Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Aplikace pro vizualizaci funkcionalit systému IS/STAG

Plzeň 2016 Petra Volenová

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 3. května 2016

Petra Volenová

Poděkování

Ráda bych poděkovala Ing. Lukáši Valentovi za vedení mé bakalářské práce, věcné připomínky a cenné rady, které mi pomohly tuto práci vypracovat. Také bych chtěla poděkovat Ing. Martinu Zímovi, Ph.D. za poskytnuté rady.

Abstract

This thesis is focused on the development of an application that provides visualization of functionalities of the information system IS/STAG with all its features and dependencies. The theoretical part deals with the research of relationship between the functionalities and business processes. There are also examined technologies that are suitable for creating the application with regard to its integration into the information system IS/STAG. The practical part describes the implementation of the selected solution and the application deployment on the information system portal.

Abstrakt

Tato práce je zaměřena na vývoj aplikace, která umožní přehledné zobrazení funkcionalit informačního systému IS/STAG se všemi vlastnostmi a závislostmi. Teoretická část se zabývá průzkumem vztahu mezi funkcionalitami a business procesy. Dále jsou zde zkoumány technologie, které jsou vhodné pro vytvoření aplikace s ohledem na její začlenění do informačního systému IS/STAG. Praktická část popisuje implementaci zvoleného řešení a postup nasazení aplikace na portál informačního systému.

Obsah

1 Úvod

Funkcionality informačního systému IS/STAG, dále jen funkcionality, popisují činnosti jeho uživatelů. Tyto činnosti jsou vykonávány během celého akademického roku a některé z nich se navzájem podmiňují. Například studijní referentka nejprve zadává nové uchazeče do systému a zapisuje je do ročníku. Dále pak jejich studium spravuje a vydává jim o něm potvrzení. Nakonec studentům studium uzavírá a tiskne diplomy. Tyto činnosti na sebe navazují a jsou provázány s dalšími povinnostmi jiných uživatelů. Příkladem může být rozhodnutí o přijetí uchazečů studijním proděkanem poté, co je studijní referentka zadá do systému.

Pro uživatele informačního systému IS/STAG je důležité, aby věděli, jakou činnost mají, v jakou dobu, na jakém místě a jakým způsobem vykonávat. A protože tyto činnosti nejsou nikde přehledně zobrazeny, je vhodné, aby vznikla aplikace, díky které se uživatelé dozví více informací o svých povinnostech.

Definice funkcionality se velmi podobá business procesu, a proto se první část práce zabývá studiem standardů a existujících nástrojů pro modelování business procesů. Tyto procesy jsou srovnávány s funkcionalitami a je zde zvažována možnost jejich vizualizace pomocí některého ze zkoumaných standardů.

V následující části jsou zkoumány technologie, pomocí kterých bude možné vytvořit nástroj pro evidenci a vizualizaci funkcionalit. Výběr technologií je ovlivněn tím, že aplikace bude součástí informačního systému IS/STAG.

Poté je popsána implementace aplikace s využitím prozkoumaných technologií. Tyto informace zahrnují vznik tabulek v databázi i popis struktury aplikace. Na závěr je zmíněn postup nasazení aplikace do portálu informačního systému IS/STAG, v rámci kterého bude aplikace provozována.

2 Zadání

Tato kapitola obecně specifikuje klíčové požadavky od zadavatele. Popisuje nezbytné části, které musí aplikace obsahovat.

2.1 Požadavky zadavatele

Zadavatel požaduje vytvoření dvou modulů. Jeden modul bude umožňovat evidenci funkcionalit a druhý je bude vizualizovat. Funkcionality jsou dále specifikovány vlastnostmi popsanými v následující části.

2.1.1 Modul pro evidenci funkcionalit

Bude vytvořen modul pro evidenci funkcionalit, jejich vlastností a vzájemných závislostí, který bude sloužit k jejich vytváření a správě. Tento modul bude přístupný pouze vývojářům a administrátorům IS/STAG jako webová aplikace.

Modul bude obsahovat formulář, který umožní zadávání všech potřebných parametrů, které se budou dané funkcionality týkat.

Funkcionalitu a její vlastnosti bude tvořit :

- název
- popis
- zda se funkcionalita týká aktuálního či budoucího období
- klíčová slova
- oblasti, ve který se nachází
- funkcionality, kterými bude podmíněna a popis této podmínky (funkcionalita může být provedena až poté, co bude provedena její podmiňující funkcionalita)
- vykonávání funkcionality:
  - objekt, kde se bude vykonávat

  - uživatelská role, která ji bude vykonávat (například studijní referentka, prorektor, fakultní rozvrhář ...)
  - dny, ve kterých se bude vykonávat
  - popis výkonu
  - odkaz na další informace

Modul uloží tato data do databáze. Dále bude umožňovat vyhledávat stávající funkcionality, které zobrazí v seznamu a povolí jejich editaci.

2.1.2 Modul pro vizualizaci funkcionalit

Dále bude vytvořen modul pro vizualizaci evidovaných funkcionalit. Tento modul je bude umožňovat vyhledávat a bude je zobrazovat. Bude přístupný všem uživatelům jako webová aplikace.

Vyhledávací filtr bude provádět vyhledávání podle:

- uživatelské role, která bude funkcionalitu vykonávat
- druhu období (aktuální či budoucí), ke kterému se funkcionalita vztahuje
- názvu funkcionality
- oblasti, ve které se nachází
- klíčových slov
- popisu funkcionality
- objektu, kde se bude vykonávat
- dnů, kdy se bude vykonávat

Vyhledané funkcionality se budou zobrazovat v seznamu, ve kterém budou uvedeny všechny jejich vlastnosti popsané v části 2.1.1. Seznam bude zobrazen podle požadavků zadavatele, které s ním budou v průběhu implementace ještě konzultovány.

3 Modelování business procesů

Kapitola se zabývá modelováním business procesů, které velmi připomínají funkcionality. Obsahuje popis několika nejpoužívanějších standardů pro modelování business procesů a zhodnocení, zda jsou tyto standardy vhodné pro modelování funkcionalit.

3.1 Funkcionality a jejich vztah k business procesům

Jednotlivé funkcionality dohromady tvoří funkční celek, zde to je informační systém. Jsou definovány vlastnostmi zmíněnými v části 2.1.1.

Obecně lze říci, že každá funkcionalita patří do určité oblasti, v určitou dobu ji vykonává konkrétní role, může být na jiné funkcionalitě závislá, nebo může další funkcionality podmiňovat. A právě tyto vlastnosti mají také business procesy, pro jejichž modelování v současné době existuje mnoho různých standardů.

3.2 Standardy pro modelování business procesů

Základními kategoriemi standardů pro modelování business procesů, jsou grafické a exekutivní standardy. Grafické standardy zobrazují business procesy pomocí diagramů. Exekutivní standardy umožňují spouštění procesů. Spojením obou druhů standardů je možné vytvořit nástroj, který zjednoduší řízení organizace.

3.3 Exekutivní standardy

Jak již bylo zmíněno výše, exekutivní standardy umožňují spouštění business procesů. To znamená, že osobu, která má proces vykonávat, bude nástroj implementující exekutivní standard vést průběhem vykonávání procesu.

Hlavními exekutivními standardy jsou BPML a BPEL.

3.3.1 BPML

BPML (Business Process Modeling Language) [REP07, KOR09, KLI14], vytvořený BPMI (Business Process Management Initiative) v roce 2002, je exekutivní metajazyk, který pomocí XML (eXtensible Markup Language) elementů popisuje strukturu procesu a sémantiku jeho spuštění, viz ukázka číslo 1. Je založen na matematickém základě PI kalkul[1].

Ukázka 1: Příklad business procesu v BPML[2]

```
<process name="ExamplePurchaseOrderHIEventWithMail">
<sequence name="Start HI Event Service">
<assign name="Assign" to="poNumber" from="DocToDOM(PrimaryDocument)/
     @Id" append="true"/>
<operation name="Human Interaction Event">
<participant name="HumanInteractionEvent"/>
<output message="HumanInteractionEventTypeInputMessage">
        <assign to="DocumentType">Purchase Order</assign>
        <assign to="FromAccount">
                SenderSterlingIntegratorUserAccount</assign>
        <assign to="NotificationBPName">WebSuiteEmailNotif</assign>
        <assign to="Operation">ADD</assign>
        <assign  to="ReferenceId"
                from="number(/ProcessData/poNumber/@Id)"/>
        <assign to="Status">Awaiting Approval</assign>
        <assign to="StorageArea">Inbox</assign>
        <assign to="SuspendAsWaiting">NO</assign>
        <assign to="SystemAccount">
                ReceiverSterlingIntegratorUserAccount</assign>
        <assign to="TemplateName">
                ExamplePurchaseOrderFromHTTPRequest</assign>
        <assign to="." from="*"/>
</output>
<input message="inmsg"><assign to="." from="*"/></input>
</operation>
</sequence>
</process>
```

[1] Popis komunikace dvou procesů založený na principu posílání zpráv.

[2] https://www-01.ibm.com/support/knowledgecenter/SS3JSW_5.2.0/com.ibm.help.svcs_adpts_a_l.doc/HI_Event_svc.html

BPML umožňuje nejenom vytvářet paralelní procesy nebo smyčky, ale také umožňuje vytvářet proměnné a rekurzivní bloky. I přesto, že je to formálně kompletní standard, po spojení společnosti BPMI s OMG (Object Management Group) již není více podporován.

Výhody BPML:

- programátoři se nemusí zabývat nízkoúrovňovým programováním, ale mohou se zaměřit na definování procesů
- zajišťuje znovupoužitelnost a škálovatelnost
- dokáže vyjádřit kompletně celý exekutivní proces

Nevýhody BPML:

- není již podporován
- může být pouze v systémech s čistým BPMS (Business Process Management Suite), jako je např. IBM MQServer nebo Websphere[3]

3.3.2 BPEL

BPEL (Business Process Execution Language) [KOR09] je exekutivní jazyk vytvořený společnostmi Microsoft a IBM kombinací jejich jazyků WSFL (Web Services Flow Language) a XLANG (XML Language). Jako BPML, viz část 3.3.1, je tvořen XML elementy, viz ukázka číslo 2. V roce 2003 byl pod názvem BPEL4WS standardizován společností OASIS (Organization for the Advancement of Structured Information Standards). Poté byl přejmenován na WS-BPEL.

BPEL postupně nahradil BPML, ačkoliv na rozdíl od něj neobsahuje všechny potřebné syntaktické značky. V současné době je nejpopulárnějším exekutivním standardem a nemá žádné vážné konkurenty.

Výhody BPEL:

- zaměřuje se spíše na procesy než na nízkoúrovňové programování

[3] http://www.ibm.com/cz-cs/

- dokáže modelovat složité interakce business procesů na pár řádek oproti konvenčním programovacím jazykům
- lze jej převést do grafického standardu BPMN, viz níže část 3.4.2
- je stabilní a riziko jeho zastarávání je minimální

Nevýhody BPEL:

- složitá syntaxe, která je obtížně implementovatelná
- standard je neúplný, některé kritické funkce v něm chybí
- chybí mu některé konstrukce procesů, a proto se spojuje např. s Javou nebo vhodným skriptovacím jazykem
- převod z grafického standardu BPMN je problematický
- nemodeluje dobře lidskou účast v business procesech

Ukázka 2: Příklad procesu v BPEL[4]

```
<process name="Assign_test_process">
        <operation name="AssignService">
                <participant name="AssignService"/>
                <output message="AssignOutputMessage">
                        <assign to="." from="*"/>
                        <!-- add constant value -->
                        <assign to="name1">value</assign>
                        <!-- add xpath value -->
                        <assign to="name2"
                                from="pathToElement/text()"></assign>
                </output>
                <input message="AssignInputMessage">
                        <assign to="." from="*"/>
                </input>
        </operation>
</process>
```

[4]http://www.worldofintegration.com/content/assign-service-bpml-code

3.4 Grafické standardy

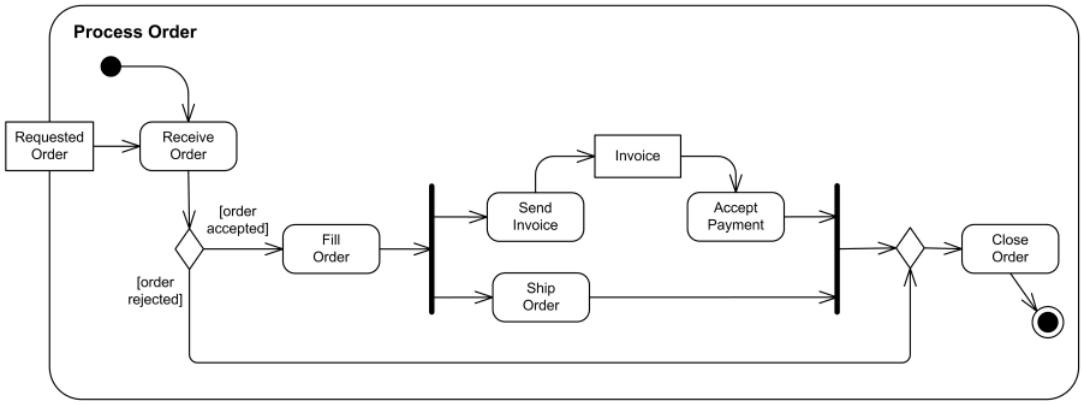
Jak již vyplývá z jejich názvu, grafické standardy pro modelování business procesů zobrazují tyto procesy graficky pomocí různých druhů diagramů. Hlavními standardy pro grafické modelování business procesů jsou UML AD a BPMN.

3.4.1 UML AD

Unified Modeling Language (UML) [REP07, KOR09, KLI14, UML15], standardizovaný v roce 2004 společností OMG (Object Management Group), vznikl jako modelovací jazyk, jehož nástroje měly při vývoji aplikací umožnit maximální využití objektově orientovaných principů. Postupem času se z něj stal obecný nástroj, který umožňuje modelování téměř čehokoliv. Jeho poslední vydaná verze 2.5 obsahuje 14 diagramů, z nichž se pro modelování procesů nejvíce hodí diagram aktivit (AD).

Diagram aktivit popisuje tok činností a přechody mezi nimi. Dokáže modelovat, kdo za danou aktivitu zodpovídá, či s jakými prostředky aktivita pracuje. Dále jej lze kombinovat s dalšími druhy diagramů a tím zachytit další skutečnosti. Z těchto důvodů je tento diagram pro modelování procesů velice často používán.

Příklad diagramu aktivit se nachází na obrázku 3.1.



Obrázek 3.1: Příklad procesu v UML AD[5]

[5]http://www.uml-diagrams.org/activity-diagrams-examples.html

Výhody UML AD:

- tento diagram lze rozšiřovat o další UML diagramy pro modelování více skutečností
- existuje zde možnost vytváření vlastních stereotypů, které slouží k zavádění nových elementů
- diagram poskytuje rozklad aktivit na dílčí aktivity

Nevýhody UML AD:

- relativní složitost pro popis jednoduchých procesů
- vlastní stereotypy nemají žádná sémantická pravidla, která by definovala jejich význam
- při použití většího množství různých diagramů a vlastních stereotypů se výsledný model může stát nepřehledným a těžko pochopitelným, zvláště pro automatické zpracování procesu
- uživatel musí mít alespoň základní znalosti více druhů UML diagramů

3.4.2 BPMN

Business Process Model and Notation (BPMN) [REP07, KOR09, BPM13, KLI14] je standard vyvinutý BPMI (Business Process Management Initiative) v roce 2004, který vznikl jako grafická reprezentace exekutivního standardu BPML popsaného v části 3.3.1. Později po nahrazení BPML standardem BPEL se stal grafickou reprezentací tohoto standardu, který je uveden v části 3.3.2. Cílem bylo poskytnutí notace, která bude čitelná pro všechny uživatele od analytiků přes vývojáře až po manažery. Necelý rok po spojení BPMI se společností OMG přešlo BPMN do správy této společnosti. V současnosti se standard vyskytuje ve verzi 2.0.2.

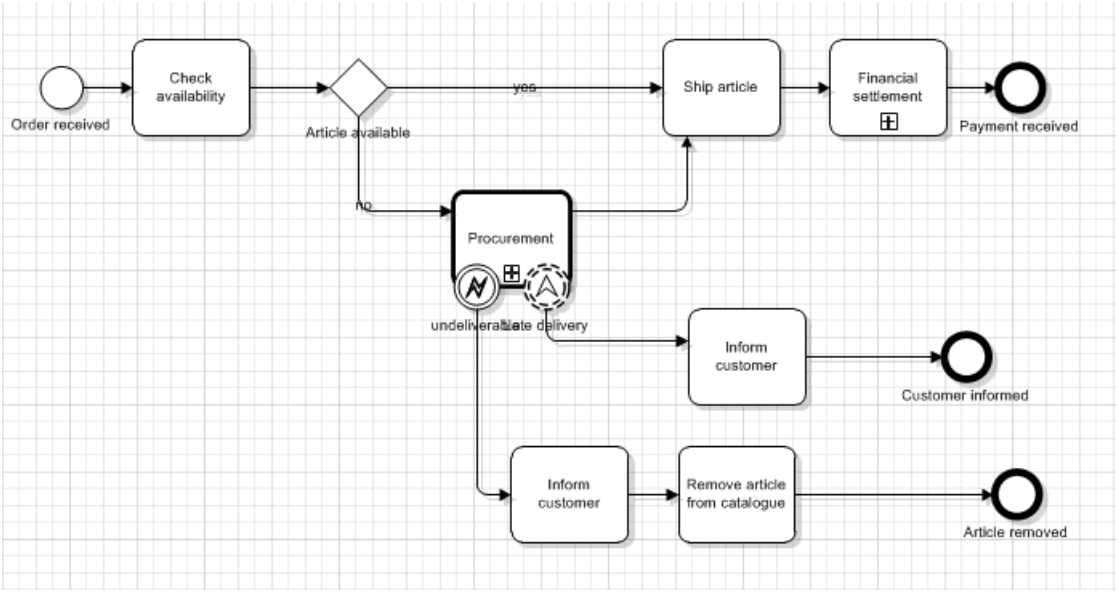
BPMN definuje Business Process Diagram (BPD), který vychází z vývojových diagramů, viz obrázek 3.2. Diagram modeluje tři druhy procesů: soukromé, veřejné a procesy spolupráce. Dále umožňuje modelovat proces z různých perspektiv podle typu uživatelů. Tento standard se postupně rozšiřuje mezi odborníky a stává se široce akceptovaným.

Výhody BMPN:

- dobře čitelný i pro netechnicky zaměřené uživatele
- dokáže obsáhnout komplexní procesy
- lze jej převést do exekutivního standardu
- došlo k formalizaci exekutivní sémantiky standardu

Nevýhody BMPN:

- při převodu do exekutivního standardu dochází ke ztrátě informací
- kvůli strojové interpretaci se zaměřuje na determinovatelnost procesů a tím potlačuje lidské aspekty procesu



Obrázek 3.2: Příklad procesu v BPMN[6]

3.5 Open Source nástroje pro modelování

V současné době existuje velké množství open source nástrojů, které umožňují modelování business procesů hlavně pomocí standardů UML, viz část 3.4.1, a BPMN,

[6]http://www.bpmn.org/

viz část 3.4.2. Některé z těchto nástrojů podporují i více druhů standardů. Hlavními nástroji pro grafické modelování business procesů jsou UMLet, Modelio, Camunda a Activiti. [BTO15, UTO15]

UMLet

UMLet [UMT15] dokáže modelovat UML diagramy. Tento nástroj má plugin do vývojového prostředí Eclipse[7].

Modelio

Modelio [MOD15] umožňuje modelování procesů pomocí standardů UML a BPMN.

Camunda

Camunda [CAM15] implementuje standard BPMN. Tento nástroj dříve fungoval na základě pluginu do vývojového prostředí Eclipse. Dnes má již vlastní grafické uživatelské rozhranní.

Activiti

Activiti [ACT15] modeluje business procesy pomocí standardu BPMN. Jako UMLet obsahuje plugin do vývojového prostředí Eclipse.

Po otestování těchto nástrojů jsem došla k následujícím poznatkům. Všechny tyto nástroje dobře umožňují modelovat business procesy v rámci možností daného standardu. Nástroje obsahující plugin do vývojového prostředí Eclipse usnadňují vývojářům integraci diagramu do projektu. Modelování procesu těmito nástroji je ale časově náročné a ve snaze o zachycení všech možných vazeb a vlastností mohou vznikat velmi nepřehledné diagramy.

[7] https://eclipse.org/

3.6 Zhodnocení

Pro účely našeho zadání je klíčovým prvkem zachycení doby, kdy se daná funkcionalita provádí. Je to proto, že na rozdíl od obvyklého business procesu mají funkcionality pevně danou dobu v roce, kdy musí být provedeny. Také je důležité zachycení všech jejich vlastností uvedených v části 2.1.1, podle kterých chceme mimo jiné zachovat možnost vyhledávání.

Na první pohled vypadá modelování funkcionalit pomocí standardů pro business procesy dobře. Potřebujeme zde zachytit určitou aktivitu v dané oblasti, kterou prování konkrétní uživatelská role. Avšak po podrobnějším prozkoumání jednotlivých standardů spolu s nástroji, které je modelují, zjistíme, že většina těchto standardů nedokáže dobře zachytit přesnou dobu, kdy bude muset funkcionalita proběhnout.

Dalším problémem je, že se nesnažíme o modelování jejich průběhu, jako je to u business procesů, ale chceme zachytit jejich vlastnosti a vztahy k ostatním funkcionalitám. A právě zachycení velkého množství těchto vlastností způsobí, že se diagramy mohou stát nepřehlednými. Také způsob uložení dat v XML na rozdíl od databázového uložení zkomplikuje vyhledávání funkcionalit podle jejich vlastností. Další nevýhodou je časová náročnost grafického modelování pomocí těchto standardů a jejich zbytečná komplexnost pro modelování funkcionalit.

Z těchto důvodů jsem došla k závěru, že pro modelování funkcionalit není použití zmíněných standardů vhodné a je lepší vytvořit nástroj nový, který se v grafické vizualizaci zmíněnými standardy pouze inspiruje.

4 Webové portálové Java aplikace

Tato kapitola se zabývá základním seznámením s portály a portlety. Stručně popisuje základní portálové komponenty specifikované vzniklými standardy. Na závěr se věnuje příkladu použití.

4.1 Vznik a vývoj

Portálové aplikace poprvé vznikly na přelomu let 1999 a 2000. Společnosti v této době přicházely s vlastními implementacemi, a proto vznikaly nekompatibilní portálové systémy s nepřenositelnými portletovými aplikacemi. Tato situace vedla v roce 2003 k vytvoření normy JSR-168 (Java Portlet Specification 1.0) [JSR1], která standardizuje portletový kontejner a obecně také tvorbu portletových aplikací. Tato norma byla přijata většinou velkých společností. Avšak již při jejím vytváření bylo zřejmé, že nebude stačit, protože byly vynechány složitější vlastnosti a funkce. Proto v roce 2008 vznikla verze 2.0 (JSR-286) [JSR2], která tyto nedostatky řeší a je s původní verzí zpětně kompatibilní.

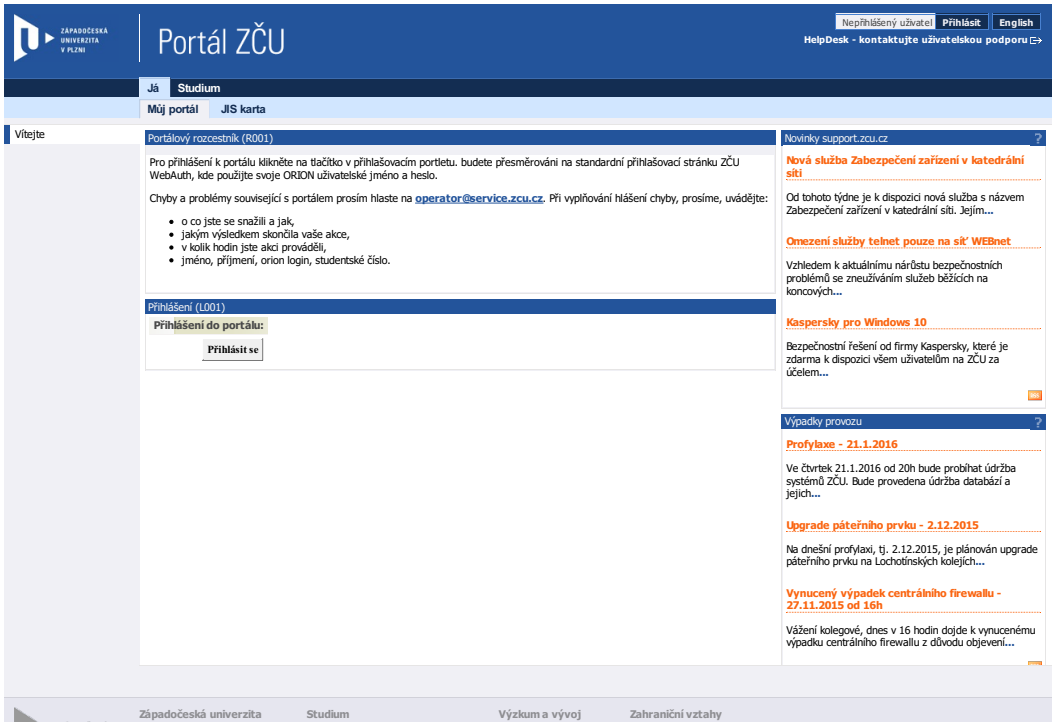
4.2 Portál

Podle standardu JSR-168 je portál webová aplikace, která uživatelům umožňuje personalizovaný přístup k informacím, jednotné přihlášení do systému, agregování obsahu z různých zdrojů informačního systému a jednotné uživatelské rozhraní.

Portál řídí celý běh systému, stará se o životní cyklus portletů, jejich správné vykreslování a vzájemnou komunikaci. Na obrázku číslo 4.1 je vidět příklad portálu Západočeské univerzity.

4.3 Portlet

Portlet je základní webová komponenta umístěná na portálové stránce, která generuje dynamický obsah. Je tvořen portletovým oknem s názvem a ovládacími prvky a obsahem, někdy také nazývaným fragment. Na portálové stránce se může současně nacházet více portletů, které nemusí patřit stejné aplikaci.



Obrázek 4.1: Portál ZČU[1]

Portlety můžeme srovnat se servlety, jelikož obě technologie fungují na podobném principu:

- jsou nasazovány do portletového nebo servletového kontejneru, který je spravuje
- generují dynamický obsah
- s klientem také komunikují pomocí požadavků a odpovědí

Najdou se mezi nimi i rozdíly:

- portlety netvoří celou stránku, ale generují pouze její fragment
- mohou se na portalové stránce vyskytnout i vícekrát
- svůj obsah musí generovat s ohledem na ostatní portlety

[1] https://portal.zcu.cz/portal/

4.4 Portletový kontejner

Portletový kontejner obsahuje portlety a řídí jejich životní cyklus. Poskytuje jim prostředí pro běh a úložný prostor pro jejich nastavení. Předává jim k vyřízení požadavky, které přijal od portálu. Portletový kontejner může být vytvořen buď jako samostatná komponenta, nebo může být součástí portálu.

4.5 Použití

Portletem je každá třída, která implementuje rozhraní `Portlet`. Doporučuje se spíše dědit třídu `GenericPortlet`, která implementuje rozhraní `Portlet` a poskytuje základní funkcionalitu.

Portletové aplikace obsahují dva deployment descriptory (DD). Prvním je `web.xml`, který specifikuje webové zdroje, jež nejsou portlety. Druhým je `portlet.xml` obsahující portletové zdroje. Každý portlet musí být uveden v `portlet.xml`. Příklad definice portletu je zobrazen v ukázce číslo 3.

Ukázka 3: Příklad `portlet.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app id="id" xmlns="http://java.sun.com/xml/ns/portlet/
             portlet-app_2_0.xsd" version="2.0">
    <portlet>
        <portlet-name>NazevPortletu</portlet-name>
        <display-name>Nazev</display-name>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>view</portlet-mode>
            <portlet-mode>help</portlet-mode>
        </supports>
        <supported-locale>cs</supported-locale>
        <resource-bundle>zdroj.Portlet</resource-bundle>
        <portlet-info>
            <title>Nazev portletu</title>
        </portlet-info>
    </portlet>
</portlet-app>
```

5 Spring framework

Kapitola se zabývá seznámením se Spring frameworkem. Popisuje jeho základní prvky a vlastnosti. Ve svém závěru podrobněji probírá některé jeho vybrané moduly.

5.1 Základní vlastnosti

Spring [SPR16] je open source framework pro enterprise Javu. Byl vydán v roce 2003 pod Apache 2.0 licencí[1]. Umožňuje vývoj webových aplikací bez nutnosti použít EJB[2] (Enterprise Java Beans) kontejner, bez kterého jde vytvořit „lehčí verze“ aplikací. Proto není potřeba Spring využívající aplikace nasazovat do J2EE (Java Enterprise Edition) kontejneru, jelikož jim pro běh stačí J2SE (Java Standard Edition) rozšířené pouze o některé knihovny. Pomocí Springu může být vytvořen vysoce výkonný, jednoduše testovatelný a znovupoužitelný kód.

Spring používá již existující technologie. Díky Dependency Injection (DI) usnadňuje například testování. Dále obsahuje dobře navržený webový MVC (Model-view-controller) framework.

5.1.1 Inversion of Control

Koncept Inversion of Control (IoC) přesouvá zodpovědnost za vytvoření a provázání objektů z aplikace na framework. Na rozdíl od procedurálního programování nejsou v aplikaci volány knihovní metody. Na vhodných místech je doplněn vlastní kód, který je následně volán frameworkem. Díky tomu je možné vytvořit nezávislý a znovupoužitelný kód.

Konkrétní implementací IoC je Dependency Injection (DI). DI zajišťuje, aby byly jednotlivé třídy aplikace na sobě nezávislé, a zároveň se stará o jejich propojení dohromady.

[1] http://www.apache.org/licenses/LICENSE-2.0
[2] http://www.tutorialspoint.com/ejb/

IoC kontejner

Spring IoC kontejner patří mezi základní funkce tohoto frameworku. Vytváří objekty, tzv. beany, které dále spravuje a řídí celý jejich životní cyklus. Pro správu jednotlivých komponent používá Dependency Injection. Dostává instrukce, co s jakými objekty provést. Tyto instrukce jsou definovány metadaty, která mohou být buď v XML formátu, jako anotace v Javě nebo jako Java kód. Pomocí Java objektů a metadat vytvoří aplikaci.

5.1.2 Aspektově orientované programování

Další vlastností Springu je aspektově orientované programování (AOP). Některé funkce programu se často rozprostírají po celé aplikaci. Může to být například logování, cachování nebo bezpečnost. AOP umožňuje tyto funkce udělat nezávislé tím, že je oddělí od objektů, které ovlivňují.

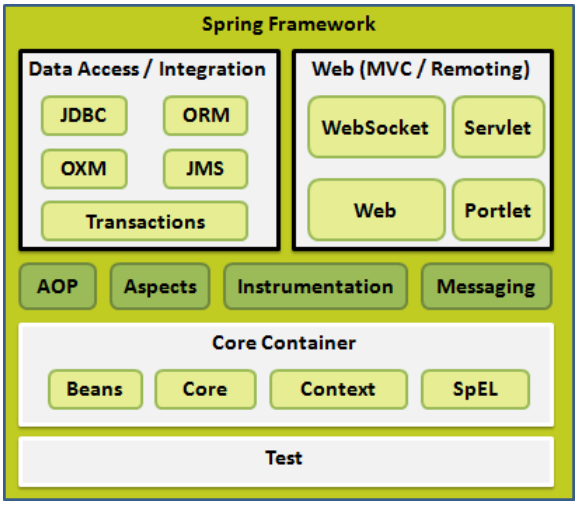
5.2 Moduly Springu

Spring obsahuje okolo 20 modulů, které jsou zobrazeny na obrázku 5.1. Dovoluje určit, které moduly se budou používat, a proto není nutné nainstalovat všechny. Moduly jsou rozděleny do oblastí podle jejich funkčnosti. V následující části je popsán modul, který byl v aplikaci použit.

5.2.1 Spring JDBC

JDBC (Java Database Connectivity) zajišťuje připojení do databáze. Zařizuje otevření a zavření spojení, přípravu a spouštění SQL (Structured Query Language) dotazů a správu transakcí. Umí také obsloužit výjimky týkající se databázového spojení. V programu stačí pouze definovat parametry připojení a SQL dotazy.

Spring JDBC poskytuje několik různých přístupů do databáze. Základním a nejpoužívanějším způsobem je použití třídy `JdbcTemplate` nebo jejích rozšíření. Běžnou praxí při použití této třídy je definování objektu `dataSource` v konfiguračním souboru. Pomocí DI je následně `dataSource` přidán do `DAO` (Data Access Object) třídy. Objekt `JDBCTemplate` je pak vytvořen v setter metodě pro `dataSource`.



Obrázek 5.1: Zobrazení modulů Spring frameworku [SPR16]

Instance třídy `JDBCTemplate` je *threadsafe*, proto ji stačí nakonfigurovat pouze jednou a následně je možné ji použít v několika `DAO` třídách. Umí spouštět SQL dotazy, volat uložené procedury nebo provádět iterace přes `ResultSet`.

6 Vizualizační technologie

Kapitola se zabývá technologiemi pro přehlednou vizualizaci dat. Neklade si za cíl popsat všechny vhodné technologie, které by bylo možné pro zobrazení dat použít. Pouze seznamuje s těmi, které budou při implementaci aplikace použity.

6.1 HTML5

HTML (HyperText Markup Language) je značkovací jazyk, který se používá pro tvorbu webových stránek. HTML5 je poslední vydaná specifikace tohoto jazyka. Tato verze nahrazuje příliš striktní XHTML (eXtensible HyperText Markup Language) a přináší nové vlastnosti.

Novinky v HTML5 [W3C14, W3S16]:

- nové API (Application Programming Interfaces) například pro geolokaci, lokální úložiště, aplikační cache, drag and drop a web workers
- nové značky pro strukturu dokumentu (article, header, footer, nav)
- nové značky pro vkládání grafických prvků (video, audio, canvas, svg)
- změna významu značek (b, i, script, small)
- odstranění značek grafické úpravy textu (big, center, font, frame)

6.2 CSS 3

CSS (Cascading Style Sheets) [W3S16] popisuje způsob zobrazení HTML (nebo XML) dokumentu na obrazovku, papír či jiná média. Definuje design, rozložení a zobrazení pro různá zařízení a velikosti obrazovek. Umožňuje řídit rozložení více stránek z jednoho místa najednou. Posledním vydaným standardem tohoto jazyka je v současné době CSS 3, které je zpětně kompatibilní s dřívější verzí CSS specifikace a přidává nové vlastnosti.

6.3 ECMAScript 6

ECMAScript [ECM15] je založen na několika technologiích, zejména na JavaScriptu[1] a JScriptu[2]. Stal se jedním z celosvětově nejpoužívanějších obecných programovacích jazyků. Je zabudovaný v prohlížečích a také ho používají servery a vestavěné aplikace. ECMAScript 6 neboli ECMAScript 2015 nabízí mnoho nových vlastností a vylepšení.

6.3.1 JQuery

JQuery [JQU16] je malá a rychlá javascriptová knihovna, která usnadňuje manipulaci s HTML dokumenty, zpracování událostí, animaci a práci s Ajaxem (Asynchronous JavaScript and XML). Stala se standardem, který je základem mnoha dalších javascriptových knihoven. Její rozhraní odstraňuje rozdíly mezi jednotlivými implementacemi JavaScriptu v různých prohlížečích.

[1] vyvíjeno společností Netscape
[2] vyvíjeno společností Microsoft

7 Implementace řešení

Kapitola se zabývá implementací zadání popsaného v kapitole 2. Popisuje databázi, samotnou aplikaci i její začlenění do ostatních aplikací informačního systému IS/STAG.

7.1 Začlenění do IS/STAG

Jelikož jsem součástí vývojářského týmu informačního systému IS/STAG, vyvíjela jsem tuto aplikaci jako jeho součást s použitím některých jeho knihoven. To, že aplikace bude začleněna do tohoto informačního systému, ovlivnilo výběr použitých technologií.

7.2 Databáze

Aplikace pro svůj běh používá databázi Oracle 12c[1], nad kterou běží celý informační systém IS/STAG.

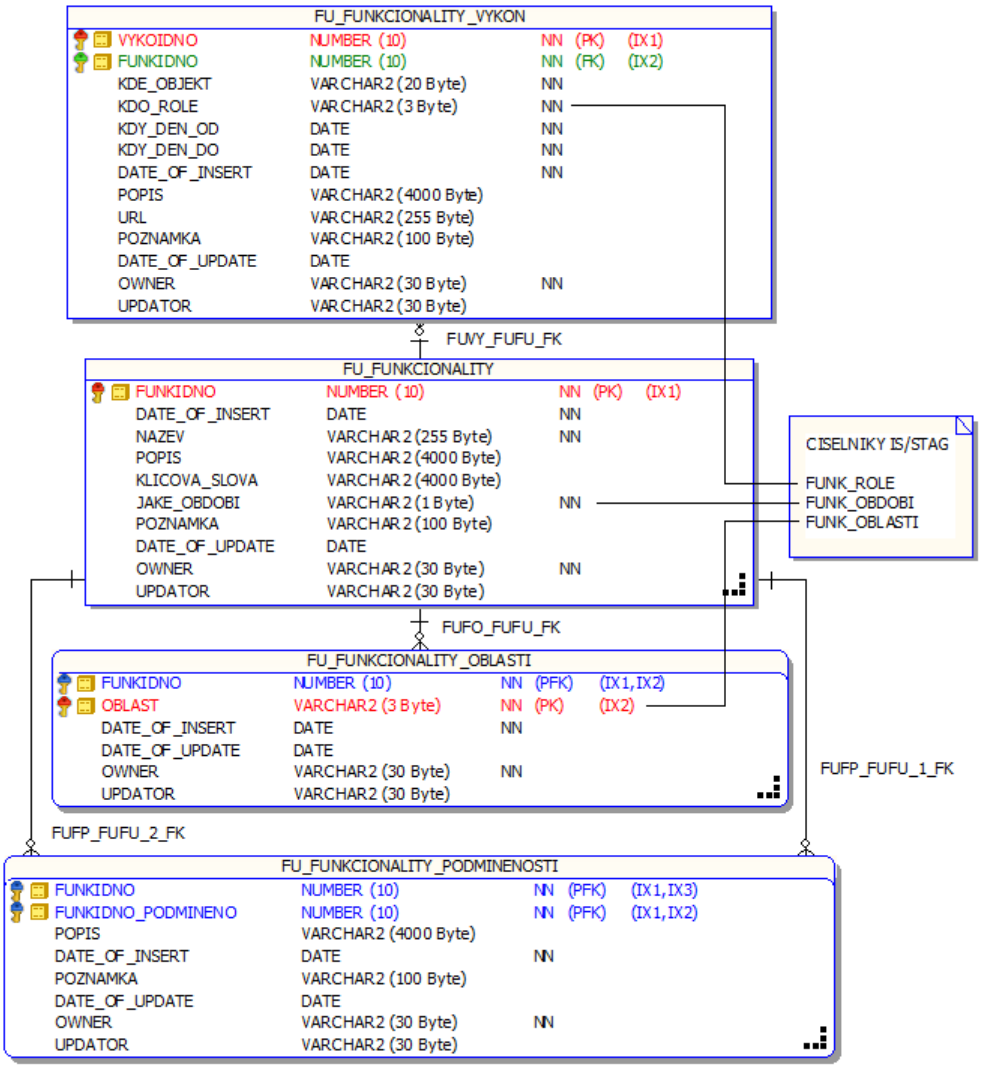
Celá struktura tabulek aplikace je zobrazena na obrázku 7.1. Tabulky dohromady uchovávají informace o funkcionalitách systému IS/STAG, jejich vztahy mezi sebou, oblasti IS/STAG, ve kterých se nachází a informace o vykonávání funkcionalit uživateli informačního systému. Všechny tabulky obsahují kromě uchovávaných dat i další provozní informace, jako je například datum vložení nebo změny, či osoba, která data vložila nebo změnila.

Tabulka `FU_FUNKCIONALITY` obsahuje základní informace o funkcionalitě. Nachází se zde její ID (`FUNKIDNO`), které je zároveň primárním klíčem, název, popis, klíčová slova, podle kterých ji bude možno vyhledat a období (aktuální nebo budoucí akademický rok), kterého se týká. Toto období odkazuje na číselník období, který je definovaný v IS/STAG.

Tabulka `FU_FUNKCIONALITY_VYKON` slouží k uchování informací o vykonávání funkcionality. Obsahuje primární klíč `VYKOIDNO` a cizí klíč `FUNKIDNO`, který vykonávání funkcionality propojuje se základními informacemi v tabulce `FU_FUNKCIONALITY` vazbou `1:N`. Dále se v ní nachází informace o místě vykonávání (`KDE_OBJEKT`), které

[1] http://www.oracle.com/us/corporate/features/database-12c/index.html

určuje formulář nebo portlet, ve kterém se funkcionalita vykonává. Uživatelská role (`KDO_ROLE`) určuje roli v informačním systému IS/STAG, která danou funkcionalitu vykonává. Tato role odkazuje na číselník rolí definovaný v IS/STAG, například to může být studijní referentka, rozvrhář či administrátor. Dále je zde evidovaná doba vykonávání funkcionality pomocí dat `KDY_DEN_OD` a `KDY_DEN_DO`. `POPIS` obsahuje popis výkonu funkcionality. V `URL` se nachází odkaz na nápovědu k formuláři nebo portletu, ve kterém se funkcionalita v rámci IS/STAG vykonává.



Obrázek 7.1: ERA model zobrazující strukturu tabulek v databázi

V tabulce `FU_FUNKCIONALITY_OBLASTI` jsou evidovány oblasti, ve kterých se funkcionalita nachází. `OBLAST` kromě určení místa slouží také jako primární klíč. Odkazuje na číselník oblastí definovaný v IS/STAG. Příkladem oblastí je například předzápis,

přijímačky nebo VŠKP. Tabulka je propojena s tabulkou `FU_FUNKCIONALITY` cizím klíčem `FU_FUNKIDNO` vazbou `1:N`.

Tabulka `FU_FUNKCIONALITY_PODMINENOSTI` eviduje podmíněnosti mezi jednotlivými funkcionalitami. To znamená, že je zde pomocí dvou cizích klíčů `FUNKIDNO` a `FUNKIDNO_PODMINENO` dvěma vazbami `1:N` do tabulky `FU_FUNKCIONALITY` uchován vztah mezi podmiňující a podmíněnou funkcionalitou. Podmiňující funkcionalita musí být vykonána před tou, kterou podmiňuje. Oba cizí klíče dohromady tvoří také primární klíč. Dále je zde zaznamenán popis podmíněnosti, který komentuje vztah mezi dvěma funkcionalitami.

7.3 Struktura aplikace

Aplikace je vytvořena jako portletová webová aplikace, která běží ve webovém kontejneru Apache Tomcat[2]. Je nasazena v portálu IS/STAG, kde je zaintegrována mezi ostatní aplikace tohoto informačního systému. Splňuje standard API JSR286, a proto ji lze nasadit na libovolném jiném portálu, který tento standard splňuje. Více informací o portálových aplikacích je uvedeno v kapitole 4.

Aplikace je postavena nad softwarovou architekturou MVC (model-view-controller), která rozděluje aplikaci na datový model, řídící logiku a uživatelské rozhraní. Tyto části jsou od sebe odděleny tak, že změna v některé z nich má jen minimální dopad na ostatní části aplikace.

Dále byl použit Spring framework, který je podrobněji popsán v kapitole 5. Připojení do databáze se uskutečňuje pomocí modulu Spring JDBC.

Aplikace běží ve Spring IoC kontejneru, který ji spravuje. Informace, které Spring potřebuje pro správu aplikace, jsou definovány v XML souborech pomocí metadat. Mezi tyto informace patří například určení implementace použitého rozhraní. Konfigurační soubory jsou spojeny dohromady v souboru `applicationContext.xml`.

Aplikace pro svůj běh používá některé knihovny z informačního systému IS/STAG. Těmito knihovnami jsou například modul pro práci s číselníky a modul pro správu a validaci fomulářů.

[2] http://tomcat.apache.org/

7.3.1 Model

Model je vrstva, která získává a zpracovává data aplikace. V tomto případě získává data z databáze. Data jsou ukládána do datových beanů, které obsahují pouze konstruktor a metody pro přístupu k datům tzv. gettery a settery.

FunkDAO

Připojení do databáze je realizováno přes třídu Springu `NamedParametrJdbcDaoSupport`, kterou dědí implementace třídy `FunkDAO`. V této třídě jsou implementovány metody pro vkládání, úpravu a mazání dat z databáze. Dále se zde nachází metody pro vyhledávání dat v databázi. Metody pro komunikaci s databází získají pomocí metody `getNamedParametrJdbcTemplate` objekt, nad kterým volají metody `update` a `query` pro úpravu nebo získání dat.

Získaná data z databáze je nutné převést do Java objektů. K tomuto převodu je možné použít některý z ORM (Object-Relational-Mapping) frameworků (např. Hibernate[3]). Avšak pro účely této aplikace se využití ORM frameworku nehodilo. Proto bylo použito řešení s vlastními SQL dotazy a ručním mapováním dat z objektu `ResultSet` pomocí objektu `RowMapper` do Java objektů.

FunkManager

Přístup k `FunkDAO` objektu je realizován v implementaci třídy `FunkManager`. Objekt je nastavem pomocí setteru a jsou zde volány jeho metody. `FunkManager` některá získaná data podle potřeby dále upravuje.

7.3.2 Controller

Controller obsluhuje požadavky od uživatele a řídí běh aplikace. V souboru `web.xml` je nastaveno umístění aplikačního kontextu. Dále je zde definován `PagesDispatcherServlet`, který obsluhuje požadavky od uživatele. Také se zde nachází seznamy výchozích a chybových stránek.

[3] http://hibernate.org/orm/

Formuláře

Aplikace obsahuje vyhledávací formulář. Tento formulář je vytvořen pomocí knihovny IS/STAG která umí validovat data zadaná do formuláře. Je definován v souboru `webForms.xml`. Zde jsou určeny jednotlivé položky formuláře a jejich vlastnosti. Knihovna z těchto definic vytvoří HTML elementy formuláře.

Dále v aplikaci existují formuláře pro vkládaní funkcionalit do databáze. Tyto formuláře jsou vytvořeny stejným způsobem jako vyhledávací formulář. Dělí se na formuláře v podobné struktuře, jako jsou data uložena v databázi.

Formuláře jsou dále registrovány ve springovském konfiguračním souboru `forms.xml`. Zde jsou uvedeny formou beanů a jsou propojeny s třídami, které se starají o jejich inicializaci a zpracování. Ve vlastnostech těchto beanů mají uveden odkaz na bean s implementací rozhraní `FunkManager`, který využívají pro přístup k datům.

Třídy pro inicializaci a zpracování formulářů dědí třídu `WebForm` z knihovny informačního systému IS/STAG, která obsahuje základní logiku formulářů. Jsou zde přednastavovány některé hodnoty formulářových prvků, celý formulář je zde inicializován a zpracováván, mohou se zde měnit části formuláře před nebo po zpracování a v případně nevalidních dat se zde nastavují chybové zprávy. Po zpracování formuláře se zde nastavuje další část controlleru, která bude data dále zpracovávat, spolu s parametry zobrazení. Data formuláře jsou ukládána ve třídě `WebFormData` a nachází se v session aplikace.

Stránky aplikace

Stránky aplikace neboli pages jsou třídy, které dědí abstraktní třídu `PageJSP` z knihovny IS/STAG. Tyto třídy se starají o zpracování a vykreslení konkrétní stránky portletu.

Jednotlivé pages jsou evidovány ve springovském konfiguračním souboru `pages.xml`. Zde jsou uvedeny v beanu, který odkazuje na portlet. Tento bean obsahuje seznam několika pages, které se v něm vyskytují. Každá page obsahuje odkaz na třídu, která ji reprezentuje, jméno, které lze používat při jejím volání a název JSP souboru, který page vykresluje.

Pages obsahují reference na rozhraní `FunkManager` a třídu `FunkFiltrForm`, která reprezentuje vyhledávací formulář. V metodě `doView` pomocí těchto objektů získávají data pro další zpracování a přípravu pro následné vykreslení. V metodě `doAction` probíhá zpracování dat.

Portlety

Aplikace obsahuje dva portlety. První portlet slouží ke správě funkcionalit, druhý je vizualizuje. Oba portlety jsou uvedeny v souboru `porlet.xml`, kde jsou definovány jejich vlastnosti, jako je například jméno, třída nebo zdroj. Příklad těchto konfiguračních vlastností je zobrazen v podkapitole 4.5.

Portlety dědí třídu `GenericPagesFormsPortlet`, která se nachází v knihovně `PortletValidation` a je přizpůsobena pro portlety používající pages a formuláře. Oba portlety vytváří vlastní `sessionBean`, který uchovává data v session aplikace.

Jelikož aplikace obsahuje dva portlety a každý vytváří jiný `sessionBean`, který se následně používá ve stránkách aplikace, je vytvořena abstraktní třída `AbstractSessionBean`, od které jsou oděděné dvě třídy přizpůsobené svým portletům.

7.3.3 View

View neboli pohled je zodpovědný za zobrazení dat. Převádí je do HTML podoby, kterou prohlížeč následně vykreslí uživateli.

JSP

Pro definování vzhledu stránky slouží JSP (JavaServer Pages), které umožňují dynamické generování stránek na straně serveru. Dovolují vkládat Java kód mezi HTML značky, které můžou být dále doplněny o další prvky.

JSP obvykle umožňují používat direktivy, deklarace, výrazy a scriptlety. V těchto elementech se nachází Java kód v určité formě, který zajišťuje vytvoření části HTML stránky. Dále se zde používá JSP Expression Language (EL), který umožňuje jednodušší přístup k objektům. JSP dále obsahují akční elementy pro řízení generování stránky.

Další součástí JSP jsou taglibs, které napomáhají k odstranění scriptletů. Kromě již existujících je možné vytvořit a používat vlastní taglib. Mezi standardní patří například JSTL (JSP Standard Tag Library), která obsahuje základní funkčnosti.

Použití

Data jsou v JSP získávána z atributů požadavku nebo session. Jednotlivé stránky jsou vytvářeny spojováním více JSP souborů. Pro přístup k datům je použita taglib `jstl`. Lokalizace textů je zajištěna přes taglib `fmt`, která načítá data podle vybraného jazyka.

Kromě základních taglibs jsou zde použity taglibs z aplikace informačního systému IS/STAG. Je to taglib `pages`, která slouží k vytváření záložek a odkazů, a `val`, která pomáhá vytvářet formuláře definované v konfiguračním souboru `webForms.xml`.

Vzhled je formátován CSS styly, které jsou částečně převzané z aplikace informačního systému kvůli vzhledovému zaintegrování a zbytek je definován přímo pro tuto aplikaci.

Ve vizualizačním portletu je použita javascriptová knihovna `qTip`[4] pro vytváření bublin s dalšími informacemi o funkcionalitách. Pomocí jQuery s CSS styly je dotvářeno barevné zobrazení podmíněností mezi funkcionalitami.

7.4 Logování

V rámci aplikace je použito logování dat, která se ukládají do databáze nebo se z ní mažou. Jelikož je tato aplikace postavena hlavně na datech, v případě chyby je nutné vědět, co se přesně kde stalo. Logování je nastaveno v konfiguračním souboru `logging.xml`, kde je definován logovací nástroj informačního systému IS/STAG.

7.5 Podpora pro monitoring

V aplikaci se nachází `TestServlet`, který slouží pro monitorování běhu aplikace. Tento servlet se nachází ve většině aplikací informačního systému IS/STAG. Z knihovny tohoto informačního systému implementuje `GenericTestServlet`, který obsahuje základní potřebnou funkcionalitu.

Tento servlet informuje o tom, zda aplikace běží, jaká je její verze a v jakém stavu se nachází. Tyto informace se pak zobrazují v aplikaci sloužící k monitorování a nasazování aplikací, ke které mají přístup administrátoři a mohou v ní vše sledovat.

[4] http://qtip2.com/

8 Nasazení a provoz aplikace

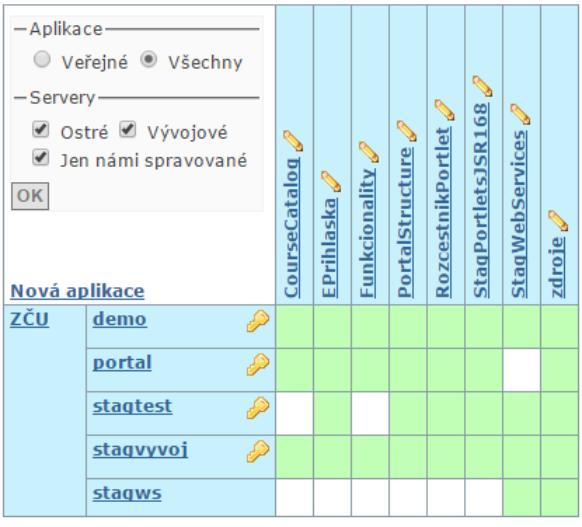
Kapitola se zabývá postupem sestavení a nasazení aplikace. Dále popisuje umístění nasazené aplikace. Zmiňuje se také o uživatelské dokumentaci.

8.1 Nasazení aplikace

Aplikace je sestavena pomocí nástroje Gradle[1]. Tento nástroj je založen na Groovy[2] a slouží k automatizaci procesů. Pro sestavení aplikace jsou vytvořeny buildovací scripty `build.gradle` a `settings.gradle`, které popisují sestavení aplikace.

Aplikaci je možné sestavit ručně po nainstalování gradlu. Druhou možností je použití buildovacího nástroje, který je používán pro sestavování a nasazování aplikací informačního systému IS/STAG. Tento nástroj běží jako webová aplikace a je přístupný pouze vývojářům.

V tomto webovém rozhraní je možné aplikaci sestavit a nasadit ji na server. Jsou zde zobrazeny již existující úspěšné i neúspěšné buildy aplikací i údaje z testovacích servletů, které podávají informace o nasazených aplikacích. Část aplikace je zobrazena na obrázku 8.1, který zobrazuje, na kterých serverech jednotlivé aplikace běží.



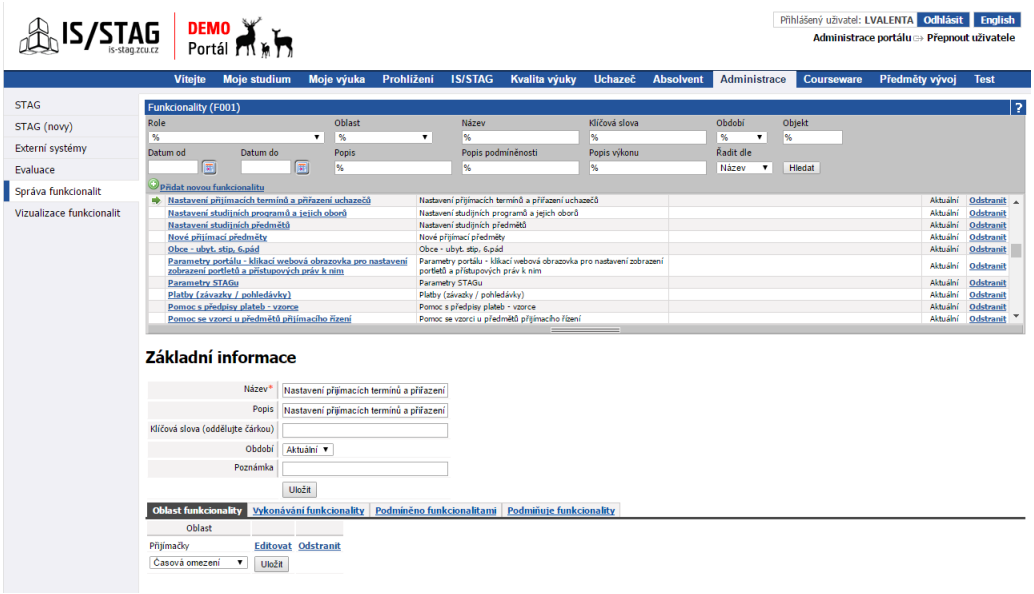
Obrázek 8.1: Část buildovacího nástroje s nasazenými aplikacemi

[1] https://docs.gradle.org/current/userguide/userguide.html
[2] http://www.groovy-lang.org/

8.2 Provoz aplikace

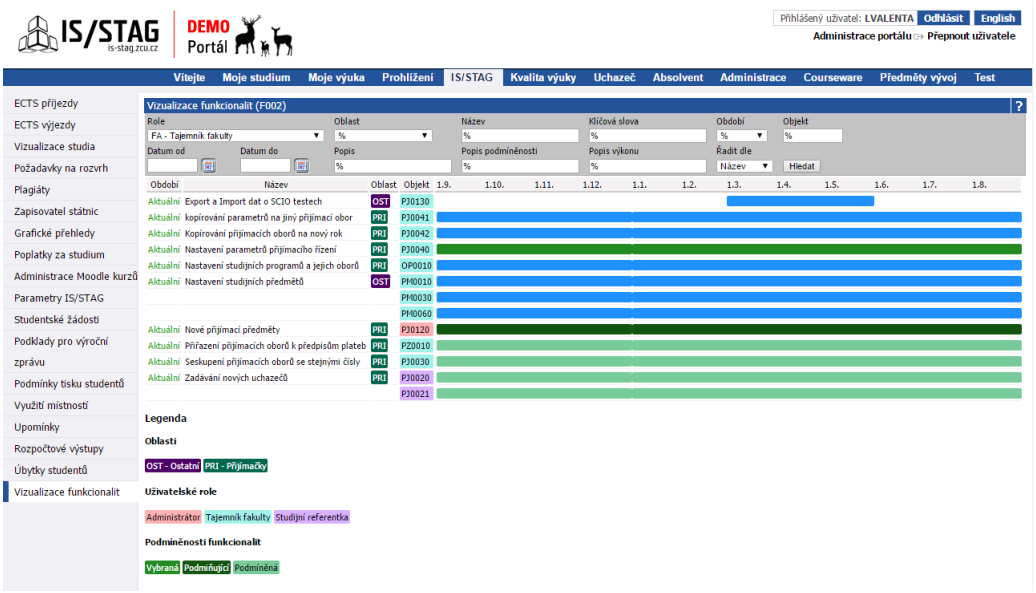
Pro účely této práce je aplikace nasazena na demo portálu, který je přístupný na adrese `https://stag-demo.zcu.cz/`. Zde je možné aplikaci otestovat. Uživatel si může v dolní části úvodní stránky zvolit, pod jakou rolí se chce do portálu přihlásit. To provede kliknutím na vybraný login.

Pro přístup k evidenční části aplikace se musí uživatel přihlásit pod rolí administrátor. Pod touto rolí má přístup i k vizualizační části. Tyto části aplikace se nachází v záložce `Administrace` na podstránkách `Správa funkcionalit` a `Vizualizace funkcionalit`. Jejich umístění v portálu je zobrazeno na obrázku 8.2, který zároveň zobrazuje vzhled editační části aplikace.



Obrázek 8.2: Editační část aplikace

Vizualizační část aplikace se dále nachází také v záložce `IS/STAG` na podstránce `Vizualizace funkcionalit`. K tomuto portletu má přístup většina uživatelských rolí, kromě role student. Umístění aplikace spolu s vzhledem vizualizační části je zobrazeno na obrázku 8.3.



Obrázek 8.3: Vizualizační část aplikace

Pro použití je aplikace nasazena na ostrém portálu ZČU, který se nachází na adrese `https://portal.zcu.cz/`. Zde je její vizualizační část umístěna v záložce `Ostatní` na podstránce `Vizualizace funkcionalit`.

8.3 Uživatelská dokumentace

Jelikož je tato aplikace součástí informačního systému IS/STAG, je její uživatelská dokumentace umístěna mezi ostatní části dokumentace tohoto informačního systému, kde se dokumentace udržují v aktualizované podobě. Je napsaná ve formátu `DocBook`, ze kterého je každý den generovaná na web do HTML a PDF.

Na dokumentaci se lze dostat po kliknutí na otazník v portletu aplikace, kde se zobrazí stránka s odkazem do dokumentace nebo přímo zadáním adresy `http://is-stag.zcu.cz/napoveda/stag-v-portalu/funkcionality.html` do adresního řádku prohlížeče.

9 Závěr

V rámci této bakalářské práce byly prozkoumány standardy a nástroje pro modelování business procesů. Bylo zjištěno, že tyto standardy nedokážou dobře modelovat datum vykonávání ani přehledně zobrazovat mnoho vlastností funkcionalit informačního systému IS/STAG. Proto nebyly tyto standardy pro modelování funkcionalit použity.

Dále byly prozkoumány technologie, pomocí kterých byla vytvořena aplikace. První část této aplikace umožňuje administrátorům zadávat a editovat nové funkcionality. Druhá část tyto funkcionality vizualizuje uživatelům, kteří se díky tomu mohou dozvědět, kdy mají jakou činnost, na jakém místě a jakým způsobem vykonávat.

Aplikace byla nasazena na portál informačního systému IS/STAG, v rámci kterého bude dále vyvíjena a upravována podle požadavků uživatelů, kteří ji budou používat. V budoucnu je plánováno rozšíření pro další školy, které používají tento informační systém. Toto rozšíření bude umožňovat administrátorům na ostatních školách vlastní modifikaci funkcionalit podle činností jejich uživatelů.

Seznam použitých zkratek

AJAX	Asynchronous JavaScript and XML
AOP	Aspect-oriented Programming
API	Application Programming Interface
BPD	Business Process Diagram
BPEL	Business Process Execution Language
BPMI	Business Process Management Initiative
BPML	Business Process Modeling Language
BPMN	Business Process Model and Notation
CSS	Cascading Style Sheets
DAO	Data Access Object
DD	Deployment Descriptor
DI	Dependency Injection
ECMA	European Computer Manufacturers Association
EJB	Enterprise Java Beans
EL	JSP Expression Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IoC	Inversion of Control
IS/STAG	Informační systém studijní agendy
JDBC	Java Database Connectivity
JSP	JavaServer Pages
JSR-168	Java Specification Request 168 (Java Portlet Specification 1.0)
JSR-286	Java Specification Request 286 (Java Portlet Specification 2.0)
JSTL	JSP Standard Tag Library
MVC	Model-view-controller
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group

ORM	Object-Relational-Mapping
SQL	Structured Query Language
UML	Unified Modeling Language
UML AD	UML diagram aktivit
URL	Uniform Resource Locator
WSFL	Web Services Flow Language
XHTML	eXtensible HyperText Markup Language
XLANG	XML Language
XML	eXtensible Markup Language

Seznam obrázků

Literatura

[ACT15] *Activiti* [online]. cit[2015-12-03]
http://www.activiti.org/

[BPM13] *Business Process Model and Notation* [online]. 2013. cit[2015-12-02]
http://www.omg.org/spec/BPMN/2.0.2/

[BTO15] *BPMN Tool Matrix* [online]. cit[2015-12-03]
https://bpmnmatrix.github.io/

[CAM15] *Camunda* [online]. cit[2015-12-03]
https://camunda.org/

[ECM15] *ECMAScript 2015 Language Specification* [online]. 2015. cit[2016-01-26]
http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

[JSR1] *JSR 168: Portlet Specification* [online]. 2003. cit[2016-01-19]
https://www.jcp.org/en/jsr/detail?id=168

[JSR2] *JSR 286: Portlet Specification 2.0* [online]. 2008. cit[2016-01-19]
https://www.jcp.org/en/jsr/detail?id=286

[KLI14] KLIMEŠ, Cyril: *Modelování podnikových procesů* [online]. Ostrava. 2014. cit[2015-12-02]
http://www1.osu.cz/ zacek/mopop/mopop.pdf

[KOR09] KO, Ryan K. L., LEE, Stephen S. G., LEE, Eng Wah: *Business process management (BPM) standards: a survey* [online]. 2009. cit[2015-12-02]
https://ryanko.files.wordpress.com/2008/03/bpm-journal-koleelee-bpms-survey.pdf

[JQU16] *jQuery* [online]. 2016. cit[2016-01-26]
http://jquery.com/

[MOD15] *Modelio* [online]. cit[2015-12-03]
`https://www.modelio.org/`

[REP07] ŘEPA, Václav: *Podnikové procesy: Procesní řízení a modelování*. 2. vyd. Praha: Grada Publishing. 2007
ISBN 978-80-247-2252-8

[SPR16] *Spring Tutorial* [online]. cit[2016-03-26]
`http://www.tutorialspoint.com/spring/`

[UML15] *OMG Unified Modeling Language* [online]. 2015. cit[2015-12-02]
`http://www.omg.org/spec/UML/2.5/`

[UMT15] *UMLet* [online]. cit[2015-12-03]
`http://www.umlet.com/`

[UTO15] *UML Tools* [online]. cit[2015-12-03]
`http://www.diagramming.org/`

[W3C14] *HTML5 Differences from HTML4* [online]. 2014. cit[2016-01-20]
`https://www.w3.org/TR/html5-diff/`

[W3S16] *w3schools.com* [online]. cit[2016-01-20]
`http://www.w3schools.com/`