

České vysoké učení technické v Praze  
Fakulta elektrotechnická

katedra počítačů

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Radek Ježdík**

Studijní program: Softwarové technologie a management  
Obor: Softwarové inženýrství

Název tématu: **Slide-it: nástroj pro HTML5 prezentace**

Pokyny pro vypracování:

Vytvořte nástroj pro tvorbu a sdílení HTML5 prezentací na webu. Webový backed umožní tvorbu a editaci slidů a jejich sdílení. Pro prezentaci použijte vhodnou JavaScriptovou knihovnu, kterou rozšířte o možnost testování čtenářů prezentace. Další požadavky na software sesbírejte na základě vytvořeného prototypu.

Očekávaný výstup aplikace:


1. Analýza, návrh a implementace webové aplikace pro tvorbu, editaci a sdílení prezentací.
2. Rozšíření prezentační JS knihovny.
3. Nasazení a otestování aplikace.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Ondřej Macek

Platnost zadání: do konce zimního semestru 2013/2014

  
doc. Ing. Miroslav Šnorek, CSc.  
vedoucí katedry



  
prof. Ing. Pavel Ripka, CSc.  
děkan

V Praze dne 15. 2. 2013



České vysoké učení technické v Praze  
Fakulta elektrotechnická  
Katedra počítačů



Bakalářská práce

## Slide-it: nástroj pro HTML5 prezentace

*Radek Ježdík*

Vedoucí práce: Ing. Ondřej Macek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

20. května 2013



## Poděkování

Rád bych zde poděkoval své rodině, která mě po celou dobu studia podporovala. Dále bych chtěl poděkovat Ing. Ondřeji Mackovi, vedoucímu této bakalářské práce, za odlehčený přístup a komunikaci v přátelském duchu.



## Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v příloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 24. 5. 2013

.....





# Abstract

The objective of this thesis is to design and develop a tool for creating and sharing HTML presentations online. The goal is to build a web application, which will provide tools for creating web-based presentations for the purpose of viewing and sharing. The content of this work covers analysis, designing, implementation and testing of the system.

# Abstrakt

Předmětem této práce je návrh a vývoj webového nástroje pro tvorbu a sdílení HTML prezentací. Cílem je vytvořit webovou aplikaci, která umožní tvorbu webových prezentací za účelem jejich prohlížení a sdílení. Obsahem této práce je analýza, návrh, implementace a testování systému.



# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
1.1	Cíle	1
1.2	Příležitosti	2
<b>2</b>	<b>Analýza</b>	<b>3</b>
2.1	User stories	3
2.2	Nefunkční požadavky	4
2.3	Doménový model	5
2.3.1	Uživatel	5
2.3.2	Prezentace	6
2.3.3	Koncept	6
2.3.4	Komentář	6
2.4	Co je HTML prezentace?	6
2.5	Výběr JavaScriptové knihovny	7
2.5.1	Reveal.js	7
2.5.2	Impress.js	8
2.5.3	Google I/O HTML5 slide template	8
2.5.4	Deck.js	8
2.5.5	Zhodnocení	8
2.6	Vize	8
2.6.1	Webové rozhraní	9
2.6.2	Editor	9
2.6.3	Prohlížení prezentace	10
2.7	Rešerše existujících řešení	10
2.7.1	SlideShare	10
2.7.2	Prezi	10
2.7.3	Rvl.io	10
2.7.4	Shrnutí	11
<b>3</b>	<b>Návrh</b>	<b>13</b>
3.1	Serverová část	13
3.1.1	Nette Framework	13
3.1.2	MVP	13
3.1.3	Architektura aplikační logiky	14
3.2	Klientská část	14

3.3	Editor	15
<b>4</b>	<b>Realizace klientské části</b>	<b>17</b>
4.1	Prototyp editoru prezentací	17
4.1.1	WYSIWYG editor	17
4.1.2	Editor založený na jednoduchém značkovacím jazyce	18
4.2	Rozložení editoru	18
4.3	Chování editoru	19
4.4	Úprava knihovny Texy!	19
4.4.1	Markdown syntaxe	20
4.4.2	Nová syntaxe	21
4.4.2.1	Syntaxe odpovědí	21
4.4.2.2	Syntaxe doplňkových textů	22
4.4.2.3	Syntaxe pro vložení osnovy	22
4.5	Rozšíření prezentační knihovny	22
4.5.1	Zvýraznění syntaxe programovacích jazyků	23
4.5.2	Matematické vzorce	23
<b>5</b>	<b>Nasazení</b>	<b>25</b>
5.1	Požadavky	25
5.2	Realizace nasazení	25
5.3	Deployment	26
<b>6</b>	<b>Testování</b>	<b>27</b>
6.1	Test Driven Development	27
6.2	Jednotkové testy	28
6.3	Integrační testy	28
6.4	Selenium testy	28
6.5	Zátěžové testy	29
<b>7</b>	<b>Zhodnocení a budoucí vývoj</b>	<b>31</b>
7.1	Budoucí vývoj	32
7.1.1	Náhledy prezentací	32
7.1.2	Notifikace	32
7.1.3	Import a export prezentací	32
7.1.4	Členění a seskupování prezentací	32
7.1.5	Bezpečnost	33
<b>8</b>	<b>Závěr</b>	<b>35</b>
<b>A</b>	<b>Seznam použitých zkratk</b>	<b>39</b>
<b>B</b>	<b>Ukázky kódu</b>	<b>41</b>
<b>C</b>	<b>Grafy průběhů zátěžových testů</b>	<b>45</b>
<b>D</b>	<b>Obsah příloženého CD</b>	<b>49</b>

# Seznam obrázků

2.1	Diagram doménového modelu . . . . .	5
3.1	Ukázka průchodu aplikace . . . . .	14
3.2	Diagram balíčků . . . . .	15
4.1	Ukázka editoru . . . . .	19
4.2	Ukázka snímku s otázkou a odpověďmi . . . . .	21
4.3	Ukázka snímku s matematickými rovnicemi . . . . .	24
C.1	Graf průběhu zátěžového testu stránky s prezentací . . . . .	46
C.2	Graf průběhu zátěžového testu stránky s prezentacemi uživatele . . . . .	47



# Kapitola 1

## Úvod

V současné době se většina aplikací, nástrojů a služeb přesouvá na internet, kde se mimo jiné klade důraz na sdílení mezi uživateli. Prezentace nejsou výjimkou. Jejich sdílení a tvorba online není originální myšlenka a na trhu existuje řada známých i méně známých služeb, které v této oblasti dosáhly dobrých výsledků. Není ale mnoho podobných služeb, které by využívaly výhod tzv. HTML prezentací, tedy prezentací založených na rodině webových technologií HTML, CSS a JavaScript. Takové prezentace pak mohou těžit maximum ze schopností moderních webových prohlížečů a technologií a zároveň není potřeba žádných rozšíření třetích stran, neboť samotná prezentace je stále jen běžná webová stránka.

### 1.1 Cíle

Tento projekt má za cíl vytvořit webovou aplikaci, která uživatelům umožní nejen snadné sdílení vlastních prezentací, ale také jejich tvorbu. Jediným nástrojem tak uživatel prezentaci vytvoří a poté ji v rámci aplikace či jiných komunikačních kanálů, jako jsou sociální sítě, může šířit dál.

Protože na trhu již existují služby, které podobné možnosti nabízejí, chtěl jsem projekt posunout dál a využít výhod, které HTML prezentace poskytují. Tvorbu a prohlížení prezentací jsem proto více zaměřil na použití ve vzdělávacích institucích či školách a nebo odborných konferencích, kde je důležité v prezentacích podat důležité informace, tedy nejen hlavní body přednášky.

Aplikace bude poskytovat prostředky pro zjednodušení tohoto cíle. V prezentacích tak bude možné barevně zvýraznit kód programovacích jazyků či zobrazit matematické vzorce pomocí syntaxe  $\text{\LaTeX}$ . Navíc bude možné do snímků vkládat kontrolní otázky, na které budou moci uživatelé odpovídat, a tak si ověřit své znalosti, případně lze přidat i doplňující text s odkazem na konkrétní snímek s odpovědí. Stejně tak je důležité, aby tyto prezentace byly použitelné i mimo přednášku a podaly významné informace i bez mluveného slova, například pro domácí přípravu nebo vlastní rozvoj.

## 1.2 Příležitosti

Od práce na tomto projektu si slibuji větší porozumění technologii HTML5 a JavaScriptu. Zajímavou příležitostí je i vyzkoušení si tvorby webových stránek, které nejsou jen dalším typickým informačním systémem. V tomto projektu lze najít spoustu problémů, ale také nové a zajímavé technologie, na kterých si vyzkouším vývoj moderních klientských aplikací.



# Kapitola 2

## Analýza

V této kapitole jsou představeny požadavky a vize vyvíjeného systému. Čtenář je uveden do problematiky HTML prezentací a s tím jsou představeny i knihovny, které zajišťují funkčnost těchto prezentací. Nakonec jsou stručně popsány podobné již existující systémy.

### 2.1 User stories

Následující odstavce uvádí uživatelské role, které se při používání systému objevují. Každá role pak má nějaká práva či možnosti využití systému. Popis je zapsán pomocí techniky zvané user story používané v agilních metodikách vývoje softwaru.

Každý odstavec je uveden názvem uživatelské role, za níž následuje výčet možností, které systém dané roli dává.

**Nepřihlášený uživatel** bude mít možnost:

1. **prohlížet veřejně dostupné prezentace**, k tomu použije:
  - 1.1. vyhledávací formulář na webu,
  - 1.2. adresu URL na stránku s prezentací, kterou dostane jiným komunikačním kanálem,
2. **vyhledat prezentace** vyhledávacím formulářem podle názvu nebo autora, aby mohl
  - 2.1. zhlédnout hledanou prezentaci,
  - 2.2. číst komentáře pod prezentací,
  - 2.3. zobrazit prezentace hledaného uživatele,
3. **přihlásit se** do systému pomocí přihlašovacího formuláře, aby získal oprávnění přihlášeného uživatele (viz níže),
4. **registrovat se** do systému pomocí registračního formuláře, aby se mohl přihlásit.

**Přihlášený uživatel** bude mít kromě *registrace* a *přihlášení* stejné možnosti jako *nepřihlášený uživatel*. Navíc bude mít možnost:

1. **vytvářet** prezentace pomocí editoru prezentací, aby je poté mohl prohlížet a sdílet,
2. **posílat komentáře** k prezentacím, aby dal zpětnou vazbu autorovi prezentace či diskutoval s ostatními uživateli,
3. **prohlížet veřejné a jemu sdílené** prezentace,
4. **vyhledat uživatele** vyhledávacím formulářem podle uživatelského jména, aby mohl:
  - 4.1. **sledovat** hledaného uživatele, aby byl upozorněn na jeho nové prezentace,
  - 4.2. **zrušit sledování** hledaného uživatele,
5. **vidět seznam jemu sdílených prezentací**, aby si je mohl prohlédnout,
6. **vidět seznam veřejných prezentací** uživatelů, které sleduje, aby si je mohl prohlédnout,
7. **změnit své heslo**,
8. **odhlásit se**.

**Autor prezentace**, tj. přihlášený uživatel, který vytvořil prezentaci, bude mít možnost:

1. **upravit prezentaci** v editoru, aby ji o něco doplnil nebo opravil chyby,
2. **smazat prezentaci**, aby ji odstranil ze systému,
3. **smazat jakýkoliv komentář** k vlastní prezentaci, aby moderoval diskuzi,
4. **sdílet prezentaci** s ostatními uživateli systému, aby si ji mohli prohlédnout.

**Autor komentáře**, tj. přihlášený uživatel, který napsal komentář, bude mít možnost smazat svůj komentář.

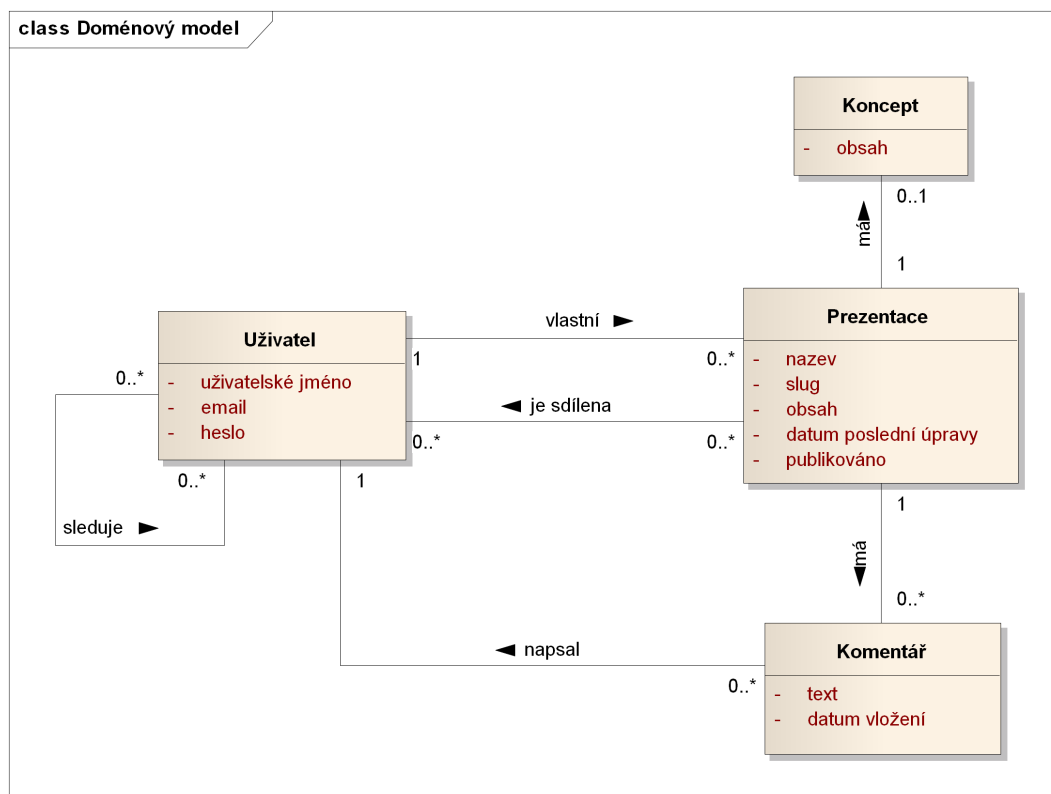
## 2.2 Nefunkční požadavky

Následující seznam uvádí nefunkční požadavky, tedy omezující podmínky, které systém musí dodržet. Podmínky vycházejí ze zadání tohoto projektu.

1. Systém bude přístupný přes webové rozhraní
2. Systém bude pro uživatele nezávislý na platformě
3. Systém bude využívat technologii HTML5
4. Pro prezentace bude systém využívat JavaScriptovou knihovnu

## 2.3 Doménový model

Z user stories vychází doménový model, který ukazuje celý systém jako provázané entity. Celý model je přehledně zobrazen na obrázku 2.1 v jazyce UML. Popis jednotlivých entit a jejich vlastností následuje v odstavcích níže.



Obrázek 2.1: Diagram doménového modelu

### 2.3.1 Uživatel

*Uživatel* představuje registrovaného uživatele systému. Registrovaný a přihlášený uživatel má výhody oproti nepřihlášenému (více na str. 3).

Pro úspěšnou registraci nového uživatele do systému musí nepřihlášený uživatel zadat:

- *uživatelské jméno*, pod kterým se bude prezentovat a přihlašovat. Stejně uživatelské jméno musí být v systému registrováno jenom jednou,
- *e-mail*, který musí být v systému jedinečný,
- *heslo* o minimální délce 4 znaků.

*Uživatelé* se mohou navzájem tzv. sledovat. Uživatel je tak snadněji upozorněn na nové prezentace od sledovaných autorů.

### 2.3.2 Presentace

Tato entita představuje jednu prezentaci, kterou uživatel pomocí systému vytvořil, tedy je jejím autorem. Má tyto povinné vlastnosti:

- *název*, pod kterým se daná prezentace bude zobrazovat ve výpisech,
- *slug* – zjednodušený název (malá písmena anglické abecedy, čísla a pomlčky), pod kterým bude prezentace dostupná přes adresu URL. Pro daného autora je jedinečný, tj. kombinace uživatelského jména a slugu je unikátní,
- *obsah*, jenž představuje složitější objekt textů jednotlivých snímků a různých nastavení prezentace.

Při vytvoření je prezentace ve stavu *nepublikováno*. Stav se změní na *publikováno* při publikaci prezentace v editoru. Pak je prezentace viditelná ostatním uživatelům v systému.

*Prezentace* může být sdílena konkrétním *uživatelům* systému.

### 2.3.3 Koncept

*Koncept* je obsah prezentace (popis snímků, nastavení), který nebyl zatím publikován. Je viditelný pouze v editoru pro autora prezentace. Při publikaci změn prezentace se obsah konceptu přkopíruje do obsahu vlastní entity prezentace, změny jsou pak viditelné i ostatními uživateli.

### 2.3.4 Komentář

*Komentář* je uživatelem napsaná poznámka k dané prezentaci. *Komentář* může vkládat každý přihlášený uživatel. Povinné vlastnosti jsou:

- *text komentáře* – obsah zprávy,
- *datum a čas*, kdy byl komentář vložen.

*Komentář* se váže k jedné *prezentaci* a jednomu *uživateli* – autorovi komentáře.

## 2.4 Co je HTML prezentace?

Prezentace založené na HTML jsou z pohledu kódu běžná webová HTML stránka. K popisu obsahu snímků se používají sémantické značky běžně používané v HTML (např. nadpisy, odstavce, obrázky). Pro úpravu grafické podoby se pak používá CSS. S použitím technologie HTML5 lze pak využít i nových možností CSS3, například animace přechodů. K oživení takové stránky a vytvoření dojmu prezentace včetně interakce s uživatelem pro přechod mezi snímky se používá JavaScript. To vše a další vylepšující funkce jsou zajištěny mnoha existujícími JavaScriptovými knihovnami.

Každá z existujících knihoven používá rozdílná pravidla pro konfiguraci prezentace či jednotlivých snímků. Následující výpis ukazuje zdrojový kód jednoduché HTML prezentace při použití knihovny deck.js.

```
<body class="deck-container">
  <section class="slide">
    <h1>Moje prezentace</h1>
  </section>

  <section class="slide">
    <h2>Nadpis snímku</h2>
    <p>Osnova:</p>
    <ul>
      <li>Bod 1</li>
      <li>Bod 2</li>
      <li>Bod 3</li>
    </ul>
  </section>
</body>
```

Výpis 2.1: Ukázka zdrojového kódu HTML prezentace (deck.js)

Zobrazení právě jednoho snímku, přecházení mezi snímky a držení stavu prezentace pak obstarává knihovna samotná. Na autorovi prezentace je pak napsat obsah snímků a dodržet konvence požadované knihovnou.

## 2.5 Výběr JavaScriptové knihovny

Knihoven pro zobrazení prezentací založených na technologii (X)HTML byla vytvořena celá řada [28]. Většina z nich je ovšem už zastaralá, nefunkční nebo neodpovídající požadovaným kvalitám, například použitelností (uživatelské či implementační) nebo zastaralým vzhledem. Následující seznam uvádí několik zdařilých knihoven, které se objevily po uvedení podpory HTML5 v prohlížečích.

- reveal.js
- impress.js
- Google I/O HTML5 slide template
- deck.js

Následující odstavce stručně popisují každou z uvedených knihoven. Nakonec je uvedeno jaká knihovna byla vybrána pro tento projekt a proč.

### 2.5.1 Reveal.js

Reveal.js [30] je vizuálně pěkná knihovna, která kromě posunu snímků horizontálně podporuje také vnořené snímky, na které se lze dostat posunem dolů. Má několik zajímavých rozšíření, např. podpora syntaxe Markdown pro psaní obsahu a barevné zvýraznění kódu programovacích jazyků pomocí knihovny highlight.js. V novém okně prohlížeče pak poskytuje zobrazení prezentace pro přednášejícího, který ukazuje aktuální a následující snímek včetně poznámek.

### 2.5.2 Impress.js

Impress.js [16] se zaměřuje na vizuální jedinečnost prezentace. Dosahuje toho složitými transformacemi přechodů mezi snímky. Ty jsou rozložené v prostoru a to nejen ve 2D, ale i ve 3D. Snímky tak lze mít v různých velikostech, pozicích a orientacích a při přechodu mezi nimi se pohled působivě otáčí a přesouvá.

Nevýhoda této knihovny je právě v tomto jednoúčelově pojatém zobrazení. Pro tento projekt je tak knihovna až moc složitá, hlavně kvůli určení pozic snímků a definici transformací, a pro informativní prezentace se příliš nehodí, neboť přechody mezi snímky spíše vyrušují.

### 2.5.3 Google I/O HTML5 slide template

Šablonu snímků od Googlu [14] používají při svých prezentacích vývojáři na konferencích Google I/O. Knihovna je proto v dobré kvalitě a velmi dobře uzpůsobená pro přednášející. Obsahuje tzv. prezentační mód podobný tomu v reveal.js. Poskytuje také zvýraznění kódu programovacích jazyků a to včetně možnosti zdůraznit jen určitou část kódu, které se přednášející chce věnovat podrobněji.

Knihovna ovšem nenabízí žádnou dokumentaci, a tak snižuje možnost pro další rozšíření. Dále má prezentace pevnou velikost, na menších displejích či oknech jsou tak snímky oříznuté. Nevýhodou je také o něco složitější konfigurace.

### 2.5.4 Deck.js

Knihovna deck.js [9] je v základu o něco jednodušší než knihovny výše uvedené, ale podporuje všechny základní funkce. Je napsaná s pomocí knihovny jQuery [17] a je velmi dobře zdokumentovaná a rozšiřitelná. Existuje pro ni řada pluginů, například již zmíněný prezentační mód či podpora Markdown syntaxe pro popis snímků.

### 2.5.5 Zhodnocení

Všechny výše uvedené knihovny umí v základu to samé, a to zobrazovat jednotlivé snímky a přecházet mezi nimi. Pro účely tohoto projektu byla vybrána knihovna deck.js, která neposkytuje tolik doplňkových možností, ale líbí se mi její celkový vzhled, jednoduchost a integrace s jQuery. Lze také využít existujících rozšíření nebo vytvořit vlastní. Knihovna reveal.js ovšem urazila během práce na tomto projektu velký kus cesty a v současné době je asi nejvíce rozšířenou a používanou knihovnou pro HTML prezentace. O knihovně již byla také napsána výuková kniha [11].

## 2.6 Vize

Následující odstavce popisují mojí vizi systému. Text je rozdělen do sekcí, jenž představují samostatné logické celky systému.

### 2.6.1 Webové rozhraní

Pro uživatele slouží web jako prostředník pro sdílení prezentací – ať už v rámci systému mezi jeho uživateli nebo posíláním URL odkazů na prezentace v emailech, příspěvcích na sociálních sítích jako Facebook nebo Twitter a dalších. V principu se tak podobá službám jako jsou YouTube, SlideShare nebo GitHub, které se zaměřují na vystavení vlastní tvorby (videí, prezentací, kódu) a její sdílení.

Sdílení prostřednictvím webové aplikace je dvojího typu. Uživatelé mohou sledovat jiné uživatele a o aktivitách těchto sledovaných uživatelů, například publikace nové prezentace, budou snadněji upozorněni. Druhý typ je pak sdílení opačným směrem – od autora prezentace jednotlivým uživatelům aplikace. Uživatelé si autor vybere podle uživatelského jména či e-mailu.

Důležitá je také zpětná vazba od uživatelů prohlížející si prezentace. Řešením je možnost přidávat komentáře k jednotlivým prezentacím.

### 2.6.2 Editor

Důležitou částí tohoto projektu je editor pro úpravy prezentace. Protože je obsah prezentace tvořen pomocí HTML, nabízí se několik možností, jak editaci pojmout, a to:

1. psaním samotného HTML kódu,
2. využitím editoru typu What You See Is What You Get (WYSIWYG),
3. využitím jednoduchého textového značkovacího jazyka (lightweight markup language).

Psaní samotného HTML kódu bylo vyloučeno bez většího váhání. HTML není nijak úsporný značkovací jazyk a pro uživatele, kteří neznají HTML, by byl překážkou. Na druhou stranu ale poskytuje největší možnou kontrolu nad podobou a funkčností prezentace. Kdyby ovšem editor měl být založen pouze na psaní HTML kódu, nebyl by vůbec potřeba – pro většinu uživatelů by pak bylo výhodnější použít nástroje, na které jsou zvyklí, a pak pouze nahrát hotovou prezentaci na web.

Editor WYSIWYG je koncept dobře známý z desktopových aplikací. Jedná se o přímou úpravu obsahu a uživatel nepřichází do styku se zdrojovým kódem. Jeho největší výhodou je pak jednoduchost pro uživatele, kteří výsledek svých úprav vidí okamžitě. Ve webovém prostředí se používá pro jednoduché vytváření stránek, například v CMS systémech. Nevýhodou je, že neexistuje standard, který by byl dodržován všemi prohlížeči. Poskytuje také menší kontrolu nad výsledným kódem.

Vybrána byla nakonec poslední možnost – jednoduchý textový značkovací jazyk, jímž se bude popisovat obsah snímku. Zdrojový text psaný v tomto jazyce se následně převádí do kódu HTML. Více je popsáno v kapitole [4 Realizace klientské části](#) na str. 17.

### 2.6.3 Prohlížení prezentace

Zajímavou vlastností, která se liší od ostatních webových služeb pro prohlížení prezentací, byla možnost poskytnout větší interaktivitu s uživatelem. Vzhledem k tomu, že je prezentace běžná HTML stránka, lze využít možností webových technologií pro vytvoření nových vlastností prezentace.

Jednou z nich je možnost odpovídat na otázky a kontrolovat správné odpovědi přímo v prezentaci. Uživatelé si pak mohou ověřit, co se z prezentace dozvěděli. Navíc je možné zobrazit doplňkový text při správné či špatné odpovědi. Ten pak může obsahovat nejen vysvětlení správné odpovědi, ale třeba i odkaz na snímek týkající se dané otázky.

## 2.7 Rešerše existujících řešení

Během analýzy byly také vyhledány a vyzkoušeny již existující služby, jejichž funkčnost se nejvíce podobá zadání tohoto projektu. Následující odstavce popisují jejich funkce a rozdíly.

### 2.7.1 SlideShare

SlideShare [33] je velmi známá webová služba pro sdílení prezentací. Obsahuje podobné mechanismy sdílení a zpětné vazby a umožňuje prezentace umisťovat na jiné stránky (tzv. embedded prezentace). Prezentace jsou ovšem velmi statické – bez animací a bez větší interaktivity s uživatelem. Některé prezentace jsou jen obrázkové, nejde tedy označit text ani následovat odkaz. Největší nevýhodou je, že prezentace nelze přímo tvořit. Lze je pouze nahrávat ve formátech PDF, PPT a dalších z lokálního úložiště. Jakákoliv změna v prezentaci se tak musí nahrát znovu.

### 2.7.2 Prezi

Prezi [29] se na rozdíl od ostatních daleko více zaměřuje na grafickou stránku a zakládá si na kreativitu uživatelů a vyjádření jejich idejí pomocí celkového vzhledu prezentace. Prezentace se značně liší od běžných prezentací, které známe například z Microsoft PowerPoint. Jejich použití se hodí hlavně pro vyjádření nějaké myšlenky či vize, méně se hodí pro podání důležitých informací. Velmi dobrou vlastností je živé sdílení, tj. procházení prezentací se synchronizuje mezi více online uživateli. Pro zobrazení prezentace používá technologii Flash.

### 2.7.3 Rvl.io

Rvl.io [31] je služba od autora knihovny reveal.js popsané na str. 7, kterou také používá pro zobrazení prezentací. Služba umožňuje sdílení i tvorbu prezentací přímo na webu. Sdílení je založeno na předávání adresy URL, poskytuje ale také vkládání prezentace do jiných stránek, podobně jako SlideShare. Pro úpravu prezentací využívá WYSIWYG editor, který ale poskytuje jen nezbytné minimum funkcí. Tato služba je velmi mladá a uvedena byla měsíc po započetí práce na tomto projektu. V době psaní této práce byla stále ve fázi beta testování.



#### 2.7.4 Shrnutí

Rvl.io jako jediný z vybraných systémů používá pro zobrazení prezentací HTML. Také poskytuje možnost tvorby prezentace a její sdílení, nejvíce se tak podobá zadání tohoto projektu. Neposkytuje ovšem žádnou přidanou hodnotu, která by se dala ve webovém prostředí využít. SlideShare staví do popředí hlavně sdílení prezentací a má proto bohaté mechanismy. Služba Prezi, i když velmi vydařená a propracovaná služba, naopak míří zcela na jinou cílovou skupinu.



# Kapitola 3

## Návrh

V následujícím textu je čtenář seznámen s detailním návrhem systému, použitými technologiemi a knihovnami.

### 3.1 Serverová část

Webový back-end jsem se rozhodl napsat v programovacím jazyce PHP s použitím českého webového aplikačního frameworku Nette [24]. Pro uchování dat byla použita databáze MySQL [23].

#### 3.1.1 Nette Framework

Nette Framework je aplikační framework, který usnadňuje řešení mnoho problémů v oblasti webového vývoje v PHP. Implementuje návrhový vzor Model-View-Presenter (MVP) rozdělující aplikaci na logické vrstvy (viz níže). Běh aplikace je řízen událostmi, tedy využívá událostmi řízené programování. Dále obsahuje komponentový model zaměřený na znovupoužitelnost.

#### 3.1.2 MVP

Návrhový vzor Model-View-Presenter řeší problém oddělení datové vrstvy, řízení událostí (např. od uživatele) a vykreslení daných dat [13]. Následuje popis jednotlivých vrstev a jejich společné interakce.

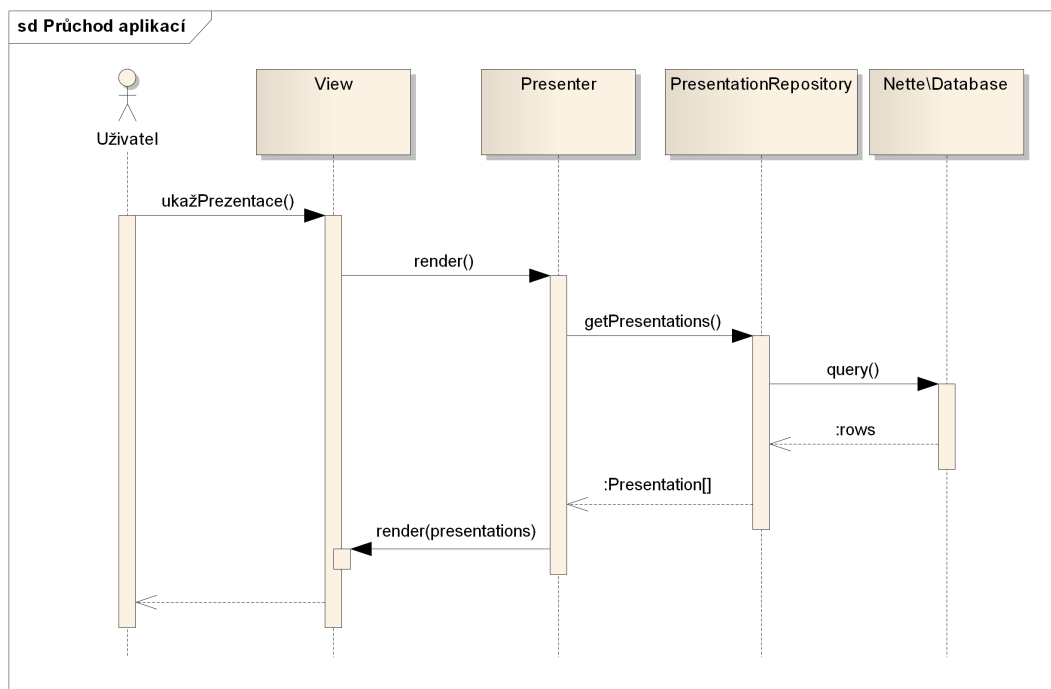
- **Model** – poskytuje rozhraní pro data určená k zobrazení
- **View** (pohled) – vykresluje data modelu a směřuje uživatelské příkazy presenteru
- **Presenter** – prostředník mezi modelem a pohledem. Získává data z modelu a upravuje je pro zobrazení v pohledu. Řeší události vyvolané uživatelem.

### 3.1.3 Architektura aplikační logiky

Nette Framework ovšem neposkytuje žádné vlastní řešení modelové vrstvy a nechává tak programátorovi volné ruce na využití jiných knihoven nebo vlastních řešení zaměřujících se na obchodní a datovou logiku. Do presenteru je pak možné tyto služby předat pomocí návrhového vzoru Dependency injection [12].

Pro tento projekt byla využita databázová vrstva Nette\Database, která poskytuje jednoduché rozhraní pro získávání dat z databáze. Komunikace s databází pak byla zabalena do repositářů, jenž poskytují metody pro navrácení jednotlivých entit a tvorbu a mazání záznamů. V případech, kdy je logika složitější a repositáře tuto funkčnost nemohou vykonat, je přidána další vrstva – fasáda.

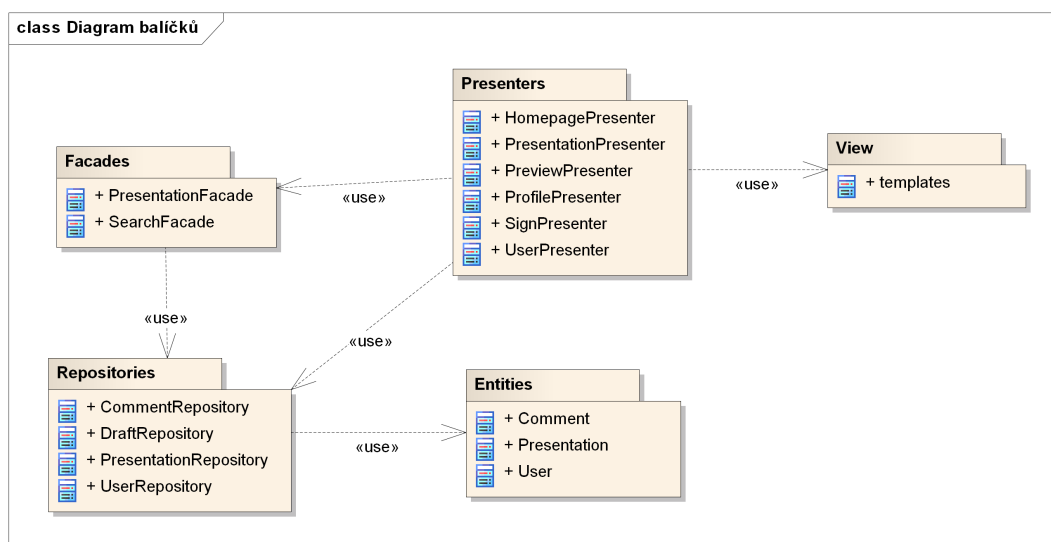
Lepší představu o průběhu chodu aplikace při uživatelském požadavku získáte z obrázku 3.1. Všechny komponenty serverové části pak přehledně zobrazuje diagram balíčků 3.2.



Obrázek 3.1: Ukázka průchodu aplikace

## 3.2 Klientská část

Na klientské prostředí jsou přenášeny výstupy v podobě HTML stránek. Pro jejich oživení a zlepšení uživatelské použitelnosti (např. klientské validace formulářů) se používá JavaScript. Vlastní kód byl však napsán v CoffeeScriptu [7] – transkompilátoru, který se kompiluje do běžného JavaScriptu.



Obrázek 3.2: Diagram balíčků

Dále byla použita JavaScriptová knihovna jQuery [17], která usnadňuje práci s HTML dokumentem a sjednocuje implementační rozdíly mezi prohlížeči.

Pro vzhled a grafické uživatelské rozhraní byl použit framework Bootstrap [5], který velmi usnadnil prototypování uživatelského rozhraní aplikace bez nutnosti tvořit vzhled a ladit ho pro každý prohlížeč zvlášť. Dále poskytuje jednoduché metody pro tvorbu nabídek či zobrazení dialogových oken. Výhodou je i to, že Bootstrap poskytuje responsivní design a tak by se web měl chovat a zobrazovat korektně i na mobilních zařízeních.

### 3.3 Editor

Editoru prezentací byl přikládán větší význam, neboť bez této komponenty by systém nemohl realizovat myšlenku tohoto projektu. Značně se také liší od ostatních obyčejných webových stránek, jelikož se jedná z větší části o ryze klientskou jednostránkovou mini-aplikaci.

Právě proto používá editor pár technologií navíc. Jako nejdůležitější se ukázalo použití JavaScriptového MVC frameworku AngularJS [2]. Jeho hlavní předností je dvoucestné provázání dat mezi pohledem (view) a modelem na straně klienta. Každá změna v modelu se pak automaticky propaguje do pohledu a naopak. Snižuje se tak množství kódu, který otrocky nastavuje nebo zobrazuje data při každé změně.

Pro popis obsahu snímku, jak bylo zmíněno v analýze, se používá jednoduchý textový značkovací jazyk, který převádí text do HTML. Pro převod tohoto textu je použita PHP knihovna Taxy! [34], která implementuje vlastní stejnojmenný značkovací jazyk. Výhodou této knihovny je snadná rozšiřitelnost a také typografické úpravy výsledného textu. Více o výběru knihovny se dozvíte v následující kapitole.

Součástí editoru je také panel nástrojů, který doplňuje textový vstup pro popis snímku. Uživateli ulehčí práci s psaním zdrojového textu, například možností vložit značky pro tučné písmo, kurzívu a další. Pro panel nástrojů byla použita knihovna Texyla [35].

## Kapitola 4

# Realizace klientské části

V této kapitole se dozvíte o průběhu realizace klientské části aplikace, tj. editoru prezentací a zobrazení prezentací, co vedlo k rozhodnutím o podobě projektu a jaké problémy vyvstaly a jak byly řešeny.

### 4.1 Prototyp editoru prezentací

V první fázi vývoje jsem se zabýval myšlenkou editoru prezentace jako hlavní komponenty, bez které by projekt neplnil požadovanou funkci. Následující text popisuje průběh práce na prototypu editoru prezentace.

#### 4.1.1 WYSIWYG editor

Při prvním prototypování editoru snímků prezentace se počítalo s použitím WYSIWYG editoru, tedy editoru, kde jsou změny vidět v reálném čase a uživatel při samotné úpravě snímků okamžitě vidí, jak bude vypadat konečný výsledek. Výhodou je, že uživatel nepotřebuje znát ani jazyk HTML, i když knihovny často poskytují i možnost kód HTML upravovat. Existuje mnoho knihoven, které WYSIWYG editor implementují. Protože ale neexistuje žádný standard, každý prohlížeč se chová trochu jinak a zároveň každá knihovna si různé věci implementuje po svém.

Nejdříve byla otestována WYSIWYG knihovna Aloha Editor [1], která se nejčastěji používá v CMS systémech. Nabízí několik zajímavých vlastností jako je např. repositář – prohlížeč obrázků, souborů či vlastních uživatelských objektů. Potíže ovšem nastaly se zobrazením panelu nástrojů a špatnou podporou CSS3 transformací, které jsou použity ve vzhledu deck.js prezentací.

Jako další editor byl vyzkoušen Mercury Editor [22], který se zdál být vhodnější a komplexnější. I když se jedná o JavaScriptovou knihovnu, je vyvíjena v Ruby pro použití v Ruby On Rails aplikacích a tak bylo hodně věcí přizpůsobeno těmto technologiím (například počítala s partial pohledy, které se posílají pomocí technologie AJAX ze serveru). Nicméně knihovna není na jazyce Ruby závislá a lze ji použít bez něj.

Zde se ale po chvíli objevily problémy technického rázu – prohlížeče se právě díky neexistujícímu standardu chovaly rozdílně nebo se nechovaly podle očekávání. Výsledkem pak byl

například špatný HTML kód nebo kód, který se už nedal nijak pomocí editoru opravit ani smazat. Jedním z největších problémů bylo vkládání nového řádku. Prohlížeč někdy vložil značku odřádkování `<br>` a někdy nový odstavec `<p>`. Chování tak nebylo předvídatelné.

Objevily se i další chyby v použitelnosti či bugy, na které jsem se pak snažil upozornit<sup>1</sup> autora knihovny.

#### 4.1.2 Editor založený na jednoduchém značkovacím jazyce

Všechny tyto problémy nakonec vyústily k rozhodnutí použít značkovací jazyk, tzv. light-weight markup language, který převádí text do HTML. V poslední době získávají tyto jazyky velkou popularitu kvůli jednoduchosti a velmi dobré podobě výsledného HTML kódu.

Asi nejvíce používaným jazykem, hlavně ve světě IT, je Markdown [20], který byl původně napsán v jazyce Perl. Vzniklo mnoho portů této knihovny do různých jazyků, včetně PHP. Port knihovny pro PHP je ovšem velmi komplexní a zdálo se nemožné ho upravit pro potřeby implementace dalších požadovaných funkcí do prezentace.

Proto nakonec bylo rozhodnuto využít jazyk Taxy! [34], který je jazyku Markdown podobný, ovšem syntaxe se občas liší. Knihovna Taxy! je ovšem velmi dobře rozdělená na logické jednotky – moduly. Jednotlivé konverze (např. nadpis nebo tučné písmo) se pak provádí ve funkcích těchto modulů. Za největší výhodu ale považuji rozšiřitelnost a upravitelnost knihovny. Moduly lze konfigurovat či vypínat a nechybí ani možnost definovat vlastní konverze. Lze se tak snadněji přiblížit známé syntaxi jazyku Markdown, to ovšem nebylo primárním cílem tohoto projektu.

Výhodou je také možnost specifikovat CSS třídy nebo určit zarovnání textu či obrázku. To je vlastnost, kterou jazyk Markdown nepodporuje. Značným kladem knihovny Taxy! je také vynikající podpora typografie. Například sekvenci znaků „+-“ nahradí za jediný znak „±“, jednoduchou pomlčku nahrazuje za spojovník (pokud je to ve větě potřeba), tři tečky nahradí za jediný znak trojtečky, dále vkládá nedělitelné mezery za spojky atp. V dokumentech jako je webová stránka nebo prezentace se tato vlastnost velmi hodí.

## 4.2 Rozložení editoru

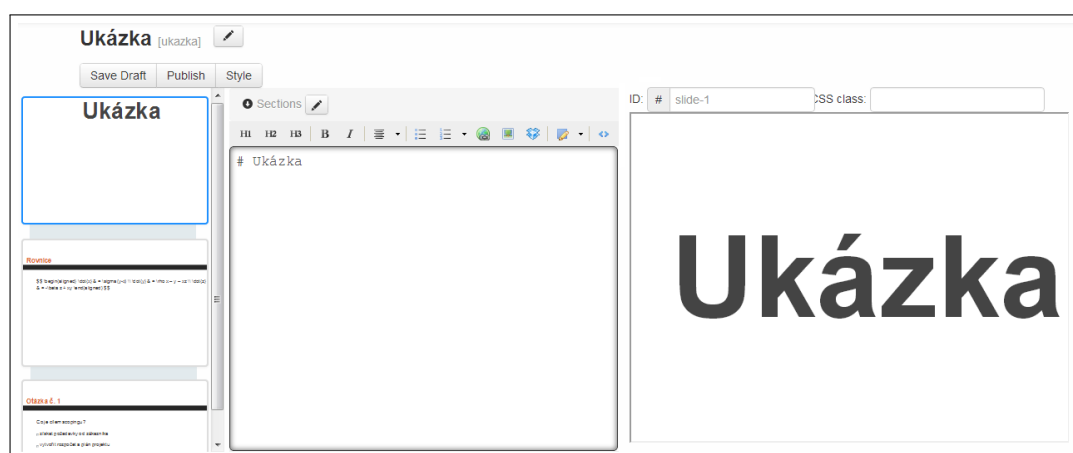
Dalším úkolem bylo navrhnout prostředí editoru. Rozložení bylo provedeno v podobném stylu jako u známých nástrojů pro tvorbu prezentací Microsoft PowerPoint a OpenOffice Impress, tedy obrazovka byla rozdělena na sloupec náhledů všech snímků po levé straně obrazovky a na editaci právě upravovaného snímku vpravo.

Protože se ale pro popis obsahu snímku používá značkovací jazyk namísto WYSIWYG editoru, bylo uživatelsky příjemnější přidat také náhled výsledného snímku po převodu zdrojového textu do HTML. Tímto se obrazovka rozdělila ještě na vstupní textové pole pro zdrojový text a velký náhled právě upravovaného snímku. Textové pole má navíc panel nástrojů, který umožňuje vkládat značky (například nadpis, nebo kurzívu) pomocí tlačítek.

---

<sup>1</sup>viz [<https://github.com/jejackson/mercury/issues/253>](https://github.com/jejackson/mercury/issues/253)





Obrázek 4.1: Ukázka editoru (vlevo náhledy všech snímků, uprostřed pole zdrojového textu, vpravo velký náhled upravovaného snímku)

## 4.3 Chování editoru

V následujícím textu je popsáno, jak editor prezentace funguje.

Při načtení stránky editoru se obsah prezentace převede na JavaScriptový objekt, který uchovává informace o jednotlivých snímcích. Součástí těchto dat je zdrojový text obsahu snímku a také jeho HTML podoba po konverzi. HTML kód každého snímku se pak vloží do panelu s malými náhledy. Malý náhled snímku je pomocí CSS pravidel zmenšen, jinak by byl obsah stejně velký jako v normálním zobrazení prezentace.

Po kliknutí na malý náhled se objeví v textovém poli pro úpravu snímku zdrojový text snímku a ve velkém náhledu vpravo se objeví kompletní snímek s inicializovanou knihovnou deck.js, tj. včetně funkční interakce.

Při změně zdrojového textu snímku se celý obsah automaticky posílá na server, kde se text pomocí knihovny Taxy! převádí na HTML kód. Výsledný HTML kód snímku se následně vrací v odpovědi. Poté se aktualizují oba náhledy snímku, malý i velký. Aby se ale na server neposílala každá úprava snímku (např. po stisku každého písmene), a tak nevhodně nezatěžovala server i klientskou aplikaci, byla implementovaná čekací doba 1 sekundy. Pokud po poslední změně snímku uplyne tato doba, během které nebyla provedena žádná další úprava, pošle se požadavek na převod zdrojového textu na server.

## 4.4 Úprava knihovny Taxy!

Následující odstavce popisují rozšíření jednoduchého značkovacího jazyka Taxy!. Bylo totiž nutné vymyslet a implementovat novou syntaxi pro interaktivní odpovídání na kvizové otázky během prezentace a další novinky. Dále jsem se pokusil upravit syntaxi značkovacího jazyka Taxy! pro lepší kompatibilitu s více rozšířeným a známým jazykem Markdown.

#### 4.4.1 Markdown syntaxe

Jak už bylo vysvětleno v předchozích kapitolách, pro popis snímků byla použita knihovna Taxy!, která implementuje vlastní jazyk stejného jména. Tento jazyk je ovšem široce používaný pouze v českých a slovenských zemích. Celosvětově ho zastiňuje mnohem více rozšířený značkovací jazyk Markdown. Oba jazyky jsou si v lecčem podobné, avšak najdou se i velké rozdíly. Proto jsem se rozhodl knihovnu Taxy! upravit a některé části přizpůsobit syntaxi jazyku Markdown.

Jeden z nejvíce patrných rozdílů byl v syntaxi nadpisů. Porovnejte rozdíly mezi výpisy 4.1 a 4.2.

```
### Nadpis 1. úrovně
## Nadpis 2. úrovně
# Nadpis 3. úrovně

nebo alternativně

Nadpis 1. úrovně
#####

Nadpis 2. úrovně
*****

Nadpis 3. úrovně
=====
```

Výpis 4.1: Ukázka nadpisů v Taxy!

```
# Nadpis 1. úrovně
## Nadpis 2. úrovně
### Nadpis 3. úrovně

nebo alternativně

Nadpis 1. úrovně
=====

Nadpis 2. úrovně
-----
```

Výpis 4.2: Ukázka nadpisů v Markdown

Jak je vidět, logika syntaxí značek u varianty před nadpisem je u obou jazyků přesně opačná. Proto byla syntaxe upravena, aby byla ekvivalentní s jazykem Markdown. Změna pak byla otestována pomocí jednotkových testů.

Protože sjednocení syntaxí nebylo prioritou tohoto projektu, nezaměřoval jsem se více na další zvýšení kompatibility.

### 4.4.2 Nová syntaxe

Dále bylo nutné rozšířit knihovnu Taxy! a implementovat vlastní pravidla konverze textu do HTML. Rozšíření se týkalo možnosti uživatele odpovídat na otázky během prohlížení prezentace a dále vytvoření osnovy prezentace s odkazy na snímky.

#### 4.4.2.1 Syntaxe odpovědí

Pro vytvoření kvizových otázek bylo potřeba vymyslet a implementovat syntaxi na vložení zaškrťovacího pole, které uživatel při prohlížení prezentace může zaškrtnout jako správnou odpověď na otázku položenou na snímku. Správných odpovědí může být více. Po delší úvaze bylo rozhodnuto, že na jednom snímku lze mít pouze jednu otázku. Nejenže by se více otázek na snímek nevešlo, ale zjednodušila se tak i složitost syntaxe, která by na tento případ musela brát ohled. Stejně tak odpadla potřeba manuálně vkládat potvrzovací tlačítko pro kontrolu zaškrtnutých odpovědí. Tlačítko se při překladau vkládá na konec snímku automaticky, pokud je na snímku použita syntaxe pro vložení zaškrťovacích políček. Ukázku syntaxe můžete vidět ve výpisu 4.3 a snímek po převodu do HTML na obrázku 4.2.

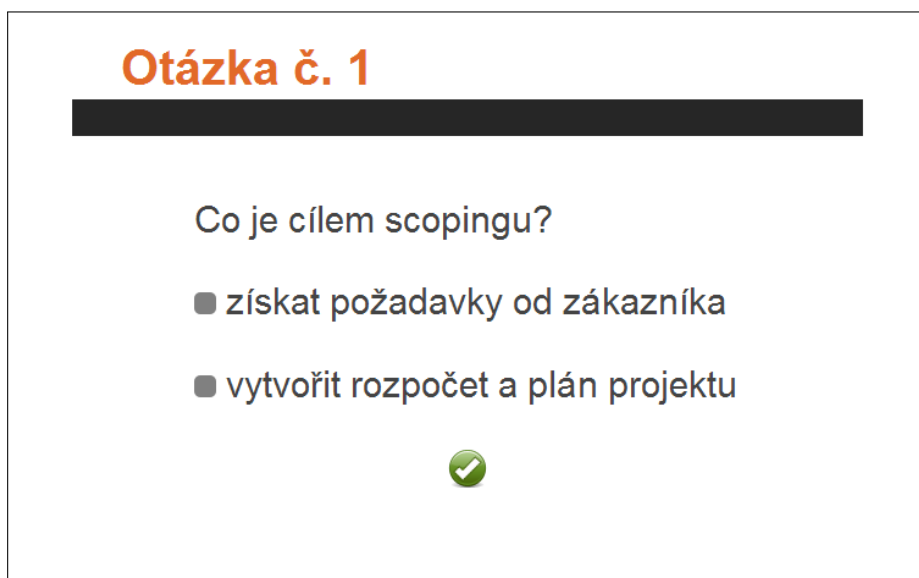
```
## Otázka č. 1

Co je cílem scopingu?

[+] získat požadavky od zákazníka

[-] vytvořit rozpočet a plán projektu
```

Výpis 4.3: Ukázka zdrojového textu snímku s otázkou a odpověďmi



Obrázek 4.2: Ukázka snímku s otázkou a odpověďmi

Pro vyhodnocení správnosti zaškrtnutých odpovědí musí uživatel stisknout tlačítko pro kontrolu. Případné chybné odpovědi mají za následek přiřazení CSS třídy ke špatně zaškrtnutým políčkům.

Samotná zaškrťovací políčka nejsou zobrazena jako HTML vstupní pole `<input>` proto, aby se daly lépe vzhledově upravit pomocí CSS stylů.

#### 4.4.2.2 Syntaxe doplňkových textů

Důležitou součástí požadovaného kvizového systému byla také možnost vkládat doplňkový text. Autor prezentace tak může uvést bližší informace k odpovědím na danou otázku nebo odkázat na jiný snímek prezentace či dokonce na jiný web.

Tento text se rozděluje na dva typy: pro správně a nesprávně zaškrtnuté odpovědi. Tento doplňkový text se dynamicky objeví až po kliknutí na potvrzovací tlačítko, přitom text pro správně zaškrtnuté odpovědi se zobrazí při zaškrtnutí pouze správných odpovědí, u textu pro špatné odpovědi přesně naopak.

Pro vložení takového bloku textu byla použita podobná syntaxe jako pro ostatní účelové bloky textu v syntaxi Texy! (např. blok kódu). Použití syntaxe pro oba doplňkové texty vidíte ve výpisu 4.4.

```
/--correct
Tento text se zobrazí při **správné** odpovědi.
\--

/--incorrect
Tento text se zobrazí při **špatné** odpovědi.
\--
```

Výpis 4.4: Syntaxe doplňkových textů

#### 4.4.2.3 Syntaxe pro vložení osnovy

V mnoha prezentacích se můžeme setkat s více tématy nebo podtématy, o kterých prezentace pojednává. Proto editor prezentace poskytuje možnost rozdělit prezentaci na logické celky. U každého snímku tak lze specifikovat až tři úrovně sekcí, do kterých snímek logicky patří. Toho pak bylo využito pro další vlastnost převodu zdrojového textu do HTML – generování osnovy (obsahu) prezentace s odkazy na snímky, které tyto sekce specifikují. Syntaxe je pak jednoduchá, stačí uvést `{table of contents}`. Při převodu je pak tato direktiva nahrazena HTML seznamem sekcí a případných podsekcí s odkazy vedoucími na dané snímky.

### 4.5 Rozšíření prezentační knihovny

Protože tento nástroj byl zaměřen na použití ve vzdělávacích institucích, bylo dobrou myšlenkou rozšířit možnosti prezentační knihovny o další pomocné funkce. Rozhodl jsem se proto implementovat funkci zvýraznění syntaxe bloků programovacích jazyků, která se určitě bude hodit prezentujícím vývojářům, a dále zobrazení matematických vzorců ve formátu L<sup>A</sup>T<sub>E</sub>X, které bude možné použít například v prezentacích z oboru matematiky.

### 4.5.1 Zvýraznění syntaxe programovacích jazyků

Pro zobrazení kódu používá knihovna Taxy! blokovou syntaxi jakou můžete vidět ve výpisu 4.5. Po převodu tohoto textu do HTML se uvedený kus kódu obalí do sebe vnořených sémantických HTML značek `<pre>` a `<code>`. Toho je pak využito při zvýraznění syntaxe, jež je zajištěno pomocí knihovny highlight.js [15], která automaticky tyto bloky kódu za použití detekce jazyka zvýrazní. Knihovna podporuje více než 50 jazyků.

```
--code
SELECT * FROM 'presentations' where 'id' = 1;
--
```

Výpis 4.5: Ukázka blokové syntaxe pro výpis kódu

### 4.5.2 Matematické vzorce

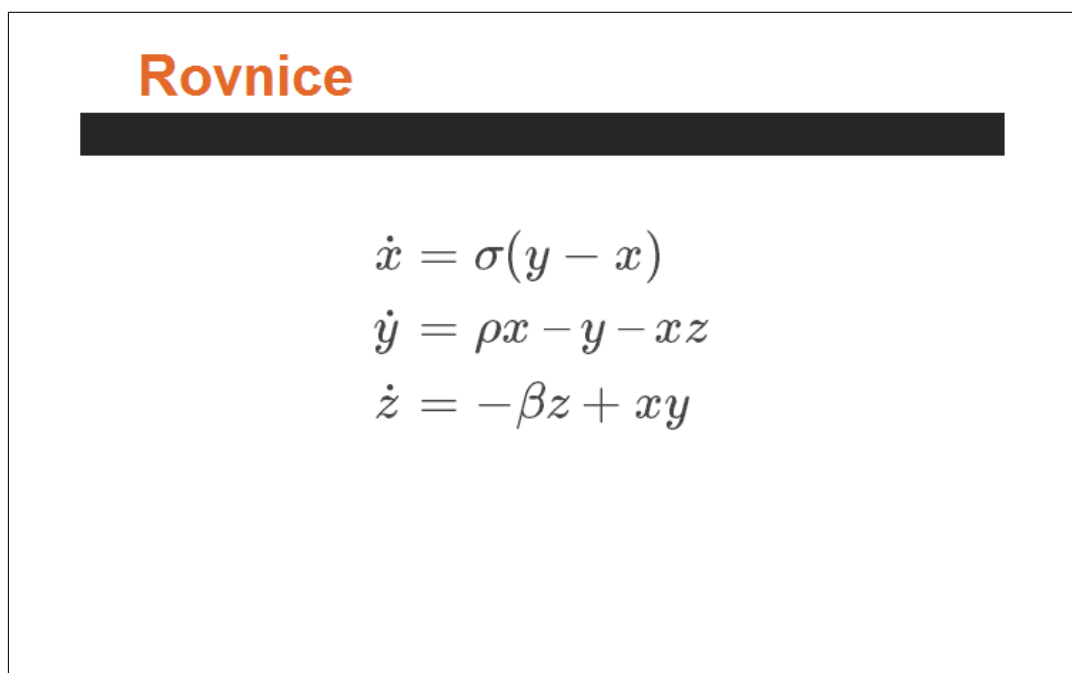
Další přidanou hodnotou do prohlížení prezentací bylo zavedení zobrazení matematických vzorců a rovnic. Není totiž nijak uživatelsky příjemné tyto vzorce vytvářet jako obrázky a vkládat je pak do prezentace. Pro implementaci této vlastnosti byla použita JavaScriptová knihovna MathJax [21], která automaticky vyhledává zdrojový text ve formátu L<sup>A</sup>T<sub>E</sub>X [18] ve specifické syntaxi a následně je vysází pomocí HTML, CSS a webových fontů. Není tedy potřeba žádných dalších technologií třetích stran, jako například Flash, a zároveň se nejedná o obrázky. Vzorce si tak zachovávají všechny vlastnosti okolního textu, jako jsou barva či velikost.

Podmínkou pro vyhledání a zobrazení vzorců je ohraničení zdrojového textu vzorců oddělovači. Ty jsou dvojího druhu, protože knihovna podporuje zobrazení jak blokových vzorců, tak tzv. inline vzorců, které mohou být součástí odstavce textu. Oddělovače pro inline vzorce jsou  $( a )$ , pro blokové vzorce pak  $\$$  a  $\$$ . Ukázku použití této syntaxe spolu se vzorovým zdrojovým textem několika rovnic vidíte ve výpisu 4.6, výsledný snímek s vysázenými matematickými rovnicemi vidíte na obrázku 4.3.

```
## Rovnice

$$
\begin{aligned}
\dot{x} &= \sigma(y-x) \\
\dot{y} &= \rho x - y - xz \\
\dot{z} &= -\beta z + xy
\end{aligned}
$$
```

Výpis 4.6: Ukázka syntaxe matematických vzorců



Obrázek 4.3: Ukázka snímku s matematickými rovnicemi

# Kapitola 5

## Nasazení

V této kapitole jsou shrnuty požadavky na nasazení aplikace a dále je vylíčen průběh nasazení projektu na produkční server.

### 5.1 Požadavky

Pro chod systému je nutný spuštěný webový server, na kterém běží PHP. Dále je pak potřeba MySQL databáze. Komponenty a jejich verze, na kterých byl projekt vyvíjen, jsou uvedeny níže.

- Apache 2.2
- PHP 5.3
- MySQL 5.5

Pro zaručení správného chodu aplikace je také dobré ověřit, zda-li server splňuje požadavky samotného Nette Frameworku. K tomu lze využít PHP skript jménem Requirements Checker, jenž je frameworkem poskytován. Více o skriptu, jeho instalaci a spuštění naleznete v dokumentaci Nette Frameworku [6].

### 5.2 Realizace nasazení

Pro nasazení na produkční prostředí jsem se nejdříve rozhodl využít cloudového systému PHP Fog. Pro malý počet aplikací a malý rozsah poskytoval své služby zdarma. V placených řešeních bylo pak největší výhodou možnost škálování instancí aplikace či operační paměti nebo velké množství rozšíření, např. analytické nástroje či optimalizační nástroje pro databázi. Server uchovával aplikaci jako Git repositář a tak pro nahrání nové verze aplikace stačil příkaz `git push`.

V listopadu 2012 bylo ovšem oznámeno ukončení služby PHP Fog a postupné odpojování bezplatných aplikací v průběhu prosince 2012. Jako náhrada byla doporučena podobná služba AppFog [3] od stejné společnosti. [26]

AppFog, podobně jako PHP Fog, je cloudová služba poskytující Platform as a Service (PaaS), tedy výpočetní platformu a k tomu funkční řešení operačního systému, webového serveru, databáze a dalších. Aplikaci tak stačí pouze nasadit a spustit. AppFog na rozdíl od PHP Fogu podporuje kromě jazyka PHP i další, např. Node, Ruby či Javu. Změněn byl i způsob nahrávání aplikace, z Git repositáře se přešlo na vlastní konzolový nástroj, který dále nabízí i komplexní správu nasazených aplikací.

Nevýhodou této služby je v současné době neexistující persistentní úložiště souborů, to znamená, že každá změna provedená na souborech na serveru, např. uložení obrázků nahraných uživateli aplikace, bude při nahrání nové verze aplikace nenávratně ztracena (databázi se to ovšem netýká). Tento problém měl být vyřešen ve stejné době jako ukončení provozu PHP Fogu, avšak ani v době dokončování psaní této práce nebyla situace vyřešena. Pro vložení obrázků do prezentací si tak uživatelé prozatím musí vystačit s jinými službami pro sdílení obrázků. Samotný editor ale nabízí vložení souborů ze služby Dropbox pomocí prohlížeče Dropbox Chooser [10].

### 5.3 Deployment

Projekt byl průběžně vyvíjen na lokálním serveru a při větším počtu změn byl aktualizován produkční server. Pro řešení závislostí použitých PHP knihoven jsem přidal balíčkovací systém Composer, který kromě závislostí obstará i stažení a aktualizaci těchto knihoven. Knihovny tak nejsou součástí projektu a při nasazení je potřeba je příkazem `install` do aplikace stáhnout a nainstalovat. Více se dočtete v dokumentaci Composeru [8].

Další významnou součástí aktualizace systému byla migrace databáze při její změně. K tomu byla použita knihovna Phinx, kterou jsem upravil, aby používala konfigurační soubor Nette Frameworku. Při změně schématu databáze je pak nutné příkazem vytvořit PHP soubor ve složce `migrate`, který obsahuje metodu pro provedení změny v databázi a také metodu pro navrácení do původního stavu. Při aktualizaci projektu je pak nutné provést příkaz `migrate`, který dosud neprovedené změny databáze provede. Více informací najdete v dokumentaci knihovny Phinx [25].



## Kapitola 6

# Testování

Testování je důležitou součástí vývoje každého softwarového systému. Jedná se o takové zkoušení systému, které nalezne nejen chyby (tzv. buggy) v softwaru, ale ověřuje také, zda-li vyhovuje zadaným požadavkům. Před několika lety, často se s tím ale můžeme setkat i dnes, se testování provádělo ručně. Zvláště v jazyce PHP se často při každé změně ručně obnovuje stránka v prohlížeči a následně se kontrolují výstupy.

V současné době se software nejčastěji testuje pomocí automatických nástrojů. To znamená, že testy definované nějakým scénářem jsou spouštěny všechny najednou a výsledkem jsou zprávy, zda všechny testy úspěšně prošly. Pokud některé testy neprošly, snadno zjistíme příčinu. Při každé změně softwaru tak můžeme rychle zjistit, zda-li systém vyhovuje požadavkům a zda neobsahuje chyby. Protože máme možnost spouštět všechny testy, můžeme mít (větší) jistotu, že změna nerozbila jiné části systému, které na první pohled se změnou nesouvisí.

### 6.1 Test Driven Development

Test Driven Development (TDD) [4] neboli vývoj řízený testy je metodika vývoje softwaru, kde se test píše dříve než kód, který má daný test ověřit. Použití této metodiky má několik výhod – test je psán bez znalosti vnitřní implementace ověřovaného kódu a je tak pro programátora těžší být ovlivněn vlastní implementací. Vedlejším efektem je pak i lepší návrh aplikace. Protože v testech se snažíme psát co nejméně kódu, programátor se pak snaží testovaný kód mít co nejjednodušší pro nastavení a inicializaci, což prospívá i jednoduchosti samotné implementace.

Tento projekt je prvním, kdy jsem se snažil používat metodiku TDD. Ne vždy se to však povedlo, neboť jsem byl zvyklý na opačný postup. Nejvíce jsem jej využil při úpravě a implementaci nových syntaxí v knihovně Taxy!, kdy jsem si pro požadované výstupy knihovny napsal test předem a poté dané změny syntaxe implementoval. Dále pak bylo TDD použito při refaktorování back-endu aplikace.

Metodika TDD se velmi dobře aplikuje a při vývoji mi pomáhala, avšak zvyklost byla často silnější. V dalších projektech se budu snažit ji používat důkladněji.

## 6.2 Jednotkové testy

Jednotkové nebo také unit testy jsou testy, které ověřují správnost jedné malé části softwaru, tzv. jednotky. Typicky jsou to metody nebo třídy. Jejich úkolem je zjistit správnost funkčnosti této jednotky nehlédě na vliv ostatních jednotek, např. ostatních tříd či metod nebo úplně jiných systémů.

V tomto projektu byly unit testy psány hlavně při úpravě syntaxe knihovny Taxy!. Bylo nutné, aby nové prvky syntaxe nebyly v konfliktu s již existujícími, což však bylo znemožněno neexistujícími testy v samotné knihovně Taxy!. Testy tak sloužily jen na ověření výstupu nové či upravené syntaxe.

Protože jsou výstupy testů syntaxe často dlouhé a ve zdrojovém souboru by byly nepřehledné, byly jak vstupy tak očekávané výstupy jednotlivých testů přesunuty do textových souborů. Vstupní soubory se pak dávkově nahrávaly do testu a výstup se kontroloval oproti souborům s očekávanými výstupy.

Unit testy byly napsány s pomocí PHP frameworku pro jednotkové testování PHPUnit [27].

## 6.3 Integrační testy

Integrační testy ověřují komunikaci mezi více jednotkami, testovaný kód je oproti jednotkovým testům již závislý i na ostatních třídách, komponentách, systémech nebo dokonce platformě či hardwaru.

Tímto způsobem byla v projektu testovaná modelová vrstva aplikace, tedy entity, repository a fasády. Testované třídy používají funkční MySQL databázi, která obsahuje testovací data. Při každém jednotlivém testu se databáze smaže a vytvoří znovu podle SQL souboru obsahující definice tabulek a data ve výchozím stavu. Je tak vyřešena jakákoliv nekonzistence mezi jednotlivými testy a není potřeba po každém testu ručně navracet databázi do výchozího stavu. Stejně jako u jednotkových testů byl použit framework PHPUnit. Ukázky integračních testů naleznete ve výpisu B.1 v příloze B.

Tabulka 6.1 ukazuje počet napsaných a spuštěných testů, jejich úspěšnost a pokrytí kódu. Pokrytí ukazuje podíl kódu aplikace, jenž byl testy vykonán.

Počet testů	57
Počet úspěšných testů	57
Pokrytí kódu	52 %

Tabulka 6.1: Přehled o testech

## 6.4 Selenium testy

Pro testování uživatelského rozhraní byl použit systém Selenium [32]. Jeho hlavním úkolem je automatizace webových prohlížečů, nejčastěji se ale používá právě jako testovací nástroj.

Selenium poskytuje rozhraní pro komunikaci s různými prohlížeči různých verzí. Testy se skládají ze scénářů, tedy předem definovaných kroků prohlížeče (např. kliknutí na odkaz nebo

vyplnění formulářového pole) a díky komunikačnímu rozhraní lze také z prohlížeče získávat informace (např. zda-li se na stránce objevuje nějaké slovní spojení). Tyto scénáře můžou být interaktivně vytvořeny pomocí Selenium IDE (pouze pro prohlížeč Firefox) a pak je lze opakovaně spouštět. Další možností je použít Selenium Remote Control (RC), který dokáže prohlížeč spustit a vykonat scénáře definované v některém z podporovaných programovacích jazyků.

Ovladač pro Remote Control jazyk PHP podporuje a zmíněný testovací framework PHPUnit poskytuje rozšíření implementující klient-server protokol pro komunikaci s RC. Navíc také nabízí specializované metody pro testování. Ve výpisu [B.2](#) v příloze [B](#) se můžete podívat, jak takový test vypadá.

## 6.5 Zátěžové testy

Zátěžové testy prověřují chování aplikace pod narůstajícím počtem požadavků. Je tak snadné zjistit, zda-li je aplikace dostatečně rychlá a zda-li ustojí větší počet aktivních uživatelů. Pro otestování tohoto projektu nasazeném na AppFog jsem se rozhodl využít webový nástroj Load Impact [\[19\]](#), jenž zdarma poskytuje zátěžové testy s maximálně 50 souběžnými virtuálními uživateli.

Zdarma ale služba nenabízí výběr místa odkud požadavky přicházejí. To se pak podepsalo na delší odezvě serveru, neboť požadavky většinou nepřicházely z Evropy, kde se aplikační server nachází. Zvyšující se zátěž ale nepředstavovala pro aplikaci větší problém a i při plném vytížení, jenž LoadImpact poskytuje, byla doba načtení konstantní.

Pro testování byly vybrány dvě stránky, které představují pro aplikaci velkou zátěž a to jak na straně PHP tak i na straně databáze. Občas se vyskytly mírné fluktuace, které jsou nejspíš vinou bezplatného účtu služby AppFog, neboť u následujícího vyššího zatížení se tyto výkyvy neprojevily. Grafy výsledků obou testů najdete v příloze [C](#).



## Kapitola 7

# Zhodnocení a budoucí vývoj

Práce na tomto projektu byla velmi zajímavá. Vyzkoušel jsem si řadu nových technologií, zejména mě zaujal AngularJS, který určitě budu dále sledovat a předpokládám, že ho využiji i budoucnu. Kromě AngularJS jsem si poprvé vyzkoušel i migrace databází v PHP a i když to není tak automatické jako třeba v Ruby on Rails, pracuje se s knihovnou velmi dobře.

Výbornou zkušeností bylo také nasazení projektu na cloudové PaaS systémy. Tyto služby jsou nejen jednoduché na použití, ale poskytují široké možnosti správy a škálování aplikací. Platforma PHP ovšem není připravena na více instancí aplikace a je proto nutné s řešením počítat již při vývoji aplikace. Placená řešení cloudových systémů ovšem mnohánásobně převyšují výhody vlastního serveru nebo běžných webhostingových služeb.

S projektem jsem celkově spokojený, i když jsem zprvu čekal mnohem rychlejší vývoj, některé doplňkové funkce, tzv. nice-to-have, tak kvůli nedostatku času či kvůli řešení jiných problémů nebyly implementovány. Problémy nastaly u prezentační knihovny a vzhledu prezentace, které zpomalovaly vývoj. Dále by bylo dobré zapracovat na zlepšení uživatelské přívětivosti a mobilní verzi.

Při práci na projektu jsem ale zjistil, že se v době responzivních designů a dalších novinek v CSS3 začínám ztrácet a řešení vzhledu webů a User experience (UX) nestíhám sledovat. Proto bych se v dalších projektech více zaměřil na vývoji back-endu aplikací a přenechal tyto problémy zkušenějším webovým designerům.

Jsem rád, že jsem použil Nette Framework, který dobře znám a tak jsem neměl tolik problémů při vývoji serverové části. Ze začátku jsem se ovšem rozhodoval, jestli si nevyzkoušet nějakou jinou technologii, například dnes populární Ruby on Rails nebo Node.js. Při výběru jsem bral v úvahu hlavně malé zkušenosti s těmito technologiemi a obával jsem se možných problémů a mnohem pomalejšího vývoje při jejich řešení. Na druhou stranu jsem se ochudil o možnost si technologie vyzkoušet a zjistit pro a proti. Při práci na tomto projektu jsem zjistil, že PHP má jednu nevýhodu, a to v nedokonalém ekosystému. Naproti tomu Ruby on Rails dokáže pár příkazy stáhnout, nainstalovat a nakonfigurovat potřebné knihovny, kterých je navíc velké množství.

### 7.1 Budoucí vývoj

Projekt je sice v použitelném stavu, ale rozhodně není dokonalý a chybí jisté funkce a vylepšení, které by uživatelé ocenili. Následující odstavce popisují nejdůležitější body mojí vize dalšího vývoje projektu. Nejedná se pouze o zlepšení UX při prohlížení prezentací, ale také nové funkce, které zlepší organizaci, přehlednost a zpětnou vazbu uživateli.

#### 7.1.1 Náhledy prezentací

V současné chvíli jsou prezentace ve výpisech na webu reprezentovány pouze názvem prezentace. Chybí tak jakákoliv grafická vazba na prezentaci. Uživatelé se ale mnohem rychleji orientují podle obrázků než podle textů. Proto by bylo dobré ve výpisech zobrazovat i náhled prvního snímku prezentace.

#### 7.1.2 Notifikace

Jako další vylepšení by mohly sloužit notifikace (upozornění) uživatelům, ať už prostřednictvím webu nebo zasíláním e-mailů. Nyní se o nových prezentacích sledovaných uživatelů nebo nasdílení prezentace uživatel nedozví. Vidí je pouze ve výpisech na hlavní stránce webu, vizuálně ale uživatel není nijak upozorněn.

#### 7.1.3 Import a export prezentací

Pro uživatele by bylo výhodné přidat i podporu pro import a export z/do různých formátů. Import uživateli ušetří čas při sdílení existujících prezentací v jiných formátech. Export pak lze využít pro prohlížení prezentací bez připojení k internetu.

Při práci na tomto projektu jsem se pokusil o nástroj pro import z formátu Microsoft PowerPoint. Jednalo se ovšem jen o jednoúčelový nástroj pro několik prezentací, značně přizpůsobený jednomu typu prezentace. Importován byl pouze text každého snímku, tj. bez vzhledu či obrázků.

#### 7.1.4 Členění a seskupování prezentací

Prezentace se nyní dají vyhledávat jen pomocí názvu nebo podle autora. Další možností by bylo přidat k prezentacím krátké popisky, tzv. štítky. Vyhledávání by pak mohlo fungovat i podle těchto štítků. Také sdílení by šlo vylepšit tím, že by uživatel mohl sledovat jednotlivé štítky a být upozorněn v případě nové prezentace s daným štítkem.

Prezentace by si také zasloužily lepší organizaci. Často se stává, že jeden uživatel má více prezentací, které spolu souvisí (například prezentace z přednášek jednoho předmětu nebo kurzu). Strukturování prezentací do jakýchsi složek nebo seznamů by tak bylo užitečnou vlastností. Spolu s tím by bylo příhodné implementovat i sdílení těchto seznamů.

### 7.1.5 Bezpečnost

Důležitou částí dalšího vývoje by mělo být zaměření se na bezpečnost systému. V tuto chvíli mě znepokojuje existující možnost zanesení škodlivého JavaScriptového kódu do prezentace při úpravě vzhledu prezentace. V některých prohlížečích totiž lze v CSS spouštět JavaScriptový kód. Řešením by bylo CSS kód automaticky projít a všechny výskyty JavaScriptu odstranit. Není to ale tak jednoduché, například kvůli komentářům či dalším technikám obfuskace kódu.





## Kapitola 8

# Závěr

Výsledkem této bakalářské práce je funkční základ webové aplikace pro tvorbu a sdílení prezentací založených na technologii HTML. Navržený systém byl nasazen na produkčním serveru<sup>1</sup> pomocí cloudové služby AppFog. Požadovaná funkčnost byla analyzována, implementována a otestována, stále je ale prostor pro další vylepšení a rozšiřování systému.

Projektu se ale nevyhnuly určité problémy, ať už technologické či časové. Nedostalo se tak na spousty nápadů, které by systém posunuly kvalitativně dál, aby mohl konkurovat zavedeným službám. Přesto jsem ale s projektem a svou prací na něm spokojen. Vyzkoušel jsem si několik nových a zajímavých technologií, které mě obohatily a které určitě využiji i v budoucnu.

---

<sup>1</sup> Adresa aplikace je `<http://slide-it.eu01.aws.af.cm/>`



# Literatura

- [1] *Aloha Editor* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://www.aloha-editor.org/>>.
- [2] *AngularJS* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://angularjs.org/>>.
- [3] *AppFog* [online]. 2013. [cit. 2. 5. 2013]. Dostupné z: <<https://www.appfog.com/>>.
- [4] BECK, K. *Test-driven development : by example*. Boston: Addison-Wesley, 2003. ISBN 0-321-14653-0.
- [5] *Bootstrap* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://twitter.github.io/bootstrap/>>.
- [6] *Nette Requirements Checker* [online]. 2013. [cit. 2. 5. 2013]. Dostupné z: <<http://doc.nette.org/cs/requirements>>.
- [7] *CoffeeScript* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://coffeescript.org/>>.
- [8] *Composer* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <<http://getcomposer.org/>>.
- [9] *deck.js* [online]. 2013. [cit. 9. 5. 2013]. Dostupné z: <<http://imakewebthings.com/deck.js/>>.
- [10] *Dropbox - Dropbox Chooser* [online]. 2013. [cit. 9. 5. 2013]. Dostupné z: <<https://www.dropbox.com/developers/chooser>>.
- [11] FERREIRA, D. *Instant HTML5 Presentations How-to*. Packt Publishing, 2013. ISBN 1782164782.
- [12] FOWLER, M. *Inversion of Control Containers and the Dependency Injection pattern* [online]. 2004. [cit. 19. 4. 2013]. Dostupné z: <<http://martinfowler.com/articles/injection.html>>.
- [13] FOWLER, M. *GUI Architectures: Model-View-Presenter* [online]. 2006. [cit. 19. 4. 2013]. Dostupné z: <<http://martinfowler.com/eaaDev/uiArchs.html#Model-view-presentermvp>>.
- [14] *Google I/O HTML5 slide template* [online]. 2013. [cit. 9. 5. 2013]. Dostupné z: <<https://code.google.com/p/io-2012-slides/>>.

- [15] *Highlight.js* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://softwaremaniacs.org/soft/highlight/en/>>.
- [16] *impress.js* [online]. 2013. [cit. 9. 5. 2013]. Dostupné z: <<https://github.com/bartaz/impress.js>>.
- [17] *jQuery* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://jquery.com/>>.
- [18] *L<sup>A</sup>T<sub>E</sub>X* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://www.latex-project.org/>>.
- [19] *Load Impact* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <<http://loadimpact.com>>.
- [20] *Markdown* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://daringfireball.net/projects/markdown/>>.
- [21] *MathJax* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://www.mathjax.org/>>.
- [22] *Mercury Editor* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://jejackson.github.io/mercury/>>.
- [23] *MySQL* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://www.mysql.com/>>.
- [24] *Nette Framework* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://nette.org>>.
- [25] *Phinx* [online]. 2013. [cit. 6. 5. 2013]. Dostupné z: <<http://phinx.org/>>.
- [26] *Urgent News about PHP Fog* [online]. 2013. [cit. 5. 5. 2013]. Dostupné z: <<https://gist.github.com/anonymous/4067487>>.
- [27] *PHPUnit* [online]. 2013. [cit. 2. 5. 2013]. Dostupné z: <<https://github.com/sebastianbergmann/phpunit/>>.
- [28] Příspěvatelé Wikipedie. *List of web based slide shows* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <[http://en.wikipedia.org/wiki/Web-based\\_slideshow#List\\_of\\_web-based\\_slide\\_shows](http://en.wikipedia.org/wiki/Web-based_slideshow#List_of_web-based_slide_shows)>.
- [29] *Prezi* [online]. 2013. [cit. 9. 5. 2013]. Dostupné z: <<http://www.prezi.com>>.
- [30] *reveal.js* [online]. 2013. [cit. 9. 5. 2013]. Dostupné z: <<https://github.com/hakimel/reveal.js/>>.
- [31] *SlideShare* [online]. 2013. [cit. 9. 5. 2013]. Dostupné z: <<http://www.rvl.io>>.
- [32] *Selenium* [online]. 2013. [cit. 2. 5. 2013]. Dostupné z: <<http://docs.seleniumhq.org/>>.
- [33] *SlideShare* [online]. 2013. [cit. 9. 5. 2013]. Dostupné z: <<http://www.slideshare.net>>.
- [34] *Texy!* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<http://texy.info/>>.
- [35] *Texyla* [online]. 2013. [cit. 19. 4. 2013]. Dostupné z: <<https://github.com/janmarek/Texyla>>.

## Příloha A

# Seznam použitých zkratek

AJAX Asynchronous JavaScript and XML

CMS Content Management System

CSS Cascading Style Sheets

HTML HyperText Markup Language

MVC Model-View-Controller

MVP Model-View-Presenter

PaaS Platform as a Service

TDD Test Driven Development

UX User experience

WYSIWYG What You See Is What You Get



## Příloha B

# Ukázky kódu

Výpis B.1: Ukázka integračních testů

```
1 <?php
2
3 class PresentationRepositoryTest extends BaseDbTestCase {
4
5     /** @var PresentationRepository */
6     private $repository;
7
8     protected function setUp() {
9         parent::setUp();
10         $this->repository = new PresentationRepository($this->
            container->database);
11     }
12
13     public function testDeletePresentation() {
14         $pres = $this->repository->create(1, 'testFoo', 'testFoo');
15         $this->assertNotNull($this->repository->getById($pres->id));
16
17         $this->repository->delete($pres->id, $pres->author->id);
18
19         $this->assertNull($this->repository->getById($pres->id));
20     }
21
22     public function testChangeNameAndSlug() {
23         $pres1 = $this->repository->getById(1);
24         $this->repository->changeNameAndSlug($pres1, 'test_foo_name',
            , 'test_foo_slug');
25
26         $pres2 = $this->repository->getById(1);
27         $this->assertEquals('test_foo_name', $pres2->name);
28         $this->assertEquals('test-foo-slug', $pres2->slug);
```

```
29
30     $this->assertEquals($pres1->name, $pres2->name);
31     $this->assertEquals($pres1->slug, $pres2->slug);
32 }
33
34 public function testChangeNameAndSlugToExisting() {
35     $this->setExpectedException('DuplicateEntryException');
36
37     $pres = $this->repository->getById(1);
38     $this->repository->changeNameAndSlug($pres, 'testFooName', '
39     test2');
40 }
41 }
```



---

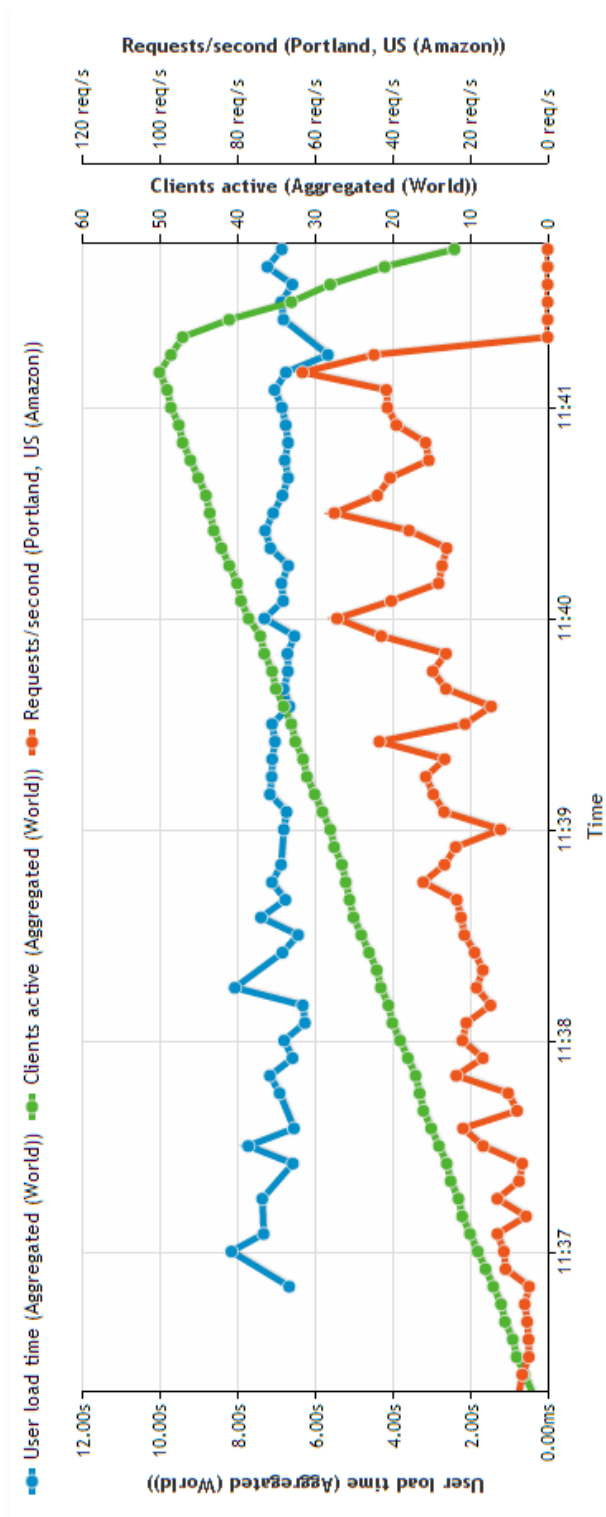
Výpis B.2: Ukázka Selenium testu

```
1 <?php
2
3 class SeleniumTest extends PHPUnit_Extensions_SeleniumTestCase {
4
5     public function setUp() {
6         parent::setUp();
7         $this->setBrowser("*firefox");
8         $this->setBrowserUrl($this->url);
9     }
10
11     private function loginTest() {
12         $this->open($this->url . '/sign/in');
13
14         // bad username
15         $this->type("id=frmsignInForm-username", "wrongTester");
16         $this->type("id=frmsignInForm-password", "test");
17         $this->click("id=frmsignInForm-send");
18         $this->waitForPageToLoad("10000");
19         $this->assertTextPresent("'wrongTester'");
20         $this->assertTextPresent("not found");
21
22         // bad password
23         $this->type("id=frmsignInForm-username", "tester");
24         $this->type("id=frmsignInForm-password", "BadBadBadPassword"
25             );
26         $this->click("id=frmsignInForm-send");
27         $this->waitForPageToLoad("10000");
28         $this->assertTextPresent("Invalid password");
29     }
30 }
```

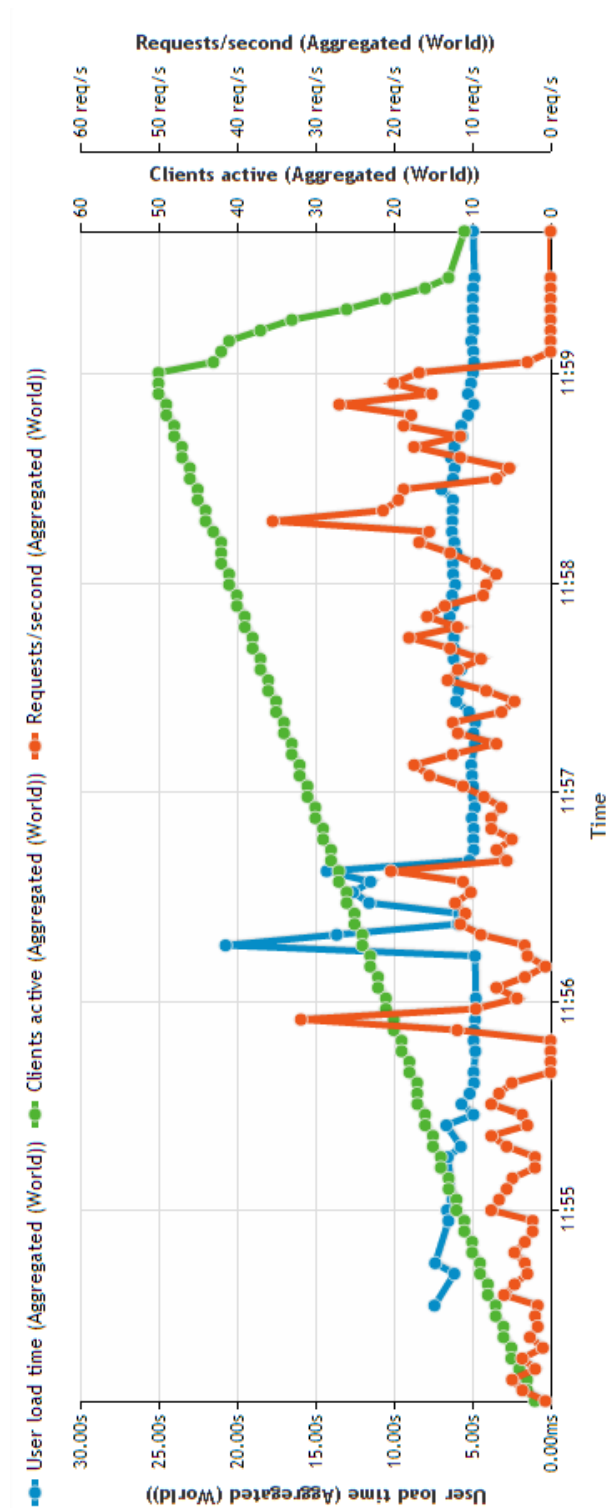


## Příloha C

### Grafy průběhů zátěžových testů



Obrázek C.1: Graf průběhu zátěžového testu stránky s prezentací ukazuje dobu načtení stránky (modrá čára), počet aktivních uživatelů (zelená čára) a počet požadavků za sekundu (oranžová čára)



Obrázek C.2: Graf průběhu zátěžového testu stránky s prezentací uživatelů. Učivo ukazuje dobu načtení stránky (modrá čára), počet aktivních uživatelů (zelená čára) a počet požadavků za sekundu (oranžová čára)



## Příloha D

### Obsah přiloženého CD

src/ .....	zdrojové soubory aplikace
text/	
src/ .....	zdrojové soubory dokumentu
jezdirad-thesis-2013.pdf .....	elektronická verze dokumentu ve formátu PDF
instalacni_manual.pdf .....	návod na instalaci aplikace
readme.txt .....	obsah CD a důležité informace