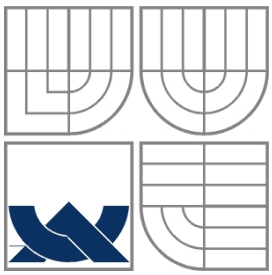
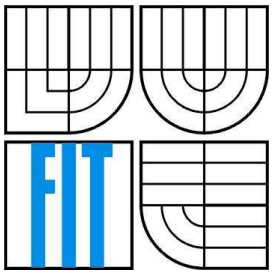


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODUL INFORMAČNÍHO SYSTÉMU PRO VYTVÁŘENÍ OBRÁZKŮ

UNIT OF INFORMATION SYSTEM FOR CREATION OF PICTURES

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — Ondřej Hanák
AUTHOR

VEDOUCÍ PRÁCE — Ing. Roman Lukáš, Ph.D.
SUPERVISOR

BRNO 2008

Abstrakt

S rozmachem elektronického obchodování přibývá prodejců nabízejících konfigurovatelné výrobky. V této práci jsou diskutována existující řešení, specifikovány požadavky a implementován univerzální systém pro nasazení v praxi. Při realizaci byl kladen důraz na použití volně dostupných technologií a maximální použitelnost a přístupnost aplikace.

Klíčová slova

informační systém, konfigurační systém, konfigurátor, vytváření obrázků, knihovna gd2

Abstract

With the boom of electronic marketing, the number of traders offering configurable goods increases. In this thesis, the existing solutions are discussed, the demands are specified and the universal system for this usage itself is implemented. During this process, the accent was put on use of freely available technologies and maximal usability and accessibility of the application.

Keywords

information system, configuration system, image creation, gd2 library

Citace

Hanák Ondřej: Modul informačního systému pro vytváření obrázků. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Licenční smlouva

Licenční smlouva je vložena v archivním výtisku uloženém v knihovně FIT VUT v Brně.

Modul informačního systému pro vytváření obrázků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením
Ing. Romana Lukáše, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Hanák
10. 5. 2008

Obsah

Úvod

Spolu s rozmachem elektronického obchodování přibývá i prodejců nabízejících ve svých internetových obchodech velké množství výrobků, které se navzájem liší jen v některých detailech, zatímco základ zůstává stejný. Nebo množství produktů, které jsou sestaveny z částí, které si zákazník může vybrat a sestavit si tak komplet co nejvíce vyhovující jeho představám. Tento přístup není nikterak nový a pro příklad z praxe nemusíme chodit daleko, třeba v oblasti výpočetní techniky na něj narazíme při výběru počítačové sestavy.

Na co však současné konfigurátory v hojné míře zapomínají, je vzhled konečného produktu. Při výběru PC sestavy to sice není ten nejdůležitější aspekt, který je navíc většinou ovlivněn pouze použitou skříní, ale třeba u výběru domovních dveří se situace zásadně mění. U relativné rozměrné a finančně nákladné věci, kterou bude mít kupující denně na očích, ho vzhled jistě zajímat bude a jen málokdo by v takové situaci kupoval příslovečného „zajíce v pytli“.

Je tedy na prodejci, aby s měnící se konfigurací produktu zákazníkovi nabídl i jeho odpovídající vizuální reprezentaci. Nejvíce názornou formou jsou bezesporu fotografie. Pokud sortiment tvoří pět modelů dveří ve čtyřech barevných odstínech, každý se třemi možnostmi prosklení a pěti typy klik, vychází to na tři sta fotografií. Jakmile do nabídky přibude jedna nová klika, je zapotřebí vyfotografovat dalších šedesát snímků. Kdyby nastavitelných položek byly desítky, počet fotografií by vzrostl řádově na tisíce a jejich pořízení by se stalo ekonomicky naprosto neúnosné. Proto je potřeba zvolit jiné řešení.

Varianta, kterou se zabývá tato práce, je uchovávání obrázků jednotlivých částí odděleně a jejich spojování dohromady až podle požadavků zákazníka. Objem samostatných souborů se tak z řádů tisíců vrátí zpátky do desítek, což se pozitivně projeví na finanční a datové náročnosti jejich pořizování, zpracovávání i uchovávání. Systém jsem realizoval jako samostatný modul, který může správce webu vestavět do již existujícího elektronického obchodu. Součástí je i on-line editor umožňující definovat nové produkty a doplňovat další volitelné součásti.

První kapitolu jsem věnoval podrobnějšímu rozboru problému – analýze existujících řešení, nalezení jejich společných rysů, kladů i záporů. Na tomto základě specifikuji požadavky na výsledný systém a jeho datový model. V druhé kapitole rozebírám požadavky na použité technologie a softwarové nástroje a zdůvodňuji konkrétní výběry v jednotlivých kategoriích včetně porovnání s případnými alternativami. Ve třetí kapitole stručně probírám konkrétní implementaci systému. Pro větší přehlednost má uživatelská i administrační část systému vlastní podkapitolu. Jako samostatnou přílohu jsem sestavil uživatelský manuál, který má sloužit správci při uvádění systému do chodu a při tvorbě datových podkladů. Na přiloženém CD se kromě samostatného modulu nachází i kompletní funkční ukázka konfigurátoru dveří.

1 Rozbor problému

V této kapitole se věnuji průzkumu existujících řešení, nalezení jejich společných vlastností, výhod a nedostatků. Na základě těchto informací specifikuji požadavky na vlastní systém a jeho celkovou koncepci včetně datového modelu.

1.1 Průzkum existujících řešení

Před započetím práce na vlastní realizací projektu jsem se rozhodl provést průzkum existujících řešení dostupných na internetu. Zaměřil jsem se na dvě odvětví: dřevozpracující průmysl, konkrétně výrobce dveří a na prodejce litých kol pro automobily.

Příklady z praxe jsem hledal pomocí české mutace vyhledávače *Google*. První odvětví bylo navrženo v zadání bakalářské práce, hledal jsem tedy frázi „konfigurátor dveří“. Relevantních výsledků bylo málo a z nich mým představám odpovídal pouze jeden – stránky Prostějovské firmy Truhlářství Martin Čuda [1]. Obrazové materiály z této stránky jsem použil pro ukázkovou aplikaci postavenou systému popisovaném v této práci. Ve výsledcích jsem mimo jiné dostal několik odkazů na stránky s konfigurátory automobilů. To mě přivedlo na myšlenku použít jako druhý příklad aplikaci pro výběr litých kol, dále jsem proto zadal frázi „konfigurátor kol“. Zde byl objem relevantních výsledků větší, jako ukázkovou aplikaci jsem z nich vybral řešení firmy ALCAR BOHEMIA, s. r. o [2].

Oblastí, ve kterých by se podobná aplikace dala použít, je ale mnohem více. Například stavebnictví – konfigurace bytů a domů, koupelen nebo kuchyní, oděvní průmysl – společenské obleky a k nim výběr různých doplňků jako kapsy, manžety, límce a knoflíky či prodej sportovního vybavení – jízdní a horská kola, lyžařské běžecké a sjezdové komplety lyží, vázání a bot.

1.1.1 Dveře Čuda

Výrobce v konfigurátoru nabízí volbu několika parametrů: primárním atributem je model dveří, jejich barva a způsob prosklení. Jako doplněk si uživatel může zvolit kliku a aby získal co nejlepší představu o tom, jak budou dveře vypadat v místě určení, má na výběr i barvu okolní stěny a podlahovou krytinu.

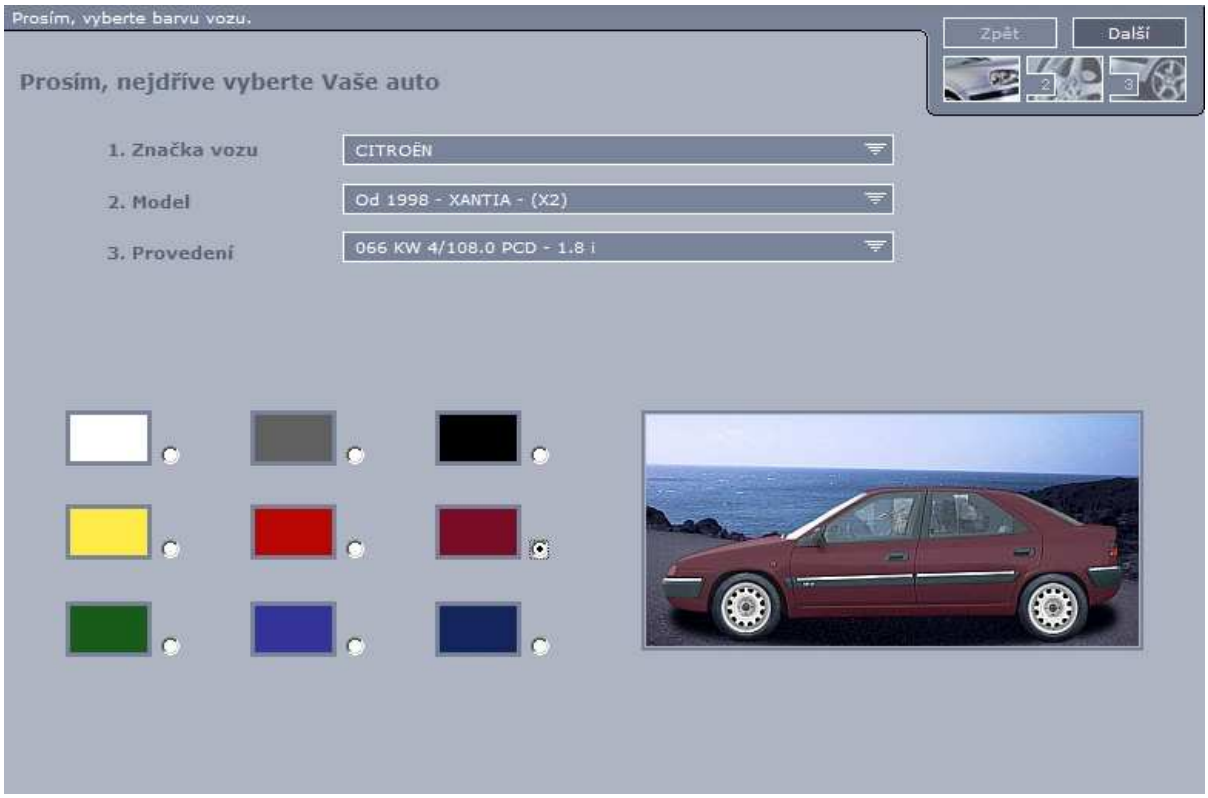
Obrázek 1. náhled konfigurátoru dveří firmy Dveře Čuda

Aplikace je realizována částečně pomocí technologie Adobe Flash a částečně s použitím JavasSriptu a pozicování HTML elementů. Práce s ní je rychlá a intuitivní, jednotlivé nabídky jsou totiž tvořeny ikonami nebo zmenšeninami komponent, které jsou danou volbou ovlivněny. Výhoda použitého řešení spočívá v rychlosti aktualizace náhledu po změně konfigurace – po stažení stránky do prohlížeče už není potřeba přenášet ze serveru žádná data. Nevýhodu spatřuji v tom, že stažení stránky trvá déle a pokud uživateli chybí podpora pro některou z použitých technologií, aplikace se stává prakticky nepoužitelnou. Může to být chybějící přehrávač Flash animací nebo z bezpečnostních důvodů vypnutý JavaScript. Konfigurace všech částí se provádí na jedné stránce.

1.1.2 Kola Alcar

Ve srovnání s předchozí ukázkou je tato aplikace rozsáhlejší, některé z konfigurovatelných položek nabízí i desítky hodnot. Možná právě proto je proces tvorby náhledu je rozdělen do tří kroků: v prvním uživatel vybírá automobil – začíná značkou, pokračuje modelem, provedením (většinou podle použitého motoru) a končí barvou. Na další stránce zatrhne požadované velikosti a designy kol. Poslední stránka už představuje automobil v kombinaci s koly, která lze vybírat z nabídky sestavené v předchozím kroku.

Nabídky v jednotlivých sekcích jsou tvořeny buď textovými vysouvacími nabídkami nebo zmenšeninami komponent podobně jako v předchozím případě. Výběrová menu nejsou tvořena standardním HTML formulářovým prvkem *select*, ale zvláštní grafickou variantou postavenou na JavaScriptu. Tím se na něm stává konfigurátor závislý podobně jako u případu dveří. Obsah těchto nabídek se sestavuje až po provedení výběru z nabídky předchozí a to přenosem dat se serveru a tím i novým načtením celé stránky. Výběr kol v druhém kroku se už provádí běžnými formulářovými prvky typu *checkbox*, ve třetím je to opět ony speciální nabídky a grafické ikony. Rozdělení výběrů na tři samostatné stránky komplikuje změnu parametrů modelu – uživatel se musí vracet, navíc jen pomocí speciálních tlačítek v konfigurátoru nebo přes klávesové zkratky, protože panel tlačítek internetového prohlížeče je skryt. Z hlediska použitelnosti a přístupnosti je tato aplikace nevyhovující.



Obrázek 2: náhled konfigurátoru kol firmy Alcar, krok 1

Obrázek 3: náhled konfigurátoru kol firmy Alcar, krok 3

1.2 Požadované vlastnosti

Na základě těchto příkladů jsem pojmenoval užitečné vlastnosti jak z hlediska funkcionality konfigurátoru, tak z hlediska návrhu uživatelského rozhraní a technického řešení ve smyslu použitelnosti a přístupnosti.

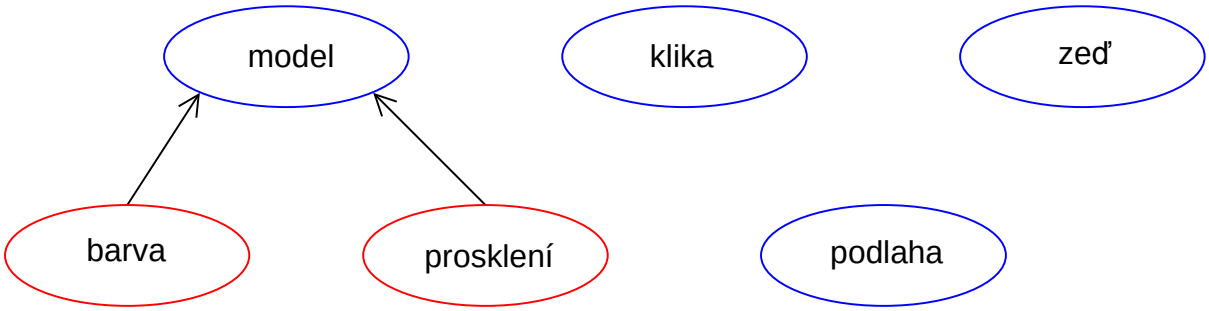
1.2.1 Funkcionalita

Z pohledu konfigurovaného předmětu jsou jeho jednotlivé části rozděleny do jakýchsi skupin, ze kterých uživatel vždy právě jednu vybere. Pro označení tohoto předmětu budu nadále používat termín model, pro skupinu součástí pojem vrstva a pro jednotlivé součásti termín komponenty.

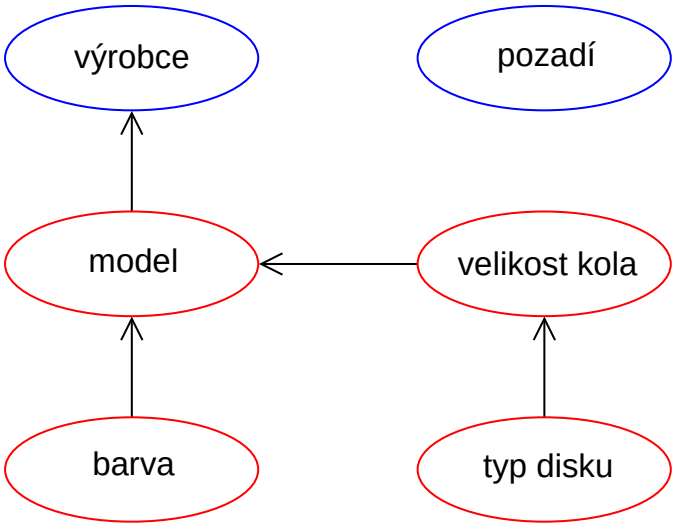
U vrstev můžeme pozorovat rozdělení na dvě skupiny. V první jsou to vrstvy nezávislé, v druhé pak závislé. Mějme například vrstvy „značka vozu“ a „model“ z příkladu konfigurátoru kol, kde druhá je závislá na první. Nabídka modelů se naplní až po výběru značky a současně se aktualizuje i při jeho změně. Nezávislá vrstva je „barva stěny“ nebo „podlahová krytina“ z příkladu konfigurátoru dveří. Ať zvolíme jakýkoliv model dveří, na vybrané barvě stěny se to neprojeví.

Z hlediska komponent přibývá další rozdělení a to sice na případy, kdy komponenta má nebo nemá grafickou reprezentaci. Toto můžeme pozorovat ve vrstvě „kliky“ u prvního příkladu: komponenta s názvem „žádná“ se v modelu neprojevuje, na rozdíl od komponent „zlatá“ nebo „stříbrná“. Některé komponenty tedy obrázek mají, jiné ne. Může nastat i situace, že čistě virtuální

bude celý obsah vrstvy – třeba volba značky vozu. Co se týče závislostí, je situace složitější než u vrstev. Zvolím-li jako automobil Škodu Octavii Scout, dostanu v závislé vrstvě „velikost kola“ rozměry 16 až 19 palců, pro model Roomster 15 až 18 zatímco pro Fabii jen 14 a 15 palců. Vidíme tedy, že nabídka v závislé vrstvě se odvíjí od výběru ve vrstvě řídící. Jednu podřízenou komponentu můžeme kombinovat s více řídícími komponentami a současně jedna řídící komponenta může povolovat volbu z více podřízených komponent. Modely závislostí v ukázkových konfigurátorech ukazují následující dva diagramy. Modře jsou v nich vyznačeny nezávislé vrstvy, červeně závislé.



Obrázek 4: diagram závislostí u konfigurátoru dveří



Obrázek 5: diagram závislostí u konfigurátoru litých kol

1.2.2 Uživatelské rozhraní

Mezi podmínkami, které jsem si na tvorbu uživatelského rozhraní stanovil, byly přehlednost a použitelnost. Zavrhnul jsem tedy vícekrokové řešení jako matoucí a soustředil veškerá nastavení na jednu stránku. Při vhodném umístění ovládacích prvků tato varianta použitelná i při větším počtu vrstev a komponent.

Z hlediska přístupnosti bylo vyloučené vytvořit rozhraní závislé na jakékoliv nadstandardní technologii, kterou by uživatel mohl ve svém prohlížeči postrádat. To ihned diskvalifikovalo komerční Adobe Flash nebo programy napsané v jazyce Java. Nakonec jsem se rozhodl nespoléhat ani na JavaScript, protože ten si může uživatel v prohlížeči vypnout. Jako jediná, 100% kompatibilní a přístupná možnost tvorby rozhraní tedy zůstalo použití standardních elementů jazyka HTML. Tady jsem se rozhodl připravit nabídky ve třech variantách: první dvě jako nečíslovaný i číslovaný seznam komponent (elementy `<ul>` / `<ol>` a `<li>`) a druhou jako výběrový box (element `<select>`). Obsahem seznamů i výběrů budou textové popisky – titulky definované zvlášť pro každou komponentu.

Výsledný náhled modelu bude tvořen pouze jedním jediným obrázkem, který se před odesláním klientovi z vybraných komponent sestaví už na serveru, takže uživatel nebude muset do prohlížeče stahovat obrázky všech dostupných komponent jako u ukázky konfigurátoru dveří, ale jen data nezbytně nutná. To mu mimo jiné umožní si vygenerovaný náhled uložit pro pozdější použití a také to zkrátí dobu nutnou pro načtení stránky.

1.2.3 Univerzální nasazení

Součástí zadání byl mimo jiné i požadavek na co nejuniverzálnější řešení, čemuž nevyhovovala ani jedna z ukázek výše. Pokud by totiž výrobce dveří zavedl do výroby nový model nebo chtěl zákazníkovi nabídnou další podlahou krytinu k náhledu, musel by upravovat Flashovou animaci a/nebo rozsáhlý kus kódu JavaScriptu. Transformovat hotový konfigurátor pro úplně jiný výrobek by bylo prakticky nemožné a musel by se naprogramovat téměř od začátku. To samé platí i u druhé ukázky. Bylo tedy nutné vymyslet jiný model.

Jako vyhovující řešení se ukázalo oddělení výkonného programového kódu a dat, která má zpracovávat. V případě užití modulu pro realizaci modelů různých předmětů by se měnila jen data, zatímco do kódu by nebylo potřeba zasahovat. Tento kód by byl soustředěn v jedné knihovně (souboru), který by výrobci poskytl funkce pro tvorbu nabídek komponent a zobrazení náhledu modelu. Knihovnu by mohl zabudovat do svých stávajících stránek bez nutnosti je nějak výrazně restrukturalizovat.

1.3 Datový model

1.3.1 Rekapitulace požadavků

Tato podkapitola shrnuje požadavky na data modelu, jenž bude v systému nutno uchovávat.

1.3.1.1 Vrstvy

Základní strukturální jednotkou modelu jsou vrstvy. Každá z vrstev je buď nezávislá, nebo závislá právě na jedné vrstvě jiné než sama na sobě. Na jedné řídící vrstvě může být závislých více podřízených vrstev. Aby měla existence vrstvy smysl, obsahuje alespoň jednu komponentu. Každá vrstva má vlastní textový titulek a pořadové číslo jednoznačně určující pořadí vykreslování jejích komponent. To má za úkol zabránit situaci, kdy by některá komponenta mohla být nevhodně překryta jinou – například že by se klika vykreslila pod prosklení dveří. Dále toto číslo určuje pořadí vrstev v konfiguračním panelu modelu. Posledním atributem je jedinečné číslo, které vrstvu mezi ostatními jednoznačně identifikuje.

1.3.1.2 Komponenty

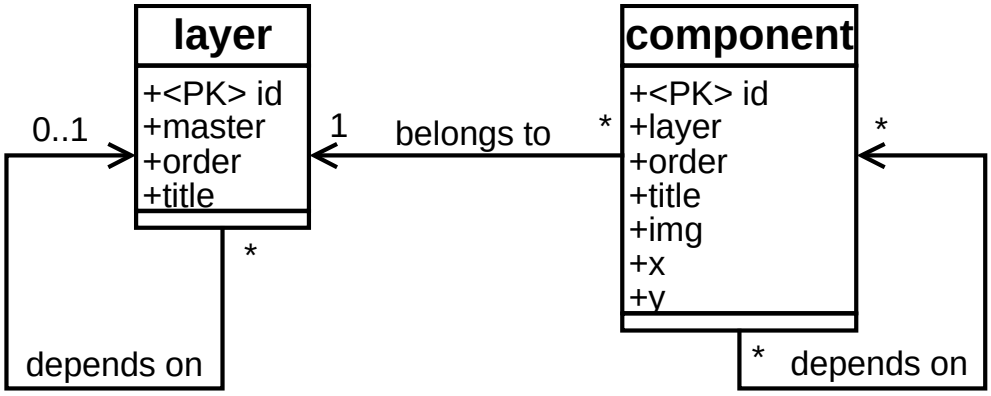
Komponent existují dva typy – vizuální a nevizuální. První typ má přiřazen obrázek, který se promítne do modelu, pokud je komponenta vybrána. Druhý pak žádný obrázek nemá a slouží pro omezení nabídky komponent v závislé vrstvě. Komponenty s definovaným obrázkem mají navíc ještě dvě souřadnice X a Y určující místo v modelu, kam se má komponenta při výběru umístit. Stejně jako u vrstev má každá definován textový popisek, který figuruje v seznamu komponent odpovídající vrstvy v konfiguračním panelu a pořadové číslo určující pořadí v tomto seznamu. Dále má každá komponenta přiřazené číslo právě jedné vrstvy, do které náleží a nakonec číslo komponentu mezi ostatními jednoznačně určující.

Jak už jsem uvedl dříve na příkladu velikost, někdy není žádoucí, aby šly vzájemně kombinovat všechny komponenty ve všech vrstvách. Proto je potřeba stanovit mezi některými komponentami závislosti tak, že po vybrání nějaké komponenty v řídící vrstvě je nabídka ve vrstvě podřízené omezena pouze na ty komponenty, pro které existuje pravidlo dovolující je s vybranou komponentou kombinovat. Aby měla definice závislosti smysl, musí být každá z komponent ve dvojici z jiné vrstvy. Dále platí, že jedna komponenta se může vystupovat ve více pravidlech ať už jako řídící, či jako podřízená.

1.3.2 Konceptuální model

Na základě shrnutí z předchozí podkapitoly jsem vytvořil ER diagram. Obsahuje pouze dvě základní entitní množiny – vrstvy a komponenty.

Názvy množin, atributů i vztahů jsem volil v anglickém jazyce, a to z důvodů přenositelnosti vývoje. To samé platí i pro názvy souborů a identifikátory ve zdrojových kódech. V budoucnu by totiž mohla nastat situace, že na dalším vývoji by pracoval tým s mezinárodní účastí nebo by kód chtěl prostě použít někdo ze zahraničí. V případě toho projektu není taková situace sice příliš pravděpodobná, ale jde o budování správného návyku.



Obrázek 6: konceptuální model dat konfigurátoru

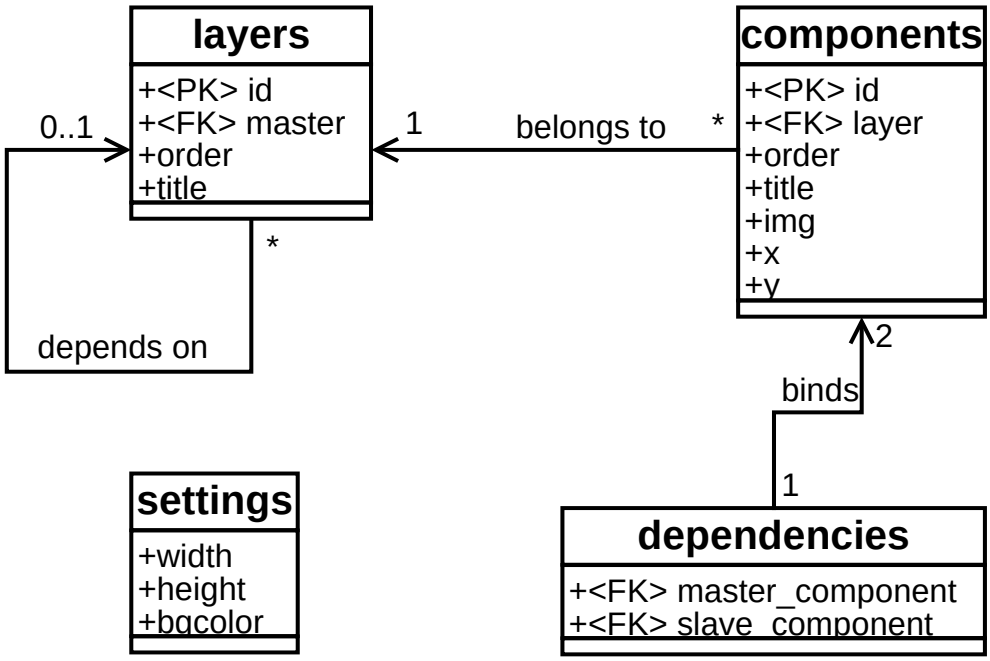
1.3.3 Volba datového úložiště

Dalším problémem, který bylo potřeba vyřešit, byla volba způsobu uložení dat. Původním záměrem bylo použít formát XML a samostatný diskový soubor pro každou z entitních množin. Pro tuto možnost hovořilo několik faktů: celková velikost dat nebude nijak závratná a data budou spíše statická, to znamená, že po prvotní konfiguraci sytému by se ze souborů převážně četlo, zápis by probíhal jen při změnách v modelu. Dále že technologie XML je pro uchovávání strukturovaných dat naprosto ideální, více viz kapitola o použitých technologiích. Podle mého mínění by byla akceptovatelná i skutečnost, že by teoreticky mohly nastat konflikty při zápisu do souborů, pokud by změny v konfiguraci prováděli dva správci současně. Vzhledem k povaze systému totiž vidím tuto situaci jako velmi málo pravděpodobnou.

Při návrhu editoru se ale ukázalo, že problém bude spíše se zápisem a opakovaným čtením, hlavně při tvorbě pravidel pro závislosti komponent. Jako alternativní možnost jsem uvažoval použití relační databáze a komunikaci prostřednictvím dotazovacího jazyka SQL. Zpočátku se mi zdálo toto řešení jako příliš robustní a pro řešený úkol poněkud těžkopádné, ale nakonec se ukázalo jako nejefetivnější.

1.3.4 Schéma databáze

Při transformaci konceptuálního modelu na schéma relační databáze bylo především potřeba vyřešit vztah M:N u závislostí komponent. To si vynutilo vznik samostatné tabulky závislostí, která má jen dva sloupce a sice čísla řídící a podřízené komponenty. Po jedné tabulce vzniklo z množin vrstev a komponent a navíc přibyla ještě tabulka *settings*, která slouží jen pro uložení obecné konfigurace modelu, konkrétně jeho rozměrů a barvy pozadí.



Obrázek 7: schéma relační databáze

1.4 Administrace

Administrační rozhraní musí správci umožnit založení nového projektu a editaci stávajícího. To znamená vytvoření a úpravy vrstev, nastavení jejich parametrů a vzájemných závislostí, vytvoření a úpravy nastavení komponent, jejich umístění do vrstev a nastavení možností jejich vzájemných kombinací a nakonec nastavení rozměrů modelu.

Původně jsem měl v plánu vytvořit editor jako klasickou off-line desktopovou aplikaci a data s databází synchronizovat přes XML export. Nakonec ale zvítězila varianta realizovat editor přímo v okně prohlížeče, aby správce mohl pracovat on-line a nemusel si na počítač instalovat žádný dodatečný software. Toto rozhodnutí sice znamenala jisté snížení komfortu při ovládání editoru, protože paleta nástrojů pro tvorbu uživatelského rozhraní v prohlížeči není tak bohatá, jako v současných grafických operačních systémech, ale zvýšená operativnost a jednoduchost to dostatečně vyvažuje.

2 Výběr technologií

Výběr technologií použitých při realizaci projektu byl nejvíce ovlivněn dvěma faktory: zaprvé to bylo samotné prostředí internetu, zadruhé pak požadavek na co největší univerzálnost a možnost vestavět knihovnu do už existujících informačních systémů.

Bylo potřeba vybrat vhodný prostředek v několika oblastech: prezentační a skriptovací jazyk pro klienta, skriptovací jazyk a databázový stroj pro server a vhodný formát pro komunikaci mezi oběma stranami.

Na základě svých dosavadních zkušeností s vývojem internetových stránek a informačních systémů a po zhlédnutí nabídky odborných knihkupectví na tato témata jsem vybral následující: XHTML a CSS, JavaScript, DOM a XML dohromady jako Ajax, PHP a MySQL. Všechny zmíněné mají veřejně dostupné standardy a specifikace a jsou k dispozici zdarma jak pro soukromé, tak pro komerční využití. Obě serverové technologie (PHP i MySQL) jsou primárně určeny pro Linuxové servery (typicky Apache), avšak úspěšně je lze provozovat i na platformě Microsoft Windows Server. Další výhodou je existence instalačních balíků „vše v jednom“ (například WAMP nebo EasyPHP) pro ladění a vývoj lokálně na stanici programátora s MS Windows, bez nutnosti pracovat se skutečným serverem. To snižuje časovou náročnost vývoje (není potřeba neustále kopírovat změněné zdrojové soubory na server) a zvyšuje bezpečnost serveru (není ohrožena jeho stabilita případnou závažnou chybou ve skriptu).

2.1 XML

XML (eXtensible Markup Language) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C [3]. Je určen především pro výměnu dat mezi aplikacemi a pro univerzální publikování dokumentů. Jazyk umožňuje popsat strukturu dokumentu z hlediska věcného obsahu, aniž by se zabýval jeho vzhledem. XML tedy samo o sobě žádné prostředky pro definici vzhledu nenabízí, nemá ani žádné předdefinované značky. Proto pokud potřebujeme data zobrazit uživateli, musíme použít některý ze stylových jazyků. Mezi nejznámější patří CSS, které lze s výhodou použít pro formátování dokumentů na obrazovce. Pokročilejší možnosti nabízí rodina jazyků XSL (eXtensible Stylesheet Language) případně XSLT (eXtensible Stylesheet Language Transformations). Ta umožňuje dokument různě upravovat a transformovat – vybírat části dokumentu nebo generovat obsahy a rejstříky. Specifikace XML 1.0 byla vydána v roce 1998, poslední platná verze 1.1 je roku 2006. Další odkazy k tématu viz [4].

Syntaxe XML je oproti HTML přísnější. Každá startovací značka musí mít odpovídající ukončovací značku, elementy mohou být vnořeny, ale nesmí se křížit – to znamená, že každý element musí být celý obsažen v jiném elementu. V dokumentu musí existovat právě jeden element

nejvyšší úrovně – takzvaně kořenový (*root*). Hodnoty atributů musí být uzavřeny v jednoduchých či dvojitých uvozovkách, opět je nelze křížit, ale opačný pár uvozovek může být použit uvnitř hodnot. Pokud dokument všechna tato pravidla splňuje, můžeme jej označit „*well-formed*“, neboli „dobře strukturovaný“. Jména elementů v XML rozlišují malá a velká písmena.

XML používám pro přenos dat mezi rozhraním editoru a serverovými skripty provádějícími operace nad databází.

2.2 XHTML

Jde o zkratku z anglického Extensible Hypertext Markup Language [5]. Jeho specifikace (verze 1) vzešla v roce 2000 z dílny konsorcia W3C jako nástupce zastarávajícího jazyka HTML 4 (z roku 1999), který reviduje a uvádí do souladu s principy dokumentů XML.

V projektu je tato technologie použita jako základní prezentační nástroj pro zobrazení nabídek komponent a hlavně pro realizaci grafického uživatelského rozhraní editoru. Zde byla volba jednoduchá, protože žádný konkurenční nástroj v této oblasti neexistuje. Šlo jen o výběr mezi tradičním HTML či novějším a perspektivním XHTML. XHTML si navíc striktně vynucuje oddělení dat strukturovaných pomocí sémantických značek od jejich vizuálního formátování, čímž přispívá k přehlednosti a čitelnosti zdrojového kódu.

2.3 CSS

Cascading Style Sheets, česky kaskádové styly, jsou dalším ze standardů prosazovaných W3C konsorciem. Smyslem této technologie je umožnit návrhářům oddělit vzhled HTML a XHTML dokumentu od jeho struktury a obsahu. V důsledku konkurenčního boje výrobců webových prohlížečů a tlaku ze strany uživatelů se do jazyka HTML dostalo spousta nesémantických elementů, které nepopisují obsah, ale způsob jeho zobrazení. Navíc vznikaly problémy s interpretací těchto značek právě mezi prohlížeči různých výrobců, protože oficiální a veřejně dostupná dokumentace neexistovala. První koncept „Cascading HTML Style Sheets“ byl zveřejněn v roce 1994 a první doporučení W3C (CSS Level 1) o dva roky později. Poslední oficiální verze z roku 1998 nese označení CSS Level 2, ale bohužel ještě ani dnes prohlížeče nepodporují všechny její vlastnosti. Specifikace viz web W3C [6] nebo česky v prameni [7].

Kromě již zmíněného oddělení vzhledu a obsahu CSS používám pro pozicování prvků v editoru, konkrétně jednotlivých komponent v náhledu modelu. Co se týče výběru mezi alternativami, byla situace stejná jako v případě XHTML, protože konkurenční technologie neexistuje.

2.4 JavaScript a DOM

Jedná se o multiplatformní objektově orientovaný jazyk interpretovaný na straně klienta. Byl vyvinut v roce 1995 společností Netscape jako nástupce podobného jazyka ECMAScript. Jeho návrh byl ovlivněn mnoha jazyky, ale s Javou má, i přes svůj název, podobnou syntaxi a onu podporu objektového programování. Samozřejmě umožňuje klasický procedurální a dokonce i funkcionální přístup k vytváření programů. Dnes ve vývoji jazyka pokračuje společnost Mozilla Foundation [8].

Mezi jeho nevýhody patří závislost chování programu na prohlížeči, přesněji na jeho implementaci interpretu, možnost uživatele zcela jej zakázat a žádná podpora přístupu k souborovému systému na klientovi (z důvodů zvýšení bezpečnost a zachování soukromí uživatele). I přesto však nemá ve své kategorii vážného konkurenta. Snaha Microsoftu prosadit v prostředí webu svůj vlastní VBScript (vycházející syntaxí z Visual Basicu) díky jeho funkčnosti pouze v prohlížeči Internet Explorer neuspěla a tento jazyk už není dále vyvíjen.

DOM (Document Object Model) je objektově orientovaná, platformně a jazykově nezávislá reprezentace XML nebo HTML dokumentu. Je to rozhraní umožňující přístup k obsahu, struktuře i styl stylu dokumentu. Historicky měl totiž každý webový prohlížeč své vlastní rozhraní k manipulaci s HTML elementy pomocí JavaScriptu. Standard vznikl pod křídly W3C opět jako reakce na tuto vzájemnou nekompatibilitu. Velmi obsáhle o problematice pojednává [9].

JavaScript ve spojení s DOM a CSS používám hlavně v editoru pro pozicování komponent a ve spojeni s XML pro tvorbu asynchronních dotazů a zpracování odpovědí od serverových skriptů pracujících s databází, více v následující kapitol.

2.5 Ajax

Asynchronous JavaScript and XML je označení pro technologii tvorby aktivních webových aplikací, kde stránky mění svůj obsah aniž by se musely znovu načítat ze serveru. Na rozdíl od klasických webových aplikací poskytují uživatelsky příjemnější prostředí, vyžadují ale použití moderních webových prohlížečů. Jak už název napovídá, Ajax ve skutečnosti není nová jednotlivá technologie, ale pojem označující použití několika prostředků dohromady.

Typicky se jedná o tyto:

- (X)HTML a CSS pro prezentaci informací
- JavaScript a DOM pro zpracování informací a dynamické změny obsahu stránky
- JavaScriptový objekt *XMLHttpRequest* pro asynchronní výměnu dat s webovým serverem, nejčastěji ve formát XML, je však možné použít libovolný jiný
- PHP jako serverový jazyk zpracovávající požadavky a vytvářející odpovědi pro klienta

Principy, na kterých je AJAX založen, nejsou novinkou: počátky lze vidět v zavedení elementu `IFRAME` v Microsoft Internet Exploreru 3.0 (rok 1996) nebo v Macromedia (dnes Adobe) Flashi (od verze 4), který umožňoval komunikaci se serverem na pozadí, bez překreslení stránky. Dále v roce 1998 představil Microsoft technologii nazvanou Remote Scripting, kdy na klientovi běžel Java applet komunikující se serverem a poskytující služby JavaScriptovým funkcím. Nakonec v páté verzi IE zavedl Microsoft objekt *XMLHttpRequest* fungující dodnes. Velkou popularitu a rozšíření AJAXu zapříčinilo jeho nasazení v aplikacích společnosti Google. Podrobněji o historii pojednává [10].

Mezi výhody patří odstranění nutnosti znovunačtení a překreslení celé stránky při každém dotazu na server, které jsou nutné u klasického modelu klient-server. Uživatel tak má pocit větší plynulosti práce, podobně jako u běžným desktopových aplikací. To s sebou nese i snížení zátěže na přenosovou síť a webové servery. Protože není potřeba při každém požadavku znovu sestavit celý HTML dokument, je množství přenášených dat výrazně nižší. Na druhou stranu AJAX může zvýšit počet vysílaných HTTP požadavků. Problémem může být také latence fyzického připojení: nízká odezva má negativní dopady na plynulost obnovování uživatelského rozhraní. Pokud uživatel nedostane jasně najevo, že aplikace jeho požadavky zpracovává, zaregistruje jen zpomalené reakce systému a může se snažit operaci spustit znovu pod dojme její předchozí ignorace. Pro uživatele může být na druhou stranu matoucí i to, že webové stránky se chovají jako plnohodnotné aplikace se složitou vnitřní logikou, nikoli jako posloupnost stránek, mezi kterými lze procházet pomocí tlačítek Zpět a Další. Praktické ukázky využití jsem čerpal z pramene [11].

Jak už bylo naznačeno výše, na asynchronní komunikaci je založen celý editor. Událostem jednotlivých formulářových prvků je přiřazeno volání JavaScriptových funkcí, které na pozadí volají serverové PHP skripty, přijímají od nich XML odpovědi a přes DOM je na stránce zpátky zobrazují.

2.6 PHP

PHP (původní význam Personal Home Page, později rekurzivní PHP: Hypertext Preprocessor) je skriptovací programovací jazyk určený především pro programování dynamických internetových stránek. Narozdíl od JavaScriptu je program vykonáván na serveru a klientovi se posílá pouze výsledek činnosti, nejčastěji ve formě HTML stánky či XML dokumentu, což je výhodné právě pro tvorbu webových aplikací. Lze v něm ale psát i konzolové či okenní aplikace.

Jazyk vznikl v roce 1994 jako soubor skriptů pro osobní potřebu autora Rasmuse Lerdorfa. Sada se postupně rozšiřovala a používalo ji stále více lidí, až v 1998 roce došlo k přepsání a zefektivnění parseru, změně názvu do současné podoby a uvolnění PHP3. Proces rozšiřování funkcionality a tvorba nových knihoven stále pokračuje. V době psaní práce aktuální verze 5.2 (z roku 2006) nabízí i podporu objektového programování. Vývojem jazyka se dále zabývá společnost The PHP Group. Mezi programátorské zajímavosti jazyka patří kombinovaná syntaxe

vycházející z C a Pascalu, dynamické typování proměnných a existence heterogenních a asociativních polí. Jazyk je platformně nezávislý a case-sensitivní. Více z historie nabízí web [12].

To, jak PHP postupně rostlo a nabalovalo na sebe další funkce a konstrukce, s sebou nezbytně nese i jisté komplikace. Je to třeba nekonzistentnost v tvorbě názvů funkcí, občas podivné a především rozdílné chování funkcí napříč verzemi. V porovnání s interpretovanými jazyky jako Java nebo .NET architektura jsou to například funkce negenerují výjimky při chybách. Některá další omezení rozebírá článek [13].

Možných alternativ pro PHP bylo více: JSP (Java Server Pages) od SUNu, ASP (Active Server Pages) či .NET rodina od Microsoftu nebo Python či Perl. I když mnohdy jsou tyto jazyky modernější a robustnější, prakticky všechny diskvalifikoval požadavek na univerzálnost řešení a možnost knihovnu jednoduše používat s již provozovanými systémy, právě kvůli rozšířenosti PHP.

V PHP je napsána prakticky celá výkonná část knihovny i editoru – komunikace s databází, sestavování nabídek dostupných komponent i sestavení výsledného obrázku modelu. Tuto část jsem řešil s využitím knihovny GD2, viz web [14]. Skriptování v PHP je od základů probíráno v [15].

Zkratka původně stála za slovy „GIF Draw“, jak se však nabídka funkcí rozšiřovala, změnil se i název na „Graphics Draw“. Knihovna je napsaná v jazyce C, vyvíjena jako opensource projekt a dostupná pro použití s mnoha programovacími jazyky. Nabídkou velkého množství funkcí programátorovi umožňuje za běhu vytvářet bitmapovou grafiku různých formátů (typicky JPEG, GIF nebo PNG). Mimo jiné dovoluje kombinovat už hotové bitmapy, měnit jejich rozměry, kreslit grafická primitiva nebo vkládat formátovaný text.

2.7 MySQL

MySQL je databázový systém vyvíjený od roku 1995 švédskou firmou MySQL AB [16]. Databáze byla od počátku optimalizována především na rychlost, takže uživatelé museli oželet některé vlastnosti známé z jiných systémů: pohledy, triggery nebo uložené procedury. Tyto „vymoženosti“ jsou doplňovány teprve v posledních letech. I přesto je MySQL ve spojení s PHP velice oblíbená jako základ pro tvorbu informačních systémů a dynamických webů. Tento stav je jistě podpořen snadnou implementovatelností (není platformně závislá), vysokým výkonem a faktem, že se jedná o volně šiřitelný software. Komunikace s databází probíhá pomocí dialektu jazyka SQL s některými rozšířeními, přístup k ní je možný prakticky z všech nejrozšířenějších programovacích jazyků. Zdarma dostupné je též komplexní administrační rozhraní phpMyAdmin. O práci s ním pojednává svazek [17].

Databáze v tom projektu slouží pro uchovávání údajů o obecném nastavení modelu, vrstvách, komponentách a jejich vzájemných závislostech. Klientská část z databáze jen čte, editor provádí čtení i úpravu, vkládání a rušení hodnot a položek.

3 Popis implementace

Při implementaci systému jsem vycházel z poznatků a návrhů z první kapitoly s využitím technologií z kapitoly druhé. Snažil jsem se využít co nejnovější verze a specifikace, ale na druhé straně takové, aby byly dostatečně rozšířené a stabilní. Verzi XHTML jsem zvolil 1.0 Strict a CSS verzi 2. Jako vývojový server jsem použil již dříve zmíněnou lokální instalaci WAMP [18] verze 2.0b obsahující webový server Apache verze 2.2.8, PHP v5.2.5 a databázi MySQL v5.0.51a. Dále jsem systém úspěšně testoval na komerčním webhostingu běžícím na OS Debian Linux s Apache v2.0.54, PHP v5.2.2 a MySQL v4.1.

Systém se z hlediska použití dá rozdělit na dvě části – uživatelskou a administrační. Ta první se integruje do stávajícího webu výrobce a stará o zobrazování nabídek komponent a obrázku modelu. Druhá část slouží k správě vrstev, komponent a jejich vzájemných vazeb.

Fyzicky je projekt rozdělen do několika souborů různých typů tak, že seskupují kód spolu nějakým způsobem související a spolupracující. Konkrétní názvy a popis obsahu uvedu v následujících podkapitolách. Oběma částem je společný soubor `mysql.php`, který vytváří spojení s MySQL databází a obsahuje některé programové konstanty (budou popsány dále). Je nutné v něm vyplnit jméno databáze, IP adresu nebo DNS jméno databázového serveru a uživatelské jméno a heslo pro přístup k němu. Aby nedošlo ke konfliktům s tabulkami, které by v databázi již mohly existovat, používá u těch svých systém předponu *cpl_*.

3.1 Uživatelská část

Popis tohoto celku jsem dále rozdělil na dvě části: konfigurační panel s nabídkami komponent a obrázek výsledného modelu.

3.1.1 Konfigurační panel

Konfigurační panel je implementován v souboru `compiclib.php`, který musí správce do stránky, na které chce funkce využívat, vložit. Název *compiclib* vychází z anglických slov *COMponent PICture LIBrary*, která jsem zvolil jako vystihující podstatu knihovny.

Pro tvorbu nabídek komponent jsem vytvořil dvě funkce: `insertConfigPanelList()` a `insertConfigPanelSelect()`. První z nich vkládá na místo volání HTML kód reprezentující výběrovou nabídku ve formě seznamu. Má dva volitelné parametry: *titletag* a *liststyle*. *Titletag* určuje HTML element, do něhož bude uzavřen nadpis skupiny komponent – tedy název vrstvy. Není-li tento parametr explicitně určen, použije se výchozí hodnota `<div>` (parametr se ale zadává bez špičatých závorek, ty se doplňují automaticky). Druhý parametr rozlišuje, zda bude

vložený seznam číslovaný nebo nečíslovaný. Očekávána je jedna z konstant `ORDERED` nebo `UNORDERED` definovaných ve společném souboru `mysql.php` (ten ale není potřeba vkládat, o to se stará přímo `compiclib.php`). Pokud parametr chybí, bude vložen seznam nečíslovaný (`UNORDERED`).

Názvy komponent jsou vloženy jako odkazy přidávající k adrese platné stránky parametr *action=cplset* jako identifikaci žádosti o provedení změny a identifikátory vybrané komponenty (*cplc=X*) a vrstvy (*cpll=Y*), do které náleží. Tyto údaje jsou „odchytávány“ ve vloženém `compiclib.php` a je podle nich upravována aktuální konfigurace modelu (identifikační čísla vybraných komponent), která se ukládá do *session*. Tento způsob jsem zvolil jako řešení překonávající bezestavovost protokolu HTTP (HyperText Transfer Protocol), kterým jsou stránky z webového serveru nahrávány. Pokud bych konfiguraci ukládal v běžných proměnných uvnitř skriptu, došlo by k její vynulování při každém načtení stránky, takže změny voleb by se opticky vůbec neprojevovaly.

Druhá z funkcí vkládá nabídku ve formě formulářových výběrových boxů. Každý box má svůj vlastní formulář, který navíc obsahuje skryté položky (`<input type=“hidden“>`) místo parametrů *action* a *cpll* v URL odkazů v seznamech z předchozí funkce. Hodnota parametru *cplc* se získává z výběrového boxu. Funkci je možné předat až tři nepovinné parametry – první z nich, stejně jako u funkce předešlé, určuje, jakou HTML entitou bude obalen titulek vrstvy (výchozí hodnota je opět `<div>`). Druhý parametr ovlivňuje chování formulářů, přesněji způsob odesílání jejich hodnot na server. Pokud má parametr hodnotu 1 (výchozí nastavení), je u každého výběrového boxu zobrazeno tlačítko a hodnoty se zpracovávají až po jeho stisku. Toto chování zaručuje 100% kompatibilitu a funkčnost napříč různými prohlížeči, na druhou stranu ale uživatele může zdržovat. Proto jsem připravil další variantu pro hodnotu parametru 0. Ta způsob, že tlačítka se nezobrazí a hodnoty budou odesílány automaticky ihned po změně výběru. Daní za toto pohodlí je závislost na JavaScriptu se všemi z toho pramenícími omezeními (viz první kapitola a specifikace požadavků). Rozhodnutí, jak se formuláře bodu chovat a zda akceptuje naznačená rizika tedy závisí na správci webu. Posledním parametrem je textový řetězec, který se v případě zobrazení tlačítek použije jako jejich popisek. Jako výchozí hodnotu jsem určil „*nastavit*“.

Obě tyto funkce jen transformují údaje o dostupných komponentách do vizuální formy, která je vhodná pro zobrazení uživateli. Proto obě volají funkci `prepareActualData()`, která načte aktuální konfiguraci modelu ze session a závislosti a údaje o komponentách z databáze a tyto informace uloží do globální proměnné `$actualdata`, se kterou funkce zobrazující seznamy dále pracují.

Obrázek 8: konfigurační panel ve formě odkazů a výběrových boxů

Abych správci usnadnil naformátování panelů přesně podle jeho potřeb za pomoci CSS, nastavil jsem třídu entit obalujících titulky vrstev na „*layer*“ (`class="layer"`) a třídu seznamu, respektive formuláře na „*components*“. Za využití kaskády a dědičnosti tak může správce jednoznačně měnit styl veškerých prvků uvnitř panelů.

3.1.2 Obrázek modelu

Obrázek celého modelu se z dílčích obrázků komponent složí už na serveru a klientovi se pošle jen jeden výsledný soubor. Obrázek se zobrazí vložením elementu `<img>` s atributem `src` odkazujícím na soubor `image.php`, v XHTML dokumentu například takto:
`<img src="compiclib/images.php" alt="model" />`. Použití PHP skriptu jako parametru obrázku je sice netradiční, ale funkční. Server totiž klientovi pošle hlavičku s informací, že následně přijatá data má interpretovat jako obrazová.

Skript přijímá i dva volitelné parametry, první *type* určuje typ posílaného obrázku a může nabývat hodnot *jpeg* nebo *png*. Pokud parametr není uveden nebo je jeho hodnota odlišná od těchto uvedených, použije se výchozí hodnota *jpeg*. Druhým parametrem je *qual* určující kvalitu, respektive

míru komprese obrázku. V obou případech se jedná o celé kladné číslo a jeho význam i obor hodnot se liší podle typu obrázku. Pro *jpeg* musí být z intervalu <0;100> a určuje kvalitu ztrátově komprimovaného formátu. Platí, že čím větší číslo, tím lepší výstup a větší datová velikost. Pro *png* má být hodnota z intervalu <0;9> a určuje míru komprese bezeztrátového formátu, tedy neovlivňuje kvalitu, ale objem přenášených dat a čas potřebný na jejich rozbalení, jinými slovy na sestavení do původní formy. Čím vyšší číslo, tím lepší komprese, menší datová velikost a větší časová náročnost zobrazení. Není-li parametr uveden nebo jeho hodnota nespadá do uvedených mezí, je jako výchozí nastavení použito hodnot konstant `JPEGQUAL` a `PNGQUAL` ze souboru `mysql.php`.

Formát obrázku (*type*)	*jpeg*			*png*		
Kvalita (*qual)*	100	80	10	0	5	9
Datová velikost (v kB)	44	12,3	4,4	330	58	55

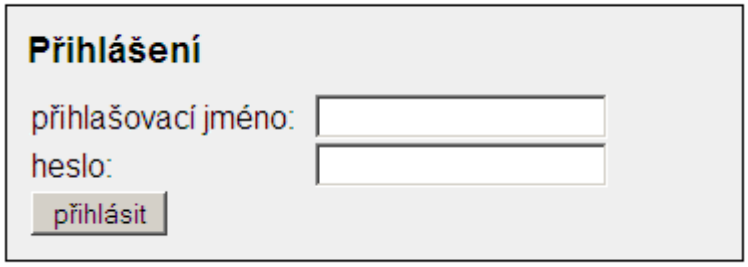
Tabulka 1: ukázka velikostí testovacího obrázku v závislosti na parametrech

3.2 Administrační část

Administrační rozhraní je založeno na technologii AJAX využívající jak skriptování u klienta (JavaScript), tak i na serveru (PHP). Výměna dat mezi stranami se děje za použití formátu XML.

Závislost na JavaScriptu u editoru, na rozdíl od veřejné části, nevnímám jako problém. S editorem bude pracovat pouze jeden správce, který si navíc může vybrat, na jakém zařízení. Na rozdíl od zákazníků, kterých je obvykle o několik řádů více a své technické vybavení si leckdy vybrat nemohou, nebo ani neumí.

Aby mohl správce se systémem pracovat, musí na přihlašovací stránce (realizované souborem `index.php`) nejdříve zadat uživatelské jméno a heslo. Jméno i heslo jsou nastaveny jako konstanty `LOGIN` a `PASSWORD` v souboru `mysql.php` a správce je může libovolně měnit. Existenci pouze jednoho uživatele jsem shledal jako dostatečnou vzhledem k povaze systému – účet slouží jen k editaci modelu, rozlišování oprávnění uživatelů ani sledování jejich akcí není zapotřebí.

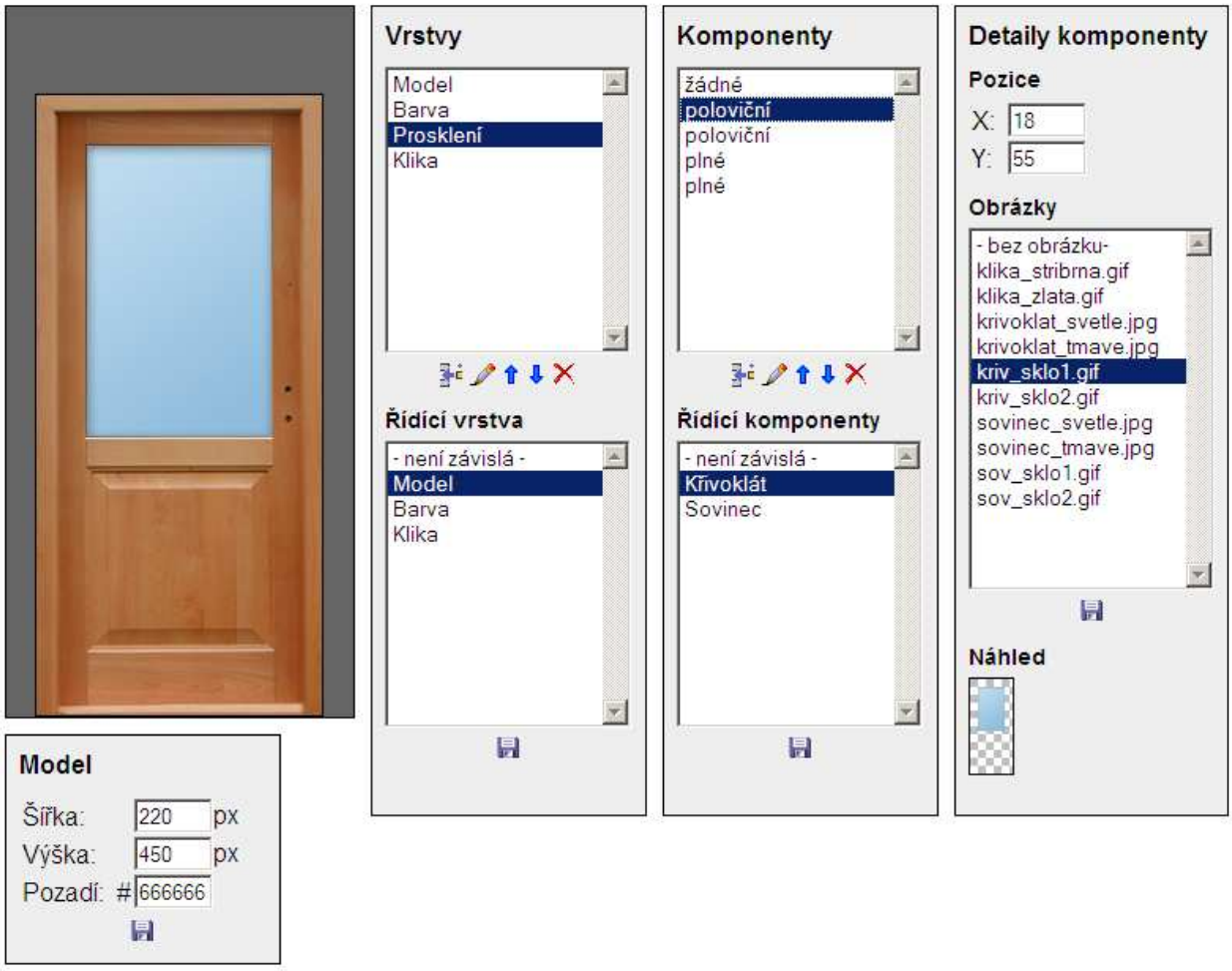


Obrázek 9: přihlašovací obrazovka

Po úspěšném přihlášení je tato skutečnost spolu s časem přihlášení uložena do session. Při pokusu provést v systému jakoukoliv další činnost je v session nejdříve zkontrolován příznak přihlášení a následně čas uplynuvší od poslední akce nebo od přihlášení. Pokud příznak přihlášení chybí, nebo čas od poslední akce je větší než konstanta `VALIDITY` (opět v souboru `mysql.php`), je uživatel přesměrován na přihlašovací stránku, kde je mu sdělen důvod a umožněno opětovné přihlášení. Čas poslední akce se aktualizuje vždy po načtení jakékoliv stránky v zabezpečené sekci. O bezpečné odhlášení se stará skript `logout.php`, který veškeré údaje o přihlášení ze session odstraní.

3.2.1 Rozhraní editoru

Uživatelské rozhraní editoru je tvořeno skripty `editor.php` (na který je uživatel přesměrován po úspěšném přihlášení) a `editor.js` a stylovým předpisem `editor.css`. Plocha editoru je



rozdělena na oblast zobrazení modelu a čtyři konfigurační panely.

Obrázek 10: uživatelské rozhraní editoru

Oblast modelu i konfigurační panely jsou pomocí CSS nastylovány jako plovoucí boxy (pravidlo `float: left`), takže se jejich rozmístění v okně prohlížeče přizpůsobuje jejich obsahu i rozlišení monitoru správce. Z pohledu tvorby rozhraní můžeme mluvit o událostmi řízeném programování. Aby došlo k vykonání nějaké akce, musí uživatel aktivovat patřičný ovládací prvek a tím vytvořit událost s odpovídající odezvou. Využité události jsou kliknutí na grafická tlačítka (ikony) v konfiguračních panelech a změny aktivní položky ve výběrových boxech tamtéž.

3.2.1.1 Oblast modelu

Na rozdíl od modelu zobrazovaného na klientovi pomocí `image.php` je model v editoru tvořen obrázky několika – každé vrstvě v modelu odpovídá jeden. Obrázky se přes DOM vytvoří v době inicializace editoru ihned po načtení stránky a nastaví se jim unikátní ID, obsah a pozice vždy podle první komponenty v odpovídající vrstvě. Pořadí vytváření obrázků odpovídá parametru *order* u vrstev, takže je zachováno překrývání tak, jak jej uvidí uživatel. Pro umisťování obrázků na odpovídající pozice využívám absolutní pozicování (CSS pravidlo `position: absolute`) a nastavování souřadnic vzhledem k levému hornímu okraji plochy modelu (CSS vlastnosti `top` a `left`).

3.2.1.2 Panel vrstev

Prvním ovládacím prvkem v tomto konfiguračním panelu je seznam (výběrový box `<select>`) všech dostupných vrstev. Zobrazuje jejich titulky v pořadí podle atributu *order*. Pod boxem je několik grafických tlačítek realizovaných pomocí obrázků `<img>` uvnitř hypertextových odkazů `<a>`, které mají nastavené JavaScriptové události `onclick`.

Dostupné akce jsou vytvoření nové vrstvy, změna titulku stávající vrstvy, posun vrstvy v seznamu nahoru nebo dolů a smazání vrstvy. Při vytváření nové vrstvy je správce dotázán na její jméno a vrstva je vložena na konec seznamu a v souladu s tím je jí automaticky nastaven atribut *order*. Podobně při přejmenování se nejdříve objeví dotaz na nové jméno, které se následně uloží do databáze a aktualizuje se obsah seznamu. Pokud správce na dotaz zadá prázdný řetězec, k přejmenování nedojde, každý titulek musí mít délku nejméně jeden znak. Přesouvání vrstev v seznamu nahoru nebo dolů funguje na principu výměny hodnot atributů *order* odpovídajících vrstev – pokud vrstvu posouvám nahoru, dochází k výměně mezi vrstvou vybranou a tou s nejmenším předchozím *order*, při přesunu dolů naopak s největším následujícím. Chce-li správce posunou první vrstvu nahoru nebo poslední dolů, nic se nestane. Poslední proveditelnou akcí je smazání vrstvy. Po kliknutí na tlačítko je správci zobrazen dotaz, zde se chce akci opravdu provést a v případě kladné odpovědi je vrstva včetně všech v ní obsažených komponent z databáze odstraněna.

Posledním ovládacím prvkem na tomto panelu je výběrový box pro nastavování závislosti vrstvy. První položka v seznamu má titulek „- není závislá -“ a po jejím vybrání a uložení pomocí jediného tlačítka pod výběrem dojde ke zrušení závislosti dané vrstvy, jinými slovy se vrstva stane nezávislou. Další položky jsou titulky vrstev kromě vrstvy vybrané v předchozím výběru – tím zabraňuji možnosti uložit nesmyslnou závislost vrstvy sama na sobě.

3.2.1.3 Panel komponent

Struktura a funkcionalita panelu pro konfiguraci komponent je velmi podobná jako u vrstev. Seznam komponent se aktualizuje vždy po změně výběru v seznamu vrstev. Při potvrzení pokusu o smazání nějaké vrstvy jsou z databáze odstraněny i veškeré závislosti ve kterých mazaná vrstva figuruje, jedno jestli na pozici řídící nebo podřízené vrstvy. Další výběrový box na panelu, seznam řídících komponent, jako první položku opět obsahuje „- není závislá -“, která způsobí, že veškeré závislosti, ve kterých komponenta figuruje jako podřízená, budou z databáze odstraněny. Jako další jsou v seznamu komponenty z vrstvy, která je pro vrstvu obsahující komponentu vybranou v prvním boxu nastavena jako řídící. V souladu s požadavky na závislosti stanovené v předchozích kapitolách je možné v seznamu řídících komponent vybrat více položek zaráz.

3.2.1.4 Panel detailů komponenty

Obsah toho panelu se od obou předešlých výrazně liší. Umožňuje nastavování všech parametrů u komponenty aktuálně vybrané v seznamu komponent na předchozím panelu. Při změně tohoto výběru se současně aktualizuje obsah polí v panelu detailů.

První dvě pole slouží k určení souřadnic, na které bude obrázek komponenty do modelu umístěn. Systém akceptuje i záporná čísla.

Dalším ovládacím prvkem je seznam obrázků dostupných v podadresáři `components` umístěném v adresáři se soubory editoru. Před započetím tvorby nového modelu je musí správce na server nahrát mocí FTP klienta. Akceptovány jsou soubory s příponami JPG, GIF a PNG. Pokud má komponenta jiný než pravoúhlý obrys totožný s rozměry jejího obrázku, musí mít nastavenou průhlednost, aby byl zachován efekt vrstvení komponent přes sebe. Průhlednost podporují formáty GIF a PNG, ale GIF má omezenou paletu na maximálně 256 současně zobrazených barev, proto se pro vkládání větších a vícebarevných komponent hodí spíše PNG. Formát obrázků komponent nemá vliv na formát obrázku modelu, který je odesílán klientovi, ten je ovlivňován pomocí volitelných parametrů, viz podkapitola o uživatelské části systému výše.

3.2.1.5 Panel nastavení modelu

Tento poslední a nejméně obsáhlý panel obsahuje pouze tři textová pole pro nastavení šířky, výšky a barvy pozadí modelu. Šířka a výška se zadává jako celé číslo a značí rozměry v pixelech. Barva pozadí je zadávána jako tři nebo šest hexadecimálních číslic pro barevné složky RGB.

3.2.2 Serverové skripty

Jde o PHP skripty, které vykonávají operace s databází a jsou asynchronně volány JavaScriptovými funkcemi ze souboru `editor.js`. Některé z nich (například dotazy na počty komponent nebo vrstev) vracejí výsledky ve formátu XML, které jsou funkcemi z `editor.js` zpracovány a zobrazeny správci, zatímco jiné (například smazání komponent) operaci pouze provedou a žádné hodnoty nevracejí.

Fyzicky jsou tyto funkce seskupeny do několika souborů: `editcomponent.php` obsahuje funkce pracující s komponentami, `editlayer.php` funkce pro manipulaci s vrstvami a poslední `editmisc.php` pro zbývající operace (práce s nastavením modelu, se závislostmi a další). To, jakou konkrétní akci má skript provést, se detekuje pomocí povinného parametru `action`, případně dalších podle povahy požadované operace.

3.2.2.1 Ukázka použití

Například po kliknutí na titulek vrstvy v konfiguračním panelu vrstev se aktivuje funkce `fillComponentList()`, která s potřebnými parametry asynchronně zavolá serverový skript `editcomponent.php?action=show&id=X`, kde `X` je identifikační číslo komponenty, na kterou uživatel kliknul. Ten provede dotaz typu `SELECT` nad tabulkou cpl_`components`, vrácené řádky transformuje do XML formátu a odešle zpět klientovi, kde si je převezme funkce `fillComponentListResponse()`. Ta XML data rozparsuje a vyplní je do výběrového boxu na konfiguračním panelu komponent.

4 Závěr

V této zprávě byly prozkoumány existující řešení konfigurátorů výrobků složených z oddělitelných komponent. Takto získané poznatky byly zobecněny a formalizovány, aby se mohly stát základem pro vývoj vlastního a především univerzálního systému pro nasazení v podobných situacích.

Při implementaci řešení byly vybrány takové technologie, aby systém nebyl závislý na žádném komerčním nebo proprietárním řešení jakékoliv firmy. Dalším požadavkem byla funkčnost systému na různých koncových zařízeních. Nedílnou součástí řešení byl i návrh a implementace editoru tak, aby v něm budoucí správce webu našel veškeré funkce potřebné při tvorbě a následné správě kompletního projektu.

Všechny body zadání jakožto i nároky, které jsem na řešení sám kladl, se mi podařilo splnit. Jako ukázku možností sytému jsem na přiloženém CD připravil demonstrační aplikaci pro výběr dveří a jejich doplňků. Při návrhu a implementaci jsem vycházel ze svých dosavadních zkušeností s tvorbou webových stránek a informačních systémů. Novinkou pro mě v této práci byla technologie AJAX, která přináší zajímavé možnosti pro tvorbu interaktivních uživatelských rozhraní.

Ale rčení „nic není tak dokonalé, aby to nemohlo být ještě lepší“ platí jistě i v případě tohoto projektu. Jako místo pro další zlepšování systému vidím tvorbu dalších typů uživatelských nabídek komponent, například ve formě ikon či zmenšenin vlastních komponent nebo umožnit vzájemnou kombinaci více typů – pro každou vrstvu moci použít jiný typ nabídky. Nadále rozvíjet by šlo i uživatelské rozhraní editoru, konkrétně proces přidávání nových komponent a určování jejich pozice v modelu, třeba za pomoci v okenních aplikacích oblíbené techniky *drag and drop*.

Literatura

[1] Webové stránky firmy Truhlářství Martin Čuda: http://www.cudadvere.cz/dvere/nabidka-dveri

[2] Webové stránky firmy ALCAR BOHEMIA, s. r. o.: http://www.alcar.cz/6588_CZ.0

[3] Webové stránky konsorcia W3C: http://www.w3.org/

[4] Wikipedie, otevřená encyklopedie: http://cs.wikipedia.org/wiki/Extensible_Markup_Language

[5] Specifikace XHTML 1.0: http://www.w3.org/TR/xhtml1/

[6] Přehled technologie CSS: http://www.w3.org/Style/CSS/

[7] Prokop M.: CSS Kaskádové styly pro webdesignéry. Mobil Media, a. s. 2003. ISBN 80-86593-35-5.

[8] Webové stránky Mozilla Developer Center: http://developer.mozilla.org/en/docs/JavaScript

[9] Flanagan, D. JavaScript - kompletní průvodce. Computer Press, Praha, 2002.

[10] Wikipedie, otevřená encyklopedie: http://cs.wikipedia.org/wiki/AJAX

[11] Darie C. aj.: AJAX a PHP tvoříme interaktivní webové aplikace PROFESIONÁLNĚ. Zoner Press 2006. ISBN 80-86815-47-1.

[12] Wikipedie, otevřená encyklopedie: http://cs.wikipedia.org/wiki/Php

[13] Snížek M.: K čemu je PHP. Článek dostupný na URL: http://www.snizekweb.cz/weblog/php-k-cemu/ (duben 2008). ISSN 1802-2103.

[14] Webové stránky knihovny LibGD: http://www.libgd.org/Main_Page

[15] Lane D., Williams H. E.: PHP a MySQL Vytváříme webové databázové aplikace. Computer Press 2002. ISBN 80-7226-760-4.

[16] Webové stránky databáze MySQL: http://mysql.com/

[17] DeLisle M.: phpMyAdmin efektivní správa MySQL. Zoner Press 2004. ISBN 80-86815-09-9.

[18] Webové stránky balíku WAMP: http://www.wampserver.com/en/

Seznam příloh

Příloha 1: CD se zdrojovými kódy, ukázkovou aplikací a technickou zprávou v elektronické podobě.