Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Dokumentácia projektu

Portálový rámec na báze technológií .NET a webu so sémantikou

Študijný program: Informačné systémy

Ročník: 1. inžinierskeho štúdia

Predmet: Tímový projekt

Ak. rok: 2007/2008

Názov tímu: M4RS

Kontakt na tím: timovyprojektfiit@googlegroups.com

Cvičiaci : Ing. Michal Tvarožek

História vývoja dokumentu

Dátum zmeny	Verzia dokumentu	Opis	Autor
08.11.2007	1.0	Funkcionálne požiadavky systému	Bc. Országh Miloš
08.11.2007	1.1	Nefunkcionálne požiadavky systému	Bc. Kajaba Michal
09.11.2007	1.2	Používateľské role	Bc. Jurík Miroslav
09.11.2007	1.3	Doplnenie Use Case modelov, work – flow modelu. Gramatická úprava.	Bc. Balocký Stanislav
09.11.2007	1.4	Korekcia celého dokumentu	Bc. Kajaba Michal
12.11.2007	1.5	Doplnenie analytickej časti a korekcia dokumentu podľa pripomienok	Bc. Országh Miloš, Bc. Jurík Miroslav
12.11.2007	1.6	Pridanie hrubého návrhu, popis architektúry systému	Bc. Červenák Roman
12.11.2007	1.61	Hrubý návrh systému, popis údajov použitých v systéme	Bc. Fris Martin
12.11.2007	1.62	Hrubý návrh systému, popis dátovodov, popis prezentačnej časti, motivácia	Bc. Balocký Stanislav
13.11.2007	1.63	Popis princípu komunikácie modulov	Bc. Červenák Roman
13.11.2007	1.7	Revízia dokumentu	Bc. Országh Miloš Bc. Fris Martin
14.11.2007	1.75	Zapracovanie pripomienok MT k dokumentu	Bc. Červenák Roman
14.11.2007	1.8	Doplnenie 4.7. Zhrnutie	Bc. Fris Martin
24.2.2008	1.9	Rozšírenie pôvodného návrhu, úprava architektúry	Bc. Országh Miloš Bc. Fris Martin
31.2.2008	1.92	Architektúra systému - zmeny oproti pôvodnému návrhu	Bc. Červenák Roman
15.3.2008	1.93	Knižnica na vývoj modulov	Bc. Fris Martin
10.4.2008	1.94	Bezpečnosť systému	Bc. Červenák Roman

Dátum zmeny	Verzia dokumentu	Opis	Autor
			Bc. Jurík Miroslav
11.4.2008	1.95	Bezpečnosť systému - doplnok	Bc. Balocký Stanislav, Bc. Červenák Roman
17.4.2008	1.96	Technická dokumentácia	Bc. Jurík Miroslav, Bc. Fris Martin
27.4.2008	1.961	Návod na inštaláciu IIS	Bc. Balocký Stanislav
28.4.2008	1.962	Inštalácia produktu	Bc. Balocký Stanislav
29.4.2008	1.963	Konfigurácia Coonda IIS	Bc. Jurík Miroslav
29.4.2008	1.964	Tvorba modulov pomocou knižnice na vývoj modulov	Bc. Červenák Roman, Bc. Fris Martin
29.4.2008	1.97	Používateľská príručka	Bc. Kajaba Michal, Bc. Országh Miloš
19.5.2008	2.0	Záverečné upravy dokumentu	Bc. Országh Miloš

Obsah

1. Úvod

1.1 Účel a rozsah dokumentu

Účelom dokumentu je analýza, špecifikácia a hrubý návrh projektu „*Portálový rámec na báze technológií .NET a webu so sémantikou*“.

Kapitola „*Opis riešeného problému*“ obsahuje analýzu portálového riešenia, integráciu aplikácií a vývojového prostredia na platforme .NET.

Kapitola „Špecifikácia požiadaviek“ zahŕňa a presne definuje správanie navrhovaného systému sformulované na základe zákazníckych požiadaviek.

V kapitole „Hrubý návrh riešenia“ je opísaná architektúra vytváraného portálového rámca od jeho dekompozície na časti a priblíženia každej z nich až po zadefinovanie typu údajov, ktoré v ňom budú používane.

1.2 Motivácia

Hlavnou motiváciou pre tento tím je možnosť naučiť sa efektívnej tímovej práci v procese a jednotlivých fázach analýzy, návrhu a implementácie informačných systémov. Záujem o tému tohto projektu v nás vyvolala odhodlanie získať praktické poznatky z oblasti tvorby webových portálov na baze technológie Microsoft .NET. Samotný námet pre tuto tému vzišiel z požiadavky Fakulty informatiky a informačných technológii STU v Bratislave, predmetu *Vývoj informačného systém v tíme I, II,* ktorej sféry výskumu smerujú aj do problematiky zadania projektu. Veľkou výzvou pre nás je teda vytvorenie portálového rámca požadovanej kvality na spomínanej platforme, ktorý bude nemalým prínosom pre potreby fakulty ako aj pre cely náš tím. V konečnej fáze teda dúfame, že si naše riešenie alebo jeho jednotlivé časti nájdu praktické využitie aj v praxi.

1.3 Skratky

WYSIWYG	What You See Is What You Get. Nástroj na modifikáciu obsahu prostredníctvom grafických prvkov.
OWL	Web Ontology Language
SAX	Simple Api for Xml
API	Application Programming Interface

REST	REpresentational State Transfer
AJAX	Asynchronous JavaScript and XML
RSS	Rich Site Summary, Really Simple Syndication
XML	Extensible Markup Language
HTML	HyperText Markup Language
WML	Wireless Markup Language
VoiceXML	Voice Extensible Markup Language
XHTML	Extensible HyperText Markup Language

1.4 Odkazy a zdroje

[1] Bieliková, M. Softvérové inžinierstvo: Princípy a manažment. Slovenská technická univerzita v Bratislave. 220 s. 2000.

[2] Bieliková, M.: Ako úspešne vyriešiť projekt. Slovenská technická univerzita v Bratislave. 158 s. 2000.

[3] Portálový rámec Cocoon, stránka projektu
http://cocoon.apache.org/

[4] Prototype Portal Class, stránka projektu
http://blog.xilinus.com/

[6] Mashup a Web 2.0
http://www.ibm.com/developerworks/web/library/x-mashups.html

2. Opis riešeného problému

V tejto časti stručne opíšeme povahu riešeného problému a všeobecný kontext vytváraného softvérového systému. Stručne charakterizujeme čo je portálové riešenie a jeho špecifiká, ďalej popíšeme technológie podporujúce tvorbu portálových riešení. Kapitolu zakončíme zhrnutí analytickej časti.

2.1 Portálové riešenie

Portálové riešenie alebo portál sa používa k označeniu webovej stránky, ktorá slúži ako vstupný bod do internetu. Obvykle umožňujú rýchly prístup k veľkému množstvu informácií na jednom mieste, od fulltextového vyhľadávania cez katalógy odkazov až po najrôznejšie novinky. Zoskupené aplikácie sú charakteristické spoločnými prvkami ovládania a štýlom grafického výzoru za účelom dosiahnutia, čo najefektívnejšej použiteľnosti portálu. Tento prístup poskytuje priestor pre viaceré používateľské role, kde sú konkrétnemu používateľovi sprístupnené len určité funkcie systému a personalizácia obsahu pomocou WYSIWYG editorov. Takisto centralizácia nástrojov pre administráciu celého systému podporuje myšlienku portálového riešenia. Kedy je administrátorovi sprístupnená konfigurácia všetkých aplikácií vystupujúcich v portálovom riešení na jednom mieste.

2.2 Mashup

Mashup je webová stránka kombinujúca dáta z viac ako jedného zdroju formou integračného nástroja. Mashup webová stránka je do značnej miery porovnateľná s portálovým riešením no je charakterizovaná modernejšími agregačnými a prezentačnými technológiami(AJAX) známymi pod označením Web 2.0 . Združený obsah pochádza od tretích strán získaný verejným rozhraním alebo API. Medzi ďalšie možné zdroje môžeme zaradiť webové služby (SOUP a REST), RSS a ATOM z ktorých sú dáta získavané automatizovanou extrakciou.

Tabuľka 1. - Porovnanie Portálu a Mashup aplikácie

	Portál	Mashup
Klasifikácia	Staršia technológia, rozšírenie štandardného modelu dynamického webu.	Novšia 2.0 Web technológia
Filozofia/Prístup	Rozdeľuje agregáciu do dvoch častí, generovanie značkovania a agregáciu označkovaných fragmentov	Adoptuje fundamentálnejší spôsob k agregácii obsahu bez ohľadu na značkovanie.
Obsahové závislosti	Agreguje prezentačne orientované značkovacie fragmenty(HTML, WML, VoiceXML, ...)	Dokáže operovať nad obsahom popísaným XML súbormi a takisto aj prezentačne orientovaným obsahom(AJAX, XHTML)

2.3 Portálový rámec

Portálový rámec predstavuje kostru pre zoskupenie konkrétnych aplikácií za účelom dosiahnutia doménovo - orientovanej služby. Konfigurovateľnosť rámca odráža základné kvality a možnosti rámca k dosiahnutiu doménových špecifík portálového riešenia. Konfigurácia spočíva v spôsobe zoskupenia aplikácií, ktoré poskytujú požadovanú dielčiu funkcionalitu. Komunikácia použitých aplikácií v rámci portálového riešenia je tiež jedna z vlastností, ktorá pridáva nielen na kvalite a vzvyšuje funckionalitu celého riešenia ale pridáva aj na efektívnosti práce s portálom. Pre zjednodušenie konfigurovateľnosti a zvýšenie modularity riešenia, budovaného na aplikačnom rámci je dôležité dosiahnuť nezávislosť jednotlivých aplikácií.

Považujeme za výhodne realizovať stavbu aplikácie formou modulov. Moduly poskytujú vysokú flexibilitu pri tvorbe samotnej aplikácie, keďže sa jedná v podstate o nezávislú časť reprezentujúcu čiastkovú logiku. Je potrebné navrhnúť moduly tak aby ich bolo možné zoskupiť a tým dosiahnuť požadovanú funkcionalitu aplikácie. Tento spôsob tvorby aplikácie v značnej mieri podporuje znovu použiteľnosť a pridáva systému na modularite a takisto v neposlednej rade je možné týmto spôsobom splniť požiadavku klienta o zásuvných moduloch.

2.4 Analýza existujúceho riešenia – portál Cocoon

Publikačný rámec Cocoon je voľne dostupná komponentovo založená implementácia v jazyku Java, ktorá poskytuje prostriedky pre dynamické generovanie dokumentov prostredníctvom definovaných zreťazení. Jednotlivé zreťazenia sú definované centrálne, pričom dáta v zreťazení produkuje generátor. Následne môžu byť transformované sériou transformátorov a nakoniec poskytnuté na výstup v cieľovom formáte prostredníctvom serializátora. Každé zreťazenie obsahuje práve jeden generátor, nula a viac transformátorov a práve jeden serializátor.

Cocoon poskytuje niekoľko štandardných generátorov, transformátorov a serializátorov, nazývaných komponenty a tiež implementačné rozhranie pre tvorbu vlastných komponentov. Najčastejšie sú implementované vlastné akcie, ktorých úlohou je zmena stavu systému, mnohokrát využívané ako prostriedok pre zmenu toku riadenia. Okrem toho Cocoon poskytuje možnosti znovupoužitia definovaných zreťazení pomocou zdrojov, ako aj pohľady určené pre sledovanie dát prúdiacich medzi jednotlivými transformátormi, bez nutnosti zmeny definície zreťazenia. Existujúce riešenie, z optimalizačných dôvodov, nie je použiteľné pre reálnu prevádzku.

2.5 Transformácie XML/XSLT/XHTML

Dátová transformácia predstavuje konvertovanie dát poskytnutých zdrojom postupnosťou krokov do požadovanej formy. Konfigurovateľné transformácie sú vhodný prostredok pre portálový rámec na spracovanie vstupov a výstupov založených na XML, XSL prípadne XHTML. Práve z dôvodu početnosti transformácií je výhodné použiť modul, ktorý dokáže spracovať XML v nami vyžadovanej miere. Jednou z možností je modul z aplikačného rámca .netnuke.

Modul XML má nasledujúce vlastnosti

- čítať XML súbory z viacerých zdrojov
- transformovať XML pomocou XSL
- generovať XML, prípadne XHTML

Znovupoužitie modulu v nami navrhovanom systéme dokáže zrýchliť vývoj a vyhnúť sa vytváraniu a dolaďovaniu vlastného riešenia pre spracovanie XML.

2.6 Integrované vývojové prostredie

Z hľadiska efektívnej tvorby softvérových systémov je dôležité použitie integrovaných vývojových prostredí, ktoré poskytujú všetku potrebnú funkcionalitu prostredníctvom jedného rozhrania. Ideálne takéto prostredie tiež podporuje spoluprácu s rôznymi ďalšími nástrojmi na riadenie verzií súborov a generovanie dokumentácie.

2.6.1 Visual Studio 2005

Integrované vývojové prostredie Visual Studion 2005 podporuje efektívnu a pohodlnú tvorbu softvéru. Prostredie je primárne určené na vývoj aplikácií na platforme .NET. Základné prostredie neponúka veľké množstvo nástrojov, je však možné ho podľa potreby rozšíriť o množstvo zásuvných modulov.

2.7 Zdieľanie výsledkov práce členov tímu

Pri riešení projektu, na ktorom pracuje väčší počet ľudí, treba vyriešiť problém zdieľania spoločných súborov, či už ide o zdrojové kódy, dokumentáciu alebo iné súbory. Základným problémom je spôsob zdieľania informácií používateľmi bez toho, aby si navzájom ničili prácu neustálym prepisovaním svojich zmien.
Copy-modify-merge model umožňuje každému používateľovi prácu nad jeho osobnou kópiou súborov z úložiska a následné spájanie týchto kópií do novej verzie v úložisku. Proces spájania môže byť do určitej miery podporovaný systémom na správu verzií, hlavná zodpovednosť však leží na používateľovi.

2.7.1 Microsoft Visual Source Safe

SourceSafe je systém na kontrolu verzií ľubovoľnej kolekcie súborov. Umožňuje efektívne zdieľanie súborov cez vývovojový nástroj Visual Studio (prípadne priamo cez Visual SourceSafe) a poskytuje vysokú bezpečnosť uloženia súborov tým, že si pamätá každú vykonanú zmenu.

Medzi hlavné prednosti nástroja SourceSafe patria:

- Posielanie difference (rozdielov medzi súbormi) v oboch smeroch.
- Každá zmena zvyšuje číslo verzie, nielen zmena obsahu súborov.
- Plne podporovaný nástrojmi pre vývoj pod platformou .NET

2.8 Zhrnutie

Cielom tejto časti bolo analyzovať portálové riešenie a prostriedky vhodné pre vývoj zadaného portálového rámca. Veľkú časť inšiprácie sme čerpali z aplikačného rámca Cocoon, ktorý rieši stavbu aplikácie pre portálový rámec formou zreťazených modulov, kde každý modul prezentuje transformačnú časť dátového toku. Takisto tento rámec trpí neprehľadnou formou konfigurácie a postráda autorizáciu viacerých používateľov pre prácu s aplikáciami. Ďalším nemalým obmedzením je slabšia výkonosť pri nasadení v prevádzke. Našim cielom je navrhnúť portálový rámec, ktorý odstráni tieto nedostatky a splní všetky požiadavky klienta. Taktiež sme analyzovali spôsob transformácie dát pri spracovaní modulmi, no keďže nebola vykonaná analýza do značného detailu navrhujeme vo fáze prototypovania overiť správanie modulu .netnuke a v prípade nenaplnenia požiadaviek navrhujeme využitie natívnych prostriedkov.

3. Špecifikácia požiadaviek

Táto časť dokumentu bližšie opisuje nároky a požiadavky klienta na funkcionalitu, používateľov a ohraničenia navrhovaného portálového rámca. Požiadavky rozdelíme na funkcionálne a nefunkcionálne a v rámci tohto rozdelenia sú požiadavky rozdelené aj tématicky.

3.1 Funkcionálne požiadavky

3.1.1 Spracovanie údajov

Spracovanie XML

Systém musí vedieť spracovávať dáta vo formáte XML (prípadne OWL), ktoré budú využívať jednotlivé moduly systému ako vstupné a výstupné dáta. Spracovanie XML dát bude založené na vzore dátovody a filtre, čo znamená, že modul, ako jedna časť dátovodu, bude prijímať, transformovať a posielať XML dáta ďalšiemu modulu na spracovanie.

Tok a transformácia údajov v jednotlivých dátovodoch na báze XML

Všetky dáta, prenášané medzi jednotlivými modulmi (spojenými do dátovodov), sú vo formáte XML. Každý modul transformuje dáta podľa vlastných pravidiel a pošle dalej na spracovanie.

Výstup procesu transformácie v rôznych formátoch

Výstupom procesu transformácie bude súbor rôzneho formátu (napr. XML, XHTML, PDF a iné), ktorý bude zobrazený používateľovi, prípadne ponúknutý na stiahnutie. Formát výstupu bude závisieť od konfigurácie dátovodu.

3.1.2 Modularita

Integrácia viacerých aplikácií do jedného používateľského rozhrania

Pod portálovým riešením sa predstavuje zoskupenie viacerých aplikácií v používateľskom rozhraní, ktoré poskytujú rozdielnu funkcionalitu. Vytvorenie rozhrania a jeho modifikácia závisí len od konfigurácie rámca bez nutnosti modifikovania a opätovného prekladania

zdrojových súborov. Konfigurácia sa odkazuje na moduly (zásuvné moduly), z ktorých sa skladajú jednotlivé aplikácie. Moduly je možné v prípade potreby nahradiť prípadne modifikovať nezávisle od portálového riešenia.

Podpora komunikácie jednotlivých aplikácií

Pri portálovom riešení je časté vytváranie používateľského rozhrania z aplikácií ktoré poskytujú rozdielnu funkcionalitu, no so zameraním na jednu doménovú oblasť. Aplikácie spadajú do jednej doménovej oblasti a preto je žiadaná komunikácia medzi nimi za účelom zefektívnenia celkovej funkcionality portálového riešenia. Pod komunikáciou sa predstavuje zmena výstupu jednej aplikácie na základe ovplyvnenia inou aplikáciou v rámci jedného portálového riešenia.

Znovupoužitie zásuvných modulov

Keďže portálové riešenie je tvorené viacerými aplikáciami a aplikácie sú tvorené z viacerých modulov, je žiadané znovupoužitie modulov za účelom obmedzenia ich duplicity, čím dôjde k sprehľadneniu a zjednoteniu konfigurácie a správy portálu.

Pridávanie nových modulov do systému bez nutnosti zásahu do portálového rámca

Portálový rámec predstavuje riešenie, z ktorého podstaty vyplýva znovupoužiteľnosť a nasadenie v rôznych sférach použitia. Z mnohonásobného využitia a potreby údržby vyplýva potreba konfigurácie portálového riešenia a možnosť pridania, odobratia, modifikovania modulov bez zásahu do samotného rámca.

3.1.3 Personalizácia a bezpečnosť

Podpora záznamu informácií o interakcii používateľa so systémom

Pre zaistenie spoľahlivosti a správnej funkčnosti portálového riešenia je potrebné vytvárať záznam informácií o interakcii používateľa so systémom. Záznam súhrnne oboznamuje správcu systému o dianí v rámci portálového riešenia a funkčnosti portálového rámca, čo umožňuje indikáciu možných problémov a ich včasné predchádzanie, prípadne okamžitú reakciu na vzniknuté problémy. Záznam interakcií používateľa so systémom takisto výrazne napomáha v oblasti bezpečnosti systému a potenciálnej hrozby so strany používateľa.

Zadefinovanie viacerých používateľských rolí s odlišným rozsahom právomocí v systéme

Z podstaty portálového riešenia a jeho rôznorodej funkcionality, ktorú poskytuje, je nutné sprostredkovať možnosť zadefinovanie používateľských skupín a ich právomocí v systéme.

Personalizácia používateľského rozhrania portálu

Portálový rámec bude sprístupňovať možnosť personalizácie používateľského rozhrania formou rozmiestnenia samotných aplikácií. Táto funkcionalita umožňuje modifikáciou používateľského rozhrania podľa osobných preferencií čo používateľovi spríjemní a zefektívni prácu so systémom.

3.2 Nefunkcionálne požiadavky

3.2.1 Konfigurovateľnosť portálového rámca

Portálový rámec by mal využívať prístupy webu so sémantikou. Rámec by mal byť ľahko a prehľadne konfigurovateľný na základe konfiguračného súboru. Konfigurácia bude obsahovať nastavenia celého portálového rámca a všetkého, čo súvisí s behom a prevádzkou portálového rámca. Takisto budú konfigurovateľné samotné aplikácie, prevádzkované na portálovom rámci.

3.2.2 Podpora bezpečnosti – prihlasovanie sa používateľov, šifrovaná komunikácia

Systém bude podporovať prihlasovanie sa používateľov a šifrovanie komunikácie na zabezpečenie dôveryhodnosti a ochrany pred zneužitím dát tretími stranami. Zabezpečenie komunikácie sa vyžaduje od momentu autentifikácie používateľa (prihlásenie sa) až po ukončenie jeho práce (ohlásenie sa).

3.2.3 Softvérová platforma

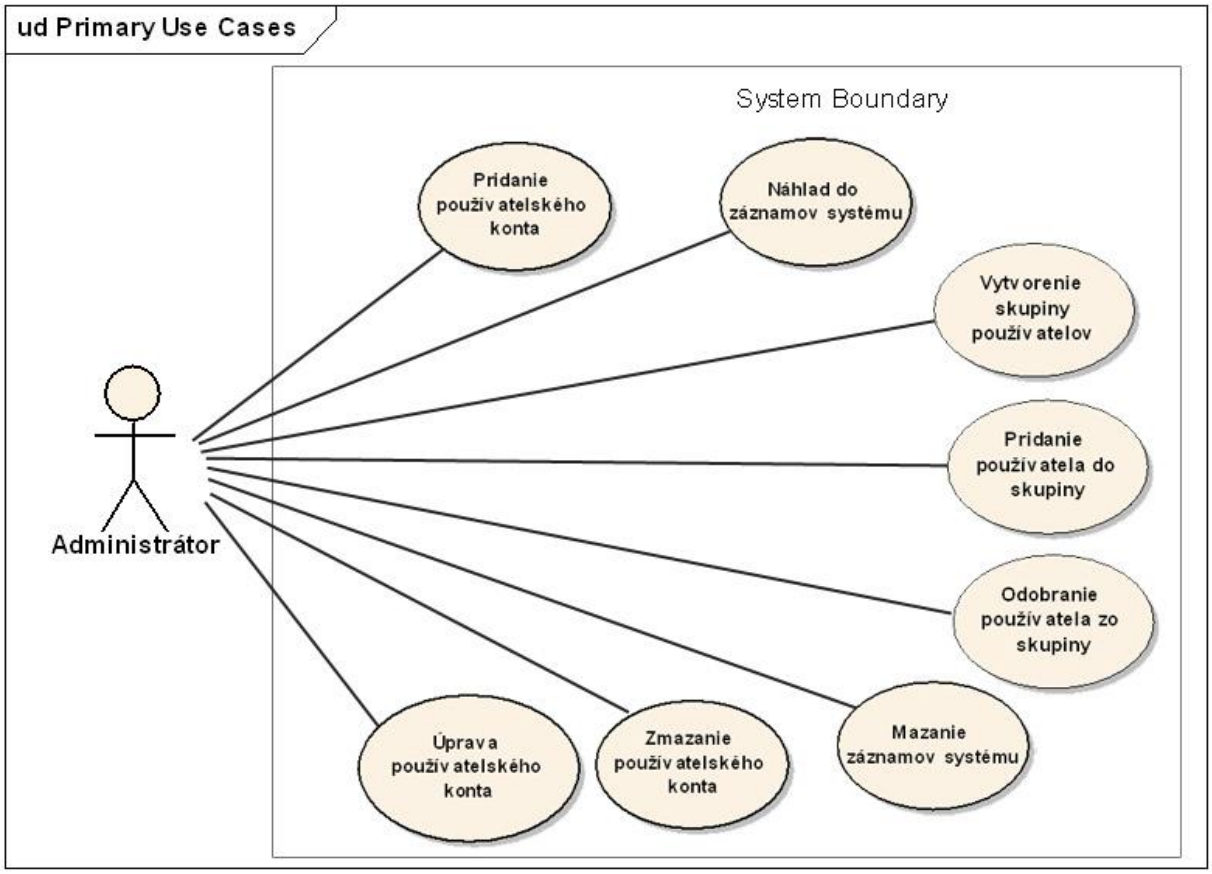
Portálový rámec je určený pre systém Windows s podporou .NET framework .

3.3 Používateľské role

V tejto časti dokumentu sú načrtnuté požiadavky na používateľov systému. Používateľom budú pridelené role so špecifickými funkciami, vlastnosťami a oprávneniami a tieto budú bližšie opísané práve v tejto časti dokumentu.

3.3.1 Administrátor

Administrátor je rola používateľa s najširšími možnosťami a oprávneniami.



Obrázok 1. - Model prípadov použitia role administrátor

Táto rola a všetky jej prislúchajúce funkcie a možnosti sú znázornené na obrázku č.1 a neskôr aj podrobnejšie opísané.

Správa používateľov

Administrátor má k dispozícii prehľad používateľov rámca a môže vytvárať, upravovať a mazať používateľské kontá portálového rámca.

Správa štruktúry a oprávnení pracovných skupín

Administrátor má možnosť vytvárať skupiny používateľov ako aj zaraďovať jednotlivých používateľov do týchto skupín. Používateľské skupiny slúžia na kategorizáciu používateľov a prispôsobenie vzhľadu aplikácie podľa potrieb a požiadaviek používateľa. Používateľské skupiny budú mať určitú sadu oprávnení, ktoré zadefinuje práve administrátor portálového rámca. Tieto oprávnenia používateľských skupín sa budú vzťahovať na jednotlivé aplikácie systému. Je potrebné,

aby používateľ nebol obťažovaný zobrazením aplikácií, ktoré ku svojej práci nepotrebuje, alebo ich dokonca nemá oprávnenie vidieť. Jeden používateľ môže byť zaradený do viacerých pracovných skupín a rovnako mu bude možné prideliť viacej používateľských rolí. Skupiny ani používateľské role nebudú organizované hierarchicky.

Prideľovanie používateľských rozhraní používateľským skupinám

Pod pojmom rozhranie rozumieme umiestnenie aplikácií na stránke spolu s grafickým spracovaním stránky ako celku. Administrátor musí mať prehľad o používateľských rozhraniach a bude mať možnosť presne zadefinovať aký typ používateľského rozhrania pridelí konkrétnej používateľskej skupine.

Správa a prehľad notifikácií

Je potrebné zabezpečiť odozvu systému prostredníctvom tzv. log súborov, alebo notifikácií, ku ktorým bude mať prístup administrátor. Jedná sa o informatčné údaje, ktoré slúžia na poskytnutie informácií o stave portálového rámca. Tieto údaje budú obsahovať informácie napr. o fyzickom vyťažení, chybách vo funkčnosti a bezpečnosti aplikácií ako aj rámca, výpadkoch zdrojov dát v jednotlivých aplikáciách. Budú uložené formou záznamu v databáze, alebo ako súbor na disku počítača. Aby sa tieto záznamy nemnožili, musí byť administrátorovi umožnené mazanie týchto súborov a notifikácií.

3.3.2 Dizajnér

Dizajnér aplikácie, ktorý vytvára počiatočnú podobu aplikácie.
Dizajnér je zodpovedný za

- Prvotný grafický návrh systému
- Vytvorenie grafickej identity aplikácie a šablón
- Rozmiestnenie grafických komponentov na stránke
- Zosúladenie grafických požiadaviek vytváranej aplikácie pre cieľovú skupinu
- Vytvorenie grafickej podoby portálového rámca, bez prvotnej funkcionality

3.3.3 Vývojár(programátor)

Vývojár (programátor) má na starosti vývoj modulov a je zodpovedný za :

- Samotnú funkčnosť modulov ako aj ich implementáciu (nie je zodpovedný len za funkčnosť, ale aj za dodržanie princípov vývoja v portálovom rámci)
- Komunikáciu modulov (moduly musia byť vytvorené tak aby sa dali v prípade nutnosti integrovať s inými)
- Prevádzku systému na testovacích dátach vo vývojovom prostredí a za plynulé nasadenie do produkčného prostredia portálového rámca

3.3.4 Návštevník

Návštevník portálového rámca má možnosť využívania iba vymedzenej funkcionality a to práve tej, ktorú mu vývojár spolu s dizajnérom ponúkli. Funkcionalita a interakcia závisí od konkrétnej aplikácie.

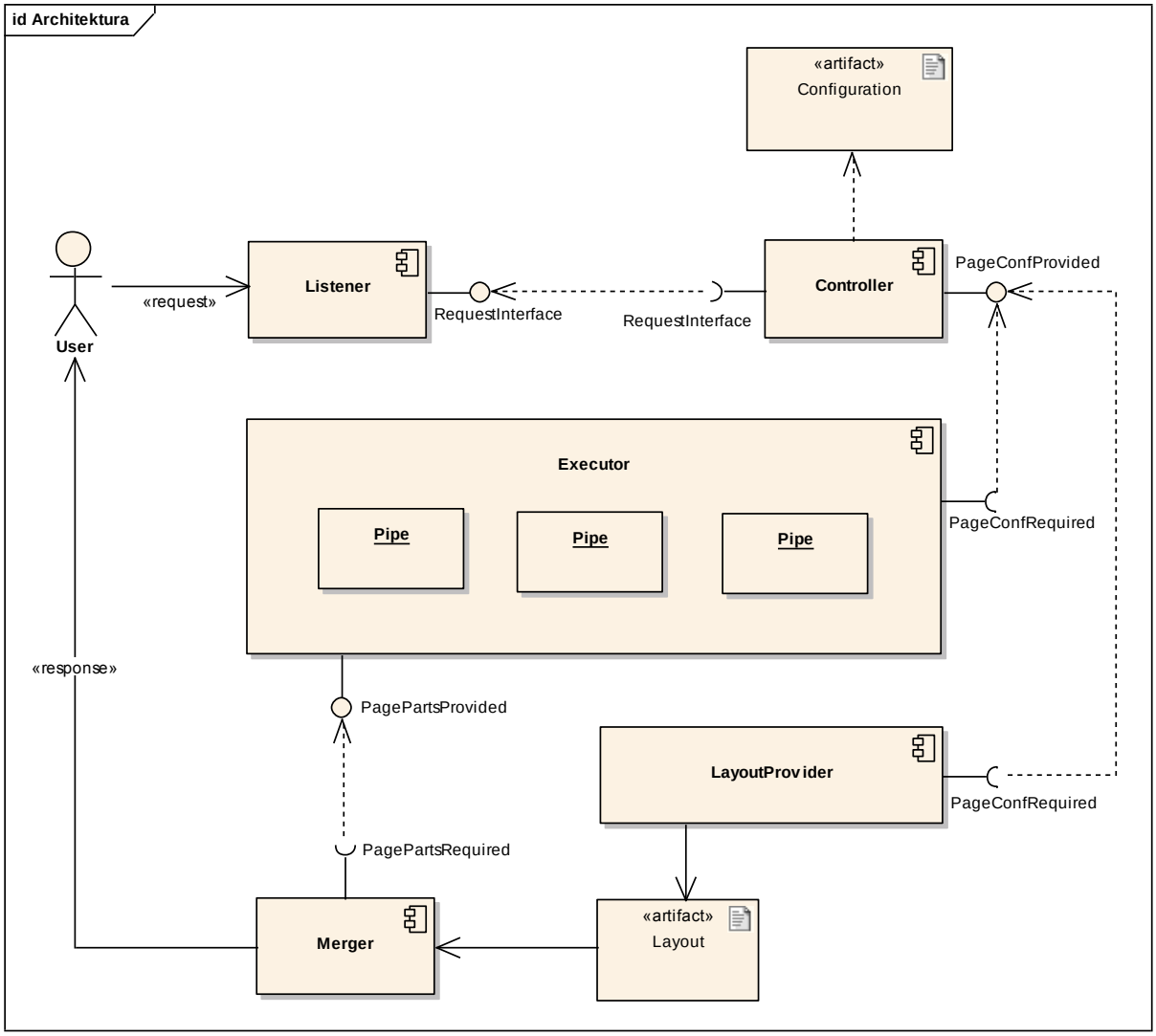
4. Hrubý Návrh riešenia

Kapitola bližšie popisuje architektonický návrh portálového rámca. Je vedená vo forme prezentácie všeobecného náhľadu na celkový systém, ktorý upresňuje pomocou dekompozície na jeho jednotlivé autonómne súčasti.

Na základe zadania sme sa rozhodli rozčleniť celý systém na dve hlavné časti a to na serverovú časť a na klientsku časť. Prvá časť kapitoly je venovaná serverovej časti navrhovaného riešenia, teda popisuje samotné jadro portálového rámca od návrhu jeho jednotlivých komponentov cez ich vstupno-výstupné rozhrania až po tok údajov medzi nimi. Za ňou nasleduje priblíženie prezentačnej časti riešenia na strane klienta, konkrétne popis používateľského rozhrania podporujúceho personalizáciu vzhľadu a rozmiestnenia komponentov na vygenerovanej HTML stránke. Poslednú podkapitolu tvorí opis logického modelu údajov.

4.1 Architektúra systému

Vychádzajúc zo zadania a špecifikácie bol systém navrhnutý na báze vzoru dátovody a filtre. Vygenerovanie obsahu stránky, typicky obsahujúcej viacero aplikácií je teda dekomponované na generovanie obsahu poskytovaného jednotlivými aplikáciami prostredníctvom rôznych dátovodov (zabezpečuje Executor). Tomuto procesu prirodzene predchádza spracovanie požiadavky (Listener), jej namapovanie na poskytovaný obsah a vyhodnotenie prístupových oprávnení (Controller). Na záver procesu sa vyberie vhodný (prednastavený alebo používateľom prispôsobený) vzhľad a výstupy aplikácií sa do neho zasadia a prezentujú používateľovi (Merger). Schematicky toto rozloženie funkcionality zobrazuje Obrázok 2 : Architektúra systému.



Obrázok 2. – Architektúra systému

4.2 Opis funkcionality komponentov serverovej časti

4.2.1 Komponent „Listener“

Komponent systému zodpovedajúci za preberanie požiadaviek (*requestov*) od používateľov, ich vhodnú transformáciu a odovzdaniu Controlleru prostredníctvom vhodného rozhrania.

Keďže vytvárané riešenie je portálový rámec pre webové aplikácie, príjem požiadaviek bude prebiehať prostredníctvom štandardného protokolu http, resp. https. Po prijatí požiadavky nasleduje vyhodnotenie jej formálnej správnosti, vhodná dekompozícia, prevedenie do interného formátu akceptovaného na rozhraní s Controllerom (založené na XML) a odovzdanie ďalej do systému.

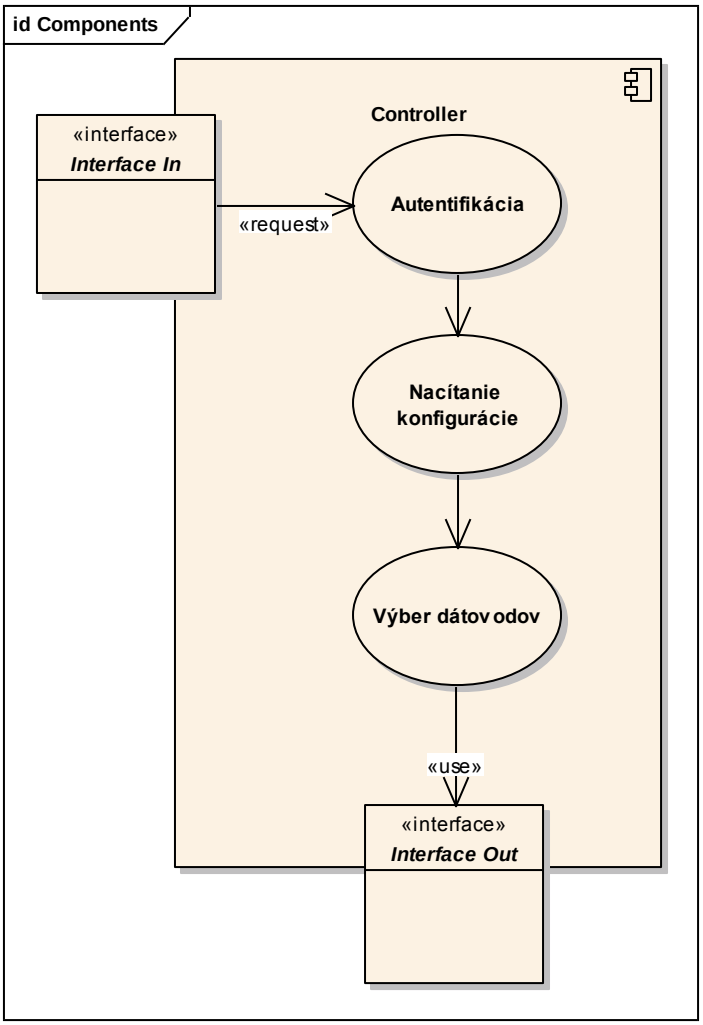
Implementácia komponentu je v súčasnej fáze návrhu systému otvoreným problémom, pričom pripadajú do úvahy dve hlavné alternatívy :

1. Implementácia vlastného listenera s vhodným http parserom. Toto riešenie by samozrejme vyžadovalo viac času i ľudských zdrojov, tak na samotnú implementáciu, ako aj nevyhnutnú prípravu – naštudovanie http protokolu, nižších sieťových služieb a rozhraní .NET frameworku atď. Na druhej strane toto úsilie by prinieslo riešenie „šité na mieru“, čiže pomerne efektívne a šetrné k zdrojom, otvorené a tým pádom podľa potrieb rozšíriteľné. Alternatívne je možné použiť komponentu tohto typu už hotovú, ak bude dostupná priamo v rámci frameworku, príp. mimo pod vodnou licenciou.

2. Nasadenie prispôsobeného http servera. V rámci platformy prichádza do úvahy hlavne MS IIS. Možnosť jeho efektívneho použitia v roli listeneru bez potreby následnej aplikačnej logiky je však zatiaľ nepreskúmanou doménou. Taktiež by toto riešenie bolo takmer určite relatívne neefektívne z hľadiska systémových zdrojov a požiadaviek na nasadenie – povinná prítomnosť MS IIS výrazne zvyšuje požiadavky na cieľový systém.

4.2.2 Komponent „Controller“

Controller je komponent zodpovedný za vyhodnotenie požiadavky v kontexte konfigurácie portálu a používateľských práv pristupujúceho užívateľa. Výstupom tohto procesu je úplná informácia pre Executor ohľadne všetkých potrebných dátovodov generujúcich výstupy pre požadovanú stránku.



Obrázok 3. – Contoller

Zjednosušene možno povedať, že Controller mapuje požiadavky na príslušné stránky portálu pozostávajúce z výstupov viacerých dátovodov. Pri tomto zároveň uplatňuje definované obmedzenia užívateľov (resp. používateľských skupín) na jednotlivé aplikácie.
Tento proces je schematicky zachytený na Obrázku 3 – Controller.

Vstupmi Controllera sú teda request na rozhraní zo strany Listenera, konfigurácia portálu v zmysle „mapy stránok“ (opäť sa predpokladá formát založený na XML), a konfigurácia používateľských oprávnení vo vzťahu k jednotlivým aplikáciám a dátovodom (tu sa z dôvodu väčšej flexibility predpokladá riešenie založené na relačnej databáze).
Výstup Controller poskytuje prostredníctvom rozhrania pre komponenty Executor a LayoutProvider, ktoré ďalej využívajú tieto informácie na spustenie samotných dátovodov generujúcich obsah, resp. výber príslušného layoutu pre danú stránku a použité dátovody.

4.2.3 Komponent „Executor“

Tento komponent bol navrhnutý na zapuzdrenie logiky pre vytváranie inštancií premenlivého počtu dátovodov, ktoré bude schopný paralelne spúšťať. Executor dostáva vstupné údaje z Controlleru vo formáte XML. Tieto údaje budú povinne zahŕňať počet dátovodov, ktoré majú byť spustené, a ich typy. Vstupné údaje môžu zahŕňať aj inicializačné hodnoty pre vybrané dátovody, prevažne v prípadoch interakcie používateľa s nejakou vybranou aplikáciou, ktorú bude prevádzkovať výsledný portál.

Po prijatí vstupných údajov vyberie Executor na základe prednastavenej konfigurácie dátovody, ktoré je potrebné spustiť ako prvé, nakoľko dodávajú vstupné údaje zvyšným dátovodom (ak je takáto závislosť potrebná a korektne zadefinovaná). Poradie aktivovania bude presne definované ku každému typu používateľskému vstupu, aby sa tak vyhlo nejednoznačnostiam pri kooperatívnom spracovaní údajov. Executor bude obsahovať riadiacu jednotku, ktorá bude schopná aktivovať vybrané dátovody po skupinách na základe priority prislúchajúcej skupiny.

Po spustení jednotlivých dátovodov bude Executor sprostredkovávať rozhranie podporujúce komunikáciu medzi nimi. Toto rozhranie bude opísané v samostatnej časti.

Po ukončení činnosti jednotlivých dátovodov sú vysielané ich výstupné údaje v serializérmi určenom formáte do naledujúceho komponentu v poradí, teda do Mergeru.

4.2.4 Komponent „Merger“

Merger predstavuje kontajner, ktorý čaká na prijatie výstupných údajov jednotlivých dátovodov Executoru.

Po prijatí dát z každého dátovodu zodpovedajúceho vybranému používateľskému dopytu je pripravený na ich finálnu úpravu do formátu, ktorý bude odoslaný používateľovi. Predpokladajú sa dve formy výstupov:

- Výstup vo forme webovej stránky
- Výstup vo formáte súboru na stiahnutie

Keďže v drvivej väčšine prípadov bude používateľovi výsledného portálu odosielaný výstup vo formáte XHTML, bude mať Merger naimplementované funkcie na spojenie XML údajov z jednotlivých dátovodov do jediného finálneho XHTML dokumentu, teda do jednej webovej stránky. Na tento účel bude využívať administrátorom a dizajnérom navrhnuté preddefinované šablóny, na základe ktorých zaradí obsahy výstupov dátovodov na im určené miesto v dokumente. Táto operácia bude zároveň obsahovať aj priradenie správneho CSS

štýlu vygenerovanému dokumentu. V prípade, že výsledná stránka bude mať preddefinované aj dynamické oblasti, bude do nej pripojený aj kód API rozhrania pre úpravu používateľského rozhrania. Ak bude šablóna dokumentu navrhnutá staticky, kód doplnený nebude, nakoľko sa jedná o administrátorsky nastaviteľnú zložku portálu.

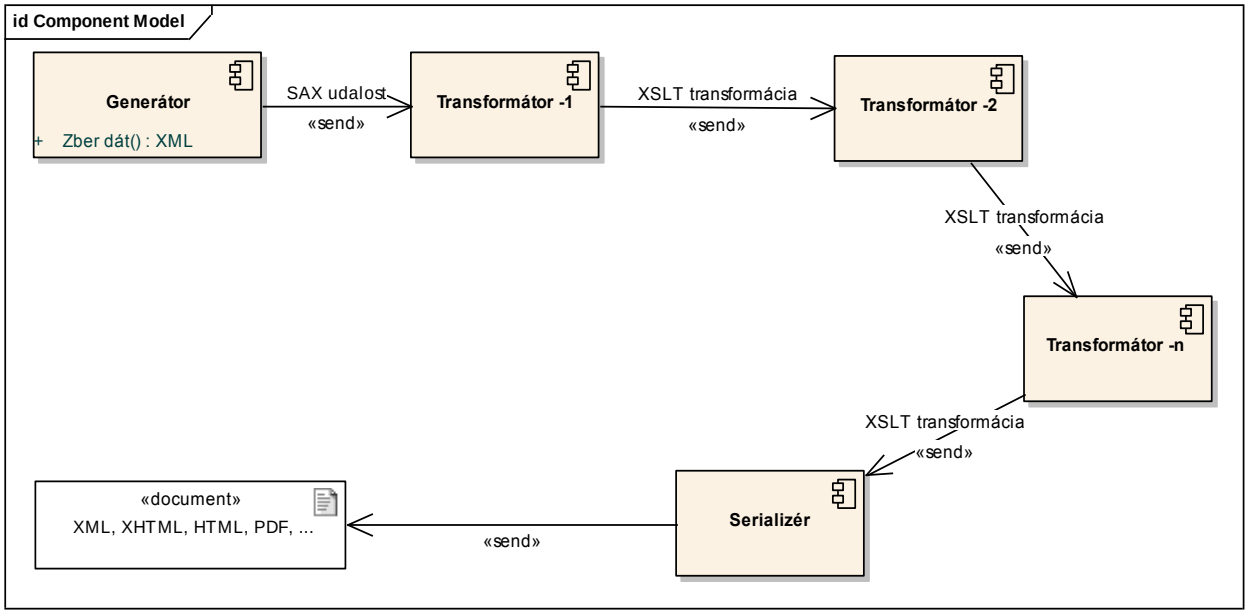
V prípade, že z Executora bude vyslaný prúd údajov v inom formáte ako je XML, bude celý tento prúd presmerovaný na používateľský výstup bez akýchkoľvek zmien. V takomto prípade sa predpokladá prúd údajov iba z jediného dátovodu.

4.3 Štruktúra a funkcionalita dátovodu

Vo fáze návrhu boli identifikované tri nosné časti dátovodu:

a) Generátor (zberač dát)
b) Transformátor (konverzia dát)
c) Serializér (export dát)

Celý proces toku údajov cez dátovod je zobrazený na Obrázku 4.



Obrázok 4. – Generovanie čiastkového výstupu jedným dátovodom

Opis jednotlivých fáz

A. Generátor zozbiera všetky potrebné dáta do XML formátu a prostredníctvom SAX udalostí ich postupne posiela prvému transformátoru.
B. Prvý transformátor v poradí pretransformuje získaný XML prúd dát pomocou implementovanej transformácie (napr. XSLT) na pozmenený tok SAX udalostí, ktorý

je zároveň vstupom pre druhý transformátor vo fronte. Takáto postupnosť krokov sa opakuje až po posledný transformátor v rade.

C. Posledný transformátor v dátovode pošle svoj výstup na vstup serializéru. Tento prvok vyexportuje prijatý prúd dát do výsledného želaného formátu, ktorým može byť samotný XML, XHTML, PDF resp. iný, dokument.

Spracovanie XML pomocou SAX rozhrania

SAX (Serial Access Parser) poskytuje vhodný mechanizmus pre čítanie dát z XML dokumentov. Je to alternatíva ku DOM (Document Object Model) rozhraniu. Hlavné charakteristiky oboch rozhraní sú:

Porovnanie SAX a DOM rozhrania

SAX

- Model založelný na udalostiach. Je obzvlášť vhodný pre veľké dokumenty. Výhoda spočíva v tom, že program stále spracúvava iba malú porciu dát. Tieto udalosti zahŕňajú začiatok a koniec dokumentu, nájdenie uzla a elementov potomkov uzla a pod.
- Pre implementáciu SAX parseru je zvyčajne potrebné napísať väčšie množstvo kódu než u DOM parserov. Výsledkom je však vlastný, priamo použiteľný objektový model dát

DOM

- Model založený na stromovej hierarchii dát. Je vhodný skôr pre menšie dokumenty, kedy program spracúva väčšiu porciu dát, nakoľko je generovaný celý objektový model (strom) pre daný dokument. V prípade potreby špecifického objektového modelu alebo len malej časti dokumentu, je tento spôsob pomerne neefektívny.

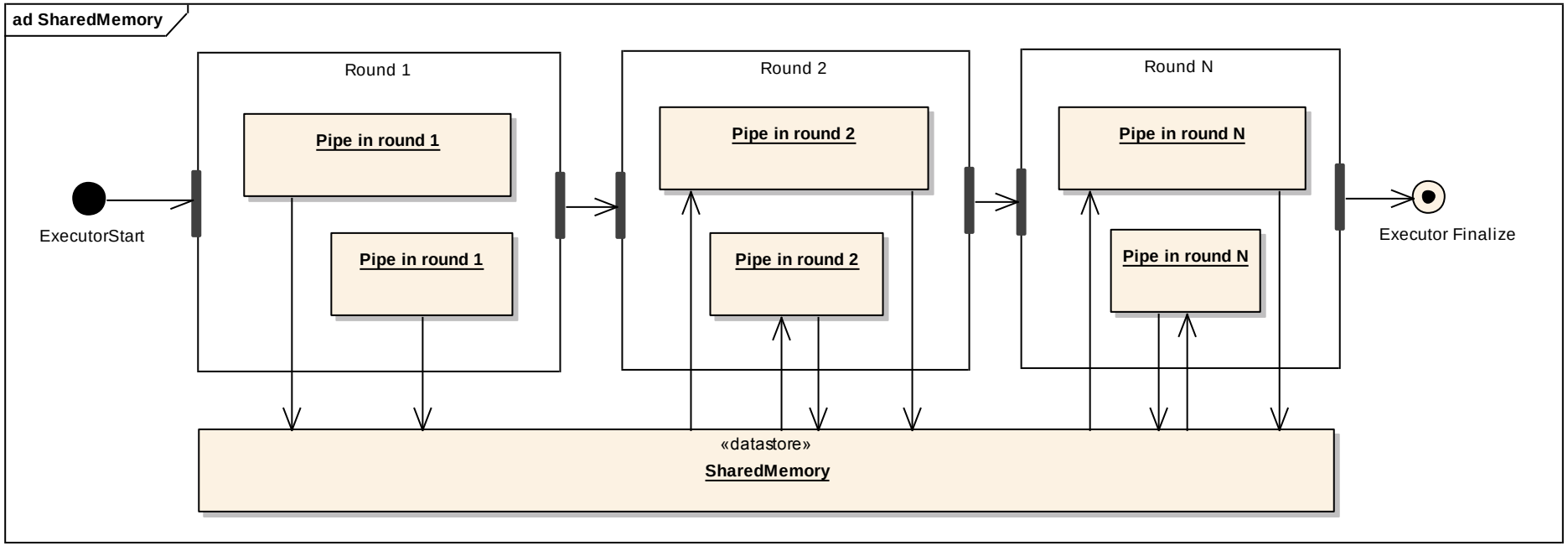
Výber rozhrania

Z daného opisu oboch prístupov usudzujeme, že rozhranie SAX aj napriek mierne vyššej náročnosti na implementáciu predstavuje pre naše potreby prijateľnejšie a pre konečný výkon výhodnejšie riešenie, pretože nami navrhovaný systém nevylučuje prácu s rozsiahlymi XML

vstupmi, a preto predstavuje lepšie riešenie hlavne z hľadiska optimálneho výkonu, nakoľko nespotrebúva až tak veľa pamäťových prostriedkov a ani procesorového času, ako rozhranie DOM.

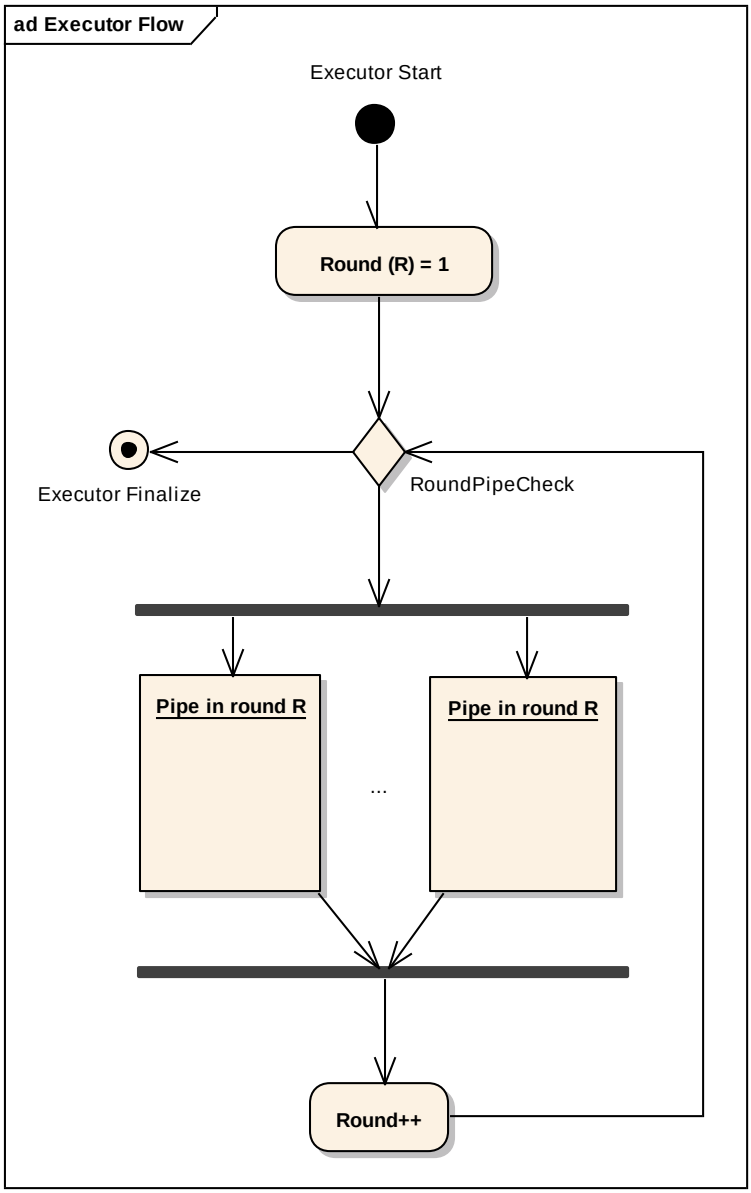
4.4 Komunikácia modulov

Analýzou možností, pre ktoré je potrebné implementovať komunikáciu modulov sme zistili, že jediným užitočným prípadom využitia tejto funkcionality je variabilná závislosť poradia spracovania vstupných údajov jednotlivými modulmi v tom zmysle, že vstupné údaje niektorých modulov môžu byť v špecifických prípadoch závislé na výstupe iných modulov bežiacich pri spracúvaní tej istej používateľskej požiadavky. Preto sme dospeli k záveru, že komunikácia modulov bude prebiehať na úrovni údajov, čím sa zjednoduší návrh aplikácie, pretože sa z neho vypustí réžia kooperácie modulov. Nakoľko bude Executorom upravené poradie spúšťania jednotlivých dátovodov, modulom bude postačovať asynchrónna komunikácia pomocou využívania zdieľanej pamäte. Tento princíp je obsahom Obrázku 6. Tento ilustruje modelový prípad vykonávania jednotlivých dátovodov v kolách („round“), v rámci ktorých Executor spúšťa dátovody nakonfigurované pre dané kolo („Pipe in round X“ – v obrázku sú pre každé kolo znázornené dve, v praxi však samozrejme bude možné používať ľubovoľné množstvo). Obrázok 6. však predovšetkým ilustruje asynchrónnu komunikáciu údajovo závislých dátovodov – pri správnej konfigurácii (zaradenia do kôl) majú závislé dátovody vždy potrebné (či dobrovoľne využiteľné) dáta v zdieľanej pamäti, kam ich predtým zaradili dátovody bežiace v predošlých kolách.



Obrázok 6. – Zdieľaná pamäť dátovodov

Dátovody spustené v skorších kolách budú pripravovať vstupné údaje následne aktivovaným závislým dátovodom, čím sa zaistí požadované poradie spracovania vstupných údajov a teda aj korektný výstup. V prípade viacnásobných závislostí bude tento proces vykonávaný postupne po krokoch (obrázok č. 7). Následnosť krokov a teda aj zadefinovanie aktivácie modulov vo vybraných kolách procesu spracovania údajov bude napevno zadané v konfigurácii portálového rámca, v časti definujúcej spracovanie používateľských požiadaviek.



Obrázok 7. – Tok údajov riadený Executorom

4.5 Údaje v systéme

Údaje používané portálovým rámcom sa budú rozdeľovať na tri skupiny, a to z hľadiska ich charakteru, ako aj z hľadiska spôsobu ich uloženia. Portálový rámec bude na ukladanie nastavení používať konfiguračné súbory. Na ukladanie používateľov, používateľských skupín a oprávnení na jednotlivé aplikácie sa predpokladá použitie relačnej databázy. Tretiu skupinu údajov používaných v navrhovanom riešení predstavujú systémové notifikácie.

4.5.1 Konfiguračné súbory

Nastavenia portálového rámca budú ukladané v súboroch vo formáte XML. Tento formát sme zvolili kvôli jednoduchosti jeho modifikácie v prípade potreby a taktiež kvôli jeho vysokej flexibilite, pretože je pomocou neho možné modelovať vzťahy medzi údajmi, ktoré sú v tomto formáte uložené.
Konfiguračné súbory budú rozdelené do nasledujúcich celkov:

- Mapa portálu (sitemap)
- Evidencia údajových úložísk
- Konfigurácie aplikácií
- Dizajnérske šablóny

Mapa portálu

Sitemap bude obsahovať základné nastavenie portálového rámca, ktoré bude využívane Controllerom. Hlavnou jeho zložkou bude zoznam aplikácií prevádzkovaných samotným portálom vybudovaným na navrhovanom portálovom rámci, teda zoznam jednotlivých dátovodov, ktoré ich predstavujú. Záznam o dátovodoch bude obsahovať jeho zloženie, teda zoznam modulov používaných v systéme, začínajúci generátorom a končiaci serializérom, medzi ktorými bude jeden alebo viac transformátorov. Pre toto použitie bude potrebné v sitemape uchovávať aj zoznam jednotlivých modulov registrovaných v portálovom rámci a ich prepojenie na prislúchajúce fyzické moduly (DLL knižnice alebo i.)
Pre korektné fungovanie Controlleru a teda aj samotného portálu budú do mapy portálu zadávané prepojenia používateľských dopytov na jednotlivé dátovody prípadne skupiny dátovodov. Controller tak bude môcť na základe dopytu presne rozhodnúť, ktoré dátovody

a v akom poradí (paralelne aktivované skupiny dátovodov) má pri obsluhovaní požiadavky aktivovať. Každému typu požiadavky bude taktiež priradená dizajnérska šablóna rozmiestnenia komponentov, ktorú bude využívať Merger pri skladaní XHTML výstupov jednotlivých dátovodov. Jej obsah však nebude súčasťou mapy portálu.

Zoznam údajových úložísk

Zoznam pripojení na zdroje dát bude pre jednoduchosť udržovaný na jednom mieste a teda v jednom globálnom XML súbore. Okrem prihlasovacích údajov do jednotlivých úložísk bude konfiguračný súbor obsahovať aj zoznam modulov, ktoré budú oprávnené dané úložiská používať. Toto riešenie bolo navrhnuté kvôli zaručeniu bezpečnosti údajov v portáli, hlavne hlavne pre prípad pridávania modulov tretích strán do systému, predovšetkým kvôli modulom, pre ktoré nemusia byť dostupné zdrojové kódy.

Konfigurácie aplikácií

Pre možnosť modifikácie správania sa jednotlivých modulov sme zhodnotili užitočnosť umožniť tvorcom jednotlivých aplikácií konfigurovať ich pomocou externých prostriedkov. Konfiguračné súbory modulov budú ukladané vo formáte XML. Tvorcovia modulov budú môcť tieto súbory používať svojvoľne, predpokladá sa však prevažne použite na načítavanie inicializačných hodnôt atribútov modulov po ich spustení.

Dizajnérske šablóny

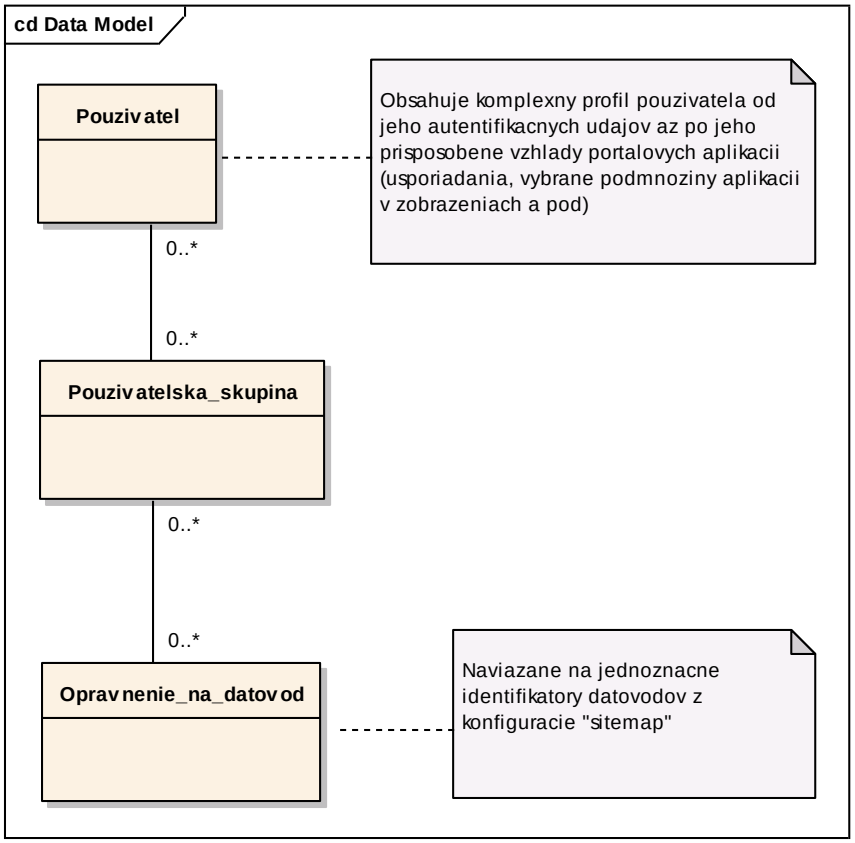
Dizajnérmi definované šablóny na rozloženie jednotlivých komponentov portálu používané pri finálnej serializácii na XHTML výstup budú ukladané vo forme XSLT štýlov, teda v podstate taktiež ako súbory XML.

4.5.2 Logický model údajov relačnej databázy

Podsystém autentifikácie, autorizácie a používateľskej personalizácie

Jednou zo základných funkcií portálového rámca je poskytovanie určitých služieb a zdieľaných údajov nasadeným aplikáciám. Typickým príkladom je autentifikácia na úrovni rámca, ktorú ďalej využívajú integrované aplikácie vo vlastnej réžii. Túto funkcionalitu bude rámec zabezpečovať pomocou komplexného profilu používateľa umožňujúceho

personalizáciu vzhľadu portálu, a taktiež viacúrovňovým systémom oprávnení postaveným na princípe používateľov, používateľských skupín (rolí) a oprávnení na používanie jednotlivých častí portálu až na úrovni jednotlivých dátovodov. Zaradenie používateľov do používateľských skupín, rovnako ako priradenie oprávnení používateľských skupinám je kardinality M:N, je teda možné vytvárať ľubovoľne komplexné autorizačné schémy. Logický model údajov tohto podsystému zachytáva Obrázok 7 – Podsystém autentifikácie a autorizácie.



Obrázok 7. - Podsystém autentifikácie a autorizácie

4.5.3 Systémové notifikácie

Na ukladanie systémových notifikácií sa predpokladá využitie obyčajných textových súborov, do ktorých budú jednotlivé udalosti ukladané v dopredu stanovenom formáte. V prípade, že by sa v priebehu ďalšieho návrhu alebo prípadného došpecifikovania požiadaviek ukázalo, že je potrebná komplexnejšia funkcionalita, bude pre tento typ údajov taktiež použitá relačná databáza.

4.6 Aplikačná a prezentačná časť

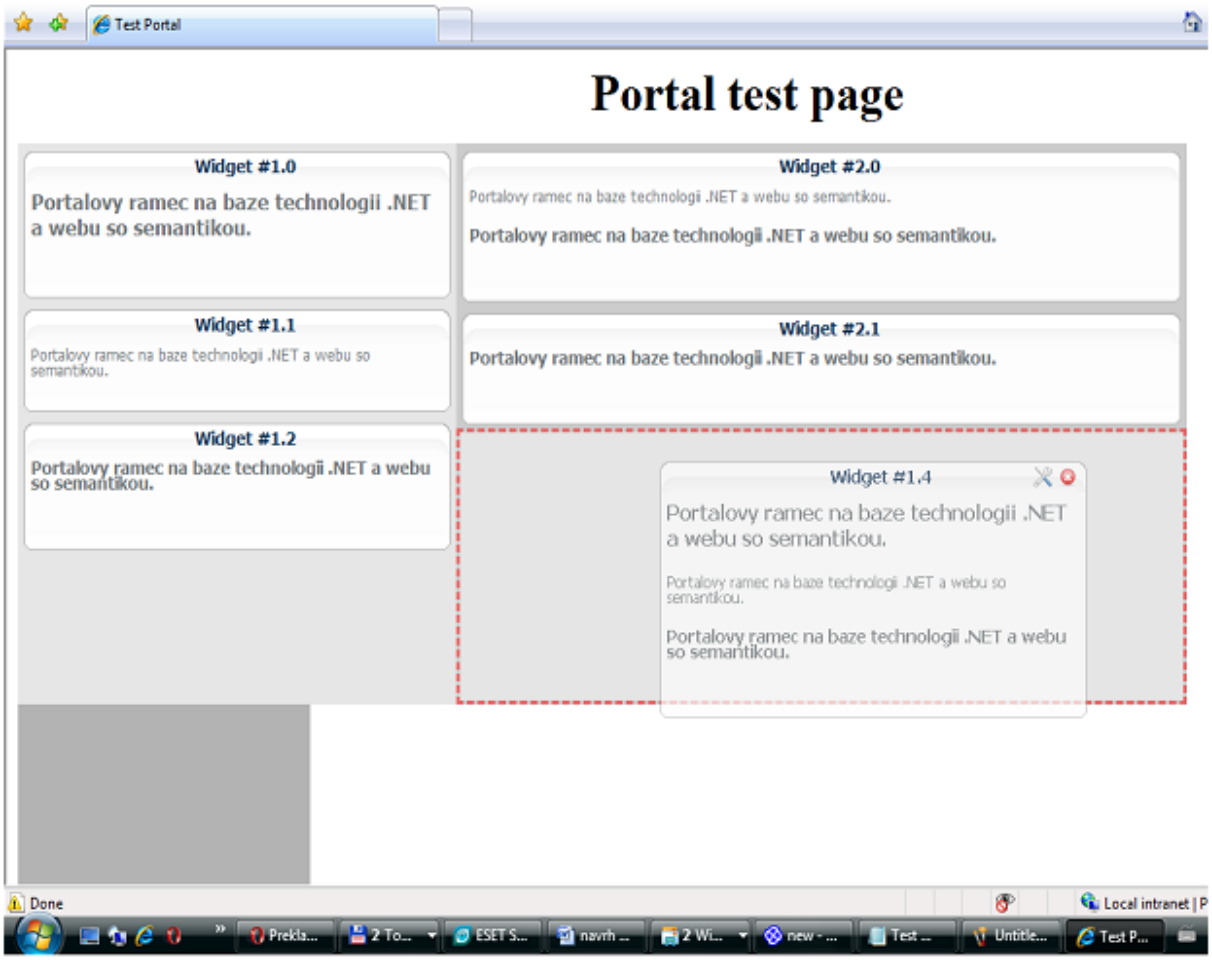
Prezentačnú vrstvu systému bude tvoriť množina samostatných aplikácií, ktorých rozmiestnenie a vzhľad na stránke si bude môcť používateľ na základe pridelených právomocí a spôsobu schémy rozmiestnenia upravovať . Tieto schémy rozmiestia pre danú používateľskú skupinu bude vytvárať systémový dizajnér.

Systém bude rozlišovať dva typy rozmiestnení:

- dynamické rozmiestnenie
- statické rozmiestnenie

Jednotlivé aplikačné okná v dynamickej časti budú polohovateľné, ich veľkosť bude meniteľná a v prípade požiadavky ich bude možné zatvoriť resp. otvoriť. Významovo podobné aplikácie budú združené do celkov, pričom aplikácie patriace jednému celku nebude možné prenášať do významovo cudzieho celku.

Aplikácie umiestnené v statickej časti si nebude môcť používateľ prispôsobovať, zatvárať resp. pridávať.



Obrázok 8. – Ukážka voľne polohovateľných okien

Na obrázku 8 je znázornený príklad rozhrania voľne polohovateľných a zatvárateľných okien v dynamickej časti stránky.

Konkrétny postup a technológia vytvárania grafického rozhrania bude spresnený v podrobnom návrhu dokumentu resp. v opise implementačnej časti, predpokladá sa však využitie JavaScriptovej knižnice Prototype Portal Class.

4.7 Zhrnutie

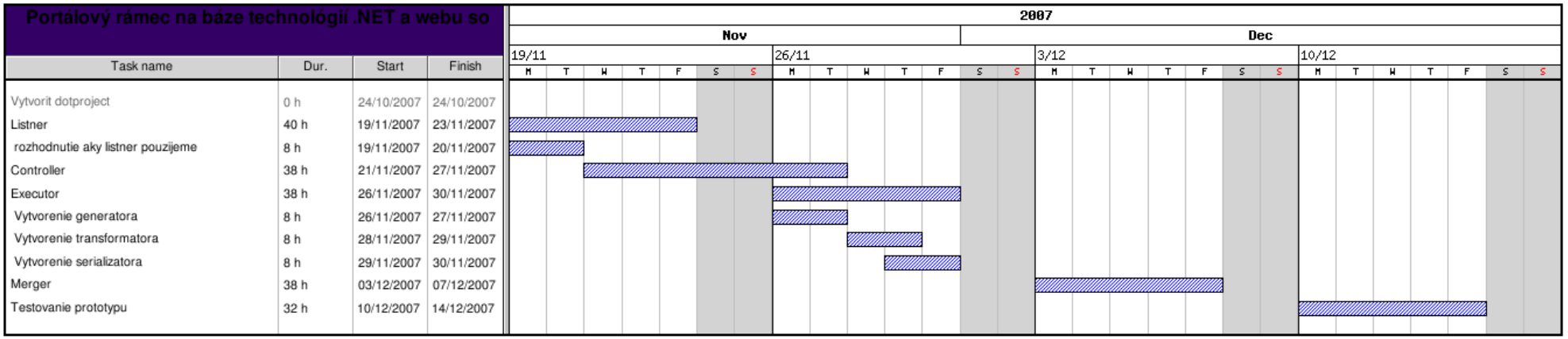
Cieľom tejto kapitoly bolo navrhnúť systém, ktorý spĺňa všetky body zhrnuté v špecifikácii projektu a navrhnúť dostatočne flexibilnú architektúru, ktorá dokáže reagovať na čo najväčšiu škálu požiadaviek na tvorbu konkrétnych portálov na báze tejto architektúry. Navrhovaný systém sme dekomponovali na časti na základe funkcionality, ktorú majú poskytovať. Jednotlivé časti sme sa pokúsili popísať na takej úrovni, aby bolo zrejmé, ako bude systém spracovávať jednotlivé používateľské dopyty. V rámci tejto témy sme sa pokúsili navrhnúť efektívny a korektný spôsob komunikácie jednotlivých častí výkonného prvku systému.

Prototyp

Našim ďalším cieľom bude zhotovenie prototypu navrhovaného riešenia. Chceli by sme naimplementovať kostru systému aspoň na takej úrovni, aby sme mohli otestovať postupnosť úkonov, ktoré je potrebné vykonať pri vygenerovaní jednoduchého dokumentu. To teda zahŕňa implementáciu všetkých komponentov (Listener, Controller, Executor a Merger) aspoň na elementárnej úrovni potrebnej na dosiahnutie simulácie behu systému. V rámci tejto simulácie predpokladáme, že transformácie v Executore budú iba formálneho charakteru. Zároveň by sme sa pri implementácii prototypu chceli definitívne rozhodnúť pre použitie komponentu Listener, teda zvážime, či je vhodnejšie implementovať vlastné riešenie, alebo použiť existujúci serverový proukt MS IIS.

4.8 Plán prác na nabližsie týždne

Týždeň	Subjekt práce	Opis práce
9.	Listener	Prototypovanie časti Listener je zamerané na zváženie možností a benefitov z vlastnej implementácie Listenera, alebo použitie MS IIS. Ďalej vyvoj rozhrania pre komunikáciu s Controllerom, ktoré zabepečuje predanie requestu na požadované zobrazenie dokumentu.
9.	Controller	Overenie rozsahu a funkčnosti navrhnutého spôsobu konfigurácie rámca v súvislosti s Controllerom a generovaním potrebnej informácie pre Executor a LayoutProvider.
10.	Executor	Prototypovanie je v tejto fáze zamerané na tvorbu základnýc typov modulov Generátor, Transformátor a Serializér a ich správne inicializovanie v dátovode prípadne viacerých datovodov podľa prijatého vstupu, ktorý bol vytvorený Controllerom.
11.	Merger	Overenie funkcionality Mergera, spojenenie výstupu z Executora do finálneho výstupu.
12.	Testovanie prototypu	Otestovanie celého prototypu a jeho častí za účelom overenia aktuálneho stavu návrhu riešenia voči požiadavkam kladeným na budovaný rámec. Cielom tejto fázy je odhaliť nedostatky a navrhnúť zmeny pre ďaľšie kroky vývoja rámca.



Obrázok 9. –Plán prác na najbližšie týždne z dotProjectu

5. Produkt

V tejto kapitole opisujeme koncepciu a základné vlastnosti jednotlivých častí výsledného produktu. Podrobný opis riešenia z implementačného hľadiska sa nachádza v technickej dokumentácii uvedenej v prílohe.

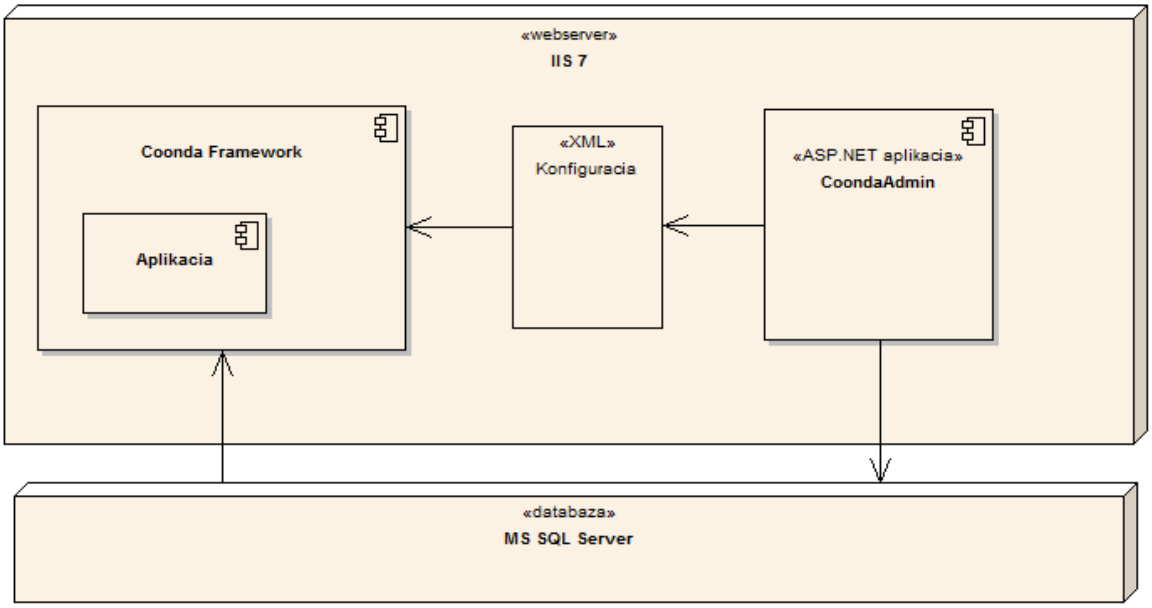
5.1 Architektúra systému - zmeny oproti pôvodnému návrhu

Návrh architektúry systému realizovaný v prvom semestri riešenia projektu sa ukázala ako pokrývajúca problém a vyžadujúca iba minimálne zmeny resp. dodatočné obohatenie. Hlavné architektonické zmeny sa týkajú :

- Použitia IIS namiesto vlastného http listenera, a s tým súvisiace vytvorenie interfejsu systému ako modulu do IIS
- Upustenie od paralelizácie vykonávania dátovodov a prejdenie na úplne serializované vykonávanie – toto riešenie sa po zvážení benefitov a komplikácií v prípade paralelného spracovania ukázalo ako výhodnejšie
- Zavedenie hlavného konfiguračného XML súboru s definíciou prístupových údajov k databáze používateľov a oprávnení
- Došpecifikovanie a implementácia jednoduchého administračného rozhrania ku konfigurácii portálového rámca – rozhranie pre administráciu používateľov, používateľských skupín, oprávnení, jednoduchú editáciu konfiguračných súborov

Funkcionalita na úrovni funkčných celkov jadra systému (Controller, Executor, Authorizator, ConfigProvider, Merger) ako aj na úrovni modulov (rozhrania, komunikácia XML streamom..) boli v plnej miere realizované v zmysle pôvodného návrhu. Výnimku tvorí Listener, ktorý, ako už bolo spomenuté, bol z projektu odstránený v prospech rozhrania so serverom IIS.

Bližšie o hlavých návrhových zmenách a rozhodnutiach k nim vedúcich, ako aj o novo implementovaných častiach systému pojednávajú ďalšie kapitoly.



Obrázok 10. - Architektúra systému

5.1.1 IIS vs. vlastný listerner

Ako už bolo spomínané v predošlej časti dokumentu, pri voľbe komponentu Listerner sme zvažovali dve alternatívy. Prvou alternatívou bolo implementovať vlastný Listener, druhou použiť MS IIS server.

Implementačné požiadavky na komponent Listeneru si vyžiadali zvoliť druhú alternatívu. Táto voľba vychádzala z množstva funkčnosti, ktoré ISS vývojárom poskytuje, a ktorú by sme boli museli v prípade implementácie vlastného komponentu sami vytvoriť. Jednalo sa najmä o vytvorenie funkcionality cookies, session, a pod. Ako sme neskôr zhodnotili, implementovanie tejto dodatočnej funkcionality, by si vyžiadalo nepomerne vyššie nároky na čas a ľudské zdroje ako sme predpokladali. Tento dôvod nás teda viedol siahnuť po hotovom riešení v podobe ISS.

5.1.2 Opis IIS

IIS poskytuje možnosť pridávania vlastných modulov, a tým rozšíriť jeho funkcionalitu resp. upraviť ju podľa vlastných požiadaviek používateľa. Tento fakt sme využili v prípade implementácie nášho Listenera vo forme prídavného modulu.

Aj tu sme narazili na dva spôsoby jeho implementácie. IIS server verzie 6.0 a nižšie podporujú pridávanie modulov napísaných v natívnom C resp. C++ kóde. Keďže je však jadro nášho systému napísane v manažovanom C# jazyku, bolo by potrebné doimplementovať rozhranie COM, ktoré umožňuje volať manažovaný kód natívnym a naopak. Toto riešenie je najmä z pohľadu rýchlosti komunikácie a zložitosti implementácie nie príliš žiaduce. Z tohto dôvodu sme siahli po IIS serveri

verzie 7.0, ktoré dovoľuje pridať okrem natívnych modulov aj moduly napísane v manažovanom kóde. Existuje však isté technické obmedzenie, na ktoré sme museli podstúpiť a to, že IIS server 7.0 je spustiteľný jedine pod operačným systémom MS Vista, resp. Windows Server 2008.

5.1.3 XML a SAX udalosti

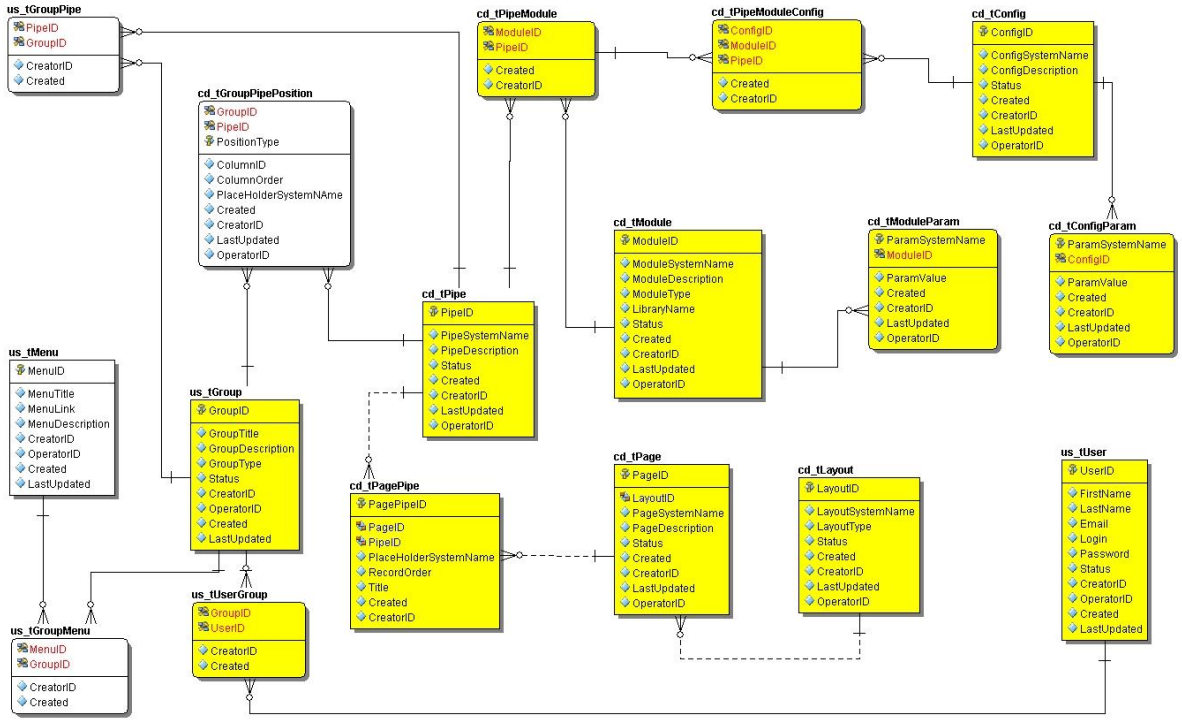
Oproti pôvodnému návrhu sme museli upustiť od úplného spracovania údajov vo formáte XML, ktoré v dátovodoch tečú medzi jednotlivými modulmi, pomocou SAX udalostí, nakoľko to v špecifických prípadoch modulov nebolo vhodné. Ako príklad môžeme uviesť modul „XSLT transformátor“, ktorý pri transformácii XML dokumentu potrebuje pracovať naraz s celým dokumentom, a nie iba s jeho časťami, ktorých obsah je poskytovaný objektu na spracovanie prúdov typu SAX.

Každý modul teda môže byť implementovaný dvoma spôsobmi – buď je možné naprogramovať tak, aby pracoval naraz s celým obsahom XML dokumentu, alebo mu môžu byť údaje posielané po častiach, závisí iba od programátora, pre ktorý spôsob sa rozhodne, a ktorý pre neho bude výhodnejší.

5.1.4 Konektivita jadra systému na databázový server

Zavedenie systému autentifikácie a autorizácie v podobe modulu Authorizator do jadra systému si pochopiteľne vyžiadalo vytvorenie konektivity na databázu ktorá systém užívateľov a užívateľských práv zabezpečuje. Platforma je natívne použitá – MS SQL Server. Konektivita je plne konfigurovateľná prostredníctvom XML konfiguračného súboru mainconf.xml.

5.1.5 Fyzický model údajov systému



Obrázok 11. – Fyzický dátový model

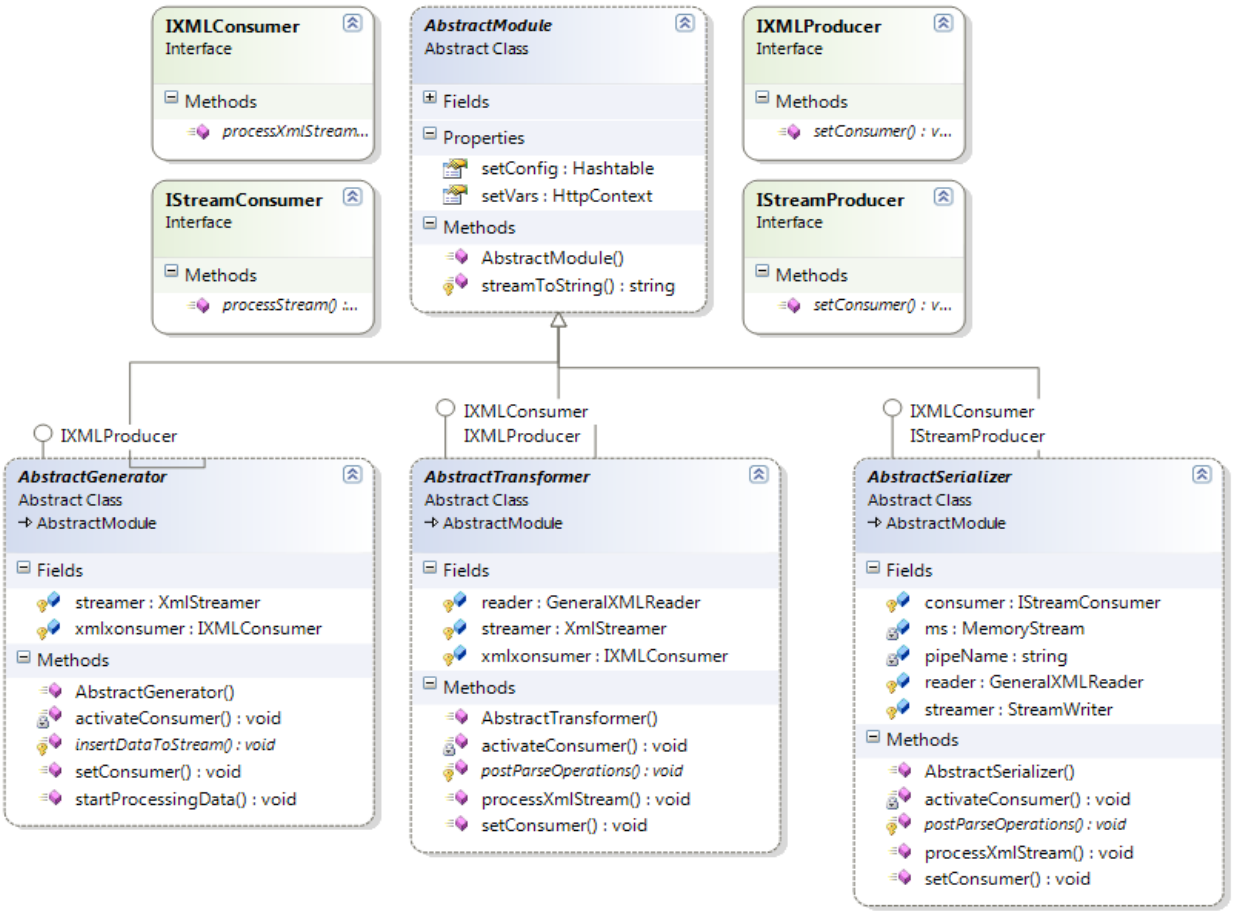
Podrobnejší popis k fyzickému modelu je možné nájsť v technickej dokumentácii na priloženom médiu.

5.2 Knižnica na vývoj modulov

Súčasťou nášho riešenia je okrem implementácie modulu do IIS, ktorý obsahuje jadro systému, riadiace beh jednotlivých dátovodov aj knižnica na vývoj modulov, ktorú sme nazvali Coonda Module Development Kit, skrátene Coonda MDK. Knižnica obsahuje abstraktné triedy potrebné pre vytváranie jednotlivých typov modulov, konkrétne generátorov, transformátorov a serializérov. Diagram tried implementovaných v Coonda MDK je zobrazený na obrázku 12. Všetky abstraktné triedy používané na vytváranie jednotlivých typov modulov (*AbstractGenerator*, *AbstractTransformer*, *AbstractSerializer*) sú zdedené od abstraktnej triedy *AbstractModule*. Jednotlivé triedy implementujú rôzne rozhrania a to vždy na základe toho, akým spôsobom spracúvajú vstupné a výstupné údaje. Pre prácu s tokom údajov typu XML boli vytvorené rozhrania *IXMLProducer* a *IXMLConsumer*. Pri návrhu sa predpokladalo, že výstupom serializéra nebude stále iba XHTML kód, preto serializéry implementujú rozhranie *IStreamProducer*. Rozhranie *IStreamConsumer* je aplikované na Merger jadra systému Coonda. Bližšie informácie o vytváraní modulov sú spomenuté v technickej dokumentácii produktu.

Knižnica taktiež obsahuje triedy používané jednotlivými modulmi, ktoré zabezpečujú vytváranie a spracovanie prúdov údajov vo formáte XML. Tieto XML pseudodokumenty môžu byť vytvárané samotnými modulmi, prípadne môžu byť prijímané zvonku (napr. z webových služieb alebo iných systémov) a priamo zapisované do prúdu. Zoznam všetkých tried je znázornený na obrázku s diagramom tried.

Pri vývoji modulov sa programátor nezaobíde ani bez možnosti výpisu chybových hlásení, preto Coonda MDK obsahuje aj základnú funkcionalitu pre prístup do záznamníku udalostí systému Windows, prostredníctvom ktorého je možný výpis chybových hlásení, ktoré sa vyskytnú za behu testovaného modulu.



Obrázok 12. - Diagram tried Coonda.mdk

5.2.1 Zoznam implementovaných modulov

Pomocou popisovanej knižnice sme implementovali nasledujúci zoznam modulov, ktoré budú v ďalšom texte bližšie predstavené:

- Modul na obsluhu procesu prihlasovania (GLoginProcessor) - Tento modul spolupracuje s jadrom systému, konkrétne s triedou zabezpečujúcou autentifikáciu používateľov, pričom zisťuje platnosť používateľského prihlásenia do systému a na základe tejto informácie generuje údaje potrebné pre zobrazenie prihlasovacieho respektíve odhlasovacieho formulára. Samotný proces overovania autentifikácie prebieha na úrovni jadra systému.
- SQL generátor (GSQLGenerator) – Modul, ktorý sa dokáže pripojiť na ľubovoľnú externú SQL databázu typu MS SQL Server, pričom jej dokáže poslať jej ľubovoľný SQL dopyt, ktorý následne spracuje a odošle vo formáte XML ďalšiemu modulu v z reťaze modulov v dátovode. Momentálna funkcionalita neumožňuje nastaviť spracovanie parametrických SQL dopytov, modul je možné nakonfigurovať iba na volanie konštantných dopytov bez akejkoľvek modifikovateľnosti závislej od používateľských vstupov.

- Generátor na výpis obsahu textových súborov (TextFileGenerator) - Jednoduchý generátor na načítanie obsahu textového súboru, ktorý je následne odoslaný na ďalšie spracovanie.
- XSLT transformátor (XSLTTransformator) - Modul, ktorý za pomoci XSLT šablóny, ktorú dostane ako konfiguračný parameter, transformuje XML z jednej podoby na druhú, pričom pravidlá transformácie sú určené spomínanou XSLT šablónou.
- XHTML serializér (XHTMLSerializer) - Funguje na tom istom princípe ako XSLT transformátor, pomocou XSLT šablóny transformuje XML dokument na XHTML dokument.
- Wrapper na externú aplikáciu „Factic – fazetový prehliadač“ (FacticGenerator) - Tento modul predstavuje wrapper pre externú aplikáciu „Factic – fazetový prehliadač“. Jeho úlohou je posielať vstupné používateľské parametre samotnému programu, ktorý na základe týchto parametrov generuje výstup.
- Wrapper na externú aplikáciu Google Maps (GoogleMapsGenerator) - Modul, obaľujúci funkcionalitu externej aplikácie Google maps. Jeho aktuálna verzia umožňuje nastaviť mu prostredníctvom konfigurácie miesto, ktoré má byť zobrazované pri prezeraní stránky portálu.

GoogleMapsGenerator

GoogleMapsGenerator je modul, obaľujúci funkcionalitu externej aplikácie Google maps. Jeho aktuálna verzia umožňuje nastaviť mu prostredníctvom konfigurácie miesto, ktoré má byť zobrazované pri prezeraní stránky portálu.

5.2.2 Definovanie šablón a štýlov

Produkt umožňuje definovať štýly zobrazovania jednotlivo každej stránky, tvorbou takzvaných XHTML šablón spolu v kombinácii s JavaScriptom a CSS štýlmi. Dáta sú do šablóny vkladane pomocou JSON(JavaScript object notation) formátu, ktorý poskytuje dostatočnú flexibilitu na ich ďalšie spracovanie. Detailný opis vkladania a použitia nájdete v technickej dokumentácii.

5.2.3 Zmeny oproti návrhu administračného rozhrania

Správa a prehľad notifikácií

Portálový rámec v prípade chyby vo funkčnosti alebo bezpečnosti aplikácie ako aj rámca, fyzického vyťaženia alebo výpadkov zdrojov, generuje notifikácie, ktoré slúžia administrátorovi systému bližšie lokalizovať a odstrániť problém. V pôvodnom návrhu bolo určený prehľad týchto notifikácií práve cez administračné rozhranie. Z dôvodu efektivity a samotnej zložitosti realizácie tohto spôsobu prezerania notifikácií bol zvolený spôsob notifikácie prostredníctvom systémového logu. To znamená,

že všetky problémy, chyby aplikácie a pod. Sú zahrnuté v systémových log súboroch samotnej server stanice a operačného systému.

Generovanie konfiguračných súborov

Portálový rámec je konfigurovaný na základe konfiguračných xml súborov, ktoré boli v pôvodnom návrhu upravované administrátorom systému priamo editáciou obsahu xml súboru. Toto riešenie sa ukázalo ako nesprávne z dôvodu integrity údajov v databáze a v xml konfiguračných súboroch. Súčasné riešenie ponúka možnosť vygenerovať xml konfiguračné súbory priamo z administračného rozhrania na základe dát z databázy. Toto zahŕňa návrh a implementáciu správy dátovodov, modulov a ich parametrov s konfiguráciami dátovodov s výsledkom bezpečnejšej konfigurácie celého portálového rámca a jeho aplikácií. Bola doplnená funkcionalita, ktorá umožňuje zobraziť len niektoré dátovody podľa zaradenia používateľa v príslušnej používateľskej skupine. Rovnako sa generuje konfiguračný súbor „sitemap.xml“ na základe rozvrhnutia stránky a umiestnenia dátovodov na stránke. To zahŕňa správu stránok a pozícií dátovodov na stránke.

Navigácia

Administrácia navigácie na portálovom rámci bola začlenená medzi funkcionalitu administračného rozhrania, nakoľko sa s ňou v pôvodnom návrhu nerátalo, ale je dôležitou súčasťou portálového rámca. Navigácia je prispôsobená používateľským skupinám na základe pridelenia v administračnom rozhraní.

5.2.4 Vstavaná šablóna a štýl

Produkt obsahuje vstavanú šablónu, ktorá podporuje interaktívne pozicovanie, zobrazovanie a ukladanie aktuálneho rozloženia jednotlivých aplikácií na stránke. Ukladanie používateľovej konfigurácie je realizované pomocou cookies. O každom používateľ ovi je vedená jedna cookieska, ktorá obsahuje konfiguráciu každej stránky v rámci portálu. Toto riešenie je limitované obmedzeniami na strane používateľa, použitým prehliadačom. Podľa odporučenia RFC 2109 vydaným W3C konzorciom, ktoré spĺňajú všetky dnes používané prehliadače, sú limity nasledovné.
Prehliadač je schopný uchovať:

- minimálne 300 cookiesiek
- minimálna veľkosť 4096 bajtov na cookiesku
- minimálne 20 cookiesiek per doménu

Alternatívne riešenie je možné realizovať pomocou ajaxových volaní a ukladať informácie o používateľovi na strane servera.

5.3 Bezpečnosť systému

Pri návrhu a implementácii systému boli brané na zreteľ aj bezpečnostné hľadiská. Keďže ide o pomerne otvorený a modulárny systém, bezpečnosť je riešená na viacerých úrovniach – od používateľských práv na úrovni operačného systému pre moduly až po údajovú bezpečnosť dát v podpornom databázovom systéme. O týchto riešeniach bližšie pojednávajú nasledujúce kapitoly.

5.3.1 Moduly tretích strán

Bezpečnostné riziká si treba uvedomiť predovšetkým pri nasadzovaní modulov tretích strán, ku ktorým nemáme dostupné zdrojové kódy, prípadne inú záruku dôveryhodnosti. Aj keď je samozrejme odporúčané takéto moduly vôbec nenasadzovať, ich nasadenie by principiálne nemalo znamenať kritickú hrozbu pre bezpečnosť. Hoci sa jedná o funkcionalitu na úrovni .NET frameworku, t.j. prakticky ľubovoľné operácie v systéme, treba si uvedomiť, že tento kód bude vždy bežať iba s používateľskými právami, aké nastavíme v konfigurácii IIS servera danému „application poolu“. Predvoleným kontom pre spúšťanie takýchto sieťových služieb v prostredí IIS je NETWORK SERVICE, čo je konto značne obmedzené čo sa týka zásahov do konfigurácie operačného systému alebo pod. V prípade, že je potrebné použiť striktnejšie obmedzenia, napr. čo sa týka vymedzenia prístupu v súborovom systéme a pod., je to ľahko možné realizovať vytvorením používateľského konta so želanými právami, a následným nastavením tohto konta pre „application pool“ portálového rámca. Konkrétny postup ako toto vykonať v prostredí IIS 7 je bližšie naznačený v časti Inštalácia systému.

Zabezpečená komunikácia

Systém plne podporuje podporu šifrovanej komunikácie HTTPS. Jedinou podmienkou je nastavenie IIS pre vykonávanie tejto služby.

5.3.2 Bezpečnosť administračného rozhrania

Bezpečnosť dát v databáze

Keďže administračná časť používa databázového používateľa CoondaWebUser, má obmedzený prístup na objekty a zároveň obmedzuje správu Coonda Servera len pre autorizovaných aplikačných používateľov. Pod aplikačným používateľom rozumieme používateľa vytvoreného v administračnej časti, pomocou správy používateľov. Títo používatelia sa autentifikujú na základe prihlasovacieho mena a hesla, bez ktorého nie sú možné vykonať žiadne operácie v administračnej časti.

Bezpečnosť aplikácie a servera

Bezpečnosť je riadená podľa právomocí priradených používateľovi, pod ktorým je spustená samotná aplikácia v IIS(Internet Information Servis). Táto funkcionalita vyplýva zo základných bezpečnostných nastavení produktu Windows Server 2008 Enterprise Edition.

5.3.3 Bezpečnosť údajov

Databázová bezpečnosť

Databázový server Microsoft SQL Server 2005 disponuje po nainštalovaní produktu Coonda databázou s názvom „CoondaWeb“. Zabezpečenie využíva funkcionalitu, vyplývajúcu z MS SQL Server bezpečnostnej politiky, a teda vytvára používateľov s rôznymi právami nad vytvorenou databázou.

Objekty v databáze a používatelia

Objekty v databáze môžeme kategorizovať na:

- Tabuľky – prístup má len vlastník databázy alebo používateľ s vyššími privilégiami
- Pohľady – prístup má okrem vlastníka aj CoondaWebUser, ktorý je používaný aplikáciou
- Procedúry – prístup má okrem vlastníka aj CoondaWebUser, ktorý je používaný aplikáciou

Základným princípom je minimalizácia prístupu používateľov na objekty typu *tabuľka(table)*, ktoré môžu obsahovať citlivé dáta, ktoré by mohli byť zneužité pri neoprávnenom prístupe. Tieto citlivé sa nedajú zobraziť na stránke, prístup k nim je zamedzený cez databázový objekt typu *pohľad*(view) a na manipuláciu sú použité objekty typu *procedúra(procedure)*.
Databázových používateľov rozdeľujeme :

- CoondaWebAdmin – má právo „vlastníka“ na všetky objekty.
- CoondaWebUser – má obmedzené právomoci na vybraný okruh objektov.

CoondaWebAdmin

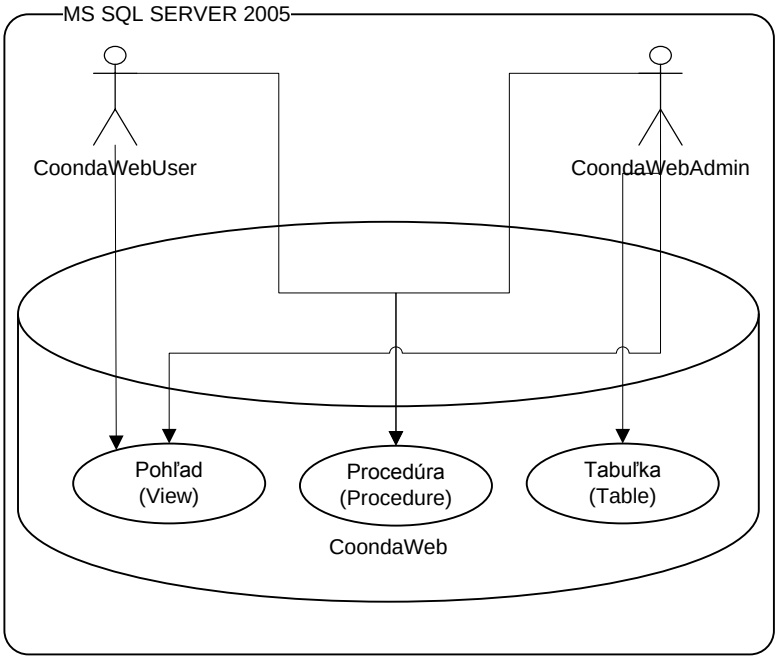
Privilégium „vlastník“ mu umožňuje akúkoľvek operáciu nad objektmi v databáze a využíva sa zásadne len pre pridávanie nových objektov. V čase prevádzky je prihlásenie pomocou tohto používateľa vypnuté, a tým minimalizovaný dopad pri neoprávnenom prieniku do systému a pri získaní hesla používateľa CoondaWebAdmin.

CoondaWebUser

Používateľ je používaný oboma aplikáciami *Coonda serverom* a jej administračnou časťou *CoondaWeb*. Oprávnenia môžeme zhrnúť týchto skupín:

- Oprávnenie pre *spúšťanie procedúr* (EXECUTE)

- Oprávnenie pre *získavanie dát z pohľadov* (SELECT)



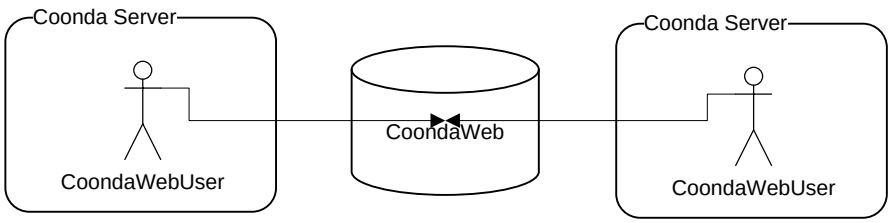
Obrázok 13. - Zjednodušená reprezentácia prístupov používateľov na databázové objekty

Jednoduchým príkladom pre pochopenie princípu je zamedzenie získaniu zoznamu hesiel pri neautorizovanom prieniku do databázy, cez používateľa používaným aplikáciou(CoondaWebUser). Používateľ v pohľade(nad tabuľkou obsahujúcou heslá) heslá nevidí a teda nevie vygenerovať ich zoznam. Príkladom manipulácie je použitie procedúry na zmenu hesla, ktorá vyžaduje okrem prihlasovacieho hesla aj hash starého hesla, ktoré je pri pokuse o zmenu porovnané so súčasným. Ďalším príkladom nad touto entitou je prihlásenie aplikačného používateľa do portálu.

5.3.4 Aplikačná bezpečnosť nad dátami

Aplikačná bezpečnosť predpokladá, že aplikácia používa na spojenie z databázou používateľa s obmedzenými právami na prácu s dátami. V jadre produktu Coonda a jej administračnej časti sa používa databázový používateľ CoondaWebUser a teda práva aplikácie sú obmedzené už databázovou bezpečnosťou.

Princíp zadefinovaný v databázovej bezpečnosti obmedzí aplikáciu len na vopred stanovené operácie s dátami, ktoré nie sú klasifikované ako citlivé.



Obrázok 14. - Používatelia aplikácií a ich napojenie na databázu

Prezentačná vrstva

Uchovávanie rozloženia aplikácií v používateľskom rozhraní je realizované prostredníctvom cookies, kedy sa každému prihlásenému používateľovi automaticky ukladá posledný stav rozloženia pre všetky stránky. Koncept cookies povoľuje manipulovať informáciu uloženú v prehliadači. Tento problém sa týka hlavne internetových kaviarní a miest kde je jeden počítač používaný viacerými používateľmi, ktorí majú k zmazaniu, alebo modifikovaniu uložených informácií prístup. Druhý prípad modifikovania používateľských nastavení, prípadne odcudzenie identity používateľa môže nastať pri použití modulov tretích strán, ktoré umožnia využitie techniky XSS (Cross-site-scripting).

5.4 Overenie výsledku

Overenie výsledku prebehlo v dvoch fázach, kedy sa najprv overovalo jadro a všetky funkcionality portálového riešenia. V druhej fáze prebehlo overovanie administračného rozhrania a jeho interakcie s portálovým riešením.

5.4.1 Overenie jadra portálového riešenia

Po nasadení portálového riešenia na testovací server prišlo k samotnému overovaniu správnej funkčnosti. Na to však bolo potrebné správne nakonfigurovať portálové riešenie pomocou konfiguračných XML súborov. Tieto konfiguračné súbory boli nastavené tak, aby zabezpečili overenie potrebnej funkčnosti ako zostavenie jednotlivých stránok (sitemap.xml), načítanie externých aplikácií, ich správne pozície a usporiadanie na stránke (sitemap.xml), uchovávanie pozície aplikácií na stránke, správne definovanie a konfiguráciu dátovodov (pipedef.xml) a jednotlivých modulov (modules.xml, moduleconf.xml) v dátovodoch a správnu XSLT transformáciu v dátovodoch.

Overovanie v tejto fáze prebehlo bez overovania funkčnosti ako prihlasovanie sa pomocou mena a hesla, kontroly oprávnenia používateľských skupín, zobrazovania stránok podľa zaradenia používateľa do používateľských skupín. Všetku túto funkcionalitu bolo potrebné overiť spolu s overovaním administračného rozhrania.

5.4.2 Overenie administračného rozhrania

Administračné rozhranie umožňuje úplnú konfiguráciu jadra portálového riešenia a preto bolo potrebné overiť hlavne správne generovanie konfiguračných XML súborov, ktoré sú generované na základe dát v databáze a vytvorené administračným rozhraním. Ako prvé bolo overené správne prihlasovanie pomocou mena a hesla a zamedzeniu neautorizovaného prístupu do administračného rozhrania. Ďalej prebehla kontrola správnej manipulácie s dátami (vytváranie, ukladanie, editácia a odstraňovanie) vo všetkých častiach administračného rozhrania (používatelia, skupiny, navigácia, stránky, rúry, moduly, šablóny a konfigurácie), správneho zobrazenia záznamov či už vo formulárových a tabuľkových zobrazeniach. Následne bolo overované správanie sa administračného rozhrania pri nevyplnení povinnej položky pri ukladaní záznamu, kedy administračné rozhranie upozornilo používateľa na nevyplnenú položku.

5.4.3 Overovanie celého systému

Overovanie administračného rozhrania prebiehalo z administrátorského konta na prázdnej databáze a po overení správnej funkčnosti rozhrania nastalo napĺňanie systému údajmi, ktoré nastavia jadro tak, aby bolo možné prezentovať všetky funkcie systému. Následne bolo možné overiť prihlasovanie sa na portál, zobrazovanie iného umiestnenia pre iné používateľské skupiny, oprávnenia používateľov na jednotlivé aplikácie ako aj samotnú interakciu administračného rozhrania s jadrom systému a to spôsobom zmeny údaju v administračnom rozhraní a následnej kontroly správneho správania na bežiacom portály. Takto boli overené všetky vlastnosti a poskytované funkcionality ako povolenie, zakázanie prístupu používateľovi, zaradenie používateľov do skupín, usporiadanie aplikácií na portály, zmena konfigurácie jednotlivých modulov a dátovodov, usporiadanie položiek navigácie a podobne.

6. Zhrnutie

Na začiatku riešenia projektu sme vykonali analýzu problémovej oblasti a existujúceho riešenia portálového rámca Cocoon. Priebežne sme sa zaoberali analýzou nástrojov a technológií vhodných pre riešenie projektu v rámci ohraničení štátneho programu. Pokračovali sme prácou na špecifikácii požiadaviek na vytváraný portálový rámec a tvorbou hrubého návrhu systému.

Po prvom kontrolnom bode sme sa venovali analýze rizík pri tvorbe rámca, ktorej cieľom bolo identifikovať kritické časti, ktoré by mohli zásadným spôsobom ohroziť úspešnosť celého projektu. Ako hlavné riziko sme identifikovali použitie a integráciu veľkého počtu nových a neodskúšaných technológií. Práve túto časť sme sa následne rozhodli prototypovať v závere zimného semestra. Tvorbou prototypu rámca sme overili našu schopnosť naštudovať vybrané technológie a nástroje, pomocou ktorých sme následne overili podstatu riešenia vo vytvorení jadra, ktoré dokázalo spracovať požiadavku od používateľa a spracovať ju pomocou rúr.

V druhej fáze riešenia počas letného semestra sme rozpracovali a upravili návrh podľa nových skutočností – využitie webového servera IIS na spracovanie požiadavky, administrácia používateľských oprávnení, konfigurácia portálu pomocou XML súborov, predávanie dát medzi modulmi pomocou SAX udalostí a transformácie pomocou XSL transformácií.

V ďalšej fáze sme sa snažili o zintegrovanie administračného rozhrania portálového rámca s jadrom a samotným výstupom vo forme modifikovateľných šablón.

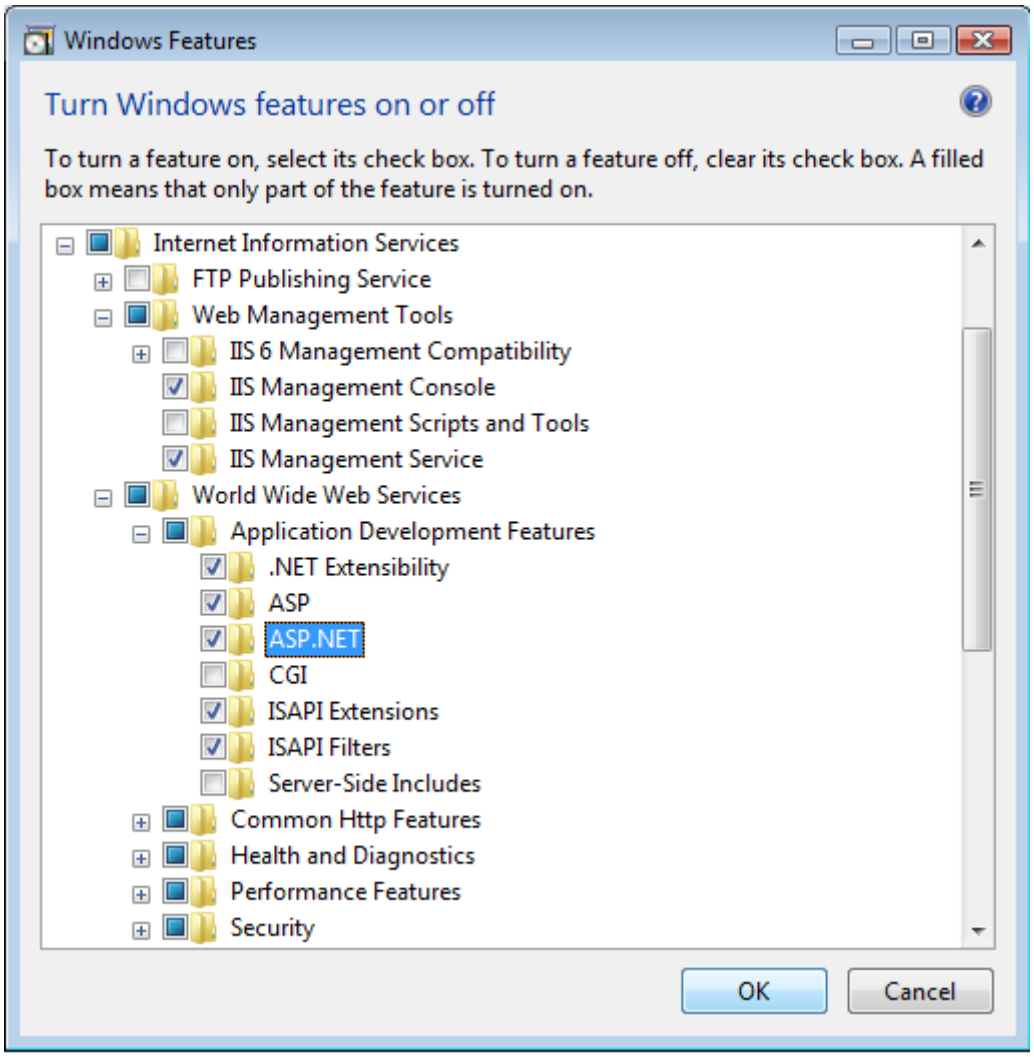
Kladne hodnotíme skúsenosti nadobudnuté tímovou prácou, ktorá sa pozitívne začala odzrkadľovať v kvalite produktu až po určitom čase, no jednoznačne sa prejavila sila tímovej práci oproti jednotlivcovi. Takisto sa nám pod ruky dostalo značné množstvo technológií, ktoré nás výrazne obohatili do ďalšieho profesijného rozvoja.

Príloha A: Technická dokumentácia

1.1 Návod na inštaláciu IIS

Pre korektnú inštaláciu IIS servera pre vytvorený systém je potrebné dodržať nasledovný postup:

1. V ovládacích paneloch Windows Vista zvoliť voľbu „Programs“.
2. V menu „Programs“ ďalej zvoliť položku „Turn Windows features on or off“.
3. V strome je potrebná nájsť a zvoliť záznam „Internet Information Services“.
4. V položke „Application Develepment Features“ zaškrtnúť „.NET Extensibility“, „ASP“ a „ASP.NET“.

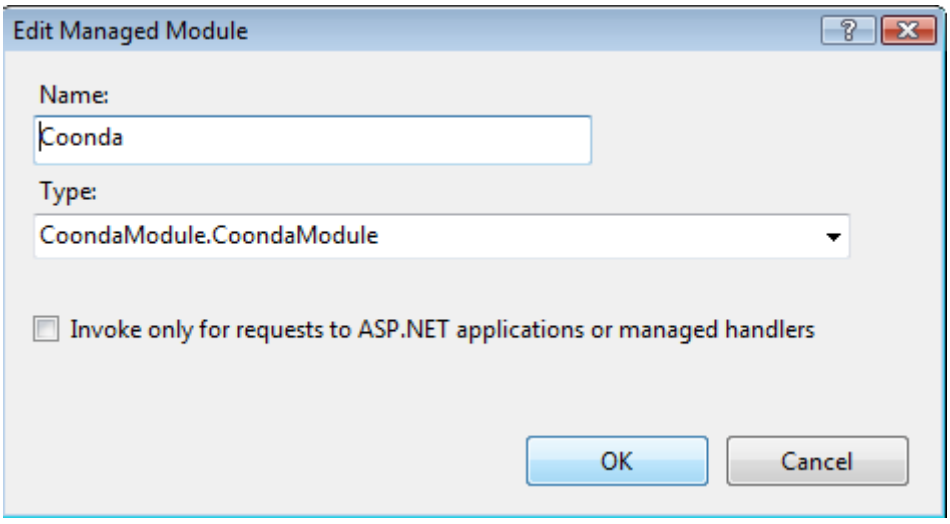


Obrázok 15. - Obrázok inštalácie ISS server 7.0

1.1.1 Návod pridania vlastného modlu do IIS servera

Pre korektné pridanie vlastného implementovaného modulu do IIS servera je potrebné dodržať nasledovný postup:

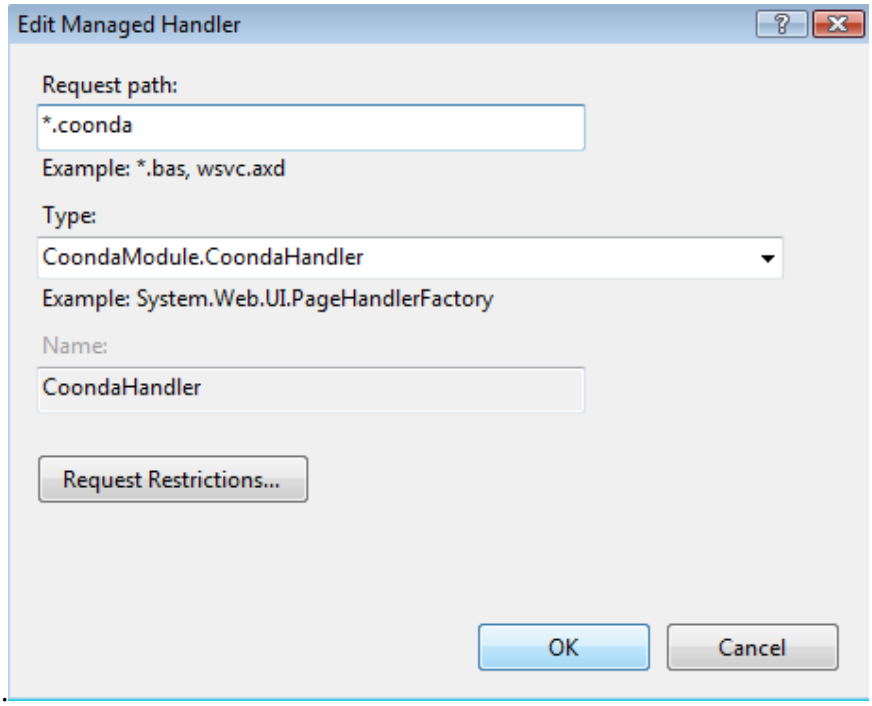
1. V menu „Administative Tools“ zvoliť voľbu „Internet Information Services (ISS) Manager“
2. V ľavom strome zvoliť položku „Sites“, neskôr „Default Web Site“.
3. V strede obrazovky zvoliť položku „Modules“.
4. V pravom menu medzi akciami zvoliť akciu „Add Managed Module“.
5. Do textového poľa „Name“ je potrebné zadať text „Coonda“ .
6. Do textového poľa „Type“ je potrebné zadať text „CoondaModule.CoondaModule“ .



Obrázok 15. - Pridávanie modulu

1. V stromovom menu sa vráťte na položku „Default Web Site“. .
2. V strede obrazovky zvoliť položku „Handeler Mappings“.
3. V pravom menu medzi akciami zvoliť akciu „Add Managed Handler“.
4. Do textového poľa „Request Path“ je potrebné zadať text „*.coonda“ .

5. Do textového poľa „Type“ je potrebné zadať text „CoondaModule.CoondaHandler“



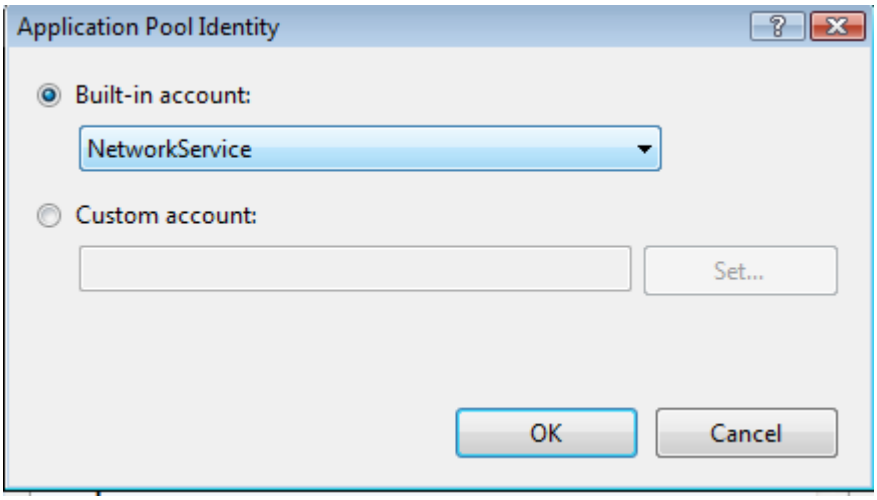
Obrázok 16. - Pridávanie Handleru

6. Reštartuje IIS server akciou „Restart“.

1.1.2 Obmedzenie používateľských práv IIS

Coonda beží na serveri ako sieťová služba (network service), ktorá v tomto nastavení neprestavuje bezpečnostné ohrozenie pre používateľa. V prípade potreby zabezpečenia väčších práv pre systém je možné túto požiadavku docieliť nastavením IIS servera takto:

1. V menu „Administative Tools“ zvoliť voľbu „Internet Information Services (ISS) Manager“
2. V ľavom strome zvoliť položku „Application Pools“.
3. Zvoľte „Default Application Pool“ a pravým tlačidlom myši vyvolajte kontextové menu.
4. V kontextovom menu zvoľte položku „Advanced Settings...“
5. V záložke „Process Model“ zvoľte položku „Identity“.
6. Vyberte predefinovanú hodnotu resp. zvolením gombíka „Custom Account“ je možné službe priradiť práva existujúceho používateľa operačného systému.



Obrázok 17. - Zmena práv systému

1.1.3 Nastavenie SQL Server 2005

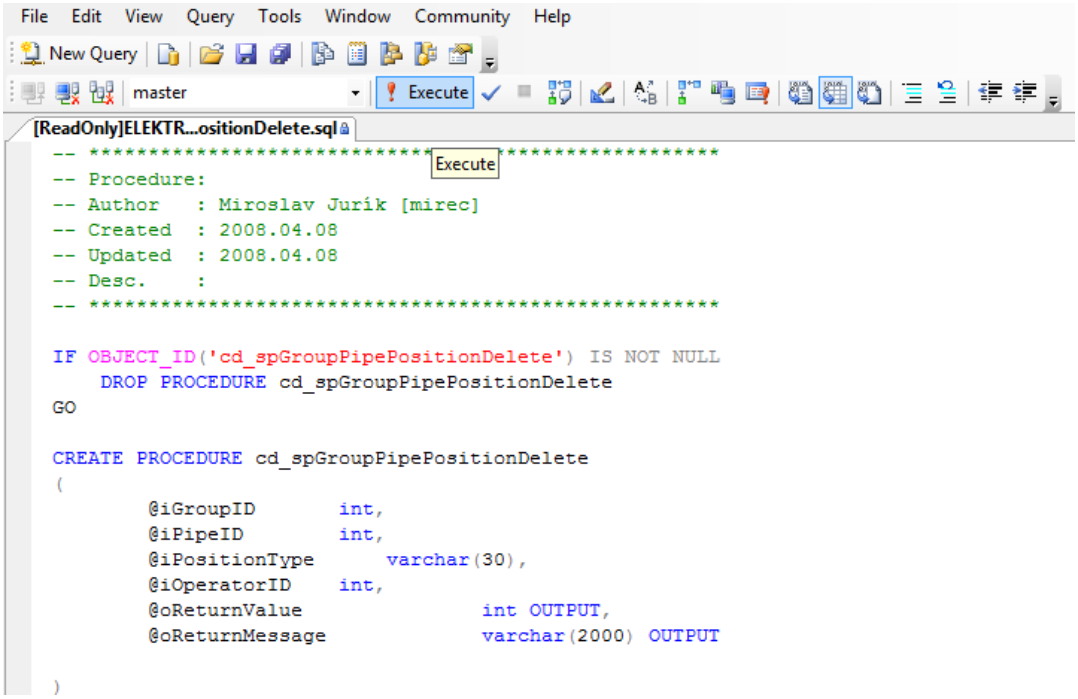
Táto kapitola opisuje sadu nastavení pre SQL Server 2005 projektu.

TCP/IP spojenie

1. Spusťte "SQL Server Configuration Manager".
2. V stromovom menu kliknite na položku "SQL Server 2005 Network Configuration".
3. Položku TCP/IP v pravom menu zmeňte z „disabled" na „enabled".

Vytvorenie databázy

1. Spusťte "SQL Server Mangment Studio".
2. Prihláste sa pod administrátorským účtom.
3. V projekte „CoondaSQL" systému Coonda sa nachádzajú adresáre „Coonda" a „Use System". V každom z týchto adresárov sa nachádzajú štyri podadresáre „Constraints", „Procedures", „Tables", „Views".
4. Pre vytvorenie databázy spusťte všetky súbory s príponou „.sql" a vykonajte ich obsah kliknutím v menu na tlačidlo „Execute".



Obrázok 18. - Vytváranie databázy v MS SQL Server 2005

1.2 Inštalácia produktu

Inštaláciu produktov Coonda rozdeľujeme na 3 základné kroky, ktoré by mali byť vykonané v nasledujúcom poradí:

1. Vytvorenie databázy CoondaWeb a jej nastavenie
2. Vytvorenie Coonda Server aplikácie
3. Vytvorenie administračnej časti Coonda Servera
4. Vytvorenie zdroja správ v záznamníku udalostí pre aplikáciu

Inštalácia predpokladá nainštalovaný MS SQL Server verzie 2005 s povolenou TCP konektivitou.

1.2.1 Vytvorenie a konfigurácia databázy

Vytvorenie databázy môžeme vykonať dvoma spôsobmi. Prvým zo spôsobov je vytvorenie databázy, používateľov a potrebných databázových objektov manuálne. Databáza CoondaWeb obsahuje početné množstvo skriptov, ktorých spustenie vyžaduje určitý čas(nehovoriac o naplnenie dátami) a preto sa tento spôsob inštalácie neodporúča. Druhým spôsobom je využitie zaužívaných spôsobov pre prenos databázy vrátane objektov, používateľov a ich oprávnení.

Druhý spôsob využíva nástroj *Backup / Restore*, ktorý sa stal od verzie MS SQL databázy 2005 voľne dostupný v produkte *Microsoft SQL Server Management Studio Express Edition.*

1.2.2 Konfigurácia databázového servera pred obnovou

Konfigurácia databázového servrea spočíva vo vytvorení používateľov v databáze. Vytvárame používateľov s názvami „CoondaWebAdmin“ a „CoondaWebUser“.

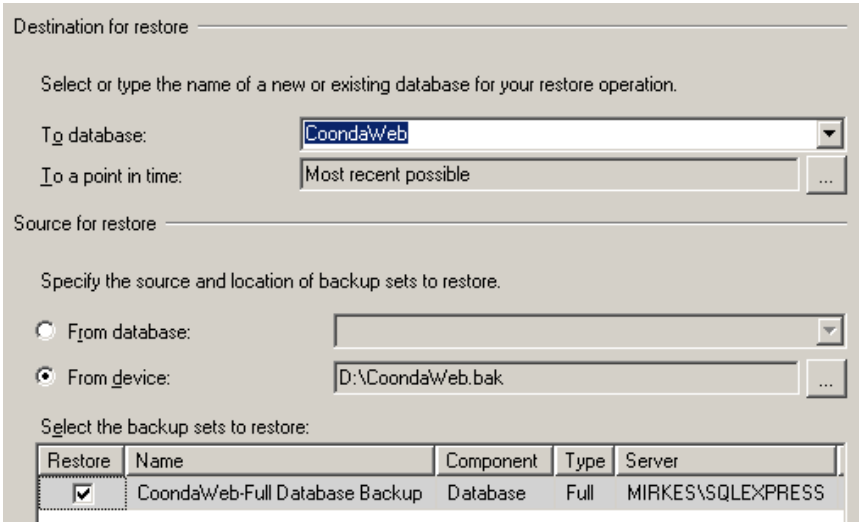
1.2.3 Obnova databázy

Na obnovu databázy sa pripojíme na náš SQL Server pomocou *Microsoft SQL Server Management Studion* príp. *Express Edition.* Obnova databázy je možná iba pod kontom, ktoré má náležité oprávnenia (napr. konto „System Account“, prihlasovacie meno „sa“).

Obrázok 19. - Prihlasovacia obrazovka nástroja Management Studion

Po prihlásení sa nám zobrazí v ľavom okne náš server, v ktorom po rozkliknutí nájdeme položku „Databases“. Kliknutím na pravým tlačidlom a výberom položky „*Restore Database*“ z kontextového menu sa nám zobrazí nové okno pre spustenie procesu obnovy databázy. Pri vypĺňaní si zadávajme hodnoty nasledovne:

- Názov databázy – CoondaWeb
- Súbor so zálohou – vyberie z inštalačného média CoondaWeb.bak



Obrázok 20. - Nastavenia potrebné pre obnovu databázy

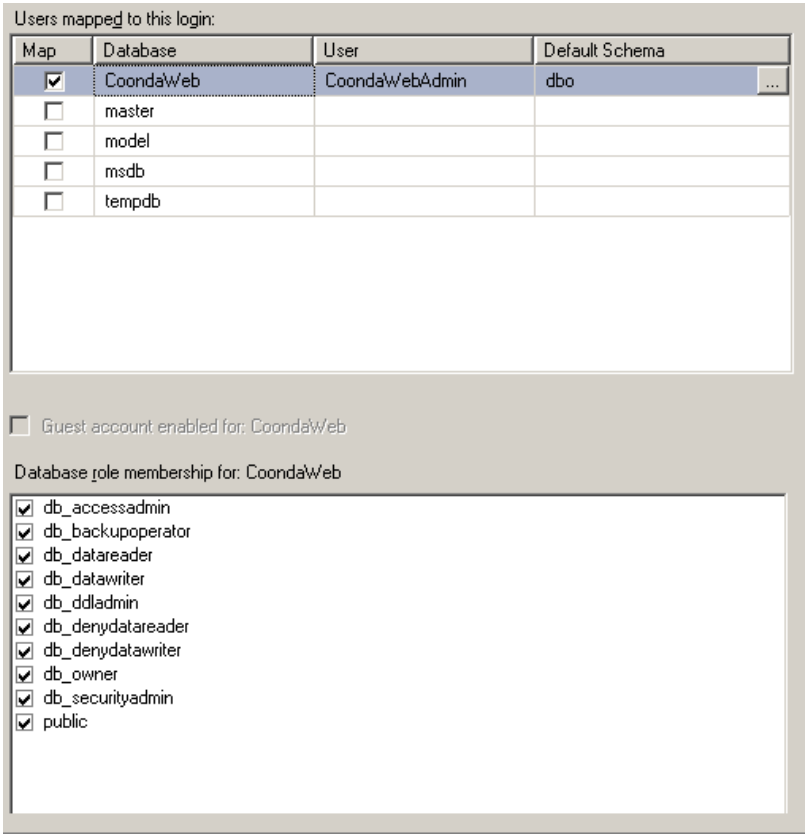
Nastavíme cesty, kde sa má uložiť fyzicky nami vytváraná databáza spolu s logovacím súborom a potvrdíme tlačidlom Restore.

1.2.4 Konfigurácia databázového servera po obnove

Konfigurácia databázového servera spočíva vo vytvorení špecifických rolí v databáze pre vytvorených používateľov CoondaWebAdmin, CoondaWebUser.

CoondaWebAdmin

V stromovej štruktúre vyberieme položku *Security -> Logins -> CoondaWebAdmin* a v kontextovom menu, vyvolaného pravým tlačidlom, potvrdíme možnosť *Properties* .V novo vytvorenom okne vyberieme z ľavého menu položku *User Mappings.* Vyberieme riadok, kde názov databázy súhlasí s meno nami vytvorenej databázy a vyberieme všetky položky z okna nižšie. Všetky položky vyberáme, z dôvodu plného prístupu používateľa CoondaWebAdmin na databázu. Tento používateľ bude používaný len ako vlastník databázy.



Obrázok 21. - Ukážka priradenia rolí používateľovi CoondaWebAdmin

CoondaWebUser

Postupujeme rovnako ako pri používateľovi CoondaWebAdmin ale s rozdielom, že nevyberáme všetky role v databáze. Vyberáme len rolu „*public*"!.

1.2.5 Vytvorenie administračnej časti

Inštalácia administračnej aplikácie spočíva v nakopírovaní potrebných skriptov na disk servera, nastavenie IIS a nakonfigurovanie aplikácie.

Nakopírovanie skriptov

Skripty nakopírujeme do systémového adresára(napríklad c:/inetpub/Coonda/CoondaAdmin) aplikácie z inštalačného CD.

Nastavenie IIS

Aplikáciu vytvárame do už vytvorenej webovej stránky(IIS Web site), ktorá bola vytvorená v predchádzajúcom kroku(časť inštalácie Coonda Servera). Do tejto stránky vytvoríme novú aplikáciu s nasledujúcimi nastaveniami:

- Alias – „CoondaWeb"

- Stránka administračnej časti bude prístupná http://nazov_stroja/CoondaWeb[1]

- Cesta – „c:/inetpub/Coonda/CoondaAdmin“

Konfigurácia aplikácie

Aplikácia a jej konfiguračné údaje nájdeme v súbore „c:/inetpub/Coonda/CoondaAdmin/web.config“. Nájdeme tam premenné akými sú:

- ConnectionString – hovorí o nastaveniach konekcie na databázu
- ProjectUrl – Url aplikácie administračnej časti
- CoondaConfigDirectory – Adresár, kde sa nachádzajú konfiguračné súbory Coonda Servera

```
<appSettings>
    <add     key="ConnectionString"     value="Password=CoondaWebUser;Persist     Security
Info=True;User ID=CoondaWebUser;Initial Catalog=CoondaWeb;Data Source=rouen.ynet.sk"/>
    <add key="ProjectUrl" value="http://mirkes.ynet.sk/CoondaWeb/"/>
    <add key="ConndaConfigDirectory" value="d:/projects/coonda/coonda/config/"/>
```

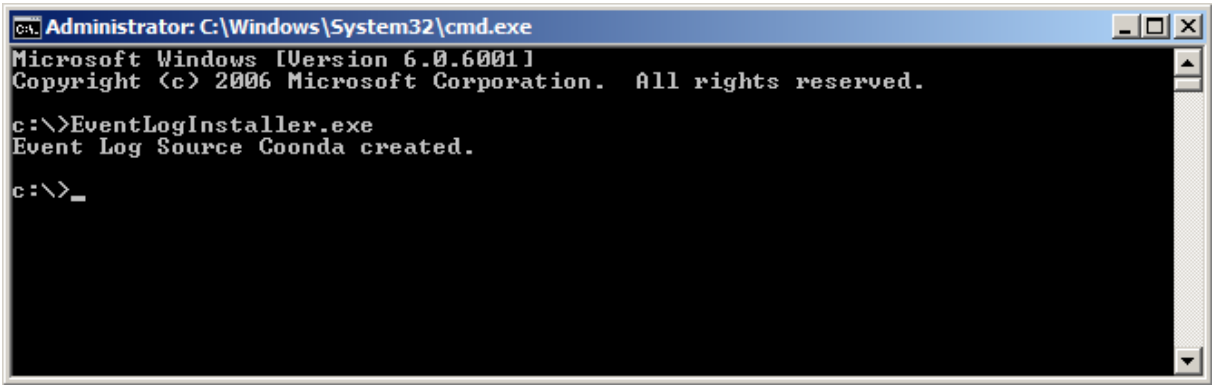
Kód 1. – Ukážka nastavení administračnej časti aplikácie

1.2.6 Vytvorenie zdroja správ v záznamníku udalostí pre aplikáciu

Aplikácia podporuje zaznamenávanie kritických udalostí počas svojho behu do systémového záznamníku udalostí (event log) systému Windows. Toto umožňuje jednoducho a prehľadne – pomocou prehliadača Event Log Viewer - zisťovať prípadné chyby konfigurácie, komunikácie s databázou, nefatálne výnimky vzniknuté za behu atď.

Ako už bolo uvedené vyššie, samotná aplikácia ako modul do IIS však nemá oprávnenia na ovplyvňovanie systémových nastavení, a teda nedisponuje ani možnosťou vytvoriť si vlastný zdroj správ v záznamníku udalostí. Na tento účel bol preto vytvorený samostatný program určený na použitie vo fáze inštalácie, nazvaný EventLogInstaller.
EventLogInstaller je jednoduchá konzolová aplikácia, ktorá musí byť v rámci inštalácie produktu spustená s príslušnými oprávneniami – typicky administrátorom. Pri spustení bez parametrov vytvorí v záznamníku udalostí tzv. zdroj udalostí pre aplikáciu s názvom Coonda, a vloží doň prvú správu o udalosti. Pri spustení s parametrom /delete tento zdroj (v prípade existencie samozrejme) zo záznamníka udalostí odstráni. Toto je teda vhodné v procese odinštalácie.

[1] Názov stroja je závislý od nastavenia IIS webovej stránky(web site) a host-header hodnoty.



Obrázok 22. – Ukážka vytvorenia logu záznamov

1.3 Konfigurácia Coonda IIS

1.3.1 Konfigurácia jadra

Nasledujúca kapitola opisuje konfiguračné súbory systému.

mainconf.xml

Súbor mainconf.xml slúži ako konfiguračný súbor jadra systému a autorizačného modulu systému.

Elementy

- „*coondamainconf*“ predstavuje záznam pre daný modul.
- „param“ predstavuje voliteľný konfiguračný parameter systému.

Atribúty

- „name“ predstavuje identifikátor modulu.
- „value“ reprezentuje voliteľný názov parametra pre jednotlivý modul.

```
<CoondaMainConf>
  <param name="default_page" value="mainpage" />
  <param name="db_host" value="rouen.ynet.sk" />
  <param name="db_user" value="CoondaWebUser" />
  <param name="db_pass" value="CoondaWebUser" />
  <param name="db_database" value="CoondaWeb" />
  <param name="configRefreshInterval" value="15" />
</CoondaMainConf>
```

Obrázok. 23 - Príklad konfigurácie

Konfigurácia stránok - sitemap.xml

Súbor sitemap.xml slúži ako konfiguračný súbor pre jednotlivé stránky.

Elementy

- „*page*“ predstavuje záznam pre jednotlivú stránku.
- „pipe“ predstavuje zoznam rúr dostupných pre danú stránku.

Atribúty

- „id“ predstavuje identifikátor pre stránku.
- „layout“ reprezentuje šablónu pre jednotlivú stránku.
- „position“ predstavuje pozíciu šablóne vo forme čísla.
- „name“ predstavuje názov rúry.
- „title“ predstavuje titulku pre dané okno rúry.

```
<page id="mainpage" layout="mainpage">
  <pipe position="2" name="Login" title="Login"/>
  <pipe position="0" name="NavMenu" title="NavMenu"/>
  <pipe position="0" name="TextInput" title="Text Input 1"/>
  <pipe position="1" name="GoogleMap" title="Google Maps" />
  <pipe position="1" name="FacticBrowser" title="Factic" />
</page>
```

Obrázok. 24 - Príklad konfigurácie

Konfigurácia dátovodov - pipedef.xml

Súbor pipedef.xml popisuje štruktúru rúr systému.

Elementy

- „*pipe*“ predstavuje záznam pre danú rúru.
- „module“ predstavuje záznam pre daný modul definovaný v rámci rúry.
- „conf“ predstavuje konfiguračný záznam modulu v rámci danej rúry.

Atribúty

- „id“ predstavuje identifikátor rúry resp. modulu v rámci rúry.
- „public“ reprezentuje viditeľnosť modulu. („1“ = modul je viditeľný aj pre neautentifikovaných používateľov. „0“ = modul je viditeľný iba pre autentifikovaných používateľov.)
- „name“ predstavuje názov modulu v rámci rúry.

```
<pipe id="FacticBrowser" public="1">
  <module id="FacticGenerator" >
    <conf name="FacticGenerator_SP" />
  </module>
  <module id="XSLTTransformer" >
    <conf name="FacticTransformer_1" />
  </module>
  <module id="XHTMLSerializer">
    <conf name="FacticTransformer_XHTML"/>
  </module>
</pipe>
```

Obrázok. 25 - Príklad konfigurácie

Zavedenie modulov do jadra - modules.xml

Súbor modules.xml popisuje zoznam modulov, ich typ a názov knižnice na pevnom disku.

Elementy

- „*module*“ predstavuje jednotlivý záznam pre daný modul, ktorý má tri atribúty.

Atribúty

- „id“ predstavuje identifikátor modulu
- „type“ predstavuje typ modulu („g“ = generátor, „s“ = serializér, „t“ = transformátor)
- „lib“ predstavuje názov knižnice na disku

```
<Modules>
  <module id="TextFileGenerator" type="g" lib="TextFileGenerator.dll" />
  <module id="XSLTTransformer" type="t" lib="XSLTTransformer.dll" />
  <module id="XHTMLSerializer" type="s" lib="XHTMLSerializer.dll" />
  <module id="GoogleMapsGenerator" type="g" lib="GoogleMapsGenerator.dll" />
  <module id="FacticGenerator" type="g" lib="FacticGenerator.dll" />
  <module id="GLoginProcessor" type="g" lib="GLoginProcessor.dll" />
  <module id="GSQLProcessor" type="g" lib="GSQLProcessor.dll" />
</Modules>
```

Obrázok. 26 - Príklad konfigurácie

Konfigurácia modulov - moduleconf.xml

Súbor moduleconf.xml slúži ako konfiguračný súbor pre jednotlivé moduly.

Elementy

- „*moduleconfig*“ predstavuje záznam pre daný modul.
- „param“ predstavuje voliteľný parameter modulu, ktorý obsahuje dva atribúty.

Atribúty

- „id“ predstavuje identifikátor modulu.
- „name“ reprezentuje voliteľný názov parametra pre jednotlivý modul.
- „value“ predstavuje hodnotu atribútu „name“.

```
<moduleconfig id="FacticGenerator_SP">
  <param name="css" value="factic.css" />
  <param name="coreRefreshInterval" value="300" />
</moduleconfig>
```

Obrázok. 27 - Príklad konfigurácie

1.4 Tvorba modulov pomocou knižnice na vývoj modulov

Tvorba modulov pre aplikáciu Coonda je spravidla jednoduchou záležitosťou. V princípe sa tento proces skladá z nasledujúcich krokov:

1. Vytvorenie projektu v prostredí Visual Studio 2005 a pridanie referenčných knižníc.
2. Vytvorenie vlastnej triedy, ktorá dedí svoje vlastnosti od jednej zo vzorových abstraktných tried.
3. Použitie minimálnej množiny predpísaných namespace.
4. Preťaženie predpísaných funkcií, ktoré obsluhujú vstupno-výstupné operácie.
5. Implementácia samotného modulu.
6. Skompilovanie dynamickej knižnice.
7. Pridanie vytvorenej knižnice do aplikácie Coonda a nakonfigurovanie modulu.

1.4.1 Vytvorenie projektu v prostredí Visual Studio 2005 a pridanie referenčných knižníc

Pri vytváraní nového projektu je nutné zvoliť typ projektu „Class library“, pričom je projekt možné implementovať v ľubovoľnom jazyku, patriacom do množiny jazykov používaných na platforme .NET. Ukážkový kód tohto dokumentu bude písaný v jazyku C#. Pri pomenovaní projektu sa odporúča v jeho názve spomenúť aj typ modulu, najlepšie na konci samotného názvu, a to pomocou prípon „Generator“, „Transformator“ a „Serializer“, ktoré zodpovedajú jednotlivým typom modulov. Teda napríklad modul na načítanie obsahu textových súborov by bol nazvaný TextFileGenerator.
Po vytvorení projektu je nutné pridať do neho referencie na knižnicu *coonda_mdk.cs.dll*, ktorá obsahuje samotnú knižnicu na vývoj modulov pre aplikáciu Coonda.

1.4.2 Vytvorenie vlastnej triedy

Hlavnú triedu modulu je potrebné pomenovať tým istým názvom, ako je názov celého projektu. Taktiež ju treba vydediť od abstraktnej triedy, zodpovedajúcej typu vytváraného modulu, teda od jednej z tried AbstractGenerator, AbstractTransformer, alebo AbstractSerializer. Nasledovne je potrebné pridať kód pre konštruktor, preťažujúci konštruktor predpísanej abstraktnej triedy, od ktorej je zdedená vytváraná trieda.

```
public class TextFileGenerator : AbstractGenerator
{
   public TextFileGenerator(HttpContext Vars, Hashtable Config,
             Hashtable sharedMemory) : base(Vars, Config, sharedMemory)
   {
   }
}
```

Kód 2. – Príklad vytvorenia vlastnej triedy

1.4.3 Použitie minimálnej množiny predpísaných namespace

Všetky moduly musia mať predpísané použitie nasledovných namespace:

```
using System;
using System.Collections;
using System.Text;
using System.IO;
using coonda_mdk;
using coonda_mdk.abstracts;
using System.Xml;
using System.Web;
```

Kód 3. Minimálna množina predpísaných namespace

V prípade vytvárania modulov, ktoré spracovávajú vstupné toky údajov vo formáte XML, teda pri transformátoroch a serializéroch, je nutné k predpísaným namespace pridať ešte nasledovné:

```
using AElfred;
using Org.System.Xml.Sax;
using Org.System.Xml.Sax.Helpers;
```

Kód 4. - Doplňujúca množina namespace

1.4.4 Preťaženie predpísaných metód

Každému typu modulu je potrebné preťažiť odlišné funkcie, ktoré sú zodpovedné za spracovanie toku údajov.

Generátor

V moduloch typu generátor je potrebné preťažiť metódu insertDataToStream(), ktorá je zodpovedná za vloženie vygenerovaných údajov vo formáte XML na začiatok dátovodu.

```
protected override void insertDataToStream()
{
}
```

Kód 5. - Metóda na preťaženie

Transformátor a serializér

V moduloch týchto typov je potrebné povinne preťažiť metódu postParseOperations(), ktorá je volaná v priebehu prúdenia XML údajov v dátovode hneď po spracovaní prichádzajúceho XML prúdu. Táto metóda je potrebná pre prípad, že by kompletný výstupný XML dokument nebolo možné vytvoriť v priebehu spracovania SAX udalostí.

```
protected override void postParseOperations()
{
}
```

Kód 6. - Metóda na preťaženie

1.4.5 Implementácia modulu

Pri implementácii modulov je povolené používať všetky možnosti, ktoré sú ponúkané platformou .NET, je však potrebné myslieť na to, že aplikácia Coonda beží s obmedzenými používateľskými právami používateľa *NetworkService*, čo zmenšuje množinu povolených operácií.
Jediné predpísané pravidlá pri implementácii modulov sa týkajú zápisu údajov vo formáte XML do výstupu modulu a spracovania vstupných tokov údajov vo forme XML.

Zápis údajov do dátovodu

Zápis údajov do dátovodu môže prebiehať v dvoch formách. Buď pomocou .NETového objektu typu XMLTextWriter a teda za pomoci metód, ktoré má tento objekt definovaný (napr. metóda na vytvorenie začiatočného a ukončovacieho elementu XML dokumentu), alebo pomocou jednoduchého objektu na zápis textových reťazcov (ktorých obsahom však nutne musí byť XML dokument). Toto riešenie je vhodné pri vytváraní wrapperov na externé aplikácie, ktorých výstupom už je hotový XML dokument, ktorý takýmto spôsobom nie je nutné ďalej spracovávať a stačí ho iba odoslať na výstup modulu.
V prípade generovania vlastného XML dokumentu (kód č.7), a teda použitia triedy XMLTextWriter, modul pristupuje na inštanciu tohto objektu prostredníctvom členskej premennej *streamer* a jej atribútu *Streamer*:

```
streamer.Streamer.WriteStartElement("Content");
streamer.Streamer.WriteAttributeString("elementId",(string)
Config["elementId"]);
streamer.Streamer.WriteAttributeString("height",
(string)Config["height"]);
streamer.Streamer.WriteEndElement();
```

Kód 7. - Ukážka kódu na zápis XML

Tento typ vytváranie výstupného dokumentu je rednastavený každému modulu implicitne.

V druhom prípade, teda **pri zapisovaní textového reťazca na výstup modulu** (kód č.8), používame atribút *stringStreamer* tej istej členskej premennej:

```
streamer.stringStreamer.Write(responseString);
```

Kód 8. - Priame vkladanie XML kódu

V tomto prípade však je nutné používanie spomenutého formátu zápisu modulu explicitne oznámiť v tele konštruktora objektu nasledovne:

```
// nastavuje sposob vytvarania xml streamu - vytvorenie textoveho
retazca
streamer.streamType = XmlStreamerType.STRING_WRITER;
```

Kód 9. - nastavenie typu vkladania XML kódu

Objekt *stringStreamer* je typu *StringWriter*, čo je štandardná trieda platformy .NET. V prípade definovania používania zápisu vo formáte textového reťazca je povolený iba tento typ zápisu počas celého vytvárania výstupného XML dokumentu.

Vyššie spomenutý ukážkový kód je potrebné umietniť do tela metód *insertDataToStream()* a *postParseOperations()*.

Čítanie údajov z dátovodu

Čítanie údajov z dátovodu môže prebiehať podobne ako pri zápise dvoma spôsobmi, a to spracovávaním SAX udalostí pri prechádzaní XML dokumentu, alebo prístupom k celému dokumentu naraz, ktorý je potom možné spracovávať ľubovoľne inak, napríklad naň aplikovať vlastné transformačné algoritmy.

V prípade spracovania SAX udalostí, ktoré je modulom na spracovanie vstupných údajov nastavené implicitne, je nutné vytvoriť si vlastnú triedu na spracovanie SAX prúdov (kód č. 10), a to nasledovným spôsobom:

```
class MySAXHandler : AbstractSAXHandler
{
   private XmlTextWriter streamer;

   public MySAXHandler(XmlTextWriter streamer)
   {
      this.streamer = streamer;
   }

   public override void StartDocument()
   {
   }

   public override void EndDocument()
   {
   }

   public override void StartElement(string uri, string name, string
qName, IAttributes atts)
   {
   }

   public override void EndElement(string uri, string name, string
qName)
   {
   }

   public override void Characters(char[] ch, int start, int length)
   {
      string str = new string(ch, start, length);
      streamer.WriteString(str);
   }
}
```

Kód 10. - Trieda na spracovanie SAX prúdov

Túto triedu je najlepšie vytvoriť ako anonymnú, nakoľko nemá zmysel, aby bola prístupná zvonku implementovaného modulu. Vo výpise kódu je zoznam metód, ktoré je nutné povinne preťažiť pri vytváraní tried na spracovanie SAX prúdov. Je možné preťažiť aj iné metódy, ich zoznam je možné nájsť v dokumentácii ku knižnici Aelffred, prípadne prostredníctvom Prieskumníka objektov v aplikácii MS Visual Studio 2005.

Pri použití vlastného handlera na SAX udalosti je potrebné pridať si ho do členskej premennej triedy modulu (kód č. 11). Pre správne inicializovanie handlera je potrebné do tela konštruktou vložiť nasledovný kód:

```
private MySAXHandler handler;

handler = new MySAXHandler(streamer.Streamer);
reader.ContentHandler = handler;
reader.ErrorHandler = handler;
```

Kód 11. - Inicializovanie handlera SAX prúdov

Tento kód zabezpečí, aby bol pri spracovaní XML dokumentu pomocou vyvolávania SAX udalostí používaný vlastný SAX handler.

V prípade, že **chceme mať v module priamy prístup na celý vstupný XML dokument** (kód č.12), musíme túto skutočnosť explicitne oznámiť modulu zadaním nasledovného príkazu do tela konštruktoru modulu:

```
reader.streamType = GeneralXMLReaderType.STRING_READER;
```

Kód 12. – Nastavenie objektu na spracovanie XML pre priamy prístup na vstupný prúd

V takomto prípade sa k vstupným dátam (kód č.13) dostaneme prostredníctvom atribútu *totalInputXML*, členskej premennej *reader*, napríklad takto:

```
MemoryStream strm = reader.totalInputXML;
```

Kód 13. - Prístup k vstupnému XML prúdu

Z vyššie napísaného vyplýva, že ako vstup nám je poskytnutý objekt typu *MemoryStream*, čo je štandardná trieda platformy .NET. Pre konverziu na textový reťazec (kód č.14) je možné použiť členskú metódu modulu , ktorá je nazvaná *streamToString()*.

```
string input = streamToString(reader.totalInputXML);
```

Kód 14. - Vytvorenie reťazca zo vstupného XML prúdu

Pridávanie vlastných kusov kódu do výsledného XHTML dokumentu

Keďže výstup aplikácie Coonda je vo formáte XHTML, je vhodné, aby moduly mohli zasahovať do jeho výslednej štruktúry aspoň v nejakej obmedzenej forme, napríklad pri potrebe načítania vlastnej CSS šablóny alebo vlastného Javascriptu do tela výsledného dokumentu. K tomuto účelu slúžia (v súčasnej dobe) dve polia typu *HashTable*, ktoré sú uložené v zdieľanej pamäti modulov.

Týmto spôsobom je teda možné pridať vlastné xhtml tagy do hlavičky xhtml dokumentu prostredníctvom poľa *CustomHeaderTags* (kód č.15) a taktiež vlastné volania funkcií pri zobrazení dokumentu, teda volania typu *<body onload="sampleFunction();">*.

Použitie je nasledovné:

```
((Hashtable)sharedMemory["customBodyLoadEvents"]).Add("Gmaps_js",
"initialize()");
```

Kód 15. - Prístup do výsledného HTML dokumentu

1.4.6 Skompilovanie dynamickej knižnice.

Ak boli pri vytváraní knižnice použité iba štandardné vyššie spomenuté metódy, mal by sa projekt úspešne skompilovať a jeho výstupom by mal byť súbor s názvom **nazov_triedy_modulu*.dll* , pričom výraz uzatvorený v hviezdičkách predstavuje názov hlavnej triedy modulu, vytvorenej v projekte.

1.4.7 Pridanie vytvorenej knižnice do aplikácie Coonda a nakonfigurovanie modulu.

Takto vytvorenú knižnicu je následne potrebné skopírovať do priečinku *bin/modules*, ktorý sa nachádza v priečinku s aplikáciou *CoondaWeb*. Následne je potrebné ho zaregistrovať do konfigurácie aplikácie Coonda. Tento proces je bližšie popisovaný v používateľskej príručke. V prípade, ak modul využíva externé súbory, zvyknú sa tieto kopírovať do priečinku */res* ktorý sa nachádza vo vyššie spomenutom priečinku *CoondaWeb*.

1.4.8 Inštalácia šablón pre používateľské rozhranie

Pre každú stránku definovanú v *sitemap.xml* je potrebné deklarovať a vytvoriť šablónu. Šablóna sa deklaruje pomocou atribútu `layout` v tagu `page`, ktorý ukazuje na adresár v adresári *layouts*, v ktorom sa nachádza súbor *layout.ctp*.

```
<page id="mainpage" layout="mainpage" title="Coonda Main page">
```

Kód 10. - Príklad definície stránky v sitemap.xml

Layout.ctp by mal obsahovať (podľa potreby) 3 "tagy" `{{custom_header_tags}}`, `{{custom_body_load_events}}`, `{{content}}`, ktoré budú nahradené obsahom vygenerovaným jednotlivými aplikáciami.

`{{custom_header_tags}}` je nahradený obsahom zo zdieľanej pamäte medzi modulmi s kľúčom `''customHeaderTags"`, určeným na vloženie potrebného obsahu väčšinou do hlavičky html stránky.

`{{custom_body_load_events}}` je nahradený obsahom zo zdieľanej pamäte medzi modulmi s kľúčom `''customBodyLoadEvents"`, určeným na vloženie potrebného obsahu väčšinou do atribútu onload tagu body html stránky.

`{{content}}` je nahradený výstupom z aplikácií a dodatočnými informáciami ku stránke vo formáte JSON. Štruktúra objektu vyzerá nasledovne.

```
{
      "page_name": "page2",
      "user_id": "",
      "page_title": "Second page",
      "widgets": [
                  {
                        "id": "Login",
                        "title": "Login",
                        "content": "obsah stránky",
                        "position": "2"},
                  {
                        "id": "NavMenu",
                        "title": "NavMenu",
                        "content": "obsah" ,
                        "position": "0"}
                  ]
};
```

Atribúty

- `page_name` – unikátny identifikátor stránky
- `user_id` – login prihláseného používateľa
- `page_title` – názov stránky
- `widgets` – pole objektov, ktoré držia obsah a pomocné atribúty k aplikáciami pre túto stránku
    - `id` – unikátny identifikátor aplikácie
    - `title` – názov aplikácie
    - `content` – obsah vygenerovaný danou aplikáciou
    - `position` – pozícia aplikácie v rámci stránky

1.4.9 Jednoduchý príklad šablóny

Nižšie uvedená šablóna spĺňa všetky požiadavky na úspešné naplnenie výstupom z jadra. Premenná *page_data* môže byť ďalej spracovávaná JavaScriptom za účelom vytvorenia vlastnej šablóny.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE      html      PUBLIC      "-//W3C//DTD      XHTML      1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
      <title>Coonda ShowCase</title>
      {{custom_header_tags}}
      <script type="text/javascript">
            var page_data = {{content}};
      </script>
</head>
<body onload="{{custom_body_load_events}}">
</body>
</html>
```

Kód 12. - Príklad najjednoduchšej šablóny

1.4.10 Vstavaná šablóna

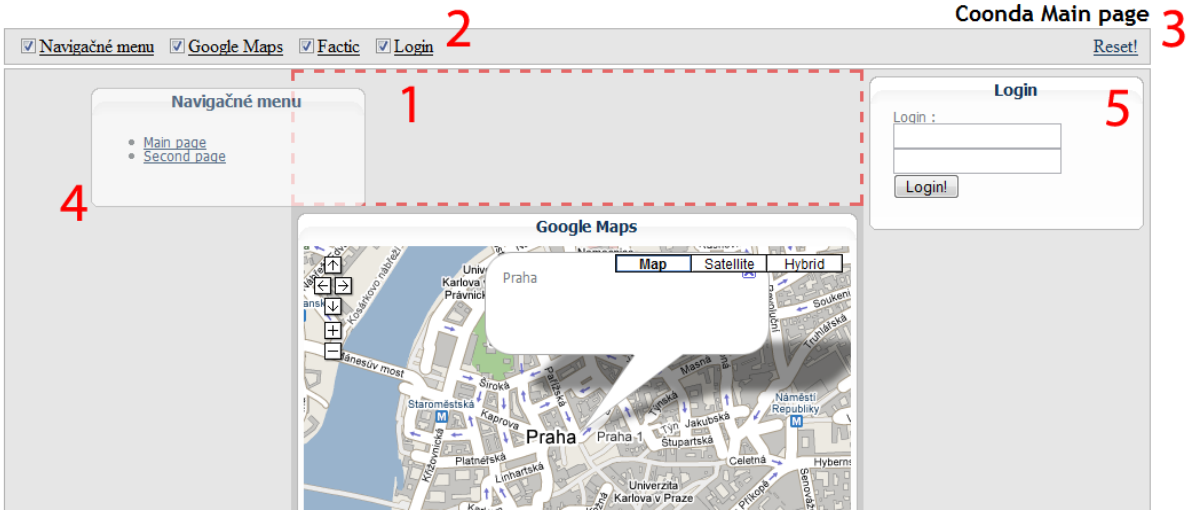
Vytvorili sme šablónu pre ukážkové predvedenie rámca. , ktorá spracováva obsah z jadra a podporuje interaktívne pozicovanie, zobrazovanie a ukladanie aktuálneho rozloženia jednotlivých aplikácií na stránke. Nachádza sa v adresári layouts/mainpage. Pre jej použite stačí skopírovať obsah adresára do nového z názvom stránky, pre ktorú sa má použiť. V layout.ctp v časti head je potrebné ešte zmeniť cestu ku štýlom a skriptom, nakoľko sme šablónu preniesli do nového adresára.

Príloha B: Používateľská príručka

1.1 Používateľské rozhranie portálu

Používateľské rozhranie je postavené na výstupe z jadra. Produkt obsahuje vstavanú šablónu, ktorá spracováva tento obsah a podporuje interaktívne pozicovanie, zobrazovanie a ukladanie aktuálneho rozloženia jednotlivých aplikácií na stránke.

1. **Premiestňovanie aplikácií**
   Aplikácie je možné premiestňovať spôsobom drang-and-drop, teda pretiahnutím za hornú časť z jedného stĺpca do druhého. Aktuálna konfigurácia je automaticky uložená a pri zobrazení znovu zobrazení stránky načítaná.
2. **Zobrazovanie/skrývanie aplikácií**
   Horné menu umožňuje aplikáciu skryť a následne znovu zobraziť. Aktuálna konfigurácia je automaticky uložená a pri zobrazení znovu zobrazení stránky načítaná.
3. **Resetovanie pozícií aplikácií**
   V pravom rohu horného menu je odkaz "Reset!", slúži na návrat k pôvodnej konfigurácii rozhrania.
4. **Navigačné menu**
   Táto aplikácia slúži na navigáciu po ďalších stránkach portálu.
5. **Login**
   Táto aplikácia slúži na prihlasovanie do portálu.



Obrázok 28. - Používateľské rozhranie

1.2 Používateľské administračné rozhranie

Administračné rozhranie je systém, ktorý slúži na správu a konfiguráciu portálu ako aj jeho súčastí. Na začatie práce s administračným rozhraním je potrebné sa prihlásiť pomocou prihlasovacieho

formulára, ktorý sa zobrazí v internetovom prehliadači po zadaní nasledovnej adresy: http://meno.servera/CoondaWeb/. Prihlásenie je umožnené iba používateľom, ktorí sú zaradení do skupiny administrátorov. Je potrebné zadať prihlasovacie meno a heslo.

Obrázok 29. - Prihlasovací formulár

Samotný systém je rozdelený do častí, z ktorých každá slúži na úpravu konkrétnej časti systému. Administračné rozhranie pozostáva z nasledovných častí:

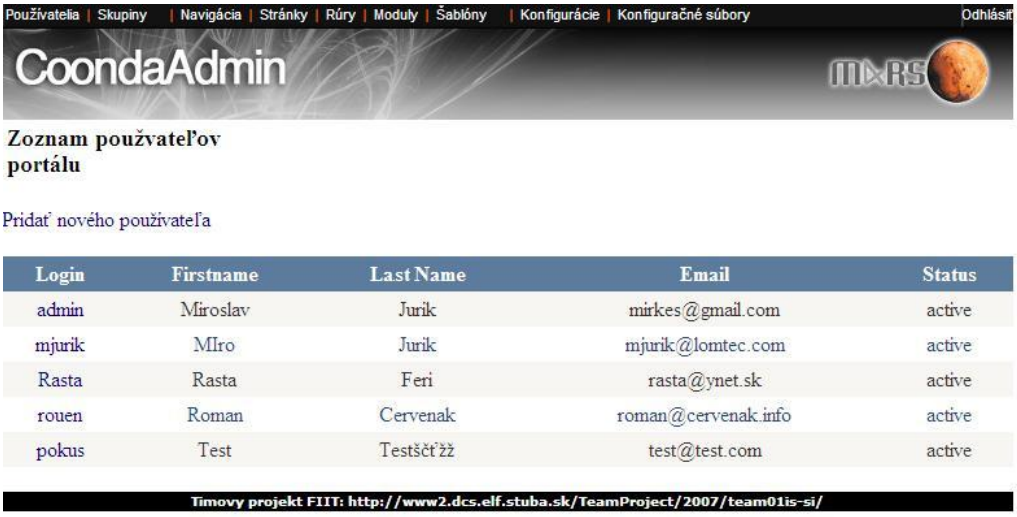
1. Správa používateľov
2. Správa skupín
3. Pozície
4. Navigácia
5. Správa stránok
6. Správa modulov
7. Správa rúr
8. Rozloženie stránok
9. Správa konfigurácií
10. Správa konfiguračných súborov

Všetky tieto časti administračného rozhrania sú prístupné z menu, ktoré sa nachádza vo vrchnej časti stránky. Pre niektoré časti rozhrania je prístupné aj menu nižšej úrovne, ktoré umožňuje úpravu ďalších vlastností konkrétnej entity.

Obrázok 30.- Administračné rozhranie s hlavným menu a podmenu pre entitu "Skupiny"

1.2.1 Používateľské rozhranie

Administračné rozhranie ponúka viacero zobrazení jednotlivých záznamov. V časti, kde je potrebné zobraziť všetky záznamy je použité tabuľkové zobrazenie s implementovaným stránkovaním. Každý riadok v tabuľke obsahuje sumárne údaje o zázname a ikonu na úpravu konkrétneho záznamu ako aj odkaz na vloženie nového záznamu (v priestore nad tabuľkou).



Obrázok 31.- Tabuľkové zobrazenie všetkých používateľov

Pre úpravu je použitá sada kontrol podľa upravovaného záznamu. Logické časti sú združené do rámikov, čo prispieva k sprehľadneniu celého administračného rozhrania.

Obrázok 32.- Použitie rôznych kontrol na úpravu alebo vloženie záznamu

1.2.2 Správa používateľov

V tomto zobrazení ma správca systému prehľad o všetkých používateľoch, zaregistrovaných v systéme. V tabuľke používateľov sú zobrazené sumárne informácie ako prihlasovacie meno, meno, priezvisko, e-mailová adresa a stav, ktorý špecifikuje, či daný používateľ je aktívny, teda či má disponuje právom prihlásenia sa do portálu, alebo je mu prístup na portál zakázaný. Pre úpravu existujúceho používateľa je potrebné kliknúť na prihlasovacie meno konkrétneho používateľa v tabuľke všetkých používateľov.

Pomocou odkazu „Pridať nového používateľa“ je možné vytvoriť nový záznam o používateľovi. V obrazovke, ktorá sa zobrazí po kliknutí na spomenutý odkaz, je potrebné vyplniť všetky údaje, inak ho rozhranie upozorní na nevyplnenie potrebného údaju. Používateľ údaje uloží stlačením tlačidla „Uložiť“, tlačidlom „Vymazať“ vymaže už existujúci záznam o používateľovi a tlačidlom „Zrušiť“ sa vráti na zobrazenie tabuľky so záznamami všetkých používateľov.

1.2.3 Správa používateľských skupín

V tomto náhľade je zobrazený prehľad všetkých používateľských skupín, ktoré majú za úlohu zjednocovať zobrazenie portálu a navigácie, špecifikovať oprávnenia na zobrazenie dátovodov. Rozhranie na manipuláciu s používateľskými skupinami (pridávanie, úprava) je podobné ako pri správe používateľov. Pri úprave existujúceho záznamu o používateľskej skupine je možné upraviť záznamy o menu položkách (navigácia) kliknutím na linku „Menu“, ktorá sa nachádza v podmenu

zobrazenia. Rovnako sa tu nachádza aj odkaz na úpravu oprávnení používateľských skupín na dátovody. Dátovody, na ktoré bude mať daná skupina právo zobrazenia, administrátor označí zo zoznamu všetkých dátovodov a tlačidlom „Uložiť“ uloží oprávnenia pre danú skupinu. Používateľ má oprávnenie na zobrazenie konkrétneho dátovodu podľa toho, do ktorej používateľskej skupiny patrí.

1.2.4 Pozície

Portál je rozdelený do viacerých častí, v ktorých je zobrazovaný výstup jednotlivých dátovodov. V tomto zobrazení je možné definovať pozíciu zobrazenia dátovodu na portáli podľa používateľskej skupiny. Týmto spôsobom je možné prispôsobiť rozmiestnenie zobrazení s dátovodmi presne podľa požiadaviek jednotlivých používateľov v používateľských skupinách.

1.2.5 Navigácia

Toto zobrazenie umožňuje definovať menu položky pre jednotlivé používateľské skupiny. Používateľ v skupine ma tak presne definovaný zoznam položiek na stránke, ktoré sú pre neho relevantné. Administrátor určuje názov menu položky, url, na ktorú sa položka odkazuje a popis.

1.2.6 Správa stránok

V tejto časti sa definujú stránky spolu s ich grafickými rozložením a zobrazenými dátovodmi. Administrátor definuje pre každú stránku ako má vyzerať (rozloženie, šablónu) a aké dátovody majú byť zobrazené na stránke. Pre určenie dátovodov, ktoré majú byť zobrazené na danej stránke, je potrebné kliknúť na linku „Rúry“ v menu druhej úrovne. Po kliknutí na linku „Stránka“ sa zobrazí úprava údajov o stránke (názov, rozloženie a popis).

1.2.7 Správa modulov

Toto zobrazenie slúži na definovanie modulov, ktoré sú prístupné v portáli. Modulu je možné definovať názov, typ (generátor, serializér, transformátor) a názov knižnice s funkcionalitou pre daný modul.

1.2.8 Správa dátovodov

Táto časť administračného rozhrania zobrazuje zoznam všetkých dátovodov ako aj jednotlivé moduly pre konkrétny dátovod.

1.2.9 Rozloženie stránok

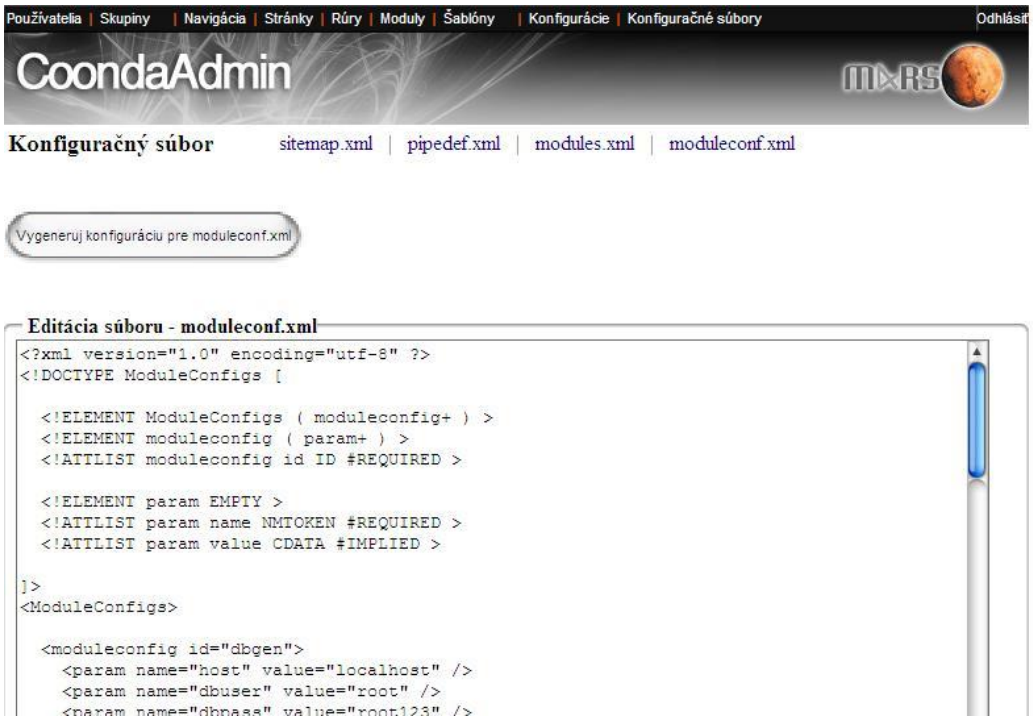
V tejto časti administrátor definuje záznamy o rozvrhnutí stránky. Pre rozvrhnutie stránky sa definuje názov, typ a popis rozvrhnutia.

1.2.10 Správa konfigurácií

Správa konfigurácií umožňuje vytvárať, upravovať a mazať konfiguračné záznamy. Každý konfiguračný záznam pozostáva z názvu a parametrov, ktorých je možné vytvoriť aj viacero. Tieto konfigurácie sa prideľujú dátovodom.

1.2.11 Správa konfiguračných súborov

V tomto zobrazení ma administrátor možnosť manuálne meniť konfiguračné súbory. Avšak takýto postup nie je odporúčaný. Mohlo by prísť k Na úpravu, generovanie konfiguračných súborov je tlačidlo „Vygeneruj konfiguráciu“, ktoré na základe záznamov v administračnom rozhraní vygeneruje a uloží zvolený konfiguračný súbor.



Obrázok 5 .- Prehľad XML konfiguračných súborov s možnosťou ich generovania