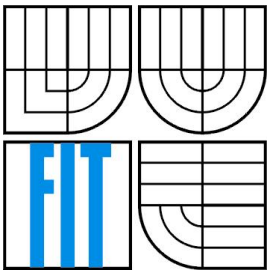
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INFORMAČNÍ SYSTÉM PRO ODDÍL ORIENTAČNÍHO BĚHU

INFORMATION SYSTEM OF AN ORIENTEERING SECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FILIP HADAČ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BARTÍK, Ph.D.

BRNO 2016

Abstrakt

Tato práce se zabývá návrhem a realizací informačního systému pro oddíl orientačního běhu v Opavě. Informační systém poskytuje uživatelům přihlašování se na události a organizaci dopravy na ně. Nedílnou součástí je vyúčtování, které manipuluje s financemi na osobních účtech uživatelů. Systém také poskytuje veřejnosti informace ohledně areálů pevných kontrol včetně výsledků a možnost přidávat své vlastní výsledky. Dalším účelem jsou příspěvky na hlavní stránce, pomocí kterých se sdělují informace a recenzují události. Práce klade důraz na jednoduchost a intuitivnost. Systém je postaven na běžně dostupné technologii Python s využitím frameworku Flask. Jeho vzhled je zprostředkován pomocí CSS frameworku Bootstrap a využitím technologií JavaScript. Výsledek této práce je dostupný na adrese www.fhadac.cz.

Abstract

This thesis is devoted to a designation and realisation of the information system of orienteering club in Opava. The information system provides users a registration for events and subsequent transport organisation. The integral part of the information system is finance management which affects the cash balance of linked personal user's accounts. The information system provides information and results of permanent courses with and upload tool for personal performances of the users. Another purpose of the information system is a post publishing function that aims to convey information and to review events. The thesis emphasises simplicity and intuition. The information system is built on Python with the use of the Flask framework. The graphic user interface is created by the Bootstrap CSS framework using JavaScript technology. The result of this thesis is available at www.fhadac.cz.

Klíčová slova

Informační systém, orientační běh, stahování dat přes API, Flask, Python, Bootstrap.

Keywords

Information system, orienteering, download data through API, Flask, Python, Bootstrap.

Citace

Hadač Filip: Informační systém pro oddíl orientačního běhu, bakalářská práce, Brno, FIT VUT v Brně, 2016

Informační systém pro oddíl orientačního běhu

Prohlášení

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

……
Filip Hadač
15. května 2016

Poděkování

Chtěl bych poděkovat především svému vedoucímu bakalářské práce Ing. Vladimíru Bartíkovi, Ph.D. za veškerou pomoc při tvorbě a cenné rady. Dále bych rád poděkoval vedení oddílu orientačního běhu za neustálou komunikaci ohledně všeho, co v systému má být a jak to má vypadat a také za pomoc při testování.

Obsah

1 Úvod

V dnešním světě si již nedovedeme představit život bez informačních systémů. Využívají se v každé firmě pro správu a chod. V každé bance pro tok částek na účtech. V každé škole pro správu školy, studentů a jejich znalostí. V každé knihovně pro evidenci knih a jejich půjčování. A nyní již i ve většině sportovních svazů a klubů.

Tato bakalářská práce pojednává o informačním systému pro oddíl orientačního běhu. Doteď veškerá komunikace, domluva a vyúčtování probíhaly pouze papírově a emailem. To se však nyní změní, a to vše díky novému informačnímu systému. Ten má nahradit tyto "zastaralé" metody a přinést do tohoto oddílu moderní technologie a usnadnit lidem práci.

Tento informační systém využívá data stažená z informačního systému Českého svazu orientačních sportů zvaného ORIS. V tomto případě se jedná o jednotlivé uživatelé (registrovaní závodníci v ČR pro daný rok) a závody (všechny pořádané v ČR pro daný rok). Pomáhá plánovat soustředění a tréninky pořádané oddílem. Řeší přihlašování uživatelů na akce včetně ubytování. Nedílnou součástí systému je tzv. *"automodul"*, který kompletně pomáhá zpracovat veškerou agendu související s dopravou na systémem registrované akce (vytížení jednotlivých osobních aut). Systém usnadňuje přihlašovateli vyúčtování po proběhlé akci, hlídá a řídí finance na osobních účtech. Umožňuje přidávat informační příspěvky. Zprostředkovává online styk s neregistrovanými uživateli (aktivními zájemci o orientační běh z řad veřejnosti), například evidenci absolvování kontrolních stanovišť v areálech pevných kontrol, které oddíl provozuje. Dále systém disponuje mnoha užitečnými funkcemi, které usnadňují administrativní a reálny chod oddílu.

První kapitola se zabývá principy tvorby webových aplikací, přesněji výčtem programovacích jazyků a frameworků, které se využívají. Ve druhé kapitole je vše potřebné pro pochopení informačního systému Českého svazu orientačních sportů ORIS. Třetí kapitola popisuje analýzu požadavků, do které je zahrnut i diagram případů užití. Čtvrtá kapitola obsahuje návrh aplikace, kde je ER diagram a použité technologie. V páté kapitole je shrnuta implementace aplikace, kde jsou popsány hlavně složitější části systému. Šestá kapitola pojednává o testování a možnostmi dalšího vývoje aplikace.

2 Principy tvorby webových aplikací

Tato kapitola se zabývá principy tvorby webových aplikací. Popisuje, co to webová aplikace je a její důležité části. Nejdůležitější je programovací jazyk, popřípadě nějaký jeho framework. Každý z těchto jazyků a frameworků má poté své vlastní principy, ty zde však nejsou uvedeny.

2.1 Webová aplikace

Webová aplikace je aplikace, kterou není potřeba instalovat na straně klienta. Tato aplikace je nainstalována na straně serveru a proto je možné ji spustit na jakémkoliv zařízení za pomocí webového prohlížeče. Může být nazývána také jako lehký klient. Webová aplikace může na první pohled vypadat jako obyčejná webová stránka, obvykle se však jedná o složitější aplikaci, která provádí různé úkony a spolupracuje s databází. V této době umí webová aplikace téměř to stejné, jako aplikace desktopová nainstalována na počítači. Pro přístup k takové aplikaci je potřeba internetové připojení, které by pro plynulý chod mělo být co nejrychlejší. [11]

Webová aplikace se skládá ze tří částí, kterými jsou server, databáze a klient. Server slouží pro spuštění aplikace a provádění veškerých akcí, které jsou zapotřebí pro daný úkon a zobrazení stránky. Pro získávání dat k tomu potřebných komunikuje s databází. Databáze je část webové aplikace, která uchovává veškerá potřebná data. Většinou se používá relační databáze formou tabulek propojených mezi sebou, může však být i jiného typu. Poslední částí je klient, což je prezentace aplikace uživateli. Klient zprostředkovává webový prohlížeč.

Výhody:

- Nemusí se nic instalovat.
- Uživatel nemusí aktualizovat verze aplikace.
- Je potřeba pouze webový prohlížeč.
- Aplikace je přístupná odkudkoliv.
- Data jsou uchovávána na straně serveru.

Nevýhody:

- Vyžaduje se připojení k internetu.
- Někdy pomalejší práce s aplikací.
- Bezpečnostní rizika.

2.2 Dostupné programovací jazyky

V dnešní době existuje obrovské množství programovacích jazyků. S těmi přichází i velké množství technologií, které se dají pro tvorbu webových aplikací využít. Největší převahu na trhu má stále PHP, ale čím dál více lidí jej přestává mít v oblibě a snaží se využívat jiné jazyky. Níže jsou uvedeny vybrané jazyky a jejich frameworky.

2.2.1 PHP

PHP (dříve: *Personal Home Page*, nyní: *Hypertext Preprocessor*) je skriptovací programovací jazyk. Jeho hlavní využití spočívá ve tvorbě internetových stránek a webových aplikací. Dá se však použít i jako běžný skriptovací jazyk pro konzolové aplikace. V případě webových aplikací se kód provádí na straně serveru. Syntaxe PHP je inspirována několika jazyky (Perl, C, Pascal, Java). Tento jazyk se využívá především pro jeho nezávislost. Může běžet na jakémkoliv systému jako server nebo se zobrazovat v jakémkoliv prohlížeči jako klient. Typickou vlastností tohoto jazyka je, že se proměnné píší se znakem "`$`" na začátku názvu. [3]

Frameworky:

- Nette
- Laravel
- Symphony

2.2.2 Python

Python je vysokoúrovňový skriptovací programovací jazyk. Jeho primární použití je hlavně ve skriptech, avšak byl původně navržen, aby zvládal plnohodnotné aplikace. Často se využívá jako skriptovací jazyk doprovodný k jinému jazyku, díky jeho jednoduchému vkládání. Jeho syntaxe je rozdílná od většiny jazyků kvůli absenci složených závorek pro třídy, funkce a výrazy. Místo těchto složených závorek se využívá znak "`:`" a následné oddělení mezerami do různých úrovní. Tento jazyk se dá teoreticky použít pro cokoliv a to je jeho velké plus. [4]

Frameworky:

- Django
- Flask
- Bottle

2.2.3 Ruby

Ruby je interpretovaný skriptovací programovací jazyk. Jeho oblast využití je stejná jako u ostatních skriptovacích jazyků. Má velice jednoduchou syntaxi a díky tomu je jednoduchý na naučení, ale stále dostatečně výkonný. Jeho výhodou ve tvorbě webových aplikací je, že se v hojném počtu využívá pro tvorbu internetových obchodů. [5]

Frameworky:

- Ruby on Rails

2.2.4 JavaScript

JavaScript je zde poslední zmíněný jazyk a jako ostatní je také skriptovací. Největší uplatnění má momentálně jako programování dynamických prvků na webových stránkách. Avšak v poslední době se začal rozrůstat i jako programovací jazyk pro server webové aplikace. Využívají se zde ve velkém funkce, které se dají použít i bez deklarace a jména. Jinak je jeho syntaxe je podobná jazykům C, C++ a Java. [6]

Frameworky:

- Node.js

3 Informační systém ORIS

Informační systém ORIS (Dostupný z: http://oris.orientacnisporty.cz) využívá Český svaz orientačních sportů. Pro komunitu orientačních běžců a zainteresované osoby v tomto sportu je velice přínosný. Namísto přihlašování, pokynů, výsledků a dalších informací na jednotlivých internetových stránkách závodů je vše v tomto systému přehledně na jednom místě. Závodníci zde mají svůj přístup na základě registračního čísla a mohou systém plnohodnotně využívat v závislosti na jejich nastavené autoritě. Základní možností je být registrován v nějakém oddílu a pod jeho záštitou se přihlašovat na závody. Rozšířenější možnosti jsou pak spíše v rámci oddílu, kde je jeden vedoucí a ten může nastavit různá práva členům daného oddílu. Těmi nejhlavnějšími rozšířenými možnostmi jsou například přihlašování jiných závodníků na závody nebo spravování soupisek štafet. Dále jsou zde možnosti registrovat nové členy a upravovat závody svého oddílu. Uživatel s administrátorskými právy oddílu může upravovat veškeré informace o členech.

3.1 Popis API

Nedílnou součástí informačního systému českého svazu orientačních sportů ORIS je také API (Dostupný z: http://oris.orientacnisporty.cz/API), neboli Application Programming Interface (rozhraní pro programování aplikací). API jsou knihovny, které nabízí různé metody obsluhy. Pomocí těchto metod se dá komunikovat se serverem dané domény, která tuto možnost poskytuje. Informační systém ORIS využívá REST (Representational State Transfer) API. Toto REST API funguje především na základě požadavků `GET` a `POST`. V některých případech má ještě další možnosti jako `DELETE` nebo `PUT`, nejsou však v sytému použity. ORIS API má cca 20 metod, díky kterým se dá se systémem komunikovat z venčí. Všechny tyto metody mohou mít nastavený výstup ve formátu XML nebo JSON. Výběr formátu výstupu je pouze na uživateli, který API využívá. Každá metoda má nějaké povinné a volitelné parametry. [7]

3.2 Metody API

Zde je výčet metod API informačního systému ORIS, které se nějakým způsobem hodí využít ve vytvářeném systému. Jedná se o metody `GET`, kdy pro žádnou z nich není potřeba být přihlášen do systému, aby vrátila výsledek.

3.2.1 getEventList

Tato metoda je volána v případě, že uživatel chce výpis všech závodů (kalendář závodů), které v systému jsou. Nemá žádný povinný parametr, tudíž bez zadaných parametrů to navrátí úplně všechny závody. Volitelných parametrů je hned několik, za zmínku stojí však jen některé.

Volitelné parametry:

- **`sport`** - identifikační číslo sportu (OB - orientační běh, LOB - lyžařský orientační běh, MTBO - orientační závody horských kol)
- **`level`** - úroveň závodu (MČR, Český Pohár atd.)
- **`datefrom`** - datum konce, v případě nevyplnění se bere první den aktuálního roku
- **`dateto`** - datum začátku, v případě nevyplnění se bere poslední den aktuálního roku

3.2.2 getEvent

Díky této metody `getEvent` můžeme získat veškeré dostupné informace o daném závodě. Mezi těmito informacemi jsou například lokalita, typ závodu, disciplíny, pořadatel a další. Tato metoda nemá žádné volitelné parametry a požaduje jeden povinný.

Povinné parametry:

- **`eventid`** - identifikační číslo závodu

3.2.3 getEventEntries

Pokud chceme získat veškeré přihlášené závodníky na daný závod nebo v případě štafet přihlášené štafety, použijeme metodu `getEventEntries`. Je zde pouze jeden povinný parametr a více volitelných, ze kterých jsou důležité jen některé.

Povinné parametry:

- **`eventid`** - očekává identifikační číslo závodu

Volitelné parametry:

- **`clubid`** - identifikační číslo nebo zkratku oddílu
- **`classid`** - identifikační číslo kategorie
- **`classname`** - název kategorie

3.2.4 getEventBalance

Metoda `getEventBalance` vrátí seznam vyúčtování pro daný závod po jednotlivých klubech. Metoda nemá žádný volitelný parametr, ale požaduje jeden povinný parametr.

Povinné parametry:

- **`eventid`** - identifikační číslo závodu

3.2.5 getEventResults

Pomocí metody `getEventResults` se získává seznam výsledků pro daný závod. Požaduje tedy jeden povinný parametr a několik volitelných.

Povinné parametry:

- **`eventid`** - identifikační číslo závodu

Volitelné parametry:

- **`classid`** - identifikační číslo kategorie
- **`classname`** - název kategorie
- **`clubid`** - identifikační číslo nebo zkratka oddílu

3.2.6 getRegistration

Poslední vybranou metodou je `getRegistration`, která vrací registrované závodníky v systému. Požaduje dva povinné parametry a žádný volitelný.

Povinné parametry:

- **`sportid`** - identifikační číslo sportu (OB, LOB, MTBO)
- **`year`** - rok

3.3 Možnosti využití API ve vytvářeném systému

Výše zmíněné ORIS API poskytuje mnoho možností, jak jej využít ve vytvářeném informačním systému. Největší předností pro bude metoda `getEventList`, která vrátí seznam všech závodů. Nebude tedy potřeba jednotlivé závody ve vytvářeném systému znovu přidávat ručně. Pomocí této metody se dostanou závody v České republice, ze kterých se potom ve vytvářeném systému může vybrat, které budou přidány do databáze. Tyto vybrané závody bude oddíl, pro který je informační systém vyvíjen, podporovat.

Při vytváření jednotlivých závodů, které se ze seznamu všech závodů vyberou, se použije metoda `getEvent`. Díky této metodě se získají veškeré informace o daném závodě a vybrané se budou moci uložit do databáze. Toto zjednoduší práci ručního přepisování jakýchkoliv informací o závodě z informačního systému ORIS do vyvíjeného systému.

Jakmile jsou jednotlivé závody vytvořené, je potřeba zjistit, kdo je na tyto závody přihlášen a uložit si to do databáze. K tomu bude sloužit metoda `getEventEntries`, která vrátí seznam závodníků, kteří jsou na daný závod přihlášení. V případě štafetového závodu to bude seznam štafet.

Dále po závodech bude potřeba zjistit dvě věci. Tou první je, jaké finanční náklady byly na závod. Měla by to zajistit metoda `getEventBalance`, která vrací vyúčtování pro daný závod. Za druhé je potřeba do příspěvku předgenerovat výsledky, pokud je to možné. Toto zvládá metoda `getEventResults`.

Kromě seznamu závodů a informací kolem těchto jednotlivých závodů, může informační systém ORIS poskytnout i údaje o registrovaných závodnících pro daný rok v oddíle. Tyto závodníky je poté možnost vytvořit také ve vyvíjeném informačním systému, aby je nemusel administrátor zadávat ručně. K tomuto pomůže metoda `getRegistration`. Pomocí ní půjde také v případě potřeby aktualizovat informace o všech uživatelích, popřípadě přidat nové. Tato metoda je bohužel veřejná a tudíž nevrátí důvěrné informace, které si budou muset uživatelé sami zadat.

4 Analýza požadavků

Oddíl orientačního běhu v Opavě má mnoho nápadů a požadavků na systém, které je potřeba nějakým způsobem zpracovat a dát dohromady. Vůbec nejlepším způsobem jak toto řešit je osobní setkání, na kterém se probere vše potřebné. Těchto setkání je však potřeba více, protože požadavků na systém přibývá. Dále je také nutné si na těchto setkání vždy upřesnit, co jak má přesně vypadat, aby nedošlo ve výsledném produktu k nějakému nedorozumění. Díky takovýmto setkáním se dá následně vše zanalyzovat a provést syntézu. Vzhledem k tomu, že autor tvořeného systému je také členem oddílu orientačního běhu v Opavě, tak není problém tyto schůzky mít kdykoliv je potřeba. Proto by měl vytvářený systém ve výsledku také splňovat veškeré požadavky a nemělo by nic zásadního chybět. Do analýzy patří také diagram případu užití, který napomáhá v ujasnění si požadavků na systém.

4.1 Požadavky na systém

- Systém bude komunikovat s informačním systémem Českého svazu orientačních sportů ORIS a stažená data se budou využívat pro co nejvíce účelů.
- Systém bude umožňovat všem členům oddílu přihlášení.
- V systému bude možno vytvářet a editovat různé události (Závody, Soustředění a Tréninky). Pro tyto události bude existovat automodul, ve kterém bude probíhat organizace dopravy. Dále bude možno ve vybraných závodech řešit ubytování. V případě štafet také přihlašování, kdo chce ve štafetách startovat. Po proběhlé události musí být možné vyúčtování, které bude předvyplněné. V neposlední řade se bude dát napsat příspěvek k dané události a v případě závodů budou v příspěvku předvyplněné výsledky.
- Systém bude mít také informace o areálech pevných kontrol, které budou dostupné veřejnosti. Při vytváření se nastaví počet kontrol a jaký kód jednotlivé kontroly mají. Veřejnost si poté může přidávat své vlastní výsledky a vidět, kolik kontrol mají správně a jak jsou na tom vůči ostatním lidem.
- Systém bude řešit veškeré finance na osobních účtech všech uživatelů a jednotlivé transakce budou vždy dostupné a možné editovat. Tyto osobní účty budou mít možnost skupinového vyúčtování, tudíž jeden vlastník bude spravovat různé účty (například rodič své děti). Po určitém období nebo když administrátor zadá příkaz, bude vyúčtování. Vyúčtování bude probíhat formou, že správcům účtů se pošle email s částkou, jakou oddílu dluží za svou skupinu (popřípadě pouze sám za sebe) a variabilní symbol, se kterým částku odešlou na daný účet.
- Systém bude umožňovat vybraným osobám přidávat příspěvky na zeď, které musí podporovat formátování textu a přidávání obrázků nebo souborů.

4.2 Diagram případů užití

Diagram případů užití neboli Use Case diagram patří do jazyka UML (Unified Modeling Language), který je používán pro grafickou vizualizaci návrhů. Tento diagram patří do skupiny diagramů chování a znázorňuje tedy chování systému. Vypovídá o tom, co má systém umět, ale neznázorňuje už jak nebo jakým způsobem. Při návrhu se tedy většinou používá jako první diagram, protože se díky něj dá určit, co vše systém musí umět. Skládá se ze dvou hlavních částí, kterými jsou případ užití a aktér. Aktérem se rozumí uživatel nebo externí systém, který komunikuje s daným systémem. Případ užití je daná funkcionalita, kterou systém umožňuje. Čárami mezi aktérem a případem užití se dá znázornit, že tato funkce je danému aktérovi zpřístupněna. Dále jsou zde šipky mezi aktéry, které znázorňují jejich dědičnost.

Obrázek 1: Diagram případů užití

4.2.1 Nepřihlášený uživatel

Nepřihlášeným uživatelem se rozumí takový uživatel, který nemá přístup do informačního systému nebo se ještě nepřihlásil. Možnosti takového uživatele jsou značně omezené. Těmi nejdůležitějšími jsou níže zmíněné.

4.2.1.1 Přihlášení

Důležitou možností nepřihlášeného uživatele je jeho přihlášení do informačního systému. Přihlášení bude přístupné v hlavní liště a bude na všech stránkách. Stránka pro přihlášení bude jednoduchý formulář pro vyplnění registračního čísla a hesla s možností zapamatování si uživatele.

4.2.1.2 Příspěvky

Náhledy příspěvky přispívané v informačním systému jsou dostupné na hlavní stránce. Uživateli je poté umožněno zobrazení jednotlivých obsahů příspěvků a následná možnost přidávat komentáře přes formulář k tomu určený.

4.2.1.3 Areály pevných kontrol

Hlavním prvkem nepřihlášeného uživatele jsou areály pevných kontrol. Uživatel si může zobrazit seznam všech areálů pevných kontrol a vybrat některý. O tomto daném areálu se zobrazí informace a odkazy s areálem související. Jedním z odkazů je přidání vlastního výsledku a tím druhým zobrazení celkových výsledků všech závodníků daného areálu. Možností je také přidávání komentářů k areálu, stejně jako je tomu u příspěvků.

4.2.2 Přihlášený uživatel

Přihlášený uživatel je běžný uživatel informačního systému. Tento uživatel má velké možnosti v systému, ve kterých jsou zahrnuty mimo jiné i možnosti nepřihlášeného uživatele.

4.2.2.1 Odhlášení

Odhlásit ze systému se může sám uživatel, který danou možnost zvolí v hlavním panelu na jakékoliv stránce. Další možností je odhlášení po určitém časovém intervalu nebo po vypnutí prohlížeče, záleží na zvolené možnosti "pamatovat si mě" při přihlašování.

4.2.2.2 Uživatelé

Přihlášený uživatel má možnosti podívat se na veškeré své informace nebo informace o kterémkoliv jiném uživateli. V rámci svých informací může i část z nich editovat. Dále si může změnit heslo, popřípadě pokud jej zapomene, tak heslo obnovit. V rámci svého účtu si může vytvořit a editovat auta. Může se také podívat na jednotlivá vlastnictví osobních účtů v oddíle a v rámci toho i na celkový a jednotlivý stav na osobních účtech.

4.2.2.3 Události

Přihlášený uživatel si může nechat vypsat seznam jakýchkoliv událostí a následně se podívat na detaily dané události. V události má uživatel mnoho dalších možností. Jednou z nich je přihlášení se na danou událost. Může také projevit zájem o ubytování. Dále má možnost zobrazit si auta, které jedou na událost. Může zde přidat některé své auto nebo se přihlásit do auta již přidaného.

4.2.3 Přihlašovatel

Přihlašovatel je přihlášený uživatel do informačního systému, kterému bylo přiděleno právo přihlašovatele na určitou událost administrátorem systému. Toto právo přihlašovatele se vztahuje pouze na daný závod, ke kterému byl uživatel přiřazen. Jeho možnosti v rámci události jsou následující.

4.2.3.1 Události

V rámci události přibude přihlašovateli několik možností, co může dělat. První z nich je editace události, ve které může měnit určité informace, popřípadě také změnit přihlašovatele. Další z možností je napsat příspěvek k dané události. Poslední možností je vyúčtování události, kde určí, kolik který uživatel za co zaplatí nebo naopak dostane na osobní účet.

4.2.4 Administrátor

Administrátor je nejvyšší role v informačním systému a má možnost dělat mnoho věcí navíc, oproti běžným uživatelům. Kromě dodatečných funkcionalit může administrátor dělat se všemi událostmi to, co přihlašovatel u své události.

4.2.4.1 Události

Administrátor má možnost vytvářet nové události, kromě závodů, které se stahují z informačního systému ORIS. Dále může na veškeré události nastavit přihlašovatele a pak už má stejné možnosti jako přihlašovatel, pouze na všechny události.

4.2.4.2 Členství a skupiny

Další možností administrátora v systému je vytváření a editace různých členství a skupin. Tyto členství a skupiny se potom dají přiřadit jednotlivým uživatelům.

4.2.4.3 Transakce

V systému probíhá mnoho možností účtování a plateb. Transakce jsou informace o všech těchto jednotlivých převodech na účtech. Tyto jednotlivé transakce se dají zobrazit a dále také upravovat.

4.2.4.4 Osobní účty

V rámci osobních účtů může administrátor nastavovat jednotlivá vlastnictví. Například otec bude spravovat osobní účty zbytku rodiny.

4.2.4.5 Platby

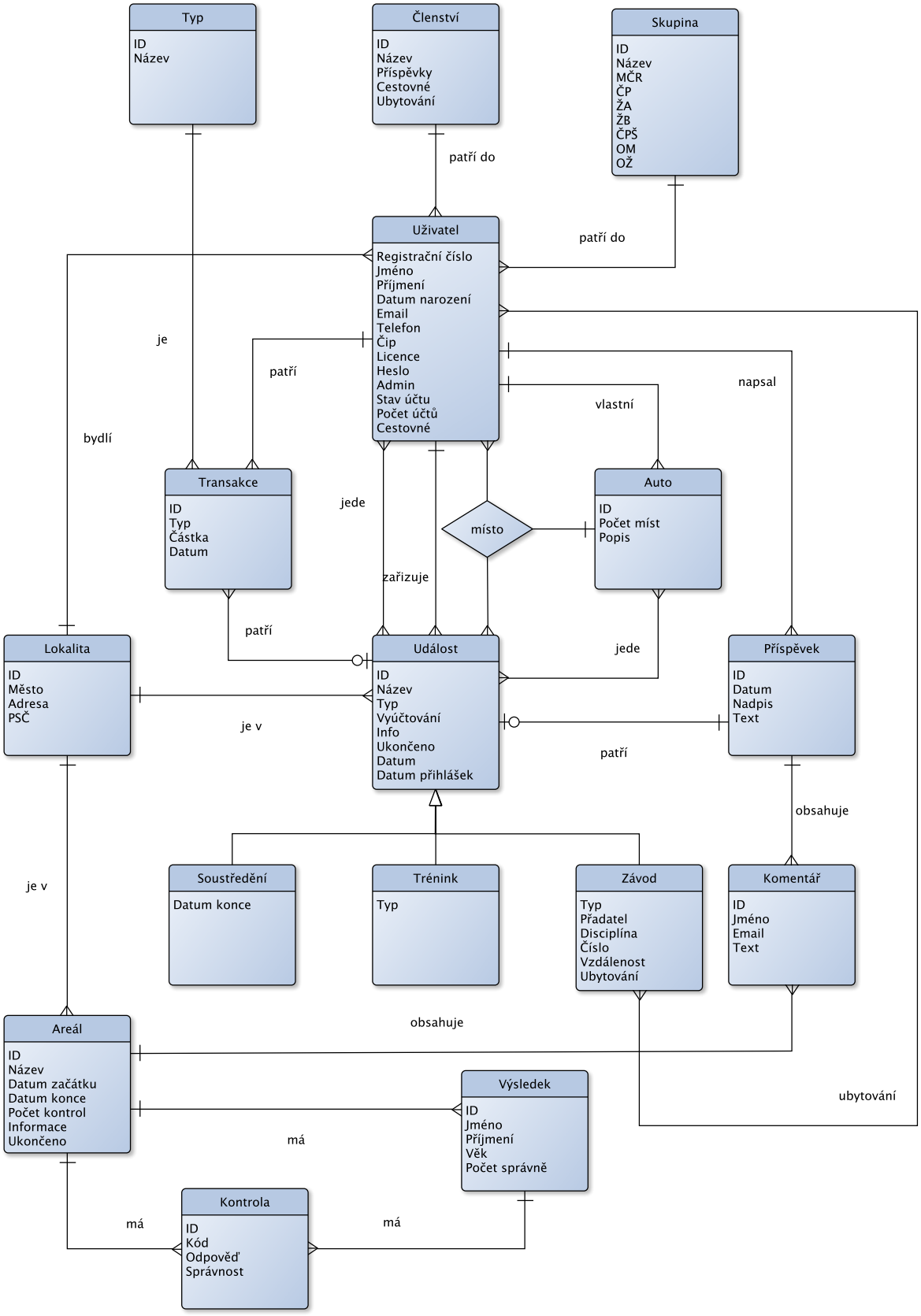
Jsou zde dva způsoby plateb a několik různých typů, které administrátor může provádět. První možnost platby je zadání nějakého typu, který vytvořil a zadání částky uživatelům. Druhou možností je vyúčtování osobních účtů, které je závislé na nastavených vlastnictvích.

5 Návrh aplikace

Tato kapitola pojednává o návrhu aplikace vytvářeného informačního systému. Bude zde zmíněn jeden diagram, který je v návrhu informačního systému nezbytný a usnadňuje následnou práci s implementací. Tento diagram se jmenuje ER diagram. Dále se v návrhu aplikace řeší, jaké budou použité technologie při implementaci.

5.1 ER diagram

ER diagram, neboli Entity-Relationship diagram je konceptuální model. Konceptuální model se využívá pro návrh znázornění dat v systému a vztahů mezi nimi. Důležitá data jsou pouze ty, které se budou ukládat, měnit, zobrazovat a mazat. Toto znázornění se při implementaci převede většinou na tabulky relační databáze, tudíž se dá hodnotit jako návrh databáze. Diagram se skládá z jednotlivých entit a vztahů mezi nimi.



Obrázek 2: Entity-relationship diagram

5.1.1 Entity

Zde je uveden výčet jednotlivých entit v ER diagramu informačního systému. Jednotlivé entity se jmenují podle toho, co ukrývají za informace. Tudíž by neměl být problém spojit si je s reálným modelem.

5.1.1.1 User

User značí reálného uživatele, který informační systém obsluhuje. Tímto uživatelem se myslí specifický člen oddílu. O jednotlivých členech oddílu se uchovává velké množství informací. Mezi tyto informace patří základní, jako je jméno, příjmení, datum narození a podobné. Ale také zašifrované heslo, zdali je uživatel administrátor nebo stav osobního účtu a mnohé další. Primárním klíčem této entity je registrační číslo závodníka v orientačním běhu.

5.1.1.2 Lokalita

Lokalita je určité místo, které v sobě ukrývá důležité informace o pozici. Těmito informacemi jsou město, adresa a PSČ. Lokalita jako taková ve vytvářeném informačním systému nefiguruje samostatně, vždy pouze ve spojení s uživatelem, událostí nebo areálem pevných kontrol.

5.1.1.3 Událost

Událost je entita, která reprezentuje vícero různých událostí. Mezi tyto události patří následující tří entity - Závod, Soustředění a Trénink. V entitě jsou skryty informace, které mají tyto tři typy událostí společné. Mezi informacemi jsou mimo jiné název, informace o události napsané administrátorem, údaj o ukončení události a vyúčtování nebo datum události a datum konce přihlášek.

5.1.1.4 Závod

Závod je jedna z událostí zděděná z entity Událost. Tato entita značí reálné závody, které jsou pro českou republiku vypsány a zvoleny oddílem jako podporované. Pro závod je v entitě uchován například typ, disciplína, pořadatel, vzdálenost a zdali se zařizuje ubytování.

5.1.1.5 Soustředění

Soustředění je další z událostí zděděná z entity Událost. Tato entita reprezentuje jednotlivá soustředění, které oddíl pořádá. Pro soustředění je důležité znát datum konce, vše ostatní má již od entity událost.

5.1.1.6 Trénink

Trénink je poslední z událostí zděděná z entity Událost. Touto entitou se rozumí tréninky, které oddíl pořádá a nabízí. Kromě informací získané z entity událost je potřebný typ tréninku.

5.1.1.7 Auto

Auto je reálné auto přidané do informačního systému některým uživatelem jako jeho vlastník. Využívá se pro automodul jednotlivých závodů, kde se auta přidávají a následně se do nich uživatelé hlásí. Auto má důležité informace jako jsou název nebo počet míst.

5.1.1.8 Areál

Areál reprezentuje reálný areál pevných kontrol vytvořený oddílem v určité lokalitě. Každý areál obsahuje několik různých kontrol a výsledků. Jeho hlavními informacemi jsou název, datum začátku a konce nebo počet kontrol.

5.1.1.9 Výsledek

Výsledek značí jednotlivé výsledky zadané závodníky do informačního systému k areálu pevných kontrol. Každý výsledek obsahuje určitý počet kontrol. Výsledek má informace jako jméno a příjmení závodníka, věk a počet správných kontrol.

5.1.1.10 Kontrola

Kontrola je každá kontrola v lese postavená pro areál pevných kontrol oddílem. Tyto kontroly jsou přiřazeny k danému areálu. Dále také značí kontroly, které byly přidány jako výsledek k některému areálu pevných kontrol. Nejdůležitější informací je tedy správnost, kód a odpověď.

5.1.1.11 Příspěvek

Příspěvek reprezentuje veřejnosti přístupné příspěvky přidané na hlavní stránku informačního systému. Tyto příspěvky mohou být vytvořeny buďto pod nějakou událostí nebo nezávisle. Tato entita nese název, datum a text daného příspěvku.

5.1.1.12 Komentář

Komentář skrývá všechny komentáře, které se v informačním systému vyskytují. Tyto komentáře přidává kdokoliv zvenčí a je možné je přidat k příspěvkům nebo areálu pevných kontrol. Obsahuje pouze jméno a text.

5.1.1.13 Členství

Členství je entita, která značí jednotlivá členství v oddíle. Na základě těchto členství se určuje, kolik peněz uživatel platí oddílu a kolik oddíl přispívá uživateli za dopravu a ubytování. Tato entita má název, příspěvek, cestovné a ubytování.

5.1.1.14 Skupina

Skupina reprezentuje jednotlivé skupiny v oddíle. Pomocí těchto skupin se zjišťuje, zdali má uživatel nárok na příspěvek od oddílu na dané úrovni závodu. Entita tedy obsahuje název skupiny a jednotlivé úrovně závodů.

5.1.1.15 Transakce

Transakce obsahuje všechny pohyby, které proběhly na osobních účtech. Těmito pohyby se myslí především příjmy a výdaje. Důležitými informacemi jsou částka, druh (příjem nebo výdaj) a datum.

5.1.1.16 Typ

Typ je entita reprezentující jednotlivý typ transakce. Za takový typ může být považováno například vyúčtování, příspěvky nebo nějaká událost. Tato entita má pouze název tohoto typu.

5.1.2 Vztahy

V této sekci jsou uvedeny některé vztahy entit, které nejsou úplně jasné z diagramu nebo obsahují nějaké informace. Dále také vztahy M:N, které musí být v databázi provedeny pomocí tabulky.

5.1.2.1 Událost - Závod, Soustředění, Trénink

Mezi Entitami Událost a Závod, Soustředění nebo Trénink je vztah generalizace/specializace. Tento vztah je zde použit jako specializace, což znamená dělení vyšší entity (Událost) na různé nižší entity (Závod, Soustředění a Trénink). Ty zdědí atributy a vztahy vyšší entity. Tyto nižší entity jsou tedy specifičtější rozdělení entity vyšší.

5.1.2.2 Místo

Vztah místo mezi entitami User, Událost a Auto je výjimečný v tom, že se jedná o vztah terciální. Terciální vztah spojuje tři různé entity a v ER diagramu se značí kosočtvercem. Místo uchovává informace o tom, který uživatel jede na událost ve kterém autě. Využívá se tedy v komponentě automodul u událostí. V relační databázi se musí implementovat pomocí samostatné tabulky.

5.1.2.3 Ubytování

Tento vztah reprezentuje ubytování u jednotlivých závodů. Zajištuje informace o jednotlivých uživatelích, kteří si na daný závod zvolili možnost, že chtějí ubytování. Je to vztah M:N a v relační databázi bude implementován samostatnou tabulkou. Má navíc také jeden atribut, kterým je dodatek uživatele k ubytování.

5.1.2.4 Uživatel-Událost a Auta-Událost

Tyto dva vztahy reprezentují jednotlivé uživatelé a auta, které jedou na daný závod. Jsou to vztahy M:N a v relační databázi budou realizovány samostatnými tabulkami. Oba mají atribut informace. Dále pak

Uživatel-Událost má atribut kategorie, kterou daný uživatel běží a Auta-Událost má atribut počet míst, kolik je v daném autě nabízeno.

5.2 Použité technologie

Pro implementaci informačního systému je nutné mít ujasněné, v čem se bude každá část programovat a jaké technologie se k tomu využijí. Toto je popsáno v této části návrhu, která se dále dělí na server, databázi a klienta. Nejsou zde popsány pouze jednotlivé technologie, ale také stručné rozhodnutí, proč jsou tyto technologie použity.

5.2.1 Server (back-end)

Mezi původními technologiemi bylo PHP, ve kterém měl být informační systém vytvářen. Toto se však změnilo během prvních měsíců po zadání bakalářské práce. Důvodem k tomu byl předmět informační systémy, ve kterém bylo potřeba vytvořit projekt menšího informačního systému. Pro tento projekt byl hledán jazyk a framework, ve kterém by tvorba byla nejzajímavější, nejpřehlednější a také nejzábavnější. Projeto bylo mnoho frameworků PHP, avšak žádný z nich nevyhovoval. To bylo nejspíše způsobeno osobní neoblíbeností jazyka PHP. Pokračování hledání šlo tedy dále. Objevil se JavaScript framework Node.js, který na pohled vypadá zajímavě, ale jeho nepřehlednost se nedá tolerovat. Jazyk Ruby a jeho Ruby on Rails také nebylo to pravé. Jasnou další zastávkou byl Python, také jeden z nejoblíbenějších jazyků. Jakmile nastalo zjištění, že existují dva velmi silné Python frameworky, bylo vyhráno. Stačilo se tedy pouze rozhodnout, který z nich zvolit. Django a Flask, to jsou jejich názvy. Django je velký komplexní framwork, zatímco Flask je minimalistický na styl puzzle. Právě díky minimalismu vyhrál framwork Flask. Ten je také ve vytvářeném systému použit.

5.2.1.1 Flask Framework

Flask framework je napsán a vytvořen v jazyce Python. Nejčastěji se o tomto frameworku mluví jako o minimalistickém nebo mikro, často se ale také najde ve spojení s puzzle. Slovo puzzle tento framework vystihuje dokonale, protože je to víceméně taková skládačka. Tento framwork nenutí vývojáře používat vybrané knihovny, protože sám vývojář si určí, které knihovny použije. Tudíž si vývojář vybere různé podporované knihovny a poskládá si z nich jeden komplexní framework, který bude sestaven přímo na míru danému systému a nebude nic chybět ani nic navíc. Díky tomuto si tento framework získává stále větší oblibu oproti například komplexnějšímu frameworku Django, který má už vše předem určené. I přesto že je to mikro framework, tak přichází s několika prvky v základním balíčku. Těmito prvky jsou například debugger, podpora unit-testů nebo zabezpečených cookies a Jinja2 šablonovací systém. [8]

MVC (Model-View-Controller)

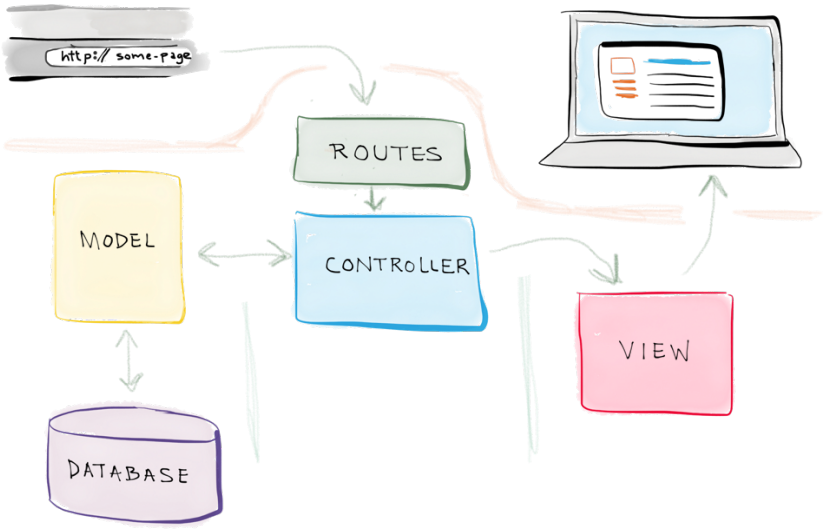
Flask se dá sestavit, aby fungoval na principu MVC. Základní myšlenkou MVC je oddělení logiky od výstupu. Rozděluje aplikaci na tři samostatné části, kterými jsou Model, View (pohled) a Controller (řadič). Zjednodušeně se dá říci, že Controller přijme vstup a rozhodne, co se má dít. Dále řekne Modelu, aby mu vrátil potřebná data a View to poté zobrazí uživateli. [9]

- **Model**

  Zajišťuje aplikační logiku a data z databáze.
- **View**

  Zobrazuje uživatelské rozhraní uživateli pomocí šablonovacího systému Jinja2.
- **Controller**

  Hlavní část, která spojuje a řídí dvě předcházející. Jeho cílem je získat data z modelu a předat je šabloně.



Obrázek 3: MVC model, zdroj: https://realpython.com/blog/python/the-model-view-controller-mvc-paradigm-summarized-with-legos/

Šablonovací systém Jinja2

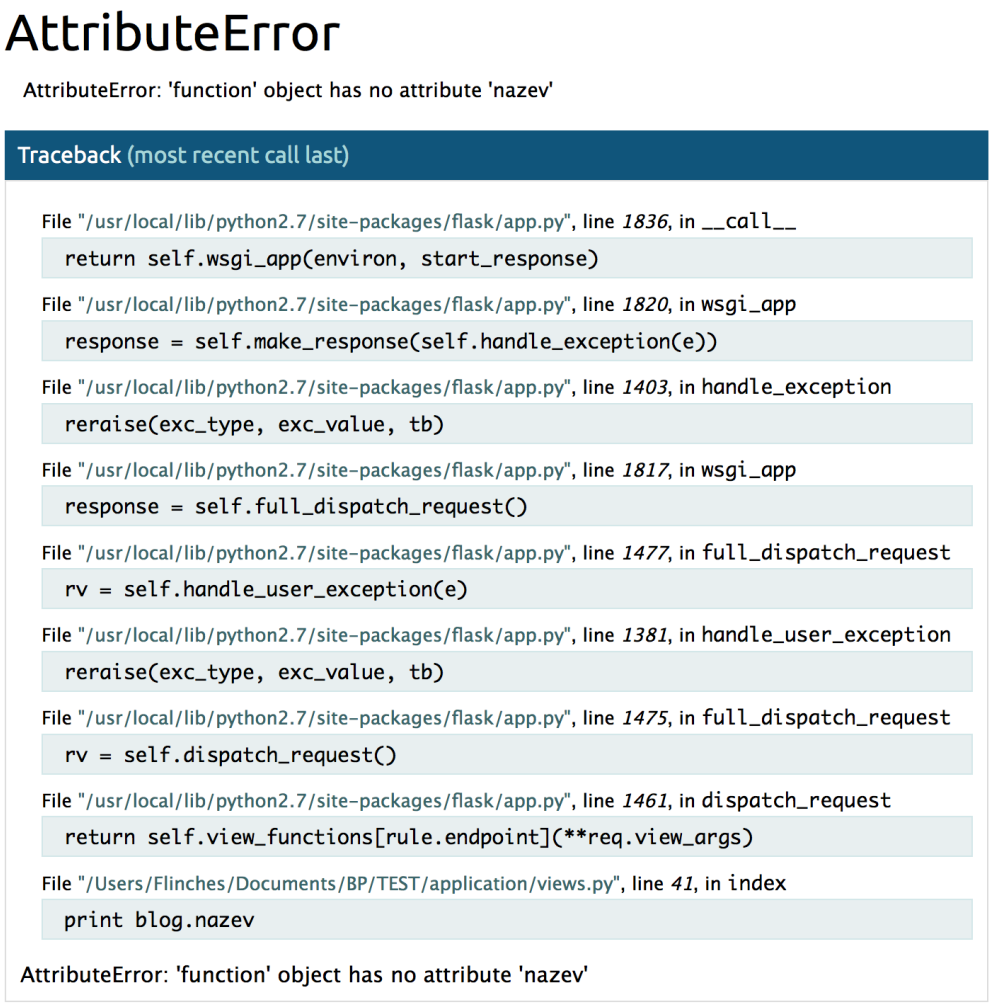
Základní důvod vzniku šablonovacích systémů je oddělení kódu aplikace od zobrazovaného vzhledu aplikace. Toto je z důvodu přehlednosti aplikace a možnosti práce programátorů odděleně od designérů, kteří nemusí znát kód aplikace. Šablonovací systém Jinja2 je zahrnut v celém balíčku Flask frameworku. Je tedy vytvořen pro jazyk Python a vychází z Django syntaxe. Jednotlivé šablony se volají z kódu serveru, ke kterým se přidají proměnné, které se chtějí používat a být přístupné z dané šablony. Pro přístup k proměnným se využívají dvě složené závorky "`{{`" a "`}}`" a pro výrazy složená závorka se znakem procenta "`{%`" a "`%}`". Za výrazy se zde považují jednotlivé bloky, podmínky, deklarace proměnných a další. Dají se zde používat také cykly, které pomohou ve vypsání něčeho, kde nevíme kolik se nachází prvků.

```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

Kód 1: Ukázka šablony Jinja2, zdroj: http://jinja.pocoo.org/docs/dev/

Ladění chyb

Pro každého programátora by měla být velice důležitá část programu hledání a ladění chyb. Ve framworku Flask je naštěstí toto zahrnuto a stačí pouze při spuštění aplikace nastavit, že chceme režim debug. Tento debugger nám při zobrazení chyby vypíše název a důvod nebo popis chyby. Dále pak jednotlivé soubory a funkce, kterými se prošlo, než se narazilo na chybu, neboli takzvaný Traceback. Tudíž je zde vidět i řádek kódu aplikace, který chybu vyvolal. Tento řádek se dá rozkliknout a zobrazí se část kódu, ve kterém se daný řádek nachází. Pro odladění, z jakého důvodu se chyba vyskytla, se dá použít konzole, která je k dispozici. Tato konzole se dá vyvolat pro kterýkoliv soubor a jeho řádek, který se nám zobrazí, počítaje i s řádky z části kódu po rozkliknutí.



Obrázek 4: Debugger frameworku Flask

5.2.2 Databáze

V případě databáze nebylo tolik rozhodování, jako v případě programovacího jazyka a frameworku pro server. Hlavní požadavky na databázi byly dobrá přístupnost, rychlost a hlavně aby byla zdarma. Tyto požadavky nesplňuje mnoho technologií na databáze. Úplně tou nejjednodušší formou je SQLite. SQLite je sice skvělé svou formou, kterou je jeden soubor na disku, ale nehodí se používat jako primární databáze. Tato technologie byla tedy zvolena spíše pro vývoj a testování, kvůli lepší přístupnosti a rychlým změnám v databázi. Dále jsou zde další dvě technologie databází, které jsou asi dva největší rivalové. Jsou to MySQL a PostreSQL. PostgreSQL má výhodu ve větším počtu funkcí a možností, avšak zaostává v jedné z důležitých částí, kterou je rychlost. MySQL není sice tak komplexní a nenabízí tolik možností, pro vyvíjený systém je však plně dostačující a na rozdíl od PostgreSQL se vyznačuje rychlostí. Použitou databázovou technologií je tedy MySQL a pro vývoj a testování SQLite.

5.2.2.1 MySQL

MySQL je databázový systém založený na objektově relační databázi. Jeho největší předností je rychlost, pro kterou byl od začátku svého vývoje optimalizován. Z tohoto důvodu také postrádal mnoho rozšíření, jako jsou například pohledy nebo triggery. To se však v poslední době již změnilo a tyto možnosti začaly být doplňovány. Jak již název napovídá, využívá se při programování jazyk SQL. MySQL je velice populární a má mnoho příznivců i odpůrců. Odpůrci tvrdí, že jeho popularita spočívá především v rozšířenosti u webhostingových poskytovatelů. Zatímco příznivci tvrdí, že je těmito poskytovateli nabízen právě z důvodů kvality. Oboje bude mít svou pravdu a není třeba o tom polemizovat. Další výhodou kromě rozšířenosti je jednoduchá spolupráce s jakýmikoliv nástroji a technologiemi třetích stran, v tomto případě hlavně s Flask frameworkem. Velké plus má také možnost použití phpMyAdmin pro správu databáze. [10]

5.2.3 Klient (front-end)

Pro vzhled, který se bude zobrazovat na straně klienta, většinou v nějakém prohlížeči, se využívá webová technologie HTML. Ta je využita prostřednictvím frameworku Flask, který má v sobě zakomponovanou šablonovou technologii Jinja2. Díky tomu je možné vytvořit HTML šablonu, která se poté volá ze strany serveru a doplňují se do ní zvolené informace. Pro vzhled HTML se většinou používá technologie CSS a není tomu jinak ani zde. Avšak není použito vlastní CSS, nýbrž technologie, která nabízí zdarma vlastní CSS soubor. Díky tomuto CSS souboru je tvorba vzhledu stránky mnohem jednodušší a přitom stále vypadá skvěle. Tato technologie se nazývá Bootstrap. Dále je zvolena technologie JavaScript pro editor příspěvků nebo galerii fotek u příspěvku.

5.2.3.1 Bootstrap

Tvorba vzhledu webových stránek nebyla nikdy pohodlnější a to vše díky technologii Bootstrap (Dostupný z: http://getbootstrap.com). Sen každého programátora, který nemá rád tvorbu vzhledu webových stránek, který si buď pronajímal designera nebo jeho vzhled nevypadal profesionálně. Pomocí Bootstrapu si stačí pouze zvolit jednu z mnoha šablon, které jsou dostupné a stáhnout si její CSS soubor. Poté už jenom v HTML kódu používat třídy a prvky, které jsou v tomto CSS souboru použity. Jeho další výhodou je také využívání komponent JavaScriptu, takže mnoho jeho prvků je dynamických, jako například vysunovací menu a podobné.

5.2.3.2 WYSIWYG editor

WYSIWYG je zkratka věty *"What You See Is What You Get"*, neboli česky *"Co vidíš, to dostaneš"*. Tato zkratka se využívá se spojením editace a znamená to, že verze zobrazená je vzhledově totožná s verzí výslednou. V tomto režimu pracují nejznámější textové editory jako například Microsoft Word nebo OpenOffice Writer. Kromě samostatných programů se dá využívat také na webových stránkách například pro tvorbu příspěvků na blogu. Uživateli to usnadní práci, nemusí psát žádné značky ani přímo HTML kód, aby jeho text měl grafickou podobu. Dále tyto editory podporují vkládání obrázků a souborů, což je potřeba. Použitá technologie WYSIWYG ve vytvářeném systému je CKEditor (Dostupný z: http://ckeditor.com).

5.2.3.3 Lightbox

Pro zobrazování obrázků a fotek v příspěvcích a informacích je použita technologie Lightbox (Dostupný z: http://lokeshdhakar.com/projects/lightbox2). Lightbox je napsán v jazyce JavaScript, tudíž umožňuje dynamické zobrazování na webové stránce. Pomocí vyskakovacího okna, které se díky JavaScriptu vypočte na velikost prohlíženého okna, se zobrazuje obsah. Toto je možné pro samostatné jednotlivé fotky, ale taky pro skupinu fotek, které se určí jako galerie. V případě galerie se poté dá listovat mezi danými fotkami.

6 Implementace

Tato kapitola pojednává o implementaci informačního systému pro oddíl orientačního běhu. K tomu byl využit framework Flask, který je popsán v kapitole 5. Při implementaci bylo čerpáno ze dvou knih [1][2], které jsou psány jako tutoriál.

Jsou zde popsány využité komponenty, které pomáhají zrealizovat určité požadavky. Dále pak způsob stahování dat z informačního systému ORIS. Poslední částí je popis jednotlivé funkčnosti, která je netriviální a pro chod systému nezbytná.

6.1 Využité komponenty

V této části je výčet použitých komponent a jejich využití ve vytvářeném informačním systému. Tyto komponenty nějakým způsobem ovlivňují implementaci nebo mění nějakou již implementovanou část na vylepšenou.

6.1.1 SQLAlchemy

Pro tvorbu databáze a komunikace s ní je použita knihovna SQLAlchemy patřící pod framework Flask. Tato knihovna je nezbytná pro uskutečnění implementace podle modelu MVC (Model-View-Controller). V MVC zastává funkci modelu, který pomáhá v komunikaci s databází. SQLAlchemy je řeší implementaci databáze pomocí ORM (Object-relational mapping). Z vytvořených modelů vytvoří celou databází, kde k jednotlivým tabulkám se potom dá přistupovat jako k objektům. Díky ORM není potřeba psát žádné SQL příkazy, protože je vše psáno v python jazyce, který je na SQL jazyk překládán.

Pro nastavení databáze, kterou bude SQLAlchemy využívat, je potřeba vyplnit proměnnou `SQLALCHEMY_DATABASE_URI`. Do této proměnné se vloží textový řetězec, který bude obsahovat následující: `dialect://username:password@host:port/database`. `Dialect` je název databázové aplikace, v případě vytvářeného systému MySQL.

Jednotlivé tabulky databáze jsou implementovány pomocí tříd, které dědí z třídy dostupné z SQLAlchemy a jednotlivé sloupce tabulky jako proměnné ve třídě, které využívají z SQLAlchemy metodu `Column`. V této knihovně jsou také jednotlivé možnosti pro sloupce tabulky, jako je například datový typ, primární klíč, implicitní nebo unikátní hodnota a cizí klíč. Vztahy mezi tabulkami jsou řešeny nejen pomocí cizích klíčů, ale taky pomocí metody `relationship`, která tyto tabulky propojí. Díky tomu se k nim dá přistupovat z obou stran jako k objektům a nemusí se využívat možnosti propojení tabulek při vyhledávání. Jednotlivé tabulky (třídy) mohou mít ještě dodatečné metody, které provedou potřebnou funkcionalitu nad danou tabulkou.

6.1.2 WTForms

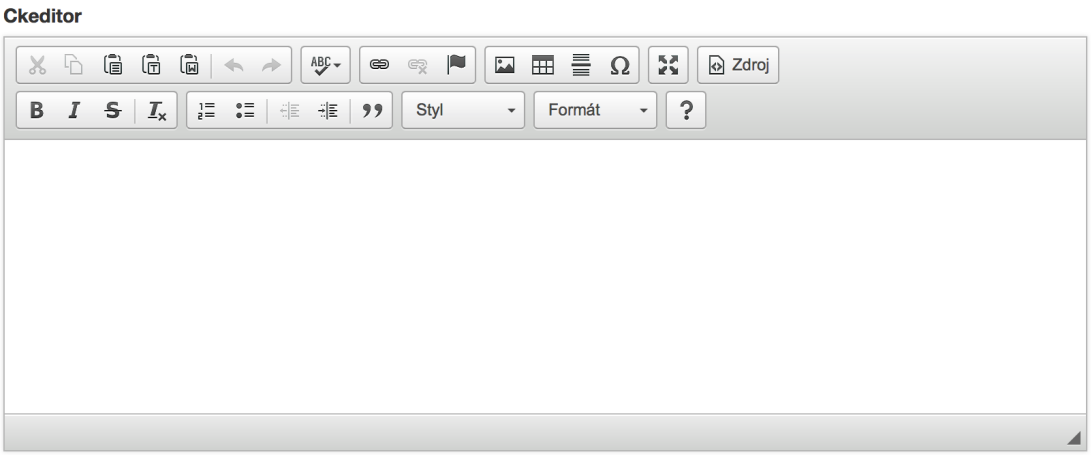
Pro usnadnění tvorby a zpracování formulářů je v informačním systému využita knihovna WTForms, která patří pod framework Flask. Knihovna přesunuje tvorbu formulářů z HTML kódu do pythonu. Na serveru se potom s jednotlivými formuláři pracuje jako s objekty a jednoduše se ověřuje, zdali byl daný formulář odeslán a na základě toho se s ním pracuje. Jednotlivé formuláře jsou implementovány jako třídy, které dědí funkcionalitu z třídy nadřazené patřící pod knihovnu WTForms. Každá taková třída má několik proměnných, které reprezentují jednotlivé prvky ve výsledném formuláři. Tyto proměnné mají nastavenou funkci, která určuje typ prvku ve formuláři, například `IntegerField` pro pole čísel nebo `StringField` pro text. Funkce určující typ prvku má dva parametry, které daný prvek ovlivňují. Tím hlavním parametrem je název, který se u daného prvku ve formuláři zobrazí. Druhým parametrem je ověřování prvku, mezi které patří například povinnost vyplnění, délka textu nebo omezení čísla. Pro každé ověření prvku se dá také vyplnit chybová zpráva, která se vypíše v případě, že formulář neprojde validátorem.

6.1.3 CKEditor

CKEditor patří pod množinu WYSIWYG editorů a je to jeden z mála kvalitních nabízených zdarma. Informace o tom, co to WYSIWYG editor je a k čemu se využívá jsou popsány v kapitole 5.

Jeho implementace do informačního systému byla poměrně snadná, pouze přidávání obrázků a souborů na server bylo komplikovanější. Jeho využití je pouze pro manipulací s textem u příspěvků a událostí. Základem tohoto editoru je prvek `textarea` v HTML, který je však vytvořen pomocí WTForms `TextAreaField`. Tento prvek je pomocí JavaScriptu převeden přímo do CKEditoru.

Důležité je doplnit funkce pro přidávání obrázků a pro přidávání souborů. Tyto dvě funkce pracují víceméně stejně až na to, že funkce pro přidávání obrázků upravuje velikost obrázků do dvou formátů. Jeden malý pro zobrazení v textu a druhý velký pro zobrazení přímo obrázku. Tyto velikosti obrázků jsou vypočítávány tak, aby menší měl na výšku 150 pixelů a ten větší 600 pixelů. Názvy souborů se generují automaticky pomocí funkce, která vytvoří časové razítko, ve kterém je rok, měsíc, den, hodina, minuta a sekunda. K tomuto časovému razítku se přidá náhodné čtyřmístné číslo a soubor se uloží na server. Ošetřeny jsou také chybové stavy, které se uživateli při přidávání obrázků nebo souborů zobrazí místo uložení.



Obrázek 5: CKEditor

6.1.4 Lightbox

Lightbox je aplikace galerie napsána v JavaScriptu, která umožňuje zobrazování obrázků. Více informací o Lightboxu je popsáno v kapitole 5.

Implementace je poměrně snadná, avšak z důvodů obrázků přidávaných přes WYSIWYG editor musely být učiněný určitá opatření, aby aplikace fungovala správně. Pro zprovoznění aplikace je potřeba do HTML elementu `<a href>` přidat atribut `data-lightbox="název"`, kde název značí název sekce obrázků. U obrázků z WYSIWYG editoru však tento HTML element není. Pomocí knihovny jQuery napsané v JavaScriptu bylo potřeba projít všechny obrázky, které budou zobrazovány a přidat jim tento HTML element. Pro zobrazení obrázků s možností listování mezi nimi musí být název v atributu `data-lightbox` stejný. V případě úvodní stránky, kde je více různých příspěvků se musely projet vždy obrázky jednotlivých příspěvků a tento atribut jim nastavit stejný.

Obrázek 6: Lightbox

6.2 Stahování dat z ORIS

Tato část popisuje jednotlivé prvky vytvářeného informačního systému, které komunikují s informačním systémem Českého svazu orientačních sportů ORIS. Je zde také popsáno jak je komunikace mezi systémy implementována.

6.2.1 Requests

Requests je python knihovna, která umožňuje využívat HTTP pro komunikaci s okolím. Tato knihovna je využita pro stahování dat z informačního systému ORIS. Ke stahování dat se využívají jednotlivé metody API zmíněné v kapitole 3. Tyto metody se volají ve tvaru URL adresy, která se zadá jako parametr funkci requests. Tato funkce navrátí výsledek jako *slovník* (*dictionary*), tudíž ve tvaru klíč a hodnota. Ke klíčům a hodnotám se dá poté jednoduše přistupovat. Dají se také třídit a hledat v nich potřebná data, která jsou důležitá.

6.2.2 Uživatelé

Pro vytváření uživatelů v informačním systému je využita metoda `getRegistration` spadající pod API ORIS. Metoda navrátí všechny registrované závodníky pro daný rok. Tyto závodníky je poté potřeba projít a vybrat pouze ty, kteří patří pod daný oddíl. Toto třízení probíhá pomocí cyklu, který projde všechny závodníky a kontroluje jejich oddíl. Závodníci, kteří zůstanou, se přidají do informačního systému a přidá se k nim zakódované heslo. V případě, že závodník již v informačním systému existuje, proběhne pouze jeho aktualizace údajů.

6.2.3 Závody

Výběr závodů administrátorem, které budou oddílem podporovány, je zprostředkován pomocí metody `getEventList`. Výsledek této metody je seznam všech pořádaných závodů v České republice pro daný rok. Jednotlivé závody jsou reprezentovány řádkem v tabulce, kde se kromě názvu nachází také důležité informace jako například datum, úroveň závodu nebo disciplína. Jeden sloupec tabulky je HTML element `input` typu `checkbox`. `Checkbox` slouží pro výběr jednotlivých závodů, které budou do informačního systému přidány. Pro každý takový závod se použije metoda `getEvent` k získání detailnějších informací. Tyto informace se následně uloží se závodem do databáze systému. Pro smazání závodů už není možné využít HTML element `input` typu `checkbox`. Smazat závod se dá pouze při editaci.

Pro každý závod v systému se stahují informace o přihlášených závodnících. K tomu se využívá metoda `getEventEntries`. Tito závodníci jsou vyhledáni v databázi systému a následně přidáni k závodu. V případě štafetového závodu metoda nevrací seznam závodníku, ale seznam přihlášených štafet. Štafety se ukládají pouze do *pole polí*, kde se pro každou štafetu ukládá jméno a kategorie.

Dále se ke každému závodu stahují informace o platbě za závod. Docílí se toho pomocí metody `getEventBalance`, která vrátí seznam plateb pro každý oddíl. V tomto seznamu se vyhledá daný oddíl a následně se tyto informace o platbě uloží do proměnné, která bude u závodu vypsána pro přehled.

6.3 Funkčnost

Zde se nachází zbývající funkčnost informačního systému, která je nějakým způsobem komplikovaná nebo zajímavá na implementaci.

6.3.1 Přihlašování a odhlašování

Přihlašování a odhlašování v systému je implementováno pomocí knihovny Flask-Login frameworku Flask. Díky této knihovně se vytvoří z tabulky uživatele v modelu databáze přihlášení do systému. Třída uživatele vytvářející tabulku dědí nějaké metody ze třídy knihovny Flask-Login, která usnadní přihlášení. Dále je vytvořena funkce, která se při vytváření připojení volá a vrací daného uživatele, který má být přihlášen. Pro tohoto uživatele se ověří zakódované heslo a pokud se shoduje, je uživatel přihlášen do systému a uložen do `session`. K přihlášenému uživateli se potom v systému přistupuje jako k objektu `current_user`. Pokud uživatel do sekce přihlášení přešel z nějaké jiné stránky než úvodní, bude po přihlášení přesměrován zpátky na tuto stránku. Při odhlášení se uživatel vymaže ze `session` a odhlásí se ze systému.

6.3.2 Odesílání emailů

Odesílání emailů se využívá pro vyúčtování. Spočítá se celková částka pro vlastníky účtů a odešle se jim na email s variabilním symbolem a číslem účtu, na který danou částku mají poslat. K posílání těchto emailů se využívá knihovna Flask-Mail frameworku Flask. Pro správnou funkčnost této knihovny je potřeba nastavit potřebné proměnné pro emailovou adresu. Těmito proměnnými jsou `MAIL_SERVER`, `MAIL_PORT`, `MAIL_USE_TLS`, `MAIL_USE_SSL`, `MAIL_USER`, `MAIL_PASSWORD`. Dále se pro nečekání na odeslání všech emailů využije vytvoření nového vlákna aplikace, ve kterém se zavolá pro každého vlastníka vytvoření a odeslání emailu.

6.3.3 Vyúčtování

Vyúčtování je jedna z komplikovaných částí systému. Vyúčtování má v tomto systému více podob, mezi které patří vyúčtování za závody, vyúčtování za administrátorem zvolenou položku a vyúčtování účtů s oddílem. Každé vyúčtování vytvoří transakci uživateli, kterému bylo provedeno.

6.3.3.1 Vyúčtování za závody

Vyúčtování za závody se provádí až po ukončení události. Toto vyúčtování může provádět administrátor nebo přihlašovatel na daný závod. Těmto dvěma se nabídne formulář, který obsahuje všechny uživatele, kteří se daného závodu zúčastnili. Pro každého uživatele jsou tři pole, které určují částku za cestovné, ubytování a doplňky. Cestovné a ubytování jsou systémem předvyplněné na základě skupiny a členství daného uživatele. Díky skupiny se zjistí, zdali daný uživatel má vůbec nárok na dotaci oddílu na daný závod. Členství potom pomáhá určit přesnou částku, která má být do polí vyplněna. Cestovné se vypočítá pomocí nastavené vzdálenosti na daný závod a koeficientu ze členství. Ubytování se pouze vyplní částka převzata z členství. Jednotlivé částky se při potvrzení přičtou nebo odečtou z účtu každého uživatele.

6.3.3.2 Vyúčtování za zvolenou položku

Vyúčtování za zvolenou položku má možnost pouze administrátor. Tomu bude nabídnut formulář, který obsahuje všechny uživatelé systému a pole pro vyplnění částky vyúčtování. První položkou ve formuláři je typ vyúčtování. Jednotlivé typy vyúčtování může administrátor libovolně přidávat do systému. V případě nastaveného typu příspěvek se automaticky všem z účtu odečte roční příspěvek oddílu podle nastaveného členství uživatele. Ve všech ostatních případech se přičte nebo odečte částka, která ve formuláři bude u jednotlivých uživatelů zadána.

6.3.3.3 Vyúčtování s oddílem

Tato možnost vyúčtování souvisí s nastavením vlastnictví účtů. Pokud má některý z uživatelů nastavené vlastnictví účtů, tak se pro něj vypočte celková částka všech účtů dohromady a s tou se pracuje. V případě že uživatel vlastní pouze svůj účet, tak se pracuje s touto částkou. Když chce administrátor toto vyúčtování provést, vyskočí na něj formulář, ve kterém jsou pouze vlastníci účtů. Těmto vlastníkům účtů poté zadá částku, kterou oddílu zaplatili na účet nebo osobně. Postupně se projedou všechny účty každého vlastníka, u kterých se postupně provede vyúčtování. Každý účet vlastníka se jeden po druhém vynulují a hodnota, o kterou byl účet změněn se přičte nebo odečte od celkové částky. U vlastníka se poté pracuje se zbývající celkovou částkou, která se vlastníkovi přičte nebo odečte na účet, podle toho jestli byla kladná nebo záporná.

6.3.4 Areály pevných kontrol

Areály pevných kontrol jsou implementovány pomocí tří částí, kterými jsou samostatné areály, jednotlivé kontroly a výsledky. Areály může vytvářet a upravovat pouze administrátor systému. Ten při vytváření určuje název, datum začátku, datum konce, počet kontrol a další informace. Dále se mu na základě zadaného počtu kontrol vytvoří formulář s předdefinovanými pořadovými čísly, ke kterým pouze přidá identifikační písmeno. Kontroly vytvořené přímo z areálu se nastaví jako primární a automaticky se ohodnotí jako správné. Nepřihlášený uživatel má možnost přidat svůj výsledek k takto

vytvořenému areálu. Vyskočí na něj formulář, do kterého doplní své jméno, příjmení, věk a dále identifikační písmena ke kontrolám. Tento výsledek uživatele se zkontroluje s primárními kontrolami areálu a zařadí jej do celkových výsledků, kde bude vidět úspěšnost. Možnost přidávání výsledků bude zablokována ve chvíli, kdy aktuální datum bude vyšší než datum konce. V takovém případě se bude dát stále podívat na celkové výsledky.

6.3.5 Možnosti kolem událostí

Možností kolem událostí je více, většina se ale točí kolem různých přihlašování spojených s událostmi. První z možností je přihlášení se na událost. To je možné pouze u tréninků a soustředění, protože u závodů se uživatelé přihlášení na událost stahují z informačního systému ORIS. Další možností je přihlášení ubytování na závod. Tato možnost je možná pouze v případě, že závod má nastavené ubytování. Poslední možností přihlašování je přihlášení se do štafety v rámci štafetového závodu. Všechny tyto přihlašování mají zároveň i možnost odhlašování. Toto vše je implementováno způsobem, že každý uživatel může přihlásit každého uživatele. V případě přihlašování jsou nabídnuti všichni uživatelé, kteří ještě nejsou přihlášení a pro odhlašování zase pouze uživatelé, kteří jsou již přihlášení. Dále se události nastaví jako ukončené samy v případě, kdy aktuální datum bude větší než datum přihlášek události.

6.3.6 Automodul

Automodul je část systému u událostí, která pomáhá organizovat dopravu na jednotlivé události. Jeho hlavními prvky jsou samostatná auta. Auta si mohou uživatelé vytvářet ve svých profilech, stačí zadat název a počet míst, popřípadě dodatečné informace. Tyto auta je potom možnost přidávat do automodulu na vybrané události. Přidání může provést pouze majitel auta, který bude automaticky nastaven jako řidič. U auta nastaví informace pro ostatní uživatelé a počet nabízených míst, který musí být roven nebo menší počtu míst daného auta. Tyto informace může majitel upravovat, popřípadě může odebrat celé auto z automodulu. Do aut, které jedou na událost (jsou přidány do automodulu), se mohou hlásit jednotliví uživatelé. Ve výpisu automodulu je vidět u každého auta, kdo v něm jede, kolik nabízí míst, kolik jich ještě zbývá volných a informace napsané majitelem.

7 Testování a možnosti dalšího vývoje

Tato kapitola se zabývá testováním a možnostmi dalšího vývoje vytvořeného informačního systému.

Testování je disciplína, která pomáhá zajistit kvalitu softwaru. Jeho hlavním cílem je nalézt chyby v softwaru a prokázat, že splňuje požadované vlastnosti. Testování se dělí do různých kategorií, například statické a dynamické, černá a bílá skříňka nebo automatické a manuální.

7.1 Testování

Testování aplikace probíhalo především průběžně, aby bylo objeveno co nejvíce chyb. Subjektů pro testování bylo několik a testy probíhaly vždy, když se přidalo do systému něco komplexnějšího. Tudíž bylo testování manuální. Dále bylo testování dynamické, protože byla potřeba pracovat se systémem samotným a v případě subjektů se jednalo o černou skříňku, jelikož neměli přístup ke zdrojovým kódům.

7.1.1 Testování programátorem

Programátor by měl svůj produkt testovat neustále. Nebylo tomu jinak ani v tomto případě, kdy programátor testoval každou novou přidanou funkci a zdokonaloval ji, dokud nebyla téměř bez chyby a schopna plného použití. Vždy po několika takto přidaných funkcích, které společně tvořily komplexnější funkci, byl systém zpřístupněn zadavateli k otestování a případným připomínkám.

7.1.2 Akceptační testování

Akceptační testování je testování zadavatelem. Zadavatel si svými prostředky otestuje, zdali výsledný systém splňuje veškerá očekávání a neobsahuje žádné chyby. Toto bylo však díky průběžné konzultaci a testování s vedením oddílu v pořádku a zadavatel potvrdil, že je systém použitelný.

7.1.3 Pilotní testování

Pilotní testování znamená testování nasazení systému. Systém tedy testují skuteční uživatelé, kteří jej budou v budoucnu využívat. Toto testování probíhalo s vybranými subjekty (členy oddílu), kteří byli ochotní pomoct. Testování neprobíhalo formálně, tudíž se nevyplňovaly žádné formuláře ani dotazníky. Subjekty měly možnost vyzkoušet úplně celý systém, který se nacházel na testovací doméně a následně jej ústně zhodnotit.

Díky předchozím průběžným testováním se již neobjevily žádné další chyby v systému. Pouze měly některé subjekty připomínky, co by mohl systém umět jinak nebo navíc.

7.2 Možnosti dalšího vývoje

Tento informační systém je vytvářen pro reálný oddíl orientačního běhu, ve kterém je autor členem. Díky tomu bude tento systém stále zdokonalován, protože cokoliv bude potřeba pro zlepšení chodu oddílu, se může kdykoliv přidělat.

7.2.1 Zdokonalení areálů pevných kontrol

Areály pevných kontrol mají momentálně stanoviště, na kterých je pořadové číslo a identifikační písmenko, pomocí kterého se určuje správnost. Každý účastník takového areálu má možnost zadat si svůj výsledek do informačního systému, který ho zařadí do výsledků. V budoucnosti je v plánu vylepšení řešení výsledků pomocí vytvoření aplikace a přidání QR kódů na kontroly. V takovém případě si účastník pouze stáhne aplikaci do chytrého telefonu, kde si vytvoří účet se svým jménem. Touto aplikaci poté načte jednotlivé QR kódy kontrol a po ukončení zadá odeslat. Výsledek by se automaticky odeslal na server a zařadil mezi všechny ostatní výsledky do informačního systému.

7.2.2 Komunikace

Hlavním prvkem rozšíření informačního systému by mohla být komunikace pomocí posílání zpráv. Zatím komunikuje pouze systém s uživatelem pomocí emailů. Neumožňuje však komunikaci uživatelů mezi sebou. Toto by mohlo být zprostředkováno jako chat, kde by mezi sebou komunikovali. Tento chat by mohl mít upozornění o nové zprávě a také by mohl mít vyskakovací okno pro rychlou komunikaci.

Další možností z hlediska komunikace by mohlo být rozšíření komunikace systému s uživatelem. Kromě posílání emailů o zaplacení příspěvků oddílu by si mohl uživatel nastavit, jaká upozornění mu na email budou chodit. Příkladem takových upozornění by mohlo být vytvoření nové události, přidání příspěvku nebo nová zpráva v případě předchozího rozšíření.

7.2.3 Pořádané akce pro veřejnost

Jednou z možností rozšíření je přidání jednotlivých stránek, které budou sloužit pořádaným akcím pro veřejnost. Momentálně jediné akce pro veřejnost v informačním systému jsou areály pevných kontrol. To by se však časem mohlo změnit a mohly by přibýt další akce jako malé veřejné závody (například Opavské Kufrování).

Dále by mohla přibýt možnost nastavení veřejnosti u tréninků. Tyto tréninky by pak byly zobrazeny podobně jako areály pevných kontrol pro veřejnost. Lidé by se v takovém případě mohli na tyto tréninky hlásit v informačním systému.

7.2.4 Vzhled

Nepatrným vylepšením by mohl být také vzhled aplikace. Prozatím je aplikace vytvořená v technologii Bootstrap, která poskytuje svou šablonu pro tvorbu internetových aplikací. Nebylo by však špatné vytvořit vlastní šablonu, která by přesně odpovídala potřebám systému a zdokonalila tak i vzhled.

8 Závěr

Cílem této práce bylo vytvořit informační systém pomocí webových technologií pro řešení organizace v rámci oddílu orientačního běhu. Zadavatelem práce byl tedy oddíl orientačního běhu v Opavě, který se velice rychle rozrůstá a tak pro něj není snadné nadále řešit vše pomocí emailů.

Bylo zapotřebí prozkoumat informační systém Českého svazu orientačních sportů ORIS a jeho API. Provedlo se tedy detailní prozkoumání tohoto systému a následně byly vybrány vhodné metody API pro stahování dat. Jedná se konkrétně o stahování jednotlivých závodů včetně podrobných informací. Dále bylo realizováno získání dat o uživatelích spadajících pod daný oddíl. Podrobněji se o tomto informačním systému a jeho metodách API pojednává v kapitole 3.

Analýza požadavků je nedílnou součástí tvorby informačních systémů a je jedna z prvních, protože určuje, co má systém vše umět. Úspěšná analýza byla provedená díky vedení oddílu orientačního běhu v Opavě, který byl ochotný přistoupit na pravidelné schůzky a konzultace ohledně funkčnosti. K tomu byl vytvořen také diagram případů užití, který graficky zobrazuje chování systému a možnosti jednotlivých uživatelů, neboli aktérů.

Na základě zanalyzovaných požadavků bylo možné sestavit návrh informačního systému. Hlavním prvkem tohoto návrhu byl ER diagram, který znázorňuje ukládání dat do databáze pomocí jednotlivých entit a jejich vzájemných vztahů. Druhou částí návrhu bylo zvolení vhodných implementačních technologií. Osobní preference jazyka Python vedly ke zvolení frameworku Flask pro tvorbu serveru. Databáze kvůli oblíbenosti, rychlosti a jednoduché použitelnosti byla zvolena MySQL. Klient je implementován standardně v jazyce HTML s využitím JavaScriptu a šablonovacího systému Jinja2 součástí frameworku Flask.

Testování systému probíhalo průběžně s pomocí vybraných členů oddílu. Proběhlo také akceptační a pilotní testování, které prokázaly plnou funkčnost informačního systému. Systém bude nadále zdokonalován a vyvíjen dle potřeb oddílu. Již nyní je mnoho nápadů, jako například přidání aplikace se čtečkou QR kódů pro vylepšení areálů pevných kontrol.

Hlavní cíle práce se povedlo splnit a tudíž implementace proběhla úspěšně. Kapitola 6 popisuje podrobně důležité a také netriviální části implementace informačního systému. Jeho aktuální verze je dostupná na adrese www.fhadac.cz.

Literatura

[1] GRINBERG, Miguel. *Flask web development*. First edition. Sebastopol, CA: O'Reilly, 2014. ISBN 1449372627.

[2] AGGARWAL, Shalabh. *Flask Framework Cookbook*. First edition. Birmingham, UK: Packt Publishing, 2014. ISBN 9781783983407.

[3] *What is PHP?* [online]. c2001-2016. [cit. 2016-05-10]. Dostupné na URL: <http://php.net/manual/en/intro-whatis.php>

[4] *Python about* [online]. c2001-2016. [cit. 2016-05-10]. Dostupné na URL: <http://python.org/about>

[5] *About Ruby* [online]. [cit. 2016-05-10]. Dostupné na URL: <http://ruby-lang.org/en/about>

[6] JANOVSKÝ, Dušan.: *Úvod do JavaScriptu* [online]. [cit. 2016-05-10]. Dostupné na URL: <http://jakpsatweb.cz/javascript/javascript-uvod.html>

[7] MALÝ, Martin. REST: architektura pro webové API. *Zdroják* [online]. 2009 [cit. 2016-05-10]. ISSN 18035620. Dostupné na URL: <https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api>

[8] RONACHER, Armin: *Welcome to Flask* [online]. 2013 [cit. 2016-05-10]. Dostupné na URL: <http://flask.pocoo.org/docs/0.10/>

[9] BERNARD, Borek. Úvod do architektury MVC. *Zdroják* [online]. 2009 [cit. 2016-05-10]. ISSN 18035620. Dostupné na URL: <https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc>

[10] ZAJÍC, Petr. MySQL (1) - pestrý svět databází. *Linux Software* [online]. 2005 [cit. 2016-05-10]. ISSN 18013805. Dostupné na URL: <http://www.linuxsoft.cz/article.php?id_article=731>

[11] Webová aplikace (Web Application). *ManagementMania* [online]. c2011-2013, aktualizováno 2016-03-25 [cit. 2016-05-10]. ISSN 23273658. Dostupné na URL: <https://managementmania.com/cs/webova-aplikace-web-application>

Seznam příloh

A Obsah CD

- **`/xhadac02_aplikace/`** - zdrojové kódy aplikace
- **`/xhadac02_instalace/`** - návod instalace a spuštění aplikace ve formátu PDF
- **`/xhadac02_BP_kod/`** - zdrojový soubor bakalářské práce ve Wordu
- **`/xhadac02_BP/`** - bakalářská práce ve formátu PDF
- **`/xhadac02_zip/`** - jednotlivé komprimované části
- **`/xhadac02_readme.txt`** - popis adresářů

B Instalace a spuštění

B.1 Instalace

B.1.1 Prerekvizity

Před samotným spuštěním aplikace je potřeba mít nainstalované určité komponenty, které jsou:

- `Flask`
- `Flask-SQLAlchemy`
- `Flask-WTF`
- `Flask-Mail`
- `Flask-Login`
- `Flask-Bootstrap`
- `Requests`
- `Pillow`
- `Databázový server`

B.1.2 Konfigurace

Je potřeba vstoupit do složky aplikace. Všechny prerekvizity kromě databázového serveru se nainstalují pomocí příkazu v terminálu:

```
pip install -r requirements.txt
```

Dále je potřeba nastavit databázi v konfiguračním souboru `config.py`. Přesněji je potřeba vyplnit danou proměnnou příslušným textovým řetězcem, kde `dialect` je použitá databázová technologie:

```
SQLALCHEMY_DATABASE_URI =
"dialect://username:password@host:port/database"
```

B.2 Spuštění

Pomocí následujícího příkazu se spustí aplikace:

```
python app.py
```

Ve výchozím nastavení se aplikace spustí v režimu `localhost` na portu `8000`.