Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

# Aplikace pro podporu projektu Jáma Lvová

***David Bernhauer***

Vedoucí práce: Ing. Jan Baier

11. května 2015

# Poděkování

Chtěl bych poděkovat panu Ing. Janu Baierovi za vedení mé bakalářské práce. Dále poděkování patří mé rodině a nejbližším přátelům za podporu během celého studia.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel licenční smlouvu o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 11. května 2015 .....

### Odkaz na tuto práci

Bernhauer, David. *Aplikace pro podporu projektu Jáma Lvová*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

# Abstrakt

Tato práce se zabývá zefektivněním procesů v rámci korespondenčního semináře Jáma Lvová. Za cíl si klade zanalyzování procesů, následný návrh databáze a implementaci prototypu webové aplikace, což povede k zefektivnění a zjednodušení procesů jak ze strany organizátorů, tak ze strany soutěžících. Z hlediska analýzy se autor zaměřil na klíčové a problémové procesy a jejich vliv na účast v semináři. Implementace samotného prototypu byla cílena na modularitu, zachování klíčových komponent a usnadnění procesu odesílání a opravování jednotlivých soutěžních řešení. Hlavním požadavkem je intuitivní ovládání a čistota kódu pro snadnou údržbu webové aplikace. Systém by díky vhodné implementaci mohl být rozšířen o modul pro podporu tvorby zadání, správu kontaktů nebo subsystém statistik.

**Klíčová slova** Jáma Lvová, informační systém, analýza procesů, modulární architektura, uživatelské rozhraní

# Abstract

This thesis deals with streamlining processes of the contest Jama Lvova. The goal of this thesis is process analysis, database design and implementation of prototype web application, which leads to streamlining and simplifying processes for organizers and contestants. In term of analysis, the author focused on important and problematic processes and their impact on participation. Implementation of the prototype was targeted on modularity, preserving important components and simplifying process of sending and correcting solutions. The goal is intuitive control and clean code for easy maintenance. Thanks to the appropriate implementation the system could be extended with a module to support creating tasks, improved contact management or statistics.

**Keywords** Jama Lvova, information system, process analysis, modular architecture, user interface

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Jáma Lvová je korespondenční seminář[1] zaměřený především na matematiku a určený studentům vyššího stupně základních škol a odpovídajících ročníků víceletých gymnázií. V každém kole soutěžící řeší celkem 5 úkolů a samotná soutěž je rozdělena na dvě věkové kategorie. Soutěžící svá řešení mohou odevzdávat poštou v papírové formě nebo elektronicky pomocí e-mailu.[1]

## Cíl práce

Cílem této práce je návrh webové aplikace, která by měla soutěžícím usnadnit odevzdávání svých řešení a organizátorům zjednodušit nejen opravování, ale i další povinnosti spojené s organizováním soutěže. Praktickou částí této práce je implementace prototypu a jeho následné otestování. Prototyp musí implementovat moduly pro publikování novinek a pro evidenci došlých řešení. Aplikace se může stát základem pro další rozšiřující moduly a je nutné klást důraz na modulární architekturu.

## Motivace

Je důležité zefektivnit odevzdávání a tím zmenšit bariéru, která pro některé soutěžící může vznikat komplikovaným odevzdáváním pomocí e-mailu. Stejně tak se jedná o bariéru pro opravující, kteří v současné době musí akceptovat několik různých druhů souborů. Aktuálnost daného tématu autor spatřuje v tom, že nyní neexistuje volně dostupné řešení, které by vyhovovalo potřebám projektu Jáma Lvová. Právě to jsou důvody, které autora motivovaly k vypracování bakalářské práce na toto téma.

[1] soutěž, při které řešitelé úlohy řeší doma a odevzdávají je pomocí papírové korespondence nebo elektronickou cestou

Během psaní této práce autor vyvíjí podobný informační systém pro fakultní korespondenční seminář FIKS, na kterém spolupracuje již 2 roky a ze kterého čerpá některé zkušenosti.

## Struktura práce

V první kapitole se autor zaměřuje na analýzu informačních systémů jiných korespondenčních seminářů, které většinou používají vlastní proprietární řešení. Kapitola rozebírá i aktuální řešení webové prezentace Jámy Lvové a problémy s tím spojené.

V druhé kapitole autor podrobně analyzuje jednotlivé procesy. Popisuje hlavní požadavky projektu a na závěr sestavuje doménový model, který je základem pro tvorbu databázového schématu.

Třetí kapitola je věnována návrhu webové aplikace, především volbě vhodných technologií pro tvorbu prototypu. Dále je popsán rozdíl mezi doménovým modelem a skutečným databázovým schématem.

Čtvrtá kapitola popisuje samotnou implementaci prototypu a jeho strukturu. Tato kapitola se také zabývá bezpečností aplikace a doporučeními pro zvýšení bezpečnosti. V závěru popisuje modulární architekturu, jednotlivé moduly a konvence pro tvorbu nových modulů.

Poslední kapitola se zaměřuje na tvorbu uživatelského rozhraní pro přesně specifikované persony. Definuje základní grafické prvky použité v prototypu a pravidla pro zachování přístupnosti. Závěr je věnován samotnému testování uživatelského rozhraní a vyvození závěrů.

Kapitola 1

# Současný stav a existující řešení

Některé korespondenční semináře využívají své vlastní řešení webové aplikace nebo informačního systému vyvinutého v rámci fakulty či univerzity. Proto se následující kapitola bude věnovat současným řešením informačních systémů různých korespondenčních seminářů.

## 1.1 Korespondenční seminář z programování

V rámci osloveného Korespondenčního semináře z programování (zkráceně jen KSP[2]) organizátoři využívají pro odevzdávání vlastní systém s názvem Odevzdávátko, který „slouží k tomu, aby mohli účastníci odevzdávat svá řešení a organizátoři se v nich pak následně vyznali.“[2] Pokud se jedná o úlohu zaměřenou na teoretické řešení problému, pak organizátoři KSP přijímají pouze soubory typu PDF, PostScript či standardní textové soubory.

Dále se v rámci KSP využívá systém CodEx[3], který automaticky vyhodnocuje programy podobně jako fakultní systém Progtest. To není v rámci Jámy Lvové prozatím třeba, ale v případě rozšíření matematických úloh či přidání speciálních matematických modelů by bylo možné využít i systém automatického vyhodnocování podobný tomuto.

Mezi další software, který si KSP vyvíjí samo dle [2], patří:

- databáze účastníků a administrátorské rozhraní k ní (správa škol, tisk obálek pro korespondenci, ...),
- automatický převod z TeXu do HTML,
- program na rozvrhování přednášek (navazující na hlasování účastníků),
- webové plánovadlo soustředění.

[2] `http://ksp.mff.cuni.cz/`
[3] `http://codex2.ms.mff.cuni.cz/project/`

## 1.2 Fiťácký informatický korespondenční seminář

Dalším korespondenčním seminářem, který využívá své vlastní řešení, je Fiťácký informatický korespondenční seminář (zkráceně jen FIKS[4]), ve kterém mohou soutěžící po přihlášení svá řešení odevzdávat ve formě textového souboru nebo ve formátu PDF. Konkrétní řešení je napsáno v programovacím jazyku PHP.

FIKS používá ve svém softwaru moduly pro evidenci účastníků a odevzdávání řešení. Posledním publikovaným rozšířením je systém pro automatické vyhodnocování programů Sfinx.

## 1.3 Jáma Lvová

Korespondenční seminář Jáma Lvová v aktuální chvíli používá webovou prezentaci, která je dostupná na webové adrese `https://jamalvova.cz/`. Tato webová prezentace obsahuje modul pro nahrávání výsledků z opravených řešení. Nahrávání spočívá ve vyplnění jedné tabulky, která obsahuje všechny účastníky, všechna kola a všechny úlohy. Z praktického hlediska se tento stav stává neudržitelným.

### 1.3.1 Existující prototyp

Na půdě Fakulty informačních technologií vznikl díky panu Tomáši Koblížkovi před 2 lety prototyp webové aplikace pro korespondenční seminář Jámy Lvové.[3] Během dvou let se projekt Jáma Lvová a jeho procesy inovovaly a změnily se tak i požadavky na informační systém.

#### 1.3.1.1 Nevýhody zvoleného řešení

S odstupem času a se změnou požadavků projektu Jáma Lvová na informační systém, je vidět několik problémů zvoleného řešení. Volbou vhodného jazyka a případného frameworku se práce zabývá v kapitole 3.1.2. Problémem prototypu je přílišná robustnost řešení a minimální organizace do jednotlivých modulů.

Většina entit (reprezentujících databázové tabulky) je umístěna v modulu `StaticBundle`, který se má dle názvu starat pouze o statické stránky. Tyto entity jsou pak načítány v jiném modulu. Druhým problémem je přílišná dekomponovanost databázových tabulek. Samotná dekompozice není špatná pro budoucí rozvoj aplikace, ale chybí k ní dostatečná dokumentace a komplikuje tak sestavování databázových dotazů. Databáze tak navíc zajišťuje veškerý chod aplikace včetně logování, které tím pádem při výpadku databáze nefunguje a nemůže tak zalogovat například pokus o útok na stránku.

[4] `https://fiks.fit.cvut.cz/`

Posledními problémy, které souvisí s novými potřebami projektu, jsou přílišná orientovanost procesů na organizátora[5] a rozšiřitelnost. Novým cílem je zapojení účastníků do procesu zjednodušení odevzdávání a opravování. Zprovoznění prototypu vyžaduje zásah do autorizační části.

### 1.3.2 Další nástroje

V rámci projektu Jámy Lvové se aktivně používají další webové nástroje pro komunikaci a uchovávání informací. Jejich výčet může být důležitý především v souvislosti s návrhem uživatelského rozhraní.

**Simple Machines Forum** Jako hlavní komunikační kanál používají organizátoři Jámy Lvové diskuzní fórum SMF[6], které je naimplementované v jazyku PHP a běží na databázovém systému MySQL [4]. Tento komunikační kanál je používán jak na organizační záležitosti, tak na komunikaci se soutěžícími.

**DokuWiki** Pro uchovávání informací, znalostí a zkušeností využívají organizátoři Jámy Lvové nástroj DokuWiki[7]. Jedná se o wiki aplikaci[5], což je nástroj, kde obsah je přidáván, upravován a organizován uživateli společně [6].

[5]Soutěžící může pomoci zlepšit procesy tím, že bude odevzdávat svá řešení přímo na server aplikace. Pro soutěžící to znamená okamžitou zpětnou vazbu o úspěšném přijetí jejich řešení.

[6]`http://www.simplemachines.org/`

[7]`https://www.dokuwiki.org/`

Kapitola 2

# Analýza

## 2.1 Procesy v projektu Jáma Lvová

Základním požadavkem, který projekt klade na webovou aplikaci, je modulární architektura s implementovanými moduly pro zveřejňování novinek a pro evidenci došlých řešení. Mezi další nefunkční požadavky patří zjednodušení práce nejen organizátorů při hodnocení, ale i soutěžících při odevzdávání.

Pro účely soutěže tak mohou být užitečná i další podpůrná rozšíření, k čemuž je žádoucí vytvořit analýzu procesů a identifikovat procesy, které je potřeba zlepšit a které lze efektivně podpořit webovou aplikací. Tato analýza by měla sloužit i k vhodnému návrhu celé aplikace v závislosti na možných změnách.

### 2.1.1 Pravidelný cyklus soutěže

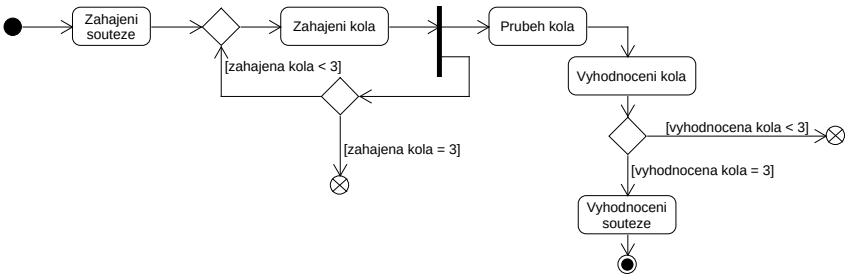
Soutěž se skládá z pravidelných ročníků, přičemž délka jedné sezóny je obvykle okolo 10 měsíců se začátkem v září a koncem v létě následujícího roku. V létě pak na úspěšné účastníky čeká tábor.

Každý ročník se skládá ze 3 kol a závěrečného tábora [1]. Délka jednotlivých kol může být variabilní a může se měnit i v rámci již probíhajícího kola (např. prodloužení odevzdávání). Některé části soutěžních kol se mohou překrývat s jinými koly, a tak se nejedná o čistě sekvenční proces, s čímž je nutné počítat i při návrhu aplikace. Možná paralelizace je pak vidět na diagramu 2.1.

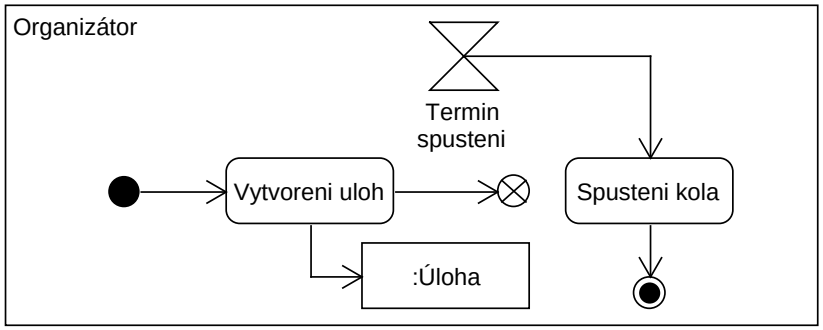
#### 2.1.1.1 Zahájení soutěžního kola

V rámci interního fóra jednotliví organizátoři navrhují zadání úloh pro nové kolo. Z těchto návrhů je vybráno celkem 5 úloh pro každou kategorii, které dostanou jednotliví organizátoři ke zpracování a sepsání „pohádkového“ zadání.

Své „pohádkové“ zadání předají sazeči, který všechna zadání zpracuje v sázecím systému LaTeX do jednoho souboru a zveřejní ve formátu PDF a PNG




Obrázek 2.1: Předpokládaný průběh ročníku soutěže



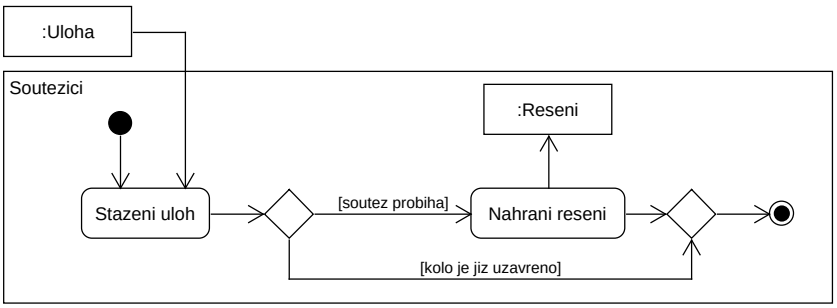

Obrázek 2.2: Zahájení soutěžního kola

na webových stránkách se začátkem kola. Pro usnadnění práce sazeče by se hodila možnost načasovat zveřejnění zadání, které by zajistil systém v přesný čas, jak je vidět na diagramu 2.2

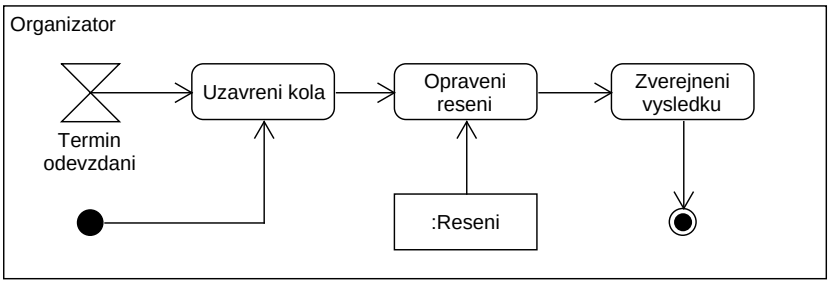
#### 2.1.1.2 Průběh kola

Během samotného kola soutěžící odevzdávají svá řešení pomocí papírové korespondence nebo pomocí e-mailu. Zde vzniká problém s nutností akceptovat více formátů, včetně papírové verze. To přináší i problémy s kontrolou odevzdání úloh do určitého data. Obzvláště v případě e-mailů bývá sporné datum odeslání s datem přijetí e-mailu. Nepříjemné faux pas může být i případná zapomenutá příloha dodaná po termínu odevzdání.

Problémy s odevzdáváním v mnoha formátech a po časovém intervalu se dají podpořit vhodnými moduly. V rámci své práce autor dělal průzkum mezi 17 studenty 6. – 8. tříd, ze kterého vyplynulo, že exportovat či tisknout do PDF umí přibližně 6% z nich. A pouze 24% studentů vlastní funkční skener pro skenování svých řešení. Proto je nutné poskytnout návod na převod textu a obrázků do formátu PDF.




Obrázek 2.3: Průběh soutěžního kola




Obrázek 2.4: Ukončení soutěžního kola

Možný průběh soutěžního kola je zobrazen na diagramu 2.3.

#### 2.1.1.3 Ukončení kola

Po oficiálním ukončení kola se přijatá řešení roztřídí dle jednotlivých úloh. Elektronická řešení se rozešlou elektronickou poštou organizátorům a papírová se předají opravujícím na nejbližší schůzce organizátorů. Odevzdaná řešení s jejich opravami (v současném řešení pouze jejich ohodnocení) je nutné evidovat a výsledky zobrazovat na příslušné webové stránce. Při opravování by se měl každý organizátor postarat o svoji úlohu, v praxi se však může stát, že odpovědnou osobou stále zůstává stejný organizátor, ale z důvodu nedostatku času úlohu opravuje někdo jiný. Průběh je znázorněn na diagramu 2.4.

### 2.1.2 Speciální kola

Vzhledem k budoucímu vývoji samotné soutěže je logické předpokládat, že může nastat situace, kdy organizátoři budou chtít uspořádat letní prázdninové kolo soutěže. Tudíž by výsledná aplikace měla počítat s možností nad-

standardních soutěží, které je nutné zaintegrovat do samotného informačního systému.

## 2.2 Analýza požadavků

### 2.2.1 Publikování novinek

Základním kamenem celé webové aplikace je subsystém pro publikování a archivaci novinek. Jeho použití musí být jednodušší než upravování samotného zdrojového HTML souboru. Protože právě tento modul je základem pro komunikaci se soutěžícími, bude toto kritická část aplikace. Předpokladem je, že část publikování novinek bude spravovat osoba, která není technickým odborníkem, ale zvládne se případně naučit jednoduchou syntaxi či použít formátovací značky HTML.

S tím se pojí i občasné zveřejnění novinky, jejíž aktuálnost a důležitost trvá pouze po určitý časový úsek, po kterém již nemá smysl nechávat novinku v archivu. Na druhou stranu její důležitost je větší než důležitost běžné novinky. Vzhledem k tomu, že tento případ bude nastávat minimálně, není nutné tuto funkčnost implementovat, mnohem efektivnější v tomto případě bude přímá editace zdrojového kódu.

### 2.2.2 Zaznamenávání a zobrazování výsledků

Druhým požadavkem na prototyp je zaznamenávání a zobrazování výsledků soutěže dle jednotlivých kol. Z hlediska zadávání by měl mít organizátor možnost zadávat výsledky pouze úloh, za něž má odpovědnost. Zde však hrozí riziko, že implementace odstínění organizátorů od opravování cizích úloh může zvýšit nároky kladené na jednotlivé organizátory, kteří jsou více aktivní ve fázi tvoření úloh. Vzhledem k malé velikosti týmu tak může být toto odstínění kontraproduktivní.

V této části vidím největší potenciál na zlepšení procesů v rámci soutěže. Tyto procesy mohou ulehčit spoustu práce jak organizátorům, tak soutěžícím.

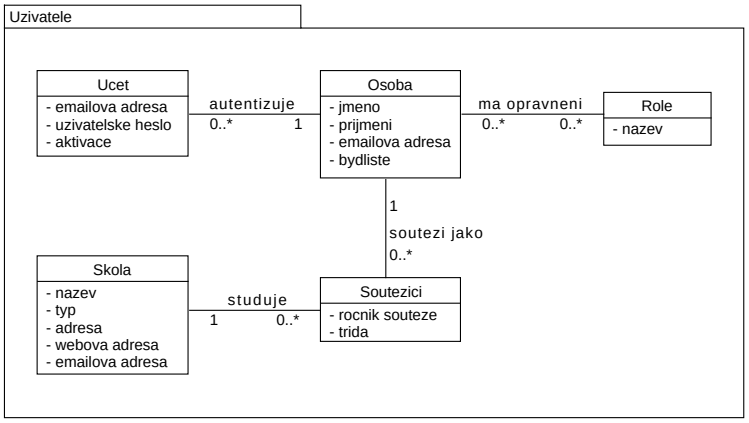
## 2.3 Doménová analýza

Pro bližší pochopení jednotlivých termínů a vazeb je vhodné sestavit a popsat doménový model. Zároveň bude sloužit jako základ pro návrh relační databáze.

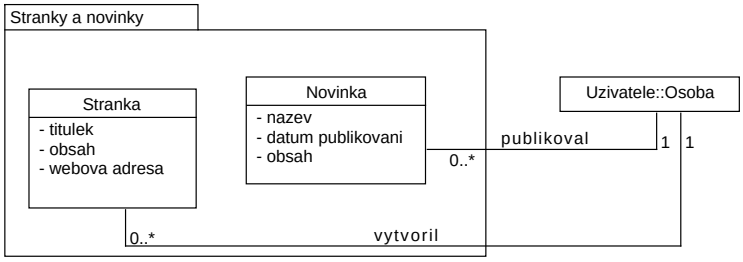
### 2.3.1 Uživatelé

Z hlediska domény je výhodné rozdělit si celý projekt do několika menších balíčků. Prvním z nich je balíček obsahující entity související s uživateli.

Zajímavou entitou je v našem diagramu 2.5 entita `Soutezici`, která reprezentuje informaci o soutěžícím studentovi a jeho škole. Tu zaznamenáváme,




Obrázek 2.5: Doménový model: Uživatelé




Obrázek 2.6: Doménový model: Stránky a novinky

z toho důvodu, že student může přestupovat mezi školami nebo že v nejhorším případě nemusí postoupit do dalšího ročníku. Záznam pro každý ročník soutěže nám tak zachovává i možnost každý rok po soutěžících vyžadovat aktualizaci jejich údajů, což je nutná podmínka k uchovávání informací o soutěžících. Z hlediska použitelnosti je uživatelům vhodné nabídnout nejpravděpodobnější scénář.

### 2.3.2 Stránky a novinky

Jediným nejasným bodem na diagramu 2.6 (reprezentující doménový model Stránky a novinky) může být webová adresa, ta se uvádí z toho důvodu, že chceme dát možnost zobrazování stránek s hezkými URL adresami bez jedinečného (pro soutěžící nicneříkajícího) číselného identifikátoru v URL. To může zlepšit indexaci soutěže ve vyhledávačích.

Tabulka 2.1: Atributy entity Kolo

| | |
|---|---|
| viditelne od | Od tohoto data soutěžící vidí, kdy se spouští (otevírá) toto kolo. Po tomto datu nelze (bez speciálních oprávnění) měnit atributy entity `Kolo` |
| spustene od | Od tohoto data soutěžící vidí zadání úloh a mohou odevzdávat svá řešení. Po tomto datu nelze (bez speciálních oprávnění) měnit zadání úloh. |
| spustene do | V tento den končí soutěžícím možnost odevzdávat svá řešení. Opravovatelé mohou začít opravovat a hodnotit. |
| zobr. vysledky od | Od této chvíle se zveřejní výsledky všech soutěžících. |

### 2.3.3 Soutěž

Doménový model soutěže je pravděpodobně nejrozsáhlejším doménovým modelem v projektu. Samotné kategorie soutěže jsou v doménovém modelu nazvány jako `Soutez`. Je to z toho důvodu, že v případě zavedení letního kola, může nastat situace, kdy letní kolo bude společné pro všechny soutěžící a nebude rozděleno na kategorii mladší a starší. Proto se název `Soutez` zdá vhodnějším.

Druhý bod, na který je potřeba upozornit, je entita `Rocnik`, která nereprezentuje ročník soutěže (jako kategorie), ale reprezentuje ročník celé soutěže jako Jámy Lvové.

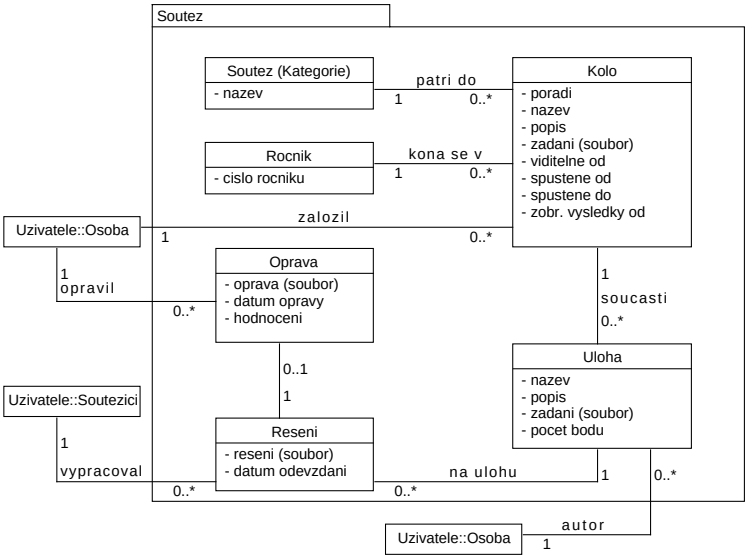
#### 2.3.3.1 Kolo

Atributy entity `Kolo` jsou samovysvětlující, až na poslední čtyři atributy, které jsou vysvětlené v tabulce 2.1. Teoreticky by mohla nastat situace, kdy organizátor bude chtít posunout odevzdávání pouze jedné úlohy. Tento požadavek by samozřejmě vyžadoval jiný přístup, a to přenesení, některých dat do entity `Uloha` nebo změnit úplně celý koncept.

#### 2.3.3.2 Další entity

Další entity neobsahují žádné nejasné atributy, ale zajímavé mohou být některé vazby mezi nimi a jejich násobnosti.

**Autor úlohy** Dle diagramu 2.7 musí každá úloha mít právě jednoho autora. Což nemusí být v praxi vždy pravda, například pokud dva organizátoři spolupracují na jedné úloze. Vazba na právě jednoho autora, byla zvolena z hlediska zlepšení procesů, kdy za úlohu je odpovědná právě jedna osoba, která sice může mít pomocníky, ale pokud nastane problém, je jasné na koho se obrátit.




Obrázek 2.7: Doménový model: Soutěž

**Řešení úlohy** Každá úloha může mít právě jedno řešení od právě jednoho soutěžícího. To v doménovém modelu nemohlo být znázorněno, ale pro organizační tým je důležité pouze poslední odevzdání, tudíž všechna jiná odevzdání jsou irelevantní. Z toho důvodu může být vazba striktnější a povolovat pouze jedno řešení k jedné úloze.

**Oprava řešení** Jedná se opět o podobnou situaci, kdy není možné mít dvě opravy jednoho řešení, např. od dvou různých organizátorů. Opět se jedná o návrh zefektivňující procesy ve fungování projektu. V aktuální chvíli je přiřazen ke každé úloze právě jeden organizátor, který opravuje všechna řešení. V případě zvýšení počtu účastníků by mohla nastat situace, že bude potřeba zvýšit počet opravujících organizátorů na jednu úlohu. Ale i poté je zcela nežádoucí, aby dva organizátoři zbytečně opravovali jednu úlohu. Tato situace by se měla řešit na procesní úrovni.

Kapitola 3

# Návrh

## 3.1 Technologie

Mezi nejdůležitější volby při programování patří výběr vhodných technologií pro implementační část. Celou aplikaci je možné rozdělit do celkem tří úrovní:

- databázová část
- aplikační část
- prezentační část

### 3.1.1 Databázová část

Implementační část této práce je odladěna na databázovém serveru MySQL[8], který je dle [7] druhým nejvíce používaným databázovým serverem. Zároveň díky integraci v mnoha distribučních balíčcích typu AMP (Apache[9], MySQL, PHP) je mnohem více dostupný začínajícím programátorům webových aplikací. Vzhledem k zvolenému frameworku a jeho knihovně pro práci s databází není problém zvolit jako databázový server i jiný typ, např. PostgreSQL[10]. Až na skript pro tvorbu základních tabulek je aplikace plně přenositelná na jiný databázový server podporovaný použitým frameworkem Nette (viz dále).

### 3.1.2 Aplikační část

#### 3.1.2.1 Programovací jazyk

Základní otázkou v případě webových aplikací je volba programovacího jazyka. Vzhledem k rozšíření programovacích jazyků a frameworků v oblasti webových aplikací se autor rozhodl zúžit výběr na celkem 3 jazyky:

[8] `https://www.mysql.com/`
[9] `http://httpd.apache.org/`
[10] `http://www.postgresql.org/`

**Python** Python se stal populární v oblasti webových technologií především díky populárnímu frameworku Django[11], který byl původně vytvořen v novinářském prostředí [8]. „Naučit se dá za týden,“[9] a nucením programátora odsazovat bloky ho učí tvorbě přehledného kódu.

**Perl** Dle [10] je Perl „jednoduchý a praktický interpretační programovací jazyk. V praxi je využívaný zejména na WWW serverech a při zpracování textu. V oblasti zpracovávání souborů patří bez nadsázky vůbec k těm nejlepším nástrojům, které existují. Ostatně právě kvůli tomu byl Perl navržen.“ Syntax Perlu dovoluje kód strukturovat tak, aby byl více pochopitelnější pro člověka, například cyklus `until` v mnoha jazycích chybí. Za zmínku stojí i fakt, že Perl stejně jako Python disponuje několika frameworky pro vývoj webových aplikací (např. Mojolicous[12] a Perldancer[13]).

**PHP** PHP neboli „hypertextový preprocesor“[11] je dle [12] 7. nejpopulárnějším programovacím jazykem. Mezi skriptovacími jazyky určenými pro vývoj webových aplikací, které operují na straně serveru, je tak prvním kandidátem. Díky velké komunitě je postaveno na PHP několik desítek frameworků, což ho činí o to populárnějším [13]. To je vidět i na grafu popularity jazyků ve službách StackOverflow a GitHub [14].

**Shrnutí** Vzhledem k možné nutnosti častých změn funkčnosti, byly z výběru automaticky vyřazeny programovací jazyky vyžadující kompilaci. To usnadnuje aktualizaci aplikace přímo na serveru v reálném čase. Pro výběr jazyka se autor rozhodoval nejen dle svých osobních preferencí, ale také dle popularity jazyka v komunitě začátečníků a dle dostupnosti informací o daném jazyce. Proto byl vybrán jazyk PHP, který je vyučován na Fakultě informačních technologií ČVUT, což zvyšuje šanci, že další programátoři budou tento programovací jazyk ovládat. „PHP, MySQL a Apache jsou ve světě Internetu nejrozšířenějšími technologiemi pro tvorbu a správu webových aplikací a webového serveru. Známé jsou především svou kvalitou a dostupností zdarma.“ jak uvádí [15] z roku 2006. Důležitým faktem v tomto případě je dostupnost zdarma, kdy začínající programátoři mohou využít velkého množství webhostingových služeb zdarma.

#### 3.1.2.2 Framework

Framework je velice podobný klasickým knihovnám až na několik rozdílů jak je uvedeno v [16]:

[11] `https://www.djangoproject.com/`
[12] `http://mojolicio.us/`
[13] `http://perldancer.org/`

- Organizace řešení nespočívá na bedrech uživatele, ale využitím principu inverze závislosti je vše v rukou frameworku, jemuž se musí jeho uživatel přizpůsobit.

- Framework má definované smysluplné implicitní chování.

- Framework není obecně modifikovatelný. Přesněji řečeno: veškeré požadavky na úpravu či rozšíření jeho funkčnosti by měly být realizovatelné bez zásahu do jeho kódu.

- Framework je ale rozšiřitelný. Jeho funkčnost lze rozšiřovat či upravovat přidáním vhodného kódu, který se požadovaným způsobem naváže na původní kód daného frameworku.

Nad jazykem PHP je postaveno hned několik zajímavých frameworků, které usnadňují vývoj aplikací (tzv. RAD) a většina z nich v sobě implementuje návrhový vzor MVC (Model-View-Controller).

**Symfony 2.3** Symfony[14] je velice robustní PHP framework usnadňující vývoj webových aplikací. Díky rozsáhlé kolekci různých komponent a knihoven, je hojně používaný ve větších projektech. Má poměrně velkou komunitu programátorů, využívá systémů Twig a SwiftMailer [17].

**CakePHP 3.0** CakePHP[15] je framework navržený tak, aby zjednodušil a ulehčil běžné úkoly při vývoji webových aplikací. Hlavní koncept CakePHP je přednost konvencí před konfigurací, což značně omezuje svobodu programátora [18].

**Nette 2.3** Nette[16] je český framework, který se pyšní velkou českou komunitou programátorů a webových vývojářů v České republice [19][20]. Nette obsahuje knihovny pro práci s databází a e-maily. Dle [21] se Nette umístilo na předních příčkách při testu výkonnosti PHP frameworků.

**Shrnutí** Jelikož aplikace pro projekt Jáma Lvová by měla být co nejvíce minimalistická, není nutné používat robustní nástroj jako je Symfony. Kvůli integraci pouze základních knihoven a omezení konvencemi, CakePHP také není vhodným kandidátem pro webovou aplikaci tohoto druhu. Proto se autor rozhodl pro použití českého frameworku Nette.

[14] `http://symfony.com/`
[15] `http://cakephp.org/`
[16] `http://nette.org/`

#### 3.1.2.3 Další nástroje

Pro získávání závislostí jednotlivých použitých balíčků autor používá správce závislostí `composer`[17]. Výhodou tohoto nástroje je automatické instalování příslušných knihoven [22]. Pro vývoj celého projektu je využito vývojové prostředí PhpStorm[18] od firmy JetBrains.

### 3.1.3 Prezentační část

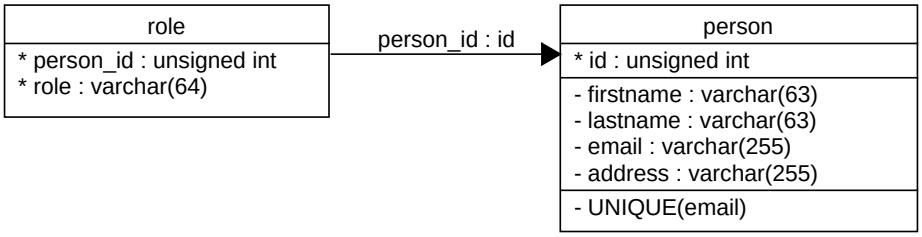
K prezentační části bude využito základních technologií (podporovaných nejnovějšími a nejrozšířenějšími prohlížeči) pro vývoj webových aplikací HTML 5, CSS 3 a JavaScriptu. JavaScript bude v našem případě použit pouze jako možnost vylepšení použitelnosti. Pro prototypování se autor rozhodl využít dle [23] nejpopulárnější HTML, CSS a JS framework Bootstrap [24]. Popularitu tohoto frameworku zobrazuje i statistika na stránkách `http://tech.wpgpl.com/CSS_Frameworks`. Je tedy velká pravděpodobnost, že budoucí grafik využije právě tento framework. Použitou JavaScriptovou knihovnou je jQuery[19], která je prerekvizitou pro Bootstrap.

## 3.2 Databázové schéma

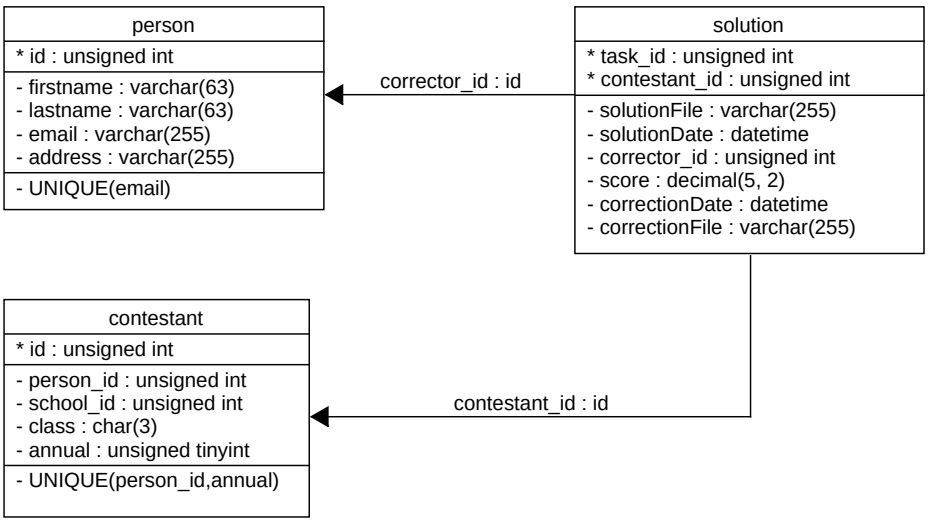
Databázové schéma vychází z doménového modelu. Podstatný rozdíl je ve zjednodušení vazby mezi uživateli a jejich oprávněními (diagram 3.1), kde je za současné situace zbytečné dekomponovat vazbu M:N. Druhým výrazným rozdílem je sloučení entit `Reseni` a `Oprava` do jedné tabulky, jak je znázorněno na digramu 3.2. Toto sloučení bylo provedeno, aby se zjednodušil počet vazeb mezi všemi tabulkami.

Kompletní databázové schéma je přiloženo na CD ve složce `src/scripts`. Je na něm mimo jiné vidět odstranění vazeb mezi uživatelem (organizátorem) a stránkami či úlohami. Toto zjednodušení napomáhá lepšímu rozdělení do jednotlivých modulů.

[17] `https://getcomposer.org/`
[18] `https://www.jetbrains.com/phpstorm/`
[19] `https://jquery.com/`




Obrázek 3.1: Vazba: Uživatel a Role




Obrázek 3.2: Sloučení entit: Řešení a Oprava

Kapitola 4

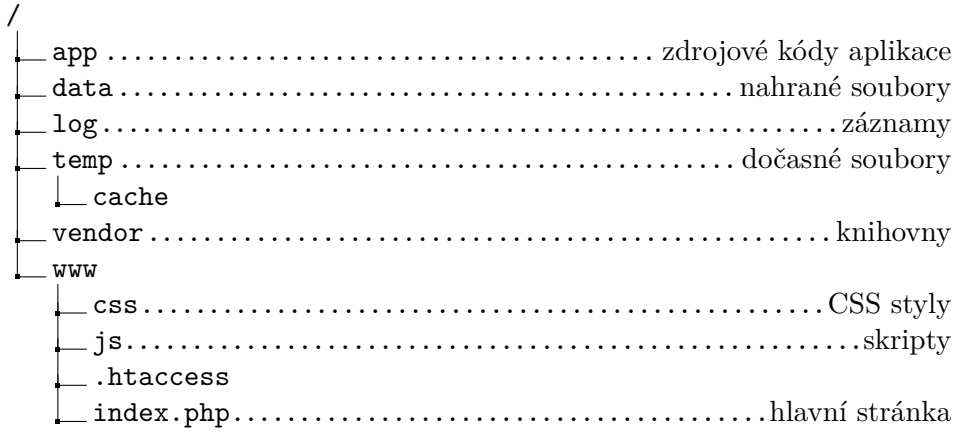
# Implementace

## 4.1 Struktura

Základní struktura projektu je rozdělena na části obsahující zdrojové kódy, dočasné soubory, nahrané soubory, knihovny a veřejné soubory. Strukturu v rámci projektu znázorňuje diagram 4.1. Struktura knihoven je určena nástrojem `composer`, který se stará o stahování a provázání všech balíčků a jejich závislostí. Struktura složek s nahranými soubory je určena implementací aplikace s tím, že o složky se záznamy a dočasnými soubory se pak stará framework Nette.

### 4.1.1 Model-View-Controller

Struktura zdrojových kódů dodržuje většinu konvencí používaných při vývoji nad frameworkem Nette. Důležitou informací však je, že Nette podporuje ná-




Obrázek 4.1: Struktura projektu

vrhový vzor Model-View-Controller (zkráceně MVC), který aplikaci rozděluje do 3 logických vrstev.

#### 4.1.1.1 Model

„Model je datový a zejména funkční základ celé aplikace. Je v něm obsažena aplikační logika.“[25] Do modelu se řadí například i validační pravidla, protože model samotný představuje doménový model a logiku mezi věcmi [26]. Některé třídy však nemusí být definovány čistě na vrstvě modelu, ale mohou být definovány na více vrstvách, protože se starají o více věcí. Model obvykle neví nic o Controlleru či View.

#### 4.1.1.2 View

„View, tedy pohled, je vrstva aplikace, která má na starost zobrazení výsledku požadavku.“[25] V našem případě se jedná zejména o použití šablonovacího systému Latte, který je dodáván spolu s frameworkem Nette.

Všechny pohledy jsou tedy uloženy ve složce `templates`. Struktura této složky kopíruje strukturu presenterů (viz 4.1.1.3) a všechny šablony mají příponu `.latte`. Speciální šablonou je šablona `@layout.latte` sloužící jako obal, který je společný pro všechny webové stránky.

#### 4.1.1.3 Controller

„Řadič, který zpracovává požadavky uživatele a na jejich základě pak volá patřičnou aplikační logiku (tj. model) a poté požádá view o vykreslení dat. Obdobou kontrolerů v Nette Framework jsou presentery.“[25] Ve frameworku Nette jsou presentery všechny třídy, které dědí od společného předka `Presenter`. Dle konvence jsou všechny presentery uloženy ve složce `presenters`.

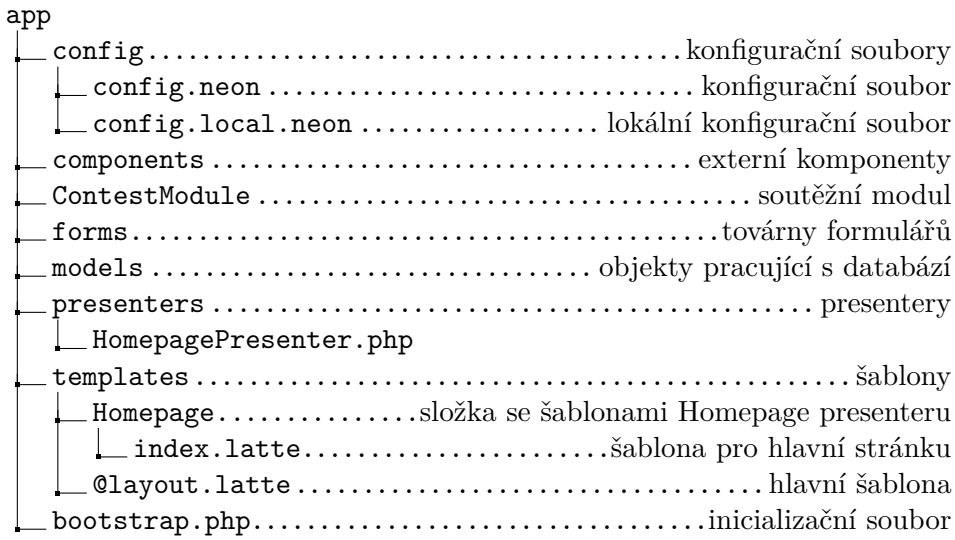
Ke správné funkčnosti presenteru je nutné nastavit ve směrovači (`Router`) správné cesty a vzory URL adres. Presenter zpracovává nejen klasické požadavky, ale také požadavky vytvořené jednotlivými komponentami[20].

### 4.1.2 Další části aplikace

Mezi další části patří komponenty a formuláře. Komponenty buď jsou definovány lokálně v presenteru pomocí tzv. továren a nebo jsou jejich definice uložené ve složce `components`. Stejně tak formuláře a jejich továrny[21], které jsou uloženy ve složce `forms`. Komponenty a formuláře jsou obvykle na pomezí vrstvy controlleru a modelu. Případně lze přidávat další moduly (více v kapitole 4.3).

[20]Komponenty jsou malé znovupoužitelné jednotky, které zajišťují logiku, která se dá využít na více stránkách. Nejčastěji se jedná o různé formuláře.

[21]Simple factory method (jednoduchá (statická) tovární metoda) je návrhový vzor, který je alternativou volání konstruktoru, jemuž se za určitých okolností chceme vyhnout.[27]




Obrázek 4.2: Struktura zdrojových kódů

Hlavní složka `app` obsahuje navíc složku `config` s konfiguračními soubory[22] a inicializační soubor `bootstrap.php`. Podrobná struktura projektu je vidět na diagramu 4.2.

## 4.2 Bezpečnost

Nejdůležitějším aspektem každé webové aplikace je úroveň poskytovaného zabezpečení. Některá bezpečnostní pravidla zajišťuje aplikace sama, jiná naopak jsou závislá čistě na správci webového a databázového serveru nebo na samotném uživateli aplikace.

### 4.2.1 Integrovaná bezpečnostní opatření

Díky použití frameworku Nette a jeho knihovnám je zajištěna bezpečnost před injektováním a XSS[23]. Tato bezpečnostní rizika byla v roce 2013 spolu s chybnou autentizací zařazena skupinou OWASP[24] mezi 3 nejzávažnější bezpečnostní rizika [28]. Správné používání Nette frameworku tato rizika eliminuje.

Vytvořený prototyp tyto vlastnosti frameworku používá, kromě úpravy statických stránek administrátorem, kde teoreticky může nastat útok typu XSS,

[22] Lokální konfigurační soubor obsahuje informace o připojení k databázi, nastavení hlavního titulku a připojení k e-mailovému serveru.

[23] Cross-Site Scripting je technika, která „útočníkům umožňuje spouštět skripty v prohlížeči oběti.“[28]

[24] Open Web Application Security Project je projekt zabývající se bezpečnostními rizika na webu.

ale pouze ze strany administrátora. V této části je žádoucí, aby administrátor měl k dispozici možnost upravit HTML. Pokud bychom chtěli maximálně zabezpečit i tuto část prototypu, bylo by nutné vytvořit vlastní značkovací jazyk, který by se příslušní administrátoři museli naučit.

**Autentizace a Autorizace** Problémy s chybnou autentizací za nás řeší Nette a třída `Authenticator`, která je zodpovědná za správné přihlášení uživatele. Z toho důvodu je třída jednoduchá, aby se autor vyhnul zbytečným chybám při implementaci. Autorizace je v prototypu zajištěna na úrovni presenterů[25], kde všechny akce jednoho presenteru vyžadují stejné oprávnění. Aplikace podporuje obranu před špatnou autorizací implementací dvou výchozích presenterů, od kterých dědí všechny[26] ostatní presentery:

- `RootPresenter` je základním presenterem pro všechny veřejně přístupné stránky.
- `SecurityPresenter` je presenter pro stránky vyžadující přihlášení.

Některé moduly (např. `CoreModule`) pak implementují svoje vlastní abstraktní presentery (např. `EditPresenter`) ověřující, zda uživatel má dostatečná oprávnění k provedení akce.

**Cross-Site Request Forgery** Dalším útokem, proti kterému Nette nabízí ochranu, je CSFR neboli Cross-Site Request Forgery. „Útočník může podstrčit obsah do prohlížeče uživatele stránek, a tím bez jeho vědomí zaslat požadavek na webový server.“[28] Nette umožňuje do důležitých formulářů vložit tzv. CSRF token, který se po přijetí požadavku na straně serveru zkontroluje a invaliduje. Tudíž každý token lze použít jen jednou a pouze po omezeně dlouhou dobu[27].

Prototyp ve všech důležitých formulářích používá právě tento CSRF token, jedinou vyjímkou jsou odkazy na smazání položek. Ty v prototypu nejsou ošetřeny z důvodu zaměření prototypu na testování uživatelského rozhraní. Ve výsledné aplikaci by na odstranění již neměly vést odkazy, ale formuláře, které se budou odesílat pomocí JavaScriptu (právě pro zjednodušení uživatelského rozhraní).

**Validace formulářů** Aby autor zabránil nesmyslným a nebezpečným vstupům od uživatele, jsou všechny formuláře validovány na straně serveru. Některé validační funkce jsou opět díky frameworku Nette navíc validovány pomocí JavaScriptu, což usnadňuje uživatelům práci s těmito formuláři.

[25] „Presenter je objekt, který vezme požadavek přeložený routerem z HTTP požadavku a vygeneruje odpověď.“[29]

[26] Kromě speciálních presenterů jako je FilePresenter, který slouží k stahování souborů.

[27] Doba trvanlivosti tokenu závisí na nastavení.

### 4.2.2 Doporučovaná bezpečnostní opatření

V následující sekci autor poskytuje výčet základních bezpečnostních opatření, o které se nemůže aplikace postarat sama. A je tedy nutné zvýšené pozornosti správce systému.

#### 4.2.2.1 HTTPS a certifikáty

Vzhledem k faktu, že aplikace posílá ve formulářových požadavcích citlivé údaje jako jsou e-mailové adresy, hesla a jména, je nutné, aby webový server, na kterém aplikace běží, plně podporoval protokol HTTPS a používal bezpečný certifikát. Ověřit bezpečnost použitého certifikátu je možné například pomocí nástroje SSL Server Test[28] od firmy Qualys. Aplikace z bezpečnostních důvodů je automaticky nastavena na používání protokolu HTTPS a vypnutí vyžaduje zásah do zdrojových kódů.

#### 4.2.2.2 Databázový účet

Je vřele doporučováno, aby přístupový účet do databáze, který se nastavuje v souboru `config.local.neon`, měl co nejvíce omezená práva. To zamezuje tomu, aby i v případě úniku tohoto hesla nebyla poškozena jiná část databáze. Samotný konfigurační soubor by měl být chráněn tak, aby nebyl dostupný z internetu (viz kapitola 4.2.2.3).

#### 4.2.2.3 Viditelnost souborů

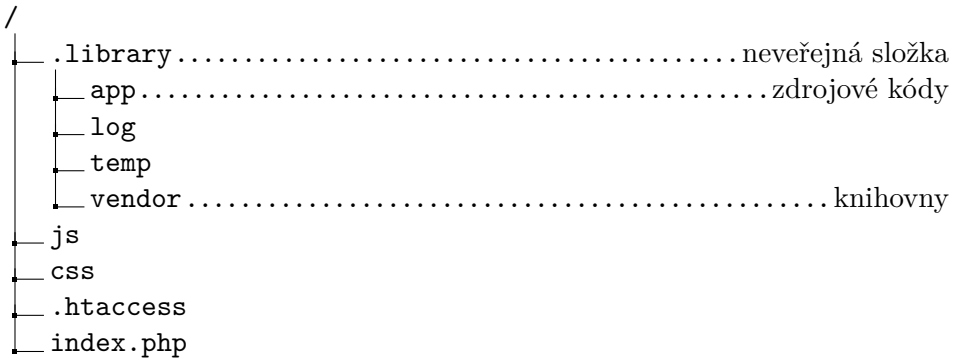
Samotná struktura projektu zamezuje tomu, aby zdrojové a konfigurační soubory byly přístupné z internetu. Pokud ovšem poskytovatel webového hostingu nepodporuje umístění souborů do neveřejné části, pak je nutné pozměnit tuto strukturu. Všechny soubory a složky, které mají být neveřejné, je doporučováno přesunout např. do složky `.library`, která při zachování aktuálního `.htaccess` souboru je nepřístupná z internetu (vyobrazeno na 4.3).

### 4.2.3 Paralelní zpracování

Povaha webových aplikací je ta, že k jedné webové stránce může přistupovat více uživatelů zároveň a obvykle upravovat data v databázi. Z hlediska soutěžícího se prakticky nemůže stát, že by dva lidé nahrávali odpověď za jednoho soutěžícího ve stejnou dobu. Místo řešení nepravděpodobného problému je v této situaci lepší poučit a vychovávat uživatele aplikace.

**Paralelní opravování** Z pohledu organizátorů se v [3] řeší uzamykání a odemykání opravování jednotlivých úkolů. Toto zamykání a odemykání je implementováno v aktuální verzi z důvodu, že k úpravě dat je využita pouze jedna

[28] `https://www.ssllabs.com/ssltest/`




Obrázek 4.3: Alternativní struktura projektu

velká globální tabulka společná pro všechny úkoly, a tak skutečně tato situace, kdy dva organizátoři upravují stejnou tabulku, avšak různé úlohy, ve stejnou dobu, může nastat.

**Řešení** Ve vytvořeném prototypu je tento problém řešen rozdělením opravování na jednotlivé úkoly a jednotlivá odevzdaná řešení. V současném stavu opravu jedné úlohy zajišťuje právě jeden organizátor. V případě většího počtu organizátorů, by bylo nevýhodné opravovat některá řešení dvakrát, proto nemůže tento problém nastat a měl by se řešit procesně. (Přidělování řešení k opravujícím by mohlo vyústit v problematickou situaci, kdy zaneprázdněný organizátor má přidělenu opravu, a blokuje tak ostatní. Navíc jeden opravující nemůže v jednu chvíli opravovat více prací.)

## 4.3 Modulární architektura

Základním požadavkem aplikace byla i architektura umožňující jednoduché přidávání nových modulů a částí. Autorovým cílem bylo zajistit modularitu aplikace tak, aby provázanost jednotlivých modulů byla co nejnižší a aby zároveň pro programátora modulu byla architektura co nejjednodušší na pochopení, včetně pochopení samotné integrace do aplikace.

Druhým autorovým cílem bylo zajistit co nejmenší počet globálních (mezi moduly sdílených) souborů, které je nutné upravovat při integraci modulu, ale zároveň se vyhnout automatickému prohledávání adresářů či zavádění konvencí, které svazují ruce programátorům a můžou být naopak nežádoucí.

### 4.3.1 Existující moduly

Kromě globálního kontextu, do kterého jsou moduly zasazeny se v prototypu nachází celkem 4 moduly, které zajišťují chod celého prototypu. Ty mohou být

dále rozšířeny či doplněny jinými moduly. Následuje výčet implementovaných modulů:

- logování (záznam) akcí,
- správa uživatelů,
- publikování novinek a stránek,
- soutěžní část.

Modul pro logování akcí slouží pro větší kontrolu nad ovládáním kritických částí. Proto je tento modul nezávislý na jiných modulech a přes sdílený kořenový presenter je tak dostupný ve všech jiných presenterech. Implementace je provedena tak, aby v případě nutnosti odpojení logovacího systému se logování pouze ignorovalo. Narozdíl od [3] se logování ukládá do souborů a paralelní zápis je řešen frameworkem Nette.

Modul pro správu uživatelů byl od globálního kontextu aplikace oddělen z důvodu plánovaného systému jednotného přihlašování do soutěží a korespondenčních seminářů ČVUT pro základní a střední školy. Tento návrh by měl po zveřejnění jednotného přihlašování usnadnit integraci nového systému s minimálním úsilím. Opět se jedná o zcela nezávislý modul, který je upraven speciálně pro účely soutěže.

Dalším modulem je publikování novinek a editace stránek. Konkrétně publikování novinek bylo odladěno pro jednoduché a rychlé používání. Tomuto modulu byla věnována největší pozornost v návrhu uživatelského rozhraní.

Posledním a největším implementovaným modulem je modul pro správu soutěže. Ten podporuje procesy spojené se zveřejňováním a odevzdáváním jednotlivých úkolů, stejně tak jako opravování a bodování odevzdaných řešení.

### 4.3.2 Konvence pro tvorbu modulů

Základními pravidly, které by měl programátor dodržovat, jsou konvence samotného Nette frameworku, které jsou důležité především ve vztahu k aktualizaci samotného jádra Nette. To by mělo mít za následek zvýšení pravděpodobnosti kompatibility s novějšími verzemi frameworku.

Rozšíření třetích stran by měla být umístěna v adresáři `vendors` a závislost uvedena v composeru. Proto by nemělo být nutné při instalaci dodávat všechny knihovny třetích stran, ale pouze konfigurace composeru, který se již o stažení potřebných knihoven postará sám. Samotné moduly by pak měly být v samotné aplikační složce `app` a každý modul by měl být reprezentován svojí složkou s (anglickým) názvem modulu a příponou `Module`. Tento název zároveň určuje namespace (jmenný prostor), ve kterém jsou jednotlivé třídy daného modulu implementovány.

#### 4.3.2.1 Poskytování vlastních služeb

Každý modul může poskytovat své služby dalším modulům, a to výhradně přes `services` (česky služby; pro rozlišení významu dále použito v anglické formě), které by měly být uloženy ve stejnojmenné složce. Každá `service` musí splňovat své neměnné (zpětně kompatibilní) rozhraní, se kterým počítají jiné moduly. Tyto `service` jsou identifikovány svým modulem a názvem skládající se z názvu poskytované služby a přípony `Service`. `Service` by neměla vykonávat žádnou náročnou logiku a měla by dodržovat návrhový vzor fasády (facade pattern), který se vyznačuje zaobalením volání konkrétních metod příslušného modulu.

Dále jednotlivé moduly poskytují rozhraní ke svým modelům, které reprezentují databázové tabulky, avšak pouze za účelem získání názvu tabulky či sloupce. Tato rozhraní by již neměla sloužit k získávání samotných dat z databáze. Data by se měla získávat pouze přes poskytované `services` (což by mělo sloužit i k většímu zabezpečení aplikace před nechtěným programátorovým pochybením, např. zasílání uživatelova hesla do modulu zajišťujícího odevzdávání úloh).

#### 4.3.2.2 Získávání cizích služeb

Každá část modulu může získat služby jiných modulů přes jejich `services`. Je však striktně zakázána cyklická vazba mezi moduly. O dodržení tohoto pravidla se musí postarat programátor sám, v případě cyklické vazby je nutné najít správné rozložení do 3 modulů, které již nebudou cyklicky závislé. Požadovat služby ze `service` může libovolná třída kromě modelů, které musí být modulově nezávislé. Modely by měly reprezentovat základní logické stavební kameny.

#### 4.3.2.3 Integrace modulů do aplikace

Integrace nového modulu do aplikace probíhá na několika úrovních, první úrovní integrace je společný kořenový presenter (`RootPresenter`), který obsahuje továrny na jednotlivé komponenty zobrazované na všech stránkách. Dále obsahuje také sdílené atributy a metody dostupné ve všech presenterech. Z toho důvodu je kořenový presenter v globálním kontextu a nepatří pod žádný modul.

Druhou úrovní integrace je zaregistrování vzorů URL adres a jejich cílů, které se definují v routeru, jenž dědí z abstraktní třídy `BaseRouter`. V této třídě je po něm vyžadována implementace metody `initialize`, kde pomocí metody `_addRoute` může zadefinovat vzory URL pro své presentery. Samotný router je pak nutné ještě uvést v konfiguračním souboru `config.neon` v sekci `services` a předat jej globálnímu routeru. Všechny presentery musí dědit ze společného kořenového presenteru `RootPresenter` (nebo jeho potomků).

```
UserModule ..................................... základní složka modulu
├── service
│   └── IUserService.php ..................poskytované rozhraní služeb
├── router
│   └── Router.php ..............................definice URL vzorů
└── services.neon ..................................... definice služeb
```

Obrázek 4.4: Vzorová struktura modulu

Třetí úrovní integrace je zásah do hlavní šablony a úprava rozložení všech stránek či stránky hlavní. Tyto úpravy se provádějí v presenteru hlavní stránky či kořenovém presenteru (v závislosti na působnosti změn). Dále v šablonách `Homepage/default.latte` a `@layout.latte`. Všechny přidávané komponenty by měly být získávány z poskytovaných služeb jednotlivých modulů.

Poslední úrovní integrace jsou služby a modely, které se definují v samostatných konfiguračních souborech `services.neon` v adresáři modulu. A tyto soubory jsou injektovány pomocí direktivy `includes` do konfiguračního souboru `config.neon`.

#### 4.3.2.4 Vzorová struktura modulu

Na dalším diagramu 4.4 je zobrazena vzorová struktura modulu, která je důležitá především při integraci nového modulu do projektu. Nedůležité části modulu (z hlediska integrace) byly vynechány.

Vnitřní struktura modulu je čistě na uvážení programátora a případné rozdělení modulu do menších submodulů je čistě v jeho režii. Základním požadavkem však je rozdělení na úrovni konceptuálního (doménového) modelu. V žádném případě by nemělo docházet k rozdělení na úrovni práv uživatele či přímo dokumentací frameworku Nette nabízené rozdělení na administrační a prezentační část [25].

Kapitola 5

# Uživatelské rozhraní

Vzhledem k faktu, že aktuální webová prezentace je nadále neudržovatelná (případně udržovatelná s velkými problémy), je nezbytné vytvořit nový grafický koncept, který bude podporovat příjemnou uživatelskou interakci se systémem. Pro zajištění kvalitního návrhu je vhodné sestavit persony organizátorů, jež budou využity pro odladění uživatelského rozhraní.

## 5.1 Persony

V našem kontextu autor identifikoval celkem 4 důležité persony, které jsou kritické pro návrh uživatelského rozhraní.

### 5.1.1 Bc. Radim Študák

Student na ČVUT, 24 let, hraje na foukací harmoniku. Jámu Lvovou již organizuje 4 roky. Jeho oblíbenými předměty na střední škole byly matematika a fyzika. Orientuje se v systému LaTeX, DokuWiki, SMF forum a OO Calc, které využívá k organizaci v rámci Jámy Lvové. Dále má základní znalosti algoritmizace (ve skriptovacím jazyku Python) a práce s Matlabem.

**Cíle** Mezi jeho základními cíli je správa soutěže, vytváření nových kol a úloh. Dále přidává oprávnění novým organizátorům a přidává účastníky, kteří nejsou zaregistrovaní.

### 5.1.2 Jana Soutěživá

Studentka střední školy v Praze se zaměřením na rozšířenou výuku matematiky, má ráda procházky a geocaching. Na základní škole se účastnila Jámy Lvové a jezdila na tábory. Po přechodu na střední školu ji byla nabídnuta účast v organizátorském týmu Jámy Lvové. Ze soutěžního prostředí má zkušenosti

s MS Wordem a SMF forum. Ze školního prostředí se na hodinách programování setkala se systémem Moodle. Do opravování ještě úplně nepronikla, ale má velice ráda český jazyk a především sloh. Na střední škole píše články do školního časopisu a velice ráda komunikuje s lidmi.

**Cíle** Její organizátorskou činností je navrhování úloh do soutěže a opravování řešení soutěžících. Dále také publikuje novinky na hlavní stránce.

### 5.1.3 Jaroslav Bílý

Žák 7. třídy základní školy, který na doporučení své učitelky matematiky začal soutěžit v Jámě Lvové. Matematika mu jde, ale zatím se nedostal na školní půdě k zajímavějším příkladům. Ve škole se na informatice učí práci se základními programy jako je malování, MS Word a MS Excel. Chová doma andulku a hraje na klavír.

**Cíle** Jeho hlavním cílem je odevzdání svých řešení a prohlédnutí si opravených řešení.

### 5.1.4 RNDr. Mgr. Lada Slavíková

Učitelka matematiky a fyziky na základní škole. Ve své třídě učí i Jardu Bílého, o kterém si myslí, že je nadaný chlapec a mohl by více procvičovat své matematické schopnosti. Účastní se šifrovaček a za své studenty skenuje jejich řešení a odevzdává je pomocí e-mailu. Zvládá základní práci na počítači, práci s vědeckými nástroji jako Matlab a základy programování v Pascalu a Pythonu.

**Cíle** Její hlavní cíl je odevzdávání řešení za své studenty, ráda by také sledovala statistiky svých studentů.

## 5.2 Vzhled aplikace

Tato příručka by měla sloužit jako zjednodušená verze HIG (Human Interface Guidelines) popisující, jak upravovat či vytvářet další části aplikace tak, aby pro uživatele bylo snadnější pochopení nové části a aby aplikace svým dojmem působila celistvě.

### 5.2.1 Základní prvky

V celém projektu autor klade důraz na jednotné prvky v globálním i lokálním měřítku. Základním prvkem celého designu je panel, který se skládá z hlavičky a těla. Hlavička musí být graficky významně oddělena od těla panelu. Všechny texty a ovládací prvky musí být zarovnány doleva (vzhledem k indoevropské

jazykové rodině). Pravá část by měla být určena pro speciální ovládací prvky (viz níže).

Všechna tlačítka menu musí vypadat stejně. Odkazy musí být napsány modrým textem (v případě tmavého pozadí bílým textem) a musí být podtrženy. E-mailové adresy budou před samotnou adresou obsahovat ikonku obálky. Externí odkazy budou obsahovat před samotnou adresou ikonku označující odchod ze stránky pryč. Všechny ikonky musí mít společný styl. (Z toho důvodu nejsou některá předchozí pravidla v prototypu dodržována, jelikož grafická úprava není cílem autorovy bakalářské práce.) Všechna tlačítka a ovládací prvky musejí být buď prvky menu nebo vypadat jako tlačítka. Tlačítka mohou mít různé barvy, dle jejich významu, podobně jako informační boxy.

### 5.2.2 Celkový vzhled aplikace

Celá stránka je rozložena na hlavičku obsahující menu a tělo stránky. V levé části menu jsou obsaženy odkazy na veřejně přístupné statické stránky. Pravá část menu je určena pro administrační část v rámci daného kontextu stránky. Kontext stránek je určen jejich modulem, do kterého patří.

V oblasti těla stránek se nachází hlavní obsah. V levé části se pak nachází navigační menu jednotlivých modulů. V pravé části (případně uprostřed) se nachází obsah stránky. Obsah stránky je opět pojat jako panel, tudíž je rozdělen na hlavičku (název stránky) a samotný text.

### 5.2.3 Rozvržení stránky

Každá stránka musí být rozdělena na hlavičku a tělo. V levé části hlavičky musí být vždy název stránky. V pravé části hlavičky mohou být speciální ovládací prvky související s kontextem stránky (např. přidání nové položky).

Tělo stránky by mělo být vyplněno strukturovaným textem, především pomocí odstavců či tabulek. V případě potřeby je možno použít jiný koncept dle vlastního uvážení (příkladem je zobrazování úloh či archiv soutěží).

#### 5.2.3.1 Tabulky

Tabulky by měly dodržovat podobná pravidla jako panely. Pro lepší přehlednost je doporučeno použití proužkovaného stylu pro odlišení řádků. Důraz je kladen na čistý vzhled aplikace, a tak by ohraničení u tabulek mělo být využíváno jen výjimečně. Nejdůležitější informace by opět měly být co nejvíce vlevo, vpravo se pak mohou nacházet ovládací prvky.

#### 5.2.3.2 Upozornění

Upozornění se musí zobrazovat v horní části stránky (nebo formuláře) a musí být reakcí na uživatelovu aktivitu. Nesmí se jednat o statickou informaci (vý-

jimku tvoří varovná upozornění). Upozornění můžeme rozdělit do 4 kategorií (dle vzrůstajícího významu):

- informativní,
- úspěšná akce,
- varování,
- nebezpečí.

Informativní upozornění by mělo být reakcí na běžné akce, které vyžadují zpětnou reakci, protože jinak by uživatel mohl být zmaten tím, že se na stránce nic nezměnilo. Text upozornění může být delší (ale neměl by překročit 80 znaků, přibližně jeden řádek). Doporučená barva je modrá a ikonka písmene „i“.

Upozornění o úspěšné akci by mělo být reakcí na různé změny v databázi, které by mělo sloužit jako potvrzení o akci, a text by měl obsahovat jaká akce se provedla. Délka textu by měla být opět přibližně jeden řádek. Doporučená barva je zelená a ikonkou je typická „fajfka“.

Varování by mělo upozorňovat uživatele na nepovedenou akci, jejíž důsledky nejsou kritické pro dosažení uživatelova cíle. Například odstranění již odstraněného záznamu v databázi. Doporučené barvy jsou žlutá či oranžová a ikonkou je vykřičník v trojúhelníku.

Upozornění typu nebezpečí by mělo být použito pouze v případě, kdy nezdar akce pro uživatele znamená kritický problém (např. nepodařilo se nahrát na server řešení), dále se běžně používá pro chybu formuláře (validační pravidla). Text zprávy by měl být krátký a výstižný, je povoleno použít i silný řez písma pro zvýraznění. Barva musí být také výrazná, proto je doporučená červená doplněná o ikonku. Typická ikonka by měla být červený vykřičník v kolečku nebo značka zákaz vjezdu v jednom směru.

## 5.3 Přístupnost

Přísné dodržování pravidel přístupnosti je doporučované, především z toho důvodu, že soutěž je určená pro všechny žáky základních škol a odpovídajících ročníků víceletých gymnázií. Některé ze základních vlastností podporujících dobrou přístupnost jsou vyžadované (v prototypu však implementované nejsou, kvůli nutnosti vytvoření kompletního grafického vzhledu). Nemusí se jednat pouze o přístupnost pro handicapované soutěžící, ale také pro uživatele mobilních zařízení.

### 5.3.1 Validita kódu a WCAG 2.0

Validita HTML a CSS kódu zajišťuje větší kompatibilitu mezi jednotlivými prohlížeči. Je proto nezbytné, aby výsledná aplikace byla validní. Dále by aplikace měla splňovat doporučení WCAG 2.0 (Web Content Accessibility Guidelines; doporučení pro přístupnost webového obsahu), alespoň na úrovni AA.

### 5.3.2 Barevnost

Použité barvy by měly být dostatečně kontrastní, ideální je použití černé a bílé barvy (případně odstíny šedé). Dále je doporučováno použít maximálně dvě další barvy. Prvním zajímavým konceptem je použít univerzitní modrou barvu, s kterou by pak soutěž reprezentovala univerzitu. Druhou možností je použít aktuální červenou barvu, protože jsou na ní uživatelé zvyklí. Bohužel při protanopii postihující až 1% mužské populace se právě červená barva jeví jako žlutá [30]. Proto je dobré doplnit všechna červená upozornění (nebezpečí) o správnou ikonku rozpoznatelnou i bez barevného rozlišení.

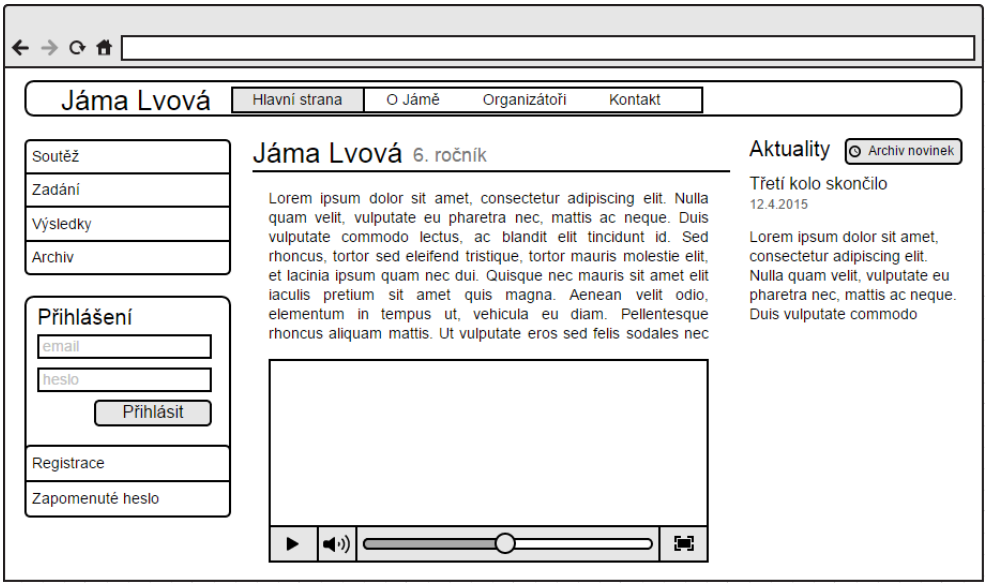
### 5.3.3 JavaScript

Použití JavaScriptu vzhledem k přístupnosti (a použitelnosti) je považováno za pouhý doplněk, který by měl usnadňovat práci s aplikací. Ale samotná práce by se měla bez JavaScriptu obejít, a to i za cenu horší použitelnosti. Je proto nutné, aby všechny panely s možností skrytí obsahu byly načteny při zobrazení stránky, ale skryty pouze s použitím JavaScriptu. Případným alternativním řešením může být vytvoření stránky pro rozkrytí každého jednotlivého panelu, což může být velice nepraktické a náročné.

### 5.3.4 Responzivní vzhled

Cílem aplikace je samozřejmě umožnit zobrazení na tabletech a dalších mobilních zařízeních. Díky tomu, že většina z nich má integrovaný fotoaparát, mohou sloužit tablety či mobilní telefony jako skener. Tudíž může tato technologie usnadnit soutěžícím převod papírové verze do digitální podoby. To však vyžaduje dostatečnou kvalitu příslušného fotoaparátu. Nejen z tohoto důvodu je nutné přizpůsobit vzhled stránek právě pro tato zařízení.

Je doporučeno zobrazit nejprve primární menu, následně jednotlivé navigační panely jiných modulů a poté obsah stránky. Navigační panely by tak měly mít výchozí nastavení takové, aby na mobilních zařízeních byly skryté a po kliknutí na hlavičku se rozbalily. Odkazy by měly přímo odkazovat na začátek obsahu stránky. Tato funkčnost může být zajištěna například pomocí JavaScriptu.




Obrázek 5.1: Drátový model

## 5.4 Drátový model

Pro ukázku základního principu výše uvedeného návrhu uživatelského rozhraní autor vytvořil drátový model (diagram 5.1) pro hlavní stránku aplikace. Na tomto modelu byly vysvětleny základní principy prototypu účastníkům uživatelského testování (podrobně v kapitole 5.5)

## 5.5 Testování prototypu

Uživatelské testování prototypu autor provedl se dvěma dobrovolníky, kteří přibližně odpovídali personě Janě Soutěživé (viz 5.1.2). Při testování se procházely následující dva scénáře:

- Chcete přidat novinku na hlavní stránku. Novinka by měla být o spuštění nového kola, proto chcete, aby se tato novinka zobrazila až za 3 dny. Zkontrolujte, zda je novinka v pořádku. Všimli jste si překlepu a rádi byste jej opravili.

- Právě jste vyřešili a naskenovali úlohu, rádi byste tuto úlohu nahráli do odevzdávátka. Zkontrolujte na výsledkové listině, že se odevzdání podařilo.

Nad personou Jany Soutěživé, jakožto organizátorky, autor testoval i odevzdávání úloh, protože Jana odpovídá personě, která dříve byla účastníkem.

### 5.5.1 Vyhodnocení

Po seznámení s principy uživatelského rozhraní se uživatelé v prototypu poměrně snadno orientovali. Přesto, že u prvního úkolu každý z obou testerů zvolil jiný postup (viz níže), tak oba dosáhli svého cíle bez sebemenších problémů. Volba delšího postupu může být zapříčeněna tím, že přidané tlačítko je drobné a může být zastíněno tlačítkem vedle, proto bylo na základě testu zvýrazněno. V následujícím seznamu jsou znázorněny dva různé postupy.

- Rychlé tlačítko plus na hlavní stránce.
- Odkaz novinky v pravém menu a poté kliknutí na tlačítko Nová novinka.

Menší problém nastal u druhého úkolu, kdy testeři dostali k nahrání soubor typu Word s příponou `.doc`, přičemž uživatelům se vypsala pouze chybová hláška: „Špatný formát souboru“. Po informování testera, že systém povoluje pouze soubory typu PDF, a po chvíli přemýšlení, oba testeři zjistili, že na počítači je nainstalována PDF tiskárna a tak problém vyřešili.

**Řešení** Tento problém s uživatelským rozhraním byl vyřešen nahrazením chybové hlášky za specifickou hlášku informující o požadovaném typu a s doporučením na použití PDF tiskárny. Oba testeři se shodli na tom, že tato informace by jim v danou chvíli pomohla vyřešit problém.

# Závěr

Na závěr této práce by autor rád připomenul nejdůležitější poznatky. Dále by rád zhodnotil jednotlivé splněné úkoly a navrhl možná rozšíření a budoucí vývoj projektu.

Vzhledem k praktickému zaměření práce byly voleny spíše technologie, které zvyšují budoucí rozšiřitelnost a udržitelnost aplikace. Některé problémy nejsou řešeny implementačně, ale jsou analyzovány na procesní úrovni. Naopak některé problémy v procesech jsou podpořeny vhodným modulem.

## Zhodnocení

V kapitole 2 byly identifikovány kritické procesy, na nichž jsou demonstrovány slabiny stávajícího řešení. Mezi ty patří především velká orientovanost na organizátora a grafický návrh, který neumožnuje rozvoj projektu. V rámci analýzy byl vytvořen i doménový model, který se stal základem pro návrh databázového schématu. Výsledné schéma bylo mírně zjednodušeno pro snazší práci se SQL a některé méně významné vazby byly vynechány.

V části návrhu byly porovnány programovací jazyky a dle srovnání byl vybrán nejvhodnější jazyk pro implementaci. V tomto bodě se autor shodl s již existujícím prototypem, ale rozhodl se pro volbu jiného frameworku. Praktická část se skládala z implementace povinných modulů a dokumentace pro tvorbu dalších subsystémů. Při implementaci byl kladen důraz na modulární architekturu. Poslední kapitola je klíčovou pro návrh uživatelského rozhraní, popisuje jednotlivé prvky a popis uživatelského testování aplikace dle sestavených person.

Vzhledem k detailní procesní analýze a doménovému modelu, byly v projektu implementovány další subsystémy nad rámec zadání. Mezi ně patří možnost dynamické tvorby webových stránek. Systém pro snadné zveřejnění a archivaci kol i jednotlivých úloh. Dále zde můžeme nalézt i jednoduchý systém na správu uživatelů.

Nad rámec dokumentace tvorby modulů byla v rámci bakalářské práce vytvořena i doporučení pro maximální zabezpečení budoucí aplikace a pro grafickou úpravu modulů s cílem vytvořit uživatelské rozhraní více intuitivní a konzistentní.

## Budoucí vývoj

Implementovaný prototyp se zaměřoval především na systém pro publikování novinek a evidenci došlých řešení. S tím, že bylo podpořeno aktivní zapojení soutěžících do procesu odevzdávání. Projekt stále nabízí možnost několika podstatných rozšíření.

- Jak ukazuje persona reprezentující učitele na základní škole, bylo by vhodné vytvořit rozhraní, které by učitelům umožnilo odevzdávat řešení svých studentů. Pro tuto funkcionalitu je již připravena podobná struktura, která pomáhá organizátorům nahrávat do webové aplikace řešení nezaregistrovaných soutěžících.
- Velmi cenným modulem, který v této aplikaci zatím chybí, je systém pro správu soutěžících. Ten by měl obsahovat systém pro komunikaci se soutěžícími prostřednictvím e-mailové korespondence (hromadné odesílání zpráv), ale i možnost generování papírových dopisů i s adresami soutěžících. Pro získání poštovních a e-mailových adres je zatím nutné přistupovat do SQL databáze.
- Dalšími podpůrnými subsystémy můžou být subsystémy pro dynamickou tvorbu menu, získávání statistik, organizaci práce a další drobné moduly pro jednodušší editaci textu. Pro dokončení vytvořeného prototypu je potřeba vytvořit kompletně nový grafický návrh.

# Literatura


[1] Jáma Lvová, ČVUT. *Pravidla soutěže* [online]. [cit. 2015-04-29]. Dostupné z: https://jamalvova.cz/soutez

[2] Hadrava, J. *KSP: Informační systém* [online]. [cit. 2014-11-24]. Osobní komunikace. Dostupné z: https://drive.google.com/file/d/0BwlhqffNQosyM2pNTDBfcEY2N28/view?usp=sharing

[3] Koblížek, T. *Information System Supporting Project Jama Lvova.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2013.

[4] Simple Machines. *Simple Machines Forum - Free & open source community software* [online]. [cit. 2015-05-02]. Dostupné z: http://www.simplemachines.org/

[5] Gohr, A. *DokuWiki* [online]. [cit. 2015-05-02]. Dostupné z: https://www.dokuwiki.org/dokuwiki

[6] Mitchell, S. *Easy Wiki Hosting, Scott Hanselman's blog, and Snagging Screens. MSDN Magazine* [online], 2008, [cit. 2015-05-02]. Dostupné z: https://msdn.microsoft.com/en-us/magazine/cc700339.aspx

[7] solid IT. *DB-Engines Ranking* [online]. [cit. 2015-04-30]. Dostupné z: http://db-engines.com/en/ranking

[8] Valoušek, M. *Django Česká republika* [online]. [cit. 2015-04-30]. Dostupné z: http://www.djangoproject.cz/

[9] Javorek, J. *Python, programovací jazyk* [online]. [cit. 2015-04-30]. Dostupné z: http://python.cz/

[10] Václavík, J. *Perl (1) - Dávka teorie na úvod. Linuxsoft.cz* [online], 2005, [cit. 2015-04-30]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=675

[11] Mach, J. *PHP pro úplné začátečníky*. Brno: Computer Press, druhé vydání, 2005, ISBN 80-7226-834-1, 167 s.

[12] TIOBE Software. *TIOBE Index for April 2015* [online]. [cit. 2015-04-26]. Dostupné z: `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`

[13] BuiltWith® Pty Ltd. *Framework Usage Statistics* [online]. [cit. 2015-04-30]. Dostupné z: `http://trends.builtwith.com/framework`

[14] sogrady. *The RedMonk Programming Language Rankings: January 2015. RedMonk* [online], 2015, [cit. 2015-05-05]. Dostupné z: `http://redmonk.com/sogrady/2015/01/14/language-rankings-1-15/`

[15] Elizabeth Naramore, a. s. *PHP5, MySQL, Apache: Vytváříme webové aplikace*. Brno: Computer Press, první vydání, 2006, ISBN 80-251-1073-7, 813 s.

[16] Pecinovský, R. *Java 8: Úvod do objektové architektury pro mírně pokročilé*. Praha: Grada, první vydání, 2014, ISBN 978-80-247-4638-8, 655 s.

[17] SensioLab. *What is Symfony* [online]. [cit. 2015-04-15]. Dostupné z: `http://symfony.com/what-is-symfony`

[18] Cake Software Foundation, Inc. *CakePHP at a Glance* [online]. [cit. 2015-04-15]. Dostupné z: `http://book.cakephp.org/3.0/en/intro.html`

[19] Nette Foundation. *Seznámení s Nette Frameworkem* [online]. [cit. 2015-03-27]. Dostupné z: `http://doc.nette.org/cs/2.3/getting-started`

[20] Skvorc, B. *Best PHP Framework for 2015 – SitePoint Survey Results. SitePoint* [online], 2015, [cit. 2015-04-05]. Dostupné z: `http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/`

[21] Daněk, P. *Velký test PHP frameworků: Zend, Nette, PHP a RoR. Root.cz* [online], 2008, [cit. 2015-03-27]. Dostupné z: `http://www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/`

[22] Nette Foundation. *Composer* [online]. [cit. 2015-04-07]. Dostupné z: `http://doc.nette.org/cs/2.3/composer`

[23] Gerchev, I. *The 5 Most Popular Frontend Frameworks of 2014 Compared. SitePoint* [online], 2014, [cit. 2015-04-04]. Dostupné z: `http://www.sitepoint.com/5-most-popular-frontend-frameworks-compared/`

[24] Otto, M. *Bootstrap* [online]. Bootstrap Core Team, [cit. 2015-02-14]. Dostupné z: `http://getbootstrap.com/`

[25] Nette Foundation. *MVC aplikace & presentery* [online]. [cit. 2015-04-07]. Dostupné z: `http://doc.nette.org/cs/2.3/presenters#toc-moduly`

[26] Bernard, B. *Úvod do architektury MVC. Zdroják.cz* [online], 2009, [cit. 2015-04-24]. Dostupné z: `http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/`

[27] Mička, P. *Simple factory method. Algoritmy.net* [online], [cit. 2015-05-03]. Dostupné z: `http://www.algoritmy.net/article/1339/Simple-factory-method`

[28] The OWASP Foundation. *OWASP Top 10 – 2013* [online]. 2013, [cit. 2015-04-29]. Dostupné z: `https://www.owasp.org/images/f/f3/OWASP_Top_10_-_2013_Final_-_Czech_V1.1.pdf`

[29] Nette Foundation. *Slovníček pojmů* [online]. [cit. 2015-05-02]. Dostupné z: `http://doc.nette.org/cs/2.3/glossary`

[30] Šikl, R. *Zrakové vnímání*. Praha: Grada, první vydání, 2012, ISBN 978-80-247-3029-5, 312 s.

Příloha **A**

# Seznam použitých zkratek

**AMP** Apache, MySQL, PHP

**CSRF** Cross-Site Request Forgery

**CSS** Cascading Style Sheets

**FIKS** Fiťácký informatický korespondenční seminář

**HIG** Human Interface Guidelines

**HTML** Hypertext Markup Language

**HTTP** Hypertext Transfer Protocol

**HTTPS** HTTP Secure

**JS** JavaScript

**KSP** Korespondenční seminář z programování

**MS** Microsoft

**MVC** Model-View-Controller

**OO** Open Office

**OWASP** Open Web Application Security Project

**PDF** Portable Document Format

**PHP** PHP: Hypertext Preprocessor

**PNG** Portable Network Graphics

**RAD** Rapid Application Development

**SMF** Simple Machines Forum

**SSL** Secure Sockets Layer

**URL** Uniform Resource Locator

**WCAG** Web Content Accessibility Guidelines

**WWW** World Wide Web

**XSS** Cross-Site Scripting

Příloha B

# Instalační příručka

Pro spuštění některých příkazů je potřeba oprávnění správce systému. Při zprovozňování aplikace dodržujte bezpečnostní pravidla popsaná v kapitole 4.2.

Pro lokální testování nemusí být žádoucí zapnuté HTTPS. Aplikace jej ve výchozím stavu vyžaduje, aby zabránila možnému bezpečnostnímu riziku. Zakomentováním řádku č. 28

```
Route::$defaultFlags = Route::SECURED;
```

v souboru `app/router/RouterFactory.php` se dá vypnout automatické přesměrování na HTTPS.

A) Instalace databáze

1. Přesvěčte se, že cílová databáze je prázdná.
2. Při použití databázového serveru jiného než MySQL bude potřeba následující skript poupravit.
3. Na databázovém serveru spusťte inicializační skript, který slouží k vytvoření databázových tabulek. Naleznete ho v adresáři `src/scripts`.

B) Instalace webové aplikace

1. Zkopírujte zdrojové kódy aplikace z adresáře `src/impl` do adresáře webhostingu.
2. V případě že webhosting neumožňuje přenastavit cílový zobrazovaný adresář na `www`.
   i. Změňte strukturu podle schématu uvedeného v kapitole 4.2.2.3.
   ii. V souboru `www/index.php` změňte cestu k inicializačnímu souboru `bootstrap.php`.
3. Pokud jste nekopírovali složku `vendors`.

i. Stáhněte nástroj `composer` z webu `https://getcomposer.org/`.
ii. Přesuňte se do složky se souborem `composer.json` a spusťte příkaz `composer update`.

4. Ověřte, že aplikace má právo zapisovat do složek `data`, `log` a `temp` (a všech podsložek).

C) Konfigurace webové aplikace

1. Otevřete si konfigurační soubor `config.local.neon`.
2. Nastavte připojení k databázovému serveru v sekci `database`.

   **dsn:** cíl připojení (výchozí `mysql, localhost, jamalvovadb`)
   **user:** přihlašovací jméno (výchozí `jamalvovauser`)
   **password:** přihlašovací heslo (výchozí `jamalvovapass`)
3. Nastavte připojení k SMTP serveru v sekci `nette:mailer`.

   **host:** adresa SMTP serveru (výchozí `smtp.server.cz`)
   **port:** port SMTP serveru (výchozí `25`)
   **username:** přihlašovací jméno (výchozí `smtpuser`)
   **password:** přihlašovací heslo (výchozí `smtppass`)

D) Přihlašovací údaje

**e-mail** admin@localhost.cz

**heslo** 12345678

Pokud vše proběhlo v pořádku, aplikace je připravená ke spuštění a Vašemu prvnímu přihlášení.

Příloha C

# Obsah přiloženého CD

- /
  - src
    - diagrams ..... diagramy
    - impl ..... zdrojové kódy prototypu
    - scripts
      - create.sql ..... inicializační skript pro databázi
      - schema.pdf ..... schéma databáze
    - thesis ..... zdrojová forma práce ve formátu LaTeX
  - thesis
    - zadani.pdf ..... zadání práce ve formátu PDF
    - thesis.pdf ..... text práce ve formátu PDF
  - readme.txt ..... popis obsahu CD