TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

KOMUNITNÍ PORTÁL S WEBOVÝMI SLUŽBAMI

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Jan Gabriel**
Vedoucí práce: Ing. Roman Špánek, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Gabriel**
Osobní číslo:	**M11000077**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Komunitní portál s webovými službami**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s možnostmi vývoje moderní webové aplikace s podporou webových služeb.
2. Navrhněte strukturu webového portálu pro specifickou komunitu Junák, včetně integrace existujícího centrálního severu organizace Junák.
3. Navrhněte grafické rozhraní aplikace a porovnejte možnosti využití v současné době dostupných technologií (HTML5, CSS3, LESS atd.) pro desktopovou i mobilním verzi webové aplikace.
4. Pro zjištěné nekompatibility najděte odpovídající řešení použitelné na obou platformách.
5. Připravte pilotní implementaci takové portálu, ověřte navržené schéma a grafické prostředí na skupině uživatelů.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**30–40 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, ISBN-10: 0201633612, ISBN-13: 978-0201633610**

[2] **Test Driven Development: By Example. Kent Beck. ISBN 0321146530 (ISBN13: 9780321146533)**

[3] **Dokumentace k databázi MySQL: http://dev.mysql.com/doc/**

[4] **A look at aspect-oriented programming http://www.ibm.com/developerworks/rational/library/2782.html**

[5] **RESTful Web Services, Web services for the real world. Leonard Richardson, Sam Ruby. ISBN:978-0-596-52926-0, ISBN 10:0-596-52926-0**

Vedoucí bakalářské práce:	**Ing. Roman Špánek, Ph.D.** Ústav mechatroniky a technické informatiky
Konzultant bakalářské práce:	**Ing. Pavel Tyl** Ústav mechatroniky a technické informatiky
Datum zadání bakalářské práce:	**10. října 2013**
Termín odevzdání bakalářské práce:	**16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -
L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Romanu Špánkovi, Ph.D. za spolupráci při její tvorbě. Dále bych rád poděkoval mému zadavateli Pavlu Konečnému za neúnavnou zpětnou vazbu a nespočet podnětů ke zlepšení uživatelského prostředí aplikace.

Abstrakt

Cílem práce je vytvoření webové aplikace, která bude sloužit k průběžné motivaci a sledování dlouhodobých cílů členů, kteří jsou ve vedoucích pozicích organizace Junák – český skaut.

Práce se soustředí na využívaní nejnovějších trendů ve vývoji webových aplikací, a to převážně v oblasti uživatelských rozhraní. Důraz je kladen na využití aplikace na všech různých zařízeních od mobilních telefonů, přes tablety až po desktopy s velkým rozlišením.

V neposlední řadě bude čtenář seznámen s problematikou zabezpečení webových služeb na rozhraní dvou serverových aplikací v síti internet.

Jednotlivé části práce postupně představují problematiku, návrh aplikace a její následnou implementaci v Nette framework.

Klíčová slova

Less, Nette, PHP, Moje předsevzetí, web webová aplikace, MySQL, HTML5, CSS3, webové služby, zabezpečení, responsive design, media queries, skaut, portál

Abstract

The aim of the thesis is to create a web application that will be used for continuous motivation and monitoring of long-term goals for members who are in leadership positions inside Junák – český skaut organization.

The work focuses on the usage of the latest trends in web development, mainly in the field of user interfaces. Emphasis is placed on the useability of the application on different devices, from mobile phones, tablet through to the desktop with a high resolution.

Finally, the reader will be acquainted with the issue of security of Web services interface between two server applications on the Internet.

The individual parts of this thesis gradually introduces problems of application design and its subsequent implementation in Nette framework.

Keywords

Less, Nette, PHP, My resolutions, web web application, MySQL, HTML5, CSS3, web services, security, responsive design, media queries, scout, portal

Obsah

Seznam tabulek

Seznam ilustrací

Seznam zkratek a pojmů

WS	(W*eb Services*) softwarový systém umožňující interakci dvou strojů v síti
CSS	(*Cascading Style Sheets*) kaskádové styly
AJAX	(*Asynchronous JavaScript and XML*) obecné označení pro technologie umožňující měnit obsah stránek bez znovu načítání
HTML	(*HyperText Markup Language*) hlavním z jazyků pro tvorbu webových stránek
JSON	(*JavaScript Object Notation*) způsob zápisu dat, který je přenositelný mezi rozdílnými systémy
HTTP	(Hypertext Transfer Protocol) internetový protokol určený pro výměnu hypertextových dokumentů
DBMS	(Database Managment System) kolekce software umožnující správu dat
SQL	strukturovaný dotazovací jazyk využívaný pro pokládání dotazů v databázi
SOAP	protokol pro výměnu zpráv založených na XML přes síť, hlavně pomocí HTTP
WSDL	popisuje co nabízí webová služba za funkce a způsob, jak se jí na to zeptat
JSON	způsob zápisu dat nezávislý na počítačové platformě
AJAX	označení technologie pro asynchronní komunikaci se serverem
PHP	(PHP Hypertext Preprocessor) skriptovací programovací jazyk určený především pro programování webových stránek
DOM	(Document Object Model) objektově orientovaná reprezentace XML nebo HTML dokumentu
RUP	(Rational Unified Process) metodika vývoje softwaru
XP	extrémní programování
SCRUM	agilní metodika vývoje software
KISS	(Keep It Simple, Stupid!) obecný návrhový vzor aplikovatelný na návrh řešení v nejrůznějších odvětvích
Latte	šablonovací jazyk, který využívá Nette
hash	unikátní datový otisk
framework	softwarová architektura, která poskytuje nástroje pro programování

1. Úvod

Ve skautské organizaci se pohybuje velké množství dobrovolníků, dávno již však neplatí, že by se všichni zabývali prací s dětmi. V řadách organizace je velké množství učitelů, právníků, informatiků nebo manažerů, kteří zastupují organizaci v nejrůznějších oblastech působení. Napříč organizací existuje mnoho menších zájmových pracovních skupin, které se nazývají odbory.

Jedním z nich je také odbor pro personalistiku, který se stará především o hospodaření s lidskými zdroji, vzděláváním, motivací a uplatněním činovníků ve všech oblastech působení organizace.

Jak již bylo nastíněno, existuje velké množství odborníků, kteří nepracují přímo s dětmi. Jsou tedy částečně izolováni od hnutí, a to jak hlavní pracovní náplní, tak i jeden od druhého geograficky. Nefunguje zde tedy sdílení společné problematiky, nepřímá viditelnost jejich pracovního úsilí na organizaci jako celku. Jejich dobrovolná práce je tedy často frustrující a velice demotivující. Navíc se stává, že už ani členové hnutí netuší, kam až organizace sahá a na čem všem je možné v zájmu dobrovolnictví spolupracovat nebo s kým spojit své síly.

A právě z potřeby motivace takovýchto lidí vychází tato práce. Touto formou by měla vzniknout platforma k motivaci členů, podpoře komunity a sdílení společných úspěchů a neúspěchů. Práce mimo výchovnou oblast totiž není tak všeobecně viditelná a lidé, kteří organizaci každoročně získávají finanční prostředky a cenné kontakty, si také zaslouží ocenění své tvrdé práce.

2. Teorie

2.1. Dostupné Technologie

2.1.1. PHP 5.*

PHP (PHP: Hypertext Preprocessor) je oblíbený skriptovací jazyk pro tvorbu dynamických webových stránek, který neustále prochází vývojem. Od verze 5.0 *PHP* obsahuje vylepšenou podporu pro objektové přístupy k programování, například třídy, dědičnost, jmenné prostory a spoustu dalších. Jedná se však o interpretovaný jazyk, což je obvykle považováno za výkonnostní nevýhodu oproti jazykům kompilovaným. Složitější aplikace mohou mít při velké zátěži problémy s výkonností serveru, to však lze řešit částečnou kompilací (jako to řeší třeba společnost Facebook). Produkt je znám pod jménem HipHop a lze se o něm dočíst například na blogu[1] Jakuba Vrány, jednoho z Českých vývojářů Facebooku.

Pro menší až střední projekty však výkon není zásadním problémem a PHP se těší velké oblibě převážně kvůli své multiplatformnosti, podpoře mnoha databází, strmé křivce učení, slušné dokumentaci a hlavně velké podpoře ze strany webhostingů. Více o PHP se lze dozvědět v knize věnované přímo programování v PHP[2] a nebo ve sborníku praktických tipů[3].

2.1.2. Nette

Nette je PHP frameworkem, postaveným nad PHP, který je vyvíjen Českou komunitou. To s sebou nese velkou řadu výhod v podobě české dokumentace, návodů a podpory v rámci fóra. Nette je podle oficiálních stránek[4] již vyspělým, moderním a bezpečným frameworkem, který: „Při programování vychází vstříc a nepřidělává vrásky. Eliminuje bezpečnostní rizika, podporuje *AJAX (Asynchronous JavaScript and XML)* a znovupoužitelnost.“ Nette je postaveno na dnes již standardním návrhovém vzoru *MVC (Model-View-Controller)*.2.2

Velký důraz klade Nette na zabezpečení a ochranu před nejrůznějšími druhy útoků. Jako příklad uvádím ochranu před takzvaným *XSS (Cross-Site Scripting)* útokem[4].

Další nespornou výhodou je šablonovací jazyk Latte[5]. Latte napomáhá výslednému formátování webové stránky přes takzvaná makra, která výraz interpretují a rozbalí do výsledného HTML kódu. To výrazně usnadňuje práci kodéra a navíc zabraňuje chybám při interpretaci výstupu a zjednodušuje se tím výstup různých závorek, uvozovek a speciálních znaků.

V neposlední řadě jsou významným pomocníkem přiložené ladící nástroje, které fungují automaticky s balíčkem Nette, pokud je nastaveno prostředí v ladícím režimu[6].

Ilustrace 1: Nette ladící nástroje[4]

2.1.3. MySQL

MySQL je relační databázový systém typu *DBMS (Database management system)*, což je vrstva oddělující aplikační logiku a data. Aby tímto termínem mohl být systém označován, musí být schopen efektivně pracovat s velkým množstvím dat, definovat jejich perzistentní strukturu, ale také s nimi manipulovat. Data musí být možné přidávat, editovat a mazat.

Databázový systém je vlastněný společností Oracle, která ho síří jako open source. Díky tomu je v současné době využitelný téměř ve všech nejrozšířenějších programovacích jazycích.

Každá databáze v MySQL je tvořena z jedné nebo více tabulek, které mají řádky a sloupce. Mezi těmito tabulkami jsou pak definovány relace, které zajišťují správnou perzistenci a redukují duplicitní záznamy.

Jak již název napovídá, jako dotazovací jazyk slouží *SQL (Structured Query Language)*, což je strukturovaný dotazovací jazyk, který se má syntaxí co nejvíce podobat běžnému anglicky položenému dotazu. Jazyk využívá pro ovládání databázové vrstvy příkazy jako `SELECT`, `DELETE`, `UPDATE`. Další možnosti lze dohledat v oficiálním manuálu[7] nebo v již zmíněné knize[2].

2.1.4. Webové služby

Webová služba je softwarový systém, zkonstruovaný k podpoře interakce mezi stroji přes síť. Má rozhraní popsané ve strojově zpracovatelném formátu *WSDL (Web Services Description Language)*. Ostatní systémy interagují s webovou službou způsobem, předepsaným jejím popisem za pomoci *SOAP (Simple Object Access Protocol)* zpráv[8].

Zprávy jsou typicky dopravované použitím nezávislých standardů *HTTP (HyperText Markup Language)* s *XML (Extensible Markup Language)* serializací ve spolupráci s ostatními webovými standardy. Aplikace si tedy mezi sebou posílají zprávy ve formátu XML, které přenášejí dotazy a odpovědi jednotlivých aplikací. Jak může vypadat taková odpověď serveru na požadavek klienta o zobrazení podrobností produktu je demonstrováno na ilustraci 2. Webovým službám z pohledu bezpečnosti se dále věnuje kapitola 2.3.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productName>Čokoláda sada 3 chutí</productName>
        <productID>827635</productID>
        <popis>Čokoláda hořka, bílá a smetanová</popis>
        <cena>98,50</cena>
        <naSkladu>ano</naSkladu>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```

Ilustrace 2: Odpověď serveru ve formátu SOAP

2.1.5. JSON

Částečnou alternativou k webovým službám může být přenos dat formátem *JSON (JavaScript Object Notation)*, který v oblasti webu stále nabírá na popularitě díky své jednoduchosti. Tato jednoduchost s sebou nese mnoho výhod i jistá omezení. JSON dokáže přenést jak indexovaná, tak neindexovaná pole, a to ve formátu srozumitelném i člověku. Tím se stává tento formát snadno parsovatelným pro libovolný jazyk. Nevýhodou jednoduchosti je potom absence dodatečných informací, komentářů a metadat, která mohou sloužit například pro určení znakové sady. JSON hojně využívá technologie *AJAX (Asynchronous JavaScript and XML)* pro specifikaci svých dat.

2.1.6. HTML5

HTML5 je nástupcem starší generace *HTML (HyperText Markup Language)*, od které se postupně vzdaluje. Novinky z HTML5 jsou pozvolna implementovány do všech používaných prohlížečů. Spolu s HTML5 přichází celá řada novinek, jako je zjednodušený zápis formátu dokumentu, nové elementy pro názornější strukturu dokumentu, nové formulářové prvky, nativní podpora multimediálního obsahu, a tak dále. Nové formulářové prvky jsou implementovány přímo v prohlížečích a poskytují tak i validaci vstupu. Na obrázku je možné vidět vstupní prvek typu `date` v prohlížeči Chrome.

Ilustrace 3: Formulářový prvek `date` v prohlížeči Chrome[9]

HTML5 přináší také spoustu zajímavých funkcí, které mohou sloužit například ke geolokaci uživatele, jako lokální databáze nebo dokonce pro používání aplikací offline. O tomto a mnoha dalších věcech pojednává česky psaný web k HTML5[10].

2.1.7. CSS3

Nová verze *CSS (Cascading Style Sheets)* s pořadovým číslem tři řeší velkou řadu problémů a nedokonalostí, které se doposud musely řešit vkládáním obrázku na pozadí. Příkladem jsou nejrůznější druhy přechodových výplní, kulatých rohů, průhledností a stínů u řady elementů. Přibyla zde i podpora pro řešení animace, kterou bylo nutné donedávna řešit přes přídavný javascript.

Mezi další novinky patří také velká spousta nových selektorů, podpora webových písem, media queries 2.4, průhledné barvy, stínování textu a tak dále. Obecně můžeme říci, že trendem je řešení implementovatelná pouze pomocí JavaScriptu přesouvat v nějaké formě do CSS3. To by mělo ve výsledku pomoci výkonu aplikace, a to i díky tomu, že některé animace v CSS jsou akcelerované hardwarově přímo v prohlížeči. O těchto a dalších novinkách pojednává například česká dokumentace k CSS3[11].

2.1.8. Bootstrap

Bootstrapem se rozumí knihovna pro rychlou tvorbu uživatelských rozhraní, která je kombinací fragmentů HTML a přiložených CSS stylů v součinnosti s javascriptem. Na ilustraci 4 je nastíněna deklarace prvků uživatelského rozhraní v HTML, ke kterým kodér dopisuje příslušné třídy, obsažené ve frameworku.


```
<!-- Single button -->
<div class="btn-group">
  <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown">
    Action <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" role="menu">
    <li><a href="#">Action</a></li>
    <li><a href="#">Another action</a></li>
    <li><a href="#">Something else here</a></li>
    <li class="divider"></li>
    <li><a href="#">Separated link</a></li>
  </ul>
</div>
```

Ilustrace 4: Tlačítka ve Frameworku Twitter Bootstrap [12]

Právě ona předdefinovaná tlačítka, formulářové prvky a typografické bloky jsou nejsilnější stránkou frameworku. S jejich pomocí je možné v krátkém čase vytvořit dobře vypadající formulářové prvky, přehledná menu a bohatá uživatelská rozhraní. Bootstrap podporuje nejnovější specifikace v HTML5 a CSS3, popsané v kapitole 2.1.6 a 2.1.7. Různé příklady použití a podrobné specifikace jsou k nalezení na oficiálních stránkách projektu[12].

2.1.9. Less

Less je dnes poměrně novým trendem, který stále nabírá na popularitě v oblasti vývoje uživatelských rozhraní pro webové aplikace. Jedná se o takzvaný CSS preprocesor, což znamená, že se jedná o interpretovaný proprietární jazyk, který vznikl jako nadstavba na CSS. Less rozšiřuje CSS o mnoho zajímavých možností. Například je možné deklarovat proměnné, psát funkce, vnořovat bloky a mnohé další. To zajišťuje modularitu stavebních bloků a zamezuje zbytečnému opakování kódu.

Funguje to tedy tak, že uživatel napíše celý styl CSS v jazyce less a spustí nad ním interpretr jazyka, který zajistí překlad na kompatibilní verzi CSS. Tento přístup řeší velké množství problémů s definováním barev nebo s kompatibilitou prohlížečů. Použití less v mé aplikaci mimo jiné řeší i škálování vzhledu pro různá rozlišení. Pro názornost přidávám dva příklady použití, více v dokumentaci[13].

```
@base: #f04615;
@width: 0.5;

.class {
  width: percentage(@width); // returns `50%`
  color: saturate(@base, 5%);
  background-color: spin(lighten(@base, 25%), 8);
}
```

Ilustrace 5: Deklarace proměnných v Less[13]

Na obrázku 5 je patrná deklarace proměnné `@base` a její následná aplikace do stylu přes funkci na saturaci barvy.

Ilustrace číslo 6 demonstruje řešení problémů s kompatibilitou CSS3 v prohlížečích. Je zde vidět takzvaná mixin funkce, která má jako vstupní parametr styl stínu. Funkci `box_shadow` je možno přidělit libovolnému selektoru a při interpretaci dojde k rozbalení obsahu funkce.

```
.box-shadow (@shadow1) {
    -webkit-box-shadow: @shadow1;
    -moz-box-shadow: @shadow1;
    -ms-box-shadow: @shadow1;
    -o-box-shadow: @shadow1;
    box-shadow: @shadow1;
}
```

Ilustrace 6: Kompatibilita CSS3 v Less[13]

Dnes již existuje množství alternativ, ale less se stále těší velké oblibě. Za to z velké části vděčí populární knihovně zvané Twitter Bootstrap 2.1.8, která je v něm napsána.

2.1.10. JavaScript

JavaScript je objektově orientovaný multiplatformní skriptovací jazyk. Nejčastěji je používán ke zlepšení uživatelského komfortu na statických webových stránkách. Pomocí JavaScriptu lze tedy libovolně měnit *DOM (Document Object Model)* strukturu HTML dokumentu a zachytávat, případně i spouštět, různé události vyvolané webovou stránkou, potažmo jejím uživatelem. Může však být použit i k tvorbě animací, ovládání různých prvků uživatelského rozhraní či vytváření jiných grafických efektů. Pro JavaScript existuje velké množství nadstavbových knihoven, jako například zepto.js, snack.js nebo jQuery.

2.1.11. jQuery

JavaScriptová syntaxe může být v mnoha případech velice neohrabaná, a proto je dnes mnohem rozšířenější používání nadstavbové knihovny zvané jQuery[14]. Ta v mnohých případech kodérovi práci značně usnadňuje a přináší mnohem přehlednější *API (Application Programming Interface)*, které má již standardní problémy webového vývoje podchyceny. Knihovna je tedy velice intuitivní a její kód je dobře čitelný, díky čemuž se rychle stala široce využívanou. Tomuto faktu dává za pravdu i následující příklad.

Změna barvy pozadí v jQuery

```
$('body').css('background', '#ccc');
```

Stejný problém v JavaScriptu

```
function changeBackground (color){
    document.body.style.background = color;
}
document.getElementsByTagName("body")[0]
.onload="changeBackground('red');"
```

2.1.12. AJAX

AJAXem souhrnně označujeme několik technologií, které slouží víceméně k asynchronní komunikaci se serverem.

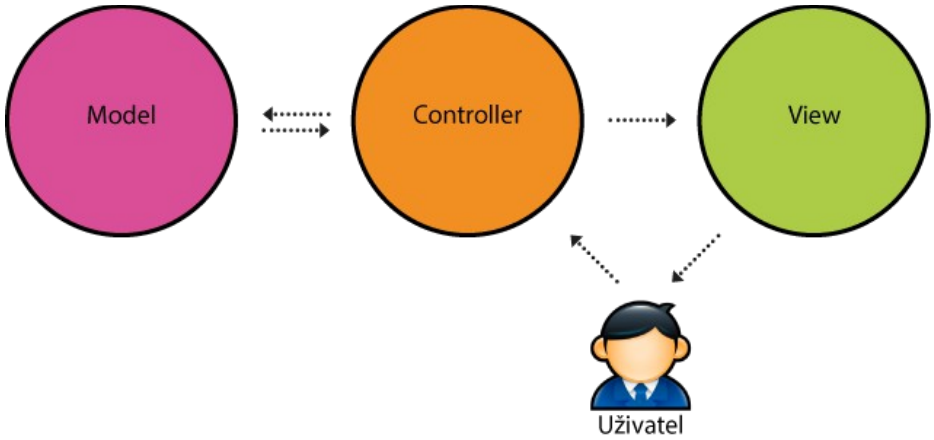
- HTML a CSS pro prezentaci informací
- DOM a JavaScript pro zobrazování a dynamické změny prezentovaných informací
- XMLHttpRequest pro asynchronní výměnu dat s webovým serverem

AJAXem můžeme měnit obsah jednotlivých stránek bez nutnosti jejich kompletního znovu načítání. Uživateli tedy umožňuje plynulé ovládání aplikace bez nutnosti znovu načítat stránku při hlasování v anketě či přidání komentáře, což zamezuje například zbytečnému přesunutí náhledu opět na začátek stránky po znovu načtení. Podpora AJAXových požadavků je implementována například i do jQuery[15].

2.2. Návrhový vzor MVC

MVC je velmi oblíbený architektonický vzor, který se těší veliké oblibě a v rozsáhlejších projektech je pro strukturu aplikace a čitelnost kódu nepostradatelný. Základní myšlenkou MVC architektury je rozdělení aplikace do menších logických celků, což je řešením takzvaného špagetového kódu, který spojuje výpočty, vykreslování i manipulaci s daty do sebe.

Jak již bylo zmíněno, architektura dělí aplikaci na tři logické celky podle ilustrace 7.



Ilustrace 7: Návrhový vzor MVC

2.2.1. Model

Model slouží v systému jako reprezentace dat aplikace, zajišťuje jejich konzistenci, ručí za správné ukládání, editaci, vyhledávání a mazaní. Zjednodušeně model vezme vstupní data od *Controlleru*, provede operaci nad modelem a výstup předá zpět *Controlleru*. Rozhodně však model neřeší, jak budou data formátována na vstupu a stejně tak neřeší, jak se výsledná data vykreslí v aplikaci.

2.2.2. View

View je v podstatě pravým opakem modelu. Neřeší, jak a kde se data vzala, ale naopak dává pozor na to, jakým způsobem získaná data zobrazit. Hlavním úkolem *View* je tedy vytvoření uživatelského rozhraní v závislosti na datech a požadovaných akcích uživatele.

2.2.3. Controller

Posledním chybějícím článkem architektury je *Controller*, který funguje jako takový prostředník mezi *Modelem* a *View*. Uživatel vybere požadovanou akci, a ta je odeslána jako požadavek na *Controller*. Ten se podle své vnitřní logiky rozhodne, jak data formátovat a odeslat do *Modelu*. Po zpracování data opět přebere, zformátuje a předá opět na *View*.

2.3. Zabezpečení webových služeb

2.3.1. XML podepisovaní

Prvním krokem k zabezpečení webových služeb je forma podepisování zpráv. Podepisování funguje srovnatelně jako například pro elektronickou poštu. Z výsledné zprávy je vypočítán *hash* a ten je podepsán soukromým klíčem odesílatele a přidán ke zprávě. Pokud je na druhé straně procesem opačným získán opět stejný *hash,* identický s *hashem* obdržené zprávy, je zpráva považována za důvěryhodnou a autentickou. Jak je jistě zřejmé, tento postup nezabraňuje přečtení zprávy při přenosu a nelze tedy považovat tento typ komunikace za bezpečný. Lze však předpokládat, že takto odeslané zprávy poskytují pravdivé informace.

2.3.2. XML šifrování

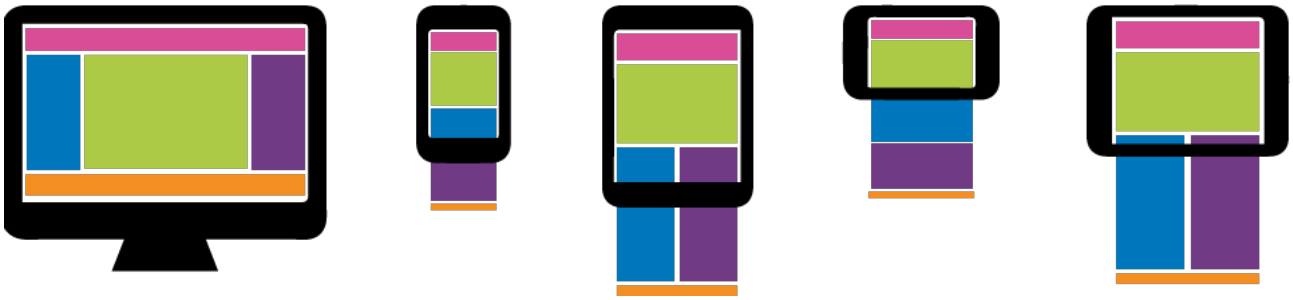
Jako druhá možnost se nabízejí všemožné druhy šifrování, které lze realizovat přes různé varianty obalových technik pro samotný *SOAP* obsah nebo rozšířením základních značek *XML*. Před šifrováním je však třeba zmínit, že řešení ve formě šifrování každé zprávy je sice nejbezpečnější, ale zároveň nejvíce náročné na výpočetní výkon, což může byt nežádoucí při velké zátěži serveru. Jelikož je však struktura *XML* dokumentu stromová, je možné zašifrovat libovolný uzel, a tím zajistit bezpečnost například pouze pro osobní údaje. Tím lze redukovat výpočetní výkon, potřebný pro šifrování, a zároveň zabezpečit kritická data[8].

2.3.3. Zabezpečený kanál

Lepší variantou je zabezpečení na úrovni přenosového media. Samostatné zprávy tedy nejsou nijak zvlášť zabezpečeny a zabezpečení je řešeno komunikací obou serverů přes šifrovaný kanál. V tomto případě se obvykle jedná o komunikaci přes *HTTPS, což* lze považovat za bezpečné řešení například i proti sledování komunikace a odposlechu zvenčí.

2.4. Responsive design

Responsive design[16] se dá nejlépe popsat ilustrací 8. Jedná se v podstatě o návrh uživatelského rozhraní, který zaručí, že zobrazení stránky bude optimalizováno pro všechny druhy zobrazovacích zařízení ve všech standardních rozlišeních. Toho je dosaženo především díky vlastnostem Media queries, která jsou zahrnuta ve specifikaci CSS3. Pomocí Media queries 2.5 lze rozpoznat vlastnosti zařízení, na kterém je stránka prohlížena, a přizpůsobit tak samotnou stránku a její obsah velikosti zobrazovacího zařízení. Více v knize Responsive design.



Ilustrace 8: Responsive design

2.5. Media query

Media query je, jednoduše řečeno, podmíněná deklarace *CSS*, která se aplikuje pouze při splnění podmínek. Taková podmínka bývá obvykle typ media, tedy zda-li se jedná o obrazovku nebo verzi pro tisk, spolu se specifikací maximální nebo naopak minimální šířky zobrazovacího zařízení. Na ilustraci číslo 9 je vidět aplikace *CSS* pouze při splnění podmínky, která v tomto případě specifikuje, že zobrazovací zařízení má maximální šířku 480px, tedy pravděpodobně mobilní telefon. Media quries přicházejí až s verzí *CSS3* a jsou jedním ze základních pilířů responsive designu.

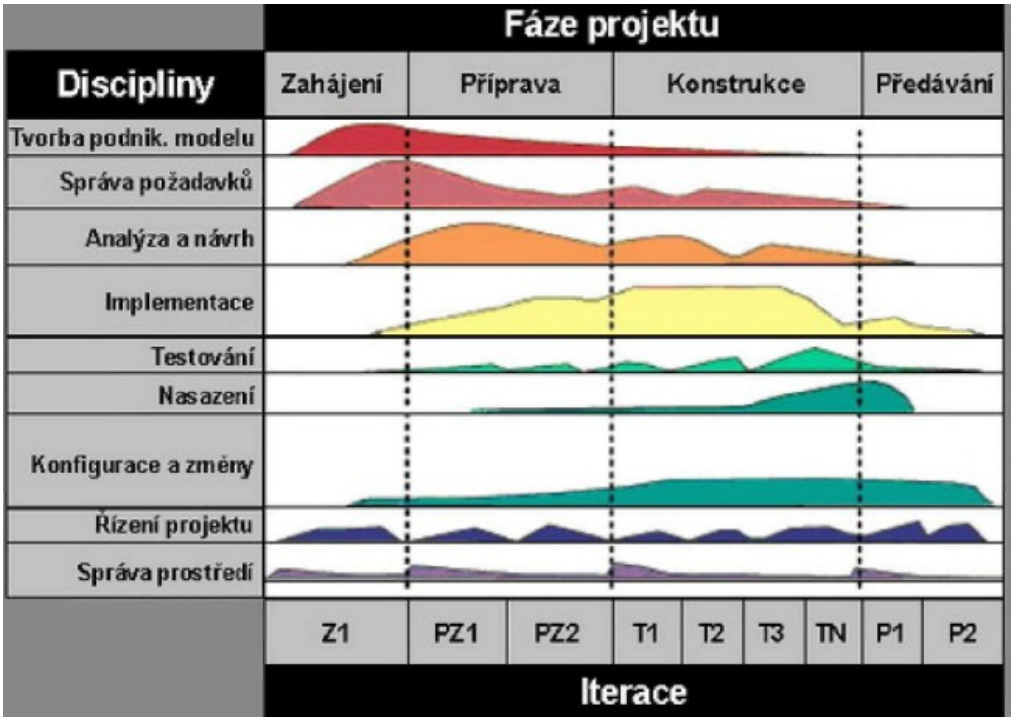
```
@media (max-width: 480px) {
  .container {
    width: auto
  }
}
```

Ilustrace 9: Media query[11]

2.6. Agilní metody vývoje

2.6.1. RUP

RUP (Rational Unified Process)[17] je metodika vývoje softwaru, vytvořená společností Rational Software Corporation. RUP je na pomezí agilního vývoje a hodí se spíše pro střední až větší projekty. Jedná se v podstatě o kolekci osvědčených metod a postupů, používaných při projektovém řízení.



Ilustrace 10: Fáze projektu dle RUP[17]

RUP obsahuje celkem čtyři základní fáze. Každá fáze obsahuje několik dalších iterací. Před započetím nové iterace musí být splněna dříve definovaná kritéria předchozí iterace. Fáze zahájení (inception) definuje účel, rozsah projektu a jeho obchodní kontext. Ve fázi projektování (elaboration) je potřeba analyzovat požadavky zákazníka i celého projektu a definovat základy architektury. Realizační fáze (construction) je nejdelší a probíhá zde tvorba zdrojových kódů. V poslední fázi předání (transition) může být projekt předán zákazníkovi nebo do dalšího cyklu.

2.6.2. Extrémní programování

Extrémní programování (dále již pouze *XP*)[18] je již oproti *RUP* agilní metodikou v pravém slova smyslu. Klade si za cíle rychle se přizpůsobovat měnícím se uživatelským požadavkům a přinášet tak co největší přidanou hodnotu s co nejnižšími náklady. Základní zásady *XP* lze specifikovat takto:

1. Komunikace

   V rámci projektu je kladen velký důraz na komunikaci mezi všemi zainteresovanými stranami. Pokud totiž například selže komunikace mezi zákazníkem a vývojáři, dostane zákazník software, který sice může být dobře naprogramovaný, ale nesplňuje přání zákazníka.

2. Jednoduchost

   V XP se programuje vždy pouze to, co je třeba ke splnění aktuálních požadavků. Nepřidáváme tedy žádný kód, který by se mohl v budoucnu možná hodit, protože je třeba mít na paměti, že požadavky se mohou velice rychle měnit a výsledný kód bychom museli měnit. V tomto bodě se nejvíce uplatňuje princip *KISS (Keep It Simple, Stupid!)*.

3. Zpětná vazba

   Zpětná vazba je velice důležitá hned na několika místech. Například programátoři neustále tvoří jednotkové testy, takže během velice krátkého intervalu dokáží zjistit, zda se změnou kódu nerozbilo něco dalšího. Dalším příkladem je zpětná vazba pro zákazníka, který prováděním akceptačních testů může průběžně sledovat pokroky ve vývoji.

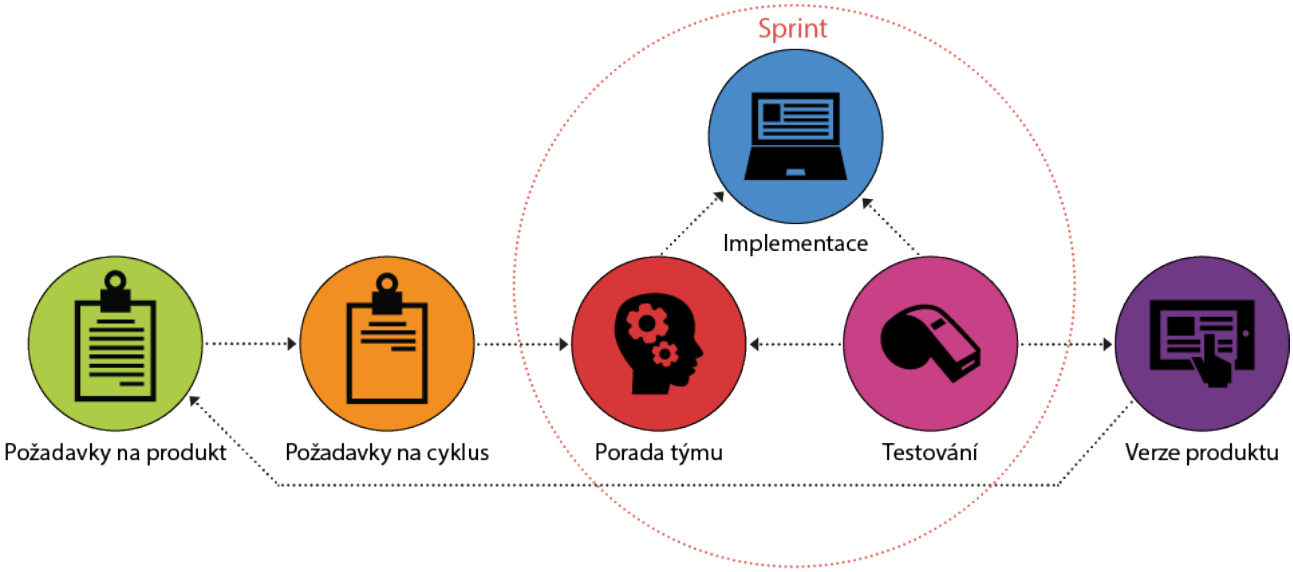
4. Odvaha

   Posledním bodem je pak odvaha, která je pro programátora velice důležitá. Pokud během vývoje narazí na komplikovaný problém, který způsobí řadu potíží na mnoha místech, je třeba velká odvaha se do takovéto úpravy pustit. Dále se dá odvahou nazvat zahození celodenní práce za předpokladu, že jiný přístup přinese požadovaný úspěch.

2.6.3. SCRUM

SCRUM je asi nejznámější metoda agilního programování. Na začátku celého procesu jsou stanoveny základní požadavky, k těm jsou přiděleny odhadované časové náročnosti. V další fázi vývoje jsou vybrány pouze ty požadavky, které budou zařazeny do takzvaného sprintu, viz obrázek 11. Sprint je časová fáze, ve které dochází k rychlému vývoji a implementaci požadavků, obvykle trvá týden až měsíc. Během tohoto cyklu probíhá ještě denní SCRUM porada, která nemusí být nijak dlouhá a každý programátor zde vysvětlí, na čem pracoval, na čem bude pracovat dále nebo kde vidí problémy.

Takových sprintů proběhne do stanoveného termínu několik. Po tomto termínu vznikne verze produktu, připravená k nasazení. Tato verze se vrací opět na začátek, kde je hodnocena zadavatelem a porovnána s požadavky na produkt. Požadavky jsou náležitě upraveny a produkt vchází do druhého sprintu. Takto se celý proces opakuje, dokud není produkt hotov. Dodatečné informace lze nalézt v seriálu o agilních metodikách[19].



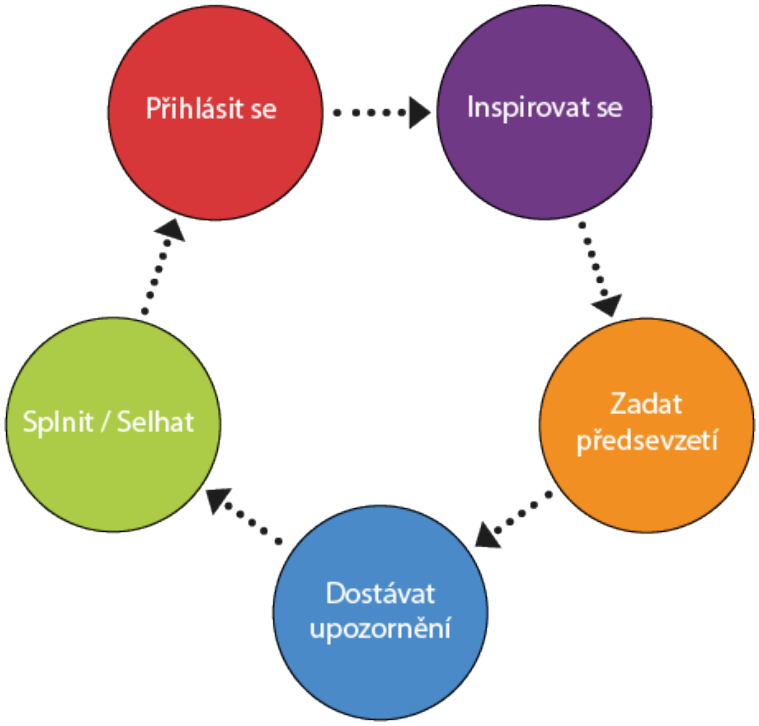
Ilustrace 11: Metodika SCRUM

3. Analýza

3.1. Specifikace požadavků

Základním požadavkem je vytvoření webové aplikace, která bude sloužit ke sledovaní dlouhodobých cílů, pro potřeby aplikace pojmenovaných jako předsevzetí. Na počátku vývoje aplikace však stála pouze mlhavá myšlenka, charakterizující princip aplikace. Proto bylo nutné postupným jednáním specifikace neustále zpřesňovat.

Každý uživatel by tedy měl mít možnost se do aplikace přihlásit, v aplikaci by mělo být možné se inspirovat předsevzetími ostatních a vzájemně se tak motivovat. Dále by mělo být možné zadat vlastní předsevzetí a nastavit si, do kdy chci předsevzetí splnit. Po tomto kroku by měl uživatel průběžně dostávat zprávy mailem podle toho, jaká předsevzetí v systému má aktivní a do kdy. Po nějaké době se uživatel vrátí do aplikace a měl by být schopen předsevzetí označit za splněné nebo nesplněné.



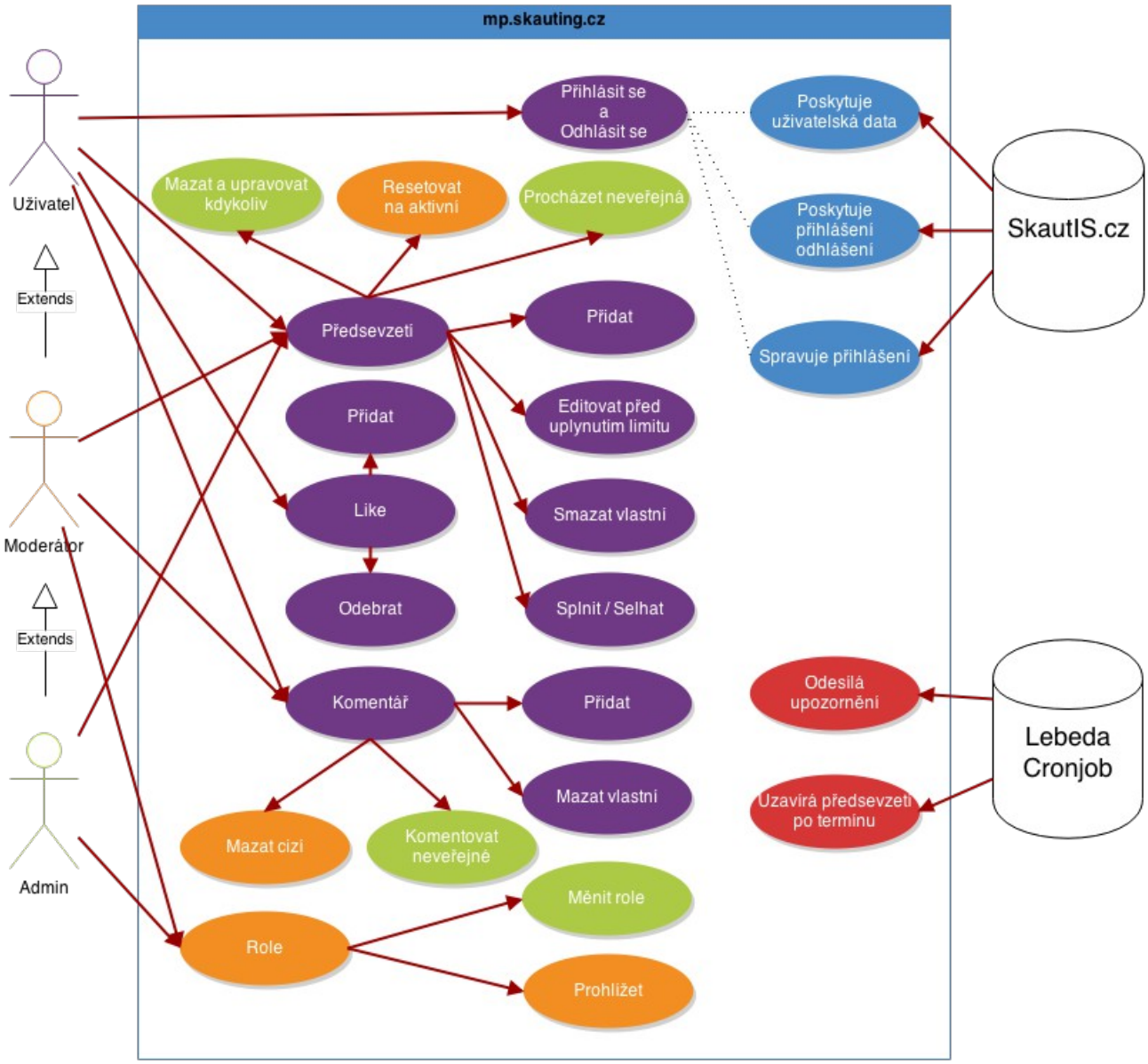
Ilustrace 12: Základní myšlenka aplikace

Postupným jednáním byly personálním odborem dílčí požadavky zpřesněny do následující podoby:

1. Aplikace bude napojena na centrální informační systém, odkud se bude možné přihlásit.
2. Je třeba přizpůsobit chování a vzhled aplikace pro mobilní zařízení.
3. Žádoucí je vyřešit zabezpečení citlivých interních dat.
4. Důraz klademe na implementaci vhodného uživatelského rozhraní, přívětivého k uživatelům ve všech standardních rozlišeních.

3.2. Požadovaná funkcionalita

Základní specifikaci požadavků bylo nutné rozpracovat do jednoduchých uživatelských scénářů. Z ilustrace číslo 13 je patrné, že v systému figurují tři základní uživatelské role.



Ilustrace 13: Usecase diagram

První rolí je prostý uživatel, jehož možnosti jsou znázorněny fialovou barvou. Druhou rolí je takzvaný moderátor, vyznačen žlutě. Ten má možnost navíc mazat nevhodné komentáře a resetovat uživatelům z jakéhokoliv důvodu uzavřená předsevzetí nebo také prohlížet si uživatelská oprávnění. Posledním uživatel je administrátor, vyznačen zeleně. Administrátor může libovolně nakládat s předsevzetími, a to i s těmi soukromými. Může tedy zároveň dohlížet na potenciálně závadný obsah, což může být z pohledu soukromí nežádoucí. Administrátor by tedy měl být důvěryhodná osoba a neměl by se vyskytovat v aplikaci příliš často.

V systému jsou navíc dvě speciální role: Zakázaný uživatel a Super Administrátor. Zakázaný uživatel nemá možnost se v sytému ani přihlásit, protože byl administrátorem zablokován. Naopak Super Administrátor je nejvyšší možná role, která nemůže být změněna, protože je v databázi nastavena napevno již od začátku.

Další kategorií jsou pak dva externí prvky, které stojí tak trochu mimo aplikaci. Jedním z nich je centrální server skautIS, který poskytuje správu přihlašování a uživatelská data. Druhým je serverový cronjob. Ten se stará o rozesílání upozornění a uzavírá předsevzetí, která propásla termín splnění.

3.3. Metodika vývoje

Specifikace požadavků od zadavatele byla velice složitá. Zpočátku nebyly definovány konečné výstupy a vývoj se stále točil v kruzích. Kvůli veliké proměnlivosti jsem byl nakonec nucen přistoupit k agilní metodice vývoje software. Nejvhodněji, vzhledem k mé situaci, se jevila metodika *SCRUM,* popsaná podrobněji v kapitole číslo *2.6.3.* Po nastudování potřebných podkladů a principů byla tato metoda nasazena při vývoji výsledné aplikace.

Zpočátku byl znám pouze základní koncept, podle kterého by měla aplikace fungovat. Z tohoto základu vznikly vždy specifické požadavky na jeden cyklus. Cyklus trval obvykle jeden týden, během kterého byly implementovány dílčí požadavky. Na konci každého týdne proběhla porada, během které jsme společně se zadavatelem prošli výstupy předchozího týdne a stanovili nové požadavky na týden nadcházející. Požadavky povětšinou pocházely přímo od zadavatele nebo od koncových uživatelů, kteří testovali použitelnost uživatelského rozhraní.

3.4. Wireframe aplikace

Na základě usecase diagramu vznikl předběžný návrh vzhledu aplikace zvaný wireframe. Ten je možné vidět na ilustraci číslo 14. Takovýto návrh je obvykle velice důležitý a dává zadavateli jasnější představu o základním vzhledu a funkcích aplikace. I přes to, že nebyl ve výsledku vůbec použit, sehrál svoji roli při dalších jednáních. Na jeho základech později vzniklo mnoho dalších, již mnohem konkrétnějších specifikací požadavků.

Ilustrace 14: Wireframe aplikace

3.5. Možnosti implementace responsive designu

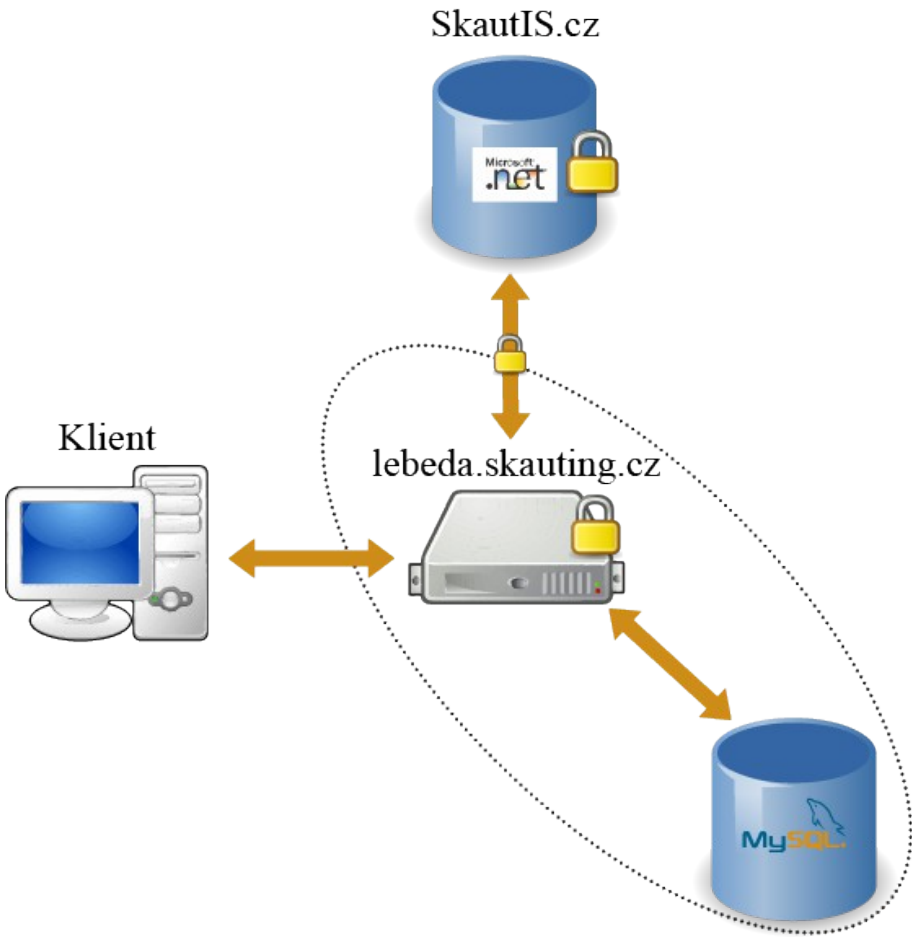
Aplikaci lze začít stavět na pomyslné zelené louce dle zásad responsive designu. Já jsem se však rozhodl využít některého z *CSS* frameworků. Do užšího výběru postoupili tři zástupci, a to Bootstrap, Foundation3 a SemanticUI. Hlavně díky jednoduchosti, komplexnosti a velkému množství příkladů jsem nakonec zvolil Bootstrap.

Většina zmíněných frameworků využívá k dosažení kýženého výsledku takzvaný *grid layout*[12], který rozděluje užitnou plochu ne na pixely, ale na sloupce. Tyto sloupce mají procentuální šířku a lze jim nastavit, jak se zalamují pod sebe při určitém rozlišení, čímž je dosaženo požadovaného rozložení stránky, a tím i dobré čitelnosti obsahu.

V mé aplikace jsem se rozhodl využívat verzi frameworku napsanou přímo v *LESS*. Díky tomu lze vhodně kombinovat deklaraci stylů v aplikaci s proměnnými deklarovanými ve frameworku. Je tedy možné například využívat stanovených šířek zobrazovacích zařízení pro vlastní Media queries. Tím je dosaženo uspokojivého rozložení stránky pro většinu rozlišení.

3.6. Zabezpečené přihlašování

Další klíčovou otázkou bylo zabezpečení citlivých dat aplikace a s tím spojené zabezpečené přihlašování. Bylo tedy nutno docílit modelu, který je k vidění na ilustraci 15. Nepřihlášený uživatel může procházet pouze veřejně přístupné sekce aplikace. Pokud si přeje se do aplikace registrovat, odešle požadavek na přihlášení, který ho přesměruje na centrální informační systém skautIS. Tam je možné se přihlásit nebo registrovat. Je-li následné přihlášení úspěšné, jsou přihlašovací data zabezpečeným kanálem poslána zpět do aplikace, kde se ověří. V případě, že je již uživatel přítomen v lokální databázi, jsou jeho data pouze aktualizována. Není-li přítomen, tak je vytvořen. Následně je uživatel automaticky přihlášen a může vstupovat i do dalších sekcí aplikace.



Ilustrace 15: Přihlášení přes skautIS

3.7. Uživatelské avatary

Pro komunitní portál by bylo záhodno vyřešit také avatary jednotlivých uživatelů. Obrázky je možné stahovat přímo z centrálního serveru, ale tam se používají pouze pasové fotografie ve vysokém rozlišení. Z toho důvodu nejsou moc oblíbené a ve výsledku tam příliš mnoho uživatelů fotografii nemá vůbec. Tento problém se nyní řeší centrálně, a proto jsem se rozhodl vypomoci si prozatím službou gravatar[20]. Ta po registraci páruje váš email s nahranou fotolitografií. Pak již stačí službu požádat o fotografii k vybranému emailu. Pokud je uživatel registrován, je zpět poslána jeho fotografie, pokud není, služba poskytne anonymní obrázek.

3.8. Použité technologie

Použité technologie vychází z velké části z možností hostingu poskytovaného organizací, ten používá oblíbené *PHP* ve spolupráci s *MySQL*. Zmíněný hosting přináší hned několik výhod. Jednou z nich je fakt, že jeho správa je plně v rukách organizace, a tím se z části řeší problém citlivých interních dat. Druhou nespornou výhodou je zabezpečený kanál mezi centrálním informačním serverem a hostingem, jak je zmíněno v kapitole číslo 2.3.3. Není tedy nutné nijak šifrovat samotné zprávy webových služeb. Použití tohoto hostingu limituje vývoj pouze na jazyk *PHP*, to však není žádnou závažnou překážkou a naopak to nahrává použití některého z oblíbených *PHP* frameworků. Pro tento projekt jsem se rozhodl použít Nette framework[4], a to především kvůli jeho kvalitám, aktivnímu českému fóru a možnosti osobní konzultace v mém okolí. Pro tvorbu vzhledu byl tedy použit již zmíněný Bootstrap[12] a k oživení uživatelského rozhraní jsem pak využil knihovnu jQuery a AJAX. Pro přehlednost uvádím tabulku číslo 1 se seznamem použitých technologií v příslušných verzích.

Serverová část	
PHP	5.4.28
MySQL	5.6
Nette	2.0.14
Klientská část	
HTML	5
CSS	3
jQuery	2.10
Bootstrap	3.0.4

Tabulka 1: Seznam použitých technologií

4. Návrh aplikace

4.1. Struktura aplikace

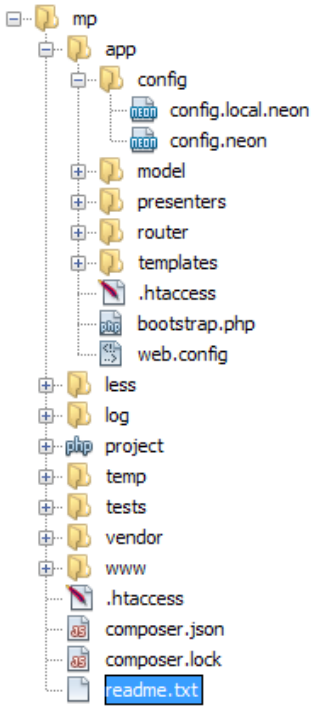
4.1.1. Sestavení aplikace

Založení a nastavení projektu je velice jednoduché, v nejnovější verzi Netbeans IDE je dokonce zabudována podpora pro Nette framework. Nově jsou podporovány i formáty specifické pro Nette, jako například konfigurační soubory Neon nebo šablonovací jazyk Latte. Zabudována je také nápověda, což přispívá značně výslednému komfortu používání Nette.

Za zmínku stojí i velice praktický nástroj na hlídání a správu závislostí, zvaný Composer[21]. Webový portál Composeru je přímo napojen na github a uživatel tedy pouze vyplní, které knihovny pro svůj projekt využívá. Composer se již postará o stažení aktuální verze a vložení do projektu. Je zde zabudována i podpora pro různé vývojové větve a je tedy možné aktualizovat například pouze určité verze knihoven. Takto je v mé aplikaci řešeno vkládání Nette frameworku a vyvíjené knihovny pro přihlašování přes webové služby[22].

4.1.2. Adresářová struktura

Aplikace je tedy postavena na Nette framework 2.1.2 a pro účely základního nastavení a zprovoznění je vhodné začít s dokumentací k základní vzorové aplikaci[23]. Pokud je navíc potřeba změnit nějaká nastavení, týkající se například přístupu k databázi, je dále vhodné projít si návod k základní konfiguraci[6]. Každá aplikace v Nette má obdobnou adresářovou strukturu jako nailustraci číslo 16.

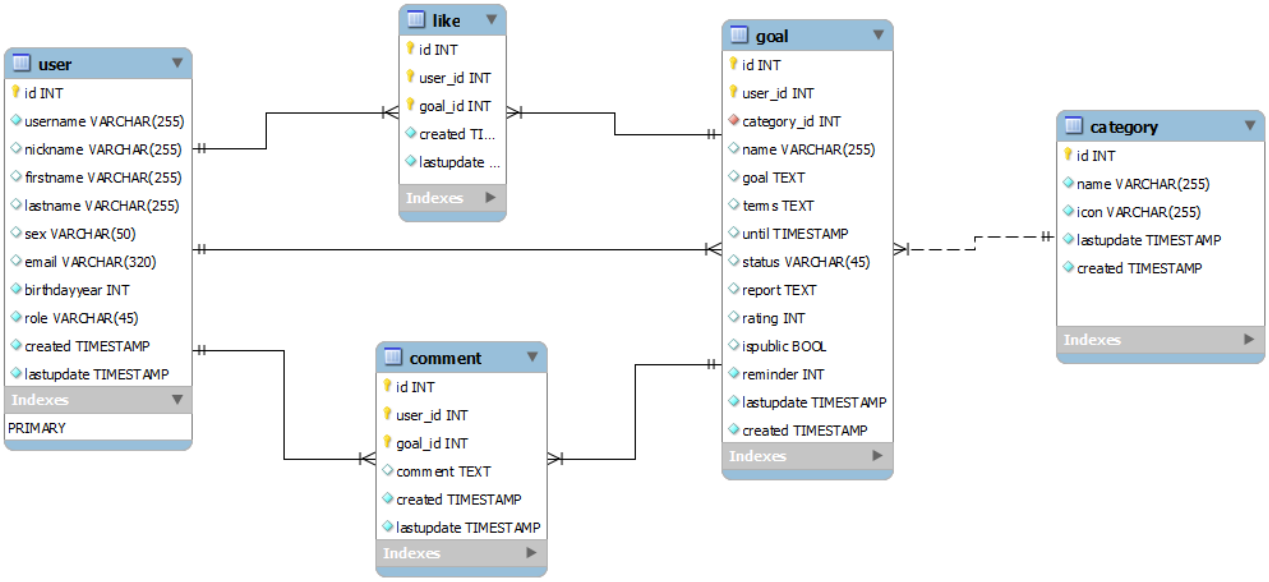


Ilustrace 16: Struktura aplikace

4.2. Databázová vrstva

Výsledný model databázové struktury je relativně jednoduchý. Obsahuje entity `user`, `goal`, `category`, `comment` a `like`. `User` se stará o ukládání dat z centrálního serveru a zároveň udržuje informace o uživatelských rolích. Entita `goal` sdružuje informace o předsevzetích a jejich výsledcích, `category` pak udává kategorii předsevzetí. Pak jsou zde dvě entity, zastřešující komunitní funkce: první entita je `comment`, sdružující komentáře k jednotlivým předsevzetím, a druhá je `like`, kterým lze ohodnotit snahu uživatelů.

Vše by mělo být patrné z ER diagramu (viz Ilustrace 17).



Ilustrace 17: Databázová struktura

4.3. Modely

V Nette aplikaci je *modelem* nazývána sada tříd, které slouží k obsluze funkcí aplikace a jejich mapování na data v databázi. Každá z výše uvedených entit je obsluhována jednou třídou, která se nazývá `Repository`. Všechny tyto `Repository` jsou pojmenovány `NázevRepository` a dědí od abstraktní třídy, která definuje základní funkce databázové logiky. Je nutné, aby jména korespondovala se jmény tabulek, vnitřní logika potom zaručí správné mapování třídy na tabulky.

Každý takovýto model musí být nastaven v konfiguračním souboru. Tím je zajištěno, že třída bude automaticky předána do systémového kontejneru a bude možné kdykoliv jí zavolat, stejně jako kteroukoliv jinou službu. Konfigurace je nastíněna na obrázku číslo 18. Jako první je uvedeno jméno proměnné, pod kterou bude možné službu volat, za ním následuje jméno třídy.

Pak se zde také nachází jeden speciální *model,* zvaný `Authenticator`. Tomu je naopak předáváno pouze `UserRepository` a má jen jedinou metodu `authenticate`. Tou je zajištěno ověření uživatelů proti lokální databázi a jejich případné přihlášení do systému.

Ilustrace 18: Konfigurace služeb

Komunikace s databází je postavena na databázovém *API,* zvaném NotORM[24]. To je integrováno přímo do každého `Repository,` což značně ulehčuje manipulaci s daty a umožňuje, aby modely byly dostatečně flexibilními při drobných změnách požadovaných parametrů.

Místo *SQL* příkazu:

```
SELECT * FROM `user` WHERE role='user' ORDER BY created LIMIT 2
```

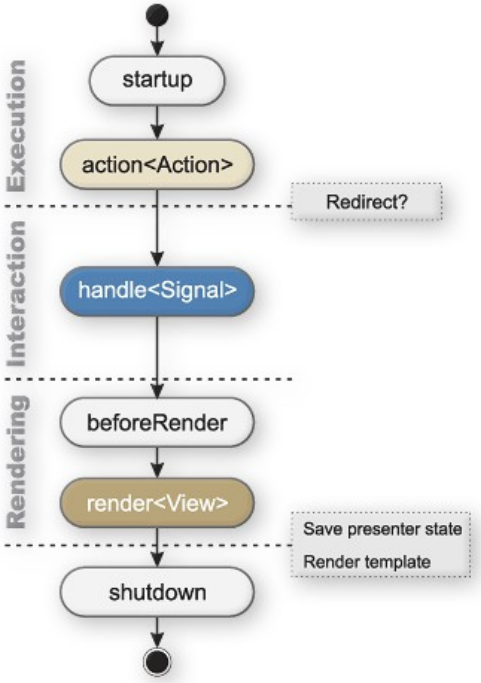
Můžeme použít přímo v kódu:

```
$userRepository->findAll()
    ->where('role'=>'user')
    ->order('created')
    ->limit (2);
```

4.4. Presentery

Presenter[25], neboli *Controller* (viz 2.2.3), je třída, která obstarává vykonání určité aplikační logiky. Je však třeba určit, který požadavek by měl obsloužit jaký *presenter*. To určuje takzvaný *Router*, což je speciální druh třídy, který podle zadaných výrazů přiděluje požadavky *presenterům*. Každý `presenter` je potomkem `Nette\Application\Presenter`. Pokud je tedy požadavek přidělen, započne jeho zpracování podle životního cyklu `presenteru,` uvedeného na obrázku 19.

Jako první se volá metoda `startup`, ve které se zajišťuje základní nastavení atributů třídy. Hned po ní se volá příslušná metoda `action<Action>`, kde parametrem je název šablony, která se vykonává. Například výchozí šablona má `action` metodu se jménem `actionDefault`. Zde se řeší přístupová práva a případné přesměrování při nedostatečném oprávnění. Metody `handle<Signal>` se provádí následně a používají se zejména pro zpracování akcí od uživatele. Jako další se provede metoda `beforeRender`, ve které se nastavují proměnné pro více šablon najednou. Posledním krokem před ukončením *presenteru* je provedení metody `render<View>`, kde je parametrem, obdobně jako u `action,` název šablony a předávají se zde proměnné, se kterými potom šablony pracují. Následuje již pouze ukončení cyklu.



Ilustrace 19: Životní cyklus presenteru[25]

Základem všech presenterů je třída s názvem `BasePresenter`, která je potomkem obecného *presenteru* `Nette\Application\Presenter`, a od kterého budou všechny ostatní dědit. Stejně jako v předchozím případě jsou *presentery* pojmenovávány `NázevPresenteru`. Zde slouží název jako implicitní jméno souboru šablony, kterou má *presenter* použít k vykreslení dat. Každý *presenter* bude zastupovat jednu sekci aplikace. Jednotlivé sekce jsou popsány níže.

- BasePresenter – základní presenter, nastavuje prvotní proměnné
- SecuredPresenter – řeší zabezpečení a vpouští pouze přihlášené uživatele
- HomepagePresenter – veřejná sekce aplikace, dědí od BasePresenteru
- MainPresenter – hlavní část aplikace po přihlášení
- ErrorPresenter – vykresluje případná chybová hlášení
- GoalPresenter – provádí operace s předsevzetími
- UserPresenter – provádí operace s uživateli
- SignPresenter – zajišťuje přihlašování a registraci uživatelů

4.5. Šablony

Jako *view* slouží v Nette šablonovací jazyk Latte. Latte je kombinací *HTML* a pomocných maker, která ulehčují vypisování proměnných. Základní šablonou je soubor jménem `@layout.latte`. Ten definuje rozložení vzhledu aplikace a určuje například hlavičky výsledné stránky, přikládané stylopisy a ostatní náležitosti, které se příliš nemění. Další šablony se nacházejí ve složce `templates` a jmenují se vždy podle příslušné metody *presenteru*, který je obsluhuje. Jednotlivé šablony se při vykreslování vkládají do této hlavní (v hierarchii nadřazené) šablony a mění se tak pouze nově definované části[5].

Tohoto chování je docíleno pomocí bloků.

```
//@layout.latte

<title>{block title} Hlavní stránka{/block}</title>

//prihlaseni.latte

<title>{block title} Přihlas se{/block}</title>
```

Pokud je tedy v použité šabloně deklarován ten samý blok jako v nadřazené, dojde k překrytí původního bloku novou deklarací, obdobně jako u dědičnosti. Výsledný kód šablony v mé aplikaci je tedy kombinací maker jazyka Latte, HTML (frameworku Bootstrap viz 2.1.8) a JavaScriptu (jQuery viz 2.1.11). Na ilustraci 20 je zobrazen kód, vykreslující varovné zprávy uživateli.

```
{block #flash}
{snippet flash}
    <div n:foreach="$flashes as $flash" class="alert alert-{$flash->type} alert-dismisable fadeout">
        <button type="button" class="close" data-dismiss="alert" aria-hidden="true">&times;</button>
        {$flash->message}
    </div>
        {/snippet}
{/block}
```

Ilustrace 20: Ukázka Latte

4.6. Integrace webových služeb

Zpočátku bylo nutné prostudovat dokumentaci webových služeb organizace[26], kde je možné svojí aplikaci i zaregistrovat. Při registraci jsou vyplněny podrobnosti o aplikaci a dvě důležité webové adresy. Obě vedou zpět do aplikace. Jedna po přihlášení, kam jsou poslána data, druhá pak po odhlášení, kde může být třeba přesměrování na úvodní stránku se zprávou o odhlášení. Tyto adresy je třeba ve své aplikaci přesměrovat na náležitý skript, který se postará o jejich zpracování.

Spolu s registrací obdrží každá aplikace svůj unikátní identifikátor, který využívá k navázaní komunikace s centrálním serverem. Server tedy může na základě identifikátoru monitorovat aktivitu aplikace a případně zamezit přístupu.

Při procházení dokumentace jsem také narazil na knihovnu, usnadňující komunikaci s webovými službami, napsanou pro *PHP*. Knihovna je však stále ve fázi vývoje a vzniká společnou silou dobrovolníků, pracujících na obdobných projektech. Po pročtení její dokumentace byla knihovna začleněna do mé aplikace, bylo zde však několik problémů a nesrovnalostí.

Jako příklad bych uvedl funkci knihovny `isLoggedIn`, která vracela `true` pouze, pokud byl uživatel přihlášen. V opačném případě vždy vyskočila nepojmenovaná výjimka, která způsobovala pád aplikace. Tato funkce byla opravena pouze na výstup logické hodnoty `true` nebo `false`.

Do knihovny jsem také zavedl aliasy pro běžně používané funkce. K hromadnému zadávání přihlašovacích dat je možné použít funkci `setLoginData` a pro hromadné odstranění `resetLoginData`. Toho je docíleno voláním `setLoginData` s prázdnými vstupními parametry, které jsou ve výchozí hodnotě nastaveny na `NULL`.

Tyto problémy jsem průběžně opravil a odeslal zpět do oficiální distribuce knihovny. Po odeslání mého kódu byly ostatními programátory navíc doplněny dodatečné testy. Nyní je tedy možné výše zmíněné funkce využívat v každé nové distribuci knihovny.

Pro správnou implementaci knihovny nesmíme zapomenout na vyplnění konfiguračních souborů, kde je nutné uvést identifikátor aplikace. Bez provedení tohoto kroku nebude možné knihovnu správně využívat.

4.7. Implementace vybraných funkcí

V následujících ukázkách je možné si všimnout, že jména proměnných jsou v anglickém jazyce a jména metod jsou naopak v českém jazyce. Výsledný kód by mohl být sjednocen do angličtiny, jména metod jsou však zároveň použita při mapování požadavků na výsledné skripty. Rozhodl jsem se tedy jména metod ponechat v českém jazyce. Toto chování je automatické a pro změnu by musela být vytvořena překládací tabulka. Proto tedy tato dvojitá pojmenování.

Stejný problém je řešen pro pojmenování *presenterů*, tam jsem se však rozhodl použít překladovou tabulku. Na ilustraci číslo 21 je možné vidět zmíněnou překladovou tabulku, implementovanou v *routeru* pomocí pole.

```
$router[] = new Route('<presenter>/<action>/<id>', array(
    'presenter' => array(
        Route::VALUE => 'Homepage',
        Route::FILTER_TABLE => array(
            // řetězec v URL => presenter
            'uzivatel' => 'User',
            'predsevzeti' => 'Goal',
            'aplikace' => 'Main',
            'zobrazit' => 'View',
            'verejna-cast' => 'Homepage',
        ),
    ),
    'action' => 'vitejte',
    'id' => NULL,
));
```

Ilustrace 21: Překládací tabulka routeru

4.7.1. Přidávání předsevzetí

Základní funkcí aplikace je stanovení si vlastního předsevzetí, které může přidat libovolný registrovaný uživatel. Jedná se v podstatě o odeslání formuláře s potřebnými daty, kvůli případné velikosti dat je však formulář odesílán metodou `POST`. To zamezí chybě při překročení velikosti dat. Jelikož je tato funkce naprosto klíčová, je přístupná přímo z hlavního menu.

Po kliknutí na přidat předsevzetí se však nenačítá nová stránka s předpřipraveným formulářem, ale dialogové okno, které původní obsah překrývá. Toho je docíleno grafickou komponentou, zvanou *modal*, kterou nalezneme v Bootstrapu. Jedná se o kombinaci *HTML* a jQuery, která má za úkol zpříjemnit práci s aplikací tím, že simuluje dialog, který je povědomý z desktopových aplikací.

Uživatel tedy vyplní data a odešle formulář. Data putují skrze router na `GoalPresenter`, kde spustí metodu `actionPridat`, kterou je možné vidět na ilustraci 22. Data jsou vytažena z kontejneru požadavku a následně ověřena oproti omezením (například čas splnění není záporný, jméno není prázdné a tak dále). Pokud jsou kontroly úspěšné, tak služba `goalRepository` přidá data do databáze. Následně je přidána stavová zpráva a uživatel je přesměrován.

```
public function actionPridat(){
    $success = true;

    $name = $this->getHttpRequest()->getPost("name");
    $category = $this->getHttpRequest()->getPost("category");
    $goal = $this->getHttpRequest()->getPost("goal");
    $terms = $this->getHttpRequest()->getPost("terms");
    $reminder = (integer)$this->getHttpRequest()->getPost("reminder");
    $ispublic = ($this->getHttpRequest()->getPost("ispublic") == "on") ? true : false;
    $date = new \Nette\DateTime();
    $until = new \Nette\DateTime($this->getHttpRequest()->getPost("until"));

    if(empty($name) or empty($goal) or empty($terms)){...4 lines }

    if($category == "0"){...3 lines }

    else if($this->categoryRepository->findById($category) == null){...4 lines }

    if($reminder > 3 or $reminder < 0 ){...4 lines }

    if($until <= $date){...4 lines }

    if ($success == true) {
        $g = $this->goalRepository->createGoal($this->getUser()->getId(), $name, $categ
        $this->flashMessage(self::GOAL_ADDED.$g["name"]);
    }
    $this->redirectBack();
}
```

Ilustrace 22: Ukázka metody actionPridat

4.7.2. Změna uživatelských oprávnění

Pro změnu uživatelských oprávnění je uživatelský komfort řešen ještě o něco lépe, používá se totiž *AJAX*. Pro správnou funkčnost je v Nette třeba splnit následující tří předpoklady:

1. Kód, který má být zpracován AJAXem, je nutné obalit tagem `snippet`. Ted vymezuje a identifikuje potřebnou oblast pro potřeby Nette.
2. Vložit JavaSciptovou knihovnu nette.ajax.js. Ta při inicializaci stránky projde všechny elementy s tagem `ajax` a nastaví jim na příslušnou událost (`onClick`, `onSubmit`) funkci, která zajistí odesílání dat AJAXem.
3. Využít ke zpracování dat metodu *presenteru* zvanou `handle`, viz ilustrace 19. Tato metoda se označuje jako *signál* a je vykonávána bez překreslení výsledných šablon. Je tedy pro tyto účely vhodná.

Ilustrace 23: Změna uživatelských oprávnění

Pro správné pochopení se podívejme na ilustraci 23. Takto vypadá ovládací prvek, který má na starost změnu uživatelských oprávnění. I když to není zřejmé, je obalen tagem snippet a na události tlačítka (`onClick`) uložit je nastavena zmíněná JavaScriptová procedura.

Po vybrání příslušné role a kliknutí na tlačítko uložit dojde k odeslání AJAXového požadavku, který opět putuje skrze router, tentokrát na metodu `handleZmenOpravneni` ve třídě `UserPresenter`. Viz ilustrace 24.

```
public function handleZmenOpravneni($profile_id, $role) {
    $profile = $this->userRepository->findById($profile_id);
    if ($profile == NULL) {
        $this->flashMessage(self::USER_DOESNT_EXIST, self::FLASH_ERROR);
    } else if (array_key_exists($role, $this->getUser()->authorizator->getRoles())) {
        $this->flashMessage(self::NO_PERMISSION, self::FLASH_ERROR);
    } else if ($this->getUser()->isAllowed("permission", "edit") and $profile->role !=
        $this->userRepository->changeRole($profile_id, $role);
        $this->flashMessage(self::USER_ROLE_CHANGE_SUCCESSFULL, self::FLASH_SUCCES);
    } else {
        $this->flashMessage(self::NO_PERMISSION, self::FLASH_ERROR);
    }
    $this->invalidateControl("permission");
    $this->invalidateControl("flash");
    //$this->redirectBack();
}
```

Ilustrace 24: Ukázka metody handleZmenOpravneni

Zde je nalezen profil, kterému chceme změnit uživatelská oprávnění. Pokud uživatel existuje a máme požadovaná přístupová práva, je za pomoci `userRepository` oprávnění změněno, pokud ne, je vrácena chybová hláška. Namísto přesměrování zpět se zde však nacházejí metody `invalidateControl`. Ty zajišťují překreslení aktuálními daty, a to té části šablony, která je uvedená v bloku identifikovaném pomocí daného textového řetězce.

4.7.3. Detail předsevzetí

Zobrazení profilu předsevzetí je ukázka metody na zobrazeni dat do šablony a využívá tedy prefixu `render`. Serverová část je velice jednoduchá a spočívá v nalezení předsevzetí, jehož identifikátor je předán z *URL (Uniform Resource Locator)*. Nalezené předsevzetí se doplní o několik dalších informací a vše se společně vloží do šablony. Na tomto kroku není nic zajímavého a dále se budu tedy věnovat pouze zpracování dat v šabloně.

Jednou z dodatečných informací, která je přiložena k předsevzetí, jsou příslušné komentáře vložené do proměnné `$comments`. Jejich výpis je patrný na ilustraci 25. Jak je možné si všimnout, tak pro výpis je použité Latte makro `foreach`, které projde každou položku v poli komentářů. Všechno co je vložené mezi se tedy zopakuje pro každé předsevzetí. Pokud má uživatel dostatečná oprávnění, zobrazí se mu i odkaz na smazání příslušného komentáře.

```
{foreach $comments as $comment}
    <div class="panel panel-default">
        <div class="panel-heading">
            {if $user->id == $comment->user_id or $user->isAllowed("comment", "delete") or $user->id == $
                <a n:href="Goal:odebratKomentar $comment->id" data-confirm="Opravdu si přejete smazat ten
            {/if}
            <h3 class="panel-title">
                <img class="img-circle" src="{$comment->user->email|gravatarUrl:25}"/> 
                <a n:href="User:profil $comment->user->id">
                    {($comment->user->nickname != ""?$comment->user->nickname:$comment->user->username)}
                </a>
                <small>({$comment->created|date:$mydateformat})</small>
            </h3>
        </div>
        <div class="panel-body">
            {$comment->comment}
        </div>
    </div>
{/foreach}
```

Ilustrace 25: Výpis komentářů v detailu předsevzetí

Nejzajímavější částí je však využití databázového *API* NotORM[24]. Při řešení podobné situace obvykle nastane problém, když je třeba vypsat jméno uživatele, který komentář přidal. Obvykle je třeba zjistit jméno uživatele a výsledné tabulky spojit a předat do šablony. V našem případě však máme pouze tabulku komentářů a přesto zápis `$comment->user->nickname` funguje bez problému. To je způsobeno tím, že pokud není pod požadovanou proměnou uvedena hodnota, NotORM provede dotaz na propojenou tabulku (se jménem proměnné). Do dotazu jsou automaticky správně doplněny identifikátory, což zajišťuje, že je výsledná hodnota doručena na požadované místo. Do jisté míry by se toto chování dalo považovat za nekorektní, protože způsobuje načítání dat z databáze v místě zpracování šablony. Toto chování je v rozporu s definicí funkce *view* v modelu *MVC*.

4.7.4. Vyhledávání

Nedílnou součástí aplikace je také vyhledávání v předsevzetích (viz ilustrace 26). V základním zobrazení lze vyhledávat podle klíčových slov a se základním filtrem podle stavu předsevzetí. Při kliknutí na ozubené kolečko se navíc zobrazí rozšířená nabídka filtrů. Vyhledávací formulář je odeslán po stisknutí tlačítka odeslat a stránka následně zobrazí výsledky vyhledávání. Formulář je vždy opětovně odeslán při jakékoliv změně filtru, což okamžitě aplikuje požadované filtrování.

Ilustrace 26: Ukázka vyhledávání předsevzetí

Na serverové straně je funkce implementována jako metoda `renderHledat` ve třídě `ViewPresenter`. Metoda přijímá jako vstupní parametry z *URL* adresy požadavku. Tyto parametry jsou postupně vloženy do pole `$by`, které slouží pro zúžení výběru vyhledání v databázi. Například pokud není uživatel administrátorem a nemůže tedy zobrazit soukromá předsevzetí, použijeme následující řádek:

```
$by['ispublic? OR user_id?'] = array (true, $current_user_id);
```

Nebo pokud chceme vyhledat pouze aktivní předsevzetí, tak použijeme:

```
$by['status'] = 'active';
```

Výsledné pole použijeme pro zpřesnění výběru z databáze.

```
$goal = $this->goalRepository->finadAllBy($by);
```

Výsledek vyhledávání je potom rozdělen na stránky komponentou `Nette/Utils/Paginator`, která zajišťuje výsledné stránkování v šabloně. Upravený výsledek vyhledávání je spolu s *paginatorem* předán pro další zpracování do šablony.

5. Zhodnocení a doporučení

5.1. Doporučení pro komunitní portály

Při vývoji portálu orientovaného na specifického uživatele jsem získal mnoho zkušeností, které lze formulovat jako sadu doporučení.. Mnoho těchto doporučení se úzce váže na zásady čistého a čitelného web designu a celkovou grafickou čistotu jakéhokoliv díla.

1. Základním požadavkem komunitního portálu je důkladná znalost specifické komunity. S dostatečnou znalostí je možné specifikovat si požadavky komunity, a ty správně interpretovat a reflektovat.
2. Velice důležitým aspektem každého uživatelského rozhraní je již dříve zmíněný princip *KISS (Keep It Simple, Stupid!)*. Pokud totiž nebude vaše aplikace snadná a jednoduše ovladatelná, nikdo ji používat nebude.
3. Uživatelé si často pamatují rozložení a barevné přechody, proto je dobré jim práci ulehčit jasným oddělením vašeho sdělení. Například červená tlačítka znamenají zápor nebo upozornění a používáme je ve většině kontextů vpravo. Ať je podobné uzpůsobení jakékoliv, mělo by být jednotné a konzistentní.
4. Jak již bylo řečeno, tak i poloha ovládacích prvků hraje svoji roli. Proto by i navigační menu měla být umístěna tam, kde je na ně uživatel zvyklý a očekává je. Výslednému efektu napomůže i jejich struktura a dělení do logických celků.
5. Z programátorského hlediska bývá často vhodné zobrazit všechny dostupné informace. To však může být někdy matoucí a nežádoucí. Je tedy dobré zaměřit se pouze na nezbytně nutné.
6. Pokud chceme zobrazit informací více, je vhodné pracovat efektivně s informační hodnotou v celkovém zobrazení. Důležité informace jsou velkým písmem a kontrastní barvou, doplňující informace jsou nekontrastní a malým písmem. Někdy může být vhodné rozšiřující informace zobrazit pouze při přejetí kurzorem nebo při kliknutí na symbol plus.
7. Během prohlížení stránek si uživatel často zapamatuje útržky informací, ke kterým by se rád vrátil. Je proto dobré mít k dispozici podrobné vyhledávání.
8. Pro komunitní portál je velice důležité, aby se uživatel necítil jako při vyplňování formulářů. Pokud je tedy třeba vyplnit nějaké informace, požadujte povinně opravdu jen ty nezbytně nutné a vyjděte maximálně vstříc při vyplňování.
9. Pokud se uživatelé chtějí k něčemu vyjádřit, dejte jim možnost. Je však důležité mít možnost diskusi moderovat, ne všichni uživatelé jsou totiž ochotni nebo ani schopni dodržovat zásady slušného chování.

5.2. Poznatky z vývoje pro mobilní platformu

Vývoj webové aplikace, která by měla být přizpůsobena jak pro velké monitory, tak pro mobilní zařízení, s sebou přináší mnohá úskalí. Na mnohá z nich jsem osobně narazil, rozšířím tedy volně předchozí kapitolu doporučení o specifika této problematiky.

1. Prvně je třeba si uvědomit, že mobilní zařízení neposkytuje mnoho prostoru pro zobrazení požadovaného obsahu. Je tedy třeba eliminovat zbytečné informace a zobrazit pouze relevantní údaje. Obecně by se dalo říci, že čím je informace důležitější, tím více by měla dostat prostoru či zvýraznění. Tyto minimalizace je často třeba udělat i na úkor některých informativních prvků. Není-li například místo pro zobrazení nejlepších článků vašeho blogu, může být lepší nezobrazovat je vůbec.
2. Druhý důležitý poznatek je fakt, že displej mobilního zařízení se může běžně vyskytovat na výšku. To je obrovský rozdíl v zobrazení, se kterým je třeba počítat. Proto je třeba některé zobrazované prvky webové aplikace zalomit pod sebe. Zde platí, že dlouhý web je lepší než široký.
3. Při úpravách je třeba si také uvědomit, že uživatelé mobilních zařízení jsou stále v pohybu. To často znamená, že někam spěchají a nebo něco rychle potřebují zjistit. Web by tedy měl být maximálně intuitivní a důležité funkce by měly být vzdáleny co nejméně kliknutí. Každé načtení stránky je drahé a takovýto uživatel nemá čas zjišťovat, jak aplikace vlastně funguje.
4. U přenosných zařízení musíme stále ještě vzít v potaz kvalitu a rychlost připojení, které může být proti běžnému domácímu připojení řádově třeba i stokrát pomalejší. Zvažme tedy následující optimalizace.
    1. Místo velkých obrázků na pozadí použijeme malou texturu nebo pouze barvu.
    2. Malý obrázek můžeme vložit kódovaný přímo do stylopisu.
    3. Bitmapové obrázky lze vyměnit za vektorové ve formátu *SVG (Scalable Vector Graphics)*, které jsou řádově menší.
    4. U obrázků je také dobré uvádět jejich velikost, zamezí to přeskakování stránky při dodatečném načítání obrázků.
    5. Používat minifikované verze přikládaných souborů.
    6. Snížit počet potřebných souborů sdružením do sebe. Tím snížíme počet vykonaných požadavků, což zvýší odezvu aplikace.
5. Dále je třeba si uvědomit, že mobilní přístroje nemají myš. Je tedy třeba zvážit použití pseudotřídy `:hover`, která se aktivuje po najetí myši. Výsledné prvky navigace by tedy nemusely správně fungovat, ne všechny prohlížeče mají tento nedostatek uspokojivě vyřešen. Stejně tak to platí pro události JavaScriptu spojené s myší.

6. V mobilních prohlížečích se také nelze spolehnout na zobrazení kompletní stránky, ať už kvůli rychlosti připojení nebo podpoře efektů či fontů. Nevsázejme tedy na předání klíčových informací pomocí obrázků či dobrovolných animací. Velké firemní logo možná vypadá dobře, ale pokud se z nějakého důvodu nezobrazí obrázek ani jméno firmy, je to veliký problém.
7. Někdy může stát za zvážení vypnutí některých grafických efektů či složitých animací, což šetří zatížení procesoru, a tím i využití baterie.
8. Skripty se obvykle na stránce spouštějí až po načtení struktury, proto je vhodné umístit elementy typu `script` až úplně nakonec. Jejich načítaní tím začne až potom, co je stránka vykreslena.

5.3. Testy pro různá rozlišení

Výsledná aplikace byla podrobena testům v různorodých zařízeních a zároveň za použití různých internetových prohlížečů. Srovnání a úspěšnost optimalizací je vyobrazena v tabulce číslo 2. Pro zjednodušení je vzhled hodnocen body od jedné do deseti, kde 10 znamená kvalitně přizpůsobeno rozlišení a naopak. Funkčnost používá stejnou stupnici a 10 zde znamená, že je aplikace pohodlně použitelná a všechny ovládací prvky fungují správně. Uvedené prohlížeče jsou použity v nejaktuálnější verzi ke květnu 2014.

Zařízení	Obrazovka	Prohlížeč	Vzhled	Funkčnost
Stolní počítač, Windows 8	22"	Chrome	10	10
		IE10	9	9
		Firefox	10	10
		Opera	10	10
		Safari	9	9
Notebook Leonovo Z500, Windows 8	15,6"	Chrome	10	10
		IE10	9	9
		Opera	10	10
Tablet Sencor Element, Android 4.0	7"	Systémový	9	9
		Chrome	9	9
Mobilní telefon LG E460, Android 4.1	5"	Systémový	8	9
		Chrome	9	9
		Opera Mini	8	8

Tabulka 2: Test aplikace v různých rozlišeních a prohlížečích

6. Závěr

V rámci této práce vznikla webová aplikace Moje předsevzetí, která bude sloužit mnoha lidem v jejich průběžné motivaci a sledování dlouhodobých cílů. Aplikace využívá na míru navržené uživatelské rozhraní, které klade důraz na uživatelskou přívětivost a přístupnost. V rámci testování aplikace bylo toto rozhraní otestováno na několika dostupných zařízeních a v různých prohlížečích. Kvůli předpokládaným změnám v požadavcích byla při tvorbě s úspěchem využita metodika SCRUM, díky které se všechny změny podařilo realizovat s minimální časovou náročností.

Za veliký přínos považuji také týmovou práci při opravách knihovny pro přihlašování do webových služeb. Společná revize kódu byla pro mne velice cennou zkušeností a až zde jsem plně pochopil některé funkce verzovacích nástrojů jako je Git. Navíc byly společně provedeny úpravy, které napomohou v realizaci budoucích projektů. Těch v současné době přibývá a knihovna je hojně využívána a je dostupná na GitHubu.

Aplikace se nyní nachází ve stavu testování a návrhy z řad uživatelů a grafiků jsou postupně zapracovávány. Web bude brzy připraven na ostrý provoz. Pravděpodobně se však vyčká s vypuštěním na vhodnou příležitost, která se naskytne až v září se začátkem nového školního roku.

Během tvorby práce jsem zavadil o mnoho zajímavých odvětví vývoje webových aplikací. Měl jsem možnost graficky navrhovat uživatelská rozhraní a testovat jejich použitelnost. Vytvářel jsem dynamické prvky stránky, díky čemuž jsem se obohatil o praktickou znalost JavaScriptu a jQuery. Z profesního hlediska to jsou pro mne všechno velice cenné zkušenosti a myslím, že mi pomohou na mé budoucí profesní dráze.

Seznam použité literatury

[1] VRÁNA, Jakub. *PHP triky* [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://php.vrana.cz/

[2] KOFLER, Michael a Bernd ÖGGL. *PHP 5 a MySQL 5: průvodce webového programátora.* Vyd. 1. Překlad Jan Pokorný. Brno: Computer Press, 2007, 736 s. Encyklopedie Zoner Press. ISBN 978-80-251-1813-9

[3] VRÁNA, Jakub. *1001 tipů a triků pro PHP.* Vyd. 1. Brno: Computer Press, 2010, 456 s. ISBN 978-80-251-2940-1.

[4] NETTE FOUNDATION. *Nette Framework* [online]. © 2008, 2014 [cit. 2014-04-10]. Dostupné z: http://nette.org/

[5] Šablony. NETTE FOUNDATION. *Nette Framework* [online]. © 2008, 2014 [cit. 2014-04-10]. Dostupné z: http://doc.nette.org/cs/2.1/templating

[6] Konfigurace. NETTE FOUNDATION. *Nette Framework* [online]. © 2008, 2014 [cit. 2014-04-23]. Dostupné z: http://doc.nette.org/cs/2.1/configuring

[7] ORACLE CORPORATION. *Dokumentace databáze MySQL* [online]. 2014 [cit. 2014-05-5]. Dostupné z: http://www.mysql.com/

[8] JAVA WORLD. *Secure Web services* [online]. 2014 [cit. 2014-04-14]. Dostupné z: http://www.javaworld.com/article/2073287/soa/secure-web-services.html

[9] *Web Projektu Moje předsevzetí* [online]. 2014 [cit. 2014-04-12]. Dostupné z: mp.skauting.cz

[10] PILGRIM, Mark. *Dive into HTML5* [online]. 2014 [cit. 2014-04-14]. Dostupné z: kniha.html5.cz

[11] *CSS3: Kóderův průvodce moderní frontend technologií* [online]. 2014 [cit. 2014-05-5]. Dostupné z: http://www.vzhurudolu.cz/prirucka/css3

[12] *Bootstrap: Dokumentace k frameworku* [online]. 2010 [cit. 2014-04-12]. Dostupné z: http://getbootstrap.com/

[13] *Less: Css pre-processor* [online]. 2014 [cit. 2014-04-14]. Dostupné z: http://lesscss.org/

[14] CHAFFER, Jonathan. *Mistrovství v jQuery: kompletní průvodce vývojáře*. 1. vyd.

Brno: Computer Press, 2013, 384 s. ISBN 978-80-251-4103-8

[15] THE JQUERY FOUNDATION. *jQuery dokumentace* [online]. 2006 [cit. 2014-04-14]. Dostupné z: http://jquery.com/

[16] FRAIN, Ben. *Responsive web design with HTML5 and CSS3: learn responsive design using HTML5 a CSS3 to adapt websites to any browser or screen size*. Birmingham: Pack Publishing, c2012, vi, 305 s. ISBN 978-1-84969-318-9

[17] HLAVA, Tomáš. *Testování softwaru* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://testovanisoftwaru.cz/

[18] WIKIMEDIA FOUNDATION. *Wikipedia.org: Extrémní programování* [online]. 2013 [cit. 2014-05-07]. Dostupné z: http://cs.wikipedia.org/wiki/Extrémní_programování

[19] *Zdroják.cz: Seriál: Agilní vývoj* [online]. 2014 [cit. 2014-04-14]. Dostupné z: http://www.zdrojak.cz/clanky/agilni-vyvoj-uvod/

[20] *Gravatar – Celosvětově uznávané Avatary* [online]. [2005] [cit. 2014-04-23]. Dostupné z: http://cs.gravatar.com/

[21] *Composer: Dependency Manager for PHP* [online]. 2014 [cit. 2014-04-14]. Dostupné z: https://getcomposer.org/

[22] *SkautIS: Knihovna pro podporu webových služeb* [online]. 2014 [cit. 2014-04-14]. Dostupné z: https://github.com/sinacek/SkautIS

[23] Píšeme první aplikaci. NETTE FOUNDATION. *Nette Framework* [online]. © 2008, 2014 [cit. 2014-04-10]. Dostupné z: http://doc.nette.org/cs/2.1/quickstart

[24] *NotORM: Dokumentace knihovny* [online]. 2014 [cit. 2014-04-14]. Dostupné z: http://www.notorm.com/

[25] MVC aplikace & presentery. NETTE FOUNDATION. *Nette Framework* [online]. © 2008, 2014 [cit. 2014-04-10]. Dostupné z: http://doc.nette.org/cs/2.1/presenters

[26] *Test SkautIS: Dokumentace pro vývojáře* [online]. 2014 [cit. 2014-04-14]. Dostupné z: http://is.skaut.cz/napoveda/programatori. Default.aspx

[27] VRÁNA, Jakub a Bernd ÖGGL. *Design patterns: elements of reusable object-oriented software*. Vyd. 1. Překlad Jan Pokorný. Boston: Addison-Wesley, 1995, 395 s. Encyklopedie Zoner Press. ISBN 02-016-3361-2

Přiložené CD

Na přiloženém CD jsou umístěny všechny zdrojové kódy aplikace vyjma konfiguračního souboru, ve kterém byla uvedena přístupová hesla a další potenciálně nebezpečné údaje. Dále je přiložena tato práce v textovém formátu a také v pdf spolu se všemi poklady a obrázky, které byly pro její potřeby vytvořeny.