VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO SPRÁVU DOKUMENTŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAKUB ŠVESTKA
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO SPRÁVU DOKUMENTŮ

APPLICATION FOR DOCUMENT MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAKUB ŠVESTKA
AUTHOR

VEDOUCÍ PRÁCE Ing. VLADIMÍR BARTÍK, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce se zabývá návrhem a realizací webové aplikace určenou pro řízení přípravy a správu dokumentů. Aplikace návštěvníkům poskytuje materiály z různých akcí (např. konference), které si mohou zakoupit ze svého kreditu, jenž si dobijí pomocí platební brány. Dalším účelem aplikace je řízení přípravy materiálů, kdy uživatelé postupně odevzdávají svou práci do systému, a správce tak má přehled o plnění úkolu. Práce klade důraz na jednoduché uživatelské rozhraní, responzivní design a přístupnost. Aplikace je postavena na běžně dostupné technologii `PHP 5.6` s použitím frameworku `Nette` od organizace *Nette foundation*, frameworku `Twitter Bootstrap` a knihovny `jQuery`. Výsledek mé práce je dostupný veřejnosti na adrese `www.monitorevent.cz`.

Abstract

This thesis deals with design and implementation of web application for the preparation control and document management. The application provides the visitors with documents concerning various events (e.g. conferences) and these documents can be purchased by means of their credit which can be topped up by means of pay gateway. The next purpose of the application is document preparation control, where the users gradually submit their work to the system and therefore the administrator can be informed about the task execution. The thesis puts the accent on user-friendly interface, responsive design and accessibility. The application is built on standard `PHP 5.6` technology with *Nette foundation* framework `Nette`, `Twitter Bootstrap` framework and `jQuery` library. The contents of my thesis is accessible to the public on the `www.monitorevent.cz` website.

Klíčová slova

správa materiálů, řízení přípravy materiálů, Nette, platební brána, Bootstrap, PHP

Keywords

document management, preparation control of documents, Nette, payment gateway, Bootstrap, PHP

Citace

Jakub Švestka: Aplikace pro správu dokumentů, bakalářská práce, Brno, FIT VUT v Brně, 2015

Aplikace pro správu dokumentů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka Ph.D.

. . . . . .
Jakub Švestka
14. května 2015

Poděkování

Rád bych poděkoval Ing. Vladimíru Bartíkovi, Ph.D. za cenné rady, osobní přístup a čas věnovaný vedení mé bakalářské práce. Dále bych chtěl poděkovat firmě *PRESCOM*, konkrétně Ing. Pavlu Stehlíkovi, za zkušenosti v oblasti řízení přípravy materiálů a pomoc při testování aplikace.

Obsah

Kapitola 1

Úvod

V rané době prvních osobních počítačů počal rozmach ve vytváření dokumentů v digitální formě. Pro autory a čtenáře tato forma nabízí řadu výhod než forma tištěná. Hlavní výhodou je okamžitá distribuce ke koncovému uživateli (a to nejčastěji pomocí síťového připojení či paměťového zařízení), nenáročná reakce na úpravy a využití různých nástrojů při práci. Stojí také za zmínku možnost práce s dokumenty pro nevidomé osoby bez nutnosti použití Braillova[1] písma či s jiným handicapem např. předčítání textu, diktování textu, zvětšení textu pro slabozraké atd.

S nástupem Web 2.0 přichází kancelářské balíky, jenž jsou provozovány jako webové aplikace – např. Google Docs (http://docs.google.com/). Využívají přívětivé uživatelské rozhraní a nabízí tvorbu pomocí *WYSIWYG*[2] metody. Uživateli tak stačí mít podporující webový prohlížeč. Hlavní výhody online tvorby je možnost okamžitého sdílení, tvorba jednoho dokumentu více uživateli zároveň a okamžité ukládání na servery poskytovatele.

V této bakalářské práci se zabývám vývojem webové aplikace pro správu a prodej materiálů, jenž slouží jako doprovodné materiály k pořádaným přednáškám/konferencím. Aplikace by měla ulehčit práci jak pořadatelům, tak i návštěvníkům (zákazníkům) tím, že vše je centralizováno na jednom místě. Pořadatel tedy má přehled o všech materiálech a zákazník nemusí od nikoho žádat přístup k nim či je dohledávat v emailové schránce.

Další část aplikace bude sloužit pro přípravu materiálů. Uživatelé (konkrétně přednášející) budou mít zadané úkoly na vypracování materiálu dle uvedeného zadání. Vypracování bude omezeno lhůtou, během které může uživatel vypracovávaný materiál postupně odevzdávat. Veškeré starší verze budou uchovávány v repozitáři. Uživatel díky tomu bude mít možnost kdykoliv se vrátit ke starší verzi materiálu, a správce tak může sledovat postupné plnění úkolu. Aplikace dále bude přehledně informovat uživatele o zbývajícím čase pro splnění úkolu a jeho aktuálním stavu.

Cílem této práce je nejprve zmínit problematiku tvorby webových aplikací, kde se zabývám procesem jejich vývoje, jenž se liší od použitých metodologických technik při vývoji běžného SW. V poslední řadě bude zmíněna jejich použitelnost a celkově jejich přístupnost. Třetí kapitola je již určena analýze a specifikaci požadavků, a to konkrétně správě a řízení přípravy materiálů. Čtvrtou kapitolu věnuji použitým technologiím, které byly pro implementaci aplikace použity. V páté kapitole popisuji vybrané případy užití aplikace a návrh ER diagramu. Dále zmíním samotný návrh grafického vzhledu, jenž je rozdělen na část uživatelskou (frontend) a část administrátorskou (backend). Šestá kapitola se zaobírá

[1] Druh písma určený pro nevidomé

[2] Akronym anglické věty „What you see is what you get“→ „co vidíš, to dostaneš“

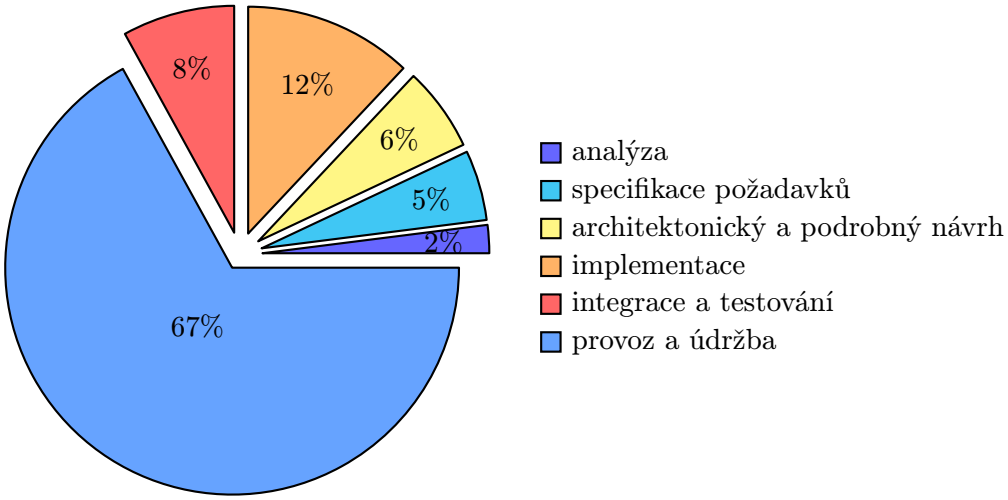
strukturou a implementací zajímavých částí aplikace. Závěr práce popisuje výsledky testování, díky němuž jsem získal zpětnou vazbu od uživatelů. Dále uvedu, jaké změny byly v aplikaci na základě uživatelských podmětů provedeny.

Kapitola 2

Problematika tvorby webových aplikací

2.1 Proces vývoje softwaru

Při vývoji softwarových produktů, tedy i webových aplikací, jsou použity metodologie, které vývoj dělí do několika etap. Jejich cílem je podpora v oblasti plánování a řízení vývoje. V případě nevyužití nebo špatném výběru metodologie, projekt nemusí odpovídat požadované kvalitě a jeho vývoj/údržba se může prodloužit/prodražit.



Graf 2.1: Etapy vývoje SW

Z grafu vyplývá, že 33 % z celkové doby životního cyklu spotřebuje vývoj softwaru. Zbylou část 67 % reprezentuje provoz a údržba softwaru. Nejdůležitější částí je etapa analýzy a návrhu, jelikož v porovnání se samotnou implementací zabírá více času a v případě zanedbání se chyby projeví v následujících etapách. Ze studií vyplynulo, že 60 % až 70 % chyb při vývoji je zapříčiněno špatnou specifikací a návrhem. [9, s. 184]

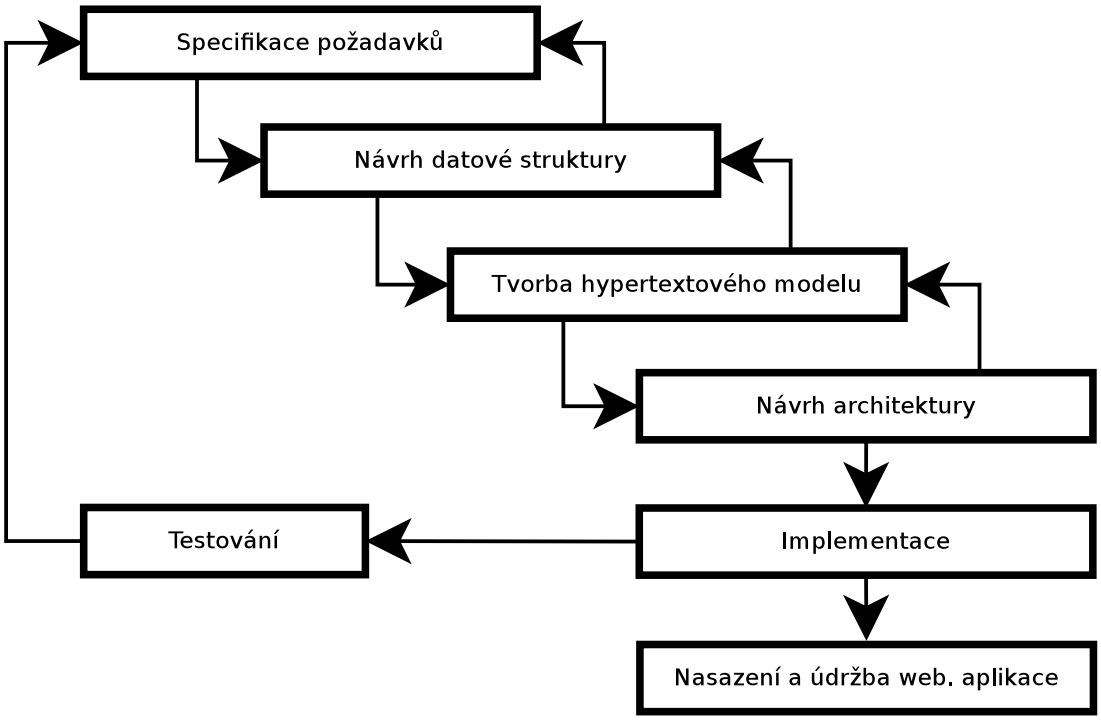
2.2 Proces vývoje webových aplikací

S rozšiřujícími technologiemi a vyššími požadavky převládá vývoj dynamických webových aplikací, které jsou oproti statickým daleko rozsáhlejší. Z důvodu jejich komplexnosti bylo nutné při vývoji využití metodologických technik. Z počátku se v této oblasti využívaly kvalitní metodologie, které byly určeny pro vývoj softwaru (např. *UML*[1]). Postupem času se ukázalo, že nejsou příliš vhodné pro oblast webových aplikací z důvodu chybějících etap, které se zabývají funkčností a použitelností ve vztahu k uživateli. Proto vznikla motivace k vytvoření metodologických technik vytvořených na míru webovým vývojářům – příkladem může být *WebML*, *OOHDM* atd.[16]

2.2.1 WebML

Je jedna z nejvíce rozvinutých webových metodik současnosti. Vznikla na italské univerzitě Politecnico di Milano a jedná se o rozšířené *UML*, jehož sada formálních grafických specifikací je navíc obohacena o definici kompletního procesu návrhu a zahrnuje kompletní životní cyklus webové aplikace. Tato metodika je pro nekomerční užití zdarma. [15]

Oproti obecnému schématu vývoje softwaru *WebML* klade podstatně větší důraz na jeho počáteční fázi. Vedle strukturálního (datového) modelu, který odpovídá diagramu tříd z *UML*, ho rozšiřuje o další tři: hypertextový, prezentační a personalizovaný model. [11, s. 2]



Obrázek 2.1: Fáze vývoje dle metodologie WebML

[1]Unified Modeling Language – grafický jazyk pro analýzu, návrh a dokumentaci softwaru

2.2.2 Strukturální model

Též nazýván datovým modelem. Návrh datové struktury dat, se kterými webová aplikace pracuje. Používá se libovolný prostředek pro datové modelování, a to nejčastěji *ER*[2] model či *UML* model tříd. Ten se pak snadno použije pro vytvoření struktury určité relační databáze.

2.2.3 Hypertextový model

Nejdůležitější částí z modelů. Dělí se na další dva modely:

- **Kompozice webové aplikace**
  Definuje jednotlivé stránky, které společně tvoří výslednou prezentaci a složení stránek, jež jsou dále tvořeny předdefinovanými elementy.

- **Navigace**
  Vyjadřuje, jak jsou mezi sebou stránky, respektive jejich elementy, propojeny. Odkazy mezi stránkami mohou být bezkontextové (stránka s detailem filmu, odkazující na hlavní stránku) či kontextové – přenášející stavovou informaci (stránka s detailem filmu odkazující na filmografii režiséra). Stále pouze definuje strukturu a funkčnost na konceptuální úrovni (není známý vzhled aplikace).

2.2.4 Prezentační model

Vstupem jsou transformace předchozích modelů. Popisuje šablonu a její grafický vzhled – výstup je generován jako *XLST*[3] transformace.

2.2.5 Personalizovaný model

Definuje uživatele a uživatelské skupiny, jež podporují uživatelský kontext webové aplikace – přizpůsobení stránek dle typu návštěvníka. Dělí se na [6, s. 202]:

- *Deklarativní personalizace*
  Stránka se přizpůsobí na základě pravidel, jejichž definice je závislá na datech specifických pro uživatele. Např. neautorizovanému uživateli webové aplikace se nezobrazí sekce pro autorizované.

- *Procesní personalizace*
  Přizpůsobení stránky se děje na základě zadaných pravidel a informací, které systém postupně získává o uživateli. Např. dle historie objednávek a splnění daného kritéria se uživateli přiřadí hodnost „nejlepší kupující.“

2.2.6 Specifikace požadavků

Úvodní fáze vývoje webové aplikace, která podstatně závisí na úspěchu projektu. Pozornost by měla být věnována požadavkům samotným a nikoliv samotné realizaci. Specifikace požadavků je rozdělena do dvou fází.

[2] Entity-relationship – metoda datového modelování
[3] Převod dat ve formátu XML do požadované datové struktury (např. HTML, jiné XML)

Sběr požadavků

Získání informací o aplikaci, a to nejčastěji ze schůzek se zákazníkem či z již vypracované specifikace požadavků zákazníkem. Formulace požadavků probíhá přirozeným jazykem, který může způsobit nejednoznačnost či mylné pochopení obou stran.

WebMl je dále rozšířen o následující druhy požadavků:

- *Funkční požadavky*
  Definice funkcionality aplikace určující rozsah aplikace. Pro jejich vymezení se dá použít UseCase diagram [4].

- *Identifikace uživatelů*
  Vyčlenění typů uživatelů a skupin, jenž budou v aplikaci figurovat. Vyjádření též pomocí UseCase diagramu.

- *Požadavky na personalizaci*
  Určují, zda je nutné personalizovat části webové aplikace pro určitě skupiny uživatelů – zavedení přístupových práv (např. běžný návštěvník nebude mít přístup do sekce pro administrátory).

- *Požadavky na data*
  Specifikace dat, která budou aplikací zpracovávaná a prezentovaná. Slouží jakou základ pro etapu datového modelování.

- *Ostatní požadavky*
  Požadavky na dostupnost (počet přístupů za jednotku času), použitelnost (snadná obsluha uživateli), bezpečnost (šifrování dat, dvoufázová autentifikace). [16]

Analýza požadavků

Převod získaných neformálních požadavků zákazníka do strukturální formy, která zajistí shodnou interpretaci mezi osobami podílejícími se na vývoji. Výstupem jsou následující poznatky:

- *Skupiny uživatelů*
  Identifikace potenciálních uživatelů a rozřazení do hierarchických skupin. Využití detailů případů užití spojených s digramem případu užití.

- *Případy užití*
  Pro každou skupinu uživatelů se definují případy užití, jenž nejčastěji pomocí tabulky definují základní posloupnost, vstupní podmínky, výstupní podmínky, alternativní posloupnost určité činnosti (např. autorizování uživatele do administrátorské sekce).

- *Datový slovník*
  Identifikace základních části aplikace (entit) – často odpovídající realitě. U každé entity se definují atributy, které ji specifikují, a vztahy s ostatními entitami.

- *Mapa webu*
  Určuje pohledy na webovou aplikaci pro danou uživatelskou skupinu. Například u neautorizovaného uživatele nebude k dispozici administrátorská sekce oproti správcům aplikace.

[4] Diagramy případu užití

- *Základní fragmenty grafického designu aplikace*
  Upřesňují rozložení a umístění jednotlivých prvků v šabloně. Dále určují barevnost aplikace, druhy fontů atd. Tato část se může pro jednotlivé zařízení lišit – rozdílná šablona pro mobilní zařízení a desktopový počítač.

- *Ověřovací testy a ostatní požadavky*
  V této fázi by mělo být určeno, jakou formou se bude výsledný produkt testovat při přebírání zákazníkem. Předejde se takto možným problémům a nedorozuměním. Pro stanovení ověřovacích testů je doporučeno použít přesné uvedení hodnot, které jsou měřitelné. [16]

2.3 Použitelnost

Běžný uživatel při přístupu na určitou webovou aplikaci by měl pouhým pohledem vydedukovat, čeho se týká a jaké nabízí rozhraní. Aplikace by tedy měla být přehledná a intuitivní, aby s ní uživatel byl schopen pracovat bez zbytečného přemýšlení a najít informaci, kterou hledal. Nevhodnou použitelnost lze přirovnat k řízení auta bez mapy oblasti, která je nám doposud neznámá, a pokud nemáme schopnost se zorientovat, jsme ztraceni[7]. Tímto danou oblast již v budoucnu nenavštívíme. To samé platí pro návštěvníka webové stránky, kdy takto docílíme faktu, že se již nebude chtít v budoucnu vrátit do naší aplikace (nebude mít k tomu důvod).

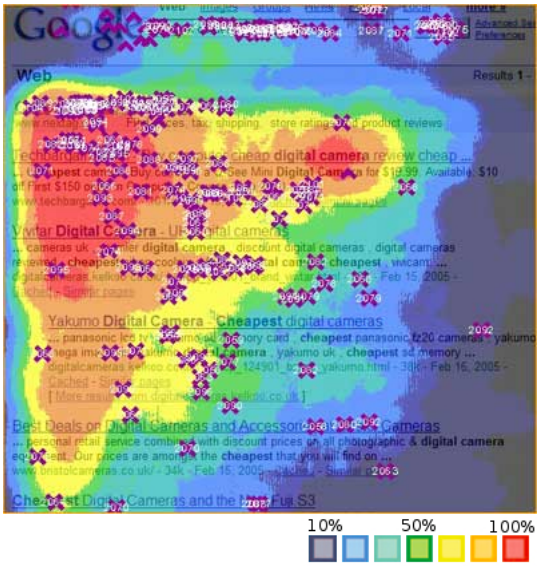
2.3.1 Stránky nečteme, ale prohlížíme

Podle zdokumentované analýzy přístupu uživatele k webové stránce vyplývá, že lidé stráví jen velmi krátkou dobu čtením stránek[8, s. 28]. Není tedy účelem zahltit uživatele vyčerpávajícími informacemi, jelikož větší část z nich je pro něj nepodstatná – výjimku samozřejmě tvoří aplikace, které jsou vyvinuty za tímto účelem, a to například zpravodajské či naučné servery (*Wikipedia*, standardy *RFC*, normy ČSN). Uživatel stránky spíše prohlíží a hledá fundamentální části textu (např. nadpisy) či komponenty (např. navigace, vyhledávací formulář). Celou posloupnost činností můžeme přirovnat k prohlížení tištěných materiálů (noviny, časopisy), kde hledáme pasáže, které jsou pro nás něčím významné – teprve pak začínáme číst či zkoumat daný prvek.

Vědecké studie uvádějí [12], že obvyklý způsob prohlížení stránky (na počítači, chytrém telefonu, čtečce knih atd.) odpovídá tvaru písmene `F`[5]. Čtení stránky začíná v levém horním rohu s následným horizontálním pohybem vpravo. Poté čtenář pokračuje vertikálním pohybem dolů na další část a opět horizontálním pohybem vpravo. Tento způsob čtení je pro nás charakteristický, jelikož jsme zvyklí číst shora dolů a zleva doprava.

Na obrázku 2.2 můžeme vidět teplotní mapu pozornosti výsledků z vyhledávání určité fráze pomocí vyhledávače Google.com. Na prvních pozicích se nachází nejvíce relevantní záznamy. Každý nalezený záznam má svůj nadpis, který je přehledně oddělen od popisu obsahu. Pokud čtenáře zaujme jeden z nich, je pravděpodobné, že bude dále následovat přečtení jeho popisu, aby zjistil, zda daný záznam je to, co hledá.

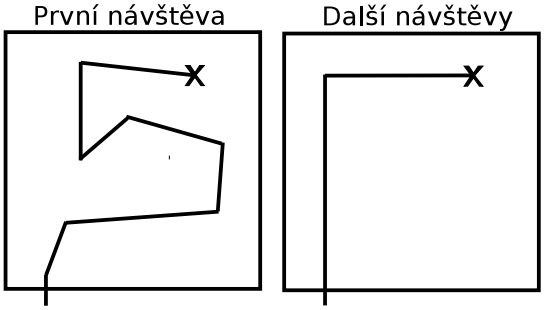
[5] F-reading pattern



Obrázek 2.2: Termomapa čtení výsledků vyhledávání, zdroj: http://www.3plains.com/faq/google-serp-heat-map/

2.3.2 Nenutit výrazně přemýšlet uživatele

Tímto je myšlena správná struktura a rozmístění obsahu šablon stránek[6] webové aplikace a dodržování standardů. Cílem není vytvořit uživatelsky nepřívětivé stránky, ale naopak podpořit uživatele v dobré orientaci při jejich použití. Jedná se o natolik závažnou část, kdy její nedodržení může způsobit okamžitý odchod uživatele z webové prezentace. Tato situace spíše nastává, pokud se uživatel na stránce nachází poprvé a neví, kde najde určité informace – znázorňuje levá část obrázku 2.3. S dalšími návštěvami se uživatel postupně učí a po určité době si je již jist, kde dané informace hledat.



Obrázek 2.3: Schéma hledání informace

Typicky se můžeme setkat s následujícími zvyklostmi:

[6] Layout – složen z jednotlivých částí (komponent)

Logo

Umístěno v záhlaví (nejčastěji v levém rohu nebo alespoň poblíž), aby ho uživatel zaregistroval jako první věc na stránce. Logo informuje uživatele, že si stále prochází stránky stejné webové aplikace a nejčastěji se sloganem reprezentuje její poslání. Dále by mělo být obohaceno funkčností odkazu, jenž po kliknutí zajistí přesměrování na titulní stránku aplikace (homepage).

Navigace

Komponenta, která v důsledku špatného návrhu může způsobit nepřístupnost webové aplikace pro uživatele, tak i pro vyhledávací roboty. Typické umístění její vertikální varianty je v levé části stránky nebo v případě horizontální varianty se nachází v horní části stránky (záhlaví). Při vytváření položek bývá výhodou dodržení následujících bodů, které eliminují špatný koncept navigace:

- název položky musí být výstižný a stručný
- přiměřený počet položek v rámci jedné navigace či použít seskupení do kategorií
- důležité položky mít na prvních pozicích
- položka `kontakt` by měla být na poslední pozici
- nepoužívat grafické položky (tlačítka), ale raději text

Viditelné odkazy

Uživateli by mělo být ze struktury textu jasné, na co může kliknout. Toho docílíme tak, že prvky budou mít jiný způsob zobrazení (barva, podtržení) nebo při najetí najetí kurzorem změníme typ ukazatele.

2.3.3 Responzivní design

Z důvodu rozšířenosti prohlížení webových stránek na mobilních zařízeních (dotykové ovládání, různé úhlopříčky), je vhodné jejich obsah přizpůsobit pomocí responzivního designu. Jedná se tedy o techniku stylování `HTML` dokumentu, která zajistí skrytí nadbytečných částí obsahu stránky a vhodně ji přizpůsobí. Úmyslem je zpříjemnit ovládání, omezit použití funkce *pinch to zoom*[7].

Po úpravách by měl mít uživatel stále přístup ke všem důležitým informacím, jako při přístupu pomocí jiného zařízení. Nemělo by se myslet pouze na přívětivý vzhled, ale i na optimalizaci. To znamená nenačítat nepotřebné skripty a vzhledy, vhodně přizpůsobit rozlišení obrázků.

2.3.4 Přístupnost

Smyslem přístupnosti je odstranění překážek, na které může uživatel narazit při používání webových stránek. Přístupnost ocení každý uživatel, jelikož stránky budou uživatelsky přívětivější a návštěvník se na nich bude lépe orientovat. Výrazné ulehčení bude možné pozorovat u handicapovaných osob, jež se potýkají s nepřístupností řady webových stránek.

[7] Provedení gesta pro přiblížení či oddálení stránky

Tuto situaci si lze představit, jako prohlížení webových stránek pouze pomocí klávesnice – při špatném návrhu nebude možné se stránkou pracovat či se dostat do dalších sekcí.

Vytváření přístupných stránek není finančně či časově náročnější – je to pouze o tom, že programátor vytváří stránky dle určitého standardu. Příkladem může být uvádění alternativního popisku u obrázků. V případě jeho neuvedení, nevidomé osoby nebudou schopny zjistit, čeho se obrázek týká, a zda patří ke kontextu hledané informace. Alternativní popisek pomůže dále indexovacím robotům vyhledávačů – ti jsou v jistém slova smyslu také handicapovaní, protože nevidí řadu věcí (mezi nimiž i obrázky).[17]

Přístupný web nepřináší užitek pouze návštěvníkovi, ale i provozovateli, a to v podobě vyšší návštěvnosti a počtu spokojených uživatelů.

2.3.5 Hodnota a stárnutí webové aplikace v čase

Tak jako spotřební elektronika, i technologie vývoje webových aplikací se z důvodů měnících se trendů rychle rozvíjí. Příkladem může být počáteční vývoj webových stránek, kdy se vytvářely pomocí tabulkové struktury. S příchodem blokově orientované struktury již nepřevládá její užití (i když tabulková struktura je v opodstatněných případech přirozenější – informační systémy, katalogy).

Technologie není pouze jediná věc, která se rapidně rozšiřuje. Markantní změny můžeme pozorovat v grafickém vzhledu stránek, který byl dříve nerealizovatelný. Kupříkladu šířka zobrazení stránky, jež je v posledních letech velmi rozmanitá z důvodu rozmachu mobilních zařízeních – důvod tvorby responzivního designu (viz 2.3.3).

Životnost webové aplikace se pohybuje přibližně v rozmezí 2-4 let[10]. Pokud je aplikace dobře navržena a pravidelně aktualizována, životnost se zvyšuje. Špatný návrh může zapříčinit prodražení úprav aplikace – v tomto případě je praktičtější provést nový a již lepší návrh. Správným krokem může být postavení aplikace na řadu nabízených redakčních systémů (Joomla[8], Wordpress[9], Drupal[10] atd.). Tímto se aplikaci zajistí možnost pravidelných aktualizací k udržení bezpečnosti, rozšířitelnosti a hlavně dojde k prodloužení její životnosti.

2.3.6 Zabezpečení

V případě statických webových stránek nehrozí žádné nebezpečí, pokud neoprávněná osoba nemá přímo přístup k souborovému systému, kde je webová aplikace umístěna. V současnosti jsou hojně vytvářeny dynamické webové stránky. V případě jejich nezabezpečení hrozí při návštěvě minimálně schopného útočníka napáchání škody. Při programování je důležité být obezřetný ke vstupním datům, které přichází ze strany uživatele. Při neošetření vstupů může dojít ke zneužití aplikace například pomocí `SQL injection`[11] či nahrání škodlivého skriptu.

Vhodně zvolený framework na straně serveru dokáže ošetřit řadu problémů s možným útokem. Například framework `Nette` automaticky ošetřuje všechny vstupy od uživatele. Pokud to je v jistých případech nežádoucí, může se zmíněné chování u daného prvku zakázat.

[8] Dostupná z http://www.joomla.org/
[9] Dostupný z http://www.wordpress.org/
[10] Dostupný z http://www.drupal.org/
[11] Technika napadení databázové vrstvy aplikace

2.4 Práce s různými typy dokumentů

Technologický pokrok nabízí díky elektronickým dokumentům řadu výhod v možnostech jejich tvorby a správy.

Jejich klíčové vlastnosti dle Gladneyho[5, s. 270] jsou:

- nezničitelnost (vlivem délky jejich uchovávání)
- nikdy se neztratí
- snadno vyhledatelné
- po celou existenci si zachovávají srozumitelnost bez ohledu na zastarávání technologií

První dva body je potřeba brát s nadhledem, poněvadž dokument může být lidskou chybou či špatným uskladněním média, na kterém je uložen, nenávratně znehodnocen (ztráta média, nechtěné smazání, poškození souboru). Z těchto důvodů jsou elektronické dokumenty populární a cenné materiální dokumenty, ať už fyzicky či hodnotně, jsou digitalizovány.

K dispozici je řada textových procesorů s `WYSIWYG` editorem (kancelářský balík MS Office, LibreOffice, OpenOffice) či použití pokročilejších nástrojů pro sazbu jako TEX. S dostupným kvalitním internetovým připojením přibývá řada online nástrojů pro jejich tvorbu a správu, jež nabízí více funkcionalit – příkladem může být služby jako Google Docs (`http://docs.google.com`) či Office 365 (`http://mail.office365.com`).

2.4.1 Google Docs

Jedná se o službu, která funguje z prostředí internetového prohlížeče. Dokumenty jsou při své tvorbě automaticky ukládány do úložiště na straně serveru. Výhodou je možnost skupinové práce a sdílení, jak v rámci uzavřené skupiny, tak i veřejně. Zajímavou funkcí je verzování dokumentů, pomocí něhož je k uživateli k dispozici procházení veškerými změnami. Procesor zvládá import dokumentů ve většině populárních formátech jako `DOC`, `HTML`, `TXT`, `ODT`, `RTF`, `PDF` atd.

2.4.2 Neplnohodnotná náhrada kancelářských balíků

Online textové procesory je třeba brát s nadhledem, poněvadž se nejedná o plnohodnotnou náhradu kancelářských balíků. Elegance tohoto směru tvorby je v jeho online povaze a díky tomu nadstandardním funkcím jako sdílení, tvorba z mobilního zařízení a přístup odkudkoliv. Dále uživateli stačí mít na zařízení internetový prohlížeč a internetovou konektivitu. Tyto nástroje najdou své uplatnění převážně v týmové spolupráci.

Kapitola 3

Analýza a specifikace požadavků

Webová aplikace bude mít za cíl ulehčit organizátorovi konferencí řízení přípravy dokumentů, přednášejícího přehledně informovat o přidělených úkolech a v neposlední řadě správu dokumentů s možností jejich zakoupení. Veškeré požadavky vycházejí z potřeb zadavatelské firmy *PRESCOM*, jenž má na starost organizaci konferencí.

3.1 Správa dokumentů

Cílem je umožnit snadnou správu dokumentů z pořádaných akcí (prezentace, příklady atd.), jak pro organizátora, tak i pro návštěvníka. Návštěvníkovi bude nabídnuta možnost získat veškeré dokumenty online, a to okamžitě a z jednoho místa. Tímto o ně nemusí žádat organizátory či je dohledávat v jiných zdrojích (např. v emailové schránce).

Adresářová struktura dokumentů

Dokumenty budou seskupovány do konferenčních bloků, a ty budou dále sdružovány do samotných konferencí. Z důvodu zajištění lepší přehlednosti a snadného dohledání, budou konference dále hierarchicky seskupovány do kategorií.

Zobrazení vystavených dokumentů

Aplikace pro své registrované i neregistrované zákazníky bude zajišťovat zobrazení vystavených dokumentů různých formátů. Registrovaný zákazník si následně může zakoupit zpoplatněné materiály pomocí úhrady ze svého kreditu. Neregistrovanému zákazníkovi bude umožněno pouze zobrazení detailu dokumentu.

Historie nákupů a transakcí

Uživatel by měl mít možnost ve svém profilu zobrazení historie pohybu svého kreditu – zakoupení konference, dobití kreditu. Dále zobrazení zakoupených dokumentů s možností filtrování dle konferencí a konferenčního bloku pro snadné dohledání.

3.2 Řízení přípravy materiálů

Tato část bude spočívat v zadávání úkolů na vytvoření potřebných dokumentů. Úkoly budou zadávány v administrační sekci, kde organizátor zadá název, popis, počáteční datum

vypracování a datum odevzdání. Každý úkol bude označen statusem, reprezentující fázi, ve které se úkol aktuálně nachází.

Dále uživatel v sekci pro přípravu materiálů přehledně uvidí přidělené úkoly, kdy po kliknutí na požadovaný úkol se zobrazí jeho podrobnosti. Součástí sekce pro vkládaní materiálů bude repozitář, kde budou verzovány nahrané soubory – uživatel tak tedy má možnost se kdykoliv vrátit ke starší verzi a organizátor může sledovat postupný vývoj vypracovávaného materiálu. Nahrát novou verzi souboru bude možné jen v případě, pokud úkol bude mít přidělený status, jenž tuto funkcionalitu povolí.

3.3 Podobné aplikace

Dle průzkumu nabízených SW řešení žádné neposkytuje stejnou funkčnost. Lze nalézt pouze úzce spojené aplikace, které slouží jen pro správu dokumentů v rámci firmy – pouze backendová část.[1] Možnost přístupu k materiálům návštěvníky či jejich zakoupení zde není možné, z důvodu zaměření aplikace pouze na účel interní komunikace a správy.

Vyvíjená aplikace má za cíl eliminovat chybějící nedostatky uvedené výše, ke kterým se již existující aplikace nehodí. Celkově bude postavena úplně jiným směrem, jelikož nebude sloužit pro interní potřeby, ale naopak bude přístupná veřejnosti. Jejím záměrem nebude pouze správa a řízení přípravy materiálu, ale i jejich prodej.

3.3.1 IS AleX – systém pro správu dokumentů

Jedná se o informační systém, který je provozován pomocí webových technologií (Apache, PHP, MySQL 5). IS nabízí základní moduly:

- Správu dokumentů
- Správu uživatelských práv
- Propojení s aplikacemi třetích stran
- Tiskové sestavy

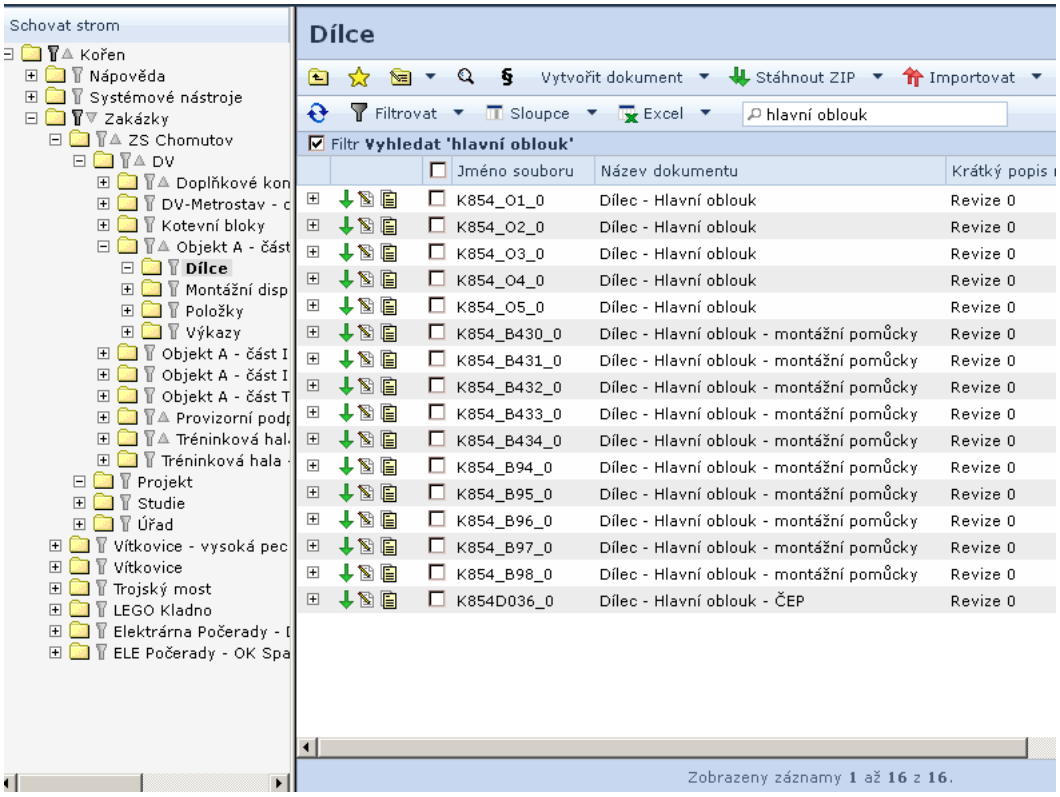
Vložené dokumenty lze hierarchicky řadit do složek. Složky a samotné dokumenty mohou obsahovat metapopisy, [2] které přispějí ke snadnému dohledání dokumentů. Díky verzování je možné se kdykoliv vrátit ke starší verzi dokumentů a v historii akcí lze vidět veškeré akce, které byly s dokumentem provedeny.

Graficky je aplikace přehledná a potřebné funkce jsou k dispozici v nástrojové liště. Výhodou je, že aplikace platformově nezávislá, protože běží na webovém serveru a koncovému uživateli je přístupná přes běžný internetový prohlížeč. Aplikaci jsem neměl možnost si vyzkoušet, poněvadž je komerční (výrobce nenabízí přístup k demoverzi) a nelze tak vyzkoušet či porovnat všechny funkce, které nabízí.

3.3.2 ELOenterprise

Jedná se o desktopovou aplikaci, která může běžet pod běžně používanými operačními systémy, jelikož běží na `JVM`. [3] Výhodou je velká škála nabízených rozšiřujících modulů

[1] Část aplikace která slouží pro určitou skupinou uživatelů, které k ní mají přístup.
[2] Forma přiřazení štítků jako např. datum vytvoření, autor
[3] **Java Virtual Machine** – virtuální stroj pro spuštění aplikací vytvořených v jazyce Java



Obrázek 3.1: Prostředí programu IS AleX, zdroj: http://www.isalex.cz/sprava-dokumentu/projektant/

umožňující například propojení s CAD systémy, napojení na kancelářský balík Microsoft Office, automatické zálohování, ...

3.3.3 Ostatní

Existují další podobné aplikace, které plní stejnou funkčnost či ji rozšiřují (např. adresář kontaktů, správa emailů, intranetové využití atd.). Většina z nich je provozována jako webová aplikace, tudíž je přístupná odkudkoliv bez potřeby instalace dodatečného softwaru. Některé dokonce umožňují vzdálený přístup pomocí mobilní aplikace.

Kapitola 4

Použité technologie

Pro vývoj webových aplikací je k dispozici řada skriptovacích a programovacích jazyků. V tabulce 4.1 je přehled nejpoužívanějších programovacích jazyků na straně serveru.

Jazyk	Využití
PHP	82.0%
ASP.NET	17.0%
Java	2.8%
ColdFusion	0.7%
Ruby	0.6%
Perl	0.5%
Python	0.2%
JavaScript	0.2%
Eriang	0.1%

Tabulka 4.1: Procentní vyžití programovacích jazyků pro tvorbu webových aplikací (na straně serveru) ke dni 27.3.2015, zdroj: http://w3techs.com/technologies/overview/programming_language/all

Jako můžeme vidět, `PHP` je hodně rozšířené a používá se u 82.0 % webových aplikací, jejichž programovací jazyk je znám. Zato `ASP.NET` není tak populární (17.0 %), jelikož oproti `PHP` je:

- závislý na platformě (Windows)
- finančně náročnější
- schopen běžet pouze na IIS serveru
- kompilovaný do `CLR`[1]

4.1 Výběr technologií

Pro aplikaci byly zvoleny běžně dostupné technologie jako Apache, PHP a MySQL (tzv. `LAMP`[2]). Důvodem byly tyto aspekty:

[1] Common Language Runtime – běhové prostředí .NET Frameworku

[2] Zkratka vytvořená z počátečních znaků použitých technologií – **L**iux, **A**pache, **M**ySQL a **P**HP

- běžně dostupné u poskytovatelů webhostingu
- open-source řešení
- nezávislé na platformě
- zkušenost s těmito technologiemi
- finančně méně náročné

4.1.1 Apache

Nejpoužívanější multiplatformní webový server [1], který umožní přístup k webové aplikaci a obsluhuje požadavky klientů. Podporuje řadu programovacích jazyků jako `Perl`, `Python` a již zmíněné `PHP`.

Díky jeho otevřenosti přibývá řada rozšiřujících funkcí (tzv. moduly), ze kterých bych následně zmínil dva hlavní, které budou v aplikaci využity:

Modul `mod_rewrite`

Jeho úkolem je překlad adres dle seznamu zadaných pravidel. Klient tedy pošle požadavek na webový server, který zkontroluje, zda požadované adrese odpovídá nějaké pravidlo (pravidla procházeny postupně shora dolů). Pokud ano, dochází k jejímu přepisu dle příslušných pravidel. V rámci pravidla se jedná o jednu z těchto akcí:

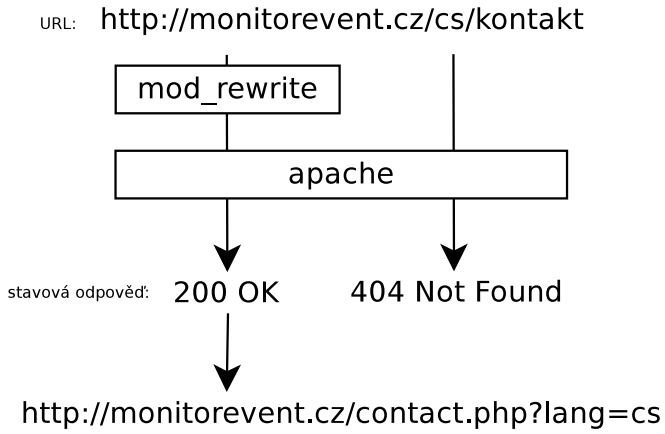
- **Přesměrování**
  Klient je přesměrován na jinou adresu – například ze staré na novou z důvodu změny struktury webové aplikace. Může se jednat o trvalé přesměrování (kód 301) či dočasné (kód 302).
- **Přepisování** (výchozí chování)
  Server vrátí prohlížeči uživatele obsah „nové“ URL adresy, ale o této skutečnosti ho neinformuje – klientem požadovaná adresa tedy nemusí fyzicky existovat. Tato metoda je často používána pro tvorbu tzv. `cool URL`, [3] především kvůli vyhledávačům, ale z části i kvůli uživatelům z důvodu lepší struktury, přehlednosti a hlavně nese srozumitelnou informaci o stránce – obrázek č. 4.1

Modul `mod_ssl`

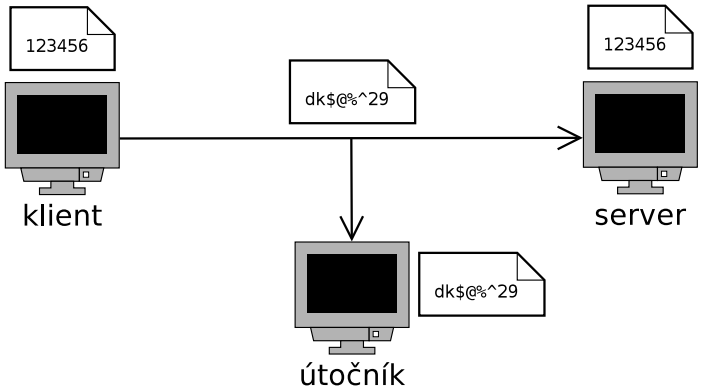
Jeho použitím je možné zabezpečit spojení pomocí mezivrstvy `SSL` mezi klientem (webový prohlížeč) a webovým serverem před odposloucháváním a podvržením dat přenášených během komunikace – příklad na obrázku č. 4.2. Šifrování samo o sobě nezaručuje bezpečnost, jelikož i přes šifrované spojení můžeme stále komunikovat s útočníkem. Proto dalším úkolem tohoto zabezpečení je ověření totožnosti protistrany pomocí digitálně podepsaného klíče (certifikátu) – ten je vydán certifikační autoritou, která při jeho vydání ověří totožnost webového serveru.

Zabezpečená komunikace probíhá přes aplikační protokol `HTTPS` (standardní port 443), jenž je nadstavbou `HTTP` (standardní port 80). Díky odlišným portům mohou běžet z jednoho webového serveru.

[3] Přátelská URL, kde například z adresy `/page.php?productCategory=19` získáváme `/produkty/domaci-potreby/`



Obrázek 4.1: Dotaz na webový server s použitím `mod_rewrite` a bez jeho použití



Obrázek 4.2: Příklad zabezpečené komunikace `HTTPS`

4.1.2 MySQL

Multiplatformní databázový server sloužící pro uchování perzistentních dat pro chod aplikace. Komunikace s relační databází probíhá pomocí standardizovaného dotazovacího jazyka `SQL`. `SQL` je hojně používaný napříč databázovými systémy jako PostgreSQL,[4] Oracle,[5] Firebird[6] atd. Databázové systémy různě rozšiřují `SQL` standard. Přenositelnost `SQL` dotazů mezi jednotlivými databázemi je omezená z důvodu implementace různých rozšíření `SQL` standardu.

Dle [2] se jedná o druhý nejpopulárnější databázový server. Za vysokou popularitu může dostupnost u každého poskytovatele webhostingových služeb, a také fakt, že se jedná o volně šiřitelný software. Další výhodou je snadná použitelnost, správa (například pomocí aplikace phpMyAdmin[7] či Adminer[8]), rychlost a spolupráce s nástroji třetích stran. Mezi nevýhody patří neefektivnost při práci s velkým objemem dat, menší škála `SQL` příkazů a některé typy

[4] Dostupný z http://www.postgresql.org/
[5] Dostupný z https://www.oracle.com/database/index.html
[6] Dostupný z http://www.firebirdsql.org/
[7] Viz http://www.phpmyadmin.net/home_page/index.php
[8] Viz http://www.adminer.org/cs/

databázových úložišť jako `MyISAM` nepodporují transakce a cizí klíče.

4.1.3 Dibi

Abstraktní databázová vrstva, jež má za cíl zjednodušit zápis `SQL` příkazů, eliminovat výskyt chyb (např. neošetření vstupu – `SQL injection`) a umožnit přenositelnost mezi databázovými systémy. Při změně tak nemusíme řešit odlišné funkce (připojení, získání výsledku) a formátování datových typů (datum, čas, atd.) specifických pouze pro danou DB. Aktuálně podporuje následující databázové systémy: MySQL, PostgreSQL, SQLite, MS SQL, Oracle, Access a ODBC.

DibiDataSource

Jedná se o třídu pro skládání dotazů na databázi. Pomocí ní můžeme původní dotaz rozšiřovat o další poddotazy. Příklad: *Získáme výpis všech zaměstnanců, který máme v objektu `DibiDataSource`. Ten předáme komponentě pro výpis záznamů, kde jsou záznamy seřazeny dle uživatelsky specifikovaného sloupce, a následně výpis omezen na určitý počet záznamů.* V této situaci proběhne pouze jeden dotaz na databázi, který si vyžádá jen skutečně vypisovaná data. Tímto není databáze zbytečně zatěžována dotazy a přenášíme jen skutečně potřebná data (což má za následek výrazné zrychlení běhu aplikace).

4.1.4 PHP

Skriptovací programovací jazyk běžící na straně serveru, určený především pro programování dynamických webových aplikací. Jazyk je dynamicky typovaný, tzn. datový typ je vázán na hodnotu proměnné a správa paměti je řešena pomocí **garbage collection**.[9] Pro složitějších programové struktury lze využít objektový návrh, který je k dispozici od páté verze.

Výhody:

- Multiplatformnost a svobodná licence
- Přehledná a kvalitní dokumentace
- Podpora na většině webhostingů
- Rychlé nasazení
- Rozsáhlá podpora funkcí a modulů (např. `FFmpeg` pro zpracování obrazu a zvuku)

Nevýhody:

- Dává programátorovi volnost – náchylné k chybám při vývoji
- Ošetření vstupních hodnot od uživatele není implicitní
- Může zapříčinit špatné návyky programování
- Horší týmová spolupráce a orientace v cizím kódu
- Špatná údržba kódu

[9] Automaticky pomocí algoritmu, který vyhledává a uvolňuje úseky paměti

Výše uvedených nevýhody lze eliminovat za použitím frameworku, [10] který již je k programátorovi striktnější a nutí ho používat určité konvence, což má za následek srozumitelný a snadno udržovatelný programový kód aplikace.

4.1.5 PHP frameworky

Pro vývoj aplikací je k dispozici zhruba třicet `PHP` frameworků[3] (které jsou volně dostupné). Pokud se na ně zaměříme detailněji, výběr se nám zúží, protože ne každý odpovídá požadavkům [4]. Některé frameworky již obsahují množství knihoven (které všechny nevyužijeme) a jsou tedy hardwarově náročnější. Jiné zase méně, ale díky jejich modulárnosti je lze snadno rozšířit o potřebné funkcionality. Dalším aspektem výběru je kvalita dokumentace, která hraje zásadní roli. Pokud je nepřehledná, neudržovaná, znesnadní nám samotný vývoj a může přivodit jiná úskalí.

Výběr frameworku

Z důvodu absence zkušeností s tvorbou webových aplikací pomocí frameworku, a potažmo využití vhodných návrhových vzorů, byl výběr zúžen na ty, které mají: kvalitní dokumentaci, dostupné (video) tutoriály, silné komunitní zázemí. Původně byl vybrán Zend Framework, ale po zhlédnutí několika záznamů o Nette z *posledních sobot*[11] a přednášce autora Davida Grudla na téma „Jak snadno tvořit aplikace v Nette Framework,“ jsem přehodnotil své rozhodnutí. Důvod změny byl hlavně v jasnějšímu pochopení použití `Nette`.

4.2 Nette framework

Český framework, v jehož počátku vývoje stál *David Grudl*, a od roku 2009 pokračuje ve vývoji založená organizace `Nette Foundation`[12]. Dle ankety SitePoint[13] se jedná o 3. nejpopulárnější `PHP` framework. Je založen na populárním návrhovém vzoru `MVP` (viz 4.2.2). Je navržen čistě objektovým návrhem a pro svůj běh potřebuje nejméně verzi `PHP` *5.3.1*.

4.2.1 Dokonalé zabezpečení

Díky implicitnímu zabezpečení před zranitelností aplikace, programátor nemusí řešit opakující se úkony jako například ošetření vstupu od uživatele. Tímto dojde k eliminaci proti útokům jako `SQL injection`,[13] `Cross-Site Scripting`,[14], `Cross-Site Request Forgery`[15] atd.

Ošetření vstupu lze také provádět automaticky na straně uživatele pomocí JavaScriptu. U každého prvku formuláře je generováno validační pravidlo, které se pro validaci použije – pravidla tak máme na jednom místě.

[10] Framework je softwarová struktura, sloužící jako podpora při programování, vývoji a vývoji jiných softwarových produktů.

[11] Setkání komunity `Nette` pořádané každou poslední sobotu v měsíci

[12] Viz stránky organizace:`http://nettefoundation.com/`

[13] Technika napadení databázové vrstvy, kdy útočník pomocí neošetřeného vstupu dojde k pozměnění `SQL` dotazu

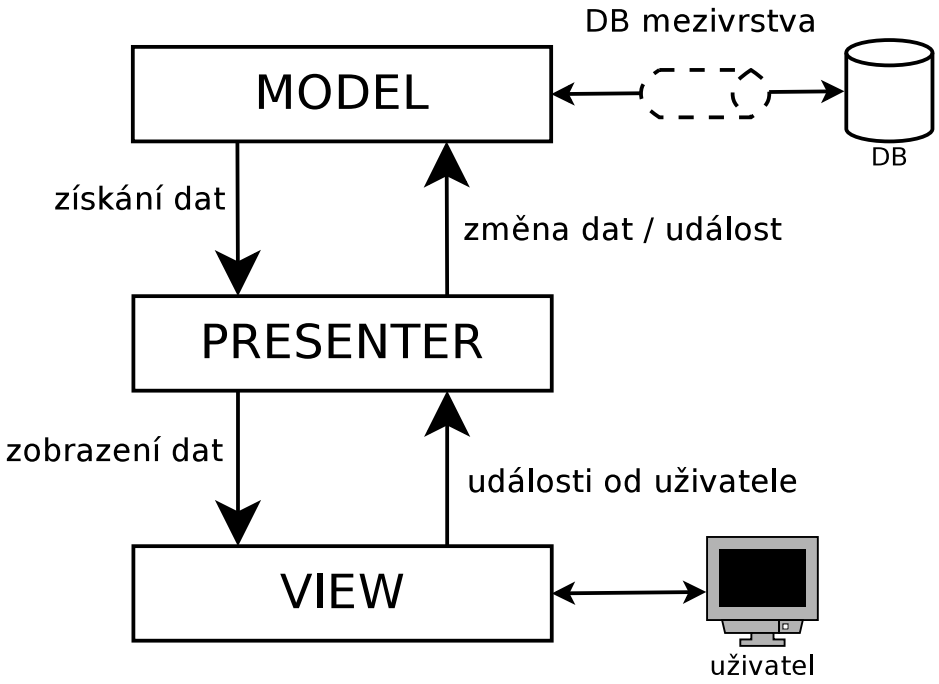
[14] Využití neošetřeného vstupu kdy útočník do stránky podstrčí svůj kód

[15] Útočník přiměje uživatele navštívit stránku, která skrytě vyvolá útok na webovou aplikaci

4.2.2 MVP (Model-View-Presenter)

Návrhový vzor odvozený od `MVC` (**M**odel-**V**iew-**C**ontroller). Hojně využívaný nejen pro tvorbu webových aplikací. Jeho použitím dojde k oddělení kódu aplikace do tří částí:

- **Model**
  Zajišťuje aplikační logiku, což **není** jen komunikace s databází. Nezávislá na reprezentaci dat.
- **View**
  Jedná se o šablonu zobrazující data z presenteru. Využívá šablonovací systém `Latte` (viz 4.2.3).
- **Presenter**
  Hlavní část, která řídí a spojuje dvě předcházející. Má na starost získat data z modelu a předat je šabloně.



Obrázek 4.3: Návrhový vzor `MVP`

4.2.3 Šablonovací systém Latte

Spojení `PHP` a `HTML` nevypadalo vždy přehledně, jelikož vždy vznikal nepřehledný špagetový kód.[16] Šablony mají za úkol zvýšit přehlednost a čitelnost kódu. Díky nim je daleko lepší týmová spolupráce, kdy kodér může pracovat nezávisle na programátorovi a bez znalosti `PHP`. V šablonách se používají tzv. `makra`, která jsou silnou stránkou `Latte` – výrazně

[16] Nesrozumitelný a neudržovatelný programový kód

zjednodušují kód a urychlují práci. Dále nabízí `helpery`, pomocí nichž snadno použijeme funkce, které pomáhají upravit nebo přeformátovat data do výsledné podoby.

Šablony nijak nesnižují výkon aplikace, protože se ukládají do cache na disk v podobě nativního `PHP` kódu. Šablony se automaticky generují po každé změně zdrojového souboru.

Příklad výpisu položek seznamu

Jak můžeme vidět, při použití šablonovacího systému `Latte` je kód daleko přehlednější a intuitivnější.

```
<ul n:if="count($items)">
        <li n:foreach="$items␣AS␣$item">{$item->name}</li>
</ul>
```

Kód 4.1: Výpis pomocí `Latte`

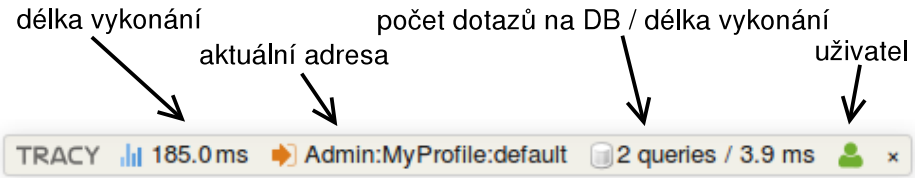
```
<?php

if(count($items)){
        echo "<ul>";
        foreach($items AS $item){
                echo "<li>".$item->name."</li>";
        }
        echo "</ul>";
}
```

Kód 4.2: Výpis běžným spojením `PHP` a `HTML`

4.2.4 Ladění a zpracování chyb

`PHP` nemá implementovány nástroje pro ladění, pouze nás upozorní na to, o jakou chybu jde a na jakém řádku se vyskytla. Proto `Nette` přichází s knihovnou Tracy (známá také pod názvem Laděnka). Při výskytu chyby vidíme část zdrojového kódu se zvýrazněným řádkem, který chybu vyvolal. Dále máme možnost procházet call stack, [17] z něhož zjistíme, které metody byly volány (včetně argumentů) a v jakém pořadí (viz obrázek 4.5).

Dalším nástrojem pro informace o aktuálním běhu aplikace můžeme získat z ladícího panelu (viz obrázek 4.4), jenž je užitečný pro kontrolu optimalizace aplikace – zjistíme celkový čas zpracování dotazu a jednotlivé časy zpracování dotazů na databázi.



Obrázek 4.4: `Tracy` debugger bar (panel pro ladění)

[17] Zásobník volání

Zmíněné ladící nástroje jsou k dispozici pouze v ladícím módu aplikace. Veškeré logy o chybách z Laděnky jsou uchovávány na disku. Tímto vývojář snadno dozví o vzniklých chybách a lehce je lokalizuje.

Obrázek 4.5: Laděnka (Nette\Diagnostics\Debugger)

4.3 Ostatní technologie

Mezi další použité technologie patří:

4.3.1 jQuery

JavaScriptová knihovna běžící na straně prohlížeče, která slouží k manipulací s DOM strukturou. Její výhody jsou:

- oddělené chování od struktury HTML
- zjednodušení práce
- přehlednost zdrojového kódu
- podpora většiny prohlížečů
- abstrahování od typu prohlížeče z hlediska vývoje

Využita bude hlavně pro interakci prvků aplikace a snadnější použití `AJAX`u. Dostupný z `http://jquery.com`.

4.3.2 Bootstrap

Front-endový framework pro vytváření moderních webových aplikací. Využit hlavně kvůli komponentám, které nabízí (modální okna, formuláře, atd.) a dále responzivnímu designu. Dostupný z http://getbootstrap.com.

Kapitola 5

Návrh aplikace

V následující kapitole se zaměřím na návrh aplikace, kde zmíním případy užití, návrh ER diagramu a samotný návrh uživatelského rozhraní, který bude rozdělen na část frontendu a backendu aplikace.

5.1 Diagram případu užití

Vytvořen kvůli zachycení chování aplikace z hlediska jejich aktérů. Tím je umožněno aplikaci rozdělit na podproblémy a zabývat se jimi zvlášť. Využití měl také při odhadu vývoje aplikace, kdy jsem si postupně odškrtával části, které již byly implementovány. V následujících sekcích bych se zaměřil na vybrané případy užití. V příloze B.3 lze nalézt vytvořený diagram, jenž je pro přehlednost více zaměřen na `Frontend` aplikace a obsahuje tři základní aktéry.

5.1.1 Nepřihlášený uživatel

V následujících odstavcích se budu věnovat možným případům užití z hlediska nepřihlášeného uživatele (nemá aktivní sezení). Takový uživatel bude disponovat základními možnosti, které jsou patrné z diagramu B.3.

Registrace

Nutný krok, pomocí kterého návštěvník vytvoří svou jedinečnou identitu v rámci aplikace a díky ní mu bude umožněno využívat poskytovaných služeb. Registrace je koncipována tak, aby pro návštěvníka nebyla nijak komplikovaná a zdlouhavá. Registrační formulář se pro přehlednost skládá ze čtyř sekcí: kontaktní, fakturační, firemní a přístupové údaje. Z čehož jsou povinné pouze položky jméno a email pro komunikaci se zákazníkem, položky uživatelské jméno a heslo pro autentizaci uživatele. Při zadávání hesla je nutné ho zopakovat z důvodu snížení rizika překlepu, který může uživatel nechtěně způsobit. Po dokončení registrace není nutné účet schvalovat administrátorem či aktivovat pomocí odkazu v emailu. Účet uživatele se takto stává okamžitě platným.

Přihlášení

Odkaz pro přihlášení a registraci bude dostupný z uživatelské navigace na každé stránce aplikace. Stránka pro přihlášení bude obsahovat pouze formulář pro zadání uživatelského

jméno a hesla; a dále instrukce pro řešení problémů s přihlášením – ztráta přihlašovacích údajů.

Odhlášení

Bude probíhat automaticky po vypršení platnosti či iniciováno ze strany uživatele. Délku platnosti bych volil v řádu hodin z důvodu rizika zneužití účtu neoprávněnou osobou.

Prohlížení vložených materiálů

Uživateli je umožněno prohlížet vložené materiály a zobrazovat jejich detail. Materiály, které bude chtít zakoupit, může průběžně vkládat do košíku. Jejich zakoupení však nebude moci provést, jelikož je vyžadováno přihlášení.

Změna lokalizace

Záhlaví bude obsahovat seznam podporovaných lokalizací – nejspíše formou vlaječek. Po kliknutí se přepne aplikace do vybrané lokalizace. Současně se změní i `URL` adresa. Čeština bude výchozí lokalizace.

5.1.2 Přihlášený uživatel

Rozšiřuje práva nepřihlášeného uživatele. Oproti běžnému návštěvníkovi je mu v rámci celé aplikace přidělena jedinečná identita. Díky ní aplikace pozná, se kterým uživatelem komunikuje.

Zakoupení materiálu

Uživatel si již může zakoupit materiály. Zakoupení lze provést dvěma způsoby. Prvním z nich je tzv. rychlé zakoupení, kdy je provedeno pomocí tlačítka „Zakoupit“ u detailu materiálu. Takto je uživateli okamžitě zpřístupněno stažení souboru. Druhý způsob slouží pro hromadné zakoupení, kdy uživatel postupně vkládá materiály do košíku a následně zakoupí všechny položky.

Správa profilu

Do této sekce se uživatel dostane po kliknutí na své jméno v uživatelské navigaci. Tato sekce se týká pouze údajů k přihlášenému uživateli a skládá se z následujících podsekcí:

- **Kontaktní údaje**
  Změna kontaktních, fakturačních a firemních údajů.
- **Přístupové údaje**
  Uživatel si může zvolit jiné uživatelské jméno či heslo. Pro změnu je nutné zadat aktuálně používané heslo.
- **Správa kreditu**
  Zde má uživatel možnost nahlédnout do historie svého kreditu a jeho dobití.
- **Zakoupené materiály**
  Stránka bude obsahovat tabulku, ve které uživatel bude mít uvedené zakoupené materiály. Z důvodu lepšího vyhledání může být výpis filtrován dle vybraných kriterií.

Řízení přípravy materiálů

Oproti analýze požadavků (viz 3.2) bylo rozšířeno o další prvky, které detail úkolu činí uživatelsky přívětivějšími. Přibyl například progress bar,[1] který vizuálně naznačuje zbývající čas pro odevzdání materiálu.

5.1.3 Administrátor

Dědí veškeré vlastnosti od přihlášeného uživatele. Díky tomu má také přístup do sekcí ve `Frontendu`. Takto si i administrátor může zakoupit materiály či celkově využívat nabízených služeb aplikace. Role je rozšířena pouze o přístup do administrační sekce, která obsahuje moduly pro správu aplikace. Z důvodu zachování přehlednosti byl diagram případu užití abstrahován od této části.

5.1.4 Super administrátor

Uživatel s neomezenými právy. Důvod jeho existence je popsán v sekci 6.2.

5.2 Návrh ER diagramu

Pro návrh relační databáze jsem vycházel z konceptuálního modelu tzv. ER diagram.[2] Ten aplikaci abstrahuje do entitních množin, kde každá entita je v rámci množiny za všech okolností jednoznačně identifikovatelná (pomocí primárního klíče). Dále entita obsahuje množinu atributů sloužící k uchování informací, které budou ukládány do databáze. Primární klíč je od běžných atributů odlišen tučným písmem. Entitní množiny jsou mezi sebou propojeny pomocí vztahů, které znázorňuje čára mezi nimi. Na každém konci tohoto vztahu je uvedena kardinalita. Následně bych se více zaměřil na vybrané části ER diagramu. Kompletní diagram lze naleznout v příloze B.1.

5.2.1 Entitní množina `acl_users` a `user_data`

Entitní množina `acl_users` je základem aplikace, jelikož její entita představuje jednoho uživatele aplikace. Takto může uživatel po přihlášení získat svou identitu. Obsahuje základní atributy o uživateli jako jeho jméno (`name`), `email`, uživatelské jméno (`username`) a heslo (`password`). Heslo je z důvodu bezpečnosti uloženo jako `md5` hash, vypočtený z konkatenace hesla a tzv. kryptografické soli.[3] Díky takovému hashi bude mít stejné heslo různý zakódovaný tvar a nebude z něj možné získat pomocí slovníkového útoku původní heslo. Další atributy jsou již pro systémové účely, které následně popíši. Atribut `super_admin` slouží k identifikaci, zda uživatel náleží do role `super admin` (viz 6.2). Atribut `registred_date` uchovává datum vytvoření účtu `last_login_date` datum posledního přihlášení – oba slouží pouze pro statistické účely. Atribut `userChangedDate` uchovává datum poslední změny profilu a uživatelských rolí, do kterých uživatel patří. Důvod vzniku je pouze kvůli cachování, kdy docílím omezení dotazů na databázi a tím se chod aplikace zrychlí (viz 6.2). Atributy

[1] Prvek uživatelského rozhraní znázorňující průběh činnosti

[2] Entity-relationship model

[3] Náhodně zvolený řetězec, který by v rámci aplikace měl zůstat neměnný. V případě jeho změny by bylo nutné vygenerování nového hashe ke všem uživatelům

reset_password_hash a reset_password_expiration slouží pro obnovu hesla, kdy je uživateli emailem zaslán časově omezený odkaz (plánuji 1 hodinu), jenž slouží pro následnou změnu. Pokud lhůta vyprší, musí si uživatel podat novou žádost.

Další částí je entitní množina user_data která má vztah 1:1 s acl_users. Kardinalita těchto dvou tabulek svádí k jejich sloučení. K tomuto návrhu mám však odůvodnění. Entitní množina acl_users obsahuje globální atributy nutné pro jakoukoliv aplikaci. Zato druhá, user_data, má již atributy specifické pro vyvíjenou aplikaci. Díky tomu lze modul pro správu uživatelů (včetně struktury tabulek) použít v jiné aplikaci a změny proběhnou pouze v entitní množině user_data (či vůbec nebude existovat).

Co se týče specifických atributů pro vyvíjenou aplikaci, obsahuje kontaktní, fakturační a firemní údaje; a dále aktuální stav kreditu (credit). Hodnota kreditu bude vypočtena agregační funkcí z entitní množiny user_credit_history pomocí triggeru.

5.2.2 Entitní množiny pro správu materiálů

Sekce s materiály jsou organizovány do složek, na což slouží entitní množina conferences_categories. Kromě primárního klíče a atributů popisující položku, obsahuje atribut dir. Pomocí něj mohu určovat, zda je daná položka je adresář, či kategorie sdružující sekce s materiály (entitní množina conference_sections). Entitní množina conferences_categories má sama se sebou vztah 1:N z důvodu hierarchického sdružování složek.

U entitní množiny conference_sections bych zmínil atribut position sloužící k nastavení pozice sekce v rámci kategorie. Takto administrátor může libovolně měnit jejich pořadí.

Entitní množina conference_files již obsahuje samotné materiály. Materiály nejsou ukládány do databáze, ale fyzicky v souborovém systému webhostingu s názvem dle vygenerovaného primárního klíče. Pouze v atributu type je uveden typ nahraného souboru a v atributu size jeho velikost. Takto se aplikace dotazuje na soubor pouze v případě požadavku na jeho stažení.

5.2.3 Entitní množiny pro přípravu materiálů

Údaje o zadaném úkolu obsahuje entitní množina preparing_materials_tasks. Atribut name obsahuje název zadaného úkolu a description samotné zadání. Pro omezení doby k vypracování materiálu slouží atribut date_begin s počátečním datem a date_end s koncovým datem. Pro odlišení hotových úkolů a zjištění jeho aktuální fáze slouží atribut status.

Entitní množina preparing_materials_files již obsahuje vložené soubory. S entitní množinou se zadaným úkolem tvoří vztah o kardinalitě 1:N z důvodu, že úkol umožňuje verzování nahraných souborů. Aktuální soubor se vždy bere dle atributu datetime, obsahující datum a čas nahrání souboru. Atribut change_description slouží k popsání změn, které byly provedeny. Podobně jako u vložených materiálů je ukládán typ souboru type a jeho velikost size.

5.3 Návrh uživatelského rozhraní frontendu

Při návrhu se dbalo na jednoduchost a pohodlné ovládání na zařízeních z různou úhlopříčkou displeje. Při návrhu jsem se snažil maximálně využít možnosti Bootstrap frameworku, který nabízí řadu připravených komponent. Jediný problém, na který jsem u něj narazil,

byl omezený balíček ikon. Jejich autor `Glyphicons.com`[4] nabízí rozšířené balíčky, ale již placené. Z tohoto důvodu jsem chybějící ikony využil z projektu `Font Awesome`,[5] který jich nabízí bezplatně podstatně více. V následujících podsekcích bych zmínil jednotlivé části grafického návrhu aplikace.

5.3.1 Hlavička

V levé části je umístěno logo se sloganem, který zároveň slouží k přechodu na hlavní stránku aplikace – rozcestník. V pravé části se nachází uživatelské menu a vlaječky pro změnu lokalizace aplikace. Položky uživatelské navigace jsou proměnlivé v závislosti na tom, zda je uživatel přihlášen. Pro nepřihlášeného uživatele se zobrazí položky košík, registrace a přihlášení. Naopak přihlášený uživatel má k dispozici košík, aktuální stav kreditu, svůj profil a odhlášení.

5.3.2 Obsah aplikace

Před obsahem se vždy nachází pruh, který informuje uživatele o tom, na jaké stránce se nachází. Pod tímto pruhem se v případě potřeby zobrazí navigace, která stránku dělí do podstránek či má jinou funkci. Následně bych popsal tři základní rozvržení obsahu pro vytvořené moduly.

Rozcestník

Výchozí stránka aplikace, kde se nachází odkazy na jednotlivé stránky. Počet odkazů na řádce je dynamický v závislosti na šířce obrazovky. Pro rychlejší orientaci jsou odkazy obohaceny o piktogram.

Správa materiálů

Složky mají stejný vzhled jako rozcestník. Po zobrazení složky s materiály jsou sekce rozděleny do řádků, které jsou odlišeny barvou pozadí. Jednotlivé řádky již zobrazují vložené materiály ve formě ikony souboru. V souboru se dále nachází ikona symbolizující typ souboru. V případě přihlášeného uživatele jsou soubory barevně rozlišeny na zakoupené a nezakoupené. Po kliknutí na soubor se pomocí modálního okna zobrazí jeho detail.

Před obsahem stránky se nachází navigace sloužící pro filtrování zobrazený materiálů. Uživatel si tak může nechat zobrazit pouze materiály, které jsou bezplatné, zakoupené či nezakoupené.

Příprava materiálů

Výchozí stránkou je tabulka se zadanými úkoly. Uživatel tak přehledně vidí zadané úkoly s jejich statusem a termínem odevzdání. Po zobrazení úkolu je stránka rozdělena do dvou sloupců, kde v levém se nachází zadání úkolu a v pravém informace o datu odevzdání včetně aktuálního stavu úkolu. Pro grafické znázornění zbývajícího času jsem využil komponentu *progress bar* z `Bootstrapu`. *Progress bar* je dále obohacen i o textový popis ve formě: *zbývá X dnů a X hodin.*

[4] Dostupné z http://glyphicons.com
[5] Dostupné z http://fortawesome.github.io/Font-Awesome/

Stránka pro nahrání nové verze materiálů se skládá ze tří částí. První z nich je samotný formulář pro nahrání, který se skládá ze dvou položek – výběr souboru a popis změny. Druhou část tvoří box obsahující poslední nahraný soubor. Poslední částí je repozitář všech nahraných materiálů včetně popisu změn.

5.3.3 Patička

Patička je umístěna fixně ke spodnímu okraji stránky. Obsahuje pouze odkazy na stránky informativního charakteru jako obchodní podmínky a kontakt na provozovatele. V budoucnu může také sloužit pro odkazy na další projekty.

5.4 Návrh uživatelského rozhraní backendu

Část určená pro administraci aplikace. Hlavní požadavek byl, aby byla přes celé okno prohlížeče z důvodu využití veškerého místa. Rozvržena je do následujících čtyř sekcí: nahoře hlavička, vlevo menu, vpravo obsah a dole patička. Zde jsou naplno využity možnosti `Bootstrapu` – od komponent až po responzivní design, umožňující snadnou práci z mobilních zařízení.

5.4.1 Hlavička

Hlavičce dominuje vlevo logo a vpravo uživatelská navigace. Ta obsahuje důležité prvky jako: informaci o přihlášeném uživateli, odkaz na přechod do `frontendu` aplikace, odhlášení a změnu lokalizace zadávaného obsahu. Navigace je plně responzivní, tudíž při menší šířce displeje dojde k jejímu skrytí. Pro následné zobrazení slouží piktogram v pravé části hlavičky.

5.4.2 Navigace

Zabírá veškerou výšku obsahu. Položky je možné libovolně hierarchicky zanořovat. Pro zajištění přehlednosti jsou vždy zvýrazněny položky od aktivní položky směrem k položce nejvyšší úrovně. Navigace má v počátečním stavu zobrazené pouze položky nejvyšší úrovně. Po aktivaci některé z nich či najetí kurzorem myši, dojde k zobrazení podúrovně. V případě menší šířky displeje by navigace zabírala potřebné místo pro obsah. Proto jsem přistoupil k jejímu skrytí při šířce displeje menší než 768px. Navigace automaticky „zajede“ vlevo. Pro její zobrazení slouží piktogram v hlavičce vlevo od loga či je možné použití gesta na dotykových obrazovkách.

5.4.3 Obsah

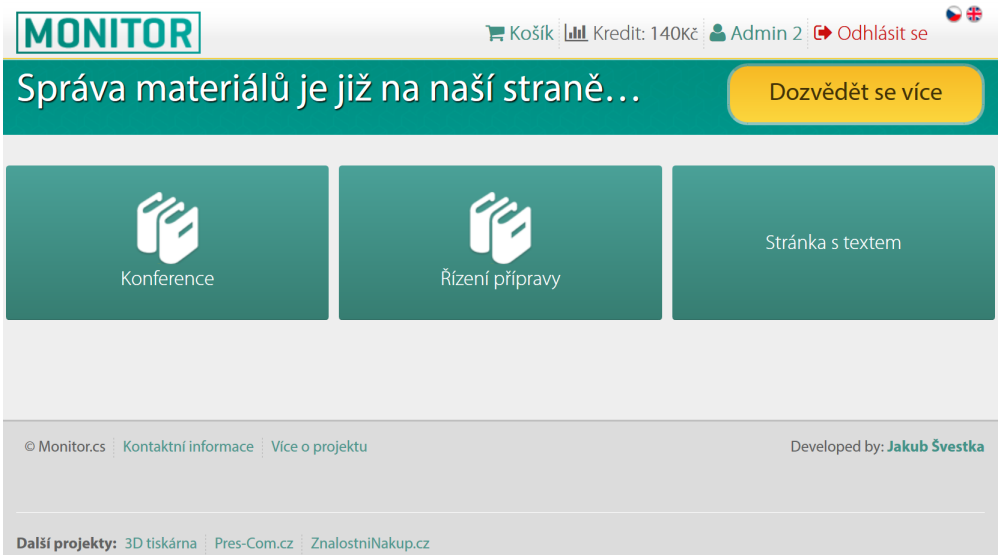
Aby uživatel na jakékoliv stránce věděl, kde se přesně nachází, umístil jsem těsně pod hlavičku drobečkovou navigaci. Ta přehledně znázorňuje cestu, pomocí níž se uživatel dostal k aktuální stránce a nabízí mu možnost se vrátit o několik kroků zpět. Obsahové okno zabírá cca 75 % šířky okna prohlížeče. Při aktivaci responzivního chování hlavní navigace dojde k jejímu rozšíření na celých 100 %.

5.4.4 Patička

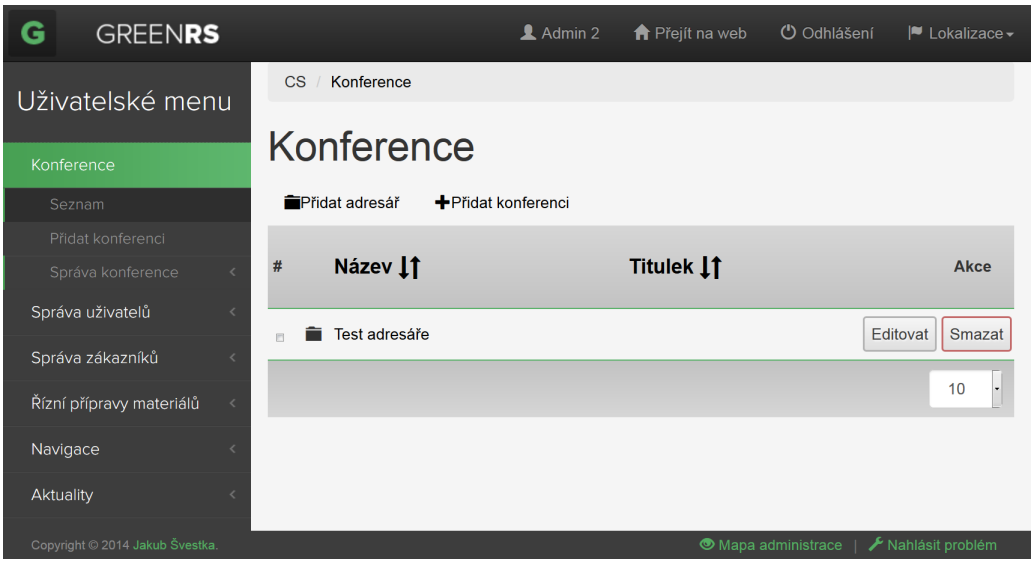
Obsahuje pouze odkaz na mapu administrace a odkaz pro nahlášení problému. Její výška je co nejmenší, aby zbytečně nezabírala místo (hlavně na menších obrazovkách).

5.5 Ukázky návrhu

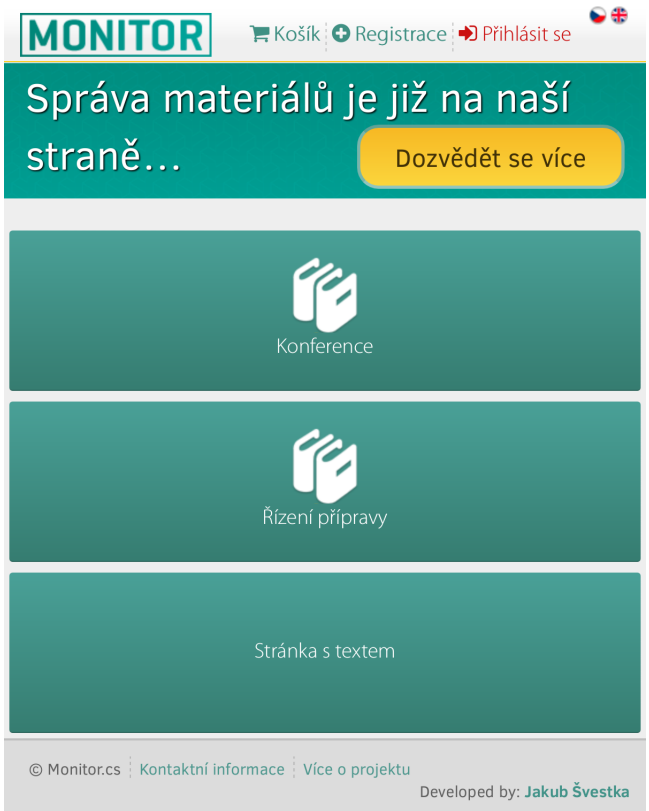
Snímky pořízeny z prohlížeče Firefox na platformě Windows a Android.



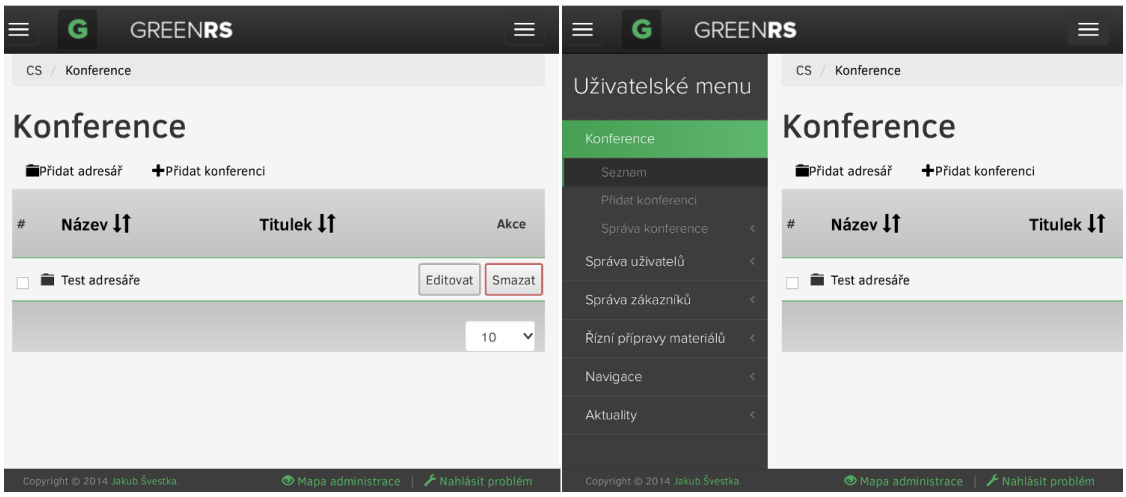
Obrázek 5.1: Desktopové zobrazení stránky s rozcestníkem (frontend)



Obrázek 5.2: Desktopové zobrazení stránky se správou materiálů (backend)



Obrázek 5.3: Mobilní zobrazení stránky s rozcestníkem (frontend)



Obrázek 5.4: Mobilní zobrazení stránky se správou materiálů (backend). Vpravo je zachycen stav, po zobrazení menu.

Kapitola 6

Implementace

V této kapitole bych se již zaměřil na popis implementace. Při implementaci jsem se maximálně držel návrhového vzoru `MVP`. Celá aplikace je řešena modulárně, tudíž lze například celou administraci či její submoduly využít u jiného projektu. Velký důraz byl kladen na zabezpečení aplikace, jelikož pracuje s virtuálními penězi formou kreditu, který by mohl být cílem zneužití neoprávněnou osobou. Z důvodu větší rozsáhlosti aplikace bych se následně zaměřil na podstatné části, které jsou implementačně zajímavé či jsem u nich řešil problémy, ze kterých bych některé rád vyzdvihl.

6.1 Struktura aplikace

Pro návrh celého systému bylo nutné zvolit vhodnou strukturu umístění kódu z důvodu rozšířitelnosti o další funkce a také přehlednosti. Danému kritériu nejlépe odpovídají hierarchické moduly, které lze různě zanořovat na tzv. submoduly. Modulem se rozumí adresář, který rozčleňuje presentery a šablony do spolu souvisejících celků. V aplikaci jsou použity dva hlavní moduly v kořenovém adresáři aplikace, kde první z nich je `Front` modul sloužící části webu viditelné běžným návštěvníkům a do určitých částí, jako procházení dostupných dokumentů, nevyžaduje autentizaci. Druhý, `Admin` modul, slouží pro správu aplikace a vyžaduje speciální oprávnění pro přístup. Moduly dále obsahují následující submoduly sdružující již specifické části aplikace:

6.1.1 Front modul

- **Materiály**
  Zobrazení detailu a zakoupení nabízených materiálů.
- **Uživatel**
  Veškeré funkcionality týkající se uživatele a jeho účtu, a to: registrace, správa profilu, správa kreditu, zakoupené materiály. Dále je zde implementován košík.
- **Příprava materiálů**
  Uživatel zde nalezne zadané úkoly. Po zobrazení úkolu má možnost procházet nahrané soubory či nahrát nový.
- **Navigace**
  Umožňuje pohyb v aplikaci – zobrazení statických stránek a přechod do jiného modulu (zakoupení či příprava materiálů).

6.1.2 Admin modul

- **Materiály**
  Vytváření hierarchických adresářů, jenž obsahují události (konference). Událost se skládá z bloků s dostupnými materiály.

- **Správa uživatelů**
  Nejedná se jen o správu uživatelů systému, nýbrž i o přístupová práva (viz 6.2).

- **Správa zákazníků**
  Zde se na uživatele systému nahlíží jako na zákazníka. Nabízí tedy správu fakturačních údajů, kreditu a materiálů, k nimž je mu umožněn přístup.

- **Řízení přípravy materiálů**
  Slouží k vytvoření úkolů k vypracování pro určitého uživatele. Administrátor má přehled o zadaných úkolech, kde po výběru úkolu lze nahlížet do repozitáře nahraných souborů.

- **Navigace**
  Slouží k vytvoření obsahu aplikace, kde vytvořená stránka může obsahovat předdefinovaný modul či se jedná pouze o běžnou stránku s textem. Další funkcí je správa ikon, které se mohou použít k dekoraci tlačítka v rozcestníku.

6.1.3 Výchozí adresářová struktura

Vychází z připraveného balíku *SandBox* – základní skeleton aplikace, který již obsahuje *Nette Framework*. Obsahuje následující adresáře:

- `/app` – Soubory aplikace, dle `MVP` obsahuje modelovou, pohledovou a prezentační vrstvu. Vše je logicky sdružováno do modulů (viz struktura aplikace v příloze C).

- `/log` – Zde jsou uchovány logovací soubory zachycující výjimky, které v aplikaci nastaly. Vhodné pro ladění aplikace.

- `/temp` – Adresář pro dočasné soubory, zkompilované šablony a cache frameworku.

- `/vendor` – Obsahuje *Nette Framework* a knihovny, které aplikace používá.

- `/www` – Veškeré soubory, které musí být přístupné pro klienta jako grafické soubory, kaskádové styly a JavaScriptové soubory. Dále obsahuje soubor `index.php`, jenž je vstupním bodem do aplikace.

6.2 Přístupová práva

Tzv. `ACL`[1] sloužící ke správě oprávnění uživatelů. Takto lze vytvořit specifickou skupinu uživatelů, kterým můžeme omezit určité částí aplikace. `Nette` framework disponuje předpřipravenou třídou `Nette\Security\Permission` pro řízení práv a přístupu. Definice přístupových práv může být provedena dvojím způsobem:

[1] Access control list

- **Staticky**
  Předem definované přímo ve třídě implementující *Permission* (Authorizator), anebo nepřímo v konfiguračním souboru `Neon`. Tento způsob je vhodnější pro rozsahově menší aplikace, kde se práva nebudou často měnit. Veškerá změna probíhá odbornou osobou – správcem, programátorem.

- **Dynamicky**
  Veškerá data týkající se přístupových práv jsou uložena v databázi. Lze je dynamicky měnit v administrační sekci. To znamená, že administrátor může variabilně vytvářet uživatelské skupiny, a takto na ně aplikovat potřebná omezení. Nevýhodou může být mírné zpoždění aplikace zapříčiněné dotazy na databázi.

Přistoupil jsem k dynamické variantě z důvodu snadné flexibility administrátorem a možnosti využití při dalších projektech. Implementuje ji třída `Greenrs\Model\Authorizator`, která načte z databáze: role, zdroje a práva s akcemi. Dále automaticky vytvoří roli *super admin*, jež má neomezená práva. V systému musí vždy existovat nejméně jeden uživatel s touto rolí z důvodu možné situace, kdy by si administrátoři omezili práva natolik, že by žádný neměl neomezená práva (byla by nutná manuální oprava v databázi).

Části přístupových práv:

- **Práva**
  Samotná práva slouží pro povolení či zakázání a skládají se z role, zdroje a akce.

- **Role**
  Sdružují skupinu uživatelů. Lze je hierarchicky zanořovat.

- **Zdroje**
  Je v nich hierarchicky uložena struktura aplikace – tzn. vazby mezi moduly, submoduly a presentery.

- **Akce** Seznam akcí, které lze provádět – například: zobrazení, editace, přidání atd.

6.2.1 Programátorský mód

Správu zdrojů a akcí nemůže provádět běžný administrátor, protože se zde pracuje s položkou `Klíč`, jež je vázána na programovou část aplikace. Jelikož jejich změna nebude častá – zpravidla pouze při návrhu, rozhodl jsem se k zavedení programátorského módu. Ten se nastavuje v konfigurační souboru aplikace v sekci `parameters` → `acl` → `prog_mode`. Pomocí něj mohu povolit či zakázat přidání/editaci zdrojů a akcí.

6.2.2 Využití cachování

Načtení přístupových práv se skládá z desítek dotazů na databázi, protože zdroje a role tvoří hierarchickou strukturu. Platí tedy, že čím větší zanoření, tím více dotazů na databázi. Při implementaci jsem tuto skutečnost značně pocítil – dotazy zabraly cca 14ms systémového času. Změny přístupových práv a celkově dat, na kterých jsou závislá, nebudou probíhat často, proto není nutné je pokaždé načítat z databáze. K tomuto jsem využil službu pro cachování (`Nette\Caching\Cache`) umožňující uložit do perzistentní formy náročně získaná data pro příští použití. Takto mohu uložit do cache celou instanci třídy `\Authorizator`. Po této změně se razantně snížil čas pro získání dat z databáze na pouhé 2ms. V případě

jakékoliv změny v právech provedu zneplatnění cache, jež obsahuje třídu `\Authorizator`, a tím dojde k nacachování její nové instance.

6.3 Identita uživatele

Po úspěšné autentizaci získá uživatel v rámci aplikace svou identitu, jež implementuje třída `Greenrs\Model\Identity` rozšiřující výchozí `\Nette\Security\Identity`. Ta obsahuje unikátní identifikační číslo uživatele, role, do kterých náleží, a uživatelovo jméno (pro zobrazení v aplikaci). Důvodem vytvoření vlastní identity bylo nežádoucí chování výchozí implementace. Konkrétně se jedná o statický stav identity, kdy v průběhu uživatelské relace neproběhne kontrola, zda je stále aktuální. Tímto může nastat situace, kdy již přihlášenému uživateli bude změněn seznam rolí, do kterých náleží, či bude ze systému odstraněn. Pak se tato skutečnost neprojeví do jeho identity – uživatel bude mít stále identitu, jež byla vytvořena při přihlášení.

Z výše uvedeného důvodu jsem přistoupil k dynamické změně identity v průběhu relace. Jelikož by nebylo ideální vytvářet identitu při každém požadavku od uživatele, vyřešil jsem to následující optimalizací. U každého uživatele je v databázi vedena položka `userChangedDate` udávající datum a čas poslední změny jeho účtu. Při jakékoliv změně ji nastavuji na aktuální hodnotu. Pokud se změna týká rolí, na starost to již mají dva trigerry definované v tabulce `acl_users_roles` – jeden je spuštěn při vložení role do tabulky a druhý při jejím odstranění.

Tímto při každém požadavku uživatele kontroluji, zda čas poslední změny koresponduje s časem zaznamenaným v identitě uživatele. Pokud se jedná o totožnou hodnotu, provedu načtení identity z cache. V případě změny cache zneplatním, a tím dojde k vytvoření nové instance identity s aktuálními daty.

6.4 Vícejazyčnost

Dle požadavků byla aplikace vyvíjena tak, aby podporovala více jazykových mutací. Aktuálně podporuje českou a anglickou. Výchozí lokalizace se nejprve volí dle atributu `Accept-Language` v hlavičce `HTTP` protokolu. Pokud z ní není zjištěna preferovaná lokalizace, je jako výchozí použita česká. Bližší popis implementace rozdělím na část věnovanou překladu statických textů a na část zabývající se dynamickými texty, které jsou uloženy v databázi.

6.4.1 Statické překlady

Jedná se o překlad textů, jenž jsou v aplikaci neměnné – nepracují s databází. Příkladem mohou být popisky formulářových prvků, chybové hlášky atd. Pro tuto část jsem vybral knihovnu `Kdyby/Translation`.[2] Je nutné ji v `Nette` zaregistrovat jako rozšíření, protože není běžnou součástí `Nette`.

Překlady jsou uloženy v adresáři `/app/lang/`, kde jsou sdružovány do třech kategorií:

- *controls* – překlady ovládacích prvků (tlačítka, popisy prvků formulářů)
- *messages* – překlady zpráv, kterými aplikace komunikuje s uživatelem
- *texts* – překlady různých textů, jenž mají informativní charakter

[2] Dostupná z https://github.com/Kdyby/Translation

Každé kategorii odpovídá soubor, jehož název odpovídá následující masce:

```
<category>.<language>.<type>
```

Jako typ souboru byl použit formát `neon`[3] z důvodu, že se jedná o prostý text – snadná úprava, není potřeba speciálních nástrojů (jak je tomu u `Gettextu`). Například pro překlad zpráv jsou použity soubory `messages.cs_CZ.neon` a `messages.en_US.neon`.

Struktura souboru s překladem

Struktura je podobná jazyku `YAML`. Překlady lze sdružovat do souvisejících celků pomocí tabulátoru (pro přístup k překladu se následně používá tečková notace). V následujícím kódu můžete vidět příklad zapsání překladu, který automaticky zajistí zvolení správné tvaru dle počtu (pluralizace).

```
preparing:
   reamingDays: "%count% den|%count% dny|%count% dnu"
   reamingHours: "%count% hodina|%count% hodiny|%count% hodin"
reaming: "zbývá|zbývají|zbývá"
```

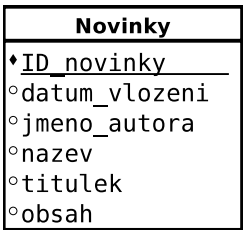
Kód 6.1: Zápis překladu pomocí `NEON`u

```
translate("reaming").": ".
translate("preparing.reamingDays", 2)." & ".
translate("preparing.reamingHours", 0);
//vypíše: zbývají 2 dny & 0 hodin
```

Kód 6.2: Použití výše zmíněného překladu pro výpis zbývajícího času

6.4.2 Dynamické překlady

Překlady potřebných atributů u záznamů vkládaných do databáze – například název a popis u přidaného materiálu. Zde se nabízí dva optimální způsoby[14], jak toho docílit. Znázornění návrhu bude vycházet z tabulky novinky (obrázek 6.1, kde je nutné uchovat překlad atributů název, titulek a obsah).

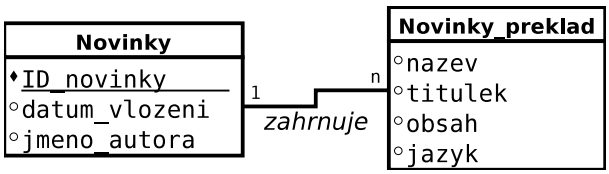


Obrázek 6.1: Struktura tabulky nepodporující uložení překladu

[3]Prostý textový soubor psaný `neon` syntaxí (podobná jazyku YAML) – viz http://ne-on.org/

Uložení překladů do samostatné tabulky

U tabulky, kde chceme podporu překladů, rozdělíme atributy do dvou kategorii. V první (hlavní) tabulce budou atributy, jenž nepotřebují překlad – datum vložení, jméno autora. V druhé se již budou nacházet atributy vyžadující překlad a doplňující atribut označující jazyk lokalizace (domény atributu budou dostupné lokalizace – cs, en). Na obrázku 6.2 můžeme vidět výslednou strukturu tabulek.

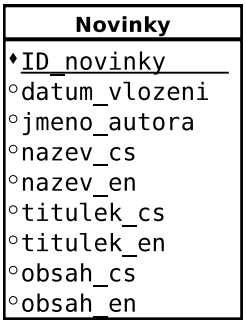


Obrázek 6.2: Uložení překladů do samostatné tabulky

Původně jsem v aplikaci použil výše zmíněné řešení z důvodu snadné rozšířitelnosti pro další lokalizace. Při implementaci jsem však narazil na potíže s položkami, které neměly přeloženy veškeré atributy. Pokud není nějaký atribut přeložen, vezme se hodnota z výchozí lokalizace (čeština), jež je dostupná vždy. Bohužel se mi požadovanou funkčnost nedařilo zajistit. Zároveň dotazy na databázi byly značně rozsáhlé a nepřehledné z důvodu spojování záznamů z tabulek (kardinalita `1:N`). Proto jsem přistoupil k následující (druhé) variantě.

6.4.3 Vytvoření atributů s překladem

Zde se atributy, včetně překládaných, nachází v jedné tabulce. Každý překládaný atribut je uveden tolikrát, kolik je potřeba uložit jeho překladů. Název překládaných atributů se liší pouze svým názvem, který obsahuje jako sufix zkratku lokalizace. Díky tomu máme všechny překlady uvedeny v dané n-tici (záznamu tabulky). Pokud daný překlad nemá atribut přeložen (obsahuje hodnotu `NULL`), vystačíme si s pouhým podmíněným příkazem `IFNULL`, jenž zajistí v případě hodnoty `NULL` získání hodnoty z atributu z výchozí lokalizace. Výslednou strukturu tabulky můžete vidět na obrázku č. 6.3



Obrázek 6.3: Uložení překladů do jedné tabulky do k tomu určeným atributům

```
SELECT ID_novinky, datum_vlozeni,
        IFNULL(nazev_en, nazev_cs) AS nazev,
        IFNULL(titlek_en, titulek_cs) AS titulek,
        IFNULL(obsah_en, obsah_cs) AS obsah
```

```
FROM Novinky
```

Kód 6.3: Dotaz na databázi, který při chybějící anglické lokalizaci atributu záznamu použije výchozí – českou

Z návrhu je patrné, že úprava škály podporovaných jazyků nebude jednoduchá a bude při ní potřeba změnit strukturu tabulky. Tuto nevýhodu lze částečně vyřešit pomocí skriptu, který nám požadované změny automaticky provede. Pokud se s tím spokojíme, dostaneme systém, kde dotazy budou pokládány elegantně a vyhodnocovány efektivně.

6.5 Kredit uživatele

Veškeré transakce s kreditem uživatele jsou zaznamenávány do tabulky `user_credit_history`. Při sečtení záznamů příslušících k danému uživateli pomocí agregační funkce v databází dostaneme jeho aktuální zůstatek kreditu. Jelikož by bylo časově náročné kredit při každém požadavku počítat (zobrazuje se na každé stránce v uživatelské navigaci), vytvořil jsem pro něj atribut `credit` v tabulce `user_data` (uchovávající data o uživateli). Pro zajištění aktuálnosti jeho hodnoty byl použit trigger, jenž se vykoná při přidání záznamu – tzn. dobití kreditu, zakoupení materiálu.

6.5.1 Dobití kreditu

Probíhá z uživatelského profilu – záložka správa kreditu. Původně se počítalo s dobití pomocí tzv. premium SMS, pro které jsem vytvořil potřebné `API`. Poté jsem ale zjistil, že tato služba je pro aplikaci nevýhodná – z původní zaslané částky od zákazníka zbude pro provozovatele zlomková část. Z tohoto důvodu jsem přistoupil k bezhotovostním transakcím (vyžaduje úkon správce, který přijetí platby potvrdí) a k online platbě pomocí kreditní karty (okamžité připsání kreditu – nevyžaduje úkon správce). Dobití kreditu je zahájeno vytvořením transakce. Při jejím vytvořením uživatel specifikuje hodnotu, o kterou chce navýšit svůj kredit. Po vytvoření má na výběr, zda transakci uhradí běžným bezhotovostním převodem či použije platební bránu. Pokud uživatel vytvořenou transakci nedokončí, má možnost se k ní v budoucnu vrátit. Implementace platební brány bude popsána v následující sekci.

6.5.2 Propojení aplikace s platební bránou

Aplikace je propojena s platební bránou *České spořitelny*[4] umožňující online platby pomocí kreditní karty. Takto má uživatel kredit okamžitě připsán na svém uživatelském účtě. Zároveň není nutný jakýkoliv zásah správce aplikace – vše je plně automatické. Převod transakce je zahájen zasláním formuláře protokolem `HTTP` metodou `POST` na stránku platební brány. Formulář obsahuje skrytá pole, v kterých je uvedeno:

- `merchantid` – jednoznačně identifikuje obchodníka, pro kterého má být platba určena (údaj přidělen v rámci smlouvy)
- `transactiontype` – udává, zda musí být platba schválena obchodníkem (zablokování částky na účtu zákazníka) či se jedná o přímý prodej (částka je z účtu zákazníka stržena okamžitě)

[4] Viz jejich nabídka na http://www.csas.cz

- `currency` – kód měny dle ISO 4217, [5] pro českou korunu byla použita hodnota „203“
- `merchantref` – unikátní identifikátor platby, použité identifikační číslo transakce
- `amount` – částka, která bude dobita – uvedená v haléřích tzn. 500 → 5.00 CZK
- `language` – jazyk platební brány

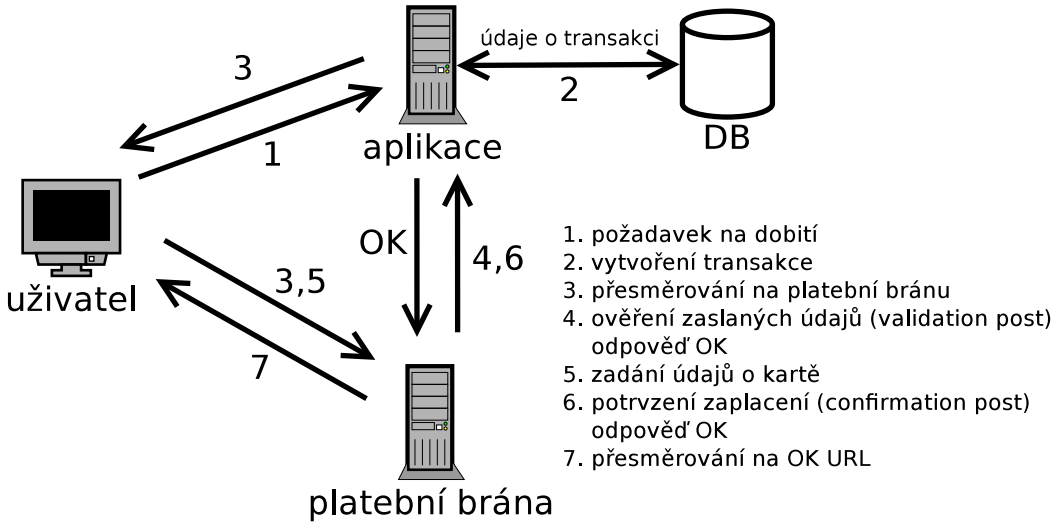
Pro ověření, zda data přijatá platební bránou odpovídají údajům uvedeným u transakce v databázi, slouží tzv. *validation post*. Ten je zaslán na speciální stránku aplikace a obsahuje veškeré údaje, jenž byly na platební bránu zaslány. Následně skript na stránce ověří vše oproti údajům v databázi, a pokud odpovídají, aplikace odpoví následující `[OK]` zprávou:

```
<html><head></head><body>[OK]</body></html>
```

Poté je umožněno uživateli zadat v platební bráně údaje o platební kartě a po zadání potvrdit platbu. Platební brána následně zasílá na stránku aplikace tzv. *confirmation post*, pomocí kterého v aplikace označím transakci za uskutečněnou a uživateli je tak navýšen kredit o hodnotu, jež koresponduje s hodnotou uvedenou v databázi u dané transakce. Aplikace následně odpoví `[OK]` zprávou, uvedenou výše. Pokud by stránka odpověděla jinak než `[OK]` zprávou, bere se transakce jako neúspěšná. V tomto případě platební brána zašle tzv. *rejection post* na stránku aplikace, pomocí něhož označím transakci jako neúspěšnou.

Pokud v průběhu nastane jakákoliv chyba (aplikace neodpoví `[OK]` zprávou), uživatel je přesměrován na **NOK URL** aplikace, kde ho informuji o chybě při provádění transakce. V takovém případě částka není uživateli stržena z účtu a kredit v aplikaci není navýšen. V opačném případě, pokud nenastala chyba, dojde k přesměrování na **OK URL** aplikace. Ta informuje o úspěšném navýšení kreditu.

Veškerá komunikace s platební bránou probíhá zabezpečeným šifrovaným protokolem `HTTPS`. Na stranu aplikace je vždy platební bránou v `POST` požadavku zasláno dohodnuté heslo, pomocí něhož ověřím totožnost platební brány.



Obrázek 6.4: Příklad komunikace s platební bránou v případě úspěšné platby

[5] Odkaz na standard: http://www.katpatuka.org/pub/doc/iso4217.htm

6.6 Způsob uložení materiálů

Veškeré soubory jsou uloženy fyzicky v souborovém systému webhostingu v adresáři `/www/-data/conferenceFiles/`. Pro propojení souboru se záznamem v databázi slouží jeho identifikační číslo, pomocí něhož je soubor pojmenován. Pro zajištění bezpečného uložení souborů, tak aby nemohly být staženy bez dostatečného oprávnění, je umístěn v tomto adresáři konfigurační souboru `.htaccess`.[6] Ten zajistí znepřístupnění všech souborů v adresáři přímým přístupem ze strany klienta pomocí protokolu `HTTP`. Soubory zůstanou zpřístupněny pouze se strany serveru, na kterém aplikace běží. Pro stažení jsou klientovi zaslány pomocí třídy `Nette\Application\Responses\FileResponse`.

6.7 Komponenty

Určité prvky aplikace, které byly použity na více místech, byly vytvořeny jako komponenty. Díky tomu se jejich implementace neopakuje a v případě jejich změny, úprava probíhá na jednom místě. Třída implementující komponentu dědí od `Nette\Application\UI\Control`. Kdybychom nahlédli do této třídy, zjistíme, že dále dědí od `PresenterComponent` a implementuje rozhraní `IRenderable`. Díky tomu je komponenta schopna přijímat signály a vykreslit se dle definované šablony.

Implementace komponent jsou umístěny v adresáři pod názvem `components`. Ten je umístěn vždy v modulu, jenž obsažené komponenty využívá.

6.7.1 Grid

Implementuje ji třída `Grid\Grid`. Slouží k výpisu dat získaných z databáze ve formě objektu `DibiDataSource` (viz 4.1.3). Komponenta umožňuje řadit výpis dle sloupce, stránkovat, nastavit počet položek zobrazených na stránce, provádět hromadné akce s označenými položkami (viz obrázek 6.5). Veškerá činnost v případě zapnutého `JavaScript`u probíhá asynchronně pomocí technologie `Ajax`. Plná funkčnost komponenty je zachována i bez zapnutého `JavaScript` v prohlížeči. Komponenta tak ale ztrácí svoji interaktivitu a při každé změně se se musí překreslit celá stránka. Z hlediska vytvořených komponent se jedná o jednu z nejpoužívanějších v celé aplikaci. Následně popíši dva základní prvky, z kterých se komponenta skládá.

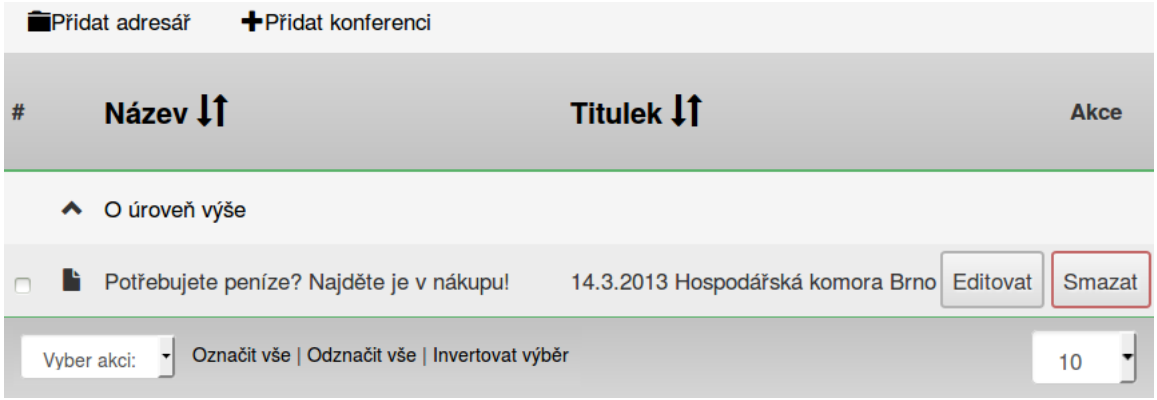
Tlačítka

Implementace v třídě `Grid\Button`. Jsou zobrazena na každém řádku záznamu. Mají určitou funkci nejčastěji editaci záznamu či jeho smazání.

Sloupce

Implementace v třídě `Grid\Column`. Sloupce jsou přidávány do `Gridu`. Těmi mu řekneme, jaké atributy záznamů má zobrazovat. Pořadí vkládání sloupců se projeví na pořadí jejich výpisu. Sloupec nemusí vypisovat pouze hodnotu z databáze, ale můžeme upravit jeho vykreslování. To má na starost metoda `setRenderer($renderer)`. Například při výpisu práv (ve správě uživatelů) u hodnoty přístupu, jenž je boolovského typu, vypisuji její grafické znázornění pomocí piktogramu – „křížek“ a „fajfka“. Takto uživatel na první pohled vidí,

[6] Konfigurační soubor služby `Apache`



Obrázek 6.5: Grafický vzhled komponenty Grid

zda daný záznam zakazuje či povoluje přístup. U sloupců, u kterých je nežádoucí funkcionalita řazení, ji může vypnout pomocí metody `disableSorting()`.

6.7.2 Detail materiálu

Komponenta určena pro zobrazení detailu o vloženém materiálu. Implementace je umístěna v `Greenrs\FrontModule\ConferencesModule\Components\FileDetail`. Komponenta se na stránce zobrazí jako modální okno pomocí signálu `handleShow($fileID)`, kde se jako argument předá identifikační číslo materiálu, pro který je detail určen.

Obrázek 6.6: Grafický vzhled komponenty pro detail materiálu

V hlavičce okna je uveden název materiálu a ikona znázorňující jeho formát. V obsahové části již nalezneme popis, velikost a formát (tentokrát v textové podobě). Spodní část (patička) slouží pro zobrazení ceny a ovládacích tlačítek. Tlačítko úplně vlevo má za úkol pouze zavřít modální okno. Tlačítko „Do košíku“ provede vložení materiálu do košíku. Naopak tlačítko „Zakoupit“ provede okamžité zakoupení, a pokud má uživatel dostatečný kredit, může si materiál okamžitě stáhnout.

Kapitola 7

Testování a zhodnocení výsledků

Test uživatelského rozhraní jsem se rozhodl udělat pro ověření funkčnosti a zhodnocení výsledků. Takto mohu zjistit, zda je aplikace pro nového návštěvníka přehledná a zda je ovládání dostatečně intuitivní. Pro tento účel jsem vytvořil testovací manuál, dle kterého účastník testu vykonával úkoly a odpovídal na otázky.

Testování bylo provedeno jednotlivě na skupině 11 osob v rozmezí věku od 16 do 38, z toho 3 uživatelé přišli do styku s aplikací již během vývoje. Test u části uživatelů probíhal vzdáleně pomocí komunikačního programu *Skype*. Abych mohl sledovat a zaznamenávat si údaje, jak uživatel vykonává jednotlivé úkoly, byla uchazečem sdílena obrazovka. Testování neprobíhalo pouze na počítači, ale i na mobilních zařízeních – pouze u uživatelů, se kterými test probíhal osobně. V případě nejasností s úkoly mne mohli uživatelé kdykoliv během testování požádat o radu.

Úkoly zasahovaly do těchto oblastí:

- **Registrace** – vytvoření účtu pouze s povinnými údaji a následné přihlášení
- **Změna uživatelských údajů** – doplnění fakturačních údajů
- **Zakoupení materiálu**
- **Příprava materiálů** – odevzdání souboru do vytvořeného úkolu
- **Proces vytvoření nového hesla** – simulace zapomenutí hesla

Další částí testování byl dotazník s několika dotazy ohledně aplikace. Dále uživatel mohl sdělit jakékoliv postřehy, jenž mne vedly ke změnám uvedeným v 7.1. Kompletní testovací manuál je v příloze D.

7.1 Vyhodnocení testů – provedené změny

Řada uživatelů zvládla testování téměř bez dotazů. Většina hodnotila rozvržení uživatelského rozhraní a umístění jeho prvků kladně. Čtyři uživatelé dokonce testování prováděli na mobilní zařízením. Takto mohla být otestována responzibilita aplikace, která měla pozitivní ohlasy. Díky testování jsem od uživatelů získal získal zpětnou vazbu, jež obsahovala řadu názorů a podnětů na vylepšení, jenž následně zmíním.

Registrace – uživatelské jméno

Dva uživatelé dali návrh na zrušení nutnosti mít jako uživatelské jméno pouze emailovou adresu. Tento návrh byl pro mne dost zajímavý, jelikož nechci uživatele při registraci nutit do něčeho, na co není zvyklý. Udělal jsem tedy kompromis, kdy si uživatel může jako uživatelské jméno zvolit to, na co je zvyklý – přezdívku či emailovou adresu.

Responzivní design – zvětšení stránky

Uživateli, který prováděl testování na mobilním zařízení, se nelíbila zakázaná funkcionalita zvětšení stránky (tzv. pinch to zoom). Když jsem se ho zeptal, k čemu by by byla nutná v mobilním zobrazení, argumentoval to tím, že někteří uživatelé (jako on) mají větší konečky prstů a někdy mají problém se správně „trefit“ na určitý prvek na stránce. Po delší diskuzi mne přesvědčil, že se opravdu jedná o pádný argument a danou funkcionalitu jsem povolil.

Favicona stránky

Jedná se o ikonu, která je zobrazena v záhlaví webového prohlížeče. Od uživatele padl dotaz, zda by ji nebylo vhodné dodělat. On sám (a jiní lidé) se podle ní orientuje v záložkách, jež má v prohlížeči uložené. Takto prý nemusí číst jejich názvy a požadovanou záložku tak daleko rychleji vyhledá pomocí ikony. S tímto návrhem jsem souhlasil a ikonu dodělal.

7.2 Slovo zadavatele

Vznik aplikace vychází z potřeby sdružení kolem webu `www.znalostninakup.cz` zajistit přípravu i nabídku materiálů pro pořádané akce. Na akcích přednáší pozvaní odborníci, především nákupčí a manažeři firem. Každý přednášející si připravuje materiály ke své prezentaci a před akcí je v určeném termínu zasílá pořadateli. Dosud bylo nutné každého přednášejícího průběžně kontaktovat a získávat koncepty jejich materiálů k revizím tak, aby bylo vše včas připravené. V aplikaci je tato činnost značně ulehčena. Je možné sledovat postup přípravy i samotné materiály.

Po skončení akce se musí zajistit elektronická distribuce materiálů, a to nejen dokumentů typu `PDF` nebo `PPT`, ale i multimediálních zdrojů.

Materiály se poskytují zdarma, ale i za poplatek pro jejich autory. Pro placený obsah je důležitý dvojí přístup k „zakoupení.“ Prvním je rychlé zakoupení, kdy dojde k okamžitému strhnutí částky z kreditu. Druhý přístup je formou košíku, pomocí kterého uživatelů jedním kliknutím zakoupí celý jeho obsah. Dobití kreditu je realizováno on-line platbou platební kartou (požadavek na implementaci služby E-commerce České spořitelny). V budoucnu bude i možné kredit získat prostřednictvím unikátního kódu z voucheru např. předaného na akci přímo jejímu účastníkovi či přednášejícímu.

Tato aplikace má mít možnost dalšího rozvoje formou integrace k dalším webovým aplikacím sdružení Znalostní nákup – rezervační systém, zasílání newsletterů, komunikace redakčním systémem.

Kapitola 8

Možnosti dalšího vývoje

Již při konzultacích se zadavatelem během implementace aplikace padly návrhy, co by se v budoucnu mohlo do aplikace dodělat a rozšířit tak její funkcionalitu. Dále se jsem si vědom, že ne všechny potřebné funkce pro pohodlnou práci s aplikací byly implementovány. Chybí například možnost vyhledávání, stažení vygenerované faktury. V následujících odstavcích bych popsal směr vývoje, kterým se bude aplikace dále ubírat.

Vouchery

Jednalo by se o další způsob, pomocí kterého by se mohl zákazník dobít kredit. Vouchery by byly rozdávány na konferencích či při jiných příležitostech. Tímto by pořadatel upozornil a zároveň přilákal návštěvníky do aplikace, kde by si mohli zakoupit materiály z pořádané konference. Voucher by měl určitou hodnotu a unikátní kód, který by uživatel zadal v sekci pro dobití kreditu. Existovaly by dva typy voucherů. První typ by byl jednorázový, jehož hodnota by mohla být čerpána pouze jednou tzn. u jednoho účtu. Druhý typ by šel použít vícekrát, ale u daného účtu pouze jednou. Tento voucher by našel uplatnění u hromadných akcích, kdy by byl vygenerován pro všechny návštěvníky akce pouze jeden společný.

Prodej publikací

Kromě prodeje nabízených materiálů by přibyl i prodej publikací. Řada přednášejících, jenž na konferencích vystupují, vydali i vlastní publikace. Prodej by vypadal podobně jako u materiálů – úhradou pomocí kreditu. Už by ale přibyl objednávkový formulář sloužící k zadání údajů pro doručení/fakturaci a následný výběr formy doručení.

Upozornění na nové materiály

V sekci uživatelského profilu by přibyla sekce, kde by si uživatel mohl povolit zasílání informací ohledně nově vložených materiálů. Informace by byly zaslány pomocí emailové zprávy. Uživatel by si mohl vybrat, zda chce zasílat informace o všech vložených materiálech či pouze o těch, kterým bude odpovídat seznam zadaných klíčových slov.

Kapitola 9

Závěr

Cílem této práce bylo vytvořit aplikaci pomocí webových technologií sloužící pro správu a řízení přípravy dokumentů. Zadání aplikace vycházelo z potřeb firmy *PRESCOM*, jenž má na starost organizaci konferencí a provozuje webovou aplikaci `www.znalostninakup.cz` sdružující komunitu v oblasti nákupčích, kteří zároveň na konferencích často přednáší. Při návrh jsem kladl důraz na jednoduchost uživatelského rozhraní a responzibilitu, která se v dnešní době stává „nutností“.

Z důvodu zkušeností programování stránek pouze pomocí `PHP`, bez použití objektového programování či vůbec návrhových vzorů (vznikal tzv. špagetový kód), bylo pro mne nejtěžším úkolem zvolit vhodný a dobře zdokumentovaný `PHP` framework. Po prostudování existujících frameworků jsem nakonec vybral `Nette` framework od české komunity. Z dalších technologií jsem zvolil javascriptovou knihovnu `jQuery` a `CSS` framework `Twitter Bootstrap` pro základ, nad kterým byl tvořen grafický vzhled aplikace (frontend a backend).

Vytvořená aplikace splňuje stanovené požadavky. Uživatel má k dispozici intuitivní a přehledné uživatelské rozhraní. Frontend i backend aplikace je plně responzibilní, kdy rozmístění a šířka okna aplikace se automaticky přizpůsobují velkosti displeje použitého zařízení. Lze s ní tedy pohodlně pracovat, jak z běžného počítače, tak i z mobilního zařízení jako chytrý telefon či tablet. Administrátor může pohodlně z jednoho místa řídit přípravu materiálů a zároveň uživatel vidí přehledně zadané úkoly. Správa materiálů dělí související materiály do složek, které jsou dále pomocí hierarchické struktury organizovány, což napomáhá lepší orientaci a přehlednosti. Přihlášený uživatel si následně může materiály zakoupit pomocí úhrady ze svého kreditu. Získání kreditu je zatím řešeno platební bránou a bezhotovostním převodem. Aplikace je nyní veřejně dostupná na webové adrese `www.monitorevent.cz`.

Díky této aplikaci se mi podařilo proniknout do světa `PHP` frameworků, čehož si cením, jelikož veškeré zkušenosti využiji při vývoji dalších aplikací. Pokud srovnám tvorbu webové aplikace s použitím a bez použití frameworku, kterých jsem dělal desítek, mohu na rovinu říci, že mi ulehčil mnoho práce a kód se stal mnohem přehlednější. Dále jsem rád za zkušenosti s propojením aplikace na platební bránu, jelikož se jedná o moderní způsob platby, jež mohu v budoucnu opět využít. Co se týče aplikace a dalšího vývoje, bude hlavně zaměřen na body uvedené v kapitole 8. Jsem přesvědčen, že aplikace je pro potřeby zadavatele použitelná a věřím, že další spolupráce povede k jejímu zlepšování a také rozšiřování.

Literatura

[1] Web Technologies Statistics and Trends [online]. http://w3techs.com/technologies/, 2015 [cit. 2015-03-27].

[2] DB-Engines Ranking [online]. http://db-engines.com/en/ranking/, 2015 [cit. 2015-03-30].

[3] Comparison of web application frameworks: PHP [online]. http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks, 2015 [cit. 2015-04-01].

[4] Daněk, P.: Velký test PHP frameworků [online]. http://www.root.cz/clanky/velky-test-php-frameworku-2008/, 2008 [cit. 2015-04-01].

[5] GLADNEY, H. M.: *Preserving digital information.* New York: Springer, první vydání, 2007, ISBN 35-403-7886-3, Dostupné z: http://fxariwibowo.files.wordpress.com/2009/08/preservingdigital-information.pdf.

[6] INGLE, D. a. B. B. M.: *International journal of advanced research in computer engineering and technology: Analyzing Web Modeling Existing Languages and Approaches to Model Web Application Design [online].* International journal of advanced research in computer engineering and technology, 2012 [cit. 2014-11-01].

[7] Jurmann, M.: The Dos and Don'ts of Website Navigation Usability [online]. http://www.chromaticsites.com/blog/the-dos-and-donts-of-website-navigation-usability, 2007 [cit. 2014-11-19].

[8] KRUG, S.: *Web design: nenuťte uživatele přemýšlet!* Brno: Computer Press, druhé vydání, 2006, ISBN 80-251-1291-8.

[9] Leondes, C. T.: *Intelligent systems.* Boca Raton, FL: CRC Press, 6 vydání, 2003, ISBN 08-493-1121-7.

[10] Meadan, B.: Planned Obsolescence and Your Website [online]. http://cybersteps.org/planned-obsolescence-and-your-website/, rok vydání neznámý [cit. 2014-12-20].

[11] Molhanec, M.: WebML - objektově orientovaná metodika pro tvorbu webových sídel [online]. http://martin.feld.cvut.cz/~molhanec/VaV/files/publik/2003/WebML-CO.pdf, 2003 [cit. 2014-11-01].

[12] Nielsen, J.: F-Shaped Pattern For Reading Web Content [online]. http://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/, 2006 [cit. 2014-11-14].

[13] Skvorc, B.: Best PHP Framework for 2015 [online]. http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/ , 2008 [cit. 2015-04-02].

[14] VRÁNA, J.: Ukládání vícejazyčných záznamů [online]. http://php.vrana.cz/ukladani-vicejazycnych-zaznamu.php , 2009 [cit. 2015-04-10].

[15] WWW stránky: The web modeling language. http://www.webml.org/webml/page37.do?UserCtxParam=0&GroupCtxParam=0&ctx1=EN.

[16] Zelenka, P.: WebML - projektování webových aplikací [online]. http://interval.cz/clanky/webml-projektovani-webovych-aplikaci/, 2003 [cit. 2014-11-01].

[17] ŠPINAR, D.: *Tvoříme přístupné webové stránky: připraveno s ohledem na novelu Zákona č. 365/2000 Sb., o informačních systémech veřejné správy.* Brno: Zoner Press, první vydání, 2004, ISBN 80-868-1511-0.

Příloha A

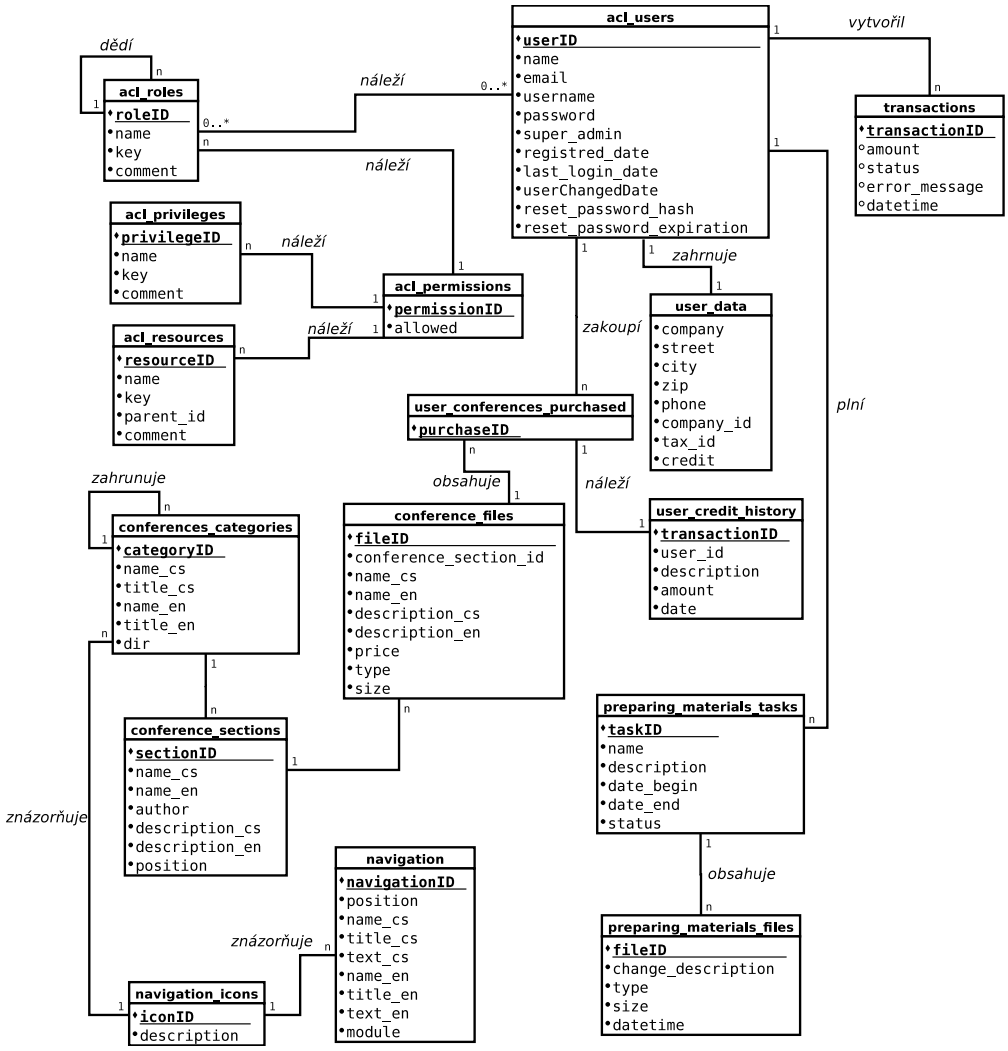
Obsah přiloženého CD

- `/xsvest05_aplikace/` – zdrojové kódy aplikace, návod na instalaci v souboru `instalace.txt`
- `/xsvest05_BP_kod/` – zdrojový kód práce pro sazbu pomocí programu LaTeX
- `/xsvest05_BP_kod/img/` – obrázky, diagramy (včetně editovatelného formátu pro program DIA)
- `/xsvest05_video/` – video na němž je zachycena práce s aplikací (tzv. screencast)
- `/xsvest05_aplikace.zip` – komprimovaný adresář ve formátu ZIP
- `/xsvest05_BP_kod.zip` - komprimovaný adresář ve formátu ZIP
- `/xsvest05_BP.pdf` – bakalářská práce ve formátu PDF
- `/xsvest05_readme.txt` – popis adresářů a souborů

Příloha B

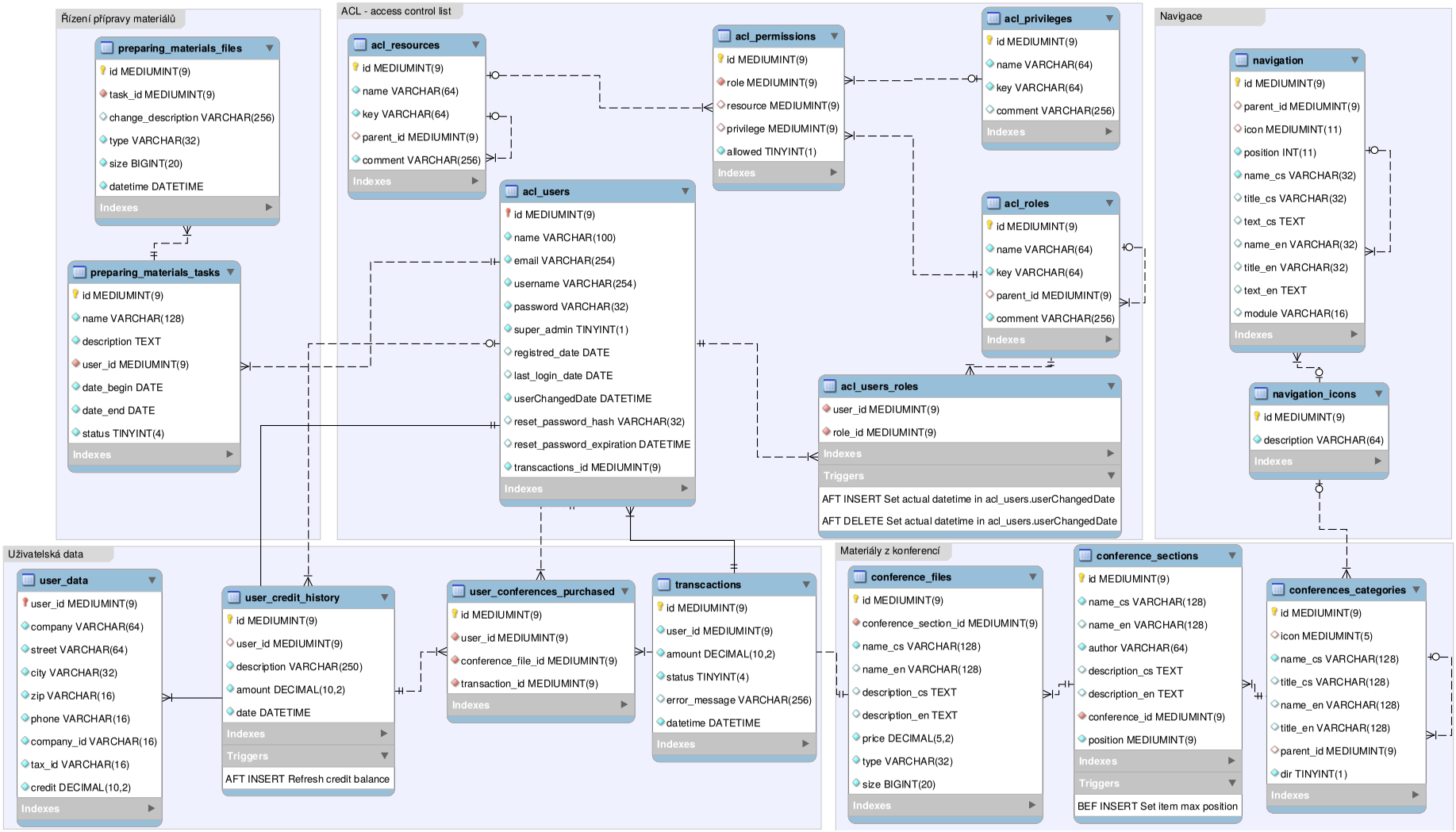
Návrhové diagramy

B.1 ER diagram



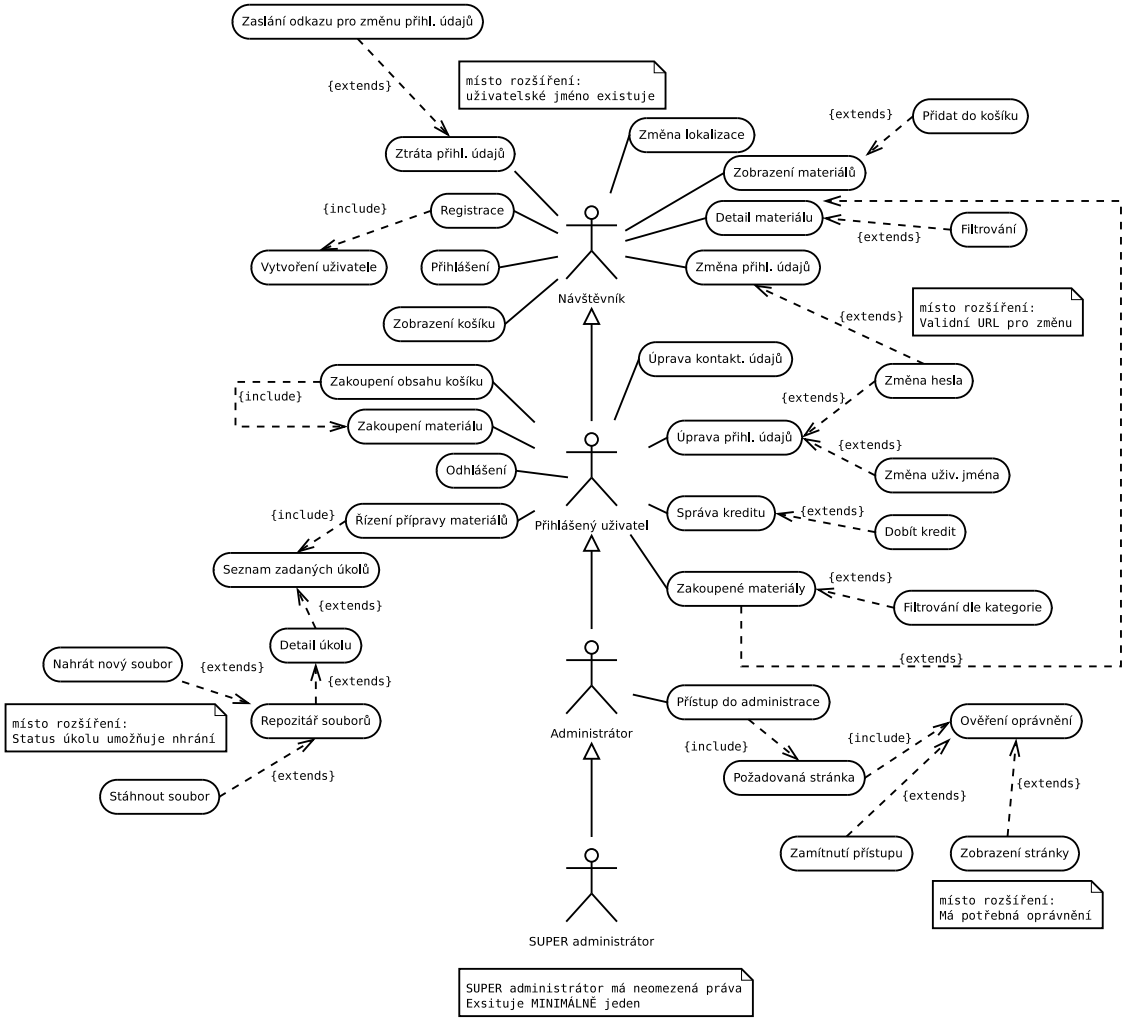
Obrázek B.1: ER diagram aplikace MonitorEvent.cz

B.2 Schéma relační databáze



Obrázek B.2: Kompletní schéma relační databáze aplikace MonitorEvent.cz

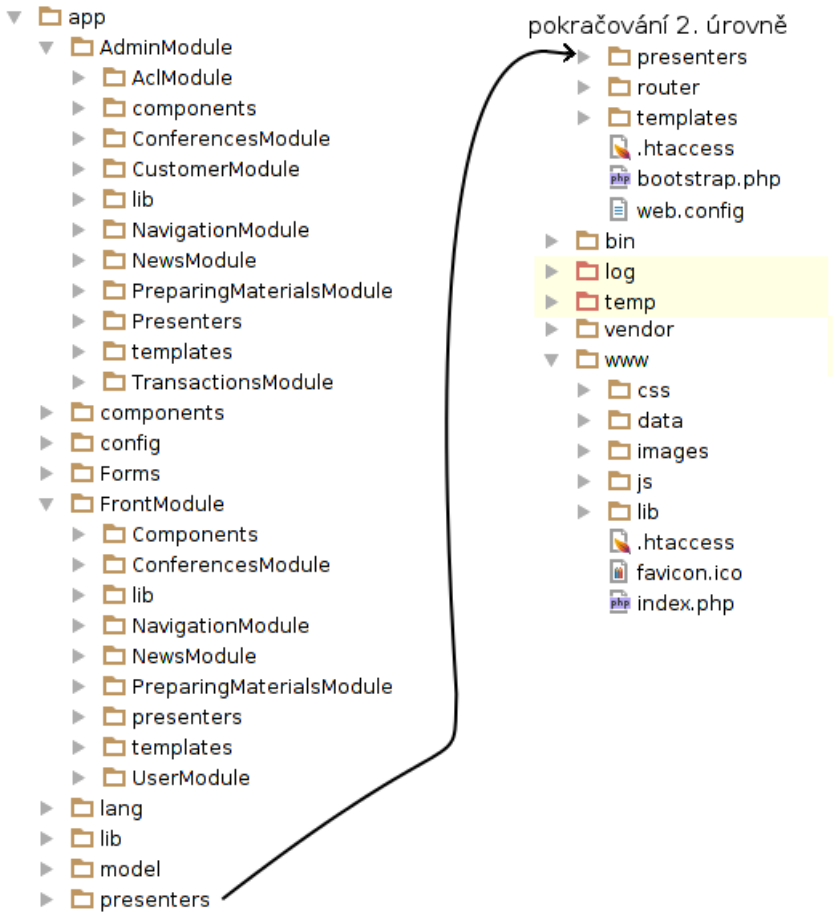
B.3 Diagram případu užití



Obrázek B.3: Diagram případu užití zaměřený na Frontend aplikace MonitorEvent.cz

Příloha C

Struktura aplikace



Obrázek C.1: Adresářová struktura aplikace MonitorEvent.cz

Příloha D

Manuál k testování

D.1 Testování uživatelského rozhraní aplikace

Právě jste byl(a) obeznámen(a) s účelem aplikace `www.monitorevent.cz`. Nyní postupujte dle následujících úkolů, jejichž cílem bude provedení několika operací s aplikací. V případě jakýchkoliv dotazů v průběhu plnění úkolů mne neváhejte oslovit. Veškeré postřehy, co se Vám v aplikaci líbilo, nelíbilo či byste vylepšili, napište prosím do dotazníku (poslední otázka), který následuje za testovacími úkoly.

D.1.1 Úkoly

1. Přejděte na stránku aplikace `www.monitorevent.cz` pomocí Vašeho preferovaného webového prohlížeče
2. **Registrace**
   - Proveďte registraci, kdy v registračním formuláři vyplňte **pouze** povinné údaje. Dbejte prosím na správné vyplnění emailu, jenž bude potřeba v posledním kroku.
   - Přihlaste se pod vytvořeným účtem.
3. **Změna uživatelských údajů**
   - Doplňte ke svému účtu **fakturační údaje**.
4. **Zakoupení materiálu**
   - Proveďte zakoupení materiálu s názvem *Program*, který se nachází v následujícím umístění: *Konference* → *2013* → *Potřebujete peníze? Najděte je v nákupu!*.
5. **Příprava materiálů**
   - Byl Vám otevřen úkol s názvem *Pokusné odevzdání* v sekci *Řízení přípravy*
   - Proveďte **nahrání** souboru k úkolu, který Vám byl poskytnut spolu s tímto testovacím manuálem.
6. **Proces vytvoření nového hesla**
   - Odhlaste se z aplikace

- Představte si, že jste se ocitli situaci kdy si nemůžete vzpomenout na heslo k Vašemu účtu. Využijte aplikací nabízenou funkci, pomocí které Vám bude umožněno nastavení nového hesla.

Testovací část je ukončena. Nyní prosím vyplňte následující dotazník.

D.1.2 Dotazník

Zde prosím odpovězte na několik otázek odpovědí, která je Vám nejbližší. U každé otázky vyberte **pouze jednu** možnost. Na závěr uveďte jakékoliv postřehy, co se Vám na aplikaci líbilo či nelíbilo.

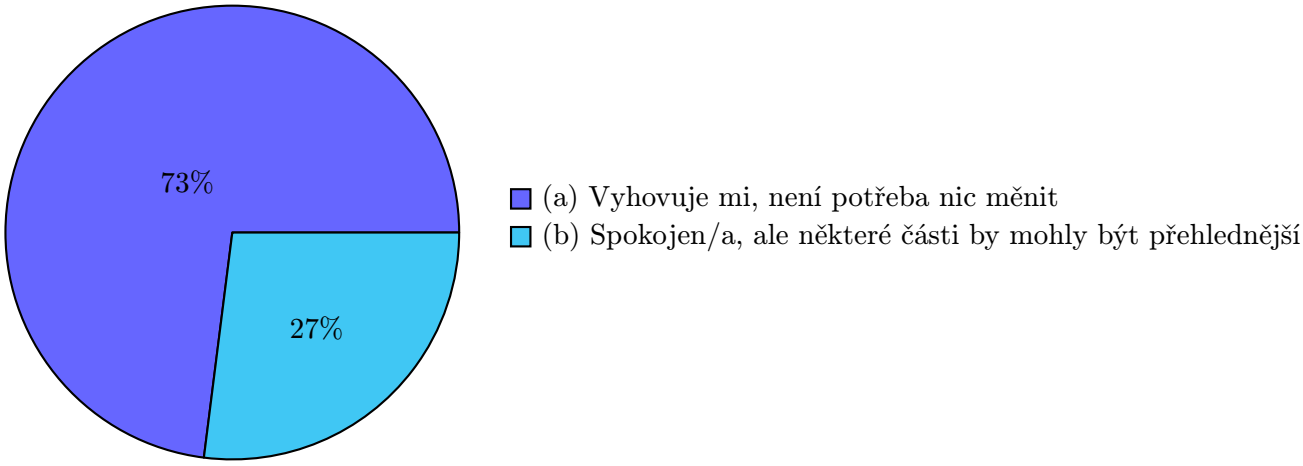
---

1. **Jak jste spokojen(a) s rozvržením uživatelského rozhraní aplikace?**
   (A) Vyhovuje, není potřeba nic měnit
   (B) Spokojen/a, ale některé části by mohly být přehlednější
   (C) Nevyhovuje mi, není dostatečně intuitivní
   (D) Nedovedu posoudit
2. **Jak by jste zhodnotil(a) grafický vzhled aplikace?**
   (A) Jednoduchý, praktický
   (B) Spokojen/a, ale některé prvky jsou až moc strohé
   (C) Líbí se mi kombinace barev
   (D) Nevyhovuje
   (E) Nedovedu posoudit
3. **Z jakého zařízení jste s aplikací pracoval(a)?**
   (A) Počítač (notebook)
   (B) Mobilní zařízení (telefon, tablet)
4. **Jak hodnotíte ovládání aplikace?**
   (A) Jednoduché, vždy jsem našel(la) co bylo potřeba
   (B) Někdy jsem nevěděl(a), jak dále postupovat
   (C) Zpočátku složité, ale pak jsme se zorientoval(a)
   (D) Nedovedu posoudit
5. **Využil(a) byste služeb aplikace aplikace?**
   (A) Ano, líbí se mi její koncept
   (B) Ne, neměl(a) by pro mne využití
   (C) Nevím

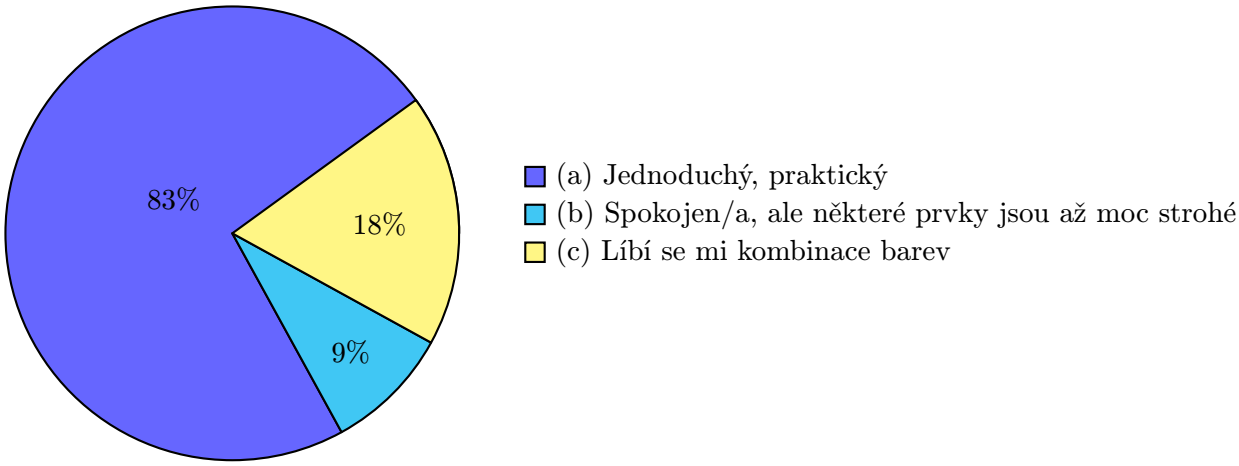
6. **Zde uveďte jakékoliv postřehy, co se Vám na aplikaci líbilo či nelíbilo**

Konec dotazníku. Děkuji za Váš čas, jenž jste věnovali jeho vyplnění.

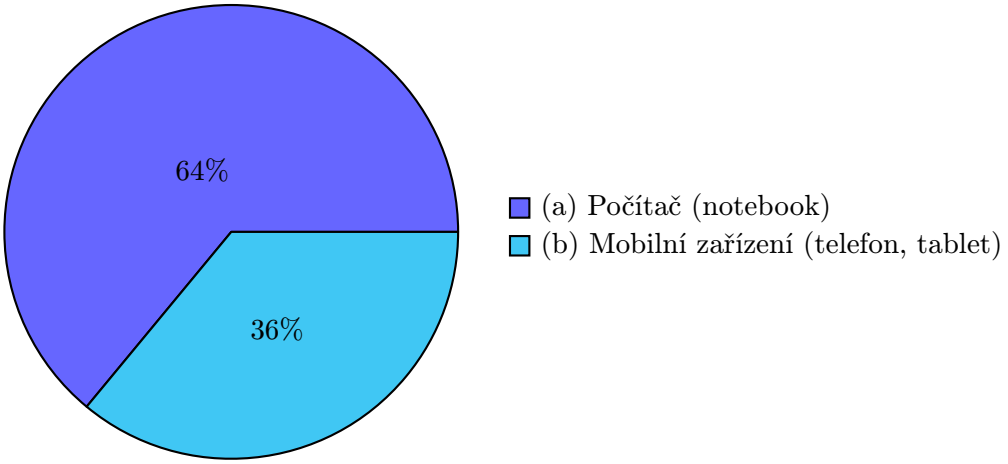
D.1.3 Výsledky dotazníku



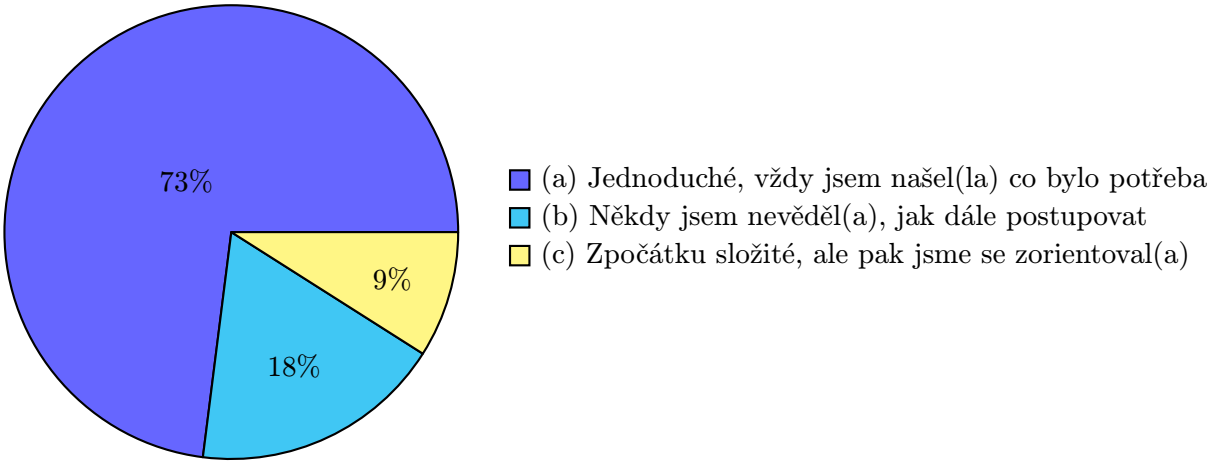
Graf D.1: Jak jste spokojen(a) s rozvržením uživatelského rozhraní aplikace?



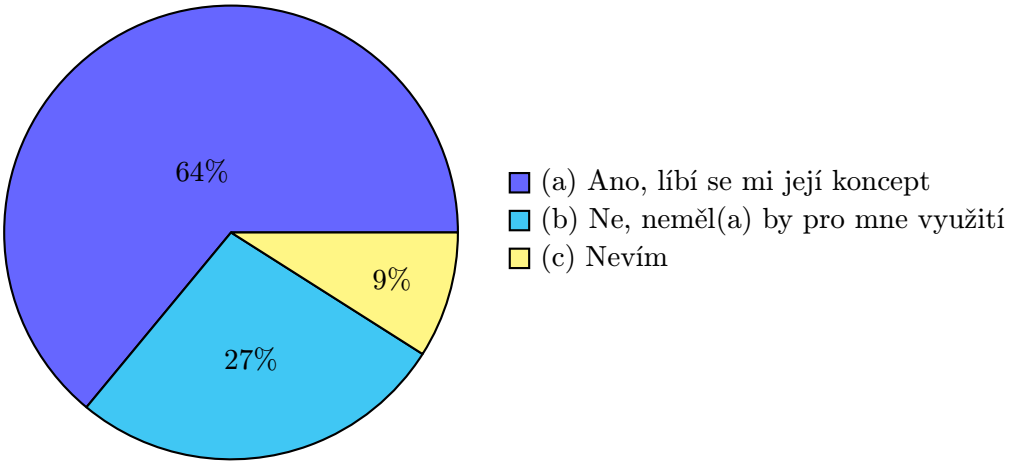
Graf D.2: Jak by jste zhodnotil(a) grafický vzhled aplikace?



Graf D.3: Z jakého zařízení jste s aplikací pracoval(a)?



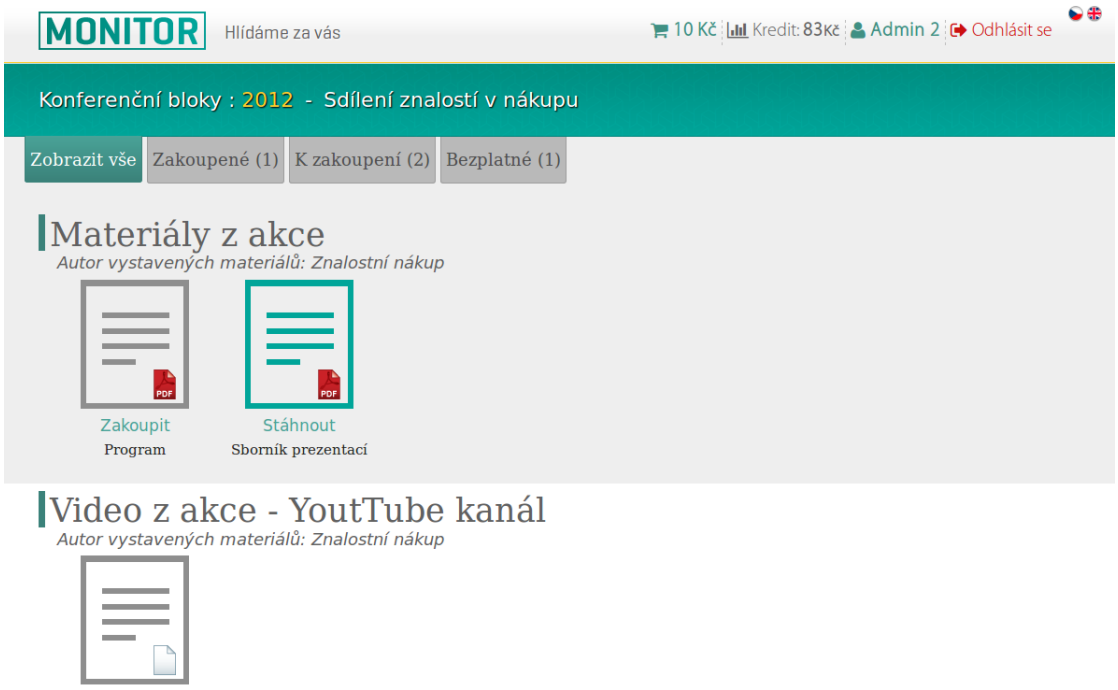
Graf D.4: Jak hodnotíte ovládání aplikace?



Graf D.5: Využil(a) byste služeb aplikace aplikace?

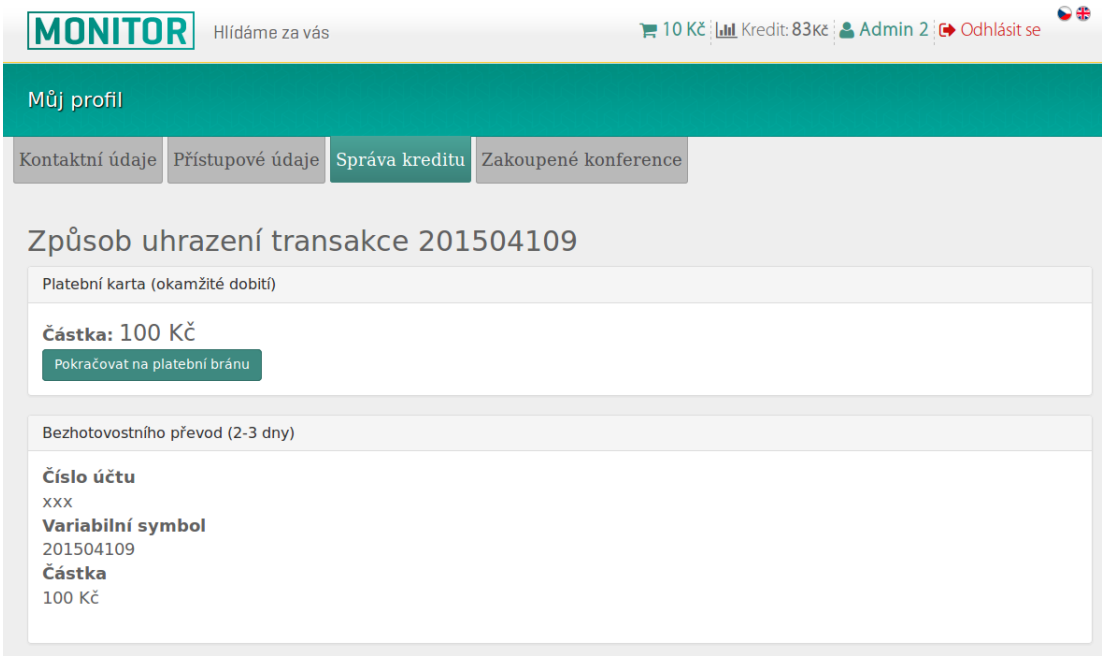
Příloha E

Snímky aplikace



Obrázek E.1: Detail složky s nabízenými materiály

Obrázek E.2: Řízení přípravy materiálu – detail zadaného úkolu



Obrázek E.3: Vytvořená transakce pro dobití kreditu