VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

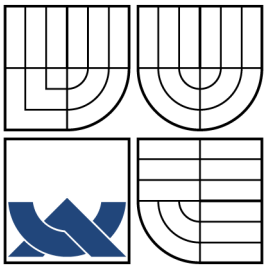
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

REALIZACE WEBOVÉHO MODULÁRNÍHO SYSTÉMU

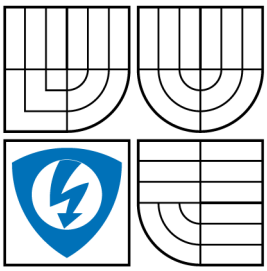
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. PETR SUCHÝ
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

REALIZACE WEBOVÉHO MODULÁRNÍHO SYSTÉMU

IMPLEMENTATION OF MODULAR WEB BASED APPLICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. PETR SUCHÝ
AUTHOR

VEDOUCÍ PRÁCE — ING. TOMÁŠ PELKA
SUPERVISOR

BRNO 2008

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ZDE VLOŽIT PRVNÍ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ZDE VLOŽIT DRUHÝ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ABSTRAKT

Diplomová práce „Realizace webového modulárního systému“ pojednává o webových modulárních systémech. Webové modulární systémy mají za cíl umožnit vytvářet univerzální webové prezentace bez nutnosti vytváření nových systémů. Diplomová práce je rozčleněna do několika částí. V úvodu práce jsou vyzdviženy výhody modulárního řešení webových systémů a posléze je analyzován současný stav dané problematiky. V další části je navržen vlastní webový modulární systém, především jeho jádro, moduly a datový model. V závěrečné části je popsána realizace vlastního navrženého systému. Největší důraz je kladen na modularitu a na bezpečnost systému. V závěru je zhodnoceno především vlastní řešení a jeho největší výhody.

KLÍČOVÁ SLOVA

webový modulární systém, modularita, bezpečnost, univerzálnost, implementace, datový model, PHP, databáze, MySQL, publikační systém, CMS, redakční systém, RS, systém pro správu dokumentů, DMS

ABSTRACT

Diploma thesis „Implementation of modular web based application“ deals with web modular systems. The aim of web modular system is to allow creating universal web presentation without necessity of creation new systems. Diploma thesis is devided into couple parts. Advantages of modular solution of web systems are highlighted in the first part and after that is analyzed the current situation of this problem. In the next part is designed own web modular system above all its kernel, modules and data model. In the final part is described own implementation of own designed system. The most stress is layed on security of the system. In the conclusion is summed up own solution and its most advantages.

KEYWORDS

web modular system, modularity, security, universality, implementation, data model, PHP, database, MySQL, content management system, CMS, document management system, DMS

SUCHÝ P. *Realizace webového modulárního systému*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2008. 92 s., 3 s. příloh. Vedoucí diplomové práce Ing. Tomáš Pelka.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Realizace webového modulárního systému“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Tímto bych rád poděkoval vedoucímu diplomové práce panu Ing. Tomáši Pelkovi za pomoc, cenné připomínky a čas, který věnoval mé diplomové práci.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM UKÁZEK ZDROJOVÝCH KÓDŮ

ÚVOD

Tato práce se věnuje návrhu a realizaci webového modulárního systému pro univerzální použití. Cílem a výstupem práce má být především návrh datového modelu, který musí splňovat požadované funkce a vlastnosti a samotná realizace navrženého systému a jeho základních modulů. Největší důraz musí být kladen na modularitu systému, jeho zabezpečení a snadnou správu. Především na tyto vlastnosti musí být brán ohled právě už v návrhu datového modelu systému, který musí být pro splnění těchto vlastností plně optimalizován a připraven.

Webový systém

Webový systém je systém, který je dostupný přes webové rozhraní pomocí protokolu HTTP (Hypertext Transfer Protocol), nejčastěji pomocí softwaru v podobě webového prohlížeče.

Modularita

Webové systémy můžeme rozdělit na modulární a nemodulární. Základní rozdíl spočívá v tom, že modulární systém obsahuje jádro, které umožňuje načítání libovolného množství modulů a tím i rozšíření funkcí systému. Jádro zpracovává veškeré požadavky inicializované uživatelem a předává je dále samotným modulům ke zpracování. Moduly jsou samostatné části systému, které jsou načteny a obslouženy jádrem a umožňují uživateli nové možnosti a funkce. Podobnou myšlenkou jsou tvořeny například operační systémy založené na bázi UNIX (monolitické jádro s jadernými moduly).

Nemodulární neboli statický systém je systém, který je přesným opakem modulárního. Tedy systém má dané specifické funkce a při potřebě nových funkcí či změn funkcí stávajících musí být změněny potřebné soubory a funkce v rámci celého systému. Kód je tedy mnohem méně přehledný a špatně spravovatelný.

Z modulární logiky vyplývá velká výhoda v možnosti globální kontroly běhu systému a jeho chování. Dále je zde také výhoda v případě potřeby zásahu do funkcí systému ve smyslu změn pouze v samotných modulech. Jádro jako takové zůstává stále stejné (samozřejmě kromě jaderných změn např. v nové verzi systému). Tím je zajištěno, že změna jednoho modulu neovlivní modul jiný. To samé platí při chybě běhu či chybném zásahu do modulu, kdy celý systém zůstane netknut a chyba se projeví jen v daném modulu. Chyba je tedy velice rychle dohledatelná, snadno identifikovatelná a opravitelná.

Jednou z hlavních předností modulárního systému je možnost a jednoduchost přidání nových funkcí. Potřeba rozšíření systému je jednoduše uspokojitelná naprogramováním a doinstalováním modulu s požadovanými funkcemi. Velkou výhodou modularity je tedy uspokojení potřeb zákazníka konkrétními moduly zabezpečujícími požadované funkce. Tím jsou pro zákazníka zajištěny minimální finanční i časové náklady.

Modulární systém má tedy výhodu nejen v jednoduché rozšiřitelnosti, ale i v lepší, jednoduší a přehlednější správě systému jako takového. Další výhodou je jednoduché sestavení systému tzv. na „míru“ - zákazník platí skutečně jen za požadované funkce, tedy za odpovídající moduly.

1 SOUČASNÝ STAV

Nynější webové systémy jsou jak modulárního tak nemodulárního typu. Většina (především dále se rozvíjejících systémů) již nyní modulární je, nebo se snaží v modulární systém přejít. Těchto systémů je nyní mnoho, ovšem také mnoho z nich nemá modularitu zpracovanou zcela úplně a dobře. U mnoha systémů nefunguje modularita skutečně bez nutnosti zásahu programátora do jádra při změnách evidentně se týkajících jen nějaké logické části systému.

V modulárních systémech je velmi důležité sadu běžných instrukcí využívaných na mnoha místech různých modulů standardizovat v metody a funkce samotného jádra. Jinak často dochází k velké redundanci, což se projevuje i na zbytečném zvětšení datového objemu a tím i vyplývajícího zpomalení celého systému. Častým problémem dnešních modulárních systémů je i jakási volnost a nejednotnost, tím že nejsou např. stanovena dostatečně pevná a přesná pravidla pro tvorbu modulů a jejich programování. U většiny systémů není také jednoduché dohledat zdroj případné chyby. Tyto chyby nejsou většinou vůbec protokolovány a jsou spíše naopak i před hlavním administrátorem systému skryty.

Z modulárních webových systémů můžeme vyjmenovat např. český Open source redakční systém phpRS (dostupný pod GNU/GPL licencí), publikační systémy Morpheus, Mambo, MagicWebDrive, systémy pro správu dokumentů iProject, openEDMS, Altus PORTAL aj.[1] Mnoho těchto systémů je založeno na myšlence Open source (např. phpRS, Mambo atd.), jiné jsou naopak komerční (Morpheus, MagicWebDrive atd.).

U Open source systémů je pro modulární systémy velkou nevýhodou často to, že nové moduly může napsat v podstatě kdokoliv, jakýmkoliv stylem apod. Poté dochází často k rozšíření modulů či úprav, díky kterým se systém stává např. pomalejším, méně vyspělým a především často i méně bezpečným. Open source systémy se tomuto snaží předejít tzv. „certifikací“ modulů. Certifikát může modulu udělit většinou jen zkušený člen vývojového týmu systému, ale ani to často nezabrání jejich šíření (ať už bez certifikace či s neodhalenou chybou při procesu certifikace).

Naopak některé z vyjmenovaných komerčních systémů dosahují vysoké úrovně jak samotného zpracování, tak především vysoké vyspělosti v ohledu modularity. V tomto ohledu můžeme vyzdvihnout především systém Morpheus vyvinutý společností Westcom s.r.o. a systém MagicWebDrive vyvinutý společnosti iMagic s.r.o. Komerční systémy podobného rázu dnes již modulární být musí, alespoň pokud chtějí mít úspěch v dnešním světě Internetu. S každým novým zákazníkem není totiž nutné psát celý systém znovu, ale stačí použít již hotové moduly. Nutností je

[1] Více informací o redakčních a publikačních systémech viz. [6], o systémech pro správu dokumentů viz. [5].

pouze upravit veřejnou část webu a případně doprogramovat moduly na zakázku, které se pak lehce do systému zahrnou (doinstalují).

Nemodulární systémy v podstatě z dnešního Internetu pomalu ale jistě mizí, právě pro jejich zmíněné nevýhody a jejich těžkopádnou možnost rozšíření. Nemodulárními systémy se dají jednoduše řešit maximálně menší projekty, ale ani u nich není nikdy jistota toho, že nebude v budoucnu zapotřebí systém rozšířit. Modulárním systémem si především zákazník zajistí jistotu jednoduché rozšiřitelnosti pro rozličné funkce a potřeby, které se mohou objevit kdykoliv v budoucnosti. Samotný autor bude mít v budoucnu také méně práce s udržováním systému a s přizpůsobováním nabídky modulů (a tedy funkcí systému) aktuální poptávce trhu.

2 NÁVRH SYSTÉMU

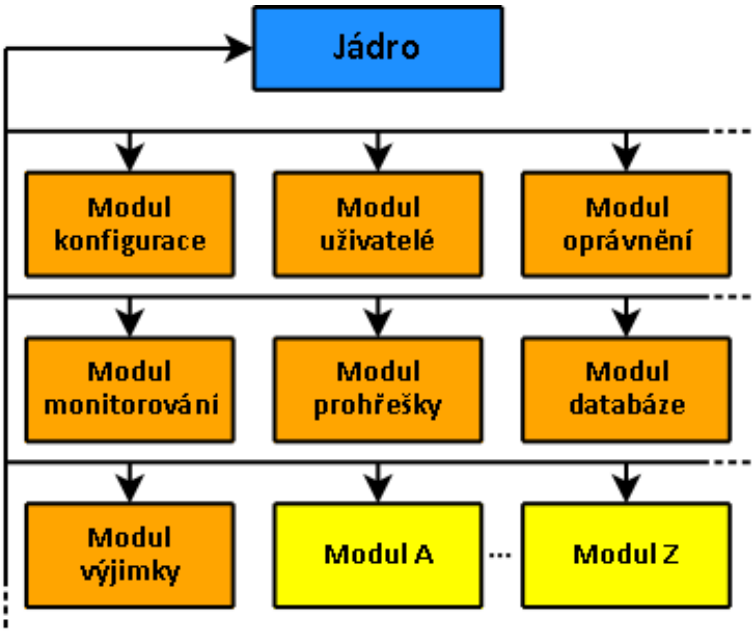
Systém bude dostupný přes webové rozhraní pomocí webových prohlížečů. Bude programován v jazyce PHP (Hypertext Preprocessor) pomocí objektově orientovaného programování. Veškerá textová data budou ukládána do databáze. Binární data budou ve většině případů ukládána na souborový server. Systém bude umožněno používat ve spolupráci s jakoukoliv běžně používanou databází, původní návrh bude zaměřen na databázi MySQL společnosti MySQL AB. Systému bude optimalizován pro provoz na webovém serveru Apache nadace Apache Software Foundation.

Systém musí být především velmi dobře zabezpečen proti neoprávněnému použití jak samotného systému, tak dat uložených jeho prostřednictvím. Toto zabezpečení bude zajišťovat především jádro ve spolupráci s uživatelským modulem. Musí být také odolný vůči útokům v podobě SQL Injection. Tento typ útoku se nejčastěji objevuje v podobě snahy o změnu dotazu na databázi provedenou útočníkem přes vstupní proměnné systému. Systém však musí být odolný i vůči jiným známým metodám útoků na webové systémy, jako např. XSS aj.

Systém bude rozdělen do dvou základních částí. Tyto části budou tvořit část systému přístupnou běžným návštěvníkům (frontend) a administrační část systému (backend). Frontend systému je tedy ta část, která tvoří veřejně přístupnou část systému bez potřeby vyžadování autentizace. Pokud je hovořeno o backendu, rozumí se tím administrační část systému, ke které je vyžadována autentizace. Každou z těchto částí bude obsluhovat jeden soubor z jádra systému, který dle požadavku (předaných parametrů v URL adrese) načte požadovaný modul a předá mu veškeré informace ke zpracování. Frontendová část bude optimalizována dle pravidel Search Engine Optimization (SEO), tedy bude mít optimalizovány URL adresy pro snadné zapamatování a lepší zaindexování vyhledávajícími stroji. K této optimalizaci bude využíván modul serveru Apache *mod_rewrite*, který právě umožňuje tzv. přepis adres např. vytvářením virtuálních adresářů.

Jak bylo již zmíněno, základ systému bude tvořit jádro a základní neodinstalovatelné moduly. Tato základní struktura systému bude představovat plnohodnotný webový modulární systém schopný nezávislého běhu. Pomocí dalších modulů bude určeno zaměření a funkce systému. Toto zaměření může být díky modularitě naprosto libovolné a bude tedy umožněno vytvořit doinstalováním potřebných modulů např. redakční systém, publikační systém (CMS - Content Management System), systém pro správu dokumentů (DMS - Document Management System), ale i jakýkoliv jiný libovolný systém dle přání a požadavků zákazníka. Systém bude moci být používán jako veřejný na Internetu, tak i jako extranetový či intranetový. Rozdělení systému na veřejný, extranetový či intranetový bude umožněno i v rámci jednoho systému právě pomocí modulu, modulů a jiného systémového nastavení. Základní

architektura systému je naznačena na obrázku 2.1.



Obr. 2.1: Architektura systému

Modrou barvou je zvýrazněno jádro systému. Dále je naznačena komunikace s jednotlivými moduly, kde moduly zvýrazněné oranžovou barvou jsou neodinstalovatelné moduly systému (takzvané jaderné moduly) a žlutou jsou naznačeny další rozšiřující moduly, kterých může být neomezené množství.

Jako první bude tedy navrženo jádro se všemi funkcemi potřebnými pro chod systému a obsluhu modulů. V další části budou popsány jaderné moduly systému. Veškeré uvedené informace jsou pouze základními pilíři při samotném vývoji. Samozřejmě návrh může být upraven při zjištění nevhodnosti nějaké volby či vhodnosti volby jiné.

Backend i případný frontend systému bude tvořen uživatelsky přívětivým prostředím pro snadnou správu obsahu a orientaci. Backend bude ve svém základu funkčně vždy stejný (změna designu je samozřejmostí), frontend bude navrhován v interakci se zákazníkem nebo podle potřeb konkrétního projektu.

Dále bude v systému umožněna plná multijazyčnost a to jak v backendu tak i ve frontendu systému. Multijazyčnost bude řešena pro celý backend globálně a ve frontendových částech dle daných pravidel individuálně. Multijazyčnost pro frontendové části webu načítané z databáze bude řešena vytvořením příslušných multijazyčných sloupců v tabulkách, kde tomu bude zapotřebí. Toto řešení multijazyčnosti je nejjednodušší a z hlediska implementace a nároků na databázi i nejlepší. V případě řešení multijazyčnosti pomocí více databází nebo více tabulek (pro každý jazyk jedna databáze nebo tabulka) se musí především pro potřeby vyhledávání vytvářet

tzv. pohledy, složitější dotazy v podobě napojování multijazyčných tabulek aj. Tyto složitější konstrukce značně komplikují práci s multijazyčným webem, jeho správou a ani programově není toto řešení vhodnější (co se rychlosti týče či zásahu do struktury databáze), i když dle pravidel datového návrhu správnější. Pokud bude daná realizace systému multijazyčná, bude volba jazykové verze umožněna při přihlášení (v případě backendu) a změna volby při práci se systémem (jak v backendu, tak ve frontendu).

2.1 Jádro

Jádro systému bude obsluhovat požadavky, které bude dále předávat ke zpracování jednotlivým modulům. Umožní také ve spolupráci se základními moduly kontrolu běhu a stavu systému včetně kontroly prací se systémem z uživatelského hlediska. S neodinstalovatelnými moduly bude představovat ucelenou jednotku schopnou nezávislého provozu bez ohledu na zaměření a další moduly systému.

2.1.1 Jaderné třídy

Funkce jádra budou zajišťovat především základní programové objektové třídy:

- Pro komunikaci s databází,
- pro kontrolu přihlašování a načítání informací a přihlášeném uživateli,
- pro načtení požadovaného modulu a informací o něm,
- základních všeobecných funkcí systému,
- pro odesílání emailů,
- session handleru,
- filtrování,
- layoutu,
- pro zachycení a zaznamenání výjimek.

Tyto třídy bude načítat třída samotného jádra. Toto stromové načítání bude prováděno z důvodu jednoduchého vypsání ladících informací o aktuálním běhu a stavu systému.

Třída pro komunikaci s databází

Třída pro komunikaci s databází bude zajišťovat komunikaci s databázovým serverem. SQL dotazy na databázový server nebudou psány v kódu systému a modulů jako takové, ale budou skládány pomocí metod této třídy jádrem systému. Tento přístup zajistí možnost kontroly vstupu a jiné výhody ať už z bezpečnostního hlediska nebo z hlediska usnadnění práce s daty uloženými v databázi. Tato třída bude tvořena na základě vytvořeného interface, nebo-li rozhraní. Použití rozhraní v objektově orientovaném programování zajistí vytvoření všech potřebných metod pro práci s daty a zamezí tak možnosti chyby již v samotné třídě (chybějící metoda, špatně implementovaná metoda apod.).

Třída pro kontrolu přihlášení a načítání informací o přihlášeném uživateli

Tato třída bude zajišťovat bezpečné přihlášení do administrace, tedy vytvoření relace s uživatelem, její udržení a také ukončení. Třída též umožní načtení informací o uživateli včetně přístupových oprávnění k modulům a k akcím v systému a včetně náležitostí ke skupinám uživatelů.

Třída pro načtení požadovaného modulu a informací o něm

Bude zajišťovat předání požadavku ke správnému modulu, načtení modulu a zprostředkování zpracování informací daným modulem. Také budou načítány veškeré informace o modulech, které budou uloženy v databázi.

Třída základních všeobecných funkcí systému

Bude obsahovat veškeré běžně používané funkce v celém systému i modulech. Jedná se o funkce zajišťující práci se soubory, bezpečnostní funkce, ale i běžné funkce jako stránkování, řazení, výpis dat ve stromové struktuře aj.

Třída pro odesílání emailů

Třída bude sloužit k odesílání veškerých emailů generovaných v systému, a to v backendu i frontendu. Bude umožňovat posílat textové emaily, emaily v HTML značkovacím jazyce i jejich různé kombinace. Samozřejmostí bude i možnost přikládání příloh a jiné.

Třída Session Sandleru

Tato třída bude obsluhovat session relace s uživateli. Session budou ukládány do databáze pro možnost jejich jednoduché vlastní správy a umožnění větší

bezpečnosti session relace. Ukládání session do databáze bude zajištěno vlastním Session Handlerem inicializovaným jádrem systému.

Třída pro obsluhu výjimek

Třída bude obsluhovat veškeré výjimky, které mohou nastat při běhu systému. Všechny rizikové části programového kódu budou zabezpečeny vyvoláním těchto výjimek, aby byla zajištěna bezpečnost dat, jejich konzistence a minimalizace škod způsobených případnými chybami. Takovéto odchycení výjimek umožní okamžité zastavení vykonávání skriptu vyvolávajícího chybu, zaprotokolování této chyby a její vypsání na obrazovku. U každé chyby bude zaznamenáno místo chyby a kód který jí způsobil (soubor, řádek, přesná část kódu aj.). Tento přístup umožní snadné a rychlé odhalení a odstranění programových chyb. Pro zachycování výjimek bude využita a rozšířena výchozí třída `Exceptions`, která je dostupná v PHP verze 5.

2.1.2 Uživatelské úrovně rozlišovány jádrem systému

Uživatelé budou jádrem systému rozlišováni třemi základními úrovněmi:

- Supervizor,
- administrátor,
- běžný uživatel.

Supervizorem je zamýšlen hlavní administrátor celého systému. Supervizor bude mít vždy na všechno plná oprávnění a jeho oprávnění nebudou nikdy ani kontrolována. Jako jediný bude moci instalovat nové moduly do systému či odinstalovávat již nepotřebné, bude moci vstupovat do modulu konfigurace a do ostatních systémových modulů (jako moduly databáze, výjimek, prohřešků atd.). Dále bude mít možnost zobrazit si ladící (debugové) informace (viz. podkapitola 2.1.5). Uživatele s úrovní supervizor můžeme přirovnat k superuživateli v operačních systémech na bázi Linuxu (tzv. „root“).

Administrátor se od běžného uživatele liší především svým vyšším uživatelským levelem, což ho automaticky opravňuje k možnosti vstupu do více částí systému (do více modulů a jejich částí) a k provádění více akcí v systému.

Běžný uživatel je uživatel s nejnižší uživatelskou úrovní s možností přístupu do administrace systému.

2.1.3 Kontrola uživatelů a záznam prohřešků

K uživatelské části jádra bude patřit i kontrola práce uživatelů se systémem. V případě zaznamenání pokusu o neoprávněný vstup uživatele do některých částí systému či k některým datům bude prohřešek okamžitě zaznamenán. Uživatel bude upozorněn na zaznamenání prohřešku a protokol o daném prohřešku bude zaslán na administrační email systému. Tímto způsobem se lehce dohledá viník problému a supervizorovi je umožněno více kontrolovat uživatele při jejich práci se systémem.

2.1.4 Adresace (URL adresy)

V backendu budou veškeré parametry URL adresy šifrovány pro zabránění možnosti vstupu uživatele do URL adresy. Tím systém znemožní vstup uživatelům do jiných částí, než do kterých budou systémem skutečně odkazováni (tedy např. znemožní zobrazení dat, na která nebude mít uživatel dostatečná oprávnění nebo zabrání nežádoucím akcím zásahem do URL adresy). Parametry URL adresy budou šifrovány a dešifrovány jádrem systému naprosto automaticky. URL adresy jednotlivých částí systému budou šifrovány i na základě aktuální relace uživatele, tedy budou proměnné (stejná „stránka“ v backendu bude mít pokaždé jinou adresu - předávané parametry budou vždy jinak zašifrovány, pomocí jiného klíče). Toto šifrování budou mít možnost dočasně vypnout pro svou relaci pouze supervizoři z důvodu testování či jiné systémové práce.

Naopak ve frontendu systému budou veškeré URL adresy optimalizovány pro vyhledávače (SEO optimalizace). Tato optimalizace bude zajištěna ve spolupráci a s využitím rozšíření Apache serveru *mod_rewrite* (pomáhá SEO optimalizaci předáváním parametrů v podobě virtuálních adresářů).

Pro bezpečné předávání a zpracování vstupních proměnných `$_GET`, `$_POST`, `$_FILES` (které může ovlivnit uživatel) budou tyto proměnné zpracovány a „očištěny“ jádrem a programově se k nim bude moci přistupovat pouze přes metody a proměnné jádra. Stejně tak bude přistupováno ke stavovým proměnným jako `$_SESSION` a `$_COOKIE` a také k proměnné `$_SERVER`. Tyto proměnné (tzv. předdefinované proměnné) jsou totiž nejčastěji používány k útoku na webové systémy a k neoprávněnému přístupu k datům uloženým ve webových systémech.

2.1.5 Ladící mód

Jak již bylo zmíněno výše, ladící (debug) mód budou moci využívat pouze uživatelé úrovně supervizor. Ladící mód bude moci být zapnut pouze dočasně v rámci jedné relace. Ladící mód umožní vypsání obsahu všech tříd a systémových proměnných.

Umožní tedy supervizorovi např. testování, hledání části systému způsobující např. nějaký problém atd.

2.1.6 Přístup do backendu

Přístup do backendu bude umožněn pouze pomocí protokolu HTTPS. Tento protokol je bezpečnou verzí protokolu HTTP (Hypertext Transfer Protocol), který umožňuje bezpečný (šifrovaný) přenos dat přes Internet. Zajistí tedy šifrování veškeré komunikace mezi serverem a klientem (uživatelem).

K přístupu do backendu bude vyžadováno uživatelské jméno a heslo s definovanou minimální délkou, omezenou platností a požadovanou složitostí testovanou pomocí vlastních metod a funkcí. Při definici hesla bude požadováno dodržení několika základních zásad pro používání bezpečných hesel, a to:

- Minimální délka hesla,
- zabránění vytvoření hesla náchylného k slovníkovému útoku typu „brute force“ (tzv. útok hrubou silou),
- donucení k vytvoření hesla z odlišných znaků,
- donucení k vytvoření hesla složeného nejen z písmen nebo z číslic, ale minimálně z jejich kombinací, nejlépe i z kombinací speciálních znaků (např. !, @, #, $, %, ; &, *, různé závorky aj.),
- zabránění vytvoření hesla pomocí sekvence prázdných znaků (tzv. „white-space“, nebo-li netisknutelných znaků).

Veškerá hesla v systému budou zhašována pomocí hašovacího algoritmu *MD5* a *SHA-1*. *MD5* i *SHA-1* jsou jednosměrné, dostatečně rychlé hašovací algoritmy využívané často k hašování hesel, nebo ke kontrole integrity dat a souborů. Dnes jsou však již známé postupy k prolomení obou těchto algoritmů, a proto bude použito zesílení tohoto hašování pomocí vícenásobného hašování a případně pomocí tzv. „zasolení“ hesla. Kryptografická metoda „zasolení“ spočívá v přidání náhodného shluku bitů či znaků k výrazu reprezentujícímu tajnou informaci, většinou heslo. Tím je informace modifikována a její rozšifrování je v reálném čase bez znalosti použité metody velmi složité, často až nereálné (díky vysoké časové náročnosti). Tvorbu hesla bude samozřejmě zajišťovat ve spolupráci s jádrem uživatelský modul (správa uživatelských účtů).

Pokud vyprší uživateli platnost hesla, bude nucen vygenerovat si jiné. Doba platnosti hesla bude nastavena v backendu systému a bude konfigurovatelná uživateli s úrovní supervizor.

Dále bude omezen počet možných pokusů o přihlášení. Pokud se uživatel pokusí přihlásit pod existujícím uživatelským jménem, ale špatným heslem, bude to u uživatelského účtu zaznamenáno. Další pokus bude moci být provedený až po vypršení určitého časového limitu. Tento limit se bude zvětšovat s počtem neúspěšných pokusů (např. 10 vteřin navíc po každém neúspěšném pokusu). Pokud bude neúspěšných pokusů více než definovaný počet v backendu systému (provedených za sebou bez úspěšného přihlášení), bude účet zablokován a jeho odblokování bude umožněno pouze podle informací odeslaných na email definovaný u daného účtu jako email uživatele. Na skutečnost zablokování uživatelského účtu bude vždy upozorněn i správce systému. Pokud se po neúspěšném pokusu o přihlášení uživatel úspěšně přihlásí, bude počet neúspěšných pokusů o přihlášení vynulován a taktéž bude vynulováno datum a čas posledního neúspěšného pokusu o přihlášení.

Pro přihlášení v backednu bude uskutečněna tzv. session relace. Třída jádra bude kontrolovat identitu daného uživatele. Přihlášení bude omezeno stanovenou dobou, po jejímž vypršení bude relace ukončena a tím uživatel odhlášen. Samozřejmě tato doba bude počítána jako doba, po kterou uživatel neprojevil jakoukoliv aktivitu (tedy nepracoval se systémem).

2.1.7 Monitorování

Systém bude monitorovat několik důležitých informací pro zajištění dohledání chyby či pokusu o napadení systému. Monitorovány budou úspěšné i neúspěšné pokusy o přihlášení uživatele, odhlášení uživatele, PHP chyby všech úrovní (chyby, varování, upozornění a další). Monitorovány budou moci být i veškeré provedené databázové dotazy. Tím se ovšem zdvojnásobí počet dotazů na databázi a proto bude tato možnost volitelná v nastavení systému. Dále budou zaznamenávána různá systémová hlášení a především budou také sledovány veškeré provedené uživatelské akce se systémem.

Všechny tyto zaznamenané informace bude mít možnost uživatel úrovně supervizor zobrazit v backendu a pracovat s nimi. I díky těmto záznamům bude možné zpětně dohledat viníka problému či nestandardního zásahu do systému.

2.2 Moduly

Moduly budou v backendu systému načítány jednotlivě dle požadavků uživatele. Každý modul bude muset splňovat programátorská pravidla pro schopnost interakce s jádrem systému (začlenění, administrace atd.). U každého modulu budou definovány jeho základní parametry, jako název, systémový název, minimální uživatelská úroveň pro umožnění vstupu aj. (viz. podkapitola 3.1.3).

Každý modul bude mít definovány uživatelské akce, ke kterým bude moci mít každý uživatel nastavené oprávnění k provedení dané akce. Pokud nebude mít uživatel toto oprávnění, nebude mít ani možnost se k dané akci dostat, natož jí provést. Každá akce bude mít definovánu položku menu modulu, která bude v době provádění akce označena jako aktivní. Každé provedení akce bude monitorováno (viz. podkapitola 2.1.7).

Každý modul bude mít také definovány položky menu. Jednotlivé položky menu budou přiřazené k akcím modulu - tedy daná položka bude sloužit jako odkaz k provedení požadované akce.

V jednotlivých modulech budou zobrazována data uložená v systému. Tato data bude možno libovolně řadit dle jednotlivých sloupců tabulek, z nichž budou data načítána a bude je také možné filtrovat snadno nastavitelným filtrem.

Pro optimalizaci jednotného přístupu k datům a jejich snadnou správu budou mít všechny tabulky identifikátor `id`, reprezentující primární klíč záznamů v daných tabulkách. Data, u nichž bude požadováno uživatelské řazení (např. moduly v systému), budou obsahovat v tabulce atribut `shift` typu `integer`, dle kterého bude možno data v databázi řadit a taktéž vypisovat.

2.2.1 Modul konfigurace

Modul konfigurace bude umožňovat konfigurování systému včetně modulů. Bude tedy zajišťovat systémovou konfiguraci, instalaci, editaci a odinstalaci modulů atd. Přístup do tohoto modulu bude striktně omezen pouze pro supervizora, protože bude umožňovat zásah do konfigurace ovlivňující celkové chování systému. Instalace modulů bude automatická a jednoduchá, provede se pouze odesláním příslušného formuláře.

Modul konfigurace umožní např. nastavit výchozí jazyk systému (každý uživatel si bude moci změnit), zda bude povolen ladící mód systému (nebude se jednat o jeho spuštění, pouze povolení ke spuštění), dále půjde zablokovat jak backend tak frontend (nezávisle na sobě), což může být vhodné nejen v případě hledání a odstranění problému, ale i při aktualizaci a jiných systémových pracích. Další konfigurační hodnoty budou i změna layoutu (celkového vzhledu webu), jeho název, povolení sledování systému (viz. podkapitola 2.1.7), systémový email (bude uveden jako odesílatel systémových emailů), email hlavního správce systému (pro hlášení prohřešků apod.) nastavení doby po jejímž vypršení bude uživatel automaticky odhlášen (pokud neprojeví po danou dobu jakoukoliv aktivitu), ale i spoustu dalších konfigurací, které budou přidávány při samotné realizaci a dalším vývoji dle zjištěných potřeb globální konfigurace.

Nastavení modulů v modulu konfigurace bude umožňovat nastavení minimálního uživatelského levelu pro samotný vstup do daného modulu, bude moci být i vypnuto kontrolování oprávnění ke vstupu do modulu (umožníme přístup do modulu všem uživatelům splňujícím podmínku požadovaného minimálního levelu), dále pořadí modulu (ovlivní pořadí ve složeném menu administrace - bude automaticky skládáno dle nainstalovaných modulů v tomto definovaném pořadí) a také bude moci být modul zablokován.

V případě zablokování backendu, frontendu či jednoho z modulů, budou jediní uživatelé úrovně supervizor moci dále využívat i tyto zablokované části systému. Zablokování bude, jak již bylo zmíněno, sloužit pro systémové úpravy apod.

2.2.2 Modul uživatelé

Modul uživatelů bude zajišťovat:

- Správu uživatelských účtů,
- správu uživatelských skupin.

Správa uživatelských účtů

Jedná se o hlavní funkci uživatelského modulu. Bude obsahovat vytvoření, editaci, blokaci a případné mazání uživatelských účtů všech úrovní. Uživatelé nebudou ale skutečně mazáni, budou pouze označováni jako smazaní. Tyto „smazané“ uživatele uvidí pouze uživatelé úrovně supervizor, pro ostatní budou uživatelé jakoby skutečně vymazáni. Bude tak činěno z bezpečnostních důvodů, protože k uživatelskému účtu bude přiřazeno spousta dat, včetně veškerých monitorovacích aj. záznamů.

Správa uživatelských skupin

V systému bude umožněno definovat libovolné množství uživatelských skupin. K těmto skupinám budou moci být nastavována stejná oprávnění jako k samotným uživatelům. Tento přístup je známý např. z operačních systémů a velmi usnadní správu uživatelských oprávnění i při větším počtu uživatelů. Do skupiny bude moci být přiřazeno libovolné množství uživatelů a každý uživatel bude moci být přiřazen do libovolného množství skupin (vztah *M:N*).

2.2.3 Modul oprávnění

Pro každý modul budou definována pravidla přístupu. Jádro systému bude při požadavku na modul kontrolovat, zda má daný uživatel k danému modulu dostatečné přístupové oprávnění. Toto oprávnění může získat jako samotný uživatel nebo jako

člen skupiny s dostatečným oprávněním. Toto oprávnění bude moci získat jen když bude uživatelská úroveň dostačující pro samotné umožnění vstupu do modulu (systémové moduly budou z většiny omezeny pouze pro vstup uživatelů úrovně supervizor, stejně tak mohou být omezeny jakéhokoliv jiné moduly).

Každý modul bude mít dále předem definované akce (jako např. pro modul uživatelů zobrazení uživatele, editace uživatele, přidání uživatele atd.) a k těmto akcím bude muset mít uživatel nastaveno oprávnění, jinak je nebude moci jakkoliv provést, ani se k nim dostat. Oprávnění k akcím půjde opět nastavit jak pro uživatele, tak pro uživatelské skupiny a taktéž bude moci být každá jednotlivá akce omezena na minimální požadovanou úroveň uživatele pro možnost přidělení daného oprávnění.

2.2.4 Modul monitorování

Supervizoři si budou moci pomocí tohoto modulu zobrazit veškeré zaznamenané záznamy pomocí monitorování jádra systému (viz. podkapitola 2.1.7).

2.2.5 Modul prohřešky

Modul prohřešky bude umožňovat uživatelům s úrovní supervizor zobrazovat zaznamenané prohřešky uživatelů (viz. podkapitola 2.1.3). Z těchto zaprotokolovaných prohřešků bude moci jednoduše dohledat viníka problému či pokus o neoprávněnou akci uživatele či napadení systému.

2.2.6 Modul databáze

Modul databáze bude opět pro uživatele uživatelské úrovně supervizor a bude sloužit pro správu databáze. Bude umožňovat jak zobrazení databázových tabulek, tak jejich sloupců, ale i jednotlivých záznamů. Dále bude umožněno jednoduše exportovat a tím tedy zálohovat aktuální databázi systému a to jak samotnou strukturu, samotná data, tak i obojí dohromady. Obdobně bude umožněn i import takto vytvořených záloh. Zálohy budou kompatibilní v rámci běžných SQL dotazů (obdoba exportu známého např. u aplikace phpMyAdmin, ale i z běžných správcovských nástrojů databázových serverů).

2.2.7 Modul výjimky

Modul výjimek bude obdoba modulů monitorování a prohřešků. Na rozdíl od předešlých dvou bude umožňovat supervizorům zobrazovat zaznamenané výjimky. Pomocí těchto výjimek bude každý případný problém v aplikaci lehce odhalitelný, naleznutelný a tím i odstranitelný (viz. podkapitola 2.1.1).

3 NÁVRH DATOVÉHO MODELU

Datový model systému bude navržen dle třech základních normálních forem (normálních forem je celkem pět) dle SQL 99. Tyto normální formy je doporučeno dodržovat pro bezproblémovou funkci databáze, její jednoduchou rozšiřitelnost a pro její použitelnost na různých databázových serverech. V některých případech se tyto normy nedodržují, zejména pokud je z vývojářského hlediska výhodnější normální formy nedodržet.

V této kapitole bude rozebrán celý datový model. Každá tabulka datového modelu bude vložena jako „screenshot“ z programu DBDesigner verze 4.0.5.6 Beta. Každá tabulka bude obsahovat sloupce *Column Name* (název sloupce), *Data Type* (datový typ), *NN* (Not Null - nenulový) a *AI* (Auto Increment - automaticky inkrementovaný). Unikátní klíče budou označeny vždy u popisu atributů jednotlivých tabulek.

Datový model bude popisován pro databázi MySQL. Všechny tabulky budou typu *MyISAM*, vzhledem k velkému rozvoji tohoto typu tabulek v posledních verzích MySQL. Především do budoucna mají tabulky typu *MyISAM* podporovat již všechny druhy relací, triggerů, které nyní podporují tabulky typu *InnoDB*. Celý datový návrh bude přiložen jako příloha v podobě obrázku formátu A3.

Všechny tabulky budou mít také prefix označující modul ke kterému patří. Tím budou tabulky přehledněji řazeny a ihned bude patrné, které ke kterému modulu patří.

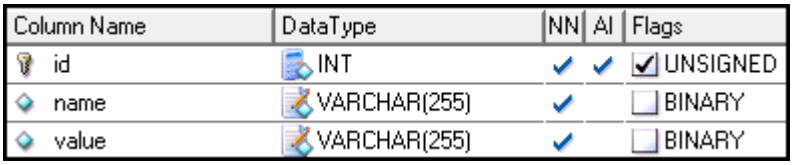
3.1 Jádro

3.1.1 Tabulka kernel_config

Tabulka `kernel_config` na obrázku 3.1 bude sloužit pro uložení globálního nastavení systému a na uložení globálních informací o systému. Uložené hodnoty budou vždy ve smyslu název - hodnota. Tato struktura tabulky umožní jednoduché doplnění konfiguračních hodnot a jejich zahrnutí do modulu konfigurace. V tomto modulu budou tyto hodnoty nastavitelné. Bude se jednat o konfigurační hodnoty rozdělené do logických částí:

- Nastavení serveru,
- nastavení systému,
- nastavení monitorování systému,
- nastavení emailů,

- nastavení webu,
- nastavení uživatelů,
- nastavení cookie,
- informace o systému.



Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
name	VARCHAR(255)	✓		☐ BINARY
value	VARCHAR(255)	✓		☐ BINARY

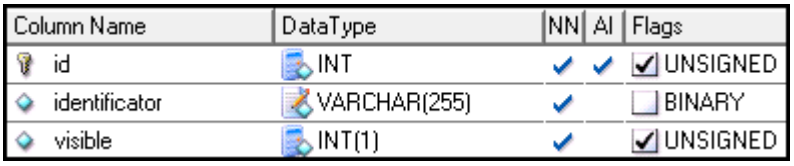
Obr. 3.1: Tabulka kernel_config

Tabulka `kernel_config` bude obsahovat následující atributy:

- `id` - identifikátor položky konfigurace, primární klíč,
- `name` - klíč položky konfigurace, unikátní klíč,
- `value` - nastavená hodnota položky konfigurace.

3.1.2 Tabulka kernel_layouts

Tabulka `kernel_layouts` (viz. obr. 3.2) bude k ukládání layoutů, které budou moci být v systému nastaveny a použity.



Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
identificator	VARCHAR(255)	✓		☐ BINARY
visible	INT(1)	✓		☑ UNSIGNED

Obr. 3.2: Tabulka kernel_layouts

Tabulka `kernel_layouts` bude obsahovat následující atributy:

- `id` - identifikátor layoutu, primární klíč,
- `identificator` - textový identifikátor layoutu, unikátní klíč,
- `visible` - zda má být layout viditelný (zda může být v systému použit).

3.1.3 Tabulka kernel_modules

Tabulka `kernel_modules` (viz. obr. 3.3) bude sloužit pro uložení informací o nainstalovaných modulech. Tyto moduly bude systém automaticky načítat a bude z nich vytvářet hlavní menu v administraci. Moduly bude umožněno pomocí modulu konfigurace jednoduše doinstalovat či odinstalovat. Vlastní základní nastavení modulů bude umožňovat opět modul konfigurace.

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
identificator	VARCHAR(255)	✓		☐ BINARY
system_name	VARCHAR(255)	✓		☐ BINARY
min_level	INT(1)	✓		☑ UNSIGNED
is_configurable	INT(1)	✓		☑ UNSIGNED
all_users	INT(1)	✓		☑ UNSIGNED
installed	INT	✓		☑ UNSIGNED
version	VARCHAR(255)	✓		☐ BINARY
shift	INT	✓		☑ UNSIGNED
blocked	INT(1)	✓		☑ UNSIGNED

Obr. 3.3: Tabulka kernel_modules

Tabulka `kernel_modules` bude obsahovat následující atributy:

- `id` - identifikátor modulu, primární klíč,
- `identificator` - textový identifikátor modulu pro předávání modulu v *URL* adrese, unikátní klíč,
- `system_name` - systémový název modulu, unikátní klíč,
- `min_level` - minimální požadovaná úroveň uživatele pro umožnění vstupu do modulu,
- `is_configurable` - zda bude umožněno u daného modulu měnit hodnotu minimálního uživatelského levelu potřebného pro vstup do modulu a zda bude možno povolit přístup všem uživatelům splňujícím podmínku minimálního požadovaného levelu bez ohledu na nastavená oprávnění,
- `all_users` - umožnění vstupu všem uživatelům s minimální a vyšší požadovanou uživatelskou úrovní nezávisle na nastavených oprávněních,
- `installed` - datum a čas nainstalování modulu do systému,
- `version` - verze nainstalovaného modulu,

- `shift` - pořadí modulu v hlavním menu backendu,
- `blocked` - integer 0/1 ve funkci boolean - zda je modul blokován.

3.1.4 Tabulka kernel_modules_actions

Tabulka `kernel_modules_actions` (viz. obr. 3.4) bude sloužit pro uložení akcí, které budou definovány v jednotlivých modulech. Těmito akcemi je rozuměna jakákoliv manipulace s daty, ve smyslu zobrazení, uložení, přidání či mazání. Tyto akce se k jednotlivým modulům přiřadí ihned při instalaci modulu a bude umožněno nastavovat oprávnění k těmto jednotlivým akcím jak uživatelům (tabulka `permissions_modules_actions_users` - viz. podkapitola 3.3.3), tak k uživatelským skupinám (tabulka `permissions_modules_actions_groups` - viz. podkapitola 3.3.4).

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_module	INT	✓		☑ UNSIGNED
id_menu	INT	✓		☑ UNSIGNED
identificator	VARCHAR(255)	✓		☐ BINARY
related_actions	TEXT			
can_assing	INT(1)	✓		☑ UNSIGNED
min_level	INT(1)	✓		☑ UNSIGNED
is_configurable	INT(1)	✓		☑ UNSIGNED
all_users	INT(1)	✓		☑ UNSIGNED

Obr. 3.4: Tabulka kernel_modules_actions

Tabulka `kernel_modules_actions` bude obsahovat následující atributy:

- `id` - identifikátor akce, primární klíč,
- `id_module` - identifikátor modulu, cizí klíč z tabulky `kernel_modules` (viz. podkapitola 3.1.3),
- `id_menu` - identifikátor položky menu, která bude zvýrazněna při provádění dané akce, cizí klíč z tabulky `kernel_modules_menu` - viz. podkapitola 3.1.5,
- `identificator` - textový identifikátor dané akce, unikátní klíč,
- `related_actions` - související akce - akce, ke kterým bude přiděleno oprávnění při přidělení oprávnění k dané akci (bude se jednat o identifikátory daných akcí oddělených znakem „:“),

- `can_assign` - integer 0/1 ve funkci boolean hodnoty - zda může být oprávnění k dané akci vůbec přiřazeno (některé akce nebudou znamenat provedení nějaké změny či zobrazení nějakých dat, bude se jednat např. jen o položku v menu - přidělování oprávnění by bylo zbytečné),
- `min_level` - minimální uživatelský level požadovaný pro oprávnění k provedení dané akce,
- `is_configurable` - integer 0/1 ve funkci boolean hodnoty, zda je možno změnit nastavení atributů `min_level` a `all_users`,
- `all_users` - oprávnění k provedení akce všem uživatelům s minimální a vyšší požadovanou uživatelskou úrovní nezávisle na nastavených oprávněních.

3.1.5 Tabulka kernel_modules_menu

Tabulka `kernel_modules_menu` (viz. obr. 3.5) bude sloužit k definování položek menu modulů v backendu systému. Bude se jednat o jednotlivé položky menu jednotlivých modulů, včetně vnoření a řazení.

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_parent	INT	✓		☑ UNSIGNED
id_module	INT	✓		☑ UNSIGNED
id_action	INT	✓		☑ UNSIGNED
identificator	VARCHAR(255)			☐ BINARY
need_id	INT(1)			☑ UNSIGNED
shift	INT			☑ UNSIGNED

Obr. 3.5: Tabulka kernel_modules_menu

Tabulka `kernel_modules_menu` bude obsahovat následující atributy:

- `id` - identifikátor položky menu, primární klíč,
- `id_parent` - identifikátor nadřazené položky menu (pro možnost zanoření), klíč odkazující na atribut `id`,
- `id_module` - identifikátor modulu, k němuž daná položka patří - cizí klíč z tabulky `kernel_modules` (viz. podkapitola 3.1.3),
- `id_action` - identifikátor akce, ke které bude daná položka odkazovat - cizí klíč z tabulky `kernel_modules_actions` (viz. podkapitola 3.1.4),
- `identificator` - textový identifikátor položky menu,

- `need_id` - zda daná položka menu potřebuje ke svému zobrazení v URL id nějaké položky,
- `shift` - řazení jednotlivých položek v rámci jednoho modulu a zanoření.

3.1.6 Tabulka kernel_sessions

Tabulka `kernel_sessions` (viz. obr. 3.6) bude použita k ukládání proměnných `$_SESSION` vlastním Session handlerem systému.

Column Name	DataType	NN	AI	Flags
id	VARCHAR(255)	✓		BINARY
time	INT	✓		☑ UNSIGNED
start	INT	✓		☑ UNSIGNED
value	TEXT	✓		

Obr. 3.6: Tabulka kernel_sessions

Tabulka `kernel_sessions` bude obsahovat následující atributy:

- `id` - identifikátor session relace (označovaný též `SID`), primární klíč,
- `time` - čas posledního obnovení session relace ve formátu *TIMESTAMP*,
- `start` - čas vytvoření session relace ve formátu *TIMESTAMP*,
- `value` - hodnota session (jednotlivé položky jsou oddělené a při výběru session z databáze se z nich vytvoří pole hodnot uložených pod daným *SID*).

3.2 Modul uživatelé

3.2.1 Tabulka users

Tabulka `users` (viz. obr. 3.7) bude sloužit pro uložení uživatelských účtů a základních informací o uživatelích, kteří budou mít umožněn přístup do backendu. V této tabulce bude výchozí záznam systémového uživatele s `ID` 1, který bude sloužit pro ukládání systémových informací. Tento účet bude ovšem blokován pro běžné přihlášení do administrace a bude označen jako vymazaný (uvidí ho pouze uživatelé s úrovní supervizor). Další uživatele bude moci přidat pouze administrátor s daným oprávněním nebo supervizor.

Tabulka `users` bude obsahovat následující atributy:

- `id` - primární klíč, identifikátor uživatele,

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
username	VARCHAR(255)	✓		☐ BINARY
user_password	VARCHAR(255)	✓		☐ BINARY
last_password_change	INT	✓		☑ UNSIGNED
level	INT(1)			☑ UNSIGNED
firstname	VARCHAR(255)	✓		☐ BINARY
surname	VARCHAR(255)	✓		☐ BINARY
email	VARCHAR(255)	✓		☐ BINARY
reg_date_time	INT	✓		☑ UNSIGNED
session	VARCHAR(255)	✓		☐ BINARY
ip_address	VARCHAR(39)	✓		☐ BINARY
hostname	VARCHAR(255)			☐ BINARY
user_agent	TEXT			
proxy_info	TEXT			
request_uri	TEXT			
http_referer	TEXT			
last_time	INT	✓		☑ UNSIGNED
last_login	INT	✓		☑ UNSIGNED
last_url	TEXT	✓		
last_login_attempt	INT			☑ UNSIGNED
count_login_attempts	INT(3)	✓		☑ UNSIGNED
blocked	INT(1)	✓		☑ UNSIGNED
blocked_reason	VARCHAR(255)			☐ BINARY
deleted	INT(1)	✓		☑ UNSIGNED

Obr. 3.7: Tabulka users

- `username` - uživatelské jméno, pod kterým se bude uživatel přihlašovat do administrace (nebude viditelné nikde ve frontendové části - pro zvýšení bezpečnosti), unikátní klíč,
- `user_password` - heslo uživatele uložené pomocí vícenásobného použití hašovací funkce *MD5* a *SHA-1* a případného zasolení hesla (viz. podkapitola 2.1.6),
- `last_password_change` - datum a čas poslední změny uživatelského hesla (pokud bude uplynulá doba od poslední změny delší než maximální povolená v administraci, bude uživatel muset své heslo změnit, jinak mu nebude umoněno se systémem pracovat),
- `level` - úroveň uživatele - 1 = běžný uživatel, 2 = administrátor, 3 = supervizor,
- `firstname` - jméno uživatele,

- `surname` - příjmení uživatele,
- `email` - email uživatele,
- `reg_date_time` - datum a čas registrace uživatele (vytvoření uživatelského účtu),
- `session` - aktuální session uživatele (pro udržení session relace s uživatelem, tedy jeho přihlášení), unikátní klíč,
- `ip_address` - IP adresa, odkud se uživatel naposledy přihlásil,
- `hostname` - DNS záznam k IP adrese, odkud se uživatel naposledy přihlásil,
- `user_agent` - informace o softwaru, který uživatel naposledy použil k přihlášení,
- `proxy_info` - informace o proxy, pokud uživatel přistupoval přes proxy (při posledním přihlášení),
- `request_uri` - poslední uživatelem vyžádaná URL adresa v systému,
- `http_referer` - poslední URL adresa, z které byl odkázán uživatel na poslední navštívenou,
- `last_time` - datum a čas poslední projevené aktivity uživatele,
- `last_login` - datum a čas posledního přihlášení,
- `last_url` - poslední URL adresy, které uživatel navštívil (pamatuje si i mezi relacemi, umožňuje tedy přesměrovat na poslední URL adresu z minulé relace a jednoduše tak uživateli umožnit pokračování v práci),
- `last_login_attempt` - datum a čas posledního neúspěšného pokusu o přihlášení (bude nastaveno na `NULL` v případě úspěšného přihlášení),
- `count_login_attempts` - počet neúspěšných pokusů o přihlášení (bude nastaveno na 0 v případě úspěšného přihlášení),
- `blocked` - zda je uživatelský účet blokován (pokud ano, není možno se pod tímto účtem přihlásit do systému),
- `blocked_reason` - důvod zablokování účtu,
- `deleted` - administrátorské nastavení, zda je uživatelský účet smazán (na uživatelský účet je navázáno spoustu dalších informací, proto je vhodné účet v systému jako takový ponechat, pouze ho označit za adminstrátorsky smazaný - tyto účty uvidí pouze uživatelé s uživatelskou úrovní supervizor).

3.2.2 Tabulka users_groups

Tabulka `users_groups` (viz. obr. 3.8) bude sloužit k ukládání uživatelských skupin. K těmto skupinám bude umožněno dále nastavovat i stejná oprávnění jako k samotným uživatelským účtům. Tento přístup velmi urychlí a usnadní přidělování oprávnění více uživatelům najednou. Umožní tedy vytváření libovolného množství uživatelských skupin složených z libovolného množství uživatelů.

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
title	VARCHAR(255)	✓		☐ BINARY
description	TEXT	✓		
added	INT	✓		☑ UNSIGNED

Obr. 3.8: Tabulka users_groups

Tabulka `users_groups` bude obsahovat následující atributy:

- `id` - identifikátor skupiny uživatelů, primární klíč tabulky,
- `title` - název uživatelské skupiny, unikátní klíč,
- `description` - popis uživatelské skupiny,
- `added` - datum a čas přidání uživatelské skupiny do systému,

3.2.3 Tabulka users_in_groups

Tabulka `users_in_groups` (viz. obr. 3.9) bude sloužit k přiřazení uživatelů do uživatelských skupin. Tabulka je *M:N* tabulkou mezi tabulkou uživatelů (`users` - viz. podkapitola 3.2.1) a tabulkou uživatelských skupin (`users_groups` - viz. podkapitola 3.2.2).

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_user	INT	✓		☑ UNSIGNED
id_group	INT	✓		☑ UNSIGNED

Obr. 3.9: Tabulka users_in_groups

Tabulka `users_in_groups` bude obsahovat následující atributy:

- `id` - identifikátor záznamu, primární klíč,

- `id_user` - identifikátor uživatele, cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `id_group` - identifikátor skupin uživatelů, cizí klíč z tabulky `users_groups` (viz. podkapitola 3.2.2).

3.2.4 Tabulka users_settings

Tabulka `users_settings` (viz. obr. 3.10) bude sloužit k ukládání uživatelských nastavení (viz. podkapitola 4.4.10).

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_user	INT	✓		☑ UNSIGNED
name	VARCHAR(255)	✓		☐ BINARY
value	TEXT	✓		

Obr. 3.10: Tabulka users_settings

Tabulka `users_settings` bude obsahovat následující atributy:

- `id` - identifikátor záznamu, primární klíč,
- `id_user` - identifikátor uživatele, cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `name` - název uživatelského nastavení,
- `value` - hodnota uživatelského nastavení.

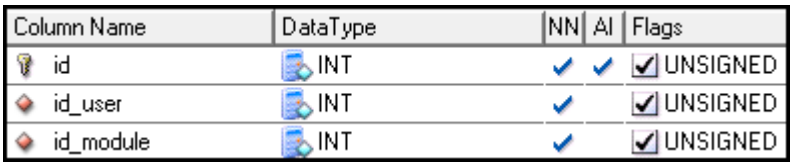
3.3 Modul oprávnění

3.3.1 Tabulka permissions_modules_users

Tabulka `permissions_modules_users` (viz. obr. 3.11) bude používána pro povolení přístupu uživatelů do modulů. Tabulka je *M:N* tabulkou mezi tabulkou uživatelů (`users` - viz. podkapitola 3.2.1) a tabulkou modulů (`kernel_modules` - viz. podkapitola 3.1.3).

Tabulka `permissions_modules_users` bude obsahovat následující atributy:

- `id` - primární klíč, identifikátor záznamu,
- `id_user` - identifikátor uživatele, cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),



Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	UNSIGNED
id_user	INT	✓		UNSIGNED
id_module	INT	✓		UNSIGNED

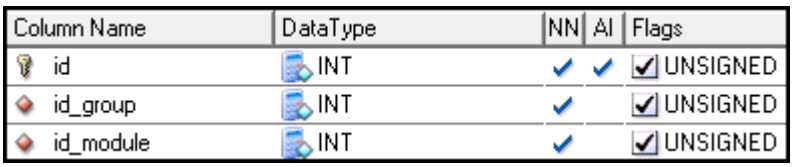
Obr. 3.11: Tabulka permissions_modules_users

- `id_module` - identifikátor modulu, cizí klíč z tabulky `kernel_modules` (viz. podkapitola 3.1.3).

Atributy `id_user` a `id_module` jsou složeným unikátním klíčem.

3.3.2 Tabulka permissions_modules_groups

Tabulka `permissions_modules_groups` (viz. obr. 3.12) bude používána pro povolení přístupu skupinám uživatelů do modulů. Tabulka je *M:N* tabulkou mezi tabulkou skupin uživatelů (`users_groups` - viz. podkapitola 3.2.2) a tabulkou modulů (`kernel_modules` - viz. podkapitola 3.1.3).



Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	UNSIGNED
id_group	INT	✓		UNSIGNED
id_module	INT	✓		UNSIGNED

Obr. 3.12: Tabulka permissions_modules_groups

Tabulka `permissions_modules_groups` bude obsahovat následující atributy:

- `id` - primární klíč, identifikátor záznamu,
- `id_group` - identifikátor skupiny uživatelů, cizí klíč z tabulky `users_groups` (viz. podkapitola 3.2.2),
- `id_module` - identifikátor modulu, cizí klíč z tabulky `kernel_modules` (viz. podkapitola 3.1.3).

Atributy `id_group` a `id_module` jsou složeným unikátním klíčem.

3.3.3 Tabulka permissions_modules_actions_users

Tabulka `permissions_modules_actions_users` (viz. obr. 3.13) bude sloužit pro přiřazení oprávnění uživatelům na akce prováděné v jednotlivých modulech. Tabulka

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	UNSIGNED
id_user	INT	✓		UNSIGNED
id_action	INT	✓		UNSIGNED

Obr. 3.13: Tabulka permissions_modules_actions_users

zajišťuje relaci *M:N* mezi tabulkou uživatelů (`users` - viz. podkapitola 3.2.1) a tabulkou akcí definovaných v modulech (`kernel_modules_actions` - viz. podkapitola 3.1.4).

Tabulka `permissions_modules_actions_users` bude obsahovat následující atributy:

- `id` - primární klíč, identifikátor záznamu,
- `id_user` - identifikátor uživatele, jemuž je přiděleno oprávnění - cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `id_action` - identifikátor akce, k níž je uživateli přiděleno oprávnění - cizí klíč z tabulky `kernel_modules_actions` (viz. podkapitola 3.1.4).

Atributy `id_user` a `id_action` jsou složeným unikátním klíčem.

3.3.4 Tabulka permissions_modules_actions_groups

Tabulka `permissions_modules_actions_groups` (viz. obr. 3.14) bude sloužit pro přiřazení oprávnění uživatelským skupinám na akce prováděné v jednotlivých modulech. Tabulka je *M:N* tabulkou mezi tabulkou skupin uživatelů (`users_groups` - viz. podkapitola 3.2.2) a tabulkou akcí definovaných v modulech (`kernel_modules_actions` - viz. podkapitola 3.1.4).

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	UNSIGNED
id_group	INT	✓		UNSIGNED
id_action	INT	✓		UNSIGNED

Obr. 3.14: Tabulka permissions_modules_actions_groups

Tabulka `permissions_modules_actions_groups` bude obsahovat následující atributy:

- `id` - primární klíč, identifikátor záznamu,

- `id_group` - identifikátor skupiny uživatelů, které je přiděleno oprávnění - cizí klíč z tabulky `users_groups` (viz. podkapitola 3.2.2),
- `id_action` - identifikátor akce, k níž je skupině uživatelů přiděleno oprávnění - cizí klíč z tabulky `kernel_modules_actions` (viz. podkapitola 3.1.4).

Atributy `id_group` a `id_action` jsou složeným unikátním klíčem.

3.4 Modul monitorování

3.4.1 Tabulka monitoring_login_logout

Tabulka `monitoring_login_logout` (viz. obr. 3.15) bude sloužit pro ukládání záznamů o přihlášeních a odhlášeních uživatelů. Budou zaznamenávána i neúspěšná přihlášení a důvod neúspěšného přihlášení.

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	UNSIGNED
id_user	INT	✓		UNSIGNED
record_type	INT(1)	✓		UNSIGNED
date_time	INT	✓		UNSIGNED
title	VARCHAR(255)	✓		BINARY
description	TEXT	✓		
request_uri	TEXT	✓		
http_referer	TEXT			
ip_address	VARCHAR(39)	✓		BINARY
hostname	VARCHAR(255)			BINARY
proxy_info	TEXT			
user_agent	TEXT			

Obr. 3.15: Tabulka monitoring_login_logout

Tabulka `monitoring_login_logout` bude obsahovat následující atributy:

- `id` - identifikátor záznamu, primární klíč,
- `id_user` - identifikátor uživatele, který byl v době zaznamenání přihlášen - cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `record_type` - typ záznamu, hodnota integer 0 (odhlášení), 1 (přihlášení), 2 (neúspěšný pokus o přihlášení),
- `date_time` - datum a čas zaznamenání,
- `title` - název záznamu,

- `description` - popis zaznamenané činnosti,
- `request_uri` - požadovaná URL adresa,
- `http_referer` - URL adresa odkazující stránky,
- `ip_address` - IP adresa uživatele,
- `hostname` - DNS záznam k IP adrese uložené v atributu `ip_address`,
- `proxy_info` - informace o proxy, pokud byl přes ní uživatel připojen,
- `user_agent` - informace o softwaru, pomocí kterého uživatel pracoval se systémem.

3.4.2 Tabulka monitoring_php

Tabulka `monitoring_php` (viz. obr. 3.16) bude sloužit pro ukládání záznamů o chybách, varováních a jiných informacích o běhu PHP. Bude tedy sloužit jako úložiště PHP Error Handleru.

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_user	INT	✓		☑ UNSIGNED
severity	VARCHAR(255)	✓		☐ BINARY
message	TEXT	✓		
filename	VARCHAR(255)	✓		☐ BINARY
line_num	INT	✓		☑ UNSIGNED
date_time	INT	✓		☑ UNSIGNED
request_uri	TEXT	✓		
http_referer	TEXT			
ip_address	VARCHAR(39)	✓		☐ BINARY
hostname	VARCHAR(255)			☐ BINARY
proxy_info	TEXT			
user_agent	TEXT			

Obr. 3.16: Tabulka monitoring_php

Tabulka `monitoring_php` bude obsahovat následující atributy:

- `id` - identifikátor záznamu, primární klíč,
- `id_user` - identifikátor uživatele, který byl v době zaznamenání přihlášen - cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `severity` - vážnost zaznamenané chyby,

- `message` - zpráva o chybě,
- `filename` - název souboru v němž došlo k chybě,
- `line_num` - číslo řádku, který způsobil chybu,
- `date_time` - datum a čas zaznamenání,
- `request_uri` - požadovaná URL adresa,
- `http_referer` - URL adresa odkazující stránky,
- `ip_address` - IP adresa uživatele,
- `hostname` - DNS záznam k IP adrese uložené v atributu `ip_address`,
- `proxy_info` - informace o proxy, pokud byl přes ní uživatel připojen,
- `user_agent` - informace o softwaru, pomocí kterého uživatel pracoval se systémem.

3.4.3 Tabulka monitoring_sql

Tabulka `monitoring_sql` (viz. obr. 3.17) bude sloužit pro ukládání záznamů o všech systémem provedených SQL dotazech. Zaznamenávání všech SQL dotazů bude ale dosti výkonově náročné (dvojnásobný počet dotazů na databázi) a proto bude tato možnost pouze volitelná.

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_user	INT	✓		☑ UNSIGNED
date_time	INT	✓		☑ UNSIGNED
sql_query	TEXT			
request_uri	TEXT	✓		
http_referer	TEXT			
ip_address	VARCHAR(39)	✓		☐ BINARY
hostname	VARCHAR(255)			☐ BINARY
proxy_info	TEXT			
user_agent	TEXT			

Obr. 3.17: Tabulka monitoring_sql

Tabulka `monitoring_sql` bude obsahovat následující atributy:

- `id` - identifikátor záznamu, primární klíč,

- `id_user` - identifikátor uživatele, který byl v době zaznamenání přihlášen - cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `date_time` - datum a čas provedení SQL dotazu,
- `sql_query` - provedený SQL dotaz,
- `request_uri` - požadovaná URL adresa,
- `http_referer` - URL adresa odkazující stránky,
- `ip_address` - IP adresa uživatele,
- `hostname` - DNS záznam k IP adrese uložené v atributu `ip_address`,
- `proxy_info` - informace o proxy, pokud byl přes ní uživatel připojen,
- `user_agent` - informace o softwaru, pomocí kterého uživatel pracoval se systémem.

3.4.4 Tabulka monitoring_system

Tabulka `monitoring_system` (viz. obr. 3.18) bude sloužit pro ukládání záznamů o systémových změnách a jiných definovaných akcích vhodných pro zaznamenání.

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_user	INT	✓		☑ UNSIGNED
date_time	INT	✓		☑ UNSIGNED
code	INT	✓		☐ UNSIGNED
title	VARCHAR(255)	✓		☐ BINARY
message	TEXT	✓		
request_uri	TEXT	✓		
http_referer	TEXT			
ip_address	VARCHAR(39)	✓		☐ BINARY
hostname	VARCHAR(255)			☐ BINARY
proxy_info	TEXT			
user_agent	TEXT			

Obr. 3.18: Tabulka monitoring_system

Tabulka `monitoring_system` bude obsahovat následující atributy:

- `id` - identifikátor záznamu, primární klíč,
- `id_user` - identifikátor uživatele, který byl v době zaznamenání přihlášen - cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),

- `date_time` - datum a čas provedení zaznamenání,
- `code` - číselný kód záznamu (každý systémový záznam bude mít své jedinečné označení),
- `title` - název systémového záznamu,
- `message` - zpráva o systémovém záznamu,
- `request_uri` - požadovaná URL adresa,
- `http_referer` - URL adresa odkazující stránky,
- `ip_address` - IP adresa uživatele,
- `hostname` - DNS záznam k IP adrese uložené v atributu `ip_address`,
- `proxy_info` - informace o proxy, pokud byl přes ní uživatel připojen,
- `user_agent` - informace o softwaru, pomocí kterého uživatel pracoval se systémem.

3.4.5 Tabulka monitoring_users

Tabulka `monitoring_users` (viz. obr. 3.19) bude sloužit pro zaznamenávání prací uživatelů se systémem. Budou zaznamenávány veškeré prováděné akce všemi uživateli systému. Díky tomu bude snadno dohledatelný například viník ztráty dat či autor nevhodné manipulace s daty.

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_user	INT	✓		☑ UNSIGNED
id_module	INT	✓		☑ UNSIGNED
id_action	INT	✓		☑ UNSIGNED
date_time	INT	✓		☑ UNSIGNED
title	VARCHAR(255)	✓		☐ BINARY
description	TEXT	✓		
request_uri	TEXT	✓		
http_referer	TEXT			
ip_address	VARCHAR(39)	✓		☐ BINARY
hostname	VARCHAR(255)			☐ BINARY
proxy_info	TEXT			
user_agent	TEXT			

Obr. 3.19: Tabulka monitoring_users

Tabulka `monitoring_users` bude obsahovat následující atributy:

- `id` - identifikátor záznamu, primární klíč,
- `id_user` - identifikátor uživatele, který provedl zaznamenanou akci - cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `id_module` - identifikátor modulu, v kterém uživatel pracoval - cizí klíč z tabulky `kernel_modules` (viz. podkapitola 3.1.3),
- `id_action` - identifikátor akce, kterou uživatel provedl - cizí klíč z tabulky `kernel_modules_actions` (viz. podkapitola 3.1.4),
- `date_time` - datum a čas provedení zaznamenání,
- `title` - název zaznamenané činnosti,
- `description` - bližší popis činnosti,
- `request_uri` - požadovaná URL adresa,
- `http_referer` - URL adresa odkazující stránky,
- `ip_address` - IP adresa uživatele,
- `hostname` - DNS záznam k IP adrese uložené v atributu `ip_address`,
- `proxy_info` - informace o proxy, pokud byl přes ní uživatel připojen,
- `user_agent` - informace o softwaru, pomocí kterého uživatel pracoval se systémem.

3.5 Modul prohřešky

3.5.1 Tabulka offences

Tabulka `offences` (viz. obr. 3.20) bude sloužit k zaznamenávání prohřešků uživatelů (viz. podkapitola 2.1.3).

Tabulka `offences` bude obsahovat následující atributy:

- `id` - identifikátor prohřešku, primární klíč,
- `id_user` - identifikátor uživatele, který prohřešek vykonal (byl přihlášen při jeho zaznamenání) - cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `date_time` - datum a čas prohřešku,

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_user	INT	✓		☑ UNSIGNED
date_time	INT	✓		☑ UNSIGNED
offence	TEXT	✓		
request_uri	TEXT	✓		
http_referer	TEXT			
ip_address	VARCHAR(39)	✓		☐ BINARY
hostname	VARCHAR(255)			☐ BINARY
proxy_info	TEXT			
user_agent	TEXT			

Obr. 3.20: Tabulka offences

- `offence` - záznam o prohřešeku,
- `request_uri` - požadovaná URL adresa,
- `http_referer` - URL adresa odkazující stránky,
- `ip_address` - IP adresa odkud byl uživatel přihlášen při vykonání prohřešku,
- `hostname` - DNS záznam k IP adrese uložené v atributu `ip_address`,
- `proxy_info` - informace o proxy, pokud byl přes ní uživatel při vykonání prohřešku přihlášen,
- `user_agent` - informace o softwaru, pomocí kterého uživatel pracoval se systémem v době vykonání prohřešku.

3.6 Modul výjimky

3.6.1 Tabulka exceptions

Tabulka `exceptions` (viz. obr. 3.21) bude sloužit pro zaznamenávání výjimek, které se vyskytnou při běhu systému (viz. podkapitola 2.1.1).

Tabulka `exceptions` bude obsahovat následující atributy:

- `id` - identifikátor výjimky, primární klíč,
- `id_user` - identifikátor uživatele, který byl v době zaznamenání výjimky přihlášen - cizí klíč z tabulky `users` (viz. podkapitola 3.2.1),
- `date_time` - datum a čas výjimky,

Column Name	DataType	NN	AI	Flags
id	INT	✓	✓	☑ UNSIGNED
id_user	INT	✓		☑ UNSIGNED
date_time	INT	✓		☑ UNSIGNED
record_type	INT	✓		☑ UNSIGNED
code	INT	✓		☐ UNSIGNED
message	TEXT	✓		
error_file	VARCHAR(255)	✓		☐ BINARY
error_line	INT	✓		☑ UNSIGNED
error_code	TEXT	✓		
debug_backtrace	TEXT	✓		
request_uri	TEXT	✓		
http_referer	TEXT			
ip_address	VARCHAR(39)	✓		☐ BINARY
hostname	VARCHAR(255)			☐ BINARY
proxy_info	TEXT			
user_agent	TEXT			

Obr. 3.21: Tabulka exceptions

- `record_type` - typ výjimky (0 = databázová výjimka, 1 = souborová výjimka, 2 = systémová výjimka),
- `code` - kód výjimky (pro jednoduché znázornění četnosti dané výjimky a její jednoduché dohledání),
- `message` - podrobná zpráva o výjimce,
- `error_file` - soubor, který způsobil výjimku,
- `error_line` - řádek v souboru, který způsobil výjimku,
- `error_code` - kód způsobující výjimku (chybu),
- `debug_backtrace` - návratová hodnota PHP funkce *debug_backtrace* (vrací ladící informace),
- `request_uri` - požadovaná URL adresa,
- `http_referer` - URL adresa odkazující stránky,
- `ip_address` - IP adresa odkud byl uživatel přihlášen při vykonání prohřešku,
- `hostname` - DNS záznam k IP adrese uložené v atributu `ip_address`,
- `proxy_info` - informace o proxy, pokud byl přes ní uživatel při vykonání prohřešku přihlášen,

- `user_agent` - informace o softwaru, pomocí kterého uživatel pracoval se systémem v době vykonání prohřešku.

4 REALIZACE

Pro realizaci systému byl vybrán programovací jazyk PHP (Hypertext Preprocessor). Jazyk PHP byl od počátku vyvíjen se zaměřením na webové aplikace. PHP je Open source jazykem, který je multiplatformní (existují verze pro operační systémy Microsoft®Windows®, Unix, Linux a Mac). Další jeho předností je jednoduchá instalace a možnost běhu pod různými webovými servery jako Apache, iPlanet, IIS a dalšími. PHP má také zabudovánu podporu pro mnoho druhů databází jako Oracle, MySQL, mSQL, PostGres, SQL Server, sBase Files, ODBC, ADO s podporou perzistentního spojení. Všechny funkce a možnosti PHP jsou bez výjimky poskytovány zdarma. Ke zvolení jazyka PHP vedl také vývoj tohoto jazyka v jeho posledních verzích, ve kterých byla zabudována mohutnější podpora pro objektově orientované programování (které je u větších projektů rozsahu tohoto systému nezbytností) aj. PHP je také nejrozšířenějším webovým jazykem, který je podporován na většině webových serverů v celém Internetu.

4.1 Systémové požadavky

Systém vyžaduje pro svou instalaci webový server Apache verze 2.0 a vyšší (s moduly mod_rewrite, mod_ssl a s instalací PHP verze 5.0 a vyšší - nejlépe jako modulu serveru Apache) a databázový server (v případě serveru MySQL je vyžadována verze 5.0 a vyšší). Instalace potřebného softwaru nebude v této práci popisována. Na klientském počítači stačí jakýkoliv webový prohlížeč umožňující nastavování *cookies*. Pro plnohodnotnou práci se systémem je doporučeno vlastnit prohlížeč se zapnutou podporou *JavaScriptu*.

4.2 Adresářová struktura

Adresářová struktura systému a účel jednotlivých adresářů:

- *admin* (administrační část systému),
  - *classes* (programové třídy administrační části systému),
  - *includes* (soubory nutné k načtení v administrační části systému),
  - *js* (javascriptové funkce používané v administrační části systému),
  - *modules* (umístění modulů - každý modul svůj adresář),
  - *phpeditor* (editor PHP souborů používaný v administrační části systému),
  - *wsweditor* (editor WYSIWYG používaný v administrační části webu),

- *classes* (třídy inicializovány jádrem a používány celým systémem),
  - *db* (třídy pro komunikaci s různými typy databází),
  - *exceptions* (třídy zachytávající různé typy výjimek),
  - *functions* (třídy funkcí často využíváných systémem i moduly),
  - *interfaces* (rozhraní používané v objektově orientovaném programování),
  - *modules* (třídy modulů určené pro frontendovou část systému),
  - *monitoring* (třídy pro monitorování systému),
  - *sessions* (umístění Session Handleru),
- *includes* (pro soubory načítané systémem),
- *js* (javascriptové funkce využívány v celém systému i modulech),
- *languages* (jazykové soubory),
  - *dictionaries* (slovníky jednotlivých jazyků, použité jako ochrana proti útoku hrubou silou),
  - *lang_xxx* (soubory daného jazyka, kde „xxx“ je nahrazeno mezinárodní tříznakovou zkratkou daného jazyka dle ISO 639-2),
    - *admin* (jazykové soubory pro administrační část systému),
      - *modules* (jazykové soubory modulů),
- *layouts* (layouty),
- *upload* (adresář pro upload souborů - potřebná práva pro zápis),
- *wsweditor* (pro umístění používaného WYSIWYG editoru).

4.3 Základní programátorská pravidla

V celém systému jsou dodržována následující základní programátorská pravidla:

- Názvy tříd jsou psány velkými počátečními písmeny, víceslovné názvy jsou psány tak, že každé nové slovo je psáno s počátečním velkým písmenem,
- názvy funkcí a metod jsou psány malými počátečními písmeny, víceslovné názvy jsou psány tak, že každé nové slovo je psáno s počátečním velkým písmenem,

- proměnné jsou psány malými písmeny (kromě proměnné `$Kernel`, která zpřístupňuje jádro systému), víceslovné názvy proměnných jsou psány tak, že slova jsou od sebe oddělena znakem „_“ (podtržítko - např. `$count_of_items`),
- pro psaní stringů, řetězcové indexy v polích, parametry funkcí a regulárních výrazů jsou používány apostrofy („’“), uvozovky jsou parsovány na speciální znaky, tedy jejich použití vede ke zpomalení,
- uvozovky jsou použity pouze u tzv. netisknutelných znaků, kde jsou nutností (např. `"\n"`, `"\t"`),
- v případě potřeby je používána i typová kontrola proměnných (PHP při porovnání proměnných „==“ porovnává pouze jejich hodnoty nezávisle na jejich typu - pro porovnání i jejich typu je nutné použít „===“),
- řetězce jsou napojovány pomocí konstrukce „.=“,
- PHP kód je v případě potřeby vložení HTML kódu přerušován pomocí sekvence znaků „?>“ a opět započat pomocí „<?php“,
- z důvodů rozlišení např. od xml dokumentů je vždy PHP kód započat sekvencí znaků „<?php“, nikoliv pouze zkráceným zapisem „<?“,
- přednostně jsou používány funkce pro práci s řetězci, v případě potřeby regulárních výrazů jsou používány regulární výrazy perlovského stylu („preg_*“), které jsou oproti kompatibilním funkcím posix „ereg_*“ rychlejší,
- nebezpečné části kódu, které mohou často skončit neovlivnitelnou chybou jsou uzavřeny do tzv. výjimek, které systém automaticky zaznamená,
- pro použití cyklů není využíván cyklus *forech*, který je velmi pomalý, ale je nahrazen cyklem *while* (s funkcí *array_keys* v případě asociativního pole, viz. ukázka 4.1),
- v případě použití relativní cesty je vždy na začátku uveden znak „.“ (tečka), který zajistí, že PHP interpret bude hledat soubor jen v jednom umístění (v aktuálním adresáři), nikoliv ve všech cestách systémové proměnné `PATH`,
- pro načtení souborů je vždy použita funkce *require_once*, která narozdíl od *include_once* v případě neexistence souboru zastaví běh skriptu (a narozdíl od funkce *require* načte soubor pouze jednou),
- SQL dotazy nejsou nikdy volány v cyklu (bylo by tím způsobeno velké zatížení databáze a zpomalení systému), je využíváno spojování tabulek pomocí syntaxe *JOIN*,

- SQL dotazy jsou optimalizovány dle optimalizačních pravidel (pro databázi MySQL viz. [8]),
- každá třída a funkce obsahuje komentář dle syntaxe nástroje *phpDocumentor*,
- kód je pravidelně komentovaný (především ve složitějších částech) pro větší přehlednost.

```
1  <?php
2  // FOREACH - asociativni pole
3  $keys = array_keys($array);
4  $i = 0;
5  while (true == isset($keys[$i])) {
6      $array[$keys[$i]];
7      $i++;
8  }
9  ?>
```

Ukázka zdrojového kóu 4.1: Nahrazení cyklu foreach

4.4 Jádro

Jádro systému načítá jeho základní nastavení z databázové tabulky `kernel_config` (viz. podkapitola 3.1.1). Jedná se o hodnoty nastavitelné modulem konfigurace uvedené v podkapitole 4.5.2. Jádro systému obsluhuje veškeré uživatelské požadavky, které v případě potřeby předává jednotlivým nainstalovaným modulům ke zpracování. Jádro systému je inicializováno ihned na začátku relace v podobě třídy *Kernel*. Třída *Kernel* zinicializuje jaderné třídy, které jsou systémem využívány. Jedná se o třídy využívané obecně ve všech částech systému:

- *Database* (viz. podkapitola 4.4.2),
- *Exceptions* (několik podtříd, zaměřujících se na výjimky určitého typu - viz. podkapitola 4.4.3),
- *Filter* (viz. podkapitola 4.4.4),
- *Functions* (několik podtříd, zaměřujících se na obsloužení funkcí určitého typu - viz. podkapitola 4.4.5),
- *Layout* (viz. podkapitola 4.4.6),
- *Mail* (viz. podkapitola 4.4.7),

- *Monitoring* (několik podtříd, zaměřujících se na monitorování určitého druhu - viz. podkapitola 4.4.8),
- *SessionHandler* (viz. podkapitola 4.4.9).

Dále jde také o třídy, které jsou využívány pouze administrační částí systému:

- *User* (viz. podkapitola 4.4.10),
- *Module* (viz. podkapitola 4.4.11).

Veškeré tyto třídy jsou dostupné svým názvem psaným malými písmeny přes jadernou třídu *Kernel* jako atributy této třídy. Tedy např. `$Kernel->functions`, `$Kernel->mail` a podobně. Jediná třída *Database* je dostupná jako atribut `$Kernel->db` a třída *SessionHandler* jako atribut `$Kernel->session_handler`.

4.4.1 Vstupní proměnné

Vstupní proměnné PHP jako `$_GET`, `$_POST`, `$_FILES`, `$_SESSION` a `$_COOKIE` (stejně tak i proměnná `$_SERVER`, tedy proměnné z tzv. předdefinovaných proměnných) jsou systémem automaticky očištěny od nebezpečného vstupu pomocí funkcí PHP *addslashes*, která přidá před znaky „´“ (jednoduchá uvozovka), „¨“ (dvojitá uvozovka), „\“ (zpětné lomítko) a hodnotu „NULL“ zpětné lomítko jako tzv. „očisťující“ znak, pro bezpečné použití v databázových dotazech (jedna z ochran před SQL Injection) a *htmlspecialchars*, která nahradí speciální znaky na jejich HTML entity (jedna z ochran před útokem XSS). Tyto proměnné jsou vyprázdněny (tedy jejich přímé použití v systému je znemožněno). Jejich očištěný obsah je uložen do proměnných jádra `$Kernel->get`, `$Kernel->post`, `$Kernel->session`, `$Kernel->cookies` a `$Kernel->server`. Pomocí těchto předdefinovaných proměnných je realizována většina útoků na webové systémy založených na snaze změnit především databázové dotazy či strukturu a obsah generovaného HTML dokumentu. Jiné předdefinované proměnné není v systému umožněno a povoleno používat.

Dalším možným útokem na webové systémy je podvržení proměnných či jejich změna. Proti tomuto typu útoků je systém chráněn pomocí šifrování (viz. podkapitola 4.4.5).

4.4.2 Třída Database

Třída *Database* slouží ke komunikace s databází. Díky ní je komunikace striktně oddělena od jiných částí systému. V celém systému je používána právě tato třída, která obsahuje řadu metod pro postupné sestavení databázového dotazu. V systému není umožněno provádět databázové dotazy jinak, než prostřednictvím této třídy.

Výhodou tohoto oddělení je nejen bezpečnost daného řešení, ale i jeho univerzálnost. Díky tomuto oddělení je umožněno použít systém s téměř libovolnou databází. Pro každý typ databáze je napsána třída s identickými metodami dle rozhraní (nebo-li *interface*) databáze (*DB_Interface*). Toto rozhraní zajistí, že daná třída musí obsahovat minimálně všechny požadované metody a atributy pro bezproblémovou spolupráci se systémem. Původní návrh systému je zaměřen na databázi *MySQL*. Jsou tedy připraveny dvě třídy pro spolupráci s touto databází přes různá rozhraní:

- *mysql*,
- *mysqli* (novější rozhraní databáze MySQL).

Jak již bylo uvedeno výše, databázové dotazy nejsou v systému psané jako celek, ale jsou skládány pomocí metod třídy *Database* přístupné přes jádro systému (`$Kernel->db`). Z bezpečnostního hlediska to přináší výhody v jednoduché kontrole dat předávaných serveru SQL, ale i kontrola samotného SQL dotazu jako takového. Všechna data jsou očištěna pomocí tzv. „escapovacích“ funkcí, které zabrání vložení nebezpečného kódu do těla SQL dotazu. Dále je touto třídou zjednodušena práce s daty uloženými v databázi právě pomocí již předpřipravených metod k získání různých vlastností ať už potřebných dat nebo samotné databáze. Sestavení jednoduchého SQL dotazu je uvedeno v ukázce 4.2.

```php
<?php
/**
 * Sestaví dotaz na databázi
 * "SELECT id FROM table ORDER BY id ASC, title ASC;"
 * provede ho
 * a vypíše všechna id nalezená v databázi v tabulce table
    .
 */
$Kernel->db->setColumns(array('id'));
$Kernel->db->setTables(array('table'));
$Kernel->db->setOrderBy(
    array(
        'id' => 'ASC',
        'title' => 'ASC'
    )
);
$Kernel->db->select();
while (true == ($data = $Kernel->db->getRows())) {
    echo $data['id'];
}
?>
```

Ukázka zdrojového kóu 4.2: Sestavení databázového dotazu

Využití metod pro jednodušší práci s databází se dá prezentovat například při potřebě pouze některých dat z databáze a zároveň potřebě zjištění počtu všech dat bez ohledu nastaveného limitu pro výběr dat. Toto je prezentováno v ukázce 4.3.

```
1  <?php
2  // vyber peti polozek od pate v tabulce
3  $Kernel->db->setColumns(array('id', 'shift'));
4  $Kernel->db->setTables(array('kernel_modules'));
5  $Kernel->db->setWhere('blocked == 0');
6  $Kernel->db->setLimit(5, 5);
7  $result = $Kernel->db->select();
8  // kolik polozek vyhovuje vcetne where i limit
9  $count_items = $Kernel->db->getNumRows($result);
10 // celkem polozek vyhovujicich where, nezavisle na limit
11 $count_total_items = $Kernel->db->getTotalNumRows($result)
       ;
12 // vypis x polozek z celkem y polozek (nehlede na limit)
13 echo $count_items . ' ' . $Kernel->lang_arrays['from'] . '
       ' . $count_total_items . ' ' . $Kernel->lang_arrays['
       items'];
14 // vypis polozek vyhovujicich puvodnimu SQL dotazu vcetne
       podminek where a limit
15 while {true == ($data = $Kernel->db->getRows($result)) {
16     echo $data['id'] . ' (' . $data['shift'] . ')';
17 }
18 ?>
```

Ukázka zdrojového kóu 4.3: Usnadnění práce s databází

Veškeré dotazy na databázi jsou kontrolovány na správnost svého provedení pomocí třídy výjimek (viz. podkapitola 4.4.3). Jakmile dojde k chybě, tak je chyba systémem zaprotokolována s veškerými potřebnými údaji k jejímu napravení. Jednoduchý protokol o chybě je vždy ihned zobrazen uživateli (v administrační části systému). Dále systém umožňuje protokolovat naprosto všechny dotazy na databázi (viz. podkapitola 4.4.8). To je opět umožněno oddělením komunikace s databází samostatnou třídou jádra systému. Taktéž je tímto oddělením snadné počítat množství provedených dotazů a časové zatížení databázového serveru. Tyto informace jsou zobrazovány uživatelům s úrovní supervizor v „patičce“ administrační části systému.

4.4.3 Třída Exceptions

Exceptions je defaultí třída výjimek dostupná v PHP verze 5 a vyšší. Jádrem systému je rozšířena na tři různé třídy zachycující výjimky různých typů. Jedná se o třídy:

- *dbException*,

- *fileException*,
- *systemException*.

Všechny tyto třídy jsou tedy rozšířením třídy *Exceptions* a implementují definované rozhraní *Exceptions_Interface*, aby nebylo v daných třídách pro zaznamenání výjimek opomenuto na žádnou potřebnou vlastnost (atribut) nebo metodu. Rozšíření výchozí třídy *Exceptions* se týká především rozšíření zachycujících informací, které jsou v případě provedení výjimky zaznamenány do databáze (viz. podkapitola 3.6.1). Jediné tyto třídy nezačínají v názvu velkým písmenem, protože nejsou jádrem doslova načítány a nejsou jeho atributem, ale jsou jádrem pouze inicializovány a poté využívány k zachycení výjimek. Každá výjimka je vyvolána a zachycena známým blokem *try - catch* (viz. ukázka 4.4).

```
1  <?php
2  // provedeni dotazu na databazi
3  $this->result = @mysqli_query($this->connect_id, $this->
       query);
4  try {
5      // test zda nedoslo k chybe
6      if ($this->getErrorNo() !== 0) {
7          // chyba - jeji zapis a ohlaseni!
8          throw new dbException($this->getError(), 1, $this
               ->query);
9      }
10 } catch (dbException $e) {
11     // zachyceni vyjimky
12     // vypsani a zaznamenani protokolu zajisti sama trida
           dbException, lze pouzit i nasledujici
13     //$e->getFormattedMessage();
14 }
15 ?>
```

Ukázka zdrojového kóu 4.4: Zachycení databázové výjimky

4.4.4 Třída Filter

Třída *Filter* slouží k zobrazení a nastavení filtru na výpisy dat v systému. U všech dat, kde je vhodné použít filtrování, je nastaven filtr pomocí této třídy. Třída je v systému dostupná jako atribut třídy *Kernel* (`$Kernel->filter`) jako ostatní třídy jádra. Nastavený filtr se ukládá pomocí uživatelského nastavení (viz. podkapitola 4.4.10). Filtr je tedy nastavený nejen pro aktuální zobrazení či relaci, ale i např. pro relaci další (nastavení filtru je pro daného uživatele uloženo v databázi a při další relaci či zobrazení znovu načteno). Filtr může být dokonce i nastaven, ale

nepoužit. Mezi jednotlivými položkami filtru se dá nastavit požadovaný vztah „A“ či „NEBO“. Příklad použití filtru pomocí této třídy je zobrazeno v ukázce 4.5.

```
1  <?php
2  $where_filter = '';
3  // begin: filter
4  $Kernel->tabs = "\t\t\t\t";
5  // identifikator filtru
6  $Kernel->filter->startFilter('users_filter_view_groups');
7  // id skupiny
8  $Kernel->filter->addLabel('id', $Kernel->lang_arrays['
       admin']['module']['users']['groups']['view']['filter']['
       id']['label']);
9  $Kernel->filter->addInput('text', 'id', 'id');
10 $Kernel->filter->addHiddenComparison('id', '=');
11 // nazev skupiny
12 $Kernel->filter->addLabel('title', $Kernel->lang_arrays['
       admin']['module']['users']['groups']['view']['filter']['
       title']['label']);
13 $Kernel->filter->addInput('text', 'title', 'title');
14 $Kernel->filter->addHiddenComparison('title', 'LIKE');
15 // zobrazeni formulare
16 echo $Kernel->filter->getFilterForm();
17 // podminka pripravena k pouziti v SQL dotazu pro nacteni
       dat
18 $where_filter = $Kernel->filter->getFilterWhere();
19 // end: filter
20 ?>
```

Ukázka zdrojového kóu 4.5: Použití třídy pro filtrovani

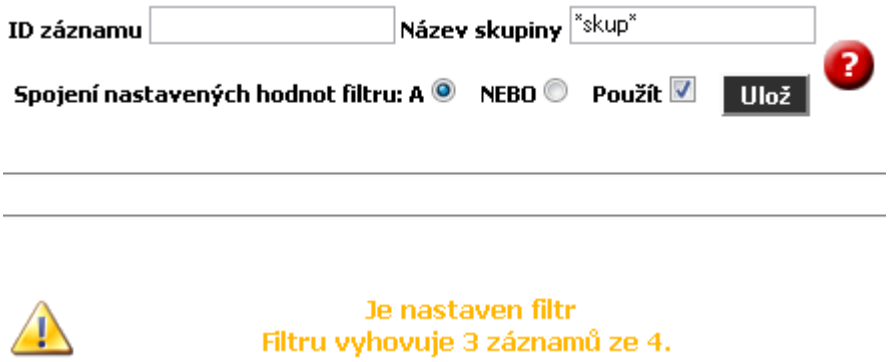
Tento kód je použit u filtru uživatelských skupin a zpřístupní filtr, pomocí kterého mohou být filtrovány skupiny uživatelů dle svého id či názvu (viz. obr. 4.1). V polích typu „text“ je umožněno používat zástupné znaky a to:

- **?** - pro zastoupení právě jednoho znaku,
- ***** - pro zastoupení žádného i více znaků.

Jak je zobrazeno na obr. 4.1, uživatel je vždy na nastavený a použitý filtr upozorněn a je informován kolik záznamů z celkového počtu danému filtru vyhovuje.

4.4.5 Třída Functions

Tato třída obsahuje spoustu podtříd, které slouží k různým účelům a jsou vyjmenovány níže. Všechny tyto třídy jsou inicializovány třídou *Functions* a jsou jejími



Obr. 4.1: Filtr skupin uživatelů

atributy. Třída *Functions* je atributem třídy *Kernel* a je tedy dostupná pod atributem třídy jádra (`$Kernel->functions`).

Třída *Functions* tedy obsahuje atributy zpřístupňující následující třídy:

- *CommonFunctions*, viz. níže,
- *CookiesFunctions*, viz. níže,
- *DateFunctions*, viz. níže,
- *FilesFunctions*, viz. níže,
- *ImagesFunctions*, viz. níže,
- *SecurityFunctions*, viz. níže,
- *ValidateFunctions*, viz. níže.

Všechny tyto třídy jsou tedy inicializovány jako atributy třídy *Functions*. Tyto atributy jsou vždy pojmenované dle dané třídy s vynecháním „Functions“ v názvu a dle zmíněných programátorských pravidel (viz. kapitola 4.3) s malým počátečním písmenem (např. třída *SecurityFunctions* je tedy v systému dostupná jako `$Kernel->functions->security`).

Třída CommonFunctions

Třída *CommonFunctions* zpřístupňuje systému běžně používané funkce v celém systému a především v jeho modulech. Jedná se o často používané funkce jako možnost stránkování záznamů, možnost uživatelského řazení záznamů dle libovolného atributu, volba počtu zobrazených záznamů na stránku, umožní ale i např. uživatelské přepnutí jazyka systému v rámci jedné relace a mnoho dalších funkcí.

Třída CookiesFunctions

Třída *CookiesFunctions* zpřístupňuje vytváření a mazání šifrovaných cookies. Vytvoření cookies zajišťuje metoda `$Kernel->functions->cookies->setCookies`, jejímiž parametry jsou název, hodnota a platnost cookie, a zda má být daná cookie dostupná pouze přes zabezpečené připojení (HTTPS). Mazání cookies zajišťuje metoda `$Kernel->functions->cookies->eraseCookie`, jejímž atributem je pouze název cookie, která má být odstraněna. Vytvoření cookie je testováno na úspěšnost a v případě neúspěchu je zachycena výjimka pomocí třídy *systemException*. Stejně tak je testováno i mazání cookie, protože jejím mazáním je myšleno nastavení cookie, které již vypršela platnost (v systému je nastavováno 1000 sekund stará cookie). Webové prohlížeče je automaticky pokládají za neplatné a fyzicky je vymažou z uživatelova počítače.

Třída DateFunctions

Systém využívá pro ukládání informací o datu a času v databázi datový typ *integer*. Je tak činěno proto, aby nebyl problém systém optimalizovat pro univerzální použití nehledě na použité databázi, které mají často datové typy pro ukládání data a času rozdílné. Do databáze se tak ukládá čas ve formátu používaném v Unixových systémech, tzv. *Unix timestamp*. Jedná se o počet sekund od tzv. *Unix Epoch*, tedy od 1. ledna 1970 00:00:00 GMT. V PHP tuto hodnotu vrací funkce *time*.

Právě proto je vytvořena v systému tato třída, která umožní jednoduché zformátování data do libovolného formátu. Výchozí formát je nastaven v konfiguraci systému. Systém umožňuje také zohledňovat letní a zimní čas (viz. podkapitola 4.5.2).

Třída FilesFunctions

Třída *FilesFunctions* zpřístupňuje systému funkce pro práci se soubory. Jedná se například o náhrání souboru na server, jeho kopírování, změna vlastníka či oprávnění k souboru, dále také otevření a načtení obsahu souboru či uložení nového obsahu aj. Většina těchto funkcí může často skončit neúspěchem, proto jsou z velké části testovány na úspěšnost a v případě neúspěchu jsou odchyceny tzv. souborové výjimky pomocí třídy *fileException* (viz. podkapitola 4.4.3).

Třída ImagesFunctions

Pro práci s obrázky je v systému definována třída *ImagesFunctions*. Pomocí této třídy je např. možno měnit rozměry obrázků, vytvářet vodoznak či zjistit velikost, rozměry aj. informace o specifikovaném obrázku.

Třída SecurityFunctions

Tato třída poskytuje systému řadu bezpečnostních funkcí. Zajišťuje například šifrování citlivých informací (obsahuje tedy metody pro šifrování a dešifrování). Pomocí těchto metod jsou šifrovány a dešifrovány názvy a hodnoty uložené v *cookies* a parametry předávané pomocí metody *get*, čímž je zabráněno jednoduché modifikaci těchto dat uživatelem systému.

K šifrování názvu a hodnot *cookies* je použita metoda *encodeCookie* třídy *SecurityFunctions* (viz. ukázka 4.6), k dešifrování pak metoda decodeCookie téže třídy (viz. ukázka 4.7). K těmto účelům jsou použity funkce PHP *base64_encode*, *base64_decode* (zakóduje a dekóduje text pomocí MIME base64 - používá se např. v emailové komunikaci pro přenos binárních dat), *urlencode* (zakóduje text pro použití v URL adrese), *mcrypt_encrypt* a *mcrypt_decrypt* (zakóduje a dekóduje text pomocí symetrické šifry *Blowfish* v režimu *ECB*).

```
1  <?php
2  /**
3   * Zakódování hodnoty nebo názvu cookie.
4   *
5   * @param string $cookie_value Název nebo hodnota cookie.
6   * @return string
7   */
8  public function encodeCookie($cookie_value) {
9      $cookie_value = base64_encode($cookie_value);
10     $iv_size = mcrypt_get_iv_size(MCRYPT_BLOWFISH,
           MCRYPT_MODE_ECB);
11     $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
12     $cookie_value = mcrypt_encrypt(MCRYPT_BLOWFISH, $this
           ->cookie_key, $cookie_value, MCRYPT_MODE_ECB, $iv);
13     $cookie_value = urlencode(base64_encode(trim(
           $cookie_value)));
14     return $cookie_value;
15 }
16 ?>
```

Ukázka zdrojového kóu 4.6: Šifrování cookies

Dále tato třída slouží ke generování bezpečného uživatelského hesla. Vygenerované heslo je vždy kontrolováno třídou pro validaci pomocí metody pro validaci bezpečnosti hesla (viz. níže, odstavec *Třída ValidateFunctions*).

```php
<?php
/**
 * Dekódování hodnoty nebo názvu cookie.
 *
 * @param string $cookie_value Název nebo hodnota cookie
 * @return string
 */
public function decodeCookie($cookie_value) {
    $cookie_value = base64_decode(urldecode($cookie_value));
    $iv_size = mcrypt_get_iv_size(MCRYPT_BLOWFISH, MCRYPT_MODE_ECB);
    $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
    $cookie_value = mcrypt_decrypt(MCRYPT_BLOWFISH, $this->cookie_key, $cookie_value, MCRYPT_MODE_ECB, $iv);
    $cookie_value = base64_decode(trim($cookie_value));
    return $cookie_value;
}
?>
```

Ukázka zdrojového kóu 4.7: Dešifrování cookies

Třída ValidateFunctions

Třída *ValidateFunctions* slouží jak k validaci obsahu proměnných, tak k validaci jejich typu. V systému je dostupná přes jádro pomocí výrazu `$Kernel->functions->validate`. Pomocí této třídy je například možné testovat sílu uživatelem zvoleného či systémem vygenerovaného hesla (viz. ukázka 4.8 z modulu uživatelů při editaci svého uživatelského účtu).

```php
<?php
// kontrola hesla zaslaneho metodou post
// zapnuta kontrola na slovnikove utoky vsech nainstalovanych jazyku se 100% presnosti
$password_check = $Kernel->functions->validate->password($Kernel->post['save_in_db']['user_password'], true, true, 100);
if ($password_check !== 0) {
    // pokud heslo neni bezpecne, je zobrazena chyba, ktera specifikuje nebezpecnost hesla
    $errors['user_password'] = $Kernel->lang_arrays['admin']['module']['users']['users']['edit']['user_password']['error']['bad_format'][$password_check];
}
?>
```

Ukázka zdrojového kóu 4.8: Kontrola síly zvoleného hesla

4.4.6 Třída Layout

Třída *Layout* slouží pro frontendovou část webu. Umožní frontendové části využívat vlastní šablonovací systém, který je založen na layoutu a dále na šablonách a samotných stránkách. Tato třída bude využita až rozšiřujícími moduly, které budou obsahovat frontendovou část (např. modul stránek, katalogu, elektronického obchodu aj.). V základní koncepci systému je třída připravena, ale jadernými moduly není nijak využívána, protože tyto moduly neobsahují frontendovou část. Jsou „pouze“ základem pro webový modulární systém jako takový, nezaměřují systém pouze na správu obsahu, dokumentů či jakýmkoliv jiným směrem. Toto zaměření bude právě až na rozšiřujících modulech a tedy na záměru zákazníka požadujícího dané moduly.

4.4.7 Třída Mail

V několika různých případech je zapotřebí, aby systém mohl odesílat emaily. Tuto funkci bude zajišťovat třída *Mail*, která obdobně jako třída *Database* (v případě sestavení SQL dotazu - viz. podkapitola 4.4.2) několika metodami sestaví postupně celé tělo emailu a email odešle. Emailem je umožněno posílat jak textové emaily, tak emaily ve formátu HTML či jejich kombinace. Samozřejmě je umožněno též přikládat soubory jako přílohy. Pomocí této třídy jsou například zasílány emaily uživatelům s novým účtem v systému či uživatelům s úrovní superuživatel (supervizor) v případě zaznamenání uživatelského prohřešku aj. Ukázka použití třídy *Mail* je zobrazena v ukázce 4.9 (z modulu uživatelů při vytvoření nového uživatelského účtu - zaslání informací na zadaný email u nově vytvořeného účtu).

```
1  <?php
2  // definice odesilatele - email a nazev z konfigurace
3  $Kernel->mail->setFrom($Kernel->getConfigValue('
       email_system'), $Kernel->getConfigValue('web_title'));
4  // definice prijemce ze zaslanych hodnot metodou post
5  $Kernel->mail->addTo($Kernel->post['save_in_db']['email'],
       $username . ' (' . $Kernel->post['save_in_db']['surname
       '] . ', ' . $Kernel->post['save_in_db']['firstname'] . '
       )');
6  // nastaveni priority, predmetu, tela a odeslani emailu
7  $Kernel->mail->setPriority(1);
8  $Kernel->mail->setSubject($title);
9  $Kernel->mail->setBody($message);
10 $Kernel->mail->send();
11 ?>
```

Ukázka zdrojového kóu 4.9: Použití třídy Mail

4.4.8 Třída Monitoring

Pro monitorování systému (viz. podkapitola 2.1.7) je vytvořena třída *Monitoring*, která zajišťuje zapsání monitorovacích záznamů do databáze. Dále tato třída zinicializuje vlastní *PHP Error Handler*, který je rozšířením výchozího Error Handleru o několik systémových položek a informací a jehož výstup je zapisován do databáze (viz. kapitola 3.4, tabulky do nichž jsou data zapisována jsou zobrazeny v podkapitolách 3.4.2, 3.4.3, 3.4.4 a 3.4.5).

4.4.9 Třída SessionHandler

Pro správu a umožnění relací je v PHP potřeba spustit a udržovat tzv. „session relaci“. PHP obsahuje výchozí Session Handler. Tento *Session Handler* ukládá proměnné `$_SESSION` na souborový server. Pro více možností a větší kontrolu nad proměnnými `$_SESSION` je vhodné definovat vlastní *Session Handler*. V systému je tedy definován vlastní *Session Handler*, který ukládá dané proměnné do databáze (viz. podkapitola 3.1.6). Umožňuje tak vlastní správu a plnou kontrolu nad proměnnými vytvořenými v rámci relace s uživateli.

4.4.10 Třída User

Třída *User* zajišťuje přihlášení uživatele, vytvoření, udržení a zrušení uživatelské relace se systémem a načítání informací o daném uživateli. Třída také načítá oprávnění daného uživatele pro práci se systémem, jeho příslušnost do uživatelských skupin a jeho uživatelské nastavení systému.

Přihlášení

Přihlášení uživatele do administrační části systému je umožněno pouze uživatelům s platným, neblokovaným uživatelským účtem. K přihlášení je zapotřebí znalost uživatelského jména a hesla.

Uživatelské jméno musí být tvořeno minimálně šesti znaky anglické abecedy nebo číslicemi (mohou být také použity znaky „_“, „-“, „.“). Uživatelské heslo musí být nejméne osm znaků dlouhé a musí se skládat pouze ze znaků s kódy 33 až 126 *ASCII tabulky*. Heslo musí být složeno z nestejných znaků (maximálně 25% délky hesla může tvořit stejný znak) a nesmí být odvoditelné ze známých slovníkových slov všech jazyků nainstalovaných v systému (obrana proti útoku hrubou silou). Splnění všech těchto podmínek pro dostatečně silné heslo i podmínek pro správný formát uživatelského jména je testováno pomocí třídy *ValidateFunctions* (viz. podkapitola 4.4.5). Každé heslo má nastavenu maximální platnost, která je nastavitelná

v konfiguraci systému (viz. podkapitola 2.2.1). Po vypršení této platnosti hesla je uživatel nucen si heslo změnit, jinak mu není umožněna práce se systémem.

Dále je umožněno již v přihlašovacím formuláři zvolit výchozí jazyk rozhraní systému pro danou relaci.

V případě neúspěšného přihlášení je uživatel informován o důvodu nepovolení ke vstupu do systému a je nastavena cookie s dobou, po kterou se uživatel nemůže znovu pokusit o přihlášení. O této době je informován i uživatel. Tato doba je nastavena v konfiguraci systému (doporučená a výchozí hodnota je 10 vteřin). Pokud bylo zadáno správné uživatelské jméno, ale chybné heslo, nastaví se tato doba i u daného účtu v databázi. V případě opakování více neúspěšných pokusů o přihlášení se tato doba násobí počtem neúspěšných pokusů (které se ukládají do databáze k danému účtu). Jedná se o další ochranu proti útoku hrubou silou. V konfiguraci systému je dále definován maximální počet neúspěšných pokusů o přihlášení. Pokud je dosaženo počtu těchto pokusů, uživatelovi je účet zablokován a o nastálé situaci je informován emailem definovaným u daného účtu. Pokud se uživatel úspěšně přihlásí, vynuluje se počet pokusů o přihlášení i doba po kterou se uživatel nemůže do systému přihlásit.

Oprávnění uživatele

Třída *User* načítá oprávnění přihlášeného uživatele, která má přiřazen přímo jako uživatel, ale i oprávnění která získává díky přílušnosti v uživatelských skupinách. Tato oprávnění jsou dále jádrem testována při používání systému. Třída také umožňuje v případě pokusu o porušení oprávnění daný pokus zaznamenat a informovat o nastálé situace hlavního administrátora definovaného v konfiguraci systému (emailem pomocí třídy *Mail*, viz. podkapitola 4.4.7). Oprávnění uživatele je načítáno z databázových tabulek `permissions_modules_users` (viz. podkapitola 3.3.1 - oprávnění samotného uživatele) a `permissions_modules_groups` (viz. podkapitola 3.3.2 - oprávnění získaná díky účasti ve skupinách) pro načtení oprávnění ke vstupu do modulů a dále z tabulek `permissions_modules_actions_users` (viz. podkapitola 3.3.3 - oprávnění samotného uživatele) a `permissions_modules_actions_groups` (viz. podkapitola 3.3.4 - oprávnění získaná díky účasti ve skupinách).

Uživatelské nastavení

V systému je mnoho hodnot, které si uživatel může přizpůsobit dle vlastního uvážení. Toto přizpůsobení umožňuje dnes většina kvalitních webových systémů, ovšem vždy jen pro jednu relaci uživatele (pomocí sessions), nebo jen pro jeden počítač uživatele (pomocí cookies). Navržený a realizovaný systém však tato uživatelská nastavení udržuje v databázi a umožňuje je tedy dále načítat i v případě dalších uživatelských

relací (nezávisle na počítači z kterého se uživatel přihlásí). Jedná se například o volbu atributů ve výpisu dat, které chce mít uživatel zobrazeny, o volbu počtu záznamů vypsaných na jednu stránku, ale i například o nastavení filtrů u jednotlivých přehledů nastavených pomocí třídy *Filter* (viz. podkapitola 4.4.4). Všechna tato data jsou ukládána do databázové tabulky `users_settings` (viz. podkapitola 3.2.4).

4.4.11 Třída Module

Proto, aby byl systém modulární, musí být zajištěno univerzální načítání modulů a tím rozšíření funkcí systému. Toto načítání modulů zajišťuje právě třída *Module*. Všechny moduly jsou rozšířením této třídy. Obsahuje řadu základních funkcí jako načtení samotného modulu a jeho vlastností, zavolání požadované funkce modulu, zjištění menu modulu, jeho akcí na které se váže oprávnění uživatelů apod.

Tato třída je vždy zinicializována samotným modulem, tedy inicializací rodiče. Jako parametr se konstruktoru předává textový identifikátor modulu, díky kterému samotná třída *Module* načte veškeré informace o modulu, jeho menu a definované akce. Na základě volání jednotlivých akcí je kontrolováno oprávnění přihlášeného uživatele k provedení dané akce a v případě neoprávněného pokusu o provedení akce je zaznamenán uživateli prohřešek (viz. podkapitola 2.1.3).

4.5 Moduly

Jak již bylo uvedeno dříve, moduly umožňují jednoduché rozšíření funkcí systému. Jakmile jádro systému zachytí požadavek o načtení modulu, zjistí zda daný modul existuje a zda má k němu přihlášený uživatel oprávnění přístupu. Pokud je vše v pořádku je modul načten a zinicializována jeho třída (viz. ukázka 4.10). Modul automaticky díky svému konstruktoru zinicializuje i třídu *Module* (viz. podkapitola 4.4.11), kterou rozšiřuje (definováno pomocí klíčového slova *extends*).

Aby mohl být modul načten, musí obsahovat soubor s vlastní třídou, která bude rozšířením třídy *Module* (viz. podkapitola 4.4.11). Dále musí obsahovat jazykové soubory pro nainstalované jazyky v systému. Modul může využívat další systémové soubory, nebo soubory vlastní, které musí být obsaženy v adresáři modulu.

Adresář každého modulu je vytvořen dle jeho textového identifikátoru v adresáři *admin/modules/* (viz. kapitola 4.2). V tomto adresáři je umístěn i soubor s programovou třídou daného modulu. Název souboru s programovou třídou modulu je vždy nazván *identifikator_modulu.class.php*.

```php
<?php
// je volan modul?
if (true == isset($Kernel->get['module'])) {
    // existuje modul?
    if (true === $Kernel->moduleExists($Kernel->get['module'])) {
        // nastartovani modulu - kernel nacte soubor se tridou modulu, kterou zinicializuje
        $Kernel->setModuleHandler($Kernel->get['module']);
        // opravneni pristupu do modulu?
        if (true === $Kernel->user-> hasEntryModulePermission ((int)$Kernel->module->getModuleInfo('id_module'))) {
            // ok ...
        } else {
            // prohresek nema opravneni
        } // end: opravneni pristupu do modulu?
    } else {
        // prohresek - neexistujici modul
    } // end: existuje modul?
} // end: je volan modul?
?>
```

Ukázka zdrojového kóu 4.10: Načtení a inicializace modulu

Jazykové soubory jsou umístěny v adresářích nainstalovaných jazyků, konkrétně v adresáři *languages/lang_xxx/admin/modules/* (viz. kapitola 4.2), kde je „xxx“ nahrazeno mezinárodní tříznakovou zkratkou daného jazyka dle ISO 639-2. Tyto soubory jsou nazývány *lang_identifikator_modulu.php*.

U modulu musí být definovány:

1. Vlastnosti modulu - jsou vkládány při jeho instalaci do tabulky `kernel_modules` (viz. podkapitola 3.1.3).

2. Menu modulu - je tvořeno položkami řazenými do stromu, které jsou uloženy do tabulky `kernel_modules_menu` (viz. podkapitola 3.1.5).

3. Akce modulu - na ně se váže oprávnění uživatelů a uživatelských skupin (viz. podkapitola 4.4.10) a jsou vkládány do tabulky `kernel_modules_actions` (viz. podkapitola 3.1.4).

Instalace modulu probíhá dle kapitoly 4.7.2.

4.5.1 Modul databáze

Modul databáze odpovídá návrhu modulu v podkapitole 2.2.6. Umožňuje tedy výpis tabulek systému, jejich strukturu a v nich uložená data. Modul je přístupný pouze

uživatelům s úrovní supervizor.

4.5.2 Modul konfigurace

Modul konfigurace odpovídá návrhu v podkapitole 2.2.1. Slouží tedy ke konfiguraci systému a k základnímu nastavení modulů.

Obsahuje tedy části:

- *Konfigurace systému* (viz. níže),
- *konfigurace modulů* (viz. níže).

Konfigurace systému

Nastavení systému je vytvořeno během instalačního procesu. Modul konfigurace ho umožňuje editovat. Nastavení obsahuje následující atributy rozdělené do několika částí:

- **Nastavení serveru**:
  - *server_name* (název domény),
  - *server_path* (cesta ke skriptům na serveru),
  - *server_port* (port serveru).
- **Nastavení systému**:
  - *default_lang* (výchozí jazyk systému),
  - *default_layout* (layout systému),
  - *default_items_per_page* (výchozí počet položek na jedné stránce),
  - *system_time_zone* (časová zóna),
  - *system_date_format* (výchozí formát času),
  - *system_dst* (zimní/letní čas),
  - *system_allow_debug* (zda má být povolen ladící mód),
  - *system_blocked_backend* (zda má být blokován backend),
  - *system_blocked_frontend* (zda má blokován frontend).

- **Nastavení monitorování systému**:
  - *monitoring_php* (zda mají být zaznamenávány chyby PHP),
  - *monitoring_sql* (zda mají být zaznamenávány provedené SQL dotazy),
  - *monitoring_system* (zda mají být sledovány změny systému),
  - *monitoring_users* (zda má být sledována práce uživatele).
- **Nastavení emailů**:
  - *email_admin* (administrátorský email),
  - *email_system* (email systému),
  - *email_signature* (podpis emailů).
- **Nastavení webu**:
  - *web_title* (název webu),
  - *web_description* (popis webu).
- **Nastavení uživatelů**:
  - *user_max_login_attempts* (maximální počet pokusů o přihlášení),
  - *user_login_time_limit_between_try* (interval před dalším pokusem o přihlášení ve vteřinách),
  - *user_login_time_validity* (maximální doba platnosti přihlášení ve vteřinách),
  - *user_max_password_validity* (maximální platnost hesla ve vteřinách),
  - *user_max_count_url_history* (počet uložených posledních navštívených stránek).
- **Nastavení cookie**:
  - *cookie_name* (název cookie),
  - *cookie_path* (cesta cookie),
  - *cookie_domain* (doména cookie),
  - *cookie_secure* (zda mají být cookies dostupné pouze přes zabezpečené připojení HTTPS),
  - *cookie_crypt_key* (klíč pro šifrování cookie).

- **Informace o systému** (neměnné):
  - *system_version* (verze systém),
  - *system_licence* (licence systému),
  - *system_installed* (systém nainstalován).

Konfigurace modulů

Konfigurací modulů v modulu konfigurace je myšleno základní nastavení modulů v systému. Umožňuje nastavit minimální požadovanou uživatelskou úroveň pro povolení vstupu do modulu, zda má být modul dostupný všem uživatelům splňujícím požadovanou minimální uživatelskou úroveň (nehledě na nastavená oprávnění), pořadí modulů v menu administrace a zda má být modul blokován. Volby minimální požadované uživatelské úrovně a povolení vstupu všem uživatelům mohou být blokovány z bezpečnostních důvodů, jak je tomu u všech systémových modulů. Do zablokovaného modulu se dostanou pouze uživatelé s úrovní supervizor (ostatní uživatelé s ním nemohou po dobu blokování modulu pracovat).

4.5.3 Modul monitorování

Modul monitorování vyhovuje návrhu modulu v podkapitole 2.2.4. Umožňuje zobrazení a mazání záznamů zaznamenaných systémem (viz. podkapitola 4.4.8) a to záznamů:

- *Systému*,
- *PHP*,
- *uživatelů*,
- *přihlašování*,
- *SQL*.

4.5.4 Modul oprávnění

Modul oprávnění vyhovuje návrhu uvedeném v podkapitole 2.2.3. V tomto modulu jsou nastavována oprávnění uživatelů a skupin uživatelů ke vstupu do jednotlivých modulů a k akcím v jednotlivých modulech. Samotné plnění podmínek oprávnění kontroluje jádro systému (viz. podkapitola 4.4.10).

4.5.5 Modul prohřešky

Modul prohřešky odpovídá návrhu v podkapitole 2.2.5. Zprostředkovává supervizorům přehled a mazání zaznamenaných prohřešků uživatelů.

4.5.6 Modul uživatelé

Modul uživatelé odpovídá návrhu v podkapitole 2.2.2. Obsahuje dvě hlavní části:

- *Uživatelé* (viz. níže),
- *skupiny uživatelů* (viz. níže).

Uživatelé

Umožňuje přehled, mazání, detail, editaci a přidání uživatelů. Mazáním je myšleno označení uživatele za smazaného.

Skupiny uživatelů

Umožňuje přehled, mazání, detail, editaci a přidání skupin uživatelů.

4.5.7 Modul výjimky

Modul výjimky odpovídá návrhu v podkapitole 2.2.7. Poskytuje uživatelům s úrovní supervizor přehled jádrem zaznamenaných výjimek (viz. podkapitola 4.4.3).

4.6 Zabezpeční

Největší důraz při samotné realizaci byl kladen na zabezpečení systému a dat v něm uložených. Zabezpečení systému se také stalo jednou z největších předností realizovaného systému. Systém využívá několika technik zabezpečení před útoky či ztrátou dat.

Administrace systému je dostupná pouze přes zabezpečený protokol *HTTPS*. Tím je zajištěn šifrovaný přenos citlivých dat a administrace systému je tak chráněna proti odposlouchávání či podvržení dat. Proti neoprávněnému vstupu do systému je chráněn přihlašovacími údaji. Tyto přihlašovací údaje se skládají z *uživatelského jména* a *hesla*. Kontrola obsahu a pravidla pro nastavení těchto údajů jsou popisovány v podkapitole 4.4.10. Heslo je v databázi uloženo po několikanásobném zhašování jednoduchými hašovacími funkcemi *MD5* a *SHA-1*. Kombinací i jednoduchých (již prolomitelných) hašovacích algoritmů je dosaženo dostatečné bezpečnosti uložení hesel. Tato bezpečnost je dále zvýšena tzv. zasolením hesel pomocí textu složeného z uživatelského jména a náhodně definovaného klíče (tedy každé heslo je

zasoleno jiným textem, tedy jiným sledem bitů). Délkou hesel, pravidly jejich definice a způsobem jejich uložení je zajištěno, že je i při případné ztrátě uložených hesel minimalizována možnost získání skutečných hesel z uložených zhašovaných a zasolených hodnot (získání hesel hrubou silou je v podstatě nereálné). Všechna hesla jsou při vytváření testována na dostatečnou složitost i proti útoku hrubou silou slovy ze slovníků v systému nainstalovaných jazyků. Další ochranou pro bezpečnost uživatelského hesla je maximální doba jeho platnosti, po jejímž vypršení je uživatel nucen své heslo změnit. Každá uživatelská relace je také časově omezena a v případě neprojevení aktivity po definovanou dobu je uživatel z bezpečnostních důvodů odhlášen.

Systém je dále chráněn proti dnes známým způsobům útoků na webové systémy. Administrace systému je chráněna použitím šifrování URL adres. Pro dané šifrování je použita symetrická šifrovací metoda *Blowfish* v režimu *ECB*. Šifrovací metoda *Blowfish* ani režim *ECB* nepatří k nejbezpečnějším, ale jedná se o dostatečně rychlou a bezpečnou kombinaci pro dané použití. Klíč je pro toto šifrování generován pro každého uživatele a každou relaci unikátní, čímž je zajištěno, že není umožněno napadnout systém pomocí zásahu do URL adres (modifikace vstupních proměnných zasílaných metodou *GET*). Šifrování zajišťuje jádro plně automaticky pomocí třídy *SecurityFunctions* (viz. podkapitola 4.4.5).

Další metodou útoku na webové systémy je modifikace a nastavení podvržených *COOKIES*. Tomuto je zabráněno opět pomocí šifrování symetrickou šifrovací metodou *Blowfish* v režimu *ECB*. Klíč je nastaven v konfiguraci systému a dá se měnit. Jeho změnou však znemožníme přečtení dříve nastavených *COOKIES*, proto se jeho změna nedoporučuje. Šifrování zajišťuje (opět plně automaticky) jádro pomocí třídy *SecurityFunctions* (viz. podkapitola 4.4.5).

Veškeré vstupní hodnoty od uživatele jsou jádrem systému „očištěny“ od nebezpečných sekvencí a speciální znaky jsou převedeny na HTML entity. Tím je systém bráněn jak proti útokům *SQL Injection*, tak proti útokům *XSS*. Proti útokům typu *SQL Injection* je také chráněn oddělením databázové komunikace do samostatné třídy (viz. podkapitola 4.4.2), kde jsou všechna data znovu podstoupena „očištění“ od nebezpečných znaků (umožňujících např. modifikaci databázového dotazu).

Vyšší bezpečnost systému je také zajištěna možností nastavení oprávnění ke všem akcím v systému. Pomocí modulu oprávnění je umožněno nastavení oprávnění na definované akce v jednotlivých modulech. Tato oprávnění je umožněno definovat přímo pro uživatele, nebo pro uživatelské skupiny. Systém dle nastavených oprávnění přihlášeného uživatele automaticky sestaví navigaci systému a kontroluje uživatelovu práci se systémem. Pokud se uživatel pokusí porušit svá oprávnění či jinak napadnout systém, je daný pokus zachycen a zaznamenán jádrem systému, které okamžitě informuje správce systému emailem (viz. podkapitola 4.4.10).

Systém dále automaticky monitoruje veškeré změny v systému, jakékoliv chyby či výjimky, SQL dotazy ale i samotnou práci všech uživatelů. Je tak zabezpečeno brzké odhalení i sebemenšího náznaku problému či zpětné dohledání viníka problému. Systém je také umožněno a doporučeno pravidelně zálohovat, čímž je zamezeno ztrátě dat v případě poruchy či neodborného uživatelského zásahu do uložených dat.

Pokud by byl požadavek na vyšší bezpečnost uložení dat, je možno použít metody třídy *SecurityFunctions* (viz. podkapitola 4.4.5) pro šifrování a dešifrování vkládaných dat.

Systém svými vlastnostmi vyhovuje ISO 27001:2005, což je certifikovatelný standard a obsahuje specifikaci pro systémy řízení bezpečnosti informací.

4.7 Instalace

4.7.1 Systému

Instalace systému bude prováděna ve třech základních krocích:

1. Vytvoření účtu s přístupem k databázi,
2. nahrání souborů systému na souborový server,
3. instalace samotného systému.

Samotná instalace bude probíhat pomocí uživatalského rozhraní přes webový prohlížeč. Toto rozhraní je přístupné po zadání adresy *https://adresa_serveru/install/*[1]. Rozhraní provede uživatele instalací v několika krocích:

1. Dle zadaných hodnot bude vytvořen konfigurační soubor `config.php` s přístupovými údaji k databázi,
2. budou vytvořeny databázové tabulky systému a budou naplněny výchozími daty,
3. bude provedeno nastavení systému dle uživatelského zadání.

Po kompletním provedení instalace by měl být adresář install z bezpečnostních důvodů smazán (umožnil by opětovnou změnu nejen přístupových údajů k databázi, ale i změnu nastavení systému jako takového).

[1]Pokud bude systém nahrán do adresáře, musí se samozřejmě objevit v cestě ke skriptu i daný adresář, dané adresáře.

4.7.2 Instalace modulu

Instalaci modulu umožní modul konfigurace (viz. podkapitola 4.5.2) pomocí jednoduchého uživatelského formuláře. Bude nutné vždy nejdříve nahrát soubory modulu na souborový (webový) server a poté provést samotnou instalaci pomocí uživatelského rozhraní.

4.8 Uživatelské rozhraní

Uživatelské rozhraní a celé zobrazení systému zajišťuje značkovací jazyk *HTML* generovaný pomocí programovacího jazyka *PHP*.

Pro pohodlné, snadné a přehledné uživatelské rozhraní jsou využívány funkce jazyka *JavaScript* a technologie *Ajax*. Práce se tak stáva pro uživatele snadnější, příjemnější a intuitivnější. Pomocí jazyka *JavaScript* je zajišťováno spousta pomocných zobrazení, nápověd a jiných uživatelských pomůcek. Systém není ovšem na těchto technologiích závislý a je plně funkční i bez jejich použití. Slouží tedy skutečně jen k vylepšení a zpříjemnění uživatelského rozhraní, což je i jejich skutečným posláním (aplikace závislá na těchto technologiích neodpovídá základním pravidlům přístupnosti). Dané technologie nejsou totiž vždy plně podporovány a uživatel jejich podporu nemůžu často ani ovlivnit.

4.9 Implementace

Implementace daného systému je pro implementátora velice snadná pro libovolné použití. Systém a jeho jádro je plně připraveno k implementaci různých projektů, bez nutnosti úprav samotného systému. Systém je právě proto koncipován velice univerzálně a především modulárně. Vše se dá velice snadno nastavit a použít díky připravenému přívětivému uživatelskému rozhraní. Případné chyby implementátora jsou ihned zachyceny a oznámeny jádrem a nedochází tak k žádným bezpečnostním rizikům. Jádro jako takové nepustí ani implementátora (natož uživatele) k „surovým“ datům a veškerá uživatelská data očišťuje od nebezpečného kódu před jejich samotným použitím. Implementátor má také svojí práci velmi usnadněnou předpřipravenými metodami a funkcemi systému, které jsou mu k dispozici. Veškeré systémové třídy, metody a funkce jsou pečlivě komentovány pomocí nástroje *phpDocumentor* s ukázkami použití. Implementace se tak stává snadná i pro méně zkušeného implementátora (programátora).

5 ZÁVĚR

Jedním z hlavních výstupů diplomové práce je navržený datový model, který splňuje veškeré požadavky modulárního systému. Datový model je plně optimalizován pro splnění všech požadovaných funkcí a pro zajištění univerzálnosti systému. Datový model je přiložen ve formě obrázku A3 (viz. příloha A).

Dle provedeného návrhu a datového modelu byl realizován vlastní webový modulární systém. K realizaci bylo použito objektově orientovaného programování v programovacím jazyce PHP. Prvotní realizace byla provedena ve spolupráci s databází MySQL, ovšem systém není omezen a může být použit ve spolupráci s libovolnou relační databází podporující SQL dotazovací jazyk. Postupně bylo realizováno samotné jádro systému a jaderné moduly. Jádro systému umožňuje automatické načítání modulů dle požadavků uživatele, což byl základní požadavek (modularita). Realizované moduly byly zaměřeny na umožnění jednoduché a komplexní správy systému, které by umožňovaly systém zcela osamostatnit od různých nástrojů a řešení druhých stran (například co se správy databáze týče). Celý systém byl realizován nezávisle na použité platformě (operačním systému) serveru a nezávisle na použité databázi. Klient může být pro použití systému vybaven libovolným operačním systémem a webovým prohlížečem.

Systém má připravený šablonovací systém pro moduly tvořící frontend systému. Šablonovací systém je tříúrovňový, a to rozlišením na layout (hlavní vzhled celého systému včetně administrace), šablony (ovlivnění vzhledu frontendových stránek) a stránky (postavené na šablonách). Tento šablonovací systém budou využívat moduly tvořící frontend, jejichž realizace a popis nebyly cílem této práce. Systém je plně připraven pro optimalizaci frontendové části pro vyhledávací stroje (SEO optimalizace).

Systém je ve své backendové části plně multijazyčný a je připraven i pro plnou multijazyčnost v části frontendové. Je tedy připraven pro multijazyčné nasazení a vytváření multijazyčných webových systémů.

Velký důraz ve všech fázích vývoje systému byl kladen na jeho bezpečnost. Systém je zabezpečen proti dnes známým a používaným útokům na webové systémy. Systém svými vlastnostmi vyhovuje ISO 27001:2005, což je certifikovatelný standard a obsahuje specifikaci pro systémy řízení bezpečnosti informací.

Systém převyšuje většinu dnes používaných i komerčních webových modulárních systémů. Byl již nasazen v praxi na několika projektech a tím byly ověřeny jeho vlastnosti a chování v zatížení běžného provozu. Výhodami tohoto řešení jsou nejen modularita, vysoká úroveň zabezpečení, ale i velké možnosti sledování chodu systému, hledání případných problémů a tím usnadnění a urychlení jejich řešení. Správce systému má tak plnou kontrolu nad systémem díky jeho navrženým

vlastnostem a jaderným modulům, aniž by musel využívat řešení druhých stran.

Systém může být použit jako základ pro webový systém libovolného druhu. Použitím požadovaných modulů je umožněno systém zaměřit v publikační systém, v systém pro správu dokumentů i v jakýkoliv jiný internetový, intranetový či extranetový systém.

LITERATURA

[1] GILMORE, Jason. *Velká kniha PHP a MySQL*. Praha: Zoner Software s.r.o., 2005. 712 s. ISBN 80-86815-20-X.

[2] SCHLOSSNAGLE, George. *Pokročilé programování v PHP5*. Praha: Zoner Software s.r.o., 2004. 640 s. ISBN 80-86815-14-5.

[3] KOSEK, Jiří. *Tvorba intraktivních internetových aplikací*. Praha: Grada Publishing, 1999. 491 s. ISBN 80-7169-373-1.

[4] GABARRO, Steven. *Web Application Design and Implementation: Apache 2, PHP5, MySQL, JavaScript, and Linux/UNIX*. New Jersey: John Wiley & Sons, 2007. 315 s. ISBN 0-471-77391-3.

[5] SUCHÝ, Petr. *Návrh systému pro správu dokumentů*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2006. 86 s. Vedoucí bakalářské práce Ing. Vít Vrba.

[6] SUCHÝ, Petr. *Publikační systémy (CMS)*. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2005. 59 s. Vedoucí semestrální práce Ing. Vít Vrba.

[7] Apache Software Foundation. *The Apache HTTP Server Project* [online]. c2008, poslední revize 25.4. 2008 [cit.2008-04-25]. Dostupné z: <http://httpd.apache.org/>.

[8] MySQL AB. *MySQL* [online]. c1995-2008, poslední revize 13.5.2008 [cit.2008-05-13]. Dostupné z: <http://www.mysql.com/>.

[9] PHP Group. *PHP: Hypertext Preprocessor* [online]. c2001-2008, poslední revize 13.5.2008 [cit.2008-05-13]. Dostupné z: <http://www.php.net/>.

SLOVNÍK POJMŮ

A

Ajax - Z anglického Asynchronous JavaScript and XML, je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. Využívá technologií HTML (XHTML) a CSS pro prezentaci, DOM a JavaScript pro zobrazování a dynamické změny v prezentaci a XMLHttpRequest pro asynchronní výměnu dat s webovým serverem (typicky je užíván formát XML).

Apache - Webový server, produkt nadace Apache Software.

B

Backend systému - Tento pojem se používá pro část systému, která slouží k administraci a není veřejně přístupná. Ke vstupu do backendu je vyžadována autentizace. Opak k frontendu systému.

Brute force attack - Útok označovaný jako „brute force attack“ (útok hrubou silou) je většinou slovníkový útok, kdy se algoritmus neustálým odesíláním formuláře snaží zjistit správné heslo různou kombinací znaků, dokud není úspěšný.

C

Certifikát - Dokument, který prokazuje nějakou skutečnost.

CMS - Jde o zkratku z anglického výrazu Content Managment System - tedy publikační systém.

Cookies, též HTTP cookie - Využívají se pro ukládání dat zaslaných ze strany serveru do souborového systému na straně klienta (internetového prohlížeče). Uživatel může mít ukládání cookies zakázáno, proto se na jejich uložení nelze plně spolehnout.

D

DMS - Jde o zkratku z anglického výrazu Document Managment System - tedy systém pro správu dokumentů.

E

Extranet - Jedná se o intranet přístupný pro uživatele z vnější.

F

Frontend systému - Tento pojem se používá pro tu část systému, která slouží k veřejné prezentaci. Ke vstupu do frontendu není obvykle vyžadována autentizace. Opak k backendu systému.

G

GMT (Greenwich Mean Time) - Čas na nultém poledníku v Greenwichi (Londýn).

GNU/GPL licence - „GPL“ znamená „General Public License“, což se dá do češtiny přeložit jako „Všeobecná veřejná licence“. Nejrozšířenější takovou licencí je GNU General Public License, zkráceně GNU GPL. Toto může být ještě dále zkráceno na GPL, pokud je z kontextu zřejmé, že se mluví o té GPL z projektu GNU.

H

HTTP - Zkratka z anglického výrazu Hypertext Transfer Protocol - jedná se o bezestavový protokol, definující pravidla přenosu textu, grafik, videa a dalších dat přes World Wide Web (WWW), původně určený pro výměnu hypertextových dokumentů ve formátu HTML. To že je protokol bezstavový znamená, že zpracovává veškeré požadavky bez jakékoliv znalosti předchozích či následných požadavků. Protokol HTTP verze 1.1 je definován v RFC 2616. Defaultně pro přenos dat používá TCP port 80.

HTTPS - Nadstavba protokolu HTTP, která poskytuje zvýšenou bezpečnost před odposloucháváním či podvržením dat. Data jsou přenášena pomocí HTTP, ale jsou šifrována pomocí SSL nebo TLS, což zaručuje ochranu proti *packet-sniffingu* i *man-in-the-middle* útokům. HTTPS implicitně komunikuje prostřednictvím TCP portu 443.

HTML - Zkratka z anglického HyperText Markup Language, jde o značkovací jazyk pro vytváření stránek v systému World Wide Web (umožňující publikaci dokumentů na Internetu).

I

Implementace - Uskutečnění, naplnění, realizace, dostání závazku.

Implementátor - Člověk provádějící implementaci (uskutečnění, naplnění, realizaci).

Interface, Object Interface, nebo-li rozhraní - V objektově orientovaném programování je tořeno kódem, který specifikuje jaké metody musí třída implementovat, bez definice jak mají tyto metody pracovat.

Internet - Celosvětová počítačová síť, která spojuje jednotlivé menší sítě pomocí sady IP protokolů.

Intranet - Systém či počítačová síť, která je soukromá a je určená pro malou skupinu uživatelů (např. zaměstnanci podniku).

J

JavaScript - JavaScript je multiplatformní, objektově orientovaný skriptovací jazyk, jehož autorem je Brendan Eich z tehdejší společnosti Netscape. Jedná se o kód zpracovávaný na klientské straně pomocí webového prohlížeče, který musí JavaScript podporovat (jinak je JavaScriptový kód ignorován).

K

Konstruktor - Jedná se o metodu třídy, která je volána vždy při vytvoření objektu třídy (jeho inicializaci).

M

MD5 - Jde o jednosměrný hašovací algoritmus využívany často k hašování hesel, nebo ke kontrole integrity souborů.

MyISAM - Jeden z typů databázových tabulek, který využívá databáze MySQL pro uložení dat do databáze.

MySQL - Jedná se o relační databázový server, produkt firmy MySQL AB. Nynější aktuální verzí je verze 5.

O

Open source, také open-source software (OSS) - Je počítačový software s otevřeným zdrojovým kódem.

P

PHP - Skriptovací jazyk na jehož počátcích byl Rasmus Lerdorf. Zkratka pochází z anglického Personal Home Page, nyní se však označuje spíše jako Hypertext Preprocessor. Jazyk PHP umožňuje spouštět na straně serveru skripty, pomocí nichž obsluhuje požadavky klienta. Nynější aktuální verzí jazyka je verze 5.

phpDocumentor - Formální standard pro komentování kódu jazyka PHP. Umožňuje automatické generování dokumentace (dostupný na adrese http://www.phpdoc.org/, aktuální verze 1.4.2).

phpMyAdmin - Open source aplikace pro snadnou správu MySQL databází (dostupná na adrese http://www.phpmyadmin.net/, aktuální verze 2.11.6).

R

Relace Session - Relace mezi klientem a serverem - umožňuje ukládání SESSION k zaznamenávání informací o činnosti klienta.

S

SEO - Z anglického výrazu Search Engine Optimization (volně přeloženo jako „optimalizace pro vyhledávací stroje“). Jde o metodologii vytváření a upravování webových stránek takovým způsobem, aby byly ve výsledcích hledání v internetových vyhledávačích zobrazeny na co nejlepších místech. Umožňuje však i snažší orientaci uživatele a příjemnější interpretaci URL adres.

Sessions - Využívají se pro zpracování relací - používá se v komplexních webových aplikacích pro přizpůsobení aplikace specifickému chování uživatele, jeho sledování a záznam akcí provedených uživatelem. Každý návštěvník webu je označen jednoznačným ID, označovaným jako ID relace (Session ID, SID). Toto SID se poté může dávat do vzájemného vztahu s dalšími daty a tak můžeme např. mapovat akce provedené uživatelem, přizpůsobovat chování aplikace dle předešlých požadavků a mnoho

dalšího. Relace sesssion se používá i pro autentifikaci uživatele, který se ověřuje dle přiřazeného SID (dokud je stále stejné i s dalšími případnými kontrolními údaji, máme jistotu, že je přihlášen stále stejný uživatel, ze stále stejného počítače i internetového prohlížeče).

Session ID, též SID - Jednoznačný identifikátor relace session.

Session handler - Slouží pro obsluhu (tvoření, ukládání, vyvolávání a mazání) session. PHP obsahuje základní Session handler, který tyto informace ukládá do filesystemu na fileserver. PHP však umožňuje vytvoření vlastního Session handleru, pro jiné zpracování session. Této možnosti se využívá především ve větších systémech, kde jsou session hojně využívány a je tedy výhodnější a rychlejší ukládat session do databáze.

SHA-1 - Jednosměrný hašovací algoritmus.

SQL 99 - Jde o poslední standard jazyka SQL, též nazývaný SQL 3.

SQL Injection - Technika napadení databázové vrstvy programu vsunutím (odtud „injection“) kódu přes neošetřený vstup a vykonání vlastního či pozměněného SQL dotazu.

Supervizor (též supervisor, superuživatel) - Jedná se o hlavního administrátora systému. Je to uživatel systému s nejvyššími oprávněními a přístupem do všech částí systému.

T

TIMESTAMP - Čas uložený v sekundách od 1. ledna 1970 00:00:00 GMT.

U

UNIX - je víceúlohový a víceuživatelský operační systém (Původně Unics, podle Unary Information and Computing Service).

URL adresa - Zkratka z anglického výrazu Uniform Resource Locator - jedná se o globální adresu dokumentů a jiných zdrojů na internetu. Její první část určuje jaký protokol se má v komunikaci použít a druhá specifikuje IP adresu, nebo doménové jméno, kde je umístěn zdroj požadované informace (požadovaných dat). URL adresa by měla být v celé síti jednoznačná.

W

Whitespace, také netisknutelný znak - Rozumí se znak (nebo shluk více těchto znaků), který reprezentuje v horizontální či vertikální rovině místo (mezeru).

WWW, též web - Zkratka anglického výrazu World Wide Web - ve volném překladu „Celosvětová pavučina“, je označení pro aplikace internetového protokolu HTTP. Je tím myšlena soustava propojených hypertextových dokumentů.

WYSIWYG editor - jedná se o editor textových dokumentů typu aplikace Microsoft®Word. WYSIWYG je zkratka z anglického výrazu What You See Is What You Get - tedy volně přeloženo jako „co vidíš, to dostaneš“.

X

XSS - Zkratka z anglického Cross-site scripting - znamená vložení nebezpečného kódu do systému pomocí vstupů systému (např. v podobě formulářů či URL adresy).

SEZNAM PŘÍLOH

A DATOVÝ MODEL

Viz. přiložený obrázek formátu A3.

B VIDEO UKÁZKY SYSTÉMU

V celé práci je popisován návrh a realizace vlastního webového modulárního systému. Pro prezentaci vytvořeného systému byla vytvořena videa s ukázkami systému.

Na přiloženém CD jsou dvě video ukázky:

- *xsuchy09 - 01 - Realizace webového modulárního systému - instalace, demo realizace.avi* - ukázka instalace a základní práce s realizovaným systémem, představení jaderných modulů a samotného systému,
- *xsuchy09 - 02 - Realizace webového modulárního systému - ukázka implementace.avi* - ukázka implementace webového projektu na realizovaném systému, spolu s ukázkou frontendové části (tvořenou především modulem stránky a dále moduly čtenáři, katalog a eshop).

Obě ukázky jsou slovně okomentovány.

C PŘILOŽENÉ CD

Obsah CD je znázorněn na obrázku C.1, kde:

- Adresář *DBDesigner* obsahuje podadresáře:
  - *Datový model* - obsahuje soubor `datovy-model.png`, což je exportovaný datový model do obrázku formátu `png` (viz. příloha A),
  - *HTML report* - obsahuje soubor `report.html`, což je HTML report datového modelu z programu DBDesigner verze 4.0.5.6 Beta,
  - *SQL* - obsahuje soubor `skript.sql`, což je SQL skript pro vytvoření navrhnutého datového modelu v databázi MySQL,
- adresář *LaTeX* obsahuje zdrojový kód tohoto dokumentu,
- adresář *Odkazy a přístupy* obsahuje textový soubor s URL adresou a přístupovými údaji do dema realizovaného systému a URL adresu s ukázkou realizovaného projektu na daném systému,
- adresář *Videa* obsahuje video ukázky realizovaného systému (instalace systému, práce se systémem, praktické ukázky).

V „kořenovém adresáři“ CD je přiložen tento dokument.

Obr. C.1: Obsah přiloženého CD

REJSTŘÍK