VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH INFORMAČNÍHO SYSTÉMU PRO EVIDENCI VIDEOZÁZNAMŮ
DESIGN OF AN INFORMATION SYSTEM FOR VIDEO RECORDINGS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jiří Psota**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Dydowicz, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Jiří Psota**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
Vedoucí práce:	**Ing. Petr Dydowicz, Ph.D.**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává bakalářskou práci s názvem:

Návrh informačního systému pro evidenci videozáznamů

Charakteristika problematiky úkolu:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrh řešení, přínos práce
Závěr
Seznam použité literatury

Cíle, kterých má být dosaženo:

Cílem práce bude navrhnout informační systém, prostřednictvím kterého bude možné spravovat databázi videozáznamů. S tímto IS bude následná manipulace a vyhledávání různých dat jednodušší, rychlejší a přehlednější.

Základní literární prameny:

BASL, J. a R. BLAŽÍČEK. Podnikové informační systémy. Podnik v informační společnosti. Praha: Grada, 2008. 283 s. ISBN 978-80-247-2279-5.

MOLNÁR, Z. Automatizované informační systémy. Praha: Strojní fakulta ČVUT, 2000. 126 s. ISBN 80-01-02269-2.

MOLNÁR, Z. Efektivnost informačních systémů. Praha: Grada Publishing, 2000. 142 s. ISBN 80-716--410-X.

PECINOVSKÝ, R. Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky. Praha: Grada, 2009. 570 s. ISBN 978-80-247-2653-3.

SODOMKA, P. a H. KLČOVÁ. Informační systémy v podnikové praxi. Brno: Computer Press, 2010. 501 s. ISBN 978-80-251-2878-7.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně dne 28.2.2021

L. S.

Mgr. Veronika Novotná, Ph.D. ředitel	doc. Ing. Vojtěch Bartoš, Ph.D. děkan

Abstrakt

Bakalářská práce se zaměřuje na návrh informačního systému ve webovém prostředí. Výsledkem je funkční celek, který nabízí širší možnosti při evidování videosouborů.

Klíčová slova

informační systém, videosoubory, evidence, datové modelování, databáze

Abstract

The bachelor thesis focuses on the design of an information system in a web environment. The result is a functional unit that offers wider options for recording video files.

Key words

information system, video files, records, data modelling, databases

Bibliografická citace

PSOTA, Jiří. *Návrh informačního systému pro evidenci videozáznamů* [online]. Brno, 2021 [cit. 2021-05-10]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/135310. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Ing. Petr Dydowicz, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 16. května 2021

…….

podpis autora

Poděkování

Rád bych poděkoval panu Ing. Petru Dydowiczovi, Ph.D. za vedení této bakalářské práce. Dále děkuji své rodině i přátelům za jejich podporu.

OBSAH

ÚVOD

Tato bakalářská práce se zaměřuje na problematiku evidování videozáznamů. Během praxí jsem měl možnost nahlédnout do oblasti regionálních televizí, získat nové poznatky a pozorovat metodiku práce. Tato problematika mne zaujala, a proto jsem se rozhodl vytvořit vlastní návrh, který zlepší výkonnost a zpřehlední evidenci videozáznamů.

V kapitole teoretických předpokladů jsou popsány prostředky, nástroje i související pojmy, s jejichž podporou jsem realizoval zamýšlený návrh.

Následující kapitola uvádí přehled o současném stavu evidence videozáznamů, také uvádím způsoby, jakými se v současné době evidují videozáznamy ve zmíněné oblasti podnikání. Uvádím také existující způsoby řešení evidence videozáznamů, jejich vzájemné porovnání a s tím spojené dedukce zjištěných nedostatků. Součástí této kapitoly je vymezení požadavků na nově vznikající návrh evidence videozáznamů, jež plyne z obsahu celé této kapitoly.

V kapitole vlastních návrhů řešení je předložen navrhovaný koncept řešení a popis sestavení datového modelu zahrnující všechny dílčí informace o uspořádání relací a jejich vazbách.

Závěr kapitoly věnuji ekonomickému zhodnocení a přínosům, které tento návrh nabízí.

VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

Cílem této bakalářské práce je navrhnout informační systém pro evidenci videozáznamů. Výsledkem je návrh funkčního celku, který usnadní evidenci videosouborů. Řešení zahrnuje zajištění plnohodnotného přístupu k informačnímu systému, což znamená následující druhy operací: vyhledávání záznamů, vytváření nových záznamů, editace a odstraňování existujících záznamů. Vedlejším cílem je přehledně zpracované grafické rozhraní pro pohodlné a intuitivní používání tohoto informačního systému uživateli.

V této práci představím prostředky, se kterými chci dosáhnout vytyčených cílů. Následně provedu analýzu současného stavu za účelem zjištění, jak je evidování videosouborů řešeno v regionálních a celoplošných televizích. Na základě analýzy provedu návrh nového informačního systému (nadále jen IS), který bude vyhovovat zjištěným požadavkům.

Průběh vytváření IS bude zahájen datovým modelováním a dále bude následovat vytvoření funkčního modelu tak, aby bylo zajištěno přívětivé uživatelské prostředí.

1 TEORETICKÁ VÝCHODISKA PRÁCE

1.1 Informační systém

„Přesná definice pojmu Informační systém neexistuje a ani ji nelze jednoduše vytvořit, neboť každý uživatel či tvůrce Informačního systému používá různé terminologie a zdůrazňuje jiné aspekty. Můžeme však říci, že Informační systém (IS) lze chápat jako systém vzájemně propojených informací a procesů, které s těmito informacemi pracují. Přičemž pod pojmem procesy rozumíme funkce, které zpracovávají informace do systému vstupující a transformují je na informace ze systému vystupující. Zjednodušeně můžeme říci, že procesy jsou funkce zabezpečující sběr, přenos, uložení, zpracování a distribuci informací.“ (1)

1.2 Databáze

V této podkapitole uvádím pojem databáze, různé datové modely a také jazyk SQL (Structured Query Language).

„Databáze je organizovaný soubor strukturovaných informací neboli dat, které se obvykle ukládají v elektronické podobě v počítačovém systému. Databáze je obvykle řízena systémem pro správu databáze (DBMS – database management system). Data a systém DBMS společně s přidruženými aplikacemi se označují jako databázový systém, často zkráceně jako databáze.“ (2)

Uvedený citát je ještě potřeba doplnit o vysvětlení rozdílu slov data a informace. Koná-li člověk rozhodnutí na základě dat, pak lze tato data nazvat informacemi, neboť datům přiřazuje význam a smysl. (3, str. 5)

1.2.1 Datové modely

Je-li projektován informační systém, existuje pět možných druhů datových modelů, které lze využít. Tyto druhy datových modelů jsou:

- lineární,
- hierarchický,
- síťový,
- relační,
- objektový (3, str. 20).

Ad lineární) V tomto datovém modelu jsou obsaženy samostatné tabulky bez vzájemného propojení. V jednoduchých systémech, bez nutnosti vytváření souvislostí, je tento model dostačující. Příkladem může být kartotéka pacientů, kdy každá zásuvka odpovídá jedné tabulce. (3, str.21)

Ad hierarchický) Tento model dokáže již lépe pracovat s daty, protože již existuje uspořádání segmentů (Segment lze přirovnat k tabulce). Například: Učitelé vyučují předměty. V tomto příkladu jsou všichni učitelé zapsáni v prvním (rodičovském) segmentu a předměty, které vyučují, se nachází v hierarchicky níže položené úrovni.

Ad síťový) Síťový model je obdobou hierarchického modelu, ale navíc s možností libovolného propojení požadovaných segmentů (3).

Ad relační) Relační model vznikl kombinací několika lineárních modelů pomocí prostředku, který se nazývá relační klíč.

Jedná se o jeden z nejpoužívanějších modelů v současné době (3, str.22).

Ad objektový) Objektový datový model je nejnovější a je vystavěn na základním prvku nazvaném objekt. Tento objekt má kromě svých atributů definované také metody, které určují chování objektu (3, str.23).

1.2.2 Relační datový model

Existují-li množiny, v terminologii teorie množin pak domény, například čísel studentů – D_1, jmen studentů – D_2 a příjmení studentů D_3, pak relace na doménách D_1, D_2, …, D_n je dvojice $R = (R, R^*)$, kde $R = R(A_1: D_1, A_2: D_2, \dots A_n: D_n)$ je schéma relace a $R^* \subseteq D_1 \times D_2 \times \dots \times D_n$ je tělo relace. (3, str. 26)

Schéma relace je tvořeno atributy. Počet atributů relace se označuje jako stupeň neboli řád relace a je konstantní (3, str. 27). Tělo relace představuje realitě odpovídající podmnožinu kartézského součinu (3, str. 27). Výsledkem násobení libovolného počtu množin je množina n-tic. Tato množina již je variabilní. Kardinalita těla relace $m = |R^*|$ se označuje jako kardinalita relace. Relaci je možné zobrazit pomocí tabulky za následujících podmínek:

1) každý řádek tabulky odpovídá jedné n-tici relace,
2) pořadí řádků je nevýznamné,
3) žádné dva či více řádků nejsou identické,

4) pořadí sloupců je nevýznamné,
5) význam každého sloupce je určen názvem atributu,
6) žádné dva názvy sloupců tabulky (atributů relace) nejsou stejné,
7) hodnoty ve sloupcích jsou atomické (tzn. dále na menší části nedělitelné), (3, str. 27)

1.2.3 Integrita relačního modelu

„Integritu modelu lze chápat jako stav, při kterém data uložená v modelu odpovídají vlastnostem objektů reálného světa. Tyto integritní omezení je možné rozlišit na: Integritní omezení pro entity (relace) a Integritní omezení pro vztahy entit (relační vazby)“ (3, str. 28).

Integritní omezení pro entity (neboli relace) se dále dělí na doménovou integritu, entitní integritu a referenční integritu (3).

- Doménová integrita

Doména je spojena s každým atributem v relační databázi. Doména se může lišit pro každý atribut nebo se mohou dvě či více domény spojit se stejným atributem. Díky pojmu domény lze definovat význam a určit zdrojové hodnoty, které může atribut obsahovat. Zjednodušeně řečeno doména popisuje množinu hodnot, kterých mohou atributy nabývat. (4)

- Entitní integrita

„Každá relace musí mít určen tzv. primární klíč – jeden nebo více atributů, jejichž hodnoty jednoznačně identifikují každý z řádků relace.“ (3, str. 29)

Primární klíč (angl. primary key, nadále PK) je definován jako množina atributů relace s následujícími vlastnostmi:

1) je jednoznačná – relace neobsahuje dvě ani více stejných n-tic, které pro konkrétní množinu atributů mají stejné hodnoty,
2) je minimální – všechny atributy jsou důležité a není možné ani jeden vyloučit způsobem, který by porušil platnost prvního pravidla.

Kromě vlastností musí PK splňovat i další podmínky, jakými jsou:

1) Atributy primárního klíče musí obsahovat vždy všechny hodnoty, žádná hodnota nesmí chybět,

2) Každá n-tice relace musí být v každém okamžiku identifikovatelná hodnotou primárního klíče (3, str. 29).

Pojem kandidátní klíč (angl. candidate key) vyjadřuje totéž, co klíč primární s tím rozdílem, že kandidátních klíčů může být více, ale primární klíč může být pouze jeden. PK se tedy vybírá z možných kandidátních klíčů. Proces volby PK závisí především na analytikovi.

- Referenční integrita

„Cizí klíč (Foreign key) je atribut, který splňuje tyto nezávislé vlastnosti:

1) *každá hodnota je buď plně zadaná nebo plně nezadaná,*
2) *existuje jiná relace s takovým primárním klíčem, že každá zadaná hodnota cizího klíče je identická s hodnotou primárního klíče nějaké n-tice této relace“*

(3, str. 29)

Díky referenční integritě lze vytvářet relační vazby mezi relacemi. Obsahuje-li relace A primární klíč, pak tento klíč může být „vložen“ do relace B jako klíč cizí a tím mezi relacemi A a B vznikne relační vazba. Tak jako entitní integrita, má i referenční integrita svá pravidla:

1) PK a CK musí být definovány na stejné doméně (soulad hodnot),
2) Žádná nesouhlasná hodnota CK se nesmí v databázi vyskytovat (3).

1.2.4 Integritní omezení pro vztahy

„Integritní omezení pro vztahy omezuje kardinalitu vztahu na poměry 1:1, 1:N, N:1, N:M. Tento poměr uvádí, kolik n-tic relací sobě navzájem odpovídá.“ (3, str. 31)

Vztah 1:1 určuje, že jedna n-tice relace A odpovídá jedné (nebo žádné) n-tici relace B. Obsahuje-li relace A datový objekt *„člověk“* a relace B datový objekt *„občanský průkaz“*, pak lze uvést v platnost následující tvrzení: *Jeden člověk může vlastnit pouze jeden občanský průkaz.*

Vztah 1:N určuje, že jedna n-tice relace A odpovídá jedné (nebo více) n-tici relace B. Tento vztah lze demonstrovat na praktickém příkladu. Vztah mezi entitami *„knihovna“* a *„knihy“* lze popsat takto: *Jedna knihovna může mít jednu, obvykle ale více, knih.*

Vztah N:1 je inverzní obdobou vztahu 1:N.

Vztah N:M určuje, že alespoň jedna nebo více n-tic relace A odpovídá jedné či několika n-ticím relace B. Jako příklad lze uvést vztah mezi entitami: *„kniha“* a *„autor“*. Jednu knihu napsal jeden nebo více autorů, ale současně lze říci, že jeden (a tentýž) autor napsal jednu nebo více knih. Přípustné varianty *„nebo více“* na obou stranách relační vazby neumožňují tuto vazbu přímo vytvořit. Řešením tohoto vzniklého problému je vytvoření tzv. průnikové entity s primárním klíčem složeným z obou primárních klíčů původních entit (3).

1.2.5 Normalizace

Normalizace představuje činnost, která spočívá v upravení návrhu datových struktur s cílem zjištění, zda jsou dodržena či porušena pravidla pro určitou normální formu (3) (4). V případě, že pravidla nejsou dodržena, je potřeba zjištěné nedostatky odstranit. Normalizaci databáze na vyšší normalizační úrovni předchází dodržení všech normalizačních pravidel nižších úrovní (3). Existuje několik forem, přičemž nejpoužívanější jsou následující tři (4).

1) První normální forma

„Relace je v první normální formě, pokud jsou všechny její atributy definovány nad skalárními obory hodnot (doménami).“ (3)

Jinými slovy lze říci, že všechny atributy, ze kterých se relace skládá, musí být jednoduché, nikoli složené či vícehodnotové. Má-li člověk více než jeden automobil, tak z hlediska této normální formy není možné všechny tyto automobily uložit do jedné relace, ani do jednoho či do několika atributů. Samozřejmě v praxi se takové případy běžně vyskytují a pro vyřešení tohoto problému se využívá dekompozice – viz podrobněji v kapitole 1.2.6

2) Druhá normální forma

Tato forma může vstoupit v platnost pouze za předpokladu, že již platí první forma. Kromě toho jsou všechny atributy relace závislé na celém kandidátním (primárním) klíči (3). Při návrhu databáze existuje množina kandidátních klíčů v relaci. Je-li z této množiny zvolen primární klíč, který není složený, tj. obsahující pouze jeden atribut, pak automaticky platí, že relace je i ve druhé normální formě (4). Pokud se skládá primární klíč z více atributů, pak musí být dodržena funkční závislost. Tato závislost popisuje vztah mezi atributy a také indikuje vzájemnou souvislost atributů.

3) Třetí normální forma

„Relace je ve třetí normální formě, pokud je ve druhé normální formě, a navíc všechny její neklíčové atributy jsou vzájemně nezávislé.“ (3, str. 60)

Tranzitivní závislosti se v n-ticích, neboli záznamech, nesmí vyskytovat. Každý neklíčový atribut, tj. takový atribut, který není součástí primárního klíče, musí být funkčně závislý na celém klíči; to vyplývá z definice druhé normální formy. V případě, že je tato závislost zprostředkována přes jiný neklíčový atribut, pak je mezi tímto atributem a klíčem tranzitivní závislost (3). Výskyt tranzitivních závislostí v relaci způsobuje redundanci dat v rámci n-tic atributu (4). Pro bližší vysvětlení uvádím příklad. Bude-li jedna relace obsahovat seznam zaměstnanců a současně údaje o pobočce, ve které pracují, pak je tranzitivní závislost patrná. Vzhledem k tomu, že na pobočce může pracovat více než jeden zaměstnanec, pak dochází k výskytu redundantních záznamů, protože např. adresa a telefonní číslo jedné pobočky budou pro dva různé zaměstnance totožné.

Tabulka č. 1: Třetí normální forma (redundantní záznamy)
(Zdroj: Vlastní zpracování)

ID_zam	jmeno_zam	Prijmeni_zam	dat_nar_zam	adresa_pobocky	tel_pobocky
001	Milan	Kratochvíl	4. 7. 1990	Havlíčkova 130, 602 00 Brno	543219778
002	Martin	Veselý	6. 6. 1979	Havlíčkova 130, 602 00 Brno	543219778
003	Magda	Hrušínská	1. 3. 1991	Koniklecová 10, 603 00 Brno	546247192

1.2.6 Dekompozice

Dekompozice se týká kardinality vztahu při poměru N:M. Cílem je rozložit tento poměr na dva poměry 1:N a N:1 za pomoci další relace. K příkladu z předchozí podkapitoly: několik lidí může vlastnit několik automobilů. Existuje jedna relace, ve které jsou údaje pouze o lidech. Pro uložení automobilů je potřeba vytvořit novou (druhou) relaci, která bude obsahovat všechny potřebné údaje pouze o automobilech. Obě dvě relace mají primární klíč, přičemž mezi těmito relacemi existuje poměr N:M. Pro vytvoření spojení

mezi těmito relacemi se vytvoří třetí relace, která toto spojení zprostředkuje. Třetí relace bude obsahovat dva atributy, jakožto cizí klíče obou předchozích relací.

1.2.7 SQL

Strukturovaný dotazovací jazyk SQL je komerčním databázovým jazykem, který byl navržen pro užívání odborníky i neodborníky (4). SQL je neprocedurální jazyk, takže stačí pouze zadat jaké údaje a z jaké relace jsou požadovány. Jsou-li požadovány údaje z několika relací současně, pak je nutné definovat, za pomoci příkazu JOIN (příp. WHERE), jakým způsobem se další relace připojí. SQL má volný formát zadání příkazů. Struktura příkazů se skládá z anglických slov: SELECT (vybrat), INSERT (vložit), UPDATE (aktualizovat), ORDER BY ASC/DESC (seřadit sestupně, vzestupně) a mnohé další příkazy. SQL zahrnuje mnoho rezervovaných slov, které jsou součástí jazyka. Uživatel je musí psát přesně (avšak velikost písmen příkazu je irelevantní) a neměl by je používat např. v názvech tabulek atp. Tento jazyk je standardizovaný databázový jazyk, který získal široké přijetí. Většina významnějších dodavatelů nabízí databázové produkty založené na SQL či s rozhraním SQL (4). Jak již bylo řečeno, SQL nabízí různí dodavatelé. Právě kvůli tomu i přes vlastnost standardizace jazyka SQL existují mírné nuance u různých dodavatelů.

Pro demonstraci jednoho rozdílu formulace dotazu uvedu příklad u 2 různých firem. Cílem bude získat omezený počet n-tic, které SQL vrátí po zpracování.

Tabulka č. 2: Ukázka rozdílu SQL serveru dvou různých společností
(Zdroj: Vlastní zpracování)

Oracle – MySQL	Microsoft – MS SQL (Lite)
SELECT * FROM tabulka LIMIT 10	SELECT TOP 10 * FROM tabulka

Oba dotazy vrátí 10 n-tic. U Microsoftu musí být omezení zapsáno za příkazem SELECT, zatímco u MySQL nezáleží, zda příkaz LIMIT bude zapsán za FROM či WHERE nebo ještě dál. Hvězdička je zástupným znakem a vyjadřuje, že uživatel požaduje absolutně všechny n-tice, které jsou v relaci obsaženy.

„MySQL je velmi rychlý a robustní systém pro správu relačních databází (RDBMS).“ (6)

MySQL server řídí přístup k datům. Díky využití architektury klient-server může k datům přistupovat několik uživatelů současně. Zajišťuje přístup pouze autorizovaným

uživatelům. MySQL server Community je veřejně a zdarma dostupný. Existují i další placené edice, například Standard, Enterprise, Cluster CGE (5).

MySQL je celosvětově nejpopulárnější open-source databáze (5). Jak již název napovídá, s databází lze komunikovat prostřednictvím jazyka SQL.

1.3 HTML

Značkovací jazyk HTML (z anglického Hypertext Markup Language) je v současnosti používán pro vytváření strukturovaných dokumentů při vývoji sekcí pro uživatele webových aplikací. Používá se pátá verze s označením HTML5 (7). Soubory HTML mají i stejnojmennou příponu, skládají se z definice typu a jedné (kořenové) části, ve které jsou vnořeny další dvě části. Na začátku dokumentu je definován typ <!DOCTYPE html>, pak následuje kořenový tag stránky <html>. Vnořenými částmi v <html> jsou <head> a <body>. Jedná se o párové tagy, které musí být také ukončeny </html>. Tag head obsahuje popisné informace. V tagu body, což je tělo stránky, se zapisuje rozložení a celý obsah stránky (7).

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4
5   </head>
6   <body>
7
8   </body>
9 </html>
```

Obrázek č. 1: Základní struktura souboru HTML
(Zdroj: Vlastní zpracování)

Příklady některých elementů, kterými HTML disponuje:

<h1>Nadpis první úrovně (velký font)</h1>,

<h2>Nadpis druhé úrovně (menší font)</h2>,

Odstavec ,

<img> obrázek (nepárový tag),

<hr> vodorovná čára (nepárový tag).

Pokud standardem definované elementy vývojáři nestačí, lze vytvořit vlastní elementy.

1.4 CSS

CSS neboli kaskádové styly (z anglického Cascading Style Sheets) je jazyk, který se využívá k popisu zobrazení dokumentů. Popisování se provádí za pomoci pravidel, které se skládají ze selektoru a deklarace množiny vlastností. Díky těmto vlastnostem lze navrhnout webovou stránku od základního rozvržení, přes výběr fontu, barev, až po přizpůsobení pro různá zařízení či využití grafických efektů (8). Selektor může odkazovat přímo na značku v HTML nebo na konkrétní jednu značku za využití selektoru *id* či *class*.

```
1 table tr {
2   padding-top: 12px;
3   padding-bottom: 12px;
4   text-align: left;
5   background-color: #4CAF50;
6   color: white;
7 }
```

Obrázek č. 2: Ukázka kódu CSS
(Zdroj: Vlastní zpracování)

Na obrázku výše je zobrazen příklad užití CSS, kdy všechny vlastnosti se vztahují na všechny řádky všech tabulek, které se v rámci HTML souboru vyskytují. Vlastnosti jdoucí za sebou vyjadřují: Nastavení vnitřního horního a dolního odsazení od obsahu, zarovnání textu vlevo, barvu pozadí a barvu textu. Tento obrázek také prozrazuje, že barvu je možné zapsat několika různými způsoby, a to buď slovem nebo hexadecimálně či dekadicky. (12)

1.5 Javascript

„Javascript je plnohodnotným programovacím jazykem, schopným provádět složité výpočty a interakce, včetně uzávěrů, anonymních (lambda) funkcí, a dokonce i metaprogramování.“ (11, str. 29)

Javascript je významnou součástí všech známých webových prohlížečů, které jsou na trhu dostupné. Mobilní prohlížeče nevyjímaje. Z toho plyne, že Javascript běží na straně klienta ve webovém prohlížeči. Díky tomuto jazyku je možné ověřovat uživatelem zadaná data ve formuláři ještě před odesláním, zda jsou korektní a zda například neobsahují nepovolené znaky, atp. Javascript je ovšem možné (např. ve standardní verzi Mozilly Firefoxu na PC) deaktivovat. Z tohoto důvodu je zapotřebí provést kontrolu přijatých údajů ještě na straně serveru.

```
1 setTimeout(function(){
2   document.getElementById("demo").innerHTML = "<p>Ahoj</p>"
3 }, 1500);
```

Obrázek č. 3: Ukázka kódu Javascript
(Zdroj: Vlastní zpracování)

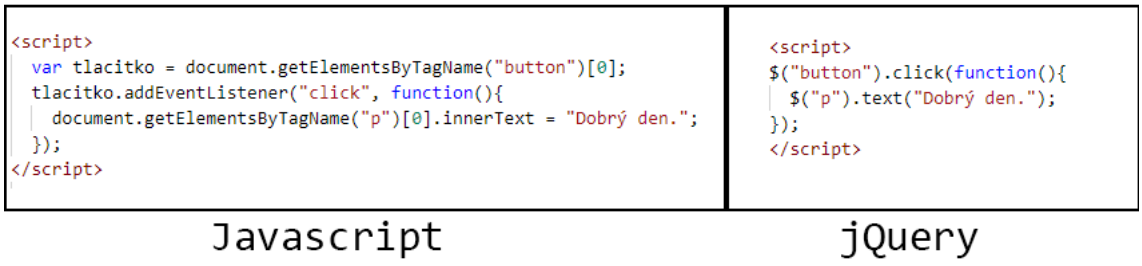
Na tomto obrázku je zobrazena funkce, která nastavuje časový odpočet. Časový limit odpočtu je v tomto případě 1500 milisekund. Jakmile uplyne tato doba, dojde ke spuštění kódu uvnitř této funkce. Kód obsahuje pouze jeden příkaz, který v HTML najde element s identifikátorem *demo* a vloží do něj nový odstavec s textem *Ahoj*. Tento příkaz lze použít i opačně, tzn. ke čtení obsahu elementu se zadaným identifikátorem.

1.5.1 jQuery

jQuery je rychlá, malá javascriptová knihovna s bohatými funkcemi. Díky snadno použitelnému rozhraní API funguje v mnoha prohlížečích. Tato knihovna podporuje také manipulaci s HTML dokumenty, událostmi a animacemi. V neposlední řadě nabízí uživateli možnost využívat asynchronní volání skriptů PHP na serveru (21).

Nechť v HTML dokumentu existují dva párové elementy tlačítko <button> a odstavec <p>. V tomtéž dokumentu existuje také párový tag <script> pro zápis kódu Javascriptu.

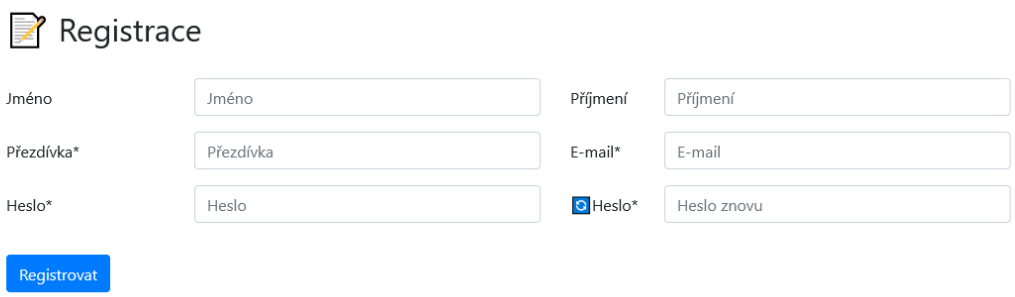
Obrázek níže demonstruje rozdíl mezi knihovnou jQuery a klasickým Javascriptem, kdy výsledkem je v obou případech zobrazení textu *Dobrý den.* po kliknutí na tlačítko.



Obrázek č. 4: Rozdíl mezi Javascriptem a jQuery
(Zdroj: Vlastní zpracování)

1.6 Bootstrap

Bootstrap je nejpopulárnější knihovna kombinující CSS a Javascript. Tato knihovna je zdarma ke stažení a používání. Bootstrap zahrnuje předdefinované vlastnosti, díky kterým je možné přesně nastavit, jak se budou příslušné prvky na stránce zobrazovat na konkrétních zařízeních (9). Použití této knihovny je velmi snadné. Nejprve je potřeba nastavit v hlavičce HTML souboru odkaz na knihovnu a následně stačí přiřadit elementům HTML atribut CLASS a zadat požadovanou hodnotu, kterou chce uživatel nastavit. Velkou výhodou Bootstrapu je, že má na svých webových stránkách připravené části HTML kódů, které lze jen zkopírovat, přizpůsobit a používat. Tato knihovna rovněž disponuje např. vysouvacími menu tlačítky, nebo kompletní visáží formulářů s různými typy tlačítek, atp.



Obrázek č. 5: Ukázka registračního formuláře s využitím Bootstrapu
(Zdroj: Vlastní zpracování)

1.7 PHP

„PHP bylo původně zkratkou (z anglického Personal Home Page), ale její význam se změnil po vzoru rekurzivní konvence pojmenování GNU (z anglického Gnu's Not Unix) – takže nyní znamená Hypertext Preprocessor.“ (6, str. 30)

PHP je skriptovací jazyk, který běží na straně serveru. Byl navržen speciálně pro web (6). Kód jazyka PHP je interpretován webovým serverem a výsledkem je HTML, které se po dokončení zpracování odešle uživateli a zobrazí se.

PHP má otevřený zdrojový kód.

```php
<?php
  $cislo1 = 5;
  $cislo2 = 5;

  echo "Součet čísel " . $cislo1 . " a " . $cislo2 . " je " . ($cislo1 + $cislo2) . ".";
?>
```

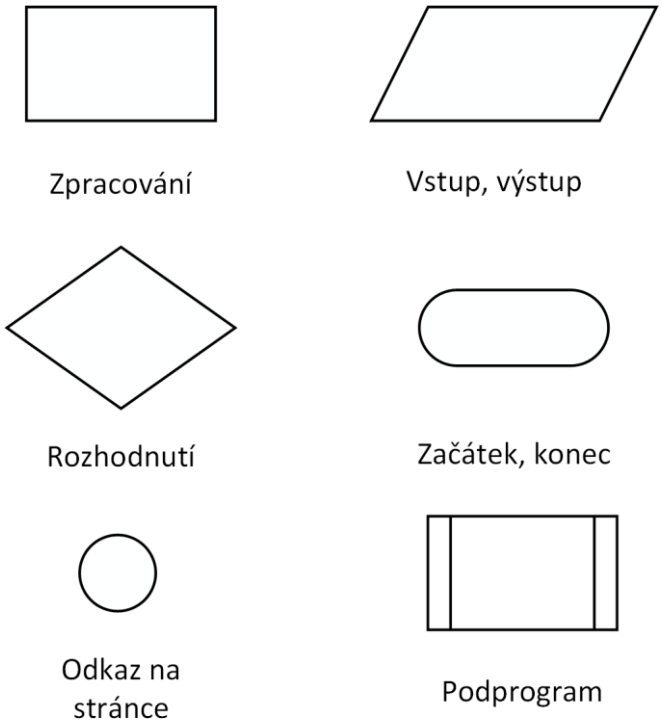
Obrázek č. 6: Ukázka kódu PHP
(Zdroj: Vlastní zpracování)

V souboru s příponou *.php* lze kombinovat zápis jazyků HTML i PHP. Tagy HTML se zapisují standardně jako v obyčejném souboru s příponou *.html*, ale kód skriptovacího jazyka PHP je nutné ohraničit jedním párovým tagem – tak jak je z ukázky na prvním a šestém řádku patrné. Dále je na ukázce zapsán příklad deklarace proměnných a příkaz *echo* sloužící ke zobrazení výstupu uživateli. Tečky se využívají ke spojování textových řetězců a znaménko plus k vykonání aritmetické operace sčítání. Výsledek, který vidí uživatel po dokončení zpracování skriptu vypadá následovně: *Součet čísel 5 a 5 je 10.*

1.8 Vývojový diagram

„Slovní zápis algoritmu má jen velmi omezené možnosti použití.“ (13, str. 6)

Díky vývojovému diagramu je možné širšímu okolí uživatelů graficky znázornit algoritmus. Pro grafické znázornění se využívají následující obrazce:



Obrázek č. 7: Některé grafické obrazce vývojového diagramu
(Zdroj: Vlastní zpracování dle: 13, str.6)

Jednotlivé obrazce se spojují tzv. spojnicemi. Spojnice je pravoúhlá a orientovaná, tzn. že má směr a zakresluje se jako šipka. Ve vývojovém diagramu se dodržuje směr shora dolů, proto není nutné svislé čáry kreslit se šipkou (13, str. 7). Zakreslení spojnice jako šipky je nutné v případě, kdy se mění její směr. Odkaz na stránce se využívá jako spojka mezi částmi diagramu. Toho se například využívá v situacích, kdy je zapotřebí, aby diagram pokračoval na další straně. Kosočtverec se využívá při rozhodování. Obvykle symbolizuje podmínku jako v programovacím jazyce. Do tohoto obrazce vstupuje jedna spojnice, ale vystupují alespoň dvě; v tomto místě dochází k vyhodnocení podmínky, zda platí či nikoli a dále dochází k větvení. Diagram musí být ucelený a všechny spojnice z jednotlivých obrazců musí mít nějaký cíl.

1.9 XML

Značkovací jazyk XML (z angl. eXtensible Markup Language) slouží k vytváření dokumentů, které jsou složeny z elementů. Tyto elementy jsou uspořádány do hierarchie a začínají kořenovou entitou. Podobně jako v HTML se i v XML dokumentech využívají tzv. tagy. Tagy mohou být párové i nepárové. Mají 3 základní podoby:

- Začínající tag,
- Ukončovací tag,
- Tag prázdného elementu (10).

Strukturu XML dokumentu lépe zachycuje obrázek níže.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<knihy>
    <kniha>
        <nazev>Komu zvoní hrana</nazev>
        <autor narodnost="americka">Ernest Hemingway</autor>
    </kniha>
</knihy>
```

Obrázek č. 8: Ukázka XML dokumentu
(Zdroj: Vlastní zpracování dle: 10)

Na obrázku č. 8 je vidět praktické využití XML dokumentu. Párový element <*knihy*> je kořenovým elementem. V tomto kořenovém elementu je vnořený další párový element <*kniha*>. Dalšími vnořenými elementy jsou <*nazev*> a <*autor*>. Díky tomuto hierarchickému uspořádání lze vytvářet přehledné dokumenty, jež se dobře strojově zpracovávají a současně jsou také dobře uživatelsky čitelné.

2 ANALÝZA SOUČASNÉHO STAVU

V následujících podkapitolách detailně přiblížím problematiku evidence videozáznamů a popíšu současnou situaci.

Objem videozáznamů neustále roste. Pro doložení svého tvrzení uvedu příklad. Z oficiální statistiky serveru Youtube.com vyplývá, že na tento server je každou minutu odesíláno mnoho videozáznamů, jejichž celková délka je více než 500 hodin (14). S neustále se zvyšujícím se počtem a objemem videozáznamů se potýkají nejen uživatelé serveru Youtube.com, kteří publikují svůj obsah, ale i televize, rádia, a mnohé další subjekty. Veškeré tyto videozáznamy je důležité evidovat a identifikovat.

2.1 Současná řešení evidence videozáznamů

Nejprve je nutné si určit kategorie, kteří lidé či organizace vůbec potřebují evidovat videozáznamy.

Běžný člověk, který občas dokumentuje své okolí prostřednictvím videozáznamů zřejmě žádnou pokročilou evidenci nevyužije. Malá firma či produkční společnost, která působí ve filmovém či podobném odvětví již podrobnější evidenci aplikuje. Do této skupiny patří i regionální televize. Do třetí kategorie patří velké produkční společnosti a celoplošné televize. Tato skupina ovšem disponuje velkými peněžními prostředky a může si dovolit využívat komplexní informační systém zahrnující archivní i odbavovací řešení.

Běžnému člověku stačí pouze propracovaný systém ukládání na fyzickém disku s výstižnými názvy souborů a adresářů. Regionální televize nemívají vhodný rozpočet na to, aby se z technologického hlediska mohly rovnat celoplošným televizím.

Oslovil jsem 6 regionálních televizí s cílem zjištění, jakým způsobem evidují videozáznamy. Pět ze šesti mi odpovědělo, že se jejich evidence nachází v prostředí Excelu či jiném tabulkovém procesoru. V takovém souboru mají definováno několik atributů, díky kterým jsou schopni dohledat, na kterém pevném disku se konkrétní videosoubor nachází.

Informaci od zbývající televize se mi nepodařilo získat, neboť tato televize považuje svůj systém evidence videozáznamů za striktně důvěrný a součástí know-how.

V následující podkapitole se budu detailněji věnovat úskalí nynějšímu používanému způsobu evidování. Dále pak uvedu požadavky, které ze zjištěných nedostatků vyplývají.

2.1.1 Úskalí

Regionální televize mívají často jednotky až nízké desítky zaměstnanců. Používání jednoho souboru pro evidenci značně omezuje výkonnost celé firmy. Pouze jeden uživatel může v jednu chvíli pracovat s tímto souborem. Samozřejmě lze tento soubor duplikovat a umožnit přístup dalšímu uživateli, jenže v případě změny přestává být aktuální. Naopak v duplikovaném souboru vznikne aktualizace, která se nemůže projevit v souboru předchozím. Jedná se o nežádoucí jev a decentralizaci dat. Z hlediska důvěrnosti lze hovořit o vysoké ochraně informací, už jen z důvodu, že k tomuto souboru mají přístup pouze zaměstnanci firmy. Nicméně na ochranu informací se v tomto odvětví nemusí brát až takový zřetel, neboť se neshromažďují citlivá osobní data. Mnohem důležitější je zde dostupnost a integrita.

Dalším problémem je lidský faktor. Ten působí vždy od doby otevření až do uzavření souboru. Při zapisování nového záznamu může uživatel jednoduše vytvořit překlep či chybně vyplnit záznam. Důsledkem tohoto omylu je nejen narušení konzistence uspořádání sloupců a řádků v souboru, ale také následná nemožnost dohledání správného záznamu.

2.2 Existující řešení

V této kapitole popíšu mnou nalezená existující řešení. Žádné z těchto existujících řešení však plně nevyhovuje potřebám pro evidenci videozáznamů, protože jsou koncipovaná pro různorodá data. Výsledkem této bakalářské práce je návrh informačního systému, který je zaměřen na evidenci videozáznamů se zohledněním požadavků a zjištěných nedostatků v existujících řešeních.

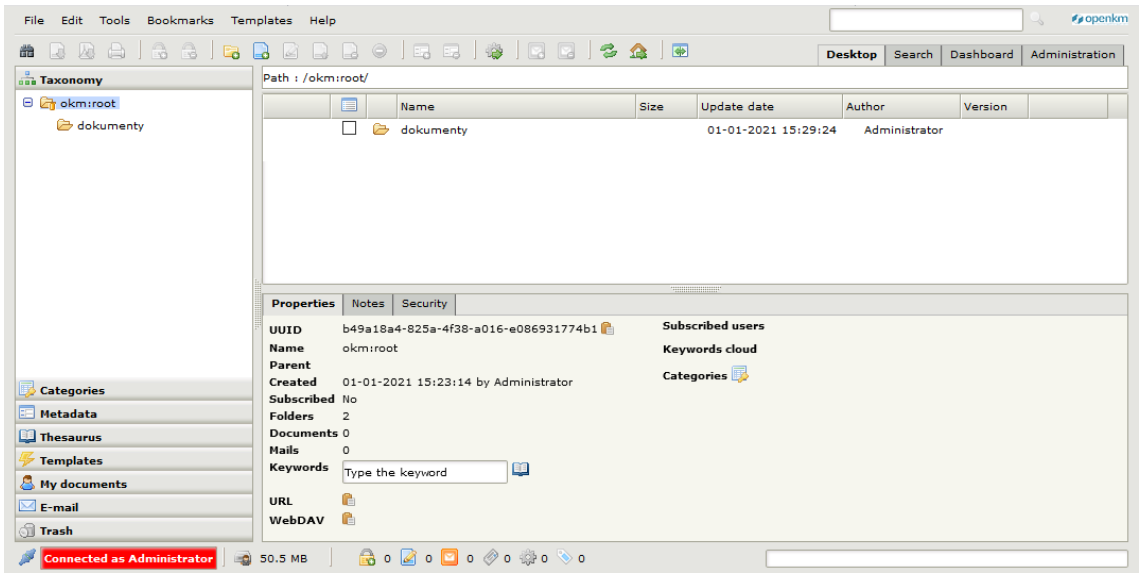
2.2.1 Provys

V současné době existuje na trhu česká firma DCIT, a.s., která vyvíjí software PROVYS TV Office. Tento software je modulární informační systém pro řízení televizních stanic (15). Hlavní funkcí tohoto softwaru je odbavování, podružná funkce je archivace. Tento software využívají celoplošné televize v celém světě (16). V ČR software PROVYS využívá např. Česká televize nebo TV Nova. Ve světě tento software využívá např. slovinská veřejnoprávní televize nebo americká televizní stanice AMC. Jde o licencovaný a komerční software.

Obrázek č. 9: Snímek z aplikace Provys TV Office
(Zdroj: Vlastní zpracování dle: 16)

2.2.2 Open Knowledge Management

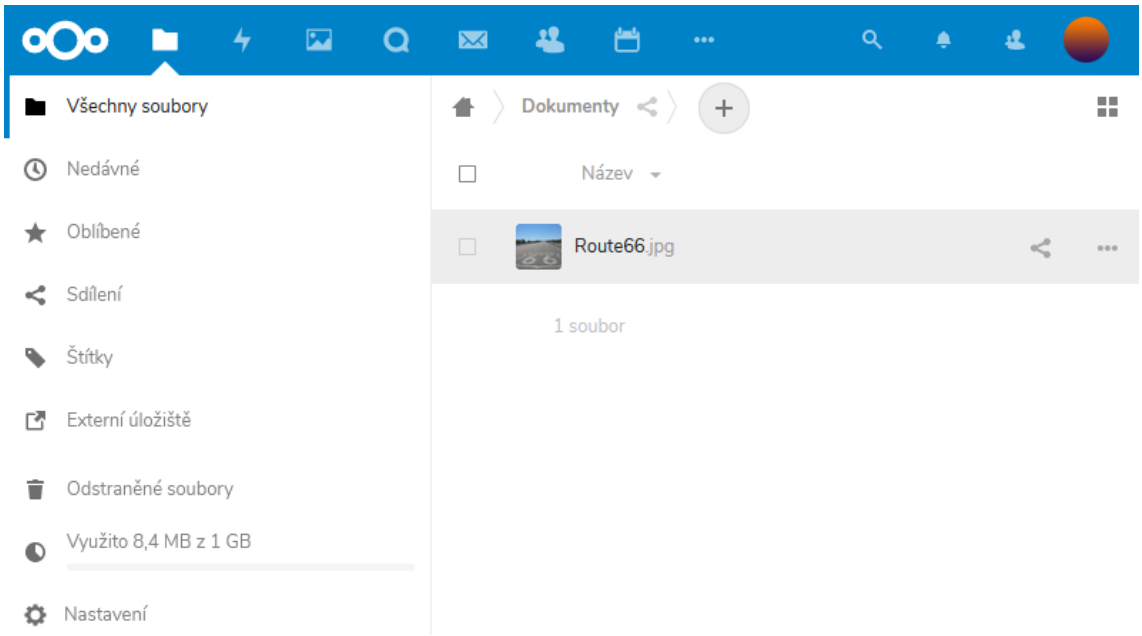
Open Knowledge Management, zkráceně jen Open KM, je systém pro správu dokumentů, který poskytuje webové rozhraní pro správu nespecifických souborů (18). Open KM potřebuje pro svůj běh server. Open KM je naprogramovaný v Javě a pro instalaci existuje průvodce v podobě spustitelného souboru s příponou *.jar*. Díky průvodci instalaci zvládne i méně pokročilý uživatel. Součástí instalace je webový server (Apache) a databázový systém (např. MySQL). Dále tento software podporuje správu metadat, přidávání poznámek k souborům, uživatelské účty, a další.



Obrázek č. 10: Snímek z aplikace Open Knowledge Management
(Zdroj: Vlastní zpracování dle: 17)

2.2.3 Nextcloud

Nextcloud je webová aplikace, která využívá architekturu klient-server, přičemž operační systém serveru může být Linux či Windows. Tato aplikace má otevřený zdrojový kód a kdokoli si může tuto aplikaci zdarma stáhnout (19). Pro svou činnost doporučuje použít databázi MySQL, ale podporuje i jiné typy databází (20). Nextcloud, díky široké paletě doplňků, poskytuje různorodé služby, např.: sdílený kalendář, přístup k souborům prostřednictvím služby WebDAV, mailový klient, konferenční videohovory a mnohé další služby. Přístup k této aplikace lze řídit uživatelskými účty. Prostředí je intuitivní a zvládne v něm pracovat i méně zdatný uživatel. Soubory lze uploadovat pouhým přetáhnutím do okna webového prohlížeče.



Obrázek č. 11: Snímek z aplikace Nextcloud
(Zdroj: Vlastní zpracování dle: 19)

2.2.4 Souhrn poznatků

Jak jíž bylo uvedeno, software Provys je licencovaný a běžně nedostupný software. To mi neumožňovalo podrobnou prohlídku funkcionalit tohoto softwaru. Další dvě aplikace jsou úmyslně vybrány tak, aby poskytovaly možnost přístupu několika uživatelů z klientských stanic s různými operačními systémy. Současně je důležité, aby koncový uživatel mohl provozovat server těchto služeb v místní síti, protože připojení k internetu v menších obcích ještě stále nedosahuje ideálních parametrů pro práci s objemným soubory. Toto kritérium splňuje jak OpenKM, tak i Nextcloud. OpenKM ukládá soubory

přímo do SQL databáze, zatímco Nextcloud umožňuje zpřístupnit zadaný adresář konkrétním uživatelům, kteří následně získají k tomuto adresáři přístup dle administrátorem nastavených pravidel. Výhodou OpenKM je to, že veškerá data se ukládají přímo do SQL databáze a díky tomu při nastavení zálohování postačí pracovat pouze s touto databází. Výhodou Nexcloudu je, že soubory jsou i nadále přístupné v adresáři na fyzickém disku. To však přináší i nevýhodu, kdy při řešení zálohování je nutné do úvahy zahrnout i tyto soubory. Obě řešení jsou nadále udržována a vychází k nim aktualizace.

2.3 Požadavky na nový informační systém

V této kapitole shrnuji poznatky, které jsem zjistil dřívějším analyzováním a doplním je o požadavky, které jsou potřebné z činnosti regionálních televizí.

Navrhovaný informační systém musí zajistit centralizaci dat a poskytovat různé úrovně přístupu, aby více uživatelů mohlo současně interagovat a manipulovat s tímto systémem. Současně je nutné, aby byl systém přístupný v místní síti, protože soubory, se kterými se bude pracovat, budou datově objemné v řádech gigabajtů. Prací se soubory se rozumí jejich importování do střihového programu za účelem vytvoření audiovizuálního obsahu. Dalším požadavkem je zajištění kompatibility s pracovními stanicemi uživatelů, jež mohou být provozovány pod různými operačními systémy. Tento požadavek již řeší uvažovaný koncept práce, tj. využití webového rozhraní s jazykem PHP.

Databáze musí být navržena tak, aby udržovala data o dvou typech archivů. Prvním typem je surový materiál. Tento materiál je přímo pořízený z videokamery bez jakýchkoli úprav. Druhým typem je materiál zpracovaný, přičemž vychází z materiálů surových. V databázi musí být uvedeno, z jakých surových materiálů se jeden zpracovaný skládá. Dále je důležité v datovém modelu zahrnout možnost přidání více úhlů záběru pro surové materiály, protože se v praxi často stává, že se jedna událost dokumentuje více než jednou videokamerou. Obdobně u zpracovaných materiálů se musí počítat s variantou přidání více verzí zpracování, např. jeden soubor s infografikou jako jedna verze a současně tentýž soubor bez infografiky jako druhou verzi. V neposlední řadě je nutné zajistit, aby existovala možnost přiřazení materiálů uživatelsky definovaným kategoriím a podkategoriím. Tímto jsou pokryty základní funkcionality týkající se datového modelu.

3 VLASTNÍ NÁVRH ŘEŠENÍ

V této kapitole se zabývám vlastním návrhem řešení problémů, které jsem identifikoval a popsal v analýze současného stavu. Při zpracovávání řešení vycházím z teoretických východisek uvedených v první kapitole. Nejprve však uvádím, z jakých dílčích částí se řešení sestává a následně blíže specifikuji jednotlivé součásti.

3.1 Koncept a dílčí části řešení

Návrh řešení musí vyhovovat všem požadavkům, které byly zjištěny v předchozí kapitole. Informační systém je realizován ve webovém prostředí a pro svoji činnost potřebuje libovolný webový server (např. Apache, Nginx) s podporou skriptovacího jazyka PHP. Na operačním systému serveru nezáleží. Dílčí části, ze kterých se návrh řešení skládá, jsou uvedeny níže:

1) Návrh datového modelu v databázi MySQL, se kterou bude komunikovat jazyk PHP.
2) Souvislost mezi úložištěm dat a databází.
3) Identifikace procesů, které probíhají v IS a jejich popis.
4) Využití Bootstrapu a zařazení jednotlivých HTML tagů do tříd, dle CSS.
5) Podpůrné funkce v jazyce Javascript.

3.2 Předpoklady

Tato práce je zaměřena pouze na návrh řešení informačního systému, který vychází z následujících předpokladů:

- Existuje vhodné úložiště pro ukládání videosouborů.
- Existuje funkční, tj. nainstalovaný a nakonfigurovaný webový server s jazykem PHP a databází MySQL.
- Existuje spojení mezi úložištěm a webovým serverem a také mezi úložištěm a uživateli.

3.3 Návrh datového modelu

Nejprve je nutné vymezit relace, které budou pro tento záměr funkčně nezbytné. Rovněž se určí primární a cizí klíče v relacích a také typy vazeb mezi relacemi včetně relačních klíčů.

Dle platné definice relace je nutné mít v každé relaci primární klíč (PK) a zajistit tak entitní integritu. V relaci může existovat několik kandidátních klíčů, ale pouze jeden z nich se může stát klíčem primárním. Z těchto kandidátních klíčů se provádí výběr PK tak, aby uvažovaný klíč byl jednoznačný a současně minimální. Význam podmínky vyžadující jednoznačnost spočívá v zajištění unikátnosti každé n-tice v rámci celé relace. Druhá podmínka vyjadřuje, že žádný atribut není možné vypustit, aniž by se porušilo pravidlo první. (3, str. 29)

Nyní začnu s definováním relace, která bude sdružovat údaje o surových videozáznamech. Název této relace je *surovy*, čímž se zkráceně míní surový materiál. Schéma relace se skládá z atributů uvedených ve druhém sloupci v tabulce níže.

Tabulka č. 3: Zobrazení relace se surovými materiály
(Zdroj: Vlastní zpracování)

Název atributu	**Popis**
datum_vytvoreni	Datum a čas, kdy byl záznam v IS vytvořen
datum_natoceni	Datum a čas, kdy byl příslušný videosoubor natočen
nazev	Název záznamu vystihující jeho obsah
misto_natoceni	Místa, ze kterých byl videosoubor pořízen
poznamky	Další doplňující prostor pro uživatele
klicova_slova	Uživatelem vypsaná klíčová slova
datum_zmeny	Datum poslední změny záznamu

Názvy atributů úmyslně uvádím bez diakritiky a bez mezer, protože následná práce s těmito textovými řetězci bude při sestavování SQL dotazů jednodušší.

Dále je nezbytné určit datové typy jednotlivých atributů. U data natočení a data vytvoření zvolím datový typ DATETIME. Jak již název tohoto datového typu napovídá, půjde o datum a čas v rámci jednoho atributu. V tomto datovém typu je přesnost času naprosto dostačující. Sekundy lze vyčíslit až na 6 desetinných míst, tzn., že zde hovoříme o přesnosti v mikrosekundách. Pro účely tohoto návrhu využívám pouze 3 desetinná místa.

Z atributů v tabulce č. 3 lze za kandidátní klíče považovat pouze datum vytvoření, neboť tento atribut, jako jediný, náležitě vyhovuje podmínkám PK. Jednoznačná bude vždy každá n-tice, protože čas plyne. I přesto není počet kandidátních klíčů konečný. Existuje možnost si vytvořit umělý PK. Takto uměle vytvořený PK se obvykle označuje zkratkou ID (z angl. slova *identification*) a někdy se doplňuje i přípona s upřesněním, co toto ID identifikuje. V tomto případě není potřeba vytvářet umělý PK, protože pro tuto relaci postačí atribut *datum_vytvoreni*. Je silně nepravděpodobné, že by v relaci vznikly buď dvě anebo více stejných n-tic z hlediska milisekund, právě když bude aktivních několik desítek uživatelů.

Následná manipulace s takto uloženými údaji bude více komfortní, než kdyby k uložení datu a času byly zvoleny dva atributy odděleně. V případě potřeby manipulace pouze s časem, lze využít funkci TIME(), přičemž vstupní proměnnou může být hodnota DATETIME. Funkce pak vrátí pouze čas. Úplně stejně lze využít funkci DATE(). MySQL má podobných funkcí mnohem víc – například: HOUR(), DAY(), atd… Další menší výhodou tohoto datového typu je například použití při chronologickém řazení n-tic v relaci v rámci SQL dotazu. S tímto datovým typem stačí v SQL dotazu napsat například: ORDER BY *nazev_atributu* ASC, zatímco v případě použití dvou atributů (*datum* a *čas*) by bylo ještě potřeba, pro úplné chronologické seřazení, zapsat zvlášť *datum* a následně *čas*. O výhodu se také jedná z hlediska fyzického místa na disku, které databáze MySQL potřebuje k uložení. V případě dvou oddělených atributů (*datum* a *čas*) je celková velikost 6 bajtů, přičemž datum má 3 bajty a čas také 3 bajty. Datum a čas v jednom datovém typu vyžaduje celkem 5 bajtů. Čas v těchto případech je uveden pouze s přesností na sekundy. Je-li potřeba uložit jednotky menší než sekundy, pak se nároky na místo zvyšují, ale v omezeném intervalu. Maximální přípustné dodatečné dynamické navýšení může být od nuly do třech bajtů. Tento rozsah záleží na počtu desetinných míst u sekund.

V následující tabulce níže jsou k jednotlivým atributům přiřazeny datové typy a také jejich délky. Délce datového typu se rozumí maximální počet míst, které zadaný údaj může využít.

Tabulka č. 4: Zobrazení relace se surovými materiály 2
(Zdroj: Vlastní zpracování)

Název atributu	Datový typ	Délka
datum_vytvoreni_PK	DATETIME	
datum_natoceni	DATETIME	
nazev	VARCHAR	255
misto_natoceni	VARCHAR	255
poznamky	VARCHAR	255
klicova_slova	VARCHAR	255
datum_zmeny	DATETIME	

Délky u dat (1.p. datum) jsou již definovány implicitně datovým typem. Místo natočení používá datový typ VARCHAR, což je zkratka dvou anglických slov *variable* a *character*. V překladu se jedná o proměnlivou délku textového řetězce, kdy při vytváření atributu v relaci se definuje maximální délka řetězce. U atributu *nazev* to v tomto konkrétním případě znamená, že lze uložit nula až 255 znaků. Výhodou tohoto datového typu je, že existuje přímá úměra mezi délkou vloženého řetězce a počtem bajtů, které využije na disku. Je-li zvolená maximální hodnota 255, pak se potřebný počet bajtů k uložení vypočítá jako součet znaků v řetězci plus jeden bajt. Změna nastává od dvou set padesátého šestého znaku, kdy se konstantě přičítá místo jednoho bajtu, dva bajty. Z tohoto důvodu lze usoudit, že při vytváření databáze není nutné dopodrobna zkoumat maximální možné délky všech atributů, ale postačí si promyslet, zda bude potřeba ukládat více či méně než 255 znaků v řetězci. Nastane-li při návrhu databáze situace, kdy by mohlo dojít k překročení maximální velikosti jedné n-tice na více než 65 535 bajtů, pak je nutné podrobně uvážit všechny atributy, jejich datové typy, délky i kódování řetězců,

aby k tomuto překročení nedošlo. Součástí relace surových materiálů je i možnost přidání klíčových slov. Zde se nabízí několik možných řešení, jak klíčová slova zahrnout do návrhu. První nejjednodušší řešení je přidaný atribut přímo v tabulce surových materiálů. Do tohoto atributu bude moct uživatel vkládat příslušná slova, jakkoli oddělená. Následné vyhledávání bude řešeno za pomocí operátoru LIKE, který bude součástí SQL dotazu u podmínky. Druhé, poněkud složitější, řešení spočívá ve využití nové relace. Vyjdeme z faktu, že mezi surovými materiály a klíčovými slovy existuje vztah N:M, protože jeden materiál může mít několik slov a jedno slovo může mít několik materiálů. Následná režie s takto definovanými klíčovými slovy bude zvýšená, neboť u poměru N:M mezi relacemi je nezbytné provést dekompozici a také přidat nástroj pro správu klíčových slov. Zvýšená režie se týká nejen programátora, ale i koncového uživatele. Toto, byť komplikovanější řešení, má bezesporu výhodu, že jím lze dosáhnout větší úspory místa na disku, protože se předejde redundanci již jednou uložených slov.

Základní atributy surových materiálů jsou vymezeny. Relaci se surovými materiály ještě doplním o další čtyři atributy. Prvním z doplňujících atributů bude připojení cizího klíče zaměstnanců, aby bylo možné identifikovat, který zaměstnanec příslušný záznam vytvořil. Další atribut, s názvem kategorie, bude rovněž s využitím cizího klíče, ke kterému bude přiřazena nová relace, kterou využiji jako číselník. Ke zbývajícímu třetímu atributu bude, podobně jako ke druhému, vytvořena nová relace s názvem *umisteni* opět ve funkci číselníku. V tabulce č. 5 je uvedena konečná podoba relace, která bude uchovávat údaje o surových materiálech.

Tabulka č. 5: Zobrazení relace se surovými materiály 3
(Zdroj: Vlastní zpracování)

Název atributu	**Dat. typ (délka)**	**Typ klíče**	**Poznámka**
datum_vytvoreni_PK	DATETIME	PK	Zadává systém
datum_natoceni	DATETIME		Zadává uživatel
nazev	VARCHAR (255)		
misto_natoceni	VARCHAR (255)		
poznamky	VARCHAR (255)		
klicova_slova	VARCHAR (255)		Klíčová slova

datum_zmeny	DATETIME		Datum poslední změny
id_kategorie_CK	*	CK	Připojení na číselník
vytvoril_CK	*	CK	Zaměstnanec, který vytvořil záznam
id_umisteni_CK	*	CK	

Pozn.: Datové typy u cizích klíčů zatím nejsou definovány, protože datové typy atributů v této relaci musí být shodné s dat. typy odpovídajících atributů relací, které dosud neexistují.

3.3.1 Úhly záběrů

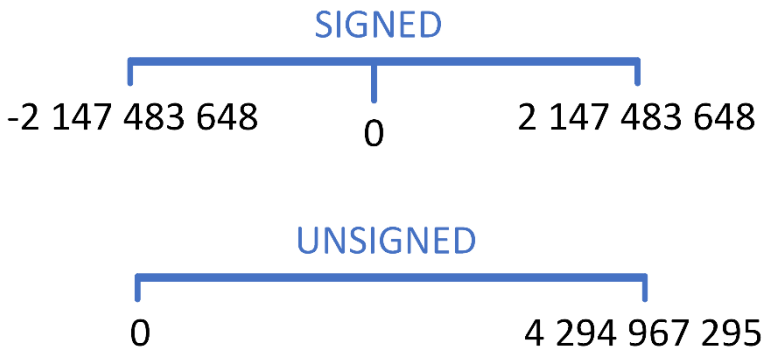
Dle analýzy současného stavu je kladen požadavek i na možnost zahrnutí více než jednoho úhlu záběru. Pro tento účel vytvořím novou relaci s názvem *uhly_zaberu*. Obsah této relace se všemi náležitostmi je v tabulce níže.

Tabulka č. 6: Zobrazení relace s úhly záběru
(Zdroj: Vlastní zpracování)

Název atributu	**Dat. typ**	**Popis**
id_uhlu	INT (10)	Umělý PK
datum_vytvoreni_CK	DATETIME	Připojený CK z relace surovy
popis_uhlu	VARCHAR (255)	Pro uživatele
kameraman_CK	*	

Mezi relací *uhly_zaberu* a relací se surovými materiály bude existovat vztah 1:N, protože jeden surový materiál může obsahovat několik úhlů záběrů. Datový typ u kameramana je s hvězdičkou, protože relaci obsahující údaje o zaměstnancích popíšu později.

Umělý PK využívá datový typ Integer, což po překladu z angličtiny znamená: celé číslo. Do datového typu Integer lze ve výchozím nastavení uložit číslo přibližně od minus dvou miliard až po plus dvě miliardy. Jsou-li celá čísla využita jako primární klíč, pak není potřeba využívat zápornou část intervalu. Nastavením vlastnosti *unsigned*, což v překladu vyjadřuje bez znaménka, lze dosáhnout posunu na pomyslné číselné ose, takže se možný interval přesouvá do kladných celých čísel včetně nuly. Konečná velikost intervalu s takto nastavenou vlastností je od nuly do přibližně čtyř miliard. Rozdíl v použití vlastností *signed* a *unsigned* je graficky znázorněn na obrázku níže.



Obrázek č. 12: Porovnání rozdílu mezi vlastnostmi signed a unsigned
(Zdroj: Vlastní zpracování)

Evidence surových materiálů je již kromě číselníků dokončená. Proto přejdu k evidenci zpracovaných materiálů. Pro tento účel využiji novou relaci s názvem *zpracovany*, což znamená zpracovaný materiál. Konečná podoba této relace je zobrazena pomocí tabulky níže.

Tabulka č. 7: Zobrazení relace se zpracovanými materiály
(Zdroj: Vlastní zpracování)

Název atributu	**Dat. typ**	**Typ klíče**	**Popis**
datum_vytvoreni	DATETIME	PK	Datum vytvoření
datum_zpracování	DATETIME		Datum zpracování
nazev	VARCHAR (255)	-	Název
poznamky	VARCHAR (255)	-	Poznámky
klicova_slova	VARCHAR (255)	-	Klíčová slova
datum_zmeny	DATETIME	-	Datum poslední změny
vytvoril_CK	*	CK	Vazba na další relaci
id_podkategorie_CK	*	CK	Vazba na číselník

V tuto chvíli jsou připraveny dvě důležité relace, tj. *surovy* a *zpracovany*, čímž jsou zajištěny dva základní typy archivů. Návrh je nutné ještě rozšířit o možnost přiřazení několika surových materiálů k jednomu zpracovanému a také, že z jednoho surového materiálu může vzniknout několik zpracovaných. Toho lze docílit využitím kardinality

vztahu N:M. Po provedení dekompozice vzniká nová relace s názvem *sur_zprac*, ve které jsou dva atributy s cizími klíči dvou výše zmíněných relací.

Tabulka č. 8: Zobrazení propojovací relace surových a zpracovaných materiálů
(Zdroj: Vlastní zpracování)

Název atributu	**Dat. typ**	**Typ klíče**	**Popis**
datum_vytvoreni_zprac_CK	DATETIME	CK	Datum vytvoření zpracovaného mat.
datum_vytvoreni_sur_CK	DATETIME	CK	Datum vytvoření surového mat.

Dále je, dle požadavků, potřeba, aby bylo možné ukládat více variant zpracovaných materiálů. Toho lze docílit přidáním další relace s názvem *verze_zpracovani*, která bude připojena k relaci *zpracovany* vztahem 1:N.

Tabulka č. 9: Zobrazení relace s verzemi zpracovaných materiálů
(Zdroj: Vlastní zpracování)

Název atributu	**Dat. typ**	**Typ klíče**	**Popis**
id_verze	INT(10)	PK	Umělý primární klíč
nazev	VARCHAR(255)		Název zpracované verze
datum_vytvoreni_zprac	DATETIME	CK	

Pro vyhovění dalšímu požadavku, který spočívá v umožnění přístupu několika uživatelů, je zapotřebí vytvořit další relaci, která bude sdružovat údaje o zaměstnancích.

Tabulka č. 10: Zobrazení relace se zaměstnaci
(Zdroj: Vlastní zpracování)

Název atributu	Datový typ	délka	Popis
id_zam	INT	10	Identifikační číslo zaměstnance
jmeno	VARCHAR	45	Jméno
prijmeni	VARCHAR	40	Příjmení
dat_nar	DATE		Datum narození
e_mail	VARCHAR	200	
telefon	VARCHAR	12	Telefonní číslo zaměstnance
heslo	TEXT		Zde se ukládá heslo jako řetězec znaků pro účel přihlášení
heslo_dat	DATETIME		Datum poslední změny hesla
adresa	VARCHAR	255	Adresa bydliště zaměstnance
dat_vytvoreni	DATETIME		Datum a čas vytvoření záznamu
id_opravneni	*	*	Připojení k číselníku s názvy oprávnění
id_pohlavi	*	*	Připojení k číselníku s názvy pohlaví

Pro celou adresu je vymezen pouze jeden atribut. Ačkoli se zdá, že takové rozhodnutí může vést k rozporu s první normální formou, v tomto případě tomu tak není. S adresou se nebude nadále pracovat.

3.3.2 Číselníky

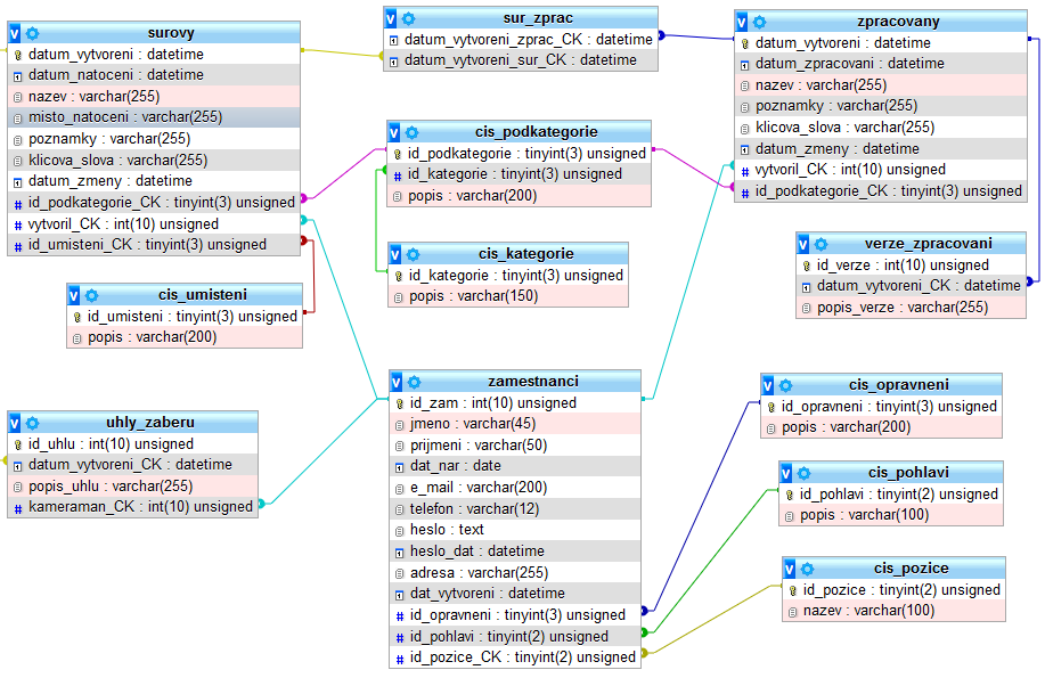
Číselníků bude zapotřebí celkem šest. První dva číselníky jsou pro kategorie a podkategorie surových a zpracovaných materiálů. Ve třetím číselníku jsou uvedeny názvy oprávnění. Čtvrtý slouží pro volbu pohlaví. Předposlední číselník se využívá k určení, na které pozici pracují zaměstnanci. Tento číselník najde uplatnění během přidávání nového úhlu záběru, kdy systém vypíše pouze ty zaměstnance, kteří jsou na pozici kameramana.

Uchovávání údajů o geografické pozici pořízení materiálů vyřeším za pomoci zbývajícího číselníku. Místo natočení v relaci *surovy* je atribut, do kterého může uživatel zadat libovolný textový řetězec. Tento číselník bude obecně definovat místo pořízení. Způsob obecných definic bude záležet na příslušném administrátorovi či společnosti, jak se společně dohodnou na zápisu evidence. Číselník může obsahovat obecná podstatná jména, např.: *muzeum, galerie,* atp. Díky tomuto kombinovaném způsobu evidence bude při vyhledávání dosaženo lepších výsledků, neboť se lze primárně řídit obecným místem a až v atributu *misto_natoceni* bude uživatelem upřesněno, na kterém místě byl videosoubor pořízen. Všechny názvy relací sloužící jako číselníky začínají 3 písmeny *cis*.

3.3.3 Výsledný ER diagram

Výsledný entito-relační diagram byl zpracován v prostředí phpMyAdmin. Jedná se o tenkého klienta, který je k dispozici ve webovém prohlížeči. Prostřednictvím tohoto klienta je možné komunikovat s databází MySQL a provádět různé operace s databázemi, relacemi a dalšími potřebnými úkony. Alternativní metodou je např. MySQL Workbench což je tlustý klient, který lze rovněž připojit k databázi MySQL.

Jednotlivé obdélníky zachycují strukturu relací. Nejprve je uveden název atributu, za dvojtečkou následuje datový typ atributu a v kulaté závorce je uvedena délka datového typu. Na některých místech se vyskytuje anglický pojem *unsigned*, který se váže na číselné datové typy. Význam je blíže vysvětlen na obrázku č. 12. V záhlaví obdélníku, tj. tučný text se světle modrým pozadím, je uveden název relace. Relace jsou vzájemně propojeny barevnými čarami. Tyto čáry znázorňují integritní omezení vztahu mezi relacemi. Každá čára má na jedné straně půlkruh a na druhé straně tečku. Nechť integritní omezení vztahu je 1:N, pak půlkruh znamená stranu N a tečka stranu 1. Vztahy mezi relacemi (čáry) jsou barevné pouze kvůli přehlednosti.



Obrázek č. 13: ER diagram
(Zdroj: Vlastní zpracování)

3.3.4 Omezení datového modelu

Nejprve je potřeba definovat omezení datového modelu už jen z hlediska určení maximální délky u datových typů všech relací. To znamená, že např. v relaci zaměstnanců není možné v atributu *jmeno* uložit textový řetězec delší než 45 znaků, atp. Další omezení je tvořeno nejen maximální délkou řetězce, ale i způsobem využití atributu *adresa*. Tento atribut sdružuje plnohodnotné údaje o místě bydliště zaměstnance, z čehož plyne, že není možné jednoduše např. vyhledávat všechny zaměstnance z konkrétní obce. Aby tato akce byla realizovatelná, převezme tuto režii funkční model. Tento model při registraci nového uživatele vyzve, aby zadal do separátních textových míst dílčí údaje adresy. Dále před zápisem do databáze funkční model sestaví ze zadaných dílčích údajů plnohodnotný řetězec obsahující adresu podle předem definované masky. Tímto krokem bude zajištěna konzistence dat a v případě potřeby bude možné využít v jazyce PHP funkce, díky nimž lze manipulovat s textovými řetězci. Tento model nepředpokládá, že by tyto potřeby musely být naplněny, a proto bylo přistoupeno ke zjednodušení datového modelu v tomto ohledu. Do datového modelu není možné uložit více než jedno telefonní číslo zaměstnance a více než jednu e-mailovou adresu.

Další omezení spočívá v neúplném zaznamenávání historie používání celého systému. Jak je z návrhu datového modelu patrné, byly vybrány pouze některé události, při kterých

se ukládá alespoň datum poslední změny přímo v relacích, kterých se změna týkala. Jde o relace: *surovy, zpracovany, zamestnanci*. V relaci *zamestnanci* je připraven prostor pouze k uložení data poslední změny hesla. Současně není možné se z IS dozvědět, kdo tuto poslední změnu provedl. Vzhledem k uvažovanému účelu použití, které je zejména pro regionální televize, kde se vyskytují nejvýše nízké desítky zaměstnanců, by tato omezení neměla mít pro používání zásadní dopad.

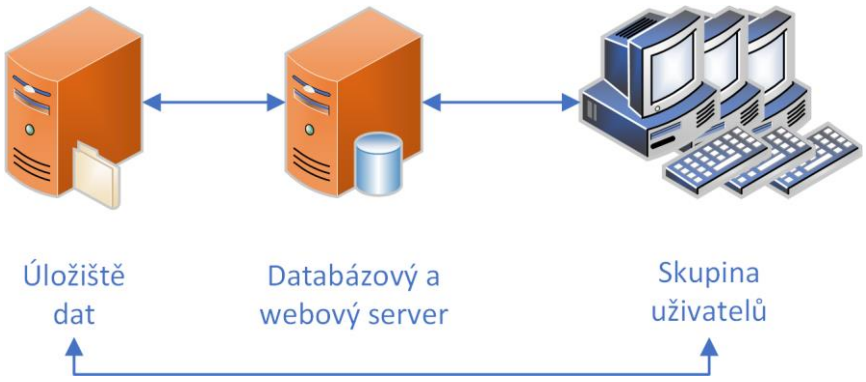
3.4 Komunikace s datovým úložištěm

V této kapitole je popsána vazba mezi datovým úložištěm a navrženou databází.

Z návrhu databáze je patrné, že v žádné relaci nebyl použit datový typ Blob (ani jeho žádná varianta Tinyblob či Longblob), který lze využít k ukládání binárních dat. Videosoubory budou ukládány prostřednictvím webového serveru přímo na datové úložiště. Tato metoda je jednodušší z hlediska použití, protože s videosoubory je nutné pracovat ve střihových a jiných editačních programech. Kdyby byla binární data ukládána pouze v databázi, nebylo by možné k datům přímo přistupovat a každý soubor by se musel stáhnout z webu na místní disk a po dokončení práce použité soubory opět smazat.

Za existence předpokladů, definovaných v bodě 3.2, lze přistoupit k upřesnění, jak by celý informační systém fungoval. Nechť datové úložiště, databázový a webový server jsou všechny s operačním systémem Microsoft Windows Server 2008 a novější. Minimální verze 2008 je důležitá, protože na starší verze není možné nainstalovat MySQL server.

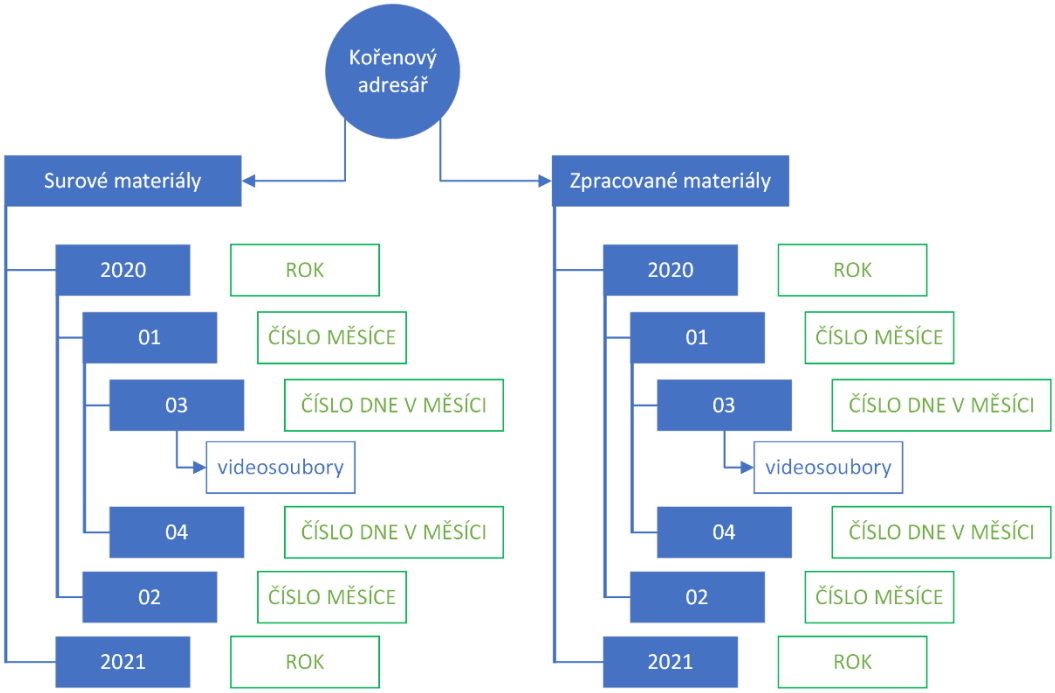
Na datovém úložišti je spuštěno sdílení souborů prostřednictvím protokolu SMB. Toto sdílení je nastavené pro dva uživatele. První uživatel má oprávnění číst a zapisovat, druhý uživatel smí pouze číst. Ve webovém serveru se namapuje síťová jednotka s přihlášeným uživatelem, který má oprávnění i zapisovat. Tuto síťovou jednotku bude následně využívat webový server, konkrétně PHP skript. Znázornění předpokládaného použití je zobrazeno na obrázku č. 14.



Obrázek č. 14: Topologie možného využití IS
(Zdroj: Vlastní zpracování)

Jedná se pouze o navrhované použití. Plně záleží na koncovém uživateli, který bude tento IS implementovat, zda oddělí datové úložiště od webového serveru či nikoli. Je vyžadován nejméně jeden server.

PHP skript bude spravovat adresář videosouborů. Primární rozdělení adresáře bude na 2 hlavní kategorie: surové a zpracované materiály. Každá z těchto dvou hlavních kategorií bude mít chronologicky tvořené podkategorie. Tato hierarchie je přehledněji zobrazena na obrázku níže.



Obrázek č. 15: Hierarchická struktura kořenového adresáře
(Zdroj: Vlastní zpracování)

Strom zpracovaných materiálů je stejný jako strom surových materiálů. Tato hierarchie je potřebná, protože ke kořenovému adresáři budou mít přímý, avšak omezený, přístup zaměstnanci, aby s těmito soubory mohli dále pracovat – pouze číst. Zápis do tohoto adresáře bude možný pouze prostřednictvím webového rozhraní, aby uživatelé nenarušili konzistenci dat.

Vazba mezi videosoubory, které jsou uloženy na datovém úložišti, a databází je založena na názvu souboru. Skript obsahuje proces, který podle předem definované masky přejmenuje nahraný soubor a současně analyzuje stav adresářů a soubor zařadí do odpovídajícího adresáře.

3.5 Návrh a popis obecného řešení IS

Celý informační systém má jednotnou strukturu a skládá se ze tří částí:

1) přidání nového surového/zpracovaného materiálu,
2) vyhledání záznamu,
3) administrace (zahrnuje např.: nastavení cílové složky uploadu, možnost aktivace anonymního přístupu, správa uživatelů, nastavení logování, atp.).

Tyto části jsou hlavními položkami menu. Např. při vyhledání souboru bude mít přihlášený uživatel možnost záznam odstranit nebo upravit. Z tohoto důvodu je nutné ještě popsat procesy, které se v tomto IS vyskytují. Procesy v IS:

- Vytvoření, editace a odstranění se tykají:
    - surových a zpracovaných materiálů,
    - uživatelů (zaměstnanců),
    - kategorií i podkategorií.
- Přihlášení / Odhlášení

Nedílnou součástí těchto možností je také přihlášení. Přihlášení má 3 druhy autorizace:

- administrátor – plnohodnotný přístup ke všem čtyřem částem,
- běžný uživatel – vyhledávat, přidávat nové, upravovat, odstraňovat, spravovat kategorie,
- anonymní uživatel – pouze vyhledávat.

Pro lepší orientaci v souborech informačního systému je struktura těchto souborů složena z jádra a čtyř podkategorií. Tyto podkategorie jsou:

- skripty zobrazitelné – zobrazují výstupy v HTML,
- skripty podpůrné – obsahují obecné funkce,
- skripty pro asynchronní volání PHP skriptů,
- soubor XML obsahující nastavení administrace.

Jako jádro je nazván hlavní soubor *index.php*, který obsahuje základní strukturu HTML a dynamicky generuje položky v menu. Aby bylo možné tyto položky zobrazit, využije se příkazu jazyka PHP *include*. Tento příkaz vloží obsah jiného souboru (např. s příponou *.html* či *.php*) do jádra. Díky tomuto řešení je možné zachovat přehlednost a modularitu. Skripty podpůrné lze volat příkazem *require*, který připojí jiný skript k jádru.

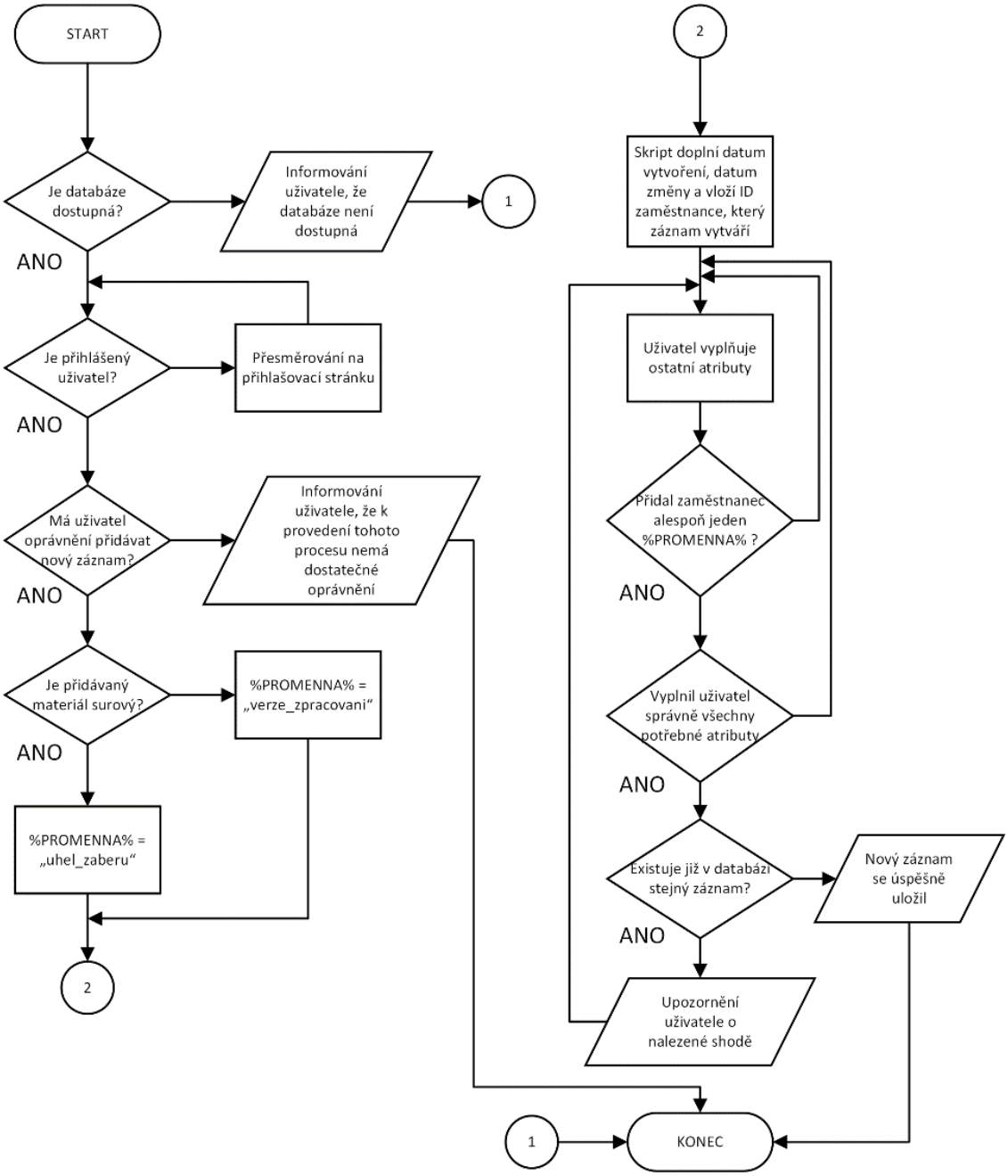
Vzhledem k tomu, že zobrazovaní jednotlivých částí IS zajišťuje příkaz *include*, je nutné každou stránku identifikovat a zajistit, aby ji bylo možné rozpoznat. Toho lze docílit využitím některé z metod PHP: GET nebo POST a definováním proměnné *stranka*. Hodnota této proměnné bude rovna textovému řetězci, jež bude obsahovat název volané stránky. Jádro následně vyhodnotí, zda volaný soubor existuje a pokud ano, zobrazí jej. V tomto případě lze přistoupit k využití metody GET, neboť nikterak nevadí, že se uživateli v navigačním řádku prohlížeče bude zobrazovat název proměnné a její hodnota. Naopak to může být kladem, protože si uživatel příslušnou stránku může uložit do záložek v prohlížeči. Nesmí se ovšem v jádře opomenout ošetřit vstupní hodnota této proměnné a při zadání neplatné hodnoty uživatele přesměrovat na výchozí stránku.

Administrace využívá jak databázi, tj. pro správu uživatelů, tak i strukturovaný XML soubor, ve kterém je uloženo nastavení IS. Toto nastavení je možné zálohovat, resp. provést export a s tím související import.

V podkapitolách níže popíši všechny důležité procesy tohoto IS pomocí vývojových diagramů.

3.5.1 Přidání nového záznamu

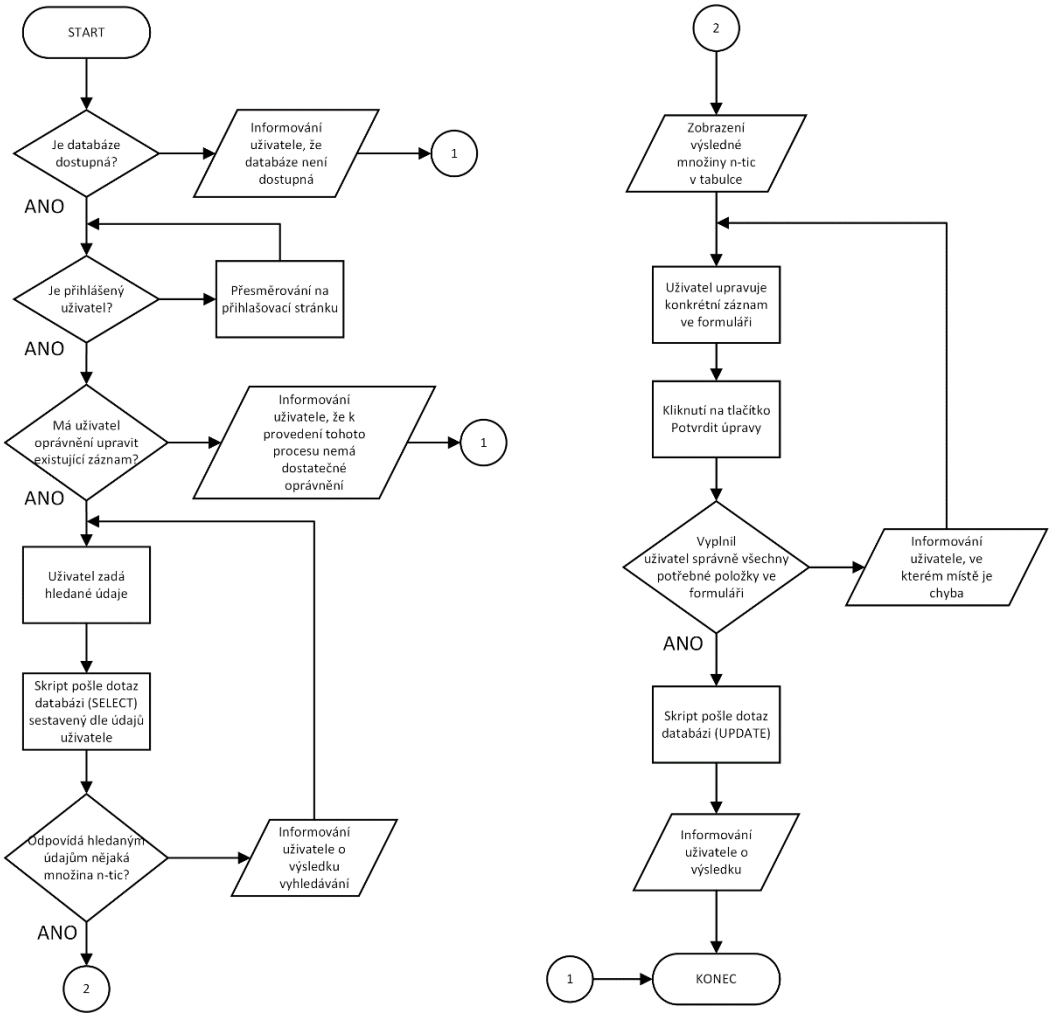
Tento proces začíná kontrolou, zda je dostupná databáze. Vznikne-li v tomto kroku nějaký problém, uživateli se zobrazí chybová hláška. Dále lze přistoupit ke kontrole, zda je nějaký uživatel přihlášen. Pokud ne, je přesměrován na přihlašovací stránku. Během přihlášení se ověřuje, zda má přihlášený uživatel oprávnění k provedení tohoto procesu. Pokud ano, smí uživatel zvolit, o jaký druh materiálu se jedná. Podle vkládaného typu materiálu se nová n-tice zapíše do příslušné relace. Datum vytvoření předvyplní PHP a uživatel jej nemůže změnit. V případě přidávání surového materiálu je nutné přidat alespoň jeden úhel záběru a obdobně u zpracovaného materiálu je nutné přidat alespoň jednu verzi zpracovaného materiálu. Správnost uživatelem vyplněných údajů ověřuje PHP. Alternativním způsobem by kontrola zadaných údajů mohla být řešena pomocí Javascriptu. Ovšem Javascript je možné v prohlížeči deaktivovat a pokud by data před vstupem do databáze nebyla řádně zkontrolována, došlo by k narušení konzistence dat. Jedná se tedy o negativní vliv a je nutné mu předejít. Součástí tohoto procesu je kontrola již existujících n-tic v relaci. V prvním případě, tj. relace se surovými materiály, se existence n-tic kontroluje v rámci tří atributů: *nazev*, *datum_natoceni* a *id_podkategorie*. Ve druhém případě, tj. relace se zpracovanými materiály, se kontrola realizuje stejným způsobem, akorát porovnávaným datem je datum zpracování. Při této kontrole je v podmínce SQL dotazu použit logický výraz AND, který pro nalezení shody zajistí, že musí být nalezena taková n-tice, která obsahuje stejné hodnoty ve všech třech atributech.



Obrázek č. 16: Vývojový diagram procesu přidání nového záznamu
(Zdroj: Vlastní zpracování)

3.5.2 Úprava existujícího záznamu

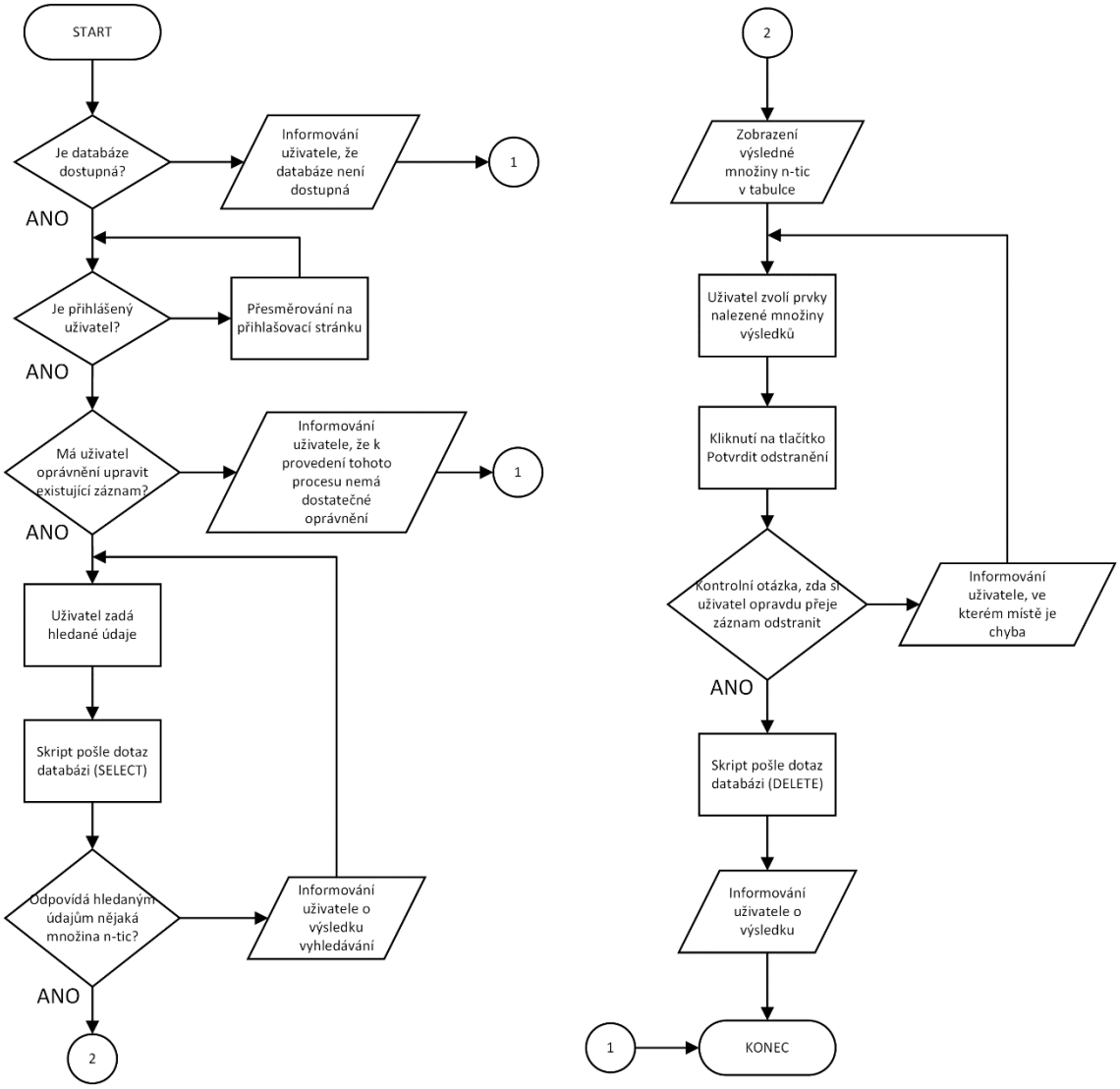
První tři kroky kontrol jsou totožné jako v předchozím procesu. Konkrétně se jedná o následující kontroly: dostupnost databáze, přihlášený uživatel, oprávnění uživatele. Dopadnou-li všechny kontroly s pozitivním výsledkem, pak skript pošle databázi SQL dotaz SELECT a požádá o hledanou množinu n-tic z relací. Tento SQL dotaz se skládá z uživatelem zadaných údajů. Byl-li nalezen výsledek, skript zobrazí data ve formuláři a toto zobrazení bude viditelně rozlišovat údaje upravitelné a neupravitelné. Neupravitelnými údaji jsou v případě surových a zpracovaných materiálů data vytvoření a autoři vytvoření. Upravitelnými údaji jsou všechny ostatní atributy, které relace obsahuje. I v tomto procesu se kontroluje správnost zadaných údajů. Jestli i tato kontrola dopadne úspěšně, dojde k zaslání dalšího SQL dotazu. Tentokrát se bude jednat o příkaz UPDATE, který aktualizuje příslušnou n-tici v relaci.



Obrázek č. 17: Vývojový diagram procesu úprava existujícího záznamu
(Zdroj: Vlastní zpracování)

3.5.3 Odstranění existujícího záznamu

Proces odstranění existujícího záznamu se od předchozího liší pouze ve změně SQL dotazu, kdy místo UPDATE se databázi posílá příkaz DELETE.

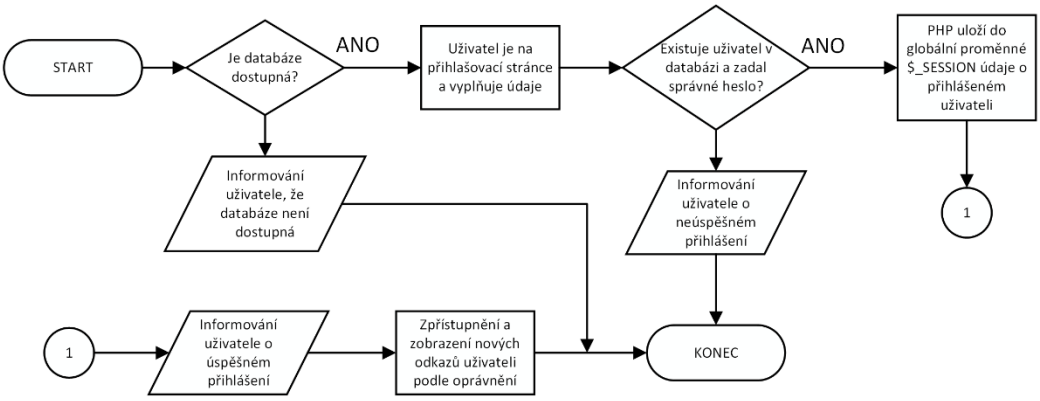


Obrázek č. 18: Vývojový diagram procesu odstranění existujícího záznamu
(Zdroj: Vlastní zpracování)

3.5.4 Proces přihlášení

Uživatel zadává své ID a heslo. Po zadání vstupních údajů uživatele dojde k ověření existence ID a hesla v databázi. V databázi je heslo uloženo jako řetězec znaků, jež byl vytvořen z běžného textu, resp. hesla ve zdrojové podobě, matematickým algoritmem. Pokud se porovnává heslo během přihlášení, skript znovu převede uživatelem vložené heslo do řetězce znaků a dochází k porovnání právě těchto dvou řetězců. Jedná se o standardní způsob ukládání hesel v databázi. Konkrétními druhy kryptografických algoritmů, sloužících k vytváření řetězců znaků, se tato práce blíže nezabývá. Je-li ověření

těchto dvou údajů v pořádku, může skript PHP definovat globální proměnnou *Sessions*. Tato proměnná je k dispozici v rámci všech ostatních stránek a je-li definována, znamená to, že je uživatel přihlášený. Oprávnění uživatele je uloženo v této proměnné. Díky těmto skutečnostem lze přizpůsobovat zobrazovaný obsah dle druhu autorizace, jak je popsáno v kapitole 3.5.



Obrázek č. 19: Vývojový diagram procesu přihlášení
(Zdroj: Vlastní zpracování)

3.5.5 Proces odhlášení

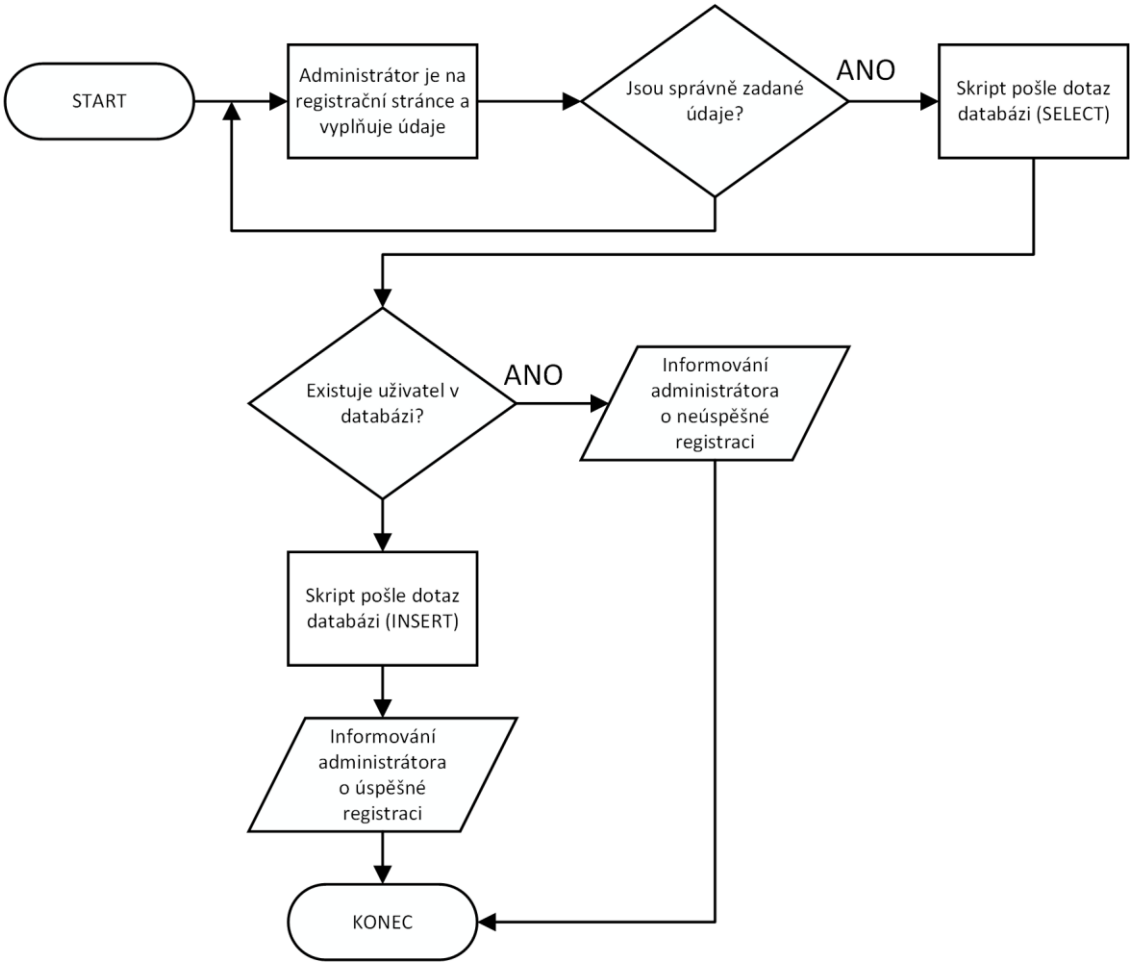
Tento proces bude provádět skript v samostatném souboru v rámci podpůrných skriptů. Jak již bylo řečeno u předchozího procesu, přihlášení zajišťuje globální proměnná *Sessions*. Odhlášení lze realizovat odstraněním proměnné.

```php
<?php
  session_start();
  session_unset();
  session_destroy();
  header("Location: ../index.php");
?>
```

Obrázek č. 20: Proces odhlášení uživatele v PHP
(Zdroj: Vlastní zpracování)

3.5.6 Proces registrace

Registraci nového uživatele smí provádět pouze administrátor. Tento krok by měl přispět k udržení konzistence dat v databázi z hlediska omezení datového modelu. Administrátor vyplní formulář a následně skript postupně zkontroluje všechny zadané údaje. Současně provede kontrolu, zda se právě registrovaný uživatel již v databázi nenachází, a to na základě těchto atributů: *jmeno, prijmeni, dat_nar* (datum narození) a *e_mail*.



Obrázek č. 21: Vývojový diagram procesu registrace
(Zdroj: Vlastní zpracování)

3.5.7 Proces správy souborů

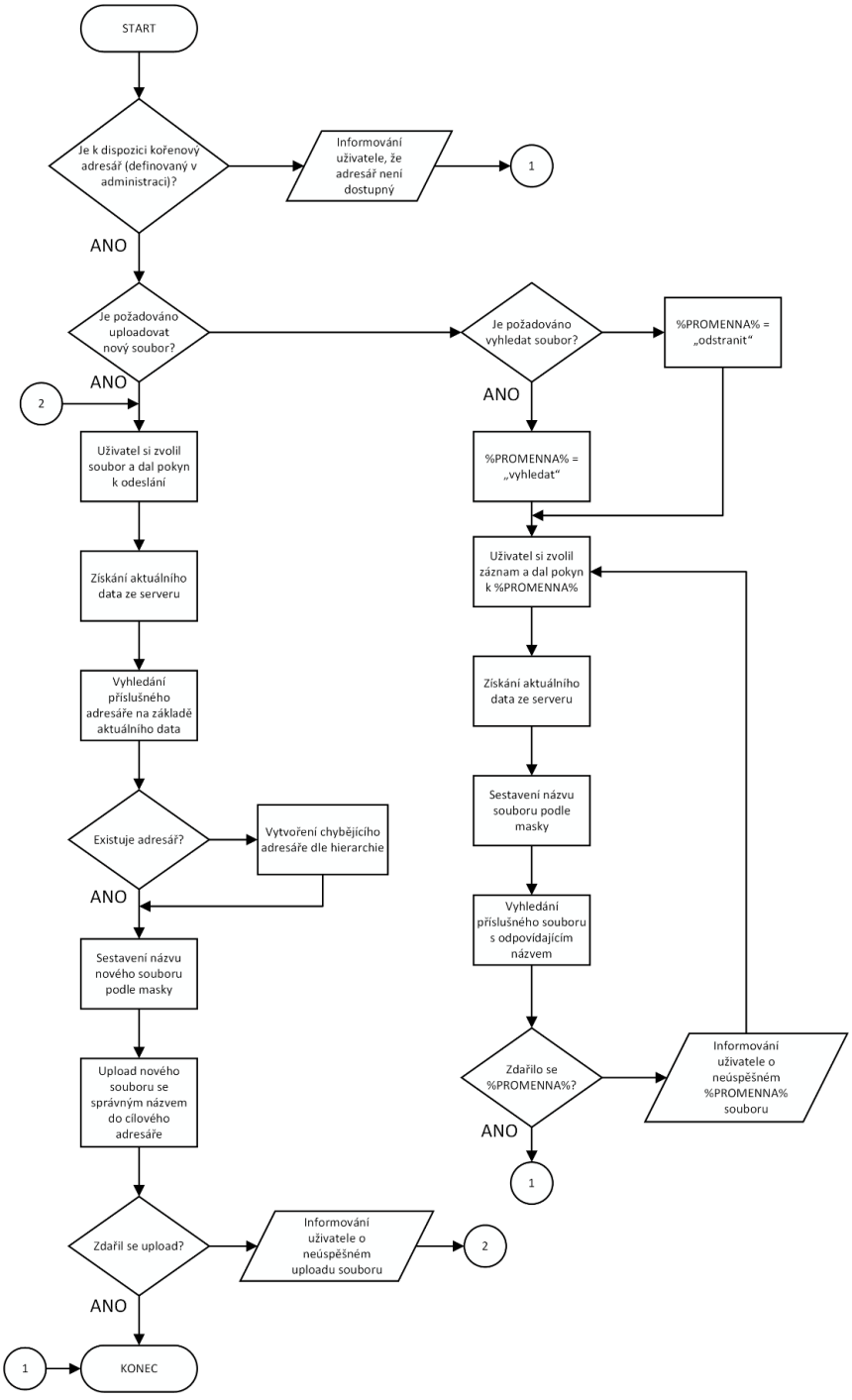
Tento proces zajišťuje propojení mezi datovým úložištěm a databází SQL. Jak již bylo řečeno, toto propojení existuje na základě přesně definované masky, podle které se generují názvy souborů. Tato maska se skládá z několika atributů n-tice.

Uvažovanou masku zobrazím pomocí zástupných písmen: YYYY-MM-DD-Z.PPP

Symboly YYYY vyjadřují rok, MM znamenají měsíc a DD je číslo dne v měsíci. Jedná se o standardní zápis data v databázích. Za poslední pomlčkou se nachází písmeno Z, které v případě surových materiálů určuje úhel záběru a v případě zpracovaných materiálů určuje verzi. Za tečkou je obvykle třípísmenná přípona. Maska obsahuje primární klíč dvou hlavních kategorií a současně primární klíč verzí a úhlů záběrů.

Nedílnou součástí tohoto procesu je také rozlišování, do kterého adresáře bude soubor zapsán. Rozlišování musí probíhat v souladu s hierarchií dle obrázku č. 15.

Pro získávání času z operačního systému se využívá PHP funkce *date()*, která podporuje široké možnosti vstupních i výstupních hodnot. Ve vývojovém diagramu je sloučen požadavek na vyhledání i odstranění souboru, neboť se jedná o velmi podobné operace.



Obrázek č. 22: Vývojový diagram procesu správy souborů
(Zdroj: Vlastní zpracování)

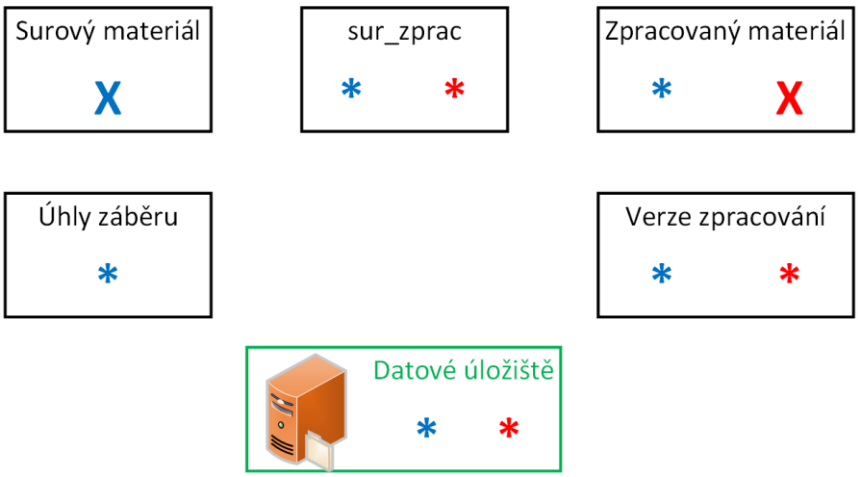
3.5.8 Vazby na procesy z hlediska operací

Procesy popsané výše obecně definují funkcionality informačního systému. Není z nich zcela zřejmé, jak konkrétně ovlivňují příslušné relace. Z tohoto důvodu je v této kapitole vysvětleno, na které relace mají procesy vliv.

Proces přidání nového záznamu vyžaduje kontrolu existence uživatele. Tato kontrola probíhá s relací zaměstnanci. Dále se dle druhu přidávaného materiálu využívají všechny nečíselníkové relace kromě zaměstnanců. V tomto procesu je každá z pěti relací ovlivněna přidáním jedné n-tice.

Proces úpravy existujícího záznamu ovlivní vždy pouze jednu n-tici upravované relace.

Proces odstranění existujícího záznamu je již komplikovanější. Záleží na relaci, ve které se uživatel rozhodne n-tici odebrat. Určí-li uživatel ke smazání n-tici ze surových materiálů, pak dojde ke kaskádovému odstranění odpovídajících n-tic ve všech ostatních nečíselníkových relací. Na obrázku níže je pomocí dvou barev zobrazen rozdíl dvou mazacích kaskád. První kaskáda se spouští při odebrání n-tice z relace se surovými materiály (modrá) a obdobně se vykoná druhá kaskáda, akorát při odebrání z relace zpracovaných materiálů (červená). Spuštění kaskád je znázorněno symbolem X a její dopad v ostatních relacích je vyjádřen symbolem *.



Obrázek č. 23: Grafické znázornění mazacích kaskád
(Zdroj: Vlastní zpracování)

Kromě relací, které jsou na obrázku znázorněny jako černé obdélníky, má mazací kaskáda vliv i na datové úložiště. Kaskáda má při mazání n-tic vliv vždy. Z uživatelského hlediska nezáleží, jak se s tímto IS pracuje. Tyto kaskády se spouští automaticky a jsou nezbytnou součástí pro udržení konzistence veškerých dat, tj. těch databázových i těch samostatně

na datovém úložišti uložených. Objem mazaných dat je přímo závislý na úrovni, ve které aktuálně uživatel pracuje. Na obrázku výše jsou zobrazeny dvě kaskády s největším dopadem, ke kterým může dojít. Na nižší úrovni, což je u úhlů záběrů a verzí zpracování, mají kaskády vliv už jen na datové úložiště. Datové úložiště je hierarchicky nejníže a žádný implementátor tohoto IS nesmí běžným uživatelům umožnit jakýkoli zásah do dat na této úrovni. Z tohoto důvodu byl také v kapitole 3.4 kladen požadavek na vytvoření dvou účtů s různými oprávněními na serveru s datovým úložištěm videosouborů, přičemž monopol pro manipulaci s daty na datovém úložišti má výhradně informační systém na webovém serveru.

3.5.9 Aplikace kaskád

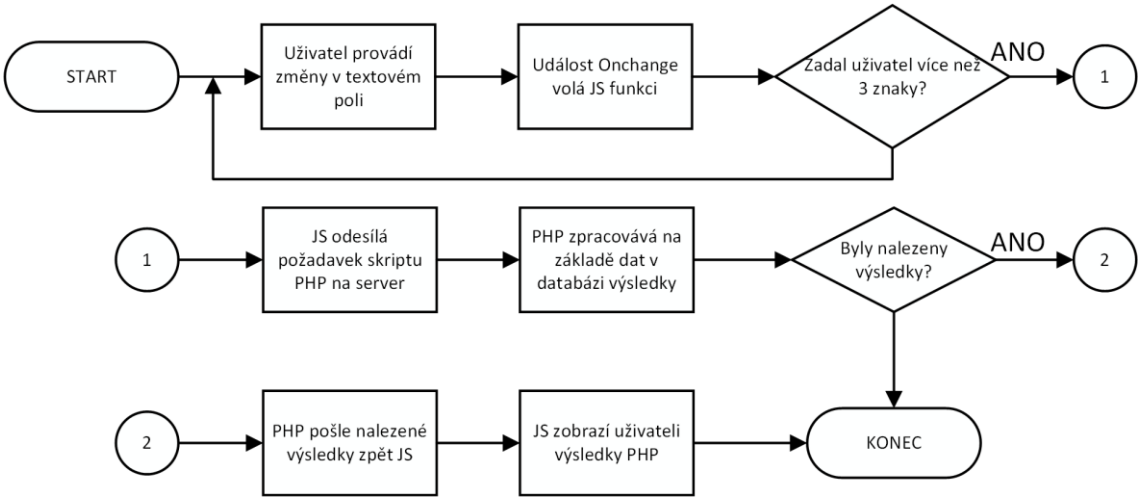
V předchozí podkapitole byla popsána problematika mazání n-tic z relací a jejich návaznosti na ostatní relace. Současně bylo uvedeno, že se tyto tzv. kaskády provádí automaticky. Automatické provádění kaskád bude předmětem řešení na následujících řádcích.

Databáze MySQL, stejně jako ostatní databázové systémy, nabízí možnost využití tzv. triggerů. Do češtiny tento anglický pojem lze přeložit jako spoušť. Tato možnost funguje na principu událostí. Když například přijde databázi dotaz s požadavkem insert, trigger může zareagovat, resp. vykonat posloupnost příkazů, a podle nově zadané hodnoty doplnit nějaký údaj do kterékoli relace. Jedná se o velmi užitečný nástroj, avšak v tomto prostředí neuplatnitelný, neboť je potřeba provádět úpravy nejen se samotnou databází, ale i s datovým úložištěm. Kvůli této skutečnosti tuto režii převezme jazyk PHP, aby mohly být kaskády kompletně zapsané na jednom místě.

3.5.10 Asynchronní volání

Pro zvýšení uživatelského komfortu při používání tohoto IS je zapotřebí využít i asynchronní volání. Jedná se o prostředek, kterým je možné spustit, prostřednictvím Javascriptu, jiný skript PHP na straně serveru. Tímto prostředkem lze dosáhnout větší dynamičnosti stránek. Uplatnění najde především při vyhledávání jako našeptávač. V Javascriptu se definuje tzv. EventListener, jež slouží k naslouchání událostí a čeká, jakmile nastavená událost nastane. V tomto případě existuje HTML tag Input typu text s atributem ID. K tomuto ID je v Javascriptu přiřazena událost Onchange. Nastane-li událost, což znamená, že došlo k přidání či odebrání textu v poli Input, pak se ihned spustí

připravená funkce. Tato funkce nejprve zjistí, kolik znaků se již v textovém poli Input nachází. Pokud již uživatel zadal více než tři znaky, dojde k asynchronnímu volání. Javascript uživatelem zadané znaky odešle na server skriptu PHP. PHP se připojí do databáze a vyhledá, zda se v příslušné relaci nachází nějaká n-tice odpovídající hledaným znakům a zpět Javascriptu pošle odpověď, zda bylo něco nalezeno. Je-li výsledek vyhledávání s nějakým nálezem, pak Javascript všechny nalezené výsledky nabídne uživateli. Na obrázku níže je pomocí vývojového diagramu podrobněji znázorněno asynchronní volání.

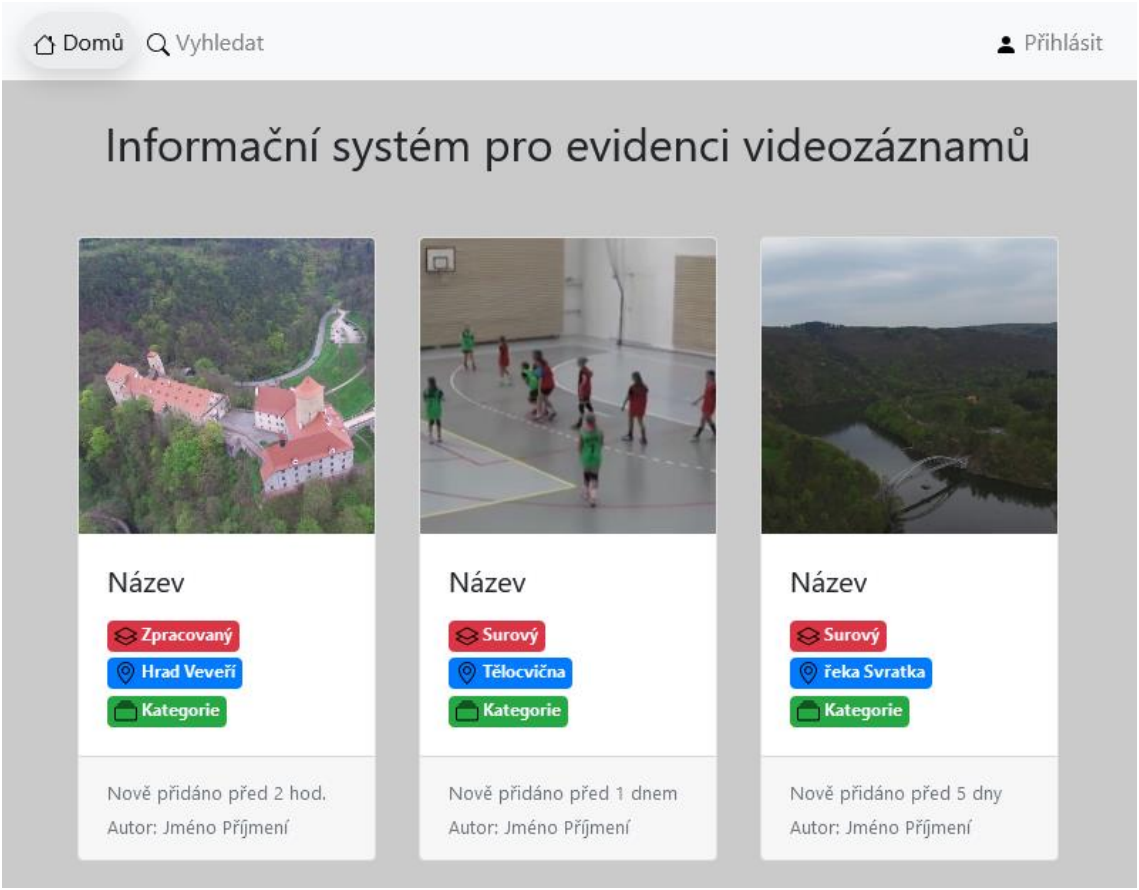


Obrázek č. 24: Vývojový diagram asynchronního volání
(Zdroj: Vlastní zpracování)

3.6 Rozvržení

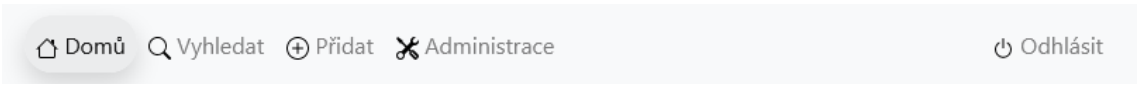
Nyní přistoupím k popisu rozvržení prvků na stránce. Jádro, což je soubor *index.php,* obsahuje základní strukturu HTML (hlavičku a tělo dokumentu). V hlavičce se nachází odkazy, které slouží pro připojení externích souborů. Zejména se jedná o zapsání relativní či absolutní cesty k souborům s kaskádovými styly a javascripty. V tomto případě je nutné importovat knihovny Bootstrap, aby jeho možnosti byly využitelné jako hodnoty atributů tříd HTML značek. Aby bylo možné využít asynchronní volání, je nutné také v hlavičce HTML importovat jQuery.

V úvodu předchozí kapitoly 3.5 byly uvedeny 3 položky, které se nachází v hlavním menu jádra. Tyto položky budou doplněny ještě o tzv. domovskou stránku, na které budou zobrazeny vždy nejnovější příspěvky. Tato možnost urychlí práci s IS, kdy uživatel nebude muset vyhledávat poslední vložené záznamy.



Obrázek č. 25: Reálná ukázka domovské stránky navrhovaného IS
(Zdroj: vlastní zpracování)

Jak je z obrázku výše patrné, menu neobsahuje všechny zmíněné položky. To je z důvodu zohlednění autorizace přihlášení. Pravý horní roh obrázku (s nápisem Přihlásit) napovídá, že je demonstrován přístup anonymního uživatele. Tento druh přístupu je možné v administraci zakázat. Náhledovými obrázky, resp. modulem, který náhledové obrázky zprostředkovává, se zabývá kapitola 3.7. Na obrázku č. 26 lze vidět menu, které po přihlášení vidí administrátor. Přístup běžného uživatele se liší pouze v odebrání položky administrace a v nemožnosti upravovat či odstraňovat záznamy.



Obrázek č. 26: Reálná ukázka menu přihlášeného uživatele
(Zdroj: vlastní zpracování)

Cílem rozvržení je jednoduchost a lehkost, aby se uživatel mohl snadno a pohotově orientovat. K dosažení tohoto cíle jsem využil tzv. Cards, což v češtině znamená karty. Jde o třídu v Boostrapu, díky které lze dosáhnout větší grafické přehlednosti. Současně je

nutné myslet i na klasičtější způsob zobrazení nalezených výsledků. Existuje možnost přepnutí zobrazení do tabulky.

3.6.1 Jádro

V této kapitole je popsána logika fungování jádra z hlediska generování položek v menu.

```
$stranky_nazev = array("Domů", "Vyhledat", "Přidat", "Administrace", "Přihlásit", "Odhlásit");
$stranky_zkratka = array("domu", "vyhledat", "pridat", "administrace", "prihlasit", "odhlasit");
$stranky_obr = array("house.svg", "search.svg", "plus-circle.svg", "tools.svg", "person-fill.svg", "power.svg");
$stranky_zobraz = array(true, uroven_prihl(0), uroven_prihl(1), uroven_prihl(2), uroven_prihl(0), uroven_prihl(0));
```

Obrázek č. 27: Ukázka části kódu jádra
(Zdroj: vlastní zpracování)

Existují 4 jednorozměrná pole, přičemž všechny mají stejný počet prvků. V prvním poli jsou uvedeny zobrazované názvy na stránce. Ve druhém poli se nachází zkráceně zapsané názvy, aby je bylo možné použít jako hodnoty proměnné *stranka*. Předposlední pole obsahuje název obrázku, který je zobrazen vždy nalevo každého názvu položky. Poslední pole obsahuje logické hodnoty: *pravda*, *nepravda*. Není-li určeno jinak, pak se tyto hodnoty získávají z funkce, která má za úkol ověřit, zda má přihlášený uživatel odpovídající oprávnění ke zobrazení příslušných položek menu.

- Anonymní přístup = 0
- Běžný uživatel = 1
- Administrátor = 2

Dále se pomocí metody GET zjistí, zda je proměnná *stranka* definována. Pokud je, skript pokračuje dále, v opačném případě je uživatel přesměrován na domovskou stránku. Spustí se *for* cyklus, což je cyklus s konečným počtem iterací. Počet iterací je stanoven na počet prvků libovolného pole ze všech 4 výše zmíněných polí. Tento cyklus má za úkol vyhledat v poli *$stranky_zkratka* takový prvek, který vyhovuje zadané hodnotě proměnné *stranka* a také vygenerovat menu s odpovídajícími položkami. Nejprve cyklus kontroluje prvky pole *$stranky_zobraz* a pokud je nalezen prvek s hodnotou *nepravda*, dojde k přeskočení této iterace, aby se tato položka vůbec v menu nevyskytla. Jakmile cyklus dokončí všechny iterace a nalezne-li shodu, dojde ke kontrole, zda v zadaném adresáři existuje požadovaný skript. Pokud i tato kontrola dopadne s úspěchem, lze použít příkaz *include* a uživateli zobrazit výsledek. V případě, kdy cyklus nenalezne shodu, je uživatel přesměrován na domovskou stránku.

3.7 Moduly

Součástí tohoto IS jsou také moduly. Tento IS byl dosud prezentován tak, že nezáleží, na jakém operačním systému bude spuštěn, ale byly vymezeny podmínky a předpoklady, které musí být pro provoz tohoto IS dodrženy. Moduly, o kterých se zde zmiňuji, jsou hotovými řešeními třetích stran. Tato řešení jsou spuštěna mimo skriptovací jazyk PHP a při jejich použití již záleží na druhu OS. Z tohoto důvodu volím taková řešení, jež jsou univerzální a dostupná pro nejpoužívanější operační systémy.

3.7.1 FFmpeg

FFmpeg je kompletní multiplatformní řešení pro nahrávání, převádění, streamování audia i videa. FFmpeg náleží pod licence GNU (22). Licence GNU znamená, že se jedná o svobodný software. Toto řešení je dostupné jak pro Linux, Windows, tak i MacOS (22). Tento nástroj je více všestranný a mimo jiné lze vytvořit snímek z videozáznamu z konkrétní minutáže. Používání tohoto modulu je podmíněno jeho stažením. Administrátor má možnost v administraci specifikovat, ve kterém adresáři je FFmpeg uložen a zatrhávacím tlačítkem jej aktivovat. Nabízí se 3 možnosti, za jakých okolností se FFmpeg bude spouštět:

1) současně s existujícími procesy (tj. procesy znázorněné vývojovými diagramy),
2) pravidelně v administrátorem definovanou denní dobu,
3) ručně.

První varianta je více závislá na použitém hardwaru, protože může docházet ke krátkodobému nárazovému zatížení hardwarových prostředků, což způsobuje negativní vliv na zpracovávání ostatních požadavků několika dalších uživatelů. Dále se nabízí možnost provádění automatické pravidelné kontroly, jež spočívá ve strojovém procházení adresářů a zjišťování, které náhledové obrázky nejsou k příslušnému videosouboru dostupné. Tato možnost řeší potíž se slabšími hardwarovými prostředky, kdy generování náhledů může probíhat během noci. Stinná stránka této možnosti spočívá ve zvýšené softwarové režii. Musí být vytvořeny různé způsoby zjišťování existence náhledových obrázků vůči videosouborům s ohledem na používaný OS. Bude-li uplatněna první varianta, pak se využije skutečnosti, že PHP již má kompletní režii kaskád a vznikne přímá závislost na existujících procesích, které jsou znázorněny vývojovými diagramy výše. Do funkce *shell_exec()* lze vložit příkaz operačního systému

a následně setrvat do dokončení zpracování za účelem zobrazení zpracovaných výsledků uživateli, tj. zda se vygenerování obrázku zdařilo.

Z výše nabízených možností spouštění FFmpegu plyne, že záleží na hardwarových prostředcích, které má implementátor k dispozici. Každou možnost spouštění FFmpegu si může administrátor aktivovat či deaktivovat v administraci.

```
krokodyl@krokokomp:~$ ffmpeg -i "zdrojovy_videosoubor.mp4" -ss 00:11:14.435 -frames:v 1
-vf scale=320:320,setsar=1:1 "vystupni_nahledovy_obrazek.png"
```

Obrázek č. 28: Ukázka příkazu FFmpeg pro vygenerování náhledového obrázku
(Zdroj: vlastní zpracování)

Výhodou FFmpegu je, že příkaz, který lze vidět na obrázku výše, je možné spustit i v FFmpegu pro MS Windows přesně s identickou syntaxí. Výsledkem zpracování je obrázek s příponou *.png*. Tento obrázek reprezentuje jeden snímek videa v 11. minutě a 14. sekundě s rozlišením 320x320px. FFmpeg podporuje široké spektrum různých kontejnerů i kodeků. Z toho důvodu by si měl poradit s nejrůznějšími příponami videosouborů, které uživatelé budou do IS nahrávat.

3.8 Distribuce

Informační systém navrhovaný v této práci bude k dispozici s otevřeným zdrojovým kódem na webovém portále GitHub.com. Tento portál je k účelu publikování (s otevřeným zdrojovým kódem) vlastních softwarů přímo vyvinut. Jedná se o webovou službu, kterou využívá přes 65 milionů vývojářů po celém světě (23). Součástí publikace budou nejen samotné skripty, ale i podrobná dokumentace, aby se případný zájemce mohl snadněji orientovat v tomto řešení. Vzhledem k tomu, že se webový portál využívá celosvětově, může být programová dokumentace uvedena i v anglickém jazyce.

3.9 Ekonomické zhodnocení

Z hlediska nákladů na tento navrhovaný informační systém se jedná zejména o náklady na čas potřebný k návrhu a také na vývoj. Tento návrh již zahrnuje veškeré okolnosti, jež budou pro následný vývoj potřebné. Rovněž se musí počítat i s rizikem, kdy se při vývoji vyskytne návrhem nepředpokládaná chyba.

Tabulka č. 11: Náklady na projekt
(Zdroj: vlastní zpracování)

Náklad	Pracnost (MD)	Cena (Kč)
Návrh aplikace	25	150 000,-
Vývoj aplikace	10	60 000,-
Vytvoření dokumentace	3	18 000,-

Určí-li se částka na 1 MD 6 000 Kč, pak celkové náklady na zhotovení kompletního informačního systému vychází na 228 000 Kč.

3.10 Přínosy informačního systému

Jak již bylo řečeno v kapitole 2.1, analýza byla primárně prováděna z hlediska regionálních televizí, a právě těmto jejich zaměstnancům by měl tento IS více zjednodušit a také zefektivnit práci. To ovšem neznamená, že použití je vyhrazeno pouze těmto televizím. Uplatnění by nalezl i v menší produkční společnosti, eventuálně u autorů video-obsahu na webových video-portálech. Díky tomuto IS je výrazně snížená režie dohledávání konkrétního záznamu více zaměstnancům současně oproti vyhledávání v souboru tabulkového procesoru. Pokud budou skutečné přínosy naplněny nebo naopak některé nedostatky zjištěny, ukáže až další zkoumání po dokončení implementace v konkrétních firmách.

Informační systém navrhovaný v této práci neposkytuje přínosy pouze ostatním osobám, ale také mně. Během navrhování jsem získal nové dovednosti a zkušenosti v několika oblastech.

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout informační systém, prostřednictvím kterého bude možné spravovat databázi videozáznamů a díky ní vést přehlednou evidenci, ve které bude mimo jiné snadná orientace a intuitivní uživatelské webové prostředí. Toto nové řešení spojuje prvky existujících řešení s požadavky regionálních televizí, přičemž tyto požadavky jsem v rámci praxí měl možnost prostudovat a popsat.

První část práce pojednává o teoretických východiscích, jež byly nedílnou součástí při zpracovávání dalších částí. Nejprve jsem uvedl pojem informační systém, dále jsem popsal pojem databáze a více přiblížil, co tento pojmem obsahuje. Součástí databází jsou různé typy datových modelů, přičemž podrobněji byl popsán relační datový model, který tvoří součást tohoto IS. Přibližně od poloviny první kapitoly jsou používány pojmy z webového prostředí, neboť všechny uvedené programovací jazyky i knihovny v této části byly nezbytné k sestavení následného řešení. Sestavené řešení se neobešlo bez popisu vývojových diagramů, prostřednictvím kterých byly schematicky vyobrazeny základní procesy, které v tomto systému probíhají. Byly uvedeny některé obrazce, které se při sestavování diagramů užívají.

V následující kapitole jsem se věnoval současnému řešení evidence videozáznamů nejprve v obecné rovině, a dále jsem se zaměřil na oblast regionálních televizí. Odvodil jsem úskalí, jež plynou z těchto stávajících řešení a pokračoval jsem výběrem a popisem existujících řešení. Celá tato kapitola vyústila závěrem, který shrnuje zjištěné informace o existujících řešeních, a také jsou odvozeny požadavky plynoucí z obsahu této kapitoly.

Poslední kapitola práce se skládá především z datového a funkčního modelování. Datový model je navrhnut od samého začátku, včetně určení vazeb mezi relacemi, vymezení primárních a cizích klíčů, stanovení datových typů jednotlivých atributů všech relací. V rámci datového modelu je uveden výsledný diagram a také omezení tohoto modelu. Dále je funkční model rozložen na jednotlivé procesy a operace, ze kterých sestává. Tyto procesy jsou znázorněny pomocí vývojových diagramů. Následuje přesnější popsání vazeb přímo souvisejících s relacemi, díky čemuž je možné lépe porozumět ovlivňování relací při provádění různých operací. V závěru kapitoly uvádím odhad ekonomického zhodnocení a přínosy tohoto nového řešení.

SEZNAM POUŽITÝCH ZDROJŮ

(1) ŠMÍD, Vladimír. *Pojem informačního systému* [online]. 2002 [cit. 2021-02-27]. Dostupné z: https://www.fi.muni.cz/~smid/mis-infsys.htm

(2) *Co je to databáze* [online]. Oracle, 2014 [cit. 2021-3-1]. Dostupné z: https://www.oracle.com/cz/database/what-is-database/

(3) KOCH, Miloš a Vysoké učení technické v Brně. *Datové a funkční modelování*. Brno: Akademické nakladatelství CERM, 2006, s. [1a]. ISBN 80-214-3252-7.

(4) CONOLLY, Thomas, Carolyn E. BEGG a Richard HOLOWCZAK. *Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází*. Brno: Computer Press, 2009, s. [1a]. ISBN 978-80-251-2328-7.

(5) *MySQL :: About MySQL* [online]. Oracle, 2021 [cit. 2021-5-10]. Dostupné z: https://www.mysql.com/about/

(6) WELLING, Luke a Laura THOMSON. *Mistrovství PHP a MySQL*. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.

(7) *HTML Standard* [online]. WHATWG, 2021 [cit. 2021-3-10]. Dostupné z: https://html.spec.whatwg.org/

(8) *HTML & CSS - W3C* [online]. W3C, 2016 [cit. 2021-3-10]. Dostupné z: https://www.w3.org/standards/webdesign/htmlcss

(9) *Bootstrap 3 Tutorial* [online]. Refsnes Data, c1999-2021 [cit. 2021-3-10]. Dostupné z: https://www.w3schools.com/bootstrap/

(10) *Extensible Markup Language (XML) 1.0* [online]. 5th ed.: W3C, 2008 [cit. 2021-3-4]. Dostupné z: https://www.w3.org/TR/xml/

(11) ZAKAS, Nicholas C. *JavaScript pro webové vývojáře.* Brno: Computer Press, 2009. Programujeme profesionálně. ISBN 978-80-251-2509-0.

(12) *CSS color Property* [online]. Refsnes Data, c1999-2021 [cit. 2021-3-10]. Dostupné z: https://www.w3schools.com/cssref/pr_text_color.asp

(13) TRETEROVÁ, Eliška. *Návrh a vývoj algoritmů: modul-vývojové diagramy a příkazy jazyka Borland Pascal.* Ostrava: Ostravská univerzita, 2003. Systém celoživotního vzdělávání Moravskoslezska. ISBN 80-7042-854-6.

(14) *YouTube for Press* [online]. Youtube, 2021 [cit. 2021-2-10]. Dostupné z: https://blog.youtube/press/

(15) *PROVYS TV Office byl úspěšně otestován na architektuře Sun - Lupa.cz* [online]. Praha: Internet Info, 2007 [cit. 2021-3-10]. Dostupné z: https://www.lupa.cz/tiskove-zpravy/provys-tv-office-byl-uspesne-otestovan/

(16) *Provys – Integrated software solution for broadcasting organisations* [online]. DCIT, 2020 [cit. 2021-2-10]. Dostupné z: https://provys.com/

(17) *Document Management System Software | OpenKM* [online]. Open Document Management System S.L., 2021 [cit. 2021-5-10]. Dostupné z: https://www.openkm.com/

(18) OpenKM. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-3-10]. Dostupné z: https://en.wikipedia.org/wiki/OpenKM
(19) *Nextcloud* [online]. Nextcloud, 2021 [cit. 2021-5-10]. Dostupné z: https://nextcloud.com/
(20) *System requirements – Nextcloud latest Administration Manual latest documentation* [online]. Nextcloud, 2021 [cit. 2021-3-10]. Dostupné z: https://docs.nextcloud.com/server/latest/admin_manual/installation/system_requirements.html
(21) *JQuery* [online]. OpenJS Foundation, 2021 [cit. 2021-3-10]. Dostupné z: https://jquery.com/
(22) *FFmpeg* [online]. 2021 [cit. 2021-5-10]. Dostupné z: https://www.ffmpeg.org/
(23) *About · GitHub* [online]. GitHub, 2021 [cit. 2021-5-1]. Dostupné z: https://github.com/about

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

CK	cizí klíč (Foreign key)
CSS	kaskádové styly (Cascading Style Sheets)
DBMS	systémem pro správu databáze (Database Management System)
ER diagram	entito-relační diagram (Entity-relationship Diagram)
GNU	General Public License (licence určená svobodnému softwaru)
HTML	hypertextový značkovací jazyk (Hyper Text Markup Language)
IS	informační systém
MD	pracnost (člověkoden Man-Day nebo jen Manday)
n-tice	uspořádaný seznam konečného počtu n objektů (konkrétní varianty nazývané uspořádané dvojice, uspořádané trojice atd.)
OS	operační systém
PK	primární klíč (Primary Key)
px	Pixel (obrazový bod)
SMB	Server Message Block (síťový komunikační protokol)
SQL	jazyk pro manipulaci s daty (Structured Query Language)
XML	eXtensible Markup Language; dokument s hierarchicky uspořádanými elementy

SEZNAM POUŽITÝCH OBRÁZKŮ

SEZNAM POUŽITÝCH TABULEK