat bude zobrazováno. Samotný Liferay portál nabízí několik možností, jak konfigurovat způsob zobrazení dat na přístrojích s malým rozlišením (např. smartphonech). Defaultně je zvoleno řešení, kdy jsou obsah a jednotlivé portlety, které jsou na velkém počítači rozmístěny do více sloupců, zařazeny pod sebe v rámci jednoho sloupce a šířka portletů přizpůsobena šířce displaye na mobilním zařízení.

Nikoliv podružnou otázkou také jsou celkové náklady na pořízení a provozování informačního systému s podporou vizualizace dat.

Tolik teorie, nyní přejděme k samotné implementaci návrhu.

Jak jsem již uvedla v úvodu této práce, je předmětem mé práce vizualizace pedagogických výkonů v databázi úvazků. A to jak dekompozice výkonů až na úroveň dílčích aktivit jednotlivých pracovníků, tak i porovnávání aktivit jednotlivých pracovníků mezi sebou navzájem.

Prvním úkolem je tedy vizualizovat stav splnění / naplnění úvazku jednotlivých pracovníků. Po důkladné analýze této úlohy jsem se rozhodla pro vizualizaci této informace zvolit sekvenční barevné schéma. V takovémto schématu je intuitivně dobře vnímán aktuální stav naplnění úvazku. V tomto typu zobrazení se pro nízké hodnoty obvykle používají světlejší barvy.

Pro oblast naplnění úvazku pracovníků jsem se rozhodla použít výsečový graf, neboť v tomto typu grafů je nejlépe viditelný podíl jednotlivých činností na celkovém úvazku.

2.5.1 Datový sklad

Jedním z cílů mé práce je vytvořit takový systém, aby byl snadno napojitelný na datové úložiště. Jako datové úložiště v našem případě slouží relační databáze PostgreSQL. V zadání se však mluví i o získávání dat z datového skladu. Dle původního plánu měly být v průběhu mé práce dokončeny 3 bakalářské práce, věnující se úvazkům. Jedna z nich měla upravit data v databázi a po návrhu vhodné struktury je měla převést do datového skladu. K tomu bohužel ani více než po roce nedošlo, a proto nebylo možné demonstrovat získávání dat z datového skladu, neboť žádná data, týkající se úvazků, v něm importována nebyla. Aby však bylo možné využít funkcionality mnou navrženého řešení, bylo přistoupeno k rozhodnutí vytvořit rozhraní, které umožní po dokončení transferu dat do datového skladu mou aplikaci snadno o propojení k datovému skladu rozšířit.

2.5.2 Srovnání dostupných metod vizualizace dat

Přehled aktuálně používaných technologií vhodných pro vizualizaci, lze včetně návodů na implementaci lze čerpat z [16]

2.5.2.1 D3.js

D3 je zkratkou pro Data Driven Documents. D3.js [7] je DSL (Domain-specific language) pro vizualizaci dat. D3.js je javascriptová knihovna pro manipulaci s dokumenty založenými na datech, která umožňuje převádět data do HTML, SVG či CSS. D3 umožňuje navázat libovolná data do Document Object Model (DOM), a pak aplikovat transformace dat. Disponuje též principy moderních prohlížečů bez vázání se na vhodný rámec, který kombinuje výkonné vizualizační komponenty a přístup dat. D3 je extrémně rychlý, podporuje velké soubory dat a disponuje též dynamickým chováním pro interakci a animace. D3 také umožňuje opakované použití kódu pro různorodou kolekci komponent a pluginů.

2.5.2.2 Google chart tools

Tento software umožňuje tvorbu grafů a vizualizačních schémat s možností integrace do vlastních stránek. Výsledný graf je stejně jako v případě předchozí knihovny vykreslován pomocí jazyka HTML5 do elementu SVG. Tento software je možné používat zcela bezplatně, avšak prozatím nabízí menší množství typů grafů, zejména pak interaktivních.

2.5.2.3 R

Jedná se o programovací jazyk, který se věnuje zejména výpočtům a vizualizaci matematických a hlavně pak statistických dat.

2.5.2.4 Prefuse

Tato sada nástrojů pro interaktivní vizualizaci dat je založena na jazyce Java. Má vlastní vizualizační framework a výsledné vizualizace je možné vykreslit pomocí produktu Adobe Flash Player. Jeho výhodou je především to, že je poskytován pod BSD licencí.

2.5.2.5 MoRiS 32

MoRiS 32 je původní český programový systém pro vizualizaci, monitorování a řízení technologických procesů společnosti BCM Control s.r.o. MoRiS slouží pro vývoj a přípravu dispečerských pracovišť technologických linek, velínů, ale také pro laboratorní a školní účely, zkrátka tam, kde je zapotřebí nejen

[7] Použití technologií pro vizualizaci dat na webu viz [17]

sledování většího množství dat v reálném čase, ale také jejich vyhodnocování, sběr a zálohování. Celý systém je řešen modulárně a je tudíž plně otevřený pro další vývoj. Veškerá práce s ním je interaktivní, využívající moderní metody tzv. vizuálního programování a také ovládání je zcela intuitivní.

2.6 Liferay

2.6.1 Instalace portalu a komponent pro vývoj portletů

Pro možnost vyvíjet vlastní portlety a naučit se s portálem dobře pracovat, doporučuji každému zájemci, stáhnout si potřebný software a po jeho instalaci si spustit vlastní Liferay portál. Vše, co budete potřebovat, lze stáhnout ve 4 archivech.

- Liferay Portal bundle with TomCat
- Liferay Plugins SDK
- Liferay IDE for Eclipse
- Software pro databázi
- JRE 1.6 či vyšší

První z výše uvedených obsahuje zdrojové kódy portálu a také software TomCat, který je zapotřebí pro provozování serveru. Druhou součástí je Liferay Plugins Software Development Kit je vývojové prostředí, které napomáhá při vývoji portletů, motivů (Themes) a rozložení stránek (Layout). Třetí položkou je samotné vývojové prostředí, které je možné nainstalovat do Software Eclipse, které vývojaři poskytuje mnoho utilit usnadňujících vývoj a lazení kódu. Pro chod portálu je nutné provozovat databázi, do které portál umístí své tabulky s daty potřebnými pro spravný chod portálu. V našem případě šlo o instalaci databáze PostgreSQL, která je používána na fakultě jako uložiště dat o úvazcích. Liferay však podporuje i další jako MySQL, SQL Server, DB2 či např. Sybase. Nutným požadavkem je instalace Java Runtime Environment.

Instalace následně probíhá ve třech krocích:

- instalace databáze
- konfigurace Eclipse a spuštění serveru s Liferay portálem
- konfigurace propojení s databází a vytvoření uživatele s plnými oprávněními

Všechny tyto kroky jsou názorně popsány []

Od verze Liferay portálu 6.2 se propojení s databází se konfiguruje snadno vyplněním formuláře po iniciálním spuštění portálu na serveru.

Liferay má několik vestavěných frameworků, které umožňují vytvořit v rámci Liferay IDE portletový plugin a nahrát ho na portál.

- MVC Portlet
- Vaadin
- Struts

Navštívíte-li stránky Liferaye s úvodním návodem pro tvorbu portletů, naleznete zde krok za krokem popsaný postup, jak vyvinout portlet, který vypíše každému programátorovi známý řetězec "Hello World!". Framework, kteremu se věnují v návodech pro snadný vývoj a začínající vývojáře je pro jeho jednoduchost přehlednost vybrán typ MVC portlet. Návod je psán pro vývoj v prostředí Liferay IDE, který lze velmi snadno nainstalovat na každý počítač, jak již bylo dříve popsáno.

2.6.2 Vývoj MVC portletové aplikace

Portál je webová aplikace nebo brána, která hostuje a spouští různé služby volané jejím uživatelem. Umožňuje integrovat různorodé aplikace, nesoucí, případně shromažďující klíčové informace, které slouží uživateli při práci se systémem. Portlety jsou zásuvné komponenty uživatelského rozhraní. Portlet kontejner je komponenta Liferay portálového serveru, která sdružuje portlety a poskytuje jim runtime environment. Stará se také o životní cyklus portletu. Tento životní cyklus se skládá ze dvou fází, fáze akční, při které portlet obsluhuje požadavek a renderovací fáze, kdy dochází k překreslení portletu do aktuální podoby. V akční fázi může být současně maximálně jeden portlet. Po ukončení této akční fáze dochází k přerenderování všech portletů na aktuální liferayové stránce.

2.6.3 Životní cyklus HTTP požadavku/odpovědi zpracovávané Liferay portálem

1. V prohlížeči je odeslán k portálu HTTP request
2. Portál tento požadavek přijme
3. Portál rozhodne, zda příchozí dotaz vyžaduje akci od některého z portletů
4. Pokud požadavek vyžaduje akci některého z portletů, portletový kontejner tuto akci vyvolá
5. Po ukončení této akce přecházejí všechny portlety do renderovací fáze, kdy jsou překresleny do aktuální podoby

6. Portletový kontejner poskytne zpět část HTML/XHTML/WML kódu příslušného portletu
7. Portál sloučí výstup portletů na portletové stránce a odešle ji zpět prohlížeči

Přístup k portálovým stránkám lze omezit na základě uživatelům předem přidělených rolí a oprávnění. Uživatele je dále možné shlukovat do komunit (uživatelů se společnými zájmy), organizací (hierarchické uskupení uživatelů) a v rámci těchto skupin i do týmů (uživatelských skupin se společnou úlohou). Stránky, které jsou ve skupině public mohou zobrazit všichni uživatele portálu, kteří nejsou přihlášeni. Na základě jednotného přihlášení k portálu se jasně stanoví, jakou stránku a které portlety uživateli zobrazit a zpřístupnit k využití.

Dosud byly vydány dvě specifikace pro Java Portlety. První z nich (JSR 168)[8] v roce 2003 a od roku 2008 do dnešního dne používaná specifikace 2.0 označovaná (JSR 286)[5]. Od verze 2.0 byla přidána možnost přijímat a odesílat portletům události a bylo umožněno sdílet veřejné vykreslovací proměnné napříč portlety.

Kapitola 3

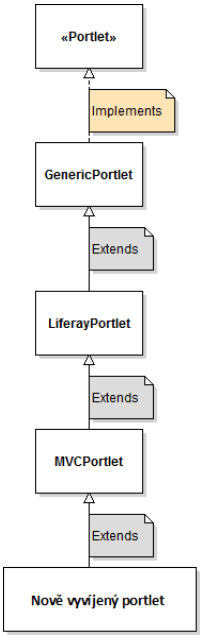
Implementační část

3.0.1 Volba technologií pro implementaci

Samostatný framework pro vývoj, Liferay, byl již stanoven v zadání práce. Jeho největší předností je jeho nezávislost na platformě. Je schopný běžet na každé, která je schopná spustit JAVA Runtime Environment a aplikační server. Můj vývoj probíhal v operačním systému Windows 8. Celý framework je založen na jazyku Java a v tomto jazyce byla vyvíjena hlavní součást mnou vytvořených portálových portletů. O dalších využitých jazycích se zmiňuji dále v podrobnějším popisu implementace, zde jen pro přehled uvádím jejich krátký výčet (JSP, HTML, JavaScript, CSS). Před samotnou fází implementace bylo rovněž potřeba zvolit si vývojové prostředí. Liferay nabízí pro různé operační prostředí vlastní IDE. Jelikož již delší dobu pro vývoj většiny aplikací používám prostředí Eclipse, rozhodla jsem se využít a vyzkoušet Liferay IDE určený jako plugin pro ně. Jako aplikační server, který lze snadno nainstalovat a spouštět i lokálně na počítači jsem zvolila produkt Tomcat, který společnost, která se věnuje vývoji produktu Liferay, doporučuje. Jako databázi jsem zvolila PostgreSQL, která je jednou z používaných databází na fakultě a v níž jsou aktuálně ukládána veškerá data o úvazcích.

Pro vývoj portletů nabízí Liferay IDE možnost vytvořit projekt založený na MVC architektuře, další - mnou nevyužitou možností - je implementovat portlety ve frameworku JSF s využitím Liferay Faces Bridge nebo Vaadin.

V MVC architektuře mluvíme o několika oddělených částech implementace, které spolu vzájemně komunikují. Všechny tyto části jsou již uvedeny v samotném názvu $M--model$, $V--view$, $C--controller$. Model se stará především o obsluhu dat, jejich získávání z databáze, jejich úpravu či aktualizaci v databázi, a umožňuje dalším částem celého systému k těmto datům přistupovat. View, česky pohled, je součást, která se stará o obsluhu komunikace s uživatelem. Prezentuje mu informace, data a zároveň pomocí user interface od něj data získává. Controller, česky řadič, je další významnou částí, která se stará o veškerou logiku a funkcionalitu aplikace. Z jedné strany



Obrázek 3.1: Hierarchie třídy portletu MVCPortlet.

komunikuje s modelem, od které získává data, a které odesílá nová či aktualizovaná data, z druhé strany komunikuje s poslední komponentou view, která je viditelná koncovému uživateli Liferay portálu.

Pokud se vývojář rozhodne pro vývoj vlastního Liferay MVC portletu, bude jeho řadič rozšiřovat funkcionalitu třídy MVCPortlet. Ta sama o sobě vznikla rozšířením třídy GenericPortlet, jak je patrné ze schématu na obrázku 3.1.

3.0.2 Popis struktury každého MVC Liferay Portletu

Takto vypadá struktura portletu, kterou připraví při vytvoření nového plugin projektu samo Liferay IDE. Nyní krátce popíši jeho části.

Hlavním adresářem je složka *docroot*, která je kořenovým adresářem portletu. Obsahuje podsložku *css*, do které je možno vkládat soubory typu .css, které specifikují formát nebo styl zobrazovaných dat. Dále veškeré soubory a knihovny psané v jazyku JavaScript můžete umístit do složky *js*.

Velmi důležité informace uvnitř svého adresáře nese složka $WEB-INF$. Ve složce *lib* je možné umístit jakékoliv knihovny ve formátu JAR. Složka *tld* může obsahovat soubory knihoven vlastních značek (tagů). Velmi důležitými pro konfiguraci a správný chod portletu jsou soubory ve formátu .xml.

V tom prvním, pojmenovaném $liferay-display.xml$, jsou uchovávány informace, do jaké kategorie portlet zařadit. Podle této informace pak portál

```
portletName
├── docroot
│   ├── WEB-INF
│   │   ├── src
│   │   ├── lib
│   │   ├── tld
│   │   ├── liferay-display.xml
│   │   ├── liferay-plugin-package.properties
│   │   ├── liferay-portlet.xml
│   │   ├── portlet.xml
│   │   └── web.xml
│   ├── css
│   ├── js
│   ├── META-INF
│   ├── icon.png
│   └── view.jsp
└── build.xml
```

správně zařadí portlet v nabídce aplikací k přidání.

Soubor $liferay - portlet.xml$ shromažďuje data o souboru ikony portletu, o umístění souborů kaskádových stylů a javascriptových souborů, obsahuje jméno portletu a názvy rolí, které umožňují nastavit portletům přístupová oprávnění či omezení. Rovněž zde jsou umístěny odkazy na tyto role.

Uvnitř souboru *portlet.xml* lze nalézt, případně upravit data o názvu souboru, ve kterém je uložena hlavní java třída řadiče i s cestou k němu. Dále nese informace o zkráceném názvu portletu i klíčová slova pro jeho vyhledání.

Dokument *build.xml* obsahuje xml skript pro vytvoření JAR souboru celého projektu Ant - systému pro sestavoování projektu a správu závislostí na externích knihovnách.

Ve složce $WEB - INF/src$ je místo, kam lze umístit veškerý zdrojový kód psaný v jazyku Java, je dobré zde umístit i samotný řadič, který bude obstarávat řízení akcí a vykreslování portletu. Umístění řadiče je potřeba vložit do souboru *portlet.xml*.

3.1 Návod pro snadný vývoj portletů

3.1.1 Implementace Řadiče

Řadič je implementován v jazyce Java jako třída která rozšiřuje třídu MVCPortlet.

Pokud je vytvořen nový Liferay Plugin project v prostředí Liferay IDE for Eclipse je zapotřebí implementovat řadič, ten by měl obsahovat minimálně dvě metody. Jednu, která bude obsluhovat portlet během vykreslovací fáze

a druhou, která bude obsluhovat akci. Akčních metod je možno implementovat v rámci jednoho portletu více, každá musí mít jiné jméno. Aby toto fungovalo, je nutné definovat v pohledu jméno této funkce a předat správnou url tlačítku nebo elementu, který akci vyvolá. Dále je možné implementovat metodu *init*, která proběhne během iniciálního nahrání portletu. Standardně používaným názvem pro metodu vykreslovací fáze je $doView$, která má dva parametry - RenderRequest a RenderResponse. V případě implementace jedné akční třídy je standardně využíván název metody *processAction* se dvěma parametry ActionRequest a ActionResponse.

Pro uchování parametrů mezi akční a vykreslovací fází a opačně je možné využít Portlet.session, do které se velmi snadno ukládají proměnné různých typů a to i seznamového typu $List <>$. Pokud se ukládají do portlet.session data vlastních tříd, je zapotřebí, aby tyto třídy implementovaly rozhraní Serializable. Portlet session má možnost uložit data pro rozsah v rámci jednoho portletu, popřípadě celé aplikace, která získat data řadiči jiného portletu v rámci jedné aplikace.

Pro předání dat do pohledu je vhodné použít metodu

$$actionRequest.setAttribute(\text{“}názevProPřístup'', proměnná)$$

v případě vykreslovací metody je potřeba uložit atribut do $RenderRequest$.

Pro načtení dat z pohledu je dobré použít třídu $ParamUtil$ a její metody. Vstup je při načtení nutno explicitně přetypovat na stejný datový typ, ve kterém byl v pohledu uložen.

Pro přístup k modelu je možné volat libovolné funkce implementované v třídách Java. V případě využití mnou implementovaného kódu - např. pro přístup do databáze, je nutné importovat na začátku souboru cestu k souboru.

3.1.2 Implementace Pohledu

Pohled je realizován v souboru typu jsp. Jeho název může být libovolný, pokud je zanesen jako inicialní parametr do souboru $liferay - portlet.xml$. Standardní, Liferay IDE při vytvoření nového projektu vytvořený, je soubor s názvem *view.jsp* Pohledových souborů je možné v rámci jednoho portletu vytvořit více. Během akční fáze, například navštívením linku nebo stisknutím tlačítka, lze obsah portletu změnit na obsah jiného pohledového souboru. K získání adresy odkazu na takový soubor je potřeba vložit za klíčový element $< portlet : defineObjects/ >$ následující kód:

```
<portlet:actionURL var="actionPageURL" name="processAction">
       <portlet:param name="mvcPath" value="/edit.jsp" />
</portlet:actionURL>
```

Odkaz se poté nachází v proměnné actionPageURL a jméno funkce řadiče, které bude pro obsluhu funkce vyvoláno bude processAction.

Získání dat z řadiče, pokud byli předány, jak je popsáno v předchozí kapitole, lze realizovat pomocí metody

$$request.getAttribute(\text{“}názevParametru'');$$

. I zde je nutno získanou hodnotu explicitně přetypovat na správný datový typ před uložením do proměnné. Jejich předání do řadiče je například možné pomocí odeslání formuláře.

V rámci pohledového souboru je možné využít programovacích jazyků Java v rámci JSP, JavaScript a běžného HTML. Veškeré formátování je možné zajistit pomocí jazyka css v souboru $WEB - INF/css/main.css$

Vyvstane-li v aplikaci chyba, nebo je-li potřeba uživateli předat zprávu o úspěšném vykonání akce, např. přihlášení, je možné využít předpřipraveného mechanismu pro zobrazení

```
<liferay-ui:error message=‘‘Zpráva o chybě.’’>
<liferay-ui:success message=‘‘Zpráva o úspěchu.’’>
```

Chyba je vykreslena červeně, hlášení o úspěchu zelenou barvou. Zpravidla jsou vypisovány na začátku portletu, kde je uživatel snadno nalezne.

Formuláře lze velmi snadno vytvořit pomocí knihovny Liferay Alloy UI.

3.1.3 Implementace Modelu

Model je implementován množinou tříd Java, které realizují připojení k datům do databáze nebo datového skladu, obsluhují dotazování nad data. Zároveň je vhodné v této části implementovat třídy a metody, která data uloží do vhodných datových struktur a typů, ve kterých je řadič nebo pohled využije pro svou funkcionalitu. Klíčové jsou tedy návratové hodnoty metod, které umožní předání parametrů do řadiče, ve kterém byla příslušná metoda zavolána.

Jelikož se jedná o běžný způsob programování v jazyce Java, nebudu se dále o modelu rozepisovat.

Kapitola 4

Vytvořené portlety

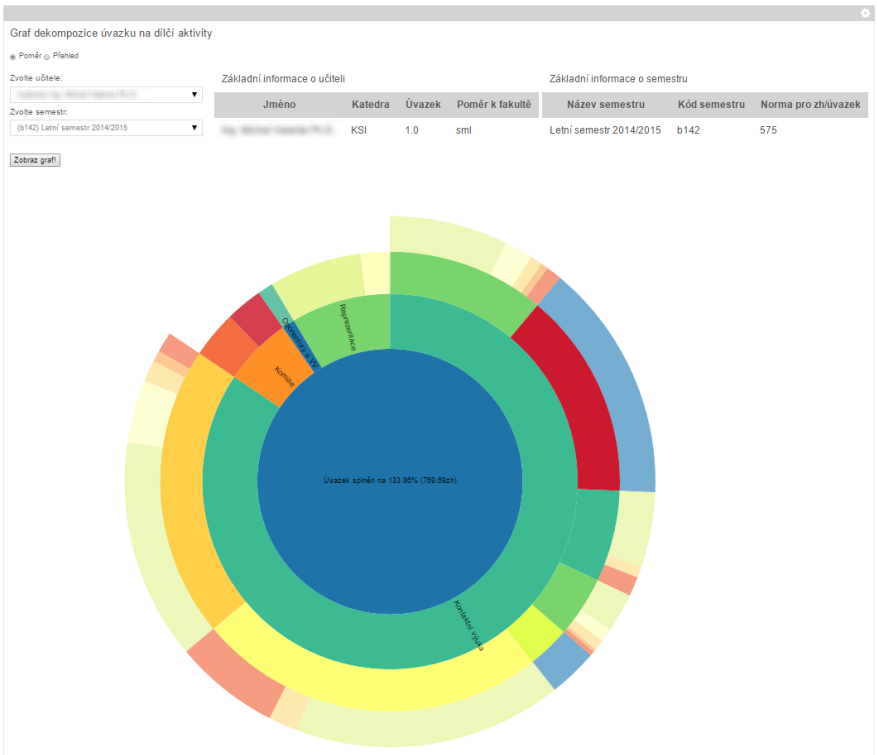
4.1 Portlet dekompozice úvazků na dílčí aktivity

První z vyvíjených portletů si kladl za cíl uživateli poskytnout informace o složení jeho úvazku, případně započítatelných hodin. Tuto informaci by měl poskytnout i osobě, která nezná strukturu dat úvazku a metody výpočtu započítatelných hodin jednotlivých položek. Každému zaměstnanci by umožnil nahlédnout do každé položky, která se na jeho úvazku, byť jen okrajově podílí. Jelikož je schéma součástí úvazku tvořeno stromovou strukturou, bylo potřeba nalézt vhodnou reprezentaci vizualizace, která by zachovala toto rozložení a umožnila procházení této stromové struktury. Jelikož strom nesoucí data o úvazků vytíženějších zaměstnanců mívá hloubku 4 vrstvy a někdy více jak 100 uzlů, bylo hledáno řešení, které by umožnilo během procházení dat se zanořovat a vynořovat mezi jednotlivými uzly v různých vrstvách. Toho bylo dosaženo implementací prvního portletu viz. 4.1.

4.1.1 Volba grafické reprezentace dat v rámci tohoto portletu

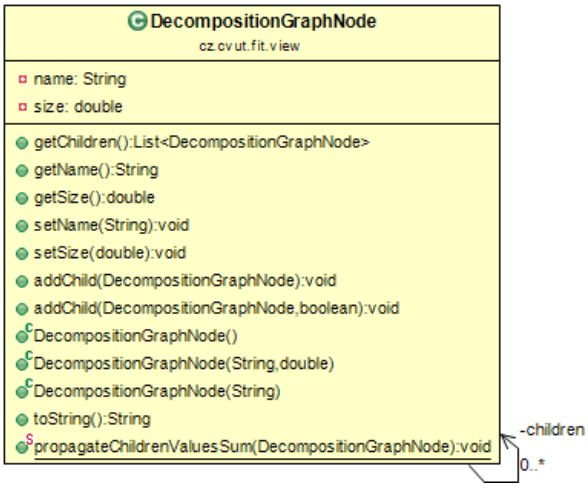
Při procházení různých dosud implementovaných grafů pomocí knihovny d3.js mě velmi zaujal graf nazvaný Sunburst [18]. Tento graf má tvar kruhu, který je dále členěn na mezikruží s postupně zvětšujícími se průměry, které odpovídají jednotlivým vrstvám stromu reprezentovaných dat. Uprostřed grafu se nalézá kruh, který znázorňuje kořen stromu. Každé mezikruží je dále členěno na menší oblasti připomínající dílky koláče. Ty odpovídají jednotlivým uzlům stromu nacházejícím se v příslušné vrstvě.

Tento statický graf sám o sobě by se pro většinu osob stal nečitelným, pokud by zůstal v jeho původní statické podobě. Každý úvazek v sobě nese mnoho informací a na ploše běžně používaných monitorů by bylo zobrazení veškerých informací včetně popisků a číselných statistik znepřehledňujícím až znečitelňujícím faktorem. Proto jsem přistoupila k doplnění další funkcionality grafu. Byla přidána možnost procházení stromové struktury grafu zanořová-



Obrázek 4.1: Screenshot portletu grafu dekompozice úvazků na dílčí aktivity.

ním a vynořováním do jednotlivých uzlů. Základní idea a matematický aparát pro přepočet informací nutných k překreslení grafu byly získány a následně upraveny a rozšířeny z grafu “Coffeewheel”[19]. Do grafu jsem přidala popisky, aby uživatel snadno zjistil, které dílky odpovídají které položce z úvazku. Jelikož jsou některé popisky dlouhé, došla jsem při procházení jednotlivých grafů úvazků k myšlence, zobrazovat tyto popisky jen u nejvyšších dvou vrstev vůči aktuálně procházené. Takto se graf stává snadno čitelným a uživatel se zorientuje v hierarchii úvazku. Možnost odkázat se do dalších vrstev kliknutím na příslušný dílek mezikruží zůstala zachována všem vrstvám grafu. Aby bylo možné vrátit se o úroveň výše, byly implementovány dva způsoby. Kliknutím do středního kruhu, na kterém je od procházení druhé vrstvy níže napsáno slůvko zpět, je možné dostat se o úroveň níže. Kliknutím přímo na slovo zpět je možné se dostat opět do kořene stromu. Jelikož toto ovládání portletu není pro všechny uživatele zřejmé, je informace o této funkcionalitě ovládání portletu uživatelům dostupná v online dokumentaci k portletům.



Obrázek 4.2: Class diagram třídy pro datovou reprezentaci koláčového grafu.

4.1.2 Datová reprezentace

Pro jednotnou datovou reprezentaci použitelnou i v jiných portletech využívajících koláčový graf (aktuálně portlet pohledu děkana, viz sekce 4.2) a snadné převádění do formátu JSON jsem navrhla třídu *DecompositionGraphNode* (viz Obr. 4.2). Tato třída byla navržena tak, aby přesně odpovídala struktuře JSON dat požadovaných implementovaným koláčovým grafem, a tudíž nebyla potřeba žádná dodatečná funkcionalita pro převod do formátu JSON, krom volání knihovny Jackson[20].

4.1.3 Parametry volby a způsobu zobrazení

Uživatel po iniciálním vykreslení portletu musí zvolit několik parametrů, které je potřeba shromáždit ke korektnímu zobrazení dat. Postačující a zároveň jednoznačně identifikující je volba osoby a semestru. Osobu, o níž jsou uchovány údaje v databázi, lze vybrat ze seznamu. Každá z nich je identifikována svým uživatelským jménem, podle něhož je seznam setříděn a pro čitelnost je přidáno celé jméno dané osoby. Semestr je vybíran z přehledu podle kódu a jeho celého jména. Zbylé parametry jsou přednastaveny defaultně a uživatel je může a nemusí měnit. O možnostech jejich nastavení a funkcionalitě, kterou ovlivňují se zmiňuji v následujících odstavcích.

Defaultně je nastaveno zobrazení grafu na volbu poměrového zobrazení, kdy každému dílku mezikruží je přiřazena jeho část na základě procentuálního zastoupení započítatelných hodin v rámci vrstvy. Někdy ovšem je toto procentuální zastoupení natolik malé (například až 0,1 %), že je dílek pro oko téměř přehlédnutelný a text lehce přetéká k sousedním. Proto byla implementována druhá volba pro uživatele, takzvaný “přehled”. Ten překreslí graf

do stavu, kdy všechny dílky získají rovnoměrnou velikost vůči počtu dílků v poslední vrstvě. Dílek v tomto případě nezohledňuje procentuální zastoupení dané položky, ale stává se čitelným a nabízí vizuálně lepší přehled o množství dílků v rámci procházené vrstvy. Toto přepínání uživatel realizuje výběrem v rámci uživatelského prostředí, které okamžitě reaguje a automaticky překreslí graf do požadované podoby.

4.1.4 Vzhled a popis komponent portletu

Hlavní součástí portletu je modifikovaný a o novou funkcionalitu doplněný graf Sunburst, který má kruhovitý charakter. Jak již bylo uvedeno, kořen stromu dat úvazku je reprezentován kruhem uprostřed, jednotlivé uzly a další vrstvy jsou viditelné v jednotlivých mezikružích, obklopujících středový kruh kořene.

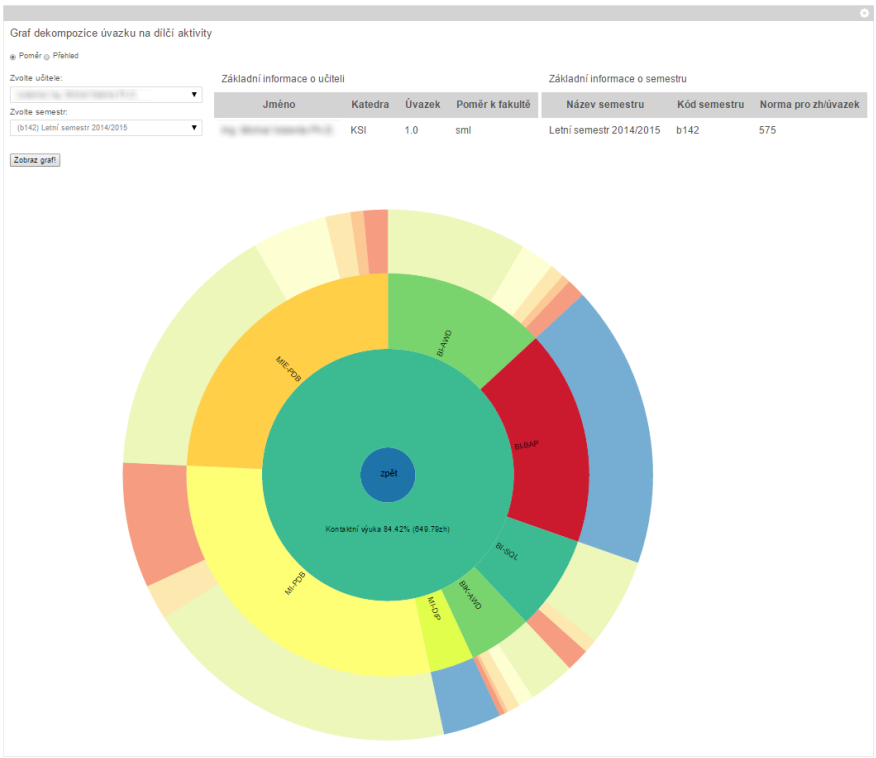
Pro každého uživatele je v kořeni zobrazen popisek, který informuje zaměstnance o naplnění jeho úvazku. Každý uživatel získá informaci o počtu započítatelných hodin a to vždy v uzlu, který je procházen. U lidí, jež mají v databázi uveden svůj úvazek, je v koření přidána informace v procentech o stavu plnění jejich úvazku vůči normě, která je přenásobená hodnotou jejich přiděleného úvazku. U osob bez úvazku procentuální hodnota v kořeni uvedena není. Zpravidla se toto děje u osob pracujících na dohodu o provedení práce, u doktorandů, školitelů a externistů, kteří jsou rovněž uvedeni v databázi úvazkostroje. Zobrazení této informace by bylo zavádějící. V nižších vrstvách je uvedeno v procházeném uzlu procentuální zastoupení této položky počítané vůči součtu započítatelných hodin získaných v rámci jednoho semestru uživatelem(tedy součtu z.h. uvedeném v kořeni stromu).

Jelikož jsou v databázi uvedené v každém semestru i osoby, které mají 0 započítatelných hodin, je pro ně uvedena informace, že pro danou osobu je v rámci zvoleného semestru počet započítatelných hodin nulový a končí vykreslením kořene stromu.

Portlet je doplněn pro uživatele dvěma tabulkami. Jednou, která nese informace o osobě, jež je vybrána k prozkoumání, konkrétně jsou zde uváděny údaje: Celé jméno včetně titulů, katedra, na které je uživatel zaměstnán, výše úvazku (nulová v případě, že úvazek v databázi uveden není) a poměr uživatele k fakultě. V druhé tabulce jsou uvedeny informace o zvoleném semestru: jeho názvu, kódu a především o normě počtu započítatelných hodin pro plný úvazek viz. 4.3

4.1.5 Způsob řešení anonymizace portletu pro běžného uživatele

Mnoho uzlů v tomto stromu nese informace, podle kterých by bylo velmi snadné odhalit, kterému ze zaměstnanců úvazek přísluší, například podle vyučovaných předmětů uvnitř kategorie kontaktní výuky. Informace o vyučujících jednotlivých předmětů jsou dostupné po přihlášení univerzitními přístu-



Obrázek 4.3: Screenshot dekompozičního grafu úvazku po zanoření do druhé vrstvy.

povými údaji jak studentům, tak zaměstnancům, například v rámci portálu timetable.fit.cvut.cz. Z tohoto důvodu by při anonymizaci muselo dojít kromě odstranění jména nebo uživatelského jména zaměstnance i k vymazání údajů o názvech všech předmětů. Zároveň se domnívám, že informace o dekompozici jiného zaměstnance by neměla být jinému běžnému zaměstnanci poskytnuta.

Toto mě vedlo k volbě jiného řešení, kde běžnému zaměstnanci je umožněno podívat se na dekompozici pouze vlasního úvazku. Uživatelům se zvýšenými oprávněními (úvazkářům, vedoucím kateder a výše postaveným osobám managementu a řízení fakulty) bude nabídnuta možnost prohlédnout si složení úvazků osob, prozatím v rámci celé fakulty. Velmi snadno bude možné vytvořit další úroveň oprávnění, např. pro vedoucí kateder tak, aby měli zpřístupněné úvazky pouze zaměstnanců své katedry. Tato poslední část nebyla dosud implementována, neboť stávající hierarchie oprávnění v aplikaci úvazkostroje má dvě role přístupových oprávnění - admin s plnými přístupovými právy a běžný uživatel s omezenými právy přístupu.

Obrázek 4.4: Screenshot portletu pohledu děkana s katedrami v první úrovni grafu.

4.2 Portlet pohledu děkana na úvazky celé fakulty

4.2.1 Volba grafické reprezentace dat

V rámci vývoje, za účelem prezentace nabytých znalostí, bylo využito již naimplementované vizualizační komponenty prvního portletu, která vykresluje graf Sunburst v mnou pozměněné podobě. K jeho znovuvyužití došlo kvůli nalezení dalšího zajímavého pohledu na data v databázi úvazků, která se hodila pro zobrazení v této podobě. V rámci tohoto portletu byla implementována odlišná struktura řadiče - především metoda pro získávání a transformaci dat do stromové struktury a formátu čitelného na vstupu tohoto konkrétního grafu. Shodné části kódu v těchto vrstvách architektury, např. metody pro dotazování do databáze, třídy nesoucí informace o osobách, semestrech a úvazcích,

byly poskytnuty v rámci sdíleného kódu jako balíčky.

4.2.2 Vzhled a popis komponent portletu

Druhý mnou zpracovaný portlet je určen výhradně do rukou děkana a osob s vysokými oprávněními k přístupu k údajům o úvazcích. O přidělení tohoto oprávnění by měla rozhodnout fakultou delegovaná osoba, která řeší otázku přístupu k osobním údajům. Tento portlet totiž sdružuje do jednoho grafu úvazky v databázi pro daný semestr uvedených osob. Nad nimi jsou postaveny ještě 3 nové vrstvy, které tvoří stromovou strukturu dat. V kořeni stromu se nachází celá fakulta, která obsahuje všechny nenulová data z databáze a součet všech započítatelných hodin. To znamená, že osoba, jež má 0 započítatelných hodin v rámci semestru, není do grafu zahrnuta.

Dvě zbývající vrstvy zařazují jednotlivé osoby do kategorií katedra a poměr k fakultě. Jelikož volba pořadí těchto dvou vrstev může poskytnout jiný pohled na data a nabízí další zužitkovatelné informace, byla při implementaci zahrnuta možnost, aby si uživatel zvolil, v jakém pořadí tyto vrstvy zařadí. Zvolí-li si parametr Katedra, budou v druhé vrstvě, hned po kořeni roztříděna data nejprve podle kateder a následně podle poměru k fakultě viz. 4.4. Při výběru varianty Pracovní poměr, bude ve vrstvě potomků kořene nejprve vizualizován přehled pracovních poměrů a až potom kateder. Potomky druhé z těchto vrstev jsou korektně zařazené osoby uvedené pod svým username a ještě jedna vrstva, která nabízí pohled na Komise, Oponentury a VV a Reprezentace. Zbývající vrstvy úvazků nebyly pro přehlednost grafu a s ohledem na možnost nahlédnout do konkrétního grafu úvazku osoby v rámci prvního, v této kapitole již zmíněného, portletu, zahrnuty.

Jelikož u některých osob je hodnota v sloupci katedra prázdná, jsou tyto osoby sdruženy v dílku grafu, který nenese žádný popisek.

Ke grafu přísluší ještě tabulka nesoucí informace o zvoleném semestru a normě započítatelných hodin pro příslušný semestr.

4.2.3 Parametry volby a způsobu zobrazení

Povinným parametrem, který si musí uživatel zvolit před vykreslením grafu je semestr. V seznamu se mu zobrazí všechny v databázi dostupné semestry se svým kódem a celým jménem. Volitelnými a defaultně přednastavenými parametry jsou volba velikosti dílků grafu dle procentuálního zastoupení v dané vrstvě (počítáno jako součet započítatelných hodin všech poduzlů vybraného dílku dělený součtem započítatelných hodin všech dílků v aktuálně prohlížené vrstvě). Druhou možností je u předchozího portletu popsaná volba zobrazení poměrem, která zvyšuje čitelnost údajů v grafu, obsahuje-li libovolná vrsta údaj s nižším než 1% zastoupením vůči předchozí vrstvě (vrstvě rodiče). Dále zde je pak možné zvolit pořadí obsahu druhé a třetí vrstvy, jak bylo uvedeno v předchozím odstavci. Graf umožňuje nejen procházení uzlů směrem k listům

Obrázek 4.5: Screenshot portletu pohledu děkana s typy pracovního poměru v první úrovni grafu.

stromové struktury dat, ale stejně jako u předchozího portletu byla implementována možnost návratu do vrstev předchůdců pomocí středního kruhu a klíčového slova zpět viz. 4.5.

4.2.4 Způsob řešení anonymizace portletu pro běžného uživatele

Otázku anonymizace v případě tohoto portletu nebylo třeba zvláštně řešit. Daný portlet je zobrazitelný pouze uživateli, kterému bylo přiděleno speciální oprávnění.

V zadání byl uveden požadavek na existenci portletu, který by umožnil srovnávat jednotlivé úvazky mezi sebou. A to i v anonymizované podobě, aby k nim mohl přistupovat běžný uživatel. Tomuto požadavku bylo vyhověno v

dalších dvou portletech.

4.3 Portlet srovnání úvazků napříč semestry

Třetí portlet obsahuje vizualizaci dat jednotlivých uživatelů napříč jednotlivými semestry uvedenými v databázi. Tato charakteristika umožňuje uživateli nahlédnout, jak se naplnění jeho úvazků v rámci po sobě jdoucích semestrů mění. Zároveň mu poskytuje informaci o tom, jak si naplnění jeho úvazku vede vůči normě počtu započítatelných hodin pro daný semestr.

4.3.1 Volba grafické reprezentace dat

Pro vykreslení uvedených dat a specifických charakteristik již není předchozí typ grafu vhodný. Proto byl hledán jiný, pro tato data vyhovující, typ grafu. Jedná se o srovnání číselných dat bez nutnosti dekompozice jednotlivých částí do více úrovní. K tomuto účelu se výborně hodí sloupcový graf, který v rámci dvoudimenzionálního prostoru zobrazuje z databáze získaná data. Jako vzorový byl využit graf [21]. Aby bylo možné stanovit, zda uživatel naplňuje dostatečně svůj úvazek, bylo potřeba zanést do grafu hraniční křivku, která ukazuje, kolik hodin v daném semestru bylo od daného uživatele požadováno normou. K tomuto účelu byl využit spojnicový graf.

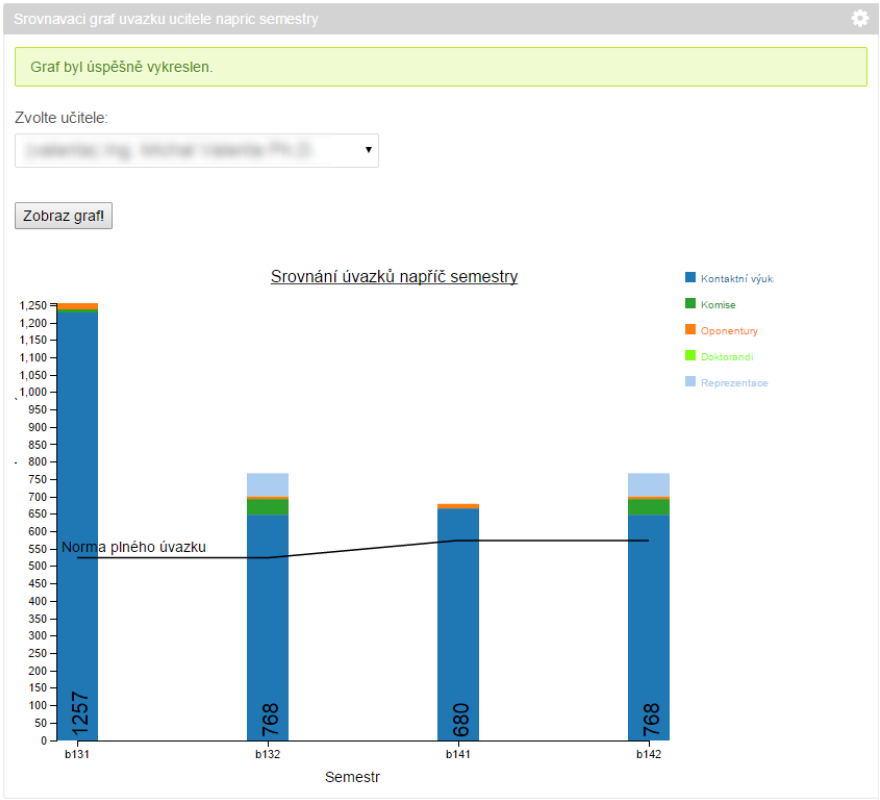
4.3.2 Vzhled a popis komponent portletu

Uprostřed porletu je po zadání parametrů vykreslen dvoudimenzionální graf. Osu X v tomto grafu tvoří jednotlivé semestry, které jsou označeny svými kódy a jsou řazeny za sebe dle časové souslednosti. Osu Y tvoří číselná stupnice, na kterou jsou vynášeny hodnoty součtů započítatelných hodin.

Každý sloupec je ještě dále rozdělen na další podčásti(obdelníky), které jsou barevně odlišeny a zobrazují dekompozici úvazku na jeho součásti do první úrovně. Uvnitř grafického zobrazení tohoto portletu je zahrnut i spojnicový typ grafu, který zobrazuje hladinu normy započítatelných hodin plného úvazku. Ke grafu je přidružena legenda viz. 4.6

Číselná hodnota součtu z.h. úvazku v rámci jednoho semestru je uvedena uvnitř každého sloupce, těsně nad osou X. Hodnoty z.h. jednotlivých dílků jsou zobrazovány v malé kontextové nápovědě, která je vyvolána najetím myší nad příslušný obdelník. Nápověda zahrnuje název dané kategorie a číselnou hodnotu udávající počet započítatelných hodin příslušné kategorie. Kategorie s nulovou hodnotou nejsou v grafu zahrnuty.

Při přidání nového semestru do databáze je automaticky přepočítána šířka sloupců a mezera mezi nimi tak, aby byly vykresleny rovnoměrně.



Obrázek 4.6: Screenshot srovnávacího portletu napříč semestry s hraniční křivkou normy na plný úvazek.

4.3.3 Parametry volby a způsobu zobrazení

Jediným povinným parametrem tohoto portletu je výběr osoby ze seznamu uživatelů, kteří jsou získáni z úvazkové databáze napříč všemi semestry.

4.3.4 Způsob řešení anonymizace portletu pro běžného uživatele

Uživatel se zvýšenými oprávněními má možnost vybrat si libovolnou osobu ze seznamu uživatelů, běžný uživatel má přístup pouze k údajům o vlastní osobě, ostatní zaměstnanci se mu v rámci seznamu pro výběr nezobrazují. Bezpečnost osobních dat při přihlášení běžného uživatele je zaručena i v rámci řadiče, kde je při volání funkce, získávající data z databáze, předáván parametr username přihlášeného uživatele. V opačném případě je vyvolána chyba.

4.4 Portlet porovnání úvazků zaměstnanců napříč katedrou

Poslední z implementovaných portletů by měl sloužit zejména vedoucím kateder a úvazkářům, ale zajímavé charakteristiky a statistické údaje může nabídnout i běžnému uživateli, kterému zakryje součást informací, které by vedly k prozrazení osobních údajů druhých osob.

Tento portlet umožňuje zobrazit úvazky jednotlivých zaměstnanců zaměstnaných na stejné katedře. V rámci tohoto grafu může snadno získat představu o výkonu svém i svých kolegů pracujících na stejném pracovišti viz.4.7 .

4.4.1 Volba grafické reprezentace dat

Pro vizualizaci těchto dat je možné využít již jednou vytvořený, lehce modifikovaný sloupcový dvourozměrný graf s jednou hraniční spojnicovou křivkou.

4.4.2 Parametry volby a způsobu zobrazení

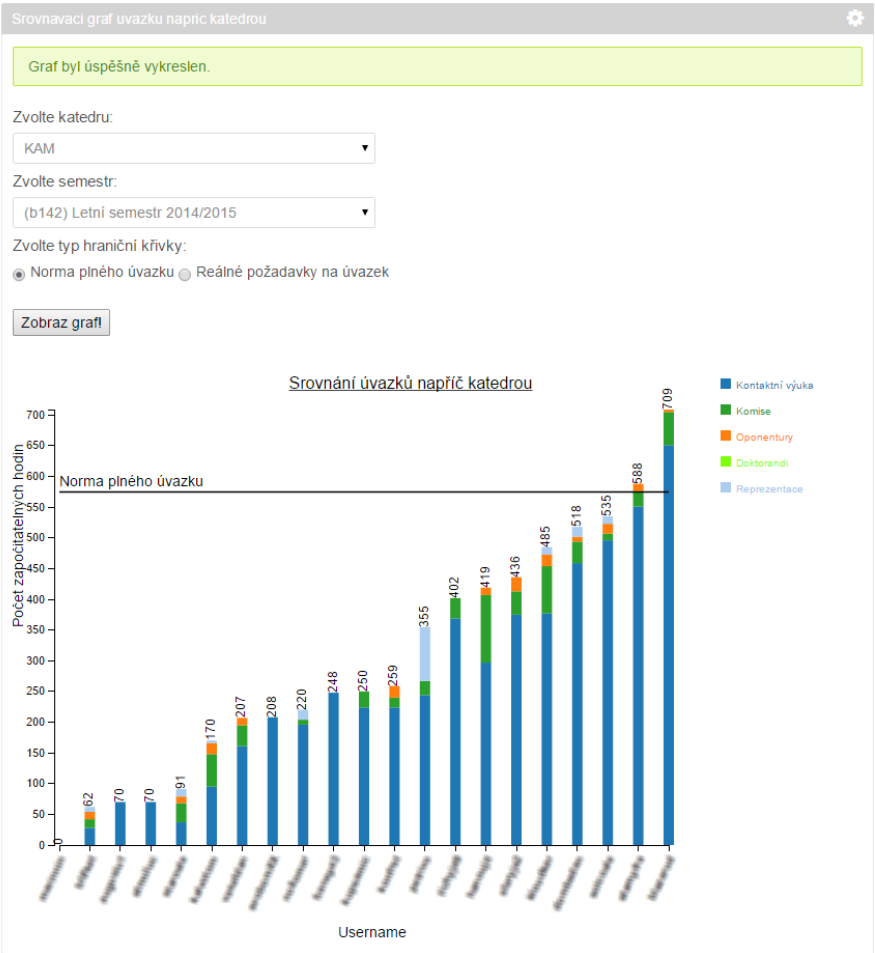
Pro vykreslení grafu v rámci tohoto portletu je potřeba, aby uživatel zvolil dva parametry. Prvním z nich je semestr, o jehož údaje se zajímá a druhým je katedra, pro kterou mají být data z databáze shromážděna a vykreslena. Jako volitelný parametr si uživatel může zvolit typ hraniční křivky. Po vykreslení grafu jsou ve sloupcích zobrazená data setřízena zleva od nejnižšího součtu započítatelných hodin jednotlivce po nejvyšší v rámci zvolené katedry.

4.4.3 Vzhled a popis komponent portletu

Osu X tohoto grafu tvoří jednotlivé osoby katedry, které lze identifikovat z popisků osy, ve kterých je uvedeno uživatelské jméno zaměstnance. Osu Y tvoří hodnoty počtu započítatelných hodin.

I tento graf zobrazuje pomocí spojnicového typu grafu hranici naplnění plného úvazku pro zvolený semestr. V rámci tohoto semestru byla implementována možnost zobrazit tuto křivku jako hodnotu normy plného úvazku přenásobené reálnou hodnotou úvazku. Bohužel u většiny osob v databázi je uvedena hodnota nulová nebo zcela schází, což ubírá na přehlednosti grafu a vlastním uživatelům může přijít nepřehledný, ba dokonce chybný. Zde pro zdokumentování, jak portlet s takto vypočtenou křivkou vypadá uvádím screenshot, který zachycuje reálný požadavek na započítatelné hodiny vzhledem k přidělenému úvazku a jeho plnění ve formě sloupcového grafu.

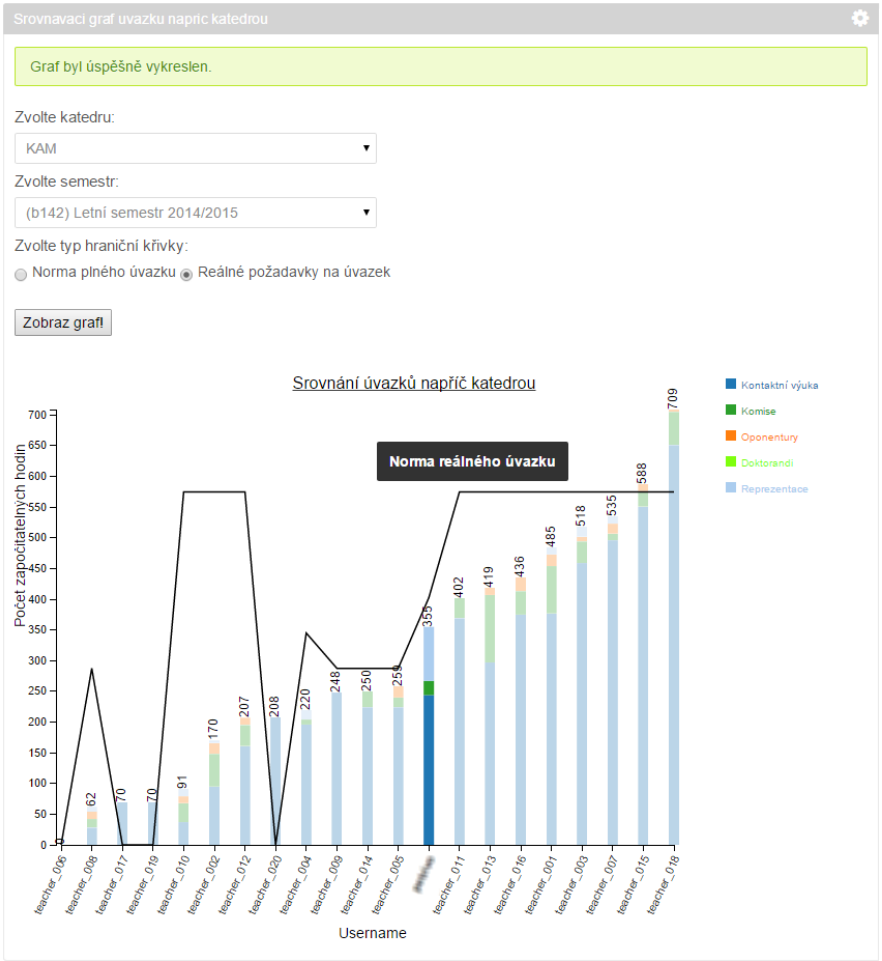
Jako barevnost tohoto grafu byla zvolena standartní, v knihovně D3.js již obsaženou desetistupňovou barevnou škálu. Ta je pestrá a díky jejímu použití lze velmi snadno od sebe odlišit jednotlivé části úvazku. Pro čitelnost grafu bylo i u tohoto portletu ponecháno bílé pozadí. Veškeré osy a popisky byly proto vykresleny černou barvou.



Obrázek 4.7: Screenshot srovnávacího portletu napříč katedrou v neanonymizované podobě s hraniční křivkou normy plného úvazku.

4.4.4 Způsob řešení anonymizace portletu pro běžného uživatele

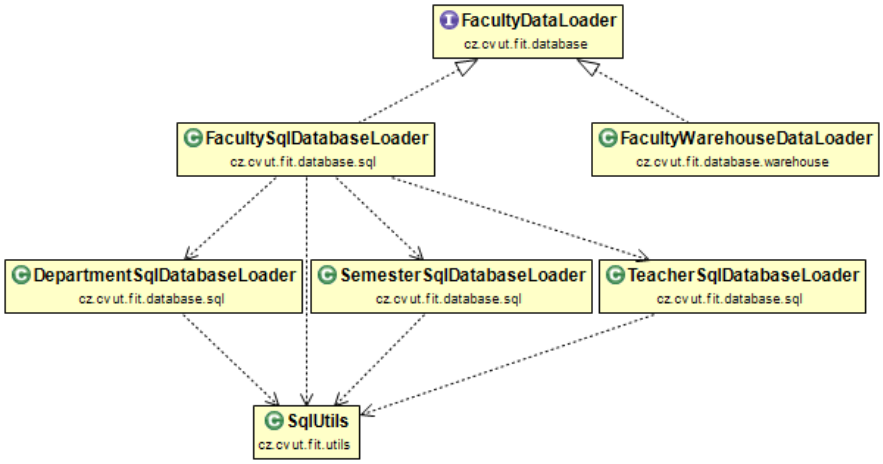
Anonymizace tohoto portletu byla implementována následovně. Aby se běžný pracovník mohl porovnat s ostatními pracovníky katedry, ale nebyl schopný ostatní pracovníky identifikovat, byly již v procesu tvorby dat pro graf nahrazeny uživatelská jména ostatních pracovníků na tvar “teacher_” a jednoznačné trojmístné číselné ID. Přihlášený uživatel své uživatelské jméno vidí v neanonymizovaném tvaru. Pro zvětšení přehlednosti, byla zvýšena průhlednost barev pro vykreslení všech vykreslených sloupců ostatních uživatelů. Samotný uživatel se tak velmi snadno v grafu najde viz. 4.8. Běžný uživatel smí nahlížet pouze do grafů své katedry.



Obrázek 4.8: Screenshot srovnávacího portletu napříč katedrou v anonymizované podobě s hraniční křivkou reálného požadavku na úvazek.

4.5 Tvorba barevných škál pro grafy

Pro každý graf byla zvolena jiná barevná škála, která dokázala vizuálně od sebe oddělit vzájemně odlišná data. V případě sloupcových grafů byla použita 5 stupňová barevná škála s různými barvami. V případě grafů typu sunburst byly použity 2 rozličné škály. Jedna s 11 různými barvami o stejném odstínu pro pohled děkana a druhá, ve které bylo vyhověno požadavku na volbu barevné škály pro barvoslepé. Zde byl pro volbu barev využit generátor barev [14] a z něho následně vyrobena barevná škála.



Obrázek 4.9: Class diagram načítání dat z databáze či datového skladu

4.6 Načítání dat pro prezentaci

V zadání práce je, mimo jiné, požadavek na možnost napojení portletů na datový sklad, který ale v průběhu vypracování mé práce nebyl k dispozici. Data jsou proto načítána z PostgreSQL databáze. Pro snadné budoucí napojení na datový sklad jsem implementovala třídy pro načítání dat tak, aby vyhovovaly návrhovému vzoru **Adapter**[22], jenž dává možnost snadné výměny databázového stroje bez změny rozhraní pro portlety (viz Obr. 4.9). Veškerý přístup k potřebným datům je specifikován rozhraním *FacultyDataLoader*, které pak implementují jednotlivé třídy pro přístup ke konkrétním druhům uložení fakultních dat.

Vzhledem k již zmiňované neexistenci řešení pomocí datového skladu jsem zatím implementovala pouze část pro načítání dat z PostgreSQL databáze (na obrázku 4.9) třída *FacultySqlDatabaseLoader*). Pro větší přehlednost kódu jsem jednotlivé logické celky načítání z databáze rozdělila do samostatných tříd pro načítání informací souvisejících s katedrou, semestrem a vyučujícími (postupně třídy *DepartmentSqlDatabaseLoader*, *SemesterSqlDatabaseLoader*, *TeacherSqlDatabaseLoader*). Samotná třída *FacultySqlDatabaseLoader* pak slouží především jako sjednocující rozhraní k těmto třídám podle návrhového vzoru **Facade**[22].

Samotné použití načítání pomocí některé implementace *FacultyDataLoader* je pak inicializováno zvlášť pro každý jeden portlet v rámci jeho *Controlleru*, v přetížení metodě *init*(). Zde je dosazena konkrétní implementace, aktuálně tedy *FacultySqlDatabaseLoader*.

Ve chvíli, kdy bude zprovozněn datový sklad s fakultními daty, stačí implemetovat jednotlivé funkce ve třídě *FacultyWarehouseDataLoader*, kterou jsem připravila jako kostru pro budoucí nasazení, a změnit inicializaci

FacultyDatabaseLoaderu v příslušných použitých portletech.

4.7 Řešené problémy s daty uloženými v databázi

Veškerá data o úvazcích jsou nyní na fakultě uchovávána v PostgreSQL databázi na serveru db-new.fit.cvut.cz. Při zadávání mé diplomové práce bylo předpokládano zpracování bakalářské práce kolegou Milanem Reschem, který měl vytvořit návrh datového skladu a data z databáze úvazků do něj importovat. Do dnešního dne však tato úloha nebyla zpracována a data z úvazkostroje se v datových skladech fakulty nenacházejí. Z tohoto důvodu bylo přistoupeno k řešení, kdy byla data čerpána výhradně z fakultní databáze.

V zadání uvedený požadavek na snadnou rozšířitelnost na datový sklad bylo implementováno rozhraní, které po dokončení úlohy transferu dat do datového skladu, umožní velmi snadné propojení mezi funkcionalitou portletů a daty, která budou v datovém skladu fakulty uložena.

Data uvedená v tabulkách učitel nemají jednotnou formu. u některých položek obsahuje sloupec jméno celé jméno učitele i včetně titulů, u jiných řádků obsahuje tento sloupec jen křestní jméno učitele a zbylé části jména získat a do řetězce ve správném pořadí spojit ze sloupců titul před, jméno, příjmení, titul za. Jelikož je celé jméno včetně titulů portlety využívano, byla implementována metoda, která data z databáze převede do jednotného formátu pro správnou funkcionalitu portletů. Funkce portletů jsou vytvořeny tak, aby po sjednocení formátu tabulek učitel v jednotlivých semestrech do libovolného z nyní společně uvnitř jedné tabulky používaných, nebyla narušena funkčnost portletů.

Během fáze implementace byly nalezeny dvě chyby v implementaci databázových funkcí, u nichž byly nalezeny vhodné úpravy, po jejichž provedení, přestaly korektně položené dotazy způsobovat neočekávané pády a zobrazovat neočekávaná chybová hlášení aplikací. Tyto úpravy byly poskytnuty správci databáze, který opravy aplikoval v online databázi úvazků, ke které jsem měla sama přidělena pouze omezená přístupová práva.

Databáze obsahuje v rámci jednotlivých semestrů mnoho osob, jež nemají žádný úvazek a často ani žádné započitatelné hodiny. Tato situace se může po zpracování bakalářské práce kolegy Milana Retsche změnit.

Kapitola 5

Testování

5.1 Uživatelské testování

V průběhu vývoje portletů byla funkčnost portletů opakovaně podrobována testování několika zaměstnanci fakulty s různými úrovněmi znalostí práce na PC a znalostí skladby úvazků a metodiky vyhodnocování úvazků.

Pro testování bylo vytvořeno několik testovacích účtů s omezenými i plnými přístupovými právy. Osoby byly za mé účasti požádány o vyzkoušení funkcionality vyvíjených portletů.

Testování probíhalo na počítači, kde byl software implementován.

Cílem tohoto testování bylo odhalit skryté chyby a získat od uživatelů nápady na vylepšení, doplnění popisu, zlepšení čitelnosti informací, případně funkcionality.

Během testování bylo vypozorováno, že ovládání dekompozičního sunburst grafu není pro každého uživatele intuitivní bez jakékoliv nápovědy či návodu k použití. Z tohoto důvodu bylo přistoupeno k rozhodnutí poskytnout ke každému portletu stručnou dokumentaci, která uživatele obeznámí se způsobem ovládání a mechanismem interakce s portletem.

Při testování částečně barvoslepou osobou byl vznesen požadavek na vytvoření barevné škály pro dekompoziční graf, který by této osobě umožnil vnímat jednotlivé dílky grafu a odlišovat vzájemně různá data i díky barvě, jak mohou ostatní. Této připomínce bylo vyhověno. Portlet dekompozice je tedy nyní prezentován a uložen s barevnou škálou pro barvoslepé, která se běžnému oku může zdát velmi pestrá.

Na základě vznesení požadavku na přidání informace o reálném plnění přiděleného úvazku bylo do grafu porovnávání úvazků napříč katedrou naimplementováno přepínání mezi hraniční křivkou normy a reálnou - očekávanou - hodnotou plnění úvazku vzhledem k přidělenému úvazku. Do tohoto grafu byly rovněž přidány popisky s číselnou hodnotou součtu započitatelných hodin všech částí úvazku jednotlivce, aby bylo snadnější ji stanovit.

Během posledního testování vzešel i nápad na vytvoření záměny dvou vrstev v portletu určenému děkanovi. Zde bylo požadováno, aby šlo na základě volby v přepínači pořadí dvou vrstev zaměnit a data a informace o součtech z.h. poduzlů byla zaktualizována na správnou hodnotu. I tomuto požadavku bylo vyhověno a příslušná funkcionalita byla vytvořena.

Závěr

Shrnutí

Cílem této práce bylo zmapovat možnosti tvorby interaktivních panelů v portálu Liferay, navrhnout a zdokumentovat postup, který umožní snadno vytvářet uživatelské dashboardy zobrazující informace a data z fakultních databází a datových skladů. Všechny tyto cíle jsem beze zbytku splnila. V práci jsem popsala snadný postup tvorby uživatelských dashboardů v portálu Liferay a demonstrovala získané znalosti na vývoji čtyř portletů, které byly vyvinuty za účelem jejich nasazení pro portál úvazkostroje. Jedná se o portlet pro dekompozici úvazku na dílčí aktivity, portlet pohledu děkana na úvazky celé fakulty, portlet srovnání úvazků jednoho zaměstnance napříč semestry a portlet pro porovnání úvazků zaměstnanců v rámci zvolené katedry.

Všechny mnou vyvinuté portlety používají jednotné rozhraní pro přístup k databázi, které umožňuje jednoduchou výměnou použité implementace změnit použitý databázový stroj či datový sklad. Zároveň byla vyvinuta implementace pro získávání dat z PostgreSQL databáze, jejíž obraz mi byl poskytnut pro účely vývoje a testování mého nasazení.

Navazující práce

Zřejmým pokračováním mé práce je nahrání připravených portletů nejprve na testovací server fakulty a následně do fakultního portálu úvazkostroje, což proběhne po domluvě se správci liferay portálu, v nejkratším možném termínu.

Možnosti navázání na tuto práci

Po návrhu způsobu uložení a dokončení transferu dat úvazku do datového skladu bude potřeba vytvořit implementaci datového rozhraní pro získávání

dat z něj, což však závisí na projektu, který dle mého nejlepšího vědomí zatím nebyl dokončen.

Potřeba by bylo vyřešit otázku sjednocení formátu dat v databázi úvazků do jednotné podoby. Zároveň by bylo dobré zabývat se otázkou, zda zařazovat do databáze osoby s nulovým počtem z.h. a bez úvazků.

Otázkou, kterou by bylo dobré se zabývat je vytvoření více úrovní přístupových oprávnění a rozhodnout o tom, jaká budou mít tato oprávnění pravomoce pro přístup k datům o úvazcích.

Zajímavé by rovněž bylo prozkoumat možnosti zobrazení statistik úvazku na displejích s menším rozlišením - pro mobilní platformu.

Další možností je tvorba dalších portletů pro získání nových statistik a informací z úvazkové databáze. Například by mohl být vytvořen graf pro srovnání počtu přednášek, cvičení a paralelek jednotlivých předmětů napříč semestry, v nichž byli vyučovány.

Zajímavou informací pro monitorování výkonu by bylo zahrnutí dat o počtu obhájených prací vedoucího, případně oponenta do databáze úvazků.

Literatura

[1] Dokumentace úvazkostroje. Dostupné z: https://rozvoj.fit.cvut.cz/uvazkostroj/dokumentace

[2] Úvazkostroj. Dostupné z: https://rozvoj.fit.cvut.cz/uvazkostroj/

[3] Němeček, J.: *Systém pro hodnocení pedagogického výkonu – verze 3.0 – metodika, datové struktury a datové zdroje.* Bakalářská práce, Praha: ČVUT, FIT, Katedra softwarového inženýrství, Praha, 2014.

[4] Wikipedia, Model-view-controller. Dostupné z: http://cs.wikipedia.org/wiki/Model-view-controller

[5] JSRs: Java Specification Requests, JSR 286: Portlet Specification. Dostupné z: https://jcp.org/ja/jsr/detail?id=286

[6] Liferay Inc. Liferay Portal. Dostupné z: http://www.liferay.com/downloads/liferay-portal/available-releases

[7] Tutorial to liferay Portlet Anatomy. Dostupné z: http://www.liferay.com/documentation/liferay-portal/6.1/development/-/ai/anatomy-of-a-portl-4

[8] JSRs: Java Specification Requests, JSR 168: Portlet Specification. Dostupné z: https://jcp.org/ja/jsr/detail?id=168

[9] Myslivec, J.; Skalská, H.: Vizualizace vícerozměrných dat symbolovými grafy. *E & M Ekonomie a Management*, ročník 12, č. 4, 2010: s. 114–129.

[10] The Champions Ring. Dostupné z: http://datavisualization.ch/showcases/the-champions-ring/

[11] Visualizing Obama's economic stimulus plan. Dostupné z: http://datavisualization.ch/showcases/visualizing-obamas-economic-stimulus-plan/

[12] D3 gallery. Dostupné z: https://github.com/mbostock/d3/wiki/Gallery

[13] Kosslyn, S. M.: Understanding charts and graphs. *Applied cognitive psychology*, ročník 3, č. 3, 1989: s. 185–225.

[14] Color Brewer. Dostupné z: http://colorbrewer2.org

[15] How We Created Color Scales. Dostupné z: http://datavisualization.ch/inside/how-we-created-color-scales/

[16] Data visualization tools list. Dostupné z: http://selection.datavisualization.ch/

[17] Murray, S.: *Interactive data visualization for the Web*. "O'Reilly Media, Inc.", 2013.

[18] Sunburst. Dostupné z: http://bl.ocks.org/mbostock/4063423

[19] Coffeewheel graph. Dostupné z: http://www.jasondavies.com/coffee-wheel/

[20] Jackson Java JSON-processor. Dostupné z: http://jackson.codehaus.org/

[21] Bar Chart. Dostupné z: http://bl.ocks.org/mbostock/3886208

[22] Gamma, E.; Helm, R.; Johnson, R.; aj.: *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

Příloha **A**

Seznam použitých zkratek

D3.js Data Driven Documents

DSL Domain specific language

GUI Graphical user interface

JSON JavaScript Object Notation

JSP Java Server Pages

MVC Model View Controller

XML Extensible markup language

z.h. započítatelné hodiny

Příloha **B**

Obsah přiloženého CD

```
/
├── src
│   ├── impl ..............................zdrojové kódy implementace
│   └── thesis ...................zdrojová forma práce ve formátu LaTeX
└── text ..................................................text práce
    └── thesis.pdf .......................text práce ve formátu PDF
```