VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM NA PODPORU TRÉNINKOVÝCH AKTIVIT

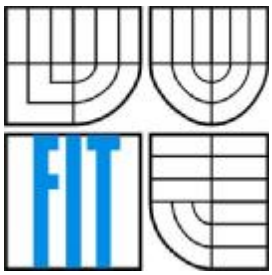
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVLÍNA SMÉKALOVÁ

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM NA PODPORU TRÉNINKOVÝCH AKTIVIT

INFORMATION SYSTEM FOR THE SUPPORT OF THE TRAINING ACTIVITIES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PAVLÍNA SMÉKALOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL OČENÁŠEK
SUPERVISOR

BRNO 2007

Zadání diplomové práce

Řešitel: **Smékalová Pavlína, Bc.**
Obor: Informační systémy
Téma: **Informační systém pro podporu tréninkových aktivit**
Kategorie: Web

Pokyny:

1. Seznamte se s technikami tvorby informačních systému v prostředí internetu.
2. Analyzujte navrhněte požadavky na IS pro podporu tréninkových aktivit. Při návrhu použijte UML. IS koncipujte jako multiuživatelský s různými rolemi.
3. Do návrhu zahrňte mj. i následující moduly: tréninkové aktivity, denní činnosti, výživa, tréninkový plán a plány výživy (automaticky / na doporučení), sledování zdravotních funkcí a parametrů.
4. IS implementujte a zahrňte i statistiky výsledků, včetně predikce fyzických a zdravotních parametrů při plnění doporučených plánů.
5. Implementujte jednoduchý import a export tabulkových hodnot tréninkových aktivit, denních činností a prvků výživy.
6. Zhodnoťte získané zkušenosti a diskutujte možnosti dalšího rozšíření.

Literatura:

- Dle doporučení vedoucího práce.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 - 3.

Podrobné závazné pokyny pro vypracování siplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz

Vedoucí: **Očenášek Pavel, Ing.,** UIFS FIT VUT

Datum zadání: 28. února 2007
Datum odevzdání: 22. května 2007

Licenční smlouva

Licenční smlouva v kompletním znění je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Výňatek z licenční smlouvy:

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zvěřejněním díla v databázi přístupné v mezinárodní síti:
   - □ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   z důvodu utajení v něm obsažených informací.
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Abstrakt

Tato diplomová práce se zabývá analýzou a tvorbou informačního systému pro podporu tréninkových aktivit. Aby měli uživatelé systém vždy k dispozici, je vytvořen jako webová aplikace. Je vytvořen informační systém, který umožňuje sledovat výživový, sportovní a silový program sportovce. Je navržen tak, aby každý sportovec mohl mít přiřazeného svého dietologa a trenéra, kteří se starají o jeho jídelníček a tréninkové aktivity. Sportovec může okamžitě sledovat své úspěchy a tím se také lépe přibližovat ke svým cílům.

Klíčová slova

Informační systém, PHP, MySQL, UML, databáze, databázový model

Abstract

This Diploma thesis describes the process of analysis and design of an information system supporting training activities. The system is based on the web pages. It is creating an information system, that would allow monitoring the nutritional plan, strength training and sports activities of an athlete. The system allows each athlete to be assigned a dietitian and a coach, who take care of his/her nutritional plan and training. The athlete has immediate feedback atd of his success and has better conditions to meet his ambitions.

Keywords

the information system, PHP, MySQL, UML, the database, the model of database

Citace

Pavlína Smékalová: Informační systém na podporu tréninkových aktivit, diplomová práce, Brno, FIT VUT v Brně, 2007

Informační systém pro podporu tréninkových aktivit

Prohlášení

Prohlašuji, že jsem tuto semestrální práci vypracovala samostatně pod vedením Ing. Pavla Očenáška. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……
Pavlína Smékalová
21.5.2007

Poděkování

Děkuji vedoucímu své práce Ing. Pavlu Očenáškovi za poskytnuté rady a odbornou pomoc.

Obsah

1 Úvod

Dnešní společnost je řízena kultem těla. Ideálním, krásným člověkem je jen jedinec štíhlý, mající trénovanou atletickou postavu. Přesto je mnoho lidí, kteří trpí nadváhou a neví, jak proti ní bojovat. Ne však každý má dostatek pevné vůle a sebezapření k tomu, aby dodržoval zásady správné výživy a chodil pravidelně cvičit. Leckdy by právě těmto lidem pomohl jen určitý druh dohledu. Každý je však individualita, a proto musí mít sobě přizpůsobenou stravu a provozovat aktivity, které jsou vhodné zvláště pro něj. Proto je vytvořen tento informační systém pokrývající problematiku výživového, sportovního a silového programu jednotlivce. Lidé cítí mnohem větší motivaci, pokud mají k dispozici systém, který monitoruje jejich úspěchy a neúspěchy. Aby měli uživatelé systém vždy k dispozici, je vytvořen jako webová aplikace.

Úvodní kapitolu této práce tvoří teoretické poznatky o používaných technologiích. Pro potřebu návrhu jsou zmíněny možnosti vizuálního modelování systému pomocí jazyka UML. Následuje stručné seznámení s jazykem PHP a databází MySQL využívané při implementaci. Také jsou popsány další techniky používané při tvorbě informačního systému (XHTML, CSS, Ajax a Pear).

Následující tři kapitoly pojednávají o samotném návrhu systému. Nejdříve jsou popsány požadavky kladené na realizovaný systém. Na základě těchto požadavků je sestaven diagram případů použití z pohledu rolí všech uživatelů přistupujících k systému. Poté je uvedeno schéma databáze a popis jednotlivých entit.

Další kapitola poskytuje pohled na již existující systémy zabývající se touto tématikou. Jsou zmíněny klady a zápory těchto systémů.

Nejrozšířenější kapitolou je popis implementace. Tato kapitola je strukturovaná do několika částí. Nejdříve je zmíněna struktura aplikace, poté již následuje samotný popis naimplementovaných tříd. Je to popis tříd pro práci s daty a třídy uživatelského rozhraní. Dále se práce zabývá popisem používání šablon, tvorbou grafů a zabezpečením samotného systému.

V posledních dvou kapitolách jsou zmíněny získané zkušenosti při vývoji aplikace a její možná rozšíření.

2 Použité technologie

Před vlastní implementací bylo nutné se seznámit s různými nástroji, které jsou používány pro tvorbu informačních systémů v prostředí Internetu. Bylo důležité si správně vybrat, aby výsledný produkt byl co nejlepší. V této kapitole jsou zmíněny jednotlivé technologie, které byly použity pro řešení této práce. Jsou zde stručně popsány a je vysvětleno, proč byly použity právě tyto prostředky pro implementaci.

Systém byl vytvořen jako internetová aplikace, která běží na serveru s nainstalovaným PHP (viz. kapitola 2.1). Tento jazyk dynamicky generuje WWW stránky. Data jsou uložena v databázi, je využíván databázový systém MySQL (viz. kapitola 2.2).

Pro návrh aplikace je použit nástroj UML (viz. kapitola 2.10), konkrétně Diagram použití a ER diagram.

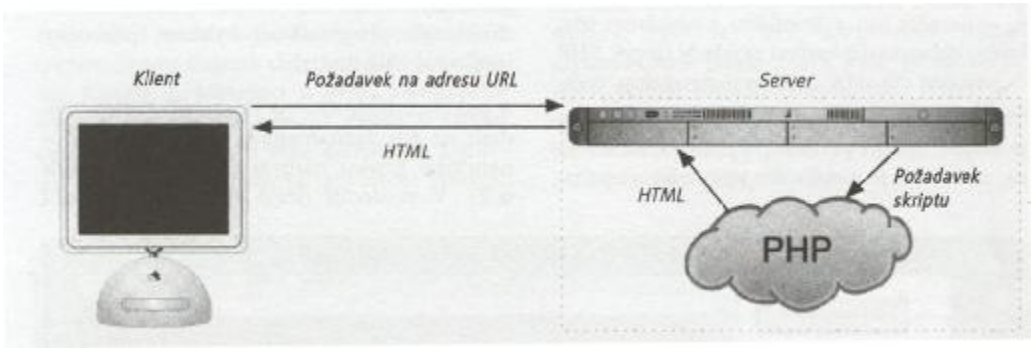
Jelikož se jedná o webovou aplikaci je používáno XHMTL (kapitola 2.3)a CSS (kapitola 2.4), pro zvětšení dynamičnosti stránek bylo použito JavaScriptu (kapitola 2.5) a Ajaxu (kapitola 2.6).

Také je zde popsán framework Pear (kapitola 2.8)a šablonovací systém Smarty (kapitola 2.9), které jsou v práci používány.

2.1 PHP

2.1.1 Co je to PHP?

PHP (rekurzivní akronym pro „PHP: Hypertext Preprocessor“) je vloženým skriptovacím jazykem, což jej přesně popisuje a vystihuje. Vložený v tomto významu znamená, že jej můžeme interpretovat přímo v kódu HTML, takže programování dynamických webových stránek je dostupnější. Skriptovací jazyk přímo reaguje na určité změny, události, na rozdíl od jazyka programovacího. PHP je nezávislé na platformě a je určen pro servery, takže je vše závislé jen na serveru a nikoli na klientovi. Další nespornou výhodou je, že je to volně šířitelný software. Obrázek 1 nám objasňuje, jakým způsobem odesílá PHP při požadavku webového prohlížeče příslušná data.



Obrázek 1 Vztah technologie PHP k modelu klient-server při požadavku na webovou stránku

2.1.2 Proč právě PHP?

Jedná se o poměrně malou aplikaci a v dnešní době, kdy jsou stále populárnější webové stránky, jsem se rozhodla o umístění aplikace na ně, čímž se informační systém stane dostupným pro každého téměř odkudkoli. Tyto stránky jsou tvořeny dynamicky a právě pro dynamickou tvorbu stránek je PHP lepší, rychlejší a jednodušší než jiné programovací jazyky. Druhou alternativou, která by mohla splňovat tyto požadavky je .NET, který ale pro tento projekt není zcela vhodný, protože je výhodnější na použití velkých projektů. Také velkou nevýhodou je jeho cena, není šířen zdarma jako PHP a pracuje pouze na platformách MS Windows. Vlastnostem tohoto projektu vyhovovalo právě PHP, které je primárně zaměřeno na webové aplikace. Jeho integrace s WWW stránkami je dobrá, stejně tak jako podpora databází.

PHP je Open Source software, takže je k dispozici zdarma. Díky tomu je stále vyvíjen, udržován a zdokonalován. Po všech inovacích stroje Zend pro zpracování skriptů je jejich běh velmi rychlý a všechny komponenty běží v paměťovém prostoru stroje PHP, což je rozdíl od jiných skriptovacích systémů, kde jednotlivé komponenty spadají do samostatných modulů. Je flexibilní vůči platformě i operačnímu systému.

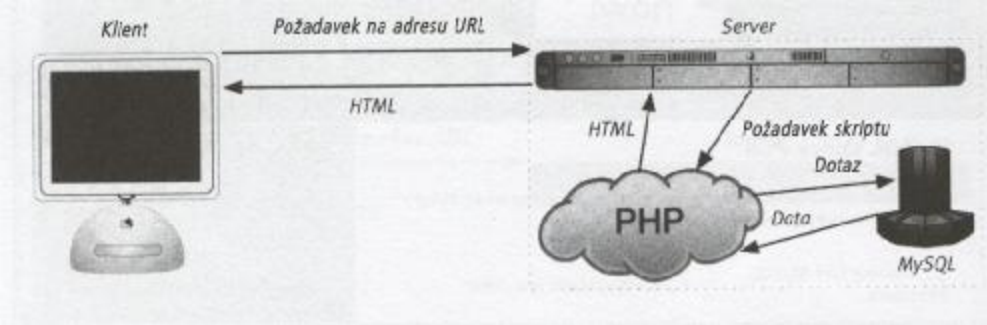
2.2 MySQL

2.2.1 Co je to MySQL?

MySQL je Open source relační databázový systém, tedy systém spolupracující s tabulkami. Řádky zastupují jednotlivé záznamy, ve sloupcích jsou pak obsaženy jednotlivé hodnoty. Tento způsob zápisu je ideální pro různé seznamy, ceníky, tedy jakákoli data zapisovaná pomocí tabulky.

Velkou výhodou MySQL je rychlost a jednoduchost. Běží totiž jako samostatný server a k práci s daty se většinou využívá skriptování na straně serveru pomocí PHP, nebo ASP. Proto tento databázový systém našel velké uplatnění právě na internetu. MySQL je lehce přenositelný a pracuje na komerčních i nekomerčních serverech a může se využívat zdarma. Na obrázku 2 je již začleněna

databáze MySQL, takže data generována strojem PHP jsou načítána na základě požadavku dotazem přímo z databáze.



Obrázek 2 Vztah technologie PHP k modelu klient-server při požadavku na webovou stránku s podporou MySQL

2.2.2 Proč právě MySQL?

Vzhledem k tomu, že v mém projektu bylo potřeba využít databázový systém, měla jsem na výběr z několika možností. Důležitým faktorem bylo, aby byla vůbec možná spolupráce s PHP. Jelikož se jedná o nekomerční projekt, tak výběr padl na freeware MySQL.

Další databázový server spolupracující s PHP je PostreSQL, který je také zdarma. Dalšími možnými SQL servery, které podporuje PHP, jsou MS SQL Server, Informix, Oracle, Sybase aj. Tyto jsou již placené, takže jsem je z tohoto důvodu nevyužila.

MySQL má několik výhod. Je rychlý, v dnešní době je patrně nejrychlejší databází, která se dá využít. Má snadné používání – je výkonný, ale relativně jednoduchý a jeho konfigurace a správa jsou jednodušší, než je tomu u velkých systémů.

2.3 XHTML

XHTML (eXtensible Hyper Text Markup Language) je značkovací jazyk pro vytváření vzhledu dokumentu. Jednotlivým značkám je přiřazena sémantika hypertextového dokumentu pro webové prostředí. XHTML je nástupcem HTML 4, byl zde kladen důraz na zapracování XML a oproti HMTL 4 je přehlednější, má jasnější strukturu, je sémantičtější a celkově lepší pro orientaci. V současné době byla vydána verze XHTML 2. Definicí jazyka pro XHTML je DTD. Jsou používány tři typy DTD, nejpřísnější forma, vylučuje ty rysy, které by se neměly používat a konsorcium W3C nedoporučuje jejich používání. Tato striktní forma je použita také při implementaci této diplomové práce. Druhou formou je tzv. přechodná forma (transitional) sloužící pro přechodnou kompatibilitu a pro stránky, které používají rámce slouží forma frameset.

2.4 CSS

CSS (Cascading style sheets) neboli kaskádové styly je jednoduchý mechanismus pro vytvoření vzhledu a jednotného stylu webového dokumentu. První návrh normy vyšel v roce 1994, první verze o dva roky poté, o čtyři roky později vyšla specifikace CSS 2 a v současné době se pracuje na verzi CSS 3.

CSS se používá k formátování obsahu HTML, XHMTL a XML stránek. Styly přesně říkají, jak který dokument bude vypadat. Umožňuje jednotné formátování, definuje jednotný vzhled elementu pro celý dokument, ale také samozřejmě můžeme nadefinovat styl pro jediný element, který se vyskytuje v dokumentu pouze jednou. Tímto formátováním se styl stává přehlednější, jasnější a hlavně v případě změny se provede pouze na jednom místě a nemusí se tedy měnit všude tam, kde je prvek použit, což může být velmi komplikované.

Pro jednotlivé elementy se pokaždé nadefinuje jednotný styl, definice se skládá ze dvou částí – selektoru a deklarace. Selektor je název elementu, pro který dané pravidlo platí a deklarace jaké vlastnosti pro něj platí. V deklaraci se uvádí vlastnost a hodnota a je uzavřena do hranatých závorek.

V CSS existuje dědičnost. Jestliže určitý element nemá svoji vlastnost nadefinovanou, dědí ji od svého nadřazeného elementu.

2.5 JavaScript

Javascript je interpretovaný, objektový jazyk. Což znamená, že se nemusí kompilovat respektive, že využívá objektů prohlížeče a zabudovaných objektů. Je používán v mnoha WWW stránkách pro vylepšení vzhledu, ověřování formulářů, animaci, efekty obrázků a v mnoha jiných funkcích. Klientská verze tohoto jazyka pracuje se strukturou DOM, používá události prohlížeče, dokument, okno apod. Tyto vlastnosti podporuje většina dnešních prohlížečů. Javascript vkládá interaktivitu do HTML stránek a může být přímo psán do kódu HTML nebo do externího souboru s příponou .js.

Stránky s Javascriptem jsou dynamické, ihned reagující na změny uživatelů, například kliknutí myši, stisknutí nebo uvolnění klávesy. Může také být použit pro určení uživatelova prohlížeče a tomu také přizpůsobit kód, který se posílá prohlížeči. Také může být využit pro ukládání a pracování s cookie.

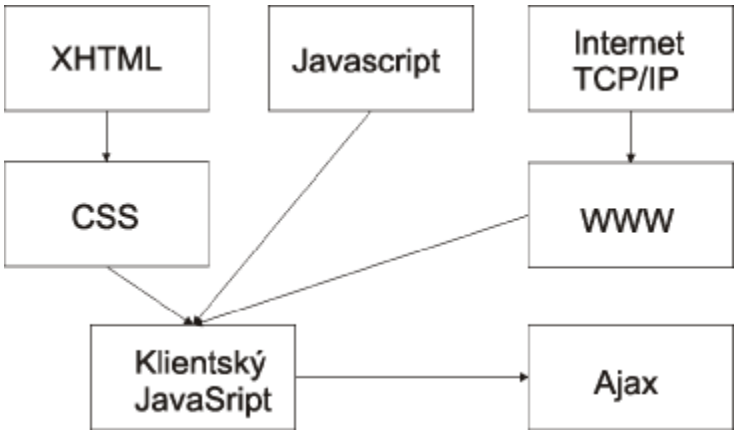
2.6 Sajax

Neboli Simply Ajax toolkit je nástroj pro programování webových stránek, který používá Ajax framework, také znám jako XMLHttpRequest. Objekt XMLHttpRequest byl přímo navržen pro komunikaci skriptů se serverem. Ajax je schopen lehce volat ASP, Cold Fusion, Io, Lua, Perl, PHP, Python nebo Ruby funkce z webových stránek přes javascript bez předchozí aktualizace prohlížeče.

Sajax volá vzdálené procedury v PHP přes javascript, takže je možné s nimi na stránce pracovat jako s ostatními funkcemi javascriptu. Ajax je využíván hlavně u WWW stránek, které obsahují formuláře. Tímto způsobem se stránky stávají více interaktivní a urychlují uživateli práci, protože nemusí být načítána celá stránka, ale jen konkrétní část formuláře, která se změnila.

2.7 Internetové aplikace

Pro přehlednost zde uvádím obrázek internetových aplikací, které jsou použity v projektu. Jejich bližší popis je uveden v předchozích kapitolách.



obrázek 3 Vztahy použitých internetových aplikací

2.8 Pear

2.8.1 Co je Pear?

Pear je zkratka pro PHP Extension and Application Repository a jsou to strukturované knihovny open-source kódu pro PHP. Kód v Pearu je strukturován do balíčků. Každý balíček je samostatný projekt s vlastním vyvíjejícím týmem, verzí, cyklem vydání, dokumentací a definovanou relací k ostatním balíčkům. V současné době archiv balíčků obsahuje 250 knihoven a rozšíření, která jsou tematický roztříděny do několika kategorií.

V mé práci jsem využila dva balíčky, první MDB2 pro práci s databází a druhý Image_graph pro vykreslování grafů. Tyto dva balíčky jsou blíže popsány v následujících dvou podkapitolách.

2.8.1.1 MDB2

Tento balíček se používá pro přístup k databázím. Jde o abstraktní databázovou vrstvu poskytující jednotné aplikační rozhraní pro většinu databází podporovaných jazykem PHP, tedy i MySQL, která je používána v tomto projektu. Samotný balíček PEAR MDB2 je napsán v jazyce PHP a obsahuje ovladače pro většinu databázových rozšíření jazyka PHP.

Tento balíček abstrahuje různé funkce, jsou to např. databázová spojení, načítání výsledků, vázání vstupných proměnných, oznamování chyb apod.

2.8.1.2 Image_graph

Image_graph je balíčkem pro zobrazování numerických dat v podobě grafů a lze pomocí něj lehce zobrazovat 14 druhů grafů: sloupcový, pruhový, spojnicový, výsečový, bodový, plošný, paprskový aj. Uživatel může grafy snadno měnit a upravovat dle svých představ, ať už se jedná o barvy, písmo, velikosti a typy. Je možné vedle sebe umisťovat několik grafů, zobrazovat za grafy obrázky, či si jinak s nimi vyhrát. Je také možný několikerý výstup obrázku do souboru: jpg, png, pdf,… Vše je jednoduché a lehce dostupné pomocí tříd tohoto balíčku.

2.8.2 Proč právě Pear?

Pro práci s databázemi bylo možné také použít ADODB, což je knihovna, která v PHP zjednodušuje mnohé úkoly, které se vztahují k databázím. Poskytuje obdobné funkce jako Pear MDB2, ovšem obsahuje také funkce, které jsou schopny zpracovávat mnohem komplikovanější databázové úkoly (automatické stránkování výsledků apod.). Pear MDB2 má ovšem lépe definovanou objektovou strukturu. A mnohem lépe se integruje s dalšími moduly Pear, protože patří do jádra knihovny Pear. Vzhledem k tomu, že ve své práci potřebuji pracovat s grafy, které obsahuje Pear – tedy balíček Image_graph, vybrala jsem si právě pro tuto práci framework Pear.

2.9 Smarty

2.9.1 Šablony obecně

V dnešní době je cílem většiny aplikací oddělení aplikační logiky od logiky prezentační. Ve webovém prostředí je toto prováděno pomocí šablon. V systému šablon je veškerá logika zobrazení obsažena v šabloně. Samotný kód, který neobsahuje žádnou prezentační logiku, zpracovává požadavek, provádí vše potřebné a výsledná data poté předá šabloně, která je formátuje pro zobrazení.

Toto oddělení dovoluje snadno modifikovat buď aplikační logiku nebo vnější vzhled stránek a navzájem se nikterak neovlivňují. Samotný kód se stává přehlednější, protože musí být jasně rozhodnuto, co je prezentační logika a co aplikační.

2.9.2 Co je Smarty?

Smarty je šablonovací systém vytvoření pomocí jazyka PHP. Smarty používá speciální značky v souborech s příponou *.tpl, které jsou poté kompilovány do cachovaného PHP skriptu. I když je Smarty určen hlavně k oddělení prezentační logiky od aplikační není omezen na tvorbu šablon jen

pomocí HTML značek. Mohou také ale být použity řídící struktury, cykly, funkce pro práce s řetězci, časem, aj. Při zpracování je každá šablona zkompilována, což je běžný PHP skript, a je uložena do speciálního adresáře. Při jejím opětovném použití není již znovu celá překládána, ale je použita její zkompilovaná verze, což je mnohem rychlejší a efektivnější. Obměnou kompilování šablon je cachování obsahu, což je podobný princip, nejsou ale vytvářeny PHP skripty, ale HTML.

Smarty používá pracovní adresáře. Je to adresář ve kterém jsou umístěny samotné šablony, dále adresář pro kompilované šablony a pro cache, které již byly zmíněny výše. Posledním, ovšem také nepovinným adresářem, je adresář pro konfigurační soubory.

2.10 UML

2.10.1 Co je UML?

Modelovací jazyk UML (Unified Modeling Language) je souhrn grafických notací k vyjádření analytických a návrhových modelů. Pomocí stejné syntaxe můžeme modelovat jednoduché i složité aplikace. I zadavatel lehce pochopí tento model, a proto se snáze ujasňují požadavky na daný systém. UML je také jazyk pro vizualizaci, specifikaci, stavbu a dokumentaci softwarových systémů.

2.10.2 UML pohledy na systém

Model v UML se skládá z různých diagramů a každý má jiný pohled na různé části systému. Žádný dvourozměrný diagram nemůže zachytit komplexní aplikaci vcelku, a proto se vždy soustřeďuje jen na jeden pohled. Jazyk UML rozeznává pět základních pohledů na systém.

- Pohled případů užití vyjadřují základní požadavky kladené na systém, všechny další pohledy jsou již vymezeny pohledem případů užití.
- Logický pohled se zabývá statickými vztahy z problémové domény zadavatele.
- Procesní pohled se soustřeďuje na chování systému, které musí splňovat požadavky a omezení z případů užití, jež jsou kladeny na průběh procesů. Jedná se vlastně o procesně orientovaný doplněk logického pohledu.
- Implementační pohled – fyzické rozdělení aplikace na samostatné komponenty a jejich závislosti.
- Pohled nasazení zachycuje komponenty na množinu fyzických výpočetních uzlů v cílovém prostředí.

Pohledy jsou blíže určeny v různých typech diagramů. Ty můžeme rozdělit do dvou základních kategorií – statické, které vyjadřují strukturu systému, a dynamické, které vyjadřují chování systému:

- Modely ukazující statickou strukturu systému – diagram tříd, diagram případů užití, objektové diagramy, diagramy komponent a diagram rozmístění

- Modely ukazující dynamické chování systému – diagram aktivit, sekvenční diagram, diagramy spolupráce a aktivit

Dále bude navržen Diagram případu užití – pohled případů užití a návrh databáze pomocí Datového modelování (ER model) – logický pohled.

3 Podobné systémy

3.1 Úvodem

V této části bych ráda zmínila stručný přehled dostupných systémů stejného zaměření, jako je má práce a uvedla bych některé jejich vlastnosti, přednosti a nedostatky.

Ve většině případů jsou to systémy, které se touto tematikou zabývají jen částečně, tedy jen z některého pohledu. Nebo jsou to jen aplikace, které se nevyskytují na Internetu, ale musí být přímo nainstalovány na pevný disk počítače. Těmito systémy jsem se nechala inspirovat a snažila se vyhnout nedostatkům, které tyto aplikace mají.

3.2 Fitlinie

Je to systém, který umožní sledovat veškerý příjem i výdej energie. Poskytuje mnoho tréninků, plánů, cvičení a aktivit, tedy vše co se týká výdeje energie, ale také veškeré příjmy energie. Obsahuje velkou databázi potravin, receptů a jídelníčků. Na základě vložených údajů přehledně zobrazuje grafy a statistiky.

Touto aplikací je můj systém nejvíce inspirován, hlavní rozdíly mezi těmito systémy je umístění. Můj informační systém bude totiž umístěn na webových stránkách, kdežto tuto aplikaci si každý uživatel instaluje samostatně na svůj pevný disk.

Aplikaci lze stáhnout [4].

3.3 Fitbook Linie Alfa 2

V tomto případě se nejedná přímo o informační systém, ale spíše o takovou elektronickou knihu. Je to učební materiál, kde se informace pouze hledají, ale již se tam žádné nezadávají, takže nelze přímo porovnávat svoje vlastní výsledky, generovat postupy, grafy apod. Jedná se tedy hlavně o texty, obrázky a tabulky, ve kterých se listuje. Obsahuje již ověřené vědecké postupy pro zlepšení kondice, postavy a zdraví. A dále obsahuje potraviny a jejich složení a energie.

Veškeré informace, registrace a program je možné najít [12].

3.4 Fitář

Tento program slouží pouze pro počítání příjmu a výdeje energie. Je to jednoduchý systém, kde se zadává výběrem, co bylo snědeno a jaký pohyb byl vykonán. Dá se vybírat jen ze zadaných údajů,

vůbec tedy není možné přidávat svoje vlastní aktivity, ani potraviny. Není zde jakýkoli tréninkový ani pohybový plán. Mimo jiné tato aplikace ještě obsahuje diář, kalendář a zápisník. Aplikaci je možné např. stáhnout z [3].

3.5 Další

Existuje samozřejmě mnoho jiných aplikací a programů, které ale obsahují vždy jen část daného problému. Na internetu jsou různé WWW stránky s recepty, cviky, či různými tabulkami pro příjem energie, nikde na internetu jsem ale nenašla informační systém, který by obsahoval vše pohromadě.

4 Požadavky kladené na informační systém

Tento informační systém by měl sloužit na podporu tréninkových aktivit a výživového plánu. Jeho hlavním posláním je plánování sportovních aktivit a výživy a kontrola dodržování naplánovaných akcí. Měl by být primárně určen pro sportovce nebo osoby, které mají zájem o sledování své přijaté a vydané energie.

Systém je víceuživatelský a po přihlášení je možné zobrazit jen ty stránky, které jsou určeny právě přihlášenému typu uživatele.

4.1 Sportovec

Je ten typ uživatele, pro kterého bude aplikace primárně napsána. Sportovec bude tuto aplikaci plně využívat, ostatní typy uživatelů pro něj „jen“ vše připravují a zvyšují tím ještě více sílu systému. Název „sportovec“ je možná trochu zavádějící, neboť se pod ním skrývá nejen uživatel, který plně trénuje a aplikaci využívá hlavně pro naplánování tréninků, ale je to i uživatel, který žádný sport aktivně neprovozuje. Systém mu slouží hlavně pro sledování své přijaté energie, tedy jeho stravování. Tito uživatelé jej využívají především pro snížení své hmotnosti. V takovém případě je ale nutné sledovat nejen přijatou energii, ale také vydanou, aby bylo vše v optimálním poměru. Musí být tedy zajištěn menší příjem energie než její výdej, proto byl název „sportovec“ použit a sportovní výkony jsou vždy zaznamenávány.

Jak již bylo naznačeno v předchozím odstavci, jsou dva hlavní úkoly, které musí aplikace splňovat. Je to sledování příjmu a výdeje energie. Ostatní části jsou hlavně doplněním hlavního poslání aplikace a jejího většího využívání.

4.1.1 Příjem a výdej energie

Zaznamenávání přijaté a vydané energie musí být přehledné, intuitivní a hlavně snadné a rychlé. Musí se vzít v úvahu, že tyto údaje budou vkládány velmi často, proto není možné, aby bylo zadávání pracné a složité. Na samostatných stránkách bude zobrazen samostatně příjem a výdej.

4.1.1.1 Příjem energie

Pro vložení příjmu energie se určí datum. Přednastavená hodnota bude aktuální datum. Poté se vybere typ jídla (snídaně, oběd,…) a již se přejde k zadání potraviny. Pro snadnější vyhledávání budou potraviny rozděleny do skupin. Po vybrání skupiny se načte seznam potravin v této skupině, kde se již konkrétní potravina vybere. Po zadání množství bude možné potravinu vložit mezi přijatou energii

daného dne. Množství bude zadáváno číslem a jednotkou. Jednotky mezi sebou budou mít přepočítací vztah, takže nezáleží na tom, v jaké jednotce uživatel potravinu vloží.

V zobrazení přijaté energie si uživatel bude moci vybrat časový interval, ve kterém chce zobrazit přijaté potraviny. V tabulce bude zobrazováno datum, název, přijatá energie v kJ a množství tuků, cukrů a bílkovin v dané potravině. U této tabulky bude také graf složení přijaté potravy. Tedy celkové množství tuků, cukrů a bílkovin.

4.1.1.2 Výdej energie

Každé aktivitě bude přiřazen typ pro snadnější vyhledávání. Při zadávání se vybere typ aktivity a podle tohoto typu se načte seznam konkrétních činností. Po jejím vybrání, zadání data a trvání aktivity se tato položka uloží mezi vydanou energii daného uživatele.

Pro zobrazení činností si uživatel vybere časový interval, ve kterém chce zobrazit vydanou energii. U každé aktivity se bude zobrazovat název, délka a vydaná energie v kJ.

4.1.1.3 Rozložení přidané a vydané energie

Pro přehledné rozložení přidané a vydané energie bude zobrazován výsečový graf, kde toto rozdělení bude procentuálně znázorněno. Uživatel bude moci toto rozložení kontrolovat, v případě požadavku o udržení váhy by mělo být rozložení 50 na 50. Jestliže chce váhu snížit musí být podíl přijaté energie menší než vydané a při požadavku o zvýšení váhy naopak. Sportovec si bude moci vybrat časový interval, ve kterém bude rozložení energie znázorněno. Implicitně bude znázorněn aktuální den.

4.1.2 Váha

Váha bude rozdělena na dvě tabulky, aktuální a požadovaná. Tyto hodnoty bude uživatel zadávat do systému sám a budou zobrazeny, jak v tabulce, tak grafu. Opět, stejně jako při zobrazování energie, bude moci vybrat časový interval zobrazení. Při zadávání aktuální hmotnosti vybere aktuální datum a aktuální váhu, při vkládání požadované hmotnosti vybere datum a váhu, kterou by chtěl k danému datu dosáhnout.

Při vykreslování grafu nebudou nezadané hodnoty znamenat nulovou váhu, ale vypočítá se přibližná hodnota, kterou by měl uživatel mít.

4.1.3 Jídelníček

Pro sportovce bude jídelníček vybírat a připravovat jeho dietolog. Uživatel se bude tímto jídelníčkem řídit a odškrtávat již přijatá jídla, což se projeví v jeho přijaté energii. Jednotlivá jídla se mu budou zobrazovat po dnech a bude moci mezi nimi přepínat. U každého dne bude vždy zobrazen typ jídla (snídaně, oběd,….) a daná potravina s množstvím, které má sportovec sníst, eventuálně nějaká poznámka.

4.1.4 Trénink

Připravovat tréninkové plány bude trenér. Sportovci se bude objevovat jeho tréninkový plán po jednotlivých dnech, mezi kterými bude moci přepínat. Po splnění tréninkové jednotky tuto skutečnost zaškrtne, to se projeví u jeho vydané energie.

4.1.5 Vybírání trenéra a dietologa

Každý sportovec si bude moci sám rušit či přidávat svého trenéra respektive dietologa. Může mít přiděleného vždy pouze jednoho z každé funkce.

4.1.6 Ostatní

4.1.6.1 Výpočet BMI indexu

Uživatel si bude moci vypočítat Body Mass Index. Po zadání své váhy a hmotnosti se zobrazí vypočtená hodnota. Na této stránce také bude tabulku, která bude popisovat možné hodnoty a také zdravotní rizika, která mohou nastat při dané hmotnosti.

4.1.6.2 Export

Možnost exportovat data bude mít uživatel do formátu CSV[1]. Bude zahrnut export vývoje hmotnosti, a svou přijatou a vydanou energii za určité časové období.

4.1.6.3 Statistiky výsledků

Pro srovnání s ostatními uživateli využívající tento systém bude možné zobrazit dva typy statistik - přijaté a vydané energie.

4.1.6.4 Predikce úbytku váhy

Ze zadaného jídelníčku a tréninku bude určen přibližný úbytek váhy pro danou osobu.

4.2 Dietolog

Hlavní úlohou dietolologa je starat se o stravu svých sportovců. Připravuje jídelníčky, přiřazuje je sportovcům. Také může měnit veškeré hodnoty u potravin, když zjistí, že jsou zadány špatně. Potraviny také může přeřazovat mezi jednotlivými kategoriemi. Další důležitou úlohou dietologa je přidávat nová data – potraviny, případně hotová jídla, což pomáhá zvyšovat hodnotu databáze.

[1] CSV (Comma-separated Values, hodnoty oddělené čárkami) je jednoduchý souborový formát určený pro výměnu tabulkových hodnot.

4.2.1 Správa sportovců

Dietolog může mít až několik přiřazených sportovců, o které se stará. Sportovce si bude moci přiřadit ze seznamu všech sportovců, kteří nemají svého dietologa. Nemůže tedy mít sportovce, který už svého dietologa má. Změnu dietologa může tedy učinit, buď sportovec, nebo původní dietolog, který danou osobu odhlásí a nový dietolog si ji přihlásí.

4.2.2 Jídelníček

Tvoření jídelníčku je nejdůležitější prácí dietologa. Musí dbát na správné složení a množství stravy pro daného jedince. Při tvorbě nového jídelníčku nejdříve určí název a počet dní. Maximální počet dní bude 14, protože pak už se dny většinou opakují, případně bude muset být vytvořen jídelníček nový. Tvorba jídelníčku bude prováděna po jednotlivých dnech pro lepší přehlednost stránky. Dietolog vybere typ jídla, potravinu a množství. Potravina bude opět, jako při vkládání přijaté energie u sportovce, přiřazena do kategorie, takže je tímto způsobem usnadněno hledání. Jestliže se daná potravina v seznamu nenachází, může ji dietolog doplnit. Po vložení položky do jídelníčku se tato skutečnost zobrazí na stejné stránce, takže dietolog bude mít přehled o již vložených hodnotách, které přísluší k danému dni. Tímto způsobem po jednotlivých dnech vyplní celý jídelníček.

Své vytvořené jídelníčky bude moci měnit obdobným způsobem jako při vkládání. Svým sportovcům je přiřadí vybráním jména a jídelníčku. Tato skutečnost se zároveň projeví také u sportovce.

4.2.3 Potraviny

Dietolog má právo měnit hodnoty u potravin v případě chybného zadání. Také bude moci přidávat potraviny nové. Tyto formuláře budou podobné. Každá potravina musí být přiřazena do kategorie. Toto rozdělení by mělo usnadnit hledání určité potraviny. Každé potravině musí být zadány následující hodnoty: energetická hodnota ve 100g a množství, tuků, cukrů a bílkovin v gramech.

4.3 Trenér

Trenéři mají přiřazeny sportovce, může jich mít několik, sportovec má ale vždy maximálně jednoho trenéra, jejich hlavní úlohou je starat se o jejich tréninkový plán. Plánovat jim tréninky a hlavně dbát na jejich vyvážení, aby nebyli přetížení nebo naopak málo zatížení.

4.3.1 Správa sportovců

Správa sportovců u trenéra je totožná jako u dietologa, jen vztažena k trenérovi.

4.3.2 Trénink

Tréninky jsou hlavním posláním trenérů. Při tvorbě musí brát v úvahu zdatnost a zdravotní stav svého sportovce. Tréninky budou vytvářeny obdobně jako jídelníčky maximálně na 14 dní a budou vkládány po jednotlivých dnech. Trenér musí vybrat kategorii pohybu a podle toho určit konkrétní aktivitu, která bude provozována. Samozřejmě může také přidat aktivitu novou. Při zadávání musí určit délku trvání daného tréninku. Po vložení se zobrazí vydaná energie. Ta je pouze z informativního důvodu. Trenér musí vědět, jestli je námaha dostatečná, nebo má ještě nějaký cvik přidat. Samozřejmostí bude vyplnění poznámky u každé aktivity. Tam bude moci být nějaká další informace k dané aktivitě, např. intenzita cvičení.

4.3.3 Aktivity

Trenér může přidávat do databáze nové aktivity případně měnit hodnoty u stávajících. Při vkládání aktivity musí zadat skupinu do které patří a její energetickou hodnotu v základní jednotce (1 kJ/min/kg).

4.4 Administrátor

Stará se, aby vše správně fungovalo. O datovou vrstvu a také, aby veškeré funkce, popsané výše, dělaly vše správně a systém byl v pořádku.

4.4.1 Správa uživatelů

Uživatele do systému může vkládat pouze administrátor. Bude zadávat jeho osobní údaje a hlavně musí určit typ uživatele, tedy vybere, zda se jedná o sportovce, trenéra, dietologa, případně administrátora. Jeden člověk může mít také více funkcí.

4.4.2 Export, import

Z důvodu zálohy databáze bude administrátorovi umožněn export a import potravin a aktivit.

5 Analýza požadavků

5.1 Model případů použití (Use Case)

Případy užití zachycují přesně funkčnost informačního systému a určují, co se v dané práci objeví, omezují rozsah práce. Model případů použití vychází z předchozí kapitoly, tj. z požadavků na systém. Všechny akce v systému jsou vyvolány aktéry, kteří budou blíže popsáni v následujících podkapitolách.

5.1.1 Aktér Uživatel

Tento aktér je nadřazeným aktérem všech ostatních. Jeho akcemi je pouze přihlášení a odhlášení do/ze systému. Tyto dvě akce jsou pro všechny společné, proto můžou další aktéři vzniknout děděním Uživatele. Ostatní aktéři tedy vzniknou specializací Uživatele po přihlášení do systému.

5.1.1.1 Přihlášení uživatele

Předpoklady: uživatel není přihlášen do systému. Cíle: přihlášení uživatele do systému, specializace aktéra podle jeho typu. Postup: uživatel vyplní do přihlašovacího formuláře své údaje (login, heslo), které se odešlou k autentizaci, v případě neúspěchu se vypíše chybové hlášení, v opačném případě se určí typ uživatele, který již bude přihlášen do systému.

5.1.1.2 Odhlášení uživatele

Předpoklady: uživatel je přihlášen do systému. Cíle: odhlášení uživatele ze systému. Postup: uživatel zadá požadavek o odhlášení ze systému a systém jej odhlásí

5.1.2 Aktér Administrátor

Stará se o bezpečnost systému a její dodržování. Edituje uživatele.

5.1.2.1 Evidence uživatele

Předpoklady: nejsou[2]. Cíle: přidání, odebrání, změna údajů uživatele. Postup: administrátor vybere akci přidání uživatele, vyplní jeho údaje a přidá je do databáze, nebo vybere přímo uživatele a změní jeho údaje, případně jej smaže.

[2] Vychází se z předpokladu, že uživatel je přihlášen.

5.1.2.2 Export tabulkových hodnot

Předpoklady: nejsou. Cíle: export tabulkových hodnot. Postup: administrátor si vybere, které údaje chce exportovat.

5.1.2.3 Import tabulkových hodnot

Předpoklady: nejsou. Cíle: import tabulkových hodnot. Postup: administrátor vloží importovaný soubor.

5.1.3 Aktér Sportovec

Sportovec eviduje svoji přijatou a vydanou energii, svoji váhu a prohlíží a plní svoje tréninky a jídelníčky. Přijatá a vydaná energii bude popisována současně, protože se jedná o podobné akce. Taktéž bude podobným způsobem rozebrána evidence váhy pro aktuální a požadovanou a pro zjištění informací o tréninku a jídelníčku.

5.1.3.1 Evidence přijaté (vydané) energie

Předpoklady: nejsou. Cíle: přidání, odebrání přijaté (vydané) energie. Postup: sportovec vybere akci pro přidání přijaté (vydané) energie, zadá potřebné údaje (potravinu (aktivitu), datum, množství (délku)), nebo vybere odebrání energie.

5.1.3.2 Evidence váhy

Předpoklady: nejsou. Cíle: přidání, odebrání, prohlížení aktuální (požadované) váhy. Postup: sportovec vybere akci přidání váhy, zadá povinné údaje (datum a váhu), nebo akci smazání váhy, případně prohlížení v určitém datovém intervalu

5.1.3.3 Prohlížení tréninků (jídelníčku)

Předpoklady: nejsou. Cíle: zobrazení tréninku (jídelníčku) sportovce. Postup: sportovec vybere akci pro prohlížení jeho tréninkových plánů (jídelníčků).

5.1.3.4 Přiřazení svého trenéra (dietologa)

Předpoklady: sportovec ještě nemá žádného trenéra (dietologa) přiřazeného. Cíle: přiřazení trenéra (dietologa). Postup: sportovec si vybere ze seznamu volných trenéru (dietologů) a označí toho, který mu má být přidělen.

5.1.3.5 Výpočet BMI

Předpoklady: nejsou. Cíle: výpočet BMI. Postup: sportovec zadá své údaje (hmotnost, výšku).

5.1.3.6 Export tabulkových hodnot

Předpoklady: v systému jsou již zadána některá data o příjmu a výdeji energie. Cíle: export tabulkových hodnot. Postup: uživatel si vybere, které údaje chce exportovat.

5.1.3.7 Zobrazení statistiky výsledků

Předpoklady: uživatel má přidělen jídelníček a trénink. Cíle: zobrazení statistik výsledků. Postup: sportovec vybere zobrazení statistik.

5.1.3.8 Predikce úbytku váhy

Předpoklady: nejsou. Cíle: zobrazení predikce úbytku váhy. Postup: sportovec vybere zobrazení predikce..

5.1.4 Aktér Trenér

Aktér Trenér reprezentuje roli uživatele, který se stará o databází veškerého výdeje energie a o svoje svěřence.

5.1.4.1 Editace tréninků

Předpoklady: nejsou. Cíle: přidání, odebrání, změna tréninků. Postup: Trenér vybere akci přidání tréninku a doplní potřebná data (počet dní, aktivitu a trvání v jednotlivých dnech), nebo vybere akci změna tréninku, kde může tato data měnit, případně akci odebrání.

5.1.4.2 Editace aktivit

Předpoklady: nejsou. Cíle: přidání, odebrání, změna aktivit. Postup: Trenér vybere akci přidání aktivity a doplní potřebná data (název, vydanou energii a typ aktivity), nebo vybere akci změna aktivity, kde může tato data měnit, případně akci odebrání.

5.1.4.3 Přiřazování sportovců

Předpoklady: nejsou. Cíle: přiřazení sportovce. Postup: Trenér vybere ze seznamu volných sportovců a provede akci přiřazení.

5.1.4.4 Přiřazování tréninků

Předpoklady: má přiřazeného aspoň jednoho sportovce. Cíle: přiřazení tréninků sportovci. Postup: Trenér vybere akci přiřazení tréninku, vybere trénink a osobu.

5.1.5 Aktér Dietolog

Aktér Dietolog reprezentuje roli uživatele, který se stará o databází veškerého příjmu energie a o své svěřence.

5.1.5.1 Editace jídelníčků

Předpoklady: nejsou. Cíle: přidání, odebrání, změna jídelníčků. Postup: Dietolog vybere akci přidání jídelníčku a doplní potřebná data (počet dní, potravinu a typ jídla), nebo vybere akci změna jídelníčku, kde může tato data měnit, případně akci odebrání.

5.1.5.2 Editace potravin

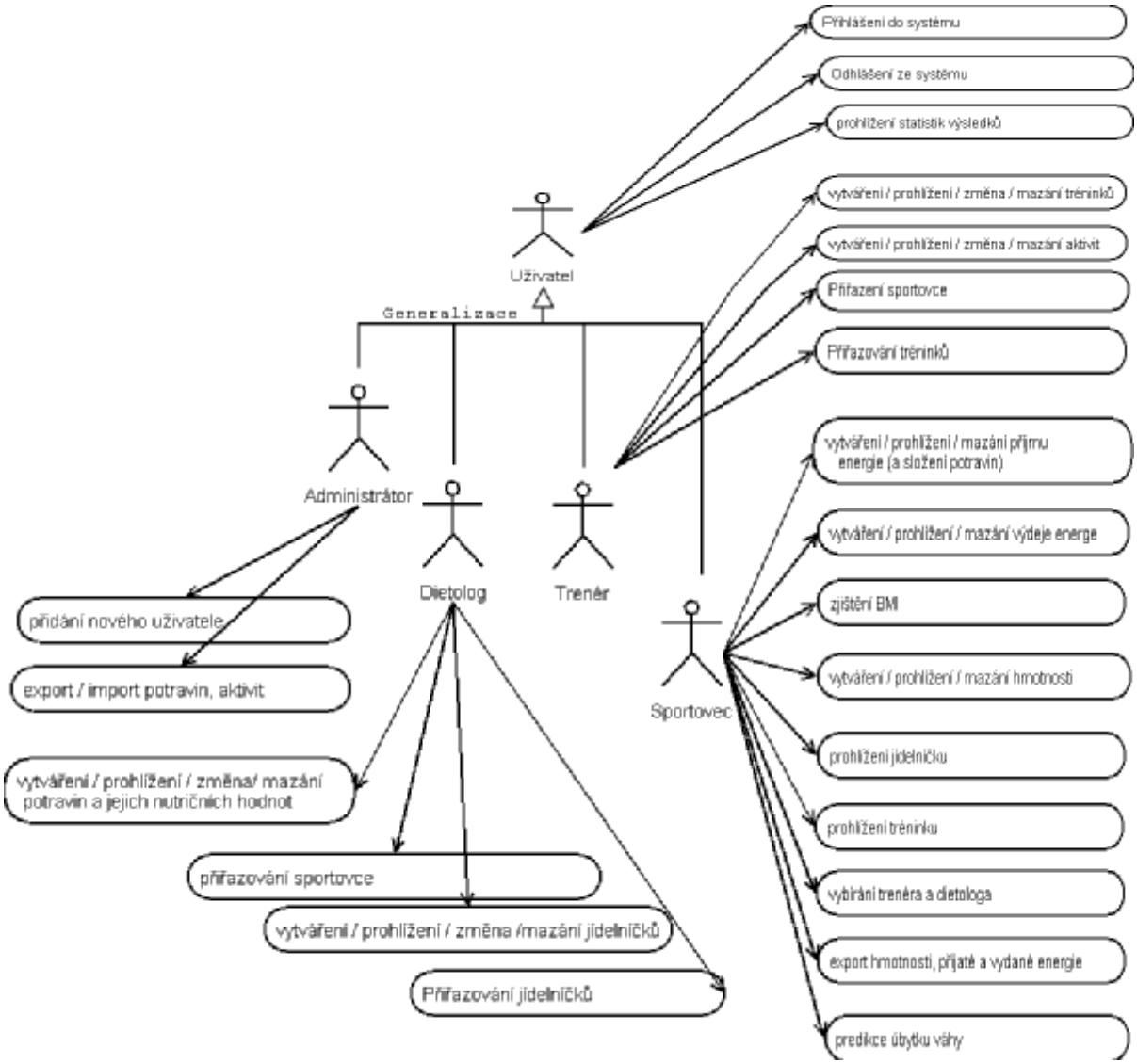
Předpoklady: nejsou. Cíle: přidání, odebrání, změna potraviny. Postup: Dietolog vybere akci přidání potraviny a doplní potřebná data (název, přijatou energii, typ potraviny a složení – tuky, cukry, bílkoviny), nebo vybere akci změna potraviny, kde může tato data měnit, případně akci odebrání.

5.1.5.3 Přiřazování sportovců

Tato akce je stejná jako u trenéra.

5.1.5.4 Přiřazování jídelníčků

Předpoklady: má přiřazeného aspoň jednoho sportovce. Cíle: přiřazení tréninků jídelníčku. Postup: Dietolog vybere akci přiřazení jídelníčku, vybere jídelníček a osobu.



obrázek 4 Diagram případů použití

6 Návrh systému v UML

6.1 Datové modelování (ER diagram)

ER model definuje požadavky na data. Je to tedy pohled, kde nás zajímají data, ne operace. Máme základní entity a vazby mezi nimi. Tento model neukazuje, jaké operace se budou s daty provádět, ale pouze jaké atributy bude obsahovat.

Tento diagram je v 3. normální formě, což znamená, že veškeré hodnoty v tabulkách jsou atomické, dále musí být všechny neklíčové atributy plně funkčně závislé na každém kandidátním klíči, takže se nesmí v řádku tabulky objevit položka, která by byla závislá jen na části primárního klíče. V tomto diagramu je v každé tabulce jen jeden primární klíč, proto je tato vlastnost automaticky splněna. Nedochází proto k redundanci dat. A dále není obsažen ani jeden neklíčový atribut, který by byl tranzitivně závislý na některém kandidátním klíči. I tuto vlastnost splňuje ER diagram tohoto systému.

ER diagram informačního systému obsahuje 19 tabulek – entit:

6.1.1 User (uživatel)

Tento systém je víceuživatelský, pro uživatele byla vytvořena entita User (uživatel). Mohou být čtyři typy, a to admin (administrátor), sportsman (sportovec), coach (trenér) a dietitian (dietolog). Uživatel má atributy, které toto rozlišují, takže entita User obsahuje 4 atributy s těmito funkcemi a každá může nabývat hodnot true/false. Tyto atributy musí být čtyři, protože jedna osoba může nabývat více typů.

Dalšími atributy, které tato entita obsahuje jsou osobní údaje uživatele: name (jméno), surname (příjmení), email, birthday (datum narození), height (výška v cm) a sex (pohlaví – muž nebo žena).

6.1.2 Weight_actual (aktuální váha)

Pro uložení informací o aktuální váze slouží tabulka weight_actual. Atributy jsou weight (váha v kg), date_created (datum vytvoření), comment (poznámka) a cizím klíčem je identifikátor uživatele. Vztah mezi těmito dvěma tabulkami je 1:N, protože uživatel postupně zadává svoji hmotnost v jednotlivých dnech.

6.1.3 Weight_demanded (požadovaná váha)

Tato tabulka je podobná jako předchozí, ale slouží pro uchování informací o požadované váze, nikoli o aktuální. Atributy jsou tedy weight (váha, tentokrát požadovaná), date_created (datum vytvoření), date_demanded (datum, ve kterém má být splněna požadovaná hmotnost), comment (poznámka) a opět cizí klíč do tabulky User.

6.1.4 Meal (typ jídla)

Tabulka sloužící pro uchování typu jídla (snídaně, oběd,…). Atributem je pouze identifikátor a name (název).

6.1.5 Food_group (skupina pro potraviny)

Entita, ve které jsou uloženy skupiny pro jídla, aby bylo zjednodušeno vyhledání v potravinách. Opět jsou atributy pouze identifikátor a název.

6.1.6 Food_unit (jednotka potraviny)

V této entitě budou uložena data se všemi jednotkami, reprezentovány svým názvem a převodním vztahem vztažené ke gramu.

6.1.7 Food (potravina)

Každá potravina je určena svým name (jménem), energy (energetickou hodnotou v kJ), lipids (tuky), sugars (cukry), albumins (bílkoviny), také svojí default_amount (což je množství, ke kterému jsou vztaženy předchozí hodnoty v dané jednotce) a default_unit (jednotka). Má dva cizí klíče a to do tabulky food_unit (tabulka jednotek), ke které má vztah 1:1, a cizí klíč k určení vazby k tabulce food_group, mezi kterými je vztah 1:N.

6.1.8 Energy_in (přijatá energie)

Tato tabulka slouží k evidenci přijaté energie. Jsou zde ukládány všechny položky o přijaté energie uživatele. Proto je v tabulce cizím klíčem identifikátor uživatele. Mezi těmito dvěma entitami je vztah 1:N, tedy uživatel může mít několik přijatých energií. Dalším cizím klíčem je v tabulce identifikátor do tabulky meal (typ jídla) a do tabulky food (potravina), v obou dvou případech je opět vztah 1:N.

6.1.9 Fare (jídelníček)

Je to hlavní entita pro jídelníček, která je vazbou spojena s fare_items, ve které jsou již konkrétní položky. Atributy fare (jídelníčku) jsou date (datum vytvoření), name (název jídelníčku), public (příznak, jestli je jídelníček zobrazen všem uživatelům nebo jen osobě, která ho vytvořila, nebo které byl přiřazen), days (počet dní, ze kterých je jídelníček tvořen), comment (poznámka k jídelníčku) a cizí klíč do tabulky uživatele, který jídelníček vytvořil. Vztah mezi těmito tabulkami je 1:N, uživatel totiž může vytvořit více jídelníčků.

6.1.10 Fare_items (jednotka jídelníčku)

Ukládá každou položku v jídelníčku. Jednotlivé položky jsou identifikovány atributy day (který to je den v dietě), comment (poznámka), amount (množství, přiřazené k určené jednotce), jednotka je určena cizím klíčem do tabulky food_unit, vztah mezi nimi je 1:1, dalším cizím klíčem je klíč do tabulky fare (přiřazen k danému jídelníčku) a do tabulky meal_id (udávající typ jídla).

6.1.11 Assigned_fare (přiřazený jídelníček)

Entita která udává, kdo má daný jídelníček přiřazen. Je spojena s tabulkami user (uživatel) a fare (jídelníček) vazbami 1:N, protože každý uživatel může mít přiřazeno několik jídelníčků a zároveň daný jídelníček může mít více lidí. Dalšími atributy jsou ještě date_from (od kterého je daný jídelníček přiřazen), date_to (do kterého dne platí) a comment (poznámka).

6.1.12 Activity_group (skupina aktivit)

Podobně jako u potravin také aktivity mají svou skupinu určenou svým jménem a identifikátorem.

6.1.13 Activity (aktivita)

Každá aktivita je určena svým name (názvem), energy (energie, která je touto aktivitou vydaná vztažená k jednotce kJ/min/kg) a poté cizím klíčem ke skupině se vztahem 1:N.

6.1.14 Energy_out (výdej energie)

Tato tabulka slouží pro zadávání uživatelova veškerého výdeje energie. Zadává se date (datum), duration (trvání v minutách), energy (vydaná energie), comment (poznámka) a cizí klíče do tabulky uživatele a aktivit, obě dvě vazby jsou 1:N.

6.1.15 Training (trénink)

Je to hlavní entita pro trénink, vazbou 1:N je spojena s training_items, která obsahuje jednotlivé tréninkové jednotky. Atributy training (tréninku) jsou date (datum vytvoření), name (název tréninku), public (příznak, jestli je trénink zobrazen všem uživatelům nebo jen osobě, která ho vytvořila, nebo které byl přiřazen), days (počet dní, ze kterých je trénink tvořen), comment (poznámka) a cizí klíč do tabulky uživatele, který trénink vytvořil. Vztah mezi těmito tabulkami je 1:N, uživatel totiž může vytvořit více tréninků.

6.1.16 Training_items (jednotka tréninku)

Ukládá jednotlivé položky v celém tréninku. Položky jsou určeny day (dnem, tedy o který den tréninku se jedná), comment (poznámkou) a during (délkou trvání). Tato tabulka je spojena s tabulkou training (trénink), mezi kterými je vazba 1:N.

6.1.17 Assigned_training (přiřazený trénink)

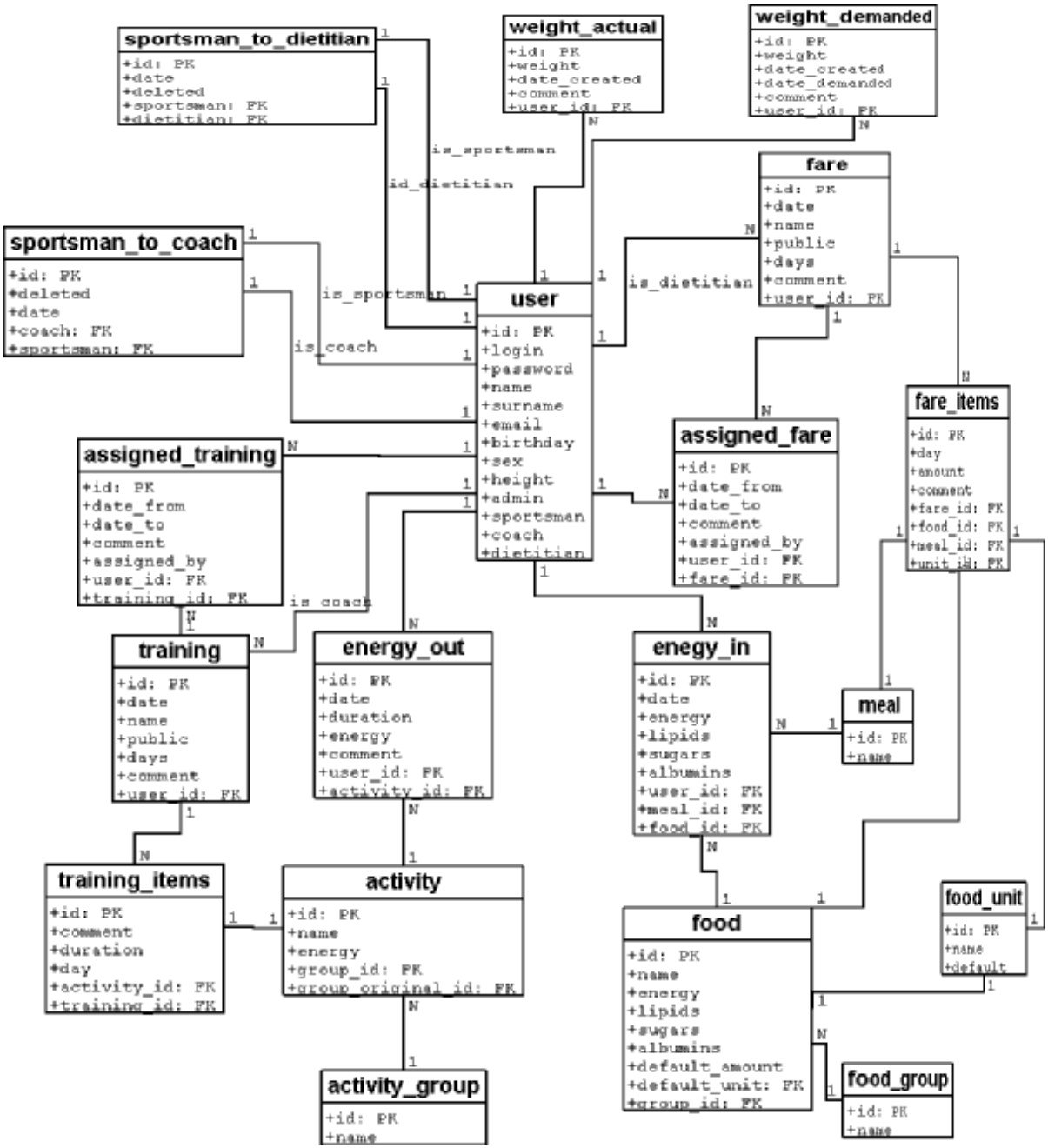
Entita která udává, kdo a který má daný trénink přiřazen. Je spojena s tabulkami user (uživatel) a training (trénink) vazbami 1:N, protože každý uživatel může mít přiřazeno několik tréninků a zároveň daný trénink může mít více lidí. Dalšími atributy jsou ještě date (datum) a comment (poznámka).

6.1.18 Sportsman_to_dietitian

Tato tabulka říká, jaký sportovec je trenéru přiřazen, respektive jaké sportovce má trenér. Je spojena s tabulkou user (uživatel) dvěma vazbami, protože záleží, zda se jedná o pohled sportovce nebo dietologa.

6.1.19 Sportsman_to_coach

Tato tabulka je stejná jako předcházející jen se jedná o vztah trenéra a sportovce.



obrázek 5 ER diagram

7 Implementace

7.1 Obecný popis

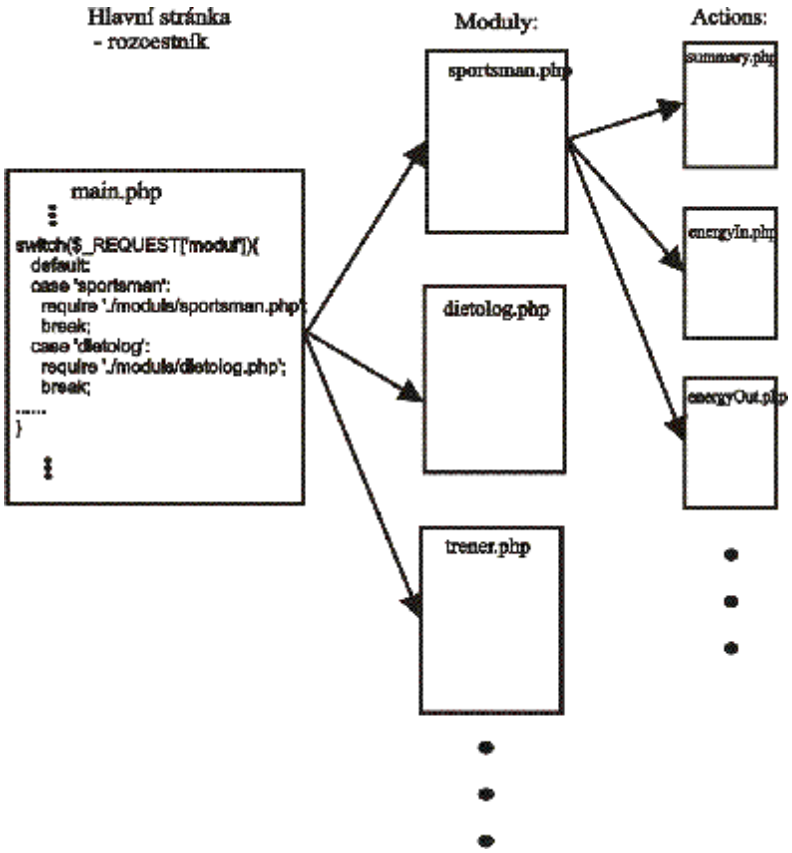
Protože je již v dnešní době objektové programování velmi rozšířené, ani jazyku PHP se nevyhnulo. Základy objektového programování byly již v malé míře napsány v PHP3, v další verzi nebylo příliš vylepšeno, protože byla požadována hlavně kompatibilita s předchozí verzí. Ovšem kvůli stále narůstajícím požadavkům na vylepšení objektově orientovaného programování, byl objektový model v PHP5 kompletně přepracován. V této práci bylo proto objektového programování plně využito. Popis jednotlivých tříd a bližší vnitřní implementace bude popsána v dalších kapitolách 7.2 a 7.3.

7.1.1 Struktura

Struktura celé aplikace byla navržena tak, aby byla jednoduchá, přehledná, ale aby zároveň obsahovala vše potřebné. Úvodní stránka index.php obsahuje pouze přihlašování, protože uživatel, který vstupuje do systému musí být vždy přihlášen. Není dovoleno prohlížet nebo provádět jakékoli akce nepřihlášenému uživateli. Po přihlášení je přesměrováno na stránku main.php, což je hlavní stránka celé aplikace. Pracuje se vlastně jenom na této stránce a do ní jsou moduly, funkce, třídy pouze vkládány.

Po odeslání požadavku o přihlášení se začne zpracovávat stránka main.php. Na této stránce se provádí všechny potřebné operace. Nejdříve se zinicializuje Ajax a jeho veškeré funkce. Poté se vloží soubor pro načtení všech tříd. V tomto soubor je tedy provedena funkce *__autoload()* pro načtení tříd, které jsou uloženy ve složce classes, zavolá se tato funkce s parametrem jménem třídy a daná třída se vloží na požadované místo. Dále je vkládán soubor, který obsahuje dvě funkce, první požaduje od serveru objekt z pole session a druhá určuje adresu url, tedy parametry modulu a akce, která je požadována. Dalším vloženým souborem je soubor verify.php, který ověřuje uživatele, zda přihlašovací údaje souhlasí s údaji v databázi, jestli ano, pokračuje prováděním dalších příkazů, v opačném případě je přesměrován na úvodní stránku index.php. Na tuto stránku je také přesměrován, jestliže byl 15 minut nečinný, tedy neprovedl žádnou akci.

V dalším kroku je vytvořen objekt pro oznamování zpráv a objekt pro šablony. Poté je zavolána ajaxová funkce *sajax_handle_client_request()*, která zpracovává sajax požadavek, pokud se požadavek zpracuje provádění skriptu, zde skončí a dále již se neprovádí a stránka nemusí být celá načtena. V opačném případě se stránka zpracovává dále. Je načteno hlavní menu a podle typu uživatele je také zobrazen modul, který by měl být pro něj nejdůležitější. Podle výběru modulu je tato stránka vložena do skriptu a prováděny operace z tohoto modulu. Vkládání skriptů je znázorněno na následujícím obrázku:



obrázek 6 Struktura projektu

Podle vybraného modulu se také načítá vertikální menu (actions), které je závislé na vybraném modulu.

Celý projekt byl rozdělen na tři části, jedná se o datové třídy, které slouží pro práci s daty jako takovými, teda nejedná se ještě o žádné vykreslování. Další vrstvou jsou třídy uživatelského rozhraní. Byly napsány pro jednodušší práci s formulářovými prvky, které se v projektu velmi často vyskytují a opakují. Vytvořením těchto tříd se stal kód přehlednější, jasnější a samozřejmě také lépe psatelným. Poslední oddělením bylo používání šablon, které již jenom vykreslují celkový vzhled stránky. Bližší popis těchto tří části je uveden v následujících třech kapitolách.

7.2 Třídy pro práci s daty

V následujících podkapitolách budou popsány důležité datové třídy, jejich atributy a metody. Mezi metodami ovšem nebudou vypsány ty, které nastavují nebo získávají jednotlivé atributy, i když je všechny třídy obsahují pro snadnou manipulaci se všemi atributy. Tyto metody jsou podobné a vypadají přibližně takto:

public function setAtribut($atribut) { $this->atribut = $atribut; }

public function getAtribut() { return $this->atribut; }

Jestliže se jedná o atribut nabývající pouze booleovských hodnot, je metoda *getAtribut()* nahrazena metodou *isAtribut()*, ale tělo funkce zůstává stejné.

7.2.1 Menu

V aplikaci jsou dvě menu, jedno vodorovné a jedno horizontální. Horizontální menu je zobrazováno podle typu uživatele. Každý typ uživatele má svůj modul a v tomto modulu je vždy jako první načtena stránka, která by měla být pro daný typ uživatele nejdůležitější.

Po vybrání položky z horizontálního menu se pro daný typ uživatele změní menu vertikální. Takže toto menu je závislé na výběru horizontálního.

Třída *MenuItem* je pro horizontální menu a pro vertikální *ActionMenuItem*. Nejprve je popsána třída *MenuItem*:

Atributy:

- *title* – titulek daného menu, tedy to co je zobrazováno.
- *action* – název modulu na které se menu odkazuje.
- *accessRights* – určuje, kterému typu uživatele se zobrazí. Všechny tyto čtyři parametry jsou předávány při vytváření objektu a v konstruktu přiřazeny.

Metody:

- *isLegal* - ověří, zda se menu zobrazí přihlášenému uživateli. Je tedy pro každý typ napsána podmínka, ve které se postupně vyhodnocuje, zda uživatel je daného typu a ve druhé části, jestli se pro dané menu má tomuto typu uživatele vůbec zobrazit. Metoda vrací true, jestliže jsou obě části splněny.
- *addPrivileges* – slouží k přidání práva typu uživatele do řetězce *accessRights*.
- *removePrivileges* – odstraní práva typu uživatele z *accessRights*.

ActionMenuItem vznikne děděním z *MenuItem*, každý nový objekt z třídy *ActionMenuItem* je položka ve vertikálním menu.

Atributy:

- *childs* – slouží k ukládání podmenu, je to pole.

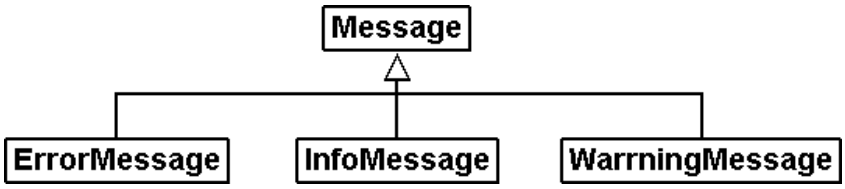
Metody:

- *addmenu* – vloží položku do svého podmenu (do pole *childs*)
- *showByAction* - zobrazí rozbalené podmenu, když je vybraný někdo z toho podmenu nebo rodič toho podmenu

7.2.2 Třídy pro vypsání zpráv

Pro jakékoli oznamování zpráv je vyhrazen prostor v horní části obrazovky. Pro tyto zprávy byla vytvořena abstraktní třída *Message* a její potomci *InfoMessage*, *ErrorMessage* a *WarrningMessage*. Abstraktní třída je třída, která obsahuje jak abstraktní metody, tak hotové metody. Abstraktní třídy

nevytváří instance, ale jsou děděny a potomci této abstraktní třídy již instance vytvářejí. Od této třídy se odvozují další třídy. Tak je tomu i v tomto případě.



obrázek 7 Třídy pro vypsání zpráv

Třída *Messsage*:

Atributy:

- *message* – zpráva, která se zobrazuje
- *typ* - může být trojího druhu: *error* – pro oznamování chybových zpráv, *info* – zobrazuje informativní zprávy (např. informace o uložení apod.) a *warrning* – zobrazuje upozornění.

Daný objekt vytváříme podle typu zprávy, kterou chceme mít. Jestliže chceme vytvořit chybovou zprávu použijeme třídu *ErrorMessage*. Ta má jako parametr předanou zprávu. V konstruktu se zavolá abstraktní třída, které předá jako první parametr danou zprávu a jakou druhý parametr typ, tedy v tomto případě *error*. Tomuto vytvořenému objektu je předána tedy zpráva a typ. V dalších dvou případech je situace obdobná jen je objekt vytvářen pomocí třídy *InfoMessage* nebo *WarrningMessage* a předáván typ *info* respektive *warrning*.

Pro zobrazování všech zpráv je naimplementována třída *MessageBoard*:

Atributy:

- *board* – pole, do kterého jsou zprávy ukládány

Metody:

- *addMesage* - přidává nové zprávy a ke každé zprávě také její typ. Poté jsou zprávy podle svého typu rozdílně zobrazovány.
- *isEmpty* – zjišťuje, zda existují nějaké zprávy pro zobrazení.

Příklad vytváření zprávy:

Vytvoření objektu *MessageBoard* pro ukládání zpráv: *messageBoard = new MessageBoard();* Vytvoření instance *errorMessage* a přidání ho do MessageBoardu: *$messageBoard->addMessage(new ErrorMessage("Chybové hlášení"))*.

Následuje výpis daných zpráv, šablona, napsána pomocí Smarty, tedy obsahuje zjištění, jestli je objekt *messageBoard* prázdný metodou *isEmpty* a jestli pole nějaké zprávy obsahuje prochází pole a podle zjištěného typu zprávu barevně vypíše.

7.2.3 Uživatel

Pro práci s uživatelem je vytvořena třída *User*.

Atributy:

- *login* - jedinečný identifikátor, přihlašovací jméno
- *password* - heslo pro vstup do systému
- *name* - křestní jméno
- *surname* - příjmení
- *email* – emailová adresa
- *birthday* - datum narození
- *height* – výška osoby
- *sportsman* – sportovec
- *coachman* – trenér
- *dietitian* - dietolog
- *admin* – administrátor
- *activeTime* - čas, kdy uživatel provedl nějakou operaci.

Metody:

- *loadFromDB* - přiřadí objektu všechny vlastnosti, které uživateli patří. Je proveden dotaz na databázi a jestliže se v databázi takový uživatel nachází jsou objektu vlastnosti přiřazeny.
- *saveToDB* – uloží data o uživateli do databáze.
- *verifyPasswordConf* – ověří, zda souhlasí heslo a potvrzení hesla, při vytváření nového uživatele.
- *verifyPassword* – ověří, zda souhlasí napsané heslo s heslem v databázi.
- *haveCoach*, *haveDietitian* – zjistí, zda uživatel má přiřazeného trenéra (dietologa), jestliže tomu tak je, tak vrátí jeho id, v opačném případě vrátí nulu.

7.2.4 Aktivity

Třída, která slouží pro práci s aktivitou, pro zjištění jejich vlastností a správné vypočítání všech hodnot pro vydanou energii uživatele.

Atributy:

- *id* – id aktivity
- *name* – název pro aktivitu
- *groupId* – skupina do které daná aktivita patří
- *energy* – vydaná energie uložená v databázi vztažená k jednotce (kJ/kg/min)

Metody:

- *loadFromDB* – podle id jsou načtena data o aktivitě z databáze.

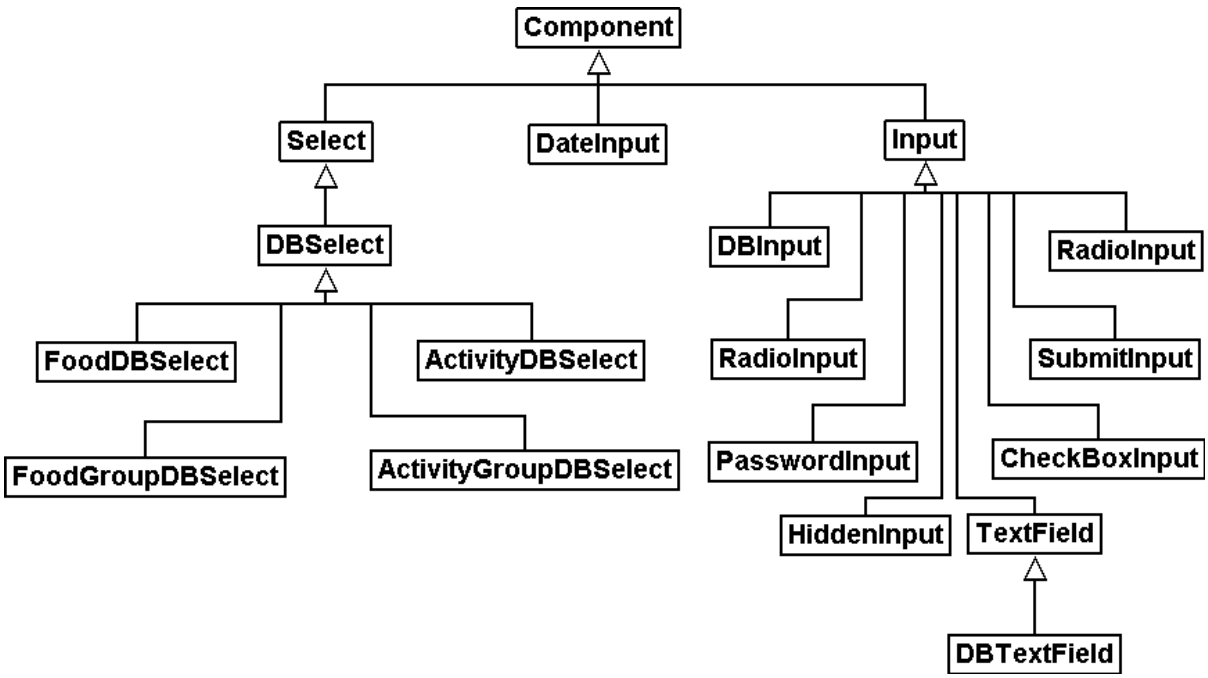
- *deleteFromDB* – smaže aktivitu z databáze.
- *saveToDB* – vloží novou aktivitu do databáze.
- *editActivity* – edituje již uloženou aktivitu, změny uloží do databáze.

7.2.5 Jídlo

Food je třída, která slouží k práci s potravinami. Vzhledem k tomu, že přijatá energie může být ukládána v různých jednotkách je vytvořena také třída *FoodUnit*. Třída *Food* obsahuje atributy a metody, které slouží k načtení hodnot, smazání a editaci jednotlivých potravin. Samozřejmostí je fakt, že potraviny musí být vztaženy k nějakému množství a jednotce, proto je vytvořena speciální třída, která se stará o editaci těchto jednotek v tabulce *food* (potravina). Množství může samozřejmě uživatel uvádět v jakýchkoli jednotkách, proto existuje pro každou jednotku převodní vztah, pomocí kterého je údaj přepočítáván a poté ukládán.

7.3 Komponenty GUI

Každý uživatel, který se přihlásí do systému komunikuje s tímto systémem pomocí webového klienta, tedy svého webového prohlížeče. Akce uživatele jsou prováděny pomocí formulářů a právě tyto formuláře jsou implementovány třídami. Tyto třídy pomáhají nejen vytvářet formuláře lépe a rychleji, ale také jejich výsledkem je strukturovanější a čitelnější kód, kde jsou psány jen nové vlastnosti k danému prvku formuláře a neopakují se tedy pořád stejné věci.



obrázek 8 Dědění tříd GUI

7.3.1 Component

Hlavní třídou je abstraktní třída *Component*. V této třídě sloužící k jednodušší práci s formuláři jsou získávány všechny informace a uloženy vlastnosti, které mají formulářové prvky společné. Při provádění skriptu jsou vytvořeny všechny prvky formuláře, nastaveny vlastnosti a vykresleny. Při tomto nastavování vlastností se vždy použije funkce *serialize* pro pozdější uložení do session. Po odeslání formuláře se tyto komponenty opět získají funkcí *unserialize* ze session. Tímto jsou zpátky vytvořeny objekty, se kterými se dále pracuje.

Atributy:

- *name* - jméno formuláře
- *class* - jméno třídy pro CSS
- *value* - hodnota formuláře
- *label* - titulek formuláře
- *labelPosition* - pozice, tedy jestli bude umístěno nad polem, nebo nalevo od pole.
- *required* - příznak o povinnosti vyplnit daný prvek formuláře. Jestliže je nastaven na true, tak se všechny tyto prvky kontrolují, v případě nevyplnění nějaké hodnoty, je uživatel požádán o její doplnění.
- *readOnly* - tedy jenom pro čtení, není možné provádět žádné změny
- *disabled* - bez hodnoty, formulářová pole učiní nepřístupnými, takže se nedají aktivovat.
- *width* - šířka prvku formuláře
- *serialize* - příznak, jestli je objekt serializován.
- atributy, do kterých jsou ukládány funkce při změně formuláře. Jsou to pole, při události se vloží do tohoto pole. Na každý typ události je samostatný atribut: *onChangeFunction* (změna ve formuláři), *onClickFunction* (při kliknutí), onKeyDown (při stisknutí klávesy), *onKeyUp* (při uvolnění klávesy), *onKeyPress* (při podržení klávesy).

Metody:

- *normalizeFunction* – přidá se na konec funkce středník, jestliže jej již neobsahuje.
- *addOnClickFunction*[3] - přidá do pole daného atributu funkci, která se má vyvolat při této události, funkce je již ukládána i s ukončujícím středníkem, protože obsahuje předcházející funkci normalize.
- *writeOnClickFunction*[4] – je procházeno pole obsahující tyto funkce, metoda vrací řetězec těchto funkcí.

---

[3] vztahuje se také na metody *addOnChangeFunction*, *addOnKeyDownFunction*, *addOnKeyUpFunction* a *addOnKeyPressFunction*

[4] vztahuje se také na metody *writeOnChangeFunction*, *writeOnKeyDownFunction*, *writeOnKeyUpFunction* a *writeOnKeyPressFunction*

- *writeFunction* – vrací všechny funkce, které se mají provést nastala-li jakákoli událost.
- *drawLabel* - vykreslí titulek prvku formuláře podle zadaných vlastností, tedy obsahuje HTML zápis pro tvorbu titulku s vlastnostmi, které daný objekt obsahuje. Pro vykreslování samotného prvku je v této třídě abstraktní metoda *draw*, která je u každého daného prvku blíže popsána.
- *draw* – abstraktní metoda, je blíže specifikována v dědících třídách.

Z této třídy *Component* dědí další tři třídy a to *Select*, *Input* a *DateInput*.

7.3.1.1 Select

Jak již název napovídá třída *Select* slouží pro vypsání formulářových prvku select, tedy výběrového pole.

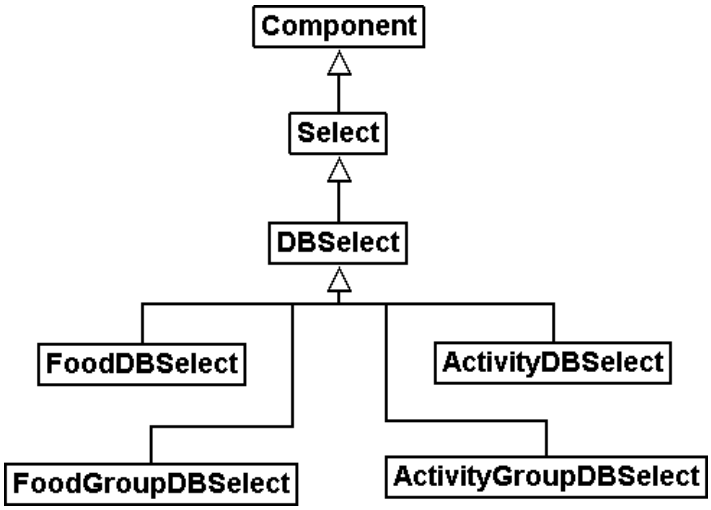
Atributy:

- *data* – položky, které se zobrazují ve výběrovém poli. Jsou ukládána do pole

Metody:

- *addKeyValue* - přidání hodnoty s klíčem do pole, této metodě se předá hodnota a klíč a jestliže daná data ještě objekt nevlastní, jsou přidána
- *removeValueForKey* - data z pole odstraňuje, jsou-li přiřazena objektu.
- *drawContent* – funkce *draw* je rozdělena do více funkcí, aby se mohla volat zvlášť a lépe kombinovat, *drawContent* je právě volaná z funkce *draw*. V této části se pro výpis nejdříve nastaví select s parametry *name* a *id* a *data*, která se budou zobrazovat, jsou načtená v poli. Toto pole se postupně prochází a přiřadí se do option. Jestliže má být daná hodnota také nastavena, je jí přiřazena vlastnost selected. Tímto způsobem jsou postupně přiřazovány hodnoty pro výpis. Celé je to potom předáváno metodě *draw*.
- *draw* – toto je již hlavní funkce pro vykreslení, která doplňuje tuto metodu a nadřazené třídy, nejdříve je nastaven styl pro výpis CSS, za kterým následuje zavolání metody *drawContent*

Dědícími třídami třídy *Select* jsou databázové třídy, ty se využívají pro výběr konkrétních dat z databáze.



obrázek 9 Zobrazení dědění ve větvi Select

DBSelect

Třída *DBSelect* slouží pro vykreslení výběrového pole s použitými daty z databáze. Vše je implementováno pro co nejjednodušší práci při vytváření výběrového pole, které je tvořeno daty z databáze. Z tohoto důvodu se zadávají jen údaje, které se skutečně mění. Co se týče databázových dotazů může se měnit pouze výsledné sloupce, tabulka a podmínka. Ale i tyto hodnoty mohou být v určitých případech shodné, např. vybírá se z tabulky vždy jen sloupce *id* a *name* (identifikátor a název) a podmínka (to, co se nachází za where) neexistuje (respektive je 1), v případě, že chceme vybrat všechny hodnoty z tabulky. Nejčastěji používané hodnoty jsou proto implicitně zadané. Při tvorbě požadavku o vykreslení tohoto objetu stačí zadat pouze název tabulky a data jsou automaticky do výběrového pole doplněna. Kód tedy vypadá takto:

$meal = new DBSelect('meal', 'meal');

$meal->setLabel('Typ jídla');

Prvním parametrem je jméno formulářového prvku, druhým jméno tabulky v databázi, kterou chceme vypsat. Tato tabulka má jen dva sloupce, id a typ jídla, který požadujeme, proto v tomto jednoduchém případě nemusíme psát žádná jiná nastavení. A jen poté pomocí smarty vykreslit tento objekt.

Samozřejmě ne vždy máme takovéto jednoduché tabulky, proto jsou zavedeny ještě další atributy, které lze objektu přiřadit a podle toho vykreslit.

Atributy:

- *table* – určuje tabulku v databázi nad kterou se provádí dotaz
- *query* - určuje podmínku v databázovém dotazu, tedy to co následuje za WHERE. Implicitně nastaven na 1.
- *keyColumn* - určuje sloupec primárního klíče, implicitně je nastaven na *id*.

- *valueColumn* - je sloupec, který chceme získat, implicitně je nastaven na *name*, proto jsme nemuseli v předchozím případě žádný z těchto atributů nastavovat a získali jsme požadovaný výsledek.
- *needRefresh* – příznak, zda byly vstupní parametry pro data změněny, a proto je třeba je načítat znovu.

Metody:

- *loadDataFromDB* – podle určených vlastností (*name*, *table*, *query*, *keyColumn*, *valueColumn*) jsou načtena data z databáze. Databázový dotaz na výběr dat je obecně napsán: *SELECT {$this->getKeyColumn()} AS id_key, {$this->getValueColumn()} AS text_value FROM {$this->getTable()} WHERE {$this->getQuery()}");* Tímto dosáhneme úplné obecnosti a zadávání hodnot, které jsou opravdu odlišné.
- *draw* – je zavolána funkce *loadDataFromDB* pro naplnění výběrového pole a zavolána metoda *draw* rodičovské třídy.

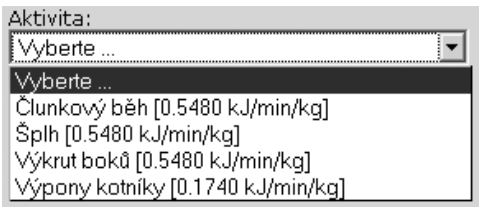
FoodDBSelect, *FoodGroupDBSelect*, *ActivityDBSelect* a *ActivityGroupDBSelect* jsou potomky předchozí třídy. Jen ji zpřesňují pro dotaz, který požadujeme. Jsou to třídy, které využíváme vždy, chceme-li získat seznam potravin, skupinu potravin respektive aktivit, skupinu aktivit. Seznam potravin a aktivit pro lepší názornost v závorce obsahuje energetickou hodnotu (příjem respektive výdej). Proto v těchto třídách provádíme konkatenaci dvou sloupců, z tohoto důvodu byla předchozí třída blíže specifikována.

Jestliže chceme tedy vytvořit seznam aktivit, stačí pouze napsat:

$activity = new ActivityDBSelect('activity', 0);

$activity->setLabel('Aktivita');

A předat proměnnou šabloně.



obrázek 10 Výsledek třídy ActivityDBSelect po vykreslení

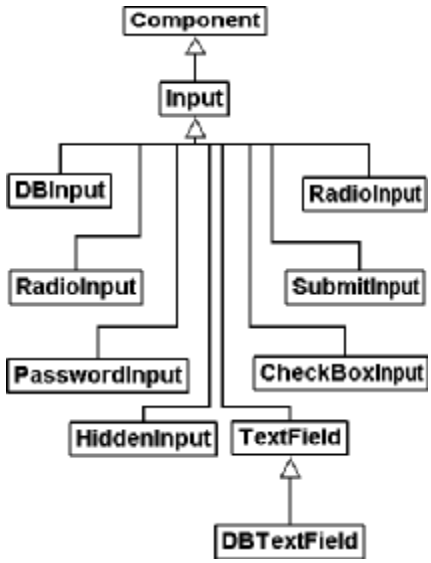
V případě výpisu skupin požadujeme, aby se po jejím vybrání, načetl seznam aktivit (potravin), který je přiřazen vybrané skupině. Tuto funkčnost zajišťuje vložená ajaxová funkce.

7.3.1.2 Input

Druhá třída, která dědí vlastnosti a metody od třídy *Component*, je třída *Input*. Slouží pro zobrazování všech formulářových prvků vstupního pole. Mohou být různého typu: text (implicitní hodnota), hidden, password, submit, reset, radio, checkbox, image, file a button. Pro zobrazování těchto typů

jsou vytvářeny potomci, které mají specifické vlastnosti lišící se od globálních. Toto opatření bylo učiněno z důvodu odlišné práce při získávání hodnot v určitých prvcích a manipulace s nimi.

Ve třídě *Input* metoda *setAccetableValue*, které předáme požadovaný řetězec pomocí regulárních výrazů, ověří, zda zadaná data vyhovují požadavkům. Používá se např. pro ověření správně napsané emailové adresy, zadaného data apod. Metoda *drawContent* a *draw* opět vše vykreslí podle zadaných požadavků.



obrázek 11 Zobrazení dědění ve větvi *Input*

PasswordInput

Určuje typ inputu jako heslo. Vlastnosti zůstávají stejné jako u typu text, jen s tím rozdílem, že při vyplňování tohoto pole se nezobrazují přímo znaky, ale jsou místo nich vypisovány hvězdičky. Také se musí určit hodnota, aby nebyl ukládán do *value* jako atributu prvku Input psaný text. Toto se nastaví v metodě *drawContent*.

HiddenInput

Tato třída slouží pro prvek *HiddenInput*, předává se v ní pouze typ *hidden* a je volán konstruktor rodičovské třídy. Takže je s ní dále pracována jako s textovým polem, jen se nezobrazuje

CheckboxInput

CheckboxInput je třída pomáhající lépe pracovat s prvky input typu checkbox (zaškrtávací pole). Jediným atributem, obohacující rodičovskou třídu, je atribut *checked*, který říká, zda je daná položka zaškrtnuta či nikoli.

Metodou *getValue* se získá hodnota (0 nebo 1), která indikuje vybrání pole. Tento příznak se pozná podle nastavené hodnoty atributu *checked*. Při vykreslování se zjišťuje, zda je tato hodnota nastavena a podle toho je buď označena nebo ne. To je důležité hlavně při editaci, při vytváření nového formuláře, nejsou hodnoty nikdy zaškrtnuty.

Co se týče metod pro vykreslování, liší se pouze *drawContent*, druhá metoda *draw* zůstává stejná jako u rodičovské třídy. *DrawContent* obsahuje hlavičku pro výpis prvku checkbox a jeho správné vypsání a nastavení, protože tento zápis se liší od nadřazených prvků a nemohl být proto použit.

Vytváření checkboxu je velmi jednoduché:

$sportsman = new CheckboxInput('sportsman');

$sportsman->setLabel('Sportovec');

RadioInput

Tato třída slouží pro vytvoření výběru ze skupiny možností, vždy pouze jedné hodnoty. Veškeré hodnoty musí být v jednom poli. Pro toto pole existuje atribut *groupData*, kde jsou jednotlivé položky, ukládány podle svého klíče. Specifika vypsání tohoto typu prvku jsou pak určena v metodách *drawRadioForKey* a v metodě *drawContent*.

SubmitInput

Slouží k vytvoření potvrzujícího tlačítka odesílající formulář. Předává se jen hodnota a typ rodičovské třídě, která se již stará o zbývající vypsání, vykreslení a práci s tímto prvkem.

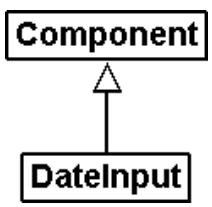
DBInput

Tato třída podobně jako *DBSelect* vybírá určená data z databáze. Opět se musí zadat jméno, tabulka, případně určit sloupce. V této aplikaci je například použito pro získání energie vybraného jídla, tato hodnota není ve formuláři samostatně zobrazena, protože má atribut *hidden*. Je již ale zobrazena u každé potraviny. Ve výběrovém poli pro potravinu se objeví název a v závorce energetická hodnota té dané potraviny.

TextField

Třída sloužící pro textové pole. Má jediného potomka *DBTextField*, která vybírá data z toho *DBTextField*, tato třída je naimplementována opět pro výběr dat z databáze.

7.3.1.3 DateInput



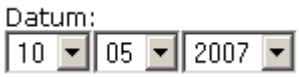
obrázek 12 Zobrazení dědění ve větvi *DateInput*

Tato třída je hojně využívaná pro zobrazení datových polí. Jsou vedle sebe zobrazena tři výběrová pole, pro určení dne, měsíce a roku. Tato třída je naimplementována tak, aby vytvořením objektu bylo

vytvoření kompaktního a kompletního zadávání data a lehké získávání jednotlivých hodnot. Vytvoření výběrového datového pole je velmi jednoduché:

```
$date = new DateInput('date');
$date->setLabel('Datum');
```

Výsledek tohoto objektu po použití metody draw() vypadá takto:



obrázek 13 Objekt třídy *DateInput*

Jestliže není explicitně řečeno jaké datum chceme mít nastaveno při zobrazení, nastaví se implicitní hodnota (aktuální datum).

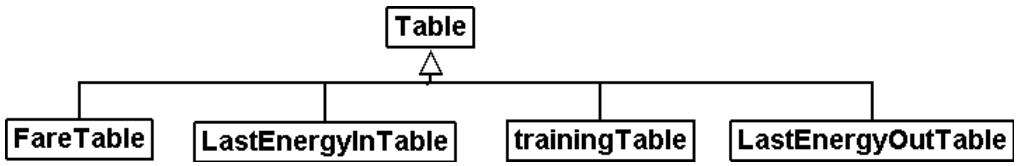
Atributy:

- *day* – určuje položku dne z datumu
- *month* – určuje položku měsíce z datumu
- *year* – určuje položku roku z datumu
- *savedValue* – uložená hodnota celého data ve formátu YYYY-MM-DD, v takovémto formátu je také ukládán do databáze.

Metody z rodičovské třídy *Component* jsou aplikovány na každou část data zvlášť a poté předány nadřazené třídě k celkovému zpracování.

7.3.2 Table

Dalším častou akcí, kromě tvorby formulářů, která se v projektu objevuje je výpis hodnot v tabulkách. Tvorba tabulek je stereotypní a stále se stejně opakující psaní, byla proto vytvořena třída *Table*, která se snaží tvorbu tabulek zjednodušit a urychlit. Tato třída je rodičovskou pro třídy, které již vypisují konkrétní věci a nebylo možné je zgeneralizovat.



obrázek 14 Potomci třídy *Table*

Při implementaci bylo uvažováno, zda tuto tabulku nevytvářet z dědící třídy *Component*, protože má mnoho vlastností společných. Odlišné vlastnosti nakonec ale převládli a byla vytvořena třída nová.

Při inicializaci objektu se musí zadat parametr *query*, což je dotaz žádající o data na databázi, která vyplní danou tabulku. Jestliže není uveden tabulka nemůže být vytvořena.

Atributy:

- *position* – aktuální pozice pro stránkování

- *limit* – počet zobrazovaných řádků
- *editLink* – odkaz pro editaci dat.
- *columns* – pole, ve kterém jsou uloženy názvy sloupců – hlavička tabulky
- *gr* – určuje, zda je řádek sudý nebo lichý, pro různé barevné odlišení
- data – pole do kterého je uložen výsledek dotazu
- *queryColumns*, *queryTables*, *queryWhere*, *queryGroup*, *queryHaving*, *queryOrder*, *queryLimit* – parametry, které jsou získány z parametru query
- *allowedRemove* (*allowedDelete*) – příznak, zda je dovoleno editování (mazání)

Metody:

- *setQuery* – zadaný dotaz, který musí být jako parametr funkce, byl rozdělen na určité části a zjištěno, jestli nějaká nechybí. Řetězec musí obsahovat rezervované slovo SELECT, sloupce, slovo WHERE a název tabulky, ostatní parametry jsou nepovinné. Při zpracování jsou přiřazeny objektu sloupce, které jsou vybírány, tabulka a v případě zadání údajů také WHERE, LIMIT, HAVING a ORDER BY. Údaje následující za těmito slovy jsou přiřazeny k jednotlivým vlastnostem třídy.
- *loadDataFromDB* – vloží výsledek dotazu do pole data.
- *setQuery* – rozloží dotaz na jednotlivé části pro jednodušší získávání požadovaných hodnot.
- *getQuery* – vytvoří dotaz ze získaných údajů, které má objekt.
- *getQueryForSum* – vrátí stejný dotaz, který je uložen v *query* ale bez nastaveného limitu, aby se mohl nastavit počet stránek.
- *drawTableRow* – touto metodou jsou vykreslovány řádky tabulky.
- *drawTableRowAsHeader* – je volána předchozí metoda s parametry zajišťující, aby nebylo zobrazováno editující a mazající tlačítko.
- *drawTableHeader* – vypíše hlavičku tabulky, s daty, které byly uvedeny a atributech při vytváření třídy.
- *drawDataRows* – vykresluje již řádky tabulky, volá metodu *drawTableRow* s parametry udávající jednotlivé řádky, které mají být zobrazeny.
- *drawListening* – nastavuje stránkování.
- *drawContent* – vytváří již celou tabulku, volá funkce *drawDataRows* a *drawTableHeader*.

7.3.2.1 LastEnergyInTable

Nastavuje specifický dotaz pro přijatou energii uživatele aktuálního dne a to pouze posledních pět údajů.. Tabulka pro vypsání navíc obsahuje řádek se součtem všech hodnot aktuálního dne.

7.3.2.2 LastEnergyOutTable

Tato třída dělá totéž co u přijaté energie, ale tentokráte se vše vztahuje na vydanou.

7.3.2.3 FareTable

Třída sloužící k zobrazování tabulek u jídelníčků, které ukazují vždy jednotlivé části dne a jejich rozepsané potraviny s požadovaným množstvím.

7.3.2.4 TrainingTable

Třída sloužící k zobrazování tabulek u tréninků, které ukazují vždy jednotlivé části dne a jejich rozepsané tréninkové jednotky s časem, po který se mají provádět.

7.4 Šablony

Oddělení prezentační logiky od aplikační je důležitou součástí každého projektu, obzvláště pak při psaní v jazyce PHP. I když je samotné PHP vlastně také strojem na šablony, neboť text uvnitř značek <?php a ?> je zpracováván a vykonán, ale text vně je zobrazen v nezměněné podobě. Ovšem v tomto jazyce se velmi lehce může stát, že se tyto dvě věci promíchávají a výpis je protkán různými podmínkami, příkazy, které s prezentací nemají nic společného. Proto je důležitým faktorem používání šablon, které tyto dvě logiky od sebe oddělují a navíc web založený na šablonách je více flexibilní. Chceme-li změnit jen design stránek, stačí nahradit pouze šablonu a veškerá jiná data zůstanou netknutá. Z tohoto důvodu jsem se rozhodla k používání šablonovacího systému a vybrala jsem si Smarty.

V úvodu práce je nejdříve nutné vytvořit adresáře pro šablony a poté je potřeba určit správné hodnoty pro konfigurační nastavení Smarty. Veškerá komunikace mezi PHP a Smarty se děje prostřednictvím objektu, který je instancí třídy Smarty, jenž je definován v souboru Smarty.class.php. V okamžiku, kdy je objekt vytvořen, musí se nastavit jeho vlastnosti tak, aby ukazovali na adresáře, které máme vytvořeny.

Jakmile je vytvořena instance objektu a nastaveny vlastnosti, může se již začít se Smarty komunikovat. Při práci se Smarty je tedy nutné nejdříve inicializovat objekt, přiřadit proměnné objektu Smarty a vytvořit šablonu s danými proměnnými. Proměnné přiřazené objektu Smarty jsou přiřazeny k šabloně v momentě vykreslování šablony.

Pro veškerou inicializaci pomocí Smarty byla vytvořena třída *MySmarty*. Při vytváření objektu z této třídy je v konstruktu provedena inicializace objektu a k vlastnostem přiřazeny cesty k adresářům.

Tato třída navíc obsahuje metodu *addComponent*, pro přiřazování k šabloně. Touto metodou jsou přiřazeny objekty, které vznikly z třídy *Component* nebo Table (a jejich podtřídy). Šabloně se tedy tyto komponenty předají jednoduchým způsobem: *$smarty->addComponent($food)* a ještě jednodušším způsobem jsou vykresleny *$food->draw*.

7.5 Práce s databází

Framework Pear má mnoho balíčků pro práci s databází. Hojně používaný je DB, který byl také využíván na začátku této práce. Poté ale byla potřeba určitá specifika funkčnosti, která DB neobsahuje. Proto byla DB zaměněna za balíček MDB2. Bohužel bylo nutné provést několik změn, protože ne vždy používají tyto dva balíčky stejnou notaci.

Pro připojení k databázi slouží třída *Database*, která obsahuje statickou třídu, připojení se provádí pomocí funkce *connect*, která má parametry jméno databázového schématu, login uživatele do databáze, heslo a databázový server.

K dotazování v MDB2 se používá funkce *query*, jako argument se jí předává sql příkaz. Tato funkce je asi nejvíce používaná, má dva možné výstupy *MDB2_result* (stejně jako select dotaz), což je správný výsledek a při chybě *MDB2_Error*. Další funkcí pro dotazování je *exec*, která jako správný výsledek vrací číslo ovlivněných řádků při manipulaci s daty (stejně jako insert dotaz). Tato funkce v balíčku DB nebyla. Pro opakované spouštění příkazů existují funkce *prepare* a *execute*.

Pro získávání výsledků dotazů je používána především metoda *fetchRow*, vrátí ze získaných dat jeden řádek. Existují samozřejmě mnohé další metody pro získávání výsledků, ty již ale nejsou v moji práci používány, proto je zde pouze vyjmenuji: *fetchOne*, *fetchCol* a *fetchAll*.

Metoda, která byla přidána, oproti balíčku DB, je zjištění posledního vloženého řádku, respektive jeho id. Tento balíček práci s databází ulehčuje a nemusí se tedy navíc ještě implementovat podobné třídy, které by podobnou abstrakci umožňovali.

7.6 Tvorba grafů

Grafy zobrazují jednotlivé hodnoty a jsou jasným a přehledným ukazatelem dosažených výsledků. Z tohoto důvodu jsou také používány v této aplikaci. Po různých úvahách jaké hodnoty a jakým typem grafu zobrazovat, byly nakonec určeny pro vykreslování tři druhy grafů, které jsou blíže popsány v následujících podkapitolách.

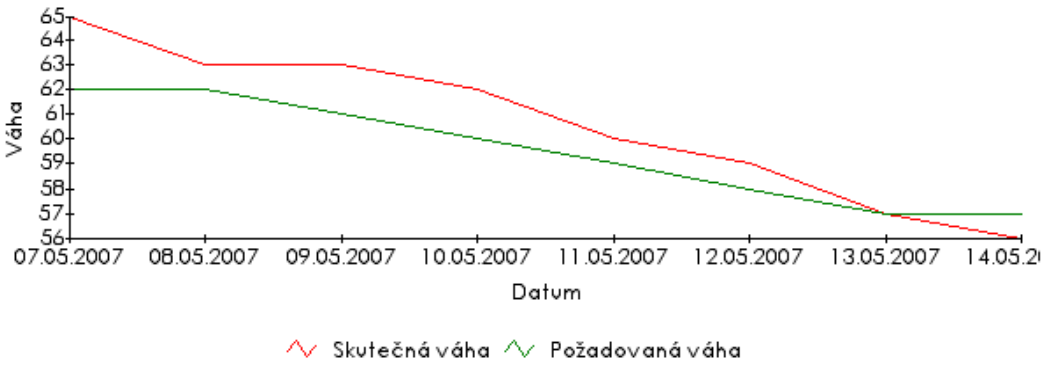
K zobrazování veškerých grafů, které se nacházejí v aplikaci bylo použito balíčku Image_graph z frameworku Pear. Je to velmi komplexní knihovna, pomocí které lze zobrazovat mnoho druhů grafů.

7.6.1 Váha

První spojnicový graf zobrazuje váhu uživatele v jednotlivých časových úsecích. Váha uživatele je porovnávaná s požadovanou váhou, kterou si uživatel zadal. V jednom grafu je tedy velmi přehledně zobrazeno porovnání daných hodnot. A uživatel může ihned vidět, zda se odklonil od požadované hmotnosti, nebo zda vše souhlasí a jeho hmotnost se v čase vyvíjí správně.

Uživatel si aktuální váhu zadává v určitých časových intervalech, z těchto údajů je pak daný graf vykreslen. Souřadnice tvoří vždy datum a hodnota. Ovšem údaje jsou zadávány také pro požadovanou hmotnost a zadané dny se většinou liší. Vykreslit se ale musí všechny hodnoty. Graf totiž nemůže v neměřených hodnotách skákat k 0. To samozřejmě není skutečná hmotnost, ale měřená hodnota také neexistuje. Z tohoto důvodu je vždy vypočítávána aproximace, která je zobrazena. Hodnoty jsou ukládány do pole, kde datum je vždy klíčem k dané hmotnosti. Tato pole musí být dvě, jedno pro požadovanou a jedno pro aktuální hmotnost. Jestliže data v polích nesouhlasí je dané datum do pole přidáno a hodnotě je přiřazena nula. Z počtu nul, které po sobě následují a z krajních hodnot se vypočítá koeficient, o který se má daná hmotnost v jednotlivých dnech změnit. Takovým to způsobem jsou vypočítány všechny hodnoty uprostřed grafu. Problém ale nastává u krajních hodnot, pro ně průměr nemůže být vypočítán vzhledem k tomu, že je ohraničen pouze jednou hodnotou. Proto krajní hodnoty přijímají hodnotu předcházející nebo následující, záleží na tom, zda se jedná o první hodnotu nebo o poslední.

Počítání těchto aproximovaných hodnot bylo implementováno pomocí Mealyho automatu[5], kde se přechází mezi stavy podle zadané hmotnosti, rozlišuje se, zda byla přijata 0 nebo nějaká jiná hodnota, tedy byla zadaná hmotnost. V případě zadané hmotnosti je hodnota ihned vykreslena, v opačném případě jsou počítány nuly následující za sebou a poté vypočítána přibližná hodnota.



obrázek 15 Graf znázorňující váhu

7.6.2 Rozložení příjmu a výdeje energie

Dalším typem grafu je výsečový graf, který zobrazuje procentuální rozdělení příjmu a výdeje energie. Tento poměr je důležitý hlavně pro osoby, které chtějí snížit svou váhu, ale samozřejmě také pro každého jednotlivce zajímající se o své tělo a tím pádem také o poměr přijaté a vydané energie. Barevně odlišený graf je rozdělen na dvě části, příjem a výdej energie, a takto přehledným způsobem znázorněn poměr těchto dvou hodnot. Uživatel má díky tomu jasnou představu o celkové energii. Osoba, která má v úmyslu snížit svou váhu, by měla mít příjem energie nižší než výdej. V případě

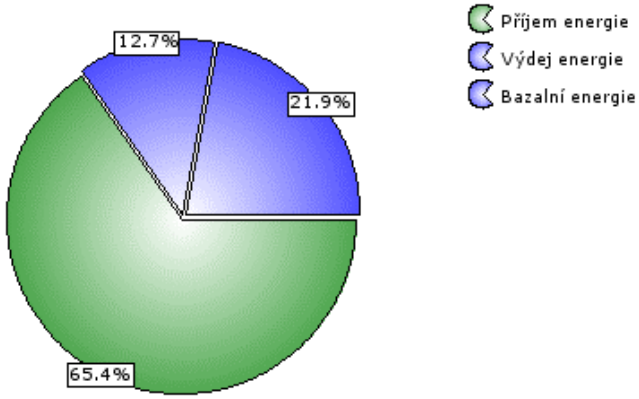
[5] Mealyho automat – konečný automat s výstupem, výstup je generován na základě vstupu a stavu ve kterém se automat nachází

přání o zvýšení své hmotnosti je tomu naopak. Udržení hmotnosti by měl signalizovat graf v rovnováze. Část grafu týkající se vydané je ještě rozdělena dvě části a to energii vydanou nějakou aktivitou a bazální energii, kterou člověk spotřebuje vždy.

Vedle grafu je samozřejmě zobrazovaná legenda, která udává o kterou energii se jedná a hodnota v kJ.

Graf je vytvořen jsou-li mu dodána minimálně tři hodnoty. Nezáleží na tom, jestli pro přijatou nebo vydanou, ale až ze tří hodnot je graf znázorněn.

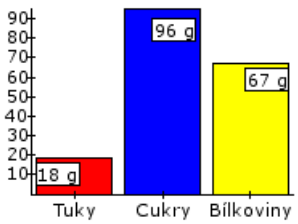
Příjem energie:	6767.01 kJ
Výdej energie:	1315.20 kJ
Bazalní energie:	2268.068 kJ
Suma:	**3183.742 kJ**



obrázek 16 Graf znázorňující poměr energií

7.6.3 Složení potravy

Složení potravy je také velmi důležité, a proto je reprezentováno sloupcovým grafem. Barevně odlišené, vedle sebe stojící tří sloupce reprezentují obsah tuků, cukrů a bílkovin v přijaté potravě. Implicitně jsou zobrazovány hodnoty aktuálního dne, ale lze samozřejmě také jednoduše nastavit jiné rozhraní.



obrázek 17 Graf složení potravy

7.7 Zabezpečení

V dnešní době je nutným předpokladem webové aplikace zabezpečení systému. Bezpečnost je důležitá nejen před „hackováním“ serveru, ale také před neúmyslným spuštěním určitého procesu, případně zadání chybných informací.

7.7.1 Přihlášení

Pro práci se systémem je nutná autentizace uživatele. Každý uživatel má přiděleno své uživatelské jméno a heslo, pomocí kterého se přihlašuje. Po zdařilém ověření uživatelovi totožnosti je uložen do cookies. Heslo je ukládáno jako zašifrovaný řetězec metodou md5, z tohoto řetězce již není možné zjistit původní heslo. Při ověřování je zadané heslo také zašifrováno a porovnáno s uloženým.

7.7.2 Udržování autentizace

Udržování autentizace, tedy předpoklad, že komunikujeme stále se stejnou osobou je důležitým předpokladem systému. Proto jsou používány již dříve zmíněné cookies. Při každé uživatelově akci se zjišťuje, jestli neuběhla určitá doba (nastavena na 15 minut), po kterou nebyla provedena žádná akce od uživatele. V případě delší časové prodlevy je uživatel přesměrován na úvodní stránku a je požádán o opětovné vypsání údajů.

7.7.3 Ověřování vstupů

Pro zajištění bezpečnosti systému jsou vstupy ověřovány, zda opravdu obsahují hodnoty, které mají. Toto omezení zabraňuje případným útočníkům, aby vložili do polí provádění nějakého skriptu, ale také samotným uživatelům nedovolí zadání nesmyslných dat. Hodnoty z formulářů jsou ověřovány, buď pomocí regulárních výrazů, jestliže je tento případ složitější, nebo v jednodušším případě je kontrolován typ přijaté informace. Je-li vložená hodnota řetězec je zpracována funkcí trim, která odstraní netisknutelné znaky ze začátku a konce řetězce.

7.8 Požadavky na systém

Tento informační systém byl navržen především tak, aby byl umístěn na WWW stránkách na internetu a jednotliví uživatelé k němu přistupovali pouze pomocí internetového prohlížeče. Proto jsou stránky umístěny na určitém serveru a uživatelé jsou klienti. Jestliže si někdo nainstaluje tento systém na svůj lokální počítač, což je samozřejmě možné, poté tento počítač slouží zároveň jako server a zároveň jako klient.

7.8.1 Server

Jelikož se jedná o webový server, předpokládá se, že na něm běží server Apache. Na tomto serveru je nutné mít nainstalované PHP verze minimálně 5.0 a na něm povolenou grafickou knihovnu GD verze 2.0 a vyšší. Dále PHP musí mít povoleno zpracovávání session. Jako databázový systém musí běžet MySQL (od verze 4.0). Co se týče různých rozšíření, je nutné použít knihovnu SAJAX a dále přichystat na používání balíčky Pear, konkrétně MDB2 a image_graph.

7.8.2 Klient

V případě klienta je pouze nutné mít nainstalované internetové prohlížeče, které podporují CSS2 a XHTML a dále povoleno zpracování cookie a použití JavaSciptu. V současné době většinou veřejností používaných prohlížečů tyto vlastnosti mají.

8 Získané zkušenosti

Při tvorbě informačního systému bylo potřebné nastudovat teorii, která se týká tréninkových plánů a výživových metod. Ze získaných znalostí v dané oblasti bylo nutné probrat ty, které jsou důležité, a proto do systému musí být zahrnuty, a které nejsou pro danou oblast zásadní. Při tvorbě části systému zabývající se výdejem energie bylo zjištěno, že do energetického výdeje musí být zahrnuta nejen vydaná energie při nějaké činnosti, ale také tzv. Bazální energie. Jedná se o energii potřebnou pro zachování základních vitálních funkcí (funkce orgánových soustav, udržování osmotické rovnováhy, chemické energie pro biosyntézy a udržování tělesné teploty) včetně růstu a vývinu. Výpočet této bazální energie je závislý na pohlaví, hmotnosti, tělesné výšce a stáří. Proto byla tato hodnota také zahrnuta mezi potřebné údaje.

Při analýze skladby potravin a určování, které hodnoty je tedy nutné do systému zakomponovat, byl zkoumán pojem energetického metabolismu[6]. Organismus dokáže získávat energii z tuků, sacharidů a bílkovin. U jednotlivých potravin je tedy nutné tyto hodnoty zadávat, aby uživatel měl o těchto hodnotách přehled, protože jsou nutné pro sledování zdravotních funkcí reagující na příjem potravy, ať už dobrý, nebo špatný.

Pro predikci úbytku váhy si bylo nutné nastudovat, jak se váha mění při výdeji energie. Tento údaj není vůbec jednoduché určit, protože to záleží na mnoha okolnostech, jak fyzických, tak genetických. Je ale udávána přibližná hodnota, která je používána při výpočtech, 32 tisíc spálené energie na 1 kilogram úbytku.

Mezi další získané znalosti je samozřejmě nutné uvést zkušenosti implementační. Důležité bylo seznámení s objektovým přístupem v PHP. Tento jazyk, ač nebyl původně pro objektový přístup navrhnut, splňuje všeobecně potřebné vlastnosti pro práci s objekty a je tedy možné pro usnadnění využít objektového přístupu.

Nová také pro mě byla práce s grafy. Používání balíčku image_graph frameworku Pear nebylo z počátku vůbec jednoduché. K tomuto balíčku zatím totiž neexistuje manuál, který by hledání potřebných informací značně ulehčil. Naštěstí ale existuje fórum, kde se dají najít potřebné informace, případně se na ně přímo zeptat. Při prohledávání tohoto fóra jsem proto několikrát našla nejen odpověď na hledanou otázku, ale také jiné zajímavé věci zabývající se problematikou grafů.

Také jsem se musela naučit používat šablonovací systém Smarty, který jasně oddělil aplikační logiku od prezentační. Využívání tohoto systému se zpočátku zdálo nepotřebné, ale čím více systém rostl, tím bylo toto oddělení potřebnější a zdrojový kód stále zůstával přehledný.

[6] Energetický metabolismus – metabolismus, ve kterém z chemické energie živin vzniká energie biologická využitelná v organismu.

9 Možnosti dalšího rozšíření

Vytvořený systém je již plně funkční a je možné jej plnohodnotně využívat na podporu tréninkových aktivit. Na každém systému se ale vždy najdou nějaké chyby a jeho možná rozšíření. Tak je tomu i v tomto případě. Jednotlivá možná rozšíření již nejsou stěžejní podstatou systému, ale mohla by vést k příjemnějšímu používání systému, případně doplnění o další možnosti využití.

Systém v současné době slouží pouze pro jednotlivce. Jeden sportovec má jednoho trenéra a jednoho dietologa. Mohl by ale také být rozšířen o používání pro kolektivní sporty. Systém by mohl sloužit pro skupinu lidí se stejným zájmem, takže by trenér nemusel plánovat trénink pro každého jednotlivce. Ale při rozšíření o skupinu by naplánoval trénink celé skupině. Jednotliví členové by se přihlásili a zjistili pokyny, které jim jejich trenér zadal. Všichni by měli pokyny stejné, ale trenér by je zadal pouze jednou. Členové skupiny by samozřejmě splněné úkoly odsouhlasili a trenér by pak viděl na jediné stránce celé své mužstvo s jejich výsledky.

Další možností by bylo automatické generování nákupního lístku. Ten by vycházel z naplánovaného jídelníčku. Vzal by jednotlivé položky s množstvím a automaticky by je vypsal do nákupního lístku.

Také by bylo vhodné vykreslovat graf při výpočtu BMI. Ten by byl barevně vykreslen pro oddělení různých hmotnostních kategorií a po zadání požadovaných hodnot by výsledek byl v daném grafu zobrazen. Tímto by bylo jasné a přehledné, v které částí se osoba nachází a jestli by bylo potřebné svoji hmotnost dále nějak upravovat.

Další možnou položkou by mohl být modul pro recepty. Zobrazování receptů, hledání v receptech a samozřejmě doplňování svých vlastních již osvědčených receptů.

10 Závěr

V této práci jsem se seznámila s technikami tvorby informačních systému v prostředí Internetu. Tyto poznatky byly využity při následné implementaci systému a tím získání kvalitního prostředku pro uchovávání informací o energetickém příjmu a výdeje uživatele, ale také pro plánování jídelníčků a tréninkových jednotek.

Byly zde formulovány požadavky, které by měl mít tento informační systém a veškeré informace, které bude poskytovat. Poté byl již realizován samotný návrh pomocí jazyka UML. Jedná se o celkem rozsáhlou databázi, proto jsem se snažila vytvořit databázové schéma, co nejpřehlednější a samozřejmě také co nejjednodušší na zadávání a získávání potřebných informací.

Samotná realizace návrhu nebyla vůbec jednoduchou záležitostí. Při bližším zkoumání dané problematiky, vždy vyvstaly nové problémy, které bylo nutné do systému zahrnout, což zapříčinilo změnu již stávajících věcí, které tímto byly ovlivněny. Výsledkem byl ovšem kvalitní návrh, podle kterého byla provedena implementace systému.

Práce obsahuje také popisy jiných systémů zabývajících se podobnou problematikou. Z některých můj návrh vycházel, z jiných jsem si vzala jen ponaučení, kterým chybám se vyhnout, aby systém byl co nejlepší a dobře použitelný.

Při samotné implementaci bylo nutné použití různých knihoven, které umožňují nestandardní možnosti, které PHP neobsahuje, nebo obsahuje pouze v omezené míře. Hlavním problémem bylo kreslení grafů. Po prohledání všech možností tvorby grafů, byl vybrán balíček frameworku Pear, který v konečné verzi systému zobrazuje požadované výstupy.

Přínosem celé aplikace je nejen funkční systém, který je možný k okamžitému používání, ale také naimplementované třídy pro zobrazování všech prvků formuláře a třídy pro automatické vykreslování tabulek.

Literatura

[1] *Diet and Weight Loss Tutorial*. Dostupné z WWW: http://www.caloriesperhour.com/tutorial_pound.html, 10.5.2007.

[2] DuBois, P. *MySQL profesionálně*, 1. vyd. Praha: Mobil Media a.s., 2004. 158s. ISBN 80-86593-41-x

[3] *Fitář*. Dostupné z WWW: http://www.slunecnice.cz/sw/fitar/, 27.12.2006.

[4] *Fitlinie*. Dostupné z WWW: www.fitlinie.cz, 27.12.2006.

[5] Gutmans, A. Bakken, S. Rethans, D. *Mistrovství v PHP 5*. 1. vyd. Brno: Computer Press, 2005. 655s. ISBN 80-251-0799-X.

[6] *Image_graph*. Dostupné z WWW: http://pear.veggerby.dk/, 7.5.2007.

[7] Kanisová, H. Muller, M. *UML srozumitelně*. 1. vyd. Brno: Computer Press, 2004. 158s. ISBN 80-251-0231-9.

[8] Kosek, J. *PHP – Tvorba interaktivních internetových aplikací*. 1. vyd. Praha: Grada Publishing, 1999. 490 s. 80-7169-373-1.

[9] *MySQL*. Dostupné z WWW: www.mysql.com, 26.12.2006.

[10] *Pear*. Dostupné z WWW: http://www.pear.php.net/, 7.5.2007.

[11] *PHP*. Dostupné z WWW: www.php.net, 26.12.2006.

[12] *Postava pro každého*. Dostupné z WWW: http://www.postavaprokazdeho.cz/, 27.12.2006.

[13] Schlossnagle, G. *Pokročilé programování v PHP 5*. 1. vyd. Brno: Zoner Press, 2004. 640s. ISBN 80-86815-14-5.

[14] Sklar, D. *PHP 5 - moduly, rozšíření a akcelerátory*. 1. vyd. Brno: Zoner Press, 2005. 341s. ISBN 80-86815-19-6.

[15] *Smarty*. Dostupné z WWW: http://smarty.php.net/manual/en/, 7.5.2007.

[16] *Úvod do CSS*. Dostupné z WWW: http://www.webtvorba.cz/css/uvod-do-css.html#historie, 7.5.2007.

[17] *Získat štíhlou linii podle počítače je jednoduché, stačí neodmlouvat*. Dostupné z WWW: http://technet.idnes.cz/ziskat-stihlou-linii-podle-pocitace-je-jednoduche-staci-neodmlouvat-1id-/software.asp?c=A061015_171757_software_dvr, 27.12.2006.

Příloha A: Obsah přiloženého CD

Pomocí přiloženého CD je možné si Informační systém na podporu tréninkových aktivit nainstalovat a vyzkoušet. Obsahuje používané verze technologie Pear. Součástí je také text této diplomové práce v elektronické podobě.

Samotné CD je rozděleno do pěti složek:

1. Zdrojové kódy
   - system – zdrojový kód IS pro podporu tréninkových aktivit
2. Text
   - doc – text diplomové práce v elektronické podobě ve formátu pdf
3. Popis instalace
   - instal – popis instalace v souboru instal.txt
4. DB
   - db - kompletní sql příkazy pro vytvoření databázových tabulek a naplnění ukázkovými daty
5. Knihovny, které jsou nutné pro správný chod systému
   - balíček Pear - image_graph
   - balíček Pear - MDB2
   - knihovna Smarty

Příloha B: Instalace systému

Pro správnou funkčnost systému je nutné splnit několik následujících kroků. Nejdříve je nutné správně nastavit server. Následuje seznam nástrojů, které je nutno nainstalovat, u každého je vždy uvedena verze na které byl systém odladěn, proto je doporučeno použít tuto verzi nebo vyšší.

- Apache v. 2.0.54
- PHP v. 5.0.4
- MySQL v. 4.1.12
- poté je nutné nahrát na server samotný systém, ten je umístěn v adresáři systém na přiloženém CD.
- pro práci s databází je možno si nainstalovat phpMyAdmin (v. 4.1.12)
- vytvořit tabulky pomocí skriptu structure.sql (umístěno v adresáři sql) a naplnit daty spuštěním skriptu data.sql.
- v souboru config.php (adresář conf) je potřeba nastavit přístupové jméno a heslo k databázi, dále název serveru a vyplnit jméno databáze, která již byla vytvořena v předchozím kroku.

Pro správnou funkčnost systému je dále nutné nahrát knihovny.

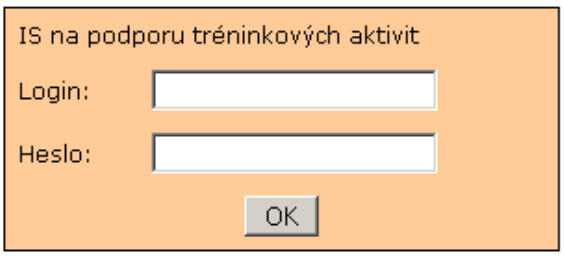
- SAJAX - tato knihovna již je součástí systému (v adresáři system/req/Sajax.php)
- Pear - jsou používány balíčky image_graph (v 0.7.2, s opravou souboru pie.php pro správné zobrazování legendy u výsečového grafu), MDB2 (v. 2.4.1.) a pro propojení tohoto balíčku s MySQL balíček MDB2_Driver_mysql (v. 1.4.1). Všechny tyto tři balíčky jsou umístěny v adresáři lib. Balíček MDB2_Driver_mysql je již součástí MDB2.
- Smarty v. 2.6.18 - nakopírovat do cílového adresáře, adresáře pro šablony jsou již vytvořeny a jsou součástí systému. Knihovna je umístěna v adresáři lib.
- Smarty i Pear musí být dostupné v PATH.

Příloha C: Uživatelská příručka

Tento informační systém pro podporu tréninkových aktivit by měl pokrývat problematiku výživového, sportovního a silového programu jednotlivce. Systém byl navržen tak, aby práce s ním byla intuitivní a snadná. Vždy samozřejmě nějaký problém může nastat, a proto byla napsána tato příručka. Je rozdělena do čtyř částí, a to na část přihlášení a další tři podle typu uživatele.

1. Přihlášení

Pro vstup do systému je nutné se přihlásit. K tomuto je nutné mít správné přihlašovací údaje, kterými jsou přihlašovací jméno (login) a heslo. Ta jsou vepsána do přihlašovacího formuláře a odeslána. Jestliže jsou tyto údaje vypsány správně, uživatel již může v systému pracovat.



obrázek 18 Formulář k přihlášení

Pro ukončení činnosti systému stačí vybrat odkaz „Odhlášení“.

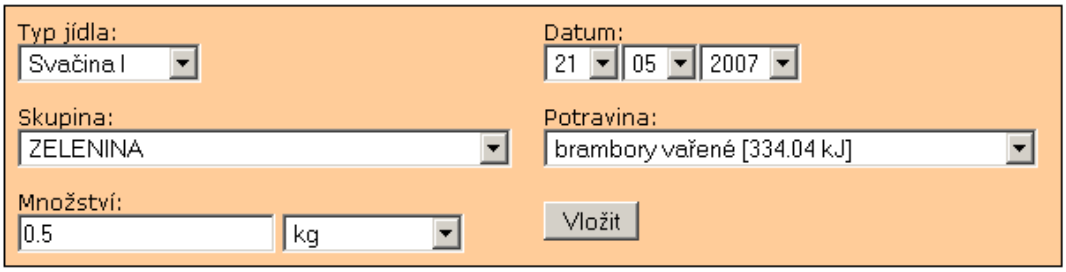
2. Sportovec

Příjem a výdej energie

Po přihlášení je uživatel přesměrován na stránku příjmu a výdeje energie, zde si může vybrat časové období, ve kterém mu má být rozdělení energie zobrazeno. Po vybraní tohoto intervalu je uživateli zobrazeno rozdělení příjmu a výdeje, do této energie je zahrnuta také energie bazální. Výsledek energetického souhrnu je textově vypsán.

Vložení jídla

Tento formulář slouží ke vkládání zkonzumovaného jídla. Je nutné zadat typ jídla, datum a vybrat skupinu potravin, do které daná potravina patří. Tím se načte výběr potravin již z této kategorie. Po jejím vybrání se ještě musí zadat snězené množství a pak již jenom výběr potvrdit.



obrázek 19 Vložení jídla

Vložení aktivity

Tato stránka slouží k vložení vykonané aktivity. Je nutné zadat typ aktivity a poté vykonanou aktivity, kde se seznam načte již podle typu. Dále se musí určit datum a délka prováděné činnosti.

Příjem energie

Na této stránce je zobrazovaná přijatá energie. Ihned po vybrání je zobrazen aktuální den, ale je možné si vybrat i jiné rozmezí, ve kterém se přijatá energie zobrazí. Je také zobrazován graf, který ukazuje rozložení dané potravy.

Výdej energie

Vydaná energie je podobně jako přijatá zobrazována na samostatné stránce. Opět se musí určit časový interval.

Přehled váhy

Jsou dva druhy určení váhy - aktuální a požadovaná. Každá se zadává zvlášť je nutné vždy určit datum, ať už dnešní pro aktuální váhu, nebo budoucí pro požadovanou váhu. Hodnoty jsou okamžitě zobrazovány v grafu.

Jídelníček

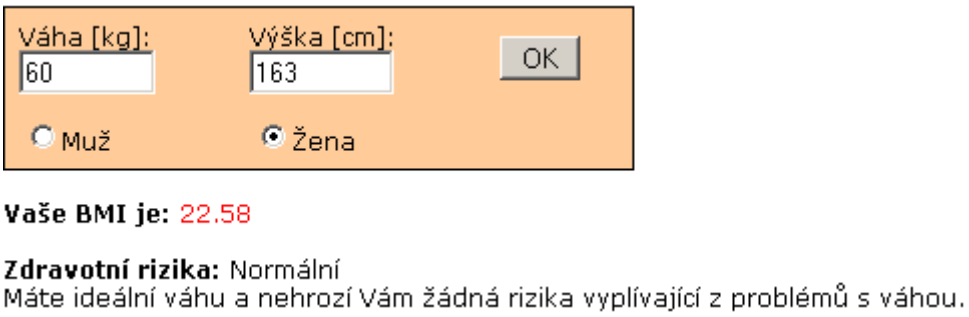
Jídelníček je zobrazován vždy po jednotlivých dnech. Je rozdělen podle typu jídel a vypsána vždy potravina a její množství, které je nutno zkonzumovat. Jakmile je jídlo snědeno, tato skutečnost se potvrdí a je přidána uživateli mezi přijatou energii.

Trénink

Sportovci je ukazován trénink vždy na určitý den, samozřejmě se lze mezi budoucími dny přepínat. Jestliže je aktivity vykonána sportovec ji potvrdí a ta se automaticky přidá mezi jeho vydanou energii.

Výpočet BMI

Jednoduchý formulář na výpočet uživatelova Body Mass Indexu. Je nutné zadat váhu v kg a výšku v cm a vybrat pohlaví.



obrázek 20 Jednoduchý výpočet BMI

Export tabulkových hodnot

Pro export dat je nutné si vybrat, která data mají být exportována. Je možnost dat příjmu a výdeje energie, nebo měnící se váhy.

Přiřazení trenéra

Je možné si změnit svého trenéra tím, že si jej vybereme ze seznamu všech trenérů, kteří jsou v databázi.

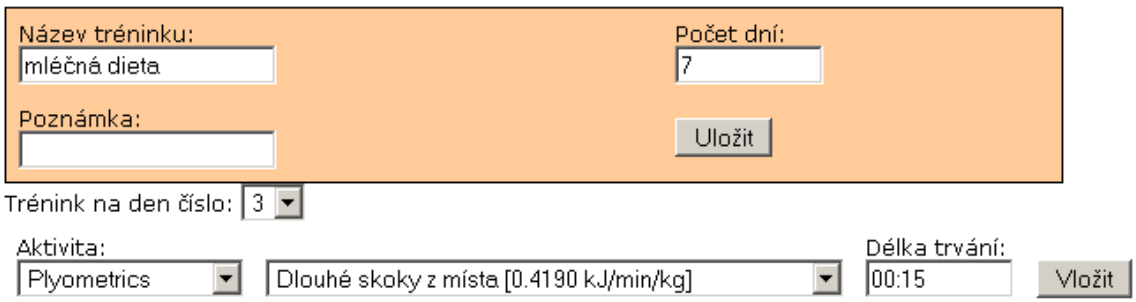
Přiřazení dietologa

Naprosto totožné jako přiřazení trenéra jen vztažené k dietologovi.

3. Trenér

Vytvoření tréninku

K vytvoření tréninku je nutné zadat jeho název a počet dní. Poté již je nutné zadávat jednotlivé cviky po jednotlivých dnech. Při zadávání aktivity se nejdříve musí vybrat typ, poté samotná aktivita. Délku provádění jednotlivé činnosti je nutné zadat ve formátu hh:mm. Po potvrzení je aktivita uložena k danému tréninku a dni.



obrázek 21 Vkládání aktivity k tréninku

Editace tréninku

Je prováděna ve stejném formuláři jako při vytváření tréninku, jen jsou již vložené údaje vypsány.

Přiřadit sportovci

Ze seznamu sportovců, kteří jsou přiřazení přihlášenému trenérovi, je nutno vybrat toho, kterému je trénink přiřazován. Dále ze seznamu vytvořených tréninků (buď veřejných, nebo vytvořených přihlášeným trenérem) se požadovaný trénink vybere a určí datum, od kterého má být plněn.

Přehled sportovců

Je možné si přiřadit pouze sportovce, který ještě žádného trenéra nemá. Trenér si jej tedy přiřadí po vybrání ze seznamu volných sportovců. Své přiřazené sportovce si také může odebírat.

Přidat /editovat aktivitu

Aktivitu je nutné přidat již do existující skupiny aktivit. Musí se určit jméno a energie v jednotce kJ/kg/min. Stejným způsobem se také potravina edituje.

4. Dietolog

Vytvoření jídelníčku

K vytvoření jídelníčku je nutné zadat jeho název a počet dní. Poté již je možné zadávat jednotlivá jídla po jednotlivých dnech a rozdělené podle typu jídla. Při zadávání potraviny se nejdříve musí vybrat typ, poté samotná potravina. Také je nutné zadat množství, které má být zkonzumováno vztažené k jednotce. Po potvrzení je položka jídla uložena k danému jídelníčku a dni.

Editace jídelníčku

Je prováděna ve stejném formuláři jako při vytváření jídelníčku, jen jsou již vložené údaje vypsány.

Přiřadit sportovci

Ze seznamu sportovců, kteří jsou přiřazení přihlášenému dietologovi, je nutno vybrat toho, kterému je jídelníček přiřazován. Dále ze seznamu vytvořených jídelníčků (buď veřejných, nebo vytvořených přihlášeným trenérem) se požadovaný jídelníček vybere a určí datum, od kterého má být respektován.

Přehled sportovců

Je možné si přiřadit pouze sportovce, který ještě žádného dietologa nemá. Dietolog si jej tedy přiřadí po vybrání ze seznamu volných sportovců. Své přiřazené sportovce si také může odebírat.

Přidat /editovat potravinu

Potravinu je nutné přidat již do existující skupiny potravin. Musí se určit jméno a energie v kJ. Dále se postupně zadají všechny údaje týkající se dané potraviny. Důležité jsou hlavně tuky, cukry a bílkoviny. Stejným způsobem se také potravina edituje.