VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

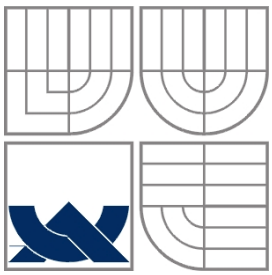
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MULTIPLATFORMNÍ PŘÍSTUP K DATABÁZOVÝM APLIKACÍM

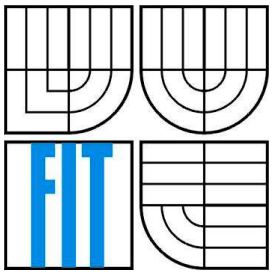
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PAVLA HROMÁDKOVÁ
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MULTIPLATFORMNÍ PŘÍSTUP K DATABÁZOVÝM APLIKACÍM

MULTIPLATFORM ACCESS FOR DATABASES APPLICATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVLA HROMÁDKOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JIŘÍ KOPECKÝ

BRNO 2008

Zadání

1. Prostudujte problematiku přístupu k databázovým aplikacím z různých zařízení (jako např. PC, mobilní telefon, PDA, …).
2. Seznamte se s požadavky kladenými na reálně použitelný systém pro cestovní kancelář Drak.
3. Navrhněte vhodnou architekturu, která bude umožňovat přístup z různých zařízení.
4. Proveďte detailní návrh aplikace umožňující intuitivní správu a ovládání systému včetně nadstandardních funkcí (systém obnovy dat, převod části dat do formátu vhodného pro program Money, …).
5. Při vývoji využijte modelovacích technik jazyka UML. Vzorový systém implementujte použitím programovacích prostředků XHTML, PHP5 a MySQL.
6. Vytvořte potřebnou podporu pro nasazení systému do reálného provozu včetně požadované migrace dat.
7. Zhodnoťte dosažené výsledky a možnosti dalšího vylepšení systému.

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato diplomová práce řeší možnosti přístupu k databázovým systémům z různých typů zařízení. Obsahuje analýzu, návrh a popis následné implementace informačního systému s využitím návrhového vzoru Model-View-Controller.

Návrh systému je proveden v modelovacím jazyce UML. Při vývoji systému byl použit programovací jazyk PHP a relační databázový server MySQL. Uživatelská vrstva byla implementována technologií HTML a WML. Výsledného vzhledu aplikace bylo dosaženo použitím webové technologie CSS.

Technická zpráva popisuje vícevrstvou architekturu a výhody jejího použití při tvorbě informačního systému. Zabývá se návrhovým vzorem MVC, jehož použití je vhodné u aplikací, kde je potřeba oddělit vzhled aplikace od komunikace s DB a logiky aplikace. Práce také popisuje analýzu, návrh i implementaci multiplatformní aplikace pro cestovní kancelář Drak. Přístup k systému je v současné době možný prostřednictvím mobilního telefonu nebo přes webový prohlížeč. Aplikace je snadno rozšiřitelná pro použití i na další platformě.

Klíčová slova

Informační systém, PHP, SQL, MySQL, HTML, XHTML, WML, CSS, JavaScript, UML, architektury informačních systémů, multiplatformní přístup, návrhové vzory, vzor Model-View-Controller, framework, Zend Framework, cestovní kancelář Drak

Abstract

This thesis deals with possibilities of accessing database systems from different devices. It contains analysis, concept and description of the following information system implementation with draft model Model-View-Controller.

Concept of the system is made in UML language. The PHP programming language and MySQL relational database server were used during the progression of the system. User interface was implemented by HTML and WML technologies. Final look of the application was achieved by using CSS web technology.

Technical report describes multilayer architecture and its advantages in the creation of the information system. It deals with MVC draft model, which is useful in the cases, where´s the need of the separation of the application look, its communication with database and application logic. The work oncemore describes analysis, concept and implementation of multiplatform application for the Drak travel agency. Systém could be currently accessed by handy or web browser. The application is easily extensible for the use in another platform.

Keywords

Information system, PHP, SQL, MySQL, HTML, XHTML, WML, CSS, JavaScript, UML, Information systeme architecture, multiplatform Access, draft models, model Model-View-Controller, framework, Zend Framework, travel agency

Citace

Hromádková Pavla: Multiplatformní přístup k databázovým aplikacím. Brno, 2008, diplomová práce, FIT VUT v Brně.

Multiplatformní přístup k databázovým aplikacím

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracovala samostatně pod vedením Ing. Jiřího Kopeckého.
Další informace mi poskytli PaedDr. Pavel Vaverka a Ing. Michael Kunc.
Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……
Pavla Hromádková
1. května 2008

Poděkování

Chtěla bych poděkovat panu Ing. Jiřímu Kopeckému, který mě během práce odborně vedl. Dále bych ráda poděkovala externímu zadavateli panu PaedDr. Pavlovi Vaverkovi, který mi poskytl potřebné informace o cestovní kanceláři Drak. V neposlední řadě patří moje poděkování i konzultantovi Ing. Michaelu Kuncovi.

Obsah

1 Úvod

Cílem této diplomové práce je vytvořit databázovou aplikaci, ke které bude možné přistupovat z počítače a mobilního telefonu. Aplikace bude snadno rozšiřitelná pro přístup i z dalších platforem, jako např. pda. Systém bude založen na architektonickém vzoru Model/View/Controller.

Toto téma jsem si zvolila z několika důvodů. Tím hlavním je skutečnost, že systém bude používán v praxi. Během studia jsme v rámci školních projektů vytvářeli různé aplikace, ale vždy se jednalo pouze o projekt, který nebude nasazen v běžném provozu. Zajímavá je i možnost prohloubení znalostí technologií, které budou využity při tvorbě systému. Největší přínos spatřuji v možnosti vyzkoušet si komunikaci se zákazníkem, zjišťování jeho prvotních požadavků a představ a jejich porovnání s výslednými požadavky na systém.

Diplomové práci předcházel semestrální projekt, během něhož jsem se seznámila s problematikou informačních systémů a přístupu k nim. Také jsem prostudovala požadavky, které na svůj budoucí systém cestovní kancelář Drak má. Teoreticky jsem se obeznámila s problematikou vícevrstvých architektur. Na základě získaných znalostí a možností daných technologií jsem zvolila jako nejvhodnější architekturu návrh založený na vzoru MVC. Vzor MVC odděluje reprezentaci informací od řadiče, který reaguje na události, a od prezentace dat, která vidí uživatel. Tím umožňuje přistupovat k aplikaci z různých zařízení použitím různých pohledů.

Kapitola dvě se zabývá cíli práce a obsahuje informace o cestovní kanceláři Drak. Na tuto kapitolu navazuje teoretická část práce, která se věnuje informačním systémům, vzoru Model/View/Controller, Zend Frameworku, UML, HTML, XHTML, PHP, MySQL, Java Scriptu, CSS a WML. Analýza budoucího systému se nachází ve čtvrté kapitole. Kapitola čtyři obsahuje i funkční a nefunkční požadavky, neformální specifikaci, prvotní analýzu požadavků, uživatele systému, diagram případů použití a specifikaci několika vybraných případů použití. Následující kapitola ve věnuje návrhu a charakterizuje vrstvy třívrstvé architektury. V šesté kapitole je popsána implementace systému, zpracování požadavků, vzhled stránek a nasazení systému v praxi. V závěru jsou shrnuty dosažené výsledky a přínosy projektu.

Při tvorbě práce jsem používala zdroje, které jsou uvedeny v seznamu použité literatury.

2 Cíle práce

V této kapitole popisuji cíle, kterých by mělo být dosaženo vypracováním diplomové práce. Vychází ze zadání práce.

V první podkapitole jsou rozebrány hlavní cíle práce. Jedna část je také věnována důvodům, které vedly ke vzniku projektu. Hlavním argumentem pro vývoj úplně nového systému byla skutečnost, že se mi nepodařilo najít žádný funkční systém, který by odpovídal požadavkům zákazníka a po mírných úpravách by jej bylo možné použít. Poslední kapitola seznámí čtenáře s cestovní kanceláří Drak.

2.1 Hlavní cíle práce

Cílem diplomové práce je seznámit se s postupy, které se používají při tvorbě systémů přístupných z více platforem, a následně tyto metody využít při implementaci aplikace. Výsledný systém by měl být nezávislý na zařízení, přes které bude k systému přistupováno. Podstatnou vlastností je i snadná ovladatelnost systému. V budoucnu s ním budou pracovat hlavně méně zkušení uživatelé. Proto je důležité, aby ovládání bylo snadné a intuitivní.

Dalším důležitým cílem je vyzkoušet si vytvoření systému, který bude použitelný v praxi. Doposud jsem pracovala na projektech určených pro akademickou půdu. V praxi ale projekty vznikají za trochu odlišných podmínek. Už samotné získávání požadavků na systém od zákazníka je zdlouhavý proces, kdy zákazník většinou ani sám přesně neví, co chce, nebo není schopný na začátku přesně formulovat všechny svoje požadavky.

Neopomenutelným cílem je i seznámení se s novými technologiemi a možnostmi při vytváření systémů. Tento poznatek je důležitý vzhledem ke skutečnosti, že po ukončení studia budu velmi pravděpodobně pracovat v této oblasti.

2.2 Důvody vzniku projektu

Téměř každá firma, škola či instituce potřebuje pro svoji činnost informační systém. V minulosti byly informace uchovávány v papírové podobě v kartotékách a různých šanonech. Vyhledávání a změna údajů byly práce na delší časové období. I přes např. abecední řazení záznamů trvalo určitou dobu, než byl záznam vyhledán. Většinou se taky informace vyskytovaly v duplicitním provedení a změna na jednom místě vyvolala nekonzistenci v datech. S nástupem informačních technologií se objevila snaha převést tyto papírové záznamy do elektronické podoby.

V dnešní době jsou již technologie finančně dosažitelné, a proto většina organizací přechází, resp. již v minulosti přešla, na elektronické informační systémy. Výběr takového systému závisel na

znalostech lidí, kteří o této změně rozhodovali. Někteří se vydali cestou vlastních systémů vyvíjených na klíč a někteří se rozhodli využívat již vytvořené a finančně dostupnější prostředky. U této druhé kategorie uživatelů ale většinou časem došlo ke zjištění, že používaný systém přímo nevyhovuje všem jejich potřebám. Měli možnost se s tím smířit a přizpůsobit se nebo mohli začít uvažovat o systému, vytvořeném přímo pro jejich potřeby. Pro takový systém „na míru“ se rozhodla i cestovní kancelář Drak, která doposud využívala produktů firmy Microsoft Office.

Jedním z hlavních důvodů pro změnu byla skutečnost, že používané programy neposkytují požadovanou funkčnost. Velké množství údajů se musí opakovaně zadávat na více místech a tím dochází ke zbytečné ztrátě času a k možnosti chyb v datech. Řešením je aplikace, která bude mít požadovanou funkčnost a data budou vkládána do systému jen jednou.

Cílem projektu je vytvořit systém, který by mohli používat zaměstnanci cestovní kanceláře Drak. Primárně je určen pro tuto CK, ale budou ho moci používat i jiné cestovní kanceláře, které mají specializaci na pobyty rodinného typu a dětské tábory. Cestovní kanceláře velmi často používají informační systémy upravené pro jejich specifické požadavky. Většina se jich ale zaměřuje na pobytové nebo poznávací zájezdy. CK Drak se však orientuje na pobyty pro rodiny s dětmi a na dětské tábory. Tomu odpovídají i specifické požadavky na informační systém. A tak použití některého z již vytvořených systémů by nutně obnášelo nemalé úpravy.

Neexistence vhodného již hotového systému vedla k rozhodnutí vyvinout úplně nový systém, který by byl vytvořen pro firmu Drak „přímo na tělo“. Poměrně významnou část pracovní doby tráví pan Vaverka[1] na cestách. Při nich vyřizuje velké množství telefonních hovorů a často potřebuje domluvené změny zanést do firemního systému. Nyní si může informace pouze zaznamenat na papír a po návratu na pracoviště je pak přenést do elektronické podoby do počítače. To je ale zdlouhavé a často také dojde ke ztrátě papírku, kam si poznamenal, co chtěl vlastně uložit. Proto jedním z požadavků je, aby byl systém přístupný z různých typů zařízení. A právě takovým systémem by měla být vytvořená aplikace.

2.3 Cestovní kancelář Drak

Celý systém je vyvíjen pro cestovní kancelář Drak. Historie firmy začíná již v roce 1991, kdy PaedDr. Pavel Vaverka zahájil podnikání v oboru pořádání dětských sportovních a rekreačních akcí, organizaci volnočasových aktivit dětí a mládeže. Nyní cestovní kancelář sídlí na ulici Božetěchova 36, Brno 612 00. IČO je 40973034.

Drak pořádá letní dětské tábory, tábory pro teenagery, tábory sportovní, jazykové, výtvarné, hudební a cyklistické. Dále má v nabídce i zimní tábory s výukou lyžování v Jeseníkách nebo

[1] PaedDr. Pavel Vaverka – majitel cestovní kanceláře Drak (viz. dále).

Beskydech. CK Drak vybudovala síť středisek na Českomoravské vrchovině – Křižanov, Meziříčko, Nesměř. V minulosti se dětských táborů zúčastnilo již více než 20 000 dětí.

Od roku 2000 pořádá CK Drak minitábory[1] pro rodiče s dětmi. V minulosti se jich zúčastnilo asi 3 000 účastníků. Další aktivitou cestovní kanceláře jsou víkendové pobyty. Tyto pobyty jsou různého typu. Hlavně pro maminky a babičky je určeno víkendové výtvarné tvoření, které nabízí možnost vyzkoušet si i netradiční techniky. Víkend pro tatínky s dětmi je určen spíše pro odvážnější muže a nabízí jedinečnou možnost posílení vztahu dítě – otec. Zároveň dává možnost maminkám si doma odpočinout a chvíli se věnovat jen sobě. Rodinný víkend s programem umožňuje strávit rodinám společný čas aktivně a v příjemném prostředí.

Cestovní kancelář Drak také zajišťuje v průběhu školního roku pobyty pro školy v přírodě, školní výlety pro základní a střední školy, pobyty pro zájmové skupiny dětí a mládeže. Za celou dobu existence táborů na nich pracovalo více než 1 000 osob (vedoucích a zdravotníků). Minitábory vedlo kolem 40-ti osob.

[1] Minitábor je pobyt pro rodiny s dětmi s bohatým programem. Je určen dětem v doprovodu obou rodičů, případně jen maminky, babičky,... Je to pobyt plný pohybu, dobré nálady a aktivního odpočinku. Program připravují pracovníci CK Drak.

3 Teoretická část

V první části se kapitola věnuje informačním systémům obecně. Představuje rozdělení systémů podle různých kritérií. Věnuje se architekturám, které se používají při vývoji systémů a obsahuje zmínku o návrhových vzorech a frameworku.

V další podkapitole je rozebírán vzor Model/View/Controller. Tento vzor je vhodné použít při vývoji multiplatformní aplikace. Podkapitola 3.3 se věnuje Zend Frameworku, který je frameworkem pro webové aplikace v PHP.

Podkapitoly 3.4 až 3.11 popisují technologie, které budou využity při implementaci diplomové práce. Jedná se o: UML, HTML, XHTML, PHP, MySQL, JavaScript, CSS a WML. Tyto technologie byly vybrány s ohledem na zadání práce.

3.1 Informační systém

Informační systém (IS) je systém, který slouží ke sběru, udržování a poskytování informací a dat[1]. Může mít různé podoby: od papírové kartotéky až po specializovaný informační systém vyvíjený přímo na míru. IS nemusí nutně být v automatizované podobě dostupný přes počítač nebo jiné zařízení. Hlavně v dřívější době byly tyto systémy převážně v papírové podobě.

3.1.1 Rozdělení systému

Systémy lze dělit následovně [23]:

- **uzavřené** x **otevřené** – podle iterace s okolím
- **deterministické** x **stochastické** – podle chování
- **statické** x **dynamické** – podle vnitřního stavu
- **spojité** x **diskrétní** – podle časových událostí

Informační systémy patří mezi otevřené - jsou provázány s okolím.

3.1.2 Architektura informačního systému

Volba architektury je velmi důležitá. Existuje více typů architektur. Prvním typem jsou centralizované systémy, které se vyznačují těsnou vazbou mezi aplikací a daty. Většinou se využívají u malých aplikací. Dvouvrstvá architektura je příkladem architektury klient/server. Jedna vrstva požaduje službu (klient) a druhá ji poskytuje (server). Tato architektura má problémy při větším počtu

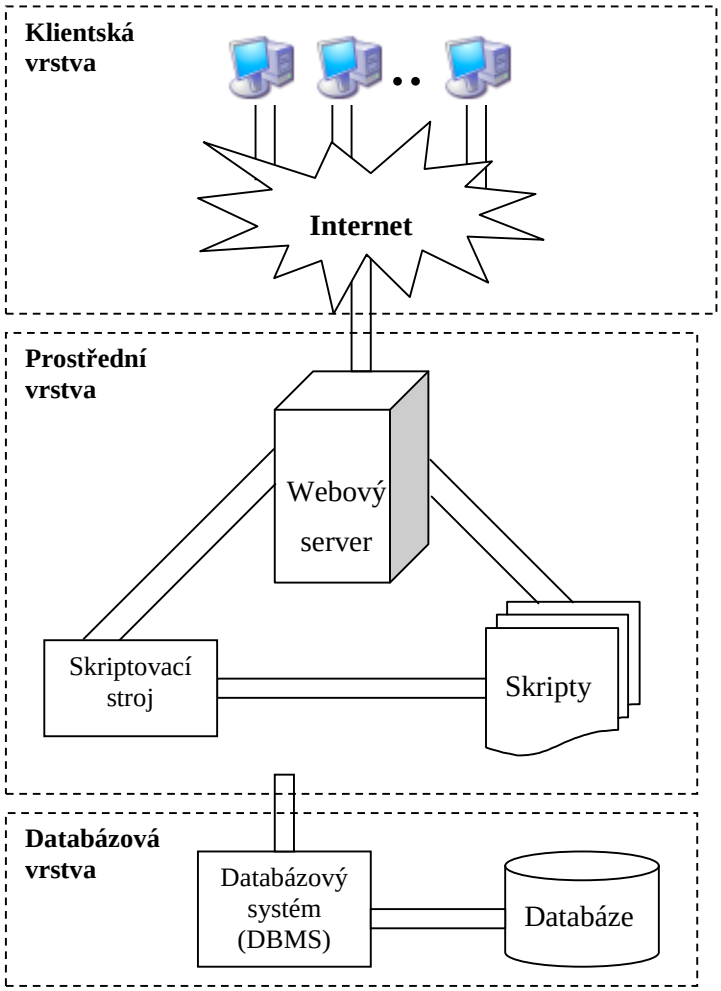
[1] Informace je sdělení, které odstraňuje nevědomost. Data jsou jakékoliv poznatky či fakta. Informace jsou data s významem.

klientů a nelze sdílet data mezi více aplikacemi. Dnes se téměř výhradně používá 3-vrstvá architektura, která je složena z:

- presentační vrstvy - komunikace s uživatelem
- funkční vrstvy - aplikace
- datové vrstvy - data

3.1.2.1 Třívrstvá architektura

Většina webových databázových aplikací používá tři vrstvy aplikační logiky. Ta je zobrazena na obrázku 3.1. Nejnižší vrstvu tvoří **databázový systém** (DNMS – Database Management System) a samotná databáze. DBMS je někdy také označován jako systém řízení báze dat – SŘBD. Tato vrstva zajišťuje správu databáze, vytváření, odstraňování a modifikaci dat i samotné dotazování ze strany uživatelů.



Obrázek 3.1: Třívrstvá architektura

Prostřední vrstvu tvoří vlastní aplikační logika, která je obvykle vytvořena pomocí nějakého skriptovacího jazyka. Ten dokáže komunikovat s příslušným databázovým systémem a následně vytvářet HTML kód. Nejvyšší vrstvu představuje klientský webový prohlížeč, který tvoří webové rozhraní aplikace a zajišťuje vlastní komunikaci uživatele s aplikací.

Klientskou vrstvu tvoří webový prohlížeč. Tento tenký klient zpracovává pouze malou část aplikační logiky. Jeho činnost spočívá v posílání požadavků HTTP na určité zdroje. Odpovědi, kterými jsou téměř vždy HTML dokumenty, zobrazuje. Tento model umožňuje, aby s aplikací pracoval libovolný uživatel, který má na počítači webový prohlížeč. Proto jsou aplikace přístupné velkému množství různých, i geograficky vzdálených, uživatelů. Klientská vrstva zobrazuje data uživateli a naopak od něj sbírá vstupní informace.

Téměř celou aplikační logiku skrývá **prostřední vrstva**. Řídí strukturu a obsah dat zobrazených uživateli, zpracovává vstupy, které uživatel zadá. Podle těchto vstupů provádí čtení a zápis dat v databázi. Vrstva se skládá z webového serveru, webového skriptovacího jazyka a stroje skriptového jazyka. Webový server má za úkol zpracovávat požadavky HTTP a formulovat odpovědi. Příklad webového serveru je Apache. Jako skriptovací jazyk lze použít například PHP. Webový server naslouchá požadavkům HTTP ze sítě, přijímá tyto požadavky, obsluhuje je a vrací HTTP odpověď s požadovanými zdroji informací.

Databázová vrstva ukládá a načítá data. Musí obstarat veškerou správu dat, jako např. paralelní přístup více procesů, zajišťovat integritu data, poskytovat potřebné zabezpečení, … Pro komunikaci s databázovým systémem se používá jazyk SQL. Pro správu dat lze použít databázový systém MySQL.

3.1.3 Návrhové vzory

Při návrhu softwaru často dochází k opakování částí zadání. Je důležité přemýšlet o znovupoužitelnosti kódu a efektivitě řešení. Proto může být vhodné použití obecného řešení problému. Takovému obecnému řešení se říká návrhový vzor. Nejedná se o nějakou knihovnu nebo část zdrojového kódu, který by se přímo vkládal do našeho programu. Nejsou to ani algoritmy, protože ty řeší pouze výpočetní problémy a nikoliv návrhové. Návrhový vzor je jen popis či šablona, která říká, jak lze problém řešit. Je to doporučený postup řešení často se vyskytujících úloh. Toto řešení pak může být použito v různých situacích. Je abstrahováno od konkrétního návrhu. Identifikuje třídy, zobrazuje jejich vzájemnou spolupráci a zodpovědnost jednotlivých tříd za určité činnosti. Použití návrhových vzorů také snižuje pravděpodobnost chyb, které by se mohly vytvořit v řešení, pokud by dané řešení vymýšlel sám vývojář. Vzory také počítají s budoucím rozšířením, a tak úpravy jsou méně náročné a výsledné programy jsou spolehlivější.

Základní rozdělení návrhových vzorů se poprvé objevuje v knížce Design Patterns - GoF[1], která vyšla v roce 1995 a stala se základní biblí návrhových vzorů. Tento katalog vzorů popisuje 23 základních, obecně použitelných, návrhových vzorů. Jsou rozděleny do tří kategorií – tvořící vzory, strukturální vzory a vzory chování.

Mezi **tvořící vzory** patří: Abstract Factory, Builder, Factory Metod, Prototype, Singleton

Strukturální vzory jsou: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy

Vzory chování jsou: Chain of Responsibility, Command, Interpreter, Iterator, Mediato, Memento, Observer, State, Strategy, Template Method, Visitor

Později vznikaly další návrhové vzory, které popisují efektivní řešení problémů ve specializovaných oblastech. Návrhové vzory jsou v katalogu zakresleny pomocí UML diagramu a jsou doplněny slovním popisem.

Nalezení vhodného vzoru není vždy jednoduché. To je způsobeno jednak tím, že je obtížné klást správné otázky během návrhu, a také znalostí vzorů.

3.1.4 Framework

Framework je softwarová struktura. Používá se jako podpora při programování a vývoji softwarových projektů. Obsahuje:

- podpůrné programy
- knihovnu API
- návrhové vzory
- doporučené postupy při vývoji

Snaží se převzít typické problémy z dané oblasti. Tím se usnadní práce návrhářům a vývojářům. Používá se ve větších projektech. Samotný framework se skládá z tzv. frozen spots a hot spots [26].

Frozen spots – definují celkovou architekturu softwarové struktury. Říkají, jaké jsou základní komponenty a jaké vztahy mají mezi sebou. Jsou to části, které se při žádném použití frameworku nemění.

Hot spots – mění se téměř při každém použití. Vytvářejí společně s kódem programátora specifickou funkcionalitu systému.

Frameworkem pro webové aplikace v PHP je Zend Framework (viz dále).

[1] Kniha Design Patterns s podtitulem Elements of Reusable Object-Oriented Software je napsaná čtveřicí autorů a za nedlouho dostala přezdívku Gang of four (banda čtyř) – zkratka GoF. Každá kniha či článek věnovaný návrhovým vzorům se odvolává na tuto publikaci.

3.2 Vzor Model/View/Controller

3.2.1 Úvod

Aby bylo možné přistupovat k systému z různých zařízení, je nutné oddělit část programu, která předzpracovává příkazy od uživatele (tj. zjištění, co má program udělat), od části zabezpečující logiku programu (tj. části, která zpracovává uživatelovy příkazy) a od části, která zobrazuje výsledek. A to právě umožňuje použití vzoru Model/View/Controller (Model/Pohled/Řadič).

Vzor rozděluje datový model aplikace, uživatelské rozhraní a řídící logiku do tří nezávislých komponent. Modifikace některé z nich nemá vliv na ostatní. Proto je tento vzor vhodný pro tvorbu multiplatformního systému. Model a řadič budou stejné, pohled se bude měnit podle zařízení.

3.2.2 Historie vzoru

Vzor byl poprvé popsán v roce 1979 Trygvem Reenskaugem. Poprvé byl použit v jazyce Smalltalk[1], vyvíjeném v Xerox research labs. Tato implementace byla inspirací pro řadu dalších projektů, které byly založeny na vzoru MVC. Dnes se MVC používá především jako architektura webových aplikací a přispívá k jejich flexibilitě, spolehlivosti a rychlému vývoji.

3.2.3 Popis vzoru

MVC (Model/View/Controller) je spíše architektonický vzor, který se používá při celkovém návrhu architektury programu. Využívá principy návrhových vzorů. Je složen ze tří druhů objektů.

Model představuje aplikační objekt. Zahrnuje celou aplikační logiku – popisuje stav aplikace a spolupracuje s daty.

View je vyjádření na obrazovce. Do této části patří nejenom vlastní GUI, ale i kód, který data pro zobrazení připravuje.

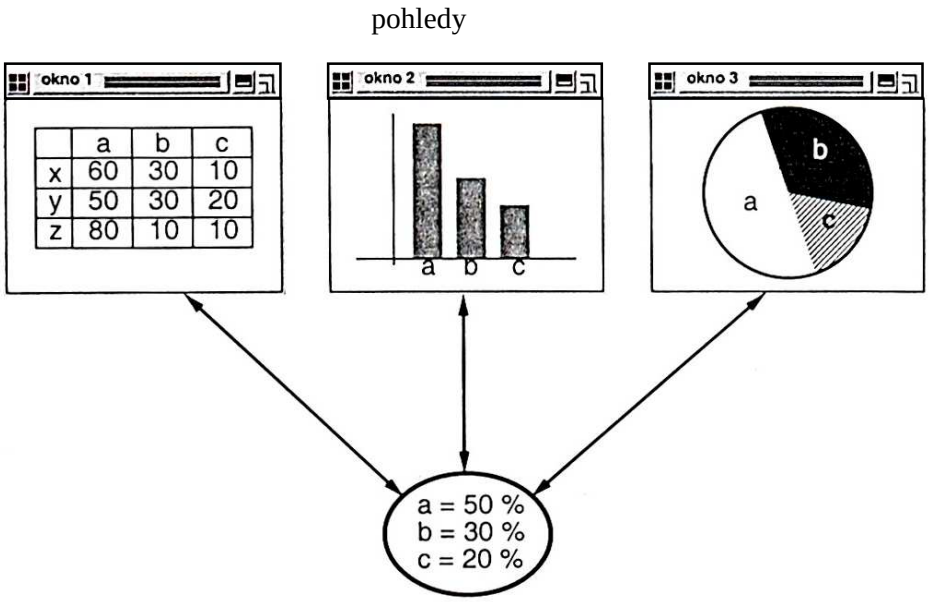
Controller definuje způsob, jak uživatelské rozhraní reaguje na uživatelský vstup.

Vzor jasně odděluje část programu starající se o vstup a předzpracování příkazů uživatele od částí reakcí a výstupu výsledků. Pokud se nepoužije tento vzor, je samozřejmě možné projekt vytvořit, ale všechny objekty budou sloučeny dohromady. To by nebylo příliš flexibilní, přehledné a ani znovupoužitelné. Využitím MVC dochází k oddělení a tím ke zvýšení tvárnosti a znovupoužitelnosti celého řešení. Aplikace bude reagovat nezávisle na tom, zda-li uživatelský vstup byl zadán z klávesnice, myši, přes hlasový vstup, …

[1] Smalltalk je objektově orientovaný programovací jazyk. Je interpretovaný virtuálním strojem. Je dynamický. První verze byla pojmenována Smalltalk-71. V dnešní době jsou známé implementace VisualWorks a VisualAge.

V tomto vzoru dochází k oddělení pohledu a modelu. Mezi těmito dvěma částmi je zřízena komunikace pomocí protokolu „přihlášení a upozorňování“ (subscibe / notify). Pohled zobrazuje stav modelu. V případě, že dojde ke změně dat v modelu, jsou modelem upozorněny všechny závislé pohledy (vzor Observer[1]). View si pak sám vyžádá od modelu potřebné informace a aktualizuje výstup. Příklad modelu a tří pohledů na daná data je na obrázku 3.2. Data, která jsou uložena v modelu, jsou zobrazena v tabulce, pomocí histogramu a výsečovým diagramem.

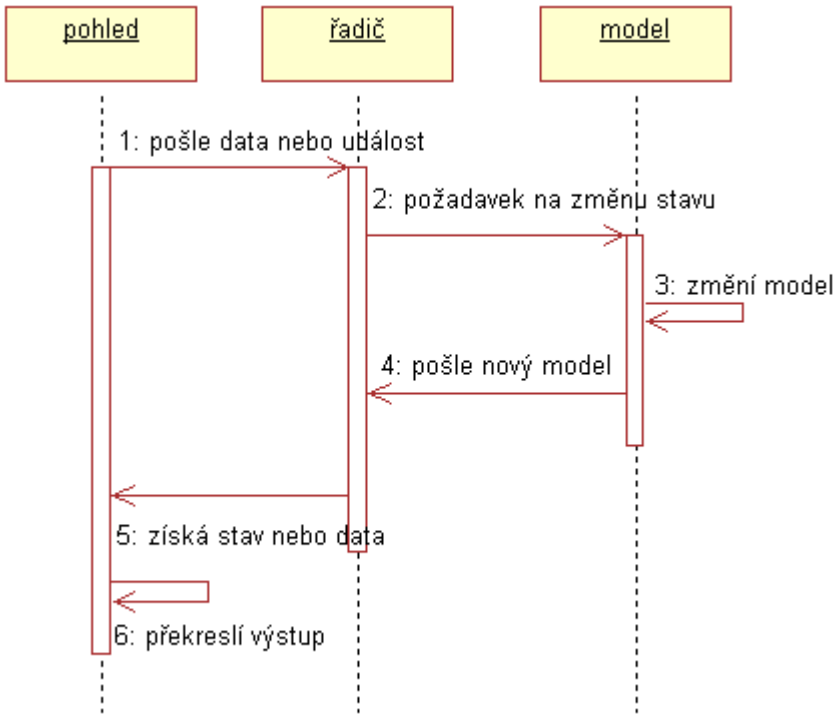
Jestliže se při tvorbě aplikací píše vše do jednoho souboru, je při jakékoliv změně (např. vzhledu) nutné tento soubor upravovat. U rozsáhlejších aplikací znamená tato úprava procházet desítky až stovky řádků kódu a hledat místo, kde je nutné provést změnu. Tento postup je zdlouhavý a nemá nic společného se znovupoužitelným kódem. Abychom se takovému zdlouhavému hledání v kódu vyhnuli, je vhodné použít obecně doporučené postupy – návrhové vzory.



Obrázek 3.2: Model a tři pohledy na data [15]

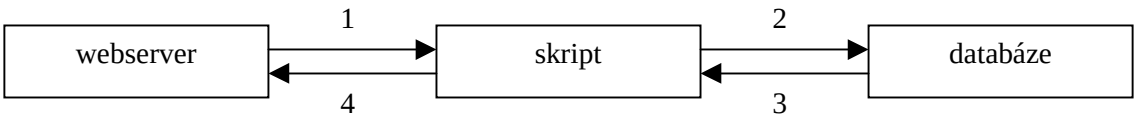
Zpracování uživatelského podnětu je na obrázku 3.3. O uživatelově akci je informován controller. Řadič příchozí událost zpracuje a pošle modelu. Ten provede aktualizaci a stav posílá zpět. Na základě těchto informací se pohled překreslí.

[1] Observer – jestliže je mezi objekty závislost, je nutné při změně objektu informovat ostatní o této změně. Nezávislý objekt šíří informaci ke všem závislým objektům – pozorovatelům, aniž by přesně věděl, kolik takových objektů je. Nezávislý objekt nemusí znát vnitřní strukturu pozorovatelů. Jestliže je pozorovatel informován o změně, musí si zjistit, co se v pozorovaném objektu změnilo, změny vyhodnotit a patřičně na ně reagovat. Počet pozorovatelů je možné dynamicky měnit.
Definice v GoF – Zavádí vztah 1:N mezi objekty. Když jeden objekt změní stav, všechny na něm závislé objekty jsou na tuto změnu upozorněny a automaticky se aktualizují.



Obrázek 3.3: Zpracování uživatelského vstupu

Obvyklá komunikace ve webových aplikacích probíhá mezi webserverem, skriptem a databází. Tato komunikace je zobrazena na obrázku 3.4. Komunikaci začíná webový server, který zvolí vhodný skript. Skript komunikuje s databází a webserveru vrací zpět HTML kód, který je potřeba zobrazit.



Obrázek 3.4: Komunikace webové aplikace [22]

U vzoru MVC obstarává činnost webserveru komponenta Controller. Za základě příchozích dat ve formě HTTP požadavku rozhodne, kam dále předá řízení.

Pohledy je v MVC možné dále vnořovat. Příkladem vnořeného pohledu může být ovládací panel s tlačítky – vnořeným pohledem jsou tlačítka. Vnořené pohledy podporuje pomocí podtřídy třídy View – třídy CompositeView. Objekty podtřídy CompositeView mají totožné chování jako objekty třídy View a lze je užít všude tam, kde lze použít pohledy třídy View.

U vzoru MVC lze změnit reakci na uživatelský vstup bez změny vizuálního vyjádření. Lze např. měnit způsob reakce na klávesnici. Odezva je zapouzdřena v objektu Controller. Pohled pak používá pouze jinou instanci podtřídy Controller k jinému typu odezvy. Změna je možná i za běhu programu.

MVC používá několik návrhových vzorů – např. Strategy[1] (vztah View - Controller), Factory Method [2](specifikace výchozí třídy kontroléru pro pohledy), Observer, Composite[3], …

V jednodušších aplikacích se někdy některé části slučují. Např. může dojít ke sloučení zobrazovací a ovládací části.

3.2.4 Princip činnosti MVC

Princip činnosti MVC lze rozdělit na několik kroků [25]:

1. První impuls pochází od uživatele – provede nějakou akci v uživatelském rozhraní – zašle požadavek. Takových akcí může být celá řada, jestliže se budeme specializovat pouze na webové aplikace, může to být např. stisk tlačítka.

2. Po akci uživatele řadič obdrží oznámení o této akci z objektu uživatelského rozhraní. Posoudí, které komponentě předá zpracování požadavku.

3. Komponenta zavolá model. Jestliže provedené uživatelské akce vyžadují aktualizaci modelu, zaktualizuje ho. Příkladem může být aktualizace obsahu nákupního košíku.

4. Model zpracuje změněná data – např. přepočítá celkovou cenu pro položky v košíku. Komponenta vrátí řadiči informaci o tom, jak změna modelu dopadla.

5. Řadič předá řízení vrstvě pohled. Ta na základě zaktualizovaného modelu přistoupí k zaktualizování dat, které se zobrazují uživateli – např. vypíše nový obsah košíku. Pohled získává data přímo z modelu, ale model nemusím mít žádné informace o pohledu. Model je zcela nezávislý na pohledu. Je ale samozřejmě možné použít návrhový vzor pozorovatel, který umožní modelu informovat všechny zaregistrované pohledy o svojí změně. V takovém případě model pošle pohledu příkaz, aby svůj obsah zaktualizoval – neposílá mu přímo doménové objekty.

6. Po změně uživatelského rozhraní se opět čeká na další akci uživatele, na zaslání dalšího požadavku. Pak se celý cyklus znovu opakuje.

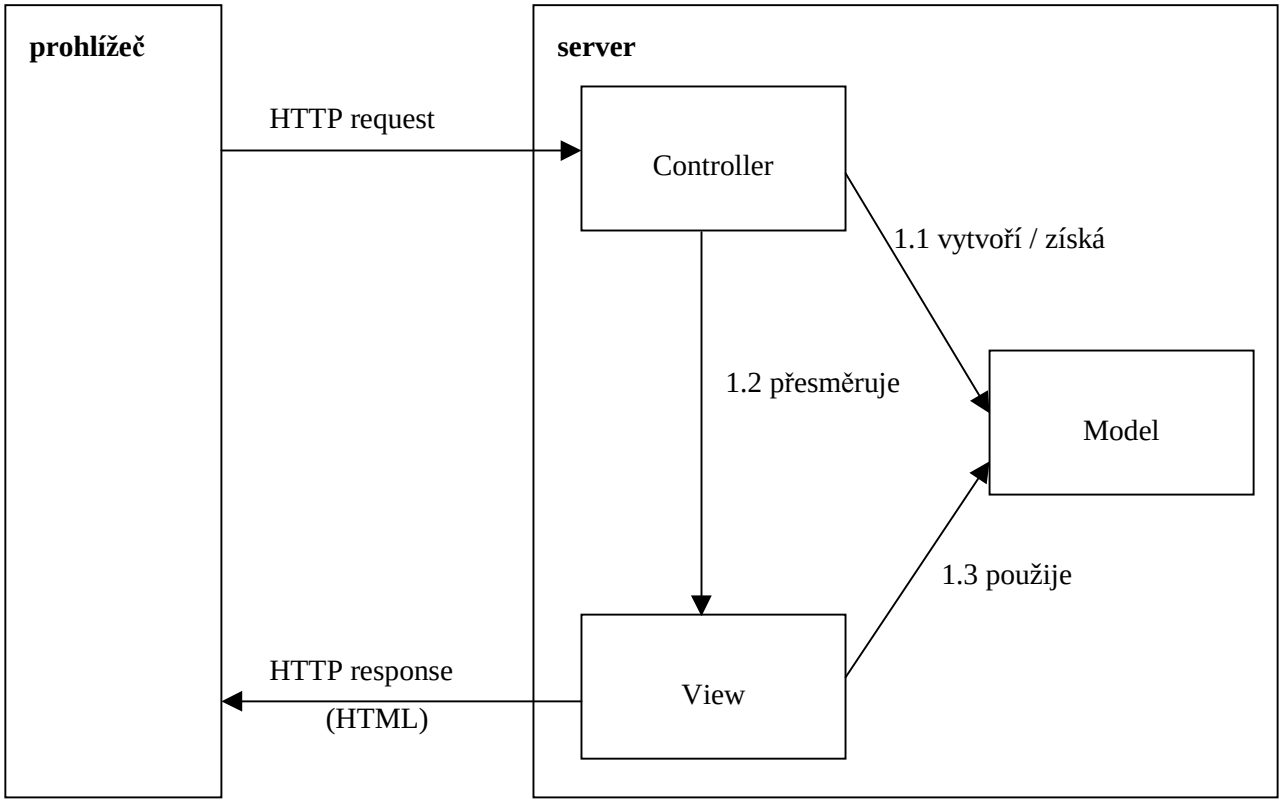
---

[1] Strategy – jestliže existuje více podobných řešení stejného problému, lze tyto algoritmy zapouzdřit do samostatného objektu a různě je zaměňovat. Sníží se tím množství podmíněných příkazů. Změny algoritmu probíhají nezávisle na klientovi.
Definice v GoF – Definuje rodinu algoritmů, zapouzdří každý z nich a nastaví je jako záměnné. Strategy umožňuje klientům zaměňovat používané algoritmy.

[2] Factory Metod – existuje několik tříd, které sice mají společného předka, ale nabízí rozdílné služby nad odlišnými daty. Třída pak nemůže předvídat třídu objektů, které má vytvářet (např. třída ví, kdy má být dokument vytvořen, ale neví, jakého má být druhu). Zodpovědnost převádí třídy na podtřídy – vytvářené objekty musí specifikovat podtřídy. Za běhu programu dojde k vytvoření instance některé z tříd.
Definice v GoF – Definuje rozhraní pro vytváření objektů, avšak rozhodnutí o tom, jakého typu bude vytvářená instance, nechává na podtřídách. Factory Method přenechává vytváření instancí podtřídám.

[3] Composite – snaží se ignorovat rozdíly mezi sestavami objektů a individuálními objekty, aby k nim šlo přistupovat jednotně. Objekty jsou uspořádávány do stromové hierarchické struktury. Listy stromu tvoří jednoduché objekty, větve jsou složené objekty. K listům i větvím lze přistupovat jednotným způsobem.

Každá část má svoji zodpovědnost a nedochází k vzájemnému míchání. Komponenty mezi sebou nemají těsnou vazbu. Možná realizace MVC je zobrazena na obrázku 3.5.



Obrázek 3.5: Realizace MVC [19]

Vzor MVC lze v prostředí webu implementovat několika způsoby. V dnešní době jsou nejpoužívanější dva hlavní přístupy, které odpovídají návrhovým vzorům a to:

- ***Front Controller***[1]
- ***Dispatcher view***[2]

Rozdíl mezi těmito dvěmi frameworky je z pohledu aplikace nad ním postavené v okamžiku volání aplikační logiky. U Front Controlleru dochází k volání aplikační logiky před předáním zpracování do pohledu. U Dispatcher view se aplikační logika volá zprostředkovaně uvnitř pohledu.

3.2.5 Výhody vzoru MVC

Použití MVC usnadňuje změnu kterékoliv ze tří částí. Lze např. změnit model a zachovat zbylé dvě složky. Nebo je možné definovat různé pohledy na stejná data (použitím vzoru Strategy) a

---

Definice v GoF – Skládá objekty do stromových struktur reprezentujících hierarchie typu část-celek. Composite umožňuje klientům pracovat s jednoduchými i složenými objekty jednotně.

[1] Front Controller řeší propojení mezi pohledem a řadičem pomocí jednoho objektu a to Front Controlleru. Jedná se o centrální objekt, který zpracovává všechny požadavky. Na jejichž základě pak rozhoduje, co bude zobrazeno. Také implementuje všechny bezpečností požadavky. Bližší informace jsou na [22].

[2] U Dispatcher View řadič předává řízení pohledu. Komunikace probíhá přes Bussines Helper. Bližší informace jsou na [23].

uživatel si může vybrat ten, který se mu zrovna nejvíce hodí. Při požadavku nového typu vstupu (např. ze souboru) lze opět pouze definovat nový vstupní modul a zbylé dvě části aplikace vůbec nezajímá, odkud pochází vstupní data.

Použití vzoru MVC usnadňuje přenositelnost mezi platformami. Jsou-li dobře oddělené jednotlivé složky aplikace, je celá aplikace velmi snadno přenositelná. Proto je tento vzor vhodný pro vytvoření mého zadání. Část Model a Controller zůstane stejná a použitím více typů View lze docílit toho, že bude aplikace spustitelná jak na počítači, tak i např. na mobilním telefonu.

3.3 Zend Framework

Zend Framework je open source[1] webový aplikační framework pro vývoj webových aplikací. Lze v něm vyvíjet aplikace založené na PHP 5. Vyžaduje minimální verzi PHP 5.1.4. Vyvíjí ho firma Zend. Poskytuje knihovny, které jsou navzájem provázané s ostatními komponentami frameworku. Zend Framework implementuje vzor Front Controller. Ve frameworku se využívá principu MVC.

Zend Framework přesně nenařizuje, jaká má být struktura aplikace. Protože je ale kód rozdělen do tří různých částí[2], je vhodné používat doporučenou strukturu adresářů [17]:

```
/application
        /controllers
        /models
        /views
                /filters
                /helpers
                /scripts
/public
        /images
        /scripts
        /styles
/library
```

Tato struktura aplikace není nutná, ale její užití je vhodné.

[1] Označení open source znamená, že software má otevřený zdrojový kód. Takový software je za dodržení určitých podmínek volně dostupný. Navíc je zdrojový kód veřejný, lze ho prohlížet i upravovat. Vývoj provádí dobrovolníci.
[2] Aplikaci tvoří tři části – model, řadič a pohled.

Adresář application obsahuje všechny tři části – model, řadič, pohled. Adresář public obsahuje kromě webu i bootstrap[1] index.php a soubor .htaccess. Soubor .htaccess obsahuje pravidlo, které přesměrovává všechny URL[2] požadavky na soubor index.php. Obsah .htaccess je ve výpisu 3.1:

```
RewriteEngine on
RewriteRule .* index.php
```

Výpis 3.1: Soubor .htaccess

Dále .htaccess může obsahovat nastavení bezpečnosti. V .htaccess se také nastavuje, že obrázky, skripty a CSS soubory se nesmí přesměrovat na bootstrap soubor.

3.3.1 Bootstrap

Zend Framework a jeho controller Zend_Controller podporuje hezká URL. Aby toho bylo dosaženo, musí všechny požadavky na aplikaci jít přes soubor index.php, který je v kořenovém adresáři aplikace. Tento soubor se označuje jako tzv. bootstrap soubor. Všechny požadavky jsou přesměrovány na tento soubor pomocí souboru .htaccess, který se nachází také v kořenovém adresáři aplikace. Kromě jiných nastavení obsahuje .htaccess pravidlo RewriteRule, které se stará o přesměrování URL na index.php.

Soubor index.php obsahuje několik nastavení. První nastavení je, že chceme být informováni o vzniklých chybách. Date_default_timezone_set nastavuje časové pásmo. Dále se nastavuje direktiva include_path a cesta k modelu – aby byly třídy snadno dostupné. Připojíme soubor Zend/Loader.php, který zpřístupní třídu obsahující statické funkce pro nahrání dalších Zend Framework tříd. Zend_Loader::loadClass nahraje zadanou třídu. Název třídy obsahuje cestu ke třídě, podtržítka jsou podadresáře a přidána je přípona .php (cesta ke třídě je tedy Zend/Controller/Front.php). V index.php také musíme nakonfigurovat, kde jsou umístěny naše controllers. Příklady uvedené v této kapitole jsou převzaty z [17].

3.3.2 Controller – nastavení

Controller je třída, jejíž název musí být následující: {jméno_Controlleru}Controller, kde {jméno_Controlleru} musí začínat velkým písmenem a ostatní písmena jsou malá. Controller se uloží do souboru pojmenovaného {jméno_Controlleru}.Controller.php do adresáře /application/controllers. Všechny action jsou public function a jsou uloženy v třídě controller. Název akce je: {jméno_action}Action. Jméno action je malými písmeny.

[1] Veškeré požadavky na aplikaci jdou přes soubor index.php. Přesměrování zajistí soubor .htaccess. - viz. dále
[2] URL (Unique Resource Locator) je jednoznačné určení zdroje. Lze pomocí něj jednoznačně zapsat umístění souboru na Internetu nebo intranetu. Definuje doménovou adresu serveru, umístění zdroje na serveru a protokol, kterým je možné zdroj zpřístupnit.

1.1.3 View – nastavení

Komponenta Zend Frameworku se jmenuje Zend_View. Komponenta umožňuje oddělit kód zobrazující stránku od kódu akcí.

Nastavení pohledu je možné vložit do každé action. Existuje ale i druhý, jednodušší způsob – inicializovat třídu view jinde a v action jen přistupovat na objekt třídy view. K inicializaci objektu view slouží v Zend Frameworku „action helper“. Zend_Controller_Action_Helper_ViewRenderer zajistí inicializaci objektu view($this->view), který též vytvoří view skript. Nastaví také objektu Zend_View cestu k view skriptu – views/scripts/{jméno_Controlleru}. Pohled se nachází v views/scripts/{jméno_Controlleru}/{jméno_action}.phtml. Obsah stránky je předáván do objektu Response, který kontroluje všechny HTTP hlavičky, obsahy body a výjimky vzniklé použitím MVC. Front Controller pak automaticky pošle hlavičku a obsah body na konci dispatchingu.

Po nastavení view je možné psát kód samotných actions. Dalším krokem je vytvoření view souborů pro aplikaci. Koncovka souborů je .phtml a jsou umístěny v podadresáři pojmenovaném podle controlleru. Výsledný vzhled pohledů je možné zlepšit použitím stylů.

3.3.4 Model – nastavení

Jestliže je nastaven Controller a View je možné přistoupit k nastavení modelu. K tomu v Zend Frameworku slouží třída Zend_Db_Table, která umí vyhledávat, vkládat, upravovat a mazat řádky tabulky databáze. Ve třídě Zend_Db_Table je potřeba nastavit typ databáze, uživatelské jméno a heslo. K tomu lze využít třídu ze Zend Framework – Zend_Config, která rychle objektově zpřístupní konfigurační soubory. Ty mohou být formátu INI nebo XML.

Zend_Config_Ini načte jednu sekci s názvem section (sekci [general]).Host, username, password a dbname bude sdruženo pod objekt `$config->db->config`.

Nejdříve jsou načteny potřebné třídy Zend_Registry a Zend_Config_Ini. Pro druhou jmenovanou je parametrem cesta k souboru config.ini a general, což je název section. Objekt $config přiřadíme objektu $registry, čímž zpřístupníme konfiguraci celé aplikaci.

Než použijeme Zend_Db_Table je nutné napřed předat načtené konfigurační údaje. To lze provést opět v boostrapu index.php.

Po nastavení databáze lze už vytvářet tabulky. Pokud není vytvořena databáze, před tvorbou tabulek je samozřejmě nutné databázi vytvořit. Do tabulek je možné vkládat data pomocí příkazů SQL. Dále rozšíříme třídu Zend_Db_Table, aby pracovala s aplikací vytvořením třídy a uložením do adresáře /application/models. Pak tuto třídu zpřístupníme při inicializaci controlleru ve funkci init(). Úpravou controllerů a pohledů se vytváří další funkčnost aplikace.

Zend Framework je možné stáhnout z adresy http://framework.zend.com/download. Aktuální verze je Zend Framework 1.0.3[1]. Velikost .zip archivu je 5.3 MB, velikost .tar.gz je 3.7 MB. Na stejném místě je dostupná i dokumentace v angličtině.

3.4 Použité jazyky

3.4.1 UML

UML (Unifed Modeling Language) je modelovací jazyk.

- UML je souhrnem především grafických notací k vyjádření analytických a návrhových modelů. Pomocí něj lze modelovat stejnou syntaxí jednoduché i složitější aplikace.
- modely jsou k ujasnění požadavků zadavatele aplikace a k upřesnění, zda návrhář správně pochopil, co zadavatel požaduje
- každý diagram se zaměřuje na jeden pohled vyvíjeného systému
- UML je jazyk s poloformální sémantickou specifikací, který zahrnuje abstraktní syntaxi, zformovaná pravidla a dynamickou sémantiku. Dokáže zachytit strukturu systémů pomocí diagramů, např. pomocí diagramu tříd, diagramů interakce objektů, stavových diagramů, sekvenčních diagramů, diagramů spolupráce, …
- tento jazyk se nesnaží nahradit textové definice, ale pouze zobrazuje celkovou strukturu systému v grafické podobě. Slouží k vizualizaci systému.
- je nezávislý na programovacím jazyce a vývojových procesech
- existují i jiné postupy, které vycházejí z UML, jako např. metodika RUP firmy Rational

3.4.2 HTML

HTML je značkovací jazyk. První verze HTML (HyperText Markup Language) vznikla v roce 1990. Od verze 2.0 se na vývoji HTML podílí firma Microsoft a Netscape.

- HTML je definován v rámci SGML[2]
- umožňuje vytvářet, formátovat a upravovat webové stránky. Lze pomocí něj upravovat text, vkládat obrázky v různých formátech, vytvářet formuláře, přes které je možné komunikovat s uživatelem.
- HTML je hypertextový jazyk, který umožňuje propojit velké množství oddělených informací

[1] Verze vydaná 30.11.2007.
[2] SGML – Standard Generalized Markup Language – je standard, který se používá k uveřejňování dokumentů. Ty obsahují text a multimediální prvky. Umožňuje definovat nové elementy a atributy, proto se používá jako systém pro tvorbu nových značkovacích jazyků.

3.4.3 XHTML

XHTML (eXtensible HyperText Markup Language) je rozšiřitelný značkovací jazyk. Je to XML[1], jehož definice typu dokumentu (DTD) obsahuje HTML.

- vznikl přeformulováním HTML 4 do aplikace XML. Lze pomocí něj vytvářet webové dokumenty, ale umí i definovat a používat vlastní formátovací značky.
- netoleruje chybně zapsaný kód. Stránky musí být napsány přesně podle definice jazyka. Např. formátovací značky a atributy musí být psány malými písmeny, značky se nesmí navzájem křížit, všechny značky musí být ukončeny, což platí i pro všechny nepárové značky. Ty se ukončují lomítkem např.
. Dále všechny hodnoty atributů musí být v uvozovkách, nelze používat minimalizované atributy.
- užití značek <head> a <body> je povinné. Atribut NAME je v XHTML nahrazen atributem id.
- povinná je také informace o typu dokumentu, vkládaná pomocí značky <DOCTYPE>

3.4.4 PHP

PHP je volně šířitelný software.

- PHP podporuje většina běžných operačních systémů, např. Windows, Unix, OS počítačů Macintosh
- PHP může pracovat s libovolným WWW-serverem, který umožňuje spouštění CGI-skriptů
- skripty v PHP jsou v podstatě HTML stránky, které jsou doplněny o příkazy
- velkou výhodou PHP je snadná spolupráce s databázemi. Pomocí příkazů jazyka SQL lze provádět libovolné úpravy dat.
- objektově orientovanou syntaxi obsahuje již PHP 3
- ve verzi PHP 5 už proměnné pracují s odkazy na objekty a ke kopírování dochází pouze na přání vývojáře
- od verze PHP 4 byla implementována podpora XML

3.4.5 MySQL

MySQL je strukturovaný dotazovací jazyk (SQL – Structured Query Language) sloužící pro správu relační databáze typu klient/server. Zahrnuje SQL server, klientské programy pro přístup k serveru, nástroje pro správu a programovací rozhranní.

- je Open Source, což znamená, že jej lze využívat zdarma
- správa MySQL je jednodušší než u velkých databázových systémů

[1] XML – Extensible Markup Language – je podmnožinou SGML. Umožňuje napsat vlastní definici dokumentu – DTD.

- používá dotazovací jazyk SQL
- zvládne současné připojení několika klientů a zároveň lze najednou přistupovat k několika databázím
- podporuje zašifrovaná připojení pomocí protokolu SSL (Secure Sockets Layer). MySQL může běžet na UNIXu, ale i na Windows a OS/2.
- typ architektury je klient/server
- od verze MySQL5 podporuje fiktivní tabulky vytvořené dotazem SQL – tzv. pohledu a uložené procedury - programy v SQL uložené v databázi
- v omezené formě také MySQL od verze 5.0 podporuje triggery - příkazy SQL, které se automaticky spustí po provedení určitých operací, jako například insert, delete, …

3.4.6 JavaScript

JavaScript je skriptovací jazyk, který se vkládá do webových stránek.

- kód je přímo vložen do webové stránky, prohlížeč jej zpracovává přímo se stránkou, tj. přímo u uživatele v prohlížeči
- JavaScript stále není plně kompatibilní ve všech běžně používaných prohlížečích
- nejčastěji se JavaScript užívá k oživení statických stránek
- lze jej využít k ověřování vstupních dat, zadávaných uživatelem
- významnou předností tohoto skriptovacího jazyka je skutečnost, že nezatěžuje server. Lze ho využít k ulehčení zaneprázdněnosti serverů.
- další využití JavaScriptu jsou cookies - malé soubory, které uchovávají informace o počítači uživatele

3.4.7 CSS

CSS (Cascading Style Sheets) slouží k formátování dokumentů. Nijak nemění obsah dokumentu.

- CSS definuje vzhled dokumentů a způsob jejich prezentace na koncovém zařízení
- určuje, jakou bude mít stránka podobu a jaký styl budou mít jednotlivé prvky
- CSS umožňuje vytvořit současně podobu i celé skupiny dokumentů, lze definovat různé styly pro různá zařízení (např. tiskárnu, monitor, čtecí zařízení, …)
- možnosti formátování CSS jsou mnohem větší, než formátování pomocí značek HTML

3.4.8 WML

Pro přístup na Internet z mobilního telefonu jsou používány služby založené na protokolu WAP[1] - Wireless Application Protocol. Některé mobilní telefony mají omezené zobrazovací možnosti - display je černobílý, má malou velikost, nízké rozlišení, … Pro dočasné uložení kódu právě zobrazované stránky mají mobily nízkou kapacitu paměti a také mají poměrně nízký výpočetní výkon a v neposlední řadě i nízkou přenosovou rychlost. Tato omezení jsou také důvodem k používání speciálního jazyka **WML** – Wireless Markup Language. Navíc prohlížeče, které jsou v mobilních telefonech, nejsou schopny zobrazovat HTML stránky, ale dokáží interpretovat jazyk WML. Tento jazyk je velmi podobný právě HTML. I přes tuto podobnost nelze HTML stránky v mobilním telefonu zobrazit.

- u WML, na rozdíl od HTML, je nutné dávat pozor na velikost písmen u názvu tagů. WML je velmi citlivý na velikost písmen.
- v kódu stránky je nutné zapisovat některé části vždy a občas i v daném pořadí. Základní struktura je dělená do dvou částí:
  - **deck** – celá stránka WML. Jeden deck je jeden WML soubor.
  - **card** – obsah, který uživatel vidí jako stránku. Každý deck se může skládat z více card. Při požadavku na stránku je stáhnutý do mobilního telefonu celý deck, ale zobrazit se může pouze jeho část – card. Výhodou je, že při přechodu na další card (navigace je většinou přes odkaz), který se součástí stejného decku, již není potřeba spojení a přenos dat ze serveru.
- při tvorbě stránek musí mít tvůrce na paměti, že velikost WML stránek je omezena maximální velikostí. Není možné vytvářet decky s obrovským množstvím card.
- deck musí obsahovat hlavičku. Je v každém souboru povinná a vždy musí být umístěna na začátku.
- jazyk WML je založen na XML.

[1] WAP využívá již existujících technologií a protokolů souvisejících s Internetem (např. využívá standardní webové servery určené pro provoz stránek HTML, protokol HTTP, …).

4 Analýza

Tato kapitola čtenáře seznámí se současným stavem systému cestovní kanceláře Drak, pro kterou je systém vyvíjen. Jednotlivé podkapitoly se věnují neformální specifikaci, prvotní analýze požadavků a uživatelům systému. Obsahuje diagram případů použití a specifikaci případů použití. Dále se zabývá současným stavem práce.

4.1 Současný stav systému CK Drak

V současné době používá cestovní kancelář Drak systém založený na MS Access. Pomocí něj provádí evidenci klientů, kteří jezdí na pobyty organizované touto cestovní kanceláří. Systém poskytuje osobní informace o účastnících táborů, minitáborů a vedoucích. Kromě osobních informací jsou u účastníků uloženy i akce, kterých se zúčastnili.

Systém není chráněný heslem a neposkytuje různé služby pro odlišné skupiny uživatelů. V době vzniku cestovní kanceláře vyhovoval aktuálním potřebám. S rozšířením poskytovaných služeb a zvětšením počtu klientů již nedostačuje stávajícím specifickým požadavkům.

Kromě systému v MS Access využívá Drak i program Money. Ten používá k fakturaci a vystavování smluv. Všechny osobní údaje do něj musí znovu ručně zadávat pracovníci CK. Při opakovaném vkládání stejných dat může snadněji dojít k chybě. Data mohou být nejen chybná, ale i neaktuální a to v případě, že se údaje změní pouze na jednom místě. Tato neaktuálnost je nežádoucí.

4.1.1 Potřeba vzniku systému

Vzhledem k rozšiřujícímu se zájmu o pořádané akce, se také začalo rozrůstat spektrum činností. Byla, a stále je, rozšiřována nabídka na období nejen letních a zimních prázdnin, ale i na dalších časová období, taktéž šíře programové nabídky se mění a rozvětvuje na věkově odlišné skupiny zájemců.

Používaný systém již zdaleka nevyhovuje potřebám cestovní kanceláře. Duplicitní zadávání údajů je časově velmi náročné. Protože je databáze uložena pouze na lokálním disku počítače, který není připojený do sítě, jsou data dostupná pouze z tohoto počítače. Tyto a další skutečnosti vedly k nápadu vytvořit systém, který by přesně splňoval specifické požadavky CK.

4.2 Neformální specifikace

Cílem je vytvořit komplexní systém pro cestovní kancelář Drak, která se specializuje na tábory pro děti a pobyty pro rodiny s dětmi s programem – tzv. minitábory. Aplikace bude umožňovat registraci a správu uživatelů, kteří s ní budou pracovat. Systém bude poskytovat informace o dětech, které jezdí

na tábory s touto cestovní kanceláří, a kontaktní údaje na jejich rodiče. Protože se CK Drak stále více specializuje na pobyty pro rodiny, musí systém uchovávat informace i o těchto zákaznících.

Dalším požadavkem je správa pobytových míst, na která klienti jezdí. Samozřejmostí je správa všech zaměstnanců, kteří pracují na táborech, ať už jako vedoucí, elévové[1] nebo členové vedení. Velmi často bude systém používán k vyhledávání dat a následnému tisku. Proto je nutné, aby obsahoval velké množství specializovaných tiskových sestav pro různé účely (např. zobrazení jednotlivých pobytů podle termínu nebo místa, seznam všech dětí daného turnusu řazený abecedně nebo podle roku narození, seznam účastníků minitábora podle ročníku narození nebo podle rodin, …). Každý systém, který pracuje s daty musí umožňovat zálohování dat. Ani tento systém nebude výjimkou. Důležitým požadavkem je vytvoření systému zálohování s možností obnovení dat. Aplikace musí umožňovat převod části dat do formátu vhodného pro program Money. Protože pracovníci, kteří budou se systémem pracovat, nejsou příliš zkušenými uživateli, je nutné, aby ovládání aplikace bylo snadné, intuitivní a uživatelsky přívětivé.

V současnosti v cestovní kanceláři Drak používají k uchovávání informací program MS Office Access. Součástí zavedení systému do provozu proto bude nejen potřebná podpora pro nasazení systému, která předpokládá instalaci všech programů, které jsou nutné k činnosti systému, ale také migrace stávajících dat. Systém by měl být také přístupný i z jiných zařízení než je počítač. Primární je přístup z mobilního telefonu, který bude využíván při služebních cestách.

1.3 Požadavky na systém

Po několika schůzkách z majitelem CK Drak a podrobném prostudování dostupných materiálů o činnosti organizace byly stanoveny základní požadavky na systém.

4.3.1 Nefunkční požadavky

Na základě analýzy požadavků plynoucích ze zadání a z cílů práce byly stanoveny následující nefunkční požadavky na systém:

- vstupy systému budou probíhat přes klávesnici a myš nebo klávesnici mobilního telefonu
- výstup bude na monitor počítače nebo displej mobilního telefonu
- aplikace bude stavová. Uživatel se k systému přihlásí. Bude moci spouštět několik vzájemně souvisejících transakcí a po dokončení práce se opět odhlásí.
- po odhlášení už s aplikací nebude možné pracovat
- aplikace bude ovládána prostřednictvím nabídek – menu
- předpokládaný počet registrovaných uživatelů nebude vyšší než 10

[1] Elévové jsou vedoucí, kteří nemají přímo přidělený oddíl a vypomáhají tam, kde je zrovna potřeba.

- systém bude oddělovat zobrazovací vrstvu od datové vrstvy a vrstvy aplikační logiky
- systém bude vytvořen programovacím jazykem PHP, relačním databázovým serverem MySQL, zobrazovací vrstva bude implementována pomocí jazyka HTML nebo WML, vzhled stránek bude implementován pomocí jazyka CSS
- na počítači bude systém funkční minimálně v prohlížečích Internet Explorer 6.0 a FireFox 2.0
- zobrazovací vrstva pro počítač bude optimalizována pro rozlišení 1024 x 768 pixelů

4.3.2 Funkční požadavky

Tato aplikace bude jediná v dané cestovní kanceláři. Proto musí poskytovat všechny potřebné funkce. Základem systému je uchovávání údajů o zákaznících, kteří využili, nebo v blízké budoucnosti využijí, služeb CK Drak. U každého klienta je nutná evidence určitých osobních údajů. Systém by měl také poskytovat přehled všech zaměstnanců. I tito lidé jsou nedílnou součástí cestovní kanceláře a v systému o nich musí být uloženy údaje. Na základě neformální specifikace a konzultace s vedoucím cestovní kanceláře Drak panem PaedDr. Pavlem Vaverkou byla vytvořena prvotní analýza požadavků. Systém bude poskytovat:

- přihlašování registrovaným uživatelům
- registraci nových uživatelů
- základní typy uživatelů - viz. kapitola 4.4
- navigaci pomocí textového menu
- evidenci účastníků dětských táborů a turnusů, kterých se zúčastnili
- evidenci účastníků minitáborů - dospělých i dětí a turnusů, kterých se zúčastnili
- přehled pracovníků táborů i minitáborů (zaměstnanců), jejich pozice a účast na jednotlivých turnusech
- tvorbu různých tiskových sestav - letní tábory: řazení podle abecedy nebo podle věku, rozdělení do oddílů, minitábory: řazení podle věku, po rodinách, podle turnusů
- export dat o zaměstnancích do formátu vhodného pro import do programu Money
- přístup z různých typů zařízení - z PC a mobilu

Toto je pouze prvotní analýza, která bude dále v průběhu práce upřesňována a doplňována podle požadavků cestovní kanceláře.

4.4 Uživatelé systému

Systém bude poskytovat informace různým skupinám uživatelů. Každá skupina bude mít ale odlišná práva podle toho, co může se systémem provádět. Všichni uživatelé budou k systému přistupovat přes webové rozhraní. Před používáním systému se budou muset uživatelé přihlásit. Přístup ke všem datům systému bude mít administrátor. Ten se bude o celý systém starat. Anonymní uživatelé nemají

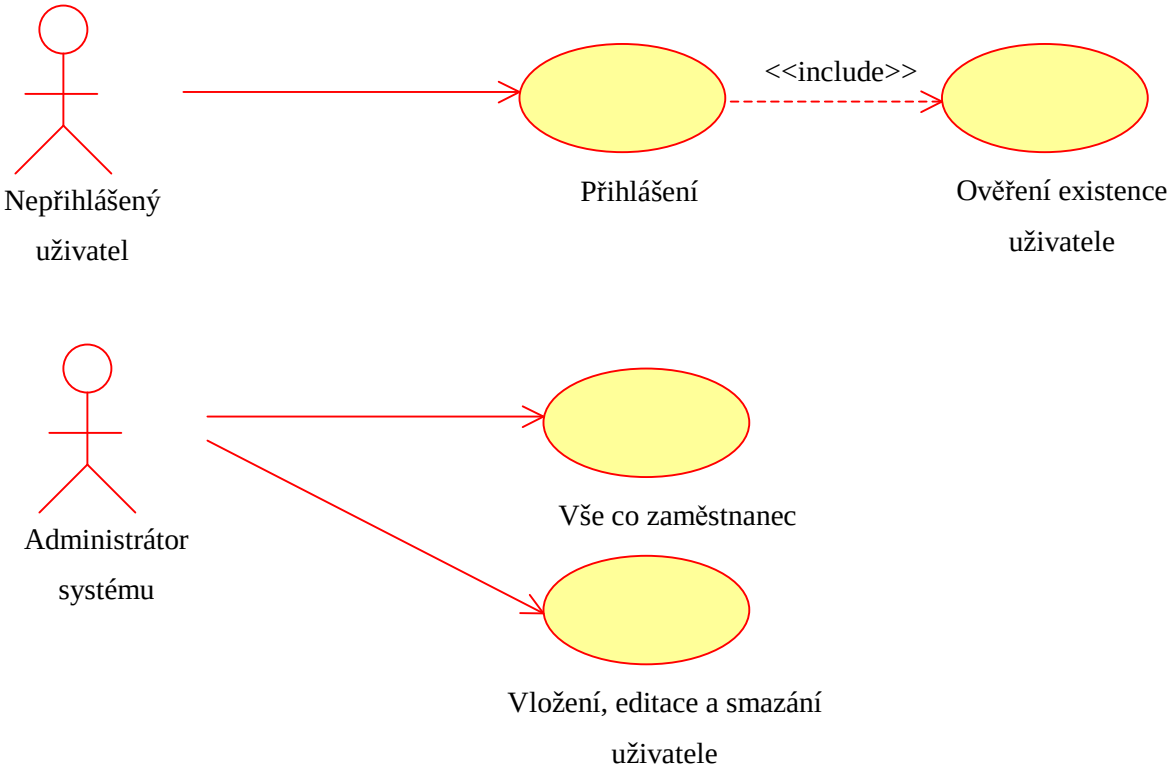
v systému žádná práva, nesmí v systému nic měnit ani si data prohlížet. Typy uživatelů jsou vytvořeny podle potřeb cestovní kanceláře Drak. Protože se nejedná o nijak velkou firmu, ani se systémem nebude pracovat velké množství uživatelů.

Skupiny uživatelů:

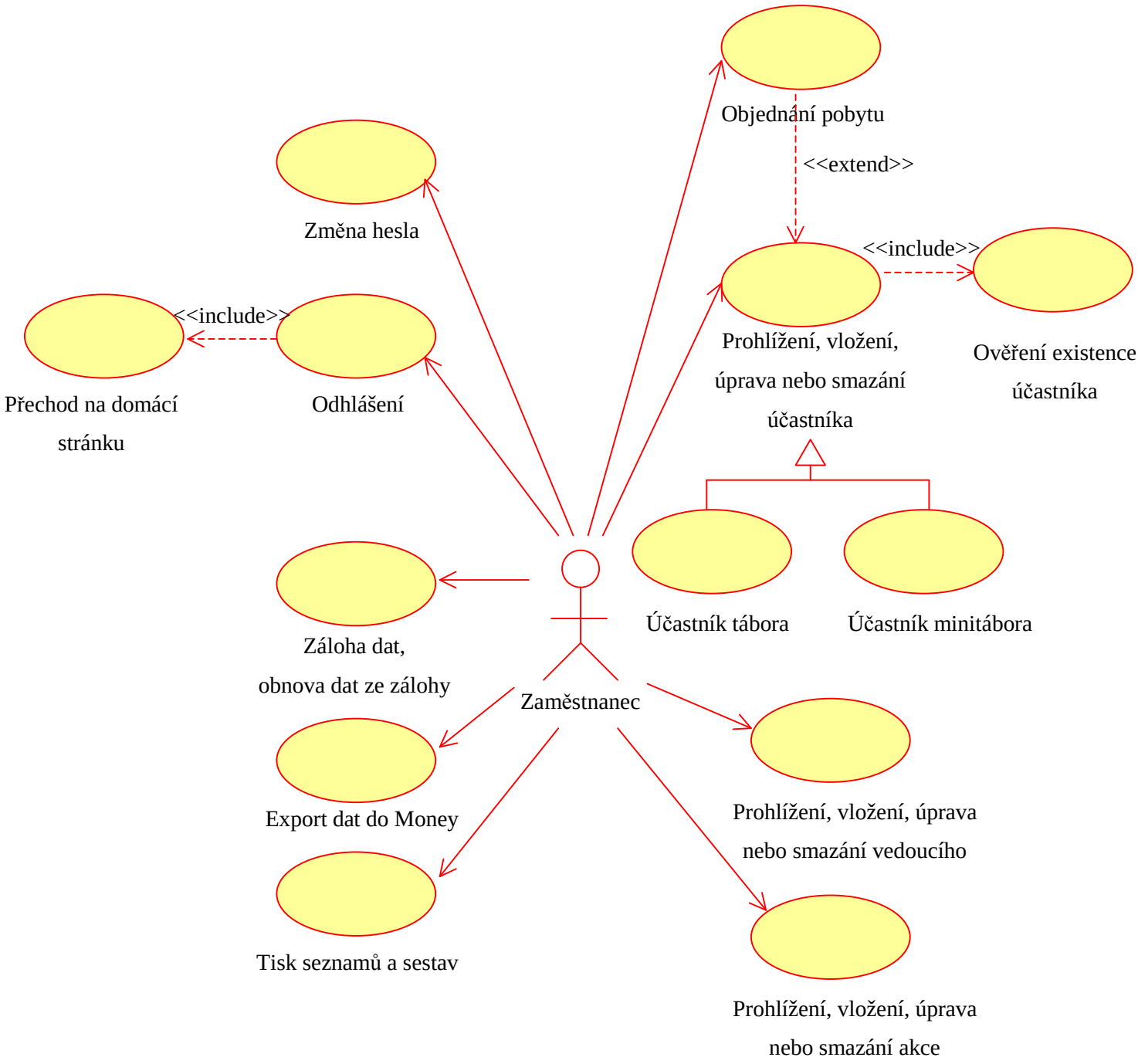
- nepřihlášený uživatel - se systémem nesmí pracovat. Nemá přístup k žádným datům, může se pouze přihlásit do systému.
- zaměstnanec - bude do systému informace vkládat i je z něj čerpat. Systém jim musí nabízet dostatečně komfortní a nekomplikované zadávání, úpravu a mazání dat.
- administrátor - bude se o celý systém starat a spravovat ho. Má přístup ke všem datům.

4.5 Diagram případů použití

Diagram případů použití systému je na obrázku 4.1. Při tvorbě jsem se snažila o jasnou a přehlednou strukturu obrázku. Vyskytují se v něm tři aktéři - nepřihlášený uživatel, zaměstnanec a administrátor. Modeluje funkční požadavky na budoucí systém. Obsahuje nejen iterace mezi případy užití a aktérem, ale i vztahy mezi případy užití samotnými. Jednotlivé vybrané případy použití jsou popsány dále v textu.



Obrázek 4.1.a: Diagram případů použití



Obrázek 4.1.b: Diagram případů použití

4.6 Specifikace vybraných případů použití

4.6.1 Specifikace případu použití „Přihlášení“

Specifikace popisuje chování systému při přihlášení nepřihlášeného uživatele. Specifikace následuje v tabulce 1.

Případ použití: Přihlášení
ID: 1
Stručný popis: Před zahájením práce se systémem je nutné, aby se každý uživatel přihlásil. Bez přihlášení není možné se systémem pracovat
Primární aktéři: Nepřihlášený uživatel
Sekundární aktéři: Žádný
Předpoklady: 1. Uživatel musí být v systému zaregistrovaný
Hlavní tok: 1. Případ použití se spustí, když Nepřihlášený uživatel vybere „Přihlášení“ 2. Uživatel vyplní uživatelské jméno a heslo 3. Systém zkontroluje formát zadaných údajů 4. Jestliže je formát správný 4.1 Systém zkontroluje, zda je uživatel registrovaný 4.2 Jestliže je uživatel registrovaný 4.2.1 Systém ověří zadané přihlašovací údaje 4.2.2 Jestliže jsou přihlašovací údaje správné 4.2.2.1 Systém přihlásí uživatele 4.2.2.2 Systém uloží identifikaci uživatele 4.2.2.3 Systém zaktualizuje model 4.2.3 Jinak 4.2.3.1 Systém vyzve ke správnému přihlášení 4.3 Jinak 4.3.1 Systém vyzve ke správnému přihlášení 5. Jinak 5.1 Systém vyzve uživatele k opravení formátu přihlašovacích údajů
Následné podmínky: 1. Nepřihlášený uživatel byl přihlášen k systému
Alternativní toky: NeplatnéPřihlašovacíÚdaje NespravnýFormátDat Storno

Tabulka 1: Specifikace případu použití „Přihlášení“

4.6.2 Specifikace alternativního toku „Přihlášení: NeplatnéPřihlašovacíÚdaje“

Tento alternativní tok nastane při zadání neplatných údajů při přihlášení do systému. Specifikace je v tabulce 2.

Alternativní tok: Přihlášení: NeplatnéPřihlašovacíÚdaje
ID: 1.1
Stručný popis: Jestliže Nepřihlášený uživatel zadá neplatné uživatelské jméno nebo heslo - nebude přihlášen k systému
Primární aktéři: Nepřihlášený uživatel
Sekundární aktéři: Žádný
Předpoklady: 1. Uživatel musí být v systému zaregistrovaný
Hlavní tok: 1. Alternativní tok se spustí po kroku 4.2.1 hlavního toku 2. Uživatel opravil uživatelské jméno a heslo 3. Systém zkontroluje formát zadaných údajů 4. Jestliže je formát správný 4.1 Systém zkontroluje, zda je uživatel registrovaný 4.2 Jestliže je uživatel registrovaný 4.2.1 Systém ověří zadané přihlašovací údaje 4.2.2 Jestliže jsou přihlašovací údaje správné 4.2.2.1 Systém přihlásí uživatele 4.2.2.2 Systém uloží identifikaci uživatele 4.2.2.3 Systém zaktualizuje model 4.2.3 Jinak 4.2.3.1 Systém vyzve ke správnému přihlášení 4.3 Jinak 3.1 Systém vyzve ke správnému přihlášení 5. Jinak 5.1 Systém vyzve uživatele k opravení formátu přihlašovacích údajů
Následné podmínky: Žádné

Tabulka 2: Specifikace alternativního toku „Přihlášení: NeplatnéPřihlašovacíÚdaje“

4.6.3 Specifikace případu použití „Vložení účastníka“

Případ použití „Vložení účastníka“ nastane při zadávání nového klienta do systému. Specifikace je v tabulce 3.

Případ použití: Vložení účastníka
ID: 2
Stručný popis: Případ použití se spustí při každém vkládání nového klienta cestovní kanceláře do systému.
Primární aktéři: Zaměstnanec
Sekundární aktéři: Žádný
Předpoklady: 1. Uživatel musí být v systému registrovaný 2. Uživatel musí být přihlášený
Hlavní tok: 1. Případ použití se spustí, když Zaměstnanec vybere „Vložit účastníka“ 2. Systém zkontroluje, zda je zaměstnanec přihlášený 3. Jestliže je zaměstnanec přihlášený 3.1 Zaměstnanec vyplní všechny povinné údaje 3.2 Jestliže jsou vyplněny všechny povinné údaje 3.2.1 Systém zkontroluje formát zadaných údajů 3.2.2 Jestliže je formát správný 3.2.2.1 Údaje o účastníkovi jsou vloženy do databáze 3.3 Jinak 3.3.1 Systém vyzve zaměstnance k opravení formátu povinných údajů 3.3 Jinak 3.3.1 Systém vyzve k vyplnění všech povinných údajů 4. Jinak 4.1 Systém vyzve uživatele k přihlášení
Následné podmínky: 1. Nový uživatel byl vložen do systému
Alternativní toky: Přihlášení NevyplněnéPovinnéÚdaje NespravnýFormátDat Storno

Tabulka 3: Specifikace případu použití „Vložení účastníka“

4.6.4 Seznam výjimek vybraného případu použití

Výjimky, které vznikají v rámci případů použití vložení účastníka tábora, a požadované chování systémů při jejich výskytu.

E1 – Prázdné přihlašovací jméno nebo heslo

Výjimka vznikne v případě, že uživatel při přihlašování nezadá login nebo heslo. Uživatel je vyzván k vyplnění povinných údajů.

E2 – Nekorektní uživatelské jméno nebo heslo

Vznikne v případě, že nepřihlášený uživatel zadá nekorektní uživatelské jméno nebo heslo. Systém zobrazí chybovou hlášku. Pak vyzve uživatele k opakovanému zadání přihlašovacích údajů.

E3 – Nevyplněné povinné údaje při vkládání účastníka

Jestliže uživatel systému nevyplní všechny povinné údaje, je na tuto skutečnost upozorněn a data nejsou vložena do db. V případě, že jsou data opravena a jsou ve správném formátu, přistoupí se k vložení dat do db.

E4 – Neplatný formát data narození

Formát data narození musí být rrrr-mm-dd. Pokud uživatel vyplní políčko Narození, je nutné data zadávat ve správném formátu. V opačném případě je vypsána chybová zpráva, která informuje uživatele o špatném formátu dat.

E5 – Nevložení údajů do DB

Výjimka, která může vzniknout ve většině případů použití a to buď vinou nezadání všech povinných údajů, zadáním chybných údajů nebo chybou při vkládání. Systém zobrazí příslušné hlášení (dle výjimky). Jestliže se jedná o chybu, kterou může uživatel opravit, je vyzván k opravě chyby. Pokud dojde k opravě chybných údajů, jsou následně údaje vloženy do databáze. Za podmínky, že chyba vznikla při vkládání údajů (chybou při komunikaci s DB), nejsou data uložena a vkládání do DB je ukončeno.

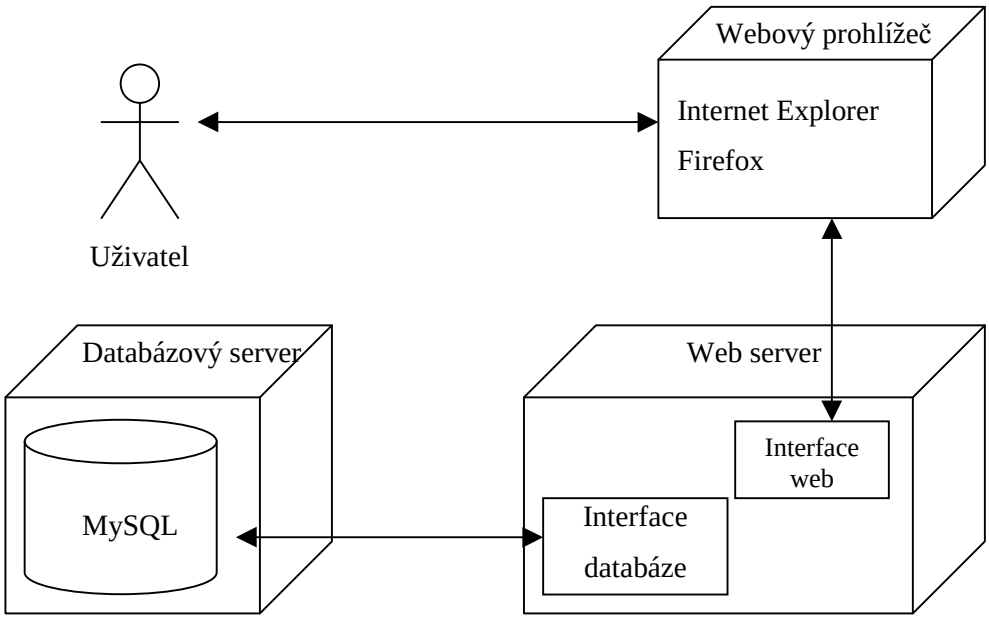
5 Návrh

Kapitola se zabývá návrhem výsledného systému. Rozebírá architekturu MVC, přístupová práva a obsahuje návrh jednoho přejímacího testu.

Návrh vychází z analýzy požadavků. Při návrhu je využito jazyka UML. Diagramy jsou modelované v programu Ration Rose Enterprise Edition.

1.1 Návrh architektury aplikace s webovým rozhraním

Návrh architektury s webovým rozhraním je na obrázku 5.1.



Obrázek 5.1: Architektura aplikace

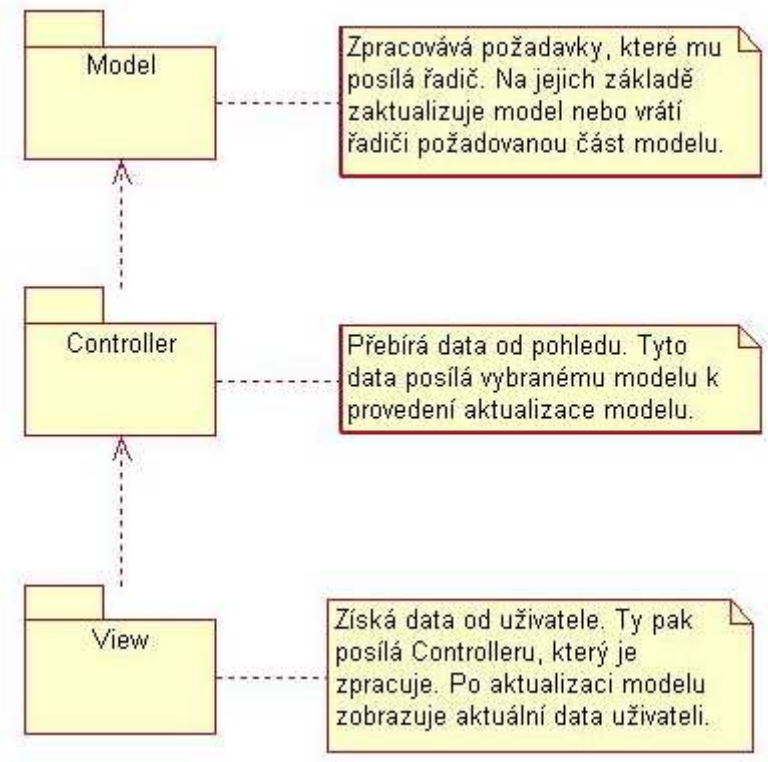
5.2 Architektura systému

Třívrstvá architektura systému rozděluje aplikaci na prezentační vrstvu, vrstvu aplikační logiky a datovou vrstvu. Prezentační vrstva zobrazuje uživateli výstup a přijímá od uživatele vstupy. O zpracování požadavků a úpravu modelu se stará vrstva aplikační logiky. Datová vrstva přistupuje k datům systému.

Architektura vyvíjeného systému bude podle návrhového vzoru Model-View-Controller. View odpovídá prezentační vrstvě. Stará se o způsob, jakým jsou data prezentována uživateli. Většinou to jsou stránky tvořené HTML nebo WML Controller je část vrstvy aplikační logiky. Stará se o

vytvoření modelu a také volá správný view. Model je datová vrstva spojená s částí vrstvy aplikační logiky. Je zodpovědný za výběr dat, která jsou pak prezentována uživateli.

Rozdělení MVC do třívrstvé architektury je na obrázku 5.2.



Obrázek 5.2: Rozdělení MVC do třívrstvé architektury

5.2.1 Funkce View

Pohled slouží k získávání vstupů od uživatele a zobrazení aktualizovaného modelu. Po zadání vstupu jsou data zkontrolována JavaScriptem (při získávání dat přes prohlížeč počítače). Pak jsou předaná controlleru k dalšímu zpracování. Ten zaktualizuje model a zpět do view vrací zprávu o úspěšném/neúspěšném provedení operace.

Při zobrazení modelu dochází k načtení pouze aktuálních dat. V případě, že je požadována jiná část modelu, controller na základě požadavku pošle do view nová data.

5.2.2 Funkce controlleru

Controller zpracovává všechny požadavky od view. Po analýze zavolá příslušný model. Od modelu pak převezme výsledek operace a předá ho do view.

5.2.3 Funkce modelu

Model provádí výběr nebo aktualizaci dat na základě požadavku. Data nebo výsledek operace posílá zpět controlleru, který jej pak posílá do view.

5.3 Vrstvy třívrstvé architektury

5.3.1 Klientská vrstva

Klientská vrstva třívrstvé architektury je totožná s vrstvou view u vzoru MVC. Funkce vrstvy jsou popsány v kapitole 5.2.1. Klientská vrstva může být vytvořena nezávisle na funkcích systému. Vytvořením klienských vrstev lze docílit toho, že aplikace bude přístupná z různých typů zařízení.

5.3.2 Aplikační vrstva

Aplikační vrstvě odpovídá controller a model ze vzoru MVC.

Funkce controlleru je popsána v kapitole 5.2.2. Po zpracování předá data modelu. Do modelu patří aplikační i datová vrstva. Fukce modelu je zmíněna v kapitole 5.2.3. V aplikační vrstvě model převezme data nebo požadavek od controlleru a zaktualizuje model nebo vrátí data.

5.3.3 Datová vrstva

V architektuře MVC patří datová vrstva do modelu. Ten přistupuje k datům. Změna datové vrstvy se týká pouze modelu. Neprojeví se v controlleru nebo pohledu. Konceptuální diagram tříd datové vrstvy je vzhledem ke svému rozsahu přiložen k technické zprávě jako Příloha 1. Konceptuální diagram datové vrstvy.

Diagram obsahuje třídu Osoba, která je předkem pro třídy Uc_tabora, Uc_minitabora a Vedouci. Tyto osoby se účastní jednotlivých akcí pořádaných cestovní kanceláří Drak. Které osoby se zúčastnily kterých akcí, eviduje atribut vztahu. Účastníci táborů a minitáborů patří do určité kategorie. Akce jsou různých typů a konají se na odlišných místech. U vedoucích je uchováváno na jakou školu chodili či chodí, kterou fakultu studovali či studují a pozice, které nejčastěji na táborech zastávají.U osob se kromě jiných údajů eviduje i dosažený titul a město trvalého příp. přechodného bydliště.

5.4 Přístupová práva

Podle požadavků budou k systému přistupovat tři typy uživatelů:

- nepřihlášený uživatel – tento uživatel nebude moct se systémem pracovat, bude mu umožněno pouze přihlášení k systému

- zaměstnanci CK – budou moci se systémem po přihlášení pracovat a provádět předem stanovené činnosti
- administrátor systému – bude mít stejná práva jako zaměstnanec plus některá navíc. Bude mít nejvyšší práva. Pro tento typ uživatele platí pravidlo, že má přístup všude tam, kde zaměstnanec.

Přístupová práva k jednotlivým stránkám nejsou definována globálně. O přístupu k dané stránce rozhoduje controller, který přístup danému typu uživatele buď povolí nebo zakáže.

5.5 Přihlášení uživatele

V systému budou uchovávány osobní informace klientů. Aby nedošlo ke zneužití těchto údajů, je nutné umožnit přístup pouze kvalifikovaným zaměstnancům cestovní kanceláře Drak. Přidávat nové uživatele do systému bude umožněno pouze administrátorovi. Ten musí zvážit, zda je vhodné, aby daný uživatel měl přístupová práva k aplikaci.

Uživatelé se budou přihlašovat pomocí loginu a hesla. Ze strany aplikace nejsou na heslo kladeny žádné požadavky, ale mělo by být voleno vhodně. Uživatelé by své heslo neměli sdělovat druhé osobě. Ani administrátor nebude mít přístup k heslu. Jestliže uživatel heslo zapomene, může administrátor systému uživatele smazat a znovu vytvořit. Změnu svého hesla může provést pouze uživatel.

Do databáze je heslo ukládáno v šifrované podobě, za použití funkce md5().

5.6 Návrh přejímacího testu

Test validuje případ použití vložení nového účastníka tábora.

Krok číslo: 1

Testuje se: Zda jsou všechny povinné údaje zadány.

Očekávaný výsledek: Výsledek je OK v případě, že všechny povinné položky jsou vyplněné.

Krok číslo: 2

Testuje se: Kontrola datumu narození.

Očekávaný výsledek: Jestliže je datum narození zadané, pole datum narození musí mít správný formát rrrr-mm-dd.

Krok číslo: 3

Testuje se: Kontrola poštovního směrovacího čísla.

Očekávaný výsledek: Pokud je pole vyplněné, zadaný formát musí být ěěě ěě.

Krok číslo: 4

Testuje se: Kontrola formátu telefonů – byt, matka zaměstnání, otec zaměstnání, mobil matka, mobil otec, mobil dítě.

Očekávaný výsledek: Jestliže jsou pole vyplněna, jejich formát musí být +ččč ččč ččč ččč.

Krok číslo: 5

Testuje se: Kontrola emailů – matka, otec a dítě.

Očekávaný výsledek: V případě vyplnění políček formuláře musí být formát vložených dat platným formátem pro emailovou adresu, tzn. musí obsahovat zavináč, na konci je tečka a dvě nebo tři písmena, ...

Krok číslo: 6

Testuje se: Kontrola faxu.

Očekávaný výsledek: Jestliže je pole vyplněné, musí obsahovat devět čísel.

Pokud všechny testy proběhnou bez problému, nahrají se do databáze data o novém účastníkovi tábora – účastník bude vložen do systému.

6 Implementace

Kapitola popisuje implementaci systému. Zaměřuji se v ní na nejdůležitější a nejzajímavější části implementace. Nejprve se věnuji používaným konvencím a popisu řešení, které jsem zvolila pro implementaci multiplatformního přístupu. Dále jsou rozebrány metody získávání vstupů od uživatele a vzhled stránek systému. Kapitola se zabývá jednotlivými částmi návrhového vzoru MVC. Věnuje se také generování tiskových sestav a provedené úpravě kódu Zend Frameworku. Ke konci kapitoly je rozebrán postup při případném rozšíření systému. Poslední část se věnuje nasazení systému do praxe.

6.1 Používané konvence

Celý systém je rozdělen na tři části - model, view a controller. Jednotlivé části se nachází v rozdílných adresářích – viz. kapitola 3.3. Vstupním bodem celé aplikace je tzv. bootstrap index.php. Všechny požadavky jsou přesměrovány na tento soubor pomocí souboru .htaccess v kořenovém adresáři aplikace.

Pojmenovávání controllerů a umístění do adresáře je blíž popsáno v kapitole 6.8. Konvence používané u modelů jsou vysvětleny v kapitole 6.9. Vše, co s týká views, je v kapitole 6.7 a 6.10.

6.2 Stránky ve View

Pojmenování stránek ve View je tvořeno názvem stránky a příponou .phtml. Nejčastější názvy stránek jsou následující:

- index - úvodní stránka s obecnými informacemi (je-li potřeba)
- add - stránka pro vložení nového záznamu
- edit - stránka pro editaci již uloženého záznamu
- delete - stránka pro smazání záznamu

Dále jsou dodržována tato pravidla:

- soubory stránek, které pracují se stejnými daty, jsou ve společném adresáři
- název adresáře je tvořen malými písmeny
- pojmenování je shodné s názvem tabulky v DB
- pokud stránka slouží vložení nových dat, je použito formuláře, který je v souboru _addForm.phtml (u view pro počítač)
- jestliže se stránka používá k úpravě dat, formulář je uložen v souboru s názvem _editForm.phtml (u view pro počítač)
- každý formulář, který je určen pro vkládání nebo úpravu dat, je před odesláním zkontrolován JavaScriptem (u view pro počítač)

Aplikace je implementována tak, že je možné k ní přistupovat z počítače a mobilního telefonu. Většinou se ale bude používat ve spojení s počítačem. Přístup přes mobilní telefon bude využíván pouze ve výjimečných případech. Proto byly po domluvě s majitelem cestovní kanceláře Drak implementovány pro mobilní telefon pouze ty pohledy, které budou využity.

Přes mobilní telefon lze přistupovat k následujícím pohledům:

- akce (zobrazení a vkládání)
- fakulty (zobrazení a vkládání)
- města (zobrazení a vkládání)
- školy (zobrazení a vkládání)
- účastníci tábora (zobrazení, vložení, smazání, vložení a smazání účasti, vyhledávání)
- účastníci minitábora (zobrazení, vložení, smazání, vložení a smazání účasti, vyhledávání)
- vedoucí (zobrazení, vložení, smazání, vložení a smazání účasti, vyhledávání)

6.3 Multiplatformní přístup k aplikaci

Aplikace je multiplatformní, tzn. je možný přístup z více typů zařízení. Podle typu zařízení se použije příslušný view. Momentálně lze systém zobrazovat na monitoru počítače nebo na displeji mobilního telefonu. Rozdělením aplikace na tři části (model, view a controller) je docíleno toho, že lze používat stejný model a controller pro různá zařízení. Jediné, co je rozdílné, jsou views.

To, který pohled se použije, je nutné nějak rozhodnout. Proto jsem vytvořila třídu ControllerSwitch. Ta určí, které pohledy se budou používat k zobrazení modelu. Toto rozhodnutí provádí na základě analýzy URL. Třída má funkci setDeviceType(), jejíž kód je ve výpisu 6.1.

```
public function setDeviceType($request, $view)
{
  $device = '';

  if ($request -> getParam('device', 0))
  {
    $device = $request -> getParam('device', 0);
  }
  else
  {
    $device = 'pc';
  }
```

Výpis 6.1.a: Výpis funkce setDeviceType()

```
switch ($device)
 {
   case 'pc': break;
   case 'mobile': break;
   default: $device = 'pc';
 }

 $view->setScriptPath("application/views/$device/scripts");

 return $device;
}
```

Výpis 6.1.b: Výpis funkce setDeviceType()

Funkce setDeviceType() provádí analýzu URL. Každý požadavek, který směřuje na aplikace, je předán této funkci. Ona pak určí, které pohledy jsou určeny pro toto zařízení, resp. ve kterém adresáři má hledat příslušné pohledy.

Rozhodnutí provádí na základě hodnoty parametru device. Jestliže parametr device obsahuje nějakou hodnotu, vezme tuto hodnotu a podle ní určí typ zařízení. Momentálně jsou pouze dvě platné hodnoty parametru device a to ‘pc‘ nebo ‘mobile‘. Parametr ‘pc‘ říká, že se jedná o počítač, parametr ‘mobile‘ označuje přístup z mobilního telefonu.

Pokud v URL nebude parametr device uveden nebo bude jeho hodnota ‘pc‘, jedná se o přístup z defaultního zařízení (tj. z počítače). Pohledy, které se mají využít, jsou uloženy v adresáři application/views/pc/scripts.

V případě, že device má hodnotu ‘mobile‘, přistupuje se k aplikaci z mobilního telefonu. Views jsou pak v adresáři application/views/mobile/scripts.

Snadno lze doimplementovat i přístup z dalšího zařízení. Pohledy pro toho zařízení budou v dalším adresáři, např. application/views/pda/scripts. Hodnota parametru pak bude /device/pda.

Příklady požadavků na stejné action, ale z různých zařízení:

http://paja.mprazak.info/drak/index/index/device/pc - požadavek na action index controlleru index z počítače

http://paja.mprazak.info/drak/index/index/device/mobile - požadavek na action index controlleru index z mobilního telefonu

Přístup z mobilního telefonu byl testován za použití simulátoru Openwave SDK. Ten je produktem firmy Openwave. Produkt lze zdarma stáhnout na adrese http://www.openwave.com. K testování jsem použila verzi Openwave SDK 6.1 HTTP. Okno aplikace emulátoru je na obrázku 6.1.

Obrázek 6.1: Okno aplikace Openwave SDK

6.4 Získávání vstupů od uživatele

Uživatel své vstupy vkládá buď prostřednictvím internetového prohlížeče v počítači nebo přes prohlížeč mobilního telefonu. Požadavky mohou být dvou typů:

- POST požadavek – tyto požadavky přicházejí např. po odeslání formuláře, který uživatel vyplnil. Obsahují data, která následně aktualizují model. Validita dat je před odesláním prověřena pomocí Java Scriptu.
- GET požadavek – tento typ požadavků vzniká např. při zadání adresy do prohlížeče, kliknutí na menu, ... Po něm většinou následuje získání dat a zobrazení modelu.

6.4.1 Zpracování požadavku

Požadavky mohou být dvojího typu – POST a GET.

POST požadavek vznikne po odeslání formuláře. Má za následek změnu modelu. Samotná změna probíhá v několika krocích:

- před samotným odesláním dat dochází ke kontrole dat na straně klienta. Pokud jsou data validní, je odeslán požadavek na aktualizaci modelu.

- požadavek na aktualizaci přijme controller. Data překontroluje a pošle modelu
- model aktualizuje data
- zpět controlleru pošle výsledek operace a případné chyby
- controller předá výsledek operace do view, který stav zobrazí uživateli.

Ukázka zpracování požadavku je ve výpisu 6.2.

```
…
if ($this->_request->isPost())
        {
          Zend_Loader::loadClass('Zend_Filter_StripTags');
          $filter = new Zend_Filter_StripTags();

          $nazev = trim($filter->filter($this->_request->getPost('nazev')));
…
```

Výpis 6.2: Zpracování POST požadavku

Nejdříve se ověří, zda byl formulář odeslaný. Z dat se odstraní zakázané znaky a data se uloží do příslušné proměnné. Jestliže jsou všechna data v pořádku, jsou předána modelu k aktualizaci modelu.

GET požadavek vzniká při zadání požadované adresy buď uživatelem, nebo aplikací. Při tomto požadavku nedochází k aktualizaci modelu, pouze se provede zobrazení modelu. Součástí požadavku mohou být i parametry předávané přes URL.

Výhodou metody GET je skutečnost, že při jiné hodnotě proměnné je vlastně voláno jiné URL. Toto je jeden z prostředků, jak předejít nechtěnému ukládání stránek do vyrovnávací paměti telefonu či na WAPové bráně. Skript mohl být již v minulosti volán uživatelem s jinými hodnotami proměnných. Při opětovném volání s novými hodnotami proměnných může být funkčnost zcela jiná, což by se neprojevilo, kdyby byla stránka již nesprávně cacheována a znovu by se pouze jen zobrazovala. Nevýhodou této metody je fakt, že délka ULR ve WAPu má své omezení, a tudíž i prostor za URL adresou je omezený.

6.4.2 Controller - zpracování požadavků

Jestliže přijde od pohledu požadavek, je přeposlán controlleru. Přesněji řečeno je volána přímo metoda daného controlleru. Který controller a která metoda má zpracovat právě daný požadavek je dáno parametry GET požadavku - první parametr je název controlleru, druhý je metoda.

Před zahájením zpracování se nejprve provede ověření přístupových práv. Jestliže uživatel není přihlášen nebo nemá právo provádět danou akci, je vypsán text, který ho o této skutečnosti informuje.

Pouze v případě, že daný uživatel má právo provádět konkrétní akci, dochází k dalšímu zpracování požadavku.

Více o controlleru je v kapitole 6.8.

6.4.3 Model – zpracování požadavků

Model se stará o aktualizaci modelu a vracení požadovaných dat. Přistupuje k databázi, provádí požadované operace a výsledek operace zasílá zpět controlleru. Model už neprovádí žádné kontroly oprávnění přistupovat k datům.

Další informace o modelu jsou v kapitole 6.9.

6.4.4 View – zpracování požadavků

Pohled přijímá a zpracovává požadavky od uživatelů. V případě potřeby vstupní data zkontroluje. Pohled pak požadavky předává příslušnému controlleru.

Bližší popis view je v kapitole 6.7 a 6.10.

6.5 Vzhled stránek

Vzhled stránek je rozdílný podle typu zařízení, na kterém se budou zobrazovat.

6.5.1 Stránky pro počítač

V celé aplikaci je jednotný vzhled stránek. Stránky jsou optimalizovány pro rozlišení 1024 x 768 pixelů. Přestože se jedná již o téměř nepoužívané rozlišení, cestovní kancelář Drak toto rozlišení stále ještě používá na většině svých počítačů.

Obrázek 6.2: Struktura stránky

Stránka je složena ze 4 částí. V horní části je hlavička a menu. Levý panel slouží k přihlášení a odhlášení uživatelů. Největší plocha je vyhrazena pro obsah. Ve spodní části se nachází patička. Struktura stránky je zobrazena na obrázku 6.2.

6.5.2 Stránka pro mobilní telefon

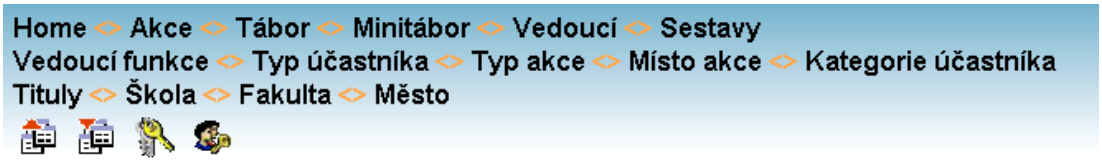
Stránky pro mobilní telefon neobsahují žádné grafické prvky. Stránky jsou co nejjednodušší, aby byly přehledné a jejich stahování netrvalo příliš dlouho. Úvodní stránka obsahuje odkaz na přihlašovací stránku. Struktura úvodní stránky je na obrázku 6.3.

Obrázek 6.3: Struktura úvodní stránky pro mobilní telefon

6.5.3 Menu pro počítač

Menu se do každé stránky vkládá spolu s hlavičkou dokumentu. Vzhled menu byl zvolen s důrazem na přehlednost a praktičnost. Velká část menu je textová. Zbylé čtyři položky jsou tvořeny ikonou.

Menu je rozděleno do čtyř částí – řádků. Rozdělení je na obrázku 6.4. Položky jsou řazeny podle frekvence používání. První řádek tvoří odkazy, které se používají nejčastěji. Druhý a třetí řádek tvoří položky, které jsou v aplikaci používány jako číselníky – tzn. např. typ účastníka – obsahuje všechny možné typy, jakých mohou být účastníci. Poslední část menu slouží k záloze databáze a následné obnově dat ze zálohy. Také si zde uživatelé mohou změnit přístupové heslo k systému. Poslední položka menu je přístupná pouze administrátorovi systému a slouží k editaci uživatelů.



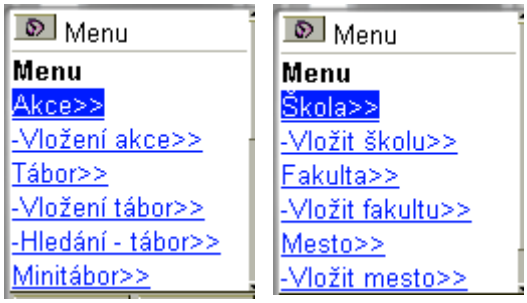
Obrázek 6.4: Menu aplikace

Menu se zobrazuje podle typu uživatele, který se přihlásil. Implicitně je zobrazené menu pro uživatele typu zaměstnanec. Pokud si nepřihlášený uživatel vybere libovolnou položku menu, zobrazí se chybové hlášení, které ho upozorní na skutečnost, že je nejprve nutné se k systému přihlásit. Položky menu se zobrazují podle přístupových práv uživatele. Každému se zobrazí pouze ty stránky, ke kterým má přístup.

Princip zobrazení pouze povolených stránek je přehledný pro uživatele. Pokud by bylo v budoucnu nutné do systému přidat další typ uživatele, který bude mít odlišná práva, lze lehce doimplementovat zobrazení pouze povolených položek menu. Jestliže administrátor systému změní uživateli jeho roli, ihned po novém přihlášení uživatele se tato změna projeví i v nabídce menu.

6.5.4 Menu po mobilní telefon

Menu se zobrazí po přihlášení do systému. Je pouze textové a je rozděleno na dvě card. První část menu obsahuje častěji používané odkazy – umožňuje práci s účastníky táborů, minitáborů, s vedoucími a akcemi. Na další card je možné přejít přes odkaz Další odkazy. Druhá card obsahuje odkazy pro práci se školami, fakultami a městy. Na první card je možné se vrátit přes První odkazy. Menu je zobrazeno na obrázku 6.5.



Obrázek 6.5: Menu první a druhé card

Vzhledem k velikosti displeje mobilního telefonu není menu neustále přístupné. Ale dá se na něj vrátit z každé stránky a to pomocí odkazu Zpět, který se nachází ve spodní části card.

6.6 Uživatelské relace

Který uživatel je právě přihlášen je uchováváno v proměnné session. Ta je aktivní po celou dobu uživatelovy práce s aplikací. IS sezení je automaticky generováno PHP a ukládáno po dobu trvání sezení u klienta. Může být uloženo na uživatelově počítači v tzv. cookie[1], nebo se předává prostřednictvím URL.

Problémy spojené s používáním cookies vyplývají z toho, že některé prohlížeče s nimi neumějí pracovat nebo že uživatelé mohou mít cookies v prohlížeči vypnuté. Navíc prohlížeče pro mobilní telefony cookies neukládají vůbec.

Proto bylo nutné implementovat funkci, která v případě potřeby předá session prostřednictvím URL. Funkce je jmenuje addSID() a jako vstupní parametr vyžaduje URL a znak pro oddělení parametrů. Kód funkce je ve výpisu 6.3.

[1] Cookie (doslovně sušenky) je malá informace, kterou může skript uložit na stroji u klienta. Cookies se posílají jako záhlaví HTTP.

```
function addSID($url, $sep = '&amp;')
{
  if (!defined('SID') || strlen(SID) == 0)
  {
    return($url); //session neni k dispozici nebo funguje predavani pres cookies
  }
  else
  {
    if (($pos = strpos($url, '?')) === false)
    {
      return $url.'?'.SID; //v URL neni uvedeny otaznik, nema zadny parametr
    }
    else
    {
      if ($pos == strlen($url) - 1) //v URL uz je otaznik
      {
        return $url.SID; //ale na posledni pozici, za nim nic
      }
      else
      {
        return $url.$sep.SID;
      }
    }
  }
}
```

Výpis 6.3: Výpis funkce addSID()

Tato funkce přidá v případě potřeby identifikaci session do URL. Pokud jsou zapnuté cookies, nepřidává k URL nic. V případě, že je nutné přidat identifikaci session k URL, provede analýzu URL. Když požadavek nemá žádný parametr, dá na konec požadavku otazník a SID (konstantu, ve které je uloženo ID sezení). Jestliže URL obsahuje otazník, ale nejsou za ním žádné parametry, vloží SID za tento otazník. Poslední případ je, že URL již obsahuje nějaké parametry. Pak přidá za parametry oddělovat parametrů a za něj dá SID.

Použitím této funkce je možné využívat session k uložení přihlášeného uživatele i při zakázaných cookies nebo v prohlížeči mobilního telefonu.

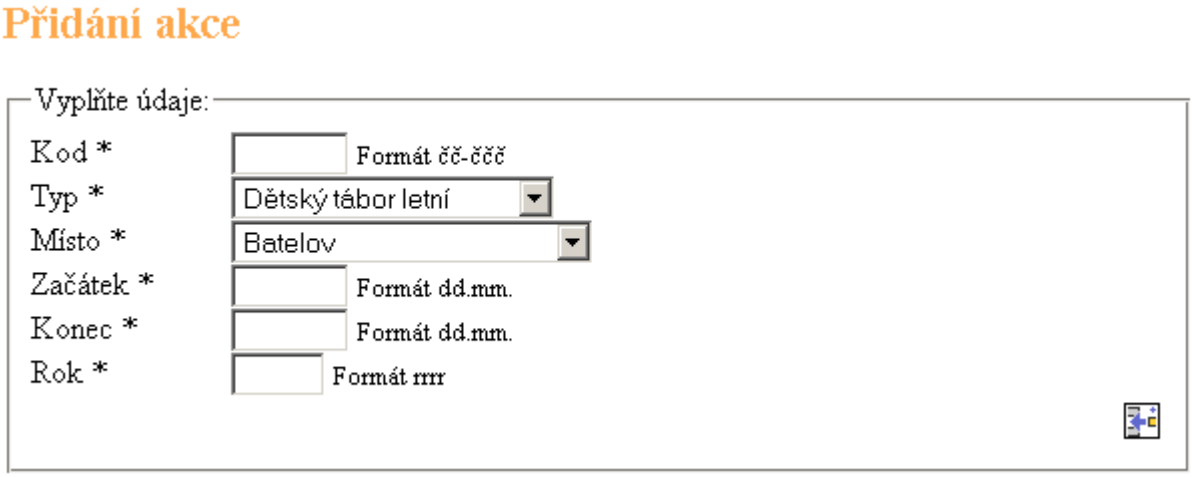
6.7 View

Klientská vrstva je přístupná minimálně z prohlížečů Internet Explorer 6.0 a Firefox 2.0. Implementace je provedena jazykem PHP a HTML. Výsledný vzhled je dotvořen pomocí CSS. Pohledy slouží k zobrazení modelu a přijímání vstupů od uživatelů.

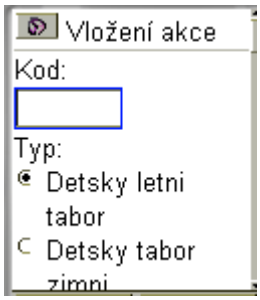
Kromě částí komponenty view, které budou popsané v této kapitole, jsou další části popsány v kapitole 6.5.1 a 6.5.2 - zobrazení stránky, menu je v kapitole 6.5.3 a 6.5.4 a zásady používané při pojmenování pohledů jsou v kapitole 6.2.

6.7.1 Formuláře

Formuláře slouží k získávání vstupů od uživatele. Obsahují textové pole, radio buttony, select boxy, ... Mají jednotný vzhled, což je přehledné nejen pro toho, kdo s formulářem pracuje, ale i pro případnou údržbu. Ukázky formulářů pro vkládání jsou na obrázku 6.6 a 6.7.



Obrázek 6.6: Formulář pro vložení akce pro počítač



Obrázek 6.7: Formulář pro vložení akce pro mobilní telefon

Data z formuláře jsou odesílána controlleru metodou POST. Před samotným odesláním dat z prohlížeče počítače probíhá kontrola vstupních dat. Ta je prováděna na straně klienta pomocí JavaScriptu. Při přístupu k aplikaci z mobilního telefonu je kontrola prováděna až v controlleru. Ukázka kontroly emailové adresy je ve výpise 6.4

```
function zkontroluj_email_1(email_1)
{
  rv = /^[a-zA-Z0-9]+([a-z0-9_.-])*@([a-z0-9.-])+\.[a-z]{2,3}$/;
  return email_1.search(rv) == 0;
}



function ValidateVedouci()
{
if( self.document.forms.vedouci.email_1.value.length != 0) {
            if (!zkontroluj_email_1(self.document.forms.vedouci.email_1.value))
   {
     alert('Vyplňte prosím správně hodnotu do pole Email 1.');
                        focus();
     return false;
   }
  }
return true;
}
```

Výpis 6.4: Kontrola tvaru emailové adresy

Jestliže pole pro email obsahuje nějaký text, je zavolána funkce zkontroluj_email_1(). Tato funkce pomocí regulárního výrazu projde text emailu. Zkontroluje, zda-li obsahuje zavináč a zda je zadaný formát adresy platným formátem pro emailovou adresu. Regulárnímu výrazu odpovídají např. emaily: email@seznam.cz, drak.ck@drak.cz, …

Kontrola probíhá jak při vkládání nových dat, tak i při úpravě dat. Pokud data nejsou validní, nedochází k jejich odeslání. Po zpracování dat a aktualizaci modelu se vrací zpět do pohledu výsledek operace.

Maximální velikost textových prvků je hlídána také vlastností textových polí (maxlength). Údaje, které nemůže uživatel měnit jsou předávána pomocí skrytých polí.

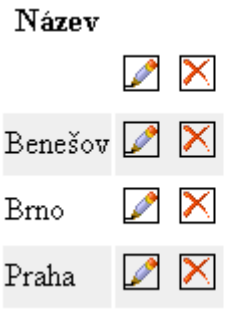
Jak již bylo řečeno, zpracování formuláře probíhá v konkrétním controlleru. Po převzetí zadaných dat se z nich odstraní případné nežádoucí znaky. Údaje jsou následně předávány modelu, který provede aktualizaci. Ukázka zpracování dat z formuláře a jejich následné předání do modelu je ve výpise 6.5.

```
…
            $kod = trim($filter->filter($this->_request->getPost('kod')));
            $typ = trim($filter->filter($this->_request->getPost('typ')));
            $misto = trim($filter->filter($this->_request->getPost('misto')));
            $zacatek = trim($filter->filter($this->_request->getPost('zacatek')));
            $konec = trim($filter->filter($this->_request->getPost('konec')));
            $rok = trim($filter->filter($this->_request->getPost('rok')));

          if ($kod != '' && $typ != '' && $misto != '' && $konec != '' && $zacatek != ''
&& $rok != '')
          {
            $data = array(
              'kod'  => $kod,
              'typ'  => $typ,
              'misto'  => $misto,
              'zacatek'  => $zacatek,
              'konec'  => $konec,
              'rok' => $rok, );

            try
            { // volani modelu
                        $akceManager -> addAkce($data);
                        $this-
>_redirect(addSID('/akce/index/device/'.$device.'/res/1','&'));
                        return;
            }
            catch (Zend_Exception $e)
            { // chyba pri vkladani
              $this->view->error = "Při vkládání do db nastala chyba!";
              //echo '' ERROR: Caught exception: '' . get_class($e) . ''\n Message: '' .
$e->getMessage() . ''\n '';
            }
          }
```

Výpis 6.5: Zpracování vstupních hodnot a předání modelu

6.7.2 Výpis dat

Stránky, které neslouží přímo k editaci dat, ale data pouze vypisují, mají také jednotný vzhled. Sudé a liché řádky ve výpisu jsou barevně odlišeny. U pohledů pro počítač jsou na konci každého řádku ikony pro editaci a smazání záznamu. Záznamy jsou řazeny abecedně, podle primárního klíče. Ukázka výpisu měst je na obrázku 6.8.



Obrázek 6.8: Výpis měst

Jestliže je předpoklad, že daných záznamů bude více než 15, jsou záznamy vypisovány postupně. Mezi jednotlivými stránkami se lze přepínat pomocí šipek ve spodní části stránky (viz. obrázek 6.9). Počet záznamů vypsaných na stránku lze libovolně upravit v kódu controlleru (proměnná $zobrazit).

Obrázek 6.9: Navigační tlačítka ve spodní části výpisu

I výpis pro mobilní telefony má barevně odlišené liché a sudé stránky. V případě, že je umožněno dané záznamy přes pohled smazat, nachází se na konci řádku odkaz ‘X’, který slouží k odstranění záznamu. Odkaz ‘R’ vloží účast dané osoby na akci. View pro mobilní telefon, který vypíše data na displej, je na obrázku 6.10.

Obrázek 6.10: Výpis účastníků tábora

6.8 Controller

Controller přijímá požadavky z view a dále je předává modelu. První controller, který aplikace používá je front controller. Tato třída je umístěna v adresáři Zend/Controller/Front.php a provádí analýzu URL. Používá třídu router a zajišťuje zobrazení správné stránky. Tato třída potřebuje znát umístění bootstrapu index.php. Od něj pak provádí analýzu URL. Nastavení cesty k tomuto souboru se provádí pomocí $frontController->setBaseUrl().

Všechny controllery aplikace jsou umístěny v adresáři ./application/controllers. Toto nastavení se provádí v bootstrapu index.php. Ukázka je ve výpisu 6.6.

```
$frontController = Zend_Controller_Front::getInstance();
$frontController->setControllerDirectory('./application/controllers');
$frontController->throwExceptions(true);
```

Výpis 6.6: Nastavení cesty ke controllerům

6.8.1 Používané konvence

Aby framework věděl, který controller se má použít, je nutné dodržovat některá pravidla. Action (stránky) se sdružují do controllerů. Jestliže je např. adresa požadované stránky http://localhost/drak/akce/index, tak název controlleru je akce a název action je index. Tzn. framework bude hledat v adresáři pro controllery controller s názvem AkceController.php. Definice třídy musí mít tvar:

class AkceController extends Zend_Controller_Action

Výchozí action pro Zend Framework je speciální action nazvaná index. Pokud se nezadá název action, bude framework předpokládat, že je požadována právě action index. Výchozím controllerem pro Zend Framework je index. Zend Framework a jeho MVC struktura podporuje moduly, které umožňují sdružovat controllers dohromady.

6.8.2 Názvy controllers

V Zend Frameworku je controller třída. Její název musí mít danou formu: {název Controlleru}Controller. {název Controlleru} musí začínat velkým písmenem. Uložen je v souboru, který je pojmenován {název Controlleru}Controller.php. Všechny controllers se ukládají do adresáře určeného pro controllery. Pro názvy action se používají malá písmena. Jméno action se shodné s názvem view, ve kterém se zobrazuje výsledek action. Základní definice controlleru může vypadat např. jako na výpisu 6.7.

```
class IndexController extends Zend_Controller_Action
{
    function init()
    {
     …
    }

    function index()
    {
     …
    }

    …
}
```

Výpis 6.7: Definice controlleru

6.8.3 Inicializace controlleru

Na začátku každého controlleru probíhá inicializace. Tu provádí funkce init(). Init provádí nastavení proměnné $baseUrl, která určuje, v jaké části aplikace se pohybujeme. V této funkci se také natáhnou potřebné třídy. Dále se naincluduje třída, která určuje typ zařízení, ze kterého se k aplikaci přistupuje. Poslední činností je nastartování session. Výpis 6.8 ukazuje funkci init pro controller AkceController.php.

```
function init()
   {
     $this->view->baseUrl = $this->_request->getBaseUrl();
     Zend_Loader::loadClass('Akce');
     Zend_Loader::loadClass('AkceManager');

     require_once('application/controllers/switch/ControllerSwitch.php');
     $this->switch = new ControllerSwitch();

     session_start();
   }
```

Výpis 6.8: Funkce init()

6.8.4 Ověření uživatele

Kontrola, zda daný uživatel může přistupovat k funkcím controlleru, se provádí na začátku každého controlleru. Je zkoumán obsah proměnné $_SESSION. Pokud pozice uživatele neumožňuje přístup k této funkci systému, není její použití umožněno. Na výpisu 6.9 je kontrola přihlášeného uživatele.

```
if ((isset($_SESSION["uzivatel"])) && (($_SESSION["uzivatel"] == 'zam') ||
($_SESSION["uzivatel"] == 'adm')))
    { // uzivatel je prihlasen
      …
    }
    else
    {  //uzivatel neni prihlasen
      $this->_redirect(addSID('/index/prihlaseni/device/'.$device.'/er/1', '&'));
                return;
    }
```

Výpis 6.9: Kontrola, zda je uživatel přihlášen

Uživatel musí být přihlášen a musí být buď zaměstnanec nebo administrátor systému. Jestliže není ani zaměstnanec ani administrátor, je přesměrován na přihlašovací stránku a přes chybovou proměnnou je informován o skutečnosti, že musí být přihlášen a mít dostatečná oprávnění.

6.8.5 Funkce controlleru

Controller ze vstupních dat odstraní zakázané znaky a v případě potřeby upraví hodnoty. Jestliže jsou data v pořádku, předá je konkrétní metodě modelu. Ten provede aktualizaci a zpět controlleru vrátí výsledek operace. Ten pak controller předá do view, který informuje uživatele o úspěšné, či neúspěšné aktualizaci modelu. Informace o úspěšném provedení se přenáší v proměnné $res. Zpráva o chybě se předává v proměnné $er.

6.9 Model

Aplikační logiku aplikace tvoří model. Provádí aktualizaci dat za základě požadavků nebo vrací požadovaná data. Model je objekt (případně kolekce objektů), který je volán ve view. Všechny modely jsou umístěny v adresáři pro modely. Cesta k němu musí být nastavená v include_path. Toto nastavení je provedeno v bootstrapu index.php - viz. výpis 6.10.

```
set_include_path('.' . PATH_SEPARATOR . './library' . PATH_SEPARATOR .
'./application/models/' . PATH_SEPARATOR . get_include_path());
include "Zend/Loader.php";
```

Výpis 6.10: Nastavení cesty k modelu

Modely jsou pak snadno dostupné. V index.php také připojíme soubor Zend/Loader.php. Ten zpřístupní třídu Zend_Loader. Tato třída obsahuje funkce pro nahrání dalších Zend Framework tříd.

Zend Framework pracuje s databází pomocí třídy Zend_Db_Table. Nejprve se musí provést konfigurace třídy Zend_Db_Table. Je nutné nastavit typ databáze a přihlašovací údaje. Je více možností, jak to nastavit, v mojí aplikaci jsem použila další třídy Zend Frameworku – Zend_Config, které objektově zpřístupní veškeré konfigurační soubory. Konfigurační soubor může mít příponu ini nebo xml. Obsah konfiguračního souboru je ve výpisu 6.11.

```
[general]
db.adapter = PDO_MYSQL
db.config.host = localhost

db.config.username = jmeno
db.config.password = heslo
db.config.dbname = db
db.config.charset = utf8
db.config.collation_connection = utf8_czech_ci;
```

Výpis 6.11: Výpis souboru config.ini

Načtení třídy Zend_Config se provádí opět v bootstrap souboru (index.php). Ukázku načtení konfigurace zobrazuje výpis 6.12. Nejdříve jsou načteny potřebné třídy - Zend_Registry a Zend_Config_Ini. Třídě Zend_Config_Ini předáme jak o první parametr cestu ke konfiguračnímu souboru config.ini. Druhým parametrem je název section (v tomto příkladě general). V aplikaci načítám pouze jednu section, ale je podporováno použití poznámky u jména section, pomocí které můžeme načíst další section. Přihlašovací údaje k databázi budou sdruženy pod objekt $config->db->config. Aby byl přístup ke konfiguraci v celé aplikaci, přiřadíme pomocí třídy Zend_Registry objekt $config do objektu $registry.

```
...
Zend_Loader::loadClass('Zend_Controller_Front');
Zend_Loader::loadClass('Zend_Config_Ini');
Zend_Loader::loadClass('Zend_Registry');

// nacteni konfigurace
$config = new Zend_Config_Ini('./application/config.ini', 'general');
$registry = Zend_Registry::getInstance();
$registry->set('config', $config);
...
```

Výpis 6.12: Načtení konfigurace – index.php

Aby bylo možné použít třídu Zend_Db_Table je nejprve nutné jí předat načtené konfigurační údaje. To se opět provede ve vstupním bodu aplikace – v souboru index.php. Předání konfigurace je ve výpisu 6.13. Prvním krokem je vytvoření instance Zend_Db. Přiřazení proběhne pomocí statické funkce Zend_Db_Table:: setDefaultAdapter().

```
...
Zend_Loader::loadClass('Zend_Db');
Zend_Loader::loadClass('Zend_Db_Table');
...

// nastaveni databaze
$db = Zend_Db::factory($config->db->adapter,
$config->db->config->toArray());
Zend_Db_Table::setDefaultAdapter($db);
...
```

Výpis 6.13: Nastavení databáze – index.php

Zend_Db_Table je abstraktní třída. Jejím rozšířením vytvoříme třídu, která bude pracovat s danou tabulkou. Nastavení modelu je na výpisu 6.14. Název třídy je stejný jako název tabulky v databázi. Je potřebné nastavit dvě protected proměnné a to: $_name a $_primary. $_name obsahuje název tabulky a $_primary říká, který sloupec tabulky je primárním klíčem. Tento údaj se nemusí třídě Zend_Db_Table předávat v případě, že primární klíč tabulky se jmenuje id a má hodnotu auto_increment.

```
class Akce extends Zend_Db_Table
{
    protected $_name = 'akce';
    protected $_primary = 'kod';
}
```

Výpis 6.14: Třída Akce, která rozšiřuje třídu Zend_Db_Table

6.9.1 Přístup k datům

Vzor MVC zajišťuje, že datová vrstva, která přistupuje k datům, je nezávislá na ostatních vrstvách. Funkce přistupující k datové vrstvě jsou implementovány v samostatných souborech. Název souboru je {název}Manager.php. Obsahuje třídu {název}Manager. Tato třída má privátní proměnnou $db. Do ní jsou zkopírovány přihlašovací údaje k databázi - viz. výpis 6.15.

```
private $db;

 public function __construct($db)
 {
  $this -> db = $db;
 }
```

Výpis 6.15: Předání přihlašovacích údajů v modelu

6.9.2 Práce s databází

6.9.2.1 Zobrazení dat z db

Data z databáze získávají jednotlivé funkce třídy modelu. Výsledek operace vrací do controlleru. Příklad select funkce je ve výpisu 6.16.

Funkce vrátí účastníky minitábora podle zadaného kódu akce. Nejdříve je načtena potřebná třída. V selectu jsou spojeny dvě tabulky (ucastnici_minitabor a reg_ucastniku_minitabora). Podmínka where omezuje počet zobrazených záznamů pouze na ty účastníky, kteří se zúčastnili vybraného turnusu. Záznamy jsou primárně řazeny podle kategorie_ucastnika a sekundárně podle narozeni. Funkce Zend_Db_Table::fetchAll() vrací Zend_Db_Table_Row-set. Controller přijme vrácená data a předá je pohledu. Ve view lze získaná data vypsat pomocí foreach cyklu.

```
public function selectUcastnikyMinitabor1($kod_akce)
 {
   Zend_Loader::loadClass('Ucastniciminitabor');

   $select = $this->db->select()
   ->from(array('u' => 'ucastnici_minitabor'),
      array('r.kategorie_ucastnika', 'r.variabilni_symbol', 'u.prijmeni', 'u.jmeno',
'u.titul', 'u.narozeni'))
   ->join(array('r' => 'reg_ucastniku_minitabora'),
      'u.id = r.id_ucastnika')
   ->where('r.kod_akce = ?', $kod_akce)
   ->order(array('r.kategorie_ucastnika', 'u.narozeni'));

   return $this->db->fetchAll($select);
 }
```

Výpis 6.16: Funkce, která z db vrátí požadovaná data

6.9.2.2 Vložení dat do db

Vložení dat do databáze se také provádí přes funkce tříd modelu. Ve výpisu 6.17 je ukázáno vložení nového účastníka dětského tábora. Nejprve se natažena potřebná třída. Pak jsou do db vložena data, která funkce dostane jako vstupní proměnnou. Výsledek operace je otestován v controlleru a úspěch či neúspěch operace je předán do view.

```
public function addUcastnicitabor($data)
 {
   Zend_Loader::loadClass('Ucastnicitabor');
   $ucastnicitabor = new Ucastnicitabor();
   $ucastnicitabor->insert($data);
 }
```

Výpis 6.17: Vložení účastníka tábora do db

Velmi obdobné jsou i ostatní databázové operace - např. editace či mazání záznamů.

6.10 View

View slouží k zobrazení modelu uživateli a k získávání vstupů od uživatele. Komponenta view Zend Frameworku se jmenuje Zend_View. Komponenta umožňuje oddělit kód action od kódu stránky. I u view je nutné nastavit cestu, kde se mají příslušné pohledy hledat. Toto nastavení je možné provést opět v bootstrapu index.php. Použití Zend_View by mohlo vypadat např. jako na výpisu 6.18.

```
$view = new Zend_View();
$view->setScriptPath('/cesta/k/pohledum');
echo $view->render('view.php');
```

Výpis 6.18: Použití Zend_View

Také by bylo možné tento kód umístit do každé action. Zend Framework ale nabízí lepší variantu. Obsahuje tzv. „action helper“, který inicializuje třídu view a v action lze už přistupovat na objekt třídy view. Zend_Controller_Action_Helper_ViewRenderer inicializuje objekt view($this->view) a také vygeneruje view skript. Cestu k view nastaví objektu Zend_View na views/scripts/{název controlleru}. Tzn. pohled se bude nacházet v souboru views/scripts/{název controlleru}/{název akce}.phtml. Obsahy stránek jsou předávány do objektu Response, který sdružuje všechny HTTP hlavičky, obsah body a výjimky vzniklé použitím MVC. Front Controller poté automaticky pošle hlavičku a obsah body na konci dispatchingu. Views tvoří především HTML nebo WML kód. Vzhled je upravován pomocí css. Soubor se styly je natažen v záhlaví stránky (header.phtml). Uložen je v adresáři /public/styles/.

6.10.1 Zobrazení dat

Data jsou do view posílána z controlleru. Metoda render() očekává, že views jsou pojmenovány podle přidružených akcí s koncovkou .phtml. Soubory musí být umístěny v podadresáři, který je pojmenován podle controlleru. Předání pořádaných akcí je ve výpisu 6.19.

```
…
$this->view->aktualniakce = $akceManager -> selectAktualniAkce($start,
$zobrazit);
…
```

Výpis 6.19: Controller - předání dat od modelu do view

Data z funkce selectAktualniAkce() jsou vloženy do proměnné aktualniakce objektu view. Vypsání je zobrazeno ve výpisu 6.20.

```
…
foreach($this->aktualniakce as $akce) :
        echo $this->escape($akce['kod']); echo “<br />“;
        echo $this->escape($akce['typ']); echo “<br />“;
        echo $this->escape($akce['misto']); echo “<br />“;
        echo $this->escape($akce['zacatek']); echo “<br />“;
        echo $this->escape($akce['konec']); echo “<br />“;
        echo $this->escape($akce['rok']); echo “<br />“;
        ?>
          <a href="<?php echo(addSID("$this-
>baseUrl/akce/edit/id/$akce[id];","&"));?>">
            <img src="<?php echo $this->baseUrl;?>/public/images/b_edit.png"
alt="Upravit" class="tlacitka"></a>
          <a href="<?php echo(addSID("$this-
>baseUrl/akce/delete/id/$akce[id]","&"));?>">
            <img src="<?php echo $this->baseUrl;?>/public/images/b_drop.png"
alt="Smazat" class="tlacitka"></a>

      <?php endforeach; ?>
…
```

Výpis 6.20: Zobrazení dat v pohledu

6.11 Generování pdf

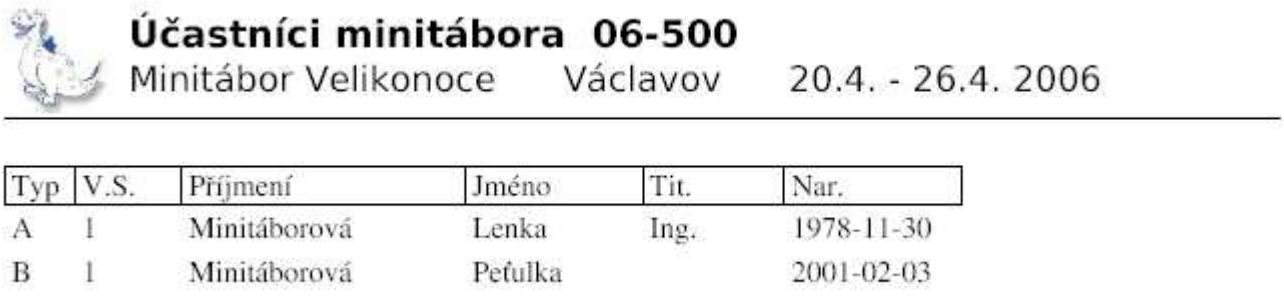
TCPDF je Open Source třída pro generování PDF dokumentů. TCPDF projekt začal v roce 2002 a je postaven na projektu FPDF. TCPDF podporuje ISO formát stránek a kódování UTF-8. Má také podporu pro kódování dokumentů. PDF stránku lze vytvořit i použitím HTML kódu. Do dokumentů lze vkládat grafické prvky a obrázky s příponou JPEG a PNG bez GD knihovny a formáty GD, GD2, GD2PART, GIF, JPEG, PNG, BMP, XBM a XPM s GD knihovnou. Podporuje vkládání odkazů a je možné používat TrueType fonty. Do všech souborů, ve kterých se mají generovat PDF, se musí includovat soubory tcpdf.php a config/lang/eng.php.

Z domovské stránky projektu TCPDF lze stáhnout dva balíčky. Jeden je pro PHP4 a druhý pro PHP5.

Tato knihovna je použita při tvorbě tiskových sestav. Aplikace v současné době poskytuje šest různých tiskových sestav ve formátu PDF. Další budou postupně doplňovány podle potřeb cestovní kanceláře Drak. Nyní jsou k dispozici sestavy:

- účastníci minitábora, řazení podle věku
- účastníci minitábora, seskupení po rodinách
- účastníci tábora, řazení podle věku
- účastníci tábora, řazení podle příjmení
- účastníci tábora po oddílech, řazení podle příjmení
- vedoucí táborů, řazení podle příjmení

U každé sestavy je možný výběr turnusu, pro který se má daná sestava vytvořit. Část výsledného pdf soubor je zobrazen na obrázku 6.11.



Obrázek 6.11: PDF sestava

6.12 Export dat pro Money

Money je ekonomický systém pro střední a malé firmy. Nabízí několik modulů, např. pro podvojné účetnictví i daňovou evidenci, adresář, fakturaci, sklady, … Vyvíjí ho firma Cígler Software.

Cestovní kancelář Drak Money využívá k evidenci mezd pro své zaměstnance – vedoucí na táborech. Aplikace vyexportuje potřebné údaje o vedoucích do textového souboru. Ten je pak k dispozici pro stažení a následný import dat do programu Money.

6.13 Úprava Zend Frameworku

Aby se v aplikaci dobře zobrazovala čeština, bylo nutné zasáhnout do kódu Zend Frameworku. Při připojení bylo nutné určit, jaké kódování se má používat (UTF 8). Do souboru config.ini se připsalo nastavení znakové sady:

db.config.charset = utf8

db.config.collation_connection = utf8_czech_ci

Výpis 6.21 ukazuje kód, který byl do Zend Frameworku doplněn.

```
protected function _connect() {
        if ($this->_connection) {
            return;
        }

        parent::_connect();

        if(isset($this->_config['charset'])) {
          $this->query("SET NAMES ".$this->_config['charset']);
          $this->query("SET CHARACTER SET ".$this->_config['charset']);
          $this->query("SET collation_connection = utf8_czech_ci");
        }

        if(isset($this->_config['collation_connection']))
          $this->query("SET collation_connection = ".$this-
>_config['collation_connection']);
    }
```

Výpis 6.21: Funkce pro nastavení znakové sady

6.14 Rozšíření systému

Systém je navržen a implementován podle požadavků cestovní kanceláře Drak. Zjištění požadavků byla věnována velká pozornost. Pokud by i přes tuto skutečnost bylo nutné v budoucnu provádět rozšíření systému, je nezbytné vytvořit model, view i controller. Rozšíření systému není nijak složité.

6.14.1 Přidání tabulky do db

První činností při rozšíření systému je vytvoření tabulky v databázi. Název tabulky musí být jedno slovo. Je vhodné použít výstižný název, aby bylo jasné, jaká data bude tabulka obsahovat. Tabulku naplníme potřebnými daty.

6.14.2 Vytvoření modelu

Dalším krokem je vytvoření modelu. Jeho jméno bude shodné s názvem tabulky v databázi. Třída bude rozšiřovat abstraktní třídu Zend_Db_Table. Do proměnné $_name se vloží název tabulky, proměnná $_primary bude obsahovat název sloupečku, který je primárním klíčem tabulky. Model se uloží do adresáře k ostatním modelům.

6.14.3 Vytvoření controlleru

Controller musí být vytvořen v adresáři pro controllery. Nezbytné je dodržení výše uvedených pravidel v pojmenovávání souboru a třídy. První funkcí třídy bude init(), která provede inicializaci controlleru. Minimálně bude obsahovat načtení třídy pro práci s modelem. Další funkce budou vytvořeny dle funkce controlleru.

6.14.4 Vytvoření view

Posledním krokem je vytvoření pohledu. Název pohledu bude totožný s názvem funkce controlleru. Uložen bude v adresáři pro views. I tady je nutné dodržet konvence ohledně pojmenovávání souborů a adresářů. Kód view bude obsahovat html nebo wml tagy. Pokud má být nově vytvořená funkce přístupná i z mobilního telefonu, je nutné vytvořit i pohled pro telefon.

6.15 Nasazení systému v praxi

Vývoj systému stále probíhá. Z tohoto důvodu není zatím aplikace přístupná v plném nasazení. Testovací verze systému je dostupná na adrese: *http://paja.mprazak.info/drak/*. Verze pro mobilní telefon běží na *http://paja.mprazak.info/drak/index/index/device/mobile.*

Na této adrese si lze v praxi vyzkoušet funkčnost systému na zkušebních datech. Přihlašovací údaje jsou:

- administrátor - admin / admin

- zaměstnanec - zam / zam

Data v systému jsou pouze demonstrativní. Nejedná se skutečné klienty cestovní kanceláře Drak.

Poslední verze systému také běží lokálně na počítači cestovní kanceláře Drak, kde systém mají nainstalovaný na dvouměsíční zkušební testování. Pokud budou s funkcí systému spokojeni, bude začátkem července 2008 spuštěna ostrá verze systému.

Samotnému nasazení systému do praxe předcházel export a čištění dat ze stávajícího systému. Bylo nutné data pečlivě vyčistit a odstranit z nich chybné a zastaralé údaje. Teprve pak bylo možné jejich natažení do nového systému.

7 Závěr

Během diplomové práce jsem navrhla a následně implementovala systém, který odděluje datovou, aplikační a zobrazovací vrstvu. Vzhledem k této skutečnosti je možné k systému přistupovat z různých platforem a to pouze změnou zobrazovací vrstvy. V současné době je implementovaný přístup k databázovému systému z počítače a mobilního telefonu.

Systém je vytvořen tak, že správa a údržba systému je snadná. Rozšíření o nové části je pro programátora nenáročné. Přidání další platformy, ze které bude možné k aplikaci přistupovat, obnáší pouze vytvoření nové zobrazovací vrstvy pro tuto platformu. Systém je přehledný a jeho ovládání je intuitivní. Celá aplikace má jednotný vzhled, což usnadňuje orientaci uživatelům a pochopení všech jeho funkcí. Menu nabízí pouze ty položky, ke kterým má uživatel přístup.

Během návrhu a implementace byla dodržena architektura návrhového vzoru Model/View/Controller. Toto rozdělení systému umožňuje poměrně snadné rozšíření aplikace i pro další typy zařízení. Aby bylo možné k aplikaci přistupovat i z dalších platforem (např. pda), stačí pouze implementovat vrstvu view pro dané zařízení. Část model a controller se nemusí nijak měnit.

K implementaci byl použit jazyk PHP, HTML, WML a relační databázový server MySQL. Aby mohl být systém použitelný v praxi, bylo nutné upravit kód adaptéru Zend Frameworku. Bez této změny nebyla plně funkční česká diakritika.

Aplikace byla vyvíjena za použití volně dostupných technologií. Systém není závislý na internetových prohlížečích.

Nevýhodou použitých technologií je skutečnost, že Zend Framework požaduje PHP 5.1.4 nebo novější a webový server podporující mod_rewrite. Na serveru musí být takové povoleny .htaccess. Najít server, který splňuje všechny tyto požadavky není úplně triviální. I přes tuto nevýhodu jsou výhody technologií převyšující. Proto byly využity právě při tvorbě aplikace.

Během diplomové práce jsem měla možnost se blíže seznámit s technologií PHP, MySQL, WML a dalšími. Také jsem měla možnost prostudovat návrhový vzor MVC a blíže jsem se seznámila s tvorbou aplikací za použití tohoto vzoru. Všechny nové zkušenosti a znalosti, které jsem získala při tvorbě reálně použitelného systému, byly důvodem, proč jsem si vybrala právě toto zadání.

Dle mého názoru jsem dosáhla stanovených cílů této práce.

Pro studium jsem použila různé dostupné zdroje. Všechny jsou uvedeny v seznamu použité literatury.

Literatura

[1] Huston, Vince: *Design Patterns*. Dokument dostupný na URL http://www.vincehuston.org/dp (prosinec 2007).

[2] Dvořák, Miloš: *Návrhové vzory (design paterns).* 16. června 2005. Dokument dostupný na URL http://objekty.vse.cz/Objekty/Vzory (prosinec 2007).

[3] Sochor, Jiří: *Návrhové vzory GoF*. 20. května 2005. Dokument dostupný na URL http://www.fi.muni.cz/~sochor/PA103/Slajdy/GoF.pdf (prosinec 2007).

[4] Holzner, Steven: *JavaScript profesionálně*, Mobil Media a.s., 2003

[5] DuBois, Paul: *MySQL profesionálně*, Mobil Media a.s., 2003

[6] Williams, Hugh E, & Lane, David: *PHP a MySQL – Vytváříme webové databázové aplikace*, Computer Press Praha, 2002

[7] Kosek, Jiří: *PHP - tvorba interaktivních internetových aplikací*, Grada, 1999

[8] Gutmans, Andi, Bakken, Stig Saether, Rethans, Derick: *Mistrovství v PHP 5*, CP Books, a.s. Brno, 2005

[9] Staníček, Petr: *CSS Kaskádové styly – Kompletní průvodce*, Computer Press Praha, 2003

[10] Kanisová, Hana, Müller, Miroslav: *UML srozumitelně*, Computer Press Brno, 2006

[11] Page-Jones, Meilir: *Základy objektově orientovaného návrhu v UML*, Grada, Praha 2001

[12] Hába, Milan: *WAP – Tvorba stránek pro mobilní zařízení*, Computer Press Praha, 2002

[13] Písek, Slavoj: *HTML a XHTML (začínáme programovat)*, Grada Publishing a.s., Praha, 2003

[14] Boumphrey, Frank, Greer, Cassandra, Raggett, Dave, Raggett, Jenny, Schnitzenbaumer, Sebastian, Wugofski, Ted: *XHTML Průvodce vývojáře*, Mobil Media a.s., Praha, 2002

[15] Gamna, Erich, Helm, Richard, Johnson, Ralph, Vlissides, John: *Návrh programů pomocí vzorů*, Grada Publishing a.s. Praha, 2003

[16] Pecinovský, Rudolf: *Návrhové vzory*, Computer Press, a.s., Brno, 2007

[17] Allen, Rob: *Getting Started with the Zend Framework*, verze 1.4.4, 2007. Dokument dostupný na URL http://akrabat.com/wp-content/uploads/getting-started-with-the-zend-framework_144.pdf (prosinec 2007).

[18] Zend Framework. *Popis developeského nástroje Zend Framework*, série článků. Dokumenty dostupné na URL http://php.interval.cz/zend-framework/ (prosinec 2007).

[19] Pichlík, Roman: *Web frameworky v Javě*, 12.10.2006. Dokument dostupný na URL http://www.sweb.cz/pichlik/archive/2006_12_10_archive.html#poznamka5 (prosinec 2007).

[20] Sun Microsystems. *Core J2EE Pattern – Front Controller*. Dokument dostupný na URL http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html (prosinec 2007).

[21] Sun Microsystems. *Core J2EE Patterns – Dispatcher View*. Dokument dostupný na URL http://java.sun.com/blueprints/corej2eepatterns/Patterns/DispatcherView.html (prosinec 2007).

[22] Kolesnikov, Pavel: *Java na web serveru nejen pro starší a pokročilé*. 23.1.2003. Dokument dostupný na URL http://interval.cz/clanky/java-na-web-serveru-nejen-pro-starsi-a-pokrocile/ (prosinec 2007).

[23] WIKIPEDIE. *Informační systém.* 5. prosince 2007. Dokument dostupný na URL http://cs.wikipedia.org/wiki/Informa%C4%8Dn%C3%AD_syst%C3%A9m (prosinec 2007).

[24] WIKIPEDIE. *Návrhový vzor*. 2007. Dokument dostupný na URL http://cs.wikipedia.org/wiki/N%C3%A1vrhov%C3%BD_vzor (prosinec 2007).

[25] WIKIPEDIE. *MVC.* 5. prosince 2007. Dokument dostupný na URL http://cs.wikipedia.org/wiki/MVC (prosinec 2007).

[26] WIKIPEDIE. Framework. 5. prosince 2007. Dokument dostupný na URL http://cs.wikipedia.org/wiki/Framework (prosinec 2007).

[27] Schlossnagle, George: *Pokročilé programování v PHP5*, Zoner Press, Brno, 2004

[28]Naramore, Elizabeth, Gerner, Jason, Scouarner, Yann Le, Stolz, Jeremy, Glass, Michael K.: Vytváříme webové aplikace v *PHP5, MySQL a Apache*, Computer Press Brno, 2006

[29] Welling, Luke, Thompsonová, Laura: *PHP a MySQL rozvoj webových aplikací*, Softpress, Praha, 2004

[30] *Menu.* Dokument dostupný na URL http://css.interval.cz/menu/ (duben 2008).

[31] Filbar. *PHP-generujeme PDF*. 13. září 2007. Dokument dostupný na URL http://www.linpro.cz/index2.php?option=com_content&do_pdf=1&id=156 (duben 2008).

[32] *PHP class for generating PDF documents.* Dokument dostupný na URL http://www.tecnick.com/public/code/cp_dpage.php?aiocp_dp=tcpdf (duben 2008).

Seznam použitých zkratek

API - Application Programming Interface

CK - cestovní kacelář

CSS - Cascading Style Sheets

GoF - Gang of four

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

IS - informační systém

MVC - Model View Controller

PHP - Personal Home Page

SGML - Standard Generalized Markup Language

SQL - Structured Query Language

UML - Unifed Modeling Language

URL - Unique Resource Locator

XHTML - eXtensible HyperText Markup Language

XML - Extensible Markup Language

WML - Wireless Markup Language

Seznam obrázků

Seznam příloh

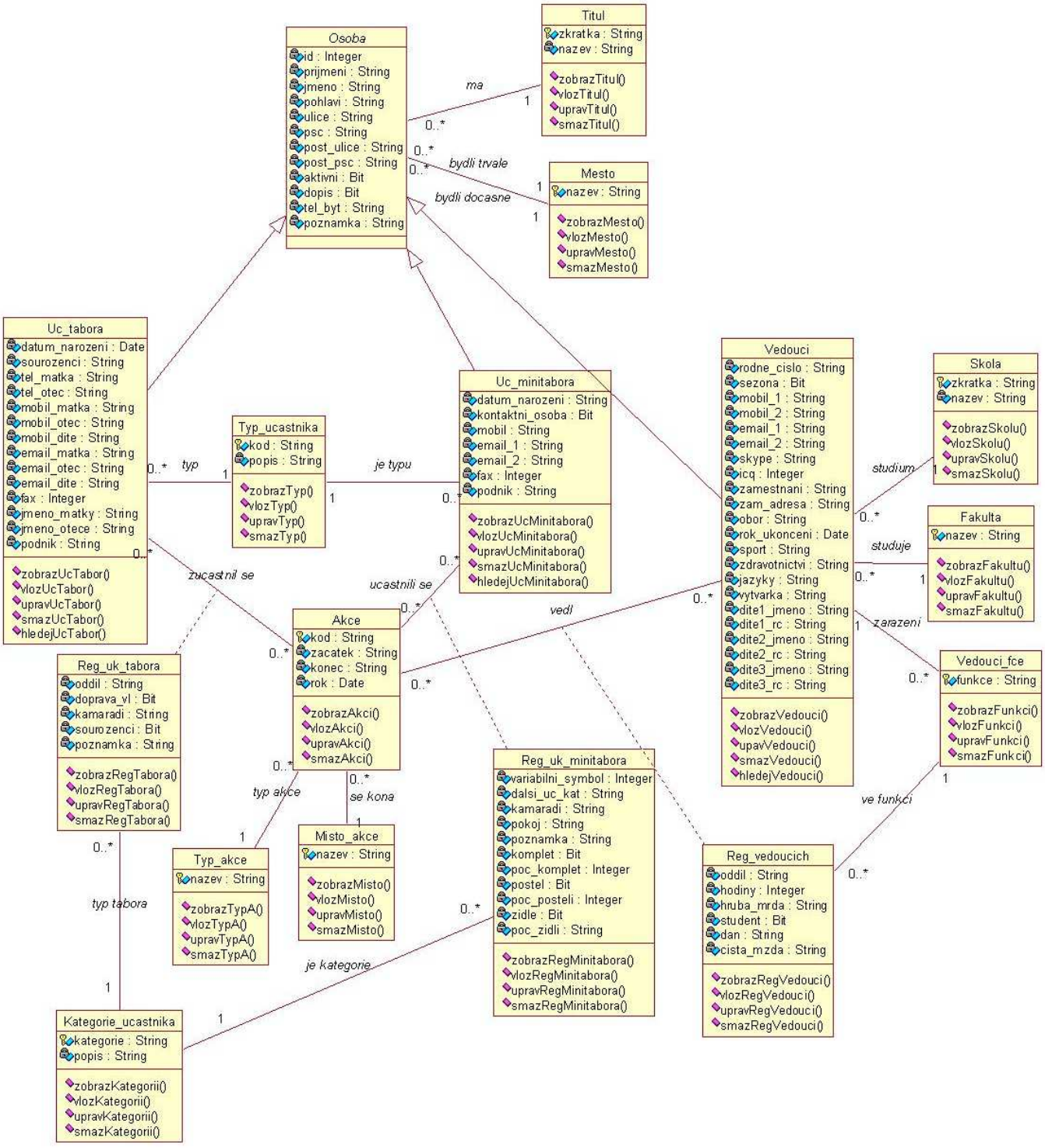
Příloha 1. Konceptuální diagram tříd - v plném rozlišení

Příloha 2. Hodnocení majitele cestovní kanceláře Drak

Příloha 3. CD-ROM nosič obsahující:

- /text - xhroma03.pdf - text této práce v elektronické podobě
- /drak - zdrojové kódy systému
- /data - skript pro vytvoření a smazání databáze
- /openwave - instalační soubor simulátoru Openwave SDK 6.1, programu pro prohlížení stránek pro mobilní telefony

Příloha 1


Osoba
id : Integer
prijmeni : String
jmeno : String
pohlavi : String
ulice : String
psc : String
post_ulice : String
post_psc : String
aktivni : Bit
dopis : Bit
tel_byt : String
poznamka : String
Titul
zkratka : String
nazev : String
zobrazTitul()
vlozTitul()
upravTitul()
smazTitul()
ma
Mesto
nazev : String
zobrazMesto()
vlozMesto()
upravMesto()
smazMesto()
bydli trvale
bydli docasne
Uc_tabora
datum_narozeni : Date
sourozenci : String
tel_matka : String
tel_otec : String
mobil_matka : String
mobil_otec : String
mobil_dite : String
email_matka : String
email_otec : String
email_dite : String
fax : Integer
jmeno_matky : String
jmeno_otece : String
podnik : String
zobrazUcTabor()
vlozUcTabor()
upravUcTabor()
smazUcTabor()
hledejUcTabor()
typ
Typ_ucastnika
kod : String
popis : String
zobrazTyp()
vlozTyp()
upravTyp()
smazTyp()
je typu
Uc_minitabora
datum_narozeni : String
kontaktni_osoba : Bit
mobil : String
email_1 : String
email_2 : String
fax : Integer
podnik : String
zobrazUcMinitabora()
vlozUcMinitabora()
upravUcMinitabora()
smazUcMinitabora()
hledejUcMinitabora()
zucastnil se
ucastnili se
Vedouci
rodne_cislo : String
sezona : Bit
mobil_1 : String
mobil_2 : String
email_1 : String
email_2 : String
skype : String
icq : Integer
zamestnani : String
zam_adresa : String
obor : String
rok_ukonceni : Date
sport : String
zdravotnictvi : String
jazyky : String
vytvarka : String
dite1_jmeno : String
dite1_rc : String
dite2_jmeno : String
dite2_rc : String
dite3_jmeno : String
dite3_rc : String
zobrazVedouci()
vlozVedouci()
upavVedouci()
smazVedouci()
hledejVedouci()
vedl
studium
Skola
zkratka : String
nazev : String
zobrazSkolu()
vlozSkolu()
upravSkolu()
smazSkolu()
studuje
Fakulta
nazev : String
zobrazFakultu()
vlozFakultu()
upravFakultu()
smazFakultu()
zarazeni
Vedouci_fce
funkce : String
zobrazFunkci()
vlozFunkci()
upravFunkci()
smazFunkci()
ve funkci
Akce
kod : String
zacatek : String
konec : String
rok : Date
zobrazAkci()
vlozAkci()
upravAkci()
smazAkci()
Reg_uk_tabora
oddil : String
doprava_vl : Bit
kamaradi : String
sourozenci : Bit
poznamka : String
zobrazRegTabora()
vlozRegTabora()
upravRegTabora()
smazRegTabora()
typ akce
se kona
Typ_akce
nazev : String
zobrazTypA()
vlozTypA()
upravTypA()
smazTypA()
Misto_akce
nazev : String
zobrazMisto()
vlozMisto()
upravMisto()
smazMisto()
typ tabora
Reg_uk_minitabora
variabilni_symbol : Integer
dalsi_uc_kat : String
kamaradi : String
pokoj : String
poznamka : String
komplet : Bit
poc_komplet : Integer
postel : Bit
poc_posteli : Integer
zidle : Bit
poc_zidli : String
zobrazRegMinitabora()
vlozRegMinitabora()
upravRegMinitabora()
smazRegMinitabora()
Reg_vedoucich
oddil : String
hodiny : Integer
hruba_mrda : String
student : Bit
dan : String
cista_mzda : String
zobrazRegVedouci()
vlozRegVedouci()
upravRegVedouci()
smazRegVedouci()
je kategorie
Kategorie_ucastnika
kategorie : String
popis : String
zobrazKategorii()
vlozKategorii()
upravKategorii()
smazKategorii()

Příloha 2

CK DRAK - dětské tábory
Božetěchova 36
612 00 Brno
IČO: 40973034

V Brně, 1. května 2008

Hodnocení spolupráce a výsledného systému

Před zahájením projektu jsme měli se slečnou Hromádkovou informační schůzku, kde jsme jí nastínili naše požadavky a potřeby. Během samotného vývoje jsme se pravidelně scházeli a to hlavně v počáteční fázi. Při těchto schůzkách jsme řešili naše požadavky na systém. Potřeby byly prezentovány i za použití stávajícího systému.

Při tvorbě nás slečna Hromádková průběžně informovala o stavu práce a postupně jsme upřesňovali naše požadavky. Před samotným předáním systému bylo provedeno školení, na kterém jsme byli seznámeni s aplikací a bylo nám předvedeno ovládání systému. Nyní máme systém nainstalovaný lokálně na našich počítačích. Provádíme jeho testování a podrobně se seznamujeme s ovládáním systému.

Vzhledem k naší spokojenosti předpokládáme, že navázaná spolupráce bude i nadále pokračovat. Uvedení do ostrého provozu je plánované na začátek července 2008. Doufáme, že tím naše spolupráce se slečnou Hromádkovou neskončí, a v případě potřeby bude možné se na ni obrátit s žádostí o rozšíření funkčnosti systému.

PaedDr. Pavel Vaverka
majitel CK Drak