VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PODPORA POŘÁDÁNÍ FESTIVALOVÝCH SOUTĚŽÍ

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MIROSLAV PAVELEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. ZBYNĚK KŘIVKA, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19317/2016/xpavel31

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Pavelek Miroslav**
Obor: Informační technologie
Téma: **Podpora pořádání festivalových soutěží**
Support for Organization of Festival Contests
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s organizací soutěží na festivalu Animefest a požadavky organizátorů pro vytvoření podpůrného informačního systému.
2. Prozkoumejte vhodné technologie pro implementaci těchto požadavků.
3. Po konzultacích s vedoucím proveďte návrh daného systému, který na zvolených technologiích implementujte.
4. V dostatečném předstihu systém uživatelsky otestujte se zapojením organizátorů soutěží. Proveďte sběr a vyhodnocení zpětné vazby. Podstatné připomínky realizujte do finální verze systému.

Literatura:

- interní dokumentace k organizaci minulých ročníků soutěží
- Nigel George: *Mastering Django: Core: The Complete Guide to Django 1.8 LTS*. GNW Independent Publishing, 2016, 646 s.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně bod 3.

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Křivka Zbyněk, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá vytvořením podpůrného informačního systému pro organizaci soutěží na festivalu Animefest. Jedná se o webovou aplikaci postavenou na aplikačním rámci Django napsaném v jazyce Python. Systém nahradí aktuální webové řešení a zároveň přinese nové funkce. Organizátorům festivalu umožní vytvářet nové soutěže a spravovat je spolu s přihláškami účastníků, soutěžícím pak registrovat se do soutěží a spravovat své přihlášky.

Abstract

This bachelor thesis is focused on making supporting information system for organization of contests on Animefest festival. It is a web application based on framework Django written in Python language. The system will replace the current web solution and will have new functionalities. For system administrator it provides the creationg and managing new contests and also for managing entries. Contestants will have the possibility to register to the contest and manage their entries.

Klíčová slova

web, informační systém, festival, soutěž, HTML, CSS, Python, Django, Bootstrap

Keywords

web, information system, festival, contest, HTML, CSS, Python, Django, Bootstrap

Citace

PAVELEK, Miroslav. *Podpora pořádání festivalových soutěží.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zbyněk Křivka, Ph.D.

Podpora pořádání festivalových soutěží

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zbyňka Křivky, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Miroslav Pavelek
17. května 2017

Poděkování

Děkuji panu Ing. Zbyňkovi Křivkovi, Ph.D., za cenné rady, odborný dohled a vedení mé bakalářské práce, a RNDr. Adamu Rambouskovi, Ph.D., který je ředitelem Animefestu, za spolupráci při návrhu a testování webové aplikace.

Obsah

1 Úvod **3**

2 Analýza požadavků **4**
2.1 Uživatelské role 4
2.1.1 Diagram případu užití 5
2.1.2 Vedoucí týmu 5
2.1.3 Člen soutěžního týmu 5
2.1.4 Správce soutěže 6
2.1.5 Pomocník správce 6
2.1.6 Porotce 6
2.1.7 Administrátor 6

3 Použité technologie **7**
3.1 Analýza webových aplikačních rámců 7
3.1.1 Požadované vlastnosti 7
3.1.2 Prvotní analýza 7
3.1.3 Podrobnější analýza 8
3.2 Popis použitých technologií 9
3.2.1 Python 10
3.2.2 Django 10
3.2.3 MySQL 10
3.2.4 HTML, CSS 11
3.2.5 JavaScript 11
3.2.6 Bootstrap 11

4 Návrh informačního systému **13**
4.1 Návrh databáze 13
4.2 Koncept 14
4.2.1 Statické stránky s informacemi 15
4.2.2 Seznam soutěží 15
4.2.3 Detail soutěže 15
4.2.4 Formuláře 16
4.2.5 Administrace soutěže 16
4.2.6 Hlavní administrace 17

5 Implementace **18**
5.1 Serverová část 18
5.1.1 Databáze 18

5.1.2 Adresářová struktura 19
5.1.3 Uživatelské oprávnění 20
5.1.4 Formuláře 22
5.1.5 Práce s JSON 23
5.1.6 Překlad dynamických položek 23
5.2 Klientská část 24
5.2.1 Šablonovací jazyk 24
5.2.2 Grafické rozhraní webové aplikace 26

6 Testování **31**
6.1 Uživatelské testování 32
6.1.1 Soutěžící 32
6.1.2 Správci soutěže 32

7 Závěr **33**

Literatura **34**

A Obsah CD **35**

Kapitola 1

Úvod

Hlavním cílem bakalářské práce je vytvoření nového podpůrného informačního systému pro organizaci soutěží na festivalu Animefest, jenž je pořádán zapsaným spolkem Brněnští otaku. Organizátorům bude umožňovat vytváření nových soutěží, jejich správu a následně kontrolu přihlášek od soutěžících. V průběhu soutěží budou soutěžící moci upravovat zadané údaje v přihlášce.

Aktuálně používaný systém běží na systému pro správu obsahu Drupal, který je velice robustní a není tak moc ohebný v nově vznikajících požadavcích na organizaci soutěží. Aktuální řešení by tak bylo složité přizpůsobit moderním webovým standardům, jako například responzivnosti nebo vícejazyčnosti i u dynamického obsahu.

Obsah práce je rozdělen do několika kapitol, které popisují tvorbu aplikace od jejího návrhu přes implementaci až k jejímu otestování a následnému nasazení. Na první kapitolu, kterou představuje tento úvod, navazuje druhá kapitola, jejímž úkolem je seznámit čtenáře s požadavky na systém ve formě analýzy požadavků. Ta zahrnuje jak konkrétní požadavky na funkčnost, tak i požadované uživatelské role. Následující kapitola se zabývá popisem využitých technologií při tvorbě a implementaci informačního systému. Ve čtvrté a páté kapitole je popsána architektura systému včetně implementace a v další kapitole je popsáno testovaní systému, zpětná vazba od uživatelů a kroky, které byly učiněny ke zlepšení systému. Následuje závěr zahrnující návrhy na další možný rozvoj a vylepšení.

Kapitola 2

Analýza požadavků

Animefest je každoroční setkání fanoušků japonského komiksu (*mangy*) a animovaného filmu (*anime*). Každý ročník obsahuje jak přednášky, tak i praktické workshopy a promítání nových i starších anime. Kromě toho se každý rok koná i několik soutěží, jako například soutěž o co nejvěrnější ztvárnění postavy nebo soutěž o nejlepší taneční scénku.

Právě na základě konzultací s ředitelem festivalu a dalšími organizátory vznikly požadavky na webovou aplikaci, která by měla sloužit k organizaci zmiňovaných soutěží. Do aplikace mají mít přístup jak samotní soutěžící, tak i administrátoři a správci soutěží, kteří zde mohou kontrolovat přihlášky. Každá soutěž je složena z několika parametrů – název, termíny, správce, popis, počet míst, formulář s přihlášením a kontaktní formulář na správce.

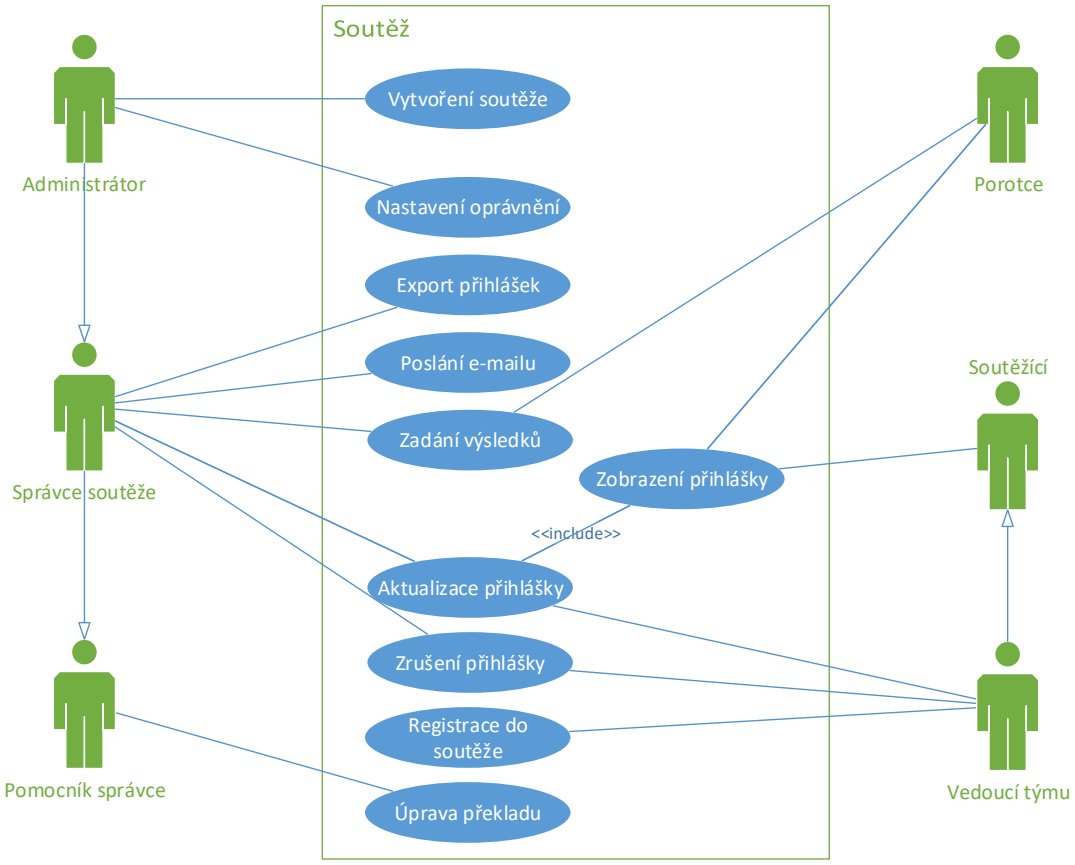
Dále organizátoři požadují podporu vícejazyčnosti, konkrétně podporu zobrazení textu ve více jazycích a překladu textů přímo z uživatelského rozhraní aplikace. Výsledky soutěže by měly být dostupné i několik ročníků zpětně, což tedy znamená nutnost zajištění dlouhodobého ukládání dat.

2.1 Uživatelské role

Zpřístupněné funkce systému se budou odvíjet od skupin uživatelů (rolí), do kterých náleží konkrétní uživatelské účty, a od konkrétních soutěží. Identifikace uživatele probíhá pomocí e-mailové adresy a hesla. Účet lze rovněž propojit pomocí účtu na Facebooku nebo Googlu. Systém bude obsahovat dohromady šest různých rolí – vedoucí týmu, člen soutěžního týmu, pomocník správce, správce soutěže, porotce a administrátor.

2.1.1 Diagram případu užití

Na diagramu případu užití na obrázku 2.1 je znázorněno všech šest uživatelských rolí a jejich možné akce. V dalších podkapitolách jsou tyto role podrobně rozepsány.



Obrázek 2.1: Diagram případu užití.

2.1.2 Vedoucí týmu

Jakýkoli uživatel se může účastnit soutěží. Přihlášení do nich spočívá ve vyplnění formuláře, v němž je potřeba vyplnit povinné položky. Až do data uzavření přihlášek má vedoucí týmu možnost si formulář prohlížet a aktualizovat jej – jak o textové údaje, tak i o nové soubory. Pokud se jedná o skupinovou soutěž, tak má vedoucí možnost uvést účty ostatních členů týmu.

2.1.3 Člen soutěžního týmu

Ve skupinových soutěžích vedoucí týmu do přihlášky uvádí i ostatní členy týmu. Ti se po přihlášení do systému mohou podívat na aktuální stav přihlášky svého týmu.

2.1.4 Správce soutěže

Každá soutěž bude mít své správce, které nastaví administrátor. Soutěž nemá limitovaný počet správců, stejně tak jeden uživatel může být správcem několika soutěží zároveň. Správce má k dispozici seznam všech přihlášených soutěžících a může přistupovat k jednotlivým přihlašovacím formulářům (včetně souborů), má možnost je upravovat a po soutěži může do systému zadat výsledné body. Formuláře i výsledky soutěží lze z této role exportovat do PDF. Dále může nakonfigurovat SMTP server pro odesílání e-mailů, jenž může hromadně posílat všem soutěžícím, například aby je upozornil na blížící se termín uzávěrky.

2.1.5 Pomocník správce

Dále je zde ještě role pomocníka správce, který má na starosti pouze překlad do cizích jazyků z administrátorského rozhraní aplikace.

2.1.6 Porotce

Poslední uživatelskou rolí je porotce. Porotce se vždy váže na příslušnou soutěž, u které si může prohlédnout seznam všech soutěžících.

2.1.7 Administrátor

Administrátor musí mít možnost provádět nad webovou aplikací veškeré možné operace ve všech existujících soutěžích.

Kapitola 3

Použité technologie

Vzhledem k široké cílově skupině je jedním z cílů informačního systému, aby fungoval nezávisle na operačním systému. Proto je IS implementován formou webové aplikace, která ke svému fungování ze strany uživatele vyžaduje pouze webový prohlížeč s podporou JavaScriptu.

3.1 Analýza webových aplikačních rámců

Jako první bylo třeba zanalyzovat, jaké technologie se pro dané požadavky hodí nejvíce. Situace na klientské části byla vcelku jasná, nicméně u serverové části bylo třeba provést analýzu. Většina programovacích jazyků sama o sobě neobsahuje dostatečný počet funkcí a knihoven pro programování webových aplikací takového typu, tyto funkcionality bývají až v aplikačních rámcích. Z tohoto důvodu byla provedena analýza dostupných webových aplikačních rámců v několika různých programovacích jazycích.

3.1.1 Požadované vlastnosti

Jako první bylo třeba určit hlavní požadavky, které musí aplikační rámec splňovat. Vzhledem k analýze jsme se rozhodli, že vybraný aplikační rámec musí obsahovat systém pro správu uživatelů, ACL (*access control list*), popř. podobně robustní systém na správu oprávnění, vícejazyčnost a podporu formulářů, ideálně i možnost pro jejich jednoduchou tvorbu pomocí GUI. Rovněž by zde měl být i nějaký vestavěný systém na překlad do jiného jazyka.

3.1.2 Prvotní analýza

Pro prvotní analýzu bylo vybráno celkem devět aplikačních rámců z pěti různých programovacích jazyků. Na tabulce 3.1 jsou uvedeny jednotlivé aplikační rámce spolu s komentářem, zda splňují požadované vlastnosti.

Název	Jazyk	Správa účtů	Správa oprávnění	Formuláře	Překlad
Symfony	PHP	ano	ano	ano	ano
Laravel	PHP	ano	ano	ne	ano
Nette	PHP	ano	ano	ano	ano
Django	Python	ano	ano	ano	ano
Flask	Python	ne	ne	ne	ne
Ruby on Rails	Ruby	ne	ne	ne	ano
Express	Node.js	ne	ne	ne	ne
Revel	Go	ano	ne	ne	ne

Tabulka 3.1: Přehled aplikačních rámců.

Z tabulky je patrné, že aplikační rámce *Flask*, *Ruby on Rails*, *Express* a *Revel* vůbec nevyhovují. Důvod je ten, že dané aplikační rámce se soustředí především na vysokou rychlost a nenáročnost, kvůli čemuž nemají prostor nabízet funkcionalitu navíc. *Laravel* ve většině ohledů vyhovoval, nicméně komponenta pro formuláře byla vyhozena z jádra aplikačního rámce a nadále je udržována komunitou, nelze tedy spoléhat na její další vývoj a podporu, proto byla vyloučena z analýzy. Do užšího výběru se tak dostaly rámce *Symfony*, *Nette* a *Django*.

3.1.3 Podrobnější analýza

V této podkapitole se podíváme na tři aplikační rámce, které se dostaly do užšího výběru.

Symfony

Jako první jsem si vyzkoušel práci v aplikačním rámci *Symfony* [10]. Konkrétně jsem se rozhodl pro verzi 3, jež byla relativně nová a nabízela velké změny oproti verzi 2. Nicméně již při zkoušení tutoriálů dostupných na oficiálních stránkách aplikačního rámce jsem narazil na několik problémů.

Prvním problémem byla zastaralá oficiální dokumentace. I v dokumentaci pro verzi 3 byly určité části, které byly stále určené pro verzi 2. Dalším problémem bylo to, že i přímo na oficiálních stránkách jsou zmíněny takzvané *Bundles* (balíčky), které mají dále rozšiřovat funkcionalitu. Nicméně tyto balíčky pochází od komunity okolo *Symfony* a nejsou pravidelně udržovány.

Další možností bylo vyzkoušet *Symfony* verze 2.8, která byla označena jako LTS (*Long term support*), tedy jako verze s dlouhodobou podporou, v tomto případě až do roku 2018. Nicméně poté by byl přechod na vyšší verzi složitý, proto jsem zavrhl i tuto možnost.

Nette

Nette je český aplikační rámec řadící se do skupiny robustnějších systému a díky svému původu patří mezi nejpoužívanější rámec u nás. Již tak na první pohled bylo patrné, že *Nette* má v našich končinách rozsáhlou a aktivní komunitu, dokonce je dostupná oficiální

dokumentace i v češtině.

Nicméně stejně jako v případě *Symfony*, qhi zde není úplně propracovaná, a tak již při seznamování se s tímto systémem bylo několik nejasností týkajících se funkčnosti. Největší potíž s dokumentací se ukázala u samotného balíčku na dynamickou tvorbu formulářů, který byl špatně dokumentovaný a od začátku s ním byly problémy.

Django

Ve svém jádru je aplikační rámec *Django* zjednodušeně kolekce knihoven napsaných v programovacím jazyce Python. *Django* volně následuje návrhový vzor MVC (*Model-view-controller*), nicméně využívá svoji vlastní logiku pro implementaci. Komponenta *controller* (řadič) je obstarána samotným jádrem *Djanga* a většina věcí probíhá v modelech, šablonách a pohledech, proto se *Django* často nazývá jako aplikační rámec MTV. MTV je v knize *Mastering Django: Core: The Complete Guide to Django 1.8 LTS*[3] popsán jako:

- **M jako „Model“**
  Jedná se o reprezentaci datové vrstvy, obsahuje tedy vše týkající se samotných dat. Složena je z bázové třídy Model, která zajišťuje automatickou konverzi dat mezi datovým modelem a relační databází. Tato technika se nazývá *objektově relační mapování* (ORM).
- **T jako „Template“**
  Template (česky *šablona*) tvoří prezentační vrstvu, jedná se o řetězec textu, jehož účelem je oddělit vzhled dokumentu od samotných dat. Šablony definují značky, které určují, jak se mají data zobrazovat uživatelům.
- **V jako „View“**
  View (česky *pohled*) pokrývá aplikační vrstvu, jedná se o spojovací prvek mezi modely a šablonami. V zásadě se jedná o funkce, které zpracovávají HTTP[1] požadavky a vrací HTTP odpověď.

Django se řídí takzvaným principem DRY[1] – „Don't repeat yourself“, což lze přeložit jako „Neopakuj se“. To znamená, že každý koncept nebo část kódu by se nikdy neměl objevovat vícekrát, je tedy kladen důraz na reprodukovatelnost kódu. Toho si můžeme povšimnout například na modulovém systému v *Djangu*. Každá aplikace postavená na rámci *Django* je složena z několika modulů, které jsou v *Djangu* nazývány jako aplikace. Každá aplikace má vlastní rozhraní a může být použitelná i napříč projekty.

Při testování funkcionalit samotného aplikačního rámce jsem nenarazil na žádné větší problémy, naopak jsem byl překvapen již vestavěnou funkcionalitou (např. základní webovou administrací) a zaujaly mě principy i dokumentace aplikačního rámce. Proto byl tento aplikační rámec použit k vývoji webové aplikace.

3.2 Popis použitých technologií

V této podkapitole jsou popsány technologie, které byly ve výsledku při implementaci použity. Jako první jsou popsány technologie na serverové části, poté technologie na klientské části.

[1] https://www.nginx.com/resources/glossary/http/

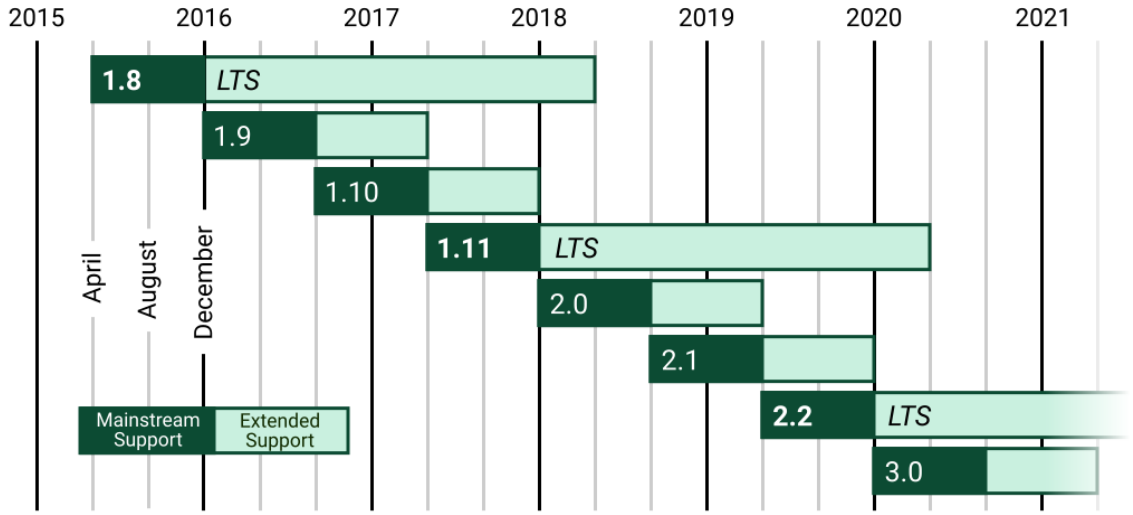
3.2.1 Python

Serverová část (back end) je implementovaná v jazyce Python. Jak je napsáno v knize *Learning Python* [6] od Marka Lutze, tak Python lze označit za univerzální programovací jazyk, který spojuje procedurální, funkcionální a objektově orientovaná paradigmata.

V dnešní době je používán jak Python ve verzi 2, tak i ve verzi 3, která s verzí 2 není zpětně kompatibilní. K implementaci byla použita verze Pythonu verze 3, která sice ještě nemá takové zázemí jako verze 2, ale většina komunity okolo Pythonu se zaměřuje již na verzi 3. Navíc verze 3 má lepší předpoklady do budoucna, neboť většina lidí se postupně zaměřuje spíše na tuto verzi.

3.2.2 Django

Jak již bylo řečeno v kapitole 3.1.3, tak k implementaci byl zvolen aplikační rámec *Django*. Konkrétně byla použita verze 1.11 LTS.



Obrázek 3.1: Diagram označující plánované vydání verzí *Djanga*, zdroj: https://www.djangoproject.com/download/.

Tabulka 3.1 znázorňuje, že verze 1.11 vyšla vyšla v první polovině roku 2017 a do konce roku má potvrzený *mainstream support*, což zahrnuje bezpečnostní záplaty, opravy týkající se možné ztráty dat a samozřejmě i opravu chyb v nových funkcionalitách. Díky tomu, že se jedná o LTS, tak má na rozdíl od ostatních verzí místo jednoho roku dva roky *extended support*, který pokrývá už pouze bezpečnostní záplaty.

3.2.3 MySQL

Všechna uživatelské data, jako například údaje o uživatelích nebo přihlášky do soutěží, je nutné ukládat na serveru. K tomu v této webové aplikaci slouží MySQL, což je databázový systém aktuálně spravovaný společností Oracle Corporation. Komunikace s databází probíhá pomocí rozšířeného jazyka SQL. Jelikož se jedná o multiplatformní a volně šiřitelný software, má v současné době vysoký podíl v používaných databázích.

3.2.4 HTML, CSS

Klientská část (front end) je implementována pomocí webových šablon, které jsou součástí *Djanga*. Nicméně v nich se kromě pro *Django* specifických tagů objevují ve velké míře i jazyky HTML (*Hyper Text Markup Language*) a CSS (*Cascading Style Sheets*).

HTML je značkovací jazyk určený pro popis webových stránek pomocí značek (tzv. tagů) a jejich vlastností (atributů). Pomocí nich určujeme sémantiku (význam) dokumentu. Dvojice tagů, které jsou uzavřeny do úhlových závorek (<, >), tvoří element. Dříve se jazyk HTML používal i pro definování vzhledu dokumentu, nicméně časem tuto funkci přebralo CSS.

Jak již z předchozího odstavce vyplývá, tak CSS slouží k určení vzhledu dokumentu. CSS umožňuje pro každou HTML značku definovat, jak bude příslušný element vypadat pomocí takzvaných pravidel. Každé pravidlo se skládá ze selektoru (element, pro který definujeme pravidla) a z bloku deklarací, v němž již určujeme hodnoty jednotlivých vlastností. Výhodou je, že jeden CSS soubor jde použít pro více HTML souborů, což ušetří hodně času. [8]

3.2.5 JavaScript

HTML a CSS zajišťují hlavně statické zobrazování obsahu, nicméně aby byla webová aplikace uživatelsky přívětivá, je třeba část obsahu validovat nebo zobrazovat dynamicky na straně klienta. K tomu slouží JavaScript a v něm vytvořený aplikační rámec jQuery.

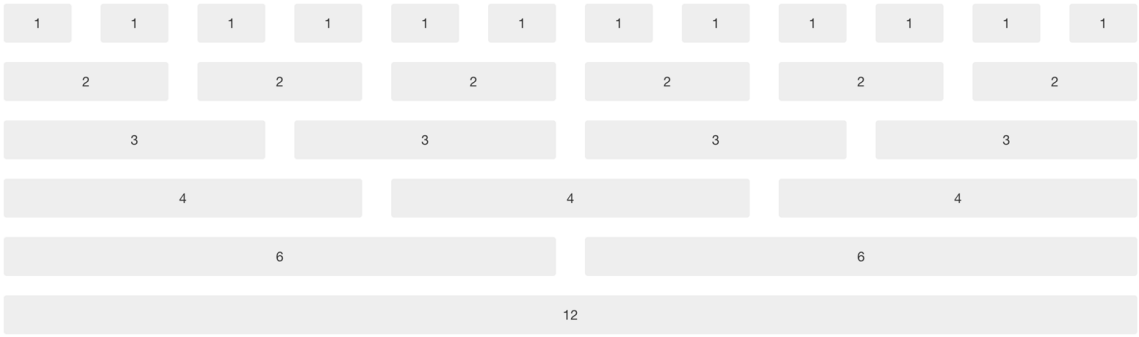
JavaScript je skriptovací, prototypově založený jazyk, který běží na straně klienta. Je tedy nutné, aby ho klientův prohlížeč podporoval a měl jej povolený. To se na první pohled zdá jako komplikace, nicméně vzhledem k tomu, že JavaScript dnes pohání většinu webových aplikací, není to v praxi problémem.

V současné době se používá především pro různé interaktivní prvky uživatelského rozhraní, jako například tlačítka nebo textová pole. Kromě toho se používá k prvotní validaci dat v textových polí, např. pokud uživatel zapomene v adrese e-mailu uvést „@“, tak jej pomocí JavaScriptu lze okamžitě upozornit.

Aplikační rámec jQuery je vyvíjen v JavaScriptu a jeho hlavním účelem je zjednodušit práci s JavaScriptem. Kromě toho automaticky řeší nekompatibilitu mezi prohlížeči a je multiplatformní. Existují i aplikační rámce (např. Node.js), které umožňují JavaScript používat i v serverové části, nicméně toto použití nebylo v této webové aplikaci vhodné. Důvod je ten, že JavaScriptové aplikační rámce se soustředí především na vysokou rychlost a flexibilitu, což vede k malé základní funkcionalitě.

3.2.6 Bootstrap

Aby nebylo nutné vytvářet CSS kompletně od nuly a znovu tak „vynalézat kolo“, byla využita knihovna Bootstrap. Jak je popsáno v knize *Bootstrap* od Jake Spurlocka[9], Bootstrap vznikl v roce 2011 původně jako čistě CSS knihovna, která se ale časem díky popularitě rozšířila i o JavaScriptové pluginy a ikony, které vylepšují vzhled a chování formulářů a tlačítek obecně.



Obrázek 3.2: Bootstrap mřížka[9].

Mezi hlavní výhody knihovny Bootstrap patří systém mřížky a i s tím související responzivity. Na obrázku 3.2 je znázorněno, jak Bootstrap dělí obsah stránky na 12 sloupců, se kterými jde pracovat jak ve skupinách, tak i jednotlivě. Tento systém velice usnadňuje pozicování v rámci stránek, protože obsah lze rozdělit až na dvanáct stejně velkých sloupců.

Navíc jejich velikost není pevně daná v pixelech, ale přizpůsobuje se velikosti obrazovky, což zajišťuje ekvivalentní zobrazení na počítačích a mobilních zařízeních. Pokud je náhodou rozlišení koncového zařízení moc malé, jsou některé sloupce umístěny pod sebe.

Bootstrap byl použit i při vytváření formulářů[2], které jsou díky němu rovněž responzivní a vypadají lépe a moderněji, než kdyby nebyly nastylované vůbec. Navíc díky velkému rozšíření Bootstrapu je rozšířené i toho stylovaní, takže i pro uživatelé je to relativně známé prostředí.

[2] http://bootstrapdocs.com/v3.3.6/docs/css/#forms

Kapitola 4

Návrh informačního systému

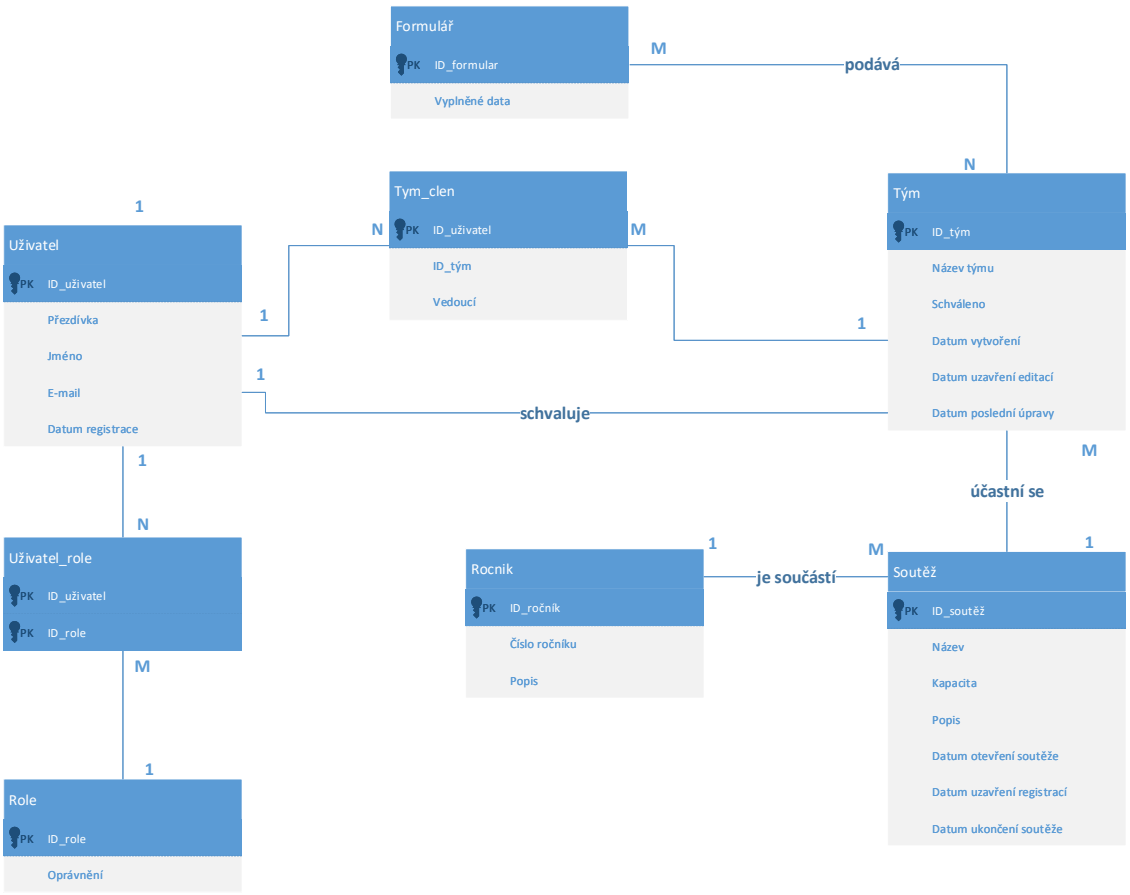
V knize *Database Systems: A Pragmatic Approach*[2] se uvádí, že při vytváření informačního systému s relačním databázovým systémem je třeba projít následujícími kroky:

1. Identifikovat entity.
2. Identifikovat vztahy.
3. Eliminovat nepotřebné vztahy.
4. Vytvořit entity-relationship diagram (ERD), object-relationship diagram(ORD) nebo jiný ekvivalentní model.
5. Normalizovat databázi.
6. Revidovat ERD, ORD nebo použitý ekvivalentní model.
7. Navrhnout uživatelské rozhraní.
8. Přesunout se k implementaci.

Autoři dále uvádí, že entita je objekt, koncept nebo věc, o které jsou uloženy data. Vztah je základní zobrazení zahrnující dvě nebo více entit. Obě tyto části jsme si částečně vymezili již v kapitole 2, tato kapitola se tedy bude dále zaobírat výsledkem z kroků 3 až 6 v podobě ER diagramu a poté zde bude letmo nastíněn krok 7, tedy návrh uživatelského rozhraní.

4.1 Návrh databáze

ERD (*entity-relationship diagram*, česky entitně vztahový diagram) slouží pro konceptuální znázornění ukládání dat do databáze. Jak již samotný význam zkratky napovídá, tak diagram popisuje abstraktní zobrazení databáze pomocí entit a vztahů mezi nimi. Diagram modelované aplikace lze vidět na obrázku 4.1.



Obrázek 4.1: Konceptuální model

4.2 Koncept

V této sekci je popsán návrh výsledného řešení a je nastíněno i celkové rozložení jednotlivých stránek. To zahrnuje i návaznost stránek, které by měl uživatel postupně navštívit.

Samotný systém se drží již dříve zmíněného principu DRY, tedy co nejmenšího opakování. To se týká jak serverové části, tak i klientské části – každá stránka by měla obsahovat unikátní informace. Samotný obsah webu lze rozdělit do pěti částí:

- statické stránky s informacemi,
- seznam soutěží,
- detail soutěže,
- formuláře,

- administrace soutěže,
- vestavěna administrace.

4.2.1 Statické stránky s informacemi

Na stránkách jsou obsaženy tři statické stránky – úvodní stránka, stránka s FAQ (*Frequently asked question*, česky často kladené otázky) a stránka s kontakty. Na úvodní stránku je uživatel implicitně nasměrován po zadání webové adresy soutěže a posléze se k ní znovu dostane prokliknutím loga stránek. Stejně tak i stránky s FAQ a kontakty jsou dostupné z horizontálního hlavního menu, které je umístěno v horní části stránky.

Každá ze stránek je editovatelná administrátorem, a to dvěma způsoby. Prvním způsobem je úprava pomocí WYSIWYG (*What you see is what you get*, česky *co vidíš, to dostaneš*) editoru, který slouží k jednoduchému a přehlednému formátování textu. Druhou možností je úprava samotného HTML v případě potřeby komplikovanějších úprav.

4.2.2 Seznam soutěží

V seznamu soutěží může každý uživatel vidět aktuální seznam jemu dostupných soutěží. Seznam není reprezentován tradiční textovou podobou, ale je v podobě obrázků. Každá soutěž má svůj náhledový obrázek, které po kliknutí směruje na detaily soutěže. Jelikož se nepočítá s velkým množstvím dostupných soutěží (aktuálně maximálně tři), je tento systém i dostatečně přehledný již na první pohled.

Existují tři situace, kdy je soutěž viditelná pro uživatele:

- Jsou otevřené registrace.
- Uživatel se soutěže účastní nebo účastnil.
- Uživatel chce soutěž spravovat.

Původně bylo zamýšleno tyto tři situace graficky odlišit v seznamu soutěží, nicméně nakonec se ukázalo, že by to snižovalo čitelnost samotného seznamu. Rovněž přidaná hodnota by nebyla tak výrazná – sami uživatelé vědí, proč se do jaké soutěže hlásí a není třeba jim to nijak připomínat.

4.2.3 Detail soutěže

Detail soutěže obsahuje jak samotné informace o soutěži, tak i přístup k přihlášce, respektive k jejímu vytvoření. Dále slouží ke vstupu do administrace konkrétní soutěže. Popis je zde zobrazen dvěmi formami. První z nich je obecný popis soutěže, který slouží spíše jako úvod a ukazuje se na stránce za všech okolností. Dále je zde popis v podobě sekcí (například sekce přihlášení, kostýmy, porota), kdy je vždy jedna aktivní a na další se lze přepnout.

Důvod několika sekcí je takový, že celkový popis soutěže je poměrně dlouhý a nechat uživatele srolovat celými pravidly by tedy nebylo moc uživatelsky přívětivé. Počet sekcí není pevně daný, správce soutěže může kdykoliv nějakou sekci vymazat nebo naopak přidat novou.

4.2.4 Formuláře

Formuláře jsou nejdůležitější součástí celého projektu. Pomocí formulářů se vytváří a upravují přihlášky obsahující informace o jednotlivých soutěžících a soutěžních týmech. Mezi hlavní požadavky na informační systém patřila možnost přihlášky editovat, a to za těchto podmínek:

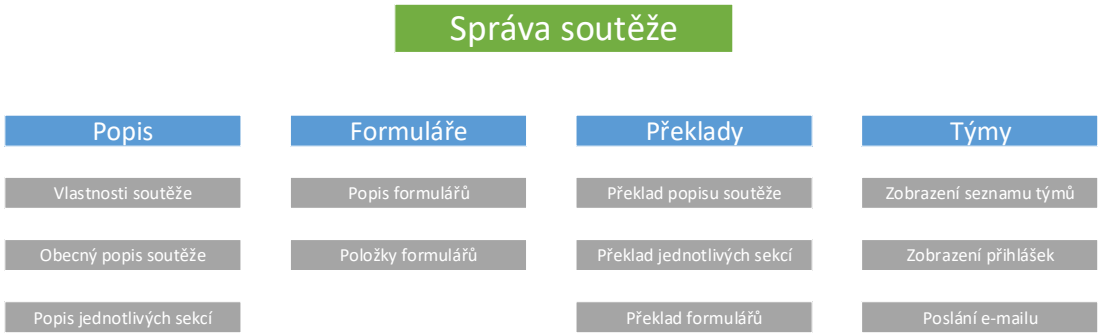
- pokud jsou otevřené registrace,
- v průběhu soutěže, pokud to je povolené v administraci soutěže.

Kromě toho jde u samotné editace nastavit, které pole lze upravovat. Tato možnost je implementována z důvodu, aby již nebylo možné změnit některé údaje o soutěžících, jako například jméno a příjmení, ale přitom stále existovala možnost, jak do formuláře přidat například aktuální fotografie kostýmu. Ve formulářích existují tyto položky:

- standardní položky:
  - text na jeden řádek,
  - text na více řádků,
  - výběr z položek,
  - přepínač,
  - celé číslo,
  - datum,
  - e-mail,
  - obrázek,
  - soubor,
  - zaškrtávací.
- speciální položky,
- zobrazení:
  - nadpis části.

4.2.5 Administrace soutěže

V sekci 4.2.3 bylo zmíněno, že do administrace soutěže se lze dostat z detailů soutěže. Konkrétně v sekci Administrace se na základě uživatelských práv pro danou soutěž nachází tlačítka, které vedou na jednotlivé části administrace, která je zobrazena na obrázku 4.2. Asistenti správců soutěže mají přístup do části *Překlady*, ve které lze zadat překlad pro formuláře a popis soutěže včetně jednotlivých sekcí. Porotci mají přístup do části *Týmy*, která slouží k prohlédnutí jednotlivých přihlášek dostupných v systému a k přiřazení ohodnocení. Správci soutěže mají přístup do všech částí administrace, tedy i do části *Správa soutěže*, která slouží k nastavení popisu soutěže, vytvoření formulářů a jejich přiřazení k soutěži.



Obrázek 4.2: Jednotlivé části správy soutěže.

Samotné vytváření formulářů je intuitivní, po přidání položky je k dispozici náhled, jak aktuálně formulář vypadá. U každé položky lze kromě názvu nastavit například i pomocný text nebo předvyplněné hodnoty. Pořadí položek se mění pomocí přetahování myší.

Administrace nabízí velké množství položek, se kterými lze pracovat i v rámci jednotlivých částí, proto bylo potřeba zamyslet se nad jejich zobrazením. Pokud má tedy nějaká část například pět podsekcí, tak každá z nich má vlastní rozevírací panel; implicitně jsou všechny panely zavřené.

4.2.6 Hlavní administrace

Již při analýze v části 2 bylo zjištěno, že *Django* obsahuje možnost vytvoření základní administrace, ve které lze upravovat jednotlivé instance objektů. Tato administrace by měla být dostupná pouze administrátorům stránek, neboť z této části stránek je možné editovat celý web a pro nastavování oprávnění není tato administrace navržena.

Kapitola 5

Implementace

V této kapitole se budeme zabývat samotnou implementací, tedy praktickým uskutečněním výše uvedených požadavků a návrhů. Jako první bude popsána implementace serverové části, na niž byl použit především aplikační rámec *Django*, a poté bude popsána klientská část, která se opírá o knihovnu Bootstrap.

5.1 Serverová část

Tato podkapitola se zabývá popisem implementace serverové části webové aplikace. Jako první je zde popsána práce s databází, poté adresářová struktura. Nakonec je zdůrazněno několik zajímavých částí serverové implementace, jako například práce s formuláři nebo zajištění překladu dynamických položek.

5.1.1 Databáze

Výsledné schéma databáze se od původního návrhu v kapitole 4 liší. Z důvodu rozšíření funkcionality o prvky, které byly v předcházející kapitole popsány pouze abstraktně, bylo zapotřebí přidat další tabulky a vazby. Jedná se například o možnost přihlašování pomocí sociální sítě Facebook nebo e-mailové adresy od Googlu, jež vyžaduje další tabulky, jednu pro samotné poskytovatele a další pro uložené tokeny konkrétních uživatelů. Několik tabulek potřebovalo i samotné *Django* ke své funkčnosti, jako například tabulku pro relace (*sessions*).

Samotné definování databáze probíhá v komponentě Model pomocí tříd. Implementace třídy pro soutěžní týmy je znázorněna v ukázce zdrojového kódu 5.1.

Zdrojový kód 5.1: Třída pro soutěžní tým.

```
1 class ContestTeam(models.Model):
2         name = models.CharField(_('Team Name'), max_length=100)
3         group = models.ForeignKey(Group)
4         contest = models.ForeignKey(Contest)
5         rating = models.PositiveIntegerField(_('Team rating'), default=0)
6         created = models.DateTimeField(auto_now_add=True)
```

V kódu 5.1 je ukázáno, jak vypadá definování modelu v kódu, a na tabulce 5.1 je znázorněn její databázový ekvivalent. Pro každou proměnnou uvedené třídy byl tedy vytvořen vlastní sloupec a spolu se sloupcem *id*, který se automaticky inkrementuje a který má na starosti unikátní označení jednotlivých položek. Za zmínku stojí i to, že ve sloupcích *contest_id* a *group_id* jsou uložena čísla (*id*) příslušných řádků z jiných tabulek, slouží tedy jako cizí klíč.

Název sloupce	**Typ sloupce**
id	int(11)
name	varchar(100)
group_id	int(11)
contest_id	int(11)
rating	int(10) unsigned
created	datetime(6)

Tabulka 5.1: Sloupce databázové tabulky pro soutěžní tým.

5.1.2 Adresářová struktura

Adresářová struktura webové aplikace se odvíjí od základní struktury definované aplikačním rámcem *Django*, nicméně bylo zde provedeno několik úprav. V adresáři s aplikací se nachází jeden soubor a devět podadresářů.

- ./manage.py
  Skript *manage.py* je vygenerován *Djangem* při vytváření nového projektu. Mezi jeho základní úlohy patří tvorba základní struktury aplikace, vytvoření databázového schématu a spuštění testovacího webového serveru.

- ./web
  Jedná se o adresář s globálním nastavením projektu. Patří zde například soubor *settings.py*, v němž jsou uloženy použité moduly, cesta k webovým šablonám apod. Dále je zde uložen soubor *urls.py*, jenž na základě regulárního výrazu přiřazuje k URL funkce, které mají zpracovat daný požadavek.

- ./contests
  Zde se nachází jádro celého projektu. V terminologii *Djanga* se jedná o aplikaci (v terminologii Pythonu balíček), která implementuje logiku soutěží.

- ./forms
  Tento adresář obsahuje další tři podsložky, které slouží k úpravě anebo rozšíření jednotlivých funkcionalit použitého balíčku *django-fobi*. První z nich se zaměřuje na samotné ukládání odeslaných přihlášek a jejich další úpravu, druhý na jednotlivé pole formuláře a třetí na grafický návrh klientské části.

- ./pages
  Tento adresář obsahuje aplikaci *pages*, obsahuje tedy pohledy, modely atd. jako ostatní aplikace. Účelem této aplikace je správa statických stránek na webu.

- ./locale
  Tento adresář obsahuje soubory užívané u vícejazyčnosti. Pro každý jazyk, do kterého je webová aplikace přeložena, je zde jeden podadresář, jenž obsahuje podadresář

LC_MESSAGES. Tento podadresář obsahuje textový soubor *django.po* s překlady a z toho zkompilovaný binární soubor *django.mo*, s kterým pracuje samotný aplikační rámec.

- ./templates
  Tento adresář obsahuje webové šablony (v terminologii *Djanga* nazývané *templates*), které určují, jak bude vypadat klientská část. Je zde umístěn soubor *base.html*, jenž obsahuje hlavní šablonu webu, tedy horní menu, blok pro obsah a patičku. Z této šablony poté dědí strukturu ostatní šablony, které jsou umístěny v dalších podadresářích.

- ./templatestags
  Samotné webové šablony mohou obsahovat uživatelsky definované značky. Ty jsou definované v tomto adresáři, který zároveň splňuje roli balíčku. V podsložce se stejným názvem lze najít tři soubory, z nichž každý definuje určitý typ značek.

- ./static
  Do tohoto adresáře jsou po spuštění skriptu *manage.py* s argumentem *collectstatic* vloženy všechny statické soubory (HTML a CSS) ze všech aplikací zahrnutých v projektu. V pojetí *Djanga* se pod pojmem statické soubory rozumí soubory, které patří k projektu, ale nejedná se o žádný kód nebo skript. Prakticky jsou zde tedy umístěny soubory CSS a JS, fonty a ikony.

- ./media
  Tento adresář je zaměřen na média, v našem případě tedy obrázky a videa. Obsahuje obrázky, které jsou součástí vzhledu webu, obsah vložený uživateli k jednotlivým přihláškám. Tyto soubory jsou dále organizovány do různých podadresářů podle účelu a podle typu, přičemž jsou zvlášť rozděleny obrázky a jiné typy souborů.

5.1.3 Uživatelské oprávnění

Pro správu přístupových práv jednotlivých uživatelů či rolí je v projektu využit systém, který je součástí aplikačního rámce *Django*. Ten obsahuje většinu potřebných tříd a metod pro autentizaci (určení identity uživatele, např. přihlášení) i pro samotnou autorizaci (např. kontrola práv).

Uživatelé a jejich hesla

Jádrem uživatelské autentizace je třída *django.contrib.auth.models.User*[1]. Objekty této třídy reprezentují jednotlivé uživatele pracující s webovou aplikací. Jedná se o jedinou třídu, která popisuje uživatele jako takového – rozdíl mezi administrátorem a obyčejným uživatelem je pouze v nastavení speciálních atributů. V systému je každý uživatel jednoznačně identifikován podle svého e-mailu a uživatelského jména. Samozřejmostí je, že každý uživatel má vlastní heslo.

Hesla z důvodu bezpečnosti nejsou v databázi ukládaná jako text, ale jako hash. *Django* využívá algoritmus PBKDF21[2] s SHA256 hashem. V případě odcizení databáze k přístupu k datům tedy nestačí její samotné vlastnění, ale je nutné prolomit šifrování.

[1] https://docs.djangoproject.com/en/1.11/ref/contrib/auth/#django.contrib.auth.models.User
[2] https://en.wikipedia.org/wiki/PBKDF2

Skupiny

Ke generickému kategorizování uživatelů slouží třída *django.contrib.auth.models.Group*[3]. V jedné skupině může být více uživatelů a zároveň jeden uživatel může být ve více skupinách. Na tyto skupiny lze aplikovat jak oprávnění, tak i nějaké jiné označení. Ve webové aplikaci jsou skupiny využity pro určení účastníků soutěže, i ke složení samotných týmů.

Oprávnění

Django obsahuje základní systém pro správu oprávnění, což prakticky znamená, že umožňuje přiřadit oprávnění specifickým uživatelům a skupinám uživatelů. V základu jsou u každé třídy objektů tři různé oprávnění:

- Add – možnost přidat nový objekt,
- Change – změnit obsah objektů,
- Delete – možnost smazat objekt.

V případě potřeby je k dispozici možnost přidat další oprávnění. V projektu je například u několika objektů přidáno oprávnění *View*, které slouží k prohlížení objektů. Nicméně i toto řešení je nedostačující, vztahuje se vždy na celou třídu objektů, ale v rámci projektu je potřebné řešit individuální oprávnění nad konkrétními instancemi objektů, tedy nad jednotlivými soutěžemi a i jednotlivými přihláškami.

Přihlašování a registrace

Django nabízí metody na správu přihlašování a registrací, ale bylo nutné k nim vytvořit jednotlivé pohledy a šablony pro zobrazení. Dále bylo třeba dodělat napojení na účty z jiných služeb, konkrétně na sociální síť Facebook a e-mailovou adresu od Googlu. Obě služby umožňují limitovaný přístup k uživatelským účtům přes HTTP službu pomocí autorizačního aplikačního rámce OAuth 2.

Jako první bylo třeba na obou službách registrovat svoji webovou aplikaci pro dvě umístění – jednou na localhost, kde byl systém vyvíjen, a podruhé na doménu, kde byl projekt umístěn. Po registraci bylo aplikaci přiděleno Client ID a Client Secret. Client ID je veřejně umístěný řetězec, podle kterého API služby identifikuje webovou aplikaci a podle kterého zároveň vytvoří autorizační URL, jež je posléze prezentována uživateli. Client Secret je používán k prokázání identity webové aplikace u API služby v momentě, kdy aplikace vyžaduje přístup k učtu uživatele; musí zůstat utajené mezi aplikací a API.

OAuth 2 nebyl v aplikaci integrován ručně, byla k tomu využita knihovna *django-allauth*[4], která integruje právě tento aplikační rámec a kromě toho poskytuje i vzorové pohledy a šablony pro registraci a přihlášení uživatele, ze kterých bylo v rámci implementace vycházeno. Navíc je v rámci této knihovny přidáno i posílání verifikačních e-mailů po registraci, což bylo s mírnými úpravami použito.

[3] https://docs.djangoproject.com/en/1.11/ref/contrib/auth/#django.contrib.auth.models.Group
[4] https://github.com/pennersr/django-allauth

5.1.4 Formuláře

Klíčovou funkcionalitou projektu je práce s formuláři. Podporu formulářů obsahuje *Django* již v základu, nicméně bylo třeba vyřešit dynamické generování formulářů včetně zobrazení v klientské aplikaci. K tomu byl použit balíček *fobi*[5], jenž má tuto funkcionalitu implementovanou a byl napsán s dostatečně velkou abstrakcí, která umožňovala pozdější úpravy pro potřeby projektu. Formuláře jsou reprezentovány modelem *FormEntry* a jsou uloženy v databázi v tabulce *fobi_formentry*.

Formuláře se skládají z jednotlivých polí. Kromě polí uvedených v kapitole 4.2.4 bylo nutné přidat ještě dvě speciální položky – výběr uživatele a název týmu. Důvodem přidání první položky byla potřeba mít konkrétně určené pole, ze kterého se budou bráni další účastníci týmu. U druhé položky je důvod téměř totožný – jedná se sice o obyčejné textové pole, ale je to unikátní typ položky, se kterým se pracuje i dále u určování názvu týmu.

Šablony pro pole formuláře

Jednotlivé šablony pro pole formuláře, jako například pole pro e-mail nebo pole pro soubor nejsou uloženy v databázi, ale jsou distribuované jako samostatné aplikace, jejichž metadata jsou obsažené vždy v souboru *fobi_form_elements.py*. Základními hodnotami jsou *uid* (unikátní identifikátor, jako například „email“ nebo „file“), zobrazovaný název, skupina pod kterou se řadí a třída, která popisuje, jak vypadá formulář pro nastavení pole.

Ve většině případů se konkrétní nastavitelné pole u formuláře skládalo z položek:

- interní unikátní název,
- označení,
- pomocný text,
- počáteční hodnota,
- zda je povinné položku vyplnit,
- vypnutí pole, tedy zrušení možností editace.

Položky konkrétních formulářů

Šablony pro jednotlivé položky jsou sice uloženy pouze v kódu v podobě aplikací, nicméně samotné položky již vytvořených formulářů jsou uloženy v databázi v tabulce *fobi_formelemententry*.

Ta v základu obsahuje sloupce *plugin_data* obsahující samotné data v JSONu, *plugin_uid* značící unikátní identifikaci šablony pole, *position*, jenž slouží k určení pořadí v rámci formuláře, a *form_fieldset_entry_id*, který standardně slouží k tvorbě skupin polí, ale v projektu není využit.

[5] https://github.com/barseghyanartur/django-fobi

5.1.5 Práce s JSON

Data položek konkrétních formulářů i samotné vyplněné formuláře jsou v databázi uložené ve formě JSONu. Jak se na oficiálních stránkách [4] píše, tak JSON (*JavaScript Object Notation*) je formát určený k výměně dat. Mezi jeho přednosti patří to, že je jednoduše čitelný i zapisovatelný člověkem a jednoduše zpracovatelný strojově. Vychází z podmnožiny programovacího jazyka JavaScript. Nicméně i přesto se jedná o textový, na jazyce zcela nezávislý formát. Široké podpory se dočkal i v samotném Pythonu[6], práce s ním tedy není komplikovaná. Pomocí příkazu *json.loads* je JSON načten do slovníku (*dictionary*), kde jej lze editovat standardním způsobem. Pro opětovný převod ze slovníku na JSON je použit příkaz *json.dumps*.

Nicméně i v tomto případě bylo třeba ošetřit výjimky, které mohly nastat. Pokud načítané pole z databáze bylo prázdné, bylo třeba odchytit výjimku *TypeError* a pracovat s prázdným slovníkem. Stejně tak je v projektu řešena i výjimka *JSONDecodeError*, která nastává, pokud JSON v databázi není validní. To by se prakticky nikdy nemělo stát, leda po ruční úpravě dat v databázi.

Dále se v souvislosti s JSON vyskytuje u slovníků výjimka *KeyError* značící, že daný klíč u daného slovníku nebyl nalezen. Jelikož různé pole formulářů nemají vždy stejné klíče, tak se tato chyba objevuje relativně často, nicméně ve všech případech je ošetřena. Nenalezený klíč je poté vytvořen a nastaven na prázdný řetězec.

5.1.6 Překlad dynamických položek

V projektu bylo nutné vyřešit problém překladu dynamických řetězců, mezi které patří například šablony formulářů, jejich jednotlivé položky a popisy soutěží. Knihovny zajišťující překlad dynamických dat většinou překlady do dalších jazyků ukládají do databáze. Zde existují dva možné přístupy – buď v databázi vytvořit novou tabulku na překlady, nebo naopak přidávat sloupce k tabulkám, ve kterých jsou obsažena původní data.

První přístup je sice čistší, neboť nejsou ovlivňována původní data, nicméně na druhou stranu s každým přeloženým řetězcem by bylo nutné vykonávat SQL klauzuli JOIN, která je výkonově pomerně náročná. Druhý přístup generuje pro každý přeložený řetězec sloupec v tabulce. To v případech, kdy je původní řetězec překládán do velkého množství jazyků (10+) může vytvářet velké množství sloupců a opět zbytečně zatěžovat databázi. Nicméně v rámci projektu je vyžadován překlad pouze do jednoho cizího jazyka a v budoucnu se nepočítá s velkými rozšířenými, proto byl zvolen druhý zmiňovaný přístup.

K tomu je využit balíček třetí strany s názvem *django-modeltranslation*[7]. Největší výhodou této knihovny je to, že přidání těchto překladů nemění původní modely, pouze k nim přidává nové položky. Stejně tak pohledy a další části kódu jsou beze změny. Provedení spočívá v tom, že je třeba vytvořit novou třídu, jenž dědí ze třídy *TranslationOptions* obsažené v modulu *translator*. Třída obsahuje pouze přiřazení do proměnné *fields*, v níž jsou sepsány pole, které mají být přeložitelné. Tuto třídu je posléze třeba registrovat do globálního jedináčku *translator*. Nakonec stačí vygenerovat a aplikovat novou migraci, která

[6] https://docs.python.org/3.4/library/json.html
[7] https://github.com/deschler/django-modeltranslation

vygeneruje na základě dostupných jazyků nové sloupce v databázi. Pokud by tedy byl do aplikace přidán nový jazyk, stačí přidat v souboru *settings.py* do proměnné *LANGUAGES* nový jazyk, vytvořit a aplikovat migraci a překlad do nového jazyka.

Samotné uživatelské rozhraní pro překlad vychází z *Django* formulářů, konkrétně z třídy *django.forms.ModelForm*, která na základě modelu sama generuje formulář. Zde až na vynechání původních polí a menší úpravy zpracování JSON dat nebyly provedeny žádné změny.

5.2 Klientská část

K tomu, aby byla webová aplikace použitelná, je třeba vytvořit i klientskou část, neboli front end. V průběhu vývoje serverové části byl front end vyvíjen pouze se základní šablonou Bootstrapu, neboť návrh v této části sloužil pouze pro představu. Ke konci vývoje, po dodání grafiky, byl integrován nový grafický návrh, který lze vidět na zachycených snímcích obrazovky 5.2, 5.3, 5.4 a 5.5.

5.2.1 Šablonovací jazyk

Při programování klientské části byl kromě standardní trojice jazyků a v nich napsaných aplikačních rámců (HTML, CSS a JavaScriptu) použit i *Django Template Language* (DTL)[8]. *Django* totiž potřebuje pohodlný způsob, jak generovat HTML dynamicky na základě aktuálního kontextu (hodnoty relevantních proměnných, například přihlášení uživatele, zvolený jazyk stránek atd.).

Nejčastější přístup k řešení je pomocí tzv. šablon. Šablony obsahují statické části požadovaného výstupu HTML (například značky `<h1>` a `</h1>` pro nadpis) a zároveň dynamický obsah pomocí speciální syntaxe (např. `{{ title }}`, který vypíše proměnnou s názvem *title*. *Django* podporuje několik různých šablonovacích jazyků, nicméně i podle dokumentace je doporučené zůstat u DTL, pokud neexistuje žádný důvod k jinému jazyku.

Syntax DTL

V šablonách existuje několik způsobů, jak pracovat s kontextem. V předchozí podkapitole je předveden příklad zobrazení proměnné, který se v šabloně zapíše jako `{{ proměnná }}`. Pomocí tečky je možné přistoupit k atributu dané proměnné, takže například `{{ soutez.nazev }}` vypíše obsah atributu *nazev* z proměnné *soutez*. Kromě toho lze s těmito proměnnými pracovat pomocí takzvaných filtrů, takže například filtr *length* v kódu `{{ prihlasky|length }}` způsobí to, že místo obsahu proměnné se vypíše počet položek v proměnné *prihlasky*. Je možné definovat i vlastní filtry.

Kromě práce s proměnnými je zde ještě jeden velký blok, a to vlastní značky (*tagy*), které v šablonách vypadají takto `{% tag %}`. Tagy jsou mnohem komplexnější než práce s proměnnými, některé vytvářejí text na výstup, některé načítají externí informace do šablony a s jejich pomocí je možné dělat i programovací konstrukce (podmínka, cyklus). Stejně jako u filtrů, i zde jde vytvářet vlastní sady tagů. Do šablony se následně přidají pomocí

[8] https://docs.djangoproject.com/en/1.11/ref/templates/language/

tagu `{% load nazev_balicku %}`.

Princip DRY popsaný v části 3.1.3 je aplikován i v šablonovacím systému v podobě dědičnosti šablon. To umožňuje vytvořit základní kostru, která obsahuje věci společné pro všechny prvky stránek a definuje *bloky*, jenž zděděné šablony mohou přepisovat. Díky tomu při správném návrhu se tedy zbytečně neopakuje kód. Bloky se definují pomocí tagů `{% block nazev %}` a `{% endblock nazev %}`. Šablony pak mohou dědit od jiné šablony pomocí tagu `{% extends cesta_k_sablone %}`.

Příklad DTL

Na příkladu 5.2 je znázorněno, jak může vypadat využití prvků popsaných v předchozích sekcích v praxi. Velice podobný kód se nachází i přímo ve webové aplikaci, nicméně zde byl z důvodu přehlednosti zkrácen.

Na prvním řádku je ukázáno nahrání rodičovské šablony, která obsahuje kostru webu. Druhý řádek znázorňuje nahrání uživatelsky definovaných filtrů a tagů. Modul *i18n* slouží k překladům, tedy k povolení bloku `{% trans %}`. Modul *bootstrap* je využit jako filtr u formulářů, v nichž upravuje původní formulář, aby byl nastylován podle rámce Bootstrap.

Řádky 3 a 24 ohraničují blok *content*, který přepisuje blok stejného názvu v rodičovské šabloně a slouží k zobrazení obsahu. Na řádcích 4 až 10 je kód vypisující nastylované zprávy na základě jejich typu. Na řádcích 11 až 23 je formulář pro odeslání dat.

Zdrojový kód 5.2: Šablona pro zobrazení zpráv.

```
{% extends "base.html" %}
{% load i18n bootstrap %}
{% block content %}
        {% if messages %}
                {% for message in messages %}
                        <div class="alert alert-{{ message.tags }}">
                        {{ message }}
                        </div>
                {% endfor %}
        {% endif %}
        <form method="post" action="">
        {{ page_form.media }}
        {% csrf_token %}
                {{ page_form|bootstrap }}
                <div class="form-group">
                <button type="submit"
                        class="btn btn-primary"
                        value="Submit"
                        name="contest_trans_submit">
                        {% trans "Submit" %}
                        </button>
                </div>
        </form>
{% endblock %}
```

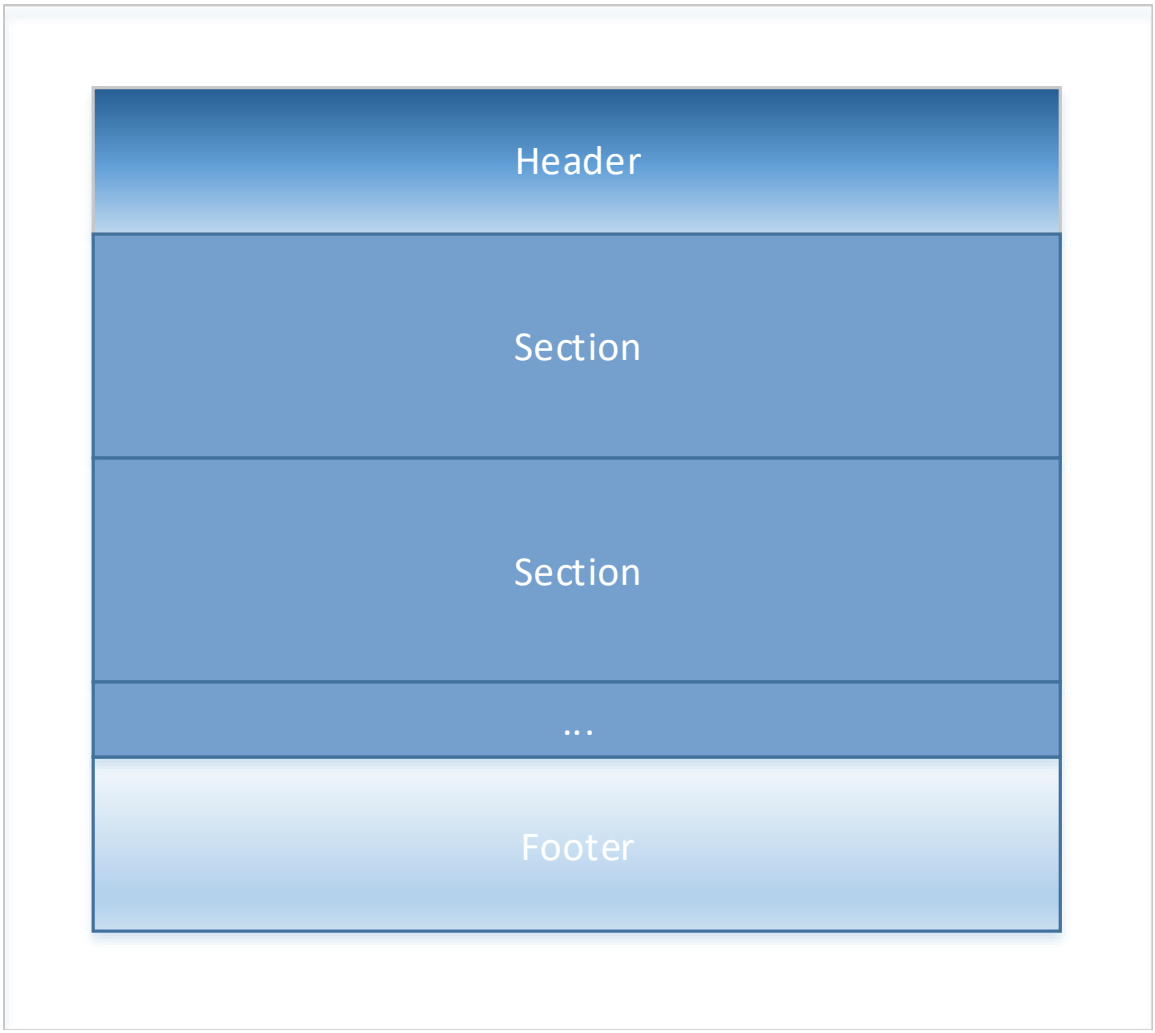
5.2.2 Grafické rozhraní webové aplikace

Grafické rozhraní systému odpovídá návrhu z kapitoly 4. Důraz byl kladen na co největší přehlednost a jednoduchost v kombinaci s moderním vzhledem splňující aktuální trendy. K tomu byla využita knihovna *Bootstrap*, která mnoho problémů, jako například zobrazení formulářů, pozicování na stránkách nebo responzivnost, z velké části řeší za programátora.

Základní rozvržení webu

Uživatelsky přívětivý web vyžaduje přehlednou a jasně definovanou strukturu a absenci bočního menu. Boční menu komplikuje celou navigaci, neboť u moderních webů jsou lidé zvyklí na menu v horní části stránky. Navíc u webů s boční navigací se komplikovaně řeší responzivita. Proto bylo rozhodnuto, že web bude mít menu v horní části.

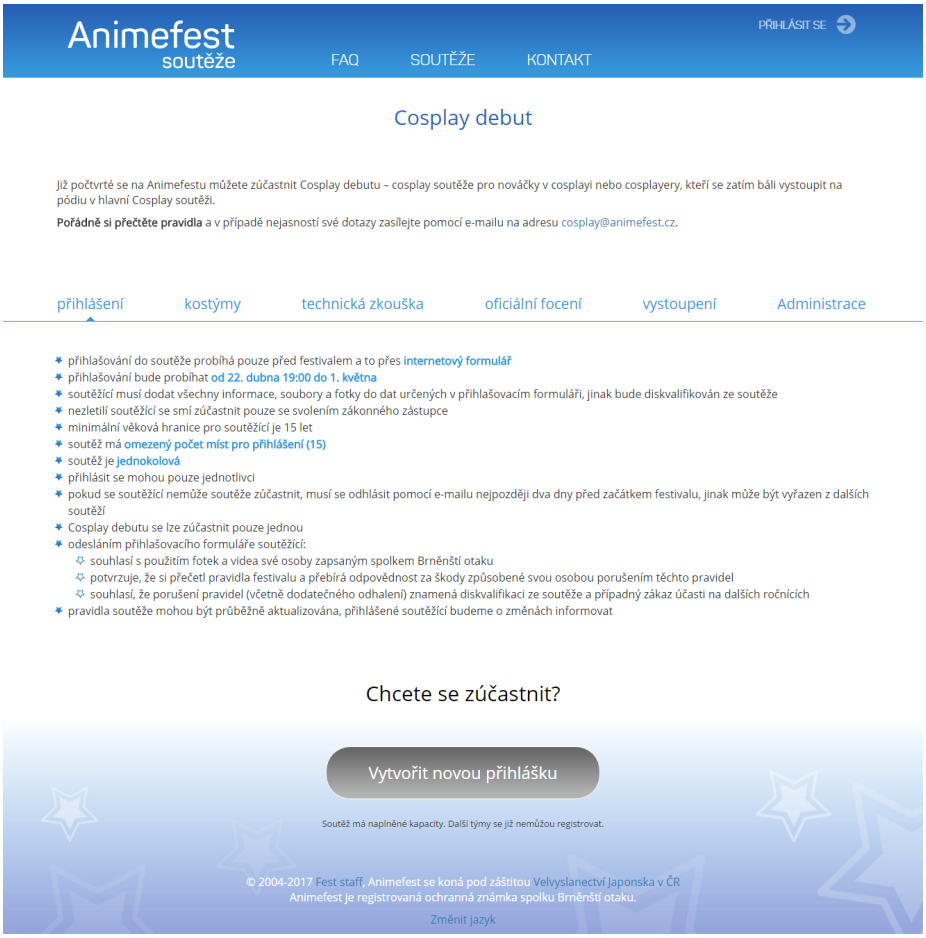
Na obrázku 5.1 je zobrazeno rozvržení webu, respektive jak vypadá samotný obsah umístěný mezi `<body>` a `</body>`. Jako první je vždy blok `<header>`, která vymezuje horní část webu, což je v našem případě logo a menu. Dále následuje 1 až N bloků `<section>`, které vždy obsahují určitou část obsahu, například popis soutěže. Na závěr je vždy blok `<footer>`, takzvaná patička, jež obsahuje především kontakt na organizátory, informaci o záštitě akce a dialog na změnu jazyka.



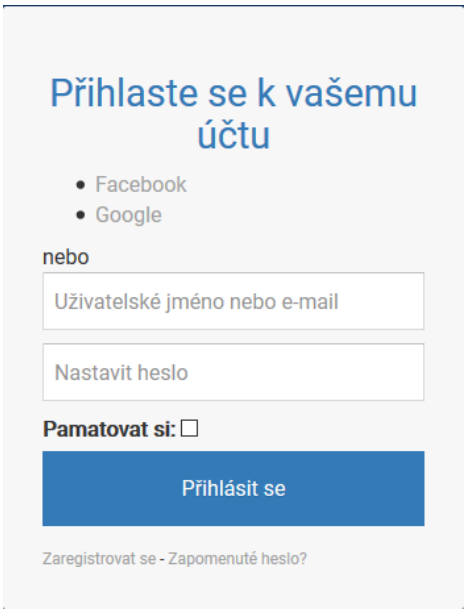
Obrázek 5.1: Struktura webu.

Konkrétní provedení popsané struktury si lze prohlédnout na obrázku 5.2. Je z něj patrné, že celý web je laděný do modré barvy. Jako první je na webu hlavička, poté následuje popis soutěže, popis jednotlivých podsekcí a nakonec patička. Pro nadpisy a menu jsou použity fonty z rodiny *Panton*[9], která byla dodána externím zadavatelem práce. Na ostatní texty až na přihlašovací obrazovku (kde je použit font *Roboto*, obrázek 5.3) je použít font *Open Sans*[10].

[9] https://www.myfonts.com/fonts/font-fabric/panton/
[10] https://fonts.google.com/specimen/Open+Sans



Obrázek 5.2: Struktura webu.

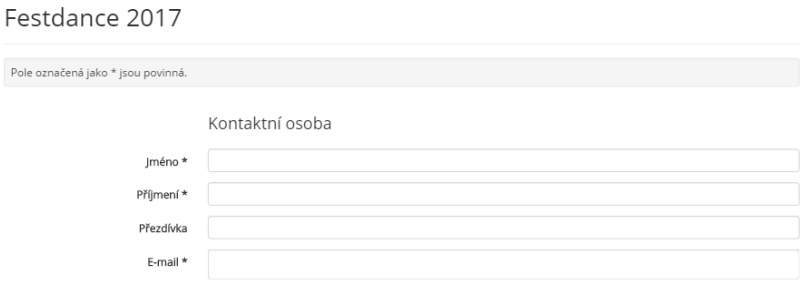


Obrázek 5.3: Přihlašovací obrazovka.

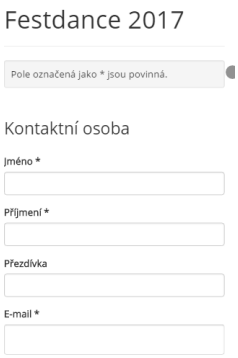
Responzivnost

Jedním z požadavků byla responzivnost webu – tedy aby se správně zobrazoval jak na počítačích, tak i na tabletech a mobilních telefonech. Část responzivnosti, hlavně co se týče formulářů, byla pokryta díky knihovně Bootstrap. Nicméně i tak bylo třeba dodělat několik věcí, jako například responzivní menu, a myslet na různé maličkosti.

Na snímcích obrazovky 5.4 a 5.5 jsou ukázany rozdíly v zobrazení formulářů na obrazovkách s velkým rozlišením (šířka nad 1200px) a obrazovkách s malým rozlišením, standardně na mobilních telefonech (šířka menší jak 767px).



Obrázek 5.4: Formulář zobrazený na velkém rozlišení.



Obrázek 5.5: Formulář zobrazený na malém rozlišení.

Stejné zobrazení na různých rozlišení si klade nárok na relativní velikost a umístění většiny formulářových prvků. To v praxi znamená, že například u odsazení se nesmí zadávat hodnota v pixelech, neboť jedna taková hodnota (např. *10px*) tvoří různé velikosti obrazovky na různých zařízeních. Stejně tak nesmí být velikost fontu určená absolutně, neboť by to na některých zařízeních mohlo způsobit problémy. Bylo tedy třeba místo pixelů používat relativní hodnotu, v aplikaci bylo přistoupeno k *em*.

V CSS je *em* obecně používání jednotka k určení velikostí ať už u odsazení, či u samotné velikosti písma. Jde používat jak vertikálně, tak horizontálně, díky čemuž dokáže nahradit většinu ostatních jednotek. Konkrétně u velikosti fontů se *em* odkazuje na velikost rodičovského elementu. To znamená, že pokud má například nadpis nastavenou velikost *2em*, tak písmo bude bude 2× větší než je v rodičovském elementu, což je často samotný element

body. Implicitně je velikost *body* určena přímo uživatelem v nastavení jeho prohlížeče, čímž je řešen problém různých zařízení. [5]

Kapitola 6

Testování

Jak se píše v knize *Art of Testing*[7], která poprvé vyšla v roce 1979, již tehdy zabíralo testování 50 % celkového času vývoje aplikace, a ani dnes to není jinak. I když se mění programovací jazyky, nástroje i metodiky, tak toto pravidlo stále platí.

V té stejné knize se můžeme dočíst, že při testování webových aplikací je třeba se zaměřit na několik faktorů, mezi které patří:

1. **Velká a rozdílná uživatelská základna**
   Uživatelé webové aplikace mají různé znalosti, používají různé prohlížeče na různých zařízeních a různých operačních systémech.

2. **Lokalita uživatelů**
   Aby byl systém uživatelský přívětivý, tak musí brát v potaz jak překlad, tak i další faktory, jako například časové zóny a formát zobrazení data a času.

3. **Testovací prostředí**
   Pro správné otestování aplikace je třeba mít testovací prostředí, které duplikuje produkční prostředí. To znamená, že by měl být použít stejný webový i aplikační a databázový server na testovacím i na produkčním prostředí. V ideálním případě by měla být nasimulována i síťová infrastruktura.

4. **Zabezpečení**
   Tím, že jsou webové stránky otevřené celému světu, je třeba je ochránit před hackery. Jedná se jak o možné zahlcení webu DoS[1] útokem nebo vyhledávacími boty[2], tak i o ochranu před ztrátou a odcizením dat.

Při testování se na tyto faktory myslelo a bylo podle nich prováděno. Vzhledem k obrovskému rozsahu potenciálních uživatelů nešlo kompletně pokrýt uživatelskou základnu, nicméně testy byly prováděny několika testery na různých zařízeních včetně různého lokálního nastavení.

Testovací prostředí na stroji autora projektu se snažilo co nejvíce přiblížit produkčnímu prostředí. To ve výsledku zajistilo hladký přechod z testovacího do produkčního prostředí a

[1] https://en.wikipedia.org/wiki/DOS
[2] https://en.wikipedia.org/wiki/Web_crawler

minimum chyb spojených s konfigurací prostředí. V tomto případě nehraje síťová infrastruktura velkou roli, její konfigurace v testovacím prostředí byla proto ignorována.

Velkou část zabezpečení, jako například o ochranu před SQL injekcí (vložení škodlivého databázového kódu přes neošetřené vstupy), zajistil již samotný aplikační rámec *Django*. Nicméně bylo třeba otestovat i naimplementovaný systém autentizace – zda nepustí uživatele k datům, ke kterým by neměli mít přístup.

6.1 Uživatelské testování

Kromě otestování funkčnosti jako takové bylo třeba se zamyslet i nad tím, jestli je systém dostatečně uživatelsky přívětivý i pro samotné uživatele. Cílem bylo mít systém natolik intuitivní, aby soutěžící nepotřeboval k používání návod. U funkcionalit pro správce soutěže (a jejich asistenty) se počítalo se zaškolením v podobě návodu ať už v podobě manuálu nebo příručky.

6.1.1 Soutěžící

Vzhledem k jednoduchosti celého webu, respektive nízkému počtu různých menu a tlačítek, nedošlo ze strany soutěžích k žádným zmatkům. Soutěžící sice neměli k testování dostupný žádný manuál, nicméně ukázalo se, že je třeba jeé v popisu soutěže mírně nasměrovat, neboť některé vlastnosti systému (např. implicitní možnost upravovat přihlášky končící při uzavření registrací) nejsou patrné z aktuálního návrhu systému.

Při testování bylo nalezeno několik nekritických chyb ve funkčnosti systému, stejně tak bylo nalezeno několik chyb týkajících se grafiky, hlavně při zobrazení webu z mobilních zařízení, protože na některých zařízení se objevil problém se odsazením. Kvůli tomu pak text nepůsobil celistvě a nebylo ve všech případech jasné, k čemu daný text patří.

6.1.2 Správci soutěže

K zaškolení správců soutěže byl potřeba základní návod osvětlující všechny funkce v systému včetně popisu, jak se k nim dostat. I přesto byly z pohledů správce v systému nalezeny určité funkcionality, které nebyly na první pohled patrné, a působily tak matoucím dojmem.

Největším problémem byla manipulace se zúčastněnými týmy – jejich mazáním, popř. s úpravami členů. U těchto úprav bylo vždy třeba více kroků a nikde nebylo přesně popsané, co a v jakém pořadí je potřeba udělat. Tento problém byl nakonec vyřešen pomocí přidání tlačítka k seznamu týmů, které umožňuje tým včetně přihlášky jednoduše vymazat.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit podpůrný informační systém pro pořádání soutěží v rámci festivalu Animefest, jenž se neustále rozrůstá. Zadavatelem práce byl zapsaný spolek Brněnští otaku, pro jehož požadavky dosavadní systém nebyl dostačující.

Nedílnou součástí tvorby informačních systémů je analýza požadavků, která zároveň probíhá jako jedna z prvních, neboť vymezuje potřebné funkcionality. Ta byla úspěšně provedena díky konzultacím se zadavatelem práce. K tomu byl vytvořen také diagram případu užití, který graficky znázorňuje chování systému a možnosti jednotlivých uživatelů.

Na základě zanalyzovaných požadavků bylo možné sestavit návrh informačního systému. Jeho hlavním prvkem byl ER Diagram, jenž znázorňuje abstraktní zobrazení databáze pomocí entit a vztahy mezi nimi. Dle požadavků na systém byla i provedena analýza dostupných technologií, na jejíž základě bylo rozhodnuto, že informační systém bude vyvíjen v jazyce Python v aplikačním rámci *Django*.

Systém byl průběžně testován autorem, měsíc před ostrým nasazením byl v připomínkové fázi testován samotným zadavatelem, který měl k systému několik podnětů, které byly postupně implementovány a znovu otestovány. Ve výsledku byly všechny požadavky zadavatele splněny a systém byl nasazen na stránkách `http://souteze.animefest.cz`.

Webovou aplikaci by bylo možné rozšířit několika způsoby. Funkcionalita z pohledu správce by se mohla upravit o možnost zakázat celé části formuláře, místo dosavadního zakázání jednotlivých polí. Rovněž by bylo zajímavé dopředu zapracovat podmínky pro možnost úpravy jednotlivých částí formuláře. Dále by šlo zjednodušit tvorbu šablon formulářů pomocí vícenásobných prvků a jejich klonování.

Co se týče soutěžících, při nahrávání obsahu (např. videa) by šlo zobrazit ukazatel, který bude ilustrovat, kolik procent souboru je již úspěšně nahráno na serveru. Jako prospěšné by se mohlo ukázat i zavedení podpory pro dohrání souboru na server v případě přerušení.

Vzhledem k tomu, že systém byl vyvíjen v aplikačním rámci ve verzi s dlouhodobou podporou, z bezpečnostního hlediska by mělo stačit po několik následujících let pouze pravidelně aktualizovat tuto verzi.

Literatura

[1] *Django documentation: design philosophies*. [Online; navštíveno 10.01.2017].
URL https://docs.djangoproject.com/en/1.10/misc/design-philosophies/

[2] Foster, E. C.; Godbole, S.: *Database Systems: A Pragmatic Approach*. Berkeley, CA : Apress, 2014, ISBN 978-1-4842-0878-6.

[3] George, N.: *Mastering Django: Core: The Complete Guide to Django 1.8 LTS*. Packt Publishing, 2016, ISBN 978-1-78728-114-1.

[4] *Introducing JSON*. [Online; navštíveno 10.04.2017].
URL http://www.json.org/

[5] Lie, H.; Bos, B.: *Cascading Style Sheets: Designing for the Web*. Pearson Education, 2005, ISBN 978-0-132-46573-1.
URL https://books.google.cz/books?id=McUjSXNvLI0C

[6] Lutz, M.: *Learning Python*. O'Reilly Media, 2013, ISBN 978-1-449-35573-9.

[7] Myers, G. J.: *The art of software testing*. Hoboken : Wiley, 2004, ISBN 978-0-471-46912-4.

[8] Robson, E.; Freeman, E.: *Head First HTML and CSS*. O'Reilly Media, 2012, ISBN 978-0-596-15990-0.

[9] Spurlock, J.: *Bootstrap*. O'Reilly Media, 2013, ISBN 978-1-449-34391-0.

[10] *What is Symfony*. [Online; navštíveno 14.01.2017].
URL http://symfony.com/what-is-symfony

Příloha A

Obsah CD

- elektronická verze písemné zprávy a návod k instalaci ve formátu pdf
- zdrojový kód písemné zprávy
- zdrojové kódy informačního systému