Tutoriál pro využití PHP frameworku Nette

Bc. Jan Kotlařík

Diplomová práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jan Kotlařík**
Osobní číslo:	**A16544**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Tutoriál pro využití PHP frameworku Nette**
Téma anglicky:	**A PHP Framework Nette Tutorial**

Zásady pro vypracování:

1. **Vypracuje literární rešerši na téma PHP frameworky.**
2. **Nastudujte současný tutoriál na stánkách Nette.**
3. **Vytvořte vlastní rozšířený tutoriál, který pro začátečníky navíc vysvětlí:**
   - **MVC architekturu, použití modelu, metody přístupu do DB.**
   - **Zabezpečení. Autentizaci uživatelů a autorizaci požadavků.**
   - **Použití sessions a cookies.**
   - **Vysvětlete Implementaci jQuery tabulek, stránkování a použití AJAX.**
4. **Tutoriál vytvořte jako tvorbu vlastního projektu krok za krokem.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **Nette Framework [online]. Praha: Nette Foundation, 2017 [cit. 2017-11-20]. Dostupné z: https://nette.org/**
2. **PHP: Hypertext Preprocessor [online]. The PHP Group, 2017 [cit. 2017-11-20]. Dostupné z: https://php.net/**
3. **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.**
4. **9 of the Best Free PHP Books. LinuxLinks.com [online]. USA: LinuxLinks.com [cit. 2017-11-20]. Dostupné z: http://www.linuxlinks.com/article/20130119004851789/9oftheBestFreePHPBooks-Part1.html**
5. **GUTMANS, Andi, Stig S. BAKKEN a Derick RETHANS. Mistrovství v PHP 5. Brno: CP Books, 2005. ISBN 80-251-0799-X.**
6. **CHAFFER, Jonathan a Karl SWEDBERG. Mistrovství v jQuery: [kompletní průvodce vývojáře]. Brno: Computer Press, 2013. Mistrovství. ISBN 978-80-251-4103-8.**

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

Datum zadání diplomové práce: **3. prosince 2018**

Termín odevzdání diplomové práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové 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 diplomová 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 diplomovou 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 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 diplomové 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 diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové 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 diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 3. 5. 2019 …Bc. Jan Kotlařík, v.r. …

podpis diplomanta

ABSTRAKT

Tato práce se zabývá tvorbou tutoriálu pro použití frameworku Nette pro začátečníky. V úvodní části je popis co to vlastně framework je, jaké má výhody a nevýhody a použití. V jednotlivých částech je vysvětleno, jak jednotlivé části používat a navazovat na sebe. Tutoriál je doplněný o ukázkovou aplikaci, ve které je popsáno použití jednotlivých částí frameworku podle zadání.

Klíčová slova:

Framework, Nette, presenter, šablona, model, metoda, databáze, pohled

ABSTRACT

This work deals with the creation of a tutorial for using the Nette framework for beginners. In the introductory part is a description of what the framework is, what it has advantages and disadvantages and uses. Each section explains how to use and build on each part. The tutorial is supplemented with a sample application that describes usage.

Keywords:

Framework, Nette, presenter, template, model, method, database, view

Poděkování

Děkuji vedoucímu práce doc. Ing. Martinu Syslovi, Ph.D. za účinnou metodickou a odbornou pomoc a další cenné rady při zpracování této diplomové práce.

OBSAH

ÚVOD

Internet a technologie s ním spojené jsou nedílnou součástí většiny aktivit člověka. Tyto technologie používáme nejen i získávání informací a obyčejné komunikaci, ale i k činnostem interaktivním, zajišťujících fungování každodenních pracovních i soukromých aktivit, tedy činností jako jsou nákupy, sledování různých aspektů jako je počasí na konkrétním místě, vzdělávací a volnočasové aktivity až po řízení lidských a výrobních zdrojů. Se stále se stupňujícími požadavky na komunikaci a řízení se zvyšují nároky na různorodost a variabilitu systémů takové požadavky zajistit.

Pro vývoj webových aplikací lze využít nejrůznější nástroje od jazyka HTML pro tvorbu jednoduchých webových stránek, přes skriptovací jazyky vytvářející dynamický web až po jejich frameworky umožňující realizaci složitých webových aplikací s jednotlivými částmi pro zpracování dat, sestavení vizuální stránky aplikace a jejich spojení pomocí aplikační logiky.

Cílem této práce je vytvořit tutoriál, který poskytne začátečníkům návod na vytvoření webové aplikace pomocí frameworku Nette 2.4 a vysvětlí jak vytvořit vhodnou strukturu projektu, způsob jak aplikovat model MVC, oddělit práci s daty od logiky projektu a jeho vzhledu. Dalším cílem této práce je využití tutoriálu při výuce studentů střední školy v oblasti tvorby webových aplikací.

I. TEORETICKÁ ČÁST

1 LITERÁRNÍ REŠERŠE

Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovnu API, návrhové vzory nebo doporučené postupy při vývoji. To znamená, že se mohou vývojáři soustředit na jádro problému projektu a nezabývat se běžnými problémy. O ty se postará právě framework. Jedná se například o rutiny jako připojování k databázi, změny databází, zpracování šablon, validace stránek a formulářů, bezpečnosti nebo vytváření SEO URL adres, které nám pomáhají dosahovat lepšího umístění ve vyhledávačích. Funkčnost a úroveň zmíněných rozšíření se u každého frameworku liší a každý z nich se zaměřuje na něco jiného[1]. Každý framework s sebou přináší nové názvy funkcí, tříd a metod. Nejsou to proměnné, které patří do daného programovacího jazyka (například PHP), ale názvy pro framework specifické. Při správném zvolení názvu to znamená volání konkrétní akce. Vývojář se musí seznámit s frameworkem a pochopit jeho princip fungování.[2]

Hlavním cílem frameworku je usnadnit práci webovým programátorům tím, že je zbaví zabýváním se rutinních prací a pomocí knihovny API poskytne prostor pro jednoduchá a efektivní řešení daného problému.

1.1 Výhody a nevýhody frameworků

Výhody použití frameworků:[6]

- Umožňují rychlý vývoj
- Poskytují dobře organizovaný, opětovně využitelný a udržovatelný kód
- Umožňují růst v průběhu času, protože webové aplikace běžící na frameworcích jsou škálovatelné
- Zbavíte se starostí souvisejících s nízko úrovňovou bezpečností webu
- Dodržují vzor MVC (Model-View-Controller), který zajišťuje separaci prezentace a logiky
- Prosazují moderní webové vývojové postupy, mezi něž patří nástroje objektově orientovaného programování

Nevýhody:

- pevně daný styl vývoje
- může být pomalý či neefektivní kód
- čas pro nastudování frameworku
- při odinstalování frameworku nebude funkční aplikace

1.2 Nette framework

Nette Framework je open source framework pro tvorbu webových aplikací v PHP 5 a PHP 7. Zaměřuje se na eliminaci bezpečnostních rizik, podporuje AJAX, DRY, KISS, MVC a znovu použitelnost kódu. Využívá událostmi řízené programování a z velké části je založen na použití komponent. [3]

Galerie aplikací a webových stránek postavených společností Nette:

- studentreality.cz
- zlateslevy.cz
- rohlik.cz
- signaly.cz
- onlinefilmy.net
- megapixel.cz
- lekarna.cz
- xfactor.joj.sk
- claimcloud.cz
- adminer.org
- a další

Je český projekt, jeho uvedení je datováno na rok 2008, jeho autor je David Grudl. V současné době se o vývoj stará Nette Foundation za stálého dohledu autora. Aktuálně stabilní verze Nette frameworku je 2.4 a vyžaduje PHP verzi alespoň 5.6 nebo vyšší a je kompatibilní s PHP 7.1. Licence frameworku je open source, tzn., že vývojáři mohou volně používat zdrojový kód a vytvářet nové návrhy a zlepšení.

1.3 Zend framework

Framework je produktem firmy Zend Technologies Inc., světlo světa spatřil začátkem roku 2006, kdy byla zveřejněna první verze Zend frameworku. Aktuální verze Zend framework je 3 a z předchozích verzí si bere to nejlepší a celkový počet stáhnutí již překročilo 116 miliónů.[4] Vyžaduje alespoň PHP 5.6 nebo vyšší. Je kladen velký důraz na problematiku zpětné kompatibility, takže je přechod na novější verzi zcela bezproblémový. Nyní je hlavním přispěvatelem frameworku společnost Rogue Wave, která se stará o to, aby byl Zend framework nadále zlepšován a rozšiřován. Jsou zde ale i společnosti jako Google, Microsoft nebo StrikeIron, které jsou partnery projektu a pomáhají s implementací nových služeb a funkcí. Zend je stejně

jako Nette framework, distribuován pod open source licencí, která opět umožňuje vývojáři využívat program prakticky bez omezení.

Zend framework je objektově orientovaný webový framework pracující v jazyce PHP, zkráceně označovaný jako ZF. ZF je vyvíjen s ohledem na jednoduchý vývoj webových aplikací. Užívá modulární architekturu, která umožňuje vývojářům používat jen ty komponenty, které potřebují. Zend nabízí dvě možnosti inicializace, bud' pomocí souboru INI nebo XML. Zahrnuje v sobě komponenty pro MVC aplikace.[4]

Nabízí spolupráci s databázovými systémy MySQL, MSSQL, Oracle, PostgreSQL a IBM DB2. Projekty Zend frameworku jsou tvořeny kombinací jazyků PHP a HTML, výsledný soubor má koncovku phtml. Jsou k dispozici validační metody pro kontrolu správného formátu datumu, mailu, IP adres, čísel a dalších. Obsahuje helpery pro usnadnění stále se opakujících kódů. ZF má na svých stránkách zpracovanou a přehlednou dokumentaci svých knihoven, stránku s tutoriály a fórum s řešenými problémy.

1.4 Symfony framework

Dalším velice známým webovým frameworkem je Symfony, který byl poprvé zveřejněn roku 2005. Myšlenka vzniku frameworku vznikla ve společnosti SensioLabs. Framework je de facto open source projekt vyvíjený nejen lidmi ze SensioLabs, ale tisícem dalších vývojářů. Aktuální nejvyšší stabilní verze nese označení 4.2.[5] Vývojové centrum ze SensioLabs si zakládá na pravidelném vydávání verzí frameworku, ať už nových nebo revizních. Bývá pravidlem, že pokud se v používané verzi najde chyba, bude v příští verzi opravena. Bezpečností strategie frameworku Symfony nabádá vývojáře, kteří narazí na chybu v programu, aby jí nezveřejňovali, ale poslali na příslušný email, kde se o ni postará tým, který má na starosti bezpečnost a provede odstranění chyby v další verzi. Na rozdíl od přecházejících frameworků je Symfony distribuován pod licencí MIT. Tato licence může být využívaná i u projektů, u kterých je uzavřený kód. U open source projektů je plná kompatibilita s GNU GPL licencí, která dovoluje kombinaci s MIT. Pro vývojáře to znamená jen to, že text licence MIT musí být uveřejněn spolu se zveřejněným dílem.

Symfony je aplikační framework psaný v jazyce PHP, minimální verze PHP pro Symfony je PHP 7.1.3. Aktuální verze je Symfony 4.2 Framework je založen na architektuře MVC. Obsahuje řadu nástrojů a tříd zaměřených na zkrácení doby vývoje webových aplikací.[5]

Symfony nabízí knihovny pro podporu:

- AJAX
- správu cache
- informace o toku dat
- šablony a helpery
- možnost pluginů

Framework stačí na server nahrát pouze jednou do adresáře, který není přístupný přes HTTP. Mnoho funkcí lze také volat přes příkazový řádek. Nastavení aplikace využívá kaskádový přístup, nejvyšší nastavení může být překryto podle nastavení projektu.
Symfony lze nainstalovat na všechny hlavní operační systémy a je kompatibilní s většinou databází např. MySQL, PostgreSQL nebo Microsoft SQL.

1.5 Ruby On Rails

Ruby On Rails je framework pro vývoj webových aplikacích v jazyce Ruby, zkráceně označován jako Rails. Poslední verze Rails 5.2.3 byla vydána 28. března 2019. Jeho základem je MVC.[6] Umožňuje psát méně kódu a zároveň dosáhnout více než v mnoha jiných jazycích. Rails nabízí řadu knihoven:

- Action Mailer - pro práci s e-maily, umožňuje odesílat e-maily i s přílohami. Také umí emaily přijímat a zpracovávat.
- Action Dispatch - zpracovává příchozí požadavky a posílá je, kam uživatel potřebuje.
- Action View - má na starosti zobrazování ve formátu HTML a XML.

1.6 CakePHP

CakePHP je zdarma šiřitelný open-source pro PHP. Aktuální verze je CakePHP 3.7. Je postaven na základě Ruby on Rails. Je kompatibilní s PHP ve verzi 5.6.0 a vyšší včetně 7.3. Nabízí automatické generování kódu, rychlé a flexibilní šablony. CakePHP je navržen pro maximální efektivitu práce.[7] Ve skutečnosti stačí napsat určitý kód pouze jednou, a pak ho používat podle potřeby. Dále pak šablony pro AJAX, JavaScript, HTML formuláře a další. Funkce pro práci s Emaily, Cookies, zabezpečením a sekcemi. Podporuje databázové

systémy DB2, Firebird, MSSQL 2008, MySQL 5.5.3, ODBC, Oracle, PostgreSQL, SQLite 3, Maria 5.5. Funguje na jakémkoliv hostingu s minimální nebo žádnou konfigurací.

1.7 CodeIgniter

CodeIgniter je framework založený na principu MVC pro vývoj webových aplikací v PHP. Aktuální verze CodeIgniter je 3. 1. 10. Řadí se mezi menší frameworky, samotná velikost je v řádu několika MB. Je jednoduchý na instalaci a používání. Snaží se co nejjednodušeji řešit problémy s co nejmenšími nároky na programátora a systém.[8] CodeIgniter běží na základě několika malých knihoven, další knihovny jsou nahrávány dle potřeby. CodeIgniter je vybaven funkcemi pro nahrávání souborů, pro práci s maily, dále obsahuje knihovnu pro zpracování obrázku, jako je například zmenšování, vkládání vodoznaku a ořez. Nabízí řadu helperů, které slouží k zpřehlednění a nahrazení stále se opakujících částí kódu. U CodeIgniter je struktura aplikace rozdělena do dvou částí:

- Systém - v této složce jsou uloženy všechny knihovny a helpery.
- Application - je část pro samotnou aplikaci a všechny součásti tvořené uživatelem. Zde je možno vkládat vlastní nebo upravený kód ze systémových knihoven nebo helperů s příponou „my“. CodeIgniter sám zjistí, zda chcete používat upravenou systémovou knihovnu a přednostně ji použije.

CodeIgniter má zpracovanou podrobnou dokumentaci, na oficiálních stránkách nabízí základní informace, návody, tutoriály, popisy helperů a knihoven s ukázkami kódu.

1.8 Prado

Prado je framework založený na komponentách a událostmi řízeného programování pro vývoj webových aplikací v PHP 5. PRADO znamená P php, R rychlá, A aplikace, D development (vývoj), O objektově orientovaný. Je zde nutné PHP 5.4.0 nebo vyšší. Licence je open source, lze jej využít k vývoji i komerčních projektů. V současné době je k dispozici verze PRADO 4.0.2.[9] Vývoj provádí parta nadšenců jmenovitě uvedená na webu frameworku. Budoucí vývoj bude zaměřen na podporu PHP 7.1.

1.9 Jelix

Aktuální verze odladěného frameworku je 1. 6. 23. Je určen pro PHP 5.2 a vyšší. Jedná se o framework, který podporuje více výstupních formátů, kromě XHTML také XUL, RSS, ATOM, XML, PDF a další.[10] Do výbavy Jelixu patří také jAcl. Jelix neobsahuje Helpery. Formuláře se ukládají jako XML soubory.

Součástí Jelixu je od roku 2006:

- Architektura v opakovaně použitelných modulech
- MVC logika a strukturovaná organizace souborů
- Zásuvné moduly pro prodloužení komponentů
- Mnoho jednoduchých API
- Výkonné jádro pro náročné webové stránky

1.10 Laravel

Je aktuálně určen pro PHP 7.1.3 a vyšší. Podporuje databázi MySQL. Krom standardních pluginů obsahuje také plugin Photo/Image Management, který umožňuje různé operace s obrázky.
Požadavky frameworku na systém je splněn použitím virtuálního stroje Laravel Homestead. Ten běží na jakémkoliv systému Windows, MAC nebo Linux a obsahuje webový server Nginx PHP 7.3, PHP 7.2, PHP 7.1, MySQL, Redis, Memcached, Node a další.[11]
Pokud nelze Homestead použít, musí se zajistit, aby server splňoval následující požadavky

- PHP 7.1 nebo vyšší
- OpenSSL PHP
- Rozšíření PHP PDO
- Mbstring PHP
- XML PHP
- PHP Ctype
- PHP JSON
- PHP pro BCMath

Na stránkách frameworku je dostatečná podpora i s ukázkami kódů a veškerou instalací a nastavením.

II. PRAKTICKÁ ČÁST

2 POUŽITÉ VÝVOJOVÉ NÁSTROJE

Pro realizaci aplikace s využitím frameworku Nette je potřeba nainstalovat určité nástroje. Doporučuje se vývoj provádět na lokálním počítači, tam jej také odladit a teprve hotový ho nahrát na server, kde bude provozován. V tomto projektu byl framework použitý jako nástroj pro vytvoření tutoriálu bez ostrého nasazení.

2.1 WAMP, Apache, MySQL, PHP

Nástroje použité pro vývoj jsou WAMP 3.1.7, písmeno W představuje operační systém, na kterém nástroj poběží, tedy Windows, písmeno A zastupuje HTTP server Apache, MySQL je skriptovací dotazovací jazyk pro práci s databází a P zastupuje skriptovací jazyk PHP. Součástí wampu je Apache 2. 4. 37, PHP 7.3.1, MySQL 5. 7. 24, Adminer 4.7.0. V podstatě se jedná o webový simulátor, kterým lze zprovoznit webový server na lokálním počítači bez nutnosti udržovat spojení se skutečným serverem.

Apache je samotný HTTP server, který zajišťuje komunikaci s klientem a jednotlivými subčástmi.

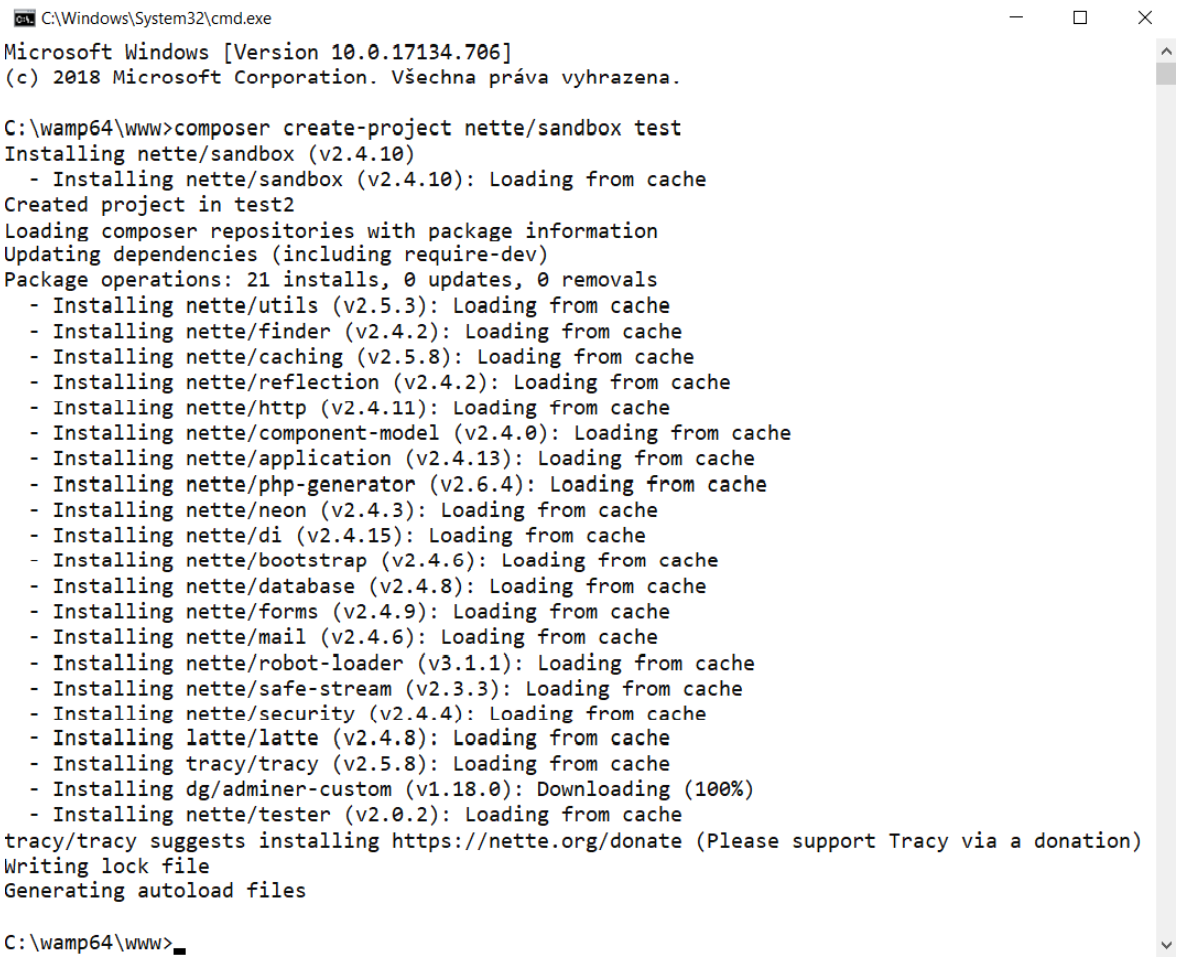
MySQL je multiplatformní databázový systém a dotazovací jazyk, je s ním spojený uživatelsky přívětivý phpMyAdmin správce databází, který je součástí instalačního balíčku a pomocí kterého lze pohodlně vytvářet a spravovat databázové systémy. Je nutností každého dynamického webu s proměnným obsahem.

PHP (Personal Home Page) je skriptovací programovací jazyk určený především pro programování dynamických webových stránek a aplikací.

Dále je použitý webový prohlížeč Chrome, aktuální verze 73.0.3683.103 a textový editor PSPad 5.0.1. U PSPadu je nutné nastavit kódování na utf-8, jinak prohlížeč vrací chybu *„Neplatný řetězec utf-8“* a odmítá zobrazovat české znaky.

2.2 Test konfigurace

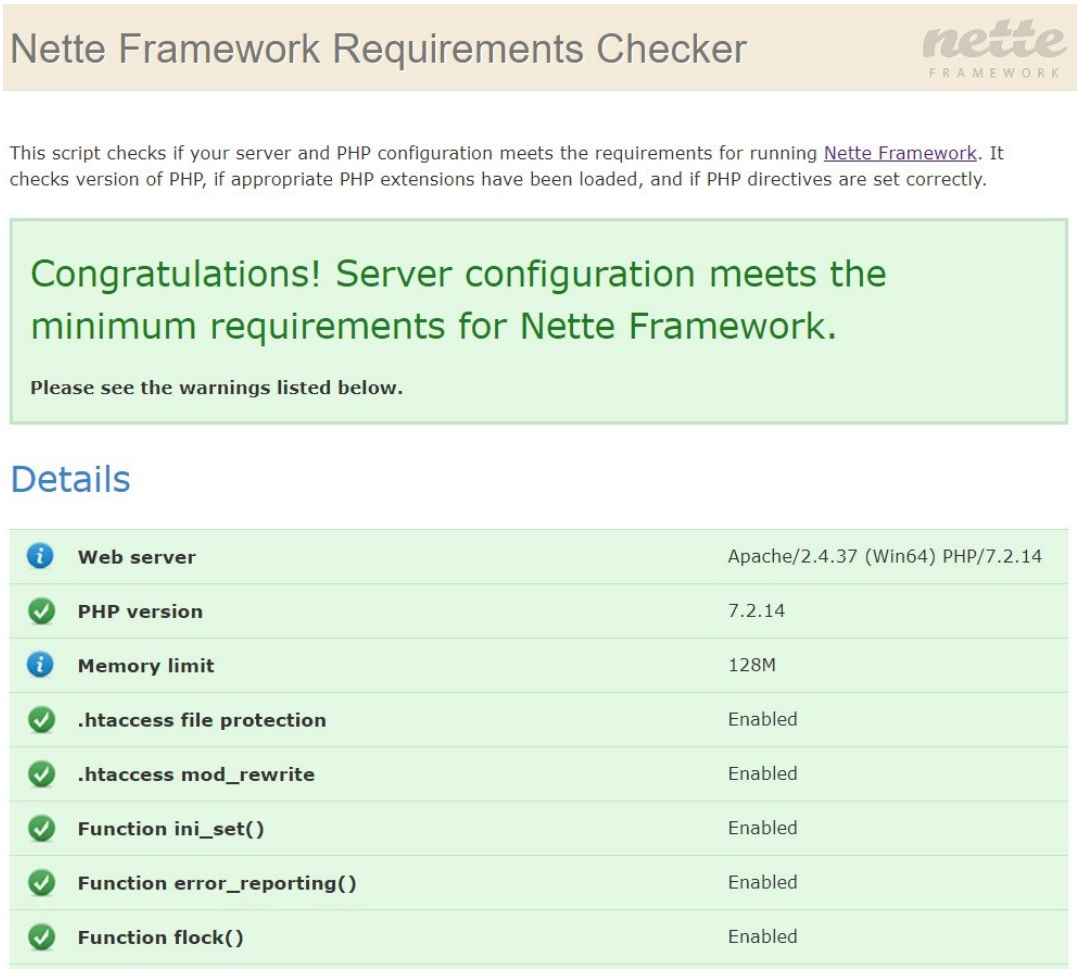
Pro otestování správné konfigurace wampu a správných verzí, je vhodné založit sandbox projekt a to pomocí composeru `composer create-project nette/sandbox test` a v prohlížeči zavolat `http://localhost/test/www/checker/`.



Obr. 1 – Instalace testovacího projektu – zdroj: vlastní

Volaný skript zkontroluje verzi PHP a konfiguraci serveru, správnost nastavení php příkazů pro spuštění Nette frameworku a výsledek zobrazí do okna prohlížeče.

Nette framework 2.4 vyžaduje PHP verze 5.6.0 nebo vyšší a klade jisté požadavky na prostředí serveru a ty je potřeba otestovat, aby aplikace mohla vůbec fungovat. Nástroj, který testuje server, se nazývá *Requirements Checker*. Ten otestuje běhové prostředí serveru a informuje o tom do jaké míry je možné jej pro framework používat. Na přehledném výpise lze snadno zjistit, jestli si bude Nette se serverem rozumět. Zelený nápis *„Congratulations! Server configuration meets the minimum requirements for Nette Framework“* znamená, že je vše v pořádku.



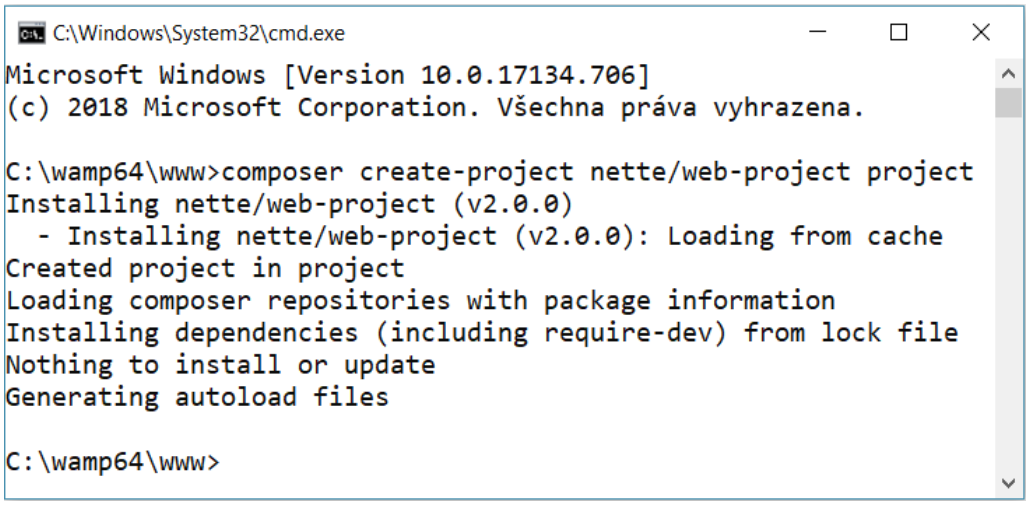
Obr. 2 - Kontrola konfigurace systému pomocí checker - zdroj: vlastní

3 INSTALACE PROJEKTU NETTE

Projekt frameworku Nette lze nainstalovat několikerým způsobem. Lze využít aplikaci composer nebo rozbalit kopii dříve vytvořeného projektu nebo využít některý verzovací nástroj.

3.1 Instalace Nette projektu

Výchozí složka localhostu je `c:/wamp64/www/`. Zde lze zakládat projekty webových aplikací, které jsou vyvíjeny a testovány lokálně. Nette projekt lze vytvořit buď pomocí composeru direktivou `composer create-project nette/nette nazev`, nebo je zde možné rozbalit již hotový připravený projekt např. z githubu. Obrázek 3 ukazuje výpis instalace webprojektu pomocí composeru.

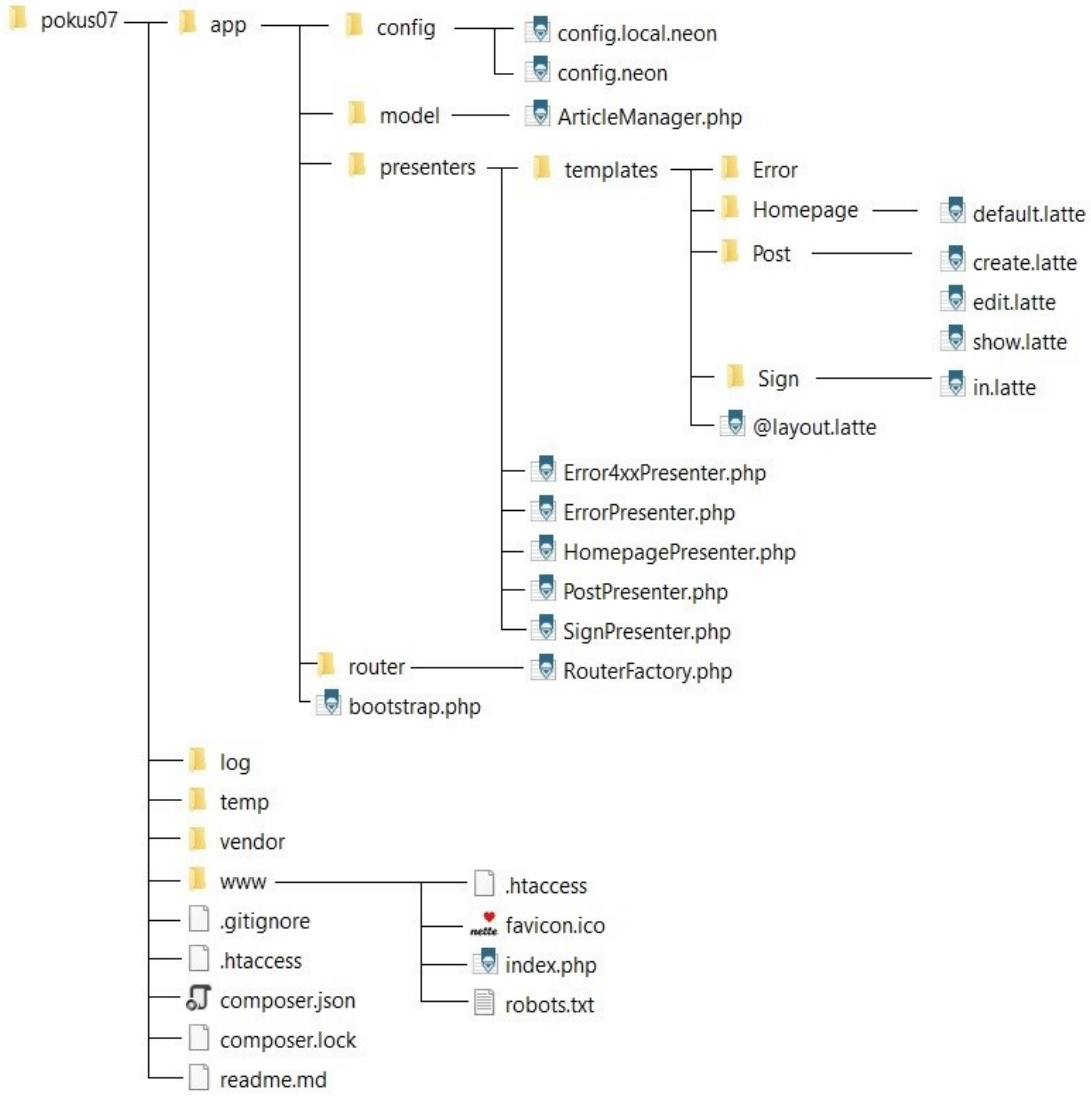


Obr. 3 – Instalace projektu pomocí composeru - zdroj: vlastní

Hlavní části projektu jsou složky config, model, presenters a templates. V konfiguračním souboru config.neon se nastavuje přístup do databáze a přihlášení k přidávání nových příspěvků, v modelu je vytvořena logika aplikace, ve složce presenters jsou jednotlivé php skripty s třídami a jejich metodami a v templates jsou vytvořené pohledové šablony celé aplikace. Tento koncept, kdy je od sebe oddělena logika, data a pohledy se nazývá architektura MVC (model-view-controller).

3.2 Struktura projektu

Struktura založeného projektu je na obrázku č. 4.

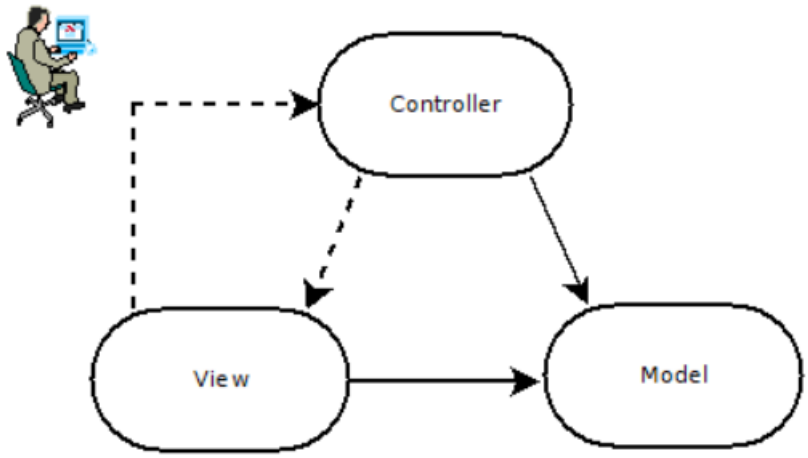


Obr. 4 - struktura projektu - zdroj: vlastní

Z pohledu vývojáře je nejdůležitější složka config, ve které se provádí veškerá konfigurace aplikace, složka presenters, v níž se nachází vytvořené php skripty a složka templates se šablonami tvořící vzhled webu.

3.3 MVC – Model, View, Controller

MVC aneb model-view-controller je návrhový vzor, objektová struktura, jež odděluje data a jejich zpracování od jejich zobrazení. Model představuje datové úložiště, obstarává získání dat a práci s nimi a vrací je controlleru, který si o ně zažádal, controller je následně předá view, který je nějakým způsobem zobrazí.[6] Pomocí MVC architektury získáme přehledný kód, lehce upravitelný a navíc okamžitě přístupnější pro týmovou spolupráci.
Návaznost jednotlivých komponent ilustruje následující obrázek 5:



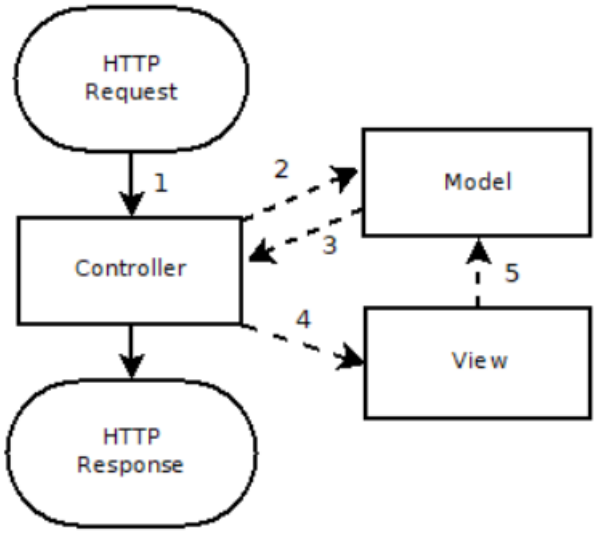
Obr. 5 – Součásti MVC - zdroj: vlastní

Model view Controller je architektura využívaná u většiny vývojových prostředí pro tvorbu webových aplikací. MVC rozděluje celou aplikaci do tří základních souborových typů, na Model sloužící ke zpracování dat, View prezentace dat a Controller
Na obrázku 6 je zobrazen HTTP požadavek (HTTP Request) na server. Zpracování požadavku pomocí MVC a zpětná HTTP odpověď (HTTP Response).[14]

1. Vše začíná zasláním HTTP požadavku na server. Tento požadavek je přijat řadičem a dále zpracováván.
2. Přijatý požadavek řadič zpracuje, zvolí vhodný model a podle potřeby přepošle žádost o zpracování dat na model.
3. Model zpracuje požadavek ke zpracování dat v databázi. Výsledek své práce vrátí controlleru.
4. Controller upraví data do správného formátu, vybere vhodné zobrazení a odešle je na View.
5. V některých případech se může stát, že si view vyžádá data přímo od modelu.

6. Na závěr controller vytvoří view a odešle view zpět do prohlížeče ve formě HTTP odpovědi.

Většinou se bodu č. 5 nevyužívá. Převládá názor, že view by nemělo mít vůbec žádný přístup do databáze.[3] Aplikace si všechna data z databáze vyžádá od modelu a následně je předá do view to již nemá potřebu do modelu zasahovat.



Obr. 6 – Průběh požadavku MVC - zdroj: vlastní

MVC si lze také představit jako tři vrstvy. Vrstva prezenční, ve které se view stará o zobrazení dat uživateli. Vrstva řídící, kterou představuje controller se stará se o chod cel/e aplikace. Vrstva datová, která přistupuje k datům. Tedy každá vrstva se stará o to své, dohromady tvoří přehledný celek.[14]

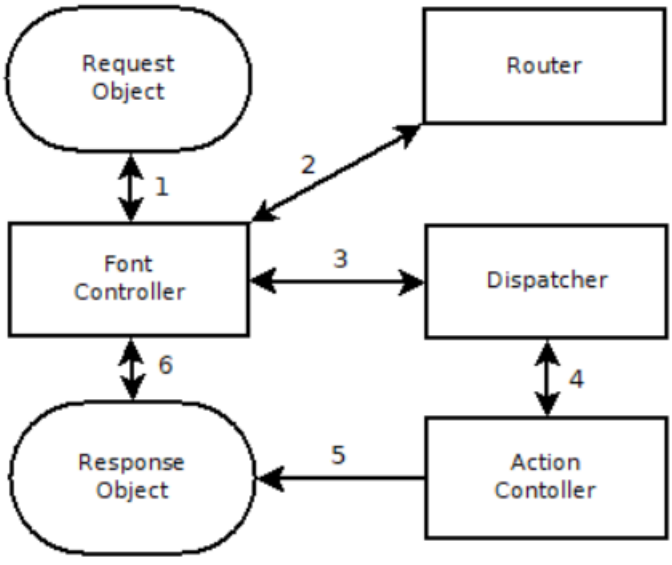
Model se stará o datovou vrstvu v MVC. Umožňuje čtení, zapisování, upravování a mazání dat. O tom, kam se data zapíší, rozhoduje a má vědomí jenom model. Ten umožňuje práci s daty pro různá datová úložiště:

- Databáze
- Textový soubor
- RSS
- Webová služba

Při tvorbě modelu je důležité vědět, jak k datům v konkrétním úložišti přistupovat. U textových dokumentů je potřeba vědět, jak data formátovat, aby byly srozumitelná pro další použití.

View – pohled se stará o prezentaci předaných dat uživateli. Naopak se nestará, odkud data jsou, nebo zda jsou správná. View se stará pouze o vložení přijatých dat do zvolené šablony, o správné naformátování a publikování ve srozumitelném a přehledném zobrazení uživateli. Zobrazení si musí poradit s formátem, v jakém data dostane, zda se jedná o pole, ve kterém budou všechna data, nebo zda budou data předána po prvcích. Zpětně se taky view stará o přijetí dat skrze formuláře. Výpis pohledu se většinou realizuje ve formátu HTML stránky, která je odeslána do prohlížeče uživatele. View obvykle umožňuje vytvořit formáty CSV nebo XML, které si uživatel může stáhnout. Dále může view vytvářet grafiku nebo PDF, které se zobrazí v prohlížeči. Všechny formáty jsou do prohlížeče odeslány jako odpověď HTTP.[14]

Controller představuje řídící vrstvu. Zavolání controlleru se vyvolá HTTP žádost z prohlížeče uživatele. Z přijaté žádosti controller zjistí, který model má být použit a v jakém view mají být data následně vykreslena.[14] Controller se skládá z více částí, které vkládají různé návrhové vzory. Jsou to front controller a action controller.



Obr. 7 – Požadavek kontroleru - zdroj: vlastní

Na obrázku 7 je zobrazen průběh požadavku v controlleru. Hlavní částí controlleru je front controller. Celý proces probíhá v následujících krocích [14]

1. Přijetí požadavku uživatele zařizuje front controller.
2. V dalším kroku front controller předá data routeru (směrovači), který zjistí, jaká akce nebo jaké akce mají být vykonány.
3. Následně je front controller pověřen dispečerem, aby vykonal další zpracování požadavku.
4. V cyklu volá dispečer postupně action controllery, aby byly vykonány všechny akce. V určitých akcích může controller přistupovat k modelu.
5. Výsledek je předán response objektu.
6. V závěru front controller odešle odpověď do prohlížeče uživatele ve formě HTTP odpovědi.

3.4 Soubor index.php

Každá webová aplikace obvykle vychází z defaultního souboru index.php. Je to skript, který se automaticky načítá jako první, když ve webovém prohlížeči zadáme cestu k internetové službě. Také projekt Nette disponuje tímto souborem. Obsah souboru není velký, nachází se v něm pouze dvě položky.

```
$container = require __DIR__ . '/../app/bootstrap.php';

$container->getByType(Nette\Application\Application::class)
    ->run();
```

Tou první položkou je předání řízení do aplikace na soubor bootstrap.php. Zaváděcí soubor, který inicializuje prostředí a vytvoří dependency injector kontejner, který následně z DI získá službu, pomocí níž spustí webovou aplikaci.

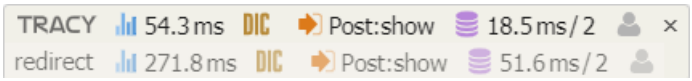
3.5 Soubor bootstrap.php

Nastavení prostředí a vytvoření Dependency Injection kontejneru má v Nette na starosti třída Configurator, který se nachází v zaváděcím souboru `bootstrap.php` umístěném ve složce `app/`.

```
require __DIR__ . '/../vendor/autoload.php';
$configurator = new Nette\Configurator;
```

PHP je jazyk s obtížně odhalitelnými chybami. Proto byl framework opatřen silným ladícím nástrojem Tracy, které je defaultně pro ladění povoleno. Je zaveden následující direktivou, kde případné chyby loguje do zadaného adresáře. Průběžný stav je vidět ve spodní části prohlížeče

```
$configurator->enableTracy(__DIR__ . '/../log');
```



Obr. 8 – Ladící hlášení Tracy - zdroj: vlastní

Nastavuje se zde také defaultní hodnota časové zóny.

```
$configurator->setTimeZone('Europe/Prague');
```

Konfigurace se při běžném provozu načte pouze jednou a uloží se do cache. To výrazně zrychluje celou aplikaci, ukládá se do složky temp.

```
$configurator->setTempDirectory(__DIR__ . '/../temp');
```

Automatické načtení tříd se zajistí pomocí RobotLoaderu. Načítá třídy z adresáře, kde je umístěn bootstrap.php.

```
$configurator->createRobotLoader()
    ->addDirectory(__DIR__)
    ->register();
```

Podle konfiguračních souborů config.neon a config.local.neon se generuje systémový DI kontejner, který je srdcem celé aplikace. V tomto projektu je využitý pouze config.neon přestože vývoj proběhl na localhostu.

```
$configurator->addConfig(__DIR__ .
                                '/config/config.neon');
$configurator->addConfig(__DIR__ .
                            '/config/config.local.neon');
```

Posledním krokem je vytvoření a předání hotového kontejneru.

```
$container = $configurator->createContainer();
return $container;
```

3.6 Šablonovací systém Latte

Latte je PHP šablonovací systém, který překládá šablony na jednoduchý optimalizovaný PHP kód a zabezpečí výstup před zranitelnostmi jako je XSS.[15] V projektu je použito několik samostatných šablon. Hlavní šablona nese název *@layout.late* a nachází se v adresáři *app/presenters/templates*.

Struktura hlavní šablony kopíruje strukturu klasické html stránky s tím rozdílem, že chování šablony je dynamické. Celý syntaxe začíná a končí párovým tagem <html>, dále je obsažena hlavička <head> s titulkem <title> a blokem pro načtení kaskádových stylů, následuje vlastní tělo stránky, do kterého je inkludován samotný obsah webu prostřednictvím dalších šablon a skriptů.

```
<!DOCTYPE html>
<html>
<head>
     <meta charset="utf-8">
     <title> </title>
    {block css}
     {/block}
</head>

<body>
    {include content}
    {block scripts}
     {/block}
```

```
</body>
</html>
```

Další šablony jsou inkludovány do hlavní šablony jako bloky s určitým názvem příkazem `{include „název bloku"}`. Tyto šablony už neobsahují tagy vytvářející stránku, ale jen jméno bloku a další jednotlivé části stránek rozdělené párovým tagem , které mohou obsahovat další identifikaci jako id nebo class. Struktura takové dílčí šablony může vypadat takto.

```
{block content}
<div id="content">
    <div class="pagination">
      …… zde stránkování ……
    </div>
    <div id="nono">
      …… zde přihlášení a odhlášení ……
    </div>
    <div class="zakaznici">
      …… zde práce s databází zákazníků ……
    </div>
</div>
```

3.7 Struktura modelu

Model zprostředkovává přístup k datům z datového úložiště, databáze a stará se o busines logiku aplikace. Obsahuje databázové dotazy, informace o tom jestli je uživatel přihlášený, obsah košíku ap. Jeho úkolem je načíst data, zpracovat a poslat zpět Modelový skript se nachází v *app/model*, soubor nese název *ArticleManager.php*.

Základní třída ArticleManager

```
namespace App\Model;

use Nette;

class ArticleManager

{

    use Nette\SmartObject;

    private $database;

}
```

Konstruktor přístupu k databázi, zajišťující přístup k datům

```
public function __construct(Nette\Database\Context $database)

    {

        $this->database = $database;

    }
```

Metoda předání dat, seřazení podle data vložení vzestupně

```
public function findPublishedArticles()

    {

        return $this->database->table('posts')

            ->where('created_at < ', new \DateTime)

            ->order('created_at ASC');

    }
```

3.8 Presenter

K tomu, aby celá struktura fungovala a vzájemně všechno fungovalo, je zapotřebí kontroleru. Kontroler se stará o funkčnost celého návrhového vzoru MVC a je jakýmsi řadičem. Presenter pracuje podobně jako kontroler s tím rozdílem, že manipuluje s požadavky, které mu pošle pohled.[14] Mění se jeho přístup ke vstupním datům, ale jeho úkol prostředníka

mezi modelem a pohledem je stejný. Pohled po požadavku uživatele volá určitou metodu presenteru, která dále může pracovat s modelem. Logika presenteru by se měla vymezit na úplnou změnu pohledu, změnu stavu aktuálního pohledu nebo příkazu pro model. Základní presenter projektu se nachází v app/presenters a má název HomepagePresenter.php a byl vytvořen již při samotné instalaci projektu.

Jeho struktura je následující:

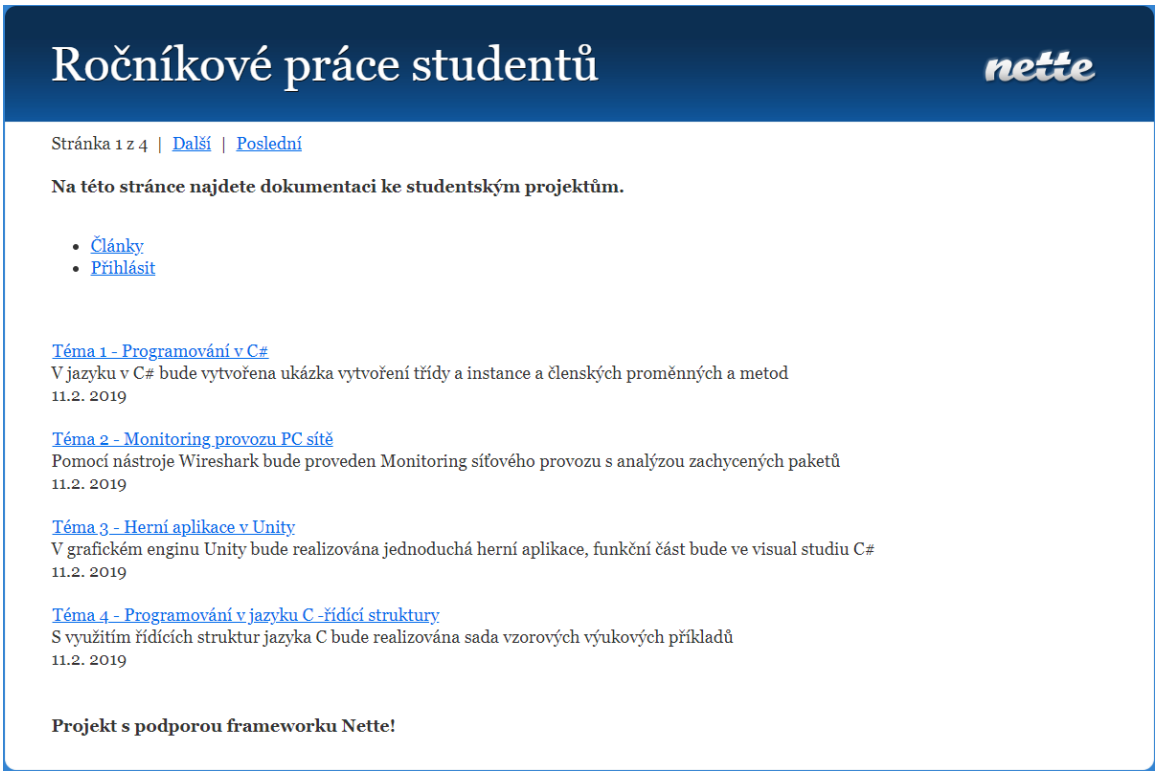
```
namespace App\Presenters;
use Nette;

class HomepagePresenter extends
Nette\Application\UI\Presenter
{
}
```

Obsahuje pouze základní prázdnou třídu, do které je vhodné umístit metody akcí probíhajících na domovské stránce aplikace. Dále se ve stejné složce nachází ještě dva presentery a to ErrorPresenter.php a Error4xxPresenter.php, které obsahuji metody pro volání různých chybových hlášení a zápis chyb do logovacího souboru.

4 ÚVODNÍ STRÁNKA APLIKACE

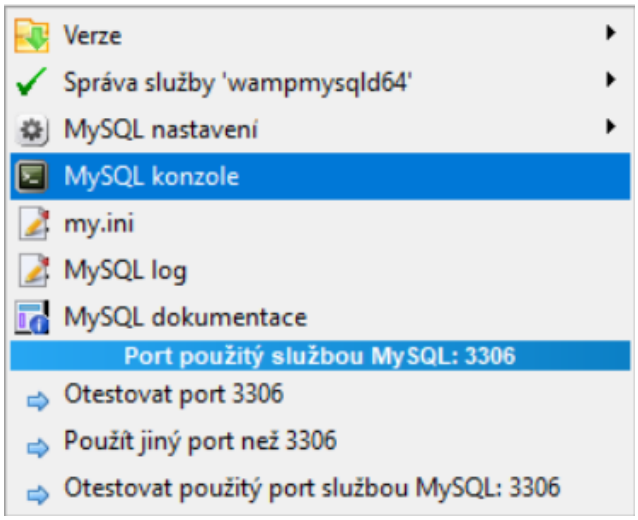
Aplikace bude zobrazovat vložené příspěvky, ke každému příspěvku bude možné přidat komentáře, bude aplikováno stránkování, přihlášení oprávněného uživatele, bude realizován přístup do databáze, vložena tabulka, stránkování tabulky a registrace uživatelů. Na úvodní stránce se bude vypisovat přehled příspěvků uložených do databáze. Databáze bude vytvořena pomocí phpMyAdmin, nastavení přístupu do databáze v config.neon a vytvoření dotazu v HomepagePresenter.php. Stránkování bude vytvořeno tak, aby se na stránce zobrazoval jen předem stanovený počet článků. S narůstajícím počtem příspěvků bude narůstat pouze počet stránek a grafická přehlednost webu zůstane zachována.



Obr. 9 – Úvodní stránka aplikace - zdroj: vlastní

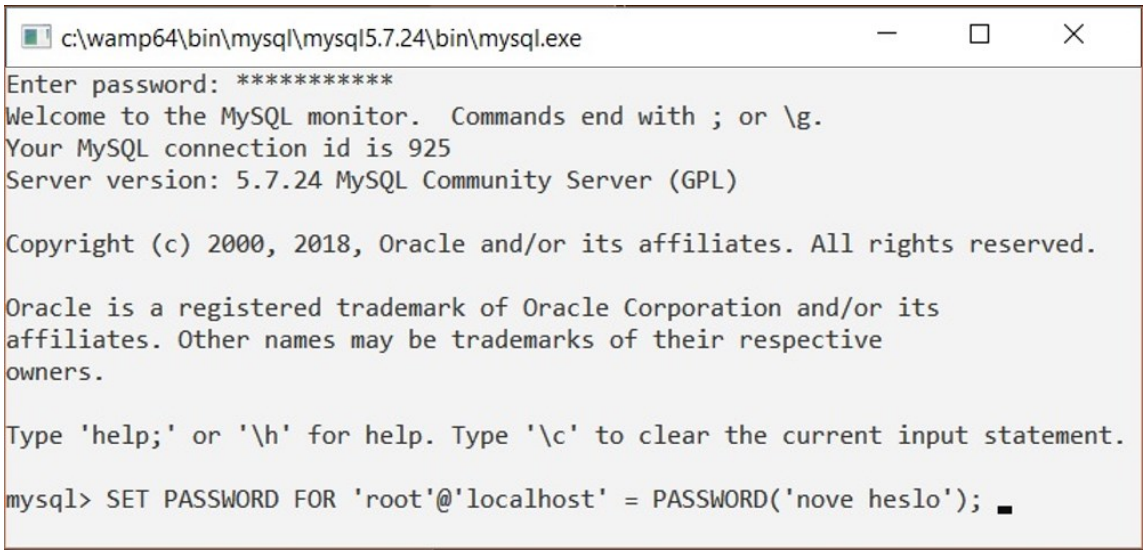
4.1 Vytvoření databáze

Přes ikonu Wampu v oznamovací oblasti se připojíme k phpMyAdmin [13]. Ve výchozím nastavení je nastaveno přihlašovací jméno ‘root‘, heslo je prázdné a typ databáze je MySQL. Je vhodné heslo účtu ‘root‘ přenastavit.



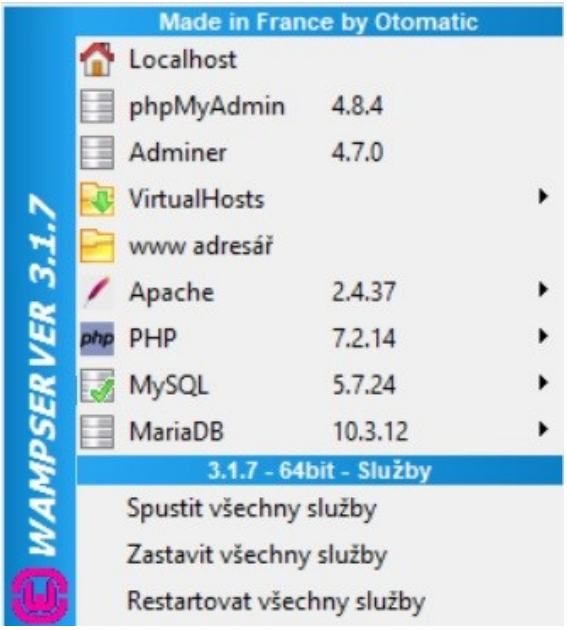
Obr. 10 – Přihlášení ke konzoli - zdroj: vlastní

To se provede tak, že se po přihlášení na wamp->MySQL->MySQL konzole direktivou SET PASSWORD FOR 'root'@'localhost' = PASSWORD('nove heslo'); nastaví nové heslo.



Obr. 11 – Změna hesla MySQL - zdroj: vlastní

Po přihlášení do phpMyAmin je možné vytvořit databázi v záložce Databáze s názvem `posts` kódováním utf-8 a dále pomocí následující syntaxe v záložce SQL stejnojmennou tabulku.

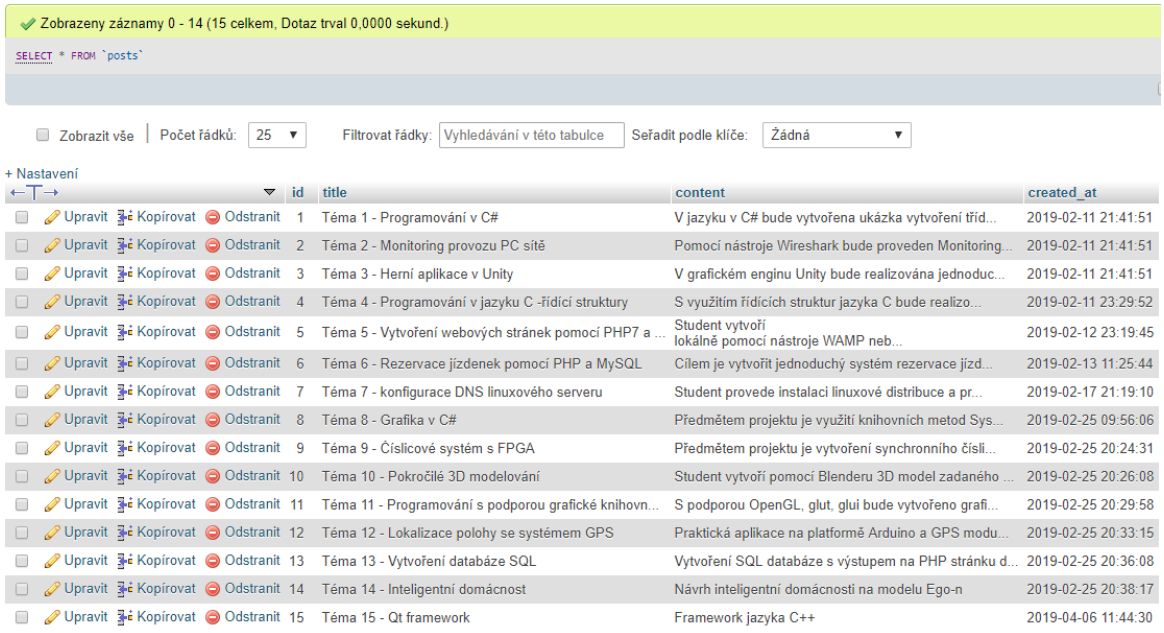


Obr. 12 – WAMP server - zdroj: vlastní

Tabulka obsahuje jedinečné id příspěvku, které se automaticky inkrementuje, titulek představující název příspěvku, samotný text příspěvku a čas vložení. Příspěvky bude vkládat přihlášený uživatel, který bude mít statut admina. Přihlášení tohoto uživatele je popsáno v kapitole 7 a 8.6.

```
CREATE TABLE `posts` (
    `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    `title` varchar(255) NOT NULL,
    `content` text NOT NULL,
    `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB CHARSET=utf-8;
```

Primární klíč je nastaven na položku id, která se automaticky inkrementuje při vložení dalších příspěvků. Dalšími atributy tabulky jsou titulek a obsah příspěvku a časový údaj, kdy byl příspěvek do databáze vložen. Nyní lze tabulku naplnit daty.



	id	title	content	created_at
Upravit Kopírovat Odstranit	1	Téma 1 - Programování v C#	V jazyku v C# bude vytvořena ukázka vytvoření tříd...	2019-02-11 21:41:51
Upravit Kopírovat Odstranit	2	Téma 2 - Monitoring provozu PC sítě	Pomocí nástroje Wireshark bude proveden Monitoring...	2019-02-11 21:41:51
Upravit Kopírovat Odstranit	3	Téma 3 - Herní aplikace v Unity	V grafickém enginu Unity bude realizována jednoduc...	2019-02-11 21:41:51
Upravit Kopírovat Odstranit	4	Téma 4 - Programování v jazyku C -řídící struktury	S využitím řídících struktur jazyka C bude realizo...	2019-02-11 23:29:52
Upravit Kopírovat Odstranit	5	Téma 5 - Vytvoření webových stránek pomocí PHP7 a ...	Student vytvoří lokálně pomocí nástroje WAMP neb...	2019-02-12 23:19:45
Upravit Kopírovat Odstranit	6	Téma 6 - Rezervace jízdenek pomocí PHP a MySQL	Cílem je vytvořit jednoduchý systém rezervace jízd...	2019-02-13 11:25:44
Upravit Kopírovat Odstranit	7	Téma 7 - konfigurace DNS linuxového serveru	Student provede instalaci linuxové distribuce a pr...	2019-02-17 21:19:10
Upravit Kopírovat Odstranit	8	Téma 8 - Grafika v C#	Předmětem projektu je využití knihovních metod Sys...	2019-02-25 09:56:06
Upravit Kopírovat Odstranit	9	Téma 9 - Číslicové systém s FPGA	Předmětem projektu je vytvoření synchronního čísli...	2019-02-25 20:24:31
Upravit Kopírovat Odstranit	10	Téma 10 - Pokročilé 3D modelování	Student vytvoří pomocí Blenderu 3D model zadaného ...	2019-02-25 20:26:08
Upravit Kopírovat Odstranit	11	Téma 11 - Programování s podporou grafické knihovn...	S podporou OpenGL, glut, glui bude vytvořeno grafi...	2019-02-25 20:29:58
Upravit Kopírovat Odstranit	12	Téma 12 - Lokalizace polohy se systémem GPS	Praktická aplikace na platformě Arduino a GPS modu...	2019-02-25 20:33:15
Upravit Kopírovat Odstranit	13	Téma 13 - Vytvoření databáze SQL	Vytvoření SQL databáze s výstupem na PHP stránku d...	2019-02-25 20:36:08
Upravit Kopírovat Odstranit	14	Téma 14 - Inteligentní domácnost	Návrh inteligentní domácnosti na modelu Ego-n	2019-02-25 20:38:17
Upravit Kopírovat Odstranit	15	Téma 15 - Qt framework	Framework jazyka C++	2019-04-06 11:44:30

Obr. 13 – Databáze příspěvků, pohled phpMyAdmin - zdroj: vlastní

Další práce s databází je vhodné provádět pomocí Admineru, který je součástí wampu. Adminer umožňuje přihlášení přímo ke konkrétní databázi, tzn., můžeme pracovat jen s databází, ke které máme oprávnění.

4.2 Připojení k databázi

Aby naše aplikace mohla s databází pracovat, je potřeba vytvořit nastavení pro připojení k databázi. To provedeme v souboru *app/config/config.neon*. V tomto souboru jsou všechna globální nastavení aplikace.

```
database:
    dns: 'mysql:host=127.0.0.1;dbname=posts'
    user: root
    password: *heslo k databázi*
```

Pro naše potřeby byl ponechán účet root se změněným heslem, ovšem při reálném provozu na serveru bude nutné použít jiný účet.

4.3 Předání databázového spojení

Do složky *app/model* se vytvoří skript s názvem *ArticleManager.php* se stejnojmennou třídou. Tento soubor představuje model projektu.

```
namespace App\Model;
use Nette;

class ArticleManager
{
    use Nette\SmartObject;
    ……
}
```

Do této třídy je umístěn konstruktor, který zajistí přístup do databáze.

```
private $database;

public function __construct(Nette\Database\Context $database)
{
   this->database = database;
}
```

Dále je potřeba přidat metodu *findPublishedArticle()*, která zajistí dotazování do tabulky post. Záznamy z tabulky se budou zobrazovat podle data vložení od nejstaršího k nejnovějšímu.

```
public function findPublishedArticles()
    {
        return $this->database->table('posts')
            ->where('created_at < ', new \DateTime)
            ->order('created_at ASC');
    }
```

Ještě je nutné provést routování na vytvořený model a to v souboru config.neon následujícím záznamem

```
services:
    router: App\RouterFactory::createRouter
    - App\Model\ArticleManager
```

4.4 Stránkování

Zde je využito souboru *HomapagePresenter.php*. Tento soubor byl vytvořen již při instalaci ve složce *app/presenters*. Obsahuje prázdnou třídu *HomepagePresenter()*. Do této třídy se umístí konstruktor, který z modelu převezme spojení na databázi

```
private $articleManager;

public function __construct(ArticleManager $articleManager)
{
     $this->articleManager = $articleManager;
}
```

Metoda *renderDefault()* si do proměnné $posts načte publikované články a následně pošle do šablony jen jejich část

```
public function renderDefault($page = 1)
{
   $posts = $this->articleManager->findPublishedArticles();
}
```

Do šablony je předána pouze jejich část omezená podle nastavení v metodě page, zobrazeny jsou čtyři položky na každé stránce.

```
   $lastPage = 0;
   $this->template->posts = $posts->page($page, 4, $lastPage);
```

Musíme také předat potřebná data pro zobrazení možnosti stránkování

```
$this->template->page = $page;
$this->template->lastPage = $lastPage;
```

4.5 Šablona stránkování

Samotná šablona je umístěna do souboru *default.latte*. Tato šablona je v *app/presenters/templates/Homepage*. Blok pojmenovaný pagination vytvoří na stránce odkazy na předchozí a následující stránky s příspěvky. Proměnné *$page* a *$lastPage* řídí zobrazování stránkování.

```
<div class="pagination">
    {if $page > 1}
        <a n:href="default, 1">První</a>
         | 
        <a n:href="default, $page-1">Předchozí</a>
         | 
    {/if}

    Stránka {$page} z {$lastPage}

    {if $page < $lastPage}
         | 
        <a n:href="default, $page+1">Další</a>
         | 
        <a n:href="default, $lastPage">Poslední</a>
    {/if}
</div>
```

4.6 Umístění do hlavní šablony

Hlavní šablona byla opět vytvořena při instalaci projektu, je umístěna ve složce *app/presenters/templates* a nese název *@layout.latte*. Struktura šablony je stejná jako struktura běžného html souboru. Obsahuje hlavičkovou část head s titulkem, vlastní tělo stránky, metadata a celá je uzavřená pomocí tagů html.

```
<!DOCTYPE html>
<html>
<head>
   <title>{ifset title}{include title|stripHtml} |
                                        {/ifset}Diplomka</title>
</head>

<body>
     {include content}
     {block scripts}
     {/block}
</body>
</html>
```

Metadata – na tomto místě se uvádí například požitá znaková sada

```
<meta charset="utf-8">
```

Blok css – je určen k zavedení formátování stránky pomocí kaskádových stylů

```
{block css}
   <link rel="stylesheet" href="{$basePath}/css/styl1.css">
{/block}
```

Flash zpráva – message box je zde použitý jako prvek pro ohlášení chybového stavu, konkrétně nezadané přihlašovací údaje

```
<div n:foreach="$flashes as $flash" n:class="flash, $flash-
>type">{$flash->message}</div>
```

Blok skriptů – slouží pro zavedení javasriptu a dalších použitých skriptů

```
{block scripts}
  <script src="{$basePath}/js/netteForms.min.js"></script>
{/block}
```

Include content – vkládá do hlavní šablony blok s názvem content. Vytvoření {block content} je umístěno ve všech šablonách projektu.

```
{block content}
     …obsah šablony
{/block}
```

Uzavírací tag bloku není povinný.

5 STRÁNKA ZOBRAZENÍ PŘÍSPĚVKU

Aby bylo možné pracovat s každým příspěvkem zvlášť, upravovat ho, přidávat k němu komentáře bude projekt doplněný o stránku příspěvku, která bude obsahovat samotný příspěvek, datum jeho publikace, formulářem pro přidávání komentářů a jejich výpisem. Do složky *app/presenters* se založí skript s názvem *PostPresenter.php*.

Kostra presenteru vypadá takto:

```
namespace App\Presenters;
use Nette;
use Nette\Application\UI\Form;

class PostPresenter extends Nette\Application\
                                        UI\Presenter
{
}
```

5.1 Připojení k databázi

Samotný přístup k databázi se provede pomocí konstruktoru. Přístupové údaje jsou uvedeny v souboru *config.neon*.

```
private $database;
public function __construct(Nette\Database\Context
                                            $database)
{
   $this->database = $database;
}
```

5.2 Metoda renderování

Metoda vrací příspěvek z databáze, jejím argumentem je id příspěvku z tabulky *posts*. Metoda *get()* převezme id příspěvku z aktuálně připojené databáze, konkrétně tabulky *posts* a předá ji do proměnné post ke zpracování v šabloně.

```
public function renderShow($postId)
{
 this->template->post = $this->database->table('
                                        posts')->get($postId);
}
```

5.3 Šablona zobrazení příspěvku

Dosavadní skript je nutné také někde vykreslit. To zajistí šablona stránky příspěvku. Její umístění je ve složce *app/presenters/templates/Post* a ponese název *show.latte*

```
{block content}
<p><a n:href="Homepage:default">zpět na příspěvky</a></p>
<div class="date">{$post->created_at|date: 'j.n. Y'}</div>
<h1 n:block="title">{$post->title}</h1>
<div class="post">{$post->content}</h1>
```

6 VYTVOŘENÍ KOMENTÁŘE PŘÍSPĚVKU

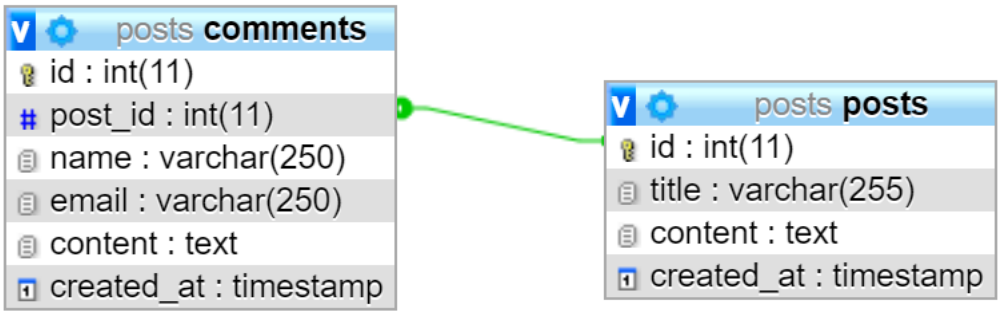
Také je možné vytvořit komentáře k jednotlivým příspěvkům a nechat je zobrazovat pouze na konkrétní stránce nejlépe podle data vložení. K tomuto účelu bude vytvořena tabulka pro ukládání komentářů s vazbou na tabulku příspěvků, dále formulář se jménem, e-mailem a samotným komentářem diskutujícího, metoda pro vložení do tabulky a úprava renderování stránky.

6.1 Tabulka komentářů

Nejprve je nutné vytvořit v databázi tabulku, kam se budou komentáře zapisovat.

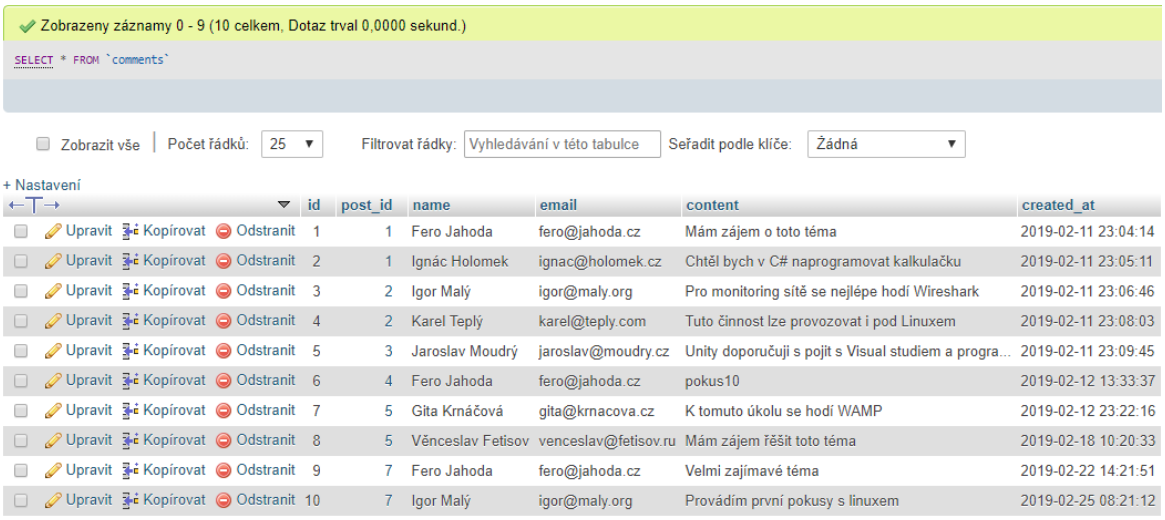
```
CREATE TABLE `comments` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
   `post_id` int(11) NOT NULL,
   `name` varchar(255) NOT NULL,
   `email` varchar(255) NOT NULL,
   `content` text NOT NULL,
   `created_at` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
   FOREGIN KEY (`post_id`) REFERENCES `posts` (`id`)
) ENGINE=InnoDB CHARSET=utf-8;
```

Tabulka komentářů je relačně vázána na tabulku příspěvků. Každý komentář má svoje jedinečné id, stejně tak i každý příspěvek. V tabulce comments je zaveden atribut `post_id`, který je v relaci s atributem `id` tabulky `posts`.



Obr. 14 – Relace tabulek – zdroj: vlastní

Tabulka obsahuje id identifikátor jednotlivých komentářů, posts_id identifikátor jednotlivých příspěvků je nastaven na jedenácti prvkové celočíselné pole, které se automaticky inkrementuje, dále tabulka obsahuje pole post_id, ve kterém jsou id identifikátory příspěvků, k nimž komentář patří, jméno komentujícího, jeho mailový kontakt, samotný text komentáře a datum jeho vložení.



Zobrazeny záznamy 0 - 9 (10 celkem, Dotaz trval 0,0000 sekund.)

```
SELECT * FROM `comments`
```

Zobrazit vše | Počet řádků: 25 | Filtrovat řádky: Vyhledávání v této tabulce | Seřadit podle klíče: Žádná

+ Nastavení

	id	post_id	name	email	content	created_at
Upravit Kopírovat Odstranit	1	1	Fero Jahoda	fero@jahoda.cz	Mám zájem o toto téma	2019-02-11 23:04:14
Upravit Kopírovat Odstranit	2	1	Ignác Holomek	ignac@holomek.cz	Chtěl bych v C# naprogramovat kalkulačku	2019-02-11 23:05:11
Upravit Kopírovat Odstranit	3	2	Igor Malý	igor@maly.org	Pro monitoring sítě se nejlépe hodí Wireshark	2019-02-11 23:06:46
Upravit Kopírovat Odstranit	4	2	Karel Teplý	karel@teply.com	Tuto činnost lze provozovat i pod Linuxem	2019-02-11 23:08:03
Upravit Kopírovat Odstranit	5	3	Jaroslav Moudrý	jaroslav@moudry.cz	Unity doporučuji s pojit s Visual studiem a progra...	2019-02-11 23:09:45
Upravit Kopírovat Odstranit	6	4	Fero Jahoda	fero@jahoda.cz	pokus10	2019-02-12 13:33:37
Upravit Kopírovat Odstranit	7	5	Gita Krnáčová	gita@krnacova.cz	K tomuto úkolu se hodí WAMP	2019-02-12 23:22:16
Upravit Kopírovat Odstranit	8	5	Věnceslav Fetisov	venceslav@fetisov.ru	Mám zájem řešit toto téma	2019-02-18 10:20:33
Upravit Kopírovat Odstranit	9	7	Fero Jahoda	fero@jahoda.cz	Velmi zajímavé téma	2019-02-22 14:21:51
Upravit Kopírovat Odstranit	10	7	Igor Malý	igor@maly.org	Provádím první pokusy s linuxem	2019-02-25 08:21:12

Obr. 15 - Databáze komentářů, pohled phpMyAdmin - zdroj: vlastní

Po úspěšném vytvoření tabulky pro vkládání komentářů, lze sestavit skript pro zobrazení příspěvku a jeho komentování.

6.2 Úprava renderování stránky

Metoda pro renderování je doplněna o test hodnoty proměnné *$postId*. V případě, že neproběhne úspěšné načtení, provede se varovné hlášení. Pokud je identifikátor úspěšně načten, předá metoda příspěvek i s komentářem k zobrazení do šablony.

```
public function renderShow($postId)
{
 $post = $this->database->table('posts')->get($postId);
 if (!$post) {
      $this->error('Stránka nebyla nalezena');
 }
```

```
 $this->template->post = $post;
 $this->template->comments = $post->related('
                       comment')->order('created_at');
}
```

6.3 Metoda komentáře do tabulky databáze

Metoda formuláře pro vložení komentáře nejprve vytvoří nový formulář. Do formuláře je postupně vloženo pole pro zadání jména, pole pro e-mailovou adresu, textové pole samotného komentáře, tlačítko pro samotné publikování komentáře a jako poslední prvek je volání metody *commentFormSucceeded()*, která převezme obsah jednotlivých polí. Na stránce se zobrazí jako komentář.

```
protected function createComponentCommentForm()
{
 $form = new Form;
 $form->addText('name', 'Jméno:')
      ->setRequired();
 $form->addEmail('email', 'Email:');
 $form->addTextArea('content', 'Komentář:')
      ->setRequired();
 $form->addSubmit('send', 'Publikovat komentář');
 $form->onSuccess[] = [$this, 'commentFormSucceeded'];
 return $form;
}
```

6.4 Metoda uložení komentáře příspěvku

Zatím je vytvořen pouze formulář pro ukládání komentářů, ještě je potřeba vytvořit metodu, která obsah formuláře zapíše do databáze a následně zobrazí. Touto metodou je metoda *commentFormSucceeded()*. Metoda přebírá postId identifikátor příspěvku a insert dotazem vkládá všechny zadané hodnoty polí. Jako poslední proběhne flashMessage se zprávou o vložení komentáře a jeho zobrazení na stránce pomocí *redirect();*
Metoda vložení komentáře do tabulky databáze:

```
public function commentFormSucceeded(Form $form, \stdClass
$values)
{
 $postId = $this->getParameter('postId');

 $this->database->table('comments')->insert([
         'post_id' => $postId,
         'name' => $values->name,
         'email' => $values->email,
         'content' => $values->content,
       ]);

 $this->flashMessage('Děkuji za komentář', 'success');
 $this->redirect('this');
}
```

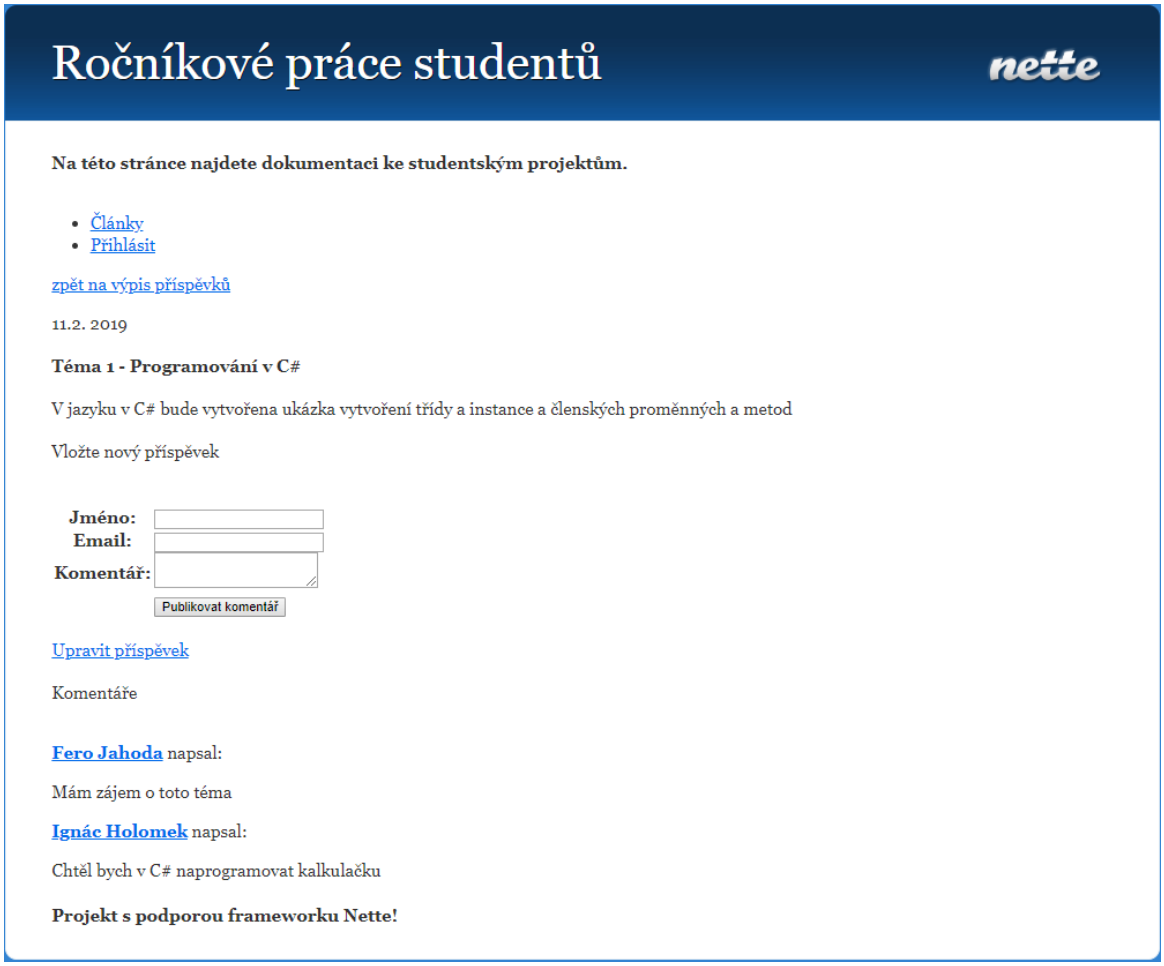
6.5 Úprava šablony

V *app/presenters/templates/Post* se nachází šablona stránky s komentářem a nese název *show.latte*. Samotná šablona už nenese znaky webové stránky tak, jako hlavní šablona projektu, ale obsahuje blok s názvem content. Pomocí tohoto bloku se inkluduje obsah do hlavní šablony. V bloku content jsou jednotlivé části rozděleny do sekcí pomocí párového tagu , nadpisů, maker a odkazů.

```
{block content}
    <p><a n:href="Homepage:default">zpět na výpis
příspěvků</a></p>
    <div class="date">{$post->created_at|date:'j.n. Y'}</div>
     <h2><b>{$post->title}</b></h2>
     <div class="post">{$post->content}</div>

     <h2>Vložte nový příspěvek</h2><br>
     {control commentForm}
    <div class="comments"><br>
```

```
    <a n:href="edit $post->id">Upravit příspěvek</a>
    <h2>Komentáře</h2>
    {foreach $comments as $comment}
        <p><b><a href="mailto:{$comment->email}" n:tag-
if="$comment->email">{$comment->name}</a></b> napsal:</p>
        <div>{$comment->content}</div>
    {/foreach}
</div>
```



Obr. 16 – Stránka příspěvku s komentáři - zdroj: vlastní

7 VLOŽENÍ NOVÉHO PŘÍSPĚVKU, EDITACE, ÚPRAVA

Aby bylo možné přidávat příspěvky do databáze a zobrazovat je na stránkované hlavní stránce webu, je nutné vytvořit další stránku aplikace, pomocí které bude možné příspěvky přidávat. Ta bude realizována pomocí presenteru a jeho šablony pro zobrazení. Příspěvky jsou zapisovány do databáze ‘posts‘ stejnojmenné tabulky ‘posts‘. Šablony pro práci s vložením, editací a úpravou příspěvku jsou místěny v *app/presenters/templates/Post/.*

7.1 Šablona příspěvků

Šablona vytvoření nového nese název *create.latte*. Jedná se o dílčí šablonu, jejíž obsah bude nahrán do celkové šablony aplikace, proto se celý její obsah nachází v bloku content a obsahuje jednotlivé html bloky , některé tagy z html kódu a umístění formuláře z metody pro vložení příspěvku.

```
{block content}
<div id="banner">
     <h1 n:block=title>Ročníkové práce studentů</h1>
</div>

<div id="content">
     <h2><b>Na této stránce najdete dokumentaci ke
                       studentským projektům.</b></h2>

    <p><a n:href="Homepage:default">zpět na výpis
příspěvků</a></p>

     <h1>Nový příspěvek</h1><br>
     {control postForm}

  <div>
    <h4>Projekt s podporou frameworku Nette!</h4>
  </div>
</div>
```

7.2 Metoda formuláře

Metoda formuláře pro vložení nového příspěvku se nachází v souboru *PostPresenter.php*. Tato metoda vytváří formulář s text boxem pro titulek příspěvku textové pole pro samotný příspěvek, tlačítko vložení příspěvku a volání metody *postFormSucceeded()*, která provede zápis obsahu formuláře do tabulky databáze.

```
protected function createComponentPostForm()
{
  $form = new Form;
  $form->addText('title', 'Titulek:')
       ->setRequired();
  $form->addTextArea('content', 'Obsah:')
       ->setRequired();

  $form->addSubmit('send', 'Uložit a publikovat');
  $form->onSuccess[] = [$this, 'postFormSucceeded'];

  return $form;
}
```

7.3 Metoda uložení příspěvku

Metoda uložení nového příspěvku do databáze přebírá argument formuláře a data z něho získané zpracuje. Otestuje, zda je uživatel přihlášený, pokud ano příspěvek přidá a vypíše na stránku, pokud ne, provede hlášení o nutnosti přihlásit se.

```
public function postFormSucceeded(Form $form, \stdClass
$values)
{
  $postId = $this->getParameter('postId');

  if(!$this->getUser()->isLoggedIn()) {
    $this->error('Pro vytvoření, nebo editování
```

```
                příspěvku se musíte přihlásit.');
  }

if ($postId){$post = $this->database->table('posts')
                                        ->get($postId);
 $post->update($values);
}
else {
 $post = $this->database->table('posts')
                                      ->insert($values);
 }
  $this->flashMessage('Příspěvek byl úspěšně
                            publikován.', 'success');
  $this->redirect('show', $post->id);
}
```

7.4 Metoda editace příspěvku

Metoda editace stávajícího příspěvku si nejprve načte obsah samotného příspěvku z databáze a umístí jednotlivé položky do příslušných oken formuláře. Při tom také otestuje, zda bylo možné příspěvek načíst. Pokud ne proběhne hlášení o nenalezení příspěvku. Také otestuje hned na začátku, jestli je uživatel přihlášený. Pokud není uživatel přihlášený, proběhne přesměrování na stránku přihlášení uživatele.

```
public function actionEdit($postId)
{
 $post = $this->database->table('posts')
                                        ->get($postId);
 if(!$this->getUser()->isLoggedIn()) {
    $this->redirect('Sign:in');
 }
 if(!$post) {
    $this->error('Příspěvek nebyl nalezen');
```

```
 }
 $this['postForm']->setDefaults($post->toArray());
}
```

7.5 Metoda přesměrování nepřihlášených uživatelů

Metoda přesměrování nepřihlášených uživatelů je aktivována v případě, že není uživatel přihlášený a klikne na odkaz upravit nebo vložit příspěvek.

```
public function actionCreate()
{
  if(!$this->getUser()->isLoggedIn()) {
    $this->redirect('Sign:in');
  }
}
```

7.6 Metoda úpravy stávajících příspěvků

Metoda *postFormSucceeded()* je upravena tak, že podle id příspěvku ukládá zpět do tabulky, resp. přepisuje původní obsah pomocí update.

```
public function postFormSucceeded(Form $form, \stdClass
$values)
{
  $postId = $this->getParameter('postId');

  if (!$this->getUser()->isLoggedIn()) {$this->error('Pro
  vytvoření, nebo editování příspěvku se musíte přihlásit.');
  }

  if ($postId) {
    $post = $this->database->table('posts')->get($postId);
            $post->update($values);
  }
  else {
```

```
    $post = $this->database->table('posts')->insert($values);
  }

  $this->flashMessage('Příspěvek byl úspěšně publikován.',
                                                    'success');
      $this->redirect('show', $post->id);
    }
```

Do složky *app/presenters/templates/Post/* byla vytvořena ještě jedna šablona *edit.latte* pro práci s upraveným příspěvkem. Její hlavní částí je link na formulář pro editování příspěvku.

```
{block content}
     <h1>Upravit příspěvek</h1>
 {control postForm}
```

8 AUTENTIZACE, AUTORIZACE, ZABEZPEČENÍ

Pod pojmem bezpečnost či zabezpečení rozumíme ochranu informačních systémů a informací, které jsou v nich uchovávány, přenášeny a zpracovávány. Patří sem např. pojmy jako bezpečnost informačních systémů, ochrana informačních systémů, ochrana informací, ochrana informačních technologií. Všechny tyto pojmy mají význam při řešení bezpečnosti a ochrany informačních systémů, informací v nich uložených a způsobu jejich zpracovávání. Jak jsou informační technologie aplikovány, stává se mnohem snazší shromažďovat data, ukládat je, manipulovat s nimi a hromadně je šířit. Proto vyvstal požadavek na zajištění informačních systémů před kybernetickými útoky, integrity dat, spolehlivosti fungování informačních systémů a zajištění trvalé dostupnosti služeb těchto systémů.[29]

Zabezpečení dat je množina pravidel a mechanismů pro zajištění důvěrnosti, integrity a dostupnosti dat především zvenčí zabezpečovaného systému.

Samotné zabezpečení se skládá z několika úrovní. Tou první je fyzické zabezpečení informačního systému na úrovni hardwaru, zálohování dat, druhou úrovní softwarovou a třetí úrovní organizační.

Kybernetická bezpečnost je podpořena i platnou legislativou, tj. konkrétními zákony v aktuálním znění, které jasně vymezují požadavky na zajištění důvěrnosti dat (secrecy), ochranu integrity dat (integrity) a zajištění dostupnosti dat a služeb (availibility).[28]

Zajištění důvěrnosti znamená, že každá operace s daty nese určitou míru utajení. Je snaha zajistit, aby jakýkoliv neautorizovaný subjekt nemohl neoprávněně vniknout do systému, tedy kontroluje oprávněnost konkrétní osoby nebo jiného subjektu k přístupu do systému. Integrita dat zhmotňuje přesnost a spolehlivost informačního systému, logickou úplnost hardware a software, které implementuje ochranné mechanismy a přispívá k zabránění neautorizovaného přístupu do systému nepovolaným subjektům.

Zajištění dostupnosti dat znamená, že je zajištěn včasný přístup ke spolehlivým ověřeným údajům, že autorizovaným uživatelům nebude přístup odepřen a naopak neautorizovaný uživatel nebude do systému vpuštěn. Je potřeba rozlišovat dva pojmy – autorizace a autentizace, které nejsou v žádném případě synonyma pro totéž, ale každý z pojmů má jasně daný význam.

8.1 Autentizace

Autentizace je proces ověření entity, zda je skutečně tou, za kterou se vydává. K tomuto účelu se používají uživatelská jména, hesla a mnoho dalších jedinečných ověřitelných identifikátorů. Mezi doplňkové identifikátory můžeme zahrnout údaje biometrické, jako je otisk prstu, sken tváře, identifikační sms, náhodné generování čísel v tokenech a další, tedy jednoznačně určit a identifikovat přihlašujícího se uživatele. Schopnost systému zachovat mlčenlivost o ověřování identity odráží úroveň bezpečnosti daného systému.

Poskytování identity musí jít ruku v ruce s procesem přidělení uživatelských práv, každý krok sám o sobě nemá žádný význam. Každý informační systém vždy uživatele provede nejprve procesem autentizace a bezprostředně po té procesem autorizace.[21]

Autentizace pomocí hesel je nejrozšířenější autentizační technikou. Při ní uživatel zadává přihlašovací jméno v kombinaci s heslem. Hlavním aspektem při tomto způsobu autentizace je kvalita hesla. To by mělo mít určitý počet znaků, znaky by měly být kombinací malých a velkých písmen a číslic. Také by se mělo heslo v přiměřených intervalech měnit.

Autentizace pomocí tokenů vnáší do procesu přihlašování další technický prvek. Podmínkou je, aby měl uživatel v okamžiku přihlašování token u sebe. Je to z toho důvodu, že se na tokenu pravidelně mění generovaný údaj. Bývá to například jedna minuta. Tokeny mají podobu čipové karty, USB klíčenky apod. Tokeny jsou spolu s biometrikou považovány za silný autentizační nástroj. Z paměťových tokenů stojí za zmínku čipová karta spolu s pinem. Karta je sama o sobě zařízení, bez kterého se nelze identifikovat, nejčastěji při manipulaci s bankomatem a v kombinaci s číselným pinem vytváří velmi odolný nástroj. Navíc konkrétně v bankomatech je autentizace omezena pravidlem třikrát a dost. To znamená, že po třech neúspěšných pokusech je karta bankomatem stornována a znehodnocena. [21]

Při procesu autentizace za pomoci biometriky, se identifikuje některá fyziologická nebo biologická vlastnost uživatele. Do této kategorie patří například otisk prstu, otisk dlaně, sken obličeje, ale i dynamika některých pohybů, např. intenzita chůze, rychlost a dynamika stisku kláves a další.

8.2 Autorizace

Autorizace je proces, při němž se ověřují práva pro přístup přihlášeného uživatele ke službám. Patří sem i ochrana integrity dat, to znamená, že je potřeba zabezpečit, aby s daty mohli manipulovat jen oprávněné osoby. Systém musí uživatele rozeznat a to dostatečně

rychle, aby nedocházelo k časovým prodlevám, ve kterých by mohlo docházet k neoprávněné manipulaci a modifikaci dat neautorizovaným uživatelem.

Vedle autorizace a autentizace existuje celá řada metod jak zajistit bezpečnost dat. Jsou to metody, které většinou následují proces identifikace a přidělení oprávnění uživateli, patří sem například zabezpečení fyzického přístupu k nosičům dat, pořizování zálohování a kopií, provádění obnovy dat ze záloh a další. Všechna opatření musí zajistit ochranu důvěrnosti a integritu dat a také samotné autentizace.

8.3 Kryptografie

Většina principů zabezpečené autentizace uživatele funguje na principu šifrování dat. Šifrování je nedílnou součástí všech autorizačních procesů. Je to proces, který převádí autentizační údaje z podoby běžně čitelné do podoby nesrozumitelné za pomoci symetrického nebo asymetrického šifrování a klíčů.[20]
Celý proces přihlašování probíhá ve třech fázích takto. V první fázi se účastníci komunikace, tedy přihlašovaný uživatel a server dohodnou na volbě šifrovacího algoritmu a délce klíče, který bude používán. Jednoduše řečeno, klient vyžadující ověření zašle požadavek, ve kterém je i nabídka algoritmů, které umí a server v odpovědi vybere jeden z nich. Ve druhé fázi se provede výměna klíčů založená na principu šifrování s veřejným klíčem a autentizaci vycházející z certifikátů. Ve třetí fázi probíhá šifrování provozu za pomoci symetrického šifrování. [20]

8.4 Zabezpečení v Nette

Nette poskytuje více způsobů jak naprogramovat autentifikaci. Implementace je pouze na programátorovi. Využívá rozhraní Nette\Security\IAuthenticator, které vyžaduje pouze jednu metodu authenticate(), která ověřuje uživatele. Nejčastější způsob ověření je pomocí hesla, uživatel poskytne jméno a heslo. Také lze využít i jiné způsoby, např., přihlásit pomocí účtu facebooku a podobně. Pro tak jednoduchou aplikaci není třeba vytvářet složitý způsob přihlašování. Použijeme připravený SimpleAuthenticator(), který je pro podobné účely připraven. Poskytuje přihlášení pomocí jména a hesla, které je uloženo v konfiguračním souboru. Konfigurační soubory jsou typu neon, může jich být i více a pak je nutné je zavést do projektu v souboru bootstrap.php následujícím příkazem.

```
$configurator->addConfig(__DIR__ . '/config/config2.neon');
```

Nette automaticky vytvoří službu s názvem authenticator v DI kontejneru.

8.5 Přihlášení a skrytí odkazu

Do konfiguračního souboru *confog.neon* se vloží sekce security a do ní se umístí údaje o přihlašovaném uživateli.

```
security:
    users:
        Admin: seCret  # user 'Admin', password 'seCret'
        Franta: Mrakota # user 'Franta', password 'Mrakota'
```

Aby bylo možné uživatele přihlásit, musí aplikace disponovat i nějakým formulářem pro zadání přihlašovacích údajů a metodami pro zpracování a ověření přihlášení. Pro tento účel je do složky *app/presenters* vytvořen soubor *SignPresenter.php* a do *app/presenters/templates/Sign* jeho šablona *in.late*.
Základem je třída stejně jako v HomepagePresenteru.

```
namespace App\Presenters;
use Nette;
use Nette\Application\UI\Form;
class SignPresenter extends Nette\Application\UI\Presenter
{
}
```

Do třídy jsou umístěny metody potřebné k přihlášení, odhlášení a otestování identity uživatele. Metoda formuláře přihlášení je pojmenována createComponentSignInForm(). Obsahuje pole pro zadání přihlašovacích údajů, při neúplnosti údajů informuje uživatele, aby je zadal všechny. Na jejím konci je volání metody *signInFormSucceeded1(),* která zpracuje data získaná z formuláře.

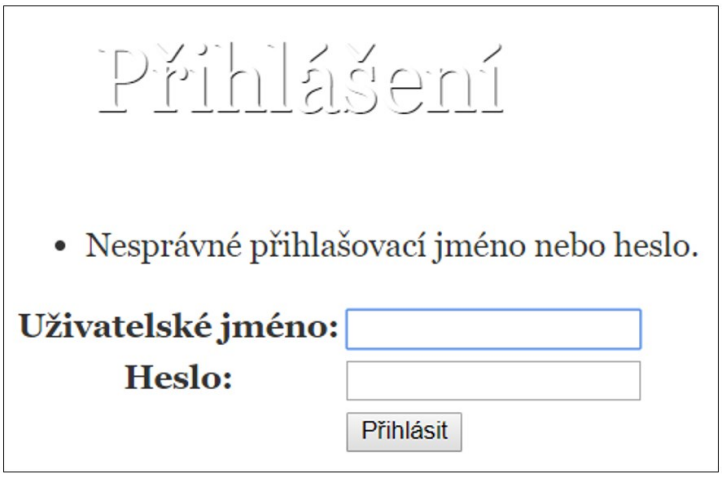
```
protected function createComponentSignInForm()
{
  $form = new Form;
  $form->addText('username', 'Uživatelské jméno:')
       ->setRequired('Prosím vyplňte své uživatelské jméno.');
  $form->addPassword('password', 'Heslo:')
       ->setRequired('Prosím vyplňte své heslo.');
  $form->addSubmit('send', 'Přihlásit');
  $form->onSuccess[] = [$this, 'signInFormSucceeded1'];
  return $form;
}
```

Obr. 17 – Formulář přihlášení - zdroj: vlastní

Volaná metoda *signInFormSucceeded1()* převezme uživatelské jméno a heslo, které uživatel vyplnil a předá to authenticatoru. Po přihlášení přesměrujeme na úvodní stránku. Pokud nesouhlasí přihlašovací údaje, je vyhozena výjimka

```
public function signInFormSucceeded1(Form $form, \stdClass
$values)
{
  try {
```

```
        $this->getUser()->login($values->username,
                                                $values->password);
        $this->redirect('Homepage:');
    }
    catch (Nette\Security\AuthenticationException $e) {
    $form->addError('Nesprávné přihlašovací jméno nebo heslo.');
    }
}
```



Obr. 18 – Nesprávném přihlášení - zdroj: vlastní

Pro odhlášení je vytvořena jednoduchá metoda, která provede samotné odhlášení, zobrazí zprávu o úspěšném odhlášení a přesměruje aplikaci na domovskou stránku.

```
public function actionOut()
{
    $this->getUser()->logout();
    $this->flashMessage('Odhlášení bylo úspěšné.');
    $this->redirect('Homepage:');
}
```

V šabloně je stále zobrazen v seznamu odkaz na domovskou stránku s příspěvky a dále pak buď odkaz na přihlášení, nebo odhlášení uživatele.

```
{block content}
<div id="content">
    <div>
        <ul class="navig">
            <li><a n:href="Homepage:">Články</a></li>
            {if $user->loggedIn}
                <li><a n:href="Sign:out">Odhlásit</a></li>
            {else}
                <li><a n:href="Sign:in">Přihlásit</a></li>
            {/if}
        </ul>
    </div>
    <h1>Přihlášení</h1>
        {control signInForm}
</div>
```

Obr. 19 – Skrytý odkaz vytvoření příspěvku - zdroj: vlastní

Mezi bezpečnostní prvky lze také zahrnout skrývání odkazů nepřihlášeným uživatelům. Neautorizovaný uživatel nemůže vidět ani stránku pro vytvoření nového příspěvku ani pro jeho editaci a ani odkazy na ně. Ty je potřeba také schovat. V šabloně default.latte je nutno upravit odkaz na napsání nového příspěvku takto.

.

```
<a n:href="Post:create" n:if="$user->loggedIn">Napsat nový
                                                příspěvek</a>
```

Ten zůstává viditelný pouze přihlášeným uživatelům. Schování odkazu je zajištěno makrem n:if a přihlášením uživatele, jinak zůstává skrytý a není možné neautentizovaně přidávat příspěvky.

Obr. 20 – Zobrazení odkazu po přihlášení - zdroj: vlastní

8.6 Autentizace proti databázi

V aplikaci jsou vytvořeny komentáře jednotlivých příspěvků. V prvotní verzi mohou komentáře vkládat neregistrovaní uživatelé. To může způsobit zaplavení příspěvků tzv. balastem. Tento problém lze vyřešit autentizací komentářů, tedy zpřístupnit komentáře jen přihlášeným uživatelům. Uživatele, kteří by se mohli přihlašovat k psaní komentářů, je

potřeba někde seskupit. K tomu bude sloužit tabulka `'users'`. V tabulce jsou údaje o registrovaném uživateli, konkrétně id, jméno a heslo.

```
CREATE TABLE `users` (
    `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    `name` varchar(255) NOT NULL,
    `password` varchar(255) NOT NULL
) ENGINE=InnoDB CHARSET=utf-8;
```

Pro ověření přístupových údajů můžeme vytvořit autenticator.

Třída authenticatoru

```
class MyAuthenticator implements NS\IAuthenticator
{
}
```

Přístup k databázi

```
private database;
public function __construct(Nette\Database\Context $database)
{
    $this->database = database;
}
```

Metoda přístupu ověření převezme prostřednictvím proměnné $credentials údaje k ověření

```
public function authenticate(array $credentials)
{
    list($username, $password) = $credentials;
    ……
}
```

Dále je potřeba přistoupit k databázi a načíst konkrétní údaje z tabulky `'users'`

```
$row = $this->database->table('users')->where('name',
                                        $username)->fetch();
```

Následuje test načtených údajů

```
if(!$row){
  throw new Nette\Security\AuthenticationException('Uživatel
                                                  nenalezen.');
}
```

A samotné ověření údajů – test hesla

```
if(!Nette\Security\Passwords::verify($password, $row->
                                                    password)){
   throw new Nette\Security\AuthenticationException('Neplatné
                                                      heslo.');
}
```

Vrácení přihlášeného uživatele

```
return new Nette\Security\Identity($row->id, row-
                                    >role,['name'=>$row->name]);
```

Je možné ověřit údaje i vlastním způsobem. V tom případě se doporučuje neukládat do tabulky hesla samotné, ale jen jejich hash. Hash je jednosměrná matematická funkce, kdy se z konkrétního hesla vytvoří pokaždé stejný otisk, opačně z otisku získat původní heslo nejde. Při přihlašování se pak ze zadaného hesla vytvoří otisk a ten se porovná s otiskem z databáze a v případně rovnosti otisků proběhne autentizace.

9 SESSIONS A COOKIES

Webové stránky pracují nad protokolem http, který sám o sobě nenese žádné prvky ukládání relací, což znamená, že při každé další návštěvě uživatele je tento považován za nového. Ten pak musí server znovu žádat o informace, které již v minulosti dostal. K tomu, aby uživatel mohl pokračovat ve stejné konfiguraci, při jaké stránky opustil, slouží soubory Cookies.[16]

9.1 Sessions

Session (relace, sezení) je potřeba k tomu, aby si aplikace mohla uchovávat informace o stavu spojení. Každý uživatel při přístupu na server obdrží jedinečný identifikátor Session ID, který se předává v Cookies. Identifikátor se ukládá na straně serveru, to umožňuje, aby po opětovném přístupu na server pamatoval, bylo možno navázat na předchozí relaci, například stav nákupního košíku.[17] Soubory Cookies se uchovávají na straně uživatele resp. prohlížeče. Konfigurace session se provádí v konfiguračním souboru config.neon v sekci session. Session standardně vyexpiruje při zavření okna prohlížeče. Dobu expirace nastavíme takto:

```
session:
      expiration: 14 days
```

Sessions startuje sama, ale lze ji konfigurací ovlivnit. Jsou k dispozici tři hodnoty true, false a smart. Doporučuje se ponechat autostart na hodnotě smart, protože pak automaticky startuje session pouze pokud je vytvořena.[3]

```
Session:

        autoStart: 'smart'
```

Na sdíleném hostingu je doporučeno zvolit vlastní adresář, kam se mají ukládat soubory s relacemi.[3]

```
session:
   savePath: "cesta/sessions"
```

Na lokálním serveru WAMP se soubory session ukládají do složky wamp64/temp/ pod názvem ses_nejakynazev.
Výpis takového souboru může vypadat takto

```
__NF|a:2:{s:4:"Time";i:1554625026;s:4:"DATA";a:1:
{s:23:"Nette.Http.UserStorage/";a:5:{s:13:"authenticated";
b:1;s:8:"identity";O:23:"Nette\Security\Identity":3:
{s:27:" Nette\Security\Identity id";s:6:"Franta";s:30:"
Nette\Security\Identity roles";a:0:{}s:29:"
Nette\Security\Identity data";a:0:{}}s:6:"reason";N;s:8:"authTime";
i:1557488686;s:11:"expireDelta";N;}}}_tracy|a:3:{s:8:"redirect";
N;s:3:"bar";a:0:{}s:10:"bluescreen";a:0:{}}
```

Obr. 21 – Výpis souboru session - zdroj: vlastní

Server předpokládá, že komunikuje stále s tímtéž uživatelem, dokud požadavky doprovází stejné Session ID. Úkolem bezpečnostních mechanismů je zajistit, aby tomu tak doopravdy bylo a nebylo možné identifikátor odcizit nebo podstrčit. Nette Framework proto správně nakonfiguruje PHP direktivy, aby Session ID přenášel pouze v cookie, znepřístupnil jej JavaScriptu a případné identifikátory v URL ignoroval. Navíc v kritických chvílích, jako je třeba přihlášení uživatele, vygeneruje Session ID nové.

9.2 Cookies

Pojmem „cookies“ se na internetu označují textové soubory, které server umísťuje do počítače uživatele webových stránek a které následně odesílají informace o chování uživatele zpět na příslušný server.[31] O používání těchto souborů musí být uživatel informován, děje se tak vyskakovacím message zprávou při návštěvě webu.
Fungování cookies je řízeno platnou legislativou Nařízení Evropského parlamentu a Rady (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES označované též jako GDPR (General Data Protection Regulation), představuje

právní rámec ochrany osobních údajů platný na celém území Evropské unie**.** Účinnosti nabylo dnem 25. května 2018. Nařízení se dotýká všech subjektů zpracovávajících osobní údaje, a to napříč odvětvími. Nahrazuje předchozí právní úpravu **-** Směrnici Evropského parlamentu a Rady 95/46/ES ze dne 24. října 1995 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů. GDPR tedy nepřináší revoluční změny v problematice ochrany osobních údajů. Subjekty, které v současné době naplňují zákon č. 101/2000 Sb.**,** o ochraně osobních údajů a o změně některých zákonů, a v případě orgánů veřejné moci též zákon č. 365/2000 Sb., o informačních systémech veřejné správy, již velmi pravděpodobně splňují i podstatu GDPR**.** [30]

Soubory Cookies umožňují uživatelům udržovat relace, tzn., zaznamená se poslední konfigurace před opuštěním stránky, ale také typicky stav nákupního košíku v elektronickém obchodě. Po opuštění stránky a po opětovném návratu si bude stránka pamatovat dříve vybrané položky a uživatel má možnost pokračovat v nákupu tam, kde naposled skončil. Sobory cookies lze upravovat pomocí funkce nastavení v každém běžném prohlížeči. Obvykle těch položek pro nastavení bývá několik a každá relace se nastavuje zvlášť.[21]

Jsou to malé textové soubory, které jsou uloženy v adresáři prohlížeče nebo podsložkách programu. Cookies se vytváří automaticky za používání prohlížeče při návštěvě webových stránek, které používají cookies pro sledování pohybu uživatelů v rámci svého webu. Tím je uživateli usnadněný pohyb po stránkách, výběr motivů, načtení předvolby a další uzpůsobení. Soubor je uložen v počítači a ukládají se do něj informace, které uživatel dobrovolně poskytl jako je e-mailová adresa apod.

Cookies soubory jsou nepostradatelné pro webové stránky, které mají obrovské databáze, potřebují přihlášení, mají přizpůsobitelná data a další pokročilé funkce. Neobsahují příliš mnoho informací, pouze adresu URL webové stránky, která jej vytvořila, trvání možností a efektů a náhodného čísla. Vzhledem k malému množství údajů, které obsahují, nelze tento soubor použít k odhalení totožnosti uživatele ani osobní identifikační údaje. Mohu však být při agresivním využití použity k vytvoření profilu návyků uživatele v oblasti surfování po webu.

Existují dva typy cookies. První soubory cookies relace jsou dočasně vytvářeny v podsložce prohlížeče během návštěvy stránek. Po opuštění webu se soubor relace zcela smaže. Druhé trvalé soubory cookies zůstávají uloženy ve složce prohlížeče a po návštěvě webu, která daný soubor vytvořil, se znovu aktivuje.[19]

Soubor cookie je malý textový soubor stažený do počítače při přístupu na určitou webovou stránku. Pomocí tohoto souboru lze odemknout paměť počítače a dovolit rozpoznat webovým stránkám uživatele, pokud se vrátí na stránky přes jinou službu. Samotný soubor neobsahuje informace, ale umožňuje po načtení počítačem zlepšit poskytované služby. V souboru najedeme název serveru, ze kterého byl soubor cookie odeslán, životnost cookie, hodnotu, obvykle náhodně vygenerované jedinečné číslo. Server pomocí tohoto čísla uživatele rozpozná po návratu na stránku a také by ho měl používat právě jen on. Soubory cookie samy o sobě nelze využít k identifikaci.

Používaný prohlížeč Chrome, ukládá soubory tohoto typu do c://user/"uživatel"/AppData/Local/Google/Chrome/UserData.

```
SQLite format 3     @ .,P □  A  g ?A 8t
□?tablecookiescookies CREATE TABLE cookies (creation_utc INTEGER NOT NULL,
host_key TEXT NOT NULL,name TEXT NOT NULL,value TEXT NOT NULL,
path TEXT NOT NULL, expires_utc INTEGER NOT NULL,is_secure INTEGER NOT NULL,
is_httponly INTEGER NOT NULL, last_access_utc INTEGER NOT NULL,
 has_expires INTEGER NOT NULL DEFAULT 1, is_persistent INTEGER NOT NULL DEFAULT 1,
priority INTEGER NOT NULL DEFAULT 1,
encrypted_value BLOB DEFAULT '',firstpartyonly INTEGER NOT NULL DEFAULT 0,
UNIQUE (host_key, name, path))-   A   indexsqlite_autoindex_cookies_1cookies f
□/tablemetameta CREATE TABLE meta(key LONGVARCHAR NOT NULL UNIQUE PRIMARY KEY,
value LONGVARCHAR)'   ;
indexsqlite_autoindex_meta_1meta            ?  ???
last_compatible_version10
version10   # mmap_status-1
   ?  ???
```

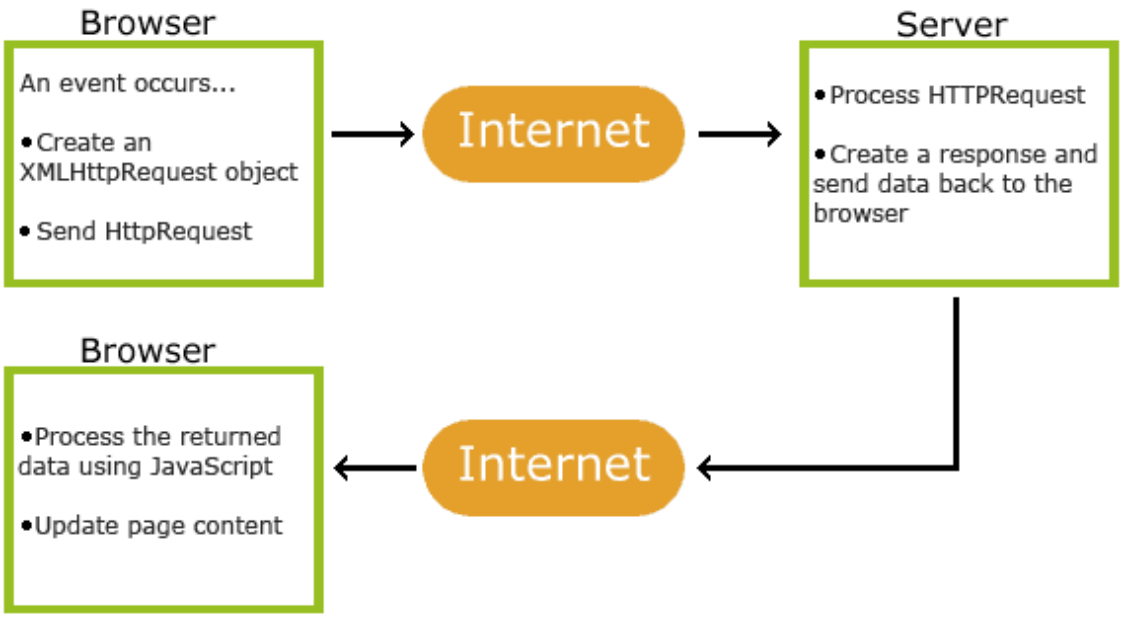
Obr. 22 – Výpis souboru Cookie - zdroj: vlastní

10 AJAX, JQUERY

Ajax je technologie umožňující posílat data na server a také je ze serveru přijímat bez nutnosti opětovného načítání celé stránky jako je tomu u klasického řešení. JQuery nabízí řadu funkcí zjednodušujících tuto technologii.[27] Čistý kód javascriptu je velmi rozsáhlý, za pomoci jQuery není potřeba psát tak dlouhý kód, ale stačí napsat to stejné pomocí několika funkcí.

10.1 Ajax

AJAX je zkratka, která znamená **A -** asynchronní **J -** javascript **A -** and **X -** XML. Uvádí se, že není programovací jazyk v pravém slova smyslu, ale funguje jako nadstavba javascriptu obohacená o xml. umožňuje aktualizovat webové stránky asynchronně výměnou dat s webovým serverem v zákulisí. To znamená, že je možné aktualizovat části webové stránky, aniž byste museli znovu načítat celou stránku [29].



Obr. 23 – Princip fungování AJAXu – zdroj [23]

Princip fungování je následující:[23]

- Na stránce se vyskytne událost, např. klik tlačítka
- Javasript vytvoří XMLHttpRequest objekt
- XMLHttpRequest objekt odešle požadavek na server

- Server požadavek zpracuje
- Server odešle odpověď zpět na webovou stránku
- Odpověď je přečtena javascriptem
- Výsledek požadavku se obrazí na stránce

AJAX používá pouze kombinaci vestavěného XMLHttpRequest objektu, který slouží pro vyžádání dat z webového serveru, JavaScriptu a HTML DOM (doucument object model). Ajaxové aplikace mohou pro přenos dat používat XML, běžný text nebo JSON.[26]

Z bezpečnostních důvodů neumožňují moderní prohlížeče přístup přes domény. To znamená, že webová stránka i soubor XML, který se pokusí načíst, musí být umístěny na stejném serveru.

Nette Framework podporuje všechnu Ajaxovou komunikaci. Nenutí k používání žádnou konkrétní javascriptovou knihovnu, takže je možné napsat vlastní obslužný skript. K tomu lze využít např. jQuery. Framework podporuje a velmi zjednodušuje zasílání fragmentů HTML. A vede programátora k tvorbě aplikací, které plně fungují i bez JavaScriptu.

10.2 Jednoduchá AJAX aplikace

V této podkapitole je jednoduchá ukázka schovávání a znovu zobrazení části stránky, konkrétně určitého nadpisu pomocí události click na jednotlivé nadpisy ve stránce. Jedná se o ajax s podporou jQuery. Kliknutím na nadpis druhé úrovně se tento pomalu schová a následným kliknutím na nadpis první úrovně se zase zobrazí.

Obr. 24 – Schování nadpisu - zdroj: vlastní

Přímo do hlavičky hlavní šablony *@layout*. Latte umístíme následující kód. Ten je tak krátký a jednoduchý, že není nutné jej umísťovat do zvláštního souboru.

```
<script>
    $(document).ready(function(){
      $("h2").click(function(){
        $("h2").hide(1000);
      });
     $("h1").click(function(){
       $("h2").show(1000);
      });
   });
 </script>
```

Obr. 25 – Zobrazení nadpisu - zdroj: vlastní

$ je znak reprezentující funkci, minimalizuje se tím psaní kódu. Než se spustí nějaký javascriptový kód, mělo by proběhnout načtení html. K tomu slouží událost Document Ready. Funkce předaná jako parametr metodě ready() se spustí po načtení stránky.

Selektor `$("h2")` vybere element, na který se bude aplikovat událost `click`. V tomto konkrétním případě je to nadpis druhé úrovně. V události click je nejprve uvedena komponenta, které se událost týká, tedy opět `$("h2")` nadpis na který jsme klikli a na něj se aplikuje činnost, kterou chceme vykonat, tj., `hide(1000)`. Ta vybraný prvek na stránce

schová. Dále máme ve funkci metody ready() ještě jednu událost click, která funguje naprosto shodně jenom se kliknutí bude provádět na jiný nadpis, tentokrát první úrovně a na schovaný nadpis se aplikuje `show(1000)`. Ta opět vrátí nadpis na stránku. Čím vyšší číslo, tím pomaleji proces schovávání a zobrazení proběhne.

Ještě je potřeba připomenout, že je nutné v záhlaví před samotným skriptem samotnou vložit knihovnu jQuery.

```
<script src="https://ajax.googleapis.com/ajax/
                    libs/jquery/3.3.1/jquery.min.js"></script>
```

10.3 jQuery

jQuery je knihovna JavaScriptu, značně zjednodušuje programování JavaScriptu a snadno se učí. Účelem jQuery je usnadnit používání JavaScriptu na webových stránkách. Pro práci s jQuery se předpokládá základní znalost HTML, CSS a JavaScriptu.[25] Mnoho velkých společností působících na poli webu používá jQuery, například:

- Google
- Microsoft
- IBM
- Netflix

JQuery řeší spoustu běžných úkolů a činností, které vyžadují mnoho řádků kódu JavaScriptu. Toho se dosáhne zabalením do metod, které můžete volat jedním řádkem kódu. jQuery také zjednodušuje spoustu komplikovaných věcí z JavaScriptu, jako jsou AJAX volání a DOM (document object model) manipulace.[18]

Knihovna jQuery obsahuje následující funkce:[22]

- Manipulace s HTML / DOM
- CSS manipulace
- Metody události HTML
- Efekty a animace
- AJAX
- Utility

Kromě toho má jQuery pluginy pro téměř jakýkoliv úkol. Přidání jQuery na stránky lze následujícím způsobem buď od společnosti Google nebo Microsoft.

```
<head>
<script src="https://ajax.googleapis.com/
ajax/libs/jquery/3.3.1/jquery.min.js"></script>
</head>
```

nebo

```
<head>
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/
jquery-3.3.1.min.js"></script>
</head>
```

Také je možné si soubor stáhnout a umístit na vhodné místo do projektu a vložit

```
<head>
<script src="jquery-3.3.1.min.js"></script>
</head>
```

10.4 jQuery – ukázka vyhledávání v tabulce

Pro ukázku funkce vyhledávání v tabulce vytvoříme zvláštní PHP soubor, který bude obsahovat jak strukturu html dokumentu, vložení všech potřebných knihoven, tak i samotný skript pro vyhledávání. Na stránce bude okénko pro zadávání vyhledávaného výrazu a samotná tabulka s daty. Po zadání výrazu po písmenech do okna pro vyhledávání se budou v tabulce zobrazovat jen údaje odpovídající zadání.[24]

Samotný ukázkový skript bude obsahovat strukturu html kódu, tedy hlavičku a tělo.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
</body>
</html>
```

Tělo stránky obsahuje dvě části. V první částí je nadpis, informace pro uživatele a vstupní okno, jakýsi textbox, pro zadávání hledaného výrazu. Textbox má identifikátor *id="myInput",* pracujeme s ním, jako s textem a také je nastaven výchozí text.

```
<h2>Filtr pro vyhledávání</h2>
<p>Do pole pro zadání zadejte vyhledávaný výraz, který hledáte v tabulce - jména, příjmení nebo e-maily:</p>
<input id="myInput" type="text" placeholder="Vyhledávání..">
```

Ve druhé části těla stránky se nachází klasická html tabulka, která má tělo tabulky uzavřené do párové značky <tbody> a opatřené identifikátorem *id="myTable"*, který bude využitý při programování jQuery kódu. Potřebujeme, aby záhlaví tabulky zůstalo zobrazeno stále bez ohledu na nalezené množství obsahu.

```
<table>
  <thead>
  <tr>
    <th>Jméno</th>
    <th>Příjmení</th>
    <th>Email</th>
  </tr>
  </thead>
  <tbody id="myTable">
  <tr>
    <td>John</td>
    <td>Doe</td>
    <td>john@example.com</td>
  </tr>
  …… <!-- další část velmi dlouhé tabulky -->
<tr>
    <td>Igor</td>
    <td>Randal</td>
    <td>iRA@gmail.com</td>
```

```
  </tr>
  </tbody>
</table>
```

Do záhlaví je ještě potřeba vložit knihovnu samotného jQuery

```
<script src="https://ajax.googleapis.com/ajax/
                libs/jquery/3.3.1/jquery.min.js"></script>
```

Také si vyrobíme soubor s formátováním vizuální stránky webu. Ten vložíme následujícím způsobem opět do záhlaví. Do tohoto souboru se umístí skript kaskádových stylů pro nastavení zobrazení.

```
<link rel="stylesheet" href="jQ08b.css">
```

Dále je potřeba vytvořit samotný javascriptový soubor, do kterého bude umístěn kód vyhledávání v tabulce. Základem je přečtení dokumentu a teprve po té aplikace skriptu.

```
$(document).ready(function(){
});
```

Do základní funkce vložíme další funkci, která vybere selektorem prvek s id myInput, to je textové pole pro vkládání vyhledávaného výrazu, a znak po znaku načte, převede na malá písmena a vrátí do proměnné value.

```
  $("#myInput").on("keyup", function() {
    var value = $(this).val().toLowerCase();
  });
```

Součástí načítací funkce je další zanořená funkce, která převezme hodnotu proměnné value. Její selektor vybere prvek s id myTable obohacený o tag tr což znamená, že bude pracovat s tabulkou po řádcích. Metoda indexOf() vrátí index prvku, ve kterém je obsažen převedený řetězec, při neexistenci řetězce vrátí hodnotu -1, metodou text() se vrátí samotný řetězec, metoda toggle() přepíná mezi skrýváním a zobrazením.

```
$("#myTable tr").filter(function() {
  $(this).toggle($(this).text().toLowerCase().indexOf
                                                (value) > -1)
});
```

Nyní máme vytvořený ukázkový příklad aplikace jQuery tabulky a vyhledávání v ní. Tento příklad sestává ze tří souborů. První soubor s názvem jQuery08_b.php (může být i *.html) obsahuje strukturu stránky sestavenou v html kódu s tabulkou, ve které se bude vyhledávat. Druhý soubor s názvem jQ08b.css nese informace o použitých kaskádových stylech, resp. o vizuální podobě stránky. Ve třetím souboru s názvem jQ08b.js je samotný jQuery kód. V tomto ukázkovém příkladu jsou všechny soubory umístěny ve stejné složce. Při aplikaci do konkrétního projektu by se musely umístit podle požadavků projektu a také by se musely upravit cesty při zavedení souborů.

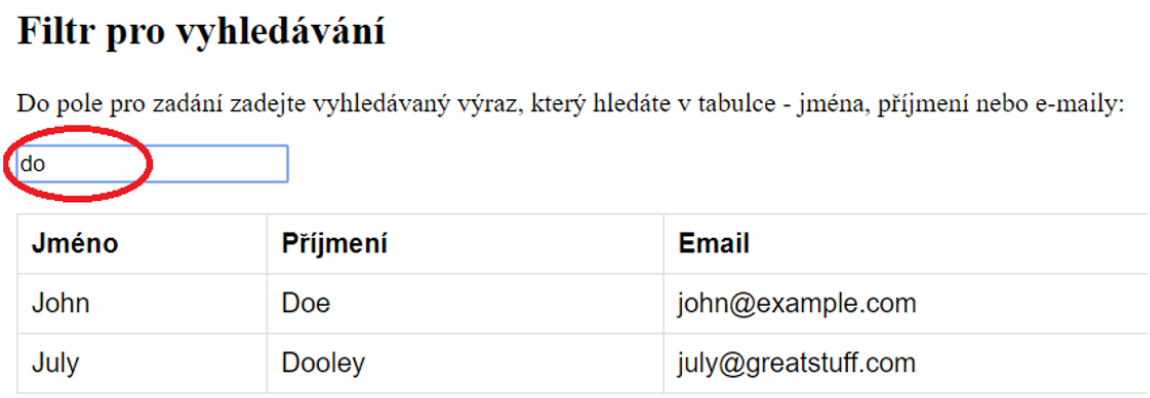
Filtr pro vyhledávání

Do pole pro zadání zadejte vyhledávaný výraz, který hledáte v tabulce - jména, příjmení nebo e-maily:

Vyhledávání..

Jméno	Příjmení	Email
John	Doe	john@example.com
Mary	Moe	mary@mail.com
July	Dooley	july@greatstuff.com
Anja	Ravendale	a_r@test.com
Claudia	Emental	EmCla@testing.com
Igor	Randal	iRA@gmail.com

Obr. 26 – Obsah celé tabulky - zdroj: vlastní



Obr. 27 – Zobrazení nalezeného obsahu - zdroj: vlastní

10.5 jQuery – stránkování tabulky

Často bývá požadavek, aby se zobrazovala jen určitá část dat z dané tabulky. Jedním z hlavních důvodů tohoto požadavku je nepřehlednost při zobrazování velkého množství dat. Tabulka, která má řádově desetitisíce položek bude prakticky nepoužitelná, pokud bychom ji zobrazili celou. Proto je vhodné data v tabulce zobrazovat přehledně po menších částech, tedy stránkách. Zobrazování může být doplněno o další nástroje jako je již zmíněné vyhledávání. Například pokud by se z databáze o padesáti tisících záznamech mělo vybrat jen pět set položek podle hledaného výrazu a ještě je doplnit stránkováním, určitě by manipulace s daty byla výrazně přívětivější a přehlednější.[24]

Obr. 28 – Tabulka se stránkováním obsahu - zdroj: [24]

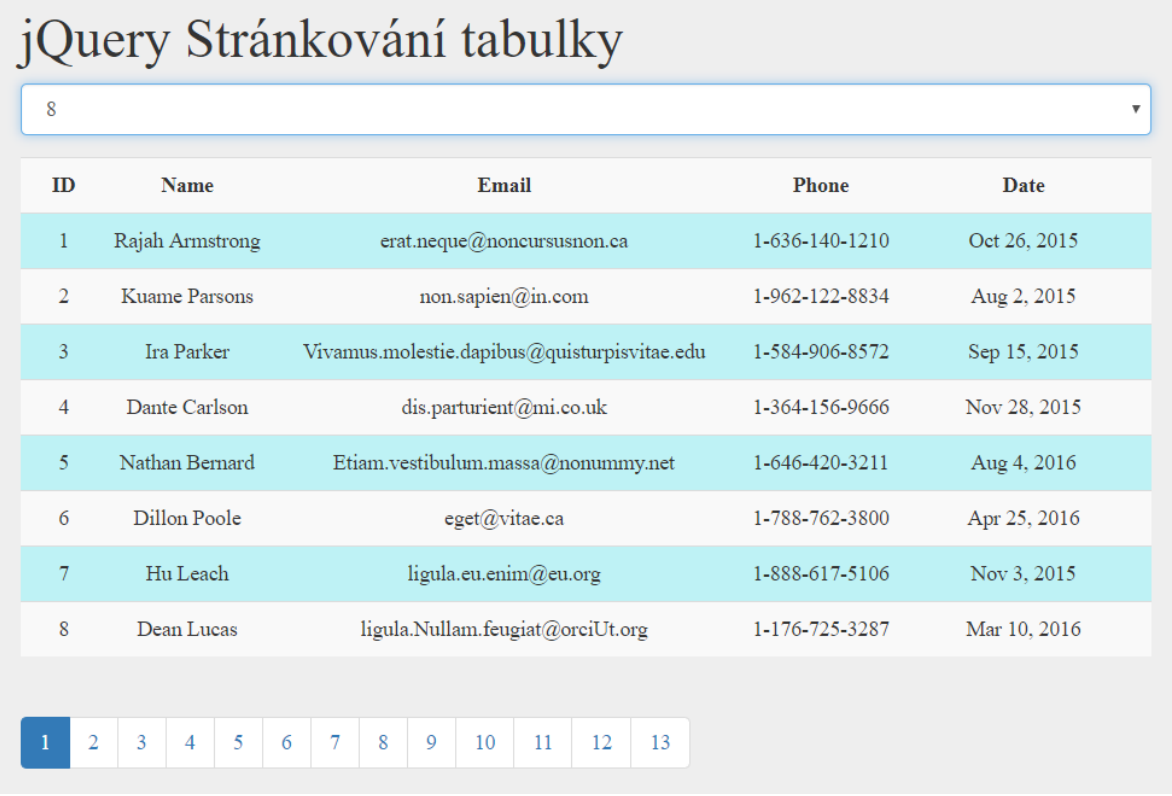
Tabulku z obrázku 28 je možné bezplatně získat z www.jqueryscript.net/table[24] pod licencí GNU. Tabulka se stránkováním obsahu využívá jQuery a je tvořena soborem index.html a skripty pagging.js a style.css. HTML soubor obsahuje kromě klasické html tabulky odkazy na knihovní soubory jquery.min.js a jquery-ui.min.js, které se načítají přímo z webu, ale je možné je i stáhnout a přidat do projektu, pak je zde jednoduchá jquery funkce pro stanovení počtu zobrazovaných položek na stránce

```
$(document).ready(function() {
                $('#tableData').paging({limit:4});
        });
```

Vše ostatní obstarává soubor skriptů pagging.js. V něm jsou obsaženy funkce pro samotné stránkování, vytvoření odkazu na předchozí či následující stránku.

10.6 jQuery – volba počtu položek na stránku

Následující ukázka je opět dostupná na www.jqueryscript.net/table pod licencí GNU. Zde je spojeno stránkování s vyhledáváním.[24]



Obr. 29 – Ukázka stránkování a volby počtu položek - zdroj: [24]

Tento ukázkový příklad je z hlediska samotného jQuery programování poměrně složitý a jQuery by samo o sobě vydalo na další samostatnou práci. Obsah tabulky je vložen do souboru index.html. Ten by šel realizovat v Nette pomocí presenteru, který získává data z tabulky a šablony, která je přes cyklus foreach zobrazuje uživateli. Funkcionalita stránkování jQuery je v souboru js.js. To je jQuery skript obsahující dvě hlavní funkce. První je stránkovací funkce *function getPagination (table){}*, která provádí výpočty stránkování a zobrazení požadovaného počtu položek a druhá jednodušší funkce, která přiřazuje každému řádku svoje ID. Samotný princip stránkování je následující. Využití zde mají dvě proměnné, do první se vloží počet požadovaných položek na stránce. Ten může být fixní, nebo se jeho hodnota může získat ze selektoru na stránce. Do druhé proměnné se vloží celkový počet položek v tabulce. Výpočet počtu stránek je již sám o sobě jednoduchý, provede se proces dělení celkového počtu záznamů s požadovaným počtem na stránku a výsledek stanoví počet stránek jQuery tabulky. Celou ukázku doplňuje soubor style.css, který nese informace o vizuální podobě stránky.

10.7 Implementace stránkování tabulky v Nette

Pro implementaci tabulky v Nette byl založen nový projekt stejným způsobem jako v kapitole 3, se stejnou strukturou projektu, tzn., v databázi posts byla vytvořena tabulka uživatelů, sloužící jako zdroj dat, dále soubory

config.neon – přístup k databázi,

```
database:
     dsn: 'mysql:host=127.0.0.1;dbname=posts'
     user: root
     password: *heslo*
```

HomePresenter.php tvoří funkční základ, obsahuje konstruktor přístupu k databázi a metodu získání dat podle id,

```
public function renderDefault()
{
$this->template->zakaznici = $this->database->table
                                                ('zakaznici')
```

```
                ->order('id ASC');
    }
```

Hlavní šablona *layout.latte* obsahuje ve svém těle načtení content blok pod šablony default.latte

```
{include content}
```

šablona default.latte vytváří ve svém content bloku vlastní tabulku jejíž fixní částí je blok tabulky a její záhlaví

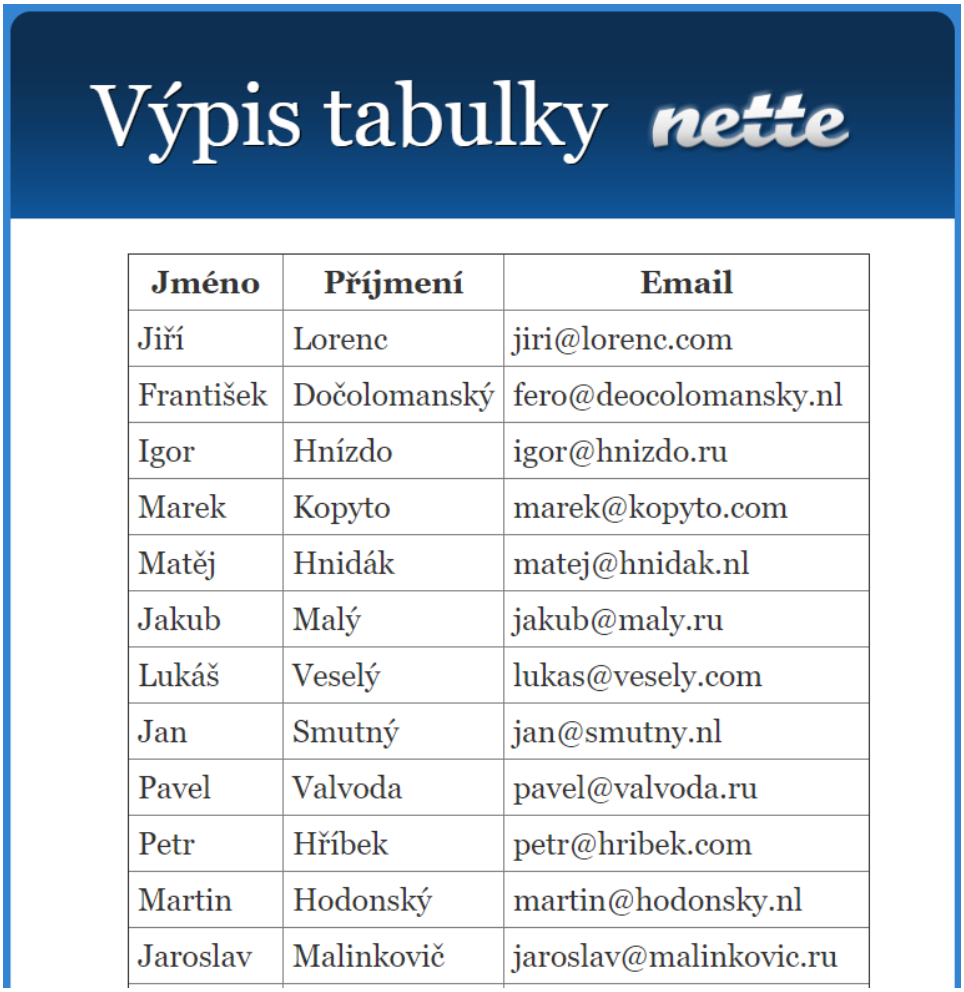
```
<table id="myTable">
  <thead>
    <tr>
      <th>Jméno</th>
      <th>Příjmení</th>
      <th>Email</th>
    </tr>
    </thead>

      ……..
  </table>
```

a tělo tabulky, které pomocí cyklu foreach převezme obsah získaný z databáze pomocí proměnné zakaznik a zobrazí je do tabulky

```
<tbody>
<div n:foreach="$zakaznici as $zakaznik" class="zakaznik">
         <tr>
           <td>{$zakaznik->jmeno}</td>
           <td>{$zakaznik->prijmeni}</td>
           <td>{$zakaznik->mail}</td>
```

```
            <tr>
        </div>
</tbody>
```



Obr. 30 – tabulka v Nette - zdroj: vlastní

Takto vytvořenou tabulku je ještě potřeba doplnit o stránkování neboť tabulky v reálném provozu mohou disponovat velkým množstvím dat a jejich zobrazení by pak bylo chaotické. Projekt je tedy doplněn o ArticleManager.php realizující model projektu, obsahuje konstruktor přístupu k databázi a metodu získání dat z tabulky, řazené podle id,

```
public function getPublicArticles()
    {
```

```
        return $this->database->table('zakaznici')
                    ->order('id');
    }
```

šablona default.latte je doplněna o zobrazení stránkování, to je řízeno proměnnými page a lastPage jimž hodnoty dodává aritmetika presenteru

```
<div class="pagination" align="center">
    {if $page > 1}
        <a n:href="default, 1">První</a>
         | 
        <a n:href="default, $page-1">Předchozí</a>
         | 
    {/if}
    Stránka {$page} z {$lastPage}
    {if $page < $lastPage}
         | 
        <a n:href="default, $page+1">Další</a>
         | 
        <a n:href="default, $lastPage">Poslední</a>
    {/if}
</div>
```

A metoda renderDefault () souboru HomepagePresenter.php je upravena tak, aby dodávala data v požadovaném množství.

```
public function renderDefault($page = 1)
{
    $zakaznici = $this->articleManager->getPublicArticles();
    $lastPage = 0;
    $this->template->zakaznici = $zakaznici->page($page, 5,
```

```
                                                    $lastPage);
        $this->template->page = $page;
        $this->template->lastPage = $lastPage;
    }
```

Do těla cyklu foreach šablony default.latte byla přidána položka id, která je zobrazena jako pořadí jednotlivých záznamů

```
{$zakaznik->id}
```

Tímto způsobem se v tabulce zobrazí přijatelné množství dat, které je možno jednoduše stránkovat neboť samotné stránkování zobrazuje číslo aktuální stránky z celkového počtu stránek.

Obr. 31 – stránkovaná tabulka v Nette - zdroj: vlastní

ZÁVĚR

V této diplomové práci byl vytvořen tutoriál pro použití PHP frameworku Nette. Tutoriál je určen pro začátečníky, kteří již mají základy skriptovacího jazyka PHP, HTML, JavaScriptu a CSS, tedy pro uživatele, kteří již chápou principy objektového programování.

Také je zde popsán vývoj webové aplikace na lokálním serveru a důležité prvky konfigurace a ověření jejich nastavení. Tím, že je aplikován model MVC nabízí framework řadu možností a funkcí, jak vzhledově, tak i funkčně. Celý návrh se tím rozdělí do samostatných částí, které je možno opakovaně aplikovat i v dalších projektech a které je možno samostatně v návaznosti na sobě zpracovávat.

Část práce byla otestována na skupině středoškoláků čtvrtého maturitního ročníku a to konkrétně – část úvodní stránka a její stránkování. Skupina sestávala z deseti studentů čtvrtého maturitního ročníku. Výsledkem bylo, že většina studentů byla schopna vytvořit pod vedením vyučujícího zkoušenou část projektu. Tutoriál bude ještě potřeba otestovat na větším počtu studentů a dále jej upravit a rozšířit. Ukazuje se, že před aplikací tohoto tutoriálu bude nutné u studentů prohloubit znalosti z jednotlivých dílčích oblastí, konkrétně práce s HTML, tvorby databází, objektového programování a kaskádových stylů. Efektivita tutoriálu také závisí na kvalitě vědomostí a znalostí studentů, kteří ho hodlají používat. Celkově by tento tutoriál mohl být do budoucna rozšířen o práci s různými druhy souborů, o práci s mailem nebo o práci se souborovým systémem.

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] Quick ‘n‘ Comfortable Web Development in PHP |Nette Framework. Quick ‘n‘ Comfortable Web Development in PHP | Nette Framework [online]. Copyright © 2008, 2019 Nette Foundation. All rights reserved [cit. 05.05.2019]. Dostupné z: https://nette.org/

[4] Home –Zend Framework. *Home – Zend Framework* [online]. Copyright © 2006 [cit. 05.05.2019]. Dostupné z https://framework.zend.com/

[5] Symfony, High Performance PHP Framework for Web Development. *Symfony, High Performance PHP Framework for Web Development* [online] Copyright © Copyright 2005 [cit. 05.05.2019]. Dostupné z: https://symfony.com/

[6] Ruby on Rails | A web-application framework that includes everything needed to database-backed web applications according to the Model-View-Controller (MVC) pattern. *Ruby on Rails | A web-application framework that includes everything needed to database-backed web applications according to the Model-View-Controller (MVC) pattern*.[online] Copyright © Copyright 2005 [cit. 05.05.2019]. Dostupné z https://rubyonrails.org/

[7] [online]. Dostupné z https://www.cakephp.org/

[8] CodeIgniter Web Framework. *CodeIgniter Web Framework* [online]. Copyright © Copyright 2005 [cit. 05.05.2019].Dostupné z: https://codeigniter.com

[9] PRADO PHP Framework. *PRADO PHP Framework* [online]. Copyright © Copyright 2004 [cit. 05.05.2019]. Dostupné z: http://www.pradoframework.net/site/

[10] [online]. Dostupné z https://jelix.org

[11] Laravel – The PHP Framework For Web Artisans. *Laravel – The PHP Framework For Web Artisans* [online]. Copyright © Taylor Otwell [cit. 05.05.2019]. Dostupné z: https://larevel.com/

[12] GILMORE, W. J. *Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály*. Přeložil Jan POKORNÝ. Brno: Zoner Press, 2005. Encyklopedie webdesignera. ISBN 80-86815-20-x.

[13] LACKO, Ľuboslav. *PHP a MySQL: hotová řešení*. Brno: CP Books, 2005. K okamžitému použití (CP Books). ISBN 80-251-0397-8.

[14] Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework. [online]. Copyright © 2008, 2019 Nette Foundation. All rights reserved. [cit. 23.04.2019]. Dostupné z: https://nette.org/cs/

[15] Latte | Nette Framework. [online]. Copyright © 2008, 2019 Nette Foundation. All rights reserved. [cit. 23.04.2019]. Dostupné z: https://latte.nette.org/cs/

[16] PHP: Sessions - Manual . PHP: Hypertext Preprocessor [online]. Copyright © 2001 [cit. 23.04.2019].

Dostupné z: https://www.php.net/manual/en/book.session.php

[17] Sessions. *Péhápko: Učebnice PHP* [online]. Dostupné z: http://www.pehapko.cz/programujeme-v-php/sessions

[18] jQuery API Documentation. jQuery API Documentation [online]. Dostupné z: https://api.jquery.com/

[19] Cookies and Sessions. akela.mendelu.cz [online]. Dostupné z: https://akela.mendelu.cz/~lysek/tmwa/articles/cookies-sessions/

[20] University information systém Mendelu. *University information systém Mendelu* [online]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=21331

[21] W3Schools Online Web Tutorials. W3Schools Online Web Tutorials [online]. Dostupné z: https://www.w3schools.com/

[22] Základy jQuery. itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. Copyright © 2019 itnetwork.cz. Veškerý obsah webu [cit. 23.04.2019]. Dostupné z: https://www.itnetwork.cz/javascript/jquery-zaklady

[23] MySQL. MySQL [online]. Copyright © 2019, Oracle Corporation and [cit. 23.04.2019]. Dostupné z: https://www.mysql.com/

[24] jQuery Table Plugins. jQuery Table Plugins [online]. Copyright © 2007 [cit. 23.04.2019]. Dostupné z: http://www.jqueryscript.net/table

[25] jQuery table pagination Plugins | jQuery Script. Free jQuery Plugins and Tutorials - jQuery Script [online]. Copyright © Copyright 2012 [cit. 23.04.2019]. Dostupné z: https://www.jqueryscript.net/tags.php?/table%20pagination/

[26] Nette Framework: AJAX - Zdroják. Zdroják - o tvorbě webových stránek a aplikací [online]. Dostupné z: https://www.zdrojak.cz/clanky/nette-framework-ajax/

[27] jQuery Ajax. Java NIO, PyTorch, SLF4J, Parallax Scrolling, Java Cryptography, YAML, Python Data Science, Java i18n, GitLab, TestRail, VersionOne, DBUtils, Common CLI, Seaborn, Ansible, LOLCODE, Current Affairs 2018, Apache Commons Collections[online]. Copyright © Copyright 2019. All Rights Reserved. [cit. 23.04.2019]. Dostupné z: https://www.tutorialspoint.com/jquery/jquery-ajax.htm

[28] NCKB. NCKB [online]. Dostupné z: https://www.govcert.cz

[29] AJAX Introduction. *W3Schools Online Web Tutorials* [online]. Dostupné z: https://www.w3schools.com/js/js_ajax_intro.asp

[30] Legislativa – Ochrana osobních údajů. *Úvodní strana – Ministerstvo vnitra České republiky* [online]. Copyright © 2019 Ministerstvo vnitra České republiky. Všechna práva vyhrazena. [cit. 10.05.2019]. Dostupné z: https://www.mvcr.cz/gdpr/clanek/gdpr-web.legislativa-legislativa.aspx

[31] Právní regulace cookies v České republice | eprav.cz. EPRAVO.CZ – *Váš průvodce právem – Sbírka zákonů, judikatura, právo* [online]. Copyright © EPRAVO.CZ a.s. 1999 [cit 10.05.2019]. Dostupné z: https://www.epravo/top/clanky/pravni-regulace-cookies-v-ceske-republice-98406.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PHP	Personal Home Page – skriptovací jazyk pro vývoj dynamického webu
HTML	HyperText Markup Language – značkovací jazyk pro tvorbu stránek
MVC	Model-View-Controller, architektura webových aplikací
AJAX	Asynchronous JavaScript and XML, technologie pro vývoj aplikací, díky níž lze aktualizovat obsah stránky bez jejího znovu načtení
DRY	Don't Repeat Yoursoft, vývoj bez duplicitních kódů
KISS	Keep ItSimple Stupid, efektivní návrh řešení v souvislosti s okolím
MySQL	Databázový dotazovací jazyk
XML	eXtensible Markup Language, rozšiřitelný značkovací jazyk
GNU GPL	Všeobecná veřejná licence
MIT	Volná licence, podmínkou je uvedení této licence
JSON	JavaScript Object Notation, styl zápisu javascriptového objektu, není závislý na platformě
WAMP	Windows, Apache, MySQL, PHP – lokální server pro vývoj webových aplikací
HTTP	Hypertext Transfer Protokol, webový protokol na principu hyperlinků
RSS	Kanál RSS (Ready, Simple, Syndication), formát pro čtení novinek na webových stránkách.
DOM	Document Object Model.

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁZEV PŘÍLOHY

Obsah přiloženého CD:

- Diplomová práce v elektronické podobě
- Zdrojové kódy vlastní aplikace
- Zdrojové kódy vlastní Nette tabulky
- Zdrojové kódy jQuery tabulek