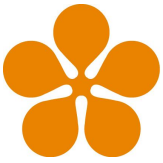


Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Automatizace analýzy výsledků informatické soutěže

Bakalářská práce

Vypracoval: Martin Tichý
Vedoucí práce: doc. PaedDr. Jiří Vaníček, Ph.D.

České Budějovice 2016

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin TICHÝ**
Osobní číslo:	**P13817**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Automatizace analýzy výsledků informatické soutěže**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Soutěž Bobřík informatiky poskytla za 7 let existence velké množství dat, které je možné využít k vylepšování úloh, běhu soutěže, ke zjišťování informatických znalostí českých žáků i pro tvorbu dalších nástrojů pro podporu výuky informatiky. Zpracování těchto dat v současné verzi je časově náročné a statistiky ukazují pouze základní údaje jako je počet zúčastněných škol, žáků a výčty výsledků. Z hodnocení není zcela patrné, zda otázky byly dostatečně srozumitelné nebo odpovídají dané lehké, střední nebo těžké úrovni.

Student vytvoří moduly popř. rozhraní v jazyce PHP, které bude výsledky automatizovaně zpracovávat a vyhodnocovat pomocí statistických metod uživatelsky přívětivou formou tak, aby byly umístitelné do webu soutěže. Celkové hodnocení by probíhalo plně automaticky s možností nastavení některých parametrů. Systém by se zaměřil také na kvalitu a náročnost otázek, jejich dopad na hodnocení žáka a skutečnou obtížnost otázek na základě výsledků v závislosti na typu otázek. Další funkcionalitou bude vymezit otázky, které rozhodují o úspěšnosti jednotlivých řešitelů.

Rozhraní se bude připojovat na stávající MySQL databázi a provádět zadané operace. Součástí systému bude také generátor grafů do formátu PNG nebo pro moderní webové prohlížeče interaktivních grafů v HTML5. Vypočtené hodnoty bude možné též exportovat do formátu XLS/XLSX, pro případnou potřebu dalších speciálních předem nespecifikovaných výpočtů.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **TOMCSÁNYI, P. Náročnosť úloh v súťaži Informatický bobor. In: Konferencia DidInfo 2009. - Banská Bystrica : Univerzita Mateja Bela, 2009. - [nestr.]. - ISBN 978-80-8083-720-4**
2. **TOMCSÁNYI, P., VANÍČEK, J. International comparison of problems from an informatic contest. In: ICTE 2009 : Information and Communication Technology in Education 2009. - Ostrava : University of Ostrava, 2009. - ISBN 978-80-7368-459-4. - S. 219-221**
3. **VANÍČEK, J. Bebras Informatics Contest: Criteria for Good Tasks Revised. In Informatics in schools: teaching and learning perspectives : 7th international conference on informatics in schools : situation, evolution, and perspectives, ISSEP 2014, Istanbul, Turkey, September 22-25, 2014: proceedings. Editors: Gülbahar, Yasemin, Karataş, Erinç. - Book series: Lecture notes in computer science, Vol. 8730. ISSN 0302-974. Cham : Springer International Publishing, 2014. ISBN 9783319099576. p. 17-28.**
4. **NIXON, Robin. Learning PHP, MySQL & JavaScript: with jQuery, CSS & HTML5. 4th edition. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2015. ISBN 978-1-49191-866-1.**
5. **MACDONALD, Matthew. HTML5: The Missing Manual. 2nd Edition. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2014. ISBN 978-1-44936-326-0.**

Vedoucí bakalářské práce: **doc. PaedDr. Jiří Vaníček, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **22. dubna 2015**

Termín odevzdání bakalářské práce: **29. dubna 2016**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 22. dubna 2015

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 2. května 2016

_____

Martin Tichý

Abstrakt

Cílem bakalářské práce je vytvoření funkčního modulu (rozhraní) pro automatizaci statistik výsledků soutěže Bobřík informatiky a následné implementování do stávajícího systému postaveném na CMS (Content Management System) Joomla. Vytvořený modul umožňuje v reálném čase provádět nastavitelné statistické výpočty z vybraného ročníku soutěže do tabulkového výpisu, interaktivních grafů a XLS výstupu; jednotlivé statistiky je možné zkoumat a vytvořit výroční zprávu soutěže. V úvodní části práce autor představuje soutěž Bobřík informatiky a následně popisuje použité technologie modulu – jazyky pro webové programování jako jsou PHP, MySQL a jejich implementace; na straně klienta jde o jazyk JavaScript a framework jQuery a použité pluginy. V další části se autor zabývá postupem vývoje jádra modulu a jeho jednotlivých částí, konfigurace a možnosti doprogramovávání dalších vlastních zkoumání nebo doplňků; dále se zabývá řešením možných problémů a implementace do stávajícího systému CMS.

Klíčová slova

Bobřík informatiky, automatizace výsledků, statistiky, vyhodnocení soutěže, PHP, MySQL

Abstract

The objective of this bachelor thesis is to develop the functional module (interface) for automatization statitics of results Bebras Informatics Contest and implementation to existing system built on CMS (Content Management System) Joomla. This module enables real-time configured statistics calculations of selected year contest to table view or interactive charts and XLS output; it is possible to examine the different statistics and creating of annual report of this contest. In the introduction of thesis autor presents the Bebras Informatics Contest and also describes used technologies of module – web programming languages as PHP, MySQL and their implementation; on client side is JavaScript language, framework jQuery and plugins which were used in. In the other part, author follows up with module development proccess and its parts, configuration and possibilities of programming a new custom examining or plugins; further analyses solving potential issues and implementation to existing CMS.

Keywords

Bebras Informatics Contest, automation results, statistics, evaluation of the contest, PHP, MySQL

Poděkování

Chtěl bych poděkovat panu doc. PaedDr. Jiřímu Vaníčkovi, Ph.D. za odborné vedení a poskytování cenných a užitečných rad při vytváření samotného modulu a poskytnutí veškerých materiálů týkajících se soutěže. Dále bych velice rád poděkoval panu Mgr. Václavu Šimandlovi za implementace modulu do produkčního webu soutěže a pomoc při nahrávání jednotlivých iterací.

Nemohu opomenout ani Janu Bartičkovou, za jejíž výraznou psychickou podporu také děkuji.

Obsah

1 Úvod

V důsledku neustálého rozvoje soutěže Bobřík informatiky již stávající řešení manuálního zpracovávání výsledků, po skončení aktuálního ročníku soutěže, nestačí. Zpracování výsledků je při současném řešení časově náročné a zkoumané statistiky jsou pouze základní. Bylo by výhodné zapojit do statistik i data z předchozích ročníků a tím možnosti statistik rozšířit.

Současné statistiky bychom mohli rozdělit podle subjektu zkoumání do kategorií Otázky, Školy a Soutěžící; z každé této kategorie vychází série výpočtů prezentována tabulkovými hodnotami nebo grafy. Zejména statistiky z kategorie Otázky jsou v současnosti vytvářeny postupnými výpočty v tabulkovém procesoru, důkladně překontrolovány, a především graficky prezentovány; tyto grafy jsou poté uloženy jako obrázky. Výroční zpráva je postupně tvořena v administraci webu – postaveném na CMS (Content Management System) Joomla – jako článek s vloženými nadpisy, popisy, tabulkovými výpočty a obrázkovými grafy a komentáři.

Zpracování výroční zprávy v novém modulu přinese její snadné vytváření bez použití tabulkových procesorů, pomocných výpočtů a jiných doplňků; implementováním interaktivních grafů dostávají statistiky nový rozměr – jednotlivé datové řady lze skrývat a zobrazovat a při najetí myší na jednotlivé body grafu okamžitě vidíme v popisku datové řady a jejich absolutní hodnoty s procentuálním zastoupením v daném bodě.

Sestava pro zprávu se bude skládat z jednotlivých bloků datových typů; mezi nejpoužívanější budou patřit: text (formátované textové pole pro vytvoření odstavce s možností vložení tzv. pole – zástupné značky s odkazem na hodnotu v databázi, která se po vytvoření zprávy hodnotou nahradí), datový objekt (blok reprezentující textové nebo grafické výpočty z databáze dle vybraných datových zdrojů a nastavených parametrů – každý blok může mít vlastní název a popis) a oddělovač. Samotné bloky bude možné přesouvat, odstraňovat a přidávat nové; každá tato sestava půjde uložit pro další použití a modifikace.

Vygenerovaná výsledná zpráva, kde jsou veškeré značky nahrazeny hodnotami z databáze, bude vytvořena jako HTML (Hypertext Markup Language) kód spo-

lečně s objekty JavaScriptu (sloužící pro konfiguraci grafů) a přiloženým CSS (Cascading Style Sheets) stylem implementována jako článek do Joomly.

1.1 Cíle bakalářské práce

Cílem bakalářské práce je vytvoření modulu (rozhraní) pro soutěž Bobřík informatiky. Modul bude implementován do stávajícího systému postaveném na CMS Joomla a následně využit k automatizovanému zpracování výsledků soutěže.

Samotné rozhraní umožňuje v reálném čase provádět nastavitelné statistické výpočty z vybraného ročníku soutěže s možností zobrazení do tabulkového výpisu, interaktivních grafů nebo XLS výstupu pro další možné zpracování. Statistiky bude možné zkoumat a následně vytvořit výroční zprávu soutěže.

V úvodní části bakalářské práce jsem zmínil metody vyhodnocování soutěže Bobřík informatiky a dále se budu zabývat použitými technologiemi, jako jsou jazyky pro webové programování PHP (PHP: Hypertext Preprocessor), MySQL, JavaScript a framework jQuery a implementované JS (JavaScript) pluginy.

Poté se budu zabývat popisem vývoje jádra celého modulu a jeho jednotlivých částí; následují způsoby možnosti konfigurace a doprogramovávání dalších vlastních statistik (z uživatelského a programátorského hlediska) nebo doplňků. Mezi další důležité oblasti bych zařadil řešení možných problémů a implementaci do stávajícího systému CMS.

Původní myšlenkou projektu bylo vytvořit pouze rozšiřující rozhraní do současného systému s použitím veškeré stávající konfigurace a doplňků. V průběhu realizace se tento způsob neosvědčil a rozhraní nepracovalo správně, popř. neukazovalo žádné výsledky. Modul je tedy stavěn tak, aby byl nezávislý a nebyl ovlivňován prostředím, kde se nachází; obsahuje proto vlastní konfigurace, připojení na databázi a doplňky.

Přidávání nových statistik a dalších možností rozšiřování nebude potřeba složitě deklarovat a implementovat, stačí se řídit podle stanovených pravidel a systém soubor sám vyhledá.

1.2 Metoda práce

Celý modul je napsán v prostém jazyce PHP bez využití frameworků a s připojením k stávající databázi MySQL. Při jeho návrhu jsem zvolil objektově orientovaný přístup programování. Pro komunikaci klientské části se serverem využívám technologii AJAX (Asynchronous JavaScript and XML) s datovou strukturou JSON (JavaScript Object Notation). Veškeré zmíněné technologie si podrobněji rozepíšeme v následující kapitole Použité technologie a zdůvodníme jejich použití.

Pro vývoj modulu považuji za nejvhodnější iterativní přístup; z počátku jsem se proto seznámil s požadavky na možnosti a funkce modulu a získal přístup k datům z předchozích ročníků – provedl jsem analýzu. V další fázi proběhl návrh celého postupu, zjištění možných problémů a jejich řešení a možnosti realizace.

Veškerá možná řešení jsem po zhodnocení implementoval do aktuální iterace modulu a v následující fázi otestoval jejich správnost a funkčnost. V případě výskytu chyb opakuji fázi implementace. Hotová iterace je pak odeslána na produkční server a očekávám připomínky a další požadavky.

Popsané fáze probíhají opakovaně, dokud není modul zcela hotový a splňuje dané očekávání.

2 Použité technologie

V aktuální kapitole si popíšeme veškeré použité technologie počínaje webovými skriptovacími jazyky přes frameworky až k využitým doplňkům. Budeme se zabývat jejich stručnou historií, postupem vývoje, použitými licencemi a výhodami a nevýhodami. Výběr použitých technologií byl ovlivněn nutností navázat na již existující systém, který využívá webový server Apache a jazyky PHP 5 a MySQL. Mimo to je důležité, aby modul byl přístupný k dalším zásahům a doplňkům.

2.1 Apache

Apache je nejpopulárnější a nejrozšířenější volně šiřitelný web server. Vznikl v roce 1995 jako volně šiřitelná modifikace původního NCSA HTTPd serveru firmy NCSA (National Center for Supercomputing Applications). Název Apache vychází z pracovního názvu „a patchy server“, kdy programátoři tvořili bezpečnostní záplaty pro stávající webový server; jeho modulární řešení umožňuje další rozšiřování o nové možnosti. [1]

Distribuce, kromě samotného **A**pache, vždy obsahuje databázový server **M**ySQL (někdy sebou také přináší PL/pgSQL) a preprocesor **P**HP – zkratka **AMP** je základem pro názvy jednotlivých distribucí a z něho jsou tedy vždy patrné součásti distribuce. Pro různé systémové platformy se můžeme setkat s distribucemi: Multiplatformní (XAMPP), Linux (LAMP), Windows (WAMP), Mac (MAMP). Dalšími součástmi distribuce může být také FTP (File Transfer Protocol) server nebo interpreter jazyka Perl.

2.2 OOP

„Moderní programovací jazyky obvykle při vývoji softwaru podporují nebo dokonce vyžadují objektově orientovaný přístup. Objektově orientované programování (OOP) se snaží vývoj programů a opakované použití kódu usnadnit vymezením objektů, kterými je modelovaný systém tvořen, a definováním jejich vlastností a vzájemných vztahů.“ [2]

Základním prvkem OOP jsou tzv. *třídy*, které slouží jako „šablona“ pro vytvoření objektu; obsahují vlastní proměnné a vlastní funkce. Každý nový objekt

vytvořený ze třídy se nazývá *instance třídy*; veškeré uložené proměnné v objektu se nazývají *vlastnosti objektu* a funkce v těle objektu jsou známé jako *metody*. [3]

Velmi důležitou vlastností OOP je tzv. *dědění*, pomocí něhož lze vytvořit hierarchické vazby tříd a podtříd. *Podtřída (potomek, dítě)* dědí vlastnosti a metody od svého předka, *nadtřídy (rodiče)*. OOP klade důraz na znovupoužitelnost a jednotlivé problémy jsou rozdrobeny do menších celků, které jsou přehledné a snadněji rozšiřitelné než strukturované programování. Díky dědičnosti je možné dále rozvíjet již hotové třídy a kód je snadněji znovupoužitelný.

2.3 PHP 5

PHP je skriptovací jazyk určený pro tvorbu jak jednoduchých webových stránek, tak zároveň velkých webových projektů nebo aplikací. Podle aktuálních informací z W3C Techs (World Wide Web Consortium) z URL http://w3techs.com je celosvětové využití PHP na serverech 81,9 %. Využívá se především díky své jednoduchosti a syntaxi obdobné jazyku C a většině programátorů je tak velmi blízký.

„Počátky vývoje PHP sahají do roku 1994, kdy Rasmus Lerdorf vytvořil jednoduchý systém evidence přístupů ke svému webu, nejdříve v jazyce Perl, poté v jazyce C. Ten se rozšířil mezi další uživatele, kteří přicházeli s požadavky na vylepšení. Vznikl tak systém Personal Home Page Tools, později Personal Home Page Construction Kit.“ [4]

Rasmus Lerdorf v roce 1995 propojil PHP se svým dalším systémem FI (Form Interpreter), který umožňoval zpracovávat data z odeslaného formuláře z webu a obsahoval řadu funkcí, které se používají dodnes. Tak vzniklo PHP/FI 2.0. Po roce 1998, kdy je již na světě verze 3.0, o poznání rychlejší a výkonnější, se rychle rozšiřuje. PHP 4 přináší v roce 2000 nové možnosti, jako jsou například Sessions (identifikace klienta na serveru v rámci relace), Output Buffering (veškeré výstupy ze skriptu nezasílá okamžitě klientovi, ale ukládá do zásobníku, jehož obsah může být kdykoliv odeslán), vyšší zabezpečení a náznaky OOP – většina funkcí z této verze je použitelných i dnes. Až v PHP 5 (vydaném v roce 2005) se PHP přibližuje ostatním jazykům novým pojetím OOP, snazší je také obsluha chyb a výjimek [4][5][6].

Mezi výhody tohoto jazyka patří zejména to, že je multiplatformní, jednodu-

chost syntaxe, není nutná kompilace zdrojového kódu – pouze jeho interpretace, propojení s Apachem a databázemi MySQL, PgSQL (PostgreSQL), široká podpora u poskytovatelů webhostingů.

Nevýhodou je pomalý vývoj oproti ostatním jazykům; některé základní zásuvné moduly (např. modul ZIP při komprimaci souborů větších velikostí) mají velké paměťové nároky a následně je skript ukončen s chybou – v určitých případech je místo modulů výhodnější využívat programy systému nebo PHP kombinovat s ostatními jazyky, především s Pythonem.

2.4 Relační databáze, MySQL

RDBMS (Relation DataBase Management System) V češtině je znám jako *Systém řízení báze dat* (SŘBZ) neboli *databázový stroj*. Jde o vrstvu mezi uloženými daty a koncovou aplikací, která jej ovládá pomocí jazyka *SQL* (Structured Query Language). Úkolem databázového stroje je manipulace s velkým objemem dat (vyčítání, ukládání a modifikace) a zpracovávání požadavků více uživatelů. Vlastnosti databázového stroje můžeme shrnout do zkratky ACID:

- *Nedělitelnost (Atomicity)* – při použití operací v rámci transakce se veškeré tyto operace chovají jako jedna, jsou atomické (nedělitelné) a musí se tedy vykonat společně,
- *Konzistence (Consistency)* – databáze je vždy po dokončení transakce konzistentní, při chybě jedné akce se provedené změny v rámci transakce navracejí do původního stavu,
- *Izolace (Isolation)* – požadavky od více uživatelů jsou zpracovány izolovaně (neovlivňují se navzájem), více požadavků současně se řadí do fronty a jsou vykonávány postupně,
- *Ochrana proti ztrátě dat (Durability)* – veškerá data jsou okamžitě ukládána na datová úložiště (pevný disk), aby se v případě výpadku žádná data neztratila. [7]

Relační databáze Základem relačních databází jsou *databázové tabulky* tzv. *entity*, složené z *řádků* (*záznamů*) a *sloupců* (*atributů*), obsahující stejný typ položek (osoby, místa, kategorie, ...). Jednotlivé položky jsou ukládány do řádků a každý sloupec má vlastní datový typ (číslo, text, datum, ...). Každý řádek by měl mít unikátní identifikátor v rámci celé tabulky, podle kterého bude možné daný záznam jednoznačně určit, jedná se o tzv. *primární klíč*. [2][7]

Název *relační* se odvíjí od vazeb (relací) mezi jednotlivými entitami. Rozlišujeme 3 druhy relací:

- **1:1** – k jednomu řádku tabulky A náleží právě jeden řádek z tabulky B,
- **1:N** – k jednomu řádku tabulky A je navázáno N řádků z tabulky B; v tomto případě je nutné založit tzv. *spojovací tabulku* obsahující ID řádků obou tabulek,
- **M:N** – M řádkům z tabulky A náleží N řádkům v tabulce B; využíváme opět spojovací tabulku.

MySQL Je relační multiplatformní databáze, velmi oblíbená a často používaná na webových serverech v kombinaci s PHP, kde bývá jeho součástí. Databáze je optimalizována pro rychlost a vysoký výkon. Mezi nevýhody bychom mohli zařadit neschopnost zvládat složitější konstrukce, narozdíl od ostatních placených databázových systémů.

K využití databáze MySQL v modulu jsem se rozhodl podle stávajícího systému. Tato databáze je již v systému použita, a proto je vhodné ji využívat i nadále.

Existuje verze zdarma pro domácí použití a verze placená. Verze pro nekomerční použití využívá GPL licenci (GNU General Public License).

Jako zajímavost bych uvedl, že označení *My* v názvu databázového systému je vlastně jméno dcery spoluzakladatele Montyho Wideniuse. [8]

2.5 HTML5

HTML je značkovací jazyk, inspirovaný jazykem *SGML* (Standard Generalized Markup Language), pro vytváření webových stránek. Vytvořil jej Tim Berners-Lee v roce 1990 pro zobrazování hypertextových stránek v interní síti CERNu. Prvotní

verze byla velmi jednoduchá a s nástupem internetu jako masově využívaného přístupu k informacím se HTML postupně více rozvíjelo.

Zatím poslední verzí je v současnosti využívané *HTML5* finálně specifikované v roce 2014. Přináší nové možnosti z pohledu designu, rozvržení stránky a použitelnosti. Kreslení grafiky pomocí objektu *canvas* a integrace přehrávače videa a audia do prohlížeče nahrazují zastaralé používání pluginů (Flash a Java Applet), které bylo nutné instalovat a někdy velmi zpomalovaly webový prohlížeč. Nové tagy, jako jsou *section*, *article*, *header*, *footer*, *nav* a další přináší lepší orientaci a snazší pochopení struktury webové stránky, především pro roboty (např. vyhledávače). Nových možností je mnoho, mezi zajímavé bych zařadil: zjišťování GPS polohy uživatele pomocí objektu *navigator.geolocation*, práci skriptů na pozadí (zajišťována objektem *Worker*) a lokální úložiště, kde oproti cookies můžeme uložit více dat a nastavit možnosti jejich expirace. [3][9]

Práce s HTML5 je vhodná především z důvodu jeho grafických možností pro využití v interaktivních grafech.

2.6 JavaScript a jQuery

Javascript Programovací jazyk vyvinula v roce 1995 společnost Netscape Communications pro svůj webový prohlížeč Netscape; samotný jazyk byl napsán poměrně rychle, a to za necelé 3 týdny.

Název JavaScript má s jazykem Java pouze minimum společného, výběr názvu byl čistě obchodní tah, normovaným názvem je *ECMAscript* normalizační společnosti *ECMA* (European Computer Manufacturers Association).

Jedná se tedy o multiplatformní a objektově orientovaný skriptovací jazyk, který dokáže „oživit“ HTML stránku. Samotný skript je implementován do HTML stránky do tagu `<script>` a interpretuje se na straně klienta. Velkou nevýhodou tohoto jazyka vždy byla jeho nekompatibilita s některými webovými prohlížeči, ale v současnosti již rozdíly v podpoře prohlížečů nejsou tak vysoké.

Velké výhody také přinášejí JS frameworky, které nabízejí nové možnosti jazyka z hlediska použitelnosti a minimalizace kódu, nezapomenu také zmínit, že samotný framework zajišťuje kompatibilitu s prohlížeči. [10]

jQuery Bohatá knihovna vytváří z prostého Javascriptu „nový“ moderní, jednotný a objektový jazyk. Slovo jednotný v tomto případě znamená, že programátor při psaní nemusí rozlišovat druhy prohlížečů na straně klienta a vytvářet pro každý jiné metody. Autor John Resig hotový produkt předvedl v roce 2006 a ten postupně získával na popularitě. V současnosti, jak je uvedeno na http://w3techs.com, je jQuery celosvětově využíváno na 69,6 % webových stránek. U programátorů je velmi oblíbený především díky vlastním metodám chybějícím v samotném JS, práci s objekty a způsoby AJAX komunikace; přístupy k prvkům HTML a DOM (Document Object Model) jsou elegantně řešeny pomocí CSS selektorů a tím je použití velmi snadné; nabízí také práci s efekty a animacemi na profesionální úrovni. Knihovna je skvělý základ pro další rozšíření; licence: MIT (Massachusetts Institute of Technology). [3]

V modulu využívám jQuery především kvůli minimalizaci kódu oproti prostému JavaScriptu a kompatibilitě s webovými prohlížeči.

2.7 AJAX, JSON, YAML

AJAX Komunikace webové stránky na straně klienta se serverem při načtené stránce probíhala pomocí – dnes již zastaralého – tagu `<iframe>`. Jednalo se o vložení stránky do stránky, kde vnitřní stránka umožňovala komunikaci se serverem tím způsobem, že se do ní načetla požadovaná URL.

Novější metoda AJAX, jak jej v roce 2005 nazval Jess James Garrett, je zkratka *Asynchronous JavaScript and XML*; jedná se tedy o soubor JavaScriptových metod umožňujících komunikaci *asynchronně* (na pozadí) mezi webovým prohlížečem klienta a serverem, bez nutnosti obnovení stránky.

Základním objektem AJAXu je *XMLHttpRequest*, skládající se z *požadavku* a *odpovědi*. Komunikace probíhá pomocí HTTP metod *GET* nebo *POST* a data jsou zasílána ve formě textu.

Jednotlivé fáze mezi odesláním požadavku a přijetím odpovědi popisuje vlastnost *readyState*, popišme si tedy jednotlivé stavy:

- 0 = `UNINITIALIZED` – objekt *XmlHttpRequest* byl vytvořen a čeká na zavolání metody `open()`,

- 1 = `LOADING` – spojení je navázáno pomocí metody `open()`, čeká se na odeslaní požadavku metodou `send()`,
- 2 = `LOADED` – požadavek odeslán metodou `send()`, z odpovědi jsou přijaty zatím jen HTTP hlavičky bez těla,
- 3 = `INTERACTIVE` – data těla odpovědi jsou postupně načítána do vlastnosti *responseText*, zatím však nejsou ještě kompletní,
- 4 = `COMPLETED` – odpověd je již kompletní a všechny operace dokončeny. [11][12]

Po přijetí odpovědi je ve vlastnosti *status* uložen číselný kód stavu zpracování požadavku serverem (např. 404), *statusText* obsahuje tento kód popsaný slovně (např. Not Found). Vlastnost *responseText* obsahuje přijatá data, která lze dále zpracovat.

Použití AJAX technologie výrazně zmenšuje množství přenášených dat a aplikace na straně klienta je dynamičtější a soběstačnější. [2]

JSON JSON (JavaScript Object Notation) je datová struktura ve formě prostého textu pro výměnu dat. Struktura je univerzální a lze ji snadno zpracovávat strojově a současně je dobře čitelná i pro člověka. Vychází z jazyka JavaScript a využívá pravidla C-rodiny programovacích jazyků (C, C++, C#, Java, Perl a dalších). [13]
Datovou strukturu můžeme využít jako:

- *seznam* – pole hodnot různých datových typů oddělených čárkou,
- *objekt* – pár *klíč-hodnota*, kde klíč může být řetězcem a pomocí klíče lze přistupovat k danému prvku.

YAML Při ukládání strukturovaných dat je nejrychlejší ukládat je jako binární soubor. Zároveň tak zaberou nejméně místa v paměti. V případě, že potřebujeme data číst i bez aplikace, nejlépe aby je uživatel mohl zpracovávat běžným textovým editorem, můžeme použít formáty složitější – to jsou například *XML* (Extensible

Markup Language), *JSON* nebo jednoduché *INI* (Inicialization). Velikosti souborů jsou pak větší než u souborů binárních.

Formát YAML přináší jednoduchost syntaxe z INI a možnosti vytvářet i složitější konstrukce v prostém textu z POX (Plain Old XML) – zjednodušená forma XML bez jmenných prostorů. Velmi často se používá jako konfigurační soubor u webových portálů.

Základními prvky YAML jsou seznamy údajů, tzv. *sekvence*, které se při načtení do PHP programu chovají jako indexované pole. V případě asociativního pole se využívá pár *klíč-hodnota* pomocí tzv. *mapy*; *dokumentem* se nazývá více logických celků oddělených třemi spojovníky. Všechny zmíněné struktury můžeme kombinovat a vytvářet tak struktury složitější, jako jsou *složená data* a *sekvence sekvencí*. [14]

– PHP –

```php
array (
  'name' => 'Block',
  'type' => 'object',
  'position' =>
  array (
    'left' => 10,
    'top' => 5,
  ),
  'values' =>
  array (
    0 => 5,
    1 => 10,
    2 => 15,
  ),
);
```

– JSON –

```json
{
    "name": "Block",
    "type": "object",
    "position": {
        "left": 10,
        "top": 5
    },
    "values": [
        5,
        10,
        15
    ]
}
```

– YAML –

```yaml
name: Block
type: object
position:
 left: 10
 top: 5
values:
 - 5
 - 10
 - 15
```

Obrázek 1: Srovnání datových struktur: PHP pole, JSON a YAML

2.8 Využité komponenty

2.8.1 Highcharts a Highmaps

Highcharts je knihovna grafů, napsaná v prostém JavaScriptu, která nabízí možnost zobrazení interaktivních grafů do webové aplikace pomocí vstupních dat zadaných ve formátu JSON. Obsahuje velké množství typů grafů a společně se sesterskou komponentou Highmaps dokáže vytvářet i interaktivní mapy. Využívá nativní technologii prohlížeče, na straně klienta tedy nevyžaduje zásuvné moduly jako Flash. Díky technologiím HTML5 a SVG (Scalable Vector Graphics) fun-

guje na veškerých moderních webových prohlížečích a to včetně mobilních. Pro starší prohlížeče (Internet Explorer 6, 7 a 8), nepodporující SVG, jsou grafy vykreslovány technologií VML (Vector Markup Language). Jednotlivé datové řady můžeme skrýt i zobrazit a v každém bodě lze ukázat popisek s informacemi pro každou datovou řadu s možností zobrazení procentuálního zastoupení vůči ostatním řadám. Část grafu je možné také zobrazit detailněji přiblížením grafu. Každý graf lze vyexportovat do bitmapových formátů PNG (Portable Network Graphics), JPEG (Joint Photographic Experts Group) a vektorových SVG a PDF (Portable Document Format). Exportování probíhá vždy na straně klienta. [15]

Výhodou je jejich dynamika a interaktivita, spolupráce obou komponent a kompatibilita i s prohlížeči nepodporujícími HTML5. Komponentu využívám pod licencí Creative Commons Attribution-NonCommercial 3.0 License.

2.8.2 PHPExcel

PHPExcel z balíku PHPOffice je rozsáhlá knihovna, založená v roce 2006, umožňující pomocí objektově orientovaného přístupu vytvářet formáty Excel 2007, Excel 97-2003, CSV (Comma-separated values), HTML a PDF. Vyžaduje minimální verzi PHP 5.2 a zapnuté rozšíření php_zip.dll. [16]

Knihovnu jsem zvolil proto, že je dle mého názoru ve své třídě nejlepší, mám s ní dlouholeté a dobré zkušenosti. Nabízí široké spektrum možností jako samotný Excel, umožňuje efektivní práci s daty a následné exporty do různých formátů. Lze s ní např. vytvářet PDF formulář mnohem lépe než se standardními *TCPDF* a *mPDF* knihovnami. PHPExcel je vydáván pod licencí LGPL (GNU Lesser General Public License), ke stažení je na oficiálních stránkách `https://phpexcel.codeplex.com/`.

2.8.3 Editor „PEN“

Tato moderní komponenta umožňuje editaci textu přímo ve webové stránce, úpravy probíhají formou běžného *WYSIWYG* (What You See Is What You Get) editoru; autor: Sofish, verze: 0.0.1, licence: MIT, URL: `https://github.com/sofish/pen`.

Editor je jednoduchý na ovládání, snadno integrovatelný a obsahuje plovoucí skrývací panel nástrojů, díky čemuž se do modulu skvěle hodí.

WYSIWYG Je textový editor, kde na obrazovce vidíme finální verzi formátovaného textu pro naformátování – uživatel tedy nemusí pro formátování textu psát speciální značky, ale jen označí část textu a přiřadí formát. Pro představu – Office Word a OpenOffice Writer jsou typickými představiteli WYSIWYG editorů. Ve webovém prostředí však nejsou tyto editory běžným standardem, stále se formátuje pomocí značkovacího jazyka HTML nebo pomocí vlastních formátovacích značek (např. phpBB) – není to však z důvodů pomalého vývoje webů, ale mnoho z těchto editorů přidává zbytečné a nevyužitelné značky a velikost dokumentu tak bývá znatelně vyšší.

2.8.4 jQuery Simple Splitter

Komponenta umožňující rozdělit blok (v mém případě okno prohlížeče) na jednotlivé části a mezi nimi vytvořit posuvník umožňující snadné zvětšování (popř. zmenšování) dané části; vytvořena v roce 2015, autor: Shunji Konishi, verze: 1.0.1, licence: MIT, URL: `https://github.com/shunjikonishi/jquery-splitter`.

2.8.5 jQuery Simple Color Picker

Tento plugin byl naprogramovaný v jQuery Tanguyem Krotoffem v roce 2012 pod MIT licencí. Jedná se o velmi jednoduchou, ale plnohodnotnou a přehlednou paletu barev. Jednotlivé barvy se vytvářejí z pomocného `<select>` tagu, který se následně skryje. Doplněk výborně spolupracuje také s Bootstrapem; stáhnout jej můžeme na GitHub: `https://github.com/tkrotoff/jquery-simplecolorpicker`.

Je na něm sympatická jednoduchost při integraci do webu a snadné ovládání pro uživatele.

2.8.6 Bootstrap Tooltip

Bootstrap je nejpopulárnější HTML, CSS a JS framework zahrnující prostředky pro designování a typografii webu. V moderní době mobilních zařízení je tato knihovna ideální zejména pro vytvoření responzivního webu (zobrazení přizpůsobené na velikost zobrazovacího zařízení).

V modulu je využita pouze komponenta *Tooltip* – bublinková nápověda pro snazší orientaci uživatele a předání navigačních informací. Autor: Twitter, verze:

3.3.5, licence: MIT, URL: `http://getbootstrap.com/`.

2.8.7 Font Awesome

Velmi oblíbená moderní knihovna vektorových ikonek pro webové stránky. Načítají se pouze dva soubory: *knihovna fontů* (ikonky) a *CSS styl* – stránky se zbytečně nezpožďují načítáním jednotlivých obrázků a ikonek na stránce. Způsob použití fontu umožňuje ikonky obarvit a transformovat bez ztráty kvality a deformace. Autor: Dave Gandy, verze: 4.5.0, licence: MIT, URL: `https://github.com/FortAwesome/Font-Awesome`.

2.8.8 Highlight.js

Komponenta napsaná v prostém JavaScriptu byla navržena v roce 2006 ruským programátorem Ivanen Sagalaevem. Dokáže zformátovat a barevně odlišit syntaxi zdrojového kódu zobrazeného na webové stránce, což výrazně usnadní orientaci v něm. Zahrnuje 150 syntaxí programovacích jazyků běžně používaných i méně známých. Druhy použitých jazyků jsou automaticky detekovány, ale lze je i nastavit ručně. Součástí knihovny je také 71 stylů ovlivňujících i zvýraznění syntaxe. Aktuální použitá verze 9.2, licence BSD3 (Berkeley Software Distribution verze 3). [17]

Doplněk můžeme stáhnout na oficiálních stránkách: `https://highlightjs.org/`.

3 Návrh modulu

V této kapitole sestavíme ze získaných požadavků podrobný popis nového modulu. Vytvoříme si náhled celého systému; popíšeme si, jakým způsobem by měl být tvořen a jaké schopnosti a funkcionality musí obsahovat.

3.1 Požadavky na modul

Nejprve si připomeňme základní požadavky a jaké funkčnosti by měl mít nový modul:

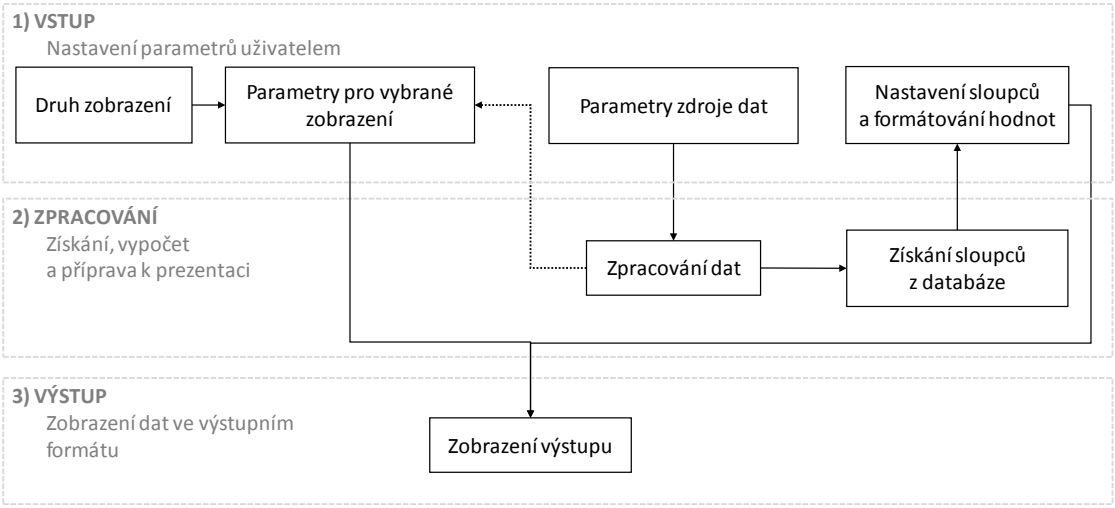
- zpracovat současné výpočty automatizovaně,
- u výpočtů bude možnost nastavit určité parametry,
- zobrazovat hodnoty tabulkově nebo graficky podle druhu grafu a použití,
- možnost exportovat hodnoty do Excelu (XLS),
- jednoduchá obsluha modulu při generování závěrečných zpráv a jejich exportu,
- určování, zda přiřazená obtížnost otázky odpovídá,
- vymezení otázek, které rozhodují o úspěšnosti jednotlivých řešitelů.

3.2 Struktura návrhu

3.2.1 Návrh zpracování dat

Mojí prvotní myšlenkou bylo implementovat modul do současného systému pouze jako rozšiřující modul pro počítání statistik. Postupem času při realizaci se však ukázalo, že modul musí být osamostatněn od systému – nemůže na něm být závislý. Ze systému využívá modul pouze konfigurační soubor s přístupy k databázi.

Na obrázku 2 si vysvětlíme, jakým způsobem bude generování dat probíhat.



Obrázek 2: Diagram návrhu zpracování a zobrazení dat se vstupními parametry

Na vstupu zvolíme parametry pro získání dat *(Parametry zdroje dat)*. Můžeme nastavit, jakým způsobem budou data prezentována *(Druh zobrazení)*; vybrané zobrazení nám ovlivní další vizuální parametry výstupu (např. pokud vybereme zobrazení jako graf, budeme moci nastavit typ grafu).

Při generování dat mohou být parametry výstupního zobrazení ovlivněny vybranými statistikami (záleží na konfiguraci každé třídy viz. Třída StatsData). Po zpracování dat může uživatel nastavit zobrazení nebo skrytí sloupců, jejich názvy, pořadí a u číselných hodnot jejich formáty.

Celý návrh modulu vychází z publikovaných výročních zpráv; má úvaha o celkovém zapracování výstupů do uživatelsky přívětivé formy vychází z přehledu témat systému Moodle, skládajících se z jednotlivých bloků řazených pod sebou. Každý blok je určitého datového typu, tedy má přednastavené vlastnosti, a jeho reprezentace je vždy odlišná.

3.2.2 Návrh bloků

Jako základní prvky zprávy jsem tedy zvolil bloky určitých datových typů jako jsou *textová pole*, *datové objekty*, *panely*, *oddělovače* a *obrázky*. Bloky je možné přemisťovat, skrývat a samozřejmě i odstraňovat. Vybráním bloku se zobrazí panel s jeho nastavitelnými vlastnostmi.

Textové pole Text s možností vkládání odstavců a formátovaného textu; tento blok jako jediný spolupracuje s proměnnou *Pole* (umožňuje vkládat do textu agregované hodnoty ze statistik viz. Pole).

Datový objekt Blok, který dokáže prezentovat výstupní data statistik tabulkovými hodnotami, grafy a mapou.

Panel Rozložením se jedná o tabulku o *r řádcích* a *s sloupcích*, kde do každé buňky můžeme přesunout bloky z šablony.

Oddělovač Slouží pouze pro optické oddělení bloků a lepší orientaci na vygenerované stránce.

Obrázek Umožňuje vložení obrázku pomocí zadané URL adresy.

Pole Do textu je nutné vkládat agregované hodnoty z databáze. Nabízí se tedy jediný způsob – vložení zástupného znaku a při exportu jej nahradit výslednou hodnotou. Inspiroval jsem se aplikací Microsoft Office Word, která právě *Pole* obsahuje a vlastní syntaxe tedy vypadá takto: `{{HODNOTA_Z_DATABÁZE}}`.

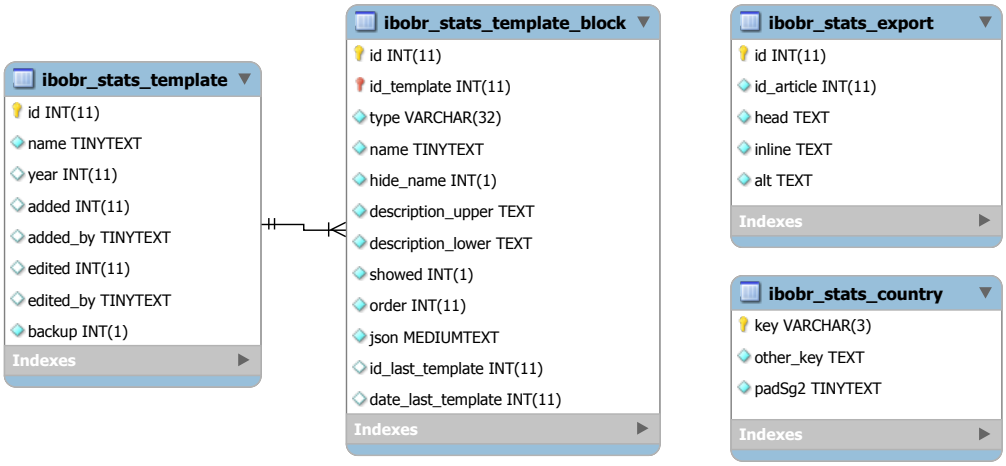
3.3 Struktura databáze

3.3.1 Databáze modulu

Modul bude integrován do stávající MySQL databáze systému. Vytvořím celkem 4 databázové tabulky:

- *ibobr_stats_template* – tabulka šablony (sestavy) jednotlivých datových bloků; umožňuje mít vlastní název a ukládá ročník, pro který je šablona tvořena,
- *ibobr_stats_template_block* – data jednotlivých bloků jsou uložena zde; jedná se zejména o datový typ, jméno, popis, pořadí a další data rozšířená pomocí datové struktury JSON,
- *ibobr_stats_export* – pomocná tabulka pro načítání JS skriptů a CSS při výpisu publikované stránky,

- *ibobr_stats_country* – seznam veškerých zemí pro výčet v otázce *Původ otázek* z třídy *Otázky*, kromě samotného klíče (identifikátoru dané země); obsahuje možné identifikátory a název země ve 2. pádu jednotného čísla.



Obrázek 3: Struktury a vazby databázových tabulek modulu

3.3.2 Databáze stávajícího systému

Každý ročník soutěže obsahuje pokaždé stejný soubor tabulek, nejdůležitější si rozepíšeme níže. V názvu tabulky je vždy obsaženo *číslo příslušného roku*, v případě aktuálního (posledního) ročníku je v názvu obsažena zkratka *akt*. V práci budu používat univerzální zápis s proměnnou *ROK* (např. *ibobr_ROK_kategorie*). Pokračujme tedy s tabulkami pro aktuální ročník.

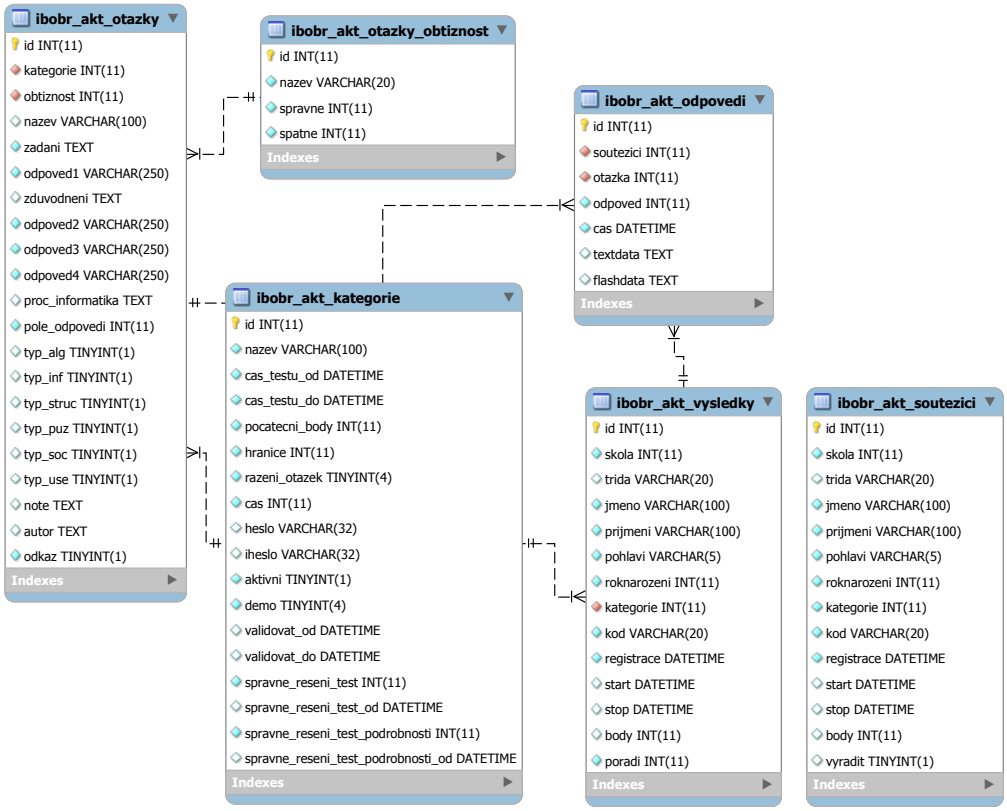
- *ibobr_akt_kategorie* – obsahuje název, čas začátku a konce soutěžení v dané kategorii, minimální a maximální bodové ohodnocení a časový limit pro dokončení soutěže,

- *ibobr_akt_otazky_obtiznost* – definuje jednotlivé obtížnosti s hodnotami bodů za správné nebo nesprávné odpovědi,

- *ibobr_akt_otazky* – dané otázky se řadí pod kategorii a obtížnost; tabulka obsahuje název se zadáním a textem otázky, kde každá otázka má 4 odpovědi, z nichž vždy ve sloupci *odpoved1* je správná odpověď; ve sloupci *note* je

většinou uložen kód (identifikátor) otázky, který je unikátní v rámci soutěže na celém světě – využívám jej pro zjišťování původu otázek,

- *ibobr_akt_odpovedi* – jednotlivé odpovědi každého soutěžícího na jednotlivé otázky s časovou značkou,
- *ibobr_akt_vysledky* a *ibobr_akt_soutezici* – obdobné tabulky s výsledky soutěže pro každého soutěžícího včetně jeho osobních dat a zařazení do kategorie; *ibobr_akt_vysledky* obsahuje oficiální výsledky s pořadím, *ibobr_akt_soutezici* obsahuje data všech účastníků soutěže.



Obrázek 4: Struktury a vazby důležitých databázových tabulek systému

3.4 Rozbor statistik

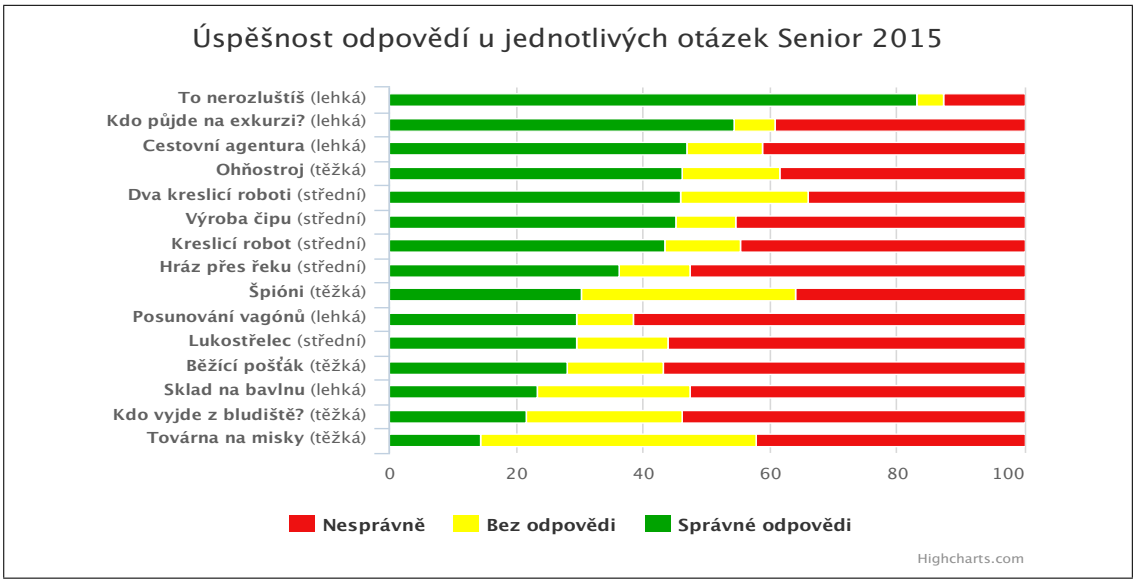
Doposud se řešení statistik provádělo výpočty v Excelu ze zdrojových dat stažených z databáze; veškeré výpočty bychom mohli rozdělit podle subjektu do kategorií

Otázky, *Soutěžící* a *Školy*. Podívejme se tedy na přehled jednotlivých statistik a způsob výpočtu. U výpočtů, které jsou primárně prezentovány grafem, je vložena ukázka grafu.

3.4.1 Otázky

Úspěšnost odpovědí u jednotlivých otázek (dle kategorie) Zkoumáme počet úspěšných, neúspěšných a nezodpovězených odpovědí. Podle procentuálního zastoupení správných a nesprávných odpovědí pak můžeme vyvodit, zda byla otázka zařazena do správné kategorie obtížnosti.

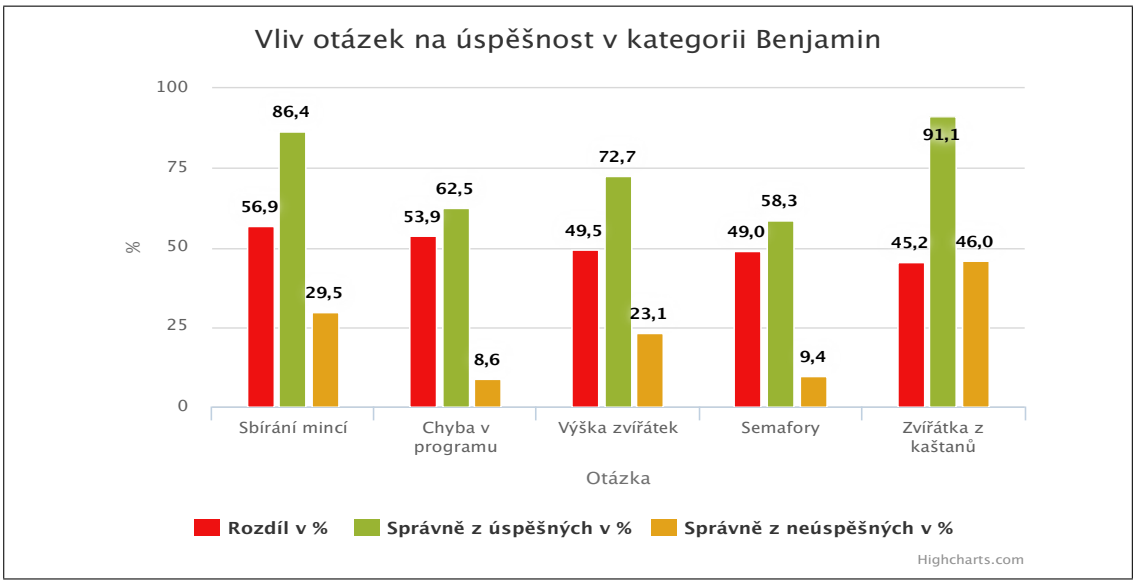
Pracujeme s databázovou tabulkou *ibobr_ROK_odpovedi*, kde $odpoved = 1$ je správná odpověď a $2 \leq odpoved \leq 4$ znamenají špatně zodpovězenou otázku. Pokud soutěžící otázku přeskočil, odmítl odpovědět nebo ji nestihl dokončit, platí výraz $odpoved = 5$.



Obrázek 5: Statistika: Úspěšnost odpovědí u jednotlivých otázek

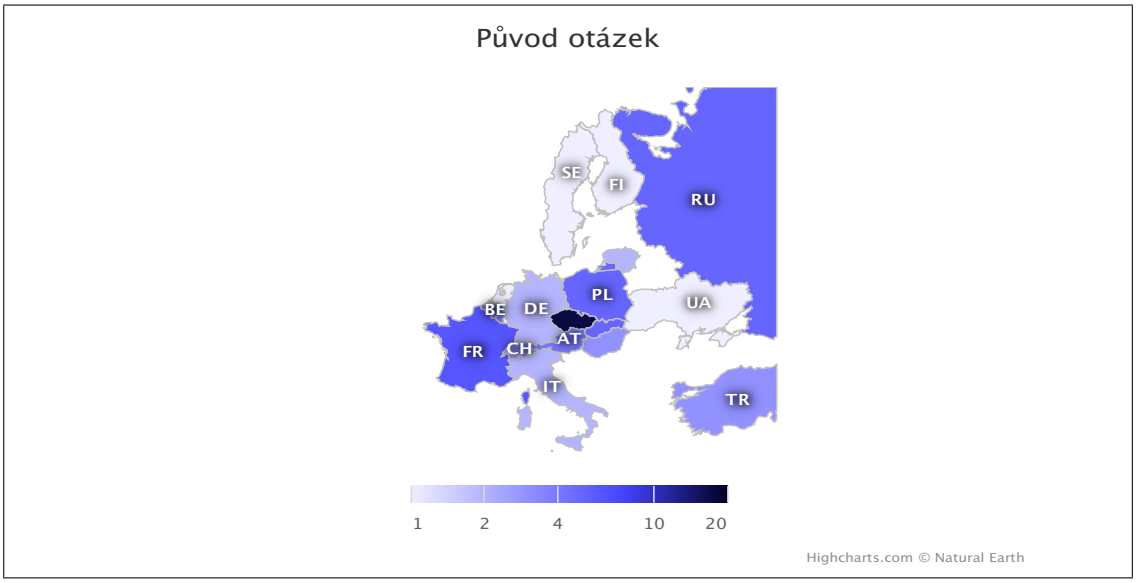
„Úroveň obtížnosti navrhuje tvůrce příkladu. Zadavatelé jsou při tom z různých zemí, proto předpokládají různé znalosti u dětí stejného věku. Při výběru úloh do soutěže se obtížnosti přehodnocují, organizátoři jsou však ovlivněni předchozím hodnocením tvůrce úlohy.“ [18]

Vliv otázek na úspěšnost Výsledek zkoumání ukazuje, které otázky nejlépe dokázaly rozdělit podle soutěžících na úspěšné a neúspěšné řešitele. Pro vybranou kategorii se zobrazí tři výsledky: počet soutěžících, kteří uspěli zároveň v soutěži i u této otázky, počet soutěžících, kteří neuspěli v soutěži ale správně zodpověděli otázku a rozdíl těchto hodnot, podle kterého je graf seřazen. Vysoký rozdíl pak ukazuje na to, že daná otázka byla jednou z rozhodujících pro úspěch v soutěži.



Obrázek 6: Statistika: Vliv otázek na úspěšnost

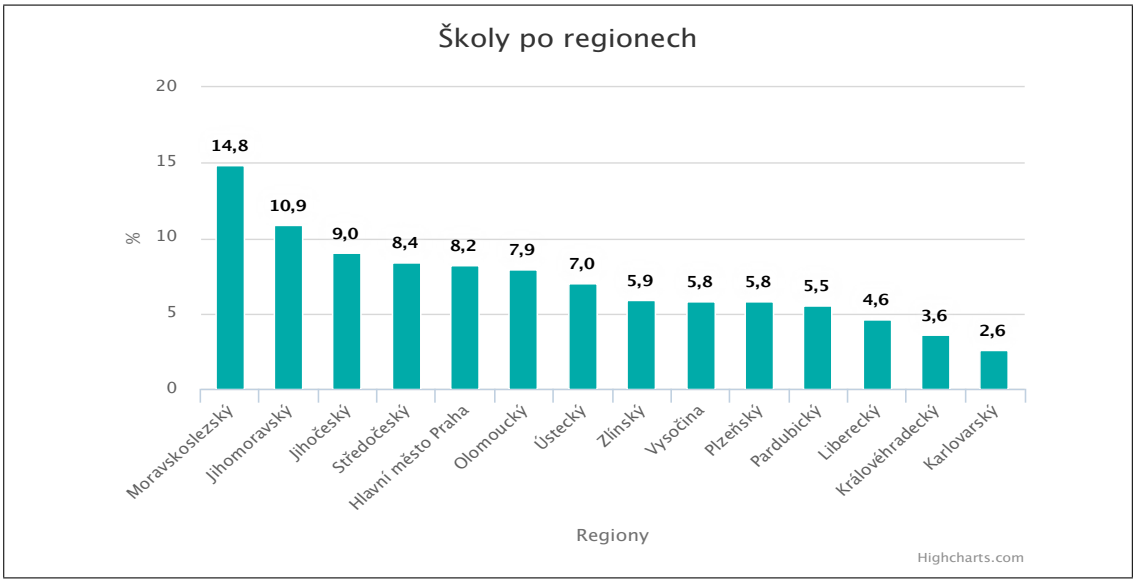
Původ otázek Statistika vytvoří souhrny, kolik otázek z různých zemí bylo v soutěži použito, podle unikátního kódu každé otázky. Kód je uložen v databázové tabulce *ibobr_ROK_otazky* ve sloupci *note* (např. 2015-CZ-05), ze kterého se extrahuje kód země a porovnává se se seznamem zemí uloženém v databázi.



Obrázek 7: Statistika: Mapa původu otázek

3.4.2 Školy

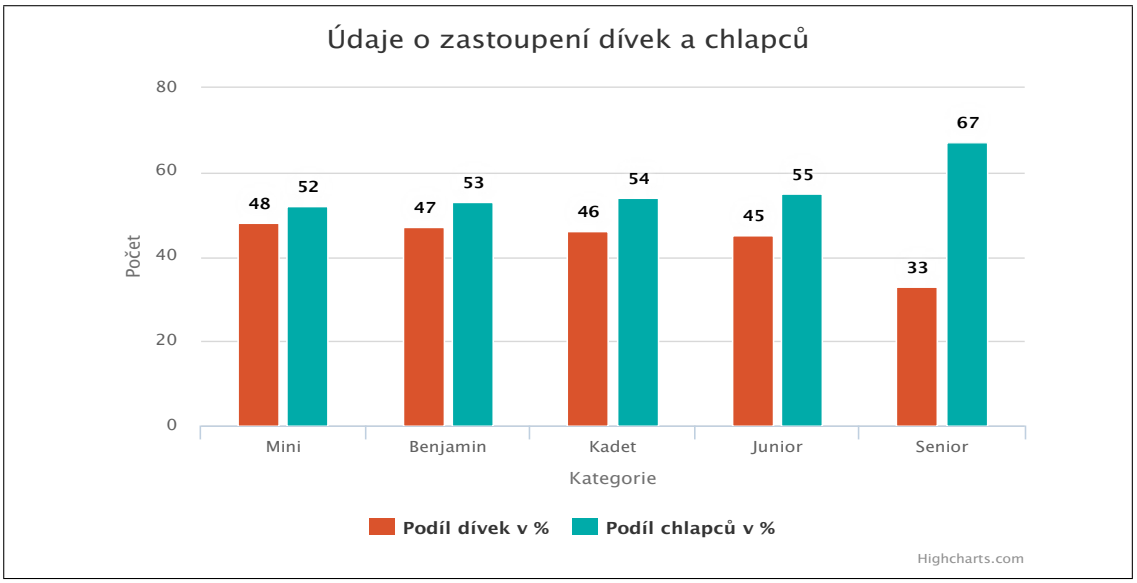
Účast škol po regionech Graf zobrazuje účast jednotlivých škol v soutěži. Můžeme vidět, kolik procent z celkového počtu zúčastněných škol je ze kterého kraje. Výpočet se provádí z počtu jednotlivých škol v krajích, do kterých spadají.



Obrázek 8: Statistika: Účast škol po regionech

3.4.3 Soutěžící

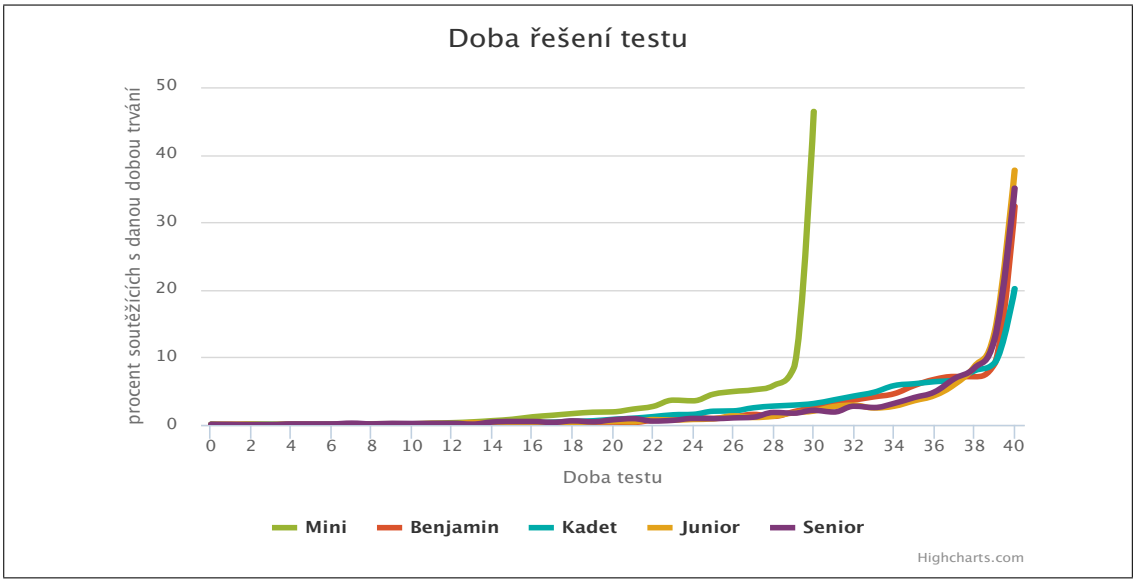
Údaje o počtech soutěžících Standardní statistika uvádí počet zúčastněných soutěžících v jednotlivých kategoriích s procentuálním rozdělením na dívky a chlapce. Můžeme vypozorovat, že čím je vyšší věková kategorie, tím menší procento dívek se soutěže účastní.



Obrázek 9: Statistika: Údaje o zastoupení dívek a chlapců

Rozdělení podle doby vyplňování testu Statistika zobrazuje, kolik procent studentů z dané kategorie ukončovalo test v závislosti na čase. Každá kategorie má vlastní časový limit. Z grafu je patrné, že nejvíce studentů odevzdávalo test až na konci časového limitu.

Zdrojová data získáme z databázových tabulek *ibobr_ROK_soutezici* a *ibobr_ROK_vysledky* obsahujících časovou značkou začátku a konce testu ve sloupcích *start* a *stop*.

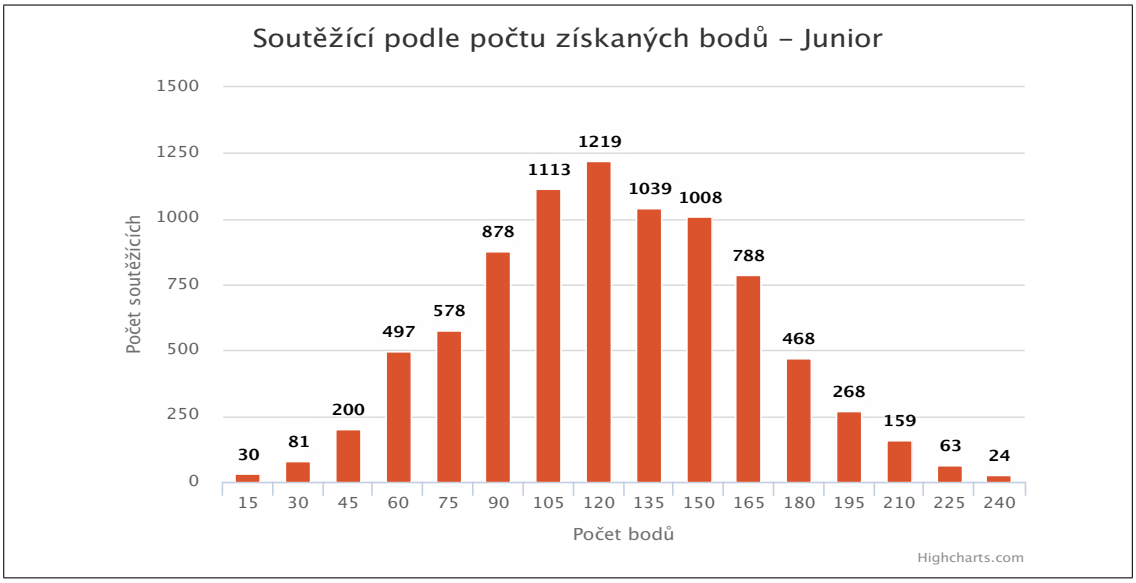


Obrázek 10: Statistika: Rozdělení podle doby vyplňování testu

Úspěšnost soutěžících Pro jednotlivé kategorie statistika udává počty všech účastníků s rozdělením na úspěšné, tedy takové soutěžící, kteří dosáhli svými body minimální hranice dané kategorie. Dalším výpočtem je počet vítězů v kategorii, jedná se o soutěžící, kteří získali maximální počet bodů v kategorii. Počty získaných bodů jsou načítány z databázových tabulek *ibobr_ROK_soutezici* a *ibobr_ROK_vysledky* ze sloupce *body*.

Průměr a medián získaných bodů Pokud chceme vědět, kolik bodů získal v průměru každý soutěžící, zjišťujeme „průměr získaných bodů“. Naopak „medián získaných bodů“ nám přibližuje, nakolik byl úspěšný průměrný soutěžící – jde o střední hodnotu bodů, získal ji soutěžící nacházející se uprostřed výsledkové listiny.

Soutěžící podle počtu získaných bodů Graf rozděluje soutěžící do skupin podle počtu dosažených bodů v předem daném rozmezí. Podle tvaru grafu pak zjišťujeme, nakolik byla soutěž pro účastníky v jednotlivých kategoriích obtížná.



Obrázek 11: Statistika: Soutěžící podle počtu získaných bodů

4 Realizace modulu

V této kapitole si podrobně rozebereme jádro celého modulu a funkčnosti veškerých tříd, způsoby získávání výsledků z databáze a publikaci výsledné zprávy.

Celý modul je psán objektově orientovaným přístupem v Notepad++ pro minimální verze PHP 5.4, MySQL 5.5 a jQuery 1.6. Využívá moderní HTML5 a CSS3 a SVG pro vykreslování interaktivních grafů. Samotné grafy jsou podporovány i staršími prohlížeči bez znalosti HTML5, vykreslování probíhá pomocí VML – jedná se o další XML formát vektorové grafiky.

Po vzhledové stránce je grafika modulu připodobněna systému, do něhož je modul integrován.

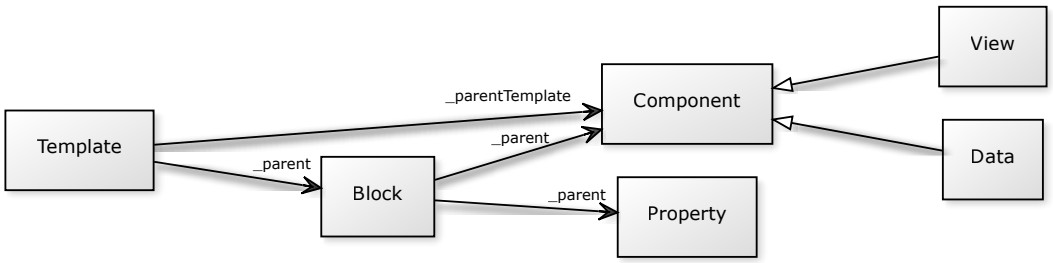
4.1 Jádro modulu

Jádro nám pokryje veškeré požadavky řešené v kapitole Návrh modulu, pro přehlednost a orientaci rozdělím třídy jádra do menších celků podle funkčnosti a zařazení:

- *Primární* – BebrasDB, Config, VariableYAML, Stats,
- *Prostředí (Template)* – StatsTemplate a StatsCountry, z adresáře *classes/statsTemplate* třídy StatsTemplateCSS a StatsTemplateJS,
- *Bloky (Block)* – StatsBlockEvent, StatsBlockProperty, StatsBlock a StatsBlockComponent a jednotlivé typy bloků v adresáři *classes/statsBlock*,
- *Data* – StatsData, StatsDataYear a StatsDataStructure a jednotlivé statistiky v adresáři *classes/statsData*,
- *Zobrazení (View)* – StatsView a StatsViewColumn a veškeré třídy zobrazení ve složce *classes/statsView*,
- *Vlastnosti (Property)* – Property, PropertyEvent, PropertyRender a PropertyControl a veškeré datové typy (rozšíření) PropertyControl ve stejnojmenné složce,
- *Publikování (Export)* – StatsJoomla,

- *Knihovny jiných autorů knihovny (Addons)* – Spyc a PHPExcel, obě umístěné v adresáři *classes/addons*.

Hierarchické závislosti objektů jsou znázorněny v obrázku 12, kde vždy můžeme přistoupit k nadřazenému objektu pomocí veřejné proměnné `$_parent` daného objektu. V případě struktury *Data* a *View* můžeme přistoupit přímo do objektu *Template* přes proměnnou `$_parentTemplate`.



Obrázek 12: Hiearchie objektů

Class diagram celého jádra, znázorňující vazby a závislosti mezi jednotlivými třídami, nalezneme v příloze 1.

4.1.1 Zavaděč StatsLoader

Zavaděč celého modulu, umístěný v *required/statsLoader.php*, načítá definice *required/defines.inc.php*, globální funkce *required/funcion.inc.php*, veškeré užívané třídy a konfigurace. Nejprve jsou nadefinovány konstanty `STATS_ROOT`, `PATH_CLASS`, `PATH_CONFIG` a `PATH_REQUIRED` určující cesty pro načítání tříd a dalších souborů. Následuje načtení konfiguračního souboru modulu *config.yml* a přístupových údajů do databáze z globálního souboru konfigurace systému *config.local.neon*. Zbývá už jen načíst soubor funkcí a jednotlivé třídy.

Třídy *StatsBlock*, *StatsData* a *StatsView* načítají vlastní rozšíření ze stejnojmenných adresářů, kde každý soubor se po načtení tzv. zaregistruje, pomocí metody `registerBlock()`, `registerData()` nebo `registerView()`, do rodičovské třídy. Tím tedy můžeme zaručit variabilitu a snadné dodávání nových možností. Podrobný popis hlaviček jednotlivých rozšíření je obsažen v podkapitole Hlavičky souborů pro správnou registraci. V poslední fázi zavaděče probíhá inicializace jednotlivých objektů a tříd.

4.1.2 Konfigurační soubory

Konfigurace databáze Přihlašovací údaje k databázi jsou uloženy v globálním konfiguračním souboru systému v datové struktuře *NEON*, jedná se o rozšířený formát YAML od Davida Grudla, autora Nette Frameworku. Konfigurační souboru nalezneme v `test/app/config/config.neon`.

Konfigurace modulu Konfigurační soubor celého modulu vychází z formátu YAML a nalezneme jej v adresáři *config/config.yml*. Samotná struktura tohoto formátu je velmi jednoduchá a pro člověka dobře čitelná. Mně se líbí, že nepotřebuje složité konstrukce a oproti struktuře JSON umožňuje vkládání komentářů; zejména u konfiguračních souborů jsou komentáře výhodou. Standardně, jako jedna z mála struktur, nabízí možnost referencí – jednotlivé struktury a hodnoty lze využít opakovaně s využitím pointerů (ukazatelů), které lze také rozšiřovat o nové vlastnosti obdobně jako u dědění v OOP.

Pro použití v konfiguračním souboru jsem potřeboval textové reference spojovat s dalšími texty – zejména pro cesty v souborovém systému. Využil jsem staženou PHP knihovnu *Spyc*, kterou jsem rozšířil na vlastní třídu *VariableYAML*, umožňující práci s proměnnými přímo v textovém souboru. Syntaxi proměnných jsem zvolil jako absolutní cestu ve struktuře k proměnné v souboru pomocí znaku (.) tzv. *dot operátoru*; celou adresu jsem uzavřel do složených závorek začínajících znakem dolaru a výsledná syntaxe tedy vypadá takto: `${a.b.hodnota}`. Jelikož se jedná o řetězec, musíme jej ještě uzavřít do uvozovek, v opačném případě nám parser knihovny *Spyc* vrátí chybu. Dosazování probíhá rekurzivním voláním metody `getPropertyPath()`, s její obdobou se ještě setkáme v podkapitole Třída Property.

VariableYAML pracuje rekurzivně tak dlouho, dokud nenajde výslednou hodnotu – můžeme tedy spolu propojovat proměnné s odkazy na jiné proměnné. Dosazování probíhá postupně, jak jsou proměnné uváděny ve vstupním souboru. V případě nalezení proměnné s odkazem na jinou proměnnou se postupně vypočítávají vnitřní proměnné, dokud není známá výsledná hodnota.

Při vypočtení (dosazení) výsledné hodnoty se hodnota uloží a při dalším výskytu se počítá již s dosazenou (výslednou) hodnotou.

4.1.3 Třída BebrasDB

Velmi důležitou částí je připojení ke stávající databázi a následné zasílání dotazů a přijímání záznamů. *BebrasDB* je statická třída umožňující připojit se k MySQL serveru pomocí MySQL ovladače, používaného v celém webu soutěže. Po upgradu PHP na verzi 7.0 a vyšší již stávající připojení přes zastaralé PHP funkce s prefixem *mysql_* nebude k dispozici, jak je psáno na oficiálních stránkách PHP.net.

Připravil jsem tedy možnost připojení pomocí novějšího ovladače MySQLi – veškeré metody ve třídě jsou společné pro oba ovladače. Pro snazší používání v programu počítá třída pouze s jedním aktivním připojením.

Mezi základní metody bychom zařadili:

- `query($sql)` – vykonání zadaného SQL dotazu v proměnné `$sql` a vrácení objektu *result*; *následující metody převolávají tuto metodu pro vykonání zadaného SQL dotazu*,
- `getRows($sql)` – vykonání dotazu a navrácení asociativního multidimenzionálního pole řádků a sloupců,
- `getRow($sql)` – totéž co metoda `getRows()`, vrací však pouze jeden (první) řádek,
- `getValues($sql)` – vykonání dotazu a navrácení jednorozměrného pole hodnot,
- `getValue($sql)` – totéž co metoda `getValues()`, vrací však pouze první hodnotu,
- `insert($sql)` – metoda vykoná zadaný dotaz pomocí metody `query($sql)` a vrátí identifikátor nově přidaného řádku,
- `exists($sql)` – vrací `TRUE`, pokud zadaný dotaz nalezne alespoň jeden záznam, v opačné případě vrací `FALSE`,
- `getColumnNames($tableName)` – výsledkem metody je pole názvů jednotlivých sloupců ze zadaného názvu tabulky `$tableName`.

Pro usnadnění zpracování proměnných v příkazu `INSERT` využívám statickou metodu `convertForInsert($key_values)`, která vrátí připravené pole řetězců s klíči *key* a *value* pro vypsání do dotazu. Klíč *key* uchovává názvy sloupců a klíč *value* zase jednotlivé hodnoty. Pro další příkaz `UPDATE` využívám obdobnou metodu `convertForUpdate($key_values)`, která nám vrátí pole již upravených parametrů „'klíč' = 'hodnota'", které stačí převést na řetězec pomocí funkce `implode()` a vložit do dotazu za příkaz `SET`.

Obě tyto metody pracují s objektem *BebrasDBSubquery*, který můžeme vložit místo hodnoty a po zpracování výše popsanými metodami se chová jako SQL vnořený dotaz.

4.1.4 Třída Stats

Statická třída modulu zajišťuje práci se *Sessions*, které jsou využívány pro dočasnou konfiguraci pro přihlášeného uživatele pomocí metod `getConfig($key)` a `setConfig($key, $value)`; obsahuje také hlavní metodu pro formátování desetinných čísel `numberFormat($number, $charLength)`. Uchovává také informaci o způsobu spuštění jádra a to *AJAX skriptem* nebo *standardním spuštěním* – tuto informaci vrací metoda `isScript()`. Třída obsahuje metodu `getAJAX($action = null, $className = null, $method = null)` pro vytváření a dosazování parametrů do vzorového URL pro jednotlivé AJAX požadavky vycházející z proměnné *path.request_link* v konfiguračním souboru.

4.1.5 Třída StatsDataYear

Statická třída uchovává veškerá čísla roků soutěže vyčtených z názvů existujících systémových tabulek databáze, jako je např. *ibobr_2015_otazky*. Spodní hranice roků zobrazovaných v modulu může být omezena „ručně" pomocí konstanty třídy `BEBRAS_START_YEAR`, kde je implicitně nastavena hodnota prvního ročníku soutěže, tedy rok 2007. Kromě čtyřmístného označení roku můžeme také využít relativní formáty počítané od vybraného roku šablony *(např. rok vybrané šablony -1 rok)*. Počet relativních formátů je omezen konstantou `RELATIVE_FORMAT_COUNT`.

4.1.6 Třída StatsTemplate

Objekt šablony s identifikátorem v proměnné `$id` obsahuje v poli `$blockList` jednotlivé bloky (viz. Třída StatsBlock) a vybraný rok šablony v proměnné `$year`, na který jsou odkázány veškeré bloky s výchozím nastavením roku. Mód zobrazení určuje proměnná `$view` podle definovaných konstant třídy:

- `VIEW_DESIGN` – návrhové zobrazení,
- `VIEW_PREVIEW` – zobrazení výstupu,
- `VIEW_PUBLISH` – zobrazení pro publikaci.

Šablony můžeme rozdělit na *standardní* a *záložní* pomocí přepínače `$isBackup`. Záložní šablona vždy existuje pouze jedna, výchozí se jmenuje *koš*, do které se přesunou bloky při smazání. V případě smazání bloku z *koše* se smaže navždy.

Renderování (vykreslování) každého bloku do šablony probíhá pomocí metody bloku `renderBlock()` pouze za předpokladu, že proměnná `$_renderManually` u daného bloku má hodnotu `FALSE`.

Načítání externích a inline skriptů a stylů zajišťují proměnné `$JS` a `$CSS`, kde `$JS` je objektem třídy *StatsTemplateJS* a `$CSS` je objektem třídy *StatsTemplateCSS*. Popišme si tedy funkčnosti a metody zmíněných objektů.

StatsTemplateJS Objekt uchovává externí JS skripty a skripty pro vykonání v závislosti na události.

- `addInclude($file)` – metoda zařídí načtení externího JS skriptu,
- `addIncludeFirst($file)` – obdobná metoda jako `addInclude()`, ale zařadí soubor na první místo v poli externích JS,
- `addInline($inlineScript)` – metoda vkládá zadaný zdrojový JS kód přímo mezi párové značky `<script>`,
- `addOnReady($inlineScript)` – metoda vychází z výše popsané `addInline()`, ale kód se vykoná v jQuery události `ready()` objektu *document*, tedy jakmile bude načten celý DOM,

- `addOnLoad($inlineScript)` – metoda také vychází z `addInline()`, ale je vykonána po dokončení načtení stránky.

StatsTemplateCSS Objekt pro CSS nabízí pouze metodu `addInclude($file)`, která zařídí načtení externích CSS stylů.

4.1.7 Třída Property

Statická třída vytváří nový přístup k objektovým proměnným pomocí tzv. *dot operátoru*, kde je celá adresa k proměnné zadána jako řetězec. Třída nabízí tři metody:

- `getPathProperty(&$obj, $stringPath, $flags)` – metoda je spouštěna rekurzivně, dokud není nalezena koncová proměnná ze zadaného argumentu cesty `$stringPath`, poté navrátí pointer nalezené proměnné; obsahuje také modifikátor zajišťující, že pokud proměnná neexistuje, metoda ji vytvoří,
- `getValue(&$obj, $stringPath)` – metoda vrátí hodnotu hledané proměnné `$stringPath` v objektu `$obj`,
- `setValue(&$obj, $stringPath, $newValue)` – zapíše hodnotu `$newValue` do proměnné `$stringPath` z objektu `$obj`.

Jako příklad poslouží proměnná objektu `$this->sourceParams->sourceType`, kde pomocí statické metody `getValue()` získáme její hodnotu pomocí cesty `"sourceParams.sourceType"`.

```
protected static function &getPathProperty(&$obj, $stringPath,
  $flags = self::NONE){
   $nullValue = NULL;
   if(!is_array($stringPath)) $stringPath = explode('.',
     $stringPath);

   //  v metode se pracuje pouze s objekty, proto pretypujeme
     veskera pole na objekt
   if(is_array($obj)) $obj = (object)$obj;
```

```
    if(count($stringPath)){
      //    vypusteni a vycteni prvniho elementu z pole cesty
      $firstVal = array_shift($stringPath);

      if(!property_exists($obj, $firstVal) && $flags == self::
        CREATE_IF_NOT_EXISTS){
        //  vytvoreni vlastnosti pri pouziti modifikatoru
          $obj->{$firstVal} = null;
      }

      if(property_exists($obj, $firstVal)){
        if(count($stringPath) == 0){
          //    jsme na konci cesty a vratime pointer
          return $obj->{$firstVal};
        } else {
          //    cesta pokracuje a volame tuto metodu rekurzivne
          return self::getPathProperty($obj->{$firstVal},
            $stringPath, $flags);
        }
      }
    }
    return $nullValue;
  }
```

Zdrojový kód 1: Metoda pro hledání proměnné pomocí cesty

4.1.8 Třída PropertyControl

PropertyControl je základní prvek všech vlastností bloků. Je přímým potomkem *PropertyRender* (obsluhuje vykreslování objektu) a jeho rodiče *PropertyEvent*, který řeší nastavení událostí.

Potomci *PropertyControl* slouží jako různé datové typy jednotlivých vlastností. Každá nová instance *PropertyControl* nebo jejích potomků vyžaduje v konstruktoru zadat cestu k dané vlastnosti (proměnné). Využívá se zde přístup k proměnným přes statickou třídu *Property* (viz. 4.1.7). Veškeré proměnné v objektech jsou skalární proměnné nebo objekty v případě, že v sobě skrývají další vlastnosti.

Mezi základní datové typy patří:

- *PropertyControlBoolean* – klasické výběrové pole s hodnotami *ano*/*ne*,
- *PropertyControlClass* – výběr třídy a její metody pomocí dvou závislých výběrových polí; pro výběr zdroje dat v modulu se využívá jeho potomek *PropertyControlClassData*, který pouze konfiguruje rodičovský objekt,
- *PropertyControlColumn* – obsahuje vykreslovací metodu `render()` sloupců datového objektu z proměnné `dataColumns`,
- *PropertyControlCheckbox* – zaškrtávácí pole s hodnotami `TRUE`/`FALSE`,
- *PropertyControlList* – výběrové pole ze zadaných hodnot v poli se plní v metodě `setList($inputArray)`, v HTML je zobrazen jako *select*,
- *PropertyControlNumber* – vstup omezen pouze na čísla; v HTML je zobrazen jako *input* prvek s typem *number*,
- *PropertyControlPanel* – soubor více vlastností, které se zařazují do panelu pomocí metody `add(&$item)`, kde `$item` je potomkem třídy *PropertyControl*,
- *PropertyControlText* – neformátovatelné textové pole; v HTML je zobrazen jako *input* nebo se při zavolání metody objektu `setLongText()` zobrazí *textarea*.

Události V *PropertyEvent* obecná metoda `addEvent($eventName, $param)` zaregistruje událost pod názvem `$eventName` s volitelnými parametry v proměnné `$param`. V současnosti jsou definovány dvě události `beforeChangeNull($arg)` a `afterChangeNull($arg)`, které před změnou (příp. po změně) hodnoty vynulují proměnné specifikované v argumentu `$arg`; obě metody převolávají `addEvent()`.

4.1.9 Třída StatsBlock

StatsBlock je základním prvkem šablony zprávy; každý blok má vlastní datový typ – vytváří se děděním základní třídy *StatsBlock*. Mezi základní typy bloků patří *text*, *objekt*, *panel*, *oddělovač* a *obrázek*.

Jedná se o potomka *StatsBlockProperty*, který obsluhuje veškeré vlastnosti daného objektu. Přiřazení vlastnosti zděděného objektu *PropertyControl* probíhá pomocí základní metody `addPropertyRef(&$propertyRef)`, naopak pokud objekt nemá obsahovat žádné nastavitelné vlastnosti, zavoláme metodu `noProperty()`.

Samotný objekt má ještě nadřazeného rodiče *StatsBlockEvent* zpracovávající události nastavené v každé vlastnosti. *StatsBlockEvent* tyto události zpracovává a metoda `_eventTrigger()` volá příslušné akce – metody s prefixem *action* (např. `actionNull()`). Ukažme si příklad na události `beforeChangeNull($arg)`, kde můžeme název rozdělit na *událost* (beforeChange) a *akci* (Null), z toho vyplývá, že při události *beforeChange* se zavolá akce *actionNull*. V objektu jsou k dispozici metody `beforeChange($propertyPath)` a `afterChange($propertyPath)`, které jsou volány při změně hodnoty z metody `setValue()` třídy *Property*.

```
$property = new PropertyControlNumber('sourceParams.dataLimit');
$property ->setTitle('Omezeni')
          ->setMin(0)
          ->setPlaceholder('neomezeno');
$this->addPropertyRef($property);
```

Zdrojový kód 2: Přidávání vlastností

Jednotlivé bloky jsou uloženy v databázové tabulce *ibobr_stats_template_block*, kde jsou uloženy fyzické vlastnosti, jako jsou titulek, typ bloku, nadřazená šablona, popis, pořadí a zda má být blok pro publikování zobrazený nebo skrytý. Další proměnné z rozšíření objektu fyzicky neexistují, vytvářejí se z definovaných proměnných objektu *PropertyControl* v inicializační metodě `propertyLoad()` a po zavedení s nimi lze pracovat jako s fyzickými; jsou uloženy v datové struktuře JSON ve stejnojmenném sloupci *json*.

Pro vytvoření objektu bloku se volá statická metoda `createObject($params)` vytvářející nový objekt podle zadaného typu v poli `$params['typ']`. Při inicializaci objektu se tedy k fyzickým proměnným zavedou proměnné z rozšíření a pod daným ID se naplní uloženými daty z databáze.
Pro manipulaci s blokem můžeme využít tyto metody:

- `save()` pro uložení všech proměnných (např. při změně),
- `remove()` pro odstranění celého bloku,
- `renderBlock()` pro navrácení vygenerovaného HTML zobrazení bloku.

Textové pole Umožňuje vkládání formátovaného HTML textu do zprávy; neobsahuje žádné nastavitelné vlastnosti použitím metody `noProperty()`. Renderování bloku probíhá přes metodu `renderField()` z bloku Pole – samotný blok se nikde nezobrazuje, využívá se pouze jeho programová část.
Jakým způsobem nastavit skrytí bloku v panelu nových bloků naleznete v Hlavičky souborů pro správnou registraci.

Pole Ačkoliv je Pole řazeno mezi bloky, nechová se tak. Můžeme říci, že se jedná o jakési „rozšíření“ textového pole o metody získání agregovaných hodnot z databáze. V návrhovém zobrazení je zobrazeno pomocí značek `{{Třída.Sloupec}}`, kde `Třída` (soubor statistik) označuje daného potomka třídy *StatsData* a `Sloupec` je daný sloupec SQL dotazu, který navrací metoda `&getFieldSQL()` z vybrané třídy.

Oproti datovým třídám zde přibyly ještě *modifikátory*, které určují druh datových tabulek:

- `vsichni` – vyčteni všichni soutěžící z tabulky *ibobr_ROK_soutezici*,
- `oficialni` – pouze oficiální výsledky z tabulky *ibobr_ROK_vysledky*.

Příklad syntaxe pole: `{{Info.Pocet_soutezicich_2015|vsichni}}`.

Způsoby přidávání vlastních hodnot viz. Naplnění Pole třídy v programátorské příručce.

Oddělovač Je standardní vodorovná čára vložená pomocí tagu `<HR />`, neobsahuje žádné vlastnosti.

Panel Blok vytváří objekty v objektu pomocí tabulky o velikosti `$cols` · `$rows`, kde v každé buňce je odkaz na ID vloženého bloku. Tato tabulka (proměnná pole) se jmenuje `$blockMatrix`.

Bloky uvnitř panelu mají nastavenou hodnotu proměnné `$_renderManually` na `TRUE`. Při renderování šablony se nám tedy nevykreslí na úrovni ostatních bloků, ale panel si je vykreslí sám.

Nás nyní bude zajímat především část, která se vykonává na straně JavaScriptu, a to jsou metody a událost:

- `detach()` – každý blok uvnitř panelu je odpojen od daného panelu a přesunut do šablony na základní úroveň,
- `attach()` – veškeré bloky, které jsou navázány podle ID bloku na buňky panelu (HTML atribut `data-join-block-id`), jsou ze šablony přesunuty do dané buňky,
- `destroy()` – zruší veškeré vazby panelu a vrátí bloky do šablony pomocí metody `detach()`,
- `afterDrop()` – událost po přesunutí daného bloku z/do panelu.

Objekt Princip bloku funguje tak, že se nejprve vytvoří *objekt vybraného zobrazení*, tedy instance potomka třídy *StatsView* (např. *StatsViewChart*), do kterého jsou načtena zpracovaná data. Tento objekt zobrazení je uložen v proměnné `$_objectView`.

Veškeré nastavení objektu nám zajišťují proměnné `sourceParams`, `viewParams`, `viewExtension` a `dataColumns`. Každá tato proměnná je souborem dalších vnořených proměnných objektu *PropertyControl*, které znázorňuje diagram na obrázku 13. Popišme si tedy funkčnosti popsaných proměnných, seřazených podle pořadí zpracování:

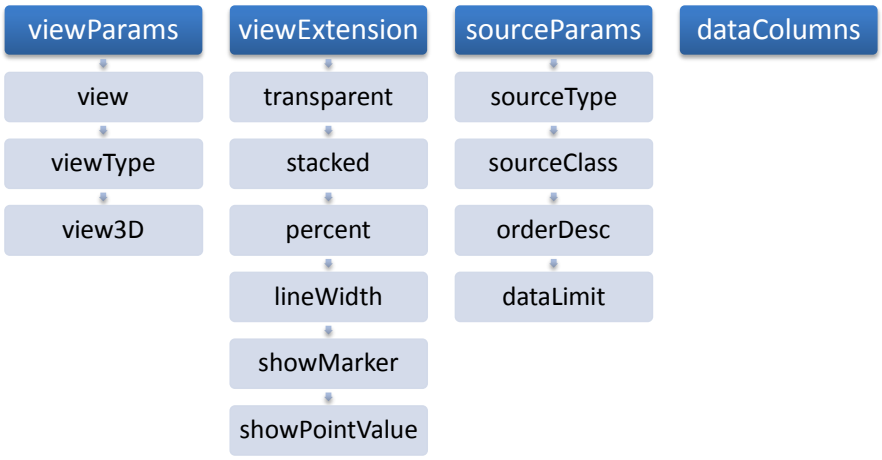
- *viewParams* – parametry pro zobrazení, které jsou v průběhu procesu neměnné,
  - ▷ *view* – určuje způsob prezentace datového objektu, podle kterého se zvolí daná instance objektu zobrazení,

  - ▷ *viewType* – další způsob zobrazení volitelný podle vybraného objektu zobrazení, např. u grafů se zde vybírá typ grafu,
  - ▷ *view3D* – u grafů se přepíná zobrazení výstupu ve 2D nebo 3D.

- *sourceParams* – nastavení datového zdroje a způsoby upravení SQL dotazu před jeho vykonáním,
  - ▷ *sourceType* – přepínač zdrojové tabulky dat *všichni soutěžící* a *oficiální výsledky*,
  - ▷ *sourceClass* – výběr datové třídy a metody (výběr dané statistiky), viz. Třída StatsData,
  - ▷ *orderDesc* – pokud je hodnota `TRUE`, záznamy budou z databáze řazeny opačně,
  - ▷ *dataLimit* – číselná hodnota určuje počet zobrazených záznamů z databáze, pokud nechceme dotaz omezovat, hodnota proměnné je `NULL`.

- *dataColumns* – obsahuje pole objektů sloupců *StatsViewColumn*, parametry jsou popsány v podkapitole Statistiky,

- *viewExtension* – další rozšíření zobrazení, které může být v průběhu procesu změněno; je závislé na vybraném zobrazení,
  - ▷ *transparent* – zprůhlední pozadí grafu nebo mapy,
  - ▷ *stacked* – tzv. skládaný graf, který sloučí všechny datové řady v rámci jednotlivých kategorií,
  - ▷ *percent* – při nastavení hodnoty `TRUE` budou veškeré zadané hodnoty grafu přepočteny na procentuální hodnoty,
  - ▷ *lineWidth* – udává tloušťku čáry v pixelech u průběhových grafů,
  - ▷ *showMarker* – každý bod grafu, při nastavení hodnoty `TRUE`, bude označen značkou,
  - ▷ *showPointValue* – zobrazí hodnoty jednotlivých bodů při nastavení hodnoty `TRUE`.



Obrázek 13: Důležité proměnné objektu StatsBlockObject a podproměnné využívané ve všech druzích zobrazení

S objektem *StatsBlockObject* úzce spolupracují dva další objekty. Podle funkce je můžeme rozdělit do dvou částí:

1. Zobrazovací (prezentační) část - viz. Třída StatsView
2. Datová část - viz. Třída StatsData

Tyto části jsou potomky *StatsBlockComponent*, který nabízí metodu pro přístup do bloku `parentBlock()`, přímý přístup do šablony pomocí `parentTemplate()` a přidávání vlastností metodou `addPropertyRef(&$propertyRef)`, která převolává stejnojmennou metodu z *StatsBlock*. Hierarchie je znázorněna na obrázku 12.

4.1.10 Třída StatsView

Zobrazování statistik do *StatsBlockObject* zajišťuje třída *StatsView*. Vytvořením instance jejích potomků je přesně určen druh zobrazení, do kterého se doplní data statistiky.

Globální konstanty zobrazení Při inicializaci třídy se v zavaděči statickou metodou `initDefines()` zavedou globální konstanty pomocí funkce `define()` pro jednotlivé druhy zobrazení. Hodnota každé konstanty je 2^k, kde $k \in \langle 0; n \rangle$ a n je počet všech zobrazení. Tyto konstanty se využívají v potomcích datového objektu *StatsData* k určování, která zobrazení jsou pro data určena.

Využívám zde *bitové operace* sloužící k práci s čísly nebo řetězci na úrovni jednotlivých bitů. [19] Vysvětleme si, jak jednotlivé operátory fungují a porovnejme s pravdivostní tabulkou 1:

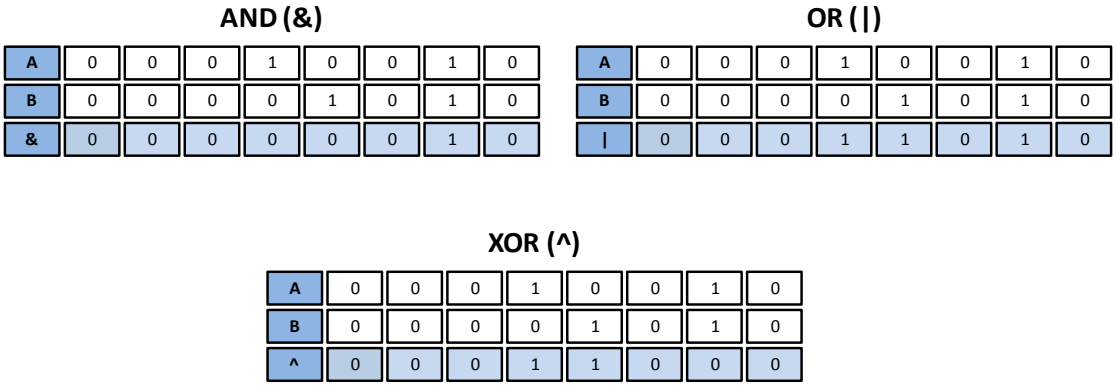
- *AND (logický součin)* – je pravdivý, pokud jsou oba operandy pravdivé,
- *OR (logický součet)* – je pravdivý pouze tehdy, pokud *alespoň* jeden z operandů je pravdivý,
- *XOR (exkluzivní logický součet)* – je pravdivý, pokud *právě* jeden z operandů je pravdivý,
- *NOT (negace)* – je operátor, kde výsledkem je *opačná* vstupní hodnota, tedy pokud je hodnota operandu na vstupu *pravda*, výsledkem bude *nepravda*.

operandy		operátor AND	operátor OR	operátor XOR	operátor NOT
`$a`	`$b`	`$a & $b`	`$a | $b`	`$a ^ $b`	`~ $a`
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

Tabulka 1: Pravdivostní tabulka bitových operací

V tabulce jsou zobrazeny číselné hodnoty pouze pro snazší pochopení, nahradil jsem zde logické hodnoty pravda a nepravda za hodnoty 1 a 0.

Ukažme si na příkladech z obrázku 14 výpočty jednotlivých bitových operací, když použijeme pravidla z pravdivostní tabulky.



Obrázek 14: Ukázka výpočtů bitových operací

Navraťme se tedy ke konstantám zobrazeným v tabulce 2 a všimněme si binárního zápisu čísel. Vidíme, že vždy následující konstanta má jedničku posunutou doleva oproti konstantě předchozí. Pokud tedy všechny tyto konstanty binárně sečteme, dostaneme konstantu `VIEW_ALL`.

konstanta	hodnota dekadicky	hodnota binárně
`VIEW_TABLE`	1	00000001
`VIEW_LIST`	2	00000010
`VIEW_CHART`	4	00000100
`VIEW_MAP`	8	00001000
`VIEW_FILE`	16	00010000
`VIEW_PHP`	32	00100000
`VIEW_ALL`	63	00111111

Tabulka 2: Globální konstanty zobrazení určené pro bitové operace

Uveďme si tedy 2 příklady, proč jsou bitové operace vhodné:

1. Chceme-li zobrazit mapu a seznam, proto sečteme `(VIEW_MAP | VIEW_LIST)`.
2. K dispozici budou všechna zobrazení vyjma mapy. Nechceme však všechna složitě vypisovat a především, při dodání nového zobrazení, by se muselo při-

dávat i do této definice. Využijeme operátor *XOR* (`VIEW_ALL ^ VIEW_MAP`), kde od součtu všech zobrazení odebereme zobrazení mapy.

Proces zobrazení Samotný objekt zobrazení nedisponuje mnoha vlastnostmi, vystačí si s titulkem `$title` a objektem `$data` uchovávajícím datový objekt *StatsDataStructure*.

Vykreslování zobrazení probíhá postupně do proměnné `$out`, která je v metodě `render()` navrácena. Tato metoda se ještě dělí na části: *záhlaví*, *tělo* a *zápatí*, kde každá část je vykonávána ve zvláštní metodě `writeHead()`, `writeBody()` a `writeFoot()`.

U každého zobrazení můžeme do statického pole `$externJS` vložit externí JS skripty, které jsou pro něj nezbytné. U tohoto pole jsou důležité dva klíče, *first* a *other*, kde pod každým klíčem je uloženo pole JS skriptů. Klíče jsou:

- *first* – skripty se načtou na prvním místě v šabloně,
- *other* – skripty jsou načítány jako druhořadé.

Určování pořadí načítání je vhodné pro skripty, na kterých jsou ostatní skripty závislé a je nutné, aby byly načteny dávno před nimi (v našem případě se jedná zejména o knihovnu *highcharts.js*).

Objekty zobrazení

- *StatsViewHtmlTable* – standardní HTML tabulka se záhlavím a tělem
- *StatsViewChart* – interaktivní graf
- *StatsViewMap* – interaktivní mapa
- *StatsViewHtmlList* – HTML odrážkový seznam
- *StatsViewPHP* – zobrazení pracovního pole v PHP pro ladění při rozšiřování modulu
- *StatsViewFile* – výstup do externího souboru

4.1.11 Třída StatsData

Třída zpracovávající data z databáze je děděna do tříd reprezentujících jednotlivé kategorie dat, jako jsou *Otázky*, *Soutěžící* a *Školy*. Každá třída obsahuje metody pro vyhodnocení statistiky s prefixem *view* (např. `viewPocet_soutezicich()`). Každá metoda může mít vlastní vstupní parametry. Rozeberme si tedy postup vyhodnocování metody při získání dat:

1. *Vyhodnocení SQL dotazu* – Jednotlivé SQL dotazy, které nevracejí žádné záznamy, jako je zavádění a plnění dočasných tabulek, zavádění proměnných a další, voláme pomocí standardní metody `BebrasDB::query($sql)`. V opačném případě vyhodnocujeme přes metodu `executeDataQuery($sql)`, která dokáže dotaz obohatit o klauzule `ORDER BY` a `LIMIT`. Pro získání názvů datových tabulek systému obsahujících v názvu rok soutěže musíme využít vlastní metodu `getTableName($sufix)`, která pracuje s vybraným rokem bloku. V případě využití datových tabulek systému *ibobr_ROK_soutezici* a *ibobr_ROK_vysledky* bychom měli použít zástupný znak `{{DATA}}`, který dosadí vybranou tabulku v závislosti na výběru typu výsledků *všichni soutěžící* nebo *oficiální výsledky*.

2. *Nastavení sloupců do výstupu z výsledku dotazu* – Zavedeme si objekt sloupce *StatsViewColumn*, který určuje datový typ a další vlastnosti, které jsou popsány v programátorské příručce v Statistiky; veškeré proměnné lze nastavovat pomocí setterů (např. `setFloatDigits(5)`).

3. *Zobrazení* – Do objektu zobrazení uloženého v proměnné `$pViewObject` nastavíme zobrazený titulek pomocí metody `&setTitle($title)` a do proměnné `$data`, obsahující objekt *StatsDataStructure* (viz. 4.1.12), předáme získaná data a další parametry.

4. *Rozšíření* – V poslední fázi se řeší nastavení rozšiřujících parametrů pomocí metody `addExtension($property, $value)`. Ukažme si například následující rozšíření `addExtension('viewParams.stacked', true)`, které nám říká, že graf bude zobrazen jako *skládaný*. Přehled vlastností je zobrazen na obrázku 13.

4.1.12 Třída StatsDataStructure

Data a informace pro zobrazení statistiky jsou strukturována do objektu *StatsDataStructure*. Tato data získaná z SQL dotazu můžeme rozdělit do dvou skupin:

- *agregovaná* – jsou taková data, kde potřebujeme získat souhrny a přehledy; v dotazu byla použita klauzule `GROUP BY`, agregační funkce `COUNT()`, `SUM()` a jiné; typy agregovaných grafů jsou *sloupcový*, *pruhový*, *výsečový*,
- *průběhová* – standardní data zobrazující určité průběhy nebo změny v závislosti na jiné proměnné (např. počet ukončených testů v závislosti na čase); typy proběhových grafů jsou *spojnicový* a *bodový*.

Popišme si metody pro zobrazené sloupce:

- `getDisplays()` – vrátí pole objektů sloupců určené k zobrazení v *tabulkovém výstupu*,
- `getPlots()`– vrátí pole objektů sloupců určené k zobrazení v *grafu*.

Nastavovací metody pro všechny druhy zobrazení:

- `setRows($val)` – argument `$val` obsahuje jednotlivé řádky získané z SQL dotazu,
- `setSQL($sql)` – `$sql` je pole vykonaných SQL dotazů; volá se v metodě `executeDataQuery($returningSql, $sqlArray)` objektu *StatsData*, kde `$returningSql` je SQL dotaz, který navrací řádky z databáze a `$sqlArray` je pole veškerých předchozích vykonaných dotazů, které výsledky nevracejí – můžeme mezi ně zařadit vytvoření, plnění dočasných tabulek a další.

Metody pro nastavení zobrazení grafů:

- `setAggregated($val = true)` – nastavení informace, zda jsou data *agregovaná*; metoda automaticky mění typ grafu v zobrazení na prvního zástupce skupiny *agregovaných* nebo *průběhových* grafů,

- `setSectionKey($val)` – argument `$val` určuje, ze kterého sloupce hodnot se vytvoří jednotlivé datové řady,
- `setTitleX($val)` – nastavuje popis osy X z argumentu `$val`,
- `setTitleY($val)` – nastavuje popis osy Y z argumentu `$val`,
- `setAxisX($val)` – argument `$val` je pole hodnot zobrazených na ose X; pokud tyto hodnoty mají být hodnotami ze sloupce z vykonaného SQL dotazu v proměnné `$rows`, můžeme jako argument dosadit volání metody `dataFromColumn($columnName)`, kde `$columnName` je název sloupce z dotazu,
- `setTooltip($val)` – nastavuje titulek při najetí na určitý bod v grafu (bublinková nápověda); nastavujeme pouze tehdy, pokud chceme výchozí titulek změnit.

Popišme si postup zpracování dat:

1. Nastavení sloupců do proměnné `$columnsToView` obsluhuje metoda `setColumnsView(&$storedColumns)`, kterou zavolá nadřazený blok *StatsBlockObject*. Argument `$storedColumns` jsou data sloupců uložených v databázi. V případě, že sloupce nejsou k dispozici (argument je tedy prázdný), vrátí se proměnná `$columnsResult` – výchozí data sloupců z vyhodnocené metody datové třídy.
2. V metodě `init()` se provádí celkové součty hodnot pro jednotlivé sloupce. Ty jsou použity pro následné výpočty procentuálních hodnot do pole `$total` pod klíčem každého sloupce. Současně se také provádí zjišťování průměrného počtu desetinných míst u čísel každého sloupce.
3. Pro grafy se v metodě `initSectionPrepare()` připraví jednotlivé datové řady a v následující metodě `initSectionWrite()` se naplní daty. Pokud je pouze jedna datová řada, proměnná `$legend` změní hodnotu z výchozí `TRUE` na `FALSE` a legenda (výběr datatových řad) se vůbec nezobrazí.

4.1.13 Třída StatsCountry

Třída, využívaná pouze ve statistice *Původ otázek*, obsahuje seznam veškerých zemí v databázové tabulce *ibobr_stats_country* vedených pod identickým klíčem (sloupec *key*), kde každá země může mít ještě další možné klíče (ve sloupci *other_key* oddělené svislou čarou „|“). Je počítáno s širším využitím, proto třída obsahuje sloupce *padSgN* a *padPlN*, kde *N* značí pořadí pádu, *Sg* (singular) jednotné číslo a *Pl* (plural) číslo množné. V současnosti se využívá pouze *padSg2*, který je též obsažen v databázové tabulce.

4.2 Návrhové zobrazení

4.2.1 Serverová část

Veškeré požadavky zasílané přes AJAX zpracovává soubor *request.php* v kořenovém adresáři modulu. Zpracovává akce pro základní manipulaci s bloky (*create-block*, *load-block*, *save-block*, *remove-block*), ke kterým můžeme také zařadit akci *set-move-position* při přesouvání bloků. Dále zpracovává požadavky při změně vlastností bloků (*get-property*, *set-property*, *sort-property*) a výstup do souboru pomocí akce *export-file*.

V případě provádění akcí šablony, kde se provádí obnovení celé stránky prohlížeče, zpracovává tyto akce handler, umístěný v adresáři *required/templateController.php*, bezprostředně po načtení zavaděče. Jedná se o akce pro manipulaci se šablonou (*create-template*, *change-template*, *rename-template*), výběr roku šablony (*change-year*) a publikování na oficiální stránky Bobříka informatiky.

4.2.2 Klientská část

Každý HTML objekt bloku je rozšířen na základní objekt bloku `IBlock` obsahující ty nejdůležitější vlastnosti a metody. Dále je objekt rozšířen o základní události `IBlockEvent` a podle typu bloku `IBlockXYZ` (např. `IBlockText`) je rozšířen o vlastnosti a metody daného typu bloku.

Popišme si události `activate()` a `deactivate()` sloužící k označení (popř. odznačení) bloku – při každém zavolání `activate()` se zavolá metoda `deactivate()`

na vybraném bloku ztrácejícím focus (aktivitu) a odkaz na nově vybraný blok se uloží do proměnné `Template.selectedBlock`.

Seznam všech bloků v šabloně nalezeneme v `Template.blockList[ID_bloku]`, kde `ID_bloku` je atribut `data-id` HTML objektu. Každý tento blok je propojen s panelem vlastností, proto se při označení vykoná akce `get-property` a načtou se vlastnosti bloku. Při opuštění bloku je volána metoda `save()`, která pomocí metody `getValue()` (vracející obsah bloku), odešle přes akci `save-block` data na server.

V případě bloku Panel – fungujícího jako tabulka o r řádcích a s sloupcích – se využívají metody `attach()` pro přiřazení bloku do panelu a `detach()` pro odstranění z panelu. Každá buňka panelu obsahuje atribut `data-join-block-id` uchovávající ID vnitřního bloku. V případě smazání celého panelu se spustí metoda `detach()` a veškeré vnitřní objekty se přiřadí pod seznam bloků s identifikátorem `block-container` (šablonu).

Vertikální rozdělení obrazovky pomocí JS doplňku *JS Splitter* musí v události `paneResized()` spouštět metodu `Template.refreshCharts()`, aby se veškeré grafy a mapy přizpůsobily a překreslily na požadovanou velikost.

4.3 Integrace do Joomly

Výsledná vygenerovaná zpráva bude zobrazena na oficiálním webu Bobříka informatiky. Jednotlivé stránky v CMS Joomla, na kterém jsou stránky Bobříka postavené, využívají základní modul *články*, které nabízejí spoustu možností. Vytváření a editace stránek se provádí ve WYSIWYG editoru v administraci článků. Způsob vytváření článku programově jsem čerpal z oficiální dokumentace pro aktuální verzi Joomly 2.5; vkládání zahrnuje 3 databázové tabulky:

- *jos_menu* – „registruje“ stránky pod daným *aliasem* (alfanumerickým identifikátorem stránek), ke kterému je přiřazena *kategorie (option)*, v našem případě se jedná o *com_content* a *zobrazení (view)* s hodnotou *article* nám zavádí modul článku,
- *jos_content* – obsahuje veškerá data článku, sloupec *title* obsahuje titulek článku, text článku je obsažen ve sloupci *introtext*, ostatní sloupce jsou da-

tumy a zakladatelé článku,

- *jos_assets* – určuje práva přístupů k článku.

Při prvním testu byla zpráva vložena jako HTML kód s externími JS skripty, inicializačními skripty pro grafy a jedním CSS stylem; na webu se vše zobrazovalo správně, dokud se neprovedla editace článku (naší vygenerované stránky) ve WYSIWYG editoru Joomly. Veškeré tagy `<script>` a `<link>` editor odstranil a pak nebylo možné grafy zobrazit. V tomto případě bylo nutné vytvořit dva vlastní pluginy do Joomly, které budou zachytávat volané události při generování obsahu. Zde jsem se inspiroval již hotovým pluginem *AddCustomCSS* vkládajícím externí CSS styly do hlavičky. Rozdělil jsem tedy tělo stránky a hlavičku, která se nyní ukládá do tabulky *ibobr_stats_export* společně s ID generovaného článku. Popišme si tedy jednotlivé pluginy:

- *jQueryLoader* – v samotném systému mezi základními JS skripty nebylo načítáno jQuery, proto jsem se rozhodl zařadit jej do pluginu, který načítá jQuery přímo v hlavičce a mohou s ním další JS pluginy okamžitě pracovat; v pluginu je volána událost `onBeforeCompileHead()`, ve které se do hlavičky dokumentu vloží pomocí metody `addCustomTag()` tag `<script>` s odkazem na externí skript,

- *BebrasStats* – využívá událost `onContentBeforeDisplay()` při načítání každého článku, kde porovná ID článku s tabulkou *ibobr_stats_export* a doplní uložené JS a CSS.

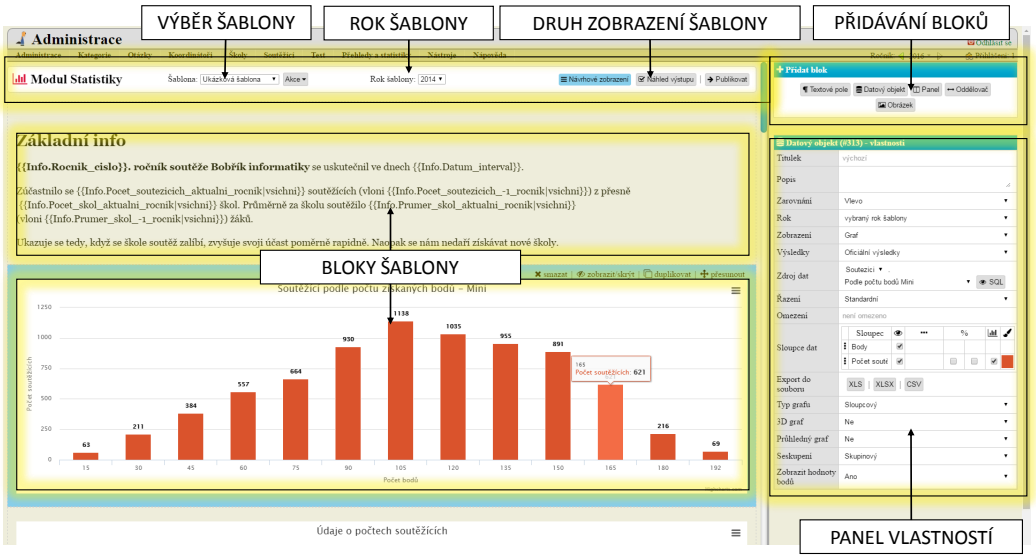
Po opětovné editaci obsahu stránky se grafy opět nezobrazily i přesto, že pluginy pracovaly správně a dodávaly data do hlavičky. Editor odstraňoval i vnořené tagy `<div>`, umístěné v blocích grafů a map. Nezbývala jiná možnost, než využít vlastní značky `{RENDER_BLOCK ID=[ID_BLOKU]}`, které nebudou měněny editorem Joomly a při zobrazení uživatelům plugin dosadí místo značky výsledné HTML objekty.

5 Popis uživatelského prostředí

Prostředí pro vytváření závěrečné zprávy je uzpůsobeno tak, aby uživatel měl snadno veškeré prvky k dispozici. Převážnou část kapitoly se budu zabývat návrhovým zobrazením a způsobem vytváření zprávy.

5.1 Seznámení s prostředím

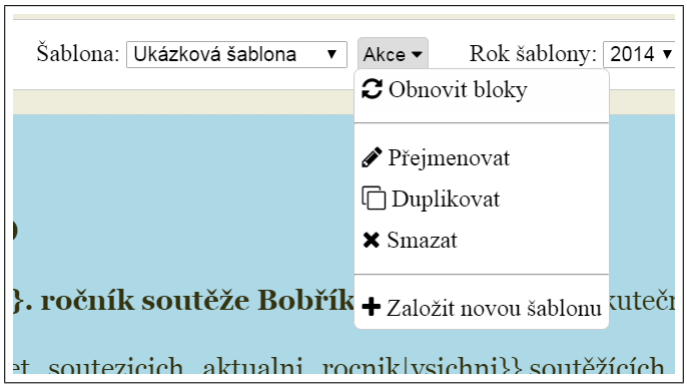
Vytváření zprávy probíhá v návrhovém zobrazení, které je rozděleno na dvě části splitterem (viz. jQuery Simple Splitter), levá část obsahuje jednotlivé bloky datových typů pod sebou, v pravé části se nachází panel pro manipulaci s těmito bloky a panel pro přidávání nových bloků.



Obrázek 15: Rozvržení návrhového zobrazení

V horním panelu vedle nadpisu modulu se nachází výběr dané šablony, zde může uživatel aktuální šablonu uložit, vytvořit její kopii, přejmenovat nebo odstranit. Pro danou šablonu můžeme v poli *Rok šablony* vybrat rok, který bude výchozí u jednotlivých bloků šablony. Užitečnou možností je záložní šablona pojmenovaná jako „koš“, do které se ukládají bloky odstraněné ze standardní šablony.

V případě potřeby lze odstraněné bloky navrátit do standardní šablony nebo je trvale smazat.



Obrázek 16: Ukázka pole výběru šablony a možné akce

V pravé části horního panelu jsou tlačítka pro druh zobrazení šablony:

- *návrhové zobrazení* – prostředí pro vytváření zprávy,
- *náhled výstupu* – ukáže vzhled zprávy před publikováním,
- *publikovat* – exportuje zprávu na oficiální stránky Bobříka informatiky.

V šabloně můžeme každým blokem manipulovat – přesouvat ho, smazat, duplikovat, určit, zda bude ve výstupu zobrazený či skrytý. Zároveň se v pravé části zobrazí panel vlastností vztažený k aktuálnímu bloku spolu s jeho typem a ID. Panel je závislý na typu bloku, pro každý nabízí jiné možnosti. Popišme si jednotlivé typy bloků.

Textové pole kromě formátovaného textu nabízí možnost vkládat do textu určitou hodnotu z databáze, např. počet soutěžících v daném roce. Je nutné využít tzv. *Pole*, které vloží do textu značku – ta je ve výstupu nahrazena danou hodnotou. Ve vlastnostech textového pole se nachází panel *pole*, kde vybereme kategorii hodnot (v současné verzi je k dispozici pouze kategorie *info*). Rozbalí se tím seznam veškerých možných hodnot, kde prostým kliknutím vložíme do textového pole zvolenou hodnotu.

Panel rozdělí blok na více částí formou tabulky – volíme si počet řádků i sloupců, kde do každé buňky můžeme přesunout libovolný blok.

Typ bloku *Datový objekt* zahrnuje nejširší spektrum možností. Volíme tu způsob zobrazení, jako je tabulkový výstup, graf, mapa či seznam – podle zdrojových dat – a další rozšiřující vlastnosti.

Pro oddělení jednolivých částí zprávy můžeme využít *Oddělovač*, který bude na výstupu prezentován horizontální čárou.

Jako doplňující blok lze vložit *obrázek*, pro zobrazení vložíme jeho externí URL adresu. Možnosti v panelu nástrojů se pak vztahují k jeho umístění, velikosti a popisku.

5.2 Vytváření zprávy

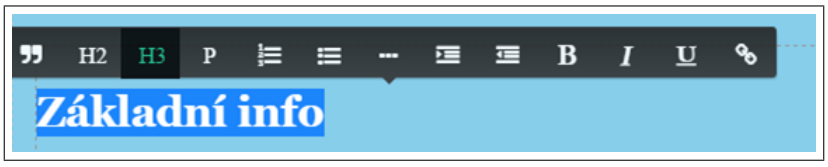
5.2.1 Vytváření bloků a práce s nimi

Pokud chceme vytvářet zcela novou zprávu, rozklikneme tlačítko *akce* v horním panelu a vybereme položku *Založit novou šablonu*. Napíšeme její jméno a potvrdíme. Pokud však chceme vycházet z již uložené šablony, vybereme požadovanou šablonu a poté v tlačítku *akce* zvolíme položku *Duplikovat*.

Z panelu pro přidávání bloků vybereme požadovaný typ bloku a klikneme na tlačítko. Blok se vloží na konec zprávy a okno šablony se na něj přesune; při najetí na blok se zobrazí panel pro jeho manipulaci – uchycením položky *přesunout* jej můžeme přesunout na požadované místo.

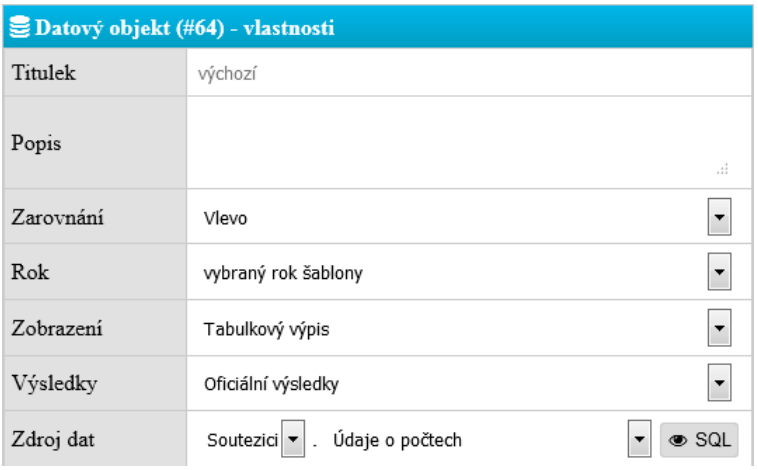
5.2.2 Editace textového pole

U textového pole se nad označeným textem zobrazí panel pro formátování, jako jsou citace, odstavce, nadpis 2. a 3. úrovně, číslovaný a odrážkový seznam, zvětšení a zmenšení odsazení zleva, tučnost, kurzíva a podtržení písma a vkládání odkazů.



Obrázek 17: Panel formátování textu

5.2.3 Editace datového objektu



Obrázek 18: Ukázka panelu vlastností

Datový objekt nabízí základní vlastnosti:

- *titulek* – nadpis objektu,
- *popis* – doplňující text vyznačený kurzívou zobrazený pod objektem,
- *zarovnání* – zarovnání vlevo, na střed a vpravo,
- *rok* – rok pro vyfiltrování zdrojových dat, ve výchozím stavu se odkazuje na nastavený rok šablony, lze nastavit přesný rok nebo relativní formát odvozený od vybraného roku šablony,
- *zobrazení* – způsob prezentace zpracovaných dat,
  - ▷ *tabulkový výpis* – zobrazuje prostá data v tabulce se záhlavím,
  - ▷ *graf* – umožňuje zvolit typy grafů, jako jsou spojnicový, bodový, pruhový, sloupcový a výsečový; data je možné seskupit do skupinového nebo skládaného grafu,
  - ▷ *mapa* – zobrazuje mapu se zvýrazněnými jednotlivými státy,
  - ▷ *seznam* – výčet zpracovaných dat (využívá se pouze u statistiky *Původ otázek*),

  - ▷ *PHP* – slouží pro řešení problémů při úpravě a vytváření nových statistik; zobrazí se obsah PHP pole.

- *výsledky* – zobrazení výsledků všech soutěžících nebo oficiálních výsledků,
- *zdroj dat* – výběr statistiky z kategorie (třídy) a druh statistiky (metoda), kliknutím na tlačítko SQL se zobrazí vykonávaný SQL dotaz,
- *řazení* – řazení dat vzestupně nebo sestupně,
- *omezení* – omezení počtu záznamů; pokud pole zůstane prázdné, počet se nebude omezovat,
- *sloupce dat* – umožňují vybrat data, která budou v grafu či tabulce viditelná, a způsob a formát jejich zobrazení; náhled je zobrazen na obrázku 19,
- *export do souboru* – exportuje vybraný zdroj dat do souboru XLS, XLSX nebo CSV.

Vlastnosti se přidávají nebo odebírají v závislosti na výběru zobrazení a zdroje dat.

Obrázek 19: Sloupce dat

Popišme si jednotlivé buňky řádku *sloupce dat* z obrázku v zobrazeném pořadí:

- uchycením tří vertikálních teček přesuneme sloupec,
- editovatelný název sloupce,
- zobrazit/skrýt v tabulkovém výstupu,
- zobrazovaný počet desetinných míst čísla (vypočte se automaticky jako průměrný počet desetinných míst ve sloupci),

- formátovat číslo jako procento (z podílu v desetinném čísle),
- zobrazovat procentuální zastoupení (výpočtem),
- zobrazit/skrýt v grafovém výstupu,
- výběr barvy datové řady

Díky těmto vlastnostem můžeme korigovat výsledný vzhled bloku podle aktuálních požadavků.

5.3 Publikace zprávy na web Bobřík informatiky

Dokončenou zprávu můžeme nyní publikovat na web. Doporučuji však ještě před tím zkontrolovat v *náhledovém zobrazení*. Publikace se provádí kliknutím na tlačítko *publikovat* v horním panelu. Jednotlivé publikace nejsou verzovány, zpráva se tedy vždy přepíše zprávou pod stejným klíčem, tzv. *aliasem*. Pokud tedy chceme původní zprávu zachovat, je nutné nejdříve v administraci stránek v CMS Joomla vytvořit její kopii.

Před publikací se zobrazí potvrzovací dialog, že bude zpráva smazána a zda chce uživatel pokračovat. Zpráva je vygenerována jako článek v HTML do Joomly, je také připravena na následnou editaci v administraci Joomly.

6 Programátorská příručka

Smyslem příručky je představit některé situace, které by při práci s modulem mohly nastat. V případě chybových hlášení rozebírá důvod jejich vzniku a následné řešení. Nastiňuje také způsob, jakým funguje komunikace se serverem a v neposlední řadě obsahuje základní informace nezbytné pro další možná rozšíření modulu.

6.1 Chybová hlášení a ladění

6.1.1 Kritické chyby

Provedeme si výčet veškerých kritických chyb, které způsobí zastavení běhu programu, protože chybějící prostředky jsou pro jeho běh nutné.

„Nenalezen konfigurační soubor modulu!“ – konfigurační soubor se načítá v souboru *required/statsLoader.php*, zavaděč je popsán v Zavaděč StatsLoader,

„Nenalezen konfigurační soubor databázového spojení!“ – problém je popsán v zde: Konfigurace databáze,

„Nepodařilo se připojit k databázi ‘XZ“‘ – připojení k požadované databázi není možné ze dvou důvodů: databázový server neodpovídá (není momentálně k dispozici) nebo jsou zadány nesprávné přihlašovací údaje. Přihlašovací údaje nalezneme v Konfigurace databáze,

„VariableYAML: Nenalezena proměnná ’YZ’ v ’XY’“ – v tomto případě není nutné složitě popisovat, v objektu *VariableYAML* je spuštěn proces pro vyhledání proměnné ze zadané adresy s pomocí metody `completeProccess()` a zadaná proměnná fyzicky neexistuje,

„Chyba při načítání seznamu států. Nebyly nalezeny žádné záznamy v tabulce ‘ibobr_stats_country‘!!!“ – chyba přesně specifikována,

„Nenalezena properta ‘XYZ“‘ – u objektů *StatsTemplateJS* a *StatsTemplateCSS* jsou metody pro zjištění hodnoty (*gettery*) a zjištění stavu (název začíná prefixem *is*) vyhodnocovány v magické metodě `__call()`, která

kontroluje existenci dané proměnné voláním metody `checkProperty()`; pokud proměnná neexistuje, nastane tato chyba,

„Chybí metoda _renderBlock() v třídě XY“ – v daném rozšíření objektu *StatsBlock* chybí metoda `_renderBlock()`,

„Neznámý typ souboru XY“ – handler akcí *request.php* vrátí tuto chybu, pokud při exportu objektu do souboru zaslaný argument `type` není známý; v případě přidání nových formátů je nutné dodat formát do proměnné `$fileTypes` v souboru *StatsViewFile*,

„Do adresare "tmp" nelze přistoupit, zkontrolujte práva.“ – při exportu do souboru (XLS, XLSX a CSV) v objektu *StatsViewFile* v metodě `createFile()` nebylo možné přistoupit do dočasného adresáře „tmp“, aby se zde soubor dočasně uložil a poté předal uživateli ke stažení; je nutné přenastavit práva přístupu k adresáři.

6.1.2 Upozornění

V některých případech program pouze zobrazí varování o chybě a pracuje dále. Tato situace nastává, jestliže chybějící informace nejsou pro běh programu nezbytné.

„Chybějí následující konstanty 'YZ' v třídě 'XY'“ – při registraci bloků se spouští metoda `registerBlock($block)` v objektu *StatsBlock*; v případě nenalezení důležitých konstant `NAME` nebo `TYPE` se zobrazí tato hláška,

„Blok typu 'XY' není k dispozici!“ – je požadován neregistrovaný (neznámý) typ bloku klíč `type` v poli `$params` v metodě `createObject($params)` objektu *StatsBlock*,

„Nenalezena třída zobrazení ‘XZ‘ u objektu #ID“ – zobrazení daného bloku *StatsBlockObject* je nastaveno na neregistrované zobrazení; chybu vrátí metoda `loadViewObject()` a program nadále pokračuje pod výchozím zobrazením,

„Nenalezena datová třída XY“ – totožná metoda jako u předchozího upozornění, jedná se však o datovou třídu, program tedy nenačte žádná data,

„Země ‘XY‘ není definována!“ – vyčítání zemí ze třídy *StatsCountry* metodou `get($countryCode)` nemůže najít požadovanou zemi pod zadaným kódem ve statickém poli `$countryList` třídy, je tedy nutné přidat do databáze novou zemi podle popisu v Třída StatsCountry.

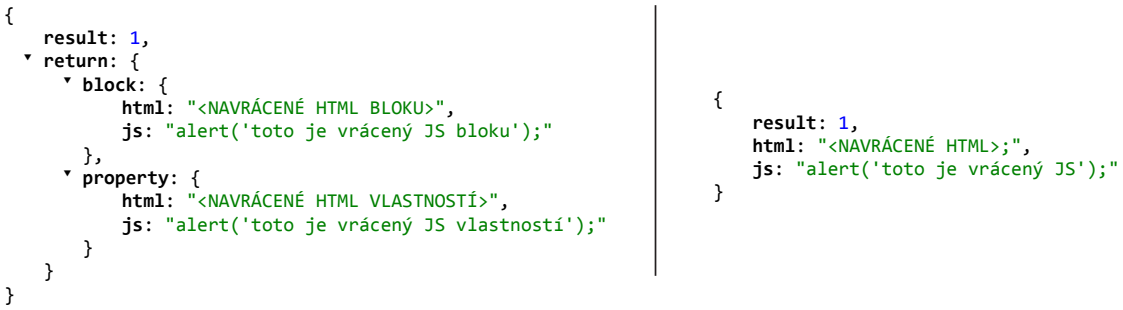
6.2 AJAX komunikace

Komunikace se serverem probíhá pomocí AJAX technologie a datové struktury JSON. Každý požadavek, který je odeslán na server, nese v URL adrese parametry pro kategorizaci, jako jsou *action*, *category* a *subcategory* a veškerá data jsou zasílána metodou *POST*. Zprávu zachytí soubor *request.php*, v kořenovém adresáři modulu, a následně vyhodnotí. Odpovědi serveru, kromě zasílaných dat, vždy obsahují stav *result* nabývající hodnot:

- 0 – požadavek se nezdařil,
- 1 – požadavek byl správně vykonán,
- 2 – neznámý požadavek, žádná akce se neprovedla.

Dalšími parametry odpovědi serveru jsou *html*, jehož HTML kód se načte do zvoleného HTML bloku, a *js*, zde jsou uloženy skripty pro vykonání na straně klienta po načtení.

V případě nastavení vlastností bloku je nutné vracet pozměněný obsah bloku a současně i seznam vlastností, protože předchozí změněná vlastnost může zpřístupnit nové vlastnosti. Vrací se tedy objekt *return* s klíči *block* a *property*, kde každý klíč obsahuje právě již dva zmíněné *html* a *js*.



Obrázek 20: Náhled JSON odpovědí ze serveru

6.3 Metody Before a After

Objekty a třídy vyvolávají události při dané akci. Tyto události můžeme rozdělit podle času volání:

- *before* – událost vyvolaná *před počátkem akce*,
- *after* – událost vyvolaná *po dokončení akce*.

Popišme si objekty a třídy společně s událostmi:

- *Property* – při volání metody `setValue()` vyvolá události `beforeChange()` a `afterChange()`,
- *StatsBlock* – v metodě `renderBlock()` vyvolá událost `beforeRender()` a `afterRender()`; v případě manipulace s bloky jsou při načítání volány události `beforeLoad()`, `afterLoad()`, při ukládání `beforeSave()`, `afterSave()` a při smazání `beforeRemove()` a `afterRemove()`,
- *StatsBlockObject* a *StatsView* – při zpracování dat a zobrazení v metodě `loadViewObject()` jsou volány `beforeDataLoad()` a `afterDataLoad()`,
- *StatsBlockProperty* – při vykreslování vlastností v metodě `renderProperty()` vytváří události `beforeRenderProperty()` a `afterRenderProperty()`,
- *StatsTemplate* – při vykreslení celé šablony pomocí metody `render()` nastanou události `beforeRender()` a `afterRender()`.

Vytvoření vlastních události se provádí zavoláním metody v určitém objektu. Je nutné nejprve zjistit, zda se volaná metoda v objektu nachází pomocí funkce `method_exists($object, $method_name)` a poté metodu zavolat.

```
if(method_exists($this, 'beforeDataLoad')){
  $this->beforeDataLoad();
}
```

Zdrojový kód 3: Vyvolání události

6.4 Rozšiřování modulu

Tato část je určena zejména pro pokročilé uživatele – programátory. Naučíme se v ní vytvářet nové bloky a statistiky a rozebereme si také konfiguraci souborů pomocí definovaných hlaviček v každém souboru.

Nesmíme však opomenout, při přetěžování rodičovských metod, tzn. vytváření stejnojmenných metod v dítěti jako v rodiči, volat zpětně rodičovskou metodu.

6.4.1 Hlavičky souborů pro správnou registraci

Bloky (Block) V každém bloku se v regionu `# <BLOCK DEFINE>` nacházejí konstanty definující chování každého bloku:

- `TYPE` – identifikátor, klíč daného bloku – nezapomeňme tedy, že musí být unikátní v celém modulu,
- `NAME` – zobrazovací jméno bloku,
- `ICON` – ikonka v panelu nástrojů, zde jsou psány CSS třídy komponenty FontAwesome (např. picture-o),
- `POSITION` – nastavení pozice zobrazení v panelu nástrojů,
- `DESCRIPTION` – rozšiřující popis funkčnosti pro uživatele,

- `SHOW_IN_LIST` – přepínač hodnot `TRUE` a `FALSE`, zda se bude blok zobrazovat v panelu nástrojů.

Zobrazení (View) Chování daného rozšíření se nastavuje definovanými konstantami v regionu `# <VIEW DEFINE>`:

- `TYPE` – identifikátor, unikátní v celém modulu,
- `KEY` – koncovka názvu rozšiřující třídy daného zobrazení,
- `NAME` – zobrazovací jméno,
- `MODE` – styl výpisu zobrazení pomocí konstant nadtřídy `parent::WRITABLE` nám říká, že výstup bude uložen do souboru; prostřednictvím konstanty `parent::VIEWABLE` bude výstup zobrazen jako HTML,
- `RENDER` – platí pouze v případě použití *StatsBlockObject*, kde určuje zda se do výstupu bude vypisovat HTML kód (`parent::STANDARD`) nebo značka (`parent::ALTERNATIVE`), která bude při výpisu nahrazena HTML kódem.

6.4.2 Vytváření bloků

Při vytváření nových bloků je nutné rozšířit třídu *StatsBlock* na podtřídu *statsBlockXYZ* a uložit do stejnojmenného souboru *statsBlock/statsBlockXYZ.class.php*. Registrace samotného bloku probíhá automaticky v *zavaděči* (viz. 4.1.1) při každém spuštění. Základní metody rozšířeného bloku jsou:

- konstruktor `__construct($params = array())`, kde jsou do vstupního parametru `$params` dosazeny uložené hodnoty vyčtené z databáze,
- metoda pro načtení vlastností `propertyLoad()`,
- metoda vracející HTML zobrazení bloku `_renderBlock()`,
- metoda `renderJS()`, která vrací doplňující JavaScripty.

6.4.3 Vytváření vlastních statistik

Statistiky Nová třída statistik je tvořena rozšířením třídy *StatsData* na podtřídu *statsDataXYZ* ve stejnojmenném souboru *statsData/statsDataXYZ.class.php*, registrace třídy probíhá též automaticky při každém spuštění v *zavaděči* (viz. 4.1.1).

V každé třídě může být neomezené množství statistik. Definice se provádí v metodě, která musí začínat prefixem *view* (např. `viewPocet_soutezicich()`). V případě, že potřebujeme statistické zkoumání pojmenovat s diakritikou a mezerami (protože to názvy metod neumožňují), využijeme statické pole `$methodList`, kde *klíčem* je název metody a *hodnota* reprezentuje vlastní název.
Popišme si tedy fáze metody:

1. Provedení SQL dotazu pomocí metody objektu `executeDataQuery($sql)`,
2. Vytvoření sloupců může proběhnout dvěma způsoby:
   (a) ručním přidáním jednotlivých sloupců do pole, kde každý element je objektem *StatsViewColumn*, s parametry v konstruktoru `$key` (název sloupce v databázi) a `$name` (zobrazovací název),
   (b) voláním metody `StatsViewColumn::fromArray($key_value)`, kde pole sloupců `$key_value` obsahuje v klíči název sloupce vyčtený z databáze a hodnota obsahuje zobrazovaný název statistiky,
3. Konfigurace sloupců – veškeré zmíněné proměnné jsou logické proměnné, nabývají tedy hodnoty `TRUE` a `FALSE`, pokud není uvedeno jinak. Ve výchozím stavu se zobrazí zaškrtávací pole umožňující zobrazit (popřípadě skrýt) hodnoty sloupce na výstupu. Zde je přehled vlastností sloupců:
   - Zobrazení v tabulkových výstupech
     - ▷ *display* – určuje, že bude sloupec označen jako zobrazen,
     - ▷ *displayNever* – v případě hodnoty `TRUE` se nezobrazí zaškrtávací pole.
   - Zobrazení v grafu
     - ▷ *plot* – určuje, že bude sloupec označen jako zobrazen,

- ▷ *plotNever* – v případě hodnoty `TRUE` se nezobrazí zaškrtávací pole,
- ▷ *plotOnly* – specifikace sloupců k zobrazení; zobrazí se pouze sloupce, kde `plotOnly = TRUE`, ostatním sloupcům se automaticky nastaví `plot = FALSE` a `plotNever = TRUE`.

- Zobrazení hodnot v procentech
  - ▷ *formatPercent* – hodnoty ve sloupci jsou formátovány jako procento tzn., že se na vstupu očekává desetinné číslo, které je vynásobeno číslem 100 a dodán znak %,
  - ▷ *percent* – veškeré hodnoty ve sloupci jsou přepočteny na procentuální zastoupení ze 100 %,
  - ▷ *floatDigits* – číselná hodnota určuje počet zobrazovaných desetinných míst číselných hodnot.
- Ostatní
  - ▷ *color* – barva datové řady určena hexadecimálně (#CCC), názvem barvy (black) nebo RGB složkou,
  - ▷ *order* – určuje číselné pořadí při zobrazení.

```
//  vytvoreni pole objektu sloupcu
  $arrColumns = StatsViewColumn::fromArray($header);

//  nastaveni parametru jednotlivych sloupcu
  $arrColumns['Pocet_divek']   -> setNumeric()
                               -> setPlotOnly();

  $arrColumns['Pocet_chlapcu']  -> setNumeric()
                                -> setPlotOnly();

  $arrColumns['Podil_divek']   -> setFormatPercent()
                               -> setPlotOnly();
```

Zdrojový kód 4: Konfigurace sloupců statistiky

Naplnění Pole třídy Doplnění nových dat do Pole se provádí v potomcích datové třídy Třída StatsData v metodě `&getFieldSQL()`, kterou můžeme přidat do nové i stávající datové třídy. Samotná metoda musí vracet SQL dotaz, kde názvy jednotlivých sloupců jsou zároveň i jejich klíče, pod kterými jsou definovány v *Poli*.

7 Závěr

Testování funkčnosti a správnosti celého modulu a vypočítaných hodnot probíhalo při generování výsledků soutěže 8. ročníku. Oba způsoby výpočtů byly prováděny souběžně a ověřovaly správnost.

Velkým přínosem a výhodou modulu je nejen usnadnění práce a ušetření času, ale i zamezení možným chybám při manuálním zpracovávání. Z připravených zpráv lze snadno vytvořit šablony a využívat je tak i pro další ročníky. Modifikováním již uložené šablony lze snadno docílit vytvoření zprávy v řádu minut. Díky interaktivním grafům je možné zobrazovat a skrývat jednotlivé datové řady v grafu a u každého bodu se při najetí myší objeví popisek s hodnotami jednotlivých datových řad a procentuálním zastoupením. Zajímavou možností je také detailní zobrazení konkrétního úseku grafu.

Mezi další výhody se řadí skutečnost, že jádro je plně otevřené novým modifikacím a je vytvořeno tak, aby bylo snadné jej využívat a rozšiřovat.

Celý modul byl otestován na moderních desktopových i mobilních webových prohlížečích Chrome 48, Opera 35, IE 11, Firefox 39 a dokonce jsou interaktivní grafy vykreslovány bez nejmenších problémů i na starších verzích prohlížeče Internet Explorer.

Použitá literatura azdroje

[1] AULDS, Charles. Linux - administrace serveru Apache. 1. vyd. Praha: Grada, 2003. Profesional. ISBN 80-247-0640-7.

[2] WELLING, Luke a Laura THOMSON. PHP a MySQL - rozvoj webových aplikací. 3rd ed. Praha: SoftPress, c2004, s. 161-164. ISBN 80-86497-83-6.

[3] NIXON, Robin. Learning PHP, MySQL & JavaScript: with jQuery, CSS & HTML5. 4th edition. 1005 Gravenstein Highway North, Sebastopol, CA, 95472.: O'Reilly Media, Inc., 2015. ISBN 978-1-49191-866-1.

[4] Úvod do PHP. Webtvorba.cz [online]. [cit. 2016-03-04]. Dostupné z: http://www.webtvorba.cz/php/uvod-do-php.html

[5] History of PHP. PHP: Hypertext Preprocessor [online]. [cit. 2016-03-04]. Dostupné z: http://php.net/manual/en/history.php.php

[6] PHP (1) – Historie a budoucnost. Linux Software. [online]. 27.5.2004 [cit. 2016-03-04]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=171

[7] 1. díl - MySQL krok za krokem: Úvod do MySQL a příprava prostředí [online]. [cit. 2016-03-27]. Dostupné z: http://www.itnetwork.cz/mysql/mysql-tutorial-uvod-a-priprava-prostredi

[8] MySQL 5.7 Reference Manual: History of MySQL. MySQL 5.7 Reference Manual [online]. [cit. 2016-03-04]. Dostupné z: http://dev.mysql.com/doc/refman/5.7/en/history.html

[9] MACDONALD, Matthew. HTML5: The Missing Manual. 2nd Edition. 1005 Gravenstein Highway North, Sebastopol, CA, 95472.: O'Reilly Media, Inc., 2014. ISBN 978-1-44936-326-0.

[10] 1. díl - Úvod do JavaScriptu [online]. 2016 [cit. 2016-04-08]. Dostupné z: http://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk

[11] The XMLHttpRequest Object [online]. [cit. 2016-03-25]. Dostupné z: http://www.w3schools.com/xml/dom_http.asp

[12] XML pro web aneb od teorie k praxi, 14.díl - XmlHttpRequest [online]. 2003 [cit. 2016-03-25]. Dostupné z: http://www.zive.cz/clanky/xml-pro-web-aneb-od-teorie-k-praxi-14dil—xmlhttprequest/sc-3-a-110710/default.aspx

[13] Introducing JSON [online]. [cit. 2016-04-04]. Dostupné z: http://www.json.org/

[14] YAML: Serializační formát pro ukládání dat - Zdroják [online]. 2009 [cit. 2016-03-22]. Dostupné z: https://www.zdrojak.cz/clanky/yaml-serializacni-format-pro-ukladani-dat/

[15] Interactive JavaScript charts for your webpage [online]. [cit. 2016-04-04]. Dostupné z: http://www.highcharts.com/

[16] PHPExcel: tabulky jednoduše [online]. 2009 [cit. 2016-03-23]. Dostupné z: https://www.zdrojak.cz/clanky/phpexcel-tabulky-jednoduse/

[17] Highlight.js: Syntax highlighting for the Web [online]. 2016 [cit. 2016-03-22]. Dostupné z: https://highlightjs.org/

[18] TOMCSÁNYI, P. Náročnosť úloh v súťaži Informatický bobor. In: Konferencia DidInfo 2009. - Banská Bystrica : Univerzita Mateja Bela, 2009. – [nestr.]. – ISBN 978-80-8083-720-4.

[19] PHP triky - Bitové operace [online]. [cit. 2016-04-01]. Dostupné z: http://php.vrana.cz/bitove-operace.php

Seznam zkratek

ACID – Atomicity, Consistency, Isolation, Durability

AJAX – Asynchronous JavaScript and XML

BSD3 – Berkeley Software Distribution verze 3

CMS – Content Management System

CSS – Cascading Style Sheets

CSV – Comma-separated values

DOM – Document Object Model

ECMA – European Computer Manufacturers Association

GPL – GNU General Public License

FI – Form Interpreter

FTP – File Transfer Protocol

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

HTTPd – HTTP Daemon

IE – Internet Explorer

INI – Inicialization

JPG/JPEG – Joint Photographic Experts Group

JS – JavaScript

JSON – JavaScript Object Notation

LGPL – GNU Lesser General Public License

MIT – Massachusetts Institute of Technology

MySQL – My Structured Query Language

NCSA – National Center for Supercomputing Applications

OOP – objektově orientovaný programování

PL/pgSQL – Procedural Language/PostgreSQL

PDF – Portable Document Format

PHP – PHP: Hypertext Preprocessor (dříve Personal Home Page)

PNG – Portable Network Graphics

POX – Plain Old XML

SGML – Standard Generalized Markup Language

SQL – Structured Query Language

SVG – Scalable Vector Graphics

URL – Uniform Resource Locator

VML – Vector Markup Language

XLS – Microsoft Excel spreadsheet file

XML – Extensible Markup Language

YAML – YAML Ain't Markup Language

W3C – World Wide Web Consortium

WYSIWYG – What You See Is What You Get

WWW – World Wide Web

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Přílohy

Přiloha 1: Přehled jednotlivých tříd a jejich závislostí

Obsah přiloženého CD

Přiložené CD obsahuje následující soubory:

1. Text této práce v souboru `tichy_bakalarska-prace.pdf`
2. Kompletní zdrojové kódy modulu v adresáři `modul_zdrojove_kody`
3. Databázové struktury tabulek v souboru `modul_databaze.sql`
4. Vygenerovaná dokomentace v adresáři `modul_dokumentace`
5. Přehled jednotlivých tříd a jejich závislostí v souboru `modul_tridy.pdf`.
6. Vlastní pluginy *BebrasStats* a *jQueryLoader* do CMS Joomla v adresáři `joomla_plugins`