

Ki-Wi Server 2.0

Technická dokumentace



Abstrakt

Technická dokumentace popisuje internetovou službu Ki-Wi Server ve verzi 2.0 společnosti Ki-Wi Digital s.r.o., která slouží k řízení digitálních informačních a reklamních panelů. Vývoj probíhá na platformě Java EE, proto text často odkazuje na související knihovny a technologie.

Obsah

Obsah.....	2
1 Úvod.....	6
1.1. O aplikaci.....	6
1.2. Princip komunikace.....	6
1.3. Podporované funkce.....	7
1.3.1. Média.....	7
1.3.2. Playlisty.....	7
1.3.3. Digitální panely.....	7
1.3.4. Reportování.....	7
1.3.5. Uživatelé.....	8
1.3.6. Správa organizací.....	8
1.4. Klientský software.....	8
1.4.1. Ki-Wi OS.....	8
1.4.2. Ki-Wi Player.....	9
1.4.3. Ki-Wi Kiosk.....	9
2 Architektura.....	10
2.1. Obecné schéma.....	10
2.2. Modelová vrstva (Model).....	11
2.2.1. Databáze.....	11
2.2.2. Objektově-relační mapování.....	13
2.2.3. Modelové třídy.....	13

2.2.4. DAO rozhraní.....	15
2.3. Servisní vrstva.....	16
2.3.1. Rozhraní a třídy.....	16
2.4. Řídicí vrstva (Controller).....	17
2.4.1. Kontrolér v aplikaci.....	17
2.4.2. Třídy řídicí vrstvy.....	18
2.5. Prezentační vrstva (View).....	20
2.5.1. JSP pro ovládací část.....	21
2.5.2. JSP pro systémovou část.....	21
2.5.3. JSP pro komunikační část.....	22
2.6. Utility.....	22
2.6.1. Třídy.....	22
2.6.2. Servlety.....	23
2.7. Klientský JavaScript.....	24
2.7.1. Player.....	24
2.7.2. Kiosk.....	24
3 Komunikace.....	26
3.1. Obecný princip.....	26
3.1.1. Architektura komunikace.....	26
3.1.2. Comet.....	27
3.1.3. Rozhraní REST.....	29
3.2. Komunikace s digitálními panely.....	30
3.2.1. Schéma komunikace.....	30
3.2.2. Řídicí kanál.....	31
3.2.3. Obsahový kanál.....	31
3.2.4. Logovací kanál.....	31
3.2.5. Systémový kanál.....	32
3.3. Komunikace s externími systémy.....	32
3.3.1. Enkódování médií.....	32

3.3.2. Amazon S3.....	33
3.4. Formát přenášených zpráv.....	34
3.4.1. CometD zprávy.....	34
3.4.2. Konfigurační zprávy.....	35
3.4.3. Systémové zprávy.....	37
4 Komponenty.....	38
4.1. Správa uživatelů a organizací.....	38
4.1.1. Organizační hierarchie.....	38
4.1.2. Správa uživatelů.....	39
4.2. Správa přehrávačů.....	40
4.3. Správa playlistů.....	40
4.4. Správa médií.....	41
4.4.1. Upload souborů.....	42
4.5. Bezpečnost.....	42
4.5.1. Komunikační část.....	42
4.5.2. Autentizace.....	43
4.5.3. Autorizace.....	43
4.6. Logování.....	44
4.7. Cometd.....	45
4.8. Servlety.....	46
4.9. Plánovač událostí.....	47
5 Implementace.....	49
5.1. Vývojové technologie.....	49
5.1.1. Operační systém.....	49
5.1.2. Java EE.....	49
5.1.3. Maven.....	49
5.1.4. Eclipse.....	50
5.1.5. Aplikační server.....	51
5.2. Spring framework.....	51

5.3. DWR.....	51
5.4. Amazon EC2.....	51
5.4.1. Certifikát pro HTTPS.....	51
5.5. Amazon S3.....	52
5.6. Enkódování médií.....	53
5.7. JasperReports.....	53
5.8. Quartz.....	53
5.9. JW FLV Player.....	53
6 Použité technologie.....	55

1 Úvod

V úvodní kapitole popíšeme základní vlastnosti Ki-Wi Serveru, princip komunikace s digitálními panely a podporované funkce v aktuální verzi systému. Nakonec stručně zmíníme software, který řídí digitální panely.

1.1. O aplikaci

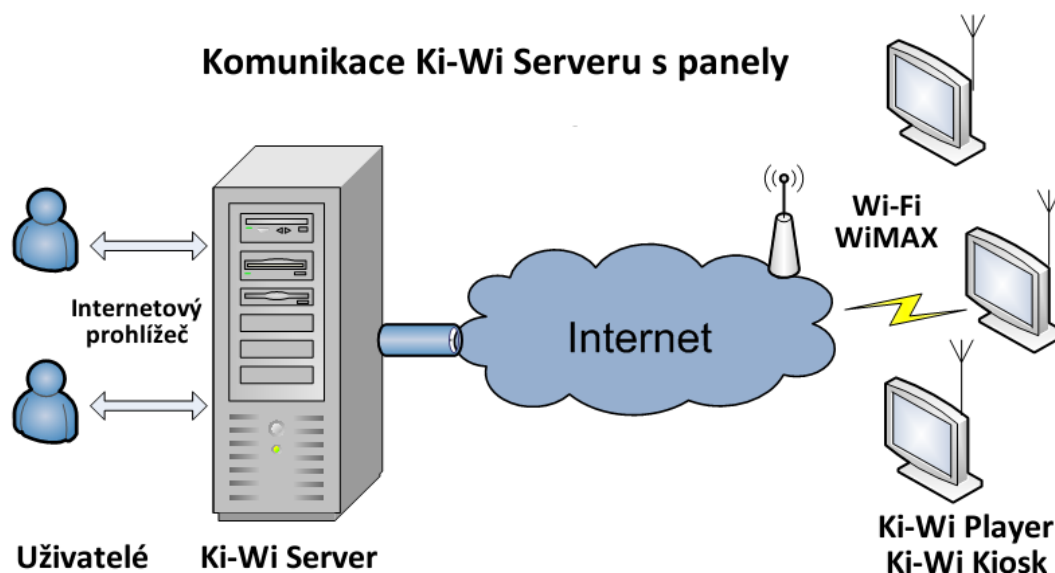
Ki-Wi server je víceuživatelská internetová služba, která umožňuje z libovolného místa připojeného k internetu vzdáleně ovládat bezdrátové digitální panely.

Služba je vhodná i pro větší společnosti (např. reklamní, franchisingové a obchodní), které mohou transparentně monitorovat a řídit displeje svých klientů (poboček).

Na bezdrátových digitálních panelech je spuštěna aplikace Ki-Wi Player nebo Ki-Wi Kiosk, která zajišťuje správné zobrazení informačního a reklamního obsahu na displeji, a interpretuje příkazy zaslané Ki-Wi Serverem.

1.2. Princip komunikace

Digitální panely se po spuštění automaticky připojí ke Ki-Wi Serveru. Uživatelé následně přes internetové rozhraní pohodlně panely ovládají.



Ovládáním je myšleno zasílání playlistů – tedy seznamů určujících, která média, a v jakém pořadí se mají na LCD displeji digitálního panelu zobrazovat. Médium je obecný název pro následující podporované typy:

- **Videa** (flv, avi, wmv, mpeg)
- **Obrázky** (jpg, png, gif)
- **Flash animace** (swf)
- **volitelně streamovaný obsah** (z TV, webových kamer)

1.3. Podporované funkce

V sekci podporované funkce uvedeme z uživatelského pohledu dosavadní dostupné vlastnosti Ki-Wi Serveru. V dalších kapitolách rozepíšeme podrobněji jejich implementaci.

1.3.1. Média

Základní část Ki-Wi Serveru. Uživatel si nahrává do centrálního úložiště své vlastní soubory, tedy obrázky, videa a animace. U všech typů médií definuje požadovanou délku zobrazení na displeji. Média lze členit hierarchicky do složek. Jednotlivé soubory musí projít schvalovacím procesem (ochrana proti nevhodnému obsahu).

1.3.2. Playlisty

Z nahraných souborů uživatel pohodlně vytvoří playlist, tedy sekvenci po sobě jdoucích médií. Playlisty lze také hierarchicky členit do složek a hromadně zasílat na všechny displeje (reklamní kampaně). Playlisty je možné tvořit ručně i automaticky (integrace s daty heterogenního systému). Z Ki-Wi serveru je dostupný také náhled médií a playlistů.

Playlisty podporují plánování s přesností na minuty, pravidelné opakování s libovolným intervalem, a nastavení týdenního i měsíčního plánu přehrávání na jednotlivých displejích.

1.3.3. Digitální panely

V Ki-Wi Serveru jsou zobrazeny všechny digitální panely patřící organizaci včetně aktuálního stavu (online, offline). Panelům lze zaslat playlist, zastavit přehrávání, vzdáleně nastavit čas vypnutí, sledovat historii přihlášení k serveru a vypsát přehraná média/playlisty.

1.3.4. Reportování

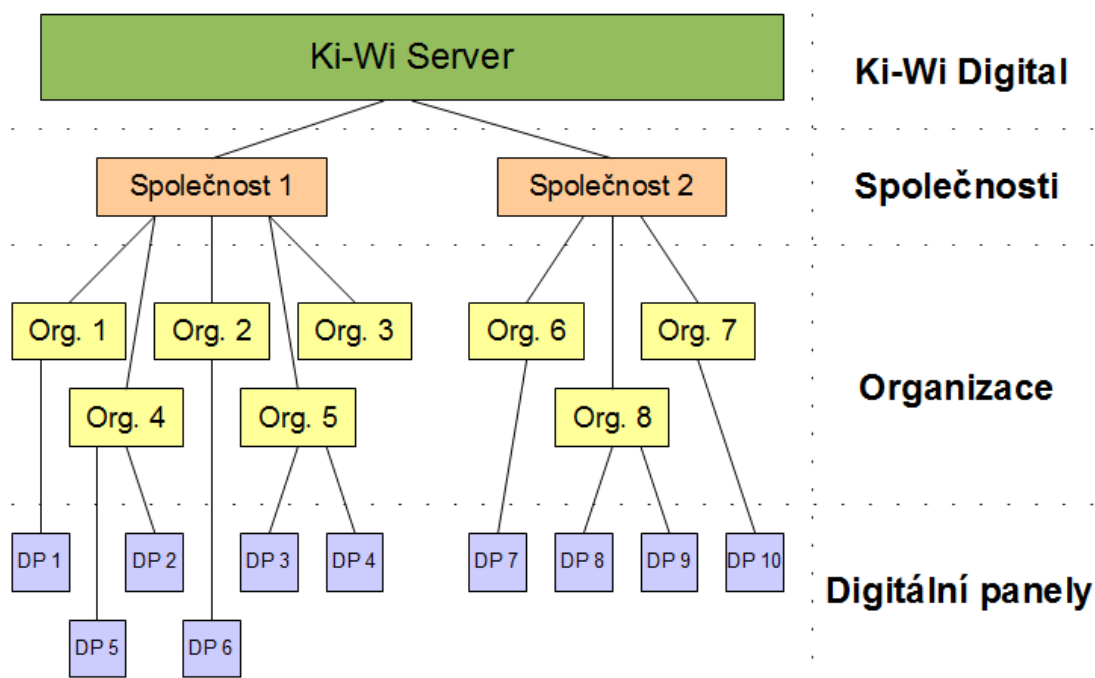
Další možností určenou pro přehled zobrazeného obsahu na displeji je denní, týdenní či měsíční reportování přehraného obsahu ve formátu textovém, PDF a HTML.

1.3.5. Uživatelé

Systém je navržen jako víceuživatelský, správce organizace může přidělovat 3 rozdílné úrovně oprávnění dalším lidem, kteří mají přístup ke Ki-Wi Serveru.

1.3.6. Správa organizací

Funkce vhodná pro větší společnosti a reklamní agentury. Administrátor společnosti získá přístup ke všem displejům svých klientů (organizací). Přístup je transparentní a může sloužit jak k monitorování jednotlivých organizací, tak k samotnému řízení jejich displejů.



1.4. Klientský software

Klientský software běží na digitálních panelech, a zajišťuje jejich správné chování. Klientské systémy podrobněji popsány v souvisejících dokumentech.

1.4.1. Ki-Wi OS

Operační systém postavený na Ubuntu 9. Slouží jako základní běhové prostředí pro aplikace Ki-Wi Player a Ki-Wi Kiosk. Grafické rozhraní postaveno na window manageru RatPoison. Ki-Wi OS

pravidelně kontroluje, zda je digitální panel připojen k serveru, v opačném případě dojde k obnovení spojení.

Více informací o operačním systému Ki-Wi OS v dokumentu s názve *[Ki-Wi-OS-Dokumentace]*

1.4.2. Ki-Wi Player

Ki-Wi Player je aplikace běžící na operačním systému Ki-Wi OS, který je instalován v digitálním panelu. Zajišťuje přehrávání informačních a reklamních smyček, které uživatel zaslal z Ki-Wi Serveru.

Na základě zaslání playlistu začne přehrávač zobrazovat jednotlivá média v pořadí a délce definované uživatelem. V první smyčce přehrávání se všechna média uloží do vnitřní dočasné paměti, čímž klesají nároky na internetovou linku.

Více informací o Ki-Wi Playeru v dokumentu *[Ki-Wi-Player-Dokumentace]*

1.4.3. Ki-Wi Kiosk

Ki-Wi Kiosk je aplikace pro digitální reklamní kiosky. Hlavní funkcí digitálního kiosku je zpřístupnění informací, většinou pomocí speciální interaktivní aplikace či internetového prohlížeče.

Ki-Wi Kiosk přidává navíc velice užitečnou vlastnost – využití okamžiku, kdy kiosk nikdo aktivně nepoužívá. V takové chvíli se přepne na obrazovce smyčka reklam či jiných informací, kterou lze pohodlně vzdáleně řídit a monitorovat pomocí internetové služby Ki-Wi Server.

Více informací o aplikaci Ki-Wi Kiosk v dokumentu s názvem *[Ki-Wi-Kiosk-Dokumentace]*

2 Architektura

V této kapitole je popsána architektura Ki-Wi Serveru s ohledem na obecný model MVCS (Model-View-Controller-Service), na němž je celý systém postaven. Od celkového schématu architektury Ki-Wi Server přejdeme podrobněji k modelové, servisní, řídicí a prezentační vrstvě. Zmíněna je také javascriptová klientská část.

2.1. Obecné schéma

Aplikace je postavena na čtyřvrstvé architektuře MVC (Model-View-Controller-Service). MVCS je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní, řídicí logiku a servisní vrstvu do čtyř nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní.

1. Model

- definuje datové objekty a stará se o přístup k samotným datům (mapování na databázi apod.)
- s databází (MySQL) komunikuje přes objektově-relační mapovací systém (Hibernate)
- objektově-relační systém používá standardní rozhraní JDBC pro přístup k databázi

2. View

- šablony, které rozhodují o tom, jaký bude vypadat výsledný výstup (HTML, XML..)
- interně reprezentovány technologií JSP (JavaServer Pages), které dovolují generovat dynamické webové stránky
- nemohou modifikovat model

3. Controller

- čeká na akce uživatele, zpracovává příchozí požadavky
- příchozí požadavky zachyceny dispatcher servletem a předány příslušnému kontroléru
- modifikuje přes servisní vrstvu model a výsledná data předává do view (JSP)

4. Service

- k zajištění menší závislosti modelu na externích systémech

- zajišťuje zachování principu jedné odpovědnosti modelové vrstvy (single responsibility)
- vrstva reprezentována rozhraním, implementaci lze změnit bez vlivu na ostatní komponenty

Prezentaci ovládacího rozhraní uživateli zprostředkovává tenký klient (internetový prohlížeč). Interakci serveru s webovým prohlížečem (uživatel) zajišťuje tzv. *Dispatcher servlet*, komunikaci digitálních panelů se serverem tzv. *Comet servlet*. Zjednodušené schéma výše popsaného znázorňuje obrázek 2.1.

2.2. Modelová vrstva (Model)

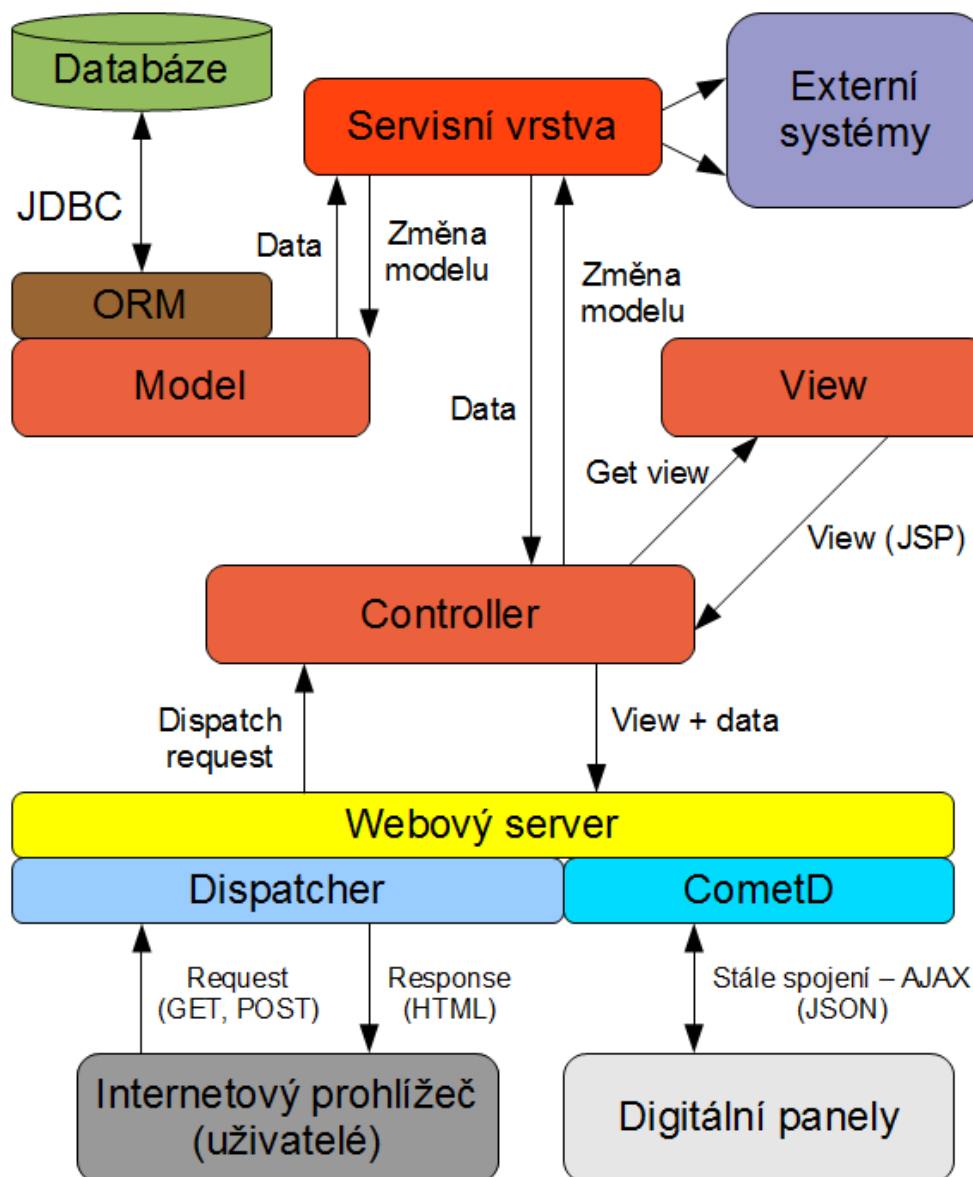
V sekci nazvané modelová vrstva popíšeme databázové schéma, objektově-relační mapování, jednotlivé modelové třídy se stručným popisem a DAO rozhraní pro přístup k těmto objektům ze servisní vrstvy.

2.2.1. Databáze

Databázové schéma je automaticky generováno z XML popisu mapování objektů na relační tabulky. K databázi (MySQL) se přistupuje pomocí rozhraní JDBC. Aplikace nepracuje přímo s tabulkami a SQL dotazy, ale vždy již získá objekt, případně kolekci objektů příslušného typu.

Jednotlivé sloupce tabulek korespondují s vlastnostmi třídy, které se na tabulky mapují. Podrobný popis vlastností je uveden v programové dokumentaci generované ze zdrojových kódů aplikací *javadoc*. Tabulky databáze kiwi:

- **activity, playlog** – monitorování aktivit uživatele a přehrávače, přehrávaného obsahu
- **corporation, organization** – společnosti vlastníci organizace, základní dělení klientů
- **directory** – adresář pro média a playlisty
- **media, track, playlist** – média, tracky s médii, a kolekce tracků určující pořadí přehrávání
- **player** – reprezentuje digitální panel
- **player_task, playlist_task, report_task** – plánované úlohy pro přehrávání a reportování
- **user** – uživatelé společnosti (administrátoři) a uživatelé organizace
- **role, user_role** – role uživatele a mapovací tabulka mezi více rolemi a uživateli



Obr. 2.1. Architektura Ki-Wi Serveru

2.2.2. Objektově-relační mapování

Objektově relační mapování slouží k tomu, aby bylo možné snadno používat relační databáze v prostředí objektově orientovaných programovacích jazyků. Vzhledem k tomu, že objektově orientovaný návrh dat není jednoznačně převoditelný na relační databáze a opačně, používají se různé formy mapování. Mapování má za účel načítat data z relační databáze a naplnit jimi příslušné datové položky objektů, případně naopak datové položky objektů ukládat do databáze.

V Ki-Wi Serveru zajišťuje mapování široce užívaný ORM systém s názvem Hibernate. Pomocí XML definujeme korespondenci vlastností třídy se sloupci databázové tabulky včetně vztahů mezi třídami. Tyto vztahy jsou automaticky reprezentovány v databázi přiřazením cizího klíče jedné tabulky k primárnímu klíči tabulky druhé.

Detailnější popis je uveden přímo v mapovacích souborech aplikace.

2.2.3. Modelové třídy

Modelové třídy jsou atomické jednotky reprezentující business logiku celé aplikace. Manipulaci s nimi zajišťují veřejné metody DAO (data access object) rozhraní. K získání potřebného objektu není proto nutné ze servisní vrstvy přistupovat přímo k objektově-relačnímu rozhraní.

Uvedeme stručně jednotlivé třídy a jejich funkci v systému, vše doplníme diagramem tříd. Popis jednotlivých vlastností a metod tříd je dostupný z programové dokumentace generované aplikací *javadoc*.

1. Základní třídy

- **BaseEntity**
 - třída společná všem objektům obsahující celočíselný identifikátor
- **NamedEntity**
 - třída společná většině objektů zděděná z BaseEntity, přidává řetězec se jménem

2. Uživatelé

- **User**
 - identifikace uživatele, který se přihlašuje do systému za organizaci nebo společnost
- **Role**
 - jedna nebo více rolí pro uživatele, které určují přístupová práva k funkcím systému

- supervisor, administrator, manager, power user, user

3. Společnosti a organizace

- **Corporation**
 - nejvyšší správní jednotka systému, vlastní administrátory a organizace
- **Organization**
 - správní jednotka systému identifikující organizaci vlastníci digitální panely
 - vlastní uživatele, přehrávače, adresáře, média a playlisty

4. Digitální panely a obsah

- **Player**
 - třída reprezentující digitální panel včetně aktuálního playlistu, reportů a úloh
- **Media**
 - třída identifikující médium pro přehrávání (obrázek, video, animace)
- **Track**
 - položka zapouzdřující médium, která se vkládá do playlistu
- **Playlist**
 - kolekce tracků s možností změny pořadí v přehrávání

5. Adresáře

- **Directory**
 - společná třída pro adresář na média, playlisty a adresáře samotné

6. Logování

- **Activity**
 - uchovává aktivity přehrávače (on/off), a uživatelů (přihlášení, zaslání playlistu)
- **Playlog**

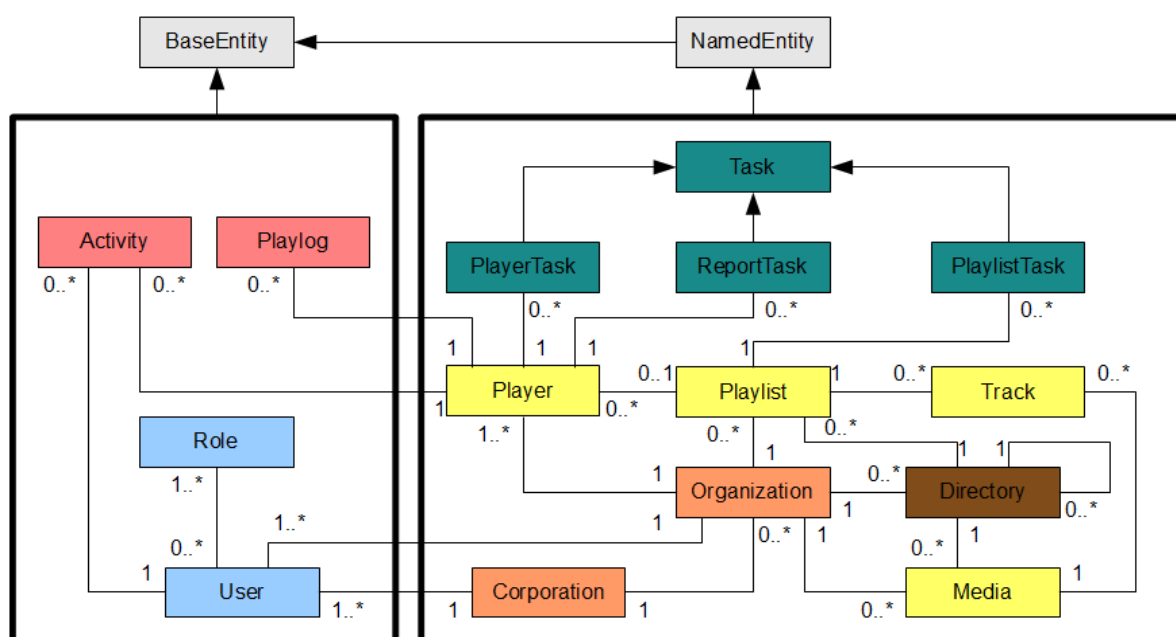
- třída pro uchování historie přehraných médií/playlistů na jednotlivém přehrávači

7. Úlohy

- **Task**
 - abstraktní třída definující úlohu v systému, která se v nastavené datum a čas provede
- **PlayerTask**
 - plánování přehrávání playlistů na digitálním panelu
- **PlaylistTask**
 - automatizovaná tvorba playlistů z externího umístění (není plně implementováno)
- **ReportTask**
 - automatické zasílání reportů o přehraných spotech na digitálním panelu

2.2.4. DAO rozhraní

Každá modelová třída obsahuje tzv. DAO rozhraní (data access object), které vystavuje metody určené pro manipulaci s objektem přes objektově-relační knihovnu, v našem případě Hibernate.



Obrázek 2.2: Diagram tříd

- **ActivityDao** – rozhraní pro přístup k aktivitám uživatelů a přehrávačů
- **PlaylogDao** – rozhraní pro přístup k informacím o přehraných médiích a playlistech
- **OrganizationDao** – DAO pro přístup k organizacím
- **CorporationDao** – DAO pro přístup ke společnostem
- **DirectoryDao** – rozhraní pro přístup k adresářům
- **UserDao** – rozhraní pro přístup k uživatelům a administrátorům
- **PlayerTaskDao** – DAO pro přístup k úlohám přehrávače
- **PlaylistTaskDao** – DAO pro přístup k úlohám pro automatickou tvorbu playlistů
- **ReportTaskDao** – DAO pro přístup k úlohám pro tvorbu reportů
- **MediaDao** – rozhraní pro přístup k médiím
- **TrackDao** – rozhraní pro přístup k trackům
- **PlayerDao** – rozhraní pro přístup k přehrávačům
- **PlaylistDao** – rozhraní pro přístup k playlistům

2.3. Servisní vrstva

Servisní vrstva slouží pro přístup k modelové vrstvě a externím systémům. Je implementována pomocí rozhraní a její metody využívá především řídicí vrstva.

2.3.1. Rozhraní a třídy

Většina rozhraní servisní vrstvy slouží jako nadstavba vrstvy modelové. Jestliže řídicí vrstva zavolá metodu pro získání určitého objektu, deleguje servisní vrstva tento požadavek na DAO rozhraní. Uvedeme tedy pouze rozhraní, která se týkají externích systémů:

EncodingManager

- rozhraní pro převod videí z různých formátů do FLV
- po nahrání videa na server volá externí službu *encoding.com*
- služba zkonvertuje video do formátu FLV a přenese do úložiště *Amazon S3*
- o dokončeném převodu uvědomí Ki-Wi Server (přes servlet)
- bližší popis této služby v kapitole Implementace

S3Manager

- rozhraní k přístupu do centrálního úložiště médií *Amazon S3*
- využívá knihovnu *jetS3t* implementující webové rozhraní REST úložiště *Amazon S3*
- po nahrání média na server dojde k zavolání metody rozhraní *S3Manager* a přenosu souboru do úložiště
- více informací o službě *Amazon S3* popsáno v kapitole Implementace

2.4. Řídicí vrstva (Controller)

Řídicí vrstva je centrálním bodem webové části aplikace. Veškeré požadavky na zobrazení stránek jsou zaslány do tříd implementujících řídicí vrstvu (kontroléry). Kontroléry následně vrátí požadovanou daty naplněnou šablonu JSP získanou z prezentační vrstvy (view). Webový server nakonec převede šablonu JSP s daty do konkrétní podoby (text, HTML, XML..).

2.4.1. Kontrolér v aplikaci

Každý kontrolér (jeho metoda) je namapován na jedinečnou URL, přičemž při přístupu na tento internetový zdroj dojde k vyvolání metody (metod), které se starají o zpracování tohoto požadavku. Výsledkem zpracování je nejčastěji vrácení požadovaného pohledu (view). Kontrolérem je tedy myšlena třída, jejíž metody implementují reakci na HTTP požadavek.

Mějme třídu *UserController*, která je namapovaná na URL zobrazující informace o uživatelích:

```
class UserController {  
  
    @Request("../users.html")  
  
    public void showUsers(ModelMap model, HttpServletRequest request) {  
  
        //ziskani dat z HTTP pozadavku (request)  
  
    }  
}
```

```
//naplneni objektu ModelMap daty pro sablonu JSP  
return "user.jsp"  
  
}  
  
}
```

Při přístupu na adresu `www.domain.com/users.html` dojde k vyvolání metody `showUsers` kontroléru `UserController`, získání dat z HTTP požadavku (parametry za URL), naplnění pomocného objektu daty z modelu a vrácení příslušné šablony (`user.jsp`), která je následně převedena webovým serverem do HTML. Každý kontrolér může obsahovat více metod, z nichž každá zpracovává o jinou URL.

Pro složitější webové stránky sloužící zejména pro zpracování formulářových dat existují pomocné třídy, které usnadňují zpracování komplexnějšího požadavku. Rodičem našeho kontroléru je poté třída řídicího frameworku, která vystavuje metody a popis, v jakém pořadí se volají. Taková třída umožňuje pomocí jedné z metod předat do JSP šablony celý objekt, který je naplněn přímo z webového formuláře. Po odeslání formuláře se vyvolají metody pro kontrolu správnosti a úplnosti parametrů, zpracování naplněného objektu, a nakonec metoda pro vrácení jiného pohledu (JSP).

V Ki-Wi Serveru a používaném MVCS frameworku Spring je takovou pomocnou třídou *SimpleFormController*. Její podrobný popis je poměrně zdouhavý, proto je vhodnější získat další informace ze zdrojových kódů Ki-Wi Serveru nebo dokumentace k Spring frameworku.

2.4.2. Třídy řídicí vrstvy

- **User**
 - `UserController`, `UserFormController`
 - informace o uživateli v organizaci a editace uživatele
 - `AdminController`, `AdminFormController`
 - informace o administrátorech společnosti a editace administrátora
- **Organization**
 - `OrganizationController`, `OrganizationEditController`
 - informace o organizacích a editace údajů o organizaci
- **Corporation**

- CorporationController, CorporationFormController
 - výpis všech společností pro supervisora systému, editace společností
- CorporationDetailsController, CorporationDetailsFormController
 - výpis detailů o společnosti administrátorům, editace společnosti
- **Player**
 - PlayerController, PlayerFormController
 - zobrazení přehrávačů v organizaci, detaily přehrávače
 - PlaylogFormController, PlayerControlFormController
 - informace o přehraných médiích a playlistech, vzdálené vypínání přehrávače
- **Playlist**
 - PlaylistController, PlaylistFormController
 - informace o playlistech organizace, editace playlistu
 - PlaylistGeneratorController, PlaylistPreviewController
 - automatické generování playlistů pro náhled a přehrávače, náhled playlistů
- **Track**
 - TrackController, TrackFormController
 - výpis tracků a editace tracku playlistu
- **Media**
 - MediaController, MediaFormController
 - výpis médií organizace, editace média
 - MediaGeneratorController, MediaPreviewController
 - automatické generování médií pro náhled, zobrazení náhledu média
- **Directory**

- MediaDirectoryFormController, PlaylistDirectoryFormController
 - editace adresářů pro média a playlisty
- **Task**
 - PlayerTaskController, PlayerTaskFormController
 - seznam úloh přehrávače, editace úlohy přehrávače
 - PlaylistTaskFormController
 - nastavení automatického generování playlistů
 - ReportTaskController, ReportTaskFormController
 - seznam reportů přehrávače, editace reportu přehrávače
- **Activity**
 - ActivityPlayerListForm, ActivityUserListForm
 - zobrazení aktivit přehrávače a uživatele
- **Ostatní**
 - CommonController
 - společný kontrolér pro více URL (změna jazyka, přihlašování, chybové stránky)
 - MessageFilter
 - filtr každého HTTP požadavku, přidává informační box, pokud kontrolér vykonal určitou činnost (např. přidání média)

2.5. Prezentační vrstva (View)

Prezentační vrstva se skládá z šablon dynamických webových stránek JSP (JavaServer Pages), které po naplnění daty převádí webový server do požadovaného výstupního formátu (text, HTML, XML). V JSP jsou používány pomocné prezentační knihovny, které se starají o následující činnosti:

- dekorování každé výstupní stránky
- kontrolu správnosti parametrů ve formulářích na klientské straně

- tagy pro implementaci bezpečnosti na webových stránkách (zobrazení příslušné části stránky na základě oprávnění přihlášeného uživatele)
- funkce pro práci s řetězci, formátování textu
- knihovny pro generování tabulek včetně naplnění daty dle zvoleného formátu
- přepočít časových informací do správného časového pásma

2.5.1. JSP pro ovládací část

Šablony pro ovládací část odpovídají ve většině případů 1:1 kontrolérům, které je vrací. Kontrolér *MediaController* vrací šablonu *mediaList.jsp*, *PlaylistController* vrací šablonu *playlistList.jsp*, atd. Seznam všech JSP pro ovládací část lze vyčíst z dokumentace ke zdrojovému kódu aplikace.

2.5.2. JSP pro systémovou část

Jedná se o šablony, které jsou využívány ve speciálních případech a některé z nich volány automaticky aplikačním frameworkem.

- **taglibs.jsp**
 - šablona vkládaná do všech ostatních šablon, zahrnuje importování prezentačních knihoven
- **messages.jsp**
 - vkládá informační box do každé ze stránek, pokud jej kontrolér *MessageFilter* nastavil
- **index.jsp**
 - úvodní stránka aplikace
- **login.jsp**
 - přihlašovací stránka aplikace
- **403.jsp**
 - při neoprávněném přístupu na určitou stránku se zobrazí hláška Access Denied
- **404.jsp**
 - přístup na neexistující stránku v aplikaci

- **error.jsp**
 - zavolána při chybě v aplikaci

2.5.3. JSP pro komunikační část

Šablony pro komunikační část slouží k nastavení klientského přehrávače playlistů a samotnému zaslání playlistů. Podrobný formát zpráv bude popsán v kapitole Komunikace.

- **playerSettingsGenerator.jsp**
 - generuje XML pro nastavení přehrávače (velikost okna, styl přehrávání, hlasitost, opakování, délka bufferu pro načítání médií, atd.)
- **playlistGenerator.jsp**
 - generuje XML s playlistem pro přehrávání (formát XSPF)
 - playlist definuje umístění média, délku, identifikaci média pro logování a jméno média

2.6. Utility

Mezi utility patří pomocné části aplikace určené pro komunikaci s externími systémy. Nejprve popíšeme používané třídy a poté vystavené webové služby (servlety)

2.6.1. Třídy

- **Scheduler**
 - soubor tříd, které se vyvolávají pravidelně ve zvoleném intervalu a vykonávají určité činnosti
 - jádro aplikace obsahuje podporu pro časové plánování akcí (*Spring Quartz*)
 - každý scheduler zjistí, které úlohy (tasky) jsou aktivní a spustí jejich činnost
 - v Ki-Wi Serveru se jedná o reportovací, plánovací a systémové úlohy
- **EmailSender**
 - třída pro odesílání emailu s reportem přehrávače
 - využívá existující implementace rozhraní *JavaMailSender*

- **PlaylogReporter**
 - utilita pro přehledné generování reportů ve zvoleném formátu (text, HTML, PDF)
 - vytvořením šablony je možné použít vlastní rozvržení reportovacího dokumentu
 - implementaci zajišťuje knihovna *JasperReports*
- **S3Utils**
 - pomocné metody pro práci s úložištěm médií na Amazon S3
- **ServerMultipartResolver**
 - nastavení progress baru stránce pro nahrávání médií do úložiště (MediaFormController)

2.6.2. Servlety

Servlet je technologie pro tvorbu webových služeb, která čekají na zvolené adrese na příchozí HTTP požadavky (request), na něž zasílají odpověď (response)

- **BayeuxStartupListener**
 - servlet pro komunikační část aplikace se vzdálenými přehrávači
 - udržuje spojení mezi přehrávači a serverem
 - podrobnější popis v kapitole Komunikace
- **EncodingServlet**
 - servlet čekající na informace o překódovaném médiu ze služby *encoding.com*
 - tyto informace (stav převodu) přiřadí příslušnému médiu v aplikaci
- **StatusServlet**
 - webová služba pro tzv. resetátor běžící na straně digitálního panelu
 - jedná se ochranu proti ztrátě spojení s displejem v případě selhání Comet části
 - servlet pravidelně odpovídá na dotazy přehrávače (online, offline)
 - dotazy obsahují také požadavky na vzdálené vypnutí, restartování, aktualizaci SW digitálního panelu a otočení displeje

- podrobnější popis v kapitole Komunikace

2.7. Klientský JavaScript

Tato sekce obsahuje popis javascriptových knihoven pro řízení a správné chování digitálních panelů včetně knihovny pro zobrazení kalendáře v ovládací části aplikace. JavaScriptové soubory se i pro přehrávače načítají ze serveru (bez uložení na klientské straně).

2.7.1. Player

Klientská část řídicího skriptu pro statické digitální panely. Obsahuje aplikaci pro přehrávání playlistů v jazyce XUL, javascriptový řídicí kód a pomocné knihovny. Přehrávač si po zapnutí stáhne tyto soubory ze serveru, naváže se Comet spojení a začne přehrávat posledně nastavený playlist. Implementace podrobněji popsána v kapitole komunikace a v dokumentaci k aplikaci Ki-Wi Player.

player.xul

- aplikace v jazyce Mozilla XUL, která se načte do programu xulrunner (standalone verze webového prohlížeče Firefox) na straně klienta. Player.xul odkazuje na řídicí javascriptový kód a knihovny.

player.js

- hlavní řídicí kód klientské aplikace. Připojí přehrávače přes technologii Comet k serveru, obnovuje přerušená spojení, zpracovává přijaté playlisty a zasílá logovací informace zpět na server.

player.swf

- široce nastavitelný a programovatelný flashový přehrávač playlistů (JW FLV Player)

Tools

- pomocné knihovny pro převod XML do JSON, pokročilý časovač a implementace fronty

Dojo

- komplexní javascriptový framework, ze kterého je pro digitální panely využívána implementace protokolu Bayeux technologie Comet

2.7.2. Kiosk

Klientská část řídicího skriptu pro interaktivní digitální panely (digitální reklamní kiosky). Obsahuje aplikaci pro přehrávání playlistů v jazyce XUL, prohlížeč webových stránek, javascriptový řídicí kód,

virtuální klávesnici a pomocné knihovny. Ki-Wi Kiosk po zapnutí zobrazí internetový prohlížeč s virtuální klávesnicí. V případě nečinnosti ze strany uživatele dojde k přepnutí na smyčku reklam, kde lze chování nastavit stejně jako u aplikace Ki-Wi Player (ve stylu spořiče obrazovky). Implementace podrobněji popsána v kapitole komunikace a v dokumentaci k aplikaci Ki-Wi Kiosk.

kiosk.xul

- aplikace zobrazující GUI kiosku (horní a spodní lišta, vyhledávací pruh, tlačítka, klávesnice) a vstupní bod do Ki-Wi Kiosk

kiosk.js

- řídicí kód pro kiosek, nastavení správného chování tlačítek, časovače pro zjišťování nečinnosti kiosku

player.js

- řídicí kód pro zobrazení smyčky reklam. Připojí kiosek přes technologii Comet k serveru, obnovuje přerušená spojení, zpracovává přijaté playlisty a zasílá logovací informace zpět na server.

keyboard.jsp

- inicializační virtuální klávesnice pro dotykové displeje

player.swf

- široce nastavitelný a programovatelný flashový přehrávač playlistů (JW FLV Player)

Tools

- pomocné knihovny pro převod XML do JSON a vykreslení virtuální klávesnice

Dojo

- komplexní javascriptový framework, ze kterého je pro digitální panely využívána implementace protokolu Bayeux technologie Comet

3 Komunikace

Komunikační část Ki-Wi Server lze rozdělit na tři části. První je komunikace s uživateli služby, tedy správná reakce na HTTP požadavky (prezentace webových stránek). Druhou částí je komunikace serveru s digitálními panely, která je navržena na technologii Comet (reverzní AJAX). Třetí částí je komunikace s externími systémy, jakými jsou služba pro enkodování videí *encoding.com* a centrální úložiště médií *Amazon S3* (využití REST rozhraní).

3.1. Obecný princip

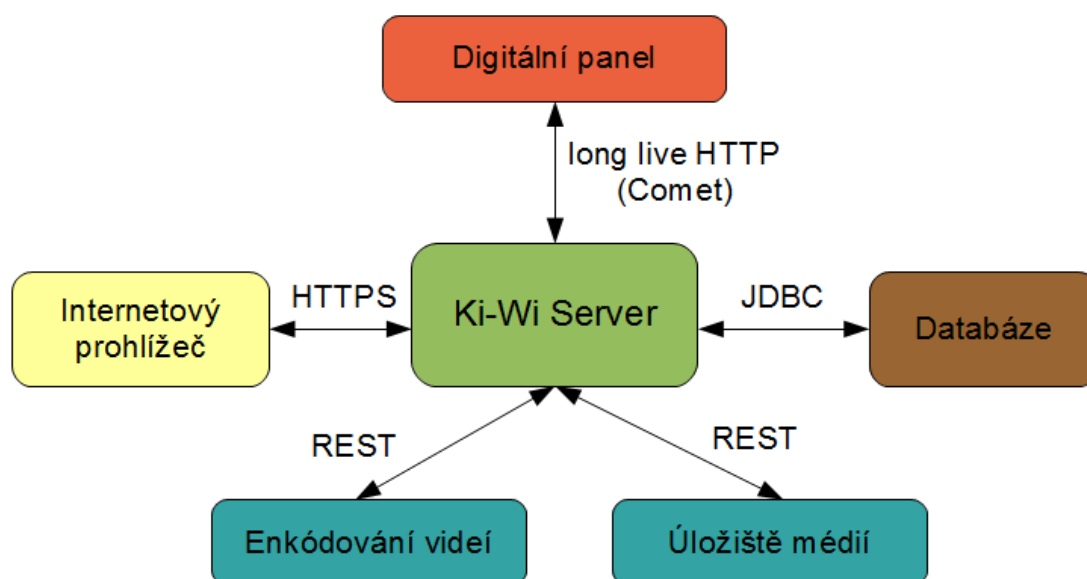
Komunikace je postavena na existujících technologiích, a to dispatcher servletu Spring frameworku pro uživatelskou část (architektura klient-server), a technologii Comet zajišťující obousměrnou komunikaci digitálních panelů a serveru. Externí systémy využívají většinou univerzální REST nebo SOAP rozhraní.

3.1.1. Architektura komunikace

Uživatel ovládá své digitální panely přes internetový prohlížeč a protokol HTTP(S). Na straně serveru očekává dispatcher servlet HTTP(S) požadavky, které předá kontroléru a následně vrátí požadovanou webovou stránku. Implementaci uživatelské část zprostředkovává Spring Framework, resp. jeho webová část (architektura klient-server).

Komunikace s digitálními panely je zajištěna pomocí technologie Comet, která bude podrobněji popsána dále. Na serveru je nasazen servlet, na který se digitální panely připojují a udržují stále spojení. Pro zajištění obousměrné komunikace se nejedná o architekturu klient-server (request-response), ale klient-klient. Server může kdykoli zaslat požadavek na digitální panel i bez prvotní reakce klienta (tzv. server push). Pro možnost komunikace server-klient je využito standardních vlastností internetového protokolu TCP/IP, a to nastavení větší délky spojení (long live request). Aplikační server musí obsahovat podporu pro takové chování, případně je nutné upravit standardní parametry (především maximální délka spojení, která je běžně omezena na kratší časový interval).

Externí systémy (webové služby) vystavují veřejné rozhraní s popisem, jak se ke službě připojit. Mezi běžné používané komunikační architektury pro webové služby patří SOAP, XMP-RPC nebo REST. Posledně jmenovanou architekturu využívají dosavadní externí systémy propojené s Ki-Wi Serverem. REST umožňuje přistupovat k datům na určitém místě pomocí standardních HTTP metod (GET, POST, DELETE, PUT). Podrobnější popis se nachází v poslední části této sekce.



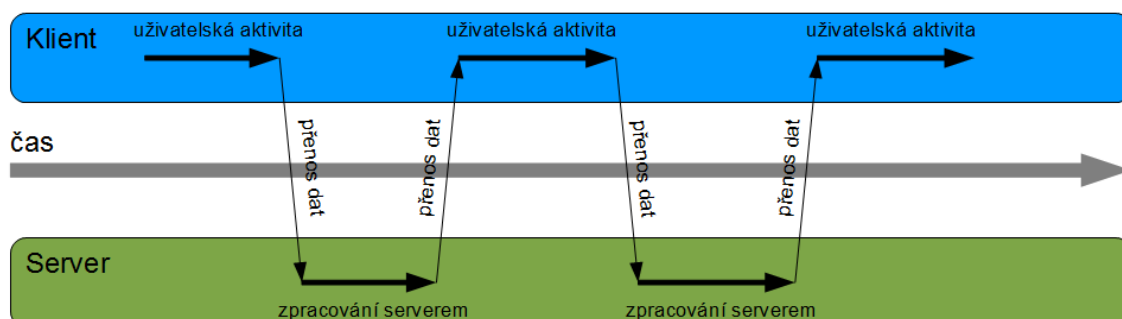
Obr. 3.1: Architektura komunikace

3.1.2. Comet

Hlavním požadavkem při vývoji ovládacího systému pro vzdálené řízení obrazek byla možnost okamžité změny přehrávaného obsahu na vzdáleném digitálním panelu. Pouze online řešení zajistí neustále spojené klienty (digitálními obrazovkami) a serverem.

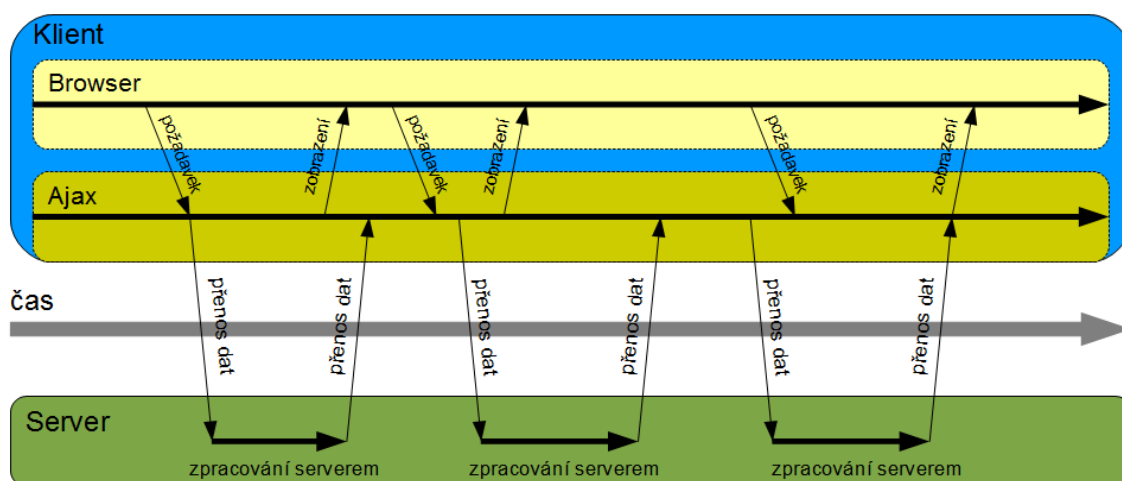
Snahou bylo se vyhnout nutnosti existence veřejných IP adres klientů, připojování k displejům přes VPN tunely, SSH, apod. Důležitá je také schopnost integrace s ovládací částí postavenou na technologii Java EE. Z těchto důvodů byla zvolena technologie Comet, která splňuje všechny požadavky kladené na komunikaci mezi vzdálenými obrazovkami a ovládacím systémem.

Technologie Comet označuje takové webové aplikace, kdy dlouhodobě otevřená HTTP spojení dovolují serveru odeslat data klientovi bez toho, aby si jej musel předtím explicitně vyžádat. Tento přístup se liší od tradičního modelu webu, ve kterém internetový prohlížeč nejprve požaduje data ze serveru. Comet využívá typicky tzv. long polling Ajax k detekci nové informace na serveru. Tento způsob komunikace je znám pod názvem Ajax Push, Reverzní Ajax, HTTP server push apod.



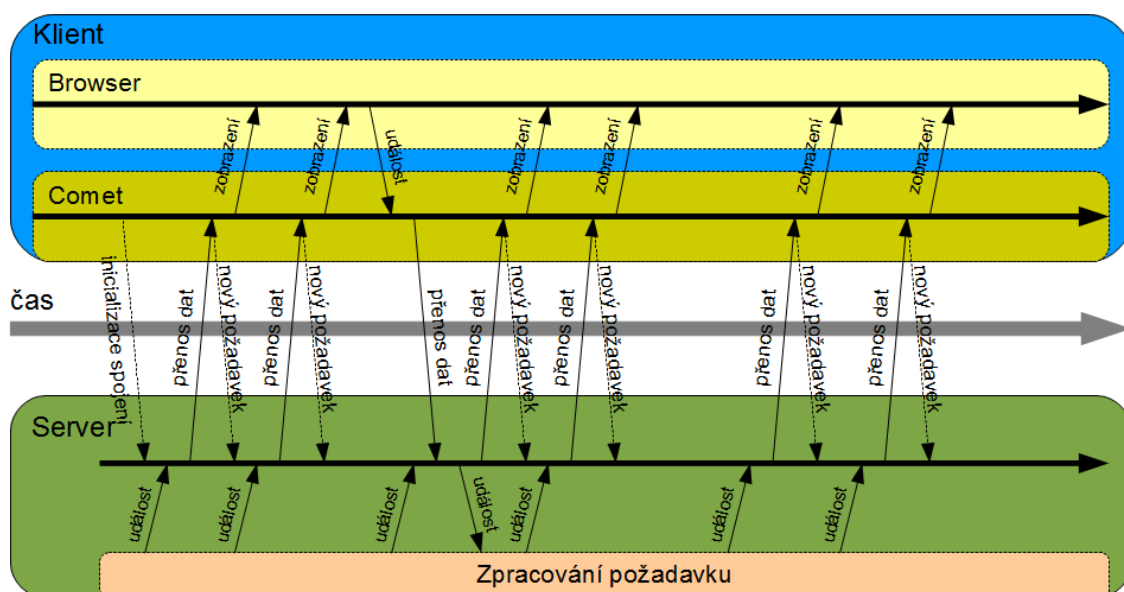
Obr. 3.2: Běžná komunikace klient-server

Long polling je varianta tradičních vyzývacích technik (pravidelné dotazování klienta na změnu stavu serveru => offline řešení). Dovoluje emulovat zaslání informace ze serveru ke klientovi. Klient získá informaci ze serveru podobným způsobem jako při běžném vyzývání. Rozdíl je v tom, že pokud nemá server žádná data připravena pro zaslání, tak místo prázdné odezvy drží otevřený požadavek, a čeká na dostupné informace. Jakmile je informace dostupná (nebo po určitém pevně daném intervalu), je zaslána ihned klientovi. Klient si poté okamžitě vyžádá další informace ze serveru, takže server má vždy k dispozici požadavek čekající na odpověď, a může doručit data okamžitě po vzniku události (např. zaslání playlistů na digitální panel).



Obr. 3.3: Komunikace klient-server přes AJAX

Samotný Comet je pouze koncept. Jednou z jeho implementací je protokol s názvem Bayeux, který v podstatě popisuje výměnu zpráv na principu publisher/subscriber. Publisher vytvoří pojmenovaný komunikační kanál, do kterého posílá data. Subscriber se na kanál zaregistruje a data z kanálu přijímá. Vztah mezi odesílatelem a příjemcem dat je běžně 1:n, kde odesílatel nemusí znát informace, kolik příjemců data odebírá. Protokol Bayeux lze využít i pro komunikaci 1:1, kde publisher odesílá data právě jednomu příjemci. K takovému účelu musíme využít speciálně pojmenovaný kanál (servisní). Prvotní spojení se serverem vytváří klient, a tedy klient nemusí být veřejně přístupný (umístění digitálního panelu za NAT).



Obr. 3.4: Komunikace klient-server přes Comet

Ki-Wi Server využívá pro udržování spojení ke vzdáleným obrazovkám projekt CometD zajišťující implementaci Bayeux protokolu v různých programovacích jazycích (JavaScript, Java, Perl, Python).

3.1.3. Rozhraní REST

REST (Representational State Transfer) je architektura rozhraní navržená pro distribuované prostředí. Na rozdíl od známějších technologií XML-RPC nebo SOAP je orientován datově, nikoli procedurálně. Webové služby běžně definují vzdálené procedury a protokol pro jejich volání. REST však určuje, jak se přistupuje k datům.

Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům (resources). Zdrojem mohou být data, stejně jako stavy aplikace. Všechny zdroje mají vlastní identifikátor URI. REST definuje čtyři základní přístupové metody, které jsou známy pod označením CRUD (Create, Retrieve,

Update, Delete). Metody jsou implementovány pomocí odpovídajících metod HTTP protokolu. GET (Retrieve), POST (Create), DELETE (Delete), PUT (Update).

3.2. Komunikace s digitálními panely

V této sekci popíšeme podrobněji princip komunikace mezi serverem a digitálními panely. Nejprve obecným principem, a následně popisem jednotlivých komunikačních kanálů.

3.2.1. Schéma komunikace

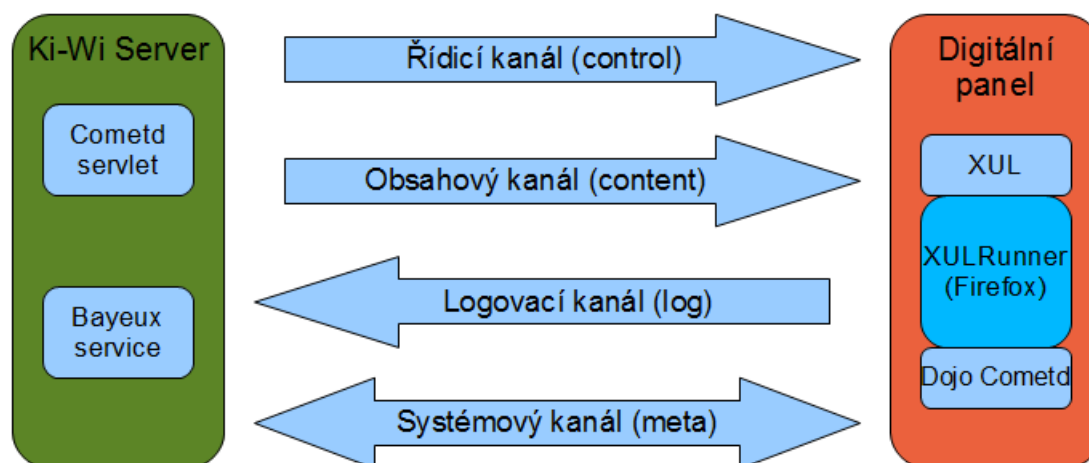
Serverová část Comet komunikace je reprezentována servletem s názvem *ContinuationCometdServlet*. Servlet je namapován na URI */cometd/**, kde čeká na příchozí požadavky na spojení, a udržuje stávající komunikaci s klienty. U servletu lze nastavit následující parametry (výpis podstatných):

- **timeout** – maximální čas, po který drží server otevřený HTTP požadavek (v ms). Po vypršení pošle klientovi žádost o opětovné připojení. Výchozí nastavení je 250 000 ms
- **interval** – timeout na straně klienta (v ms). Určuje čas, jak dlouho čeká klient, než pošle serveru znovupřipojující požadavek. Výchozí nastavení je 0.
- **maxInterval** – maximální interval, do kterého musí klient odpovědět serveru na požadavek. Pokud není odpověď v tomto limitu zaslána, dojde k odstranění klienta ze seznamu připojených. Výchozí nastavení je 30 000 ms.
- **multiFrameInterval** – timeout na straně klienta pro případ, když je z jednoho prohlížeče detekováno více spojení na server (výchozí nastavení 1500 ms).
- **filters** – filtry zpráv mezi klientem a serverem (např. odstranění zpráv s určitým klíčovým slovem)
- **loglevel** – úroveň logování Comet komunikace (0=none, 1=info, 2=debug)

Vysoká hodnota parametru *timeout* má za negativní vliv opožděnou reakci na výpadek klienta. Naopak, nízká hodnota vytěžuje více komunikační linku. Vysoký *maxInterval* je vhodný pro horší kvalitu linky a častější krátkodobé odpojování, nižší hodnota u *maxInterval* je vhodná pro kvalitnější spojení a rychlejší detekci výpadku.

Modul *BayeuxService* slouží jako prostředník mezi uživatelem a komunikační Cometd částí. Získává informace o přihlášených klientech a ukládá aktuální stav do databáze. Odstraňuje odhlášené klienty a zpracovává logovací informace o přehraných médiích a playlistech. Stará se také o zasílání řídicích informací a playlistů na digitální panely.

Klientská část Comet komunikace je spuštěna v prohlížeči xulrunner založeném na jádře Gecko společnosti Mozilla Foundation. Implementaci Cometd části zajišťuje javascriptová knihovna Dojo. Po spuštění aplikace xulrunner dojde k načtení stránky s řídicím javascriptem a připojení klienta na server. Jednotliví klienti jsou rozlišováni na základě unikátního identifikátoru (GUID).



3.2.2. Řídicí kanál

Jednosměrný kanál určený pro přihlašování klientů na server označený `/service/kiwi/control`. Modul BayeuxService zaregistruje funkci pro zachytávání klientů (subscribe). Klienti odesílají svoji identifikaci a přístupový kód na server (publish). Jako identifikátor je používána unikátní GUID hodnota. Přístupovým kódem je uvažována MAC adresa digitálního panelu (v současné době není tento parametr využíván).

3.2.3. Obsahový kanál

Jednosměrný kanál pro zasílání konfiguračních nastavení přehrávače a playlistů na digitální panel. Na straně serveru je vytvořena zpráva ve formátu JSON s adresou XML souboru s konfigurací přehrávače, který obsahuje mj. i URL k playlistu pro přehrávání. Konfigurační a playlistové XML zdroje jsou vytvářeny dynamicky na serveru. Konfigurační zdroj je předán přehrávači flashových videí, který se automaticky přizpůsobí nastavení a stáhne druhý XML soubor s playlistem. Jméno kanálu je `/service/kiwi/content`.

3.2.4. Logovací kanál

Jednosměrný kanál pro odesílání logů o přehraných médiích/playlistech na server. Obsahový kanál pojmenovaný `/service/kiwi/log` definuje, zda se mají zasílat logy jednotlivých přehraných médiích

nebo celých playlistech. Obslužné listenery přehrávače zajistí zavolání logovací funkce po každém přehraném médiu a dojde k zaslání zprávy na server (publish).

3.2.5. Systémový kanál

Kanál pro automatickou obnovu klientského připojení po výpadku a obnovení registrace do kanálů. Zaregistrováním klienta do systémového (*/cometd/meta*) kanálu získáme informace o stavu spojení. Obslužná funkce je automaticky zavolána po obnovení ztraceného spojení a můžeme opět zaregistrovat přehrávač a zajistit opětovné odebrání požadovaných kanálů. Tento způsob obnovy spojení je v případě selhání nahrazen nízkourovňovým resetováním spojení ze strany operačního systému (Ki-Wi OS).

3.3. Komunikace s externími systémy

V aktuální verzi Ki-Wi Serveru je využíváno spojení se dvěma externími systémy, a to *encoding.com* a *Amazon S3*. Stručně uvedeme princip komunikace.

3.3.1. Enkódování médií

Pro enkódování médií je využívána služba *encoding.com*. Zasláním speciální XML zprávy na enkódovací službu dojde k převodu videa do požadovaného formátu (např. FLV). Po ukončení převodu je enkódovací servlet na straně Ki-Wi Server upozorněn na dokončení úlohy a v objektu média nastaven příslušný status. Převedené video je automaticky uloženo na Amazon S3 s původním jménem a příponou FLV. Kompletní formát zprávy je uveden na adrese <http://www.encoding.com/wdocs/ApiDoc>.

- **Formát požadavku na převod:**

<query>

```
<userid>[UserID]</userid>    //přístupový kód ke službě
<userkey>[UserKey]</userkey> //přístupové heslo ke službě
<action>[Action]</action>    //požadovaná akce (přidat médium, zastavit převod..)
<mediaid>[MediaID]</mediaid> //unikátní ID média
<source>[SourceFile]</source> //URL s originálním video souborem
<notify>[NotifyURL]</notify> //URL pro notifikaci dokončení převodu
```

<format>

```
<output>[Output format]</output>    //výstupní formát média (flv)
<video_codec>[Video Codec]</video_codec> //video kodek (flv)
<bitrate>[Video bitrate]</bitrate>    //datový tok
<size>[Size]</size>                  //výstupní velikost videa
<destination>[DestFile]</destination> //cíl pro uložení média
```

</format>

</query>

- **Formát zprávy pro notifikaci převodu:**

<?xml version="1.0"?>

<result>

<mediaid>[MediaID]</mediaid> //ID média dle nastavení v XML zprávě

<source>[SourceFile]</source> //zdrojové URL originálního média

<status>[MediaStatus]</status> //stav média (Finished, Error)

<description>[ErrorDescription]</description> //popis chyby při převodu

<format>

<output>[OutputFormat]</output> //výstupní formát souboru

<destination>[DestFile]</destination> //URL k převedenému souboru

<status>[TaskStatus]</status> //stav úlohy (Finished, Error)

<description>[ErrorDescription]</description> //popis chyby převodu

<suggestion>[ErrorSuggestion]</suggestion> //vysvětlení chyby

</format>

</result>

3.3.2. Amazon S3

Komunikace se službou Amazon S3 je zprostředkována výhradně přes knihovnu jetS3t. JetS3t dává k dispozici veškeré funkce a objekty pro práci se soubory z úložiště.

V Ki-Wi Serveru je vytvořena jediná instance připojení ke službě Amazon S3 na základě přístupového klíče a hesla (*AWSCredentials*). Každá komunikace s úložištěm obsahuje právě tyto identifikační údaje.

Účet Ki-Wi na službě S3 má vytvořen tzv. bucket pro uživatelská média. Bucket reprezentuje „disk“ na S3, a jeho název je unikátní pro celou službu (např. *ki-wi.s3.amazonaws.com*). Služba Amazon S3 nepodporuje adresářovou strukturu pro ukládání souborů, pouze tzv. klíče. Klíč může mít tvar *location\$subLocation\$file.ext* nebo také *location/subLocation/file.ext*. Nutné je dodržet unikátnost klíčů. Použitím lomítek lze vizuálně simulovat adresářovou strukturu a s mírnou úpravou prezentace adresářové struktury docílit přehledného umístění souborů.

Pokud chceme nahrát soubor do S3 úložiště, vytvoříme nový objekt reprezentující S3 soubor:

```
S3Object s3object = new S3Object(repository + path + key); //unikátní klíč
```

```
s3object.setDataInputStream(inputStream); //stream s daty souboru
```

```
s3object.setContentType(contentType); //typ souboru
```

```
s3object.setContentLength(size); //velikost souboru
```

Vytvořený objekt nahrajeme do S3:

```
S3ServiceMulti s3ServiceMulti = new S3ServiceMulti(s3Service, e);  
//s3service obsahuje přístupové údaje k S3 pro účet Ki-Wi  
//objekt e je listener pro sledování stavu uploadu souboru na server  
s3ServiceMulti.putObjects(bucket, s3object); //uložení na S3 do  
bucketu
```

Úložiště S3 lze zabezpečit tak, aby z něj nešlo veřejně získat soubor. Pro přístup je třeba vytvořit speciální URL (signed URL), která obsahuje přístupový klíč na Amazon S3, a hash vytvořený z hesla k S3 a doby platnosti URL (přístup je možné časově omezit).

```
signedUrl = S3Service.createSignedGetUrl(S3Utils.DEFAULT_USER_BUCKET, key,  
cred, getExpiryDate());  
//bucket, cesta k souboru (klíč), přístupové údaje, datum platnosti)
```

Výsledná URL má poté tvar:

```
http://kiwi.s3.amazonaws.com/f/picture/3.jpgAWSAccessKeyId=3X7QJDT8ERS8M9ST  
QZ82&Expires=1293840061&Signature=lNdfhMjey7nDnuFcGvV3O6VWhjg%3D
```

3.4. Formát přenášených zpráv

Poslední sekce této kapitoly popíše formát přenášených zpráv mezi digitálními panely a serverem. CometD zprávy se týkají řízení a udržování spojení, konfigurační zprávy definují nastavení přehrávače a playlist, systémové zprávy zajišťují nízkouúrovňovou ochranu proti odpojení digitálního panelu včetně dalšího řízení na úrovni operačního systému.

3.4.1. CometD zprávy

- **Registrace klienta na server:**

```
publish('/service/kiwi/control', { //zpráva na kontrolní kanál  
  
    'id': playerId, //unikátní ID přehrávače  
  
    'playlistId' : playlistId, //ID přehrávaného playlistu  
  
    'idle' : idle, //stav přehrávání (IDLE = 1, PLAYING = 0)  
  
    'accessCode' : accessCode //přístupový kód (MAC adresa)  
  
});
```

- **Zaslání obsahu na klienta:**

```
publish('/service/kiwi/content', {  
    'player': control{STOP}, //řízení přehrávače (STOP, PLAY)  
    //URL s XML pro nastavení přehrávače  
    'config': './XMLsettingsGenerator.htmlid=129&device=GUID\\',  
    'log': 0 //úroveň logování (MEDIA=0, PLAYLIST=1, NONE=2)  
});
```

- **Zaslání logu na server:**

```
publish('/service/kiwi/log', {  
    'device': playerId, //ID přehrávače  
    'itemId': itemId, //ID přehrané položky (média,playlistu)  
    'type' : '1' //typ logování (MEDIUM=0, PLAYLIST=1)  
});
```

3.4.2. Konfigurační zprávy

- **Nastavení přehrávače:**

- nastavení pro přehrávače JW FLV Playe verze 4.3 ve formátu XML
- konfigurační soubory dynamicky generované na serveru (JSP)
- podrobné nastavení na stránce <http://developer.longtailvideo.com/trac/wiki/FlashVars>

```
<config>  
    <file>URL</file> //URL s playlistem  
    <repeat>always</repeat> //opakovat pro přehrání playlistu  
    <displayclick>none</displayclick> //zobrazit ikony pro kliknutí  
    <controlbar>none</controlbar> //navigační pruh  
    <playlist>none</playlist> //zobrazení playlistu na boční straně  
    <stretching>exactfit</stretching> //přizpůsobení médií oknu  
    <bufferlength>2</bufferlength> //délka bufferu (vteřiny)
```

```
<icons>false</icons> //ikony na přehrávači  
<resizing>true</resizing> //roztáhnutí média  
<width>400</width> //šířka přehrávače  
<height>300</height> //výška přehrávače  
<autostart>true</autostart> //automatický start  
<volume>80</volume> //hlasitost (0..100)  
  
</config>
```

- **Playlist:**

- formát playlistu XSPF

- soubor dynamicky generovaný na serveru (JSP)

```
<playlist version='1'>  
<title>Ki-Wi playlist</title>  
<info>XSPF</info>  
<tracklist>  
  <track>  
    <title>${tracku.name}</title> //název tracku  
    <info>${playlist.id}</info> //ID playlistu  
    <location>URL_S3</location> //zdroj média (z Amazon S3)  
    <meta rel="duration">media.duration</meta> //délka média  
    <meta rel="type">media.type</meta> //typ média  
    <annotation>${media.id}</annotation> //ID média  
  </track>  
  <track>  
    ....  
    ....  
  </track>
```

```
</tracklist>
```

```
</playlist>
```

3.4.3. Systémové zprávy

Systémové zprávy jsou k dispozici na adrese `/status/playerID`, kde *playerID* je unikátní identifikace klienta (přehrávače). Pravidelně spouštěný skript na straně přehrávače kontroluje parametr status. Pokud je offline, provede nízkourovňový restart spojení.

Dále se kontroluje požadovaná akce, která nelze provést z javascriptového skriptu (bezpečnostní omezení JS). Pokud klient vykoná požadovanou akci, zašle informaci na server a nastaví se stav *idle*.

- **Systémový servlet:**

```
status:status //offline,online,error
action:action // akce pro vykonání
    action:shutdown?wakeup=date_time //vypnutí klienta včetně
                                     data a času zapnutí
    action:idle //žádná akce, výchozí stav
    action:rotate?direction=[left,right] //rotace displeje
    action:update //aktualizace SW klienta
```

4 Komponenty

Tato kapitola uvede informace o každém modulu systému z hlediska významu a implementace v Ki-Wi Serveru. Od základních ovládacích součástí aplikace přejde k zabezpečení systému, způsobu autentizace, logování, nastavení servletů a implementaci Comet. Na závěr popíšeme princip plánovače událostí.

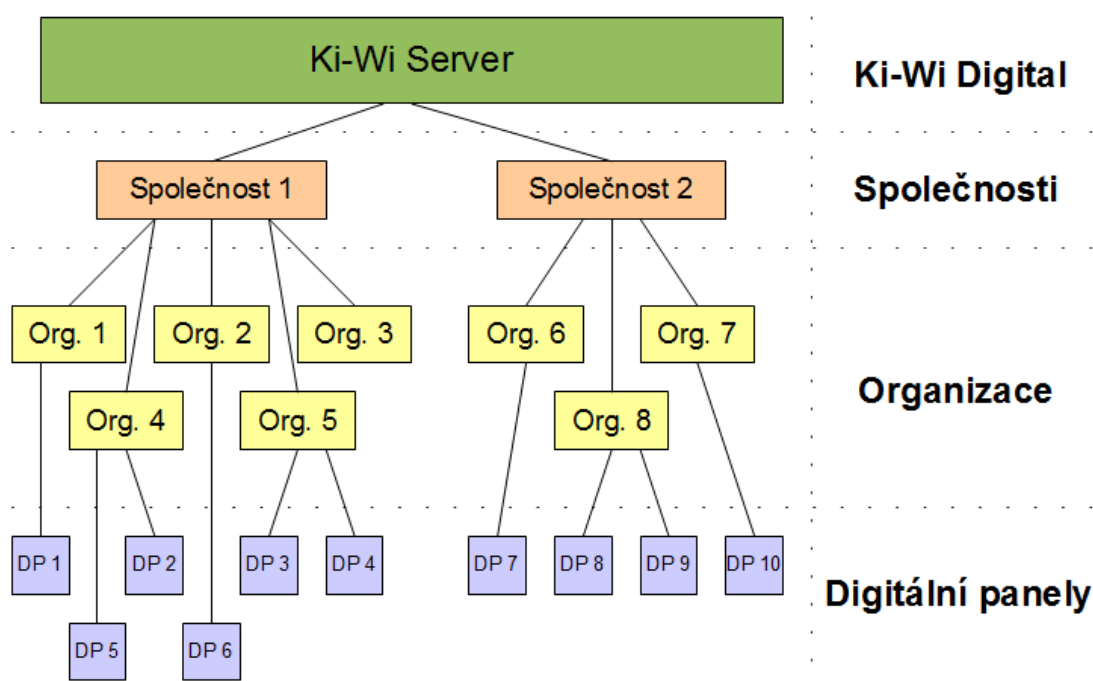
4.1. Správa uživatelů a organizací

V této sekci vysvětlíme organizační strukturu systému s rozdělením na společnosti a organizace. Dále uvedeme význam rolí uživatelů.

4.1.1. Organizační hierarchie

Na vrcholné úrovni Ki-Wi Serveru je objekt s názvem Corporation (společnost). Společnost je organizační jednotka Ki-Wi Serveru, která nevlastní žádné displeje, ale může transparentně řídit a monitorovat displeje svých klientů (organizací). Jednou ze společností je např. Ki-Wi Digital, další může být např. reklamní agentura.

Organizace je základní správní jednotka pro běžné uživatele, kteří ovládají digitální panely a tvoří playlisty z nahraných médií. Všechna média, playlisty, přehrávače a uživatelé se odkazují na organizaci, která tyto objekty vlastní.



4.1.2. Správa uživatelů

Uživatelům v Ki-Wi Serveru jsou přiřazeny tzv. role. Každá role určuje činnosti, které lze v systému provádět. Uživatele může mít rolí více, každá vyšší role zpravidla zahrnuje všechny povolené činnosti role nižší.

1. Supervisor

- uživatel s nejvyššími právy v Ki-Wi Serveru, může vytvářet a odstraňovat společnosti, organizace a přidávat digitální panely do systému. V systému existuje pouze jeden supervisor.

2. Administrator

- správce společnosti, kterému je dovoleno monitorovat více organizací a provádět činnosti jako manager.

3. Manager

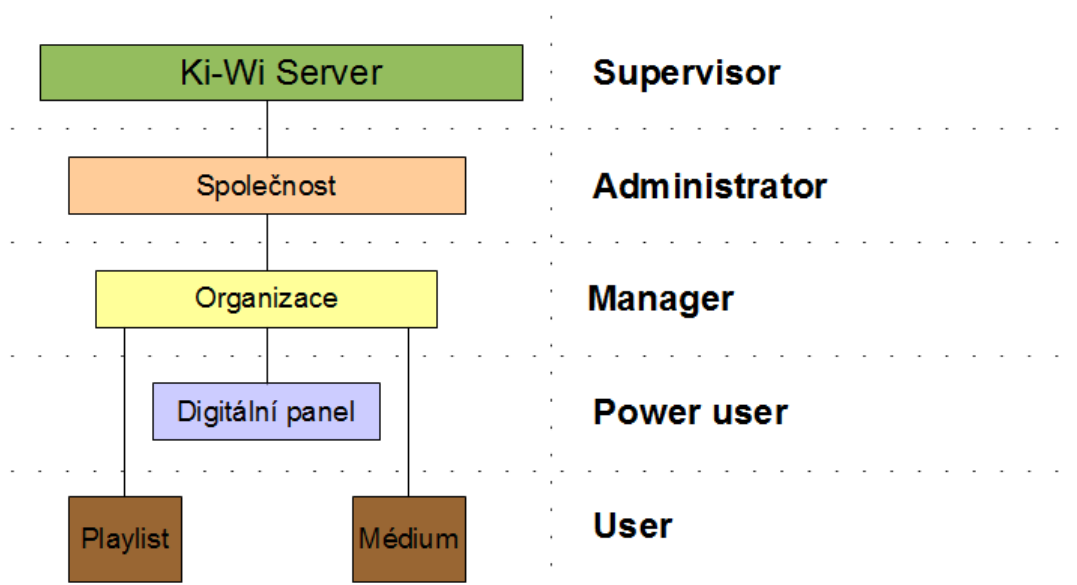
- správce jedné organizace, který není uvnitř této organizační jednotky omezen

4. Power user

- role s omezením editace organizace, plánování playlistů a úpravou informací o uživateli

5. User

- role pro přidávání médií do systému a tvorbu playlistů



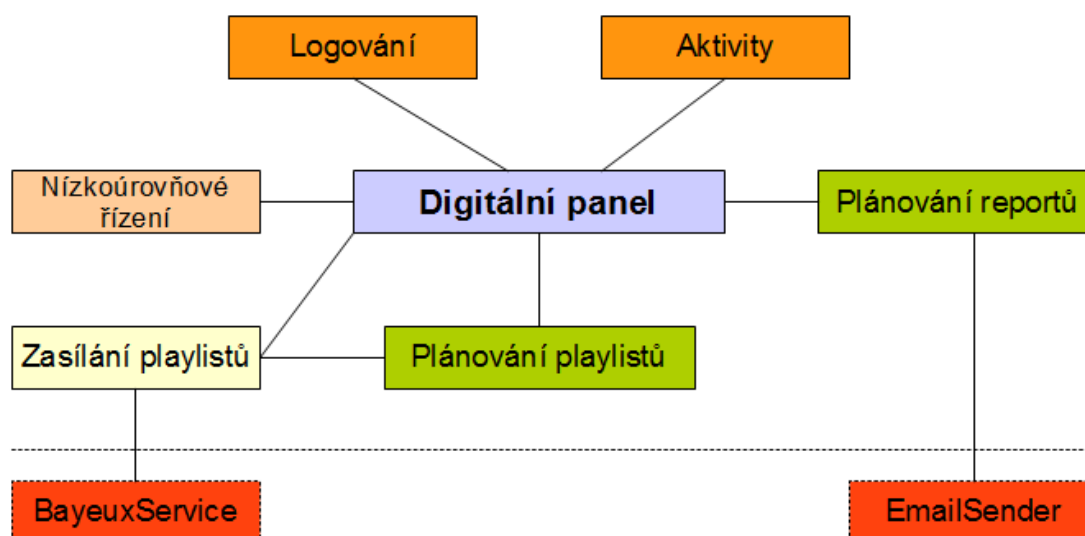
4.2. Správa přehrávačů

Správa přehrávačů je komponentou systému, která se stará o vzdálené řízení digitálních panelů. Součástí je monitorování aktivit přehrávače (přihlášení na server, odhlášení ze serveru, zaslání playlistu) a výpis logovacích informací (záznamy o přehraných médiích).

K přehrávači se také vztahuje plánování playlistů a reportů. Přes formuláře nastavíme datum a čas automatického zaslání playlistu na digitální panel, plánovač Ki-Wi Serveru v daný okamžik provede tento požadavek. Reportování zahrnuje zasílání informací o přehraných médiích (playlistech) na nastavený email v pravidelném intervalu a ve zvoleném formátu (HTML, PDF, text).

Při změně playlistu na panelu zavoláme obslužnou funkci objektu BayeuxService, která přes Comet zašle příkaz na server.

Nízkoúrovňové řízení přehrávače, jakým je vypnutí, restart, aktualizace softwaru nebo otočení obrazovky je provedeno nastavením parametrů, které zprostředkuje systémový servlet řídicí smyčce na digitálním panelu.

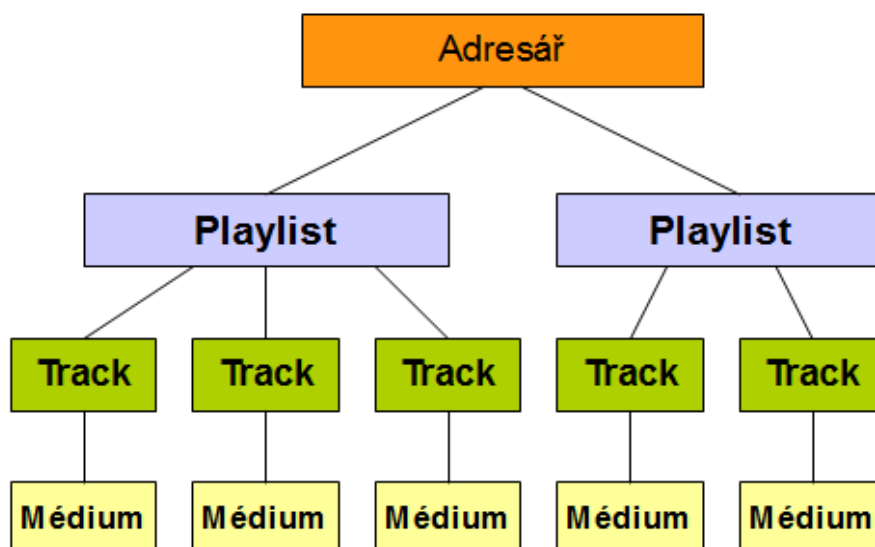


4.3. Správa playlistů

Playlist je uspořádaná kolekce tracků, která se v definovaném pořadí přehrává na digitálním panelu. Každý track může obsahovat právě jedno médium. Playlists se sdružují v adresářích, které jsou

navenek hierarchicky uspořádány ve stylu cest v operačním systému Linux (/adresar1/podadresar1/playlist).

Při zaslání playlistu na digitální panel se z vnitřní objektové reprezentace vytvoří XML soubor s popisem jednotlivých médií v definovaném pořadí.



4.4. Správa médií

Správa médií slouží pro načítání médií do Ki-Wi Serveru a následnou práci s nimi. Načtením média se vytvoří objekt třídy Media a zároveň se soubor nahraje do centrálního úložiště S3. V S3 je soubor reprezentován klíčem z prefixu organizace (např. /k/kiwi/...), cestou k souboru (/adresar1/fotky/..) a vlastním jménem souboru. Výsledný klíč má po tvar např. /k/kiwi/adresar1/fotky/picture.jpg.

Pokud se do systému nahraje videosoubor s jinou příponou než flv, dojde ke konverzi. V takovém případě se uloží soubor do S3, a zavolá se služba encoding.com s odkazem na soubor v S3. Po konverzi dojde k nahrazení klíče z tvaru /prefix/cesta/file.avi na /prefix/cesta/file.avi.flv. O této změně je informován enkódovací servlet a nový klíč k souboru je přepsán v objektu Media.

U každého média lze definovat čas přehrávání, tzn. délku zobrazení na displeji. Pokud je zadáno 0 vteřin, dojde u videa k celému přehrávání a u fotek k zastavení přehrávání na tomto médiu. Každé médium je třeba schválit managerem organizace z důvodu ochrany proti nevhodnému obsahu.

Stejně jako playlisty, i média se ukládají do adresářů. Adresáře lze libovolně pojmenovat a musí být unikátní v rámci jedné úrovně. V současné době nelze média přesouvat mezi adresáři.

4.4.1. Upload souborů

Načítání médií na server a následně do S3 může být u větších souborů poměrně zdlouhavé. Pro sledování stavu načítání média jsou do formuláře vloženy dva progress bary. První zobrazuje přenos na server včetně již přenesené velikosti a druhý přenos ze serveru do úložiště S3 včetně zbývajících času. Při lokálním spuštění Ki-Wi Serveru má význam druhý progress bar, kdy je vzdálená služba S3. U produkčního nasazení serveru na EC2 je důležitý první ukazatel, protože linka mezi EC2 a S3 je velice rychlá.

Implementace progress baru při přenosu na server je zajištěna pomocí pravidelného volání metody na straně serveru (přes DWR), která vrací informace o aktuálním průběhu uploadu (`ServerUploadInfo`). Třída `ServerFileUploadListener` je zděděna z `ProgressListener` knihovny Apache Commons, která umí zjišťovat stav uploadu z objektu `MultiPartFile` (objekt uchovávající informace o nahrávaném souboru).

Progress bar přenosu souboru na S3 je aktualizován třídou `ThreadWatcher` knihovny `jetS3t`, která monitoruje přenos souboru do úložiště. Pro přenos je třeba použít vícevláknou implementaci REST rozhraní, a to třídu `S3ServiceMulti`. Aktualizace ve webovém prohlížeči je opět zajištěna asynchronně pomocí DWR.

4.5. Bezpečnost

Bezpečnost v aplikaci je třeba řešit na dvou místech. Prvním je bezpečnost komunikační části aplikace, tedy zabezpečení pomocí přenosového protokolu HTTPS a autentizace. Druhou částí je bezpečnost při práci s uživatelským rozhraním (autorizace). Aplikace musí povolit uživateli přístup pouze na ta místa, ke kterým má oprávnění.

Autentizace a autorizace je implementována pomocí části Spring frameworku s názvem Spring Security. Tato komplexní knihovna dovoluje systém široce zabezpečit v závislosti na potřebě daného systému. Veškerá nastavení jsou uložena v souboru `security.xml`.

4.5.1. Komunikační část

Komunikace s Ki-Wi Serverem je možná jak přes protokol HTTP, tak HTTPS. Doporučená je druhá možnost, první zůstává prozatím podporovaná z technických důvodů. Spring Security dokáže pro vybrané internetové stránky vnutit použití protokol HTTPS, např. pro přihlašování do aplikace.

4.5.2. Autentizace

Autentizací je myšleno ověření identity uživatele ve vztahu ke Ki-Wi Serveru. Po odeslání přihlašovací dialogu ověří objekt třídy `UserDetailsService` existenci uživatele v databázi a správnost hesla. Pokud jsou údaje správné, vytvoří se v session uživatele objekt `UserDetails`, který obsahuje veškeré užitečné údaje pro pozdější práci se systémem (jméno a příjmení uživatele, prefix úložiště pro média, časová zóna, atd.).

Při přístupu ke každému webovému zdroji je kontrolována platnost těchto přihlašovacích údajů (tzv. credentials). Přihlášení lze uchovat i při zavření internetového prohlížeče zaškrtnutím volby `remember-me`, kdy dojde k uložení cookie do klientského prohlížeče. Počet otevřených session pro uživatele není omezen.

Hesla jsou v databázi ukládána jako SHA-1 hash hesla a dalšího řetězce (salt). V následující verzi bude implementována ochrana proti vícenásobnému přihlašování s dočasným zamezením přístupu.

Autentizace digitálních panelů je v současné době prováděna pomocí unikátního identifikátoru (GUID) a systém je připraven na zasílání dalšího údaje – např. MAC adresa síťové karty na základní desce digitálního panelu.

4.5.3. Autorizace

Autorizace určuje, k jakým zdrojům může přihlášený uživatel v závislosti na přiřazených rolích přistupovat. V Ki-Wi Serveru provádíme tři typy autorizace:

1. přístup k internetovým stránkám
2. volání metod
3. kontrola ruční změny URL

Přístup k internetovým stránkám je nastaven deklarativně anotací v souboru `security.xml`. Pro přístup k internetové stránce kromě `login.jsp` musí být uživatel vždy přihlášen (resp. mít nějakou roli). Dále je pro vybrané webové stránky nastaven přístup pouze pro určité role. Následující deklarací se omezení přístup k výpisu společnosti pouze pro supervisory a administrátora:

```
<intercept-url pattern="/**/corporation.html*" access="ROLE_ADMIN, ROLE_SUPERVISOR" />
```

V Ki-Wi Serveru je také omezen přístup k některým servisním metodám pro uživatele. Jedná se o doplňující/alternativní způsob, jak omezit funkcionalitu pro vybrané role.

Deklarativní omezení nám nezajistí způsob, jak ručně měnit parametry URL a tím získat nežádoucí přístup k údajům např. jiné organizace. Z tohoto důvodu jsou ve webových kontrolérech části kódu,

který zjistí, zda právě přihlášený uživatel má přístup např. k přehrávači s určitým ID. V opačném případě je vyvolána výjimka 404 – Access Denied.

4.6. Logování

V Ki-Wi Serveru se v současné době zaznamenávají vybrané činnosti prováděné uživatelem. Implementace je provedena pomocí tzv. aspektů. Aspektově orientované programování je vhodné pro funkcionalitu, která je překrývá všechny části systému. Typickým příkladem je právě logování, které zasahuje do všech částí programu.

Aspektově orientované programování je postaveno na několika jednoduchých myšlenkách úpravy chování programů v přesně definovaných místech (tzv. joinpoint) – při spouštění metod, přístupu k atributům třídy nebo při vyhození výjimky. Úprava chování spočívá ve vložení nového kódu, označovaného jako advice, před, po nebo okolo joinpoint. Advice může také měnit vstupní parametry metod, nebo dokonce vůbec zabránit jejich spuštění a sama poskytnout návratovou hodnotu. Aby se jednotlivé advice aplikovaly na správná místa, je třeba vytvořit speciální výraz (nazývaný pointcut), který identifikuje konkrétní joinpoint; a kromě jejich vyjmenování může obsahovat i složitější podmínky, jak bylo naznačeno již v předchozím odstavci. Skupinu advice a jim odpovídajících pointcut, které zajišťují konkrétní funkčnost, pak shrnujeme do aspektu jako základního modulu v AOP.

V praxi to znamená vytvoření třídy s anotací `@Aspect`. V třídě vytvoříme metody, před jejíž hlavičku přidáme pointcut (odkaz na metodu, jejíž provedení chceme zaznamenat). Pointcut je vložen do anotace, která určuje vyvolání kódů metody (Before, After, AfterReturning, AfterThrowing, atd.). Každá metoda má parametr typu `JoinPoint`, který nese vstupní, případně výstupní parametry, může vstoupit do vykonávání kódu sledované metody.

Následující kód je vyvolán po návratu z metody `DisabledAccountManagerImpl.disableUserAccount` a uloží logovací záznam do databáze.

```
@After("execution(* cz.kiwi.server.service.impl.DisabledAccountManagerImpl.disableUserAccount(..))")
```

```
public void logAfterDisabledAccount(JoinPoint joinPoint) {  
    String userName = (String) joinPoint.getArgs()[0];  
    User user = userManager.getUserByUsername(userName);  
    Activity activity = new Activity(ACTIVITY_TYPES.ACTIVITY_USER_ACCOUNT_LOCKED, new Date(),  
    "Account has been temporarily locked");  
    activity.setUser(user);  
    activityManager.saveActivity(activity);  
}
```

4.7. Cometd

Princip Cometd byl podrobně popsán ve třetí kapitole. Nyní si vysvětlíme, jak je Cometd integrován do Ki-Wi Serveru. Cometd se skládá ze dvou částí:

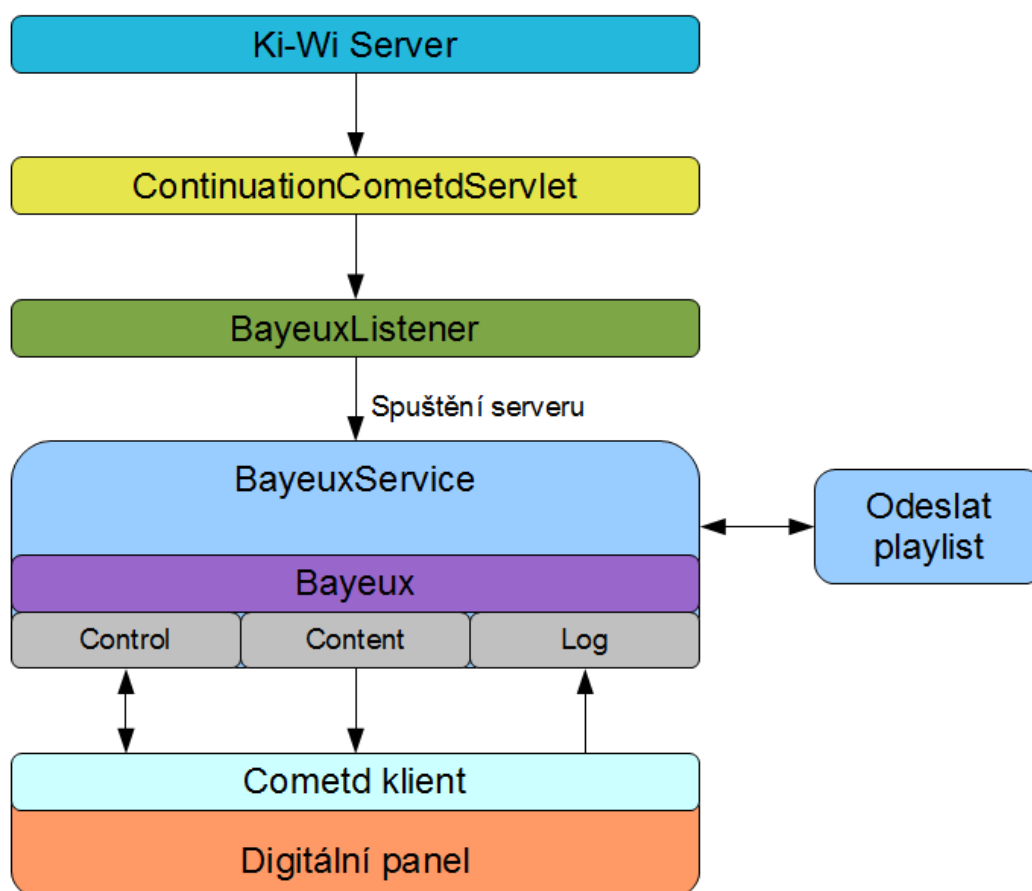
1. Cometd servlet na straně serveru (java)
2. Cometd klient na straně digitálního panelu (javascript)

Cometd servlet čeká na příchozí spojení od digitálního panelu (Cometd klienta), které následně automaticky udržuje v aktivním stavu. Digitální panel se na serveru identifikuje unikátním kódem (GUID). Z důvodu ochrany proti přihlášení cizího klienta se stejným GUID bude nutné zasílat společně s GUID např. MAC adresu síťové karty digitálního panelu (jejíž hodnota se uloží předem na server pro kontrolu).

Cometd spojení následně automaticky získá ze serveru náhodně vygenerovaný řetězec, který slouží pro vnitřní rozlišení klientů v rámci Cometd komunikace. Tento řetězec uložíme do databáze, a používá se při každé komunikaci s digitálním panelem ze strany serveru (změna playlistu). Po opětovném připojení Cometd klienta dojde k přepsání dříve uloženého náhodného řetězce.

Po spuštění Ki-Wi Serveru dojde k namapování `ContinuationCometdServletu` na zdroj `/cometd/*`. V servletovém kontextu se uloží instance objektu `Bayeux` implementujícího protokol Comet. `BayeuxListener` je servlet listener, který vytvoří jedinou instanci služby `BayeuxKiwiService` a předá ji objekt `Bayeux` získaný ze servlet kontextu. Celý Ki-Wi Server komunikuje následně s digitálními panely právě přes objekt `BayeuxKiwiService`. Stejně tak vstupním bodem do Ki-Wi Serveru pro digitální panely je objekt `BayeuxKiwiService`.

`BayeuxKiwiService` po spuštění vytvoří kontrolní a logovací kanál, na kterém naslouchá zprávám od klientů. Problematická je současná implementace třídy `BayeuxService` (rodičovská třída `BayeuxKiwiService`), neumožňuje jednoduchou integraci do Ki-Wi Serveru přes dependency injection a instanci `BayeuxKiwiService` proto musíme uchovávat ručně ve `WebApplicationContext`.



4.8. Servlety

Ki-Wi Server v současné době vystavuje pět servletů (webové služby).

1. Dispatcher (DispatcherServlet)

- systémový servlet Spring Frameworku zachytávající všechny požadavky na prezetační část Ki-Wi Serveru (HTML stránky) a vracejí příslušný pohled (view reprezentovaný jsp šablonou).
- namapován na všechna URL končící „.html“

2. DWR (DwrSpringServlet)

- servlet pro asynchronní komunikaci se serverem (AJAX)
- namapován na URL /dwr/*

3. Cometd (ContinuationCometdServlet)

- Cometd komunikace s digitálními panely
- namapován na URL /cometd/*

4. Encoding (EncodingServlet)

- servlet pro získání informací o stavu převodu videa ze služby encoding.com
- namapován na URL /encoding/*

5. Status (StatusServlet)

- servlet, který využívají digitální panely pro kontrolu spojení v případě nestandardního výpadku cometd
- namapován na URL /status/*

4.9. Plánovač událostí

Server obvykle odpovídá na zaslané požadavek od klienta, tzn. naslouchá na určitém portu bez dalších činností. Pokud požadujeme automatické vykonávání určitých činností bez zásahu klienta, potřebujeme aktivovat takový mechanismus.

Spring Framework poskytuje podporu pro automatické spouštění úloh (Quartz). Hlavní plánovací třídě předáme reference na objekty definující intervaly určující frekvenci aktivace činností. Spuštěním úlohy je myšleno zavolání předem určené metody, která vykoná požadovanou činnost.

1. SchedulerFactoryBean

- základní třída knihovny Quartz, které předáme reference na objekty obsahující parametry spouštění úlohy (CronTriggerBean)

2. CronTriggerBean

- třída, která definuje interval spouštění a referenci na spouštěcí objekt
- zápis je ve formě CRON výrazu (např. 10 * * * * ? , tzn. spuštění každých 10 minut)

3. MethodInvokingJobDetailFactoryBean

- třída definující objekt a metodu, která se má spustit

Ki-Wi Server spouští následující automatické činnosti:

- **ActivitySchedule**
 - mazání aktivit starších 45 dnů
- **PlayerSchedule**
 - spustí úlohy pro automatické odesílání playlistů na přehrávač (kontrola aktivních objektů PlayerTask)
- **PlaylistSchedule**
 - tvorba playlistů z nastaveného zdroje – není zcela implementováno
- **PlaylogSchedule**
 - mazání playlogů starších 45 dnů
- **ReportSchedule**
 - spustí úlohy pro automatické odesílání reportů na email (kontrola aktivních objektů ReportTask)

Každá úloha je tvořena abstraktní třídou Task. Ta definuje čas spuštění a konce, aktivitu úlohy, parametry spuštění, interval opakování, atd.

1. PlayerTask

- třída pro vytvoření úlohy přehrávače uchovávající playlist a datum zaslání
- každý přehrávač obsahuje kolekci objektů třídy PlayerTask

2. ReportTask

- třída pro odeslání reportu z playlogů daného přehrávače
- uchovává email příjemce, frekvenci odesílání a formát dokumentu
- každý přehrávač obsahuje kolekci těchto úloh

3. PlaylistTask

- úloha pro vytvoření playlistu ze zadaného zdroje
- tvoří především kolekci internetových stránek

5 Implementace

Implementace probíhala inkrementálně, s postupným přidáváním jednotlivých funkcí. Popíšeme si prostředí implementace, použité technologie, externí služby a knihovny.

5.1. Vývojové technologie

Při vývoji byl kladen důraz na přenositelné a opensource technologie. Z důvodů předpokládané komplexnosti aplikace je hlavním vývojovou platformou Java EE. Správů balíčků a buildovací proces zjednodušuje software s názvem Maven. Vývojovým prostředím je IDE Eclipse.

5.1.1. Operační systém

Pro vývoj ani produkční nasazení nezáleží na použitém operačním systému. Pro spuštění Ki-Wi Serveru postačuje nainstalované běhové prostředí JRE. V produkční verzi je Ki-Wi Server nasazen na operačním systému Ubuntu 9. Spuštění produkčního serveru je řízeno automatickými skripty vytvořenými přes službu RightScale (<https://my.rightscale.com/>).

5.1.2. Java EE

Java Platform, Enterprise Edition (Java EE) je součást platformy Java určená pro vývoj a provoz podnikových aplikací a informačních systémů. Ki-Wi Server je od počátku vyvíjen na této platformě, která dovoluje téměř neomezenou rozšiřitelnost, je široce využívána po celém světě a existuje pro ni velké množství knihoven.

5.1.3. Maven

Maven je softwarový nástroj pro správu a sestavování projektů vytvořených v jazyce Java. Funguje podobně jako aplikace Ant. Rozdíl je v tom, že sestavovacímu programu Ant sdělíme „jak“ danou činnost udělat, zatímco nástroji Maven předepíšeme „co“ udělat.

Pro sestavení projektu využívá Maven konstrukci nazvanou Project Object Model (POM), která zahrnuje informace o softwarovém projektu, závislostech, externích modulech a buildovacím pořadí.

Maven dovoluje aplikační server s projektem spustit (využívané ve vývoji), v produkční verzi však nasazen sestavený projekt v souboru WAR (WAR vygeneruje také Maven).

Níže je ukázka konfiguračního POM souboru z Ki-Wi Serveru:

```
<project ...>
  <!-- obecné informace o projektu -->
  <modelVersion>4.0.0</modelVersion>
  <groupId>cz.kiwi</groupId>
  <artifactId>kiwi-server</artifactId>
  <packaging>war</packaging>
```

```
<name>Ki-Wi Server</name>
<version>0.0.0-SNAPSHOT</version>

<build> <!-- informace pro sestavení a spuštění projektu -->
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>2.0.2</version>
      <configuration>
        <source>1.5</source>
        <target>1.5</target>
        <debug>false</debug>
      </configuration>
    </plugin>
    ...
  </plugins>

  <resources> <!-- zdroje k projektu -->
    ...
  </resources>
</build>

<repositories> <!-- úložiště knihoven -->
  <repository>
    <id>java.net</id>
    <url>http://download.java.net/maven/2</url>
  </repository>
</repositories>

<dependencies> <!-- závislosti, automaticky stahované v definované verzi -->
  <dependency>
    <groupId>org.mortbay.jetty</groupId>
    <artifactId>cometd-client</artifactId>
    <version>6.1.20</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>2.0.4</version>
  </dependency>
  ...
</dependencies>
</project>
```

5.1.4. Eclipse

Eclipse je vývojové prostředí určené pro programování v různých jazycích s podporou Java EE. Výhodou je automatické generování projektu z maven pluginu a spolupráce s SVN.

5.1.5. Aplikační server

Ki-Wi Server běží na kompaktním aplikačním serveru Jetty, který mj. podporuje technologii Cometd, přes kterou komunikují digitální panely. Jetty je v provozním prostředí spouštěn přes standalone verzi, ve vývojovém pomoci pluginu maven-jetty.

5.2. Spring framework

Spring framework je aplikační rámec pro vytváření nejnáročnějších informačních systémů v prostředí Java EE. Obsahuje v základní verzi mnoho podpůrných knihoven a velice usnadňuje tvorbu rozsáhlých webových aplikací.

5.3. DWR

DWR je knihovna pro vzdálené volání objektů ze strany klienta na serveru (AJAX). Pro vystavený objekt a jeho metody vytvoří javascriptový kód, který vložíme na webové stránky. Poté můžeme přímo z prohlížeče volat funkce pro modifikaci objektů a obnovit data na stránce bez nutnosti znovuzobrazení celé internetové stránky.

V Ki-Wi Serveru je DWR použit pro dynamickou aktualizaci progress baru při uploadu souborů. Třída `ServerFileUploadListener` zobrazuje průběh nahrávání souboru z prohlížeče na server, třída `S3FileUploadListener` průběh nahrávání ze serveru do centrálního úložiště Amazon S3.

5.4. Amazon EC2

Amazon Elastic Compute Cloud (EC2) nabízí zákazníkům pronájem virtuálních serverů. Po vytvoření serverové instance o požadovaném výkonu máme k dispozici škálovatelný výpočetní uzel. V současné době se servery nachází v USA a Evropě.

Ki-Wi Server běží nyní na malé instanci EC2 (small) zahrnující 1.7 GB paměti, 160 GB diskový prostor a jedno výpočetní jádro o frekvenci 1.2 GHz.

Hostovaný operační systém je spravován pomocí SSH konzole a podpůrné externí služby RightScale, která dovoluje vytvořit skripty pro automatické spuštění a nastavení serveru.

5.4.1. Certifikát pro HTTPS

Certifikát na HTTPS je generován společností godaddy (<http://www.godaddy.com/>), jejíž kořenové certifikáty obsahují ve výchozím stavu všechny běžně používané internetové prohlížeče. Certifikát platí jen pro subdoménu `server.ki-wi.cz` a jeho platnost je 12 měsíců.

5.5. Amazon S3

Amazon S3 je další služba společnosti Amazon, která nabízí prakticky neomezené webové úložiště. S3 má některé specifické vlastnosti, které jej odlišují od běžného pevného disku. Jednak je snadno škálovatelné – volný prostor je prakticky neomezený. Dále je trvalé – data jsou fyzicky ukládána redundantně na různých místech, takže ani výpadek, ani poškození HW by neměl data nijak ohrozit.

S3 tedy rozhodně není vhodné místo pro ukládání pracovních souborů, do nichž se neustále zapisuje či které se často mění, mažou, vytváří, ale naopak vhodné právě pro data s delší trvanlivostí.

Architektura úložiště se skládá především z následujících částí:

Object

- základní jednotka uložené informace – skládá se z vlastních dat a metadat. Data mohou mít velikost od jednoho bajtu až po 5GB. Metadata jsou tvořena dvojicemi klíč/hodnota. Mezi nimi jsou některé standardní údaje, známé z HTTP protokolu (např. Content-Type), ale můžeme si specifikovat i vlastní atributy dle potřeby.

Bucket

- místo, do něhož jsou objekty ukládány. Bucket má v rámci celého S3 jedinečné jméno, a každý soubor v něm uložený musí mít opět jedinečné jméno. Bucketů můžeme mít na jednom AWS účtu maximálně 100.

Key

- jednoznačná cesta k danému objektu v určitém bucketu. Každý objekt uložený v S3 má právě jeden unikátní klíč (key). Například objekt s názvem *obrazek.jpg*, uložený v bucketu *kiwi*, bude mít klíč `http://kiwi.s3.amazonaws.com/obrazek.jpg`.

V S3 neexistuje možnost vytváření podadresářů či „zanořených bucketů“ – úložiště není hlouběji strukturované pod úroveň bucketu. Přesto lze, alespoň navenek, podadresáře simulovat – pokud do bucketu *kiwi* uložíme objekt pojmenovaný *media/obrazek.jpg* (klíč), tak jeho přístupová adresa bude `http://kiwi.s3.amazonaws.com/media/obrazek.jpg`.

S3 nabízí dvě API pro přístup k souborům – klasické SOAP a REST. Knihovna *jets3t*, kterou Ki-Wi Server využívá pro snazší komunikaci s centrálním úložištěm nabízí implementaci obou zmíněných rozhraní. Pro přístup k centrálnímu úložišti je přiděleno přístupové jméno a heslo, souborům lze nastavit ochranu (čtení, zápis) a stahovat zabezpečeně přes HTTPS.

5.6. Enkódování médií

Přehrávač JW FLV Player podporuje pouze videa ve formátu FLV, proto je nutné všechna ostatní zkonvertovat. K tomuto účelu slouží online služba *encoding.com*, která za měsíční poplatek dovoluje automaticky enkódovat běžně používané videoformáty do požadovaného výstupu.

Po nahrání média do centrálního úložiště je zaslán požadavek na službu *encoding.com* (xml formát), který specifikuje zdrojovou a cílovou adresu souboru, výstupní formát a adresu pro zaslání výsledků převodu. Ki-Wi Server má vytvořen servlet, který čeká na výsledky konverze a u média v databázi mění stav na aktuální (úspěšný převod, chyba).

Služba *encoding.com* je placená, výše částky má vliv především na rychlost konverze.

5.7. JasperReports

JasperReports je reportovací nástroj, který dovede vytvářet dokumenty ve formátech PDF, XML, HTML, CSV, XLS, RTF a TXT. Design dokumentu se tvoří pomocí XML. Určuje se zde jednak layout dokumentu i práce s daty.

XML soubor, ve kterém je dokument nadefinován, má koncovku `jrxml`. Jeho kompilací vznikne soubor s koncovkou `jasper`. Pro běh JR je třeba několik knihoven - Jakarta Commons (Jakarta Commons BeanUtils, Jakarta Commons Collections, Jakarta Commons Logging, Jakarta Commons Digester). Pro vytváření PDF souborů je potřeba knihovna iText.

iReport je IDE sloužící pro vytváření zdrojových (`jrxml`) souborů. Pomocí grafického návrháře v něm navrhujeme formulář a připojíme datasource.

V Ki-Wi Serveru zajišťuje tvorbu reportů třída `PlaylogReporter` generující výstupní formáty TXT, HTML a PDF.

5.8. Quartz

Quartz je opensource systém od společnosti OpenSymphony pro plánování úloh, který je přizpůsoben integraci do J2EE kontejnerů. Plánovat lze na konkrétní čas, dny, roky, opakování nastavit pouze do určitého data. Spring Framework obsahuje přímou podporu Quartz.

Intervaly spouštění jsou tvořeny stejnou syntaxí jako u Unixového plánovače Cron (<http://en.wikipedia.org/wiki/Cron>)

5.9. JW FLV Player

Klíčová součást přehrávače médií na digitálním playlistu. JW FLV Player je modifikovatelný přehrávač, který z předaného playlistu začne v definovaném pořadí přehrávat jednotlivá média. Nevýhodou je omezení formátu videí pouze na FLV.

JW FLV Player podporuje přehrávání formátu Adobe Flash Player, formáty FLV, MP4, MP3, AAC, JPEG, PNG a GIF. Také nabízí podporu RTMP, HTTP, živého vysílání, různých formátů skladeb, široký rozsah nastavení, a rozsáhlé Javascript API. Dovoluje také kompletně přizpůsobit vzhled, včetně podpory pluginů.

6 Použité technologie

Kapitola stručně zmíní technologie a nástroje pro vývoj a provoz serveru a klienta, včetně operačního systému, na kterém je klient postaven.

1. Ki-Wi Server

Java EE 6

- Java Platform, Enterprise Edition, součást platformy Java pro vývoj a provoz podnikových aplikací a informačních systémů

- <http://java.sun.com/javaee/>

Spring Framework 2.5.6

- open source aplikační framework pro platformu Java EE. Usnadňuje vývoj aplikací v oblasti komplexních informačních systémů a podnikových aplikací. V současnosti nejpoužívanější framework pro vývoj na platformě Java EE.

- <http://www.springframework.org/>

Hibernate 3.2.6

- open source ORM knihovna pro jazyk java poskytující framework, který řeší mapování objektů do relační databáze.

- <http://www.hibernate.org/>

MySQL 5.1.6

- relační databázový systém

- <http://www.mysql.com/>

Jetty 6.1.19

- webový server a servletový kontejner implementovaný v jazyce Java

- <http://www.mortbay.org/jetty-6/>

Maven 2.0.9

- software pro správu projektů, používaný pro automatické sestavení projektu a kontrolu závislostí jednotlivých balíčků

- <http://maven.apache.org/>

Direct web remoting (DWR) 2.0.3

- knihovna pro asynchronní komunikace prohlížeče se serverem

- <http://directwebremoting.org/dwr/index.html>

Quartz 1.6.1

- knihovna pro nastavení a automatické spouštění naplánovaných úloh

- <http://www.opensymphony.com/quartz/>

JasperReports 3.5.3

- knihovna pro generování reportů

- <http://jasperreports.sourceforge.net/>

Eclipse 3.5.1

- vývojová platforma a prostředí pro tvorbu aplikací v jazyce Java

- <http://www.eclipse.org/>

Jets3t 0.6.1

- knihovna pro komunikaci se službou jets3t

- <http://jets3t.s3.amazonaws.com/>

2. Ki-Wi Player

XUL 1.9.1

- vykreslovací jádro Gecko (Firefox) v multiplatformní standalone verzi

- základ klientských aplikací Ki-Wi Server a Ki-Wi Kiosk

- <http://www.mozilla.org/projects/xul/>

JavaScript

- multiplatformní, objektově orientovaný skriptovací jazyk, využíván pro řízení přehrávače

JW FLV Player 4.3

- flashový přehrávač videa ve formátu FLV, MP4 včetně podpory zobrazování obrázků, podporuje velké množství playlistů definujících přehrávaný obsah

- http://www.jeroenwijering.com/?item=JW_FLV_Media_Player

Dojo Toolkit

- open source DHTML toolkit napsaný v JavaScriptu, podpora implementace technologie Comet

- <http://www.dojotoolkit.org/>

CometD 1.0.beta8

- javascriptová knihovna s klientskou implementací Cometd pro připojení na Ki-Wi Server server

- <http://www.cometd.org>

3. Ki-Wi OS

Ubuntu 9.03

- operační systém Linux, na kterém běží digitální panely
- obsahuje jednoduchý správce oken RatPoison