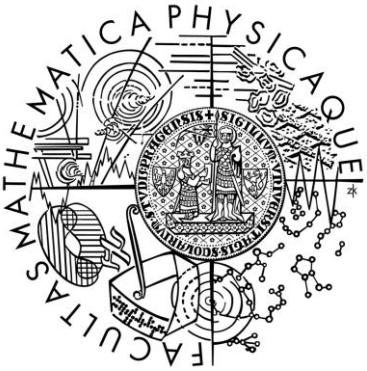
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Jindřich Flídr

Fashionspace Portal

Kabinet software a výuky informatiky

Vedoucí diplomové práce: RNDr. Tomáš Holan, Ph.D.

Studijní program: Informatika

Studijní obor: Softwarové systémy

Praha, rok 2011

Děkuji vedoucímu diplomové práce RNDr. Tomášovi Holanovi, Ph.D. za cenné rady, připomínky a metodické vedení práce.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 4. 4. 2011 Jindřich Flídr

Název práce: Fashionspace Portál
Autor: Jindřich Flídr
Katedra / Ústav: Kabinet software a výuky informatiky
Vedoucí diplomové práce: RNDr. Tomáš Holan, Ph.D.

Abstrakt: V rámci práce byla navrhnuta architektura a implementována kostra webové aplikace „Fashionspace“. Pomocí této aplikace je možné v interaktivním prostředí vytvářet a sledovat virtuální módní přehlídky v podobě videa. Klientská aplikace je implementována pomocí platformy Adobe Flash a pomocí služeb komunikuje s CMS Drupal. V rámci aplikace je integrována řada technologií, jako je online komunikace uživatelů využívající protokol XMPP nebo streaming HD videa. Architektura aplikace je modulární a je postavena na principech MVC.

Klíčová slova: webové technologie, RIA, Adobe Flex, CMS Drupal, video streaming

Title: Fashionspace Portal
Author: Jindřich Flídr
Department: Department of Software and Computer Science Education
Supervisor: RNDr. Tomáš Holan, Ph.D.

Abstract: The aim of this thesis is to design architecture and implement web application “Fashionspace”. This application allows user to create virtual fashion shows based on videos in an interactive environment. The client application is implemented using the Adobe Flash platform and communicates with the CMS Drupal using services. The application is integrated with many technologies, such as online user communication using the XMPP protocol or HD video streaming. The architecture of this application is modular and is built on the principles of MVC.

Keywords: web technologies, RIA, Adobe Flex, CMS Drupal, video streaming

Obsah

1. Úvod

Cílem práce je navrhnout a implementovat webovou aplikaci pro virtuální módní přehlídky, které v systému prezentuje předpřipravené video, s komentáři a možností chatu uživatelů v reálném čase. V rámci návrhu architektury aplikace je nutné zejména vyřešit zpracování videa, a to od jeho vložení až po živé vysílání v HD kvalitě. Dále je nutné integrovat robustní řešení v podobě chatovacího serveru a navrhnout systém pro tvorbu interaktivního prostředí.

Celý systém je dále nutné navrhnout tak, aby ho bylo možné snadno rozšiřovat a upravovat. Úpravou je myšleno použití projektu v jiných oblastech, než je móda. Systém je nutné rozdělit tak, aby zvládal co nejlépe vysokou návštěvnost.

Cílem práce není vytvořit kompletní funkční web, ale připravit prototyp, v rámci kterého budou vyřešeny hlavní problémy, s nimiž by se realizace podobného systému potýkala. V celé práci je kladen větší důraz na použité technologie, architektonický návrh a jeho další rozšiřování, než na samotnou implementaci aplikace.

Webová aplikace bude rozdělena na klientskou a uživatelskou část. Aplikace běžící na serveru bude postavena na CMS Drupal, který bude rozšířen zejména pomocí nových modulů. Klientská část aplikace bude vytvořena pomocí technologie Adobe Flash a frameworku Flex SDK.

Kromě těchto základních částí systému, které je nutné implementovat tak, aby splňovaly zadání, je zásadní částí práce integrace služeb třetích stran, které poskytují již hotová řešení, například v oblasti streamování videa, jeho konverze a zálohování.

2. Technologie

Architektura projektu je postavena na modelu klient/server. Podrobněji je tato problematika rozebrána v páté kapitole, která se věnuje architektuře. Z technologického pohledu je nejzajímavějších následujících pět technologií v projektu použitých. Jedná se o CMS Drupal na straně serveru, dále se jedná o platformu Adobe Flash na straně klienta. Komunikaci mezi klientem a serverem zajišťuje protokol AMF. Pro online komunikaci mezi uživateli je použit protokol XMPP, dříve známý jako Jabber. Pro přehrávání videa a streamování slouží knihovna OSMF. Výběru konkrétních technologií a dále zvažovaným alternativám se věnují kapitoly šest až osm.

2.1. CMS Drupal

CMS Drupal je Open Source systém pro správu obsahu naprogramovaný v jazyce PHP. Drupal původně napsal Dries Buytaert. První verze byla vytvořena v roce 2001. Poslední verze 7.x byla představena na začátku roku 2011. Aplikace využívá verzi 6.x, která byla v době vzniku práce aktuální.

Drupal umožňuje snadnou tvorbu internetových aplikací, jako jsou online magazíny, internetové obchody, blogy nebo jiné komplexní webové systémy. Drupal je postaven modulárním způsobem a využívá principy MVC. Jako svou filozofii udává přehlednost kódu a otevřenost API.

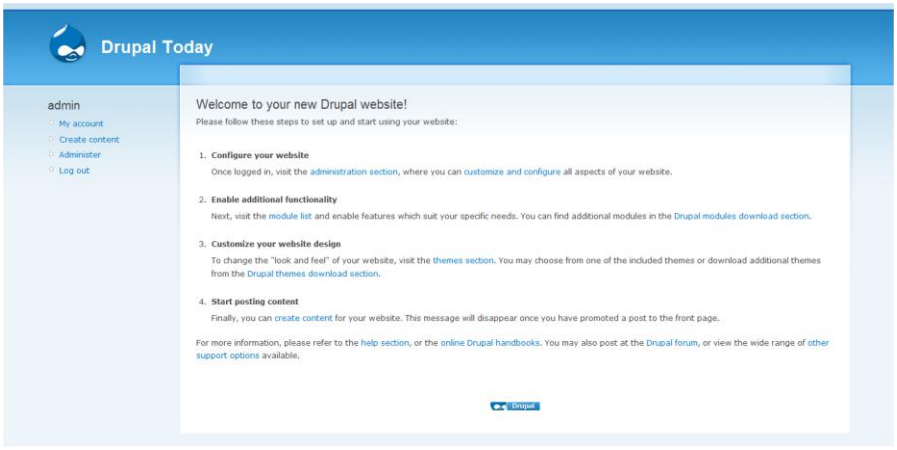
Moduly je možné stáhnout na stránkách Drupalu a rozšiřují jeho funkčnost do oblastí, jako je například obchod a vzdělávání. Počet podporovaných a vyvíjených modulů se pohybuje v řádu tisíců. Drupal je pro implementaci webu využíván například webem úřadu prezidenta Spojených států amerických Whitehouse.gov.

Kromě Drupalu existuje celá řada kvalitních konkurenčních řešení, jako je například Mambo nebo Joomla. Pro každý projekt je vhodný jiný systém. Drupal byl vybrán,

protože nejlépe splňuje požadavky aplikace Fashionspace. Řada požadavků na systém jde často proti sobě. Klíčové požadavky na portál Fashionspace jsou následující:

- Široké možnosti systému jsou důležitější než jeho snadná ovladatelnost
- Flexibilní a robustní architektura je důležitější než její jednoduchost
- Systém je obecný a snadno rozšiřitelný, ne jen specificky zaměřený

Systém bude dále rozšiřován jak celou řadou modulů třetích stran, tak i modulů vlastních, a proto jsou předchozí body velice důležité.



Obrázek 1 - Ukázka základní instalace CMS Drupal

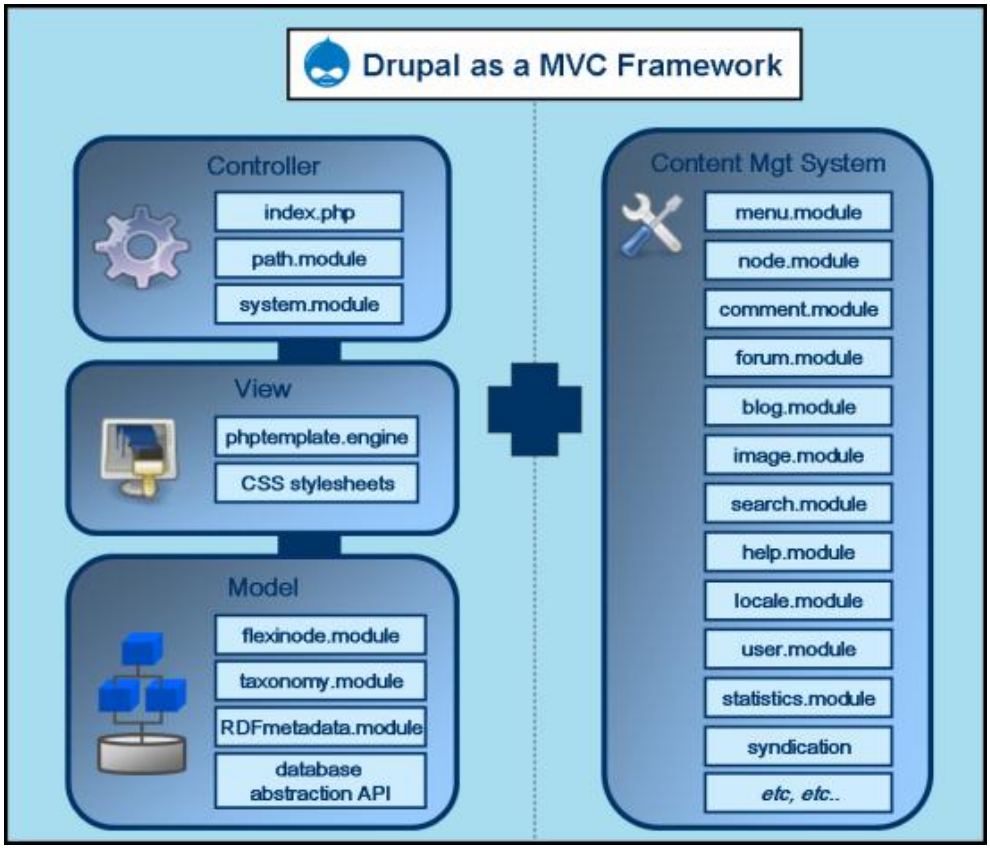
Mezi klíčové vlastnosti Drupalu patří:

- Snadné nasazení aplikace a podpora většiny hostingů
- Modulárnost a snadná tvorba vlastních modulů
- Aktivní komunita a tisíce podporovaných modulů
- Možnost cokoli v systému změnit
- Snadná tvorba vlastních datových typů bez zásahu do databáze
- Databázová abstraktní vrstva

Mezi nevýhody systému patří zejména:

- Jádro aplikace je minimální a obsahuje jen několik modulů, vhodné moduly pro konkrétní aplikaci je nutné najít
- Drupal není napsán pomocí OOP
- Komplikovanost systému

V následujícím textu práce se bude často vyskytovat architektonický vzor Model-view-controller (MVC). Podle [43] je Drupal podle MVC navržen. Rozložení systému na jednotlivé části je vidět na následujícím obrázku.



Obrázek 2 - Drupal jako MVC [43]

2.2. Platforma Adobe Flash a Flex SDK

Platforma Adobe Flash je široké veřejnosti většinou známá jako technologie, nicméně základní principy, formáty, nástroje a jazyky, které využívá, již tolik známé nejsou. V roce 2010 se Flash dostal do širšího povědomí díky diskusi, v níž byl porovnáván s technologií HTML 5. Cílem této kapitoly není tyto technologie porovnávat, ale dát čtenáři základní náhled na platformu Adobe Flash, jejíž součástí je program Flash Player, ale také jazyk MXML nebo open source SDK Flex pro vývoj Rich Internet Applications - RIA.

Adobe Flash je multimediální platforma používaná k vytváření animací, přehrávačů videa a interaktivních webových aplikací. Flash je často využíván k tvorbě her a reklamních aplikací. Méně známé je jeho využití pro tvorbu aplikací v podnikové oblasti nebo v oblasti komplexních intranetových firemních aplikací.

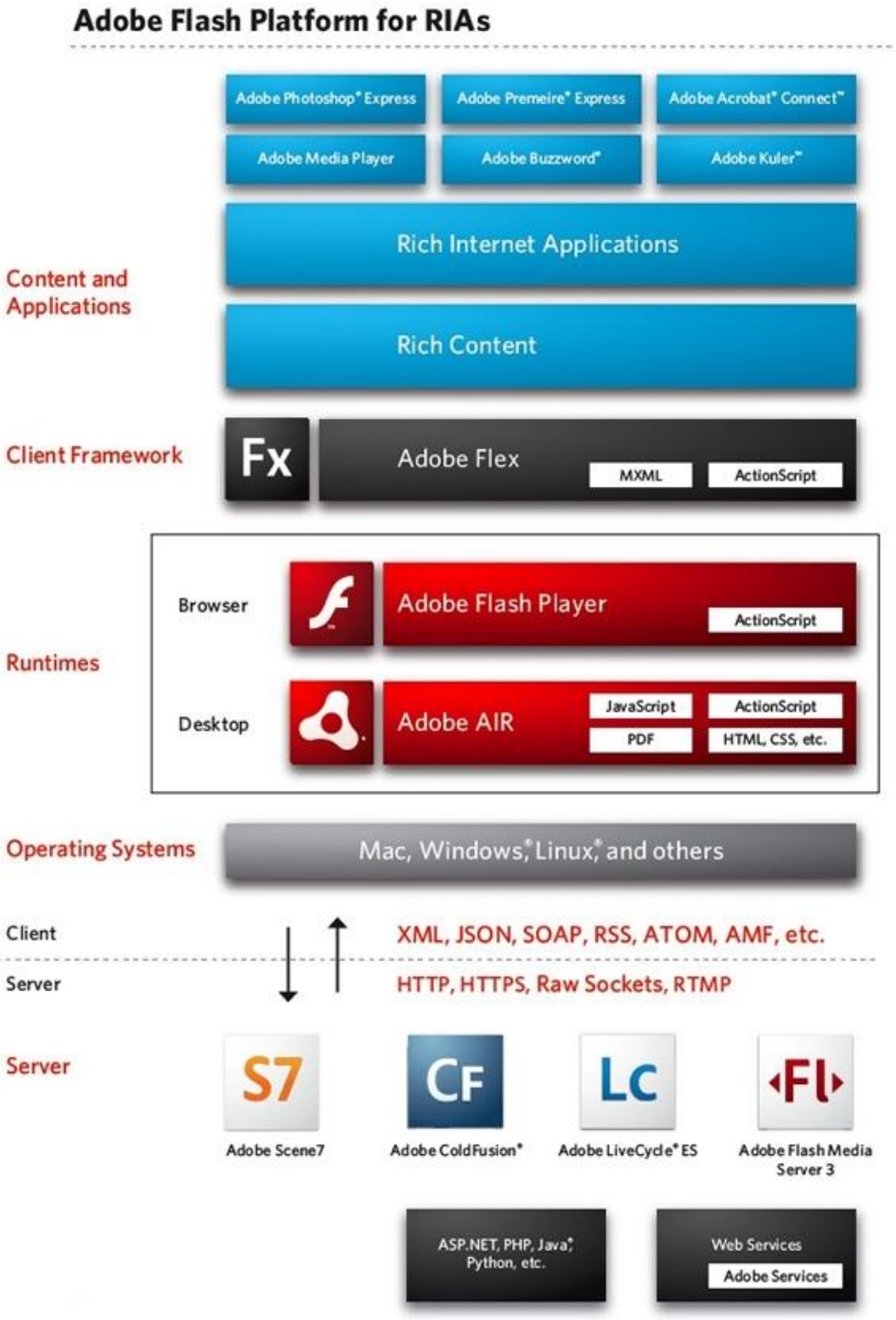
Součástí platformy jsou dvě běhová prostředí, která jsou dostupná napříč platformami. První je plugin pro prohlížeče Flash player, jehož rozšíření na PC se podle [6] pohybuje kolem 99%. Druhé běhové prostředí, Adobe AIR, je vhodné pro instalaci desktopových aplikací. Pomocí tohoto nástroje je možné snadno převést webové aplikace na desktop a zároveň pomocí stejných jazyků a vývojových prostředí vytvářet plnohodnotné desktopové aplikace.

Kromě běhového prostředí jsou důležité i frameworky a knihovny, které usnadňují tvorbu aplikací. Mezi hlavní oficiální open source frameworky patří Flex SDK. Aktuální je verze 4.5. Součástí frameworku je široká škála GUI komponent, konektorů na webové služby, lokalizace, práce s médii atd.

2.2.1. Historie

Historie Flashe začala podle [30] programem SmartSketch, který napsal Jonathan Gay. Aplikace byla v roce 1995 obohacena o frame-by-frame animaci a přejmenována na FutureSplash Animator. V následujícím roce 1996 aplikaci kupuje

firma Macromedia a slučuje ji se svým projektem Shockwave. Vzniká tak Shockwave Flash . Slovo Flash je kombinací slov Future a Splash.



Obrázek 3 - Pohled na platformu Adobe Flash podle [3]

Macromedia rozšířila interaktivní schopnosti Flashe. Přidává komunikaci s JavaScriptem, formulářová pole a knihovny objektů. V páté verzi je skriptovací jazyk přepracován podle specifikace ECMA a je představena první verze ActionScriptu. Sedmá verze přináší objektově orientovanou verzi ActionScriptu 2.0 a komponenty pro použití zvuku MP3 a video kontejneru FLV. V roce 2004 vzniká aplikační framework Flex, který jako první implementuje ActionScript 3.0, jenž je následně použit pro zbytek platformy.

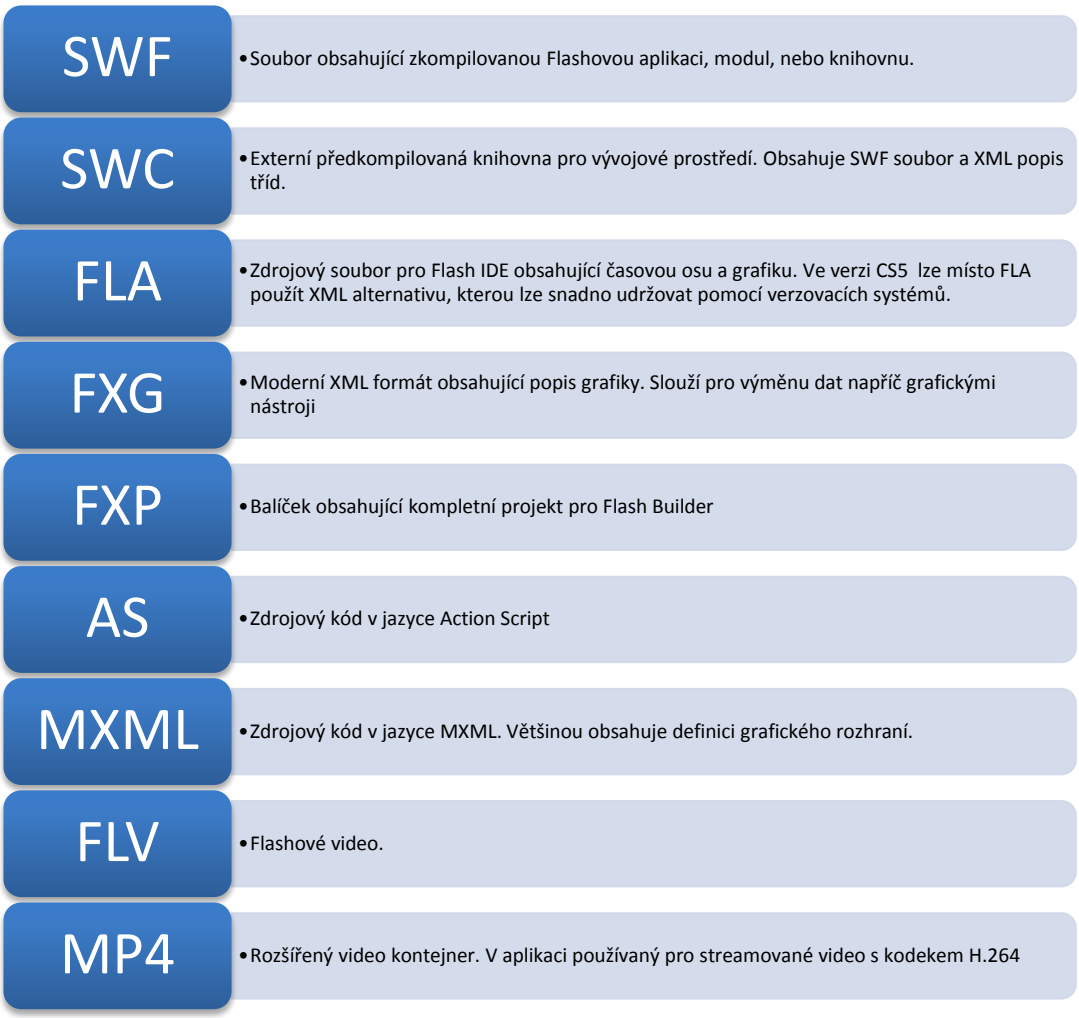
V roce 2005 kupuje Adobe Systems Macromedii. Postupně dochází ke slučování produktů obou společností a nadbytečné produkty zanikají. Navzdory obavám komunity [30] Adobe dobře pochopilo potenciál platformy. Ta se v současnosti pozvolna stává důležitým multimediálním pojítkem mezi jednotlivými produkty portfolia firmy Adobe, nyní navíc zastřešenými nástrojem Adobe Flash Catalyst a jednotným grafickým formátem FXG.

2.2.2. Adobe labs

Adobe labs [7] zaštiťuje mimo jiné vývoj nových technologií pro platformu Adobe Flash. Mezi méně známé skutečnosti patří to, že platforma Flash obsahuje většinu technologií a nástrojů, které spadají do kategorie open source. Například se jedná o framework Flex SDK, multimediální framework OSMF, ale i kompilátor mxmlc a jiné.

2.2.3. Formáty souborů

Součástí platformy Adobe Flash je několik základních datových formátů, které jsou popsány v následující tabulce. Mezi klíčové patří zejména typ .swf, který obsahuje zkompilovanou aplikaci. Definice tohoto formátu je veřejně dostupná, stejně jako většina klíčových formátů a protokolů. Hlavní zlom v této oblasti znamenalo založení uskupení Open Screen Project v roce 2008. Firmy jako Intel, AMD, Google, Nokia, Samsung a další se v něm snaží o vytvoření jednotného prostředí napříč platformami, na kterém poběží Flash Player.



Obrázek 4 - Formáty související s Platformou Adobe Flash

2.2.4. Jazyk AS3

ActionScript 3.0 (AS3) je objektově orientovaný jazyk, který znamenal velký skok v možnostech Flash Playeru [5]. Motivací pro vznik nové verze jazyka bylo vytvoření vhodného nástroje pro tvorbu RIA aplikací, které se staly nedílnou součástí webového prostředí.

Základem jazyka je ECMAScript, který představuje mezinárodní standardizovaný programovací jazyk pro skriptování. ActionScript 3.0 je kompatibilní se specifikací třetí verze ECMAScriptu (ECMA-262). Obsahuje ale také řadu vlastností čtvrté generace ECMAScriptu.

ActionScript je interpretován v ActionScript Virtual Machine (AVM), která je součástí Flash Playeru. Aktuální verze Flash Playeru 10.2 obsahuje AVM1 pro historické verze postavené na ActionSriptu 1 a 2. Dále je přítomné AVM2 pro ActionScript 3.0. Jazyk AS3 je vhodný pro programování logiky aplikace, ale pro definici uživatelského rozhraní, formulářů a deklarativní programování je nevhodný. Z tohoto důvodu byl vytvořen nový jazyk postavený na XML, který řeší všechny jmenované problémy.

2.2.5. Jazyk MXML

Soubory ve formátu MXML obsahují vzhled uživatelského rozhraní, tj. typy komponent, jejich velikost a pozici na scéně, vlastnosti a funkce, podle kterých se zkompiluje výsledný SWF soubor. Pro tento obsah Adobe zvolilo formát XML, konkrétně MXML skinovatelný pomocí CSS. Potřebný kód v ActionScriptu 3 může být součástí MXML souboru, nebo může být umístěný do samostatných souborů.

Jazyk MXML usnadňuje zejména tvorbu uživatelského rozhraní. Pomocí poslední verze GUI komponent Spark, které jsou součástí SDK 4.x, je možné od sebe striktně oddělit vzhled komponent a jejich logiku. V XML je možné definovat libovolnou grafiku podobně jako v SVG. Kromě grafiky deklarativní programování usnadňuje i definování logiky, například pomocí techniky databindingu. Díky ní je možné jednoduše spojovat objekty mezi sebou nebo objekty s vlastnostmi komponent. Po změně jedné strany se změní i strana druhá, a to bez nutnosti ručně sledovat a ošetřovat událost změny.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
    width="300" height="200">
    <mx:Script>
        <![CDATA[
            private function clickHandler():void {
                myLabel.text = "Hello, World!";
            }
        ]]>
    </mx:Script>
```

```
    <mx:Panel
      title="My Application">
       <mx:Button id="myButton" label="Click Me!"
click="clickHandler();"/>
    </mx:Panel>
</mx:Application>
```

Ukázka Hello World v jazyce MXML

2.2.6. Přehled nástrojů

Pro implementaci aplikace byla použita celá řada nástrojů z platformy Adobe Flash. Základní přehled je na seznamu níže.

Adobe Flash Builder

Hlavní IDE projektu, ve kterém je napsána klientská aplikace. Slouží ke kompilaci aplikace a editaci zdrojových kódů ve formátu .as a .mxml. Pro vývoj byla použita verze Flash Builder 4. Aplikace je následníkem aplikace Flex Builder 3.

Adobe Flash Catalyst CS5

Nástroj určený k přípravě grafiky ve formátu .fxg a skinování komponent. Nástroj využívá novou sadu GUI komponent Spark, které jsou dostupné v SDK Flex 4.x. Jako grafický vstup je možné použít grafiku vytvořenou mimo jiné v grafickém programu Adobe Illustrator CS5.

Adobe Flash CS5

Animační program, ve kterém jsou vytvořeny animace a efekty. Program jako zdrojový formát využívá soubory typu . fla.

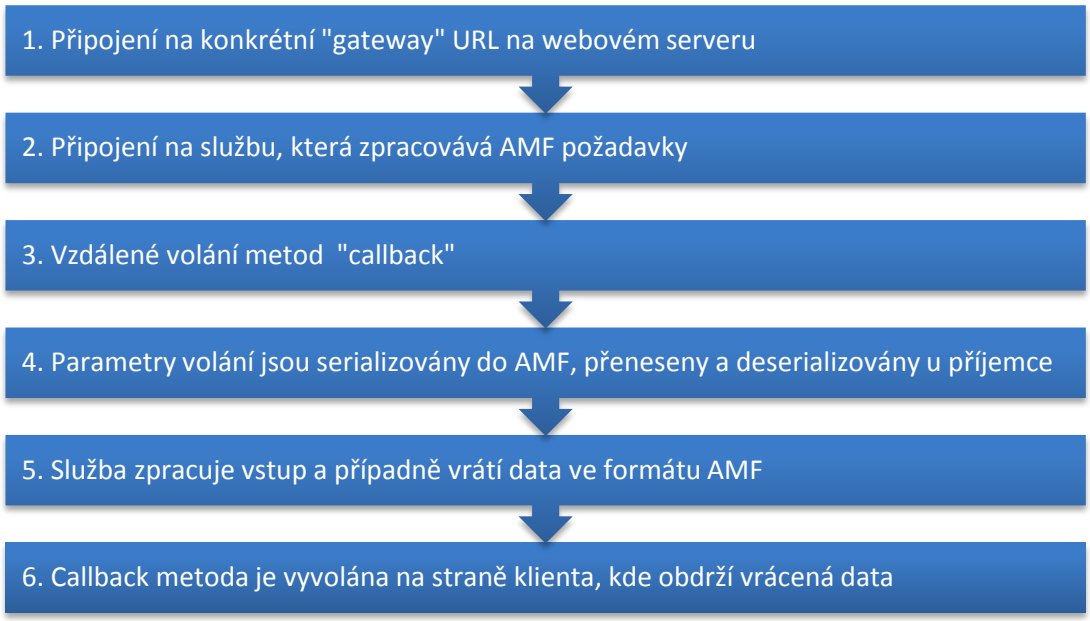
Eclipse for PHP Developers

Pro vytvoření rozšiřujících modulů pro CMS Drupal bylo použito vývojové prostředí Eclipse ve verzi 3.6 (Helios). Flash Builder je postavený na Eclipse a je proto možné si Flash Builder nainstalovat do standardní instalace Eclipse jako plugin.

2.3. Protokol AMF

AMF (Action Message Format) je binární protokol používaný pro serializaci, kompresi a výměnu ActionScript objektů. Je používán primárně pro výměnu dat mezi aplikacemi platformy Adobe Flash a vzdálenými službami (remote services). AMF0 byl představen spolu s Flash Playerem 6. Nejnovější verze AMF3 přináší podporu složitějších typů a byla představena u Flash Playeru 9 v roce 2006.

Kromě formátu AMF by bylo možné použít například různé podoby webových služeb, nebo REST protokol. Mezi hlavní důvody, proč bylo vybráno právě AMF, patří jeho rychlost při přenosu dat a snadná integrace pomocí knihovny AMFPHP. Oproti REST protokolu má AMF výhodu automatického mapování přenášených objektů na AS3 VO (value objects).



Obrázek 5 - Průběh komunikace pomocí AMF protokolu

2.4. Protokol XMPP

Extensible Messaging and Presence Protocol (XMPP), podle [11] dříve známý jako Jabber, původně vznikl jako protokol pro IM síť Jabber. Následně se ukázalo, že kromě IM může být s výhodou použit i pro vzájemnou komunikaci programů. Protokol se následně stal součástí RFC dokumentů - základní normy jsou RFC 3920 (obecná specifikace

protokolu) a RFC 3921 (samotný instant messaging a zobrazení stavu). O vývoj protokolu se stará XMPP Standards Foundation. XMPP je implementací formátu XML. Specifikace jsou zcela otevřené všem, kdo mají zájem o implementaci software s podporou XMPP. Serverové aplikace s XMPP protokolem běží standardně na TCP portu 5222. Pro vzájemnou komunikaci serverů je pak vyhrazen port 5269.

Protokol XMPP zapouzdřuje zprávy do XML. Síť využívající XMPP není centralizovaná, jak je běžné u většiny ostatních IM, ale je distribuovaná na lokální servery, na kterých je možno si založit uživatelský účet. Identifikátory uživatelů JID jsou v základním tvaru syntakticky i sémanticky podobné e-mailovým adresám. Například ve tvaru user@server.

Uživatel se připojuje vždy pouze ke svému serveru, protože jenom tento server je schopen ověřit jeho identitu, například pomocí uživatelského jména a hesla. Kromě serverů a klientů se v síti vyskytují další služby, jako například služby víceuživatelských diskuzí, uživatelské adresáře a transporty. Transporty jsou brány mezi XMPP sítí a IM sítí pracující na jiném protokolu.

2.4.1. Openfire a XIFF

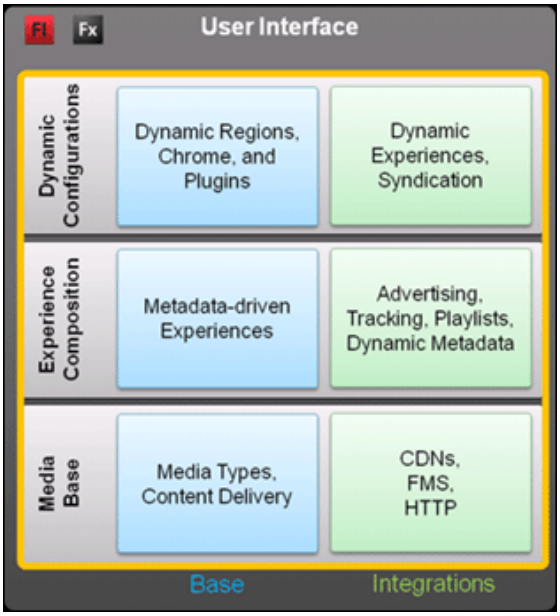
Kromě komunikačního protokolu bylo též nutné vybrat vhodnou serverovou aplikaci a SDK. Výběru a zvažovaným možnostem se věnuje samostatná kapitola. Jako nejvhodnější kandidát byl vybrán server Openfire a SDK XIFF [13] pro AS3. Oba produkty poskytuje firma Jive Software pod licencí Apache licence [12]. Mezi klíčové vlastnosti serveru patří:

- Snadná administrace pomocí webových stránek
- Snadná rozšiřitelnost pomocí pluginů
- Podpora šifrování
- Snadná instalace, platformová nezávislost díky implementaci v Javě
- Široká škála databázových konektorů a podpora LDAP

2.5. Projekt OSMF

Open Source Media Framework (OSMF) je open source framework určený pro usnadnění tvorby Flashových aplikací pracujících s videem. Framework využívá ActionScript verze 3.0. Framework se primárně zaměřuje na tvorbu univerzálního přehrávače videa, který je možné snadno skinovat a rozšiřovat pomocí pluginů třetích stran. Pluginy mohou přidávat reklamu, zlepšovat vykreslování, upravovat streamování nebo řešit práci s chráněným obsahem. OSMF podporuje streaming videa pomocí Real Time Messaging Protocol (RTMP) a také pomocí protokolu http, pokud protokol RTMP není dostupný. Kromě streamingu zvládne přehrávač použít i metodu progressive download.

Ve verzi 1.6 vydané v únoru 2011 je integrována nová funkce Stage video, obsažená ve Flash Playeru 10.2, která přináší rapidní snížení požadavků na HW počítače díky široké podpoře hardwarové akcelerace přehrávání videa na grafické kartě. Video není podle [10] součástí DOM Flashové aplikace, ale je vykreslováno pod ní pomocí DirectX, nebo OpenGL. Video v HD kvalitě je tak možné použít i v aplikacích, které běží na méně výkonných PC.



Obrázek 6 - Architektura OSMF podle [15]

3. Podobné projekty

Po technické stránce identická aplikace není prozatím známa. Aplikace v sobě integruje prvky různých sociálních sítí, video databází a jiných webových aplikací, které spojuje termín "Web 2.0". Důležitým znakem těchto aplikací je to, že v nich byl pevný obsah webových stránek nahrazen prostorem pro sdílení a společnou tvorbu obsahu [57]. Nejen pro samotnou implementaci, ale také pro návrh architektury aplikace Fashionspace je důležité se podívat na technické řešení webových aplikací, jako je například Youtube, Facebook a Google talk.

3.1. Youtube

Pro portál Fashionspace je z aplikace Youtube nejdůležitější systém zpracování videa a jeho distribuce uživatelům. Podle [16] do roku 2010 Youtube využíval pro přehrávání videa čistě Flashový video přehrávač. Podle [17] bylo zastoupení Flash videa na internetu na konci roku 2009 zhruba 75%. Od roku 2010 existuje experimentální varianta video přehrávače postaveného na HTML5. Podporované jsou prohlížeče, které dokáží přehrát video ve formátu H.264 a WebM. Všechna videa nejsou prozatím konvertována. Oproti projektu Fashionspace je video na stránkách Youtube zobrazeno pouze v samostatném přehrávači a není do aplikace natolik integrováno jako v projektu Fashionspace.

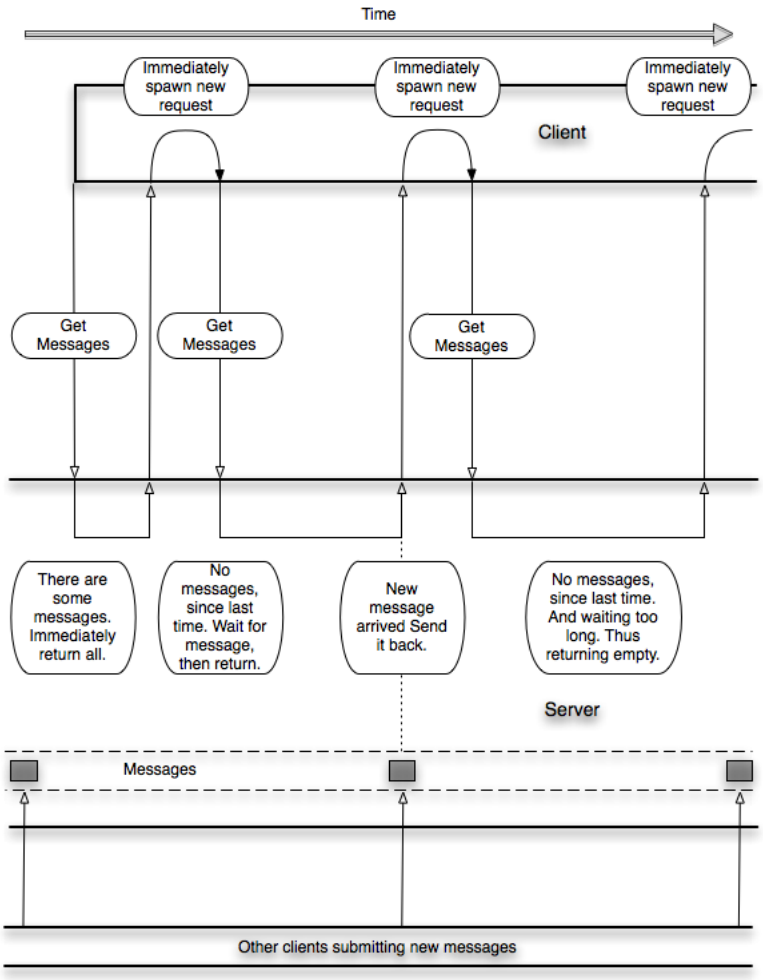
Mezi komplikované části zpracování videa patří jeho upload na server. V roce 2006 Youtube zavedl limit 10MB pro velikost uploadovaného videa. Tento limit byl následně navyšován. Aktuálně je možné přímo na webové stránce nahrát video o velikosti 2 GB. Pomocí rozšířeného Java uploaderu je možné nahrát video o velikosti 20 GB. Mezi podporované kontejnery pro video patří: AVI, MKV, MOV, MP4, DivX, FLV, OGG a OGV. Tyto kontejnery zahrnují video formáty jako MPEG-4, MPEG, VOB, a WMV. Podporované je též video ve formátu 3GP pro upload videa z mobilních telefonů.

Podle [18] je možné pozorovat rychlý nárůst použití videa kompatibilního s HTML5, zejména v podobě H.264, který na konci roku 2010 tvořil zhruba 50% veškerého video obsahu na internetu.

3.2. Facebook

Podle [19] je Facebook rozsáhlý společenský webový systém sloužící primárně k tvorbě sociálních sítí, komunikaci mezi uživateli, sdílení multimediálních dat, udržování vztahů a k zábavě. Podle [20] má Facebook 650 milionů aktivních uživatelů a je jednou z největších společenských sítí na světě. Systém je plně přeložen do 68 jazyků.

Z technického pohledu je pro projekt Fashionspace nejzajímavější návrh a implementace chatu, který je součástí Facebooku. Chat byl integrován na začátku roku 2008 a je postavený na modelu Comet [21]. Svými vlastnostmi je tento chat blízký IM desktopovým aplikacím.



Obrázek 7 - Model komunikace Comet podle [23]

Comet je model webové aplikace, který využívá dlouhé http požadavky (long-held HTTP request) pro posílání dat prohlížeči bez specifického vyžádání prohlížečem. Na straně serveru je stále otevřený HTTP request, na který může server kdykoli odpovědět. Comet je zastřešující termín pro řadu technologií, které jsou schopné docílit této interakce. Comet a jeho princip je znám pod řadou dalších jmen, jako je Ajax Push [24], Reverse Ajax [25], Two-way-web [26], HTTP Streaming[26] a HTTP server push [27].

Podle blogu programátorů Facebooku [22] je serverová část aplikace napsaná pomocí jazyka Erlang a C++. Jazyk Erlang patří do kategorie funkcionálních jazyků. Byl vyvinutý v roce 1986 firmou Ericsson a v současnosti je dostupný jako open source.

3.3. Google talk

Mezi další známé IM webové aplikace patří Google talk uvedený v roce 2005 [28]. Služba využívá protokolu XMPP. Dále přidává možnost hlasové komunikace, na jejíž specifikaci spolupracuje s XMPP Standards Foundation. V další fázi vývoje přibyla i možnost video hovorů.

Využití služby, která je zcela zdarma, je podmíněno registrací účtu na Google nebo na poštovním serveru Gmail, s nímž je úzce propojen (sdílení kontaktů, oznamování nové pošty, integrovaný klient ve webovém rozhraní). Službu Google Talk lze používat i na jiných klientech pro instant messaging, jakými jsou například Miranda, Pidgin a Trillian.

4. Požadavky

Cílem této kapitoly není popsat kompletní seznam požadavků na vyvíjenou aplikaci a nahradit tak formální specifikaci požadavků Software Requirements Specification (SRS). Cílem je popsat jen klíčové požadavky, které ovlivní hlavní návrh aplikace, její architekturu a výběr technologií.

4.1. Funkční požadavky

FR1. Aplikace by měla být hravá a uživatel by měl mít možnost snadno upravovat prostředí, například pomocí přesouvání GUI komponent. Dále by se tento stav měl ukládat bez nutnosti přihlášení. Každá stránka by měla obsahovat možnost (tlačítko) pro navrácení stránky do původního stavu.

FR2. Návrháři mohou do aplikace snadno vložit video s přehlídkou a okomentovat modely, které jsou na ní prezentovány.

FR3. Video na přehlídkách by mělo byt velice kvalitní, aby ho bylo možné bez problému sledovat ve fullscreen módu.

FR4. V základní podobě musí systém podporovat dvě lokalizované verze (cz/en). Předpokládá se rozšíření na další jazykové mutace.

FR5. Systém by měl obsahovat jednoduché administrační prostředí, kde bude správce vkládat reklamy do databáze. Systém musí podporovat různé velikosti a formáty.

FR6. Systém by měl obsahovat katalog přehlídek, a to jak historických, tak aktuálních a připravovaných. Katalog bude obsahovat stránku s hodnocením a výběrem komentářů pro staré přehlídky.

4.2. Uživatelé

UR1. Stránky by měly být pro návštěvníky přístupné bez přihlášení. Tento typ uživatelů získá přístup k starším přehlídkám (například dva týdny)

UR2. Uživatelé mohou získat vstupenku, pomocí které vstoupí na aktuální přehlídku.

UR3. Návrháři se budou muset přihlašovat, aby bylo možné uchovávat stav jejich rozdělané práce.

4.3. Technicke požadavky

TP1. Aplikace by měla být co nejvíce multiplatformní.

TP2. Aplikace by měla být optimalizovaná na rozlišení 1024x768

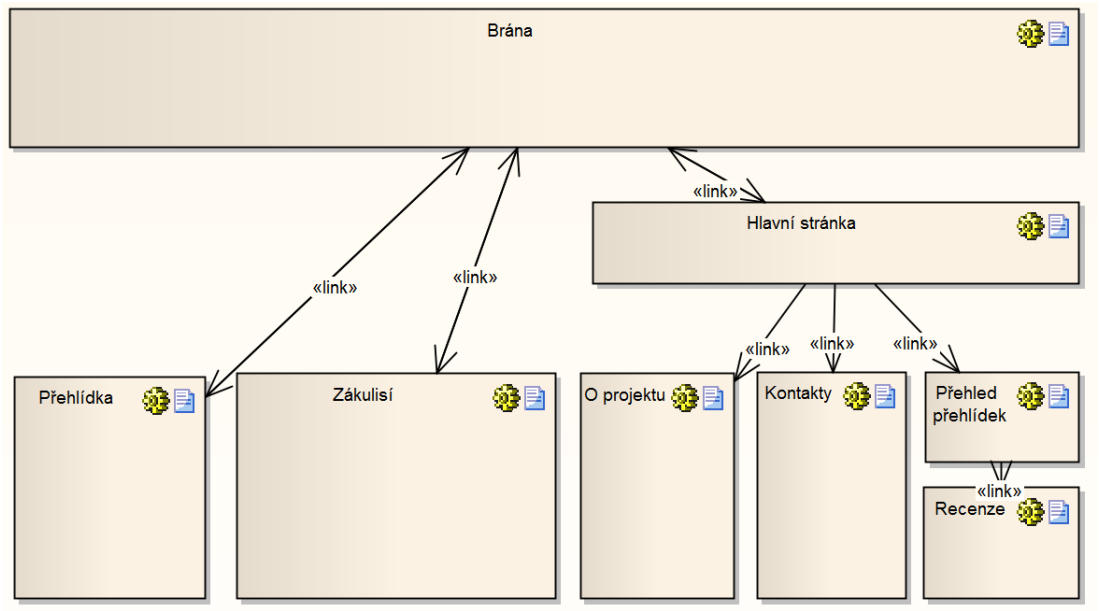
TP3. Aplikaci by mělo být možné ideálně provozovat i na slabších PC, jakými jsou například netbooky.

TP4. Aplikaci by mělo být možné snadno skinovat a upravovat pro jiné použití, než jsou módní přehlídky.

TP5. Architektura aplikace by měla být komponentová a snadno rozšiřitelná.

4.4. Základní struktura stránek

Klientská část aplikace by měla obsahovat hlavní rozcestník se třemi částmi. První je tvořena stránkou s přehlídkou. Druhou část tvoří editace přehlídky - zákulisí. Poslední část obsahuje stránky informující o projektu a obsahující historii starých přehlídek s recenzemi.



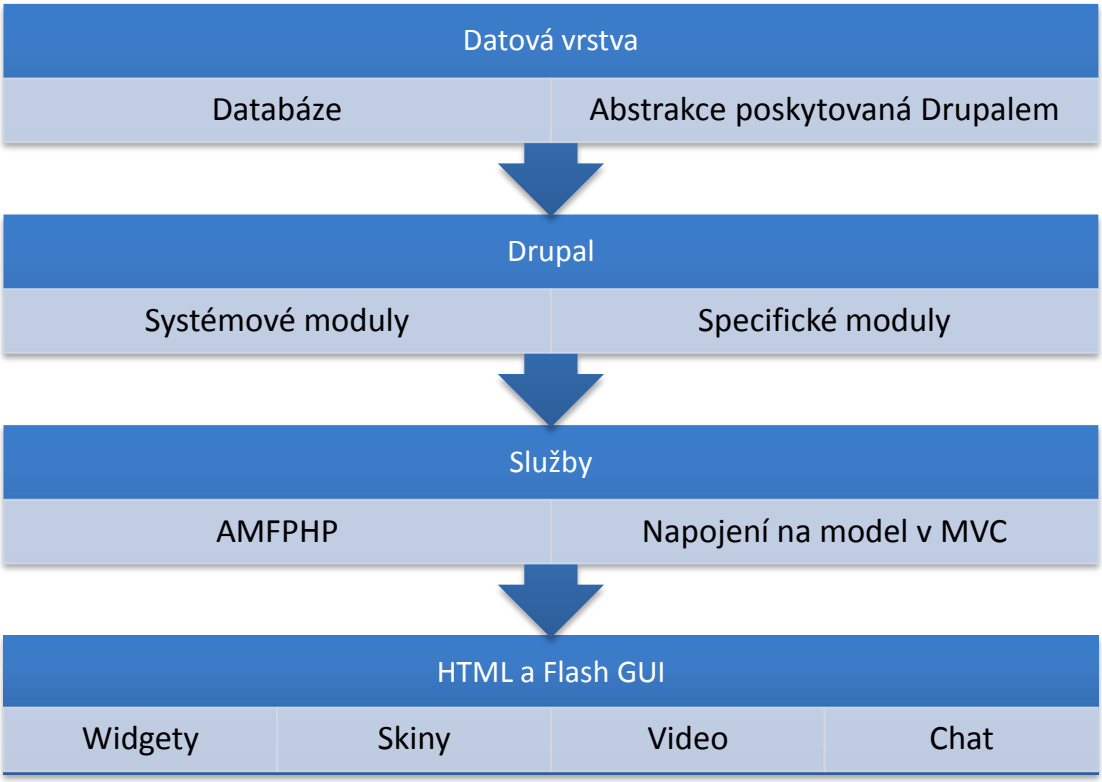
Obrázek 8 - Základní struktura stránek klientské aplikace

5. Architektura

Portál je tvořen webovou aplikací, která využívá širokou škálu technologií a knihoven. Aby byla aplikace dobře udržovatelná a rozšiřovatelná, obsahuje její návrh celou řadu ověřených vzorů na jednotlivých úrovních návrhu. Jedná se například o model Klient/Server, Model-view-controller (MVC), dependancy injection, ale i o klasické návrhové vzory jako je Observer, Strategie a další.

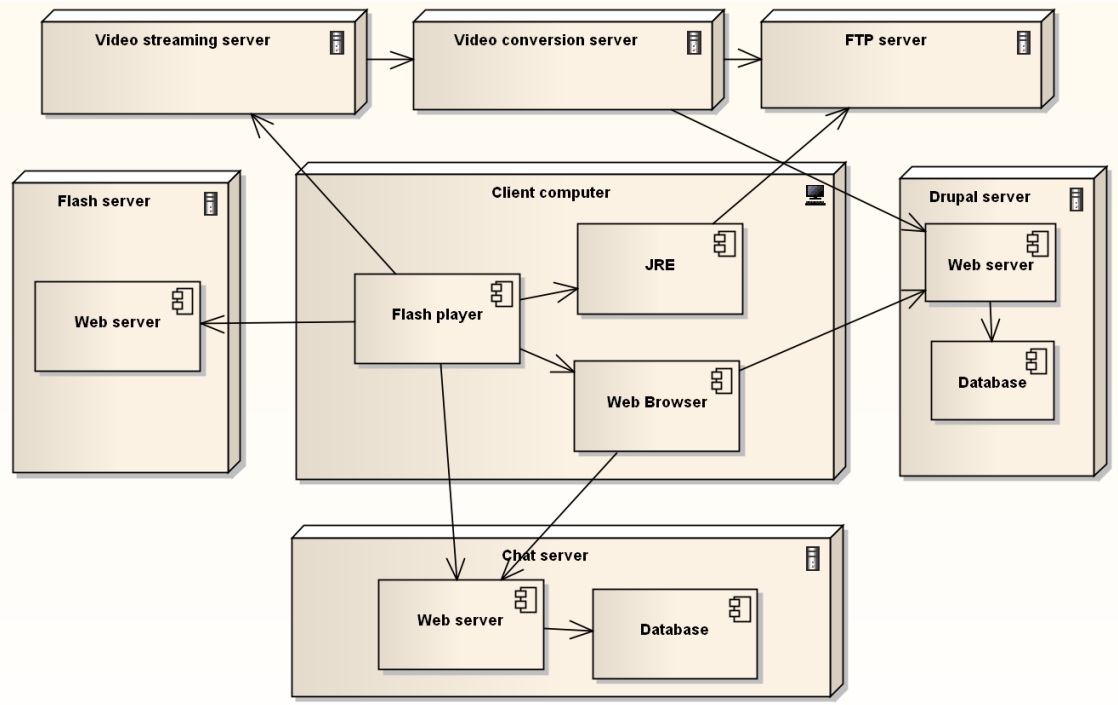
5.1. Struktura aplikace

Aplikace je postavena na modelu Klient/Server. Serverová část je tvořena CMS Drupal, který poskytuje služby klientské části, kterou tvoří Flashová aplikace. Server zajišťuje zejména persistenci dat. Klientská aplikace běží v prohlížeči uživatele a tvoří grafické rozhraní aplikace. Middleware je tvořen službami, které CMS Drupal poskytuje pomocí frameworku AMFPHP. Komunikace probíhá pomocí protokolu AMF.



Obrázek 9 - Základní architektura systému

Kromě CMS Drupal a Flashové aplikace je součástí systému ještě Java Applet, sloužící k uploadu videa na FTP server. O online komunikaci se stará Openfire server. Pro video je v systému vyhrazen FTP server, transcoding server a streaming server. Do budoucna je možné systém rozšířit o LDAP server a integraci s S3 službami od firmy Amazon, které by sloužily k zálohování. Vazby mezi jednotlivými komponentami jsou označené na obrázku 10.



Obrázek 10 - Nasazení aplikace Fashionspace

5.1.1. Administrace

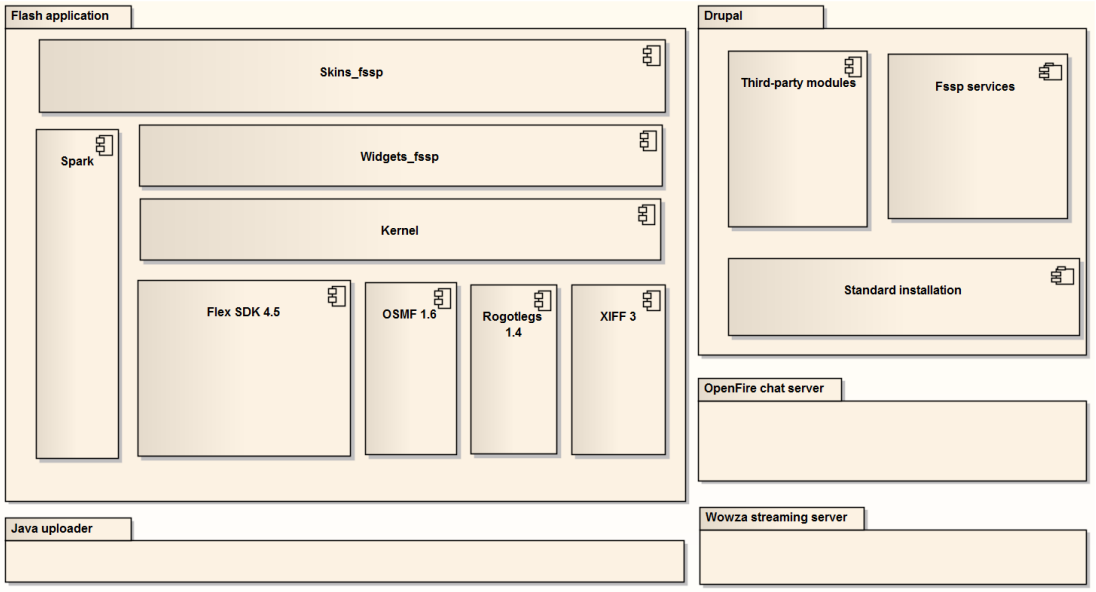
V textu práce je řešena zejména část aplikace pro návštěvníky portálu. Kromě této části je samozřejmě součástí portálu také administrace. Administrace portálu probíhá čistě přes webové rozhraní, které poskytuje CMS Drupal. Vytvoření tohoto rozhraní pomocí složení vhodných modulů a vytvoření požadovaných formulářů není technicky náročné, nicméně velmi pracné. Z tohoto důvodu není součástí textu této práce, která se zaměřuje na architekturu a vytvoření kostry aplikace

5.1.2. Základní komponenty systému

Kromě popisu nasazení aplikace je klíčové i rozdělení na základní komponenty a moduly. Rozdělení je znázorněno na obrázku 11, podrobný popis komponent je dostupný v následujících kapitolách.

5.2. Klientská část aplikace

Klientskou část řešení tvoří aplikace napsaná v jazyce AS3 a MXML. Základ aplikace je postavený na frameworku Flex SDK 4.5 a sadě GUI komponent Spark. Kromě tohoto základu aplikace využívá Framework OSMF 1.6 pro práci s videem. Dále je použit Framework Robotlegs 1.4 pro implementaci MVC a XIFF 3 pro podporu XMPP protokolu. Nad tímto základem stojí jádro aplikace, widgety a jejich skiny. Pro logování chyb a chybovou konzoli je použita knihovna ThunderdBoltAS3.



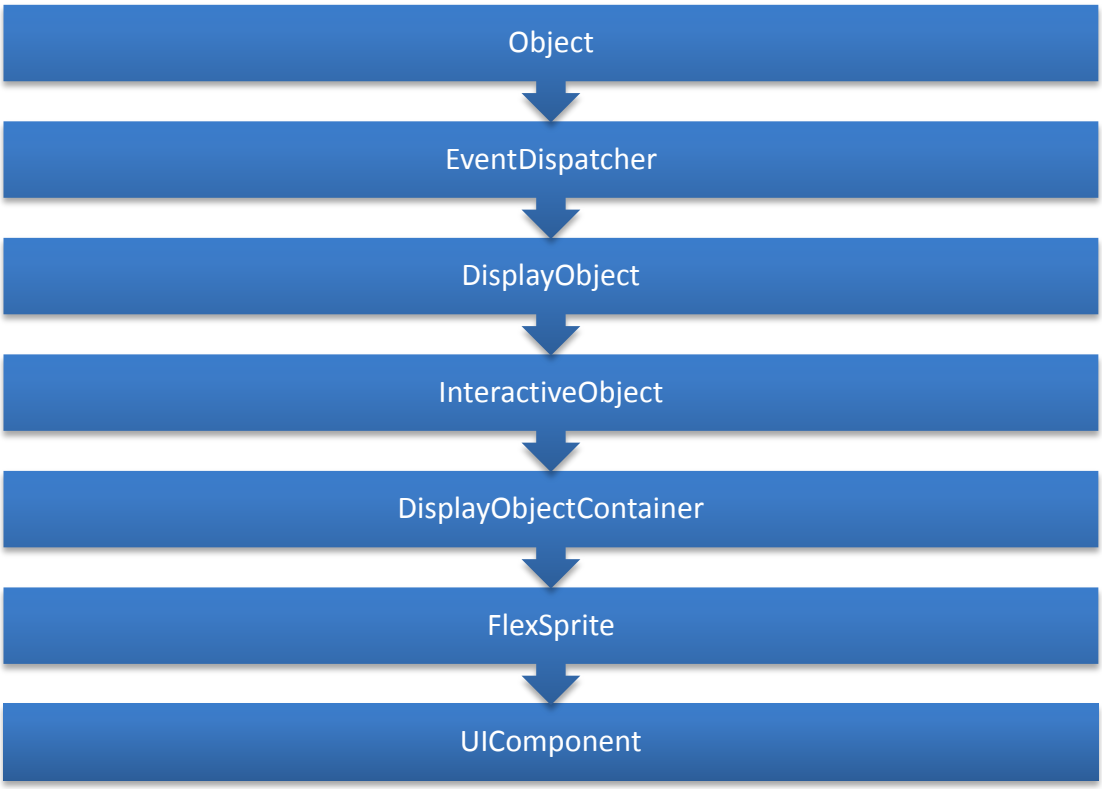
Obrázek 11 - Základní komponenty aplikace Fashionspace

5.2.1. FlashPage a Widgety

Základem systému je kontejner FlashPage, který představuje jednu stránku aplikace. Každá stránka může obsahovat celou řadu komponent (widgetů). Každá stránka v systému je popsána pomocí XML, ve kterém je definována její struktura. Toto XML může být uloženo jako soubor, může být uloženo jako node v CMS Drupal nebo zjednodušeně pro testovací účely a urychlení načítání přímo v kódu aplikace. Přesný popis struktury XML a jeho lokalizace bude popsán v následujících kapitolách.

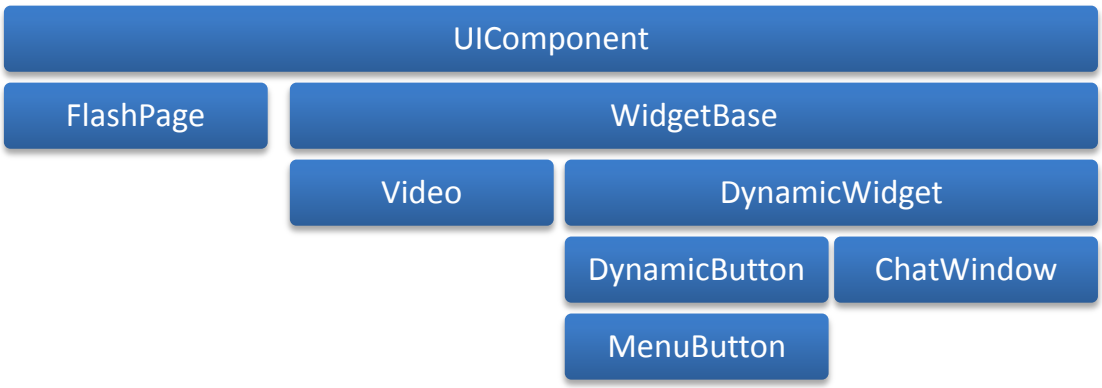
Třída *FlashPage* a všechny widgety jsou odvozeny od standardní flexové třídy *UIComponent*. Základní struktura tříd GUI pro Flex SDK 4.5 je na obrázku dvanáct. U widgetu je klíčové, zda se jedná o dynamickou variantu, kterou muže uživatel

libovolně upravovat, zejména posouvat, nebo zda se jedná o statické elementy, které jsou pevně ukotvené.



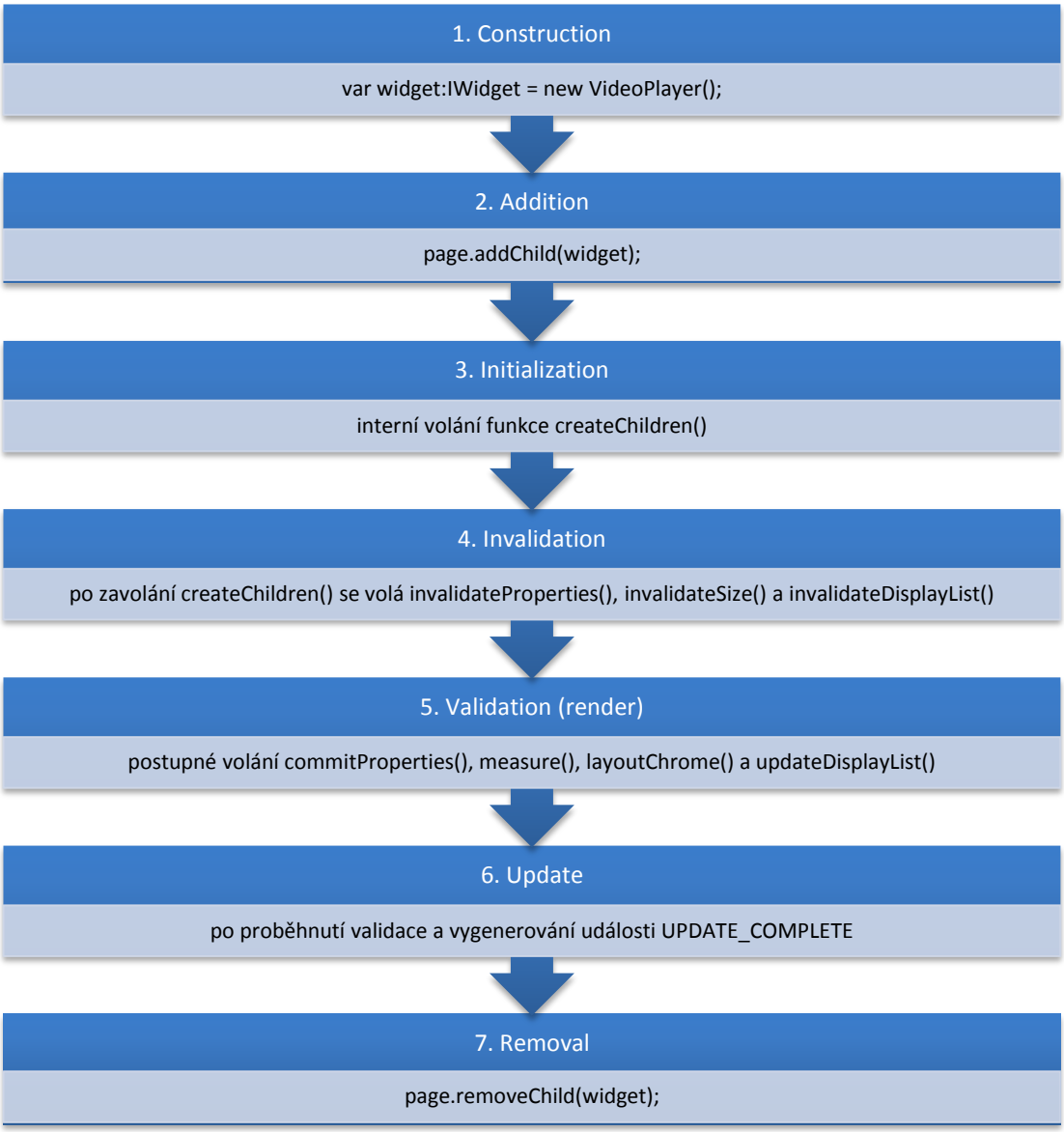
Obrázek 12 - Základní hierarchie tříd pro UIComponent ve Flex SDK 4.x

U dynamické varianty dochází k ukládání případných změn komponent do lokálního uložiště pomocí Local Shared Objects. Informace o změnách nejsou zásadní a z tohoto důvodu se ukládají jen lokálně. Důležitá data se ukládají na server.



Obrázek 13 - Základní návrh GUI komponent

Widgety a FlashPage jsou postaveny na třídě *UIComponent*. Jejich životní cyklus je na obrázku 14. Nejdříve dojde ke konstrukci komponenty a jejímu přidání na plochu. V dalším kroku jsou vytvořeny dětské subkomponenty. Po ukončení konstrukce je komponenta inicializovaná. Dále proběhne dvojice funkcí *invalidation* a *validation*, které se volají po každé změně. Komponenta je následně aktuální a existuje až do destrukce.



Obrázek 14 - Životní cyklus flexové komponenty UIComponent

Na obrázku třináct je ukázka základního členění widgetů. Třída *WidgetBase* slouží jako základní třída widgetů, která obsahuje podporu pro integraci do systému, komunikaci mezi widgety a ukládání lokálních dat. Přímým potomkem této třídy je

například widget pro přehrávání videa. Třída *DynamicWidget* přídává podporu pro přesouvání komponenty a náhodný pohyb komponenty. Potomkem této třídy je například komponenta zobrazující chatovací okno ChatWindow.

V mnoha situacích spolu potřebují widgety na stránce komunikovat. Aby byla zaručena co nejmenší závislost (loose coupling) widgetů a modulů, jsou pro komunikaci použity události a návrhový vzor Observer. Pokud potřebuje widget1 zavolat funkci fn1 na widgetu2, vygeneruje potřebnou událost. Pokud widget2 odchytává odeslanou událost, může na základě informací obsažených ve zprávě vyvolat zavolání funkce fn1. Události putují po centrální sběrnici (bus), kterou definuje MVC model popsaný dále.

Obrázek 15 - Proces komunikace mezí widgety na jedné stránce

Vazby mezi widgety pomocí akcí a reakcí je možné definovat pomocí XML popisujícího jednotlivé stránky

5.2.2. XML popis FlashPage

Každou stránku aplikace reprezentuje XML popis. Třída *FlashPageParser* tyto definice prochází a generuje události s informacemi o tom, které komponenty je třeba vytvořit, jaké reakce zaregistrovat atd. Pro práci s XML je použito rozšíření jazyka AS3 a to ECMAScript for XML (E4X). Popis stránky v podobě XML je důležitý z několika důvodů:

- Oddělení obsahu stránky od systému
- Při změně textu nebo posunu komponenty není nutné kompilovat znovu celou aplikaci
- V podobě XML je možné popis umístit do nodu v CMS Drupal, v němž se snadno online edituje. XML je možné vložit do samostatného souboru, kde následně nezatěžuje webovou aplikaci. Pro urychlení vývoje nebo načítání je možné jej začlenit přímo do kódu aplikace
- XML se snadno lokalizuje. Libovolný tag může mít svoji lokalizovanou alternativu
- XML dokument je možné snadno skládat z fragmentů
- XML se snadno edituje
- Jazyk je podobný jazyku MXML

Ilustrační příklad definice stránky obsahující několik widgetů:

```
<?xml version="1.0" encoding="utf-8"?>
<flashpage>
   <widgets>
      <box>
            <widget
                  id="logo" type="SplashLogo" x="100" y="100"
                  componentState="randomMoveState"
                  allowDragging="true"
                  nodeLink="home_node"/>
            …
      </box>
      <canvas>
            <widget
                  id="footer" type="Footer"
                  bottom="0" horizontalCenter="0">
                  <copyright id="12" />
            </widget>
            …
            </canvas>
      <widgets>
```

```
	<actions>
		<action
			sender="countdown" event="startShow"
			responder="video" action="play" />
		…
	</actions>
	<fragments>
		<fragment name="basic_interface" />
		…
	</fragments>
</flashpage>
```

Každá stránka je složena ze čtyř hlavních částí. První je element <*box*>, který obsahuje definici komponent vykreslených v oblasti o velikosti 955x600 pixelů. Toto je oblast, která je vycentrovaná uprostřed stránky a je vždy vidět, pokud má uživatel alespoň minimální podporované rozlišení 1024x768. Jedná se například o texty, editor přehlídek, reklamy atd., tedy o komponenty vytvořené s fixní velikostí.

Druhou část představuje sekce <*canvas*> obsahující komponenty, které se mají přichytit okraje obrazovky nebo se roztahují po celé její ploše. Jedná se například o podkladové video, patičku aplikace nebo chatovací panel. Sekce <*canvas*> a <*box*> se nacházejí v tagu <*widgets*>.

Třetí sekce <*actions*> obsahuje popis interakce widgetů na stránce. Pro definice je použit systém událostí v jazyce AS3. Každý typ události má unikátní identifikátor. V definici jde o atribut *event.* Atribut *sender* definuje ID komponenty, která může tuto událost generovat. Atribut *responder* definuje ID komponenty, která má na událost reagovat. Poslední atribut, *action,* definuje název funkce, která se má v důsledku události na komponentě *responder* vykonat.

Poslední sekce <*fragments*> slouží k zamezení opakování obsahu XML souborů pro různé stránky. Napříč stránkami je například stejná patička nebo menu. XML soubor nebo node FragmentsNode obsahuje definici všech fragmentů. Každý fragment má stejnou strukturu jako flashpage. V rámci fragmentů je možné přidat libovolný nový widget nebo reakci. Pokud dojde ke kolizi id widgetů a typ widgetů je stejný, potom fragment přepíše vlastnosti widgetu.

Každý widget je popsaný svými atributy. Kromě atributů je možné použit i dětské nody, které jsou rovnocenné atributům se stejným jménem. Pokud je použit jak atribut, tak node, potom je zahlášena chyba a přednost má node. Atributy jsou vhodné pro jednoduché textové a číselné vlastnosti widgetů, zatímco nody jsou vhodné zejména pro složitější datové typy nebo vlastnosti, které je nutné lokalizovat.

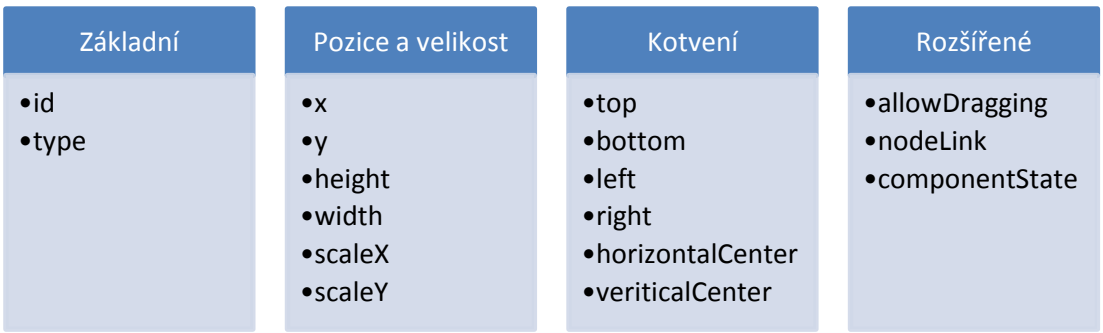
Při vytváření komponent je použita technika reflection. Jméno vytvářené instance widgetu obsahuje atribut *id*. Jméno třídy obsahuje atribut *type*. Atribut *id* by měl být unikátní v rámci jedné obrazovky. Zjednodušená ukázka použití reflection:

```
//Bez reflection
var foo:Foo = new Foo();
foo.hello();

//S  reflection
var cls:Object = getDefinitionByName("Foo");
var foo:Object = new cls();
foo.hello()
```

Při použití reflection je důležité použít správné nastavení domény. Domény jako takové jsou podrobněji popsány na konci deváté kapitoly.

Struktura XML souboru popisujícího jednotlivé stránky je úzce svázána s jazykem MXML. Pomocí XML je možné definovat většinu vlastností a stylů komponent, které komponenty reálně mají. Kategorie vlastností komponent jsou zobrazeny na následujícím obrázku.



Obrázek 16 - Atributy elementů v XML popisujícím vzhled stránky

Atributy z kategorie „pozice a velikost“ jsou přímo napojené na standartní atributy třídy *UIComponent*. Atributy z kategorie „kotvení“ jsou napojeny na styly

komponent. Souřadnice *z* u widgetů je definována jejich pořadím v XML. První komponenta je nejníže a ostatní ji překrývají. Pomocí atributu *allowDraging* je možné povolit uživatelské přesouvání komponenty. Atribut *nodeLink* obsahuje odkaz na jinou stránku, která se má zobrazit. Atribut *componentState* definuje stav komponenty. Mezi základní stavy komponenty patří statický stav, kdy se komponenta nehýbe, a stav, kdy se náhodně pohybuje po obrazovce.

V XML je možné pro formátování textů používat podmnožinu HTML, která podporuje Flash. Základní ukázka formátování:

```
<text>
      <font size="40" color="#7f6168" face="Verdana">
            <b>Název kategorie</b>
      </font>
</text>
```

Mezi podporované tagy patří <a>, <b>,
, <font>, <img>, <i>, <li>, <p>, , <textformat>, <u>. Tabulky nejsou podporovány.

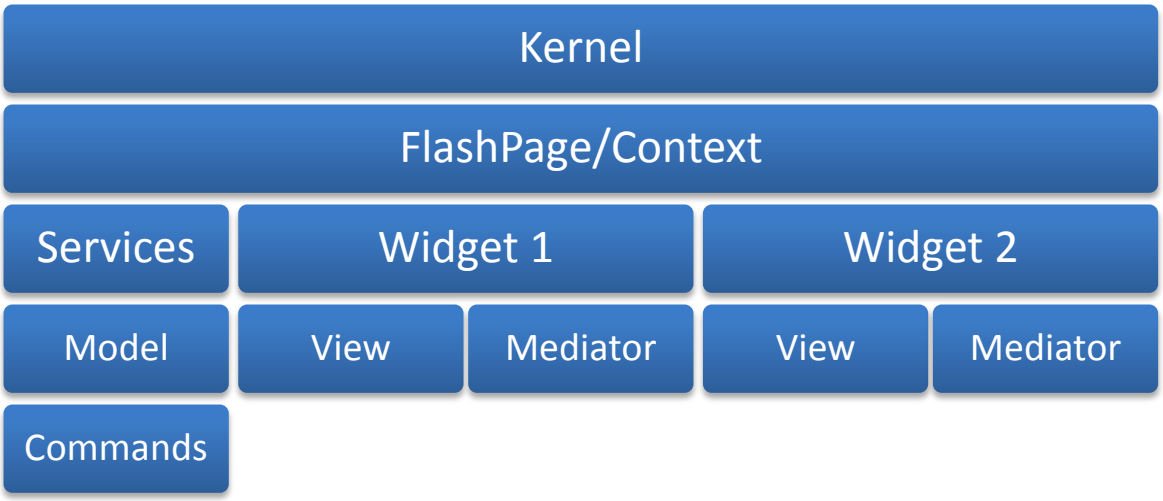
5.2.3. Lokalizace

Pomocí XML definice je možné provést základní lokalizaci bez nutnosti překompilování aplikace. Libovolný tag, v definičním XML, který má atribut *id* je před zpracováním nahrazen jeho lokalizovanou alternativou pro aktuální jazyk, pokud taková alternativa existuje. Takto je možné lokalizovat jak texty, tak měnit komponenty zobrazené na obrazovce, včetně jejich obsahu. Pro lokalizaci slouží samostatné XML, které obsahuje změny. Atribut *id* definuje párování. Jazyk definuje atribut *set*.

```
<localization>
      …
      <value id="1" set="1">Zákulisí</value>
      …
      <value id="59" set="1">
            <value id='0' label='Základní'/>
            …
      </value>
      …
<localization>
```

5.2.4. MVC

Pro architekturu klientské aplikace a implementaci vzoru MVC je použit Framework Robotlegs. Podrobný popis výběru vhodného frameworku je popsaný v šesté kapitole. Ve stejné kapitole je framework Robotlegs podrobně popsán.

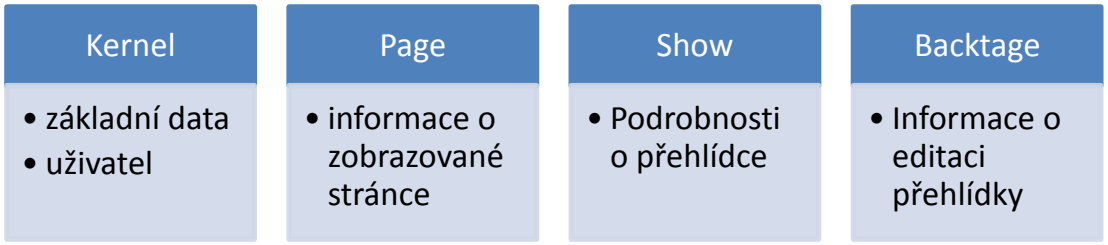


Obrázek 17 - Fyzický pohled na strukturu MVC

Jádrem aplikace je třída *FlashPage* a související třídy *FlashPageContext,* která tvoří jádro MVC frameworku. V rámci této třídy jsou registrovány všechny třídy typu Command a dále všechny vazby tříd pomocí Dependancy injection. Všechny tyto informace nejsou dostupné v době kompilace. Z tohoto důvodu jsou zde další registrace prováděny vždy při načtení dalšího modulu. Součástí *FlashPage* je také třída *FlashPageMediator a FlashPageView,* která osahuje GUI komponenty, které mají být na všech stránkách. Jedná se především o komponentu videa s hardwarovou akcelerací, která musí být na souřadnici *z* pod všemi komponentami. Jedná se o provizorní řešení chyby v knihovně OSMF 16.

Součástí MVC jsou dále služby, které zajišťují komunikaci se serverovou částí aplikace. Podrobněji se jim věnuje kapitola 5.4. Data, která se pomocí služeb získávají, jsou ukládána do modelu. Třídy modelu, stejně jako většina tříd potřebných pro běh MVC, jsou vytvářeny automaticky pomocí dependancy injection až ve chvíli, kdy jsou skutečně potřebné.

Každý widget obsahuje dvě základní třídy. První z nich, *view*, obsahuje GUI komponenty. Dále je to třída *mediator*, která se stará o propojení widgetu s MVC, a to jednak pomocí generování událostí, které obsluhuje MVC event bus, jednak díky přístupu k modelu. Na následujícím obrázku jsou čtyři základní části lokálního modelu s jejich popisem.



Obrázek 18 - Základní části lokálního modelu

Konkrétním komponentám, třídám a implementaci se věnuje kapitola 5.6.

5.2.5. Životní cyklus aplikace

Životní cyklus a zejména načtení aplikace má deset základních kroků, které jsou schematicky popsány na obrázku 19. V první fázi dojde k načtení SWF souboru, obsahujícího jádro aplikace. Následně se načtou všechny RSL knihovny. Dále se načte základní modul s widgety a jejich skin. V této fázi je stažena kompletní aplikace, která může začít komunikovat se serverovou aplikací.

Nejdříve dojde k základnímu přihlášení aplikace ke službám. Následně se stáhnou všechny základní nody z Drupalu, které aplikace ke svému běhu potřebuje. Jedná se o nody s lokalizací, konfigurací a fragmenty stránek.

Všechny následující kroky se týkají každé nové stránky, která se bude v rámci aplikace nahrávat. Nejprve se načte XML obsahující definici konkrétní stránky. Dále jsou v tomto XML provedeny všechny změny vyplývající z lokalizace a skládání fragmentů. XML je následně proparsováno a je podle něj zkonstruována stránka s widgety. Widgety jsou dále upraveny podle případných lokálních dat (obdoba cookies). Stránka je zobrazena a uživatel s ní může pracovat. Může se například přihlásit pod jiným uživatelským účtem, ukládat data pomocí služeb nebo si zobrazit novou stránku.

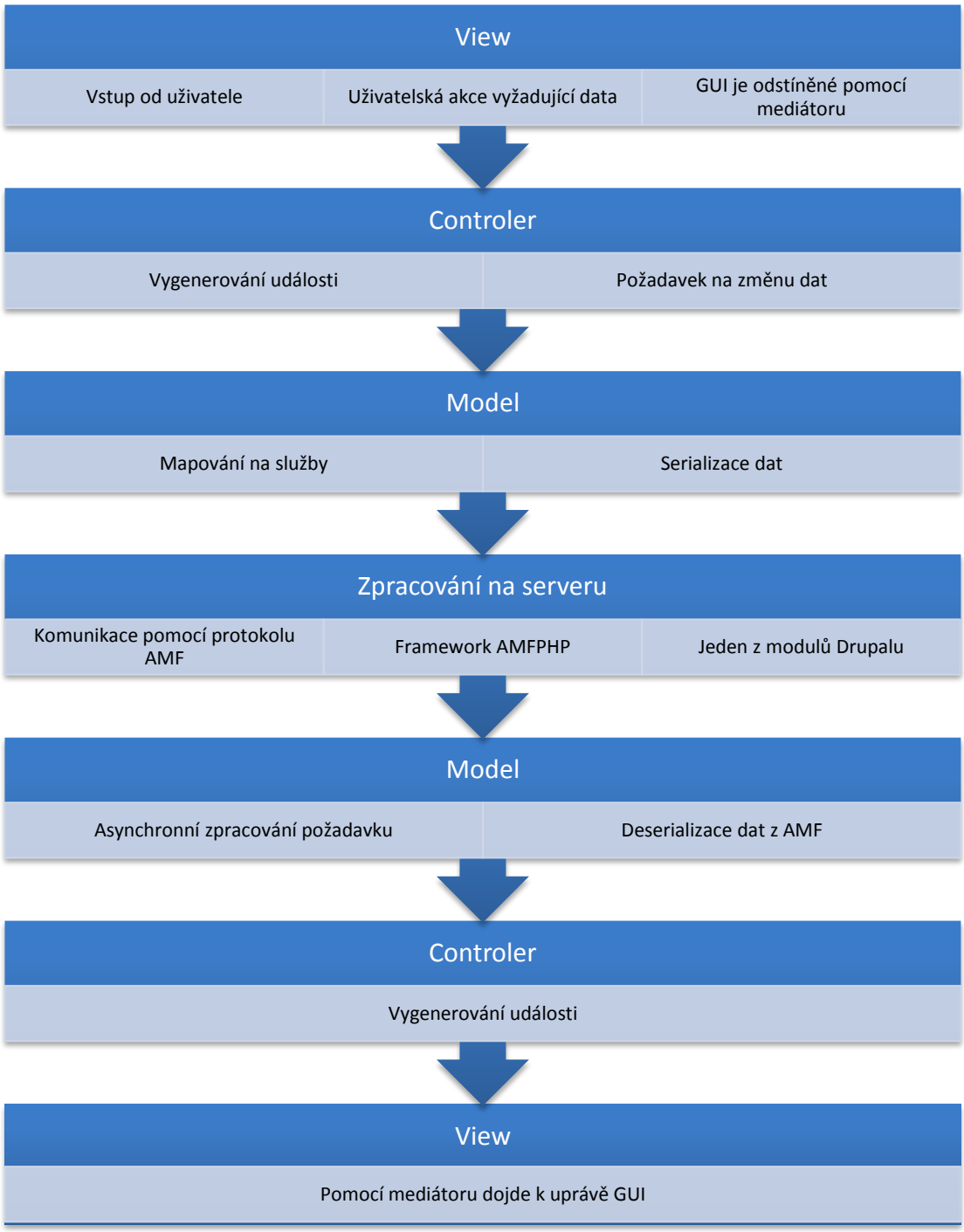
Obrázek 19 - Životní cyklus načtení aplikace

5.2.6. Integrace s prohlížečem

Mezi praktické techniky použité v aplikaci patří Deep linking, která díky virtuálním URL umožňuje snadno odkazovat na částí aplikace – stránky. Pomocí JavaScriptu je tak možné docílit obdobného chování, jaké má klasická HTML stránka vzhledem k historii procházení. Pro implementaci této vlastnosti systému byla použita knihovna SWFAdress [56].

5.3. Integrace pomocí služeb

Služby jsou v rámci klientské části aplikaci Fashionspace chápány jako třídy, které komunikují se serverem pomocí protokolu AMF. Proces využití služeb v rámci aplikace je na obrázku 20. Služby jsou v rámci klientské aplikace využívány zejména k operacím Create, Read, Update and Delete (CRUD).



Obrázek 20 - Komunikace klientské aplikace a serveru pomocí modelu MVC

V aplikaci je možné služby snadno zaměňovat a používat jich celou řadu. Jediné, co je nutné zaručit, je to, aby služby měly stejné rozhraní. Tato vlastnost je velice důležitá v době testování, kdy se služby nahradí mock variantou, která vrací testovací data. Dále je možné podle situace automaticky použít takovou implementaci služeb, která je v danou chvíli nejvýhodnější. V případě migrace systému je tak možné aplikaci snadno napojit na jinou serverovou aplikaci.

Pro každý základní datový objekt existuje v systému služba, která se stará o datové operace nad tímto objektem. Kompletní seznam objektů je v kapitole "Datový model". Jedná se například o objekty uživatel, reklama, video a přehlídka.

Základem implementace služeb je třída *RemoteObject,* která zastřešuje připojení na AMF bránu. Každá služba obsahuje mapování PHP funkcí, které jsou implementovány na straně serveru a jsou propojeny s AMF bránou do AS3 funkcí.

Volání funkcí u služeb je v AS3 asynchronní. V rámci služby jsou definované reakce pro případ, kdy funkce proběhne v pořádku, ale také pokud její zavolání selže. Kromě mapování funkcí jsou mapované i předávané objekty. Základní mapování nepodporuje dědičnost. Z tohoto důvodu je použito vlastní mapování, které dědičnost podporuje a zároveň dokáže AS3 třídy odstínit od specifických vlastností datových typů v Drupalu. Datovým typům, jejich struktuře a mapování se věnuje kapitola "Data a datový model".

5.3.1. CMS Drupal a služby

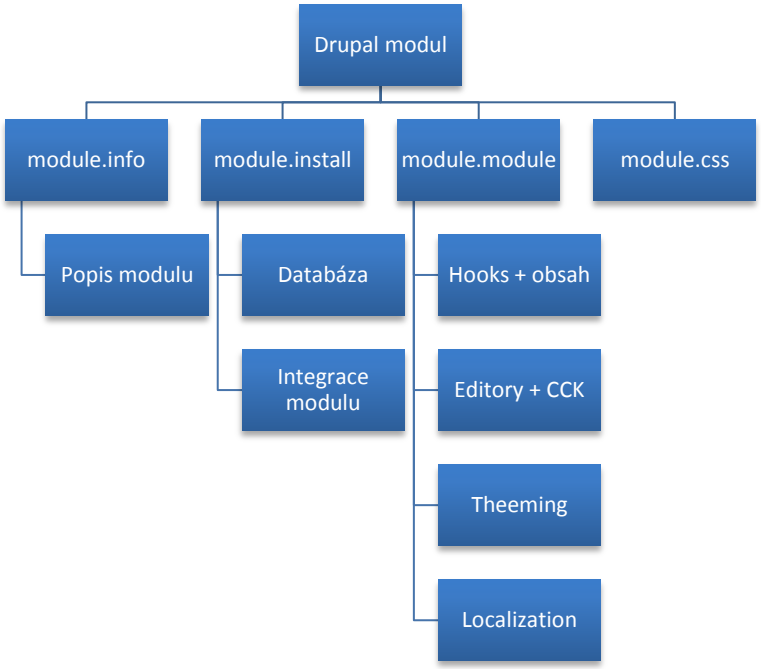
Služby, tak jak je tento pojem chápán v rámci SOA (Service-oriented architecture) nejsou součástí CMS Drupal. Pomocí modulu *services* je ale možné základní podporu služeb do systému integrovat. Pro konkrétní podobu služeb, jako například REST, je nutné do CMS Drupal instalovat nový modul. Jako vhodná implementace služeb byl vybrán modul *amfphp*, který využívá stejnojmennou knihovnu pro vytvoření AMF brány, která je optimalizovaná pro komunikaci s Flashovými aplikacemi.

5.4. Serverová část aplikace

Serverovou část aplikace tvoří zejména instalace CMS Drupal. Základní instalace CMS Drupal je rozšířená o řadu modulů, nových datových typů a služeb.

5.4.1. Moduly CMS Drupal

CMS Drupal je možné snadno rozšiřovat pomocí modulů. V rámci modulu je možné ovlivňovat datový model aplikace, reagovat na události v systému a měnit jeho chování.



Obrázek 21 - Základní struktura modulu v Drupalu

Každý modul v CMS Drupal obsahuje svůj popis v souboru *modul.info*. Instalace a nastavení databáze je v souboru *module.install*. Samotný modul s většinou kódu v PHP je v souboru *module.module* a v souborech do něj linkovaných.

5.4.2. Externí instalované moduly

Základní instalace CMS Drupal obsahuje minimum funkcionality a modulů. U reálného nasazení je tak často nutné nainstalovat až desítky modulů, které rozšíří funkčnost aplikace na potřebnou úroveň. Mezi nejdůležitější externí moduly patří následující:

- **ad** – administrace reklam, sledování přístupu
- **amfphp** – AMF brána nutná pro připojení Flashe ke službám
- **cck** – editor pro snadnou tvorbu nových datových typů
- **content_profile** – s uživatelským profilem lze pracovat jako s nodem
- **date** – usnadněná práce s datumy
- **filefield** – podpora datové položky soubor
- **imagefield** – podpora datové položky obrázek
- **services** – podpora externích služeb
- **views** – podpora pro vytváření pohledů nad daty v podobě nových stránek

5.4.3. Nově vytvořené moduly

Všechny nově vytvořené moduly jsou dostupné ve složce *fssp_service* . Každá služba pak následně tvoří jeden modul. Každý modul obsahuje minimálně následující soubory:

- XY_service.info – popis modulu, verze, vazby
- XY_service.module – definování rozhraní modulu
- XY_service.inc – implementace služeb

Mezi implementované služby patří *advertisements_service*, která zpřístupňuje reklamy, které se mají zobrazit. Dále se jedná o *designer_service, video_service, shows_service, invitation_service* a *models_service*.

5.4.4. Uživatelé a zabezpečení

Pro uživatele a role je plně využit systém CMS Drupal. Uživatelský profil je rozšířen pomocí modulu *content_profile* tak, aby jej bylo možné editovat jako node. V systému existují následující role:

- **Systém** – základní učet nutný k přihlášení ke službám
- **Administrátor** – přístup do administrace
- **Návštěvník** – vstup uživatele, který má vstupenku
- **Návrhář** – uživatel editující přehlídku
- **Anonymní** – obyčejný návštěvník stránek

Pomocí Drupalu je možné snadno nastavit v systému práva pro každou operaci nad daty. Dodatečné oprávnění je možné přidat do každého volání služby. Každá funkce má připravené ověřovací volání podle přihlášeného uživatele a stačí ho dle požadavků implementovat.

5.4.5. Data a datový model

Datový model aplikace obsahuje základní návrh, který bude nutné dále rozšiřovat. Na datový model je možné se podívat z několika pohledů. První pohled patří databázi a tabulkám v CMS Drupal. Díky databázové abstrakční vrstvě umožňuje CMS Drupal snadno definovat nové datové typy a nody tak, aby uživatele odstínil od přímého přístupu k databázi. Databáze obsahuje skoro sto padesát tabulek a její zobrazení není proto přehledné. Z tohoto důvodu bude v této kapitole rozebrán datový model z pohledu mapování dat do tříd jazyka AS3 na straně klientské aplikace. I zde se bude jednat jen o základní náhled. Podrobnější náhled je dostupný v podobě UML modelu aplikace.

Základním kamenem dat na klientské stráně je třída *Vo*, která reprezentuje node v CMS Drupalu. Obsahuje základní vlastnosti:

- **nid** – unikátní identifikátor nodu
- **uid** – identifikátor vlastníka

- **type** – datový typ nodu
- **title** – popisek
- **body** – textový obsah

Kromě datových položek je důležitou součástí též transformace třídy na univerzální objekt, který se pomocí AMF přenáší. Vzhledem k tomu, že se pomocí služeb přenášejí objekty, které mají vazby na jiné objekty, je nutné řešit, do jaké hloubky odkazů se data stahují. Toto je pro každou službu řešeno specificky. Každé volání služby je náročné časově a zároveň zatěžuje server. Z tohoto důvodu existuje několik typů vlastností jednotlivých tříd modelu:

- **S prefixem ro_XY** – obsahují položku, kterou je možné jenom číst, nikoli ukládat. Slouží jako optimalizace proti dalším voláním. Například se může jednat o jméno vlastníka nodu. Vlastnost reprezentuje getter a je tak zaručeno, že je skutečně jen pro čtení
- **Se sufixem XY_link** – obsahuje pouze identifikátor položky. Například se jedná o odkaz na další node nebo uživatele. Používá se zejména u funkcí, které vrací dlouhé seznamy
- **Ostatní vlastnosti** - obsahují data, která je možné pomocí služeb jak číst, tak ukládat

Mezi základní datové objekty, které jsou potomky třídy *Vo* patří:

- **UserVO** – reprezentuje uživatele a jeho profil
- **ShowVO** – informace o přehlídce
- **VideoVO** – záznam o videu a všech jeho variantách
- **InvitationVO** – reprezentuje pozvánku na přehlídku, zejména unikátní klíč
- **ModelVO** – informace o jednom modelu
- **NewsVO** – novinky zobrazované před přehlídkou
- **AdvertisementVO** – jedna reklama v systému

Kompletní seznam tříd je dostupný v podobě UML diagramů v následující kapitole.

5.5. Struktura projektů a zdrojové kódy

Aplikaci lze rozdělit na dvě základní části. První část tvoří zdrojové kódy klientské a serverové aplikace. Druhou část tvoří špatně přenosná konfigurace všech serverů, které aplikace využívá (streaming, IM, FTP, …).

Zdrojové kódy klientské aplikace, které tvoři většinu, reprezentují čtyři projekty pro Flash Builder. Projekt je možné exportovat do balíčku FXP, který lze snadno importovat.

- **FlashPage** – hlavní aplikace obsahující loader
- **Kernel** – RSL knihovna obsahující základní prostředí
- **Widgets_default** – základní sada widgetů použitá v projektu
- **Skin_default** – skin pro základní sadu widgetů

Projekt *FlashPage* je jednoduchý a obsahuje jen grafiku loaderu, která nemůže být z technických důvodů součástí skinu a vazby na RSL knihovny. Projekt *Skin_default* obsahuje jedno centrální CSS s vazbami na všechny skiny. Z CSS se generuje jeden SWF soubor.

Projekt *Widgets_default* obsahuje základní sadu widgetů. Každý widget XY se nachází ve vlastním balíčku *com.fssp.wigets.XY*. Každý widget reprezentuje minimálně jeden soubor XY.mxml tvořící view a soubor XYMediator.as reprezentující mediátor. Všechny widgety musí být následně registrované ve třídě *com.fssp.widgets.Widgets_default*.

Jádro aplikace reprezentuje projekt *kernel.* Mezi nejdůležitější balíčky aplikace patří:

- ***com.fssp.kernel.flashpage*** – Zejména třída FlashPage vykreslující stránku
- ***com.fssp.kernel.parser*** – Zpracování definičního XML
- ***com.fssp.kernel.widgets*** – Základní třídy pro implementaci widgetů

Následující balíčky jsou z praktických důvodů také součástí jádra. V případě použití aplikace pro jinou doménu, než jsou přehlídky, by bylo nutné tyto třídy přesunout do samostatného modulu, protože jsou doménově specifické.

- ***com.fssp.kernel.model*** – Třídy reprezentující model aplikace
- ***com.fssp.kernel.commands*** – Příkazy pro MVC
- ***com.fssp.kernel.services*** – Implementace služeb
- ***com.fssp.kernel.services.vo*** – Mapování objektů
- ***com.fssp.kernel.services.data*** – Lokální verze nodů

Zdrojové kódy pro CMS Drupal reprezentuje projekt pro Eclipse. Součástí projektu jsou všechny služby v podobě PHP kódu. Struktura modulů je popsaná v kapitole 5.4.3.

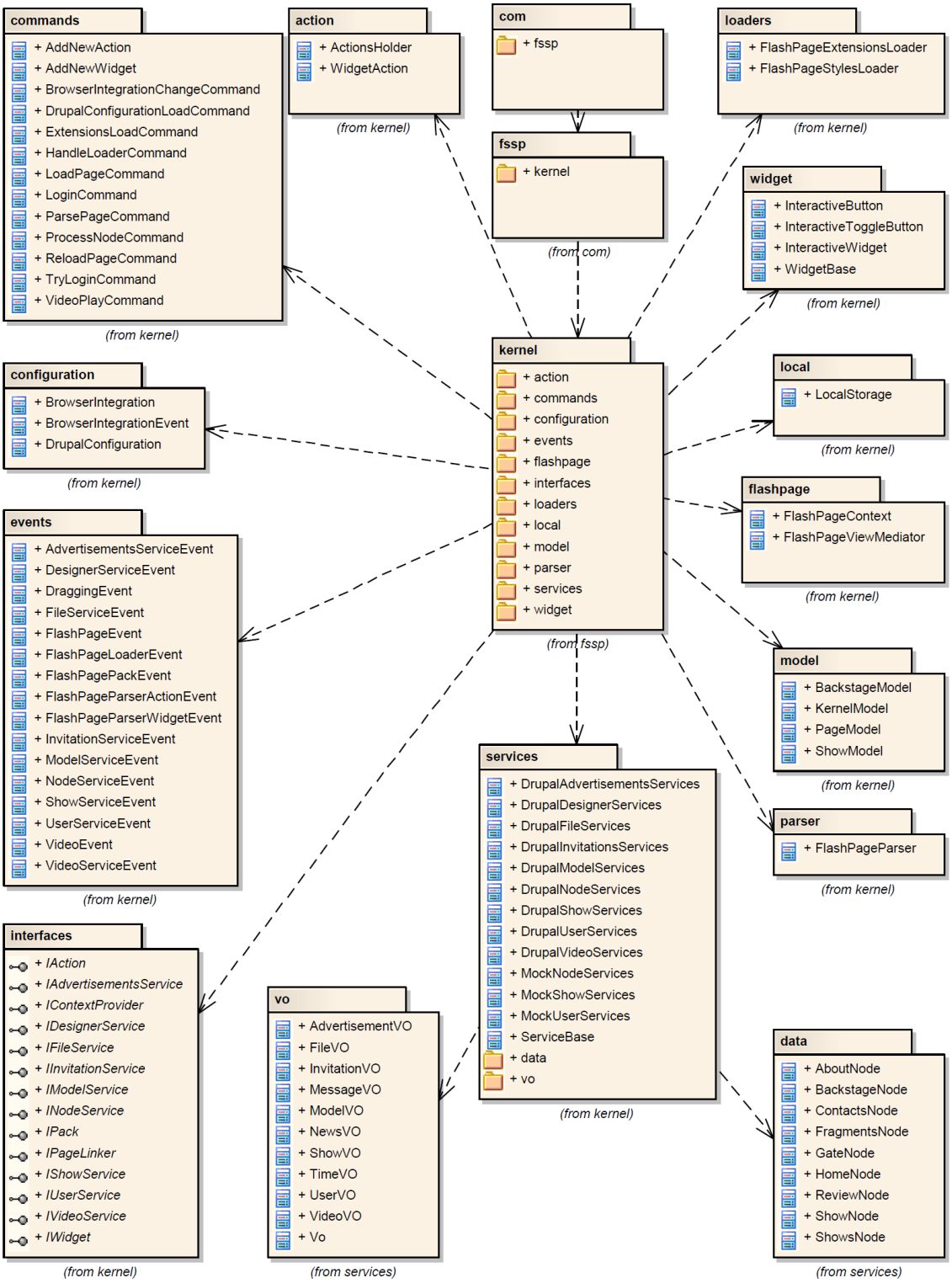
5.5.1. Přehled implementované funkčnosti

Množství implementovaného kódu je popsáno v následující tabulce. Rozsah kódu vytvořeného pro klientskou aplikaci řádově převyšuje rozsah kódu serverové aplikace. Hlavním důvodem je to, že vývoj aplikace využívající CMS Drupal je zaměřen spíše na konfiguraci a skládání modulů, než na implementaci nového kódu.

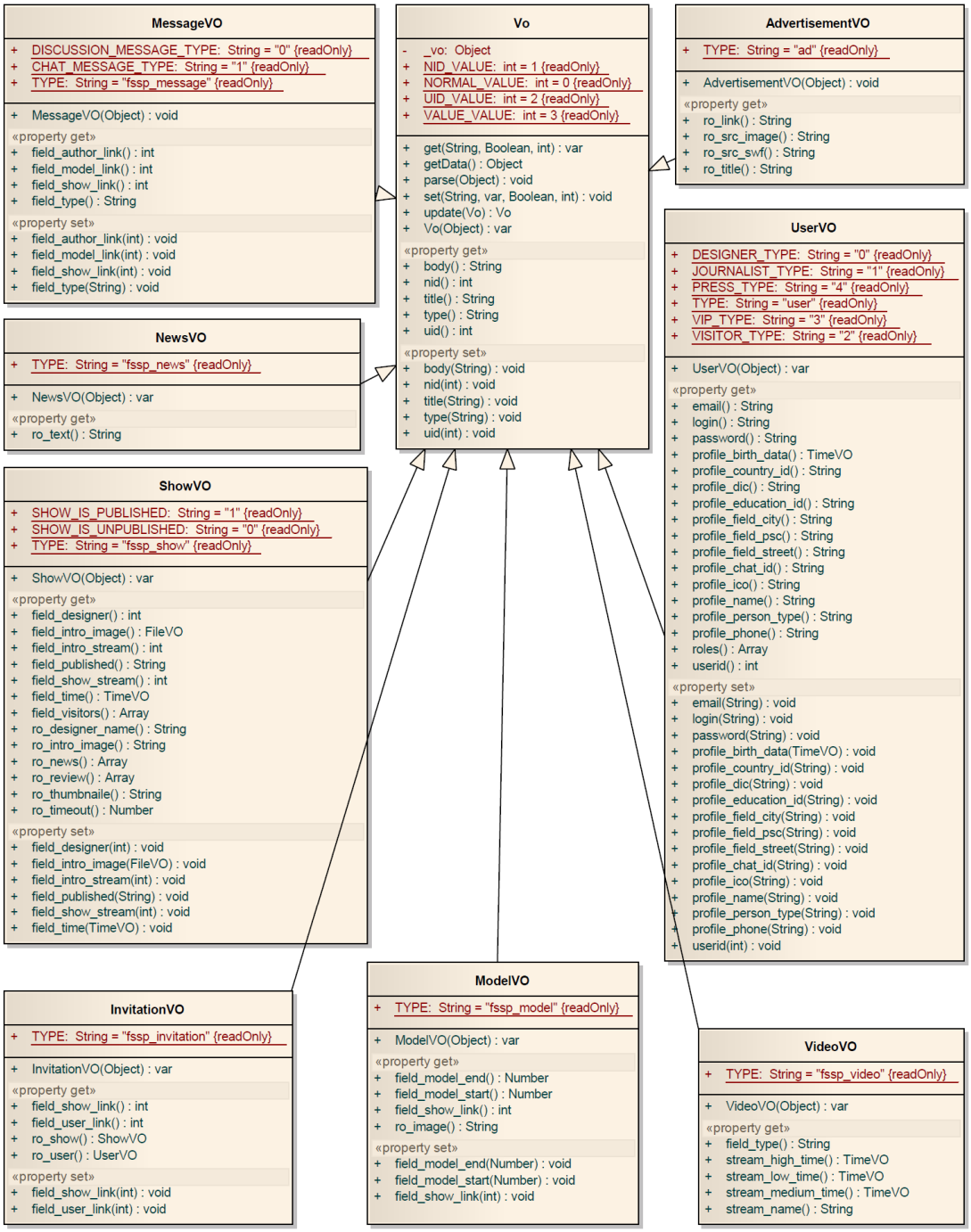
Modul	Rozsah	SLOC	Velikost
Klient – jádro	98 tříd	~5500	~200 KB
Klient – widgety	100 tříd / 30 widgetů	~6000	~200 KB
Klient – skin	60 tříd	~4500	~200 KB
Server – služby	15 nových služeb	~1000	~30 KB

Tabulka 1 - Rozsah implementovaného kódu

Na obrázcích 22, 23 a 24 jsou dostupné UML diagramy pro jádro aplikace a widgety.

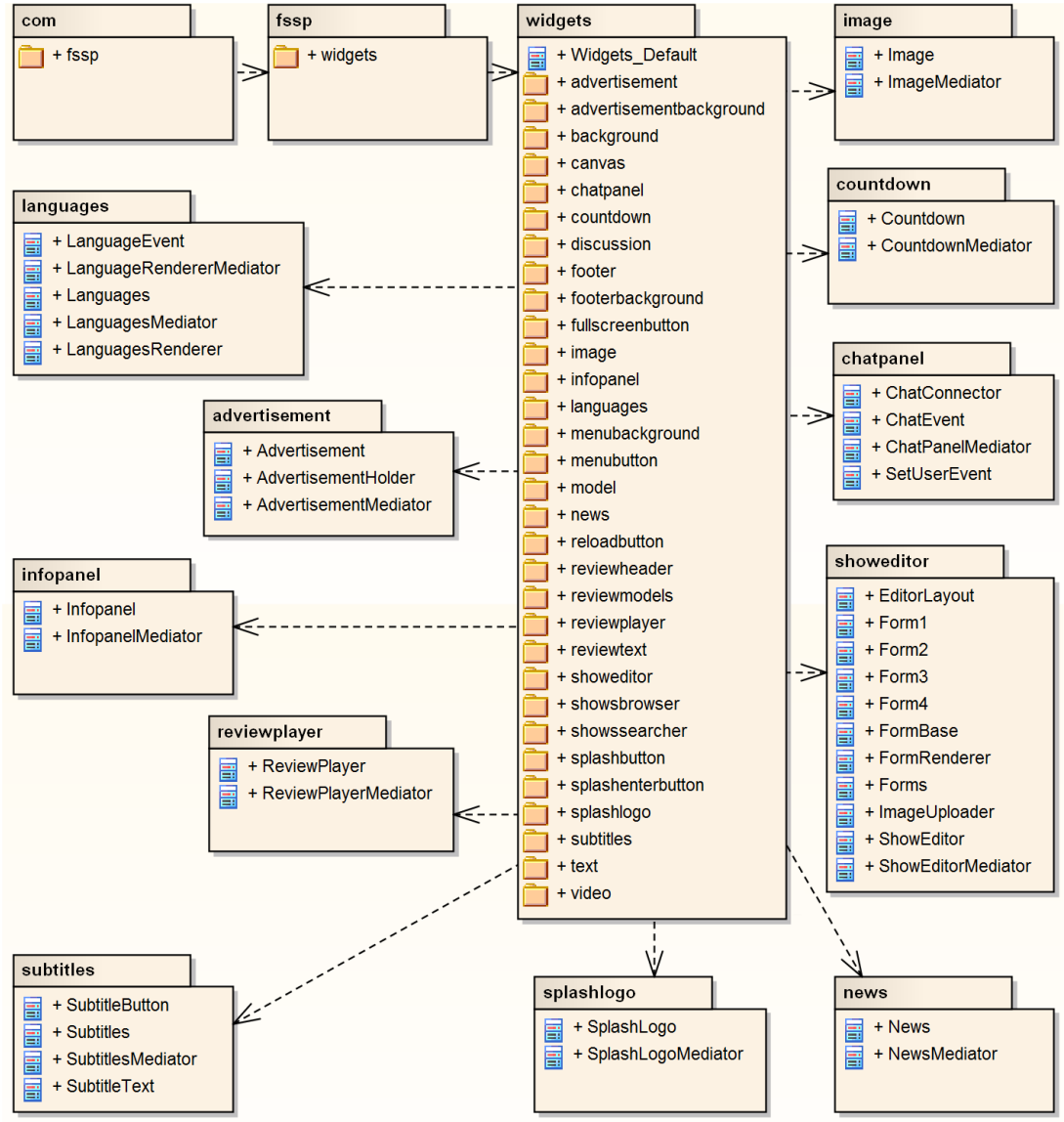


Obrázek 22 - UML pro třídy implementované v rámci jádra aplikace



Obrázek 23 - Část UML obsahujícího mapování datové vrstvy do tříd AS3

Po funkční stránce je kompletně implementované jádro aplikace, které se stará o vykreslení stránek, správu widgetů a komunikaci se serverem. V rámci práce bylo dále implementováno 30 základních widgetů, jejichž přesný seznam je na obrázku 24. Pomocí základních widgetů, jako je text, video přehrávač, menu nebo interaktivní logo, jsou vytvořeny základní stránky popsané na obrázku 8.



Obrázek 24 - Část UML obsahující implementované widgety

Pro online komunikaci je připravený widget *chatpanel*, pomocí kterého je dostupný seznam uživatelů přihlášených na přehlídce. Widget dále umožňuje komunikaci s libovolným návštěvníkem. Další funkčnost zajišťuje widget *discussion*, který umožňuje hromadnou diskusi všech návštěvníků na jednom místě. Po přihlášení návštěvníka se zobrazí i předešlá historie diskuse. Pro diskusi je připravený a nakonfigurovaný server Openfire.

Pro editaci přehlídky je dostupný widget *showeditor*, v rámci kterého je možné založit přehlídku, přidat k ní základní informace a nahrát video na FTP server. Tento widget pro svůj běh používá FTP server, streamingovou a transcodingovou službu.

Aplikace je nastavená, aby využívala služeb streamhosting.cz. Instalace CMS Drupal je nakonfigurována tak, aby bylo možné editovat jak data aplikace, tak spravovat uživatele. Dále jsou vytvořené pohledy pro snadný přehled přehlídek v systému.

V základní podobě jsou tak implementované všechny stránky z obrázku 8 a jsou předpřipravené a spojené všechny komponenty systému z obrázku 10. V rámci implementace tak bylo prokázáno, že danou aplikaci je možné pomocí navržené architektury implementovat. Pro reálné použití by ale bylo nutné implementovat ještě celou řadu funkcí, widgetů a upravit administraci aplikace v CMS Drupal.

6. Výběr MVC a mikroarchitekura

Architektura aplikace je postavena na architektonickém vzoru Model-view-controller (MVC), který patří mezi nejrozšířenější softwarové architektury, používané pro návrh rozsáhlejších klientských aplikací postavených na frameworku Flex. Této situaci také odpovídá množství knihoven a projektů, které se snaží navrhnout vhodnou implementaci MVC. Základním principem MVC je rozdělení aplikace na tři části [40], a to:

- **Model** - reprezentuje doménově specifickou reprezentaci dat, s kterými aplikace pracuje
- **View** - převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli
- **Controller** - reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu

Kromě MVC existuje celá řada dalších softwarových architektur, jako například Autonomous View [41]. Tyto architektury ale nejsou buď vhodné pro velké aplikace, nebo nejsou tolik známé a podporované.

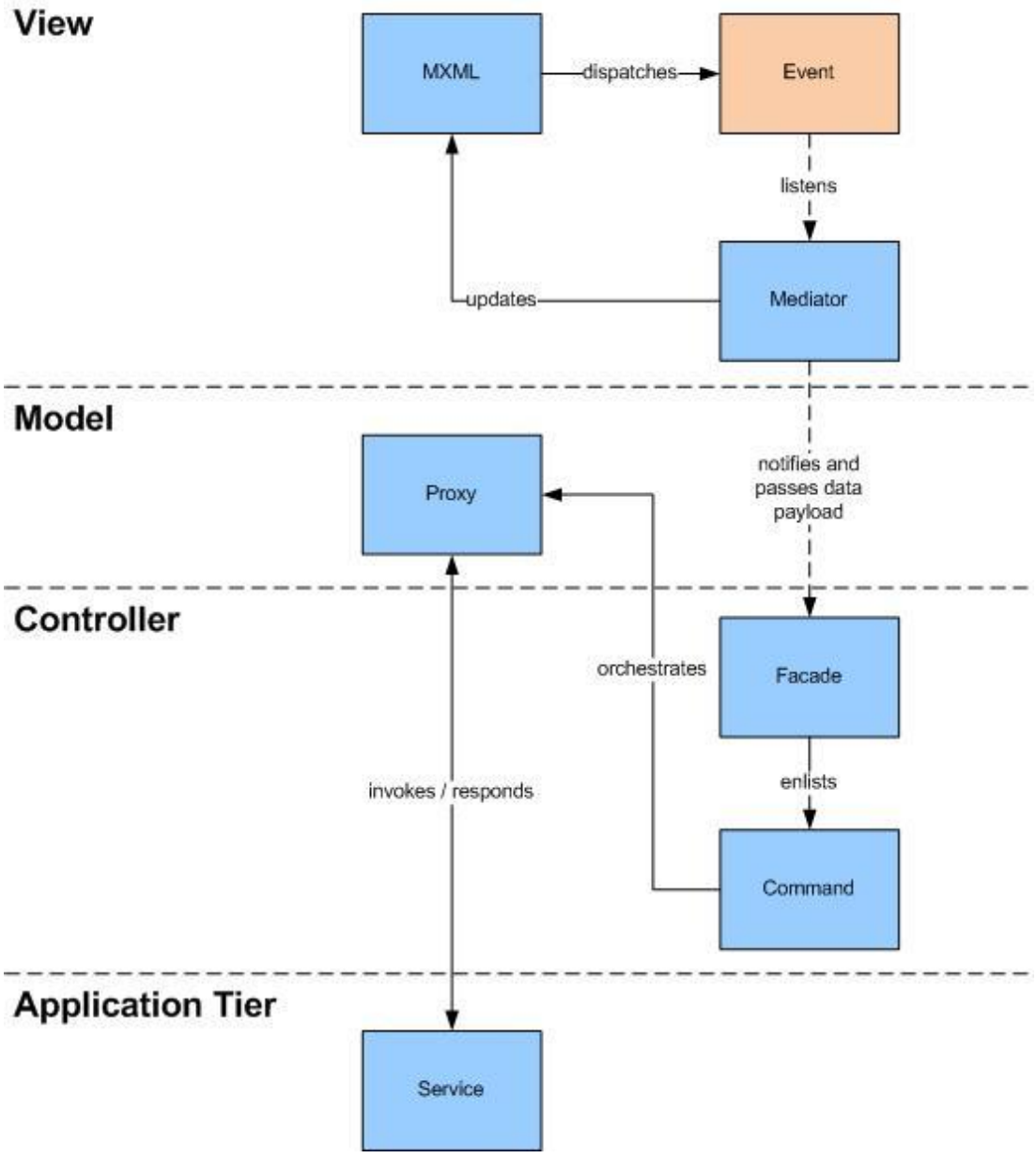
6.1. PureMVC a Cairngorm

Mezi nejznámější frameworky patří PureMVC a Cairngorm. Oba frameworky jsou velice rozšířené a dobře podporované. Pro oba projekty je dostupná kvalitní dokumentace a automatické generátory kódu.

Obrázek 25 - Základní komponenty systému PureMVC

Framework PureMVC je možné použít v celé řadě jazyků a platforem, jakými jsou například Java, C#, PHP, JavaScript, Ruby, Python, AS3, AS2 a jiné [42]. Tato

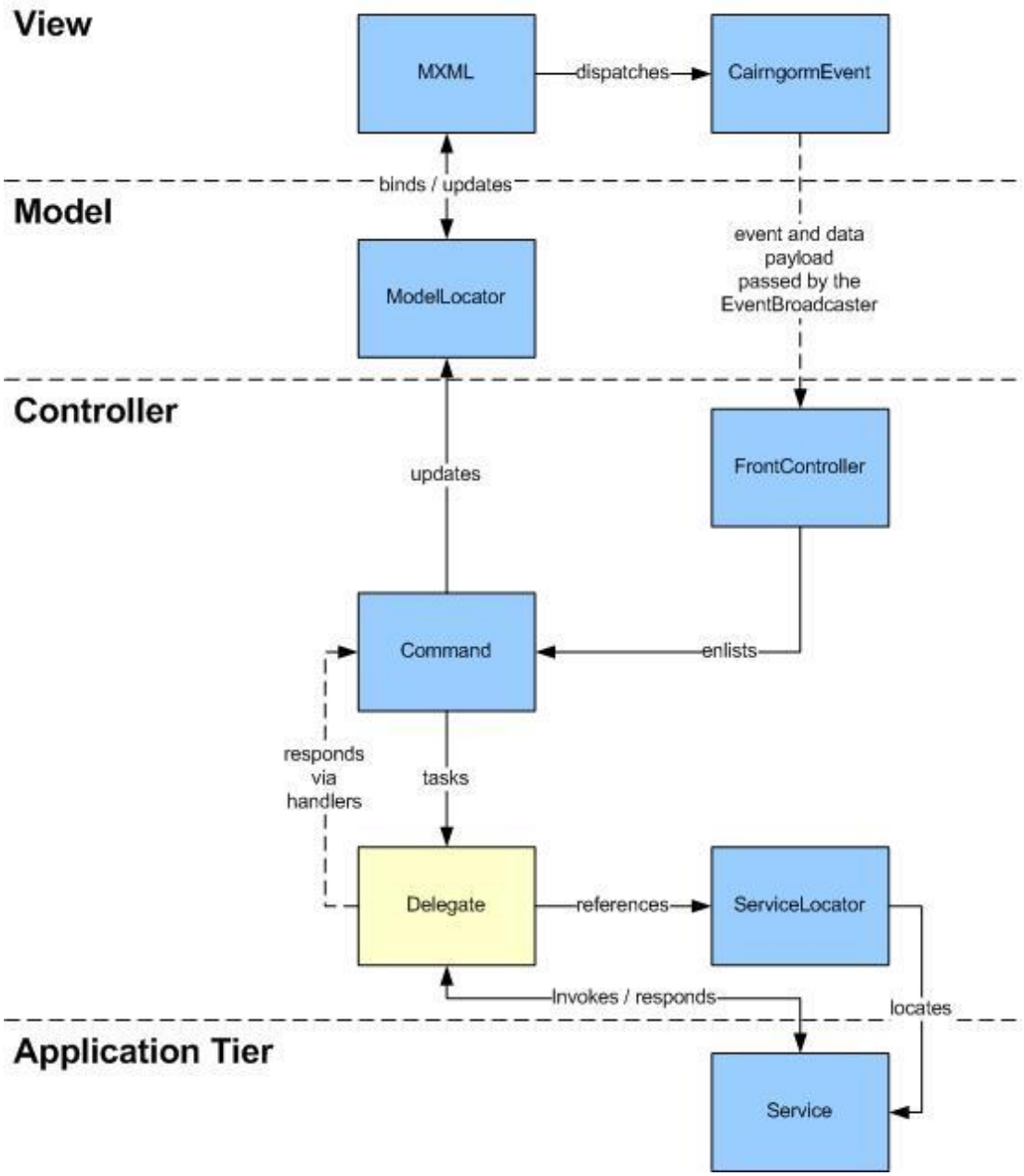
multiplatformnost u frameworku bohužel vede k tomu, že nejsou dostatečně využitá specifika konkrétních jazyků a platforem. Výsledný kód je sice robustní, ale je ho potřeba daleko více, než u jiných frameworků. V rámci frameworku je například implementovaný vlastní systém pro správu událostí a nevyužívá se interní optimalizovaný systém Flash Playeru. Dále není použitý databinding a aktualizace UI je nutné naprogramovat.



Obrázek 26 - PureMVC podle [44]

Framework je vhodný zejména pro programátory, kteří přecházejí z jiných programovacích jazyku nebo platforem, protože neklade vysoké požadavky na znalost Flex SDK.

Framework Cairngorm je oficiálně podporovaný firmou Adobe a tomu odpovídá kvalita dokumentace a provázanost s Flex SDK. Framework je jednoduchý a přímočarý.

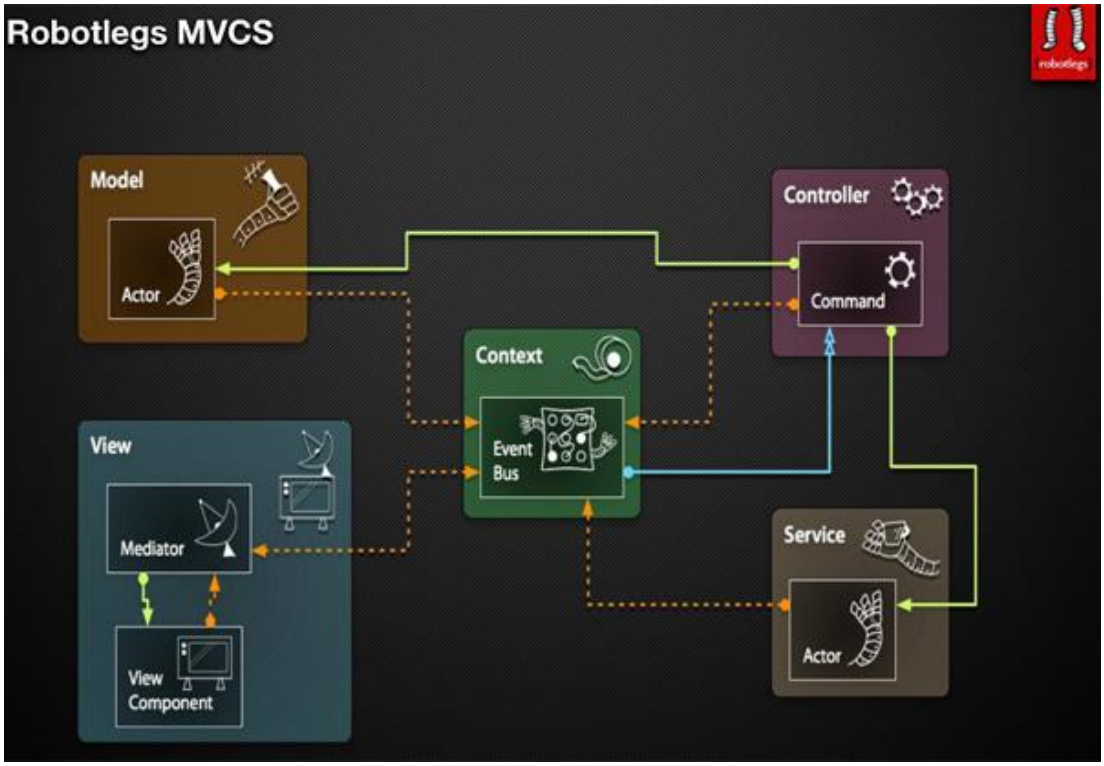


Obrázek 27 - Cairngorm podle [44]

PureMVC a Cairngorm lze považovat za klasické frameworky, které jsou použity pro celou řadu projektů. Nová generace frameworků, jako například Robotlegs nebo Mate, v rámci implementace MVC přináší řadu zlepšení.

6.2. Robotlegs

Pro implementaci projektu Fashionspace byl vybrán framework Robotlegs. Tento framework není prostou implementací MVC, jedná se již o komplexnější mikro architekturu pro vývoj Flash a Flex aplikací.



Obrázek 28 - MVCS u frameworku Robotlegs [45]

Autoři frameworku místo klasického MVC používají pojmenování MVCS - Model View Controller and Service, protože součástí návrhu jsou i služby. Framework plně využívá možnosti Flexu. Zdrojový kód je díky tomu přehledný a pro implementace jednotlivých částí je nutné minimum kódu.

6.2.1. Dependency Injection

Mezi klíčové techniky, na kterých je framework Robotlegs postavený, patří Dependency injection (DI). Tato technika se používá v rámci celého projektu a je důležité ji podrobněji popsat. Podle [46] je DI programovací technika používaná

v OOP pro vkládání závislostí mezi jednotlivými komponentami programu tak, aby jedna komponenta mohla používat druhou, aniž by na ni měla v době sestavování programu referenci. DI je možné chápat jako novější název pro Inversion of Control (IoC), ale v užším mínění. Jedná se o konkrétní techniku IoC.

Pokud nepoužijeme DI, potom objekt A, který chce využít služby jiného objektu B, zodpovídá za celý životní cyklus komponenty B. Pokud však aplikujeme DI, bude objekt A odstíněn od této správy, neboť ji zajišťuje dependency provider. Objekt A potom potřebuje už jen referenci na dependency providera a provider je mu schopen dodat hned několik různých komponent, které splňují jeho očekávání. Každá z těchto komponent tak může objektu A poskytovat rozdílné služby, a právě v tom spočívá síla DI.

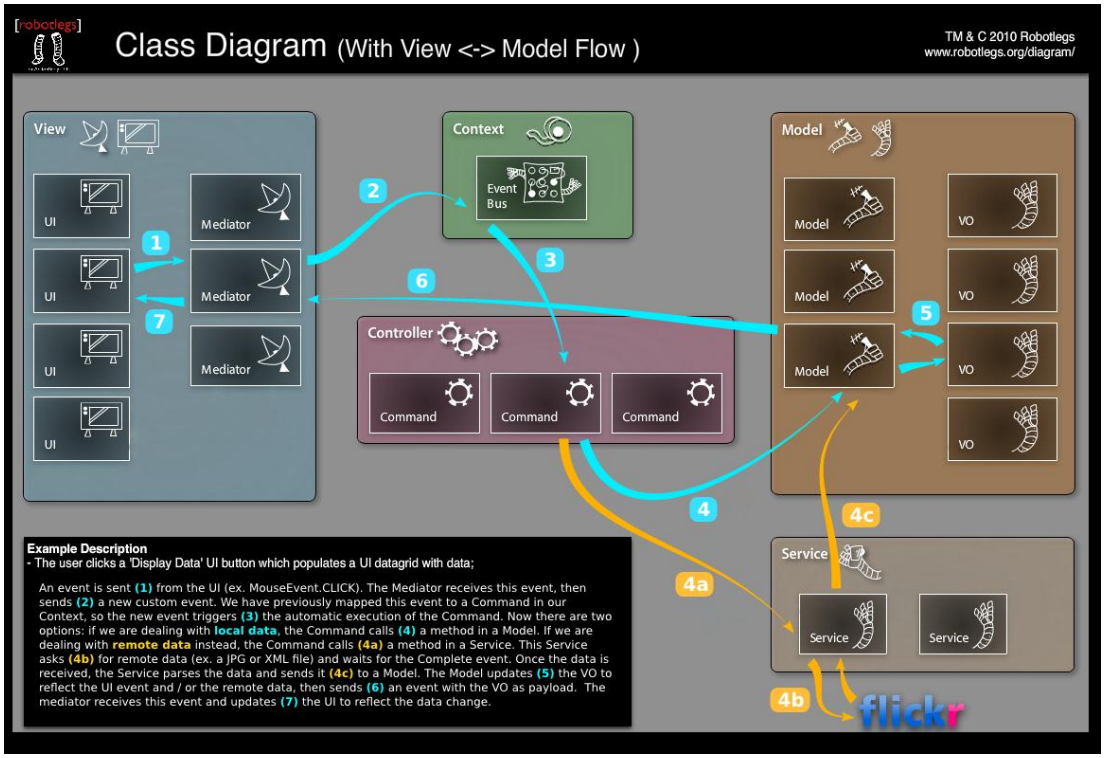
Robotlegs plně využívá síly DI, díky které je možné snadno testovat aplikace. Bez nutnosti změny celé aplikace, pouze úpravou jedné definice, je možné například nahradit napojení na server za testovací (mock) připojení.

V rámci tříd MVC modelu není nutné řešit dependency providera a všechna pravidla je možné definovat pomocí metadat. V třídách je možné mít property v podobě interface. Jaké konkrétní třídy a v kolika instancích se vytvoří, je zcela na DI. Jedná se o velký posun v urychlení práce a přehlednosti kódu. Díky GC není nutné vždy explicitně definovat destrukci objektů. Díky definovaným pravidlům pro DI, není nutné již ručně definovat konstrukci objektů. Objekty se vytvoří ve chvíli, kdy je jich potřeba, a jsou tak snadno sdílené napříč aplikací.

6.2.2. Struktura frameworku

Jádrem frameworku je potomek třídy Context, v kterém jsou definice pro DI. Na začátku se vytvoří komponenty GUI. Následkem této operace se automaticky vytvoří potřebné mediátory. Po provedení uživatelské akce je událost odchycena nejdříve mediátorem a následně zpracována v Controlleru pomocí potomka třídy Command. Podle potřeby dojde k vytvoření tříd modelu. Pokud informace z modelu jsou dostatečné, je následně aktualizováno GUI. V opačném případě jsou získána

aktualizovaná data pomocí služeb ze serveru a následně dojde přes mediátor k aktualizaci GUI.



Obrázek 29 - Podrobný popis fungování frameworku Robotlegs podle [47]

7. Video a streaming

Mezi klíčové části práce patří práce s videem. Proces zpracování je možné rozdělit na tří základní části. V prvním kroku dojde k nahrání videa od klienta na server. V následující fázi dojde k jeho konverzi. Pokud je video zkonvertované do podoby, kterou je možné streamovat, může jej uživatel použít k následné editaci přehlídky a zveřejnění ostatním uživatelům.

7.1. Upload

Vzhledem k tomu, že je nutné, aby bylo možné nahrávat soubory přesahující velikost 2 GB, tak nebylo možné použít klasický upload pomocí HTTP. Ideální variantou se proto jeví varianta uploadu na FTP server. Tuto variantu lze považovat za standard mezi poskytovateli transcodingu.

Pro upload souboru na FTP by bylo ideální použít možnosti Flexu, který umožňuje snadné připojení pomocí socketu. Po dlouhém testování se toto řešení ukázalo jako nepoužitelné. Aby se Flexová aplikace mohla na FTP server připojit, musí být na tomto severu na portu 516-523 dostupné crossdomain informace definující domény, které mají právo na FTP server přistupovat.

Pro komunikaci přes FTP neexistuje v AS3 žádná standardizovaná knihovna. Knihovny pro práci s FTP existují, nicméně nejsou spolehlivé pro velké soubory, protože Flash Player již několik let obsahuje chybu v generované události o dokončení přenosu jednoho fragmentu souboru. Problém lze částečně obejít, ale není možné garantovat plnou spolehlivost na všech typech internetového připojení a všech velikostech souborů.

Protože nebylo možné pro upload použít Flash, používá aplikace Fashionspace pro upload souborů již standardizovaný postup, který používá například aplikace Youtube, a to Java applet.

Pro upload souborů pomocí Javy na FTP existuje celá řada knihoven a aplikací. Pro hledané řešení bylo klíčové následující:

- Integrace s Flashovou aplikací tak, aby měl uživatel pocit jednolitého prostředí
- Aplikace by měla umožňovat určit název uploadovaného souboru
- Aplikace by měla umožňovat pokračovat v uploadu při výpadku spojení
- Aplikace by měla vyžadovat minimum dodatečné práce a měla by se snadno integrovat

Ze všech testovaných řešení se tomuto nejvíce přiblížila JavaScriptová knihovna IntegralFTP [48], která pro upload používá Java applet. IntegralFTP je unikátní tím, že je snadno ovladatelná pomocí JavaScriptu, pomocí něhož lze splnit všechny požadavky výše popsané. Do budoucna by bylo vhodné pro upload vytvořit desktopovou aplikací, která bude sloužit jako alternativa k webové aplikaci a části uživatelů by mohla lépe vyhovovat.

7.2. Konverze

Mezi nejznámější nástroje pro konverzi Flash videa na severu patří FFmeg [49]. Pro tento nástroj existuje i modul pro CMS Drupal. Pomocí tohoto nástroje by bylo možné vytvořit vlastní konverzní server, který by byl napojený na streamovací server. Toto řešení by bylo možné, ale již se vzdaluje návrhu a řešení portálu Fashionspace, který představuje hlavní náplň práce.

V současné době, kdy dochází k silnému posunu v chápání IT a software jako služby (SaaS) [51] a objevují se fenomény jako Cloud computing [50], se mohou sofwarové projekty čím dál více zaměřovat pouze na svoji doménu a využívat hotových řešení třetích stran jako služeb. Z těchto důvodů se ukazuje jako klíčový prvek projektů integrace a standardizace.

Konverzní služby nabízí celá řada poskytovatelů, jako například heywatch.com, nebo encoding.com. Služby nabízejí integraci přes FTP, ale i pomocí jiných webových uložišť, jako je například S3 od firmy Amazon. Konverzi je následně možné ovládat pomocí webových služeb a využit notifikace prostřednictvím e-mailu.

Pro konverzi a následně i streaming byla vybrána služba od stremhosting.cz. Konverzi bylo možné snadno konfigurovat pomocí definičního XML, ve kterém byl popsán proces konverze do požadovaných formátů. Jako vstup pro video slouží FTP server. Po konverzi je video nahrané na streamovací server, využívající aplikaci Wowza Media Server.

7.3. Streaming

U streamování videa je situace obdobná jako u konverze, škála poskytovatelů je však daleko širší. Podle [52] existuje celá řada světových poskytovatelů. Velké firmy jako Akamai a Amazon poskytují své služby pro uložení a transport dat specializovaným firmám, které se zaměřují i na streaming videa.

Vedle služeb je též možné použít vlastní servery. Mezi nejznámější aplikace, které je možné použít, patří například oficiální řešení od firmy Adobe a to Flash Media Server. Jedná se o robustní řešení, jehož cena je však pro začínající projekty nedostupná, protože se pohybuje kolem 4500$. Existují samozřejmě i levnější varianty, například již zmíněný Wowza Media Server nebo kvalitní open source varianta Red5.

Pro streamování videa byl vybrán formát H.264 a kontejner MP4, který je v současnosti považován za standard u HD videa, a to nejen pro Flash.

8. Online komunikace uživatelů

Online komunikace se v systému vyskytuje ve dvou podobách. První reprezentují komentáře na přehlídce, které by měli všichni najednou vidět. Druhou část tvoří komunikace mezi návštěvníky přehlídky.

8.1. Moduly pro CMS Drupal

Jako první testovaná varianta byly vybrány systémy, které je možné snadno integrovat do CMS Drupal jako modul. V CMS Drupal neexistuje žádný standardizovaný modul pro chat a je tedy nutné vybrat nejvhodnější z celé řady možností.

V první fázi testování se jednalo o modul Chat Room [32]. Modul využíval plně datovou strukturu CMS Drupal a bylo možné jej snadno administrovat. Integrace klientskou aplikací proběhla bez problémů. Problém nastal při testování robustnosti řešení. Když počet uživatelů dosáhl zhruba deseti, dosahovala latence systému desítek vteřin, což bylo nepřijatelné. Alternativní modul Drupalchat pro komunikaci používá techniku long-polling [33], která zajišťuje menší zátěž, než klasické http dotazy. Po výkonnostní stránce se však od modulu Chat Room neliší.

Modul Dxmpp [34] představuje daleko výkonnější řešení, než předchozí moduly. Toto řešení je postavené na protokolu XMPP a komunikace využívá principu Comet. Modul je bohužel stále ve verzi beta a není ho možné snadno použít.

Mezi moduly, které vytvářejí pouze konektor na samostatný chat server, patří 123FlashChat [35]. Jedná se o systém rozšířený, nicméně komerční a se špatnou možností úpravy klientské aplikace. K dalším testovaným modulům patřil konektor na Phpfreechat [36], s nímž bylo možné dosáhnout maximálně desítek paralelních připojení.

8.2. Integrace samostatného serveru pro IM

Aplikace Fashionspace by měla zvládat i sto uživatelů, kteří budou spolu na přehlídce komunikovat. Proto bylo nutné vybrat vhodný robustní chatovací systém, který bude možné snadno integrovat do CMS Drupal.

Jako první možnost se nabízí použití aplikace Adobe Flash Media Interactive Server 4. Jedná se o robustní řešení, které je možné snadno integrovat do Flashové aplikace. Bohužel se ale jedné o komerční řešení, které není pro chat primárně určené a bylo by nutné velkou část logiky serverové aplikace doprogramovat.

Vzhledem k tomu, že obdobné projekty, jak bylo podrobněji popsáno ve třetí kapitole, využívají standardizované protokoly jako XMPP a pro komunikaci se nejčastěji používá principu Comet, byl vybrán pro základní instalaci server Openfire ve verzi 3.7. [37], který toto splňuje. Server je primárně určený pro real time collaboration (RTC) a je dostupný pod open source licencí GPL. Server se snadno instaluje, jeho součástí je webová administrace. Server dále nabízí kvalitní zabezpečení a vysokou výkonnost.

Součástí projektu Openfire je také framework XIFF, který umožňuje snadnou integraci protokolu XMPP do Flashových aplikací. Pro komunikaci klienta se serverem je možné použít jak HTTP protokol pomocí techniky Bidirectional streams Over Synchronous HTTP (BOSH), tak v případě přímého spojení optimalizovanější Real Time Messaging Protocol (RTMP).

U prototypu je integrace uživatelů CMS Drupal a serveru Openfire řešena triviálně, kdy na serveru Openfire je předgenerována sada uživatelů. Stejný princip generování pak využívá i CMS Drupal. Při reálném nasazení je pak možné využit pro integraci uživatelů LDAP server, který podporuje jak Openfire, tak CMS Drupal.

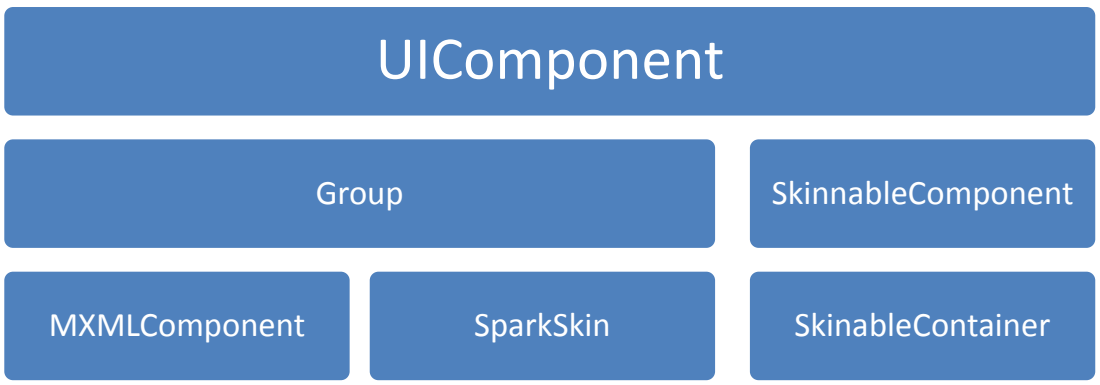
Kromě Openfire existují další servery, jako například Ejabberd [38] a jabberd14 [39]. Oproti serveru Openfire mají méně přívětivou administraci a instalaci. Další zásadní výhodou Openfire je knihovna XIFF. Kromě vlastní instalace XMPP serveru je možné použit i hostingy, jejichž součástí jsou spravované instalace XMPP serveru.

9. Skinování aplikace

Pro skinování Flexových komponent existuje celá řada možností. Od verze Flex SDK 3.x je možné využít Halo komponenty, které lze skinovat pomocí CSS, nebo pomocí vytvoření skinu v podobě SWF souborů. V této verzi však není zcela oddělena logika od vzhledu a možnosti skinování jsou omezené. Tyto možností rozšiřují různé frameworky, jako například Degrafa, ale není to ideální řešení. Nejen z těchto důvodu je součástí SDK 4.0 nová generace komponent Spark.

V sadě komponent Spark je oddělena striktně logika od vzhledu. Je zde vytvořený propracovaný systém stavů vzhledu a komponent, který má elegantní deklarativní syntaxi v jazyce MXML. Každému atributu je možné přiřadit sufix stavu oddělený od názvu atributu tečkou, který definuje hodnotu pro určitý stav. Dále je možné pro libovolný node nastavit, v jakém stavu se vyskytuje. Pro definici vzhledu je použit jazyk FXG, díky kterému je možné snadno využít celou řadu nástrojů z rodiny Adobe Creative Suit, zejména Adobe Flash Catalyst.

V současné době nemají všechny GUI komponenty Spark variantu a je nutné kombinovat Halo a Spark komponenty. Halo komponenty se při použití SDK 4.x daleko hůře skinují, protože musí být kompatibilní se Spark komponentami, kde je skinování řešeno zcela jinak. Spark komponenty jsou jistě cesta dobrým směrem, jen je nutné převést co největší množství komponent do této podoby a přidat plnou podporu automatizace napříč celou rodinou produktů Adobe Creative Suite.

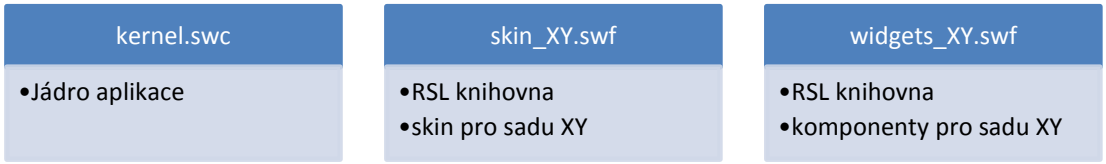


Obrázek 30 - Základní třídy nutné pro skinování Spark komponent

9.1. Balíčky se skiny

Klientská část aplikace je složena z řady projektů. Části se kompilují do SWC knihoven, které se stávají součástí aplikace. Další projekty se kompilují do podoby SWF knihoven, které se nahrávají až za běhu aplikace (RSL). Pro každý modul XY musí existovat dvě části, a to část se skinem skin_XY.swf a část s implementací komponent widgets_XY.swf. Takto je do budoucna zaručeno, že bude velice snadné změnit vzhled nebo funkčnost komponent bez nutnosti zásahu do celého systému.

Aktuálně se v systému nahrává jen jeden modul. Pří skládání částí modulů se používá jedna doména, aby byly třídy skinů a widgetů dostupné v rámci celého systému. V opačném případě by nebylo možné třídy vytvářet pomocí metody reflection. Každý skin obsahuje centrální CSS, jenž obsahuje všechny vazby skinů Spark komponent a další nastavení, které CSS pro Flash poskytuje. Centrální CSS je pak kompilované do jednoho SWF souboru, který obsahuje všechny potřebné třídy a grafiku.



Obrázek 31 - SWF soubory v systému nutné pro skinování

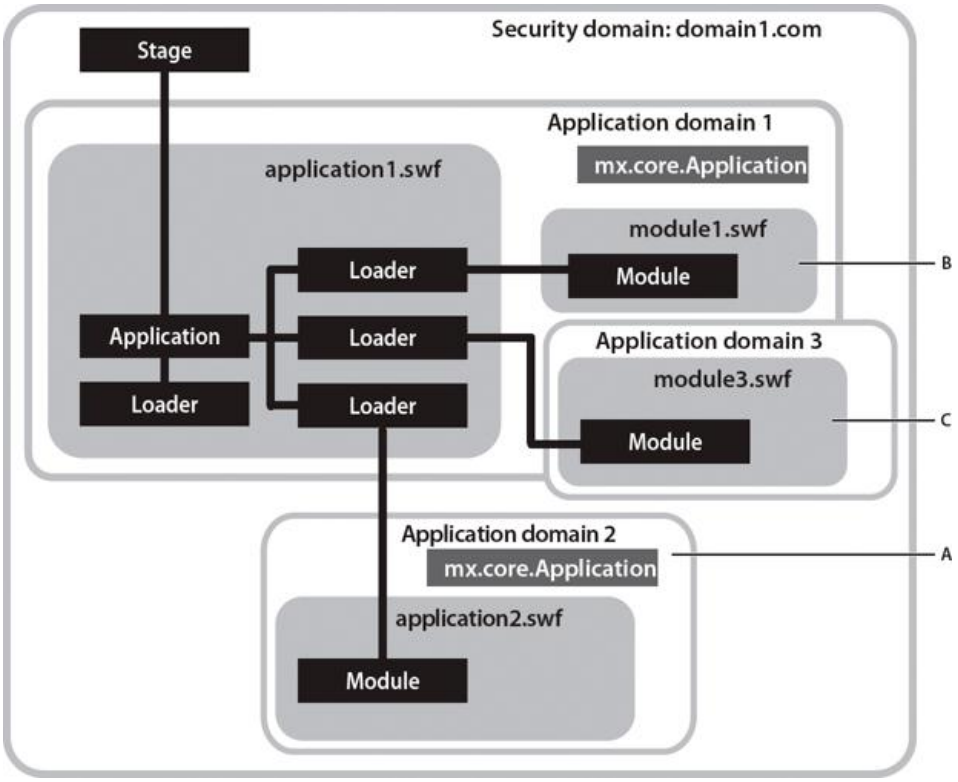
Každý SWF soubor a třída v něm patří do nějaké domény. Mezi doménami platí bezpečnostní opatření, která omezují mezidoménovou viditelnost tříd, jejich vzájemné přepisování a možnosti komunikace. Problematika domén není moc známá, ačkoli je pro správné použití externích SWF souborů klíčová. Při nahrávání externích souborů mohou nastat tři základní situace, které jsou znázorněny na následujícím obrázku.

A. Zcela nová doména. Třídy jsou od sebe oddělené. Nemůže dojít je konfliktům definic, protože definice tříd jsou oddělené a vzájemně nedostupné.

B. Třídy ze SWF souboru se přičlení do domény rodičovského souboru. Toto chování je vhodné pro RSL. Pokud se nepoužívá pro loadování RSL

automatický systém generovaný kompilátorem, tak je nutné toto nastavení zaručit ručně. Protože skiny a komponenty nejsou dopředu při kompilaci známé, je toto nutné nastavit při loadování SWF souborů obsahujících skiny a komponenty v projektu Fashionspace.

C. Doména nahrávaného SWF souboru je potomkem domény rodičovského SWF souboru. Definice všech tříd jsou vzájemně dostupné. Rodičovské definice tříd předefinovávají definice v dětském SWF.



Obrázek 32 - Práce s doménami při použití externích SWF souborů podle [31]

10. Závěr

V rámci práce byla navrhnuta kompletní architektura aplikace Fashionspace a implementována její základní kostra. V rámci práce bylo nutné nejdříve vyřešit způsob propojení klientské a serverové aplikace. Zde byl vybrán princip služeb postavených na protokolu AMF, který umožnil snadnou a rychlou výměnu dat.

V dalším kroku bylo nutné vybrat architekturu klientské aplikace. Zde byla vybrána varianta MVC, reprezentovaná frameworkem Robotlegs, který nabízí plné využití jazyka AS3 a používá další principy, jako je například dependency injection, díky kterým není zbytečně navýšené množství kódu nutného pro jeho použití.

Pro online komunikaci mezi klienty byl vybrán aplikační protokol XMPP a serverová aplikace Openfire, které umožňují zvládnout bez problému komunikaci stovky klientů a zároveň je pomocí transportního protokolu BOSH zajištěna široká dostupnost přes NAT, firewally atd.

Pro práci s videem jsou využívány služby třetích stran. Video je nejdříve nahráno na FTP server, kde čeká na konverzi do použitelného formátu. Po konverzi je umístěno na streamovací server, z kterého je umožněna jeho distribuce klientovi. Zejména v této oblasti práce se již plně projevil současný trend využívat software jako službu (SaaS). Nejen streamovaní videa, ale i konverze videa je již nabízena jako služba. Je možné si vybrat libovolného poskytovatele bez nutnosti si vše implementovat a zajišťovat provoz.

V době dokončení práce byla oficiálně vydána verze Flash Playeru 10.2, která přinesla plnou podporu hardwarově akcelerovaného videa, což umožnilo to, že aplikaci s HD videem je možné provozovat i na méně výkonných zařízeních. Pro práci s videem je použit Open Source Media Framework (OSMF), který usnadňuje práci se streamovaným videem.

V rámci práce bylo nutné integrovat celou řadu technologií. Toto bylo možné jen díky celé řadě standardů a protokolů, které jsou v současné době podporované. Právě

díky standardizaci a integraci se podařilo systém navrhnout tak, že je ho možné použít bez nutnosti vlastnit HW a administrovat vlastní aplikace.

Implementovanou kostru aplikace je do budoucna možné snadno rozšiřovat o další komponenty, nový vzhled atd. Lze ji též snadno nasadit v jakékoli jiné oblasti, než je móda. V dalším rozšíření práce by bylo možné se věnovat hlubší analýze chování uživatelů a generování statistik. Též se nabízí možnost zpracovat systém online plateb a mikro plateb, případně se zaměřit hlouběji na zabezpečení a ochranu aplikace.

Seznam použité literatury

[1] Wikipedia.org [online]. 2011 [cit. 2011-03-25]. Drupal. Dostupné z WWW: <cs.wikipedia.org/wiki/Drupal>.

[2] Interval.cz [online]. 2006 [cit. 2011-03-25]. Drupal - seznamte se. Dostupné z WWW: <interval.cz/clanky/drupal-seznamte-se/>.

[3] Adobe.com [online]. 2008 [cit. 2011-03-25]. Adobe Flash Platform ActionScript reference for RIA development. Dostupné z WWW: <www.adobe.com/devnet/actionscript/articles/atp_ria_guide.html>.

[4] Wikipedia.org [online]. 2011 [cit. 2011-03-25]. Adobe Flash. Dostupné z WWW: <en.wikipedia.org/wiki/Adobe_Flash>.

[5] Adobe.com [online]. 2006 [cit. 2011-03-25]. ActionScript 3.0 overview. Dostupné z WWW: <adobe.com/devnet/actionscript/articles/actionscript3_overview>.

[6] Adobe.com [online]. 2009 [cit. 2011-03-25]. Flash Player penetration. Dostupné z WWW: <adobe.com/products/player_census/flashplayer/version_penetration>.

[7] Adobe.com [online]. 2011 [cit. 2011-03-26]. Adobe Labs. Dostupné z WWW: <labs.adobe.com>.

[8] Wikipedia.org [online]. 2011 [cit. 2011-03-26]. Action Message Format. Dostupné z WWW: <en.wikipedia.org/wiki/Action_Message_Format>.

[9] Wikipedia.org [online]. 2011 [cit. 2011-03-26]. Open Source Media Framework. Dostupné z WWW: <en.wikipedia.org/wiki/Open_Source_Media_Framework>.

[10] Adobe.com [online]. 2011 [cit. 2011-03-26]. Stage video. Dostupné z WWW: < www.adobe.com/devnet/flashplayer/articles/stage_video.html>.

[11] Wikipedia.org [online]. 2011 [cit. 2011-03-27]. Extensible Messaging and Presence Protocol. Dostupné z WWW: <cs.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol>.

[12] Wikipedia.org [online]. 2011 [cit. 2011-03-27]. Openfire. Dostupné z WWW: <http://en.wikipedia.org/wiki/Openfire>.

[13] Inginiterealtime.org [online]. 2011 [cit. 2011-03-27]. XIFF API. Dostupné z WWW: <http://www.igniterealtime.org/projects/xiff/>.

[14] Wikipedia.org [online]. 2011 [cit. 2011-03-27]. Extensible Messaging and Presence Protocol. Dostupné z WWW: <cs.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol>.

[15] Adobe.com [online]. 2009 [cit. 2011-03-27]. Flash-based media player standards. Dostupné z WWW: <http://blogs.adobe.com/flashmedia/2009/07>.

[16] Wikipedia.org [online]. 2011 [cit. 2011-03-27]. YouTube. Dostupné z WWW: <http://en.wikipedia.org/wiki/YouTube>.

[17] Bbc.co.uk [online]. 2009 [cit. 2011-03-27]. Flash moves on to smart phones. Dostupné z WWW: <http://news.bbc.co.uk/1/hi/8287239.stm>.

[18] Tipb.com [online]. 2010 [cit. 2011-03-28]. iPhone, iPad friendly HTML 5 video penetration hits 54%. Dostupné z WWW: <tipb.com/2010/10/27/iphone-ipad-friendly-html-5-video-penetration>.

[19] Wikipedia.org [online]. 2011 [cit. 2011-03-28]. Facebook. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Facebook>.

[20] Facebook.com [online]. 2011 [cit. 2011-03-28]. Press room - Statistics. Dostupné z WWW: <http://www.facebook.com/press/info.php?statistics>.

[21] Wikipedia.org [online]. 2011 [cit. 2011-03-28]. Comet. Dostupné z WWW: <http://en.wikipedia.org/wiki/Comet_(programming)>.

[22] Facebook.com [online]. 2008 [cit. 2011-03-28]. Facebook Chat. Dostupné z WWW: <http://www.facebook.com/note.php?note_id=14218138919>.

[23] Svarovsky-tomas.com [online]. 2008 [cit. 2011-03-28]. Sproutcore and NodeJS are stars and comets. Dostupné z WWW: <http://www.svarovsky-tomas.com/nodejs-sproutcore.html>.

[24] Oracle.com [online]. 2010 [cit. 2011-03-28]. JavaOne Online Technical Sessions and Labs. Dostupné z WWW: <http://www.oracle.com/technetwork/java/index-jsp-156726.html>.

[25] CRANE, Dave; MCCARTHY, Phil. Comet and Reverse Ajax: The Next Generation Ajax 2.0. Apress, 2008

[26] MAHEMOFF, Michael. Web Remoting. Ajax Design Patterns. O'Reilly Media, 2006

[27] Bluishcoder.co.nz [online]. 2005 [cit. 2011-03-28]. More on Ajax and server push. Dostupné z WWW: <bluishcoder.co.nz/2005/11/more-on-ajax-and-server-push.html>.

[28] Wikipedia.org [online]. 2011 [cit. 2011-03-29]. Google Talk. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Google_Talk>.

[29] Ibm.com [online]. 2011 [cit. 2011-03-29]. Meet the Extensible Messaging and Presence Protocol (XMPP). Dostupné z WWW: <ibm.com/developerworks/webservices/library/x-xmppintro/index.html>.

[30] Jakpsatweb.cz [online]. 2011 [cit. 2011-03-29]. Úvod do platformy Adobe Flash. Dostupné z WWW: <http://flash.jakpsatweb.cz/adobe-flash/>.

[31] Adobe.com [online]. 2010 [cit. 2011-03-29]. Working with application domains. Dostupné z WWW: <http://help.adobe.com/en_US/as3/dev/WSd75bf4610ec9e22f43855da312214da1d8f-8000.html>.

[32] Drupal.org [online]. 2006 [cit. 2011-03-30]. Chat Room. Dostupné z WWW: <http://drupal.org/project/chatroom>.

[33] Drupal.org [online]. 2010 [cit. 2011-03-30]. DrupalChat. Dostupné z WWW: <http://drupal.org/project/drupalchat>.

[34] Drupal.org [online]. 2010 [cit. 2011-03-30]. DXMPP. Dostupné z WWW: <http://drupal.org/project/dxmpp>.

[35] Drupal.org [online]. 2011 [cit. 2011-03-30]. Drupal Chat Module Introduction. Dostupné z WWW: <http://www.123flashchat.com/drupal-chat.html>.

[36] Drupal.org [online]. 2006 [cit. 2011-03-30]. PHPfreechat. Dostupné z WWW: <http://drupal.org/project/phpfreechat>.

[37] Ignitrealtime.org [online]. 2006 [cit. 2011-03-30]. Openfire. Dostupné z WWW: <http://www.igniterealtime.org/projects/openfire/>.

[38] Wikipedia.org [online]. 2011 [cit. 2011-03-30]. Ejabberd. Dostupné z WWW: <http://en.wikipedia.org/wiki/Ejabberd>.

[39] Wikipedia.org [online]. 2011 [cit. 2011-03-30]. Jabberd14. Dostupné z WWW: <http://en.wikipedia.org/wiki/Jabberd14>.

[40] Wikipedia.org [online]. 2011 [cit. 2011-03-30]. Model view controller. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Model-view-controller>.

[41] Root.cz [online]. 2011 [cit. 2011-03-30]. Seriál MVC a další prezentační vzory. Dostupné z WWW: <http://zdrojak.root.cz/clanky/alternativy-k-mvc-a-zaverecne-poznamky/>.

[42] Puremvc.org [online]. 2006 [cit. 2011-03-30]. PureMVC. Dostupné z WWW: <http://archivemati.ca/2006/01/21/drupal-as-a-mvc-framework/

[43] Archivemati.ca [online]. 2006 [cit. 2011-03-30]. Drupal as a MVC Framework? Dostupné z WWW: <http://archivemati.ca/2006/01/21/drupal-as-a-mvc-framework/>.

[44] Flexmaster.blog.co.in [online]. 2010 [cit. 2011-03-30]. Flex Tutorials – Selecting the Right Flex Application Framework. Dostupné z WWW: <http://www.robotlegs.org>.

[45] Robotlegs.org [online]. 2011 [cit. 2011-03-30]. Robotlegs. Dostupné z WWW: <http://www.robotlegs.org>.

[46] Wikipedia.org [online]. 2011 [cit. 2011-03-30]. Vkládání závislostí. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Vkládání_závislostí>.

[47] 314bits.com [online]. 2010 [cit. 2011-03-30]. Robotlegs class diagram. Dostupné z WWW: <http://314bits.com/blog/2010/09/robotlegs-class-diagram-2/>.

[48] Enterprisedt.com [online]. 2007 [cit. 2011-03-31]. Integral FTP. Dostupné z WWW: <http://www.enterprisedt.com/products/integralftp/overview.html>.

[49] Ffmpeg.org [online]. 2007 [cit. 2011-03-31]. FFMPEG. Dostupné z WWW: <http://www.ffmpeg.org/>.

[50] Wikipedia.org [online]. 2011 [cit. 2011-03-31]. Cloud computing. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Cloud_computing>.

[51] Wikipedia.org [online]. 2011 [cit. 2011-03-31]. Software as a service. Dostupné z WWW: <http://en.wikipedia.org/wiki/Software_as_a_service>.

[52] Adobe.com [online]. 2011 [cit. 2011-03-31]. Flash Video Streaming Service. Dostupné z WWW: <http://www.adobe.com/products/flashmediaserver/fvss/>.

[53] POLZER, Jan. Drupal - Podrobný průvodce. Computer Press, 2008.

[54] TIDWELL, Travis. Flash with Drupal. Packt Publishing, 2009.

[55] WINBORN, Aaron. Drupal Multimedia. Packt Publishing, 2008.

[56] Asual.com [online]. 2011 [cit. 2011-03-27]. SWFAddress - Deep linking for Flash and Ajax. Dostupné z WWW: <http://www.asual.com/swfaddress/>.

[57] Wikipedia.org [online]. 2011 [cit. 2011-03-27]. Web 2.0. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Web_2.0>.

Seznam obrázků

Seznam použitých zkratek

AMF - Action Message Format

API - Application Programming Interface

AS3 – Action Script 3.0

BOSH - Bidirectional streams Over Synchronous HTTP

CMS - Content Management Systém

CRUD - Create, Read, Update and Delete

DI - Dependency injection

DOM - Document Object Mode

E4X - ECMAScript for XML

FXG - Flash XML Graphics

GC - Garbage collector

GPL - General Public License

GUI - Graphical User Interface

IM - Instant messaging

IoC - Inversion of Control

JID – Jabber ID

LDAP - Lightweight Directory Access Protocol

MVC - Model-view-controller

MXML - Macromedia Extensible Markup Language

NAT - Network Address Translation

OOP - Object oriented programming

OSMF - Open Source Media Framework

RFC -Request for comments

RIA - Rich Internet Application

RSL - Runtime Shared Libraries

RTC - Real time collaboration

RTMP - Real Time Messaging Protocol

SaaS - Software as a Service

SOA - Service-oriented architecture

SRS - Software Requirements Specification

SVG - Scalable Vector Graphics

SDK - Software development kit

UML - Unified Modeling Language

VO - Value Object

XML - Extensible Markup Language

XMPP - The Extensible Messaging and Presence Protocol

Obsah přiloženého CD-ROM

CD-ROM obsahuje text práce a zdrojové kódy aplikace.

- Sources – zdrojové kódy aplikace
- Sources\Drupal – moduly pro Drupal
- Sources\Flex – projekty pro Flash Builder 4
- Sources\UML – UML diagramy
- Thesis – text diplomové práce
- Readme.txt – popis struktury CD-ROM