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. Juríik 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. Juríik 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

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/

[5] XML modul pre DotNetNuke, stránka projektu http://www.dotnetnuke.com/Products/Development/Projects/ModuleXML/tabid/836/Default.aspx

[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 špecifik 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

3.1.1.1 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.

3.1.1.2 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.

3.1.1.3 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

3.1.2.1 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.

3.1.2.2 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.

3.1.2.3 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.

3.1.2.4 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

3.1.3.1 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.

3.1.3.2 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.

3.1.3.3 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

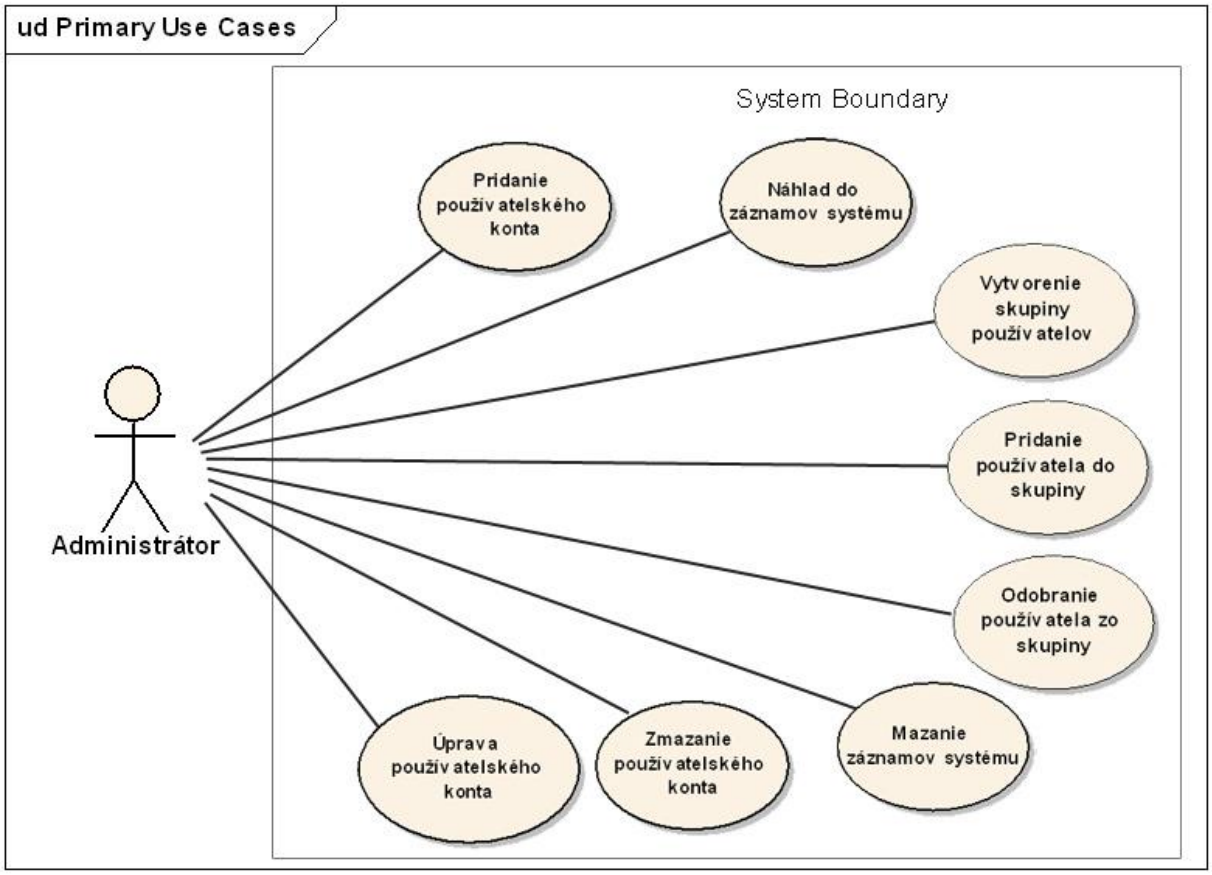
Portálový rámec je určený pre systém Windows s podporou .NET framework .

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.

3.3.1.1 Správa používateľov

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

3.3.1.2 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.

3.3.1.3 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.

3.3.1.4 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.

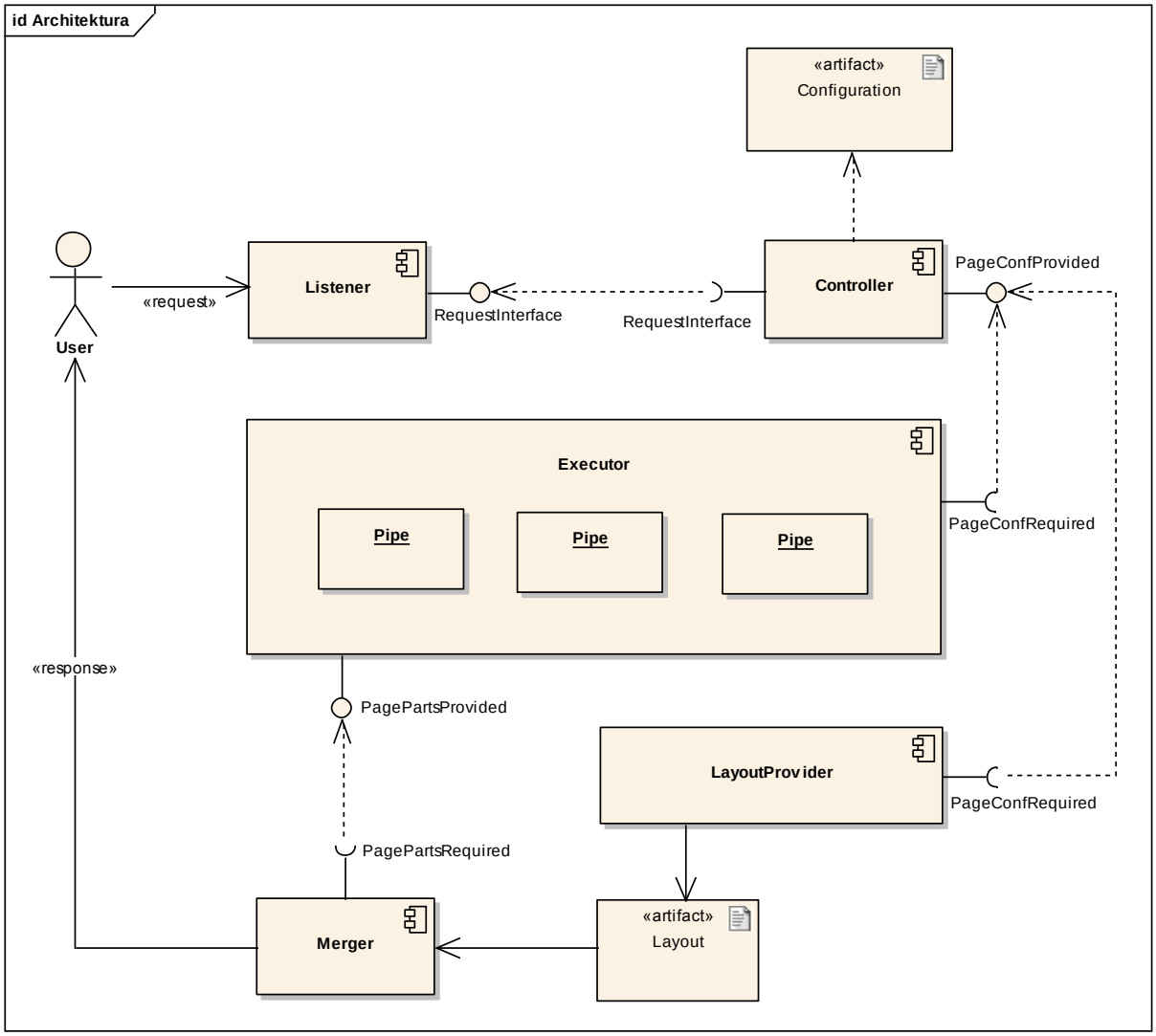
4. Hrubý Návrh riešenia

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.

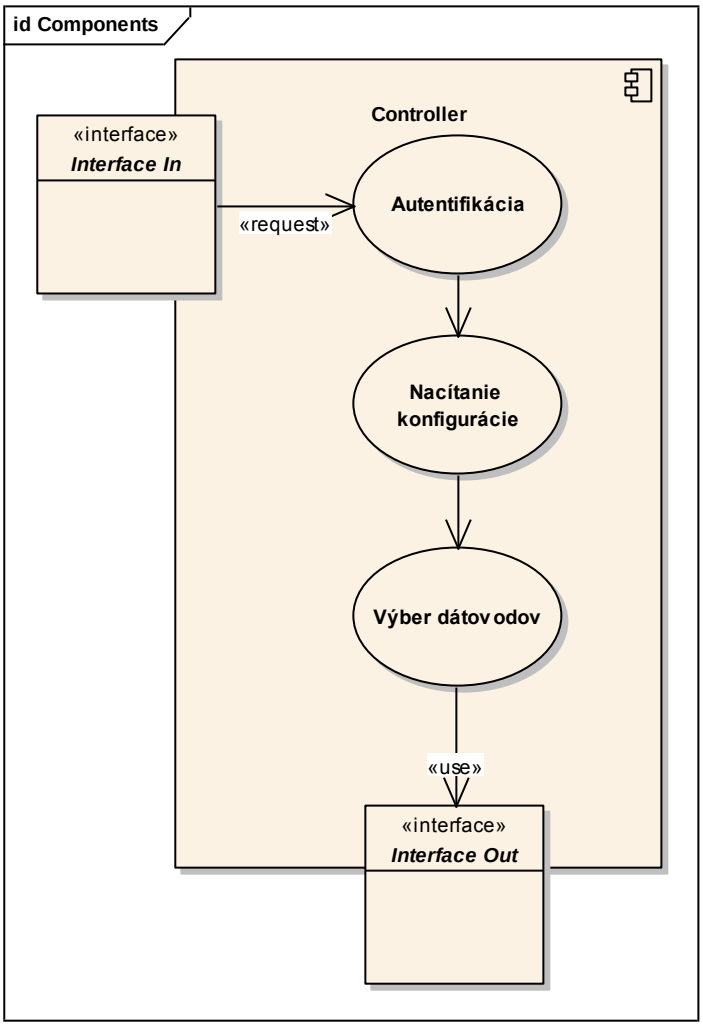
Implementácia komponentu je v súčasnej fáze návrhu systému otvoreným problémom, pričom pripadajú do úvahy dve hlavné alternatívy :

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.

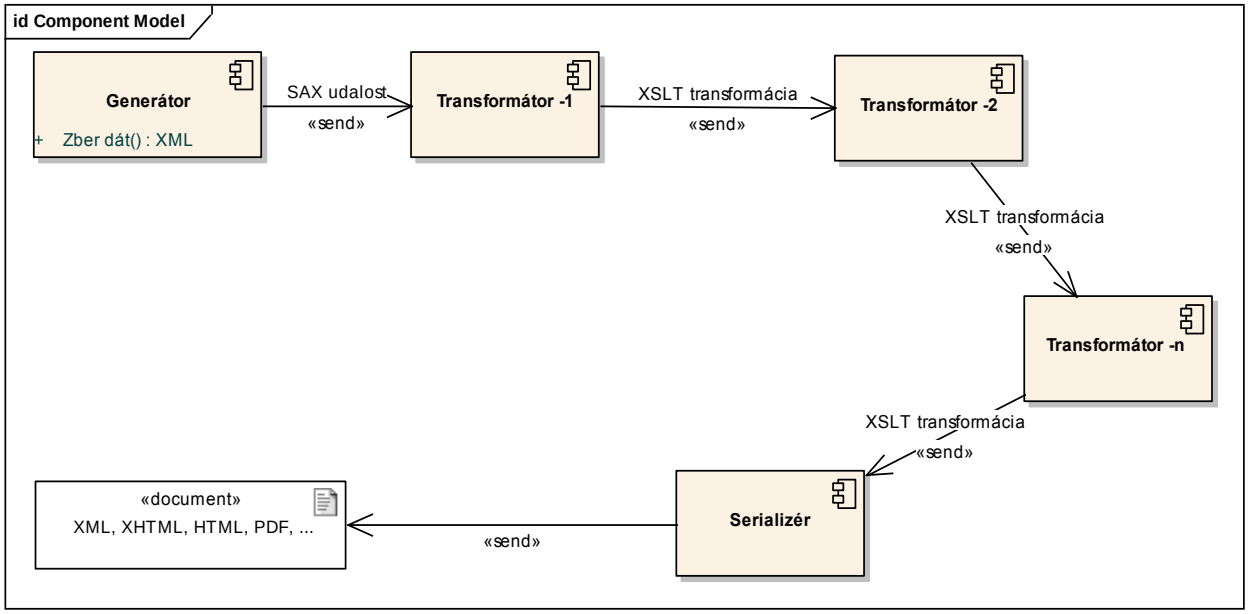
V prípade, že z Executora bude vyslaný prúd údajov v inom formáte ako je XML, bude celý tento prúd presmerovaný na používateľský výstup bez akýchkoľvek zmien. V takomto prípade sa predpokladá prúd údajov iba z jediného dátovodu.

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.

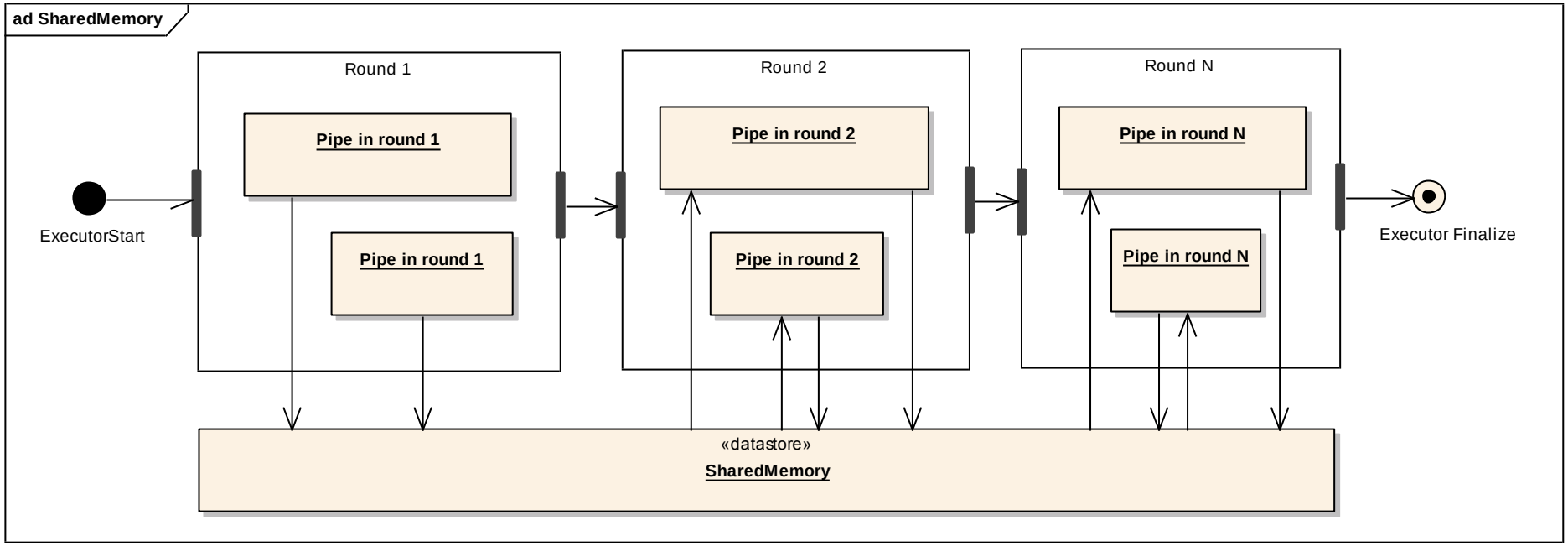
Výber rozhrania

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.

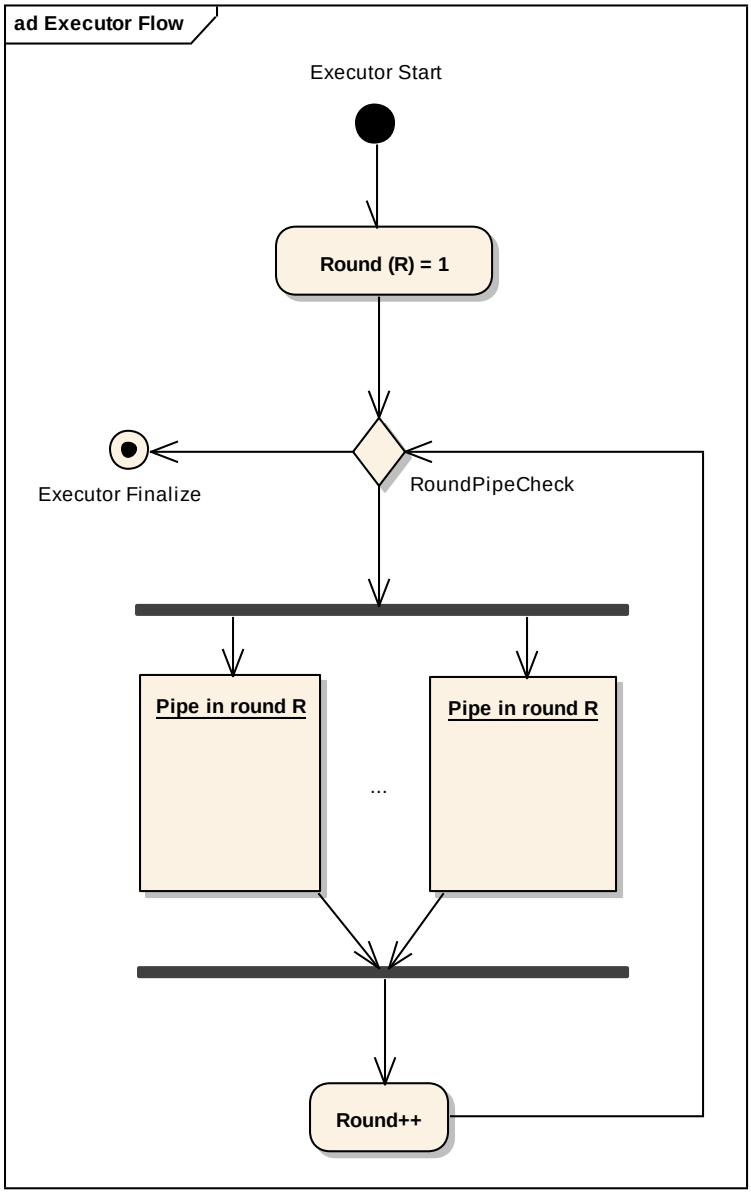
4.4 Komunikácia modulov

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

4.5.1.1 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.

4.5.1.2 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.

4.5.1.3 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.

4.5.1.4 Dizajnérske šablóny

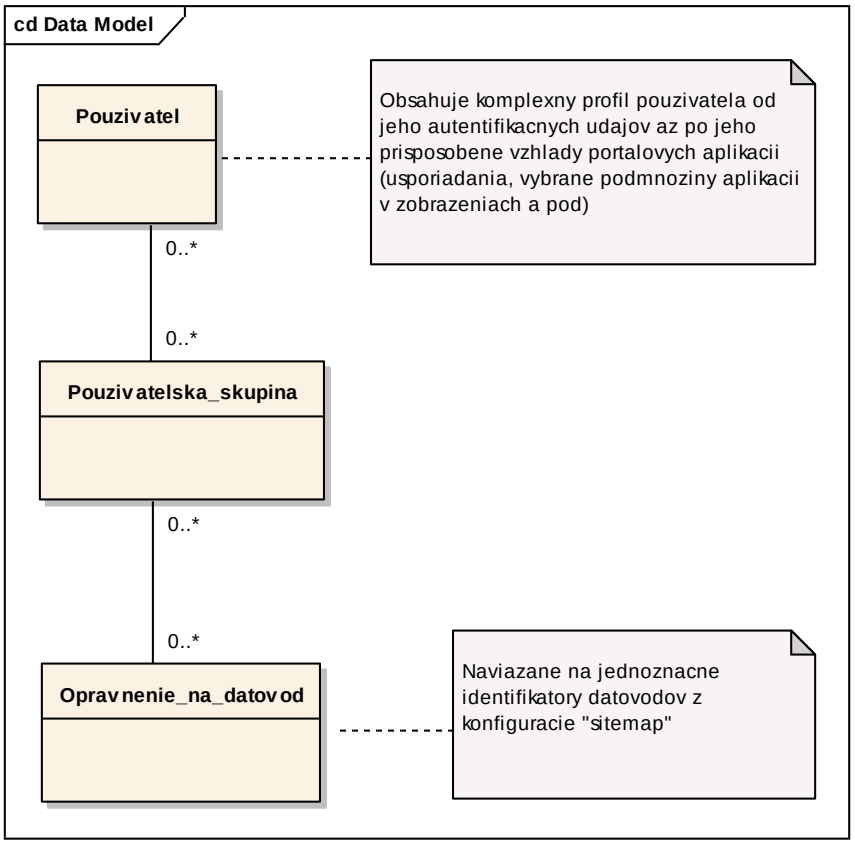
Dizajnérmi definované šablóny na rozloženie jednotlivých komponentov portálu používané pri finálnej serializácii na XHTML výstup budú ukladané vo forme XSLT štýlov, teda v podstate taktiež ako súbory XML.

4.5.2 Logický model údajov relačnej databázy

4.5.2.1 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

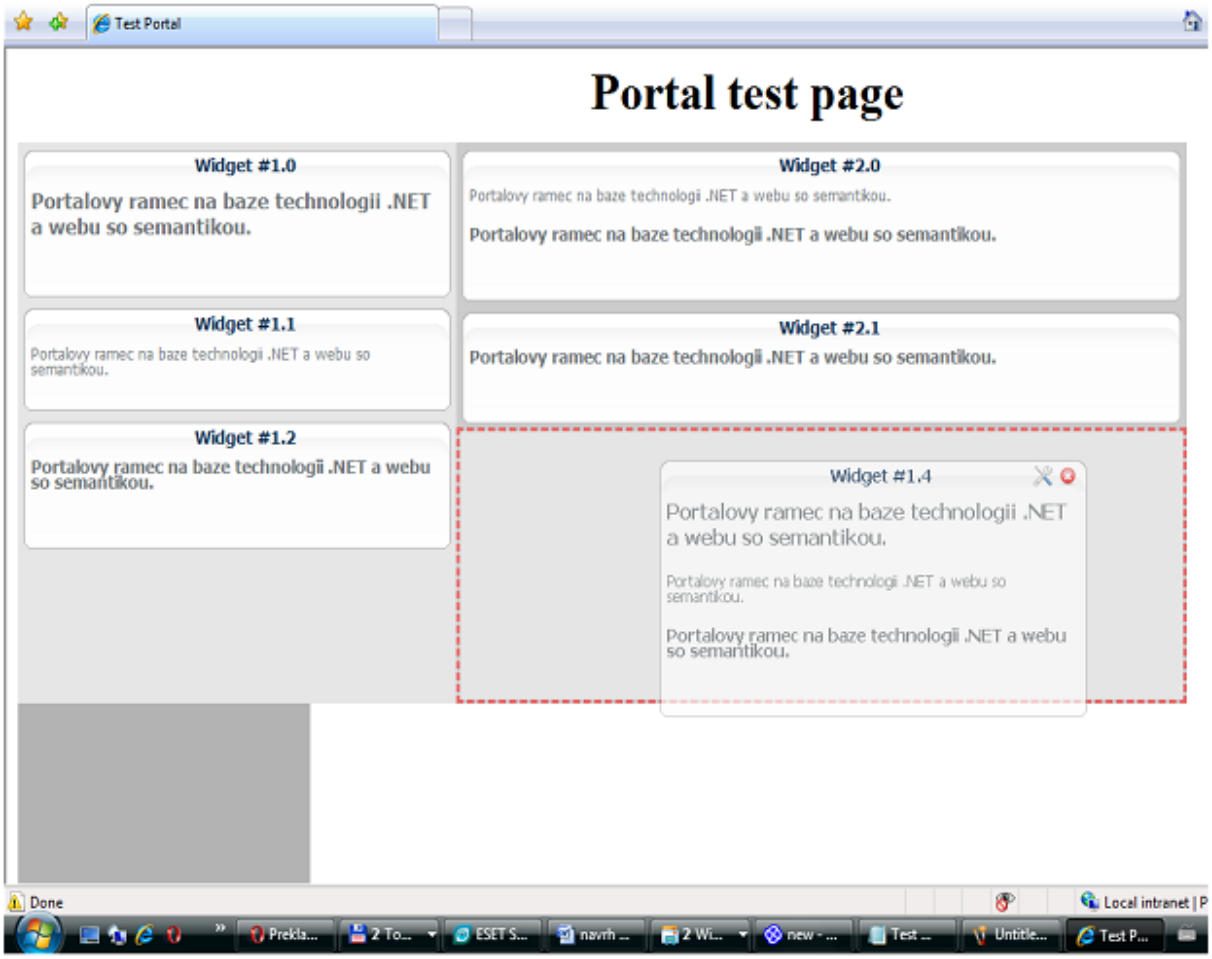
Prezentačnú vrstvu systému bude tvoriť množina samostatných aplikácií, ktorých rozmiestnenie a vzhľad na stránke si bude môcť používateľ na základe pridelených právomocí a spôsobu schémy rozmiestnenia upravovať . Tieto schémy rozmiestia pre danú používateľskú skupinu bude vytvárať systémový dizajnér.

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

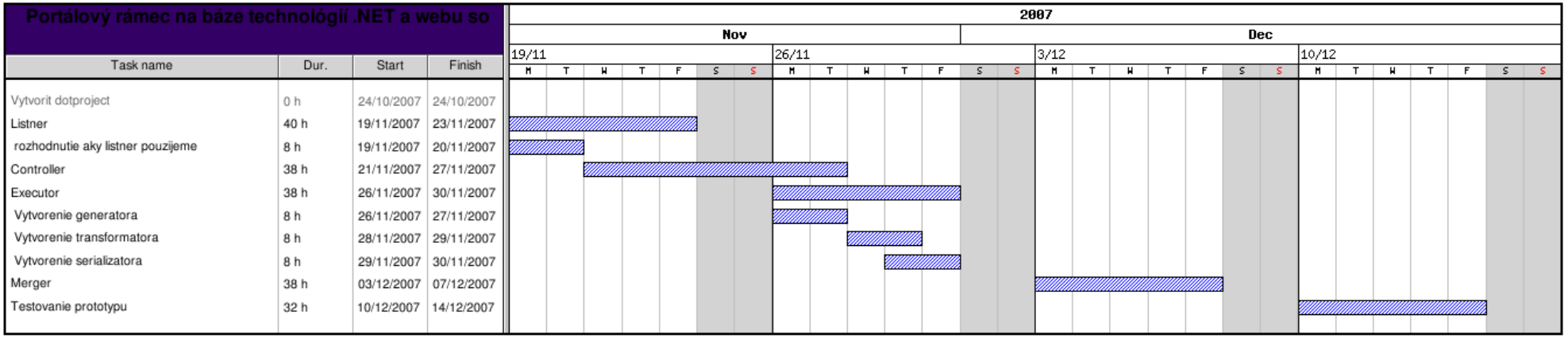
Cieľom tejto kapitoly bolo navrhnúť systém, ktorý spĺňa všetky body zhrnuté v špecifikácii projektu a navrhnúť dostatočne flexibilnú architektúru, ktorá dokáže reagovať na čo najväčšiu škálu požiadaviek na tvorbu konkrétnych portálov na báze tejto architektúry. Navrhovaný systém sme dekomponovali na 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