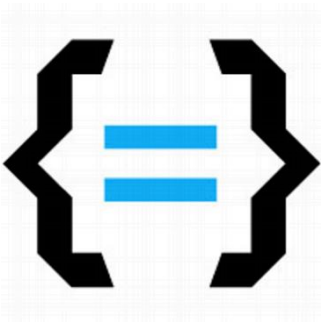
Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod



Mobilní aplikace pro české a mezinárodní hudební soutěže

Diplomová práce

Autor: Bc. Jan Motyčka
Studijní obor: Aplikovaná informatika, 2. ročník

Vedoucí práce: Ing. Pavel Kříž, Ph.D.

Hradec Králové Srpen 2018

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 04.08.2018 Jméno a příjmení

Poděkování:

Děkuji vedoucímu diplomové práce Ing. Pavlu Křížovi, Ph.D. za pomoc a odborné rady při zpracování této práce.

Anotace

Práce se zaměřuje na popis tvorby aplikace, vytvářené pod záštitou firmy JPH Software. Bude sloužit základním uměleckým školám a jednotlivcům z hudebního odvětví pro celkovou správu českých a mezinárodních hudebních soutěží. V práci jsou popisovány jednotlivé kroky vývoje mobilní a webové aplikace od prvotních uživatelských cílů přes analýzu a implementaci až k uživatelské příručce. Mimo jiné jsou zmíněny použité platformy, na kterých je aplikace postavena a také jisté přínosy pro uživatele. Na samotném závěru je zhodnoceno splnění cílů, které byly vytyčeny a také zpětná vazba od koncových uživatelů a potencionální budoucí směr, jakým by se mohla aplikace ubírat.

Annotation

Title: Mobile application for Czech and international music competitions

This work focuses on the description of the creation of an application created under the auspices of JPH Software. It will serve basic arts schools and individuals from the music industry for the overall management of Czech and international music competitions. The thesis describes individual steps of mobile and web application development from initial user targets through analysis and implementation to the user manual. Among other things, the platforms on which the application is built are also mentioned, as well as certain benefits for the user. In the end, there are the goals that were established, as well as feedback from end-users and the potential future direction the application could take, are evaluated.

Obsah

1. Úvod

Dnešní doba si stále více žádá využívání informačních systémů v nejrůznějších činnostech. Důvodem je rostoucí chuť si co nejvíce ulehčit práci a minimalizovat chyby vzniklé lidským faktorem. Je to tedy jeden z důvodů, proč se oblast informačních technologií velice rychle rozvíjí.

Informační technologie již delší dobu pronikají i do odvětví umění a vzniká tak větší poptávka softwaru na míru i v této oblasti. Tak vznikl požadavek vytvořit aplikaci, která by byla uplatnitelná na poli českých, ale i mezinárodních hudebních soutěží.

Cílem práce je vybrat vhodné vývojové prostředky a poté navrhnout a vytvořit uživatelsky přívětivou a užitečnou aplikaci pro správu hudebních soutěží. Hlavním přínosem je ušetřit práci pořadatelům soutěží tak, aby se nemuseli starat o vybíraní bodovacích lístků porotců, sčítání jednotlivých bodů a vyhodnocování vítěze. Všechny tyto potřeby pokryje nový systém na míru a přidá další užitečné funkce, do kterých se řadí např. export bodovacích lístků, programu soutěže atd. do různých formátů pro uložení nebo tisk, obsluha přihlášek, odhlašování soutěžících, kontrola bodování apod. Funkční požadavky jsou stanovovány ve spolupráci se Základní uměleckou školou Evy Randové z Ústí nad Labem.

Systém je určen pro personál ve vedení soutěže, porotce, ale i soutěžící a je vyvíjen pod záštitou softwarové firmy JPH Software z Ústí nad Orlicí.

2. Analýza a návrh systému

2.1 Systém a analýza požadavků

Vytvářený systém bude sloužit ke správě vedení hudební soutěže. Funkční požadavky jsou navrhovány podle potřeb z reálné praxe dle Základní umělecké školy Evy Randové z Ústí nad Labem. Systém se zakládá na několika entitách, jež tvoří celkovou strukturu systému. Základním prvkem jsou soutěže, od kterých se pak odvíjí většina částí systému. Pod konkrétní soutěží se nachází vždy celá její struktura, a to konkrétně kola, kategorie apod. Obsahem kategorií jsou jednotlivá čísla, spojená s jednotlivými skladbami soutěžících, která tvoří dohromady program v dané kategorii. Každá soutěž dále musí obsahovat hodnocení, které se skládá z jednotlivých kritérií a také nastavení umožní odlišit některé funkční možnosti jednotlivých soutěží.

Nedílnou součástí každého systému jsou uživatelské role, jež umožní oddělit práci s jednotlivými zdroji. Systém obsahuje hned několik typů uživatelských rolí. Role, která má nejvyšší práva, se nazývá správce soutěže, pro členy poroty jsou připraveny role předseda poroty a porotce a pro účinkující je to role soutěžící. Pro potřeby kontroly hodnocení je připravena role sčítač bodů.

2.1.1 Fungování systému

Systém, který je vytvářen v rámci diplomové práce pod záštitou firmy JPH Software, bude sloužit především personálu z vedení hudebních soutěží, kterým značně ulehčí práci. Veškerá funkčnost je směřována především pro webovou platformu a nejdůležitější části systému jsou určeny i pro mobilní zařízení. Nedílnou součástí je multijazyčnost. Systém nabízí 3 jazyky. Jsou jimi angličtina, čeština a slovenština.

Důvod proč vyvíjet tento systém i pro mobilní platformu je takový, že všude ve světě je masivní rozmach mobilních aplikací a koncoví uživatelé předpokládají, že jejich oblíbený systém lze spustit na mobilních zařízeních jako plnohodnotný mobilní systém. Dalším důvodem je i větší uživatelská přívětivost přístupu k systému pomocí mobilní aplikace, jestliže se uživatelé rozhodli využívat systém pomocí mobilních zařízení.

Sady funkcí jsou vždy určeny pro konkrétní typ uživatelské role. V systému existuje pouze jediná sada funkcí, která je určena pro všechny uživatele. Je to možnost odhlášení, změna hesla, osobních údajů, jazyka nebo zobrazení vyhodnocení za celou soutěž nebo za konkrétní kategorii v daném kole. Uživatelská role se dělí na správce soutěže, sčítače bodů, předsedu poroty, porotce a soutěžící.

Soutěžícím je umožněn vstup do systémů ze dvou míst. Prvním místem jsou přihlášky do soutěže. Po vyplnění je soutěžící zaregistrován do programu soutěže a získá také přístupové údaje i do části, která vyžaduje přihlášení. Pokud již uživatel někdy obdržel přístupové údaje do systému, stačí je pouze vyplnit spolu s dalšími údaji a odeslat přihlášku, poté už uživateli samozřejmě není vytvářen nový účet. Druhým místem je uživatelská část po přihlášení, kde má soutěžící přehled o struktuře soutěže včetně důležitých datumových informací. Dále bude moct přejít opět na přihlášku.

Předseda poroty a porotce mají téměř totožné funkcionální možnosti. Po přihlášení vidí strukturu všech soutěží, do kterých jsou začleněni. Toto začlenění do soutěže provede správce soutěže. Hlavní funkcí pro tyto 2 druhy uživatelů je vyplnění bodovacích lístků ke každé kategorii v daném kole. Během bodování si uživatel může prohlédnout i skladby, které soutěžící během svého představení prezentuje. Udělené body nesmí přesáhnout danou hranici stanovenou správcem soutěže. Dále nesmí hodnotit soutěžící, kteří pocházejí ze stejné školy jako porotce. Tato funkce je ovšem nastavitelná.

Role sčítače bodů slouží pouze pro kontrolu bodování. Tato role se dá připodobnit k fungování archivátora. Do funkcí pro tuto roli tak patří zobrazení struktury soutěží, export bodovacích lístků nebo export programu se soutěžícími.

Pod roli správce soutěže spadá nejvíce funkčních možností. První možností je vytvoření soutěže, jež zahrnuje tvorbu struktury obsahující např. kola a kategorie, dále hodnocení včetně nastavení maximální hodnoty bodování a v neposlední řadě i přidání personálu (porotci, ostatní personál). Do vytváření soutěží spadá i možnost speciálního nastavení, které upravuje způsob hodnocení, kdy se mohou odečítat nejhorší a nejlepší body pro odstranění subjektivních hodnocení, kontrolu bodování všech porotců nebo zda porotce může hodnotit žáky ze stejné školy.

Do pravomocí správce soutěže patří i editace a případně odstranění soutěží, kol, kategorií, odhlašování soutěžících, uzamykání kol před hodnocením, uzavírání vyhodnocení nebo export bodovacích lístků, soutěžících nebo hodnocení.

2.1.2 Požadavky na systém

Požadavky pomáhají definovat specifikaci toho, co by mělo být implementováno. V podstatě rozlišujeme dva typy požadavků:

- Funkční požadavky, jež symbolizují nabízené chování systému,
- nefunkční požadavky, specifikující technické pozadí aplikace.

Požadavky tvoří základní stavební kameny každého návrhu aplikace. Dalo by se říci, že je to vyjádření toho, co by měl daný systém dělat. Jsou jediným vyjádřením, co by měl systém dělat, ale rozhodně ne toho, jak by to měl dělat. Lze určit, co by měl systém dělat a jaké chování by měl poskytovat, aniž bychom cokoli říkali o způsobu, jak bude dané funkce dosaženo [1].

Systém řeší evidenci soutěží zahrnující kola, kategorie, druhy hodnocení, soutěžní program, bodování porotců, export bodovacích lístků, export programu soutěže, export celkového bodování. Dále se stará o vyhodnocení a kontrolu bodování. Evidovány jsou i základní umělecké školy a státy, ale manipulace s nimi není možná.

2.1.2.1 Funkční požadavky

- **Oprávnění a evidence uživatelů**

  V systému budou evidování jednotliví uživatelé, kteří se rozdělí do několika uživatelských rolí a kteří budou s tímto systémem pracovat. Každý uživatel bude mít právě jednu roli, která je určena k práci s konkrétními zdroji. Každý uživatel bude mít své uživatelské jméno (ve tvaru emailu), heslo, roli, jméno a příjmení. Autentifikace a autorizace bude prováděna systémem během přihlášení uživatele. Každému uživateli bude umožněna změna hesla a jazyka, dále změna osobních údajů a odhlášení ze systému. Přihlášky do soutěží pak budou dostupné pro všechny uživatele, tedy ještě před přihlášením.

  Role s nejvyšším oprávněním se nazývá správce soutěže. Tento uživatel bude pracovat s kompletní správou soutěže. Bude moct vytvářet novou soutěž včetně nadefinování struktury a hodnocení, přidání personálu a stanovení výchozího nastavení soutěže. Všechny své vytvořené soutěže bude moci i editovat. Dále bude tomuto uživateli umožněno odhlašovat a odstraňovat soutěžící, vytvářet nové uživatele a provádět export bodovacích lístků, seznamu soutěžících a hodnocení a v neposlední řadě také uzamykání kategorie před bodováním nebo uzavírání hodnocení.

  Roli soutěžící bude umožněno využívat všeobecné základní funkcionality jako je např. zobrazení struktury soutěží, vyhodnocení za

kategorii nebo i za soutěž, odesílání přihlášek apod. U soutěžícího bude navíc evidován rok narození, rodné číslo, stát a škola.

Sčítač bodů bude mít také, kromě společné základní funkcionality, i přístup k bodovacím lístkům a bude tak moct kontrolovat a tisknout tyto bodovací lístky ale i např. seznamy soutěžících a také hodnocení v kategorii.

Předseda poroty a porotce smějí nad rámec základních funkcionalit přistupovat k bodování za danou kategorii. Tuto možnost mohou využívat jako jediní ze všech uživatelských rolí. U těchto rolí bude evidováno, z jaké školy pocházejí.

- **Evidence struktury soutěží**

  Pouze správce soutěže bude mít možnost evidovat soutěže. Evidovat se bude název soutěže, název úvodního kola a názvy jedné či více kategorií. Veškeré názvy budou vždy evidovány ve třech jazycích.

- **Evidence hodnotících kritérií**

  Tato evidence spadá pouze pod oprávnění správce soutěže. U hodnotících kritérií se tedy bude evidovat název ve třech jazycích a horní bodovací mez.

- **Přidělení personálu**

  Možnost přidělovat personál k soutěži, tedy porotce nebo ostatní personál, patří do funkcionality spadající uživateli s oprávněním správce soutěže.

- **Evidence nastavení soutěže**

  Každá soutěž může obsahovat specifické nastavení. Ovšem výchozí nastavení je již definováno systémem. Jedná se o kontrolu bodování všech porotců, odečítání nejlepších a nejhorších bodů a zda může porotce hodnotit žáky ze stejné školy. Veškeré změny nastavení provádí pouze správce soutěže. Evidován bude klíč označující každý druh nastavení a data, která popisují konkrétní nastavení.

- **Evidence přihlášek**

  Přihlášku do soutěže může odeslat kdokoliv. Pod pojmem přihláška si lze představit soutěžícího se svým souborem skladeb. Soubor takových přihlášek poté tvoří program v daném kole a kategorii. U přihlášky bude evidována soutěž a kategorie, do které se daný soutěžící hlásí a v neposlední řadě také seznam skladeb včetně stopáže.

- **Evidence skladeb**

  Pracovat s evidencí skladeb může správce soutěže. Soutěžící má přístup pouze ke svým skladbám. U každé skladby je evidován její autor, název a také délka v minutách.

- **Evidence bodů**

  Bodování provádí pouze porotce či předseda poroty. Zaznamenávány jsou informace, jaký porotce, k jakému číslu a k jakému druhu hodnocení udělil body. Nepochybně je třeba evidovat také hodnotu bodování.

  Možnost procházet evidenci bodů má každá uživatelská role.

- **Uzamykání kategorie**

  Uzamykání kategorie slouží jako ochrana před nežádoucím bodováním porotců např. ve chvíli, když se již mají sčítat body. Tuto možnost ovládá pouze správce soutěže.

- **Uzavírání kategorie**

  Tuto funkci využívá pouze správce soutěže vždy pro vyhodnocení soutěžících a jejich následný postup do dalšího kola. Po uzavření kategorie není již možnost bodovat a ani nijak jinak upravit danou kategorii.

- **Export bodovacích lístků**

  Tato možnost je určena především pro sčítače bodů, ale využívat ji může i správce soutěže. Je určena pro uložení v interních dokumentech organizace a pro případnou kontrolu. Export je prováděn za každého porotce v dané kategorii do formátu PDF[1], XLSX[2]. Takto vyexportovaný lístek by měl obsahovat náležitosti soutěže včetně prostoru pro podpis.

- **Export soutěžících**

  Této možnosti smí využít pouze sčítač bodů a správce soutěže. Exportovat lze do formátu PDF a XLSX. Vygenerovaný dokument by měl obsahovat kromě seznamu soutěžících také prostor pro podpis předsedy poroty a sekretářky.

[1] Portable Document Format. Je to přenosný formát dokumentů
[2] Formát souboru pro ukládání dat z tabulkového editoru

- **Export hodnocení**

  Export je možný opět do formátu PDF a XLSX. Tato možnost je umožněna jen pro sčítače bodů a správce soutěže. Ve vygenerovaném dokumentu by se pak měl nacházet součet a průměr bodů.

- **Odhlášení soutěžícího ze soutěže**

  Odhlašování provádí buď sám soutěžící nebo správce soutěže. Tato funkce je zavedena z důvodu případné neúčasti buď před nebo během soutěže.

- **Kontrola bodování všech porotců**

  Kontrola bodování se provádí vždy při uzavírání kategorií pro postup do dalšího kola. Toto ošetření je třeba, aby správce neuzavřel kategorii ještě předtím, než by měli hodnotit porotci. Eliminuje tak chybné vyhodnocení. Možnost kontroly je ovšem volitelná a vypnout ji může pouze správce soutěže.

- **Odečítání nejnižších a nejvyšších bodů**

  Využití této funkce je opět volitelné a je možné ji zcela vypnout, což učiní pouze správce soutěže. Někdy se bohužel stává, že porotce upřednostňuje nebo naopak poškozuje jinak objektivní hodnocení hodnocením subjektivním, proto na řadu přichází tato funkce, která má za úkol odstranit extrémy z bodovacích listin.

- **Ověření shody škol porotce a soutěžícího**

  V některých případech se stává, že v porotě sedí porotce, jež pochází ze stejné základní umělecké školy jako soutěžící. Pro účely zabránění

zaujatosti přichází do popředí funkce, jež porovná školy zúčastněných stran a v případě shody znemožní tak bodování některých porotců u konkrétních soutěžících.

2.1.2.2 Non-funkční požadavky

- Systém musí být přístupný z webového prohlížeče
- Databáze bude implementována pomocí PostgreSQL
- Implementace systému bude na webové platformě pomocí PHP[3], a to za použití Nette Framework[4]
- Heslo bude šifrováno alespoň 160 bitovým klíčem
- Webová aplikace bude responzivní
- Vizuální podoba webové aplikace bude implementována za pomoci Bootstrap[5] 4
- Verzování bude prováděno přes GIT[6].
- Správa závislosti v PHP bude prováděna pomocí nástroje Composer[7]
- Aplikace bude dostupná i pro operační systém Android 4.4[8] a vyšší

2.2 Diagram případů užití

Diagram případů užití (Use Case Diagram) vyjadřuje chování systému z pohledu uživatele. Cílem diagramu je popsat sadu funkcí, které systém nabízí, a to takovým způsobem, aby dostatečně vyjádřil, co má systém dělat, ale ne jak to má dělat. Jedná se o jeden z prvních diagramů, které jsou na

[3] Hypertext Preprocessor. Jde o skriptovací programovací jazyk
[4] Framework je softwarová struktura, jež ulehčuje práci při tvorbě aplikací. Je tvořen knihovnami nebo doporučenými postupy při programování
[5] Framework pro tvorbu designu webových aplikací
[6] Systém správy verzí vytvořen Linusem Torvaldsem
[7] Nástroj na správu závislostí v PHP
[8] Mobilní operační systém

začátku tvorby informačních systémů a patří mezi základní nástroje UML[9]. V prvé řadě je třeba vymezit jasné hranice systému a dostatečně využít abstrakce pro stanovení, co má systém umět a až poté přichází na řadu úvaha, jak daný úkol řešit [2].

Při sestavování diagramu je třeba postupovat po jednotlivých logických krocích. Nejprve se tedy vymezí, co daný systém má umět, poté se zvolí aktéři a naleznou se případy užití, u kterých je třeba stanovit specifikace a scénář.

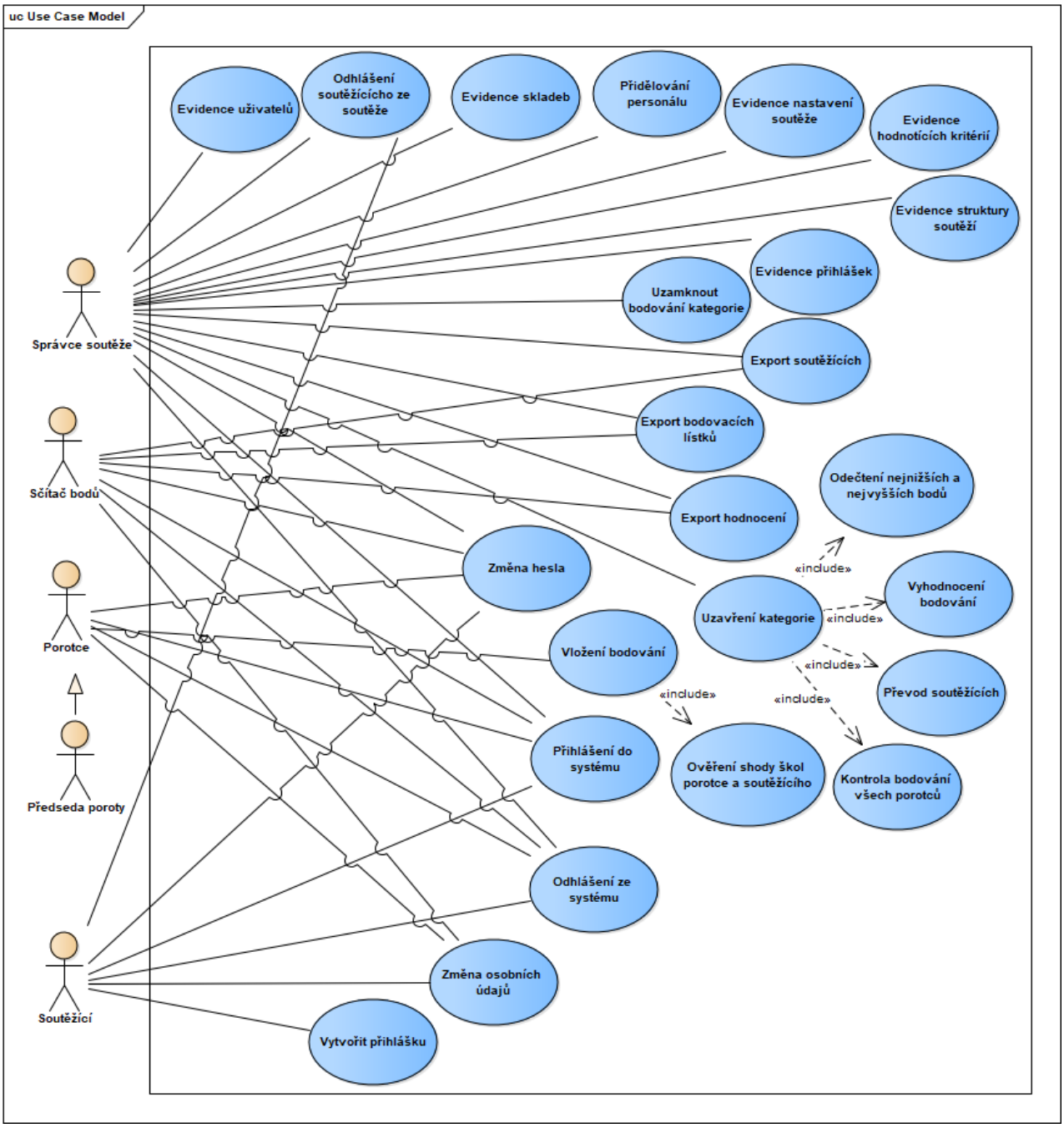
Konkrétní případ užití představuje jednu funkcionalitu. Může např. obsahovat přidání soutěže nebo provést různé kontroly. Ovšem tyto další akce už v diagramu zachyceny nebudou.

Při modelování je lepší postupovat tak, že nebude využívat funkční dekompozice, ale využije se tzv. blackboxu (černé skříňky), přičemž se skryje vnitřní logika, čímž se docílí pouze práce s komponentami. Tento přístup vede k zamezení výskytu tzv. CRUD[10] operací [2].

Diagram případů užití obsahuje mimo jiné relace *<<include>>* a *<<extend>>*. Relace *<<include>>* dle [1] vyčleňuje funkcionalitu, společnou pro několik případů užití, do samostatného případu užití, který je pak zahrnut do příslušných případů užití. U relace *<<extend>>* podle [1] platí, že do existujícího případu užití vkládá nové chování.

Diagram případů užití pro navrhovaný systém je znázorněn na obrázku 1.

[9] Jedná se o jazyk pro specifikaci dokumentace programových systémů
[10] Create, Read, Update, Delete. Jedná se o operace se záznamem v trvalém uložišti



Obrázek 1 – Diagram případů užití pro navrhovaný systém (zdroj: autor)

2.2.1 Detailní popis vybraných případů užití

Případ užití: Změna osobních údajů
ID: UC01
Stručný popis: Uživatel si mění své uživatelské údaje
Hlavní aktéři: Správce soutěže, sčítač bodů, porotce, předseda poroty, soutěžící
Vedlejší aktéři: Žádní
Vstupní podmínky: Uživatel je přihlášen do systému
Hlavní scénář: 1. Aktér iniciuje změnu osobních údajů 2. Systém zobrazí formulář 3. Aktér změní údaje
Výstupní podmínky: Osobní údaje byly změněny
Alternativní scénář: Žádný

Případ užití: Přihlášení do systému
ID: UC02
Stručný popis: Uživatel se přihlásí do systému
Hlavní aktéři: Správce soutěže, sčítač bodů, porotce, předseda poroty, soutěžící
Vedlejší aktéři: Žádní
Vstupní podmínky: Uživatel není přihlášen do systému
Hlavní scénář: 1. Aktér iniciuje přihlášení do systému 2. Systém zobrazí formulář 3. Aktér vyplní údaje 4. Systém ověří přihlašovací údaje 5. JESTLIŽE jsou přihlašovací údaje správné 5.1. Systém aktéra přihlásí 6. JINAK: 6.1. Systém zobrazí varovnou informaci o neúspěšném přihlášení
Výstupní podmínky: Uživatel je přihlášen do systému
Alternativní scénář: Žádný

Případ užití: Ověření shody škol porotce a soutěžícího
ID: UC03
Stručný popis: Systém ověří, zda není porotce ze stejné školy jako soutěžící
Hlavní aktéři: Porotce, předseda poroty
Vedlejší aktéři: Žádní
Vstupní podmínky: Uživatel je přihlášen do systému
Hlavní scénář: 1. Případ užití začíná až uživatel začne bodovat 2. Systém ověří shodu škol porotce a soutěžícího 3. Aktér vyplní údaje 4. JESTLIŽE není nalezena shoda 4.1. Systém povolí bodování 5. JINAK: 5.1. Systém zobrazí varovnou informaci o nepovoleném bodování
Výstupní podmínky: Porotce a předseda poroty mohou bodovat
Alternativní scénář: Žádný

Případ užití: Vložení bodování
ID: UC04
Stručný popis: Porotce a předseda poroty vkládají body soutěžícímu
Hlavní aktéři: Porotce, předseda poroty
Vedlejší aktéři: Žádní
Vstupní podmínky: Uživatel je přihlášen do systému
Hlavní scénář: 1. Zahrnout (Ověření shody škol porotce a soutěžícího) 2. Systém zobrazí bodovací formulář 3. Aktér vyplní formulář 4. Systém uloží bodování
Výstupní podmínky: Porotce a předseda poroty mohou bodovat
Alternativní scénář: Žádný

2.3 Diagram tříd

Dle [3] je diagram tříd diagramem implementace. Vývojáře vede k přemýšlení v kontextu celého systému. Podle [4] poskytuje statický pohled na třídy systému, jejich atributy, operace a vzájemné vztahy.

Diagram tříd lze rozdělit do základních úrovní, kterými jsou analytický a návrhový model tříd. UML určuje, jak má vypadat vizuální syntaxe diagramu tříd, ale postup tvorby včetně rozdělení na analytickou a návrhovou část určuje metodika.

2.3.1 Analytický model

Jak je popsáno v [4], analytický model tříd modeluje obchodní doménu systému, tedy typy objektů a vztahy mezi nimi. Snaží se zachovat přehlednost a jednoduchost bez jakýchkoliv implementačních detailů, tedy např. bez operací a případně i bez atributů. Není tedy závislý na konkrétním programovacím jazyku.

Pro nalezení analytických tříd se používá několik možných postupů. Odborná literatura [1] rozděluje tyto postupy na:

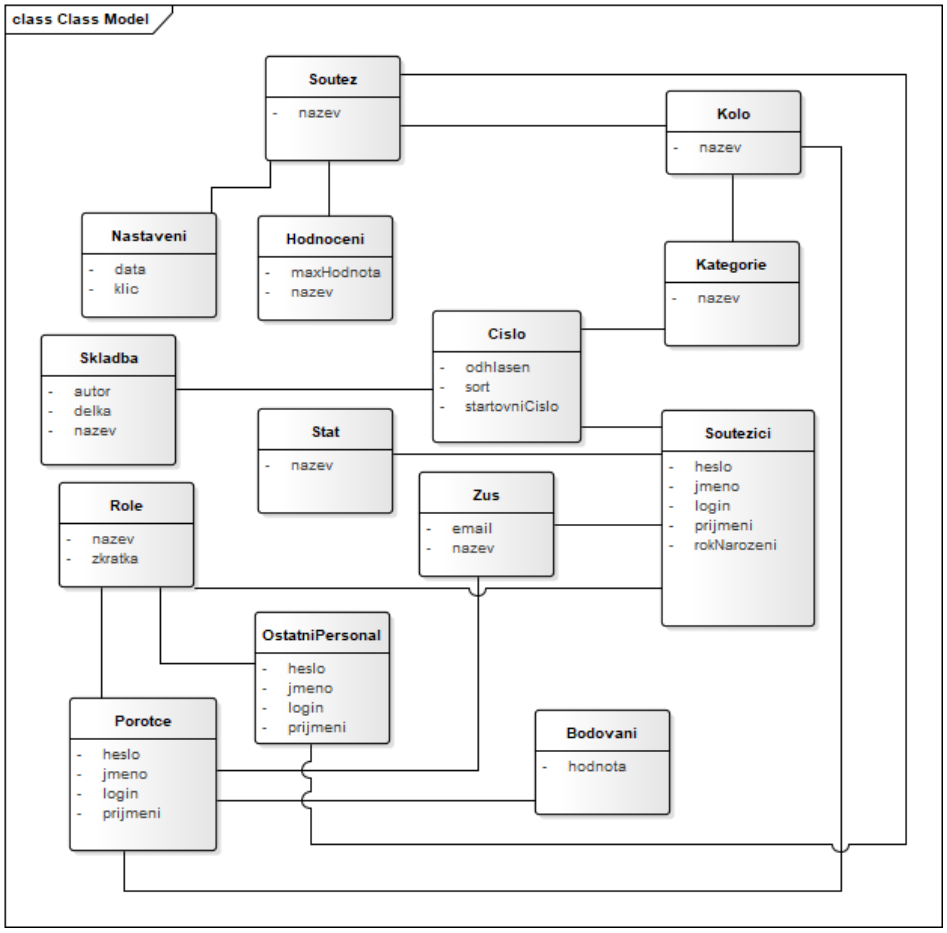
- **Analýzu podstatných jmen a sloves** dostupných dokumentů (specifikace nebo dokumentace případů užití)
- **Metodu štítků CRC**[11], která spočívá v okamžitých nápadech, kdy jsou důležité aspekty zaznamenávány na samolepících štítcích
- **Metodiku RUP**[12] využívající stereotypů *<<boundary>>*, *<<control>>*, *<<entity>>*
- **Analýzu dalších zdrojů** (firemní dokumenty, externí systémy, fyzické objekty)

[11] Class, Responsibilities and Collaborators. Třída, odpovědnosti a spolupracovníci
[12] Rational Unified Process. Objěktově orientovaný iterativní přístup k životnímu cyklu softwaru

Při postupu tvorby analytického modelu tříd se lze řídit body dle [4] takto:

1. Nalezení tříd, základních atributů, operací a spolupracovníků
2. Určení dědičnosti mezi třídami
3. Zachycení vztahů pomocí asociací
4. Pojmenování asociací nebo rolí na nich
5. Určení násobností relací
6. Zachycení závislostí
7. Doplnění dalších atributů a operací, které mají analytický charakter

Pro provedení tohoto postupu poté vznikne model pro navrhovaný systém, který je znázorněn na obrázku 2.



Obrázek 2 – Analytický model tříd navrhovaného systému (zdroj: autor)

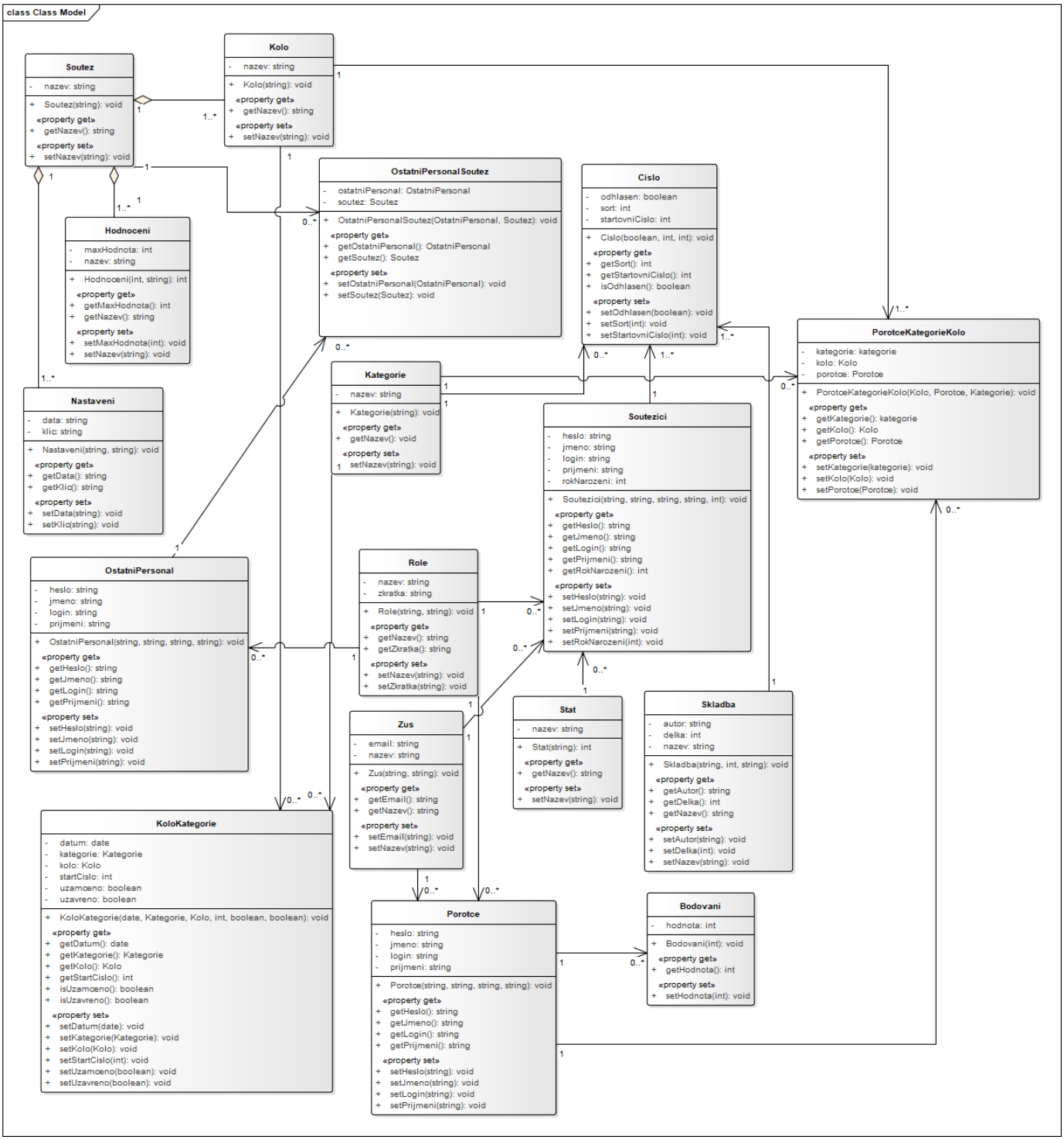
2.3.2 Návrhový model

Je rozšířením analytického modelu o implementační třídy a detaily. Velmi často obsahuje až několikanásobně více tříd. Na rozdíl od analytických tříd jsou návrhové třídy na takové úrovni abstrakce, že je lze implementovat. Jak se uvádí v [6], návrhový model tříd je závislý na zvolené technologii a přiřazuje zodpovědnosti. Dle [1] by měla správná návrhová třída obsahovat kompletní sadu atributů a jejich detailní specifikace včetně názvů, typu, viditelnosti a nepovinně i implicitní hodnoty.

Způsobů převodu z analytického do návrhového modelu je zajisté více. Jeden z možných postupů je uveden v [5] a lze ho tedy nastínit takto:

1. Nalezení návrhových tříd
2. Doplnění detailů návrhových tříd (atributy, metody)
3. Doplnění rozhraní, šablon, vnořených tříd, stereotypů
4. Doplnění chybějících relací
5. Upřesnění analytických relací
6. Aplikace návrhových vzorů

Výsledek tohoto postupu je reprezentován na obrázku 3.



Obrázek 3 – Návrhový model tříd navrhovaného systému (zdroj: autor)

2.4 Datové modely

2.4.1 E-R model

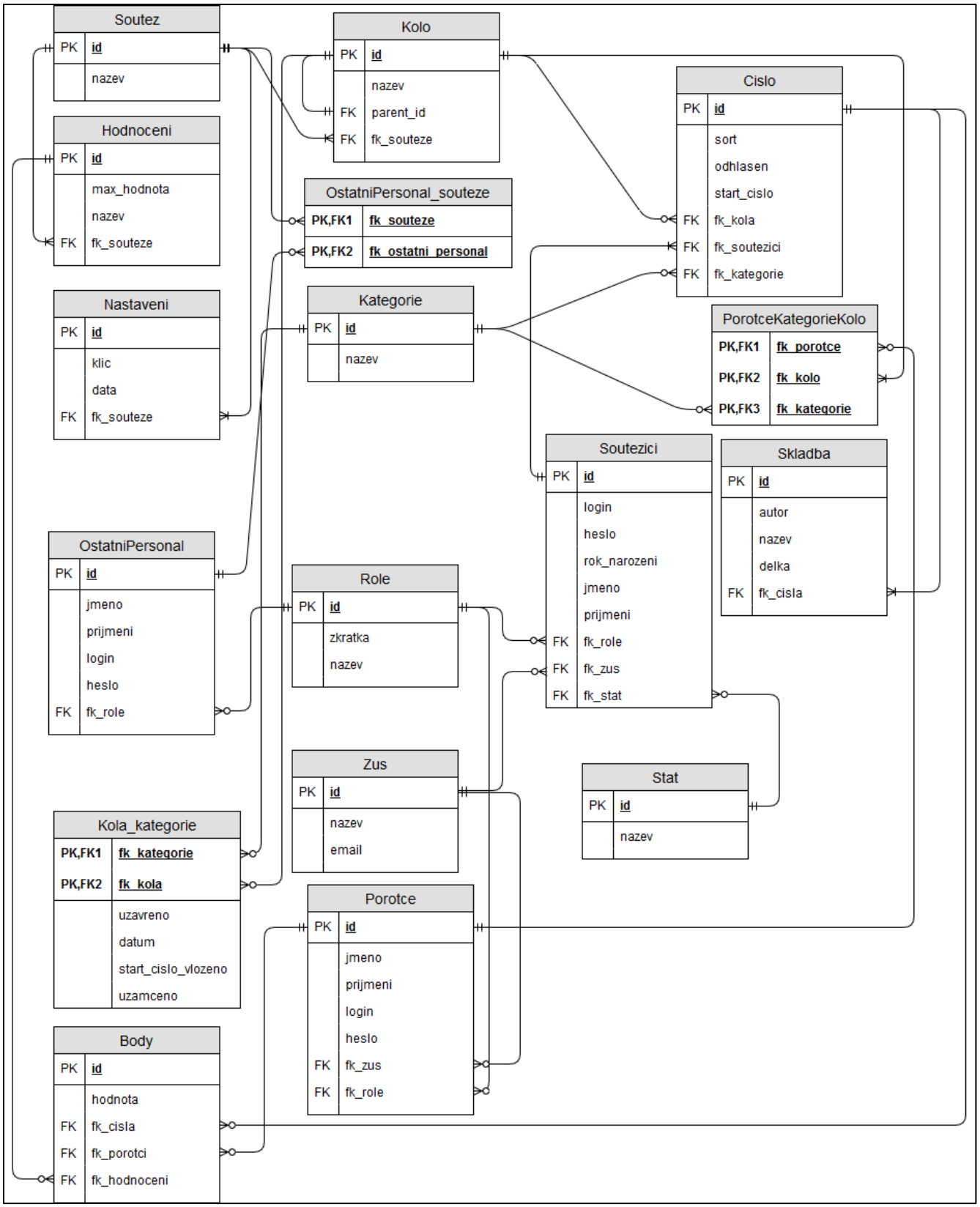
V systémech, které se běžně používají, objem dat roste stále větším tempem, a proto je nutné v takových systémech striktně dodržovat správnou strukturu dat např. už proto, aby byla práce s takovými daty co nejefektivnější.

Způsobů, jak přistupovat ke správné struktuře, je více. V literatuře [7] se uvádí, že prvním a zásadním krokem je identifikace entit a jejich vzájemných vztahů. Výsledkem tohoto snažení je pak entitně-vztahový diagram, který je často ne zcela šťastně označován jako entitně-relační. Jeho konkrétní podoba pro navrhovaný systém je znázorněna na obrázku 4. Takový diagram pak lze poměrně snadno převést na relační databázové schéma.

E-R diagram patří do oblasti konceptuálního modelování a to znamená, že je zcela nezávislý, bez ohledu na to, jak budou fyzická data uložena. Není zatížen technologickou koncepcí řešení ani jeho implementačními detaily. Dá se tedy říci, že odpovídá na otázku, *co* je obsahem systému.

Mezi vybrané základní stavební kameny E-R diagramu, dle [8] patří:

- **Entity** – reálná nebo imaginární část světa, která nás zajímá a ke které chceme sbírat data, např. porotce, kategorie, skladba
- **Atributy** – vlastnost entity, která nás zajímá v daném kontextu, např. jméno, příjmení, název kategorie
- **Vztahy mezi entitami** – asociace mezi entitami, např. kategorie s názvem soutěžící do 12 let *obsahuje* číslo s pořadím 1
- **Parcialita vztahu**
- **Kardinalita vztahu** (1:1, 1:N, M:N)



Obrázek 4 – E-R diagram pro navrhovaný systém (zdroj: autor)

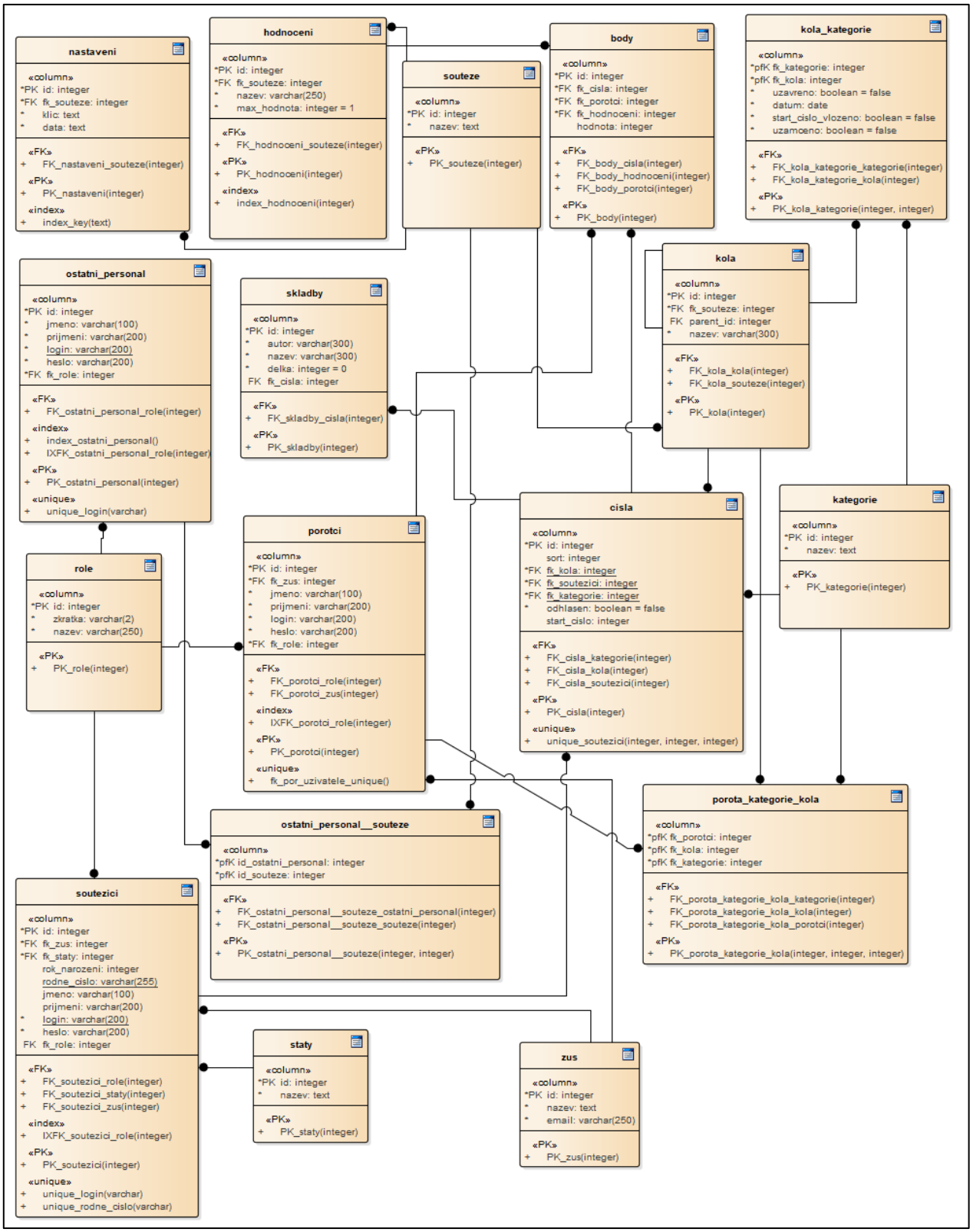
2.4.2 Fyzický model

Fyzický model je dalším krokem v tvorbě správné struktury dat. Reprezentuje vlastní návrh implementace pro konkrétní databázový systém. V případě tohoto navrhovaného systému se jedná o databázi PostgreSQL. Tento model je závislý na vybrané databázi, protože přiřazuje datové typy, délku či přesnost jednotlivým sloupcům. Z datových modelů je nejpodrobnější, je ale platný pouze pro určitou databázi. Nabízí mnoho informací, ale může se stát u větších modelů, nepřehledným.

Fyzický model vzniká převodem z konceptuálního modelu (ER diagramu). Bude tedy implementován relační databází, která je na obrázku 5. Převod z konceptuálního modelu na fyzický přináší s sebou i změnu terminologie, která je popsána v tabulce 1 .

Konceptuální	Fyzický
Entita	Tabulka
Instance	Řádek
Atribut	Sloupec
Primární identifikátor	Primární klíč
Sekundární identifikátor	Unikátní klíč
Vztah	Cizí klíč

Tabulka 1 – Změny terminologie při převodu konceptuálního modelu do fyzického



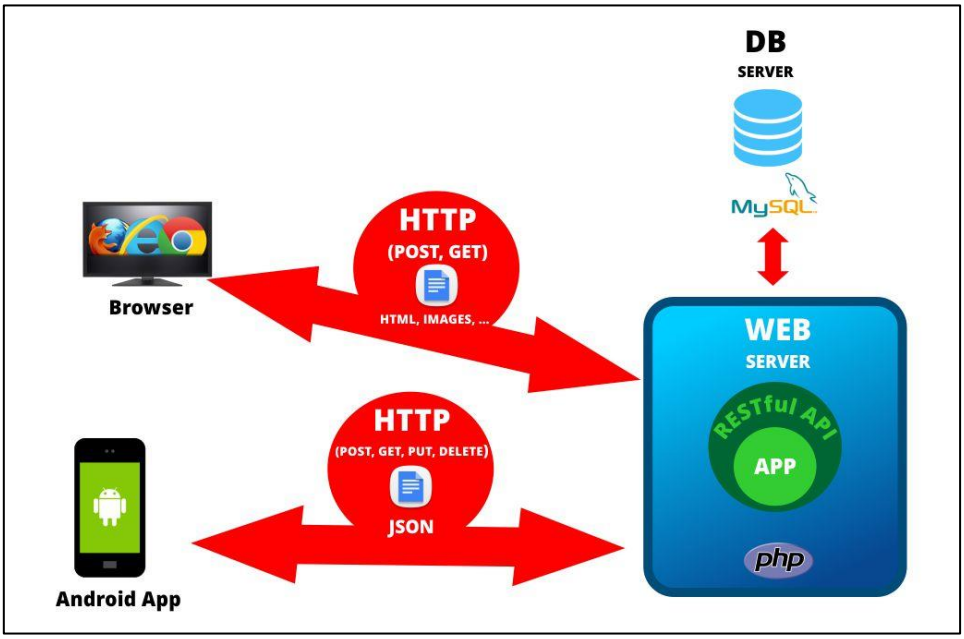
Obrázek 5 – Relační databáze navrhovaného systému (zdroj: autor)

3. Implementace systému

Implementace systému je proces, kdy je uskutečňována teoretická myšlenka, jež vznikla ve fázi analýzy. Podle [1] implementace spočívá v převodu modelu v návrhové fázi do podoby spustitelného kódu. Smyslem implementace je, dle analytika či návrháře, tvorba požadovaného implementačního modelu.

V architektuře systému na straně jedné hraje důležitou roli webový server, jež zpracovává PHP skripty a komunikuje s databází. Na straně druhé se poté vyskytuje klientská část, která se skládá z uživatelského prohlížeče nebo z mobilní aplikace s operačním systémem Android. Tyto strany spolu komunikují pomocí protokolu HTTP[13] za použití protokolových metod.

Tato kapitola popisuje, jakým způsobem je komunikace realizována a jak jsou hlavní části systému implementovány. Na obrázku 6 je pak znázorněna grafická podoba struktury systému.



Obrázek 6 – Struktura systému (zdroj: autor)

[13] Hypertext Transfer Protocol. Internetový protokol pro výměnu hypertextových dokumentů

Systém je rozdělen, z hlediska zpracování, na dvě části. První částí je webová aplikace, která je přístupná z webového prohlížeče a umožňuje využívat veškerou nabízenou funkcionalitu systému. Tato podoba aplikace má zásadní výhodu v tom, že je přístupná odkudkoliv a z jakéhokoliv zařízení, které má v sobě webový prohlížeč. Tato webová aplikace je vytvořena na platformě PHP s využitím Nette Frameworku, ve vývojovém prostředí NetBeans 8.2., vlastněné a sponzorované firmou Oracle Corporation.

Implementace webové části systému nese s sebou povinnost vytvořit RESTful[14] API[15], které by nabízelo možnost vzdálené autentifikace uživatele a v případě úspěšného ověření uživatele také možnosti vytvořit, číst, editovat nebo smazat data. V případě, že se v budoucnu bude webová aplikace, tedy včetně API, přesouvat na jiný webhosting, je potřeba v mobilní aplikaci přepsat adresy, které na toto zmiňované aplikační rozhraní vedou.

Druhou podobou je aplikace cílená pro mobilní operační systém Android. Jedná se o aplikaci, která využívá jako zdroj dat již zmiňované RESTful API. Umožňuje modifikaci a následné odeslání změněných dat na server opět pomocí RESTful API. Jedním z hlavních důvodu implementace mobilní aplikace přímo pro Android je fakt, že nárůst nákupu chytrých telefonů s mobilním operačním systémem Android stále roste. Dle analytické agentury Gartner [9] meziroční nárůst na určitém vzorku lidí mezi prvním kvartálem roku 2017 a prvním kvartálem roku 2018 byl 1,3 %, dále pak v prvním kvartálu roku 2018 byl podíl mobilního operačního systému Android 85,9 % na vzorku čítající 383 503 900 koncových uživatelů. Vývoj aplikace pro Android probíhá ve specializovaném a velmi uživatelsky přívětivém Android Studiu, postaveném na společných základech produktu IntelliJ IDEA a provozované společností Google. V minulosti se k tvorbě mobilních aplikací pro Android využívalo prostředí Eclipse s pluginem

[14] Representational state transfer. Způsob pro vytvářením editování, čtení a mazání informací pomocí HTTP protokolu
[15] Application Programming Interface. Rozhraní pro programování aplikací

Android Developer Tools. Výhodou Android Studia oproti Eclipse je skutečnost, že přímo obsahuje kompilátor Android, základní emulátory a Android SDK Tools a to bez nutnosti jakéhokoliv doinstalování.

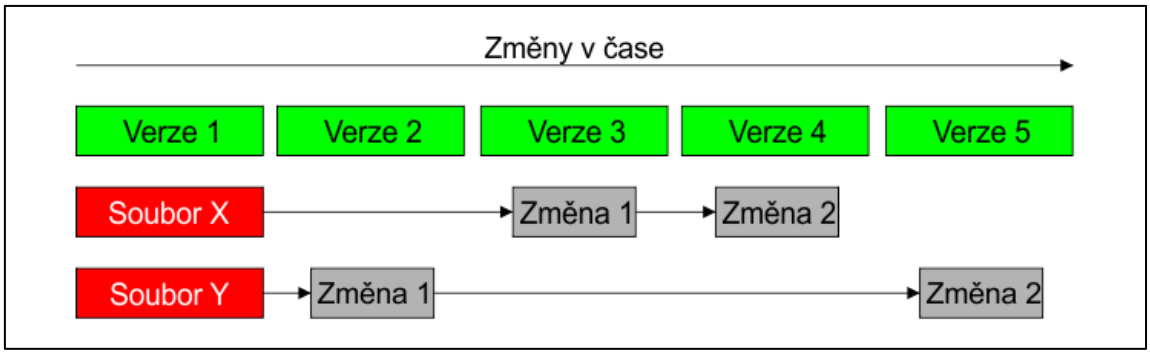
3.1 Git

Vzhledem k rozsáhlosti systému je třeba mít kontrolu nad prováděnými změnami v rámci vývoje. Tato potřeba vedla k využití verzovacího nástroje. Takových nástrojů je více, ale nejběžnějším standardem je právě Git, a proto byt zvolen. Mimo jiné umožňuje práci více vývojářů nebo dokumentaci změn během vývoje. Git je v rámci vývoje použit kromě zálohování dat také např. na vrácení změn do předchozích verzí a hledání problému v rámci servisního ošetření aplikace.

Git je dle článku [10] systém pro verzování souborů, který byl vyvinut Linusem Torvaldsem v roce 2005 a to pro vývoj jádra Linuxu. Tento nástroj dokáže uchovávat jednotlivé verze souborů, zobrazovat rozdíly nebo také slučovat změny více uživatelů apod. Do repozitáře (lokání nebo vzdálené uložiště) se ukládají kromě zdrojových souborů také obrázky či jiné binární soubory, u kterých ale nelze zobrazit rozdíly, avšak verzovat se dají.

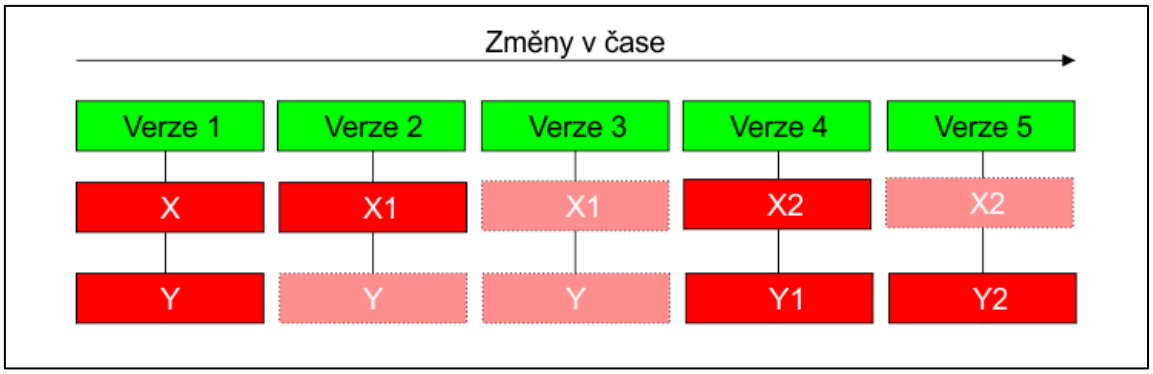
Existují i konkurenční nástroje Gitu. Jsou jimi např. SVN[16], BitKeeper, Darcs, Monotone a další. Dle oficiálního tvrzení Gitu [11] rozdíl spočívá ve zpracování dat. Konkurenční systémy ukládají data jako seznamy změn jednotlivých souborů, jelikož chápou uložené informace jako sadu souborů a seznamů změn těchto souborů v čase, což znázorňuje obrázek 7.

[16] Apache Subversion. Systém pro správu a verzování zdrojových kódů



Obrázek 7 – Ukládání dat jinými systémy jako změny v každém souboru (zdroj: autor)

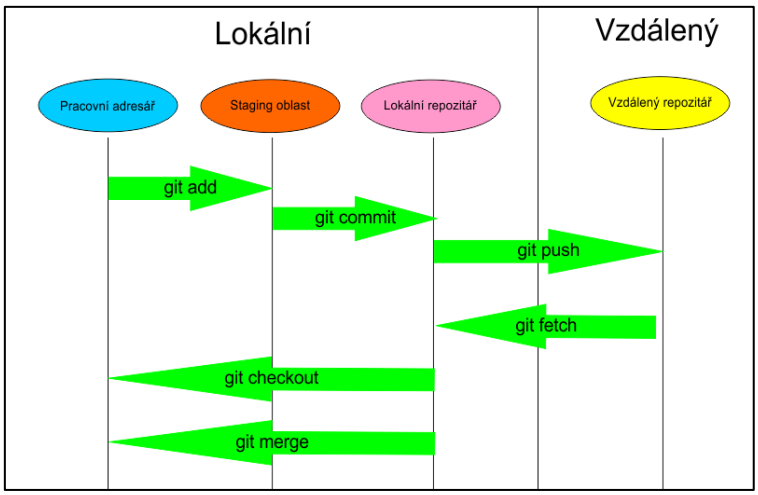
Git zpracovává data odlišně. Blíže je tento princip znázorněn na obrázku 8. Chápe uložené informace spíše jako sadu snímků vlastního malého systému souborů. Při každé změně zaznamená, jak vypadají všechny soubory v dané chvíli, a to v podobě reference na obraz celého systému souborů.



Obrázek 8 – Ukládání obrazu projektu v čase Gitem (zdroj: autor)

Git pracuje se základními čtyřmi pracovními prostory. Prvním je pracovní adresář, což je složka, ve které se nachází soubory, se kterými se pracuje. Není však povinnost mít všechny soubory ve složce a zároveň i v repozitáři. Pro určení, s kterými soubory se má pracovat se využívá staging oblast, která by se dala připodobnit k virtuální složce pro Git. Vše, co se pak nachází v této oblasti, bude pomocí commitu přeneseno do lokálního repozitáře. V lokálním repozitáři se vyskytuje celá historie projektu, tedy všechny předchozí verze systému. Posledním důležitým prvkem je vzdálený

repozitář. Pro potřeby navrhovaného systému byl použit GitLab. Jedná se o repozitář, kam se nahrává lokální repozitář. Celý princip je pak znázorněn na obrázku 9.

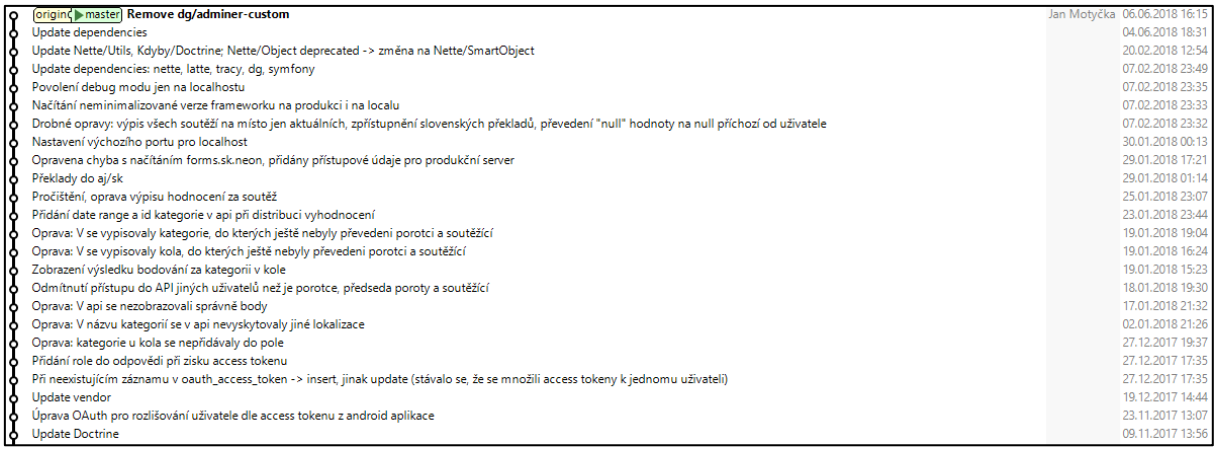


Obrázek 9 – Princip spolupráce lokálního a vzdáleného repozitáře (zdroj: autor)

Existuje cesta, která vede i opačně, tedy stažení souborů ze vzdáleného repozitáře a následný import do lokálního repozitáře. Poté je možnost obnovit lokální repozitář do aktuálního stavu vzdáleného repozitáře nebo také do stavu při jednom z commitů.

Git lze ovládat pomocí příkazu z příkazové řádky, ale také pomocí GUI[17]. Pro potřeby tohoto navrhovaného systému bylo využito grafického prostředí SmartGit. Na obrázku 10 lze vidět, jakým způsobem vypadá strom, do kterého jsou řazeny commity týkající se webové aplikace, které jsou odesílané do vzdáleného repozitáře.

[17] Graphical User Interface. Grafické uživatelské rozhraní.



Obrázek 10 – Strom commitů v grafickém prostředí SmartGit (zdroj: autor)

3.2 Webová aplikace

Jak již bylo naznačeno, v rámci implementace bude vytvářena mimo jiné také webová aplikace, zahrnující serverovou a klientskou část, která je začleněná do celkového konceptu vyvíjeného systému. Mezi důvody tvorby webové aplikace zajisté patří fakt, že je přístup možný téměř odkudkoliv a bez ohledu na použité zařízení pouze s nutností nainstalovaného webového prohlížeče. Během vývoje je dbáno na dodržování non-funkčních požadavků, které byly na začátku stanoveny.

Mezi vývojovými nástroji, které byly využity pro implementaci webové aplikace, patří Nette Framework. Dále je to značkovací jazyk HTML 5[18] a kaskádové styly CSS 3[19]. Pro vyšší dynamičnost a další potřeby tvorby front-endu byla využita javascriptová knihovna jQuery, která má širokou podporu prohlížečů a klade důraz na interakci mezi Javascriptem a HTML. Pomocí ní bude možné ovládat nejrůznější události nebo také efekty či animace.

Užitečným doplňkem tvorby front-endu je knihovna Bootstrap 4. Tato knihovna má výhodu v tom, že nabízí uživatelsky přívětivé a responzivní komponenty, jež jsou založené na HTML, CSS a ovládané pomocí javascriptu.

[18] HyperText Markup Language. Značkovací jazyk pro tvorbu webových stránek.
[19] Cascading Style Sheets. Nástroj pro grafické stylování webových stránek.

Jedná se o hojně využívanou knihovnu, která je dnešním standardem, proto je využita i pro tuto webovou aplikaci.

Součástí webové aplikace bude aplikační rozhraní, které bude sloužit pro komunikaci s okolními systémy včetně zamýšlené mobilní aplikace. Přístup ke zdrojům pomocí tohoto rozhraní bude pak chráněn vzdáleným uživatelským ověřováním pomocí principů OAuth 2.0.

Pro persistentní ukládání dat je zvolena, dle non-funkčních požadavků, databáze PostgreSQL.

3.2.1 Serverová část

3.2.1.1 PHP

Běh webové aplikace je plánován na webovém serveru Apache HTTP Server. Proto je nutné vybrat jazyk, který může na tomto serveru pracovat. S ohledem na non-funkční požadavky je proto zvolen jazyk PHP a bude v něm tedy vytvořena kompletní serverová část webové aplikace.

PHP je dle [12] programovací jazyk, který slouží pro tvorbu tzv. skriptů na straně serveru. Skriptem je myšlen popis chování dané stránky. Dále umí přizpůsobit obsah stránky situaci a případně i reagovat na interakci od návštěvníka. Jednou ze zásadních výhod použití jazyka PHP je fakt, že má strmou křivku učení, což dokazuje i jeho časté využívání na různých projektech. Mezi takové projekty se řadí např. Facebook nebo Wikipedia. Důkazem hojného využívání jazyka PHP jsou i statistická data, uvedená v [13], která tvrdí, že k 18. červnu 2018 využívá PHP 83,5 % webových stránek. Výhoda strmé křivky učení se může v některých případech proměnit i v nevýhodu, jelikož lze v jednoduchém přístupu udělat spoustu chyb.

Jazyk PHP prošel dlouhým vývojem, a to již od roku 1995 kdy Rasmus Lerdorf vydal první verzi jazyka tak, jak je znám dnes. Jazyk je často kritizován za svoji nepříliš oslnivou rychlost zpracování. Dle [14] je to způsobeno tím, že

jsou skripty při každém požadavku načítány a interpretovány. Řešení může spočívat v použití akcelerátoru. Touto cestou se vydal Facebook, který z těchto důvodů vytvořil akcelerátor s názvem HipHop for PHP. Další z vad na kráse PHP býval fakt, že neumožňoval typovou kontrolu pro skalární datové typy nebo zápis návratové hodnoty funkce přímo do její definice.

Zásadní inovace přináší oproti verzi PHP 5 verze PHP 7. Tyto inovace, které budou dále popsány, přispěly k důvodům využití verze PHP 7, tedy přímo PHP 7.1, ve vývoji webové aplikace.

Vybrané inovace, jež přináší PHP 7 dle [15] a které byly využity ve vývoji webové aplikace:

- **Nový engine Zend**

  Od roku 1999 pohání PHP Zend Engine a to od verze PHP 4. Je to open-source engine napsaný v jazyce C, jež je interpretem. Verze PHP 7 je vybavena novým enginem pod označením PHP#NG (Next Generation).

- **Dvojnásobná rychlost**

  Nejlépe rozpoznatelnou předností PHPNG enginu je jeho výrazné navýšení výkonu. Využívání paměti je optimalizované a je přidaná i kompilace právě včas (just-in-time), jež má výhodu v tom, že umožňuje kompilovat až za běhu, tedy ne před zahájením vykonávání. Důkazem zásadního zrychlení jsou testy výkonnosti, které poskytl tým Zend [16]. Kód se vykonává rychleji a zároveň je třeba méně serveru k obsluze stejného množství uživatelů.

- **Deklarace typů**

Jednou z nejvíce medializovaných inovací je deklarace typů, které byly velice žádanou novinkou. Na obrázku 11 je vidět, jak lze v definici metody *getContestant()* uvést návratový typ *Contestant* či hodnotu *null*, která je zde reprezentovaná jako otazník. Dále pak je možné uvést skalární datový typ *int* jako vstupní parametr.

Hodnotu *null* jako návratový typ metody lze uvést až od verze PHP 7.1 a to znamená, že v PHP 7.0 k dispozici není, což vedlo během vývoje k rozhodnutí použít PHP 7.1 místo 7.0.

```
public function getContestant(int $id) : ?Contestant {
    return $this->entityManager->getRepository(Contestant::class)->find($id);
}
```

Obrázek 11 – Ukázka návratové hodnoty u metody v PHP 7.1 (zdroj: autor)

S ohledem na hashování např. hesel apod. byl standardním hashovacím algoritmem do verze 7.1 *bcrypt*, který je využitý pro hashování uživatelských hesel i v rámci tohoto vyvíjeného systému a je stále (psáno v roce 2018) vysoce bezpečným. Jeho bezpečnost lze ještě navýšit zvýšením počtu iterací, čímž ale také dojde ke zpomalení během vytváření hashe.

V době vývoje tohoto systému vyšla nová verze PHP 7.2, která mimo jiné přináší nově ještě více bezpečnější hashovací algoritmus pojmenovaný *Argon2.* Tento algoritmus dokonce vyhrál v roce 2015 soutěž Password Hashing Competition, ve které porazil všech 23 konkurenčních algoritmů. S výhledem do budoucnosti může být přechod od *bcrypt* k *Argon2* dalším krokem, jak v tomto vyvíjeném systému ještě více zvýšit zabezpečení.

PHP 7.1 přináší novinky, kterými v neposlední řadě jsou: podpora 64 bitových systémů Windows, nové operátory „kosmická loď“ a koalescence, anonymní třídy, odstranění zastaralých funkcionalit, rozšíření a další.

V případě, že vývojář využívá jazyk PHP, může využít nástroj pro správu závislostí v PHP, který se na tomto jazyku zakládá. Tento nástroj se jmenuje Composer.

3.2.1.2 Composer

V rámci webové aplikace vzniká potřeba využít komponent, které jsou k dispozici pro serverovou část webové aplikace. Při dodávce takové komponenty do systému vzniká závislost, která se váže např. na aktuální verzi apod. Proto je ve webové aplikaci přistoupeno k nástroji, který v aplikaci spravuje závislosti na dané komponenty včetně nové instalace či aktualizace. Jedním z takových nástrojů je právě Composer, který tuto roli bude vykonávat ve vyvíjené webové aplikaci.

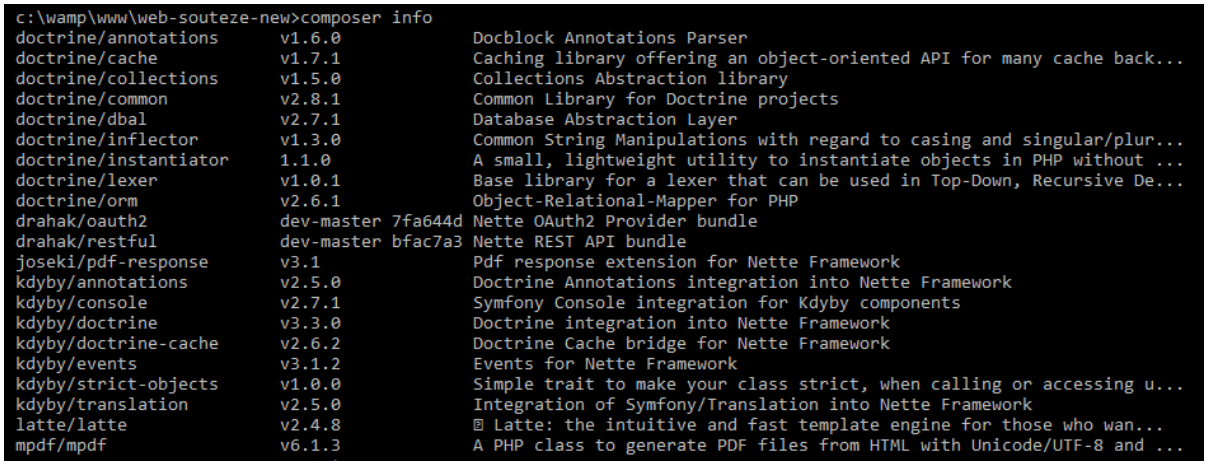
Composer je dle [17] multiplatformní nástroj pro správu závislostí v PHP. Výhodou je, že dokáže deklarovat závislosti a následně je instalovat či aktualizovat. Proto byl využit v rámci vývoje této aplikace a je jeho důležitou součástí.

Oficiálním repozitářem tohoto nástroje je Packagist, odkud Composer právě získává závislosti. Na Packagistu lze publikovat vlastní projekty, které budou dostupné ostatním vývojářům.

V kořenové složce webové aplikace lze najít soubor *composer.json*, kde jsou definovány všechny závislosti. Těmto závislostem lze nastavovat požadované verze a ty se pak stahují do složky *vendor*. Aktuální stav závislostí ze souboru *composer.json* je znázorněn na obrázku 12.

Mezi nejpoužívanější příkazy v rámci této aplikace patří:

- *composer install* – stažení závislostí do složky *vendor*
- *composer require* – přidání závislostí do projektu
- *composer update* – aktualizace závislostí
- *composer info* – informace o aktuálním stavu balíčků



```
c:\wamp\www\web-souteze-new>composer info
doctrine/annotations      v1.6.0            Docblock Annotations Parser
doctrine/cache            v1.7.1            Caching library offering an object-oriented API for many cache back...
doctrine/collections      v1.5.0            Collections Abstraction library
doctrine/common           v2.8.1            Common Library for Doctrine projects
doctrine/dbal             v2.7.1            Database Abstraction Layer
doctrine/inflector        v1.3.0            Common String Manipulations with regard to casing and singular/plur...
doctrine/instantiator     1.1.0             A small, lightweight utility to instantiate objects in PHP without ...
doctrine/lexer            v1.0.1            Base library for a lexer that can be used in Top-Down, Recursive De...
doctrine/orm              v2.6.1            Object-Relational-Mapper for PHP
drahak/oauth2             dev-master 7fa644d Nette OAuth2 Provider bundle
drahak/restful            dev-master bfac7a3 Nette REST API bundle
joseki/pdf-response       v3.1              Pdf response extension for Nette Framework
kdyby/annotations         v2.5.0            Doctrine Annotations integration into Nette Framework
kdyby/console             v2.7.1            Symfony Console integration for Kdyby components
kdyby/doctrine            v3.3.0            Doctrine integration into Nette Framework
kdyby/doctrine-cache      v2.6.2            Doctrine Cache bridge for Nette Framework
kdyby/events              v3.1.2            Events for Nette Framework
kdyby/strict-objects      v1.0.0            Simple trait to make your class strict, when calling or accessing u...
kdyby/translation         v2.5.0            Integration of Symfony/Translation into Nette Framework
latte/latte               v2.4.8            Latte: the intuitive and fast template engine for those who wan...
mpdf/mpdf                 v6.1.3            A PHP class to generate PDF files from HTML with Unicode/UTF-8 and ...
```

Obrázek 12 - Ukázka seznamu závislostí pomocí Composeru (zdroj: autor)

Kromě správy závislostí umí Composer také zakládat nové projekty, tzn., že pro vývojové účely bude využit pro stažení balíčku Nette Frameworku a to příkazem *composer create-project nette/web-project*.

3.2.1.3 Nette Framework

S ohledem na non-funkční požadavky bylo stanoveno využití Nette Frameworku pro vývoj serverové části webové aplikace. Tato volba také vyhovuje požadavku, že aplikace má fungovat na jazyku PHP. Nette Framework je ve webové aplikaci využíván např. pro vytvoření a obsluhu formulářů, zpracování a poskytování dat, autentifikaci a mnoho dalšího.

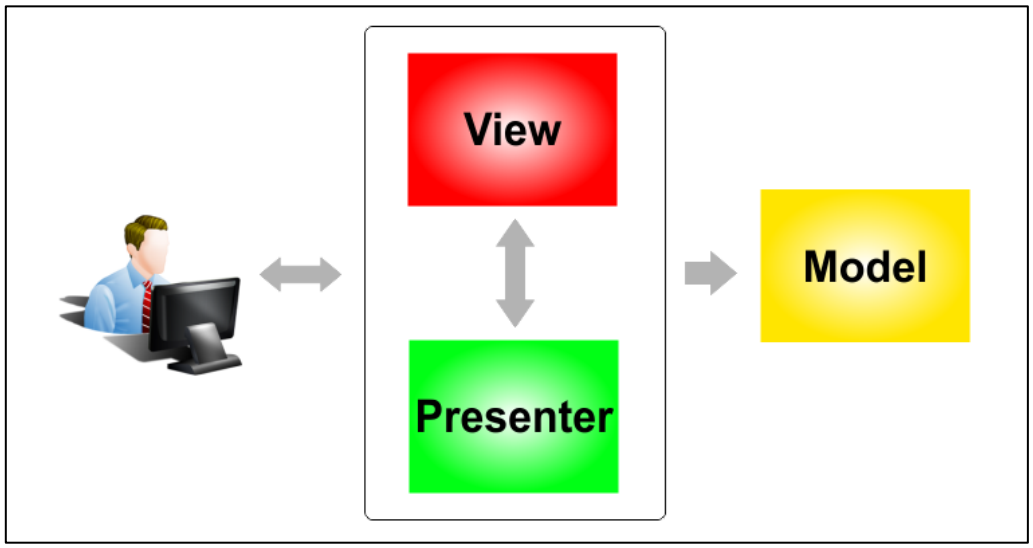
Nette Framework lze zařadit do skupiny open source frameworku, jež jsou založeny na programovacím jazyku PHP. Jak uvádí [18], Nette framework má značnou výhodu v tom, že velmi usnadňuje práci, disponuje vynikajícím šablonovacím systémem Latte a bezkonkurenčním ladícím nástrojem Tracy. Důraz klade především na DRY[20], KISS[21], MVC[22] a znovupoužitelnost kódu. Autorem tohoto frameworku je český vývojář David Grudl, jenž také přispívá

[20] Don`t repeat yourself. Způsob vývoje systému, jež se zaměřuje na snižování duplicitních informací
[21] Keep It Simple, Stupid! Princip, který se snaží o co největší jednoduchost bez zbytečností.
[22] Model-view-controller. Softwarová architektura, která rozděluje aplikaci na datový model, uživatelské rozhraní a řídící logiku.

k rozvoji. Celý framework je minimálně v České republice velmi oblíbený, což dokládá také fakt, že ho využívají takové projekty jako jsou např. GE Money, DHL, ESET, Slevomat, CSFD, Uloz.to, Socialbakers, Harley Davidson Praha a v neposlední řadě také oficiální web ex-prezidenta Václava Klause. V roce 2015 získal 3. místo v anketě *Best PHP Framework for 2015* pořádané magazínem SitePoint.

Nette framework je postavený na architektuře MVC resp. MVP, což je softwarová architektura, znázorněná na obrázku 13, která vznikla pro potřeby rozdělit kód obsluhy (controller), kód zobrazovací logiky (view) a kód aplikační logiky. V Nette je místo kontroléru presenter. MVP se od MVC liší dle [19] takto:

- View plně kontroluje uživatelský vstup a výstup
- Oddělení view a presenteru spíše z architektonického důvodu nežli z důvodu ošetření vstupu
- Přímá vazba view na presenter
- Presenter pracuje přímo s view a je zde tedy silnější vazba



Obrázek 13 - Schéma návrhového vzoru MVP (zdroj: autor)

Model

Model lze chápat, jako datový základ aplikace. Základem je aplikační (business) logika, která určitým způsobem komunikuje s kontrolérem, v případě Nette frameworku, s presenterem. Obsahuje také datové entity, jako jsou např. soutěže, kategorie, uživatel apod. Model je v podstatě izolovaný, tzn. že si spravuje svůj vnitřní stav sám a s okolím komunikuje pomoci daného rozhraní. O existenci kontroléru a view neví.

View

View neboli pohled je vrstva uživatelského rozhraní, jež má na starost zobrazení. Presenter získává od modelu surová data, která předá šabloně, jež si je pomocí grafického formátování zpracuje do lépe vnímatelné podoby uživatelem. Lze jej chápat jako výstup aplikace, který je v Nette frameworku reprezentován Latte šablonami.

Latte je šablonovací jazyk, jež umožňuje vkládat data z PHP. Jak uvádí [20], je prvním PHP enginem, který přichází s context-aware escapováním a kontrolou odkazů. Nette pracuje s latte šablonami tak, že jsou v debugovacím režimu pokaždé překládány, v produkčním režimu se přeloží pouze jednou, a to tak že z kódů, který tvoří převážně HTML a speciální Latte značky (makra, filtry apod.), vznikne PHP soubor obsahující HTML a PHP kód, který je uložen ve složce *temp* pro dočasné soubory.

Presenter

Presenter se stará o řízení mezi pohledem (view) a modelem. Presenter je objekt, který přijímá požadavky od uživatele a vygeneruje odpověď. Odpověď nemusí být nutně jen HTML stránka, ale také např. obrázek, JSON[23], přesměrování atd. Funguje tak, že předá data modelu, ten je zpracuje a pošle

[23] JavaScript Object Notation. Formát dat, který je zapisován pomocí javascriptové syntaxe a slouží pro přenos dat organizovaných v polích či objektech.

zpět presenteru a ten je dál předá šabloně k vykreslení. Součástí práce presenteru je i validace uživatelských vstupů.

Nette Framework díky své propracovanosti výborně řeší bezpečnost a nabízí velké množství balíčků, díky kterým se dá dostatečně obohatit funkcionalita vyvíjené webové aplikace. Dané balíčky jsou použitelné i samostatně bez využití celého Nette Frameworku.

3.2.1.4 Využité balíčky

Nette disponuje sadou samostatných a znovupoužitelných balíčků. Na začátku vývoje vznikal Nette framework jako monolit, ale postupem času se začal rozpadat do menších, nezávislých a samostatně použitelných balíčků. V tabulce 2 jsou popsány, jaké balíčky frameworku jsou využity a jaký hrají význam v aplikaci.

Název balíčku	Popis	Význam v aplikaci
Application	Webové aplikační jádro	Zasílání response, routing,
Bootstrap	Bootstrap webové aplikace	Nastavení pro start aplikace (konfigurace běhového režimu, dočasné a logovací složky, instalace doplňků)
DI	Dependency Injection Container	Registrace a následné získání služeb pro vstříknutí závislostí
Forms	Webové formuláře	Tvorba uživatelských vstupů pro modifikaci dat
Http	Zapouzdření http request, response	Ukládání/čtení cookie
Latte	Šablonovací systém	Tvorba uživatelského rozhraní
Mail	Odesílání e-mailů	Zasílání informací o resetu a změně hesla
Robot Loader	Autoloading	Načítání externích knihoven
Security	Správa přístupových práv	Přihlášení, odhlášení, autentifikace, autorizace uživatele

Tracy	Debugovací nástroj	Debugování aplikace, logování chyb
Utils	Pomocné třídy	Enkódování/dekódování JSON, manipulace s poli, datumem, časem a řetězci

Tabulka 2 – Využité balíčky pro webovou aplikaci a jejich význam

3.2.1.5 Řešení zabezpečení

Bohužel se stává, že je často hlášena bezpečnostní díra z nejrůznějších aplikacích. U každého vývojáře by proto mělo být samozřejmostí kladení důrazu na vysokou úroveň zabezpečení. Nette framework naštěstí na tuto problematiku velice důrazně myslí.

Ve webové aplikaci probíhá přihlášení uživatele mimo jiné také přes uživatelské jméno a heslo. Odstrašujícím příkladem nevhodného použití hashovacího algoritmu např. *md5()* nebo *sha1()*, je případ z roku 2017, kdy z Mall.cz unikla hesla, jež byla šifrována právě takto slabými algoritmy. Všechna uživatelská hesla ve vyvíjené webové aplikace jsou proto uložena pomocí bezpečného algoritmu *bcrypt* s 10 ti iteracemi. S tímto algoritmem pracuje statická třída *Passwords*, která se používá ve webové aplikaci. Pro zahešování hesla je využito funkce *hash()*. Pro ověření, zda heslo odpovídá danému otisku, se pak volá funkce *verify()*. Ověření hesla probíhá ve třídě *CredentialsAutheticator* v metodě *authenticate()* v rámci procesu autentifikace.

Dalším častým útokem je XSS (Cross-Site Scripting), který spočívá ve zneužití neošetřených vstupů. Princip spočívá v podstrčení vlastního kódu na stránku, což může vést k odcizení citlivých údajů nebo k poškození vzhledu stránky. Nette framework proto využívá technologie Context-Aware Escaping, která všechny výstupy automaticky ošetřuje.

Nette framework nabízí i ochranu proti útoku CSRF (Cross-Site Request Forgery), který je vykonáván na aplikaci, kde je uživatel přihlášen a může tak

skrytě např. smazat článek bez povšimnutí. Řešením je jednoduché zavolání metody *addProtection()* na formuláři.

Mezi další útoky, kterým se snaží Nette zabránit, jsou URL attack, control codes, invalid UTF-8, session hijacking, session stealing, session fixation a další.

3.2.1.6 Doctrine 2

Webová aplikace vyžaduje práci s databází a proto, aby byl vývoj o něco snazší a čistší, bylo přistoupeno k volbě využití Doctrine 2. Ve webové aplikaci zajišťuje především to, že při potřebě získat data z databáze rovnou namapuje tato data do objektů, což je bezesporu výhoda, kvůli které byla Doctrine 2 zvolena.

Doctrine 2 je, jak se uvádí v [21], ORM[24] framework pro jazyk PHP. Zajišťuje mapování relační databáze do objektů a opačně. Výhodou je, že je vývojář oproštěn od přímého psaní SQL[25], tím vzniká nezávislost na konkrétní databázi a práce s daty se týká objektů.

Pro použití v Nette frameworku je k dispozici balíček na Packagistu *kdyby/doctrine*, který lze nainstalovat pomocí Composeru. Po nainstalování je třeba provést konfiguraci v konfiguračním souboru NEON[26] ve složce *config*.

Samotný Nette framework nabízí vrstvu pro práci s databází, tzv. *nette/database.* Nicméně se nejedná o plnohodnotný ORM framework, a proto bylo spíše využito řešení s Doctrine 2.

Jedním ze stěžejních prvků Doctrine je Entity manager, který je správcem entit. Má za úkol ukládat nebo mazat nové entity. Základním kamenem Entity manageru je jeho jádro, které využívá vzoru Unit of work. To má za úkol pracovat s frontou všech změn, které provede i v databázi. Pro

[24] Object Relational Mapping. Metoda mapování relační databáze na objekty.
[25] Structured Query Language. Dotazovací jazyk, jež je používán v relačních databázích.
[26] Nette Object Notation. Formát souboru pro serializaci PHP pole nebo objektu

zařazení entity pod správu Entity manageru se používá metoda *persist()* a pro provedení všech změn i v databázi je nutné zavolat metodu *flush()*.

Doctrine disponuje i vlastním dotazovacím jazykem s označením DQL (Doctrine Query Language). Rozdíl mezi SQL a DQL není tolik markantní. Jediná odlišnost je, že místo s tabulkami a sloupci se pracuje s objekty a jejich atributy.

Druhým způsobem, jak vytvářet dotazy a který je také nejčastěji využíván při vývoji, je Query Builder. Je to zvláštní třída, jež umožňuje vytvořit databázový dotaz pomocí posloupnosti volaných funkcí. Na obrázku 14 je vidět, jakým způsobem pomocí Query Builderu je získáván seznam ostatního personálu soutěže.

```
public function getOtherStaffByCompetition(int $competitionId) : array {
    $qb = $this->entityManager->createQueryBuilder();
    return $qb->select("os")->from(OtherStaff::class, "os")
        ->join("os.otherStaffCompetitionRelationship", "osc")
        ->where("osc.competition = ?1")
        ->orderBy("os.lastName")
        ->setParameter(1, $competitionId)
        ->getQuery()->getResult();
}
```

Obrázek 14 – Využití Query Builderu při sestavování databázového dotazu (zdroj: autor)

Třetím způsobem vytváření databázových dotazů v Doctrine je využití nativního SQL. Zápis je stejný jako u klasického SQL, tzn. že se využívá tabulek a sloupců. Tohoto způsobu není ale úplně vhodné využít a když je to třeba, tak jen v krajních případech, jelikož vývojář přichází o možnost nezávislosti na konkrétní databázi a také vzniká nutnost manuálního mapování na objekty.

Díky tomuto frameworku lze poměrně jednoduše získávat data z databáze, která jsou pak distribuována pro další zpracovávání nebo mohou sloužit jako vstupní data pro jiné knihovny, které tyto data zpracují do nejrůznějších podob pro potřeby webové aplikace. Jako příklad knihoven lze

uvézt PHP Excel pro XSLX soubory nebo mPDF pro PDF soubory. Možností je také distribuovat data získaná pomocí Doctrine 2 také do okolních aplikací pomocí Rest API.

3.2.1.7 PHP Excel

Jedním z požadavků zadavatele systému bylo, aby webová aplikace uměla generovat různé seznamy pro Microsoft Excel, a to s podporou vzorečků. To znamenalo, že nelze použít jednoduchý formát CSV[27]. Z tohoto důvodu bylo využito knihovny PHP Excel, která požadovanou funkcionalitu zvládá.

Dle [22] PHP Excel používá *lazy loader,* a to znamená, že se načtou pouze požadované třídy, což vede k menší zátěži procesoru a paměti, a tak bude vykonání rychlejší. Mimo jiné umí PHP Excel také načíst data z různých formátů jako je XLS, OpenOffice nebo CSV a také do těchto formátů zapisovat. Součástí této knihovny je nabídka široké škály funkcí, jež jsou obsaženy i v samotném Excelu a byly využity v rámci generování požadovaných dokumentů jako jsou např. bodovacích lístky, seznamy soutěžících, soutěžní vyhodnocení. Ve webové aplikaci obstarává generování třída *ExcelBuilder*.

3.2.1.8 mPDF

Knihova mPDF slouží k tvorbě PDF dokumentů na straně serveru, což je jeden z požadavků zadavatele, a proto je příhodné využít tohoto nástroje ve vyvíjené webové aplikaci, a to pro generování následujících PDF dokumentů: bodovací lístky, seznamy soutěžících a soutěžní vyhodnocení. V praxi funguje tak, že na vstupu obdrží HTML soubor i s dalšími parametry a výstupem je pak PDF dokument.

[27] Comma-separated values. Souborový formát pro tabulková data sestavená z řádků a oddělena čárkou

Pro svůj chod vyžaduje minimálně PHP 5.6 a je kompatibilní s PHP 7.0, 7.1 a 7.2. Do projektu ji lze začlenit pomocí Composeru příkazem *composer require mpdf/mpdf.*

3.2.1.9 Rest API

REST (representional state transfer) je dle [23] množina pravidel v rámci návrhu architektury distribuovaného systému mezi rozhraním a komponentami. Roy Fielding představil REST poprvé ve své dizertační práci v roce 2000. Fielding je mimo jiné také spoluautorem protokolu HTTP. REST pracuje nad komunikačním paradigmatu klient-server se zaměřením na HTTP webové služby.

Touto technologií bude disponovat ve své serverové části i vyvíjená webová aplikace, což byla povinnost zejména kvůli tomu, aby měla zamýšlená mobilní aplikace přístup ke chráněným zdrojům, jež jsou společné pro oba druhy aplikací. Do webové aplikace lze použít veřejně nabízenou implementaci Rest API, která je cílena pro Nette Framework. Nese název *drahak/Restful* a je možné ji přidat do projektu pomocí Composeru příkazem *composer require drahak/restful*. Takové Rest API lze testovat např. pomocí programu Postman od společnosti Google.

Základním prvkem, o které se Rest API opírá, je tzv. zdroj. Zdroj může být jakkoliv reprezentován, např. jako webová stránka, objekt v databázi, dokument nebo také jako stav aplikace apod. Nutnou podmínkou zdroje je disponovat vlastní URL[28].

V Rest API klient a server získávají zdroje pomocí HTTP. Rest API umožňuje díky této komunikaci vyvinout jakoukoliv aplikaci, která využívá

[28] Uniform Resoure Locator. Řetězec znaků, který přesně specifikuje umístění zdroje.

všechny HTTP operace. Tyto operace lze zařadit do CRUD[29] (Vytvořit, Číst, Aktualizovat, Odstranit) a jsou to následující:

- **GET**

Jedná se o základní operaci, která vykonává čtení zdroje, nikoliv jeho modifikace. Operace se považuje za bezpečnou, jelikož nemění stav zdroje. Předpokládá se, že je vždy idempotentní, což znamená, že všechny výsledky vrácené touto metodou, jsou identické do doby, než se provede jiná operace, která modifikuje zdroj. Metoda GET je ve vyvíjené webové aplikaci v rámci Rest API využita pro získání seznamu soutěží, bodovacích lístků, vyhodnocení soutěže.

Bodovací lístek

Parametry:	URL:
competitionId – id soutěže *roundId* – id kola *categoryId* – id kategorie *access_token* – přístupový token	*/api/v1/scoring/get-scoring-list?competitionId=45&roundId=69&categoryId=59&access_token=9ea6d5f8e55f7098afd201822e35b74d8f6e52c4a85963011d1584542f200452*

Seznam soutěží

Parametry:	URL:
access_token – přístupový token	*/api/v1/competitions/get-all/?access_token=d45ec47187d47975fa202f3b2d3055200cd3afcdee65b8b319f44fc18e6dcb2e*

Vyhodnocení soutěže

Parametry:	*competitionId* – id soutěže

[29] Create, Read, Update, Delete. Základní operace nad uložištěm záznamů.

access_token – přístupový token

URL*:*
/api/v1/scoring/competition-evaluation?competitionId=40*&access_token=d45ec47187d47975fa202f3b2d3055200cd3afcdee65b8b319f44fc18e6dcb2e*

- **POST**

Tato metoda slouží k vytvoření nového zdroje. Nedá se považovat za bezpečnou metodu, jelikož mění stav zdroje. Při volání této metody není znám přesný identifikátor, jelikož zdroj ještě neexistuje. Používá se tedy domluvený identifikátor. Ve webové aplikaci v rámci Rest API slouží pro přihlášení uživatele pomocí OAuth 2.0.

Přihlášení uživatele pomocí OAuth 2.0

Hlavičky*:*
Content-Type – application/x-www-form-urlencoded

URL:
/api/v1/authorization/token

Parametry*:*
Grant_type – typ oprávnění OAuth
username – uživatelský email
password – uživatelské heslo
client_id – id aplikace

- **PUT**

PUT je určen pro aktualizaci stávajícího zdroje, ale může být také použit pro vytvoření nového, pokud není daný zdroj k dispozici. Není považována za

bezpečnou metodu, jelikož má schopnost měnit zdroj. Nové hodnoty jsou předávány v těle. Ve webové aplikaci v rámci Rest API je PUT metoda použita pro změnu hesla, kdy se v těle zasílá JSON.

Změna hesla

Hlavičky*:*

Content-Type – application/json

Parametry*:*

access_token – přístupový token

URL:

/api/v1/user/change-password/?access_token=63f819ae2686ecaf0bba09fe8d476cc6afc0a62efefa10879d7557a9ccc290a9

Tělo:

{ "oldPassword": 123456,"
newPassword": 1234567 }

Vyplnění bodovacího lístku

Hlavičky:

Content-Type – application/json

Parametry*:*

access_token – přístupový token

URL:

/api/v1/scoring/fill-scoring-list/?access_token=9ea6d5f8e55f7098afd201822e35b74d8f6e52c4a85963011d1584542f200452

Tělo:

{ "competitionId": 45, "roundId": 69," categoryId": 59,"scores": [{ "scoreId": null, "evaluationId": 65, "numberId": 191, "value": null }] }

- **Delete**

Metoda delete smaže daný zdroj ze serveru na základě předem zadaného id zdroje. Řadí se mezi méně bezpečné metody, jelikož mění stav zdroje. Ve webové aplikaci v rámci Rest API nebylo její využití potřeba.

3.2.1.10 Struktura aplikace

Adresářová struktura vychází tedy z kostry aplikace (sandboxu), který je oficiálně poskytován se zdrojovými kódy k Nette frameworku.

app	adresář s aplikací
AdminModule	modul s prezenční logikou chráněnou uživatelským oprávněním
config	konfigurační soubor
forms	komponenty, resp. formuláře aplikace
FrontModule	modul s veřejně přístupnou prezenční logikou
lang	jazykové překlady
model	aplikační logika
presenters	presentery pro chybové stavy
router	továrna spravující adresy
templates	šablony pro chybové stavy a emaily
bootstrap.php	zaváděcí soubor aplikace
libs	rozšiřující knihovny aplikace
log	chybové hlášení
temp	dočasné a session soubory
vendor	zdrojové kódy Nette frameworku
www	adresář pro podporu designu
css	kaskádové styly
fonts	fonty
icons-reference	doplňující ikony

img	obrázky
js	javascriptové soubory
index.php	inicializace aplikace
.htaccess	konfigurace webového serveru

AdminModule

- LoginPresenter – tvorba komponent pro přihlášení a resetování hesla
- AdminBasePresenter – společný presenter pro administrační modul
- CompetitionsPresenter – tvorba komponenta a předání dat pro zobrazení a editace soutěží, personálu v soutěži, kol, kategorií
- ExportPresenter – Exportování dat do PDF a XLSX
- OptionsPresenter – tvorba komponenta a předání dat pro modifikaci nastavení soutěže
- ScoringPresenter – tvorba komponenta a předání dat pro bodování a vyhodnocování v soutěži

ApiModule

- AuthorizationPresenter – přidělování access tokenu, kontrola oprávnění přes OAuth 2.0
- CompetitionsPresenter – Distribuce struktury soutěží pomocí API
- ScoringPresenter – Distribuce bodovaní pomocí API

FrontModule

- ApplicationPresenter – tvorba komponent a předání dat pro přihlášky do soutěže

Model (popis balíčků)

- Authenticators – autentifikátory – pomocí uživatelského jména a hesla, access tokenu
- Builders – sestavování XLSX a PDF souborů
- Mail – zasílání uživatelských emailů
- Repository – uložiště entit, služeb, sekvencí
- Responses – uložiště pro HTTP response např. XLSX souborů

3.2.2 Klientská část

Podstatnou součástí konceptu webové aplikace je klientská část, což jsou skripty, které zasílá server internetovému prohlížeči a ten je zpracuje. Mezi takové skripty lze řadit HTML stránku, obrázky, CSS soubory, JSON, XML[30], JavaScriptové soubory.

Client-side v sobě zahrnuje grafické rozhraní aplikace, které je velmi důležitým prvkem systému. Takové rozhraní je třeba vždy tvořit tak, aby bylo uživatelsky přívětivé a uživatele neodradilo od jejich každodenního využívání. Také je třeba, aby se vzhled přizpůsoboval moderním trendům jako je např. responzivní design apod. K tomuto faktu mohou pomoci moderní nástroje jako je např. Twitter Bootstrap či jQuery. Tyto nástroje se běžné používají webovými kodéry a návrháři uživatelského rozhraní.

3.2.2.1 Sass

Zápis běžného CSS není pro vyvíjenou webovou aplikaci dostačující, jelikož se nedá dostatečně dobře udržovat a rozvíjet, tak jak by bylo potřeba. Proto je přistoupeno k volbě CSS preprocesoru Sass, jež umožňuje ve webové

[30] eXtensible Markup Language. Značkovací jazyk pro formát dokumentů.

aplikaci lépe minimalizovat opakování kódu, a proto se lépe udržuje. Celkové kódování CSS je proto mnohem pohodlnější.

Sass je CSS preprocesor, který slouží pro tvorbu kaskádových stylů. Rozšiřuje syntax CSS o podmínky, mixiny, cykly, funkce aj. Zásadní výhodou oproti psaní klasického CSS je fakt, že Sass je přehlednější a snadněji se udržuje, což je právě důvod, proč je využit pro implementaci kaskádových stylů ve vyvíjené aplikaci.

V rámci vývoje je použita syntaxe SCSS, tedy Sassy CSS. Po napsání kódu je třeba takový soubor zkompilovat do CSS, které je již čitelné samotným prohlížečem. Výsledek kompilace může mít více podob a jednou z nich je komprimovaný zápis pro lepší optimalizaci, jež je využit v aplikaci v souboru *display.css*.

Pro potřeby vývoje jsou využity mixiny a partials. Mixiny jsou části kódu, jež se často opakují a jsou umístěné ve vyhrazeném souboru *_mixins.scss*. Jedním z mixinů je *border-radius($radius)*, jež umožní vkládat vlastnost *border-radius* včetně prefixů. Partials jsou soubory, jež obsahují SCSS kód, nicméně nejsou určeny k přímé kompilaci. Mezi tyto soubory se v aplikaci řadí *_mixins.scss* a *_utils.scss*. Výhodou tohoto řešení je, že se dá logika vyčlenit a tím zmírnit odpovědnost a zvětšit přehlednost.

Na Sass přešel také Twitter Bootstrap 4, který dříve využíval Less do verze 3. Díky Sassu tak lze využívat ve webové aplikaci mixiny apod., které spadají do zdrojového kódu Bootstrapu.

3.2.2.2 Twitter Bootstrap 4

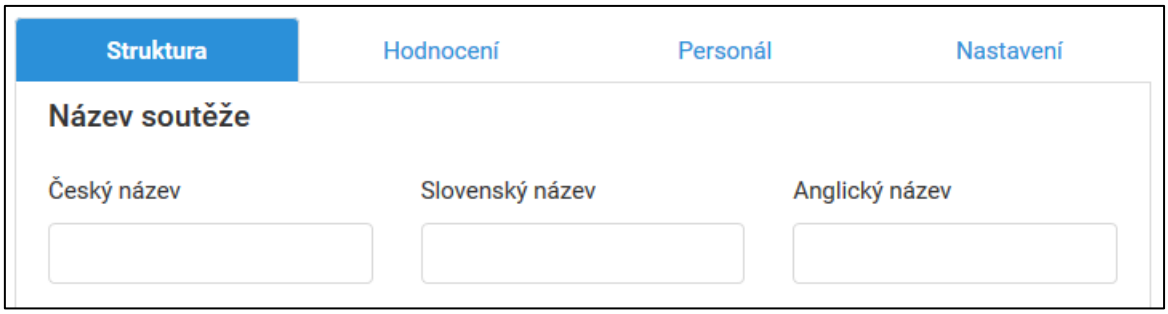
Ve webové aplikaci je třeba klást důraz na design včetně responzivity. Bootstrap je v tomto ohledu zaběhnutá a vynikající knihovna, která pomáhá řešit tyto problémy, proto vznikla tendence využít takto nápomocnou knihovnu. V rámci designu obstarává v aplikaci mimo jiné grid systém a umožňuje využít prvky uživatelského rozhraní apod.

Bootstrap je podle literatury [24] webový front-endový framework, jež slouží pro tvorbu responzivního vývoje napříč prohlížeči a který byl vyvinut Markem Ottem a Jacobem Thorntonem z firmy Twitter. Pod pojem responzivní vývoj lze zařadit vývoj takových aplikací, které jsou flexibilní při různém rozlišení cílového zařízení.

Je postaven na technologiích HTML, CSS, JavaScript. Proto je nutné, aby vývojář disponoval znalostmi těchto technologií, ale také znal JavaScriptovou knihovnu jQuery, jelikož je vyžadována pro správnou funkcionalitu. Pro zprovoznění je dále třeba nalinkovat samotné knihovny Bootstrapu. Možné je využít plné verze nebo zkompilované v souboru *bootstrap.min.**.

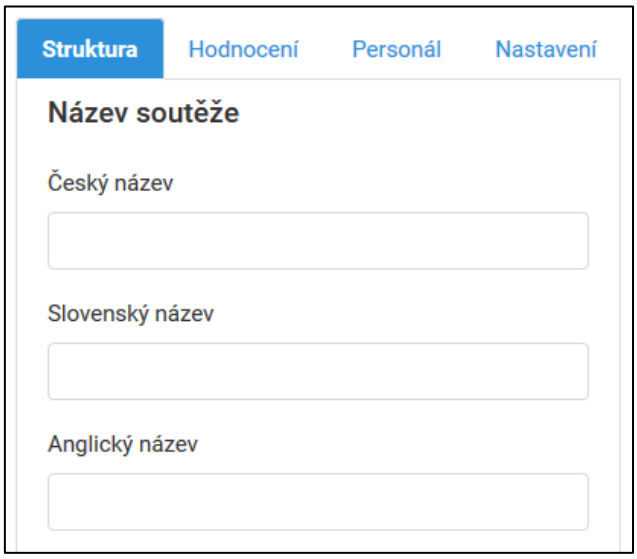
Bootstrap poprvé vyšel jako open source již v roce 2011. Za tuto dobu již prošel mnohým vývojem až po aktuální verzi 4.1.3. (psáno v roce 2018). Během vývoje webové aplikace nastala otázka, zda využít hojně využívanou verzi 3 nebo novější verzi 4. Důvodů pro přechod na verzi 4 je poměrně více. Prvním důvodem může být fakt, že vývoj na starší verzi není krok vpřed a verze 3 může být dříve či později bez podpory. Dalším důvodem je, že Bootstrap 4 přichází s novou mřížkou a flexboxy, což byla mezi vývojáři velice žádaná novinka. Nová mřížka poskytuje 5 rozlišovacích bodů šířky okna, oproti původním čtyřem, které nabízel Bootstrap 3. Dalšími důvody jsou nová *rem* typografie nebo globální nastavení stínů či gradientů.

Ve webové aplikaci byly využity přednosti, mezi které se bezesporu řadí responzivní design. Obrázek 15 znázorňuje, jak vypadá rozložení vstupních polí u názvu soutěže během vytváření nové soutěže při šířce okna 780 px, tedy např. na tabletech.



Obrázek 15 - Zobrazení rozložení prvků při šířce 780 px (zdroj: autor)

Na obrázku 16 je vidět, jakým způsobem Bootstrap přeskládá vstupní pole pro použití na mobilních zařízeních s displejem menším než 768 px.



Obrázek 16 - Zobrazení rozložení prvků při šířce menší než 768 px (zdroj: autor)

3.2.2.3 jQuery

Pro potřeby dynamických změn na straně klienta lze využít JavaScriptu. Nicméně nadstavba nad JavaScriptem, tedy knihovna jQuery, usnadňuje práci s DOMem za použití známých CSS selektorů, a to vede k rozhodnutí nasadit právě tuto knihovnu. Ta v aplikaci zajišťuje spolupráci s Bootstrapem, obsluhu událostí, modifikaci DOMu a mnoho dalšího.

Jak uvádí literatura [25], jQuery je knihovna, jež nabízí víceúčelovou abstraktní vrstvu pro běžné webové skriptování. Je velice jednoduchá a také

intuitivní. Umožňuje vykonávat spoustu úloh, mezi které patří přístup k elementům dokumentu. To je jeho velká výhoda, jelikož bez tohoto přístupu jsou vývojáři nuceni procházet celý strom modelu DOM[31], aby vyhledali určité části dokumentu HTML. Umí dále např. upravovat vzhled webové stránky, měnit obsah dokumentu, reagovat na akce uživatele nebo animovat změny v dokumentu atd. Nicméně je třeba dodat, že se v moderních single page aplikacích již nevyužívá, jelikož nestačí na vše, tzn. že míchá dohromady zpracování událostí, aplikační logiku a manipulaci s DOM. Nahrazuje se tedy nástroji jako je např. React nebo AngularJS, které pracují na odlišném principu. Ovšem jQuery je pro potřeby vývoje tohoto systému dostačující a jedná se také o zaběhnutou knihovnu.

Díky této knihovně lze rozvíjet dynamičnost vyvíjené webové aplikace. jQuery je využita např. pro přepínání záložek u vytváření nové soutěže, reakci na změnu velikosti okna, skrývání či zobrazení různých prvků, client-side validaci formulářů. Na obrázku 17 je vidět implementace funkce, která slouží pro přepínání záložek při vytváření nové soutěže.

```
function nextTab() {
    var activeIndex = $("#competition-menu a.active").index("#competition-menu a");
    var count = $("#competition-menu a").length;

    if (activeIndex !== (count - 1)) {
        $("#competition-menu a").eq(activeIndex + 1).tab("show");
    } else {
        $("#competition-menu a").eq(0).tab("show");
    }
}
```

Obrázek 17 - Přepínání záložek pomocí jQuery (zdroj: autor)

Problémem výběru ze široké škály prohlížečů je fakt, že se mohou objevit různé odchylky, jež se liší od standardního a očekávaného chování. Naštěstí jQuery řeší tyto definované problémy napříč prohlížeči a sjednocuje tak běžné úkoly, což vede ke zmenšení velikosti zdrojového kódu.

[31] Document Object Model. Objektově orientovaná reprezentace HTML nebo XML dokumentu.

Tuto knihovnu lze rozdělit na několik částí. První je jQuery UI, která se zaměřuje na uživatelské rozhraní. Druhou částí je jQuery Mobile, která disponuje optimalizací pro mobilní zařízení. Posledními částmi jsou Sizzle nebo QUnit.

Schopností jQuery je také komunikace s PHP a případné překreslení stránky. Tomuto principu se říká AJAX.

3.2.3 Ajax

Ajax (Asynchronous JavaScript and XML) je technologie, která pracuje z části na serveru a z části u klienta v prohlížeči. Jedná se o způsob, jakým asynchronně lze změnit obsah stránky, aniž by bylo potřeba její znovunačtení. Ajax pracuje se základními třemi prvky, kterými jsou:

- HTML a CSS pro zobrazení dat
- DOM a JavaScript pro dynamickou změnu obsahu stránky
- XMLHttpRequest pro výměnu dat s webovým serverem. Typicky se používá XML, někdy ale lze využít i JSON či jiný formát.

Výhodou je fakt, že není třeba nutnost načíst znovu stránku, stačí pouze překreslit potřebnou oblast. Vyvolání takové akce lze docílit interakcí uživatele či systému. Server po obdržení požadavku od klienta vloží nová data pouze do části stránky, kterou poté odešle. Této části stránky se v Latte říká snippet. JavaScript poté zachytí příchozí požadavek a překreslí danou stránku dle změn snippetu.

Nevýhoda spočívá ve faktu, že pokud není uživateli dostatečně jasně signalizována asynchronní operace, může uživatel, bez toho, aniž by to věděl, takovou operaci přerušit. Ve webové aplikace je tento problém řešen zobrazením obrázkového spinneru. Mezi další nevýhodu lze zařadit i to, že se zvýší počet HTTP požadavků.

V kódu webové aplikace se často vyskytuje volání metody *redrawControl()*, jež se stará o samotné překreslení snippetu. Nicméně Nette Framework nedisponuje implementací Ajaxu na straně klienta. Proto je třeba použít vhodnou knihovnu, která dokáže pracovat se serverem a překreslit danou oblast stránky. Pro tuto potřebu je proto zvolena knihovna od autora Vojtěcha Dobeše s názvem *nette.ajax.js*, pro jejíž spuštění je zapotřebí mít nalinkovanou knihovnu jQuery. Na obrázku 18 je znázorněno, jak lze funkci z této knihovny předat parametry a ta poté interaguje se serverem jehož odpověď je vrácena v JSON a výsledek je zobrazen uživateli.

```
function fillRoundForm(roundId, competitionId, btn) {
    showLoadingSpinner($(btn).closest(".dropdown"));
    $.nette.ajax({
        url: {link fillRoundForm!},
        data: {
            roundId: roundId,
            competitionId: competitionId
        },
        type: "GET",
        complete: function () {
            hideLoadingSpinner($(btn).closest(".dropdown"));
            $("#roundFormModal").modal("show");
        }
    });
}
```

Obrázek 18 – Asynchronní naplnění dat formuláře založené na komunikace PHP a jQuery (zdroj: autor)

3.3 OAuth 2.0

Dle specifikace [26] je OAuth 2.0 autorizační framework, který umožňuje třetím stranám získat omezený přístup ke zdrojům na základě

souhlasu. Tato specifikace také definuje, jak může klient přistoupit ke zdroji, aniž by mu musel vlastník poskytnout svoje přístupové údaje.

Ryan Boyd ve své publikaci [27] rozděluje a popisuje několik klíčových aktérů OAuth takto:

- **Server zdrojů (Resource server)**

  Jedná se o server, na kterém jsou chráněné zdroje, k nímž se přistupuje. Typicky se jedná o API poskytovatele, který chrání různé zdroje jako jsou především data v databázi apod.

- **Vlastník zdroje (Resource owner)**

  Vlastník zdroje je typicky uživatel, jenž má přístup ke zdrojům, které může získávat, modifikovat či mazat.

- **Klient (Client)**

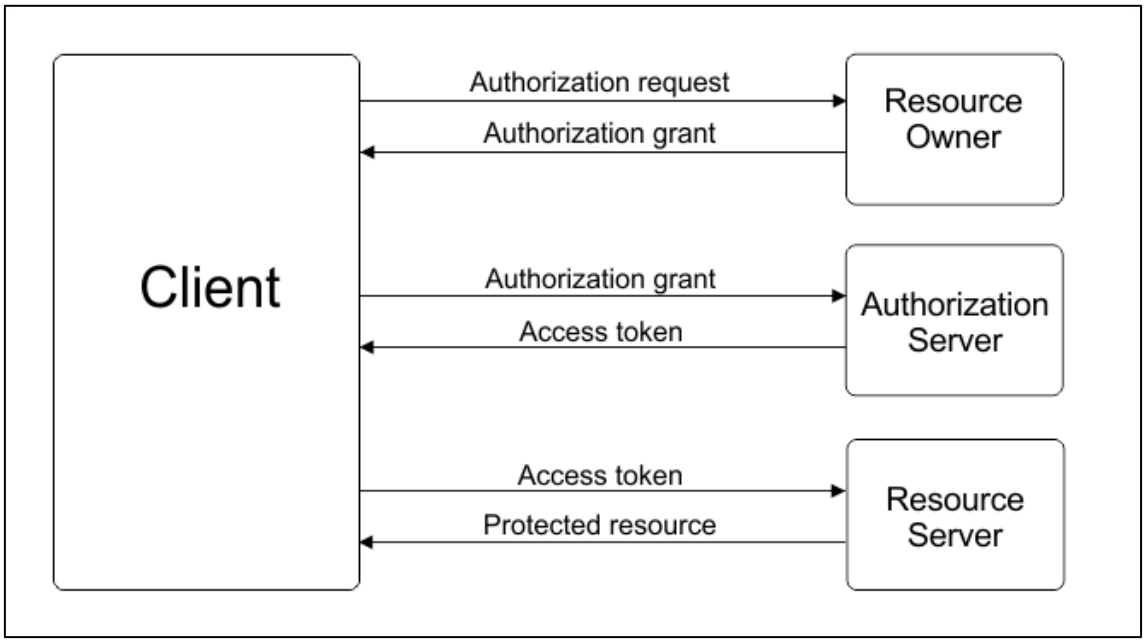
  Je to aplikace třetí strany, která komunikuje s API a žádá server o poskytnutí chráněných zdrojů jménem vlastníka zdroje s jeho oprávněním.

- **Autorizační server (authorization server)**

  Autorizační server vyžaduje souhlas vlastníka zdroje a uděluje přístupový token klientům pro přístup k chráněným zdrojům. U menších poskytovatelů API může mít autorizační server a server zdrojů stejnou adresu.

V prvé řadě je nutné, aby provozovatel zaregistroval klienta na autorizačním serveru. Pro přístup klienta ke zdrojům je třeba získat tzv.

přístupová token (access token). Jedná se o náhodný a tajný řetězec znaků, který opravňuje klienta k přístupu ke zdrojům. Jeho platnost má časové omezení a může být např. serverem zdrojů zrušena. Server zdrojů typicky ověřuje přístupový token na autorizačním serveru. Princip fungování je blíže znázorněn na obrázku 19.



Obrázek 19 – Princip fungování OAuth 2.0 (zdroj: autor)

OAuth 2.0 je nezbytnou součástí celého konceptu systému, ve kterém hraje důležitou roli ochrany zdrojů dat, jež jsou uložené na webovém serveru. Je tedy nutné využít této technologie, a proto je do projektu přidán doplněk OAuth 2 provider, který vykonává funkcionalitu, znázorněnou na obrázku 19, ze strany serveru. Takový doplněk lze přidat pomocí Composeru příkazem *composer require drahak/oauth2:@dev* od autora Drahomíra Hanáka. Ze strany klienta, v rámci vyvíjeného systému, bude přistupovat mobilní aplikace pro potřeby získat data ze serveru zdrojů.

3.4 Mobilní aplikace pro Android

V rámci systému je vyvíjena také mobilní aplikace. Důvod rozhodnutí vytvořit aplikaci také pro mobilní platformu je ten, že využívání aplikací nainstalovaných přímo v mobilním zařízení v dnešní době stále roste a je tedy trendem, aby byly takové aplikace nabízeny zákazníkům. Důkazem je dle analytické agentury Gartner [9] nárůst nákupu chytrých telefonů mezi prvními kvartály roků 2017 a 2018 o 1,3 % na daném vzorku lidí. V rámci tohoto šetření se také nabízí informace, na jaký mobilní operační systém se zaměřit. Dle statistických dat [9] využívá Android 85,9 % v prvním kvartálu roku 2018 ze vzorku 383 503 900 uživatelů. Obě tato tvrzení proto tedy vedou k tomu vyvinout mobilní aplikaci, a to přímo pro operační systém Android. Nespornou výhodou může být také fakt, že mobilní zařízení obsahují různé druhy senzorů např. pro geolokaci, což může být do budoucna myšlenka pro další vývoj.

Android je tedy mobilní operační systém, založený na jádře Linuxu. Jedná se o tzv. open source, což je software s otevřeným zdrojovým kódem.

Doporučovaným vývojovým prostředím je Android Studio, jež má základ na produktu IntelliJ Idea. Android Studio umožňuje vývoj pomocí jazyka Java nebo Kotlin, proto je potřeba znalost alespoň jednoho z nich. Z tohoto důvodu je třeba mít nainstalován JDK (Java Development Kit) a SDK (Software Development Kit) pro Android. Android Software Development Kit nabízí sadu nástrojů pro vývoj, mezi které se řadí:

- **SDK Tools** – správa Android Virtual Devices, nástroje pro testování, Android emulátor apod.
- **Android SDK platforms** – systémový obraz, ukázky kódu, knihovny apod.
- **USB Driver** – umožňuje ladění aplikace na živém zařízení
- **Google API** – podpora pro Google Maps apod.

3.4.1 Popis aplikace

Mobilní aplikace slouží pro určité uživatelské role jako alternativa k webové aplikaci. Nabízí pohodlnou práci se systémem soutěžícímu či porotci nebo předsedovi poroty, což jsou jediné role, pro které je aplikace určena.

V roli soutěžícího lze z mobilní aplikace po přihlášení přistupovat k hodnocení, a to jak k hodnocení v daném kole a kategorii, tak k hodnocení za celou soutěž.

Porotce má pravomoc přistupovat k bodování v soutěži opět po předchozím uživatelském přihlášení. Během bodování je omezen dle nastavení soutěž, tzn. že např. nesmí bodovat soutěžícího ze stejné školy, pokud je tak definováno, dále smí bodovat až po maximální hranici, která je definována v nastavení soutěže.

Pokud chce uživatel využívat aplikaci na svém mobilním zařízení s operačním systémem Android, je třeba vlastnit alespoň verzi 4.4 KitKat.

3.4.2 Základní součásti aplikace pro Android

Autor publikace [28] Ľuboslav Lacko dělí základní prvky operačního systému Android na *aktivity*, jež reprezentují obrazovku, dále *service* pro vykonání akcí na pozadí, *content providers* pro přístup ke zdroji dat, *broadcast reciever* pro zpracování příchozích oznámení. Tyto prvky mohou mezi sebou komunikovat pomocí *intentů.*

- **Activity**

Aktivita reprezentuje obrazovku, na které jsou prvky grafického uživatelského rozhraní pro interakci s uživatelem. Její využití je např. pro vyplnění bodovacího lístku, zobrazení soutěží a bodování atd.

Výhodou zařízení s větším rozlišením je realizace komplexnějších úkonů. Jako příklad lze uvést aktivitu master-detail, jež umožňuje zobrazit seznam položek a detail položky ve dvou obrazovkách na menších zařízeních, zatímco na větších obrazovkách proběhne zobrazení na jedné obrazovce, tedy seznam položek na straně jedné a detail položky na straně druhé.

Z určitého pohledu je aktivita náročná záležitost, jelikož se při prvním spuštění vytvoří nový proces a alokuje se paměť pro komponenty uživatelského rozhraní. O životní cyklus aktivity se stará Activity Manager, který pracuje se zásobníkem, kde jsou zaznamenány spuštěné aktivity. Na vrcholu zásobníku je aktivita, jež je uživateli právě zobrazena.

Stavy aktivit, ve kterých se volá patřičný kód, lze rozdělit do čtyřech stavů:

Activity starts

Stav, sloužící pro inicializaci aktivity. V prvé řadě je volána metoda *onCreate()*, kde se vytváří uživatelské rozhraní, např. vstupní pole pro zadání bodování. Také se zde konfigurují proměnné a objekty pro běh aktivity. Dalšími metodami jsou *onStart()* a *onResume()*. Jedná se o metody pro přechod do/z pozadí a mají tedy odpovědnost za skrytí/zobrazení uživatelského rozhraní.

Activity is running

Během tohoto stavu je aktivita v popředí. Aktivita ale může také přejít do pozadí v důsledku upřednostnění jiné aktivity, přičemž dojde k zavolání metody *onPause()*. V případě nedostatku paměti může ale také dojít k přechodu do stavu *proces is killed*. Jestliže metoda není viditelná, dojde k zavolání metody *onStop()*. Pokud se daná aktivita probudí, je volána metoda *onRestart()*, čímž může dojít až k zavolání *onStart()*. Při tomto přechodu ale může opět dojít k nedostatku paměti, tedy stavu *proces is killed*. Během zničení aktivity se volá metoda *onDestroy(),* tedy přechod do stavu *aktivity is shut down*, pokud nebyla předtím zavolána metoda *onStop().*

Process is killed

Activity Manager může rozhodnout o zničení aktivity, a to z důvodu nedostatku paměti. Pak tedy aktivita přechází do stavu *process is killed*. Také ale může dojít ke znovuspuštění aktivity a to, pokud uživatel takový signál vyšle. V tomto případě by došlo k zavolání metody *onCreate().*

Activity is shut down

Pokud Activity Manager rozhodne o ukončení aktivity, dojde k přechodu do stavu *aktivity is shut down* a taková aktivita již nevyužívá žádnou paměť.

- **Service**

Service neboli také služba realizuje operace, jež se vykonávají na pozadí. Od běžných aktivit se liší tím, že nepotřebují uživatelské rozhraní. Umožňují pracovat asynchronně, tedy paralelně s hlavním

vláknem a procesy tedy mohou trvat delší dobu. Od třídy *AsyncTask* se liší tím, že vykonávají dlouho trvající operace. Třída *AsyncTask* byla ve vyvíjené mobilní aplikaci využita a slouží např. pro jednorázové získání dat ze serveru.

- **Broadcast receiver**

  Broadcast receiver naslouchá událostem a poté na ně reaguje. Stejně jako service ani broadcast receiver nedisponuje uživatelským rozhraním. Události jsou reprezentované jako objekty typu *Intent*, tedy záměr a jsou broadcast receiver vysílány či přijímány.

- **Content provider**

  Content provider (poskytovatel obsahu) umožňuje ukládání a sdílení dat mezi více aplikacemi či procesy. Využívají rozhraní podobající se databázovému, nicméně poskytovatel obsahu je více než jen databáze.

3.4.3 Použité třídy

HttpUrlConnection

Jedná se o třídu, jež pracuje s HTTP protokolem. Ve vyvíjené mobilní aplikaci slouží pro přenos dat mezi serverem a mobilní aplikací. Jako formát dat je zvolený JSON. K vykonávání své funkce potřebuje objekt typu URL, pod kterým se může skrývat např. adresa k seznamu soutěží. Tato třída takto získá potřebná data a předá je dalšímu zpracování, čímž je parsování JSON souboru. Také ji lze použít k zaslání dat serveru danou metodou.

JSONObject, JSONArray

Obě třídy slouží pro parsování JSON formátu, kdy je na vstupu textový řetězec ve formátu JSON a na výstupu lze tento formát přímo číst pomocí rozhraní, které tyto třídy nabízí. Samotné čtení probíhá pomocí volání metod, jež odpovídají datovým typům daných uzlů v JSON.

AsyncTask

AsyncTask je užitečná ve chvíli, kdy je potřeba vykonat operaci, jež má běžet asynchronně, tedy mimo hlavní vlákno. Použitelná je pro krátké asynchronní operace, jakým může být např. získání či uložení bodování. Pro svůj chod využívá 4 základní metody:

- **onPreExecute()**

  V metodě *onPreExecute()* se vykonává kód před samotným spuštěním na pozadí, např. signalizace probíhající operace

- **doInBackground(Params...)**

  Tato metoda vykonává kód, jež se má provést na pozadí mimo hlavní vlákno.

- **onProgressUpdate(Progress...)**

  Pro potřebu indikace stavu průběhu operace běžící na pozadí slouží metoda *onProgressUpdate(Progress...).*

- **onPostExecute(Long result)**

  Tato metoda slouží pro vykonání kódu, jež se má stát po ukončení operace na pozadí.

NetworkInfo

NetworkInfo je třída, jež se dá použít pro detailnější informace o stavu připojení k síti. Ve vyvíjené mobilní aplikaci je využita pro zjišťování, zda je dané zařízení připojené k internetu.

SharedPreferences

Důležitou součástí mobilní aplikace je uložiště *SharedPreferences*. Jedná se o uložiště, které udržuje data i po ukončení či restartu aplikace. Data jsou ukládána jako párové hodnoty typu klíč-hodnota. Použita je k trvalému uložení access tokenu, který se ukládá po ověření uživatele. Společně s ním také uživatelská role pro rozlišení, jaké uživatelské prostředí má být předpřipraveno. Důležitým prvkem je také jazykové nastavení, které je ukládáno.

Pro zápis do *SharedPreferences* se používá *SharedPreferences.Editor*, který nabízí následující metody:

- putString(String key, String value)
- remove(String key)
- putInt(String key, int value)

Poté, co se provede zápis je třeba potvrdit změny, což se provádí podobně jako v databázové terminologii zavoláním *SharedPreferences.Editor.commit()*.

Naopak pro čtení jsou k dispozici následující metody:

- getAll()
- getBoolean(String key, …)
- getString(String key, …)

3.4.4 Balíčky aplikace

Balíčky mobilní aplikace jsou rozděleny na aktivity, komponenty, služby, výjimky, entity, URL zdroje a pomocné třídy (utils).

cz.jphsw.activities

- **CompetitionEvaluationActivity** – zobrazení vyhodnocení v daných kategoriích (za jednotlivá kola, za celou soutěž), bodování jednotlivých porotců
- **CompetitionsActivity** – poskytuje seznam soutěží
- **EvaluationActivity** – zobrazení vyhodnocení v daných kategoriích za dané kolo
- **LoginActivity** – zobrazení přihlašovacího rozhraní pro uživatele
- **ProtectedActivity** – společná aktivita pro všechny aktivity. Definuje jazykovou mutaci a zdroj adres
- **RoundActivity** – zobrazení detailu daného kola včetně seznamu kategorií
- **ScoringActivity** – zobrazení bodovacího formuláře
- **SecuredActivity** – společná aktivita pro aktivity, jež jsou dostupné až po přihlášení

cz.jphsw.model.components

- **CategoryListAdapter** – adapter pro naplnění seznamu kategorií
- **CompetitionEvaluationPlaceholderFragment** – třída reprezentující fragment se seznamem hodnocení za celou soutěž
- **CompetitionEvaluationSectionsPagerAdapter** – adapter pro naplnění menu položkami pro přepínání fragmentů reprezentující kolo

- **CompetitionListAdapter** – adapter pro naplnění seznamu se soutěžemi
- **RoundListAdapter** – adapter pro naplnění seznamu s koly
- **ScoringListAdapter** – adapter, jež naplní seznam pro bodování

cz.jphsw.model.exceptions

Balíček, jenž obsahuje vlastní výjimky, např. pro autentifikační proces nebo pro nesprávný požadavek.

cz.jphsw.model.repository.entities

Balíček obsahující třídy, jež reprezentují entity

cz.jphsw.model.repository.urlresources

Balíček se třídami s uvedenými adresami pro testovací a produkční server

cz.jphsw.model.services

- **CompetitionsService** – třída umožňující získání seznamu soutěží
- **ScoringService** – třída umožňující získání a modifikování bodování soutěžících
- **SecuredService** – společná třída pro třídy, jež využívají přístupu ke chráněnému zdroji dat
- **Service** – společná třída pro všechny třídy
- **UserService** – třída umožňující uživatelskou autentifikaci

cz.jphsw.model.utils.evaluation

Balíček obsahující třídy, jež reprezentují seznamy pro hodnocení, které vykonávají různé matematické výpočty

cz.jphsw.model.utils

- **InputFilterMinMax** – validace vstupních prvků bodování pro kontrolu minimální a maximální hodnoty
- **NetworkUtil** – třída poskytující informaci o stavu internetového připojení
- **NumberHelper** – třída pro pomocnou aplikační logiku
- **RoundsList** – třída reprezentující seznam kol se schopností serializace
- **ScoringList** – třída reprezentující bodovací list s možností převodu do formátu JSON
- **UrlConnector** – třída, pomocí níž lze odeslat data na server pomocí *put* nebo *post* metody nebo lze získat data ve formátu JSON prostřednictvím URL adresy.

4. Testování

V rámci testování je třeba nasadit aplikaci na produkční server. Pro testovací nasazení proto byl vybrán hosting www.hukot.net. Výběr probíhal trochu obtížněji, jelikož každý hosting nenabízí kombinaci PostgreSQL a PHP 7.1 a proto padla volba právě na www.hukot.net. Webová aplikace je tedy dostupná na adrese www.souteze-jphsw.cz.

Během testování webové aplikace byly kladeny nároky na správnou funkčnost aplikační logiky a také na flexibilitu responzivního designu na různých zařízeních počínaje mobilními telefony, přes tablety až po notebooky

a desktopy. Na těchto zařízeních se nevykytovaly žádné zásadní problémy, a proto bylo rozhodnuto o způsobilosti pro pilotní nasazení.

Mobilní aplikace byla testována na dvou typech zařízení. Prvním byl emulátor, což je virtuální zařízení nabízené Android Studiem, které nabízí vytvoř emulaci libovolné verze Androidu. Aplikace byla testována na verzích od Android 4.4 až po verzi 8.1 a nebyly vykazovány žádné znatelné problémy. Druhým typem bylo přímo reálné zařízení s Android 5.1 a 7.0. Ani tam nevznikaly zásadní komplikace.

První pilotní nasazení proběhlo pod záštitou firmy JPH Software na ZUŠ Evy Randové v Ústí nad Labem na mezinárodní klavírní soutěži Virtuosi per musica di pianoforte. Systém, který byl zde nasazen, nevykazoval žádné zásadní problémy, avšak vyvstaly nové poznatky, jakou cestou by mohl vývoj dále směřovat.

5. Stručná uživatelská příručka

Uživatelská příručka si klade za cíl seznámit uživatele s některými funkčními možnostmi, které se nemusejí zdát na první pohled zcela jasné. Rutinní funkce, které jsou srozumitelné na první pohled a známé z jiných systémů, vynechává. Zaměřuje se jak na webovou aplikaci, tak na mobilní aplikaci.

5.1 Webová aplikace

5.1.1 Podání přihlášky

Podání přihlášky může vykonat každý nepřihlášený uživatel nebo soutěžící, jenž byl předtím uživatelsky ověřen. Tato možnost slouží k závaznému přihlášení do soutěže. Uživatel, jenž není přihlášen, musí vybrat do jaké soutěže a potažmo do jaké kategorie se chce přihlásit. Dále je třeba

vyplnit, jaké skladby bude prezentovat a jejich stopáž. V případě, že je uživatel přihlášený, není povinen vykonat registraci. V opačném případě je ještě povinné vyplnit další údaje spojené s registrací. 20 znázorňuje formulář přihlášky pro přihlášeného uživatele.

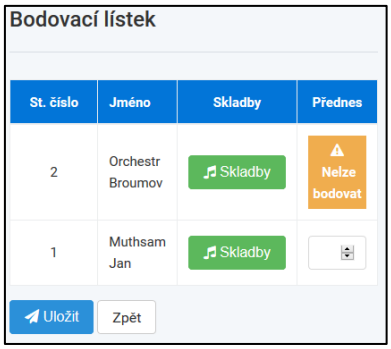
Obrázek 20 – Formulář pro přidání přihlášky přihlášeného uživatele (zdroj: autor)

5.1.2 Vložení startovacího čísla

Tato funkce slouží pro vygenerování pořadí soutěžících. Ovládat ji může pouze správce soutěže, a to po tom, co organizátoři soutěže vylosují, jakým startovacím číslem bude seznam soutěžících začínat. Systém poté vygeneruje seznam, jímž soutěžící s daným startovacím číslem začíná. Tuto funkci je třeba vykonat vždy na začátku soutěže.

5.1.3 Bodování

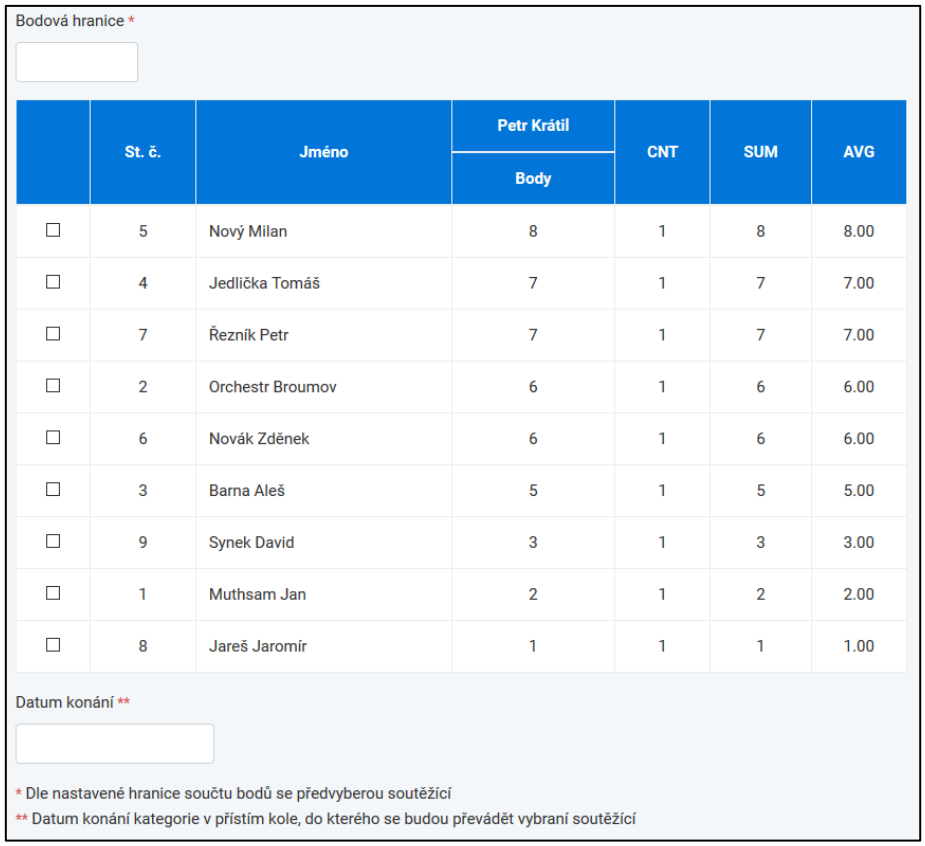
Možnost bodovat má k dispozici pouze porotce nebo předseda poroty. Samotné bodování, jež je zobrazeno na obrázku 21, ovlivňuje také nastavení soutěže, což znamená, že porotce či předseda poroty smí, či nesmí bodovat soutěžící ze stejné školy nebo je udělování bodů limitováno horní hranicí.



Obrázek 21 – Bodovací lístek (zdroj: autor)

5.1.4 Vyhodnocení soutěžících v daném kole

Jedná se o funkci, jež v případě, že se nejedná o poslední kolo soutěže, provede uzavření bodování a zobrazí celkový výsledek soutěže za vyhodnocovanou kategorii. V případě, že se vyhodnocuje kategorie v kole, které není poslední, je třeba, aby správce soutěže vybral soutěžící, kteří mají postoupit do dalšího kola a také zadal datum konání v dalším kole, do kterého se převod provádí. Tuto možnost, znázorněnou na obrázku 22, smí využívat pouze správce soutěže, a to ve chvíli, kdy je vloženo startovací číslo a alespoň jeden porotce provedl bodování. V nastavení soutěže je zapínatelná kontrola, která ověřuje, zda provedli bodování všichni porotci.

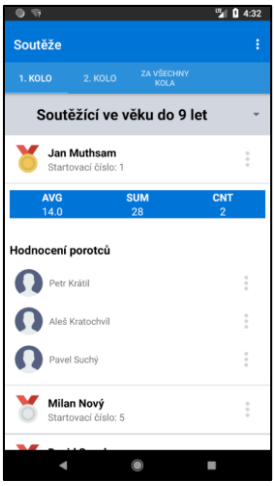


Obrázek 22 – Vyhodnocení kategorie v prvním kole (zdroj: autor)

5.2 Mobilní aplikace

5.2.1 Zobrazení vyhodnocení

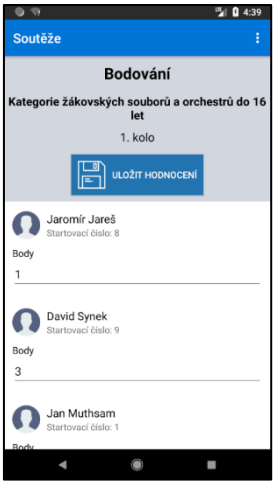
Zobrazení vyhodnocení je možné jak z role soutěžícího, tak z role porotce či předsedy poroty. Dostat se k tomuto vyhodnocení lze ze seznamu soutěží přes rozcestník. Uživateli se poté zobrazí obrazovka znázorněná na obrázku 23. K dispozici je menu v horní části obrazovky, jež umožňuje volit vyhodnocení za dané kolo, či za celou soutěž. Pod menu se nachází přepínač volby kategorie. Po ťuknutí na jméno soutěžícího se otevře detail bodování. V detailu se nachází i seznam porotců, u kterých lze opět ťuknutím na jejich jméno otevřít detail jimi udělených bodů z různých hledisek bodování.



Obrázek 23 – Vyhodnocení soutěže na mobilní aplikaci (zdroj: autor)

5.2.2 Bodování

K zobrazení bodování se dostane pouze porotce, či předseda poroty, a to ze seznamu soutěží přes výběr kola a ťuknutí na volbu kategorie. Uživatel, který vkládá bodování, má opět stejná omezení jako ve webové aplikaci. Obrazovka, která se zobrazí porotci či předsedovi poroty, je znázorněna na obrázku 24.



Obrázek 24 – Formulář pro zadání bodování v mobilní aplikaci (zdroj: autor)

6. Závěr

Primárním cílem této práce bylo vytvořit pod záštitou firmy JPH Software komplexní systém pro správu českých a mezinárodních hudebních soutěží, jež by zákazníkům otevřel nové možnosti a ulehčil práci se sčítáním bodů, vyhodnocováním, shromažďováním přihlášek a dalšími rutinami.

Na začátku práce bylo třeba shromáždit funkční a nefunkční požadavky od zákazníka, vytvořit analytický a návrhový model. Poté bylo přistoupeno k samotné implementaci za použití vývojových prostředků. Nakonec proběhlo testování a pilotní nasazení u prvního zákazníka ZUŠ Evy Randové Ústí nad Labem na soutěži Virtuosi per musica di pianoforte, na který se každoročně sjíždějí desítky soutěžících z celého světa.

Výsledkem vývoje je webová a mobilní aplikace. Webová aplikace umožňuje podat soutěžícím přihlášky a nahlížet na výsledky bodování. Porotce a předseda poroty má možnost, za určitých podmínek vkládat bodování. Správce soutěže má oprávnění pro vytváření soutěže, export bodovacích lístků či seznamu soutěžících a jejich výsledku. Dále je mu umožněno uzavírat respektive vyhodnocovat bodování nebo vložit startovací číslo a tím spustit proces vygenerování pořadí. Mobilní aplikace pro Android slouží pro všechny porotce, předsedy poroty a soutěžící. Pro porotce a předsedu poroty je připraven bodovací formulář. Soutěžíci má umožněno sledovat průběžné a konečné bodování.

Během testování a nasazení byly zjištěny poznatky, kterou mohou být přínosem pro budoucí vývoj systému. Např. směřovat vývoj mobilní aplikace pro širší spektrum operačních systémů.

Celý systém bude bezesporu přínosem pro spoustu základních uměleckých škol a jiných subjektů, které pořádají hudební soutěže, jelikož zásadním způsobem ulehčuje a zrychluje práci oproti manuálním úkonům.

Seznam obrázků

Seznam tabulek

Literatura

[1] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

[2] ČÁPKA, David. 2. díl – UML – Use Case Diagram. *ITnetwork.cz* [online]. [cit. 2018-01-16]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram

[3] ČÁPKA, David. Lekce 5 - UML – Class diagram. *ITNetwork.cz* [online]. [cit. 2018-06-07]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model

[4] ZIMMEROVÁ, Barbora. *Analytický model tříd - 1. část* [online]. [cit. 2018-06-07]. Dostupné z: https://www.fi.muni.cz/~buhnova/PV167/06_AnalytickyModelTrid_I.pdf

[5] ZIMMEROVÁ, Barbora. *Návrhový model tříd - 1. část* [online]. [cit. 2018-06-08]. Dostupné z: https://www.fi.muni.cz/~buhnova/PV167/10_NavrhovyModelTrid_I.pdf

[6] MLEJNEK, Jiří. Návrh – návrhové třídy a vzory [online]. [cit. 2018-06-09]. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-SI1/prednasky/BI-SI1-P06m.pdf

[7] HOLUBOVÁ, Irena, Jiří KOSEK, Karel MINAŘÍK a David NOVÁK. *Big Data a NoSQL databáze*. Praha: Grada, 2015. Profesionál. ISBN 978-80-247-5466-6.

[8] TYL, Pavel. *Konceptuální modelování* [online]. [cit. 2018-06-12]. Dostupné z: https://www.tutorialspoint.com/dbms/er_model_basic_concepts.htm

[9] *Gartner Says Worldwide Sales of Smartphones Returned to Growth in First Quarter of 2018* [online]. [cit. 2018-06-13]. Dostupné z: https://www.gartner.com/newsroom/id/3876865

[10] KUTÁČ, Pavel. *Jak na git díl 0 - Co, proč, jak?* [online]. [cit. 2018-06-14]. Dostupné z: http://www.kutac.cz/blog/pocitace-a-internety/jak-na-git-dil-0-co-proc-jak/

[11] *Úvod – Základy systému Git* [online]. [cit. 2018-06-14]. Dostupné z: https://git-scm.com/book/cs/v1/%C3%9Avod-Z%C3%A1klady-syst%C3%A9mu-Git

[12] MENCL, Michal. *Základní kurz 1: Úvod* [online]. [cit. 2018-06-18]. Dostupné z: http://www.pehapko.cz/zakladni-kurz/1-uvod

[13] Historical trends in the usage of server-side programming languages for websites. *W3Techs.com* [online]. [cit. 2018-06-18]. Dostupné z: https://w3techs.com/technologies/history_overview/programming_language

[14] JELÍNEK, Lukáš. *HHVM: co to je, proč používat, jak na to* [online]. [cit. 2018-06-18]. Dostupné z: https://www.linuxexpres.cz/software/hhvm-co-to-je-proc-pouzivat-jak-na-to

[15] MONUS, Anna. *PHP 7: 10 Things You Need to Know* [online]. [cit. 2018-06-18]. Dostupné z: https://www.hongkiat.com/blog/php7/

[16] Turbocharging the Web with PHP 7. *Zend.com* [online]. [cit. 2018-06-18]. Dostupné z: https://pages.zend.com/rs/zendtechnologies/images/PHP7-Performance%20Infographic.pdf

[17] PETROFČÍK, Matúš. Composer. *ITnetwork.cz* [online]. [cit. 2018-06-22]. Dostupné z: https://www.itnetwork.cz/php/ostatni/composer

[18] Seznámení s Nette Frameworkem. *Nette.org* [online]. [cit. 2018-06-19]. Dostupné z: https://doc.nette.org/cs/2.4/getting-started

[19] BOREK, Bernard. *Prezentační vzory z rodiny MVC* [online]. [cit. 2018-06-20]. Dostupné z: https://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/

[20] Latte. *Nette.org* [online]. [cit. 2018-06-20]. Dostupné z: https://latte.nette.org/cs/

[21] TICHÝ, Jan. Doctrine 2: úvod do systému. *Zdroják.cz* [online]. [cit. 2018-06-22]. Dostupné z: https://www.zdrojak.cz/clanky/doctrine-2-uvod-do-systemu/

[22] KUTÁČ, Pavel. PHPExcel – generování souborů MS Excel. *Kutac.cz* [online]. [cit. 2018-06-22]. Dostupné z: http://www.kutac.cz/blog/weby-a-vse-okolo/phpexcel-generovani-souboru-ms-excel/

[23] HORDĚJČUK, Vojtěch. REST. *Voho.eu* [online]. [cit. 2018-06-23]. Dostupné z: http://voho.eu/wiki/rest/

[24] MORETO, Silvio, Matt LAMBERT, Benjamin JAKOBUS a Jason MARAH. *Bootstrap 4 – Responsive Web Design*. Birmingham: Packt Publishing, 2016. ISBN 978-1-78839-731-5.

[25] CHAFFER, Jonathan a Karl SWEDBERG. *Mistrovství v jQuery: [kompletní průvodce vývojáře]*. Brno: Computer Press, 2013. Mistrovství. ISBN 978-80-251-4103-8.

[26] HARDT, Dick. The OAuth 2.0 Authorization Framework. *IETF Tools* [online]. [cit. 2018-06-26]. Dostupné z: https://tools.ietf.org/html/rfc6749

[27] BOYD, Ryan. *Getting started with OAuth 2.0*. Sebastopol, Calif.: O'Reilly, c2012. ISBN 978-1-449-31160-5.

[28] LACKO, Ľuboslav. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

Přílohy

Obsah přiloženého CD-ROM

- Zdrojové soubory webové aplikace (web-souteze.zip)
- Zdrojové soubory mobilní aplikace (android-souteze.zip)
- Záloha databáze (db.backup)
- Instalační soubor pro mobilní aplikaci (app.apk)