VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ APLIKACE ZALOŽENÁ NA FRAMEWORKU ASP.NET MVC 3/RAZOR

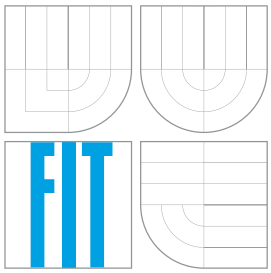
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ KVAPIL

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÁ APLIKACE ZALOŽENÁ NA FRAMEWORKU ASP.NET MVC 3/RAZOR

WEB APPLICATION BASED ON ASP.NET MVC 3/RAZOR FRAMEWORK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ KVAPIL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RUDOLF KAJAN

BRNO 2012

Abstrakt

Cílem bakalářské práce bylo vytvořit webovou aplikaci založenou na návrhovém vzoru ASP.NET MVC 3/Razor s využitím různých moderních technologií. Pro aplikaci byla zvolena problematika komunikace designéra s klientem. V textu je popsán kompletní postup od analýzy existujících řešení přes návrh až po implementaci samotné aplikace. Na závěr je navrženo její možné rozšíření.

Abstract

The main goal of this bachelor's thesis was to create web application based on the ASP.NET MVC 3/Razor framework and several other modern web technologies. The application deals with issues in communication between graphic designer and client. In this text you can find an analysis of existing solutions, design and implementation of the application itself. Possible extensions for this project are sugested at the end of the text.

Klíčová slova

ASP.NET MVC 3/Razor, návrhový vzor MVC, webová aplikace.

Keywords

ASP.NET MVC 3/Razor, MVC framework, web application.

Citace

Jiří Kvapil: Webová aplikace založená na frameworku ASP.NET MVC 3/Razor, bakalářská práce, Brno, FIT VUT v Brně, 2012

Webová aplikace založená na frameworku ASP.NET MVC 3/Razor

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Rudolfa Kajana. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal. U částí práce které nejsou původní jsem vždy uvedl jejich autora.

.....
Jiří Kvapil
16. května 2012

Poděkování

Tímto bych rád poděkoval panu Ing. Rudolfu Kajanovi za poskytnuté rady při vypracování bakalářské práce.

Obsah

Kapitola 1

Úvod

Název bakalářské práce je *Webová aplikace založená na frameworku ASP.NET MVC 3/Razor*, to je velice široké téma, pod kterým si můžeme představit jakoukoliv aplikaci. Jedinou podmínkou je programovací jazyk ASP.NET a použití frameworku MVC 3/Razor. Právě možnost takové volnosti ve vyběru konkrétní aplikace mě vedla k této variantě bakalářské práce.

Kterou aplikaci jsem si tedy vybral? Výstupem této práce bude *Systém pro komunikaci mezi grafickým designérem a zákazníkem*. Proč právě tato problematika? V tom, že tento systém má důvody pro vytvoření, mě utvrdila i vlastní praxe s tím, jak to v komunikaci mezi designérem a zákazníkem chodí. Nápříklad jsem si objednal zhotovení plakátu, poté mi přišla jeho první verze na e-mail. Mé připomínky jsem poslal zpět pomocí elektronické pošty a další upravená verze mi byla opět odeslána na mou e-mailovou adresu. Takto se to opakovalo až do finální verze a k mé spokojenosti. Tento způsob používá velké množství designerů, reklamních agentur, apod., přestože má hned několik nedostatků a ty má za cíl naše aplikace odbourat.

V kapitole 3, **Existující systémy** provedeme analýzu aplikací s podobnou tématikou a systémů použitelných pro komunikaci designéra a zákazníka. Takovými jsou e-mail, prevue.it, dropmark.com. Každému bude v textu věnován vlastní prostor, ve kterém probereme jejich klady a zápory. Výstupem této analýzy pro nás bude sada poučení a zkušeností, které nám umožní navrhnout vlastní aplikaci. Tomu se bude věnovat kapitola 4, **Návrh vlastního systému**. Probereme jednotlivé funkce budoucího systému z pohledu designéra i zákazníka. Jednou z podmínek je vytvoření aplikace za použití moderních technologií tak, aby se implementace nesla v duchu *Neprogramovat něco, co už bylo naprogramováno*, tyto technologie budou popsány v kapitole 5, **Technologie**.

Po provedené analýze a návrhu budou v kapitole 6, **Implementace**, popsány nejdůležitější a zajímavější části programování aplikace. Implementační proces bude seřazen chronologicky a bude obsahovat seznam a popis všech použitých postupů a technologií.

V následujícím textu se velmi často budeme setkávat se slovem *designér*, jakožto hlavním uživatelem našeho budoucího systému. Pod tímto slovem si ale můžeme představit i celé reklamní agentury, umělce, architekty či nezávislé tvůrce - veškeré profese, které mohou svou práci sdílet s okolím pomocí obrázků, zvuku nebo videa v digitálním formátu. V některých případech bude aplikace nazývána pracovním názvem „Workflow“.

Kapitola 2

Hlavní technologie ASP.NET MVC 3/Razor

V této kapitole si řekneme nejprve něco o technologii ASP.NET a frameworku MVC 3/Razor. Vyjmenujeme výhody a důvody, proč jsme vybrali tento programovací jazyk a vývojové prostředí.

2.1 ASP.NET

Tato technologie je součástí .NET Frameworku, který je souborem technologií v softwarových produktech tvořících celou platformu, která je dostupná nejen pro Web ale i Windows a Pocket PC. ASP.NET je odvozen od starší technologie ASP pro vývoj webů. .NET Framework disponuje vlastností CLR (Common Language Runtime), umožňující programátorům programovat v nejrůznějších jazycích. Je tak mnohem jednodužší přejít např. od programování desktop aplikací k aplikacím pro web. Nemusíme se učit nový jazyk, jako např. PHP, ale můžeme napsat nový web z celé řady jazyků ať už Visual Basic.NET, JScript.NET, C#, Managed C++ nebo z řady dalších. Pro tuto bakalářskou práci byl vybrán jazyk C#.

Aplikace v ASP.NET jsou mnohem rychlejší v porovnání se skriptovacími jazyky, kde se stránky sestavují při každém přístupu k nim, neboť jsou předkompilovány do jednoho nebo několika souborů DDL. Více o ASP.NET v knize [2].

2.2 ASP.NET MVC 3/Razor

Je to framework postavený na ASP.NET. Co je ASP.NET jsme si už řekli. MVC je zkratka z Model View Controller. My použijeme již 3. verzi tohoto frameworku (v době psaní tohoto textu už vyšla MVC 4 v beta verzi, která např. přichází s větší podporou pro tvorbu aplikací pro mobilní zařízení). Za každým slovem se skrývá jiná část nějaké aplikace. Přeložit by se to dalo jako datový model aplikace (Model), uživatelské rozhraní (View) a řídicí logiku (Controller).

Datovým modelem můžeme chápat například tabulky v naší databázi, včetně kompletního popisu typů sloupců tabulek a vazeb mezi nimi. Jejich povolené hodnoty, formát apod. od kterého se pak odvíjí jejich automatická validace (pokud to v aplikaci podporujeme). Nemusí se to týkat pouze tabulek a databáze, můžeme si namodelovat jakýkoli objekt. Např. model e-mailové zprávy, která by měla pevně daný formát - odesílatel, příjemce, předmět, text zprávy.

View, uživatelské rozhraní definuje, jak budou data předávána návštěvníkovi stránky. Zahrnuje možnosti interakce s uživatelem. Tato část určuje, jak bude web, každá jeho stránka a její data vizuálně vypadat.

Controller (řídicí logika) reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu.

A co je Razor? Při vytváření projektu v prostředí Visual Web Developer jsme dotázání na výběr *View engine*. To je způsob, jakým budou napsány ty části spadající do View části MVC. Kombinuje bloky kódu HTML s bloky jazyka, který byl vybrán pro programování v ASP.NET, v našem případě C#. Razor není nový jazyk. Popisuje novou syntaxi jak kombinovat v předešlé větě zmíněné jazyky na jedné stránce a to dva odlišné jazyky. Jeden pro zobrazení dat, druhý pro manipulaci s daty. Druhý „View engine“ je ASPX, ten byl typický při vytváření aplikace ve WebForms. My se budeme držet způsobu Razor.

Obrovskou výhodou je možnost použití vývojového prostředí, my použijeme Microsoft Visual Web Developer (na www.asp.net ke stažení zdarma), které nám v mnoha ohledech usnadní práci. Rychle vytváří projekty, má širokou podporu pro MVC framework, našeptává použitelné metody nad objekty, automatické simulování serveru bez nutnosti přístupu na internet. Nebo při vytvoření datového modelu (např. modelu pro databázové tabulky) můžeme několika kliknutími automaticky vytvořit řídící metody „smazat, přidat, editovat“ a dalšími pár kliknutími vytvořit způsob, jak se budou data zobrazovat na konkrétní stránce, a mnoho dalších výhod, které jsou popsány v celém následujícím textu vstahujícím se ke konkrétním kapitolám. Podpora je i ze strany vývojářů, jak pro ASP.NET tak i pro framework MVC v podobě mnoha návodů na oficiálních stránkách.

Kapitola 3

Existující systémy

Jak bylo naznačeno v úvodu, předtím, než se pustíme do návrhu konkrétní aplikace, je nutné provést analýzu existujících řešení daného problému, nebo analýzu aplikací, které se pro danou problematiku používají, přestože pro ní nejsou primárně sestrojeny. V této kapitole si probereme celkem tři systémy. Aplikací která se přímo zabývá touto problematikou, tedy problematikou komunikace designéra a zákazníka je *prevue.it*. Velmi používaná je pak i *e-mailová komunikace*. Velkým přínosem pro nás bude i analýza nové webové aplikace *dropmark.com*. Vybrané systémy budeme rozebírat z pohledu designera i z pohledu zákazníka, probereme jejich výhody i nevýhody. Jako první rozebereme e-mailovou komunikaci.

3.1 Elektronická pošta

Takřka všem známá služba, má jistě svá pozitiva i negativa. My budeme analyzovat elektronickou poštu z pohledu jejího využití pro komunikaci designera a jeho zákazníka. Pokud bude závěrem, že je systém nevhodný, neznamená to, že je celkově elektronická pošta k ničemu. Vypíšeme si tedy výhody a nevýhody, některé budou rozebrány podrobněji.

3.1.1 Výhody e-mailové komunikace

- Obrovská rozšířenost,
- lehké vytvoření,
- zdarma,
- dostupnost,
- třídění,
- možnost vlastního hostingu.

Obrovská rozšířenost je ten hlavní důvod, proč jej většina designerů používá. Můžou si být téměř jistí, že když se zeptají klienta na jejich e-mailovou adresu, bude jejich odpověď kladná. Podle The Radicati Group, Inc. má počet uživatelů elektronické pošty z 1.4 miliard k roku 2009 narůst na 1.9 miliard uživatelů v roce 2013. Denně se pak v roce 2009 poslalo neuvěřitelných 247 miliard zpráv, v roce 2013 to má být 507 miliard [8]. Rozšířenosti elektronické pošty využijeme i v naší aplikaci, o tom však později.

Lehké vytvoření je míněno především z pohledu mnoha poskytovatelů e-mailových služeb, ze kterých je možné vybírat. Samotné vytvoření je pak otázkou několika minut.

Dostupnost. Největší rozmach zažila elektronická pošta při rozvoji internetu. Její dostupnost je pak závislá na dostupnosti internetového připojení. Přístup k naší poště je pak skrz on-line služby jako email.seznam.cz, nebo skrz desktop klienty jako Thunderbird, nebo Outlook.

Třídění. Důležitá vlastnost pro dobrou orientaci v množství došlých zpráv.

3.1.2 Nevýhody e-mailové komunikace

- Velké množství zpráv,
- omezená kapacita zprávy,
- změna vzhledu e-mailového klienta,
- přiřazování komentářů ke konkrétním návrhům (zprávám),
- hledání e-mailu na designéra,
- kapacita celé schránky,
- dostupnost služby.

Velké množství zpráv. Jak bylo řečeno v předchozí kapitole, denně se pošle obrovské množství zpráv. V jedné e-mailové schránce tak můžeme mít za malou chvíli tisíce zpráv. Ať už vyžádaných, tak nevyžádaných, od různých lidí, s přílohou nebo bez. To pak ztěžuje vyhledávání.

Omezená kapacita zprávy. To může být problém ve chvíli, kdy má designér klientovi odeslat například výkres, kde je stěžejní vysoké rozlišení, je tedy i velká velikost souboru.

Změna vzhledu e-mailového klienta. Změny ve vzhledu umožňují především online e-mail služby. Tuto změnu má ve své režii ale každý uživatel sám a designér nemůže ovlivnit, v jakém prostředí jeho klient práci uvidí.

Přiřazování komentářů ke konkrétním návrhům. Ty největší problémy pak může přinášet komentování návrhů ze strany zákazníka. Složité slovní přiřazení k určitému obrázku, nebo části obrázku. Špatné dohledávání komunikace v chaotickém prostředí, kde pokud někdo nepřiloží předešlou komunikaci, musí se složitě dohledávat.

Kapacita celé schránky. Omezená velikost je problém ve chvíli, kdy chce mít designér historii všech svých prací a má jich mnoho.

Dostupnost služby. Jak ukázat práci zákazníkovi, když je můj e-mailový klient mimo službu tak, abych ho neobtěžoval s novými servery pro sdílení dat? a co když má naopak klient plnou elektronickou schránku? To je věc, kterou sám designér nemůže ovlivnit.

3.2 prevue.it

Jednoduchý nástroj pro ukládání a sdílení svých prací z kategorie aplikací, které byly vytvořeny pro designéry, umělce a kohokoli, kdo může sdílet svou práci skrz obrázky. Ze všech systémů, které tu budou analyzovány, je tento nejblíže tomu, který by měl být i výstupem této bakalářské práce. Stejně jako elektronická pošta je poskytován zdarma. Popišme si ale další výhody a nevýhody:

3.2.1 Výhody prevue.it

- Vzhled prostředí se dá měnit,
- propracované komentování prací ze strany zákazníka,
- poznámky u projektů,
- rychlé sdílení se sociální sítí twitter.com,
- portfolio ze všech prací.

Vzhled prostředí se dá měnit. Velká výhoda. Designér má možnost ovlivnit jak přesně zákazník jeho práci uvidí. Každý projekt může mít jiné prostředí. Měnit se dá např. pozadí projektů.

Propracované komentování prací ze strany zákazníka. Každou práci je možné komentovat přímo do konkrétního obrázku, z něj se vybere část a k ní se přiřadí poznámka. Další možností je komentování celého projektu jako celku.

Poznámky u projektů. Velkým plus je i funkce poznámek u každého projektu, kam si může designér přidávat své připomínky. Má tak vše v jednom systému, bez nutnosti psát si poznámky na papír.

Portfolio ze všech prací. Prevue.it nabízí vygenerování portfolia, kde se budou všechny vaše zakázky zobrazovat. Je možnost projekty ochránit heslem, pak se v portfoliu nezobrazují.

3.2.2 Nevýhody prevue.it

- Omezený počet prací,
- omezená velikost nahrávaných fotek,
- pouze pro soubory obrázkových formátů,
- chybí funkce *drag and drop*,
- chybí propracovanější sdílení,
- vybírání konkrétních prací do portfolia.

Omezený počet prací. Po vytvoření účtu dostane každý možnost přidat přesně 30 různých návrhů/obrázků. Navýšení tohoto počtu je umožněno skrze doporučování prevue.it.

Omezená velikost nahrávaných fotek. Velikost fotografie nebo obrázku nesmí přesáhnout 1Mb.

Chybí funkce *drag and drop*. Velkým nedostatkem zvláště s přihlédnutím k uživatelskému komfortu je absence funkce drag and drop souborů přímo z operačního systému/-plochy do aplikace.

Chybí propracovanější sdílení. Rychlé sdílení je pouze do twitteru. Obecně je při sdílení vždy vygenerována URL adresa projektu a uživatel sám distribuuje tuto adresu svým klientům.

Vybírání konkrétních prací do portfolia. Uživatel systému má možnost zakázat zobrazování celého projektu. Chybí zde ale možnost volby zobrazovat pouze ty obrázky, které chceme.

3.3 dropmark.com

Systém fungující zatím jen pod beta verzí (k 27. 12. 2011). Nástroj pro sdílení organizaci souborů různých formátů. Propracovaný systém pro sdílení a spolupráci v reálném čase.

3.3.1 Výhody dropmark.com

- Drag and drop,
- propracované sdílení,
- velké množství podporovaných formátů a dat,
- on-line chat.

Drag and drop. Funkce uchopení souboru a vložení na příslušné místo v aplikaci nahrazuje klasické vybírání souboru z disku vašeho počítače, kdy musíte z výchozí složky najít cestu až k souboru. Drag and drop je uživatelsky přívětivé a urychluje nahrávání souborů. U dropmark.com můžete vkládat soubory nejen z vašeho počítače, ale i z webového prohlížeče.

Propracované sdílení. Ke sdílení souborů s ostatními stačí zadat jejich emailové adresy do příslušného formuláře u daného souboru/souborů. Možnost privátního a veřejného sdílení.

Velké množství podporovaných formátů a dat. Mezi podporovaná data patří: mapy z google maps, videa, fotografie, dokumenty, odkazy.

3.3.2 Nevýhody dropmark.com

- Více než je potřeba,
- komentování konkrétních souborů.

Více než je potřeba. *Méně je někdy více.* Pro naši aplikaci je potřeba jen několika málo formátů. Přílišná složitost může uživatele mást. V našem případě vytváříme aplikaci velice konkrétního uplatnění, naproti tomu dropmark.com, přestože je použitelný pro naší problematiku, byl vytvořen pro komplexní sdílení souborů a celých alb především s přáteli a známými.

Komentování konkrétních souborů. Komentáře může například k fotografiím přidávat jen jejich autor, lépe řečeno ten, kdo soubor přidává. Je tedy jednomu z hlavních požadavků (komentování návrhů ze strany zákazníka) nevyhověno.

Kapitola 4

Návrh vlastního systému

Po provedené analýze existujících aplikací použitelných pro komunikaci designéra se zákazníkem můžeme přistoupit k návrhu našeho vlastního systému. Nejprve si vypíšeme hlavní požadavky na budoucí systém. Některé budou popsány podrobněji. Až se seznámíme se všemi požadavky, podíváme se na nejkomplikovanější blíže a navrhneme jejich efektivní řešení.

4.1 Hlavní požadavky na budoucí aplikaci

V této kapitole si řekneme hlavní potřeby ze strany designéra i ze strany zákazníka. Nejprve však technické parametry.

4.1.1 Technické nároky

- Typy souborů: audio, video, obrázky,
- velká kapacita datového prostoru.

Typy souborů. Systém bude schopný přijímat soubory zvukových, video a obrázových formátů. Po jejich nahrátí aplikace příslušnou formou data uloží a poté bude zobrazovat.

Velká kapacita datového prostoru. Soubory chceme ukládat v co největším rozlišení a nejlepší kvalitě. V případě výkresů velkých formátů nebo videí jsou nároky na kapacitu úložného prostoru veliké. Této problematice se budeme věnovat v kapitole **Řešení největších problémů**.

4.1.2 Nezbytnosti ze strany designéra

- Vytváření, mazání a úprava projektů,
- změna vzhledu prostředí,
- přidávání poznámek ke klientům,
- přidávání a mazání jednotlivých návrhů,
- sdílení prací se zákazníkem,
- možnost vytvoření portfolia,

- všeobecné nastavení.

Změna vzhledu prostředí. Velice důležité je, jak výsledný návrh zákazník uvidí. Plakát jinak vypadá, když bude např. na žlutém pozadí a jinak, když bude na černém pozadí. Je tedy velice důležité, aby měl designér možnost měnit prostředí, ve kterém klient jeho práci uvidí.

Přidávání poznámek ke klientům. Snaha aby všechno bylo v jednom systému tak, aby si uživatel ani poznámky k jednotlivým klientům nemusel psát vedle, ale měl je zapsané v naší aplikaci na příslušném místě.

Sdílení prací se zákazníkem. Aplikace se bude snažit usnadnit uživateli práci, tedy i samotné odeslání hotové práce klientovi bude otázkou jednoho kliknutí, při kterém se zákazníkovi odešle elektronická zpráva s odkazem na práci, případně dalšími údaji.

Všeobecné nastavení. Nastavení týkající se aplikace a účtu, jako například změna loga, omezení notifikací, nastavení svých kontaktů apod.

4.1.3 Požadavky ze strany zákazníka

- Prohlížení projektů,
- přidávání poznámek k návrhům.

Prohlížení projektů. Jedna z hlavních a samozřejmých funkcí. Klient musí mít možnost prohlížet zadané zakázky a to v dostatečné kvalitě.

Přidávání poznámek k návrhům, bude formou vkládání poznámek přímo do obrázku - do určitého místa např. na konceptu plakátu se vloží příslušný komentář vztahující se k dané ploše návrhu. Tyto poznámky jsou přístupné designérovi, který podle nich provede konkrétní změny (samozřejmě, dle svého uvážení).

4.1.4 Společné nároky

Tato kapitola obsahuje požadavky, které mají designér i jeho klient společné.

- Chráněné přihlašování do systému.
- Vzájemná komunikace.

Vzájemná komunikace. Budoucí systém má usnadňovat život oběma stranám. Bude tak co nejvíce služeb a funkcí na jednom místě. Vzájemná komunikace je stěžejní. Aby tedy klient nemusel do své e-mailové schránky, bude v systému možnost posílat soukromé zrávy, přidávat komentáře k návrhům, nebo jak bylo zmíněno přidávat anotace do konkrétních obrázků.

Pro sprovoznění veškeré funkčnosti, která byla navržena však nebude bohužel prostor. Aplikace, která bude výstupem této práce tak bude obsahovat hlavní a nejpodstatnější prvky. Z navrhovaných funkcí však bude chybět jen naprosté minimum. Na možné rozšíření tak bude čekat funkce *změny vzhledu prostředí změna notifikací* a další podporované typy dat, jako *audio a video*.

4.2 Řešení největších problémů

Nyní si popíšeme nejzávaznější problémy a jejich řešení.

4.2.1 Velký objem dat

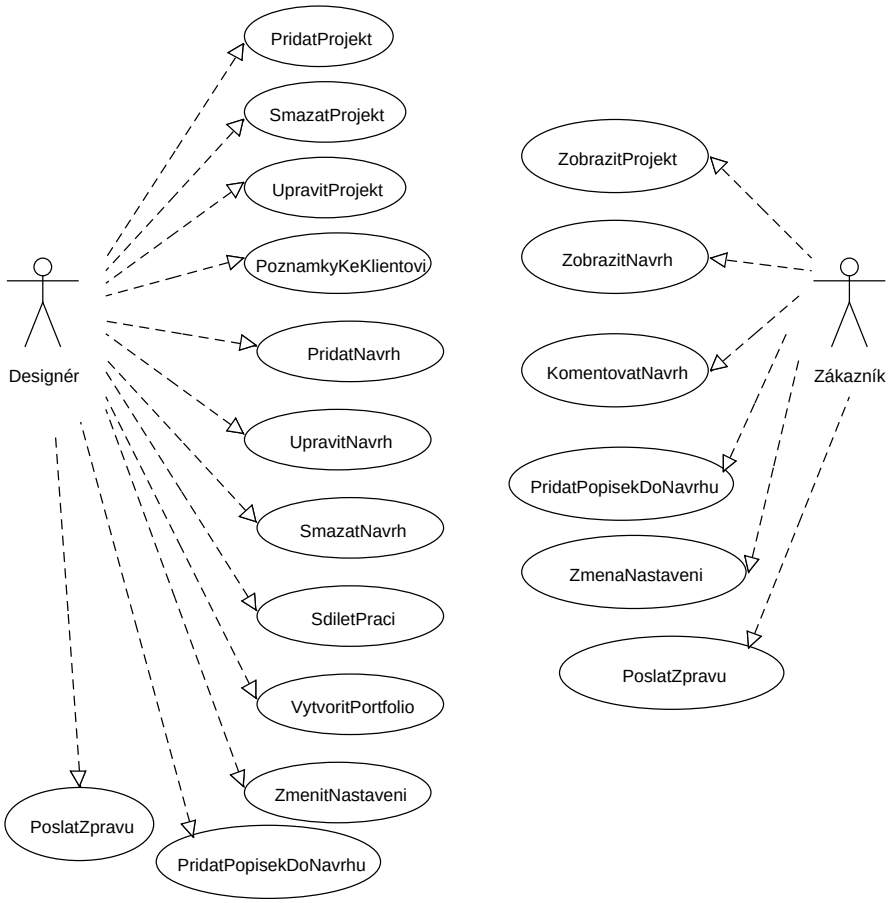
Aplikace použitelné pro komunikaci designéra se zákazníkem, které jsme si probrali v kapitole **Existující systémy**, dávají jen určitý prostor, který nám tvůrci aplikace poskytli. Fungují na způsobu registrace účtu, u kterého není možnost regulovat objemy dat.

Řešení tohoto problému v naší aplikaci bude následující: systém bude v samostatném balíčku, ten se zkopíruje na příslušný server, nainstaluje se a bude závislý na tom prostoru, jaký mu dá konkrétní „vlastník“ staženého balíčku. Tento způsob můžeme vidět například u systému internetového obchodu Prestashop (www.prestashop.com/). Odpadá tím problém sdílení jednoho datového prostoru pro veliké množství případných uživatelů budoucího systému.

Nevýhodou tohoto přístupu je nutnost větší informační gramotnosti, např. zajištění webhostingu, zvládat kopírování skrz FTP nebo vytvoření databáze.

4.3 Use-case diagram

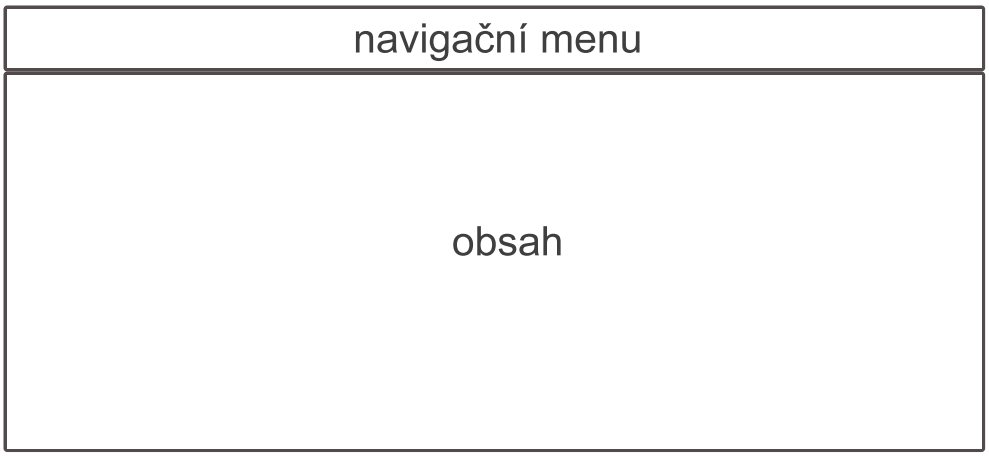
Následuje diagram případů užití našeho systému. Na obr. 4.1 můžeme vidět akce proveditelné z pozice designéra a zákazníka.



Obrázek 4.1: Use case diagram pro náš systém.

4.4 Základní rozvržení prvků na stránce (layout)

Na obr. 4.2 vidíme základní rozvržení v naší aplikaci. Aplikace je tvořena s důrazem na uživatelskou přívětivost a jednoduchost. Ve svém základu se tak skládá pouze ze dvou částí, navigační menu a samotný obsah stránky.



Obrázek 4.2: Základní Layout aplikace.

Navigační menu obsahuje jednotlivé odkazy pro úvodní stránku, projekty, klienty, individuální, nastavení, přidat, zprávy. Jednotlivé stránky si rozebereme. Více o navigačním menu v samostatné podkapitole **Navigační menu**.

V *obsahu* se pak bude zobrazovat potřebné, co se vztahuje k dané stránce.

Layout stránek pro zákazníka bude v podstatě totožný. Budou tam ale chybět takové části, které pochopitelně nesmí být klientovi přístupné, např. *Drag & drop pole*, neboť nechceme, aby vkládal projekty do systému.

4.5 Jednotlivé stránky aplikace a jejich obsah

Tato kapitola popisuje některé stránky aplikace, jejich obsah, kdo má k nim přístup a jaká je jejich funkce. Nakonec je v této kapitole více rozebráno navigační menu, které je přístupné z každé stránky aplikace.

Hlavní stránky, ke kterým má přístup designér.

- Úvodní stránka,
- klienti,
- projekty,
- individuální,
- nastavení,
- tzv. závislé nastavení,
- zprávy.

Hlavní stránky, ke kterým má přístup klient.

- Úvodní stránka,
- projekty,
- nastavení,
- zprávy.

Stránka, ke které má přístup každý návštěvník aplikace.

- Portfolio.

Tyto stránky mají svůj odkaz v hlavním navigačním menu. Existuje spousta dalších stránek, vyjmenovávat je všechny, by však nebylo úplě podstatné. Řekneme si jen o některých. Další podstránky se týkají pouze editací, mazáním, detailem různých objektů. Z hlavních stránek si popíšeme jen ty nejdůležitější. U ostatních jejich název mluví za vše.

Úvodní stránka. Na této stránce, jak v části klienta, tak designéra, se objevují nejnovější události. Hned po přihlášení do systému tak vidí nejnovější komentáře, anotace, zprávy nebo poslední projekty.

Individuální. Pro případ, kdy chce designér uložit obrázek do systému, ale nechce jej přiřazovat k žádnému konkrétnímu projektu, je zřízena stránka „Individuals“. Tam jsou odeslány všechny obrázky, které nebyly přiřazeny k projektu. Odtud je samozřejmě možné je k nějakému projektu zpětně přiřadit.

Nastavení. Zde je samozřejmá změna hesla daného účtu. Na straně designéra je tu však ale několik dalších údajů, které se promítají do portfolia. Jsou to například jméno firmy, logo, kontakty nebo popis.

Závislé nastavení. Toto tzv. „Závislé“ nastavení je položkou hlavní navigační lišty. Je aktivní jen v určitých situacích, je závislé na určitých stránkách. V základu je vypnuté. V případě, kdy se ocitneme například na detailu obrázku, projektu nebo klienta, zaktivuje se a poskytuje nabídku akce k dané stránce, např. smazat projekt, upravit projekt.

Zprávy. Designér může poslat zprávu klientovi, klient designérovi. Jsou to vnitřní zprávy, ne e-mailové zprávy. Evidují se na účtu uživatelů systému, mají možnost vidět přečtené, nepřečtené, odeslané.

Detail projektu. Zde jsou k vidění jednotlivé obrázky přiřazené k danému projektu. Celá obsahová část detailu projektu je akivní jako drop zóna. Pokud tedy chceme přidat další obrázek k otevřenému projektu, stačí přetáhnout soubor obrázku do místa projektu v našem webovém prohlížeči. Odtud je možné každý z obrázků smazat, upravit, poslat do portfolia, přejít na jeho detail.

Detail obrázku. Když přejdeme na stránku s detailem obrázku, máme opět možnost jej mazat a upravovat. Dále je tu lokální statické menu s tlačítky pro přidání anotace do obrázku (nebo její mazání) a přidání komentáře. Komentovat a přidávat anotace do obrázku může designér i klient.

Portfolio. Nachází se v hlavní kořenové stránce domény (na hlavní index stránce). Přístup k ní má každý. Najdeme zde informace o firmě (viz. Nastavení), nebo práce (obrázky), které byly do portfolia poslány.

4.5.1 Navigační menu

Vyjmenovali jsme si různé stránky, které může uživatel systému navštívit. Jejich odkazy jsou v hlavní navigační liště, která je přístupná z každého místa aplikace (samozřejmě přístupné jen přihlášenému uživateli), kromě portfolia. Navigační lišta obsahuje jednotlivé odkazy, ty případně rozbalují další podmenu. Jednou z nejdůležitějších položek je tzv. „Plus“, které má funkci Drag & Drop. Kromě toho že po přesunutí souboru do této zóny bude soubor možné nahrát, je po najetí na tuto zónu aktivní i jako klasická položka menu, která rozbalí nabídku na přidání nového klienta, projektu či obrázku.

Navigační menu bude nehybatelné (pomocí CSS nastylované do statické polohy). To pomůže v případě, kdyby byla stránka příliž dlouhá tak, aby měl uživatel systému vždy odkazy po ruce a nemusel se vracet na vrchní část stránky. Podobné navigační lišty můžeme vidět na www.facebook.com, nebo www.dropmark.com.

Kapitola 5

Technologie

V této kapitole si popíšeme technologie použité při implementování navržené aplikace. S každou technologií se nejprve seznámíme, pak vystvětlíme proč ji chceme použít a objektivně zmíníme její výhody a nevýhody.

5.1 jQuery

Nejprve si řekneme co to vlastně jQuery je, v další části budou popsány důvody proč chceme tuto technologii v našem projektu použít a jaká je podpora ze strany vývojového prostředí pro MVC projekty.

jQuery je open source knihovna jazyka Javascript. Vyznačuje se svojí jednoduchostí oproti samotnému jazyku Javascript, umožňuje používání výhod tohoto jazyka bez jeho větších znalostí. Např. když chceme nějaký prvek na stránce dinamicky skrývat nebo odkrývat a to ještě hezky postupně a ne skokově, museli bychom si napsat v Javascriptu funkci. S jQuery nám stačí přiložit potřebné zdrojové kódy k dokumentu a začít používat obrovské množství funkcí, jako třeba zmíněná skrýt/odkrýt bez jakéhokoli programování. Velkou výhodou je kvalitní dokumentace. Mnoho nezávislých vývojářů vytváří další funkce/programy využívajících jQuery. Pak stačí dát dohromady jejich zdrojové kódy s jQuery knihovnou a může se používat. Další výhodu nese tato technologie v podobě AJAX (Asynchronous JavaScript and XML)- další technologií, která umožňuje měnit obsah webu bez znovuobnovení stránky ale se zapojením serveru. Více o jQuery v kapitole 6.5.

Co se týče podmínek, ve kterých je aplikace pro designéry vyvíjena, hned při vytvoření projektu s využitím frameworku MVC se vytvoří i složka *Scripts* obsahující nejzákladnější jQuery knihovnu. Pokud budeme chtít specielnější funkce jako je například aplikace Colorbox pro zobrazování a procházení fotografií, musíme zdrojové kódy této aplikace přidat nejlépe do složky Scripts (může být i v jiné složce, je však vhodné dávat soubory stejného typu do stejné složky) a poté použít v našem dokumentu.

V naší budoucí aplikaci použijeme tuto knihovnu a aplikace postavené na této technologii pro:

- vkládání poznámek přímo do fotografie, obrázku,
- zobrazování fotek a obrázků,
- lupa, přibližování detailů na fotografii nebo obrázku,
- validace vstupů formulářů,

- pro veškeré akce, které nepotřebují proběhnou na serveru,
- pro komunikaci se serverem bez obnovení stránky (AJAX),
- tzv. progressbar, neboli ukazatel průběhu vykonávání nějaké akce (např. ukládání dat).

Javascript jazyk se vykonává přímo v prohlížeči, není vázán na žádný vzdálený server, ten tedy nezatěžuje.

5.1.1 Vypnutý Javascript

Nevýhodou, takřka rizikem je, že si jej může každý uživatel ve svém prohlížeči vypnout-pak tedy není Javascript a ani jQuery použitelné. S takovými uživateli ale pro naší aplikaci skrátka není počítáno. Existuje mnoho webových aplikací, které pro své bezchybné použití musí mít Javascript zapnutý. Například i český server www.aukro.cz nenahraje fotografie při vytváření nového zboží, bez Javascriptu, nebo služba Google Analytics je postavena na Javascriptu. Nehledě na to, že ti uživatelé, kteří mají Javascript vypnutý se sami ošizují o jeho výhody a stránky jím vylepšeným. Z článku [9] na developer.yahoo.com si můžeme udělat lepší představu o počtu uživatelů bez Javascriptu. Tento počet není absolutní (odhadovaná chyba 0.2%, pro Brazílii 0.04%), v průměru je takových uživatelů do 2%. V USA 2.06% naproti tomu v Brazílii jenom 0.26%.

5.2 LINQ

Jazyk přinášející nový způsob manipulace s daty, usnadňuje třídění, jejich propojování i vyhledávání v nich. LINQ má několik variant podle dat, se kterými má jazyk manipulovat. Nás bude zajímat především LINQ to SQL, z názvu je patrné, že slouží na dotazování nad SQL databází.

Velikou výhodou je automatická podpora tohoto jazyka ze strany frameworku MVC 3. Hned po vytvoření projektu máme LINQ k dispozici. Do jednotlivých dokumentů se pro použití LINQ přiloží příslušný balíček.

5.3 HTML5

Existuje řada důvodů, proč použít nejnovější verzi značkovacího jazyka HTML, ty si uvedeme.

Informační technologie jdou neúprostně kupředu, není tedy rozumné zůstávat u starých a zastaralých technologií, jen to by mohlo být dostačujícím důvodem, popíšeme si to ale více. HTML5 je vyvíjen s důrazem na odbourání nedostatků předchozí verze 4.01. Hlavní má být zjednodušení jazyka. Poskytuje prostředky pro vyznačení částí webu jako je hlavička, menu nebo patička, narozdíl od možností univerzálního tagu DIV, více tagů tedy ponese sémantický obsah. To bude šikovou výhodou pro webové vyhledávače. Další novou funkcí, pro nás důležitou, bude lepší podpora pro multimediální obsah. HTML5 garantuje i zpětnou kompatibilitu.

Opomenout nemůžeme vývojové prostředí Microsoft Visual Web Developer 2010 express, které umožňuje při vytváření projektu ve frameworku MVC 3 jednoduchým zaškrtnutím políčka povolit programování šablon v jazyce HTML5. Při psaní kódu v rovině

View je použito HTML5. MS Visual Web Developer má zabudovanou funkci automatické validace, při které nám v případě chyby text označuje a popisuje danou chybu.

5.4 oAuth.net

Je otevřený protokol pro bezpečnou autentizaci a autorizaci pomocí třetí strany. Odbourává problém nechuti uživatelů předávat další aplikaci svá hesla. OAuth umožňuje aplikacím přihlašovat se pomocí jiné existující služby/aplikace, například pomocí OpenID, Google, Yahoo nebo Twitter účtu. Tam možnosti tohoto protokolu nekončí. Skrze OAuth je možné danou službu, u které jsme provedli autentizaci, i využívat.

Vedle klasického vytvoření účtu s heslem a jménem bude v naší aplikaci možnost využití OAuth pro přihlašování tak, aby uživatelé nemuseli prozrazovat svá hesla, nebo vymýšlet pro náš systém nové heslo, které by si museli zapamatovat.

OAuth má podporu pro většinu jazyků jako je PHP, C#, JAVA nebo pro nás podstatnou technologii ASP.NET.

Toto bylo jen malé představení technologií, které pro aplikaci použijeme. Více o nich a jejich použití bude v kapitole 6, **Implementace**.

Kapitola 6

Implementace

Provedli jsme analýzu existujících řešení, navrhli jsme novou aplikaci, vybrali potřebné technologie a je na řadě samotná implementace. V této kapitole se čtenář dozví, jak bylo potřeba postupovat při snaze vytvořit navrženou webovou aplikaci. Dočte se jak byly jednotlivé problémy a potřebná funkčnost vyřešena. Tato kapitola také obsahuje ukázky zajímavějších úseků kódu.

6.1 První krok, vytvoření šablony aplikace

Jak bylo uvedeno aplikace je vyvíjena v prostředí MS Visual Web Developer Express 2010, která má silnou podporu MVC 3/Razor. Prvním krokem je vytvoření šablony/nového projektu v tomto prostředí. Začneme volbou **File** poté **New Project**, zde se vybere možnost programovacího jazyka (C#) a šablony (MVC 3), napíše se jméno projektu. V dalším kroku se vybere, jestli chceme aplikaci postavit na prázdné šabloně, nebo na předpřipravené (v našem případě byla vybrána možnost prázdné šablony). V tomto kroku je také na výběr, jestli chceme pro naše „View“ stránky používat Razor nebo ASPX. Další možností, kterou jsme zvolili, je HTML5 semantické značkování.

Pro naučení základů s MVC 3/Razor byly jako učební pomůcky použity oficiální návody. Např. návod MVC Music Store [5].

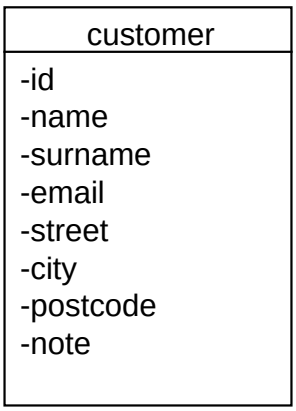
6.2 Model, View, Controller v praxi

V této kapitole si popíšeme postup, který byl opakovaně použit pro různé datové modely. Datovým modelem myslíme například model po zákazníka, projekt či obrázek. Tento postup si ukážeme na příkladu se zákazníkem. Nejprve vytvoříme onen datový model, poté příslušnou řídící logiku (controller) a nakonec zobrazení (view).

6.2.1 Model

Nejprve musíme navrhnout model pro zákazníka. Musíme si uvědomit, jaké informace chceme o zákazníkovi uchovávat. Datový model si popíšeme následujícím diagramem 6.1.

Jednotlivé modely jsou v souborové struktuře MVC ve složce Models. Tam vytvoříme novou třídu s názvem Customer. Obsah této třídy vidíme na ukázce zdrojového kódu 6.1. Můžeme zde vidět množství klíčových slov v hranatých závorkách popisujících data. Popsat data tímto způsobem nám umožňuje balíček *System.ComponentModel.DataAnnotations*,



Obrázek 6.1: Diagram modelu zákazníka.

z tohoto popisu pak vychází i automatická tvorba tabulky databáze, nebo automatická validace vstupů pomocí javascriptových knihoven. Jednotlivým datům tak můžeme striktně přiřadit jaký mají mít datový typ, formát, délku, jestli jsou povinné, případně jim přiřadit chybovou hlášku apod.

Za velkou zmínku stojí řádek *public virtual ICollection< Project>Projects*. Takto vytvořená proměnná asociovaná s modelem zákazníka pak v našem případě obsahuje sbírku veškerých projektů, které patří k danému zákazníkovi. Tohoto docílíme, pokud máme příslušný cizí klíč zákazníka přiřazený u modelu projektu. Vše je automatcké. Nemusíme provádět žádné další „join“ dotazy do databáze.

```
public class Customer
{
    [Key]
    [ScaffoldColumn(false)]
    public int Id { get; set; }
    [Required(ErrorMessage = "User Name is required")]
    public string Name { get; set; }
    [Required(ErrorMessage = "User Surname is required")]
    public string Surname { get; set; }
    [StringLength(24)]
    public string Phone { get; set; }
    [Required(ErrorMessage = "Email address is required")]
    [RegularExpression(@"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}")]
    [DataType(DataType.EmailAddress)]
    public string Email { get; set; }
    public string Street { get; set; }
    [StringLength(40)]
    public string City { get; set; }
    [StringLength(10)]
    public string Postcode { get; set; }
    public string Note { get; set; }
    public virtual ICollection<Project> Projects { get; set; }
}
```

Zdrojový kód 6.1: Obsah model třídy zákazníka.

6.2.2 Controller

Sice je v názvu MVC Controller až poslední, v hierarchii programování v našem případě následuje hned po vytvoření nějaké model třídy. Veškeré třídy řídící logiky (controller) jsou ve složce Cotrollers. Vytvoříme tam třídu nazvanou CustomerController. Každá třída může být vytvořena s různým nastavením. Buďto jako prázdná, nebo s předpřipravenými metodami pro akce Index, Create, Edit, Delete, nebo jako třetí možnost s kompletně funkčními metodami pro vyjmenované akce s použitím Entity Frameworku (EF). Při třetí možnosti musíme určit k jakému modelu daný Controler patří a jaký se k němu vztahuje Data Context. V našem případě byla v složce Models vytvořena třída SystemForDesignersEntities.cs, která reprezentuje náš datový kontext. V ní jsou obsaženy veškeré datové modely, které vytvoříme. Pokud vytvoříme Controller pomocí EF, vytvoří se ve složce Views i příslušná podsložka s názvem řídící logiky obsahující jednotlivé soubory, kde se už píše konkrétní kód, který uživatelé aplikace vidí. Pomocí EF je tak možné co nejautomatizovaněji vytvořit kompletně fungující aplikaci.

V našem případě bylo využíváno spíše prázdné šablony pro jednotlivé řídicí třídy tak, abychom si napsali jen ty metody, které opravdu potřebujeme. Pokud to vztáhneme na náš příklad se zákazníkem, vytvořili jsme 8 různých metod pro různé akce (např. Index, Edit, Delete, Create, AjaxDeleteClient). Každá z těchto metod vykonává určitou činnost (o jakou jde je patrné z jejich názvů) nad objektem nějakého modelu a poté vrací například konkrétní View, nebo v případě AjaxDeleteClient Json objekt. O některých zajímavých metodách si řekneme v dalších kapitolách.

6.2.3 View

View je poslední komponeta, určující jaká data a jak budou prezentována uživatelům aplikace. Jednotlivé tyto pohledy (View) jsou ve složce Views, každý má pak svou podsložku pojmenovanou stejně jako některá třída z řídicí logiky. Speciální je pak složka Shared, kde je vhodné vkládat například takové soubory, které určují Layout. View soubory mají při použití Razor koncovku *.cshtml*. V těchto souborech pak můžeme psát v HTML, CSS, C#. Pro oddělení HTML a C# kódu se používá čistější Razor syntaxe (oproti další dostupné možnosti ASPX).

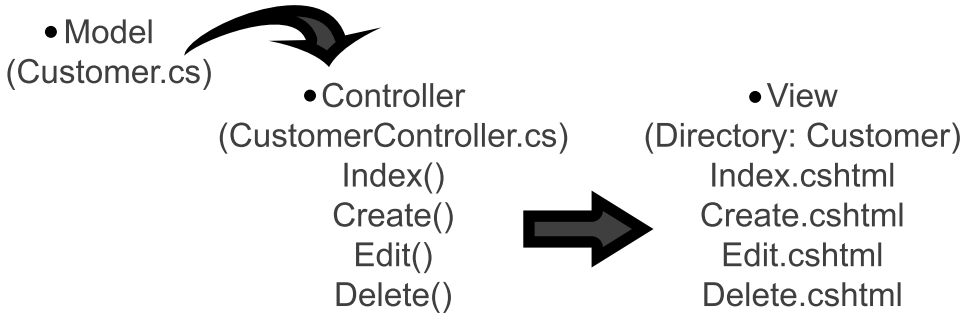
Na brázku 6.2 můžeme názorněji na příkladu zákazníka vidět vzájemnou návaznost mezi jednotlivými vrstvami MVC. Řídicí třída CustomerController přijímá ve svých metodách (Index, Create, Edit,...) obejkty modelu Customer. Tyto metody, v MVC nazývány jako akce, pak s daným modelem provedou příslušnou akci a vrátí určitý výsledek na View se stejným jménem, jako má tato akce. Jednotlivé pohledy jsou pak s příponou *.cshtml* ve složce, která je pojmenovaná stejně jako řídicí třída, tedy Customer.

A jak se na View dostanou potřebná data?

Podobná situace jako se zákazníkem byla i s dalšími entitami, které v budoucí aplikaci budou hrát roli. V další kapitole si je vyjmenujeme a popíšeme vazby mezi nimi.

6.3 Datové entity

Při návrhu systému jsme se setkali s potřebou vytvoření několika tabulek uchovávajících potřebná data. Zjistili jsme, že je tedy potřeba navrhnout strukturu několika entit. V následujícím seznamu si je vypíšeme, v závorce je pak anglicky název příslušného modelu. Pro



Obrázek 6.2: Vzájemná návaznost jednotlivých entit MVC.

samotné programování, názvy funkcí, metod a psaní poznámek byl zvolen anglický jazyk. Na obr. 6.3 pak vidíme ER diagram s názvy jednotlivých sloupců tabulek a jejich propojení.

- Zákazníci (Customer),
- projekty (Project),
- obrázky (Image),
- nastavení (Setting),
- poznámky (Note),
- komentáře (Comment),
- zprávy (Message).

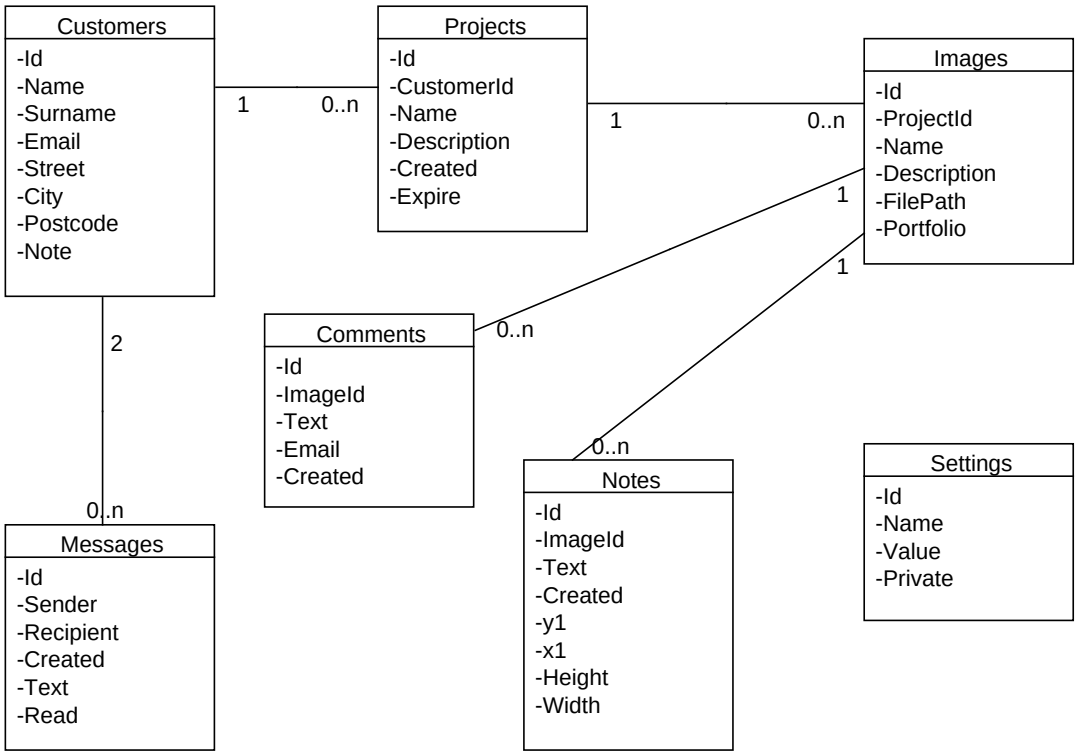
Z vytvořených modelů se pro každou entitu automaticky vytvořili příslušné tabulky. Často se při programování setkáme s dilematem, zda tabulky v databázi pojmenovávat množným, nebo jednotným číslem. Názory jsou různé. V ASP.NET MVC 3 je standardně automaticky každá tabulka převedena do množného čísla (přidáním „s“ na konec jména). Oba přístupy nemají na funkčnost vliv. Pokud však někdo preferuje jednotné číslo, může standardní chování jedním příkazem zrušit. O této problematice a tvorbě datového modelu pomocí Entity Frameworku v online návodu [4].

6.4 Tři roviny aplikace

Aplikace musí řešit celkem tři různé roviny z hlediska přístupových práv. V tabulce 6.1 jsou tyto 3 roviny vypsány. V prvním sloupci tabulky jsou definováni uživatelé, v druhém je název role a v posledním je ukázka URL kde je tato role pro přihlášení nutná.

První rovina je viditelná všem návštěvníkům webové aplikace, jedná se standardně o domácí stránku nějaké webové domény. Zde je umístěno portfolium, kam je možné odesílat jednotlivé obrázky.

Druhá rovina je pro uživatele s přístupovými právy administrátora, tedy v našem případě nějakého deisgnéra. Veškeré řídící třídy, modely a soubory s uživatelským rozhraním spadajícími pod administrátora jsou umístěny v hlavním adresáři aplikace.



Obrázek 6.3: ER diagram aplikace.

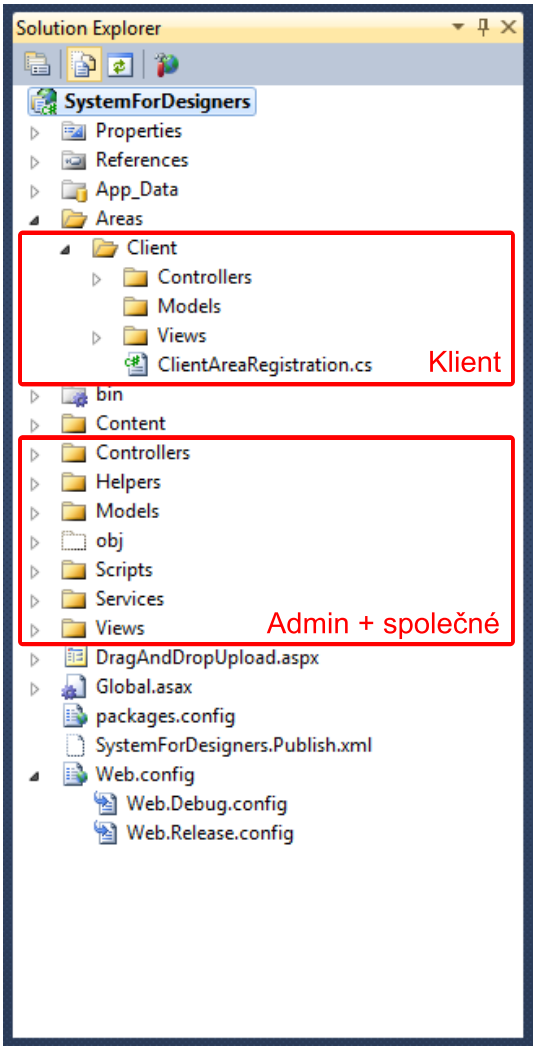
Přístup	role	ukázka url
Pro všechny		www.web.cz
Designér	Administrator	www.web.cz/Admin
Klient	Client	www.web.cz/Client

Tabulka 6.1: Tabulka tří rovin aplikace z hlediska přístupových práv.

Třetí rovina je pro klienty, ta obsahuje stejné pojemnované řídící třídy, nebo pohledy (Views) jako administrátor, toto činilo potíže. Pokud bychom chtěli aby se jak o klienta tak o administrátora aplikace starali stejné řídící třídy (Controller) museli by se podl určitých atributů měnit především různá přesměrování. Takový přístup by však velice komplikoval kód. MVC 3 framework však pro tuto situaci nabízí řešení v podobě *Areas* rozšíření. Tento způsob nám umožňuje vytvořit libovolné množství takových „oblastní“, které pak mohou mít vlastní řídící třídy, modely či uživatelské rozhraní. Pro větší názornost slouží obrázek 6.4, kde vidíme rozmístění jednotlivých zdrojových kódů do správných adresářů.

6.5 jQuery

Tuto technologii už jsme představili v kapitole o technologiích. V této kapitole si ale řekneme jak, k čemu a proč jsme ji použili. Výhody jQuery můžeme používat při přidání její knihovny do naší aplikace, ještě větší funkčnost pak dostaneme, když ji „spojíme“ s dalšími pluginy



Obrázek 6.4: Rozdělení zdrojových kódů podle 3 různých rovin.

(moduly), ať už oficiálními, nebo od nezávislých tvůrců. Jednotlivé moduly budou zmíněny v pořadí, v jakém byly nasazovány při implementaci. Z oficiálních modulů vydaných tvůrci jQuery bylo použito rozšíření Datepicker pro interaktivní výběr data (více o tomto a dalších rozšířeních na www.jqueryui.com).

6.5.1 Modul Colorbox

Tento modul je nejčastěji používán jako prohlížeč fotografií, kdy nemusíte každou otevírat samostatně v novém okně, místo toho vyskakuje do popředí box s obrázkem, uživatel však stále zůstává na stejné stránce, stránka se nemusí znovu načítat. Takových prohlížečů fotografií je velké množství (např. Lightbox, FancyBox, GreyBox), Colorbox má ale další výhodu, umí zobrazovat i tzv. inline úseky kódu, poradí si s AJAX technologií i iframe, kromě toho nabízí ve svém základy hned několik různých vzhledů. Pro naší aplikaci byl použit především z důvodu schopnosti zobrazovat inline kód, tedy takový kód, který se někde nachází na stránce. Konkrétní použití nalezl tento modul při potřebě zobrazovat

- chybové hlášky,
- dotazovací hlášky,
- oznamovací hlášky,
- zobrazování stavu akce (tzv. progressbar)
- zobrazování oken pomocí kterých se přidával obsah (zákazníci, projekty, fotografie, apod.),
- odesílání zpráv.

Tento modul byl také použit pro uživatelskou přívětivost, pro uživatelský komfort, kdy šetřil čas pro jinak potřebnou obnovu stránky. Autorem tohoto modulu je Jack Moore, více o autorovi a pluginu na www.jacklmoore.com/colorbox.

6.5.2 AJAX

ASP.NET MVC framework umožňuje velice automatizované vytvoření funkční webové aplikace. Taková se dá ale vylepšit o další technologie, pro které už je ale nějaké progamování nutné. My vylepšíme naší aplikaci o technologii AJAX. Konkrétně AJAX v kombinaci s jQuery. Hlavní funkcí této technologie je odbourávání nutnosti znovuobnovení celé stránky při nějaké akci uživatele aplikace. Např. při hlasování v nějaké anketě, kde by stačilo změnit pár čísel, musí se bez AJAX technologie odeslat dotaz na server, který nám odpoví celou stránkou, kterou překreslí. Ale s technologií asynchronního Javascriptu můžeme docílit mnohem plynulejší práce, blížící se komfortní interaktci v desktopovém prostředí, můžeme na pozadí odesílat příkazy na server, který nám pak odpovídá příslušnou formou. Více o technologii AJAX v knize [2], kapitola 19, *Technologie AJAX a ASP.NET*.

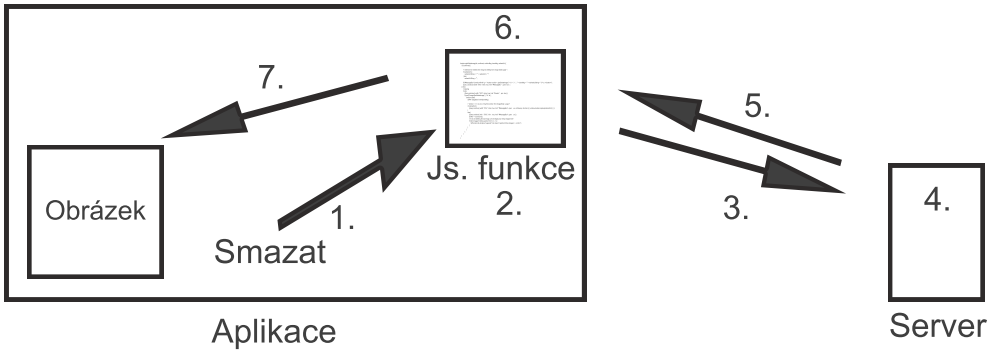
Nejprve si v následujícím seznamu uvedeme jednotlivé akce uživatele, pro které jsme zvolili jejich vykonávání pomocí technologie AJAX.

- Mazání obrázků na stránce detailu projektu,
- mazání projektu,
- mazání klienta,
- přidávání anotací do obrázku,
- mazání anotací z obrázku,
- přidávání komentářů,
- odesílání zpráv.

Na příkladu mazání obrázku si vysvětlíme, jak AJAX funguje a jak byla tato technologie nasazena do aplikace. V ostatních případech je pak situace obdobná. Nejprve na obr. 6.5 vidíme zjednodušené schéma pro naší situaci. Jednotlivé kroky si pak vysvětlíme.

Na obr. 6.5 vidíme dvě hlavní části, aplikaci a server. Podrobnější vysvětlení jednotlivých kroků je v následujícím seznamu

- 1. uživatel provede nějakou akci, v našem případě chce smazat obrázek,



Obrázek 6.5: Volnější abstrakce AJAX komunikace.

- 2. mazání je zahájeno voláním Javascriptové funkce, ta převezme dané parametry, jako je jméno obrázku k mazání, id prvku na stránce apod. Před samotným mazáním vyvolá okno Colorboxu s výzvou na potvrzení mazání,
- 3. po potvrzení odesílá data příslušnou jQuery funkcí na server - na URL, která směřuje na konkrétní metodu, která AJAX post zpracuje,
- 4. na serveru jsou data zpracována, smaže se obrázek z databáze a souborového systému,
- 5. server nakonec vrátí kladnou nebo zápornou odpověď v podobě Json objektu,
- 6. na tuto odpověď čeká jQuery funkce *.post()* a zpracuje příchozí odpověď od serveru,
- 7. funkce zobrazí kladnou nebo zápornou dpověď o úspěšnosti akce. Pokud bylo mazání úspěšné, vymaže funkce pomocí jQuery *.remove()* danou fotografii z DOM. Tímto způsobem zmizí z dané stránky, aniž by musela být obnovena.

Na následujícím kousku kódu 6.2 vidíme část funkce, která se starala o odeslání AJAX dotazu, přijmutí odpovědi ze serveru a provedení změn. Zdrojový kód je pro přehlednost komentován na příslušných řádcích.

```
$.post("/Image/AjaxDeleteImage", { "id": id }, /* url, kam je směřován AJAX
dotaz a jednotlivé POST atributy */
    function (data) { // Json odpověď je odchycena do proměnné data
      $("#MessageBox").html(data.msg); /* konkrétní odpověď od serveru
      o úspěchu či neúspěchu mazání se uloží přímo do HTML objektu
      s id = MessageBox */
      if(data.state){ // pokud bylo mazání úspěšné state == true
        $("#id-" + id).remove(); // smazat obrázek i z DOM
      }
      /* zobrazení zprávy o úspěšnosti akce: (Colorbox zobrazí inline HTML
      kód z objektu s id = MessageBox, do kterého jsme vložili odpověď ze
      serveru) */
      jQuery.colorbox({ width: "50%", inline: true, href: "#MessageBox",
      open: true });
    }
);
```

Zdrojový kód 6.2: jQuery kód pro AJAX komunikaci se serverem.

6.5.3 Modul hoverIntent

Tento jQuery modul byl vybrán pro rozbalování jednotlivých podmenu v hlavní navigační liště. Rozbalování a sbalování podmenu jde lehce docílit jen pomocí jQuery knihovny a funkce *.hover()*. HoverIntent má ale další funkci, rozbalí podmenu jen v případě, kdy se skutečně zastavíme nad nějakou položkou menu. Dají se zde lehce v kódu nastavit časy rozbalení, citlivosti na přejetí kurzorem myši apod. Autorem tohoto pluginu je Brian Cherne, více o modulu a autorovi na www.cherne.net/brian/resources/jquery.hoverIntent.html.

6.5.4 Modul imgAreaSelect

V kapitole o návrhnu budoucí aplikace jsme kladli velký důraz na potřebu přidávání poznámek přímo do jednotlivých obrázků. Tento další jQuery modul, jak z jeho názvu vyplývá, umožňuje vybírat z obrázku určitou část. Jednoduše se klikne např. do nějaké fotografie, kde se myší vybere její libovolně velká čtyřúhelníková oblast. Autorem tohoto modulu je Michal Wojciechowski, více o autorovi a pluginu na http://odyniec.net/projects/imgareaselect.

6.5.5 Modul imgnotes

Tento jQuery plugin využívá a rozšiřuje předchozí modul pro výběr oblasti obrázků o možnost přidávat poznámky ke každé vybrané ploše, dynamicky ukládá každou tuto poznámku s ohraničenou oblastí přímo do dané fotografie/obrázku. Autorem je Tarique Sani a více o tomto jQuery rozšáření naleznete na http://www.sanisoft.com/blog/2008/05/26/imgnotes-jquery-plugin. Na tento modul ale ještě musela být napojena javascriptová funkce, která pomocí AJAX komunikace se serverem ukládala údaje o přidaných poznámkach do databáze, aby tak byly poznámky v obrázku dostupné i po obnovení stránky.

6.5.6 Modul filedrop

Další z důležitých funkcí vytyčených v kapitole návrhu aplikace byla možnost uchopení souboru typu obrázku z desktopu (operačního systému) přímo do okna prohlížeče (do aplikace). Pro realizaci této funkce byl zvolen jQuery modul využívají HTML5 file api. Jeho použití je velice snadné, umožňuje navolit maximální možný počet souborů pro přenos, nebo jejich velikost. Má přednastavenou možnost reagovat na akce jako *dragLeave, uploadStarted, beforeEach, dragOver* a další. Tento modul ve svém základu nahrává soubory hned na server. Tuto vlastnost jsme v naší aplikaci potlačili, neboť chceme, aby uživatel případně přidal jméno souboru, popis nebo jej přidal k nějakému projektu. Hned po vložení souboru do příslušných oblastí aplikace (tzv. drop zón) se vyvolá okno Colorboxu. Uvnitř bude náhled obrázku, který v tuto chvíli ještě není nahrán na server. Uživatel podle potřeby doplní zmíněné informace o obrázku a ten se nahraje až po jeho odeslání příslušným tlačítkem.

HTML5 file api a „drag & drop“ bohužel nemá podporu u všech webových prohlížečů, modul filedrop tak v současné chvíli nepodporují ani u nejnovějších verzí. Podporu prokázal jen prohlížeč Firefox 3.6+ a Google Chrome. Ještě větší podporu HTML5 slibují Safari [6] a Internet Explorer [3] od příštích verzí (Safari 6, IE 10). Autorem modulu filedrop je weixiyen, více na https://github.com/weixiyen/jquery-filedrop.

6.6 HTML5

Při použití HTML5 jsme museli dávat pozor, jakou mají jednotlivé části tohoto jazyka podporu u webových prohlížečů. Použili jsme například nový tag *<nav></nav>*, který sémanticky nese význam navigační části s odkazy. V naší aplikaci byl použit pro hlavní navigační lištu. Podporu pro tento tag mají všechny nejpoužívanější web. prohlížeče (IE až od verze 9). Vše o novém standardu HTML na www.w3schools.com/html5.

6.7 Jazyk aplikace, lokalizace

Jedna z prvních věcí, která byla při programování navržena a vytvořena, byla možnost výběru jazyka, nějakého způsobu lehkého přeložení do dalších jazyků. V open source projektu Prestashop (systém pro internetový obchod) je pro překlad do jiných jazyků používáno asociativní pole. Malá ukázka na zrojovém kódu 6.3.

```
$_LANG['404_1d3ac0a5826e47e3c2761085e97877f6'] = 'Stránka není dostupná'.
```

Zdrojový kód 6.3: Ukázka pole jazyka ze systému Prestashop.

Tento způsob byl převzat pro naší aplikaci. Byla vytvořena třída, při jejíž použití se konstruktoru předají jméno řídicí logiky (konkrétní controller třída) a jméno akce (jedna z metod onoho controlleru). Tato třída nazvaná LanguageController pak obsahuje seznam s jednotlivými řetězci. Ty jsou pak použity v jednotlivých „View“ souborech, do kterých je z určitého řídícího souboru (controller) seznam s jazykem odeslán ve speciální datové proměnné ViewBag. Ukázka jak vypadá seznam, jeho inicializace a použití vidíme na zdrojovém kódu 6.4.

```
/* obsah třídy LanguageController.cs: */
SortedList languageList = new SortedList(); // vytvoření seznamu jazyka
languageList.Add("Description", "Popis"); // konkrétní text
languageList.Add("Created", "Vytvořen");

/* ProjectController.cs: inicializace jazyka, odeslání jazykového pole na příslušný pohled
    (View) */
ViewBag.Language = new Language(this.RouteData.Values["controller"].ToString(), this.
    RouteData.Values["action"].ToString()).getLanguage();

/* použití tohoto jazyka na stránce detailu projektu (v souboru Details.cshtml) */
@{
  var Language = ViewBag.Language;
}
<b>@Language["Created"]</b> @Model.Created
<b>@Language["Description"]</b> @Html.DisplayFor(model => model.Description)
```

Zdrojový kód 6.4: Způsob jazykové podpora v naší aplikaci.

6.8 Jiné varianty jazykové podpory

Obecně existují dva různé způsoby, jak řešit potřebu podpory více jazyků.

- 1. pro každý jazyk existuje samostatná šablona (každý „View“ musí být tolikrát kolik je podporovaných jazyků),
- 2. každý řetězec na přeložení je generován nějakou proměnnou.

V aplikaci byl použit druhý způsob. Je však nutno konstatovat, že bylo zbytečné vymýšlet na to nějaký vlastní postup, třídy apod., neboť ASP.NET MVC 3 má skvělou vlastní podporu pro lokalizaci, kdy může podle jazyka webového prohlížeče nasadit konkrétní jazykovou sadu. Místo tvorby vlastní třídy, stačilo použít výhody tzv. zdrojů (Resources). Této problematice se podrobně věnuje návod [1].

6.9 Uložení obrázkových souborů

Z předchnozích kapitol vyplývá, že nejdůležitějšími daty budou obrázkové soubory. Bude jich velké množství, budou mít různé velikosti a formáty. Přirozeně tak stojíme před volbou co nejvhodnější varianty pro jejich uložení. Obecně existují dvě možnosti:

- 1. uložení obrázů v databázi,
- 2. uložení na filesystému serveru.

Jako speciální možnost by se dala použít kombinace dvou těchto přístupů. Ze článku publikovaném na Microsoft Research [7] vyplívá, že soubory menší 256K je nejlepší ukládat v databázi, soubory nad 1M naopak na filesystému. Při návrhu aplikace byl ale kladen důraz na možnost ukládat soubory větší než 1M, proto byla zvolena možnost ukládání na disky serveru.

Výhoda 1. varianty je v manipulaci s daty. Přístup k nim je jako ke klasikým datům v databázi. Mazání je také jednodužší. Ovšem samotné zobrazování fotografií a obrázků pak požaduje trochu víc programování.

Naproti tomu 2. varianta má jednodužší zobrazování. Stačí mít uloženou cestu k souboru (v naší aplikaci v tabulce Images). Při mazání ovšem nestačí smazat příslušný záznam z databáze, musí se mazat i konkrétní soubor z filesystému serveru. Další výhodou je případná možnost sdílet takové soubory s dalšími aplikacemi, neboť k nim je jednodužší přístup.

Všechny obrázky, které jsou uživateli nahrávány do aplikace, se ukládají do stejného adresáře */Content/Uploads/Images*. Musel být zvolen vhodný způsob pojmenování jednotlivých souborů. Lehce se může stát, že by uživatel nahrával soubor se stejným jménem, který už v systému existuje, tímto by se mu přemazal novým souborem. Vyřešit to můžeme jednak ohlášením kolize jmen, v naší aplikaci byl však použit jiný model. Byla použita speciální struktura Guid, globální unikátní identifikátor. Každý soubor tak dostane jméno z této struktury. Nemůže se tak stát, že by došlo k problémům se stejnými jmény. Jak konkrétně se tvoří jméno nového souboru, vidíme na následující ukázce zdrojového kódu 6.5.

```
string filename = Guid.NewGuid().ToString() + fileExtension;
```

Zdrojový kód 6.5: Tvorba nových jmen pro obrázkové soubory.

6.10 Přihlašování do aplikace

Původně byl navrhován klasický model uživatelů v databázi, kde by byla tabulka s uživateli, hesly apod. Nic takového ale opět není potřeba programovat a sestavovat. ASP.NET MVC 3 framework poskytuje membership balíček, který spolupracuje s databází ASPNETDB.MDF (přítomna od začátku v adresáři App_Data). Obsahuje různé třídy a metody, které nám umožňují vytvářet nové užvatele, jejich autentizaci pomocí cookies apod. Pro

potřeby přihlašování byla z oficiálního návodu MVC music store převzata třída AccountController.cs s dílčími změnami použitelná pro naší aplikaci. Framework nám dále umožňuje tvořit různé role, k těmto rolím pak přiřazujeme jednotlivé uživatele. Pomocí těchto rolí můžeme velice jednoduchým způsobem omezit přístup k jednotlivým třídám (tedy i stránkám aplikace) nebo dokonce metodám. Ukázka omezení třídy AdminController ve zdrojovém kódu 6.6.

```
[Authorize(Roles = "Administrator")]
public class AdminController : Controller
{
```

Zdrojový kód 6.6: Omezení přístupu k třídě jen pro uživatele s rolí „Administrator“.

Ke všem stránkám, které jsou tak vázané na řídící třídu AdminController, má přístup jen ten uživatel s oprávněním (rolí) „Administrator“. Při omezení konkrétní metody je situace obdobná. Omezení může platit pro libovolný počet rolí. V naší palikaci byly vytvořeny dvě role, „Adminitrator“ a „Client“.

6.11 oAuth přihlašování

V kapitole o technologiích jsme si řekli, že můžeme tento protokol použít pro přihlašování do aplikace pomocí nějaké třetí strany. My umožníme přihlašování pomocí Google účtu, Yahoo účtu a OpenID. Pokud by však mohl každý, kdo disponuje s takovými účty, vstoupit do zabezpečených částí aplikace, byla by aplikace prakticky k ničemu. Při přihlašování pomocí oAuth se tedy testuje, jestli je daný email, pod kterým se přihlašuje, registrovaný i v systému.

Pro ASP.NET MVC3 byla použita varianta *dotnetopenauth*. Více o této technologii na www.dotnetopenauth.net. Dokumentace na tomto webu, nebo na www.oauth.net není příliš propracovaná a při implementaci nebyla příliš nápomocná.

Vyšel jsem z existujícího opensource projektu www.nerddinner.com, který je napsán v ASP.NET MVC 3/Razor a používá také oAuth přihlašování. Více o tomto projektu na http://nerddinner.codeplex.com.

Pro použití oAuthDotNet je nejprve nutné tento balíček k projektu přidat. To se dělá jednoduchým příkazem *Install-Package DotNetOpenAuth* v tzv. balíčové konsoli (Package Manager Console) ve Visual Web Developer. Tím se nám do projektu přidá podpora oAuth protokolu pro ASP.NET. oAuth komunikaci zajišťovala třída AuthController.cs, jejímž utorem je Andrew Arnott.

Na obrázku 6.6 je ukázka stránky s přihlášením do aplikace.

6.12 Sdílení s klientem

Jednou z hlavních funkcí, která má uživatelům aplikace usnadnit a umožnit co nejrychlejší komunikaci, je automatizované sdílení. Od začátku bylo sdílení navrhováno ve stylu „jedním klikem“.

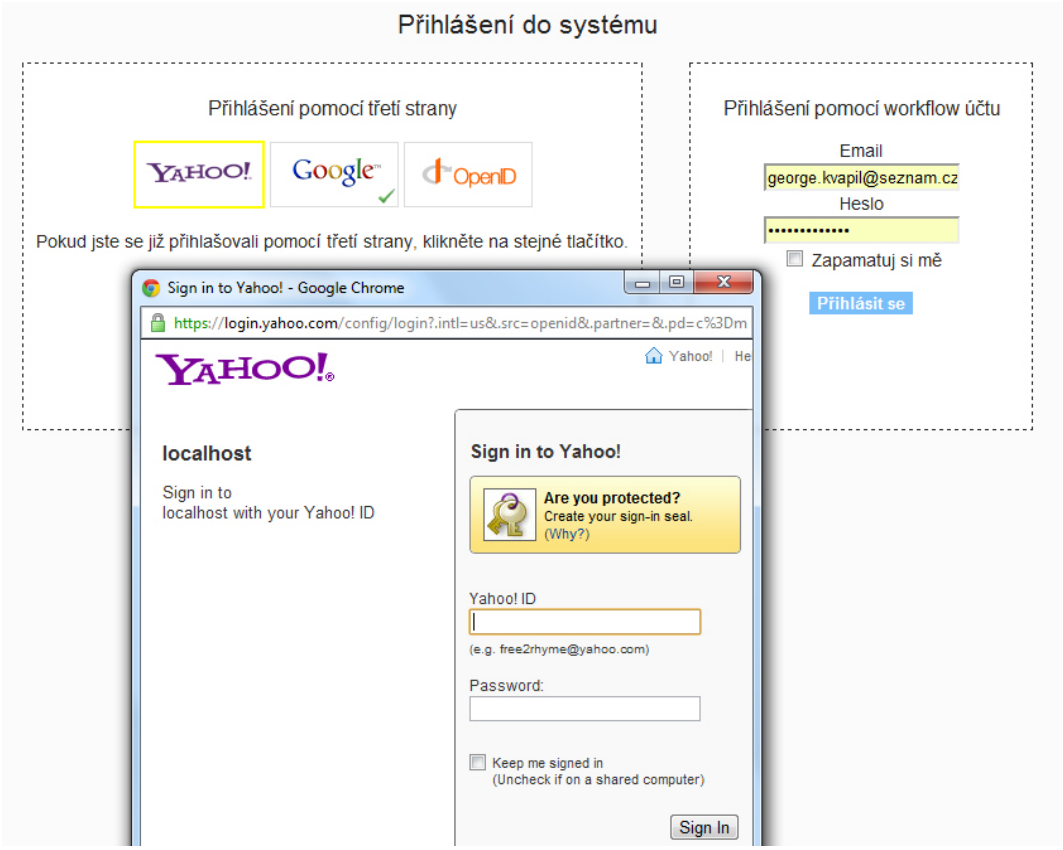
Sdílení funguje následujícím způsobem. Jednotlivé akce si uvedeme v seznamu. Pořadí akcí demostruje ukázkové použití např. nějakým designérem.

- 1. vytvoření klienta.
- 2. vytvoření projektu, který je vázán na onoho klienta.

- 3. do projektu se přidají návrhy (obrázky).
- 4. když je projekt připraven, klikne se na tlačítko „Sdílet s klientem“.
- 5. klientovi se na jeho e-mailovou adresu zadanou při jeho vytvoření v aplikaci odešle zpráva s vysvětlujícím textem, přihlašovacím heslem a odkazem na projekt. Klient je také vyzván ke změně hesla. Pokud je klientovi odesílán další projekt ke schválení, už je informační e-mail bez přihlašovacích údajů.

Pro potřeby odesílání e-mailových zpráv byl použit balíček ActionMailer. Pro nainstalování opět stačil příkaz *Install-Package ActionMailer*. Autorem tohoto balíčku je Scott Anderson.

Více o balíčku ActionMailer na https://bitbucket.org/swaj/actionmailer.net/wiki/Home.



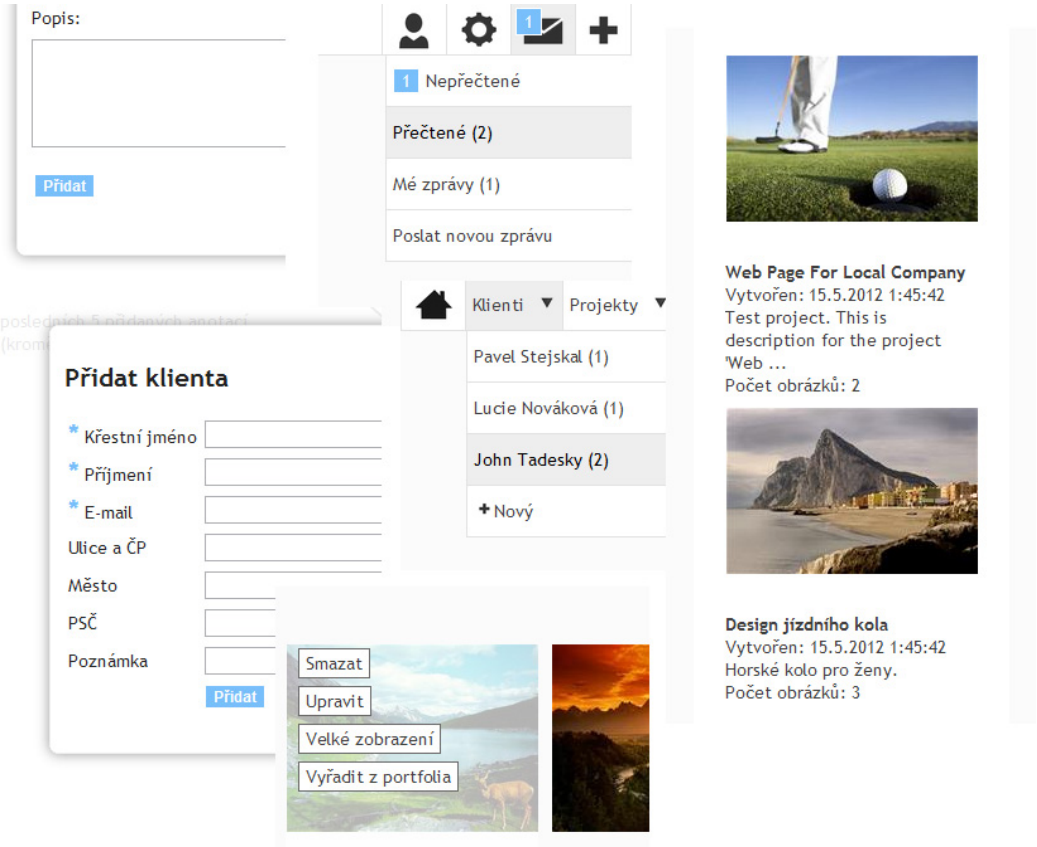
Obrázek 6.6: Ukázka přihlášení do aplikace.

Kapitola 7

Design aplikace

Pro aplikaci byl zvolen minimalistický design. V kapitole 4.4 jsme rozdělili layout aplikace pouze na 2 části, navigační lištu a obsah konkrétní stránky. Samotná aplikace pak obsahuje pouze 4 různé barvy. Design je navržen tak, aby byl co nejjednodužší. V navigační liště byly použity ikony, které co nejlépe vystihují jejich význam. Ovládání musí být co nejednodužší a intuitivní, aby osvojení aplikace bylo co nejrychlejší.

Pro co nejlepší představu slouží následující obrázek 7.1 s různými částmi aplikace.



Obrázek 7.1: Minimalistický design. Různé části aplikace.

Kapitola 8

Závěr

Hlavním z cílů této práce bylo navrhnout a poté implementovat bohatou webovou aplikaci založenou na frameworku ASP.NET MVC 3/Razor, který poskytuje obrovské množství funkcí a rozšíření, a jedině po důkladném studiu tohoto návrhového vzoru mohlo být přikročeno k návrhu vlastní webové aplikace.

Ze zadání nebylo specifikováno zaměření aplikace. Bylo zvoleno téma sdílení práce grafického designéra s klientem. Tento cíl se podařilo splnit. Designér je schopen své práce ukládat do systému a sdílet s klientem či veřejností. Mezi sebou pak mají umožněnou komunikaci v podobě soukromých zpráv, komentářů či přidávání popisek do konkrétních návrhů.

Před návrhem vlastní aplikace byla nejprve provedena analýza existujících řešení. Systémů, které by se úzce specializovaly na sdílení práce s klientem, není příliš mnoho. Byly tak analyzovány i takové způsoby, které se pro tuto problematiku používají, přestože pro ni nejsou primárně určeny. Analýza proběhla u tří existujících systémů, obecně u e-mailových služeb, www.prevue.it a www.dropmark.com.

Jedině po důkladné analýze jsme mohli navrhnout aplikaci, která by si vzala z existujících řešení jen klady a naopak odbourala jejich nevýhody. Na přidanou hodnotu v podobě funkcí a šetření času uživatelům aplikace při jejím použití byl kladen velký důraz. Vzhled aplikace je minimalistický, navržen tak, aby bylo použití systému lehké a intuitivní.

Dále bylo cílem bohatou webovou aplikaci implementovat. Slovo „bohaté“ v tomto kontextu popisuje aplikaci, která používá moderní technologie či postupy. Před implementací bylo nastudováno množství moderních internetových technologií, které jsou pro takovou aplikaci použitelné. Výsledná aplikace pak tyto technologie využívá. V práci je popsáno nasazení jednotlivých technologií. Čtenář se dočte o procesu implementace od prvního kroku, popsána budou použitá vývojová prostředí a problémy, které nastaly při programování.

Co se týče rozšíření, může být systém vylepšen o velkou řadu služeb a funkcí. Může podporovat více druhů dat, aplikace podporuje pouze obrazové formáty. Chybí tak videa, zvukové formáty nebo další libovolné soubory. Dále může obsahovat komunikaci v reálném čase bez potřeby obnovovat stránku (chat), či propojit jednotlivé instalace této aplikace tak, aby se mohli jednotliví uživatelé (designéři) domlouvat. Aplikace by mohla obsahovat aparát pro nasazování různých modulů pro rozšíření funkčnosti, či podporu různých šablon vzhledů. Její použití by mohlo být jak v podobě stažitelného balíčku, tak formou sdílené aplikace s nutností registrace, kde by však byl omezený prostor.

V rámci zadání měla být vytvořena internetová prezentace projektu. Pro tuto potřebu byl využit web www.codeplex.com pro „open source“ projekty od společnosti Microsoft. Domácí stránka vytvořené prezentace je https://workflowapp.codeplex.com. Samotná aplikace, její zdrojové kódy se dají stáhnout v záložce „SOURCE CODE“.

Aplikace úspěšně řeší nepopiratelné problémy při komunikaci klienta s dalším subjektem, kde informační technologie můžou touto formou velice pomoci.

Literatura

[1] Afana, N.: ASP.NET MVC 3 Internationalization. 2011, [online].
URL http://afana.me/post/aspnet-mvc-internationalization.aspx

[2] Bill Evjen, D. R., Scott Hanselman: *ASP.NET 3.5 v jazycích C# a Visual Basic.* Holandská 8, 639 00 Brno: Computer Press, a.s., 2009, ISBN 978-80-251-2069-9.

[3] Bright, P.: First preview of Internet Explorer 10 arrives with new features. 2011, [online].
URL http://arstechnica.com/information-technology/2011/04/first-preview-of-internet-explorer-10-arrives-with-new-features/

[4] Dykstra, T.: Creating an Entity Framework Data Model for an ASP.NET MVC Application. 2011, [online].
URL http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application

[5] Galloway, J.: MVC Music Store. 2011, [online].
URL http://www.asp.net/mvc/tutorials/mvc-music-store

[6] Marshall, G.: Safari 6 rumours: what you need to know. 2011, [online].
URL http://www.techradar.com/news/internet/web/safari-6-rumours-what-you-need-to-know-943486

[7] Russell Sears, J. G., Catharine Van Ingen: To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem. 2006, [online].
URL http://research.microsoft.com/apps/pubs/default.aspx?id=64525

[8] The Radicati Group, I.: Email Statistics Report, 2009-2013. 2009, [online].
URL http://www.radicati.com/?p=3237

[9] Zakas, N. C.: How many users have JavaScript disabled? 2010, [online].
URL http://developer.yahoo.com/blogs/ydn/posts/2010/10/how-many-users-have-javascript-disabled/

Seznam příloh

Příloha A

Obsah DVD

1 Zdrojové kódy aplikace.

2 Písemná zpráva v PDF.

3 Zdrojové kódy písemné zprávy.

4 Plakát.

5 Dokumenty příloh.

Příloha B

Manuál

Pro zprovoznění aplikace na lokálním stroji doporučuji Microsoft Visual Web Developer 2010 ke stažení z http://www.asp.net/mvc zdarma.

Jediné nutné nastavení aby aplikace bez chyb fungovala na lokálním počítači je vytvoření adresáře pro odchozí e-mailovou poštu. Vytvořte tedy složku *mails* na disku *C*. Z kódu B.1 umístěném v konfiguračním souboru *Web.config* je patrné proč je to nutné. Na lokálním stroji, kde není aplikace v ostrém provozu nechceme aby byly elektronické zprávy odeslány. Pro demonstraci funkčnosti však necháváme tyto zprávy odeslat do vytvořené složky *C:\mails*.

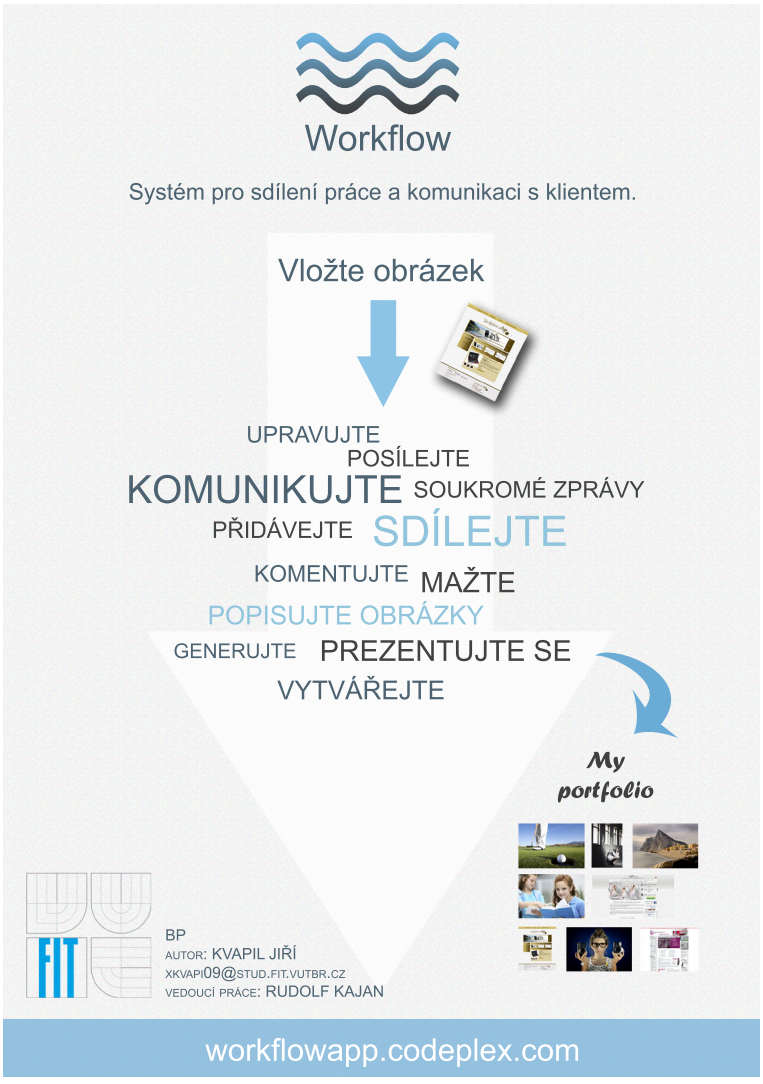
Tam se budou vyvářet soubory s koncovkou *.eml*, které se dají pro kontrolu otevřít např. v programu Windows Live Mail.

```
<mailSettings>
  <smtp deliveryMethod="SpecifiedPickupDirectory">
    <specifiedPickupDirectory pickupDirectoryLocation="C:\mails" />
    <network host="localhost" />
  </smtp>
</mailSettings>
```

Zdrojový kód B.1: Nastavení pro odchozí poštu.

Příloha C

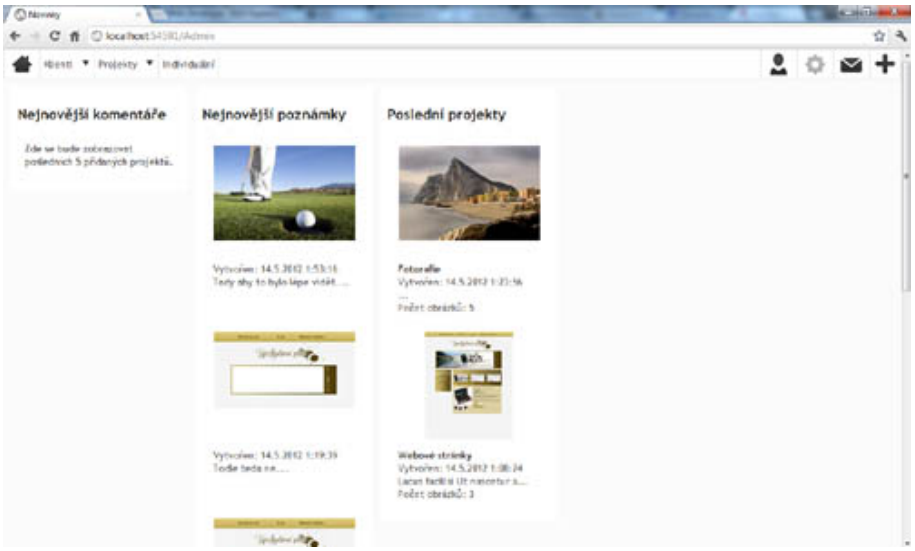
Plakát



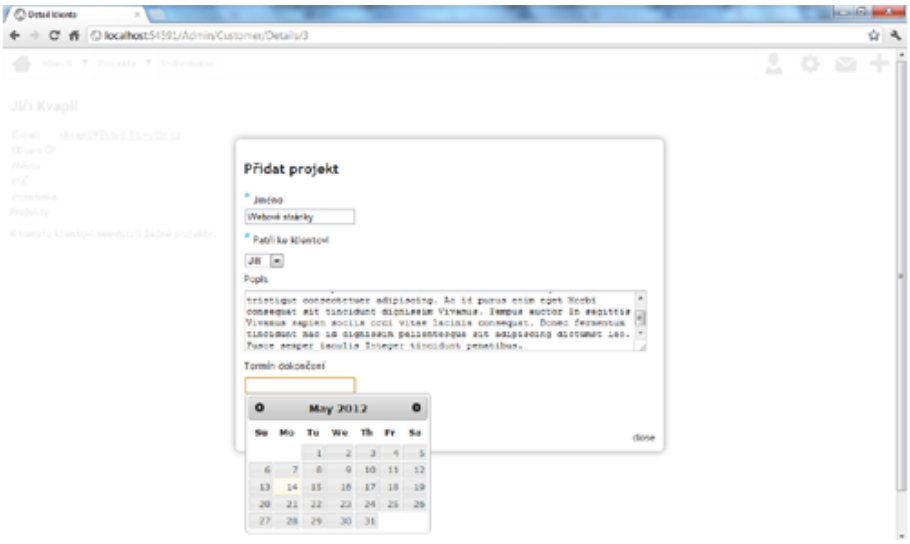
Obrázek C.1: Plakát, jehož vytvoření bylo dané zadáním.

Příloha D

Náhledy aplikace

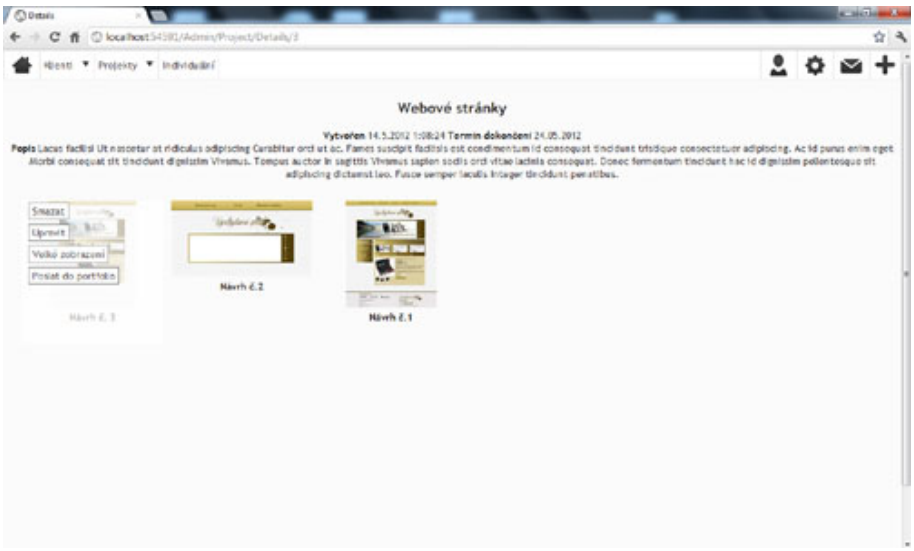


Obrázek D.1: Úvodní stránka administrace zobrazující poslední aktivitu v systému.

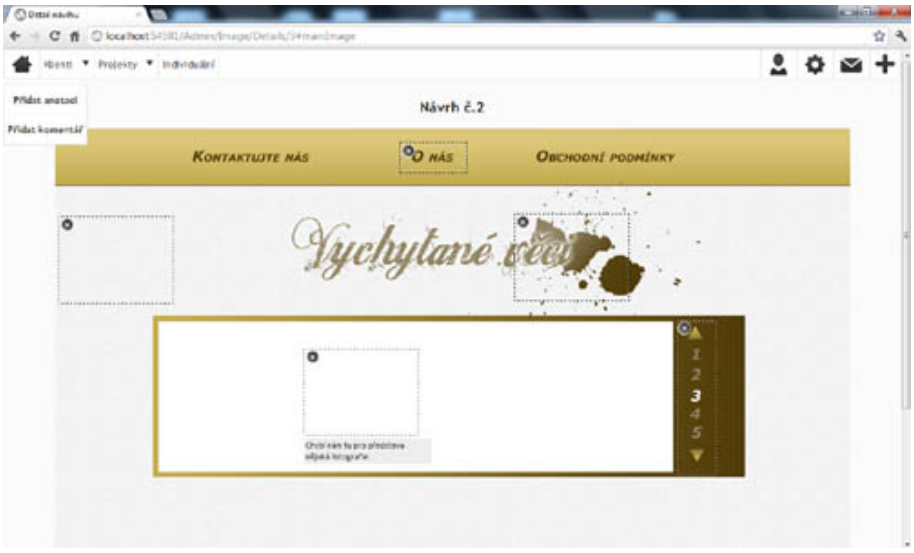


Obrázek D.2: Přidání projektu.

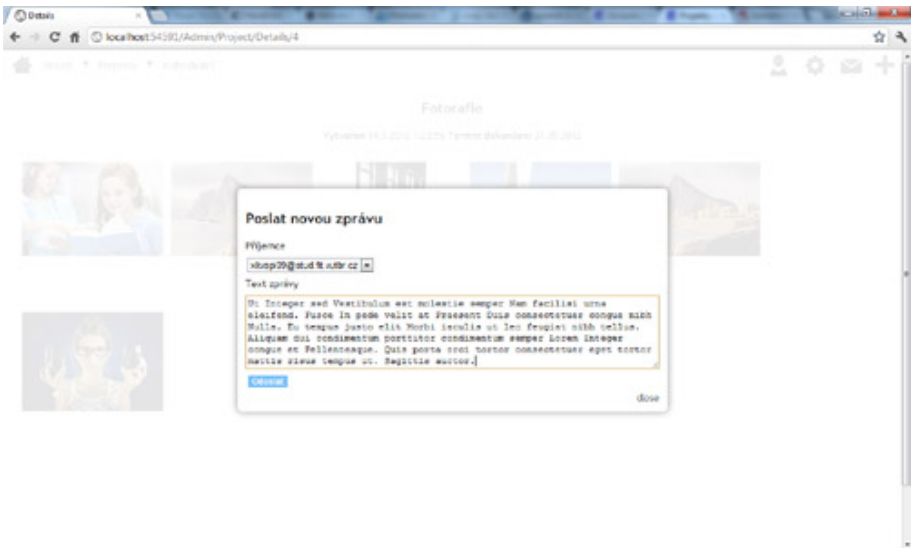
Obrázek D.3: Přidání obrázku k projektu.



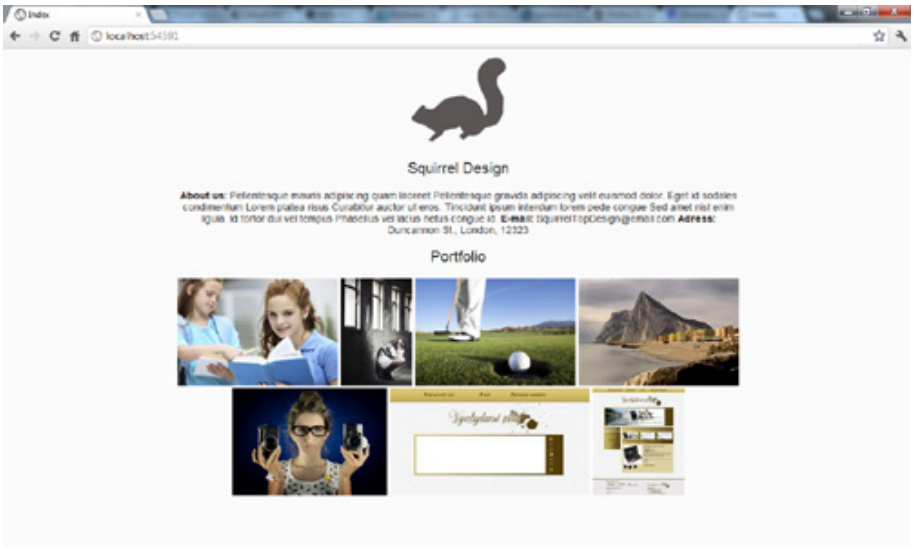
Obrázek D.4: Stránka s detailem vloženého projektu.



Obrázek D.5: Stránka s detailem konkrétního obrázku s několika popiskami.



Obrázek D.6: Odeslání soukromé zprávy.



Obrázek D.7: Portfolium, kam byly odeslány jen vybrané obrázky.