Webové stránky základní školy

Aleš Bartoš

Bakalářská práce
2018

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2017/2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Aleš Bartoš**
Osobní číslo:	**A15044**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Webové stránky základní školy**
Téma anglicky:	**Web Pages for a Primary School**

Zásady pro vypracování:

1. **Analyzujte požadavky na komunikaci mezi učiteli, žáky a rodiči.**
2. **Navrhněte komunikační portál ke sdílení informací.**
3. **Dbejte na správné strukturování informací.**
4. **Vytvořte vhodnou správu uživatelů.**
5. **Věnujte zvýšenou pozornost zabezpečení.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **GILMORE, W. J. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Nové, 3. vyd. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2011. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.**
2. **GUTMANS, Andi, Stig S?ther BAKKEN a Derick RETHANS. Mistrovství v PHP 5. Brno: CP Books, 2005. ISBN 80-251-0799-X.**
3. **CHAFFER, Jonathan a Karl SWEDBERG. Mistrovství v jQuery: [kompletní průvodce vývojáře]. Brno: Computer Press, 2013. Mistrovství. ISBN 978-80-251-4103-8.**
4. **KOLEKTIV AUTORŮ. Mistrovství v PHP 5. Vyd. 2. Brno: Computer Press, 2007. ISBN 978-80-251-1519-0.**
5. **KOLEKTIV AUTORŮ. Vytváříme webové aplikace v PHP5, MySQL a Apache. Brno: Computer Press, 2006. ISBN 80-251-1073-7.**
6. **PHP: Hypertext Preprocessor [online]. ?2001-2017 [cit. 2017-11-13]. Dostupné z: http://php.net/**
7. **OWASP Foundation. [Online]. [Cit. 2017-11-13] Dostupné z https://www.owasp.org/.**

Vedoucí bakalářské práce: **doc. Ing. Martin Sysel, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **27. července 2018**

Termín odevzdání bakalářské práce: **28. srpna 2018**

Ve Zlíně dne 15. prosince 2017

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Hlavním cílem této bakalářské práce je vytvořit webovou aplikaci pro základní školu zabývající se sdílením výukových materiálů a informacím žákům a rodičům. Aplikace je napsána v PHP. Na backendu funguje open source framework Nette. Frontend je vytvořen v CSS s využitím Bootstrapu. Aplikace pracuje s databází MySQL.

Práce řeší stávající problém konkrétní základní školy, pro sdílení informací a výuky pro žáky pomocí síťové komunikace. Kde se na webhosting nahraje statická webová stránka, která neobsahuje žádné zabezpečení ani oprávněný uživatelský přístup. Tady by mohl nastat problém s nepřehledností a strukturováním dat. Proto ve své práci vytvářím informační a výukový systém, který zamezí možné nepřehlednosti a chybám ve strukturování dat, který lze uplatnit i všeobecně.

Klíčová slova: Webová aplikace, CMS, MVC, Informační a výukový portál

ABSTRACT

The main object of this bachelor´s thesis is to create a web application for elementary schools that deals with sharing educational material and information with pupils and parents. This application is written in PHP. The open source framework Nette works on the backend. Frontend is created CSS using Boostrap. The application saves data to a MySQL database and is working with them afterwards.

The work deals with the ongoing issue of a specific elementary school with sharing information and education for pupils using network communication, where a static web page is uploaded on webhosting, which doesn´t include any security nor authorized user access. This is where a big problem happens through confusion and structuring of the data. For this case, an informational and educational system will be made for a current issue , that can be used in general as well.

Keywords: Web Application, CMS, MVC, Information and Learning Portal

Rád bych poděkoval vedoucímu mé bakalářské práce, panu doc. Ing. Martinu Syslovi, Ph.D., za odborné vedení, cenné rady a věcné připomínky, které mi poskytoval při zpracování této práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- Že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis diplomanta

OBSAH

ÚVOD

V dnešní době je již internet velmi důležitým a užitečným nástrojem každodenního lidského života. V mnoha ohledech je jeho role nezastupitelná. Internet je hlavní prostředek komunikace a sdílení informací. Pomocí internetu dokážeme získat potřebné informace jen pomocí pár kliků myší na webových katalozích či webových stránkách.

Webové stránky jsou nedílnou součástí internetu a jejich počet exponenciálně stoupá a denně přibydou stovky takových stránek. Můžeme je prohlížet, číst články nebo komunikovat se známými apod. Bez jejich existence bychom byli oběti příkazového řádku a sofistikovaných způsobů komunikace, které by obyčejný smrtelník bez specifických znalostí nebyl schopen využít.

Všechny nově vznikající webové stránky se musí vyrovnat s příchodem nových směrnic jako jsou GDPR. Tento požadavek klade mnohem větší důraz na znuvupoužitelnost a rychlost tvorby. Proto se začali vytvářet tzv. CMS a obdobné redakční systémy, které urychlují práci a hlavně se nemusí neustále opakovat tytéž příkazy. CMS si můžeme stáhnout a upravit dle vlastních potřeb, anebo v nejlepším případě vytvořit si vlastní.

S velkým přibývajícím procentem uživatelů na internetu a webových stránek, které jsou využívány, vzrůstají také útoky na výše zmíněné objekty, a proto práce bude postavena na výkonném a vysoce zabezpečeném frameworku Nette, který má v sobě ošetřené jak základní bezpečnostní prvky, tak také ty sofistikovanější. Určitě je výhodnější využít nějakého osvědčeného frameworku, než začít řešit bezpečnostní problémy, jak se lidově říká, od píky.

Cílem práce je vytvořit informační a výukový portál pro školství, díky kterému bude uživatel (učitel) moci jednoduše a efektivně administrovat jak svůj předmět, tak i svou třídu. Naopak ze strany žáků a rodičů se budou získávat informace a potřebné materiály o úkolech, učebních materiálech a nadcházejících povinnostech. Celý systém bude vysoce zapečený jak z technického hlediska, tak z osobního. Žákům usnadní práci jak při běžném studiu jako takovém, tak v případě nepřítomnosti, například ze zdravotních důvodů.

I. TEORETICKÁ ČÁST

1 WEBOVÉ STRÁNKY

Webová stránka je soubor nebo soubory dat, která zobrazuje webový prohlížeč v přívětivé uživatelské formě. Pracuje pomocí tzv. WWW (world wide web), kde data jsou prezentována pomocí hypertextu, který je tvořen pomocí značkovacího jazyka HTML či XHTML. Celá stránka se skládá s těchto značek (tagů) dále textu, multimediálních dat a odkazů vedoucí na další stánky. [5]

1.1 Responzivita

Od dne 5. listopadu 2007, kdy na veřejnost vstoupil první operační systém Android od společnosti Google pro mobilní zařízení, se webové stránky musely začít adaptovat. Již nestačilo mít pevně dané rozměry webové stránky a jednotky výjimek pro nejpoužívanější webové prohlížeče. [9]

S rozmachem systému Android přišla velká revoluce mobilů a tabletů, kde každé zařízení mělo jiné zobrazovací a technické parametry. Android hlavně disponoval velice stabilním prohlížečem, který jeho předchůdci (jako byl např. Symbian) neměli tak propracovaný. [9] Lidé začali používat mobil v každodenním životě a webové stránky, které nebyly responzivní se dlouho načítaly nebo se zobrazovaly v nepoužitelném stavu, proto takové stránky nebyly navštěvovány a ztrácely potencionální návštěvníky.

1.1.1 CCS 3

Je třetí verzí tzv. kaskádových stylů neboli (Cascading style sheets 3), které slouží k nastolování HTML tak, aby stránka byla uživatelsky přijatelná. CSS 3 byla vydaná již v roce 2005, ale pouze jako pre-alfa a postupně se přidávaly další funkce a implementovaly se do webových prohlížečů. Důležité datum pro CCS 3 je 19. června 2012, kdy byla vydána funkce Media Queries, která usnadnila responzivitu webů.

1.2 Dynamické web stránky

Dynamická stránka se aktualizuje v závislosti na čase, aktivitě návštěvníka apod. Z pohledu uživatele mění svůj vzhled a obsah. [5] Taková stránka využívá technologie jako jsou PHP, Ajax, Javascript atd. pomocí těchto technologií stránka reaguje na pohyb myši, identifikuje uživatele, generuje obsah v závislosti aktivity návštěvníka (nákupní košík v e-shopu), matematické výpočty, odesílání dat pomocí formulářů apod.

1.2.1 Jak dynamické stránky fungují

Aby dynamické stránky fungovaly, je zapotřebí skriptovacích jazyků většinou ve spolupráci s databázemi, aspoň na větších projektech. Skripty lze rozdělit na:

- Na straně serveru (PHP)
- Na straně klienta (Javascript ve zkratce JS)
- Kombinace serveru a klienta (AJAX)

1.2.1.1 Na straně serveru

Základním princem je, že uživatel skrze stránku pošle dotaz na server, kde se provede rutinní logika a výsledek se pošle zpátky uživateli jako nová stránka s aktualizovanými daty. Výhodou takového procesu je bezpečnost, kde se data odesílají jako výstup a bezpečnostní data zůstávají na serveru chráněna, např. hash heslo. [5]

Nejsou zde jenom výhody, ale jsou zde i nezanedbatelná úskalí, jako třeba velká zátěž komunikace mezi klientem a serverem, ze které vychází náročný proces, při kterém dochází ke zpomalení akce hlavně při pomalejším internetovém připojení.

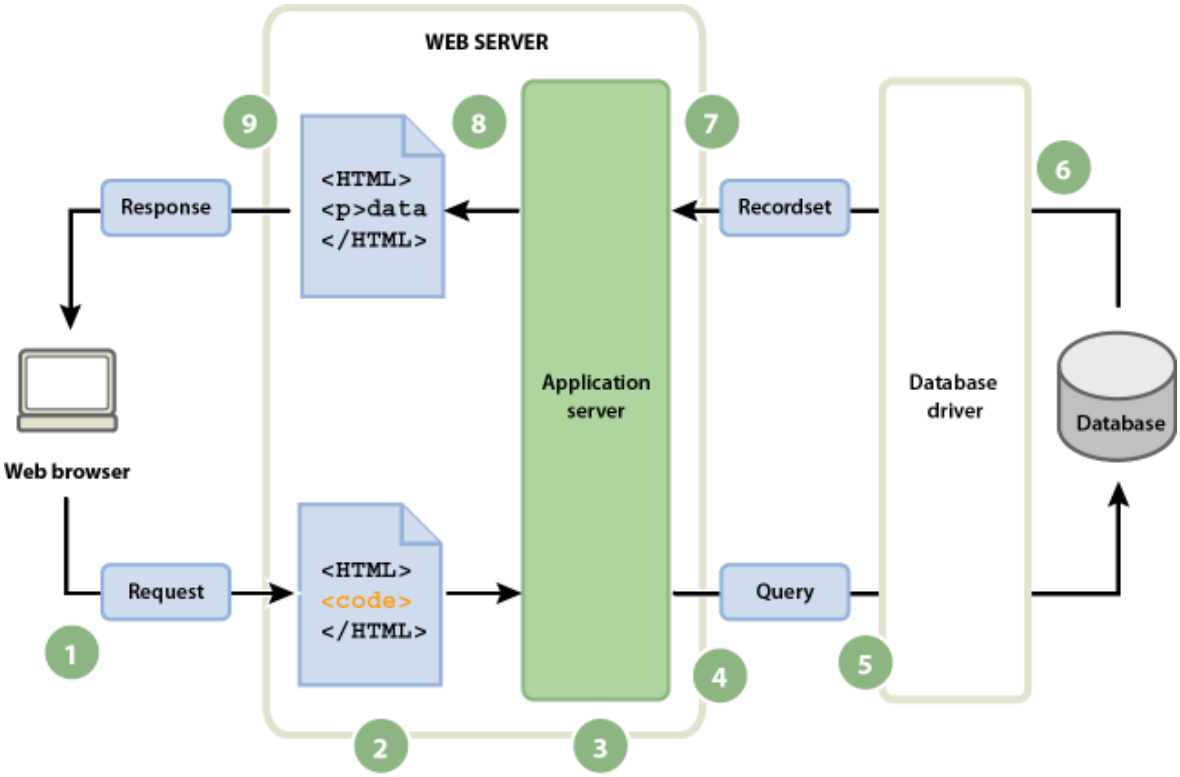
1.2.1.2 Na straně klienta

Na straně klienta je to o mnohem jednoduší. Zde se o to stará JS, který v dnešní době dokáže spoustu užitečných věcí. Hodně se mluví o knihovně jQuery, která v sobě nese nespočet zjednodušujících funkcí usnadňujících práci a také odstraňující výpočetní složitosti. [3] Přednostní určitě je eliminace potřeby komunikovat se serverem, což urychlí celý proces. Také dokáže v reálném čase reagovat na uživatelské chování.

JS by se neměl používat na všechno. Například pro bezpečnostní účely je JS velice nepraktický a nebezpečný. Hlavním důvodem je, že se zobrazuje na straně klienta a lze s ním lehce manipulovat. Je tedy jasné, že nejlepší variantou je používat kombinaci serveru a klienta. [5]

1.3 Webová aplikace

Webová aplikace je kolekcí statických a dynamických webových stránek. Častou chybou se stává, že lidé chápou pojem webová aplikace a dynamické stránky jako jeden termín. Jedná se kombinaci statických, dynamických stránek a databáze. Jejich realizaci si ukážeme na obrázku životního cyklu webové aplikace. [5]



Obrázek 1 - Životní cyklus webové aplikace (zdroj: https://helpx.adobe.com/)

Prohlížeč pošle žádost o dynamickou stránku, kde webový server vyhledá stránku a pošle ji aplikačnímu serveru. Poté aplikační server najde stránce instrukce a dokončí přijatou stránku, kterou hned poté posílá zpátky na webový server. Webový server zpracuje požadavek a odesílá již kompletní požadovanou stránku.

1.4 Databáze

Každá vetší webová aplikace potřebuje uchovávat systém souboru záznamů, díky kterému budeme moci měnit jeho obsah. Databáze může byt pouhý textový soubor, který v sobě nese patřičné údaje o uživatelích, které dostaneme z jeho přečtení. Takový systém je však zastaralý a velice nepraktický.

V dnešní době jsou databáze automaticky brány jako sofistikovaný a odladěný nástroj. Řeší za nás mnohé problémy a díky jeho jednoduchosti se ho naučí používat víceméně každý. Pomocí navrženého jazyka SQL, který vychází z logické interpretace lidského požadavku na vyhledávání je použití elementární. S nadsázkou lze říct, že píšeme kód stylem „vlož do tabulky uživatele jeho jméno: Petr, příjmení: Mazáník heslo: 32SDA5sda5Sc… last_log: time stamp“

user_id	name	surname	password	last_log
1	Petr	Mazáník	32SDA5sda5Sc…	12.6.2016
2	Ludmila	Zábranská	A&da52PorsD…	2.5.2018
3	Josef	Babica	ađŁßs56SdS…	3.8.2017

Tabulka 1 - Část tabulky *users*

1.4.1 MySQL

Je rozšířený model databáze pro webové aplikace uplatňující relační model databáze. Celá komunikace funguje pomocí SQL jazyka plus vlastní rozšíření, co přidala po převzetí firma Oracle Corporation

MySQL databáze se soustředí hlavně na rychlost. Také nabízí několik typů databázových uložišť pro různé typy projektů. Typy takových uložišť jsou:

- ARCHIVE
- BLACKHOLE
- **CSV**
- MEMORY
- **InnoDB**
- MARGE
- **MyISAM**

Zvýrazněné typy jsou nejvíce používané. Ostatní jsou ukázkou, které se používají zřídka nebo jako dozvuk minulosti.

1.4.2 Relační databáze

Jde o databázi, která je založena na principu tabulek, jak jsme si ukázali výše, jen s tím rozdílem, že více tabulek může mezi sebou navazovat relace (neboli vztah mezi nimi), který propojuje požadované informace na požadovanou událost. Například máme tabulku user, propojovací tabulku mezi nimi (říkejme jí propojovací) a tabulku role, která nám nese informaci o možných rolích v systémy. Díky propojovací tabulce můžeme uživatelům přidělovat role díky relaci M:N.

user_id	Name	surname	info_id	role
1	Petr	Mazáník	1	učitel
2	Ludmila	Zábranská	NULL	žák

3	Josef	Babica	NULL	žák

Tabulka 2 - Výpis relační tabulky

V tabulce je ukázka logiky na následující strukturu databáze, kde by uživatel s rolí žák nikdy neměl mít žádný údaj ve sloupci info_id, protože informace může zasílat jednotlivým třídám pouze jejich učitel nebo ředitel školy.

1.4.3 Procedury

Procedura v databázích se dá chápat jako funkce událostí, které se mají vykonat. Jedná se o část programu, který je funkčně oddělený od svého okolí. Komunikuje pomocí modulů programu, který na jeho zavolání proceduru vyvolá. Využívá se zejména hlavně pro četné opakovací události.

2 VÝUKOVÉ A INFORMAČNÍ SYSTÉMY

Jedná se o takový webový portál nebo přímo aplikační software, který pomáhá zvládnout každodenní administrativu pomocí systému, který je připraven na změny v rozvrhu, zapomenuté žákovské knížky, komunikaci mezi školou, rodinou a spoustu dalších funkcí, které ušetří desítky hodin času, a to jak škole, tak rodičům.

Takový systém si segment školství nechává vytvořit přesně podle svých parametrů, které odpovídají požadavkům dané školy nebo se adaptují na již vytvořený systém, který se zakoupí.

2.1 Základní funkce výukových a informačních systémů

Výukové a informační systémy obsahují základní funkce pro jejich strukturu. Specifická jsou:

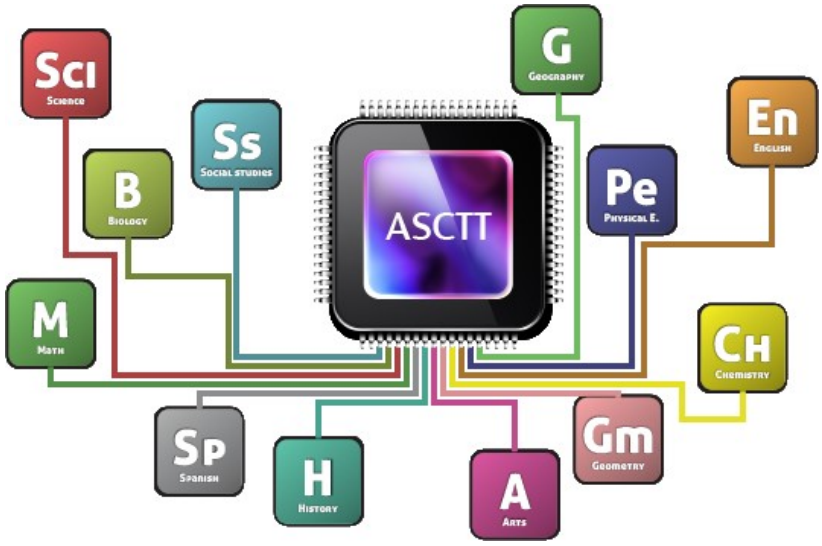
- Sdílení informací a dat
- Hierarchie uživatelů
- Aktuálnost (real time)
- Stabilita a robustnost
- Jednoduchost
- Zálohy

2.1.1 Pokročilé funkce

- Rozvrh
- Klasifikace
- Absence
- Mobilní verze
- Knihovna
- Technická podpora
- Propojení student a rodič

2.2 Asctimetables

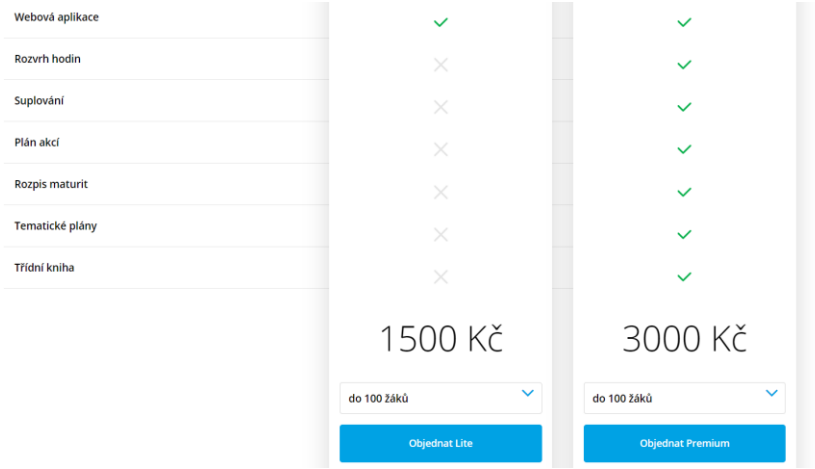
Webový portál zaměřující se hlavně na tvorbu tzv. dokonalých rozvrhů. Soustředí se na rychlou a snadnou tvorbu rozvrhu pro jakoukoliv školní instituci. Obsahuje také rozšířenější funkce jako import dat, suplovaní učitelů, třídní knihu apod.



Obrázek 2 - Asctimetables (zdroj: https://www.asctimetables.com/)

2.3 Bakaláři

Nejrozšířenější systém v ČR s nejdelší tradicí. Obsahuje nespočet funkcí a rozšiřujících modulů. Technická podpora pro systém je 24 hodin denně. Všechny tyhle přednosti mají za důsledek vysokou pořizovací a udržovací cenu. Zakoupené verze programu obsahují několik dílčích částí, které nelze separovat od celkového modulu. Některé funkce tedy škola ani nevyužije.



Obrázek 3 - Ceník systému Bakaláři (zdroj: https://www.bakalari.cz/)

2.4 Ostatní systémy

Výše uvedené systémy patří mezi nejrozšířenější a nejpopulárnější systémy pro výuku a sdílení informací. Internet nám však nabízí spoustu dalších jako jsou:

- dm Software
- eTřídnice
- iškola
- SAS
- Škola OnLine
- Moodle

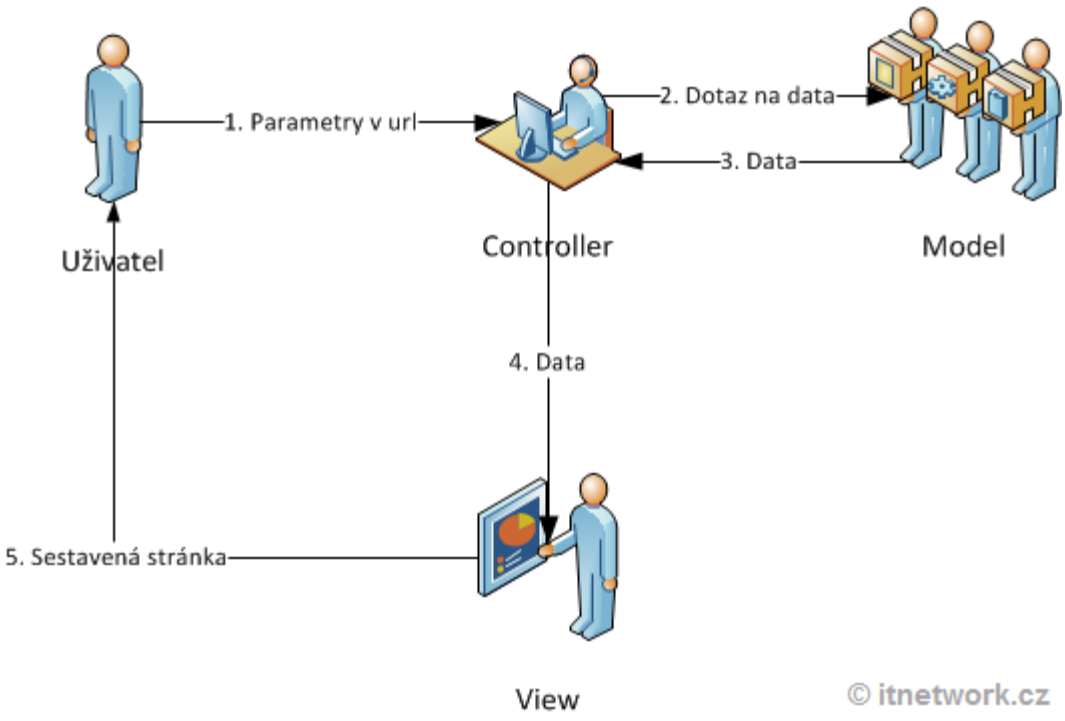
3 MVC ARCHITEKTURA

Je nejpoužívanější modelový vzor, který se využívá zejména na webových stránkách, i když jeho historie začala na desktopech. MVC architektura se uchytila především na webových frameworcích, jako jsou např. Zend, Nette, Laravel apod.

Základním stavebním kamenem MVC je oddělení logiky od výstupu. Řeší problém, kdy máme v jednom souboru logické operace zároveň s renderováním výstupu. Tomuto problému se také říká „špagetový kód“. V jednom souboru máme tedy Databázové dotazy, logiku a neuspořádané HTML. [8]

Takový kód sice funguje, ale z hlediska dlouhodobého provozování webu je velice nepraktický. Kdybychom chtěli přidat například novou funkci nebo tu, která závisí na předešlé verzi, musela by se pracně předělávat velká část kódu.

Aplikace se dělí na 3 typy komponentů, hovoříme o modelu view (pohled) a controlle (kontrolorech), z kterého vychází zkratka MVC. Kde komponenty jsou třídy oddělené od abstraktních tříd MVC.



Obrázek 4 - Diagram MVC architektury (zdroj: https://www.itnetwork.cz/)

3.1 Model

Logika, výpočty, databázové dotazy, validace apod. spadá pod správu Modelu. Model se nestará o vystup. Jeho hlavní práce je příjem údajů a vydání výsledných dat zpátky. [8] Není potřeba, aby model věděl, odkud data přišla, o existenci view nebo controller neví.

Při ORM (objektově – relační Mapovaní) koresponduje model přímo s databázovými tabulkami. Kde model máme např. uživatel nebo článek a jejich instance jsou atributy přímo z databáze. U uživatele to bude atribut jméno, příjmení a tak dále.

3.2 View

Neboli česky pohled se zabývá zobrazením výstupu uživateli. Z velké časti to bývají HTML šablony obsahující HTML stránku a tagy nějakého značkovacího jazyka. Díky tomuto systému je možné vkládat do šablony různé operace a cykly apod. bez velké námahy. Každý pohled je logicky rozdělen do kategorií, aby vypisoval jen určité údaje. Všechny tyhle šablony lze vkládat do sebe, aby se zamezilo vytváření duplicitních kódů.

Nejedná se o pouhou šablonu, ale zobrazovač výstupu. Obsahuje jen elementární logiku, která je pro výpis nutná. View stejně jako model vůbec netuší, odkud mu data přišla, jediné co má za práci, je zobrazit je uživateli. [8]

3.3 Controller

Aby to celé fungovalo, nesmí se zapomenout na controller, který celému systémů dodá funkčnost. Dá se chápat jako takový prostředník, se kterým je ve spojení uživatel, model a view. Spojuje celý systém a drží ho pohromadě a všechny komponenty propojuje. Celý cyklus života stránky si ukážeme níže.

3.4 Cyklus života stránky

Všechno začíná u uživatele, který zadá do prohlížeče adresu webu a parametry. Požadavek jako první zachytí tzv. router. Ten podle parametrů pozná, který controller voláme. Tímto nám řekne, jakou podstránku si přeje zobrazit např. www.utb.cz/fai/uzivatel/5, čímž říká, že chce zobrazit uživatele s identifikačním číslem 5. [8]

Požadavek putuje do tzv. routeru, který podle parametru pozná, jaký controller voláme. Podle požadavku controller pozná, co se po něm požaduje. Jeho další akce je, že zavolá

model, díky kterému vyhledá uživatele a vrací jeho údaje, které si controller ukládá do proměnných. Po dokončení tohoto procesu začne rendrovat view. Přijatá data od controller view vloží do připravené šablony a tím je naše stránka hotová a zobrazená uživateli. [8]

4 FRAMEWORK NETTE

Jak bylo již výše zmíněno framework Nette je open source software pro tvorbu webových aplikaci v PHP. Jeho velkou předností je eliminace bezpečnostních rizik, produktivita a čistý kód. Výhodou Nette je velká aktivní komunita v ČR. A jedná se o 3 nejpopulárnější framework na světě. [4]

4.1 Instalace Nette

Nastavíme si server na základní požadavky pro správnou funkci PHP. Buď stáhneme příslušný program např. Wamp server. Také je možnost založit si projekt na webhostingu zcela zdarma na doméně třetího řádu a nainstalovat Nette. Většina společností, která provozují web hostingové služby, by měla mít tyhle minimální požadavky znázorněny na obrázku níže.

Obrázek 5 - Minimální požadavky pro chod Frameworku Nette (zdroj: https://nette.org)

4.2 Composer

Nástroj pro správu závislosti v PHP. Dokáže vytvořit tzv. skeleton, dříve nazývaný jako sandbox webové aplikace a spojit dohromady závislosti jednotlivých knihoven. Stačí si stáhnout instalátor v podobě .exe souboru a naistalovat.

V dokumentaci knihoven (Nemusí se jednat přímo o Nette. Je spousta dalších projektů, které dokáže composer naistalovat.) najdeme příkaz, který nám vytvoří základ projektu. U Nette spustíme příkazový řádek, navolíme výchozí adresář (nejlépe přímo na lokálním severu), kde chceme projekt vytvořit a pomocí níže uvedeného příkazu nainstalujeme. [11]

```
composer create-project nette/sandbox nazev-projekt
```

4.3 Adresářová struktura nette

```
sandbox/
├── app/                  ← adresář s aplikací
│   ├── config/           ← konfigurační soubory
│   │   ├── config.neon   ← konfigurační soubor
│   │   └── config.local.neon
│   │
│   ├── forms/            ← třídy formulářů
│   ├── model/            ← modelová vrstva a její třídy
│   ├── presenters/       ← třídy presenterů
│   │   ├── HomepagePresenter.php  ← třída presenteru Homepage
│   │   └── templates/    ← adresář se šablonami
│   │       ├── @layout.latte ← šablona společného layoutu
│   │       └── Homepage/     ← šablony presenteru Homepage
│   │           └── default.latte  ← šablona akce default
│   ├── router/           ← třídy routerů
│   │
│   └── bootstrap.php     ← zaváděcí soubor aplikace
│
├── log/                  ← obsahuje logy, error logy atd.
├── temp/                 ← pro dočasné soubory, cache, ...
│
├── vendor/               ← adresář na knihovny (např. třetích stran)
│   ├── nette/            ← všechny knihovny Nette Frameworku
│   │   └── nette/Nette   ← oblíbený framework nainstalovaný Composerem
│   ├── ...
│   │
│   └── autoload.php      ← soubor který se stará o načítání tříd nainstalovaných
│
└── www/                  ← veřejný adresář, document root projektu
    ├── .htaccess         ← pravidla pro mod_rewrite
    ├── index.php         ← který spouští aplikaci
    └── images/           ← další adresáře, třeba pro obrázky
```

Obrázek 6 - Adresářová struktura (zdroj: https://nette.org)

4.4 Vlastnosti Nette

Jedná se o soubor PHP funkcí a tříd, které slouží jako usnadňující nástroj programátorovi při vývoji webových aplikací. Jeho pozitivní vlastnosti jsou:

- Bezpečnost
- OOP PHP
- MVC technologie
- Aktivní komunita a aktuální dokumentace na stránkách
- Rozšiřující balíčky funkcí
- Vysoká výkonost

4.5 Zabezpečení před zranitelností

Nette klade velký důraz na bezpečnost a snaží se držet krok s dnešním světem a zacelovat všechny díry v systému. Jako podklad možných útoků a chyb si Nette jako většina světa bere příklad z dobročinné organizace OWASP. Díky tomu je ošetřena před spousty typů útoků od těch nejjednodušších (BFA) až po ty složitější (CSRF). [11]

5 ZABEZPEČENÍ

Každý systém může být napaden mnoha způsoby, od serveru ať po klienta. Pomineme-li zabezpečení samotného webového serveru, nejčastěji k útokům dochází mezi prohlížečem a aplikací. Tyto útoky by se daly rozdělit na tři typy:

- Aplikační
- Komunikační
- Na straně klienta

5.1.1 OWASP

Mezinárodní organizace představující ustanovení technických odborníků pro největší slabiny webu a komunikace na internetu neboli Open Web Application Security Project. Zaměřuje se především na bezpečnost softwaru. [7]

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Obrázek 7 - OWASP top 10. 2013 vs 2017 (zdroj: https://www.owasp.org/)

5.1.1.1 Injection

Nejčastějším typem je SQL injection, který je častý problém mnoha webů. Tenhle typ útoku probíhá tak, že útočník upravuje SQL dotaz (nejčastěji SELECT, UPDATE, INSERT a WHERE). Útočník najde nechráněný formulář, který posílá dotazy na aplikační server do databáze. [7] Pomocí speciálních znaků jako je apostrof, uvozovky a pomlčka, přidá nebo pozmění SQL dotaz a bez velkých problémů se může dostat do backendu webové aplikace.

5.1.1.2 Cross-site scripting (XSS)

Metoda narušení webových stránek zneužívající neošetřených výstupů. Díky tomu útočník může změnit stránku dle vlastní úvahy a tím dezinformovat nebo dokonce získat citlivé údaje o uživateli. [7]

5.1.1.3 Cross-site request forgery (CSRF)

Základní myšlenka CSRF je přinutit uživatele navštívit stránku, který tajně provádí útok na webovou aplikaci, na kterou je uživatel zrovna přihlášen. Pomocí toho lze pozměnit nebo smazat článek bez povšimnutí uživatele. [7] Nette tenhle útok ošetřuje pouze jednou metodou *addProtectin();* která se přiřadí k nechráněnému formuláři. [11]

6 GDPR

Je novinka letošního roku, která se zabývá ochranou osobní údajů neboli Obecné nařízení na ochranu osobních údajů, ve zkratce GDPR (General Data protection regulation). GDPR bylo přijato již v roce 2016 s účinností od 25. května 2018, kde se bude jednat o nejvíce ucelený soubor pravidel o ochraně dat na světě.

Firmy, instituce i fyzické osoby, které nakládají s osobními údaji, jejíž ochranou se nařízení zabývá, budou muset zabezpečit ochranný režim osobních údajů v souladu s přijatou směrnicí. V případě závažného porušení GDPR budou hrozit vysoké pokuty ve výši deseti tisíce eur nebo 4 % celkového obratu společnosti. [10]

6.1 Dopady GDPR

Změny, které sebou nese směrnice GDPR dopadnou na každého, kdo pracuje při svém podnikaní s osobními údají. Každý správce i zpracovatel osobních údajů musí dodržovat GDPR po celou dobu zpracování. S tím přichází velká technická a administrativní zátěž, kde se bude muset manipulovat jen s nezbytnými osobními daty a přikládat řádný dokument, že se tak koná.

GDPR obsahuje nezanedbatelnou část z již právně platné dosavadní úpravy ČR. Toto zabezpečuje Úřad pro ochranu osobních údajů (ÚOOÚ). GDPR má však sofistikovanější systém, který zavádí nové povinnosti např. pro zpracovatele údajů, kteří byli kryti jen subjektem správce údajů.

Správci musí umožnit přístup k údajům, které jsou o subjektech shromažďovány. To vyžaduje další modifikace nynějších systémů nevyhovujících směrnici. Ideální přístup by měl být přímý a online, i když to není přímou podmínkou. Občan má nárok na rozšířené právo na výmaz ze systému, jinak řečeno právo být zapomenut, které nesmí být zbytečně odkládáno bez doložení právního důvodu např. v případě trestní činnosti.

Velkou změnou projde definice osobní údaj, kde budou přidány technické parametry subjektu nebo například genetické a biometrické údaje. [10]

Obavu vzbuzuje především skutečnost oznamovací povinnosti v případě narušení bezpečnosti údajů například hackerským útokem. Tuhle skutečnost ilustruje kauza Yahoo, kdy v letech 2013 a 2014 došlo ke dvěma masivním únikům dat, kde se jednalo o miliardy údajů.

Yahoo několik měsíců tuhle skutečnost tajila až v roce 2016 pod tlakem medií vyšla s pravdou ven.

6.2 Co jsou osobní údaje

Osobní údaje definuje GDPR jako údaje, kterými mi je pohlaví, věk, datum narození, osobní stav, dále IP adresu, fotografický záznam, rasový či etnický původ, politické zaměření, náboženské nebo filozofické vyznání a údaje o trestních deliktech či pravomocném odsouzení. Směrnice se vztahuje i na fyzické osoby v podnikaní, kde jsou zahrnuty i organizační údaje jako e-mailová adresa, telefonní číslo či identifikační údaj daný státem.

Vyloučeny z působnosti GDPR jsou anonymní údaje o zemřelých osobách a dále také údaje nabité v rámci činnosti čistě osobní povahy, které nemají obchodní či institucionální charakter. Jedná se tedy o údaje zpracované pro osobní potřebu, které není potřeba sdílet veřejně.

6.2.1 Genetické údaje

Jsou osobní údaje o získaných či zděděných genetických znacích dané fyzické osoby, které vyplývají z analýzy biologického vzorku dotčené fyzické osoby nebo z analýzy jiného prvku umožňujícího získat rovnocenné informace. Dále údaje charakterizující zdravotní stav vypovídající o duševní nebo tělesné způsobilosti člověka.

6.2.2 Biometrické údaje

Jsou osobní údaje vyplývající z technického zpracovaní fyzických nebo fyziologických znaků či znaků chovaní fyzické osoby umožňující unikátní identifikaci. Jako biometrický údaj můžeme chápat např. záznam oční duhovky, otisk prstu, fotografie obličeje, audio záznam hlasu, ba dokonce i podpis.

6.3 Povinnosti institucí a firem

Každá firma a instituce má povinnost zavést procesní, organizační a technická opatření bez ohledu na velikost nebo počet zaměstnanců v souladu s DGPE. A to v bodech:

- Implementace ochrany dat
- Pseudonymizace osobních údajů
- Vedení záznamu o činnostech zpracování
- Jmenovaní DPO (Data Protection Officer)
- Posouzení vlivu na ochranu osobních údaji neboli v anglickém jazyce DPIA

DPIA je horkou novinkou. Tento dokumentu musí instituce či firma vypracovat, pokud jejich systém je prováděn automaticky a pracuje s rozsáhlými daty, které profiluje bez účasti člověka. Zářným příkladem jsou pojišťovny, banky a leasingové firmy. Kde algoritmus systémů porovnává osobní údaje všech uživatelů a vyhodnocuje nejpotenciálnějšího zákazníka. [10]

Další povinností je pro společnosti uchovávat záznamy o zpracovávaných subjektech. Každý správce firmy nebo zprostředkovatel bude muset spolupracovat s dozorovým úřadem a na jeho žádost mu data zpřístupnit, podobně jako občanovi zmíněnému výše.

Takový záznam musí obsahovat následující informace:

- Důvod zpracovaní
- Časový interval pro zachovaní jednotlivých dat
- Hierarchie příjemců, kterým byly nebo budou subjekty zpřístupněny
- Technické a organizační opatřeni
- Informace o mezinárodních transakcích dat
- Kontaktní údaje správce a DPO
- Důvod zpracovaní

II. PRAKTICKÁ ČÁST

7 VÝUKOVÝ A INFORMAČNÍ PORTÁL

7.1 Cíl praktické části

Cílem praktické části je vytvořit informační a výukový portál pro školství, který bude usnadňovat práci školství jako je sdílení informací a výukových materiálů žakům.

7.2 Klíčové vlastnosti

Seznam funkcí, které musí informační a výukový portál obsahovat:

- Rozdělení jednotlivých tříd
- Možnosti stáhnutí a zobrazování materiálu pro konkrétní třídu
- Přidávání učitele
- Propojení žáka a rodiče
- Registrace žáka a jeho přiřazení k určité třídě
- Automatická aktualizace každý rok

Tohle jsou základní a klíčové vlastnosti, které systém musí umět pro správnou funkčnost.

7.3 Autentizace a autorizace

Celý systém je chráněn přístupovým systémem, jak kvůli zobrazování jen podnětných materiálů, tak zabezpečením osobních údajů. Nejdřív se provede autentizace daného uživatele, následně mu přidělí roli a pokračuje do systému.

Na základě role se autorizuje stránka, jaké má uživatel práva. Žák má právo nahlížet do svých souborů a materiálů. Učitel zase přidávat úkoly, oznámení a spravovat svoji třídu.

7.4 Modulárnost

U webových aplikací je důležitým aspektem jejich modulárnost, která programátorovi nebo správci stránek umožňuje v krátkém čase a hlavně jednoduše přidávat nové prvky a funkce na stránky.

7.5 Vybrané případy užití

Náš systém, jak již bylo zmíněno výše, má 4 typy uživatelů student (rodič) učitel a admin. Student prohlíží a stahuje. Učitel přidává a edituje a admin edituje a spravuje celý systém. Vypíšeme si základní operace všech typů uživatelů.

7.5.1 Student (rodič)

- registrace pomocí klíče -> přihlášení -> zobrazení hlavních událostí -> odhlášení
- přihlášení -> třída -> zobrazení třídních událostí -> odhlášení
- přihlášení -> předměty -> zobrazení úkolů / stažení materiálů -> odhlášení
- přihlášení -> profil -> zobrazení svého profilu-> odhlášení
- přihlášení -> profil -> změnit -> Změna údajů -> odhlášení
- přihlášení -> profil -> změnit -> Změna hesla -> odhlášení
- přihlášení -> profil -> změnit -> Vyžádaní rodičovství -> odhlášení
- přihlášení -> profil -> změnit -> Vyjádření na stav rodičovství -> odhlášení

7.5.2 Učitel

- přihlášení -> zobrazení hlavních událostí -> odhlášení
- přihlášení -> třída -> vybrat třídu -> výběr třídy-> odhlášení
- přihlášení -> třída -> Změna klíče -> odhlášení
- přihlášení -> předměty -> zobrazení předmětů -> odhlášení
- přihlášení -> předměty -> zobrazení svých úkolů -> odhlášení
- přihlášení -> předměty -> přidaní úkolu -> odhlášení
- přihlášení -> předměty -> smazaní svých úkolu -> odhlášení
- přihlášení -> profil -> zobrazení svého profilu-> odhlášení
- přihlášení -> profil -> změnit -> Změna údajů -> odhlášení
- přihlášení -> profil -> změnit -> Změna hesla -> odhlášení
- přihlášení -> profil -> změnit -> přihlásit se do třídy -> odhlášení
- přihlášení -> profil -> změnit -> odhlásit se ze třídy -> odhlášení

7.5.3 Admin

- přihlášení -> zobrazení všech událostí -> odhlášení
- přihlášení -> smazání události -> odhlášení
- přihlášení -> editace události -> odhlášení
- přihlášení -> přidaní události -> odhlášení
- přihlášení -> třída -> zobrazení všech tříd -> odhlášení
- přihlášení -> třída -> změnit klíč -> odhlášení
- přihlášení -> předměty -> zobrazení všech předmětu -> odhlášení
- přihlášení -> předměty -> přidat předmět-> odhlášení

- přihlášení -> předměty -> editovat předmět -> odhlášení
- přihlášení -> předměty -> smazat předmět -> odhlášení
- přihlášení -> předměty -> přidání úlohy -> přidat úlohu -> odhlášení
- přihlášení -> předměty -> smazat úlohu -> odhlášení
- přihlášení -> profil -> zobrazení svého profilu-> odhlášení
- přihlášení -> profil -> změnit -> Změna údajů -> odhlášení
- přihlášení -> profil -> změnit -> Změna hesla -> odhlášení
- přihlášení -> uživatelé -> zobrazení uživatelů -> odhlášení
- přihlášení -> uživatelé -> přidaní uživatele -> odhlášení
- přihlášení -> uživatelé -> smazaní uživatele -> odhlášení
- přihlášení -> uživatelé -> editovat -> editace uživatele -> odhlášení
- přihlášení -> nový rok -> začít nový rok -> odhlášení

8 KONFIGURACE FRAMEWORKU NETTE

Aby se práce stala plnohodnotným projektem, je potřeba vycházet ze správné konfigurace frameworku, který se nastavuje v souborech *config.neon* a *config.local.neon*. Zápis je prováděn pomocí formátu NEON.

V *config.locl.neon* nastavíme základní parametry pro přihlášení databáze. Jako je databázový server, název databáze, login a heslo. To je důležitý zaklad, bez kterého bychom se nepřipojili k DB. Poté upravíme sekundární parametry jako jsou *Debugger* (panel v Tracy baru), *explain* (dotazy v Tracy baru) nebo *autowired*. [11]

8.1 Vypnutí cache a ACL

Vývoj aplikace je náročný proces a není na škodu si ho co nejvíc ulehčit a zpříjemnit. Vypnutím *cache* si ušetříme mnoho problémů, zvlášť když pracujeme na localhostu. Díky deaktivaci *cache* zamezíme reakci na změny jako je např. přímý zásah do databáze. V konfiguračním souboru *config.neon* stačí napsat následující řádky pro jeho deaktivaci. Jedná se o speciální implantované uložiště, které vlastně neukládá.

```
services:
    cacheStorage:
        factory: Nette\Caching\Storages\DevNullStorage
```

Obrázek 8 - Ukázka kódu pro deaktivaci cache (zdroj: vlastní)

Dále je vhodné vypnout ACL, také pomocí konfiguračního souboru. Tohle se hodí především u jednodušších aplikacích nebo když si vytváříme vlastní přístupovou logiku. [11] Vypnout ACL lze napsaním *false* v sekci *aclAllowed*. Prostým zásahem do konfiguračního souboru nám tak odpadne nutnost změny kódu napříč celou aplikací. [6]

8.2 Session

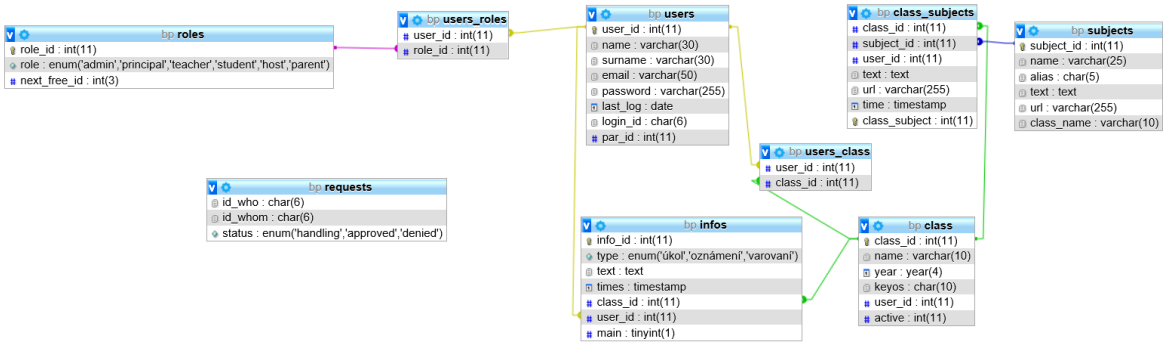
Je permanentní spojení mezi klientem a serverem, dokáže uchovávat data, které jsme nasbírali procházením určitého webu. Používá se zejména na e-shopech, my ho ale použijeme k uchování ID přihlášeného uživatele a jeho role. Také v případě, že je na studenta přihlášený jeho rodič.

9 DATABÁZE

Celá aplikace běží na MySQL databázi, která je nejvíc rozšířená pro webovou komunitu. Pro MySQL existuje PHP nástroj *phoMyAdmin,* který byl použit pro zaplnění testovacích dat.

9.1 Schéma databáze

Databáze je tvořena takovou strukturou, aby splňovala všechny patřičné podmínky a nemusela se modifikovat. Samozřejmě celý systém je navržen tak, aby šla provádět jak migrace, tak přidávat novou strukturu možnosti pro výukový portál.



Obrázek 9 - Schéma databáze (zdroj: vlastní)

Základní tabulkou je *users,* která je určená pro záznam uživatelů a je vázána relací M:N k tabulce *roles*, která určuje roli v systému daného uživatele například ředitele, učitele nebo žáka. Dále je významná tabulka *class* nesoucí informaci o startu studia, její ročník a přihlašovací klíč, díky kterému se žák přihlásí do příslušné třídy. Táhle tabulka je podobně jako *roles* propojena s tabulkou *users* pro identifikaci, který žák patří do jaké třídy a kdo je jejich třídní učitel. Dále je tabulka *class* v relaci s tabulkou *subject,* zde je systém více sofistikovaný a tabulka *subject* obsahuje všechny předměty ve škole a jejich tematicky plán. Jejich propojovací tabulku slouží k přidávání úkolů a materiálů dané třídě pro určitý předmět. V poslední řade je tu tabulka *infos.* Táto tabulka slouží pro obecné oznámení, jako jsou výlety, ředitelská volna apod. class_subject je tabulka pro přidávání konkrétních úkolů konkrétním třídám. Nezbytnou tabulkou je *requests*, která pracuje se žádostmi o rodičovství a jejich aktuálními stavy.

9.2 Atributy v databázi

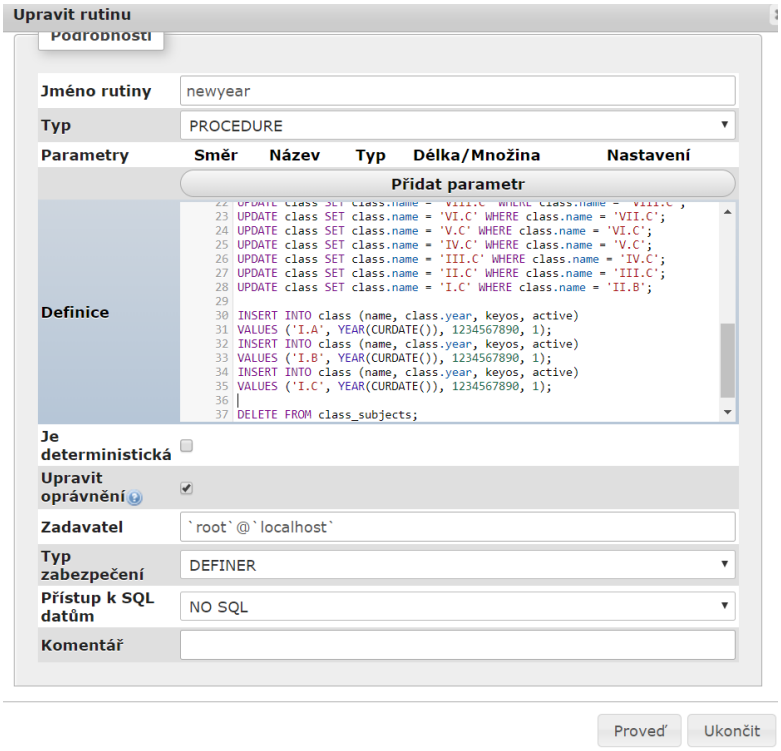
- Users
    - User_id – identifikační číslo uživatele (primární klíč)

  - Name – jméno uživatele
  - Surname – příjmení uživatele
  - Email – email uživatele
  - Password – heslo uživatele
  - Last_log – poslední přihlášení
  - Login_id – přihlašovací kód
  - Par_id – identifikační číslo rodiče
- Roles
  - Role_id – identifikační číslo role (primární klíč)
  - Role – Role uživatele (Admin, teacher, student, host)
  - Next_free_id – Dostupné identifikační číslo kódu
- Users_roles
  - User_id – identifikační číslo uživatele (cizí klíč)
  - Role_id – identifikační číslo role (cizí klíč)
- Class
  - Class_id – identifikační číslo třídy (primární klíč)
  - Name – jméno třídy (I.A, IX.B apod.)
  - Year – start studia (rok)
  - Keyos – klíč k zápisu do třídy
  - Activate – bool pro zjištění aktivní třídy
- Subject
  - Subject_id – identifikační číslo předmětu (primární klíč)
  - Name – plný název předmětu
  - Alias – název předmětu ve zkratce
  - Text – obecný popis předmětu a jeho osnova
  - url – adresa souboru pro předmět
  - class_name – Naze třidy pro daný předmět
- Class_subject
  - Class_id – identifikační číslo třídy (cizí klíč) třída daného úkolu
  - Subject_id – identifikační číslo předmětu (cizí klíč) předmět daného úkolu
  - User_id – identifikační číslo uživatele (cizí klíč) autor úkolu
  - Text – text ohledně daného úkolu
  - URL – adresa k případnému materiálu

  - Time – čas vzniku události
  - Class_subject – identifikační číslo úkolu (primární klíč)
- Infos
  - Info_id – identifikační číslo (primární klič)
  - Class_id – identifikační číslo třídy (cizí klíč)
  - User_id – identifikační číslo uživatele (cizí klíč)
  - Text – text informace
  - Type – typ události (oznámení, úkol, varovaní)
  - url – rrl adresa souboru materiálu
  - time – čas vzniku události
  - main – hlavní příspěvek
- Requests
  - Id_who – ID odesilatele (rodiče)
  - Id_whom – ID příjemce (student)
  - Status – stav požadavku

9.3 **Procedury**

Procedura se nám náramně hodí pro jednu konkrétní akci jako je událost nový rok. Kde se vykoná mnoho příkazů naráz. Jestli je výhodnější nechat procedury na úrovni databáze nebo programového modelu je diskutabilní otázka. Co člověk, to názor.



Obrázek 10 - Vytváření procedury (zdroj: vlastní)

10 ZABEZPEČENÍ V NETTE

Velmi často je hlášena bezpečnostní díra na dalším významném webu, nebo je díry zneužito. A taková situace je nepříjemná. Pokud vám záleží na zabezpečení vašich webových aplikací, je Nette Framework zcela jistě tou nejlepší volbou. [10]

10.1 Oprávnění

Po přihlášení se uživatel autorizuje a hned se mu přiřadí identifikační číslo a role. Díky těmto dvěma údajům může nahlížet do šablon a pracovat s presentery. Metody presenteru jsou z velké časti podmíněny argumentem, který bez jejích hodnoty nefunguje.

```
public function handleDelete($id)
{
    $this->ClassManager->GetClass()->where('user_id',$id)->update([
    'user_id' => 0
    ]);
}
```

Obrázek 11 - Ukázka kódu reset třídního učitele (zdroj: vlastní)

Třeba tato nebezpečná metoda, která odstraní třídu uživateli, kde je hodnota uživatele rovna příchozímu argumentu.

10.2 Zabezpečení proti XSS

Cross-Site Scripting je metoda narušení webových stránek zneužívající neošetřených výstupů. Nette Framework přichází s revoluční technologií Context-Aware Escaping, která vás provždy zbaví rizika Cross-Site Scriptingu. Všechny výstupy totiž ošetřuje automaticky a tak se nemůže stát, že by kodér na něco zapomněl. Přiklad v šabloně {$výstup}.

10.3 Zabezpečení proti CSRF

Cross-Site Request Forgery je útok spočívající v tom, že přimějeme uživatele navštívit stránku, která vykoná útok na webovou aplikaci, na které se zrovna nachází uživatel. Lze takto modifikovat článek bez povšimnutí uživatele. Proti útoku se lze bránit generováním a ověřováním autorizačního tokenu.

Ochránit formulář před útokem Cross-Site Request Forgery lze v Nette Frameworku pouhým jedním příkazem *$form->addProtection();*

10.4 **Zabezpečení proti Injection**

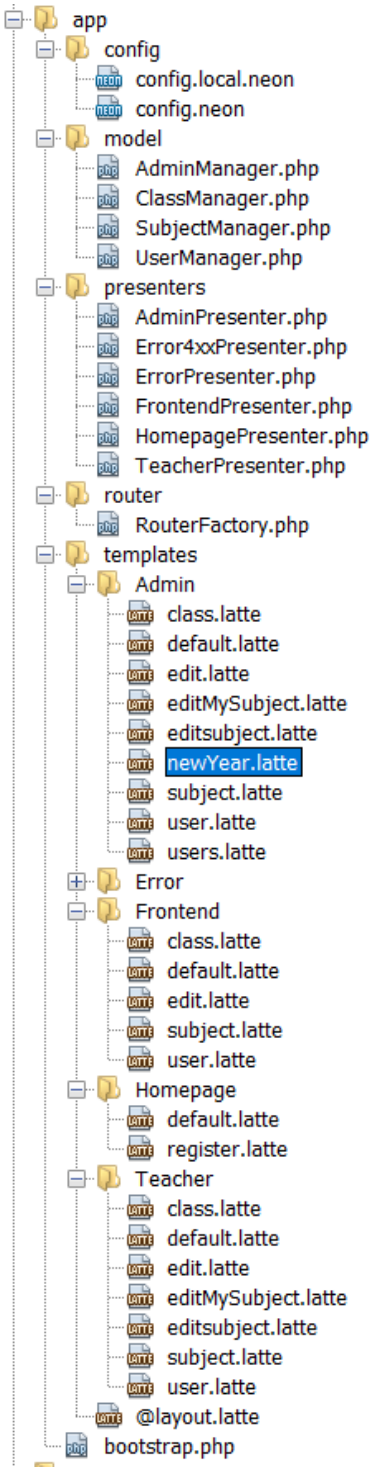
Nette využívá pro práci s databázi ovladač PDO. Tento ovladač není jen snahou zjednodušit už tak jednoduchý jazyk, ale zvýšit jak modulárnost, tak především jeho zabezpečení.

```
public function GetClassByTeacherID($ID) {
    return $result = $this->database->fetchField('SELECT name FROM class WHERE user_id = ?', $ID);
}
```

Obrázek 12 - Příklad zápisu v PDO (zdroj: vlastní)

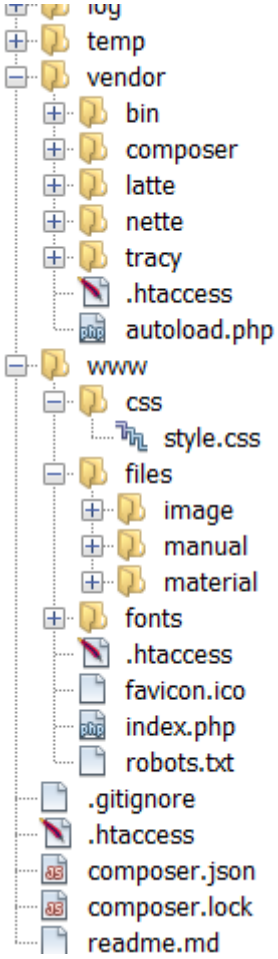
Celý ovladač má v sobě zahrnuté všechny možné metody injection. Takže nehrozí žádná modifikace bez vůle uživatele.

11 ADRESÁŘOVÁ STRUKTURA



Obrázek 13 - Adresářová struktura složky app (zdroj: vlastní)

Po dokončení konfigurace Nette se vytvořila základní adresářová struktura pro webovou aplikaci. Tamplete se dal o adresář výš kvůli lepší přístupnosti. Založil se adresář model, kde se vytvořily 4 základní modely pro naši práci. Tyhle modely jsou především pro manipulaci s databází. Pro jejich editaci vypisovaní apod.

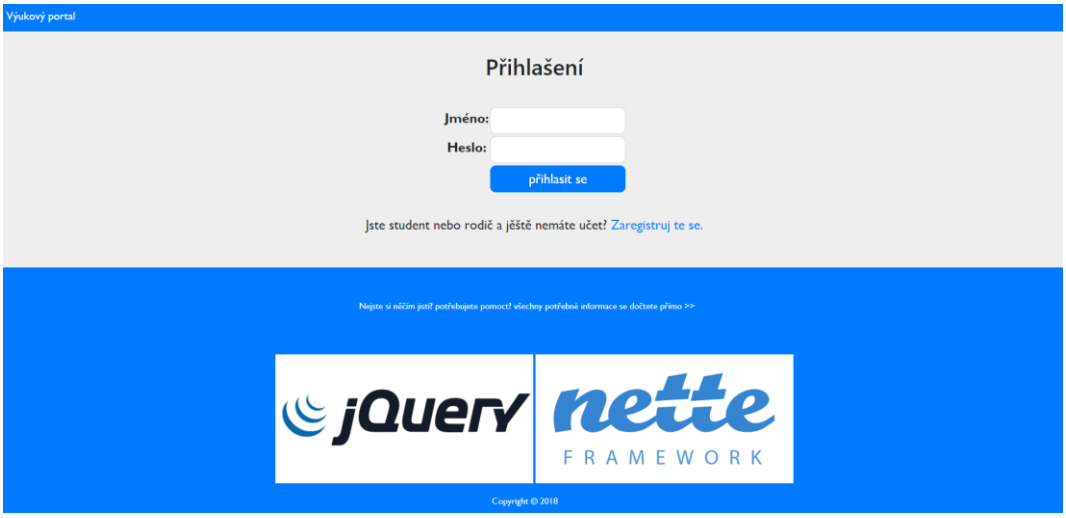


Obrázek 14 - Adresářová struktura ostatní (zdroj: vlastní)

Kořenový adresář www slouží pouze k přesměrovaný na presenter webové aplikace. Zde lze upravit jen .htaccess nebo robot.txt pro indexovaní stránky vyhledávacími katalogy. Dále zde máme adresář pro materiály, fonty a styly. Ke každému z adresářů se dostaneme pomocí *{$basepath}*.

12 AUTENTIZACE A AUTORIZACE

Po zadaní webové adresy na naši webovou aplikaci se zobrazí výchozí šablona pro přihlášení (také pro registraci pro nové žaky, jak je zvykem u každé stránky). Přihlašovací údaje se skládají z přihlašovacího kódu a hesla.



Obrázek 15 - Ukázka přihlašovaní (zdroj: vlastní)

Po odeslání formuláře se data pošlou na presenter, který pošle požadavek na model pro autentizaci, aby zjistil, je-li uživatel v databázi uložen. Model nám vrátí uživatele dle přihlašovacích údajů a presenter mu přidělí identifikační číslo, roli a patřičné místo v aplikaci.

```
public function loginFormSucceeded(UI\Form $form, $values)
{
    if($this->UserManager->GetUserByEmail($values->name)->email != NULL) {
        if(hash('sha512', $values->password.'@lg@vi') == $this->UserManager->GetUserByEmail($values->name)->password) {


            //$user->login($username, $password);
            $this->getSession('myApp')->id = $this->UserManager->GetUserByEmail($values->name)->user_id;
            $this->getSession('myApp')->role = $this->UserManager->GetUserByEmail($values->name)->role;
            $this->flashMessage('Byl jste úspěšně registrován.');
            if($this->getSession('myApp')->role == 'admin')
                $this->redirect('Admin:');
            if($this->getSession('myApp')->role == 'teacher')
                $this->redirect('Teacher:');
            if($this->getSession('myApp')->role == 'student')
                $this->redirect('Frontend:');
        }
        elseif(hash('sha512', $values->password.'@lg@vi') == $this->UserManager->GetUserByEmail($values->name)->password) {
            $this->getSession('myApp')->role = 'student';
            $this->getSession('myApp')->parent = 1;
            $this->flashMessage('Byl jste úspěšně registrován. JAKO rodic');
            $this->redirect('Frontend:');
        }
        else
            $this->flashMessage('spatné Heslo.');
    }
    else $this->flashMessage('Saptny email.');
```

Obrázek 16 - Autorizační kód (zdroj: vlastní)

Tímto krokem se uživatel autorizuje a pokračuje již na správné hierarchické vrstvě, kterou má povolenou. Všechny vyšší vrstvy práv jsou mu zablokovány a ani přes přímou URL adresu se nedostane, například do rozhraní pro admina.

12.1 Přihlašovací kód

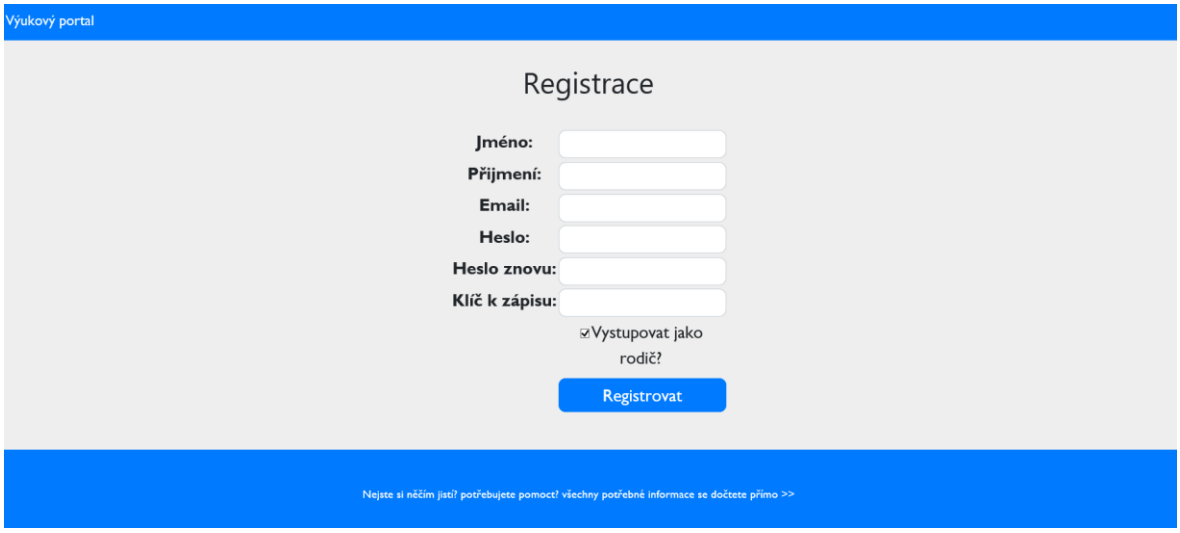
Přihlašovací kód se skládá ze šesti znaků. První znak určuje role uživatele (A – admin, U – učitel, S – student, R – rodič), druhý znak je rok registrace (např. rok 2018 = 18). Poslední tří ciferní číslo slouží jako identifikační číslo, které se vytváří podle pořadí registrace.

Obrázek 17 - Schéma přihlašovacího kódu (zdroj: vlastní)

Velká výhoda použití tohoto systému je vyhnutí se problémům ohledně GDPR. Zde se pracuje jen s pouhým kódem a neunikají žádné osobní údaje (jméno, příjmení, email apod.) dokud to není nezbytně nutné.

12.2 Registrace

Účet lze získat dvěma způsoby, buď admin vloží uživatele přímo do databáze a potom předá přihlašovací údaje dané osobě (tímhle způsobem se budou přidávat do systému hlavně učitelé, protože jedině admin může nastavit jejich roli na *teacher* a tím jím dát veškerá práva), anebo klasickou registrací, která je dostupná přes hlavní přihlašovací stránku, jak je zvykem na většině webů.



Obrázek 18 - Registrační formulář (zdroj: vlastní)

Přihlásit se může kdokoliv, kdo má přihlašovací klič k zápisu, který identifikuje, do které třídy se má student zařadit. Tenhle formulář je zejména pro studenty a rodiče. Klíč k zápisu získají od vyučujících, kteří spravují třídy. Pokud se jedná o rodiče, zaškrtne se patřičné tlačítko (tzv. checkbox) a bude do systému přiřazen pod rolí rodič a bude tak dále vystupovat.

Potřebné údaje jako je typ uživatele a přihlašovací kód se odesílají na zadanou emailovou adresu pro případ zapomenutí těchto údajů.

13 MVC

Základní myšlenkou MVC architektury je oddělení logiky od výstupu. Řeší tedy problém tzv. "špagetového kódu", kdy máme v jednom souboru (třídě), logické operace a zároveň renderování výstupu. Soubor tedy obsahuje databázové dotazy, logiku (např. PHP operace) a různě poházené HTML tagy. [11]

13.1 MODEL

Modely se navrhly tak, aby odpovídaly základní struktuře webové aplikaci a obsahovaly všechny potřebné funkce. Máme tedy tyhle modely:

- AdminManager
- ClassManager
- SubjectManager
- UserManager

Všechny zmíněné moduly jsou především databázové dotazy s logikou, které vrací potřebná data. Kde dostává v argumentech potřebná pravidla pro výstup, který následně posílá presenteru. Jedná se o krátké příkazy, které ošetřují logiku výpočet nebo například validaci.

```
public function InsertSubject($data) {
    $result = $this->database->table('subject')->insert($data);
}
public function DeleteSubject($id) {
    $result = $this->database->table('subject')->where('info_id',$id)->delete();
}
public function GetUsers(){
    return $result = $this->database->table('users');
```

Obrázek 19 - Ukázka kódu modulu (zdroj: vlastní)

Nette složku pro moduly přímo nevytváří. Hodně jednoduchých stránek se bez nich obejde, např. mikro weby. Pro takové stránky stačí mít statický obsah, která se nezmění ani za deset let.

13.2 Controller (presenter)

Presentery obsahují potřebné události, které zpracovávají data a posílají je do šablon. V presenterech je nejzákladnější logika programu. Presenter spojuje všechny tři základní složky (uživatel, model, pohled). Díky tomu aplikace funguje modulárně a každá svoje logika programu je oddělená od té druhé. Naše presentery jsou:

- AdminPresenter
- FrontendPresenter
- HomepagePresenter
- TeacherPresenter

Aby se na stránce objevil požadovaný obsah, musí presenter vyrenderovat výstup a ten poslat šabloně. Pro každou šablonu je unikátní vykreslovaní. Funkce pro vykreslení se značí klíčovým slovem *render* + *Název*.

```
public function renderEditsubject($ID)
{
    $this->template->logged = $this->getSession('myApp')->id;
    $this->template->role = $this->getSession('myApp')->role;
    $this->template->infos = $this->UserManager->info();
    $this->template->user = $this->UserManager->GetUserByID($this->getSession('myApp')->id);
    $this->template->subject = $this->SubjectManager->GetSubjectOneById($ID);

}
public function renderClass()
{
    $this->template->logged = $this->getSession('myApp')->id;
    $this->template->role = $this->getSession('myApp')->role;
    $this->template->classes = $this->ClassManager->GetClass();
```

Obrázek 20 - Příklad renderování (zdroj: vlastní)

Z obrázku výše vidíme, že některé rendrovací metody přijímají argument. Tenhle argument přichází od dotazovací metody skrze komunikaci HTTP. Z nejčastějšího se využívá jako posílání identifikačního čísla pro danou událost. Např. ze stránky editace uživatelů klikneme na uživatele, což nás přesměruje na stránku editaci uživatele s jeho identifikačním číslem.

Nette kvůli bezpečnosti vytvořil třídu pro vytváření formulářů. Díky kterému lze vytvořit funkční a bezpečný formulář pomocí již naprogramovaných metod.

```
protected function createComponentEditUser()
{
    $form = new UI\Form;
    $form->addProtection();
    $form->addText('name', 'Jméno:')->setRequired()->setValue($this->UserManager->GetUserByID($this->getSession('myApp')->id)->name);
    $form->addText('surname', 'Přijmení:')->setRequired()->setValue($this->UserManager->GetUserByID($this->getSession('myApp')->id)->surname);
    $form->addEmail('email', 'Email:')->setRequired()->setValue($this->UserManager->GetUserByID($this->getSession('myApp')->id)->email);
    $form->addSubmit('as', 'Změnit');

    $form->onSuccess[] = [$this, 'checkEditUser'];

    return $form;
}
public function checkEditUser(UI\Form $form, $values)
{
    $postId = $this->getSession('myApp')->id;
    if(($this->UserManager->GetUserByEmail($values->email) == NULL) or ($values->email == $this->UserManager->GetUserByEmail($values->email)->email) ) {
        if ($postId) {
            $this->UserManager->EditUserByID($postId, $values);
            $this->flashMessage('Změněno');
            $this->redirect('Admin:user');
        } else {
            $this->flashMessage('Nemuže nastat');
        }
    } else $this->flashMessage('Duplikace emailu');
}
```

Obrázek 21 - Příklad zabezpečeného formuláře (zdroj: vlastní)

Třídy se vytváří v tzv. *compomentu*. Díky tomu lze do *view* snadně přidávat již navržené vzory od frameworku, a tím si ulehčit několikanásobně práci. Pomocí klíčového slova *createComponent + Název* vytvoříme proceduru, kterou v šabloně zavoláme pomoci *{control Nazev}*.

Další předdefinována metoda je *handle,* která umožnuje reagovat na akci uživatele a přenést informaci presenteru do konkrétní vrstvy, se kterou lze nadále pracovat.

13.3 **View (pohled)**

Po konečném sestrojení modelu a presenteru se dostane náležité odměny a zobrazí se naše práce v uživatelském podaní. Šablonovaný systém je spravován v .latte. Latte je šablonovaný systém pro značkovací jazyk obsahující minimální potřebou logiku např. podmínky.

Celý šablonový systém vychází z defaultní šablony @default.latte. Nemusí být pouze jedna, zaleží na struktuře programu. Pro naše účely však jedna bohatě stačí.

Na výchozí šablonu navazují ostatní šablony upravené pro svůj účel. Zobrazují se pomocí tzv. blocku, které se inicializují pomocí *{block contect}*. Celý tenhle blok má ve vlastním pohledu další podbloky jako např. *{control}*. Pomocí téhle hierarchie se sestaví kompletní pohled pro uživatele.

Tenhle systém usnadňuje práci z hlediska dlouhodobé úpravy. Nemusí se díky tomu změnit značná část souboru, ale stačí pozměnit pouze jeden. Tak se přenese celá logika do ostatních vrstev.

14 HOW TO

Celý systém a jeho ovládaní je popsán v manuálu, který má pomoci všem typům uživatelů k jeho používaní a správě. Systém, jak bylo již výše popsáno, byl tvořen od základních pravidel až po ty sofistikovanější, ke kterým jsou potřeba uživatelská práva. Takže uživatelé s větším uživatelským oprávněním mají zděděny vlastnosti a funkce uživatelů předcházejících.

14.1 Student (rodič)

Student už po přihlášení vidí základní stránku s informacemi pro celou školu a pro jeho danou třídu, které ho upozorňují na následující události, a to zvýrazněné důležitostí pomocí barevného spektra, kde čím teplejší barva události, tím důležitější informace.

14.1.1 Změna osobních údajů

Ve frontendu se přesuneme pomocí menu na rubriku profil. Odkaz nás přesměruje na stránku s tabulkou našeho profilu, kde jsou zobrazené osobní údaje. Na konci tabulku je hypertextový odkaz *změnit,* který nás přesune do modu editace našich osobních údajů. Je třeba mít na paměti všeobecně známá pravidla jako jsou např. neduplicitní přihlašovací údaj, validní email apod. Uživatelské údaje jako jméno, email atd. jsou osamostatněny od změny hesla z bezpečnostních důvodů. Provedeme námi potřebné změny a potvrdíme tlačítkem *změnit*. Stránka nás zase přesměrovala na profil, kde vidíme změněné údaje.

14.1.2 Získaní materiálů

Začneme v rubrice *předměty,* kde nám vyjedou v tabulce všechny předměty, ve kterých byla zadána úloha pro naši třídu. Může se jednat pouze o textový výstup bez odkazu na materiál. V takovém případě neuvidíme v řádku materiál odkaz na *download*..

14.1.3 Moje třída

Třída je další infomační stránkou pro studenta, která mu znázorňuje informace o jeho vazbě s danou třídou a události na jeho třídu.

14.1.4 Úkoly

Všechny úkoly, které byly kdy třídě dány jsou k zobrazení v *předměty,* kde je stručný výpis obsahující předmět, o který se jedná a textový materiál k vypracovaní úkolu. Pokud existuje k úkolu i soubor, je možné ho stáhnout za pomocí klíčového slova *download.*

14.1.5 Rodičovství

Aby mohl uživatel s rolí rodič sledovat aktivity svých děti, musí nejdříve poslat žádost danému studentovy, který takovou žádost schválí či nikoliv. Díky tomu se snižuje administrativní náročnost a zamezuje se neoprávněným rodičovstvím.

14.2 Učitel

Učitel má díky svojí roli mnoho funkcí na administraci systémů. Samozřejmě má všechny funkce studenta, jako je editace profilu, oznámení apod.

14.2.1 Předměty

Učitel má právo editovat a přidávat úkoly předmětům, nikoliv však vytvořit předmět samotný. Takové právo má pouze admin. Má však právo svoje úkoly mazat. Všechny předměty, které kdy učitel přidal, se mu zobrazují hned pod seznamem všech předmětů v systému. Při editaci je stejný systém jako u každé editace. Přepíšou se hodnoty, které nám nesedí a klikneme na změny.

14.2.2 Třída

Každý učitel začíná s čistým štítem bez třídy, bez předmětů apod. Třídu si však může hned vybrat ve své profilové editaci, pokud již není zabraná. Poté lze sledovat strukturu studentů v rubrice *třída*. Také zde najdeme tlačítko *Nový klíč,* které změní přihlašovací klíč té třídy, v které právě je učitel správcem (třídním). Tahle hodnota se změní náhodně na desetimístné číslo. Tohle číslo bude vždy zobrazeno v rubrice *třída.*

14.3 Admin

Zde je role admina ze všech nejdůležitější. Admin má všechny předešlé funkce ostatních rolí. Na starost má hlavně aktivaci nového školního roku. Kde pomocí dotazu funkce v SQL

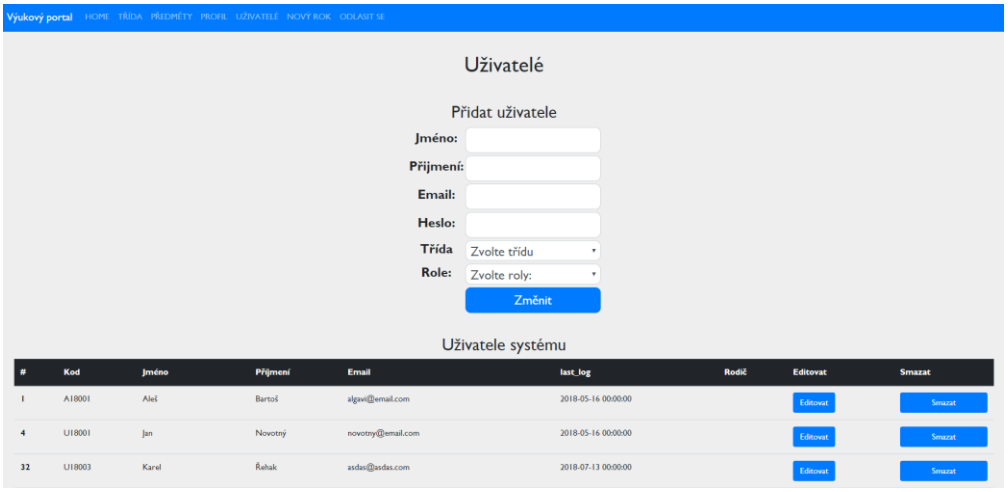
spustí nový školní rok. Třídy se posunou do nových ročníků a resetují se jim učitelé a povinnosti z minulých roků. Případné anomálie ošetří učitel nebo admin. Jediné, co se zachová, je závěrečný ročník, který se jen v databázi změní na neaktivní třídu.

14.3.1 Oznámení

Hned po přihlášení má admin vypsané všechny veřejné a cílené oznámení na hlavní stránce, které může dle své libosti upravovat. Také je zde funkce na přidaní nové události a taky její zaměření.

14.3.2 Editace uživatelů

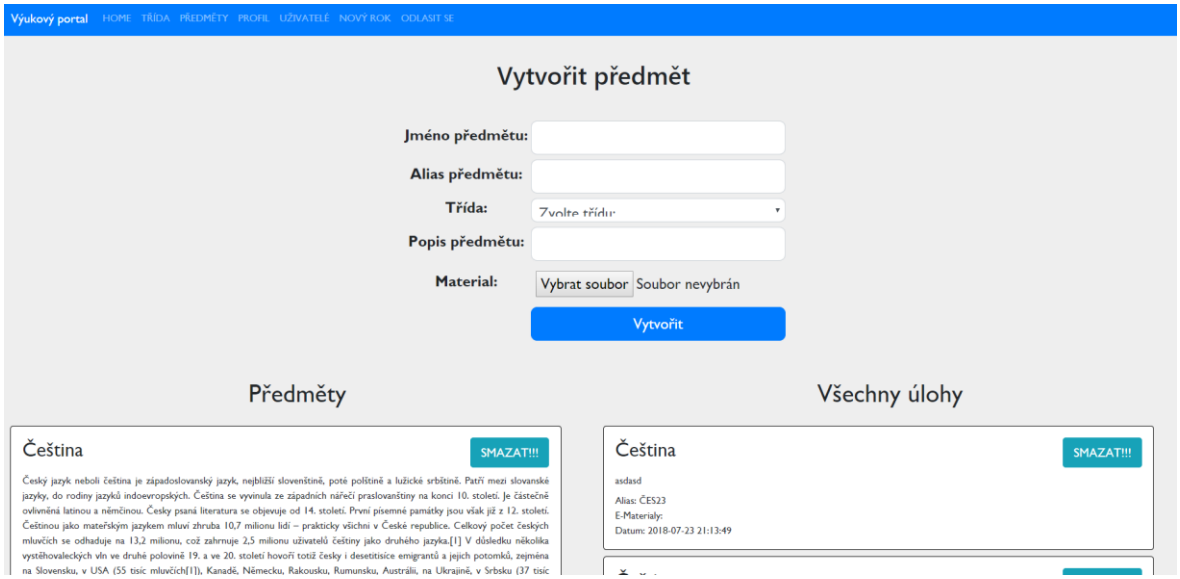
Admin muže libovolně mazat, přidávat a editovat (insert, delete, update) uživatele. V rubrice *uživatelé* má tabulkový výpis všech uživatelů. Pomocí grafického znázornění lze lehce manipulovat s vypsanými daty. Díky téhle tabulce může mazat třeba uživatele, kteří opustili školství nebo ty, kteří nebyli dlouho přihlášeni.



Obrázek 22 - Editace uživatelů (zdroj: vlastní)

14.3.3 Předměty

V rubrice předměty lze přidávat nové předměty pro systém. Po vložení se okamžitě objeví v seznamu předměty, kde je lze zpětné upravovat nebo vymazat. Po pravé straně od předmětu máme seznam přidaných úloh, které lze smazat pouhým kliknutím.



Obrázek 23 - Správa předmětu (zdroj: vlastní)

14.3.4 Nový rok

Je velice důležitá rubrika, která v sobě má jen textový vstup (konstantní heslo pro potvrzení operace) a tlačítko *začít nový rok.* Po zadání validního hesla admin potvrzuje, že opravdu chce provést akci nový rok. Táto akce je nevratná a bez zálohy nebo dobře nastavených procedur se nelze vrátit do původního stavu bez poškození dat.

ZÁVĚR

Cílem bakalářské práce bylo vytvořit informační a výukový systém pro základní školu.

Teoretická část je věnována tématům pro naši problematiku. Nejprve jsou popsány webové stránky až po webovou aplikaci s databázemi. Dále velmi krátce popsán princip CMS systémů. Poté vysvětlený pojem MVC architektura, na kterém funguje bakalářská práce. Je zde popsaná logika adresářové struktury a celkový cyklus života stránky. Po seznámením s termínem MVC je popsán NETTE framework, na kterém webová aplikace běží. Je tu také věnována kapitola bezpečnosti systému na internetu. V poslední řadě je zde zmíněna směrnice GDPR.

Praktická část na začátku zdůrazní věci z teoretické části, které jsou použité na bakalářskou práci. Dále pokračuje na konfigurační nastavení celého programu, na který navazuje databázová struktura. Zde máme graficky znázorněnou celou databázi i s relacemi. Poslední podkapitolka je věnována proceduře. Dále kapitola o bezpečnosti uplatněné v praxi. Zdůrazňuje se tu také autentizace a autorizace v systému a jeho základní princip. V kapitole MVC je popsána nejzákladnější logika programu a vysvětlení, jak vlastně funguje. A nechybí ani popis ovládaní webové aplikace, který může být použít jako manuál.

Po analýze požadavků pro komunikaci mezi učiteli, žáky a rodiči se navrhla základní struktura aplikace a začaly se vyvíjet stavební prvky práce. První byla vytvořena adresářová struktura. Po dokončení adresářové struktury se postupně vyvíjely funkce, které se spojovaly dohromady s programem. Celý proces byl zakončen testováním a laděním.

Aplikace obsahuje všechny požadované vlastnosti pro správu uživatelů (admin, učitel, student, rodič), předmětů, tříd, přístupových práv, generovaní klíčů pro přihlášení, vypisování událostí a jejich editaci a automatické nastavení v DB pro nový školní rok. Celý systém je zabezpečen pro všechny hlavní hrozby podle OWASP. Celý systém je podporovaný modulem, který pracuje real-time, což je vhodnější pro přehled údajů a událostí.

Bakalářská práce řeší stávající problém základní školy tohoto zaměření, a to sdílení informací a výuky studentům a rodičům. Díky webové aplikaci lze zproduktivnit výuku a eliminovat chyby vznikající pouhým sdílením souboru na FTP server.

Webová aplikace byla podrobená průběžnému testování a debuggování pomocí Tracy debugger od frameworku Nette. Díky tomu se všechny chyby zobrazovaly v real-time a mohly se následně opravovat. Celý systém je vyvíjen modulárně, takže jej lze rozšířit o spoustu

dalších přídavných modulů a funkcí, aniž by byl stávající program narušen. Například přidaní zájmových kroužků, docházky studentů, klasifikace atd., až po objednávky na obědy ve škole.

SEZNAM POUŽITÉ LITERATURY

[1] GILMORE, W. J. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Nové, 3. vyd. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2011. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.

[2] GUTMANS, Andi, Stig Sæther BAKKEN a Derick RETHANS. Mistrovství v PHP 5. Brno: CP Books, 2005. ISBN 80-251-0799-X.

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

[4] KOLEKTIV AUTORŮ. Mistrovství v PHP 5. Vyd. 2. Brno: Computer Press, 2007. ISBN 978-80-251-1519-0.

[5] KOLEKTIV AUTORŮ. Vytváříme webové aplikace v PHP5, MySQL a Apache. Brno: Computer Press, 2006. ISBN 80-251-1073-7.

[6] PHP: Hypertext Preprocessor [online]. ©2001-2017 [cit. 2017-11-13]. Dostupné z: http://php.net/

[7] OWASP Foundation. [Online]. [Cit. 2017-11-13] Dostupné z https://www.owasp.org/.

[8] MVC architektura. [online]. Copyright © 2018 itnetwork.cz. Veškerý obsah webu [cit. 17.05.2018]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

[9] MARCOTTE, Ethan a [FOREWORD BY JEREMY KEITH]. *Responsive web design*. New York: A Book Apart, 2011. ISBN 9780984442577.

[10] GDPR | Obecné nařízení o ochraně osobních údajů — prakticky. GDPR | Obecné nařízení o ochraně osobních údajů — prakticky[online]. Dostupné z: https://www.gdpr.cz/

[11] Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework. Quick 'n' Comfortable Web Development in PHP | Nette Framework [online]. Copyright © 2008, 2018 Nette Foundation. All rights reserved. [cit. 17.05.2018]. Dostupné z: https://nette.org/cs/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ACL	Seznam oprávnění pro připojení k objektu.
Aplikační server	Software pomáhající webovému serveru zpracovávat webové stránky obsahující skripty na straně serveru.
Backend	Je část webové aplikace, která slouží jako administrační část webu.
Default	Výchozí hodnota.
Cache	Mezipamět, uchovává data pro rychlejší přistup k dalším datům.
CMS	Redakční systém / publikační systém.
DB	Databáze, soubor dat.
BFA	Útok hrubou silou.
Broadcast	Je zpráva kterou přijmou všechna připojená síťová rozraní.
Controller	Řadič, který reaguje na události a zajišťuje změny v modelu nebo v pohledu.
E-shop	Elektronický obchod.
Framework	Softwarová struktura sloužící při programovaní a vývoji dalších softwarových prací.
Frontend	Část webu viditelná běžným uživatelem.
FTP	(File Transfer Protocol) Protokol pro přenos souborů mezi počítači pomocí počítačové sítě.
Hash	Matematický funkce pro převod vstupních dat do malého čísla
Model	Model je doménově specifická reprezentace informací.
Open source	Je software s otevřeným kódem.
ORM	Objektově relační mapovaní.
Rendrovaní	Vykreslovaní šablony.
Time stamp	Česky: časová značka je sekvence znaku vyjadřující čas vzniku události.
View	Pohled, který převádí data reprezentována modelem do podoby vhodné interaktivní prezentaci uživateli.

Webhosting	Pronájem prostoru pro webové stránky na cizím serveru.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK