TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Návrh a realizace backendu webového portálu pro základní školu

Diplomová práce

Studijní program: N6209 – Systémové inženýrství a informatika
Studijní obor: 6209T021 – Manažerská informatika

Autor práce: **Bc. Martin Letáček**
Vedoucí práce: Ing. David Kubát, Ph.D., Ing.Paed.IGIP

Liberec 2018

TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

Akademický rok 2017 / 2018

Zadání diplomové práce

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Martin Letáček**
Osobní číslo:	E15000551
Studijní program:	N6209 Systémové inženýrství a informatika
Studijní obor:	N6209T021 – Manažerská informatika
Zadávající katedra:	katedra informatiky
Vedoucí práce:	Ing. David Kubát, Ph.D., ING.PAED.IGIP
Konzultant práce:	Ing. Michal Bohuslávek Dubax s. r. o., programátor

Název práce: **Návrh a realizace backendu webového portálu pro základní školu**

Zásady pro vypracování:

1. Možnosti nových backendových technologií.
2. Návrh datové struktury portálu.
3. Návrh a tvorba databáze.
4. Realizace a testování funkčnosti.
5. Zhodnocení funkcionality a použitých postupů.

Seznam odborné literatury:

MELONI, Julie C. 2017. *PHP, MySQL & JavaScript All in One, Sams Teach Yourself.* Indianapolis: Sams Publishing. ISBN 978-0672337703.

ZANDSTRA, Matt. 2016. *PHP Objects, Patterns, and Practice.* Barkley: aPress. ISBN 978-1484219959.

ULLMAN, Larry. 2016. *Php and mysql for dynamic web sites: visual quickpro guide.* Barkley: Peachpit Press. ISBN 978-0134301846.

DUBOIS, Paul. 2014. *Mysql Cookbook: Solutions for Database Developers and Administrators.* Sebastopol: Oreilly & Associates Inc. ISBN 978-1449374020.

PROQUEST. 2017. *Databáze článků ProQuest* [online]. Ann Arbor, MI, USA: ProQuest. [cit. 2017-09-28]. Dostupné z: http://knihovna.tul.cz/

Rozsah práce:	65 normostran
Forma zpracování:	tištěná / elektronická
Datum zadání práce:	31. října 2017
Datum odevzdání práce:	31. srpna 2019

Technická univerzita v Liberci
Ekonomická fakulta

L.S.

prof. Ing. Miroslav Žižka, Ph.D.
děkan Ekonomické fakulty

doc. Ing. Klára Antlová, Ph.D.
vedoucí katedry

V Liberci dne 31. října 2017

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Tímto bych rád poděkoval vedoucímu mé diplomové práce Ing. Davidu Kubátovi, Ph.D., Ing. Paed. IGIP za vedení této práce a také za poskytnuté rady a připomínky. Děkuji i kolegům ze společnosti Dubax s.r.o., kteří mi poskytli odbornou spolupráci při realizaci celého projektu. Mé poděkování patří také rodině a přátelům, kteří mě během studia podporovali a motivovali.

Anotace

Hlavním tématem této diplomové práce je backend webových stránek a jeho vývoj. V teoretické části této práce jsou představeny dílčí vrstvy webových stránek a dostupná technologická řešení k jejich realizaci. Praktická část této práce již popisuje průběh realizace konkrétního webového portálu pro Základní a mateřskou školu ve Stráži pod Ralskem. Navazuje přitom bezprostředně na témata otevřená v první části této práce a to volbou technologické platformy pro datovou a aplikační vrstvu webových stránek. Další část je věnována návrhu a realizaci těchto vrstev. Nechybí zde ani detailní popis datové vrstvy a ukázky zdrojového kódu. V závěrečné části je shrnuta výsledná funkcionalita administračního prostředí webových stránek a průběh testování.

Klíčová slova

Webový portál, webová stránka, backend, PHP, Nette, databáze, MySQL

Annotation

DESIGN AND IMPLEMENTATION OF A WEB PORTAL BACKEND FOR AN ELEMENTARY SCHOOL

Main topic of this diploma thesis is a backend of websites and its development. Theoretical part presents website layers and available technological platforms for their implementation. Second part of this thesis describes the process of design and implementation of a web portal backend for an elementary school in Stráž pod Ralskem. It begins with a choice of technological platform for data and application layer of this website. Following part shows details of the development processes of both layers. Finally there is a summary of all features included in administration system that we created followed by a description of testing process.

Key Words

Web portal, website, backend, PHP, Nette, database, MySQL

Obsah

Seznam zkratek

API – Application Programming Interface

CMS – Content Management System

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

IoT – Internet of Things

MVC – Model, View, Controller

OOP – Objektově Orientované Programování

PHP – Hypertext Preprocessor

WYSIWYG – What you see is what you get

Seznam obrázků

Úvod

V současné době jsou webové portály nezbytným komunikačním prostředkem mezi organizací a veřejností. Provozovatelem takovéhoto portálu mohou být například podnikatelské subjekty, jež usilují o oslovení potenciálního zákazníka. Využití webových portálů je však daleko širší. Dalšími typickými představiteli vlastníka webového portálu jsou také neziskové či příspěvkové organizace. Konkrétně můžeme jmenovat například školy, školky, dětské domovy apod. Pro tyto organizace představují webové portály prakticky jediný informační kanál s veřejností, snad jen vyjma osobního kontaktu. Mít svůj vlastní webový portál či webové stránky se tak pro mnoho organizací stalo nejen samozřejmostí, ale také nutností.

Tato práce je zaměřena zejména na backendovou část webových stránek, která je za běžných okolností uživatelům skryta. Pro čtenáře tak představuje jednu z možností, jak nahlédnout do tvorby webových aplikací. Hlavními tématy teoretické části této práce jsou současné technologické trendy a postupy při vývoji dynamických webových stránek. Pozornost je věnována také doprovodným nástrojům užívaným při vývoji.

Cílem této práce je realizace zcela nového webového portálu pro Základní a mateřskou školu ve Stráži pod Ralskem. Nový portál by měl být moderní, přehledný a v neposlední řadě by měla být zajištěna také snadná správa obsahu celého portálu. Pro tyto potřeby mělo vzniknout administrační prostředí na míru, které umožní upravovat a přidávat obsah veřejně přístupné části webu. Celý proces vývoje tohoto webového portálu je detailně popsán v praktické části této práce. Poměrně obsáhle je vylíčena i nově navržená databázová struktura včetně všech náležitostí. V neposlední řadě je čtenářům představena také obecná struktura webových stránek a veškeré funkce administračního prostředí.

1. Webový portál a současné trendy při jeho návrhu

1.1 Definice webového portálu

Webový portál je zvláštním typem webové stránky, jenž se odlišuje zejména tím, že shromažďuje více informací z různých zdrojů na jednom místě. Každé informaci je přitom přidělen prostor na speciálně vyhrazeném místě, jejž nazýváme „portlet“. Jednou z užitečných komponent webového portálu je například vyhledávač, který umožňuje prohledávat vnitřní obsah webu. Webový portál se skládá z jednotlivých webových stránek. [10]

1.2 Rozdělení dle účelu

Webové stránky obecně můžeme rozdělit na tři typy dle účelu, ke kterému slouží: webová prezentace, e-shop a webová aplikace.

Webová prezentace, jak již název napovídá, slouží zejména k prezentaci případně propagaci požadovaného předmětu. Daným předmětem může být obvykle firma, osoba nebo nějaký produkt či služba. Návštěvník na tyto prezentační weby zavítá povětšinou za nějakým konkrétním účelem, kdy například potřebuje nalézt kontaktní či jiné informace.

Vedle webových prezentací firmy využívají také druhý typ webových stránek, jímž jsou e-shopy. E-shop překračuje prezentační rámec a zaměřuje se přímo na prodej vlastních produktů či služeb. Jinými slovy e-shop na rozdíl od webové prezentace nenapomáhá k zisku, ale přímo jej generuje. Uživatelé od e-shopů očekávají přehlednost, nízké ceny a rychlost vyřízení objednávky. Vlastníci si od e-shopu naopak slibují nižší náklady oproti kamenné prodejně a vyšší množství zákazníků.

Posledním typem webových stránek, který budeme rozlišovat, jsou webové aplikace. Pod tímto pojmem se může ukrývat širší spektrum od sebe v zásadě velmi odlišných webových stránek. Prakticky do této kategorie spadají všechny aplikace, které si získávají uživatele svým obsahem nebo funkcemi. Účelem bývá získat vysoké počty pravidelných návštěvníků a generovat tak zisk skrze reklamu umístěnou na webu. [1]

1.3 Rozdělení dle způsobu generování obsahu

Jedna z prvních věcí, kterou je nutné vyřešit hned v počátku návrhu webových stránek, je volba mezi statickými a dynamickými webovými stránkami. Vedle těchto dvou typů se někdy hovoří také o tzv. hybridních webových stránkách, které kombinují vlastnosti obou výše zmíněných. [11]

Statické webové stránky jsou velice jednoduché na tvorbu, avšak mají spoustu omezení. Jsou to takové stránky, jejichž obsah je napsán pouze v jazyce HTML a je nedílnou součástí jednotlivých HTML souborů. Tyto soubory jsou mezi sebou vzájemně propojeny skrze statické hypertextové odkazy například v menu nebo na jiném místě v textu, obrázku apod. Kromě jazyka HTML statické webové stránky rovněž často využívají kaskádové styly. Vzhledem k povaze takovýchto stránek je případná úprava či nástavba obsahu náročná a může ji provést pouze někdo se znalostí jazyka HTML. Úprava obsahu tedy spočívá ve změně kódu některého ze souborů, načež je ještě nutné tento soubor aktualizovat (přepsat) na webovém serveru. Z tohoto důvodu se statické webové stránky hodí pouze pro užší sortu zákazníků. Obvykle je to někdo, kdo se chce prezentovat na webu, avšak nepovažuje za nutné, aby obsah pravidelně aktualizoval. Takové stránky ovšem bývají pro uživatele méně zajímavé a informace na nich mohou snadno „zestárnout“. Výhodou je zpravidla nižší počáteční investice než v případě stránek dynamických. [12][13]

Dynamické webové stránky jsou naproti tomu kolekcí stránek s dynamicky se měnícím obsahem. Zobrazovaný obsah je přitom čerpán z databáze a k jeho generování je nutné použití nějakého skriptovacího jazyka. Skriptovací jazyky můžeme rozdělit na dva základní typy podle toho, kde dojde ke spuštění daného skriptu. Naprogramované skripty se provádí (spouštějí) buď na straně serveru, nebo na straně klienta.

Jazyky z první zmíněné skupiny se v praxi používají následujícím způsobem. Uživatel nejprve odešle požadavek na server. Na serveru pak je, aby provedl daný skript, který zpracuje data a vygeneruje požadovaný obsah do HTML stránek. Nakonec jsou uživateli odeslány již hotové HTML stránky. Typickým představitelem skriptovacího jazyka využívaného pro tyto účely je jazyk PHP. PHP je v současnosti nejpoužívanějším jazykem,

jenž běží na serveru a slouží k výše zmíněnému účelu. Mezi další patří například ASP.NET, Java, static files, ColdFusion, Ruby, JavaScript, Perl, Python apod.

Druhá skupina jazyků spouští skript na straně klienta a umožňuje tak měnit určité prvky na stránce v reálném čase. K tomuto účelu se nejčastěji používá JavaScript či některé z něj vycházející frameworky jako například JQuery. Běžně slouží například k vytvoření stránkování, „pop up“ oken, dynamickému zobrazování či skrývání prvků jako například detailu náhledu textu apod.

Výhodou dynamických webových stránek je zejména jednoduchá editace a možnost vkládání nového obsahu. To je většinou zajištěno pomocí administrační části webu, ke které může přistupovat například vlastník nebo další redaktoři apod. Nevýhodou je větší časová i finanční náročnost takového projektu. [14][15][16]

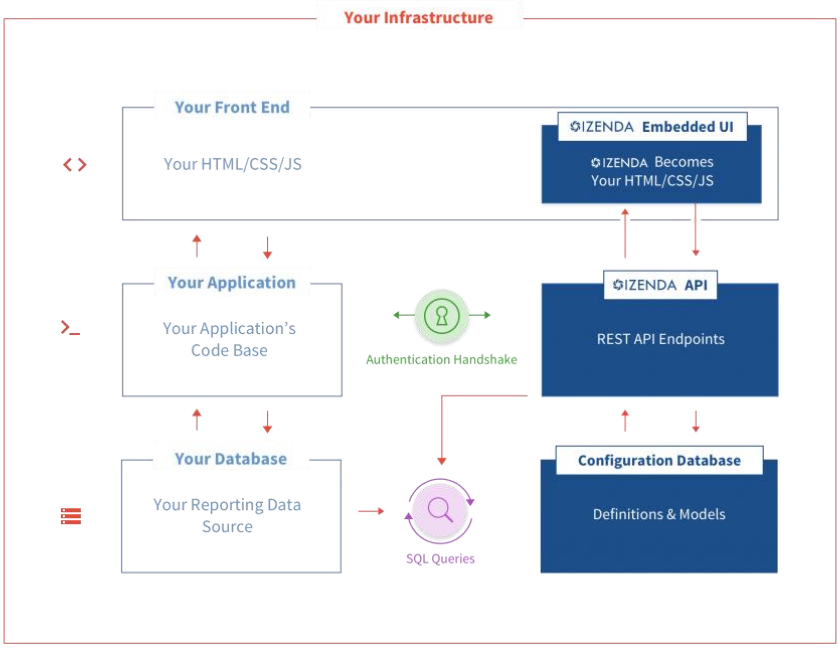
1.4 Dostupné technologie

Je zřejmé, že webové stránky prošly v posledních dvou dekádách obrovským vývojem. Ruku v ruce s tím, jak se web měnil z pohledu uživatele, samozřejmě procházel evolucí i samotný proces vývoje webových stránek a k tomu využívané technologie. Tato kapitola popisuje, jaké technologie jsou dnes vývojářům k dispozici a co konkrétně je vhodné použít. V předchozí kapitole již bylo zmíněno několik programovacích a kódovacích jazyků, které bývají k vývoji webových stránek běžně používány. Výběr konkrétního jazyka však nebude v této kapitole popsán. Mimo to existují i jiné způsoby realizace webových stránek než jen vlastní vývoj. Pokud se ovšem vývojář rozhodne pro vlastní vývoj, naskýtá se také spoustu možností a směrů, kterými se může ubírat. Příkladem lze uvést vývoj na bázi vybraného frameworku, který je jakousi nadstavbou programovacího jazyka, na němž je postaven. Všechny výše zmíněné možnosti budou hlouběji představeny v dalších částech této práce.

Při vývoji komplexnějšího softwaru se lze často setkat s dělením na vrstvy. Za zmínku také stojí, že existují dokonce dva odlišné druhy vrstev. Prvním typem jsou vrstvy logické (angl. layers). Druhým typem jsou vrstvy fyzické (angl. tiers). Ať už je řeč o vrstvách logických nebo fyzických, v obou případech se takovéto dělení aplikace obecně nazývá

jako vícevrstvá architektura. Počet vrstev se může u různých projektů lišit, avšak u těchto webových stránek, postačí tři základní abstraktní vrstvy. Každá taková vrstva představuje jakousi logickou část aplikace, potažmo webového portálu. Tyto části by od sebe měly být oddělené, přičemž pro každou vrstvu je vhodná určitá množina nástrojů, technologií, platforem či jazyků. Základní třívrstvá architektura má následující podobu. Nejníže je vrstva datová, následuje vrstva aplikační a na pomyslném vrcholu je vrstva prezentační, která je nejblíže uživateli webových stránek. Některé zdroje rozlišují vrstev více, jako příklad může posloužit vrstva interakční. Při popisu třívrstvé architektury lze pojmy fyzická a logická vrstva částečně zaměňovat, neboť jsou mezi sebou velmi úzce propojeny. [17]

Malou odbočkou z výše nastíněného směru vývoje budou takzvané redakční systémy (zkráceně také CMS). Redakčních systémů se nabízí celá řada a jedná se o v podstatě předpřipravená řešení, která umožňují jednodušší a rychlejší vývoj vlastních webových stránek pro širší okruh lidí. Toto téma je detailněji popsáno ještě v závěru této kapitoly, který následuje po podrobném vysvětlení a popsání jednotlivých vrstev třívrstvé architektury.



Obrázek 1: Infrastruktura webových stránek
Zdroj: https://www.izenda.com/blog/5-benefits-3-tier-architecture/

- Datová vrstva
- Aplikační vrstva

- Prezentační vrstva
- (Interakční vrstva)

1.4.1 Datová vrstva

Do této vrstvy spadá veškerý obsah webových stránek. Obsahem jsou myšleny například články, jejich nadpisy, komentáře uživatelů, grafy, ankety, ale i audiovizuální soubory, dokumenty ke stažení apod. Jinými slovy jsou to data, která jsou prostřednictvím webových stránek prezentována uživateli. U statických webových stránek prakticky nelze tuto vrstvu oddělit od zbylých vrstev. Naproti tomu u dynamických webových stránek je tato vrstva nejčastěji tvořena datami, která jsou uložena v databázi. Jejím úkolem je data uchovávat, zajistit k nim přístup a také zaručit jejich konzistenci. Vedle databáze může být součástí datové vrstvy rovněž souborový systém, webová služba nebo jiná aplikace. [18][19]

Databázových řešení je více, avšak v následující kapitole butou porovnány pouze tři aktuálně nejpoužívanější technologie. Při výběru byla zohledněna statistická data ze serveru db-engines.com, přičemž na prvních příčkách jsou umístěny již několik let tyto tři níže jmenované databázové systémy.

1.4.1.1 Oracle database

Oracle database je relační databázový management systém od společnosti Oracle Corporation. Je považován za jeden z nejdůvěryhodnějších a nejrozšířenějších relačních databázových systémů. Oracle database (někdy také Oracle DB, Oracle RDBMS či pouze Oracle) je konstruována jako běžná relační databáze, k jejímž datovým objektům lze přistupovat přímo za pomocí jazyka SQL. Oracle DB je přímým konkurentem Microsoft SQL Serveru na trhu podnikových databází. Výhodou je, že struktura obou technologií si je velmi podobná, což zákazníkům usnadňuje případný přechod z jedné na druhou. Klíčovou vlastností Oracle DB je oddělená logická a fyzická architektura. Díky této struktuře oplývá vysokou výkonností a velkou škálovatelností. Za zmínku stojí také to, že Oracle DB je možné provozovat napříč všemi běžnými platformami, jako je Windows, UNIX, Linux a Mac OS. Nevýhodou obzvláště pro menší projekty může být zpoplatnění tohoto softwaru. Oracle DB je distribuován ve čtyřech různých verzích od nejrobustnější

Enterprise Edition, přes Standard a Express Edition až po Oracle Lite pro mobilní zařízení. Předposlední jmenovaná verze je dokonce volně distribuována. [20]

1.4.1.2 MySQL

MySQL je obdobně jako Oracle DB plnohodnotný relační databázový management systém. Byl vytvořen švédskou společností MySQL AB a přestože je dnes již ve vlastnictví Sun Microsystems (dceřiná společnost Oracle corporation), stále je distribuován jako freeware. MySQL je napsán v jazyce C a C++ a je rovněž kompatibilní se všemi běžnými operačními systémy.

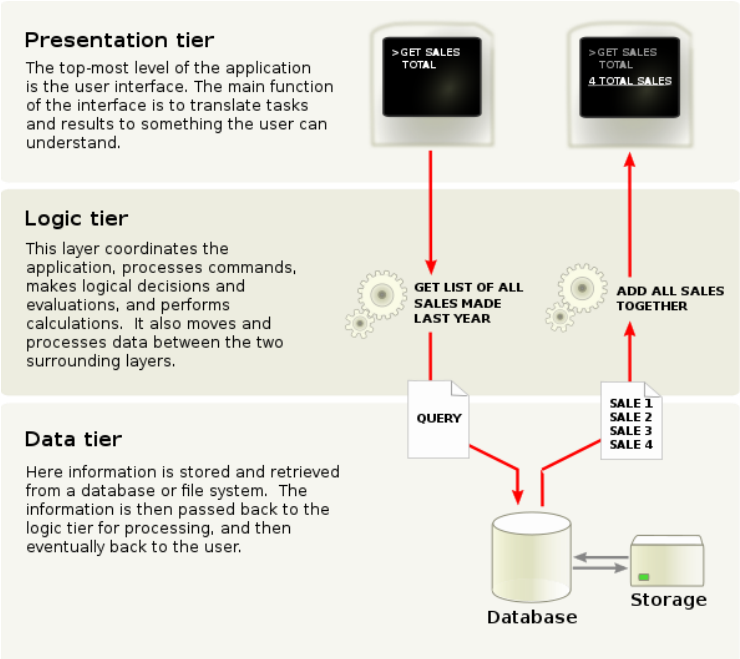
MySQL je velmi populární mezi web-hostingovými aplikacemi zejména díky množství optimalizovaných funkcí jako např. HTML datové typy a samozřejmě také díky bezplatné dostupnosti. MySQL je součástí takzvané LAMP architektury, jež je zkratkou pro Linux, Apache, MySQL a PHP. Tato architektura je kombinací nejpoužívanějších platforem používaných k vývoji a provozu pokročilých webových aplikací. Na MySQL běží i některé velmi známé webové stránky jako jsou Google, Facebook a Wikipedia, což nasvědčuje tomu, že se jedná o velmi solidní a vysoce stabilní technologii navzdory své volnější filozofii. [21]

1.4.1.3 Microsoft SQL Server (MSSQL)

Microsoft SQL Server se rovněž označuje jako RDBMS a jak již název napovídá, k jeho administraci využívá opět jazyka SQL. Na rozdíl od svých konkurentů však k jazyku Microsoft navíc přidává vlastní nadstavbu jazyka SQL v podobě Transact-SQL, která jej rozšiřuje o další funkcionality. První verze MSSQL vznikla již v roce 1989 a během let 1995 až 2016 Microsoft vydal celkem 10 verzí MSSQL. Z počátku byl MSSQL zaměřen spíše na menší aplikace pro různá firemní oddělení či pracovní kolektivy. Postupným vývojem však Microsoft schopnosti MSSQL rozšiřoval, až z něj vytvořil plnohodnotný a konkurenceschopný „enterprise“ (celopodnikový) DBMS. V následujících letech Microsoft dále začlenil nové funkcionality pro řízení a správu dat a nástroje pro analýzu dat. MSSQL se může pyšnit také tím, že podporuje širokou škálu aplikací na správu transakčních procesů, business intelligence a další obecné analytické nástroje pro podniková IT prostředí. [22]

1.4.2 Aplikační vrstva

Aplikační vrstva je v prvé řadě jakýmsi prostředníkem mezi datovou vrstvou a prezentační vrstvou. Někdy bývá tato vrstva označována také jako logická vrstva nebo business logic vrstva. Aplikační vrstva je jádrem celé aplikace. Jejím úkolem je koordinace všech procesů a příkazů a zajištění logických vyhodnocení a výpočtů. V praxi tato vrstva zajišťuje zpracování dat, která jsou uložena v datové vrstvě. Skrze aplikační vrstvu mohou být data pouze získána z datové vrstvy a následně zobrazena uživateli (v prezentační vrstvě). V jiných případech lze před konečnou prezentací uživateli data v této vrstvě ještě dále zpracovávat, například součty, průměry apod. Kromě čtení dat a jejich zpracování aplikační vrstva zajišťuje i zápis dat do datové vrstvy nebo například autentizaci uživatelů. Aplikační vrstva tedy zajišťuje obousměrný pohyb dat mezi vrstvou datovou a prezentační.



Obrázek 2: Grafický nákres třívrstvé architektury
Zdroj: https://en.wikipedia.org/wiki/Business_logic#/media/File:Overview_of_a_three-tier_application_vectorVersion.svg

Aplikační vrstvu zajistíme pomocí některého z jazyků, které se spouštějí na straně serveru. V současné době je takových jazyků na výběr mnoho. Pro potřeby této práce bylo vybráno následujících sedm programovacích jazyků, které zde budou obecně představeny a porovnány. [23][24][25]

1.4.2.1 PHP

PHP je nejčastěji využívaným programovacím jazykem pro aplikační vrstvu webových stránek. Různé zdroje uvádí, že až 80% z 10 milionů nejnavštěvovanějších webů používá PHP. Jedním z důvodů, proč je PHP tak populární, je stále častější využívání redakčních systémů, které jsou rovněž napsané v PHP. PHP bylo navrženo jako jednoduchý skriptovací jazyk pro vývoj webových aplikací. Poslední verze PHP je verze 7, která oproti předchozí verzi 5 vyřešila mnoho nekonzistencí a fatálních chyb.

PHP nabízí velmi rozsáhlé množství funkcí a širokou komunitu vývojářů. A právě díky této komunitě a vysoké popularitě PHP máme k dispozici velkou škálu naučné literatury v knižní i elektronické podobě. Vedle naučné literatury jsou k dispozici i diskuzní fóra, kde začínající programátor snadno nalezne odpovědi na všemožné dotazy. Neméně podstatnou výhodou je rovněž samotná distribuce PHP, která je zaštítěna prakticky neomezující licencí Open-source. Tento typ distribuce tak dále přispěl k rozmachu PHP a umožnil mimo jiné vznik nemalého množství frameworků.

PHP framework tvoří základní platformu, jež umožňuje vyvíjet webové aplikace. Stručně řečeno framework vývojáři usnadňuje práci a zpřehledňuje celkovou strukturu kódu i projektu jako takového. Použitím PHP frameworku ušetří vývojář obrovské množství času a zároveň je ušetřen neustálého opakováníá se při psaní kódu. V neposlední řadě framework usnadňuje připojení k databázi. Mezi nejznámější frameworky patří například Laravel, Yii Framework, Symfony nebo Cake PHP. V tuzemsku je pak velmi oblíbený domácí framework Nette.

PHP jakožto jazyk, který je velmi flexibilní, neoplývá vysokou rychlostí, avšak aktuální verze PHP7 učinila jistá zlepšení. Za zmínku stojí i fakt, že Facebook v minulosti zainvestoval nemalé sumy do vývoje PHP. Z těchto investic vzešel také virtuální stroj s názvem HipHop, kterým lze hotový kód převést a docílit tak vyšší rychlosti. Výsledná rychlost provedení operací samozřejmě nedosahuje rychlosti staticky typovaných jazyků jako je Java, ale i tak lze dosáhnout podstatného zlepšení.

Přestože PHP7 přineslo řadu zlepšení, obliba PHP pomalu upadá a to jak kvůli mnoha nedostatkům toho, jak je jazyk navržen, tak i kvůli vzestupu novějších platforem, jako je Ruby on Rails nebo Node.js. [26, 27 28]

1.4.2.2 Java

Java je široce používaný programovací jazyk, jenž byl navržen tak, aby byl vhodný zejména pro prostředí internetu. Mimo jiné je také nejpoužívanějším jazykem pro vývoj aplikací pro smartphony s prostředím Android a jedním z nejpopulárnějších jazyků pro vývoj softwaru pro zařízení, která spadají do IoT. Java byla navržena tak, aby vypadala podobně jako jazyk C++. Zároveň zde však byla snaha o prosazení objektově orientovaného programování, jednoduchou syntaxi a snazší programování. V prostředí internetu lze Javu využít pro vývoj appletů, malých aplikací, ale i komplexních aplikací, které běží na mnoha serverech. Faktem ovšem je, že Java se těší popularitě spíše u vývojářů komplexních webových aplikací. Typicky je Java používána například při vývoji webových aplikací pro vládní organizace, banky a podobné masivní organizace.

Podobně jako v případě jazyka PHP, má i Java své frameworky, které jsou postavené na jejím základě. Velmi populárním a pravděpodobně i jedním z nejlepších je framework Spring MVC. Mezi dalšími lze jmenovat například Struts, Hibernate, Vaadin nebo Grails. Poslední jmenovaný, tedy Grails, bývá často doporučován pro malé a středně velké projekty. Tím se trochu odlišuje od ostatních frameworků postavených na Javě a vyplňuje tak tuto pomyslnou mezeru. [29, 30, 31, 32]

1.4.2.3 C++

C++ je všestranný objektově orientovaný programovací jazyk. C++ bylo vyvinuto jakožto rozšíření jazyka C. C++ lze kódovat ve „stylu C“ i v objektově orientovaném stylu. V určitých situacích může být kódován oběma způsoby a jedná se tedy o dokonalý příklad tzv. hybridního jazyka. Některé zdroje tvrdí, že C++ je vysokoúrovňový programovací jazyk. Jiné naopak říkají, že se pohybuje někde uprostřed mezi nižšími a vyššími, neboť zahrnuje funkce jazyků nízkoúrovňových i vysokoúrovňových.

C++ je velice populárním jazykem, avšak jeho zaměření je primárně na systémové aplikace, ovladače, aplikace typu klient-server a vestavěný firmware. Nejpodstatnější vlastností C++ jsou předdefinované třídy. V těchto třídách jsou uložené datové typy, které mohou být vícekrát instancovány.

C++ je obecně velice schopným jazykem, avšak pro vývoj běžných webových aplikací jej nelze doporučit. Jeho síla tkví zejména ve vysokém výpočetním výkonu. Naopak rychlost

vývoje patří mezi slabší stránky tohoto jazyka. Ovšem i ve webovém vývoji má C++ svůj smysl. Používá se například pro velké projekty, kde je snaha o maximalizaci výkonu na úkor delšího vývoje. Příkladem lze uvést nemalou část sociální sítě Facebook, konkrétně například celá funkcionalita, jež obstarává živý „chat“ byla kompletně přepsána do C++. [33, 34, 35]

1.4.2.4 Python

Python je silný vysokoúrovňový objektově orientovaný programovací jazyk. Python má jednoduchý typ syntaxe, jež se velmi snadno používá a je tak vhodný i pro začínající programátory, kteří mají zkušeností buď málo, nebo se teprve chystají s programováním začít. Podobně jako Java a C++ i Python je velmi všestranným jazykem. Python se často využívá pro vývoj webů, grafických uživatelských rozhraní, ale i pro vědecké a matematické výpočty. Jak již bylo zmíněno v úvodu, syntaxe je jednoduchá a délka kódů relativně krátká. Obecně se v Pythonu dobře pracuje a umožňuje vývojáři soustředit se na problém, který řeší namísto syntaxe. Stejně jako ostatní výše zmíněné jazyky je i Python volně distribuován jako Open-source a jeho užití je tak zcela zdarma. Python nezaostává ani v dalších aspektech jako je portabilita, neboť může být spouštěn na téměř všech platformách, zahrnujíc Windows, Mac OS X a Linux.

Čistý Python lze využít i na vývoj webové aplikace, avšak ještě vhodnější je využít některý z frameworků, které jsou na Pythonu postaveny. Populárními platformami jsou například Django, Flask, Pyramid a Plone. Na Pythonu jsou postaveny také některé redakční systémy jako například Django CMS, které lze pro vývoj webové aplikace rovněž využít.

Python je všestranný a snadný jazyk nejen pro začínající programátory. Drobnou nevýhodou může být rychlost, za niž je zcela objektivně po právu kritizován. Python je ve srovnání s C++ nebo Javou skutečně pomalejší. Na druhou stranu procesy lze často optimalizovat pomocí vhodných knihoven a dosáhnout tak srovnatelného času vykonání operace. Proti nižší rychlosti Pythonu lze však argumentovat tím, že některé projekty upřednostňují rychlost vývoje před rychlostí výkonu operace (tedy naopak než je tomu u C++). Kvality Pythonu potvrzují i některé známé webového portály, které jsou v Pythonu napsány. Jedná se například o Reddit, Instagram, Pinterest, Dropbox nebo i některé poměrně velké části Google včetně YouTube. [36, 37, 38]

1.4.2.5 Ruby

Ruby je objektově orientovaný, multiplatformní, interpretovaný skriptovací programovací jazyk. Ruby byl navržen tak, aby učinil programování zábavnějším. Syntaxe Ruby je tedy opět velmi jednoduchá na pochopení i pro začátečníky, podobně jako tomu je i u konkurenčního Pythonu. Ruby je velmi flexibilním jazykem a nemá jasná a tvrdá pravidla pro to, jak psát funkce. Dá se říci, že je velmi blízko mluveným jazykům, což mnoho začínajících programátorů jistě ocení. Navíc je Ruby více tolerantní k chybám a je ochotný některé chyby „odpouštět".

Na druhou stranu to, co doposud zní jako samé výhody, může z jiného pohledu vyplynout jako zcela zásadní nevýhoda. S tím jak aplikace v Ruby roste a stává se komplexnější, se mohou stále častěji vyskytovat chyby, které není jednoduché vystopovat a opravit. Aby bylo možné takový kód snadno udržovat, je již zapotřebí značných zkušeností a „know how". Řešením k tomuto potenciálnímu problému může být například zkušený mentor, s nímž bude efektivita učení a řešení problémů na mnohem vyšší úrovni. Mimo to má i Ruby velmi početnou komunitu vývojářů, takže se má začínající programátor s problémem vždy kam obrátit.

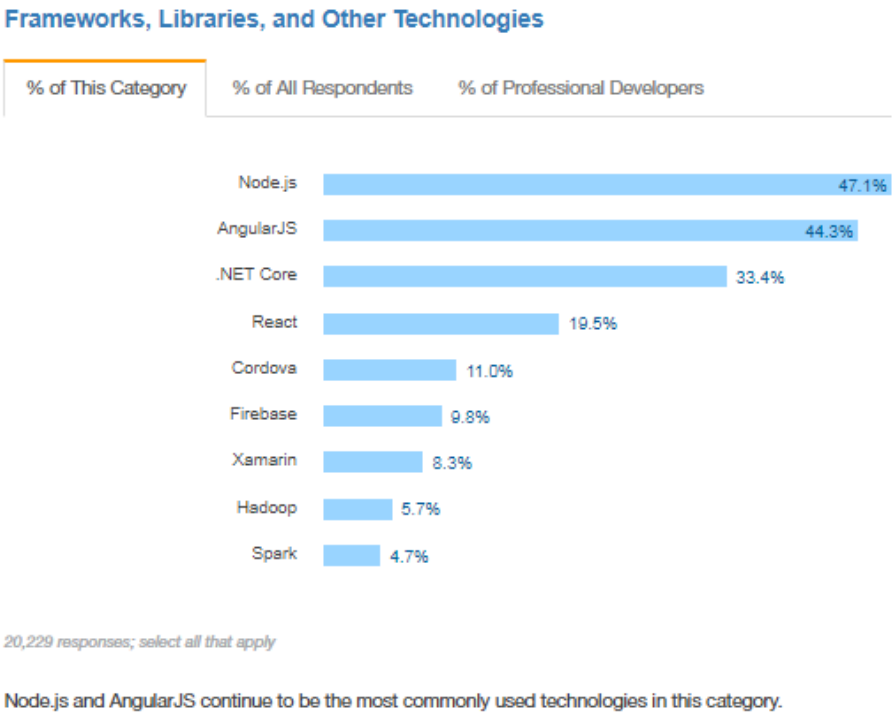
Ruby se také stejně jako Python řadí spíše k těm pomalejším jazykům. Je to tím, že jsou oba tyto jazyky příliš flexibilní. Existují zde i alternativy, jako je JRuby, jenž může být podstatně rychlejší, přesto však nedosahuje takových rychlostí jako například Java.

Relativně nižší rychlost ovšem nemusí být vždy překážkou a tak si i Ruby našla uplatnění hned u několika známějších webových aplikací. Ve výčtu takových aplikací nechybí například Twitch, Airbnb, GitHub, AngelList a Hulu. [39]

1.4.2.6 JavaScript & Node.js

Když se řekne JavaScript, většina si asi vybaví spíše frontend než backend nebo aplikační vrstvu. A skutečně původním teritoriem JavaScriptu byl především frontend, potažmo interakční vrstva. V kapitole o interakční vrstvě je JavaScript zmíněn také, avšak v současnosti má JavaScript své místo i v backendu, konkrétně v aplikační vrstvě. V roce 2009 totiž nastal zlom a první pokus o to dostat JavaScript mezi „server-side" jazyky. Tím počinem, který se o to zasloužil byl Node.js. Node.js rozdělil komunitu vývojářů a odborníků na dva antagonistické tábory, kde jej jedni oslavují a druzí zavrhují. Jak už bývá

zvykem, aby něco mohlo budit tolik kontroverze, musí to být také populární. Nutno říci, že co se do popularity týče, není na tom JavaScript ani Node.js vůbec špatně. Grafy níže názorně zobrazují výsledky ankety na populárním vývojářském webu Stack Overflow.



Obrázek 3: Popularita Node.js dle ankety na StackOverFlow
Zdroj: https://insights.stackoverflow.com/survey/2017#technology

Kromě toho je aktuálně Node.js jazykem s nejrychlejším tempem růstu popularity během posledních pěti let. Stejně tak je na vzestupu i JavaScript. Oproti tomu tradiční jazyky jako C# či PHP naopak v oblíbenosti pomalu ztrácejí.

Vysoká popularita Node.js jistě není bezdůvodná. Důvody, proč tolik vývojářů volí právě Node.js, jsou následující. Za prvé, je velice pravděpodobné, že JavaScript bude ve větší či menší míře použit při vývoji frontendu a v takovém případě je tato univerzálnost kódu velkou výhodou. Za druhé, nástroje jako je „webpack“ umožňují opětovně použít některé části kódu na obou stranách projektu, čímž je zachována logická souvislost v celém systému. Některé zdroje mohou tuto vlastnost označit jako negativum a tvrdí, že JavaScript nutí vývojáře používat jej skrze celý projekt, avšak to není tak úplně pravda. Je například možné kombinovat Node.js s knihovnami napsanými v Pythonu. Další vlastnost, kterou Node.js disponuje je „async/await“, který změnil pohled na to, jakým se píše asynchronní kód. Způsob, jakým zde funguje asynchronní kód a jak vypadá, připomíná spíše kód

synchronní. Všechny tyto vlastnosti dělají z Node.js výborný jazyk a to zejména pro níže vyjmenované typy aplikací.

První kategorií jsou aplikace, které běží v reálném čase. Konkrétně do této kategorie spadá vše od živých „chatů“ až po online hry a všechny tyto aplikace přitom mohou těžit z architektury Node.js. Tyto aplikace totiž operuji v daném časovém rámci, jejž uživatelé vnímají jako aktuální a bezprostřední. Node.js je více než dobrým řešením pro rychlou odezvu, kterou tyto programy potřebují k tomu, aby mohly efektivně fungovat. Krom toho Node.js umožňuje zpracování požadavků od mnoha uživatelů zároveň a dovoluje opětovně užívat balíčky knihoven. Díky tomu pak může synchronizace dat mezi uživatelem a serverem probíhat velmi rychle. Druhou kategorií jsou jednostránkové webové aplikace, které načtou jednu jednoduchou HTML stránku a poté dynamicky doplňují na stránku informace, které uživatel vyvolává pomocí interakčních prvků na stránce. Tyto stránky se však běžně tvoří skrze JavaScript ve frontendu, tedy takový, který se spouští na straně klienta. Takto napsané webové stránky s sebou ovšem nesou jedno negativum, kterým je horší SEO. Oblíbeným řešením tohoto problému je právě Node.js běžící na straně serveru.

Na co se naopak Node.js příliš nehodí, jsou opravdu velké projekty. Ne že by nebylo možné je vytvářet, ale může se stát, že zde vývojář narazí na jistá omezení. Dokonce i tvůrce Node.js Ryan Dahl si tato omezení uvědomil a ze stejného důvodu se rozhodl od tohoto projektu odstoupit. Přitom se nechal slyšet: „Myslím, že Node není nejlepší systém na realizaci obřího serverového webu. Na to bych použil Go. A popravdě, to je také důvod, proč jsem z Node odešel. Byl to ten okamžik, kdy jsem si uvědomil: aha, tohle ve skutečnosti není ten nejlepší server-side systém, jaký kdy existoval." [17]

Kromě Node.js existuje ještě mnoho dalších populárních frameworků založených na JavaScriptu. Tím nejpopulárnějším je Express.js, který vychází z Node.js. Je to rychlý a minimalistický Framework pro tvorbu webových aplikací. Dalším frameworkem je Meteor, který naopak obaluje čistý JavaScript a Node.js do mnohem větší architektury. Meteor se hodí spíše pro vývoj robustních serverových aplikací. Vedle těchto frameworků dále stojí Sails.js, který je „real-time“ MVC frameworkem. Byl navržen tak, aby emuloval MVC architekturu Ruby on Rails rozšířenou o podporu požadavků moderních aplikací. Ta je zajištěna skrze datově řízená API se škálovatelnou architekturou orientovanou na služby. Poslední Framework, který zde bude uveden je Koa.js. Koa.js byla představena jakožto

Framework nové generace pro Node.js. Koa.js je menší, expresivnější a nabízí pevnější základy pro webové aplikace a API. Populárních JavaScriptových frameworků je samozřejmě mnohem více, namátkou jsou to jména jako Nest.js, Hapi.js, Socket.io, Mean.js, Derby.js a Keystone.js. [40, 41]

1.4.2.7 Go

Go je programovací jazyk, který byl vyvinut malým týmem ve společnosti Google. Jedná se o poměrně mladý jazyk, neboť představen byl až v listopadu roku 2009. Je to kompilovaný, staticky typovaný jazyk se syntaxí odvozenou od jazyka C. Oproti jazyku C však nabízí souběžné programování ve stylu CSP (communicating sequential processes), ochranu paměti, limitované strukturální typování a „odvoz odpadu“ (garbage collection). Jazyk go včetně všech nástrojů a kompilátoru je zdarma a Open-source.

Go jakožto jeden z mála jazyků, které byly vytvořeny až po nástupu víceprocesorových počítačů, je napsán tak, aby byl schopný z toho co nejvíce vytěžit. Go je kompilovaný jazyk, což znamená, že napsaný kód je převeden do strojového a poté může být zpracován procesorem. Na rozdíl od ostatních nových jazyků, jako je Swift nebo Rust, go nezávisí na LLVM (překladač – Low Level Virtual Machine) ani na žádné jiné vrstvě virtuálního stroje. Go má jednu zásadní vlastnost, která se nazývá „goroutines“. „Goroutines“ je efektivnější a méně náročný způsob běhu souběžných procesů.

Hlavní motivací vzniku Go byla nespokojenost se současným stavem na poli programovacích jazyků. Autoři tvrdí, že ostatní jazyky nabízejí buď efektivní kompilaci nebo výkonost a nebo jednoduché programování, avšak žádný z běžných jazyků nenabízel tyto všechny tři vlastnosti zároveň. [3] Tyto tři vlastnosti se tak staly hlavními pilíři při návrhu tohoto programovacího jazyka. Časy kompilace jsou vskutku rychlé a tak bezprostřední spuštění programu se tváří jakoby Go byl jazyk skriptovací. Go také netrpí neustálými změnami jazyka jako například Swift. Oproti tomu standardní knihovna Go nabízí spoustu užitečných prvků a je naprosto dostačující například k napsání serveru bez použití dalších vrstev frameworků na rozdíl od Ruby nebo JavaScriptu.

Go si tedy jistě najde projekty, pro které je tím nejvhodnějším nástrojem. Již nyní jej využívá mnoho webových stránek a aplikací. Na prvním místě je to samozřejmě Google,

kvůli kterému celý projekt také vznikl, dále jej využívají společnosti jako Adobe, SoundCloud, Reddit, BBC, eBay, BitBucket, Netflix a Uber. [4][42][43][44][45]

1.4.3 Prezentační vrstva

Prezentační vrstva leží nejvýše z hlediska struktury třívrstvé architektury. Někdy se také nazývá jako vrstva klienta, neboť je klientovi ze všech vrstev nejblíže. Hlavním úkolem prezentační vrstvy je zprostředkování komunikace mezi koncovým uživatelem a aplikační vrstvou. Vrstva datová a vrstva aplikační zajišťuje a zprostředkovává obsah webových stránek. Avšak bez aplikační vrstvy by koncový uživatel neměl možnost daný obsah číst nebo zadávat své požadavky. Uživatel je tedy v přímém kontaktu s prezentační vrstvou, s níž komunikuje prostřednictví myši a klávesnice. Prezentační vrstva určuje, jakým způsobem bude uživatel moci komunikovat s aplikační vrstvou a také jak budou uživateli prezentována zobrazovaná data, případně obsah webových stránek.

Z hlediska jednoduchého dělení vývoje na dvě základní části spadá již veškerý kód napsaný v některém z níže uvedených jazyků do takzvaného frontendu. Pozorný čtenář si jistě povšimne, že tato práce se týká vývoje backendu webového portálu a není tedy jejím cílem se dopodrobna zabývat prezentační vrstvou - potažmo frontendem. Pro ucelenost tohoto textu však budou představeny alespoň základní informace o těchto vrstvách. Pro vývoj prezentační vrstvy se v současnosti využívá téměř výhradně kombinace HTML, CSS a JavaScriptu.

HTML (HyperText Markup Language) je jednoduchý značkovací jazyk, jenž slouží k vytvoření základní struktury webových stránek. Tento jazyk neustále prochází vývojem, tak aby vyhovoval stále vyšším požadavkům internetového publika. O revize a vývoj se stará organizace W3C, jež je pověřena správou a údržbou HTML. Již od konce října roku 2014 se můžeme těšit z nejnovější verze tohoto jazyka, jež je označována jako HTML5. Oproti předchozí verzi z roku HTML4 1997 přinesl mnoho změn, z nichž za zmínku stojí například přímá podpora přehrávání multimédií v prohlížeči.

CSS neboli kaskádové styly rozšiřují HTML nebo jiný značkovací jazyk o styly. Obdobně jako HTML byly kaskádové styly navrženy organizací W3C. Primárním účelem kaskádových stylů je oddělení vzhledu, a potažmo struktury od obsahu webových stránek.

Konkrétně se jedná například o pozici či barvu ploch, velikost písma, jeho font apod. CSS je spolu s HTML při tvorbě webových stránek prakticky nepostradatelné. Bez CSS nemáme nad vzhledem dokumentu dostatečnou kontrolu a případné změny jsou komplikované. Obvykle se kaskádové styly užívají tak, že jednotlivé HTML stránky (části webu) společně sdílí jeden (či více) souborů s kaskádovými styly. Tím je zajištěn jednotný vzhled celého webu. Zároveň tento způsob použití vede k odstranění repetitivního kódu, zvyšuje flexibilitu designu a snižuje zbytečnou komplexnost HTML kódu. [46][47][48]

1.4.4 Interakční vrstva

Interakční vrstvu lze považovat za jakousi nadstavbu k vrstvě prezentační. Hlavní roli zde zastávají tzv. „client-side“ jazyky, což jsou jazyky, které spouští dané skripty na straně klienta (návštěvníka webu). Tímto způsobem chování doplňují „server-side“ jazyky, kterým je například oblíbené PHP, jež často tvoří základ aplikační vrstvy. Veškeré jazyky, které spadají do této kategorie, jsou v podstatě volitelnou komponentou při vývoji webových stránek. Lze tedy říci, že tyto jazyky nejsou ani pro moderní funkční web nezbytně nutné. Ze stejného důvodu nejsou v tomto textu součástí vrstvy prezentační, neboť ta může být objektivně považována za kompletní i bez interakčních prvků, které „client-side“ jazyky nabízí. V současnosti se však jedná o velmi populární technologii, neboť pozitivně přispívá k vyšší interaktivitě uživatelského prostředí webu. Běžně se využívají k vytvoření interaktivních prvků, pomocí nichž může uživatel dodatečně měnit jemu zobrazovaný obsah webových stránek.

Nejpopulárnějším nástrojem v této kategorii je JavaScript. JavaScript je objektově orientovaný skriptovací jazyk, na jehož základě vznikla spousta knihoven, které přináší snadnější práci s některými prvky. Příkladem nejznámější JavaScriptové knihovny může být například jQuery či pokročilejší frameworky jako je Angular JS a React. [49][50][51]

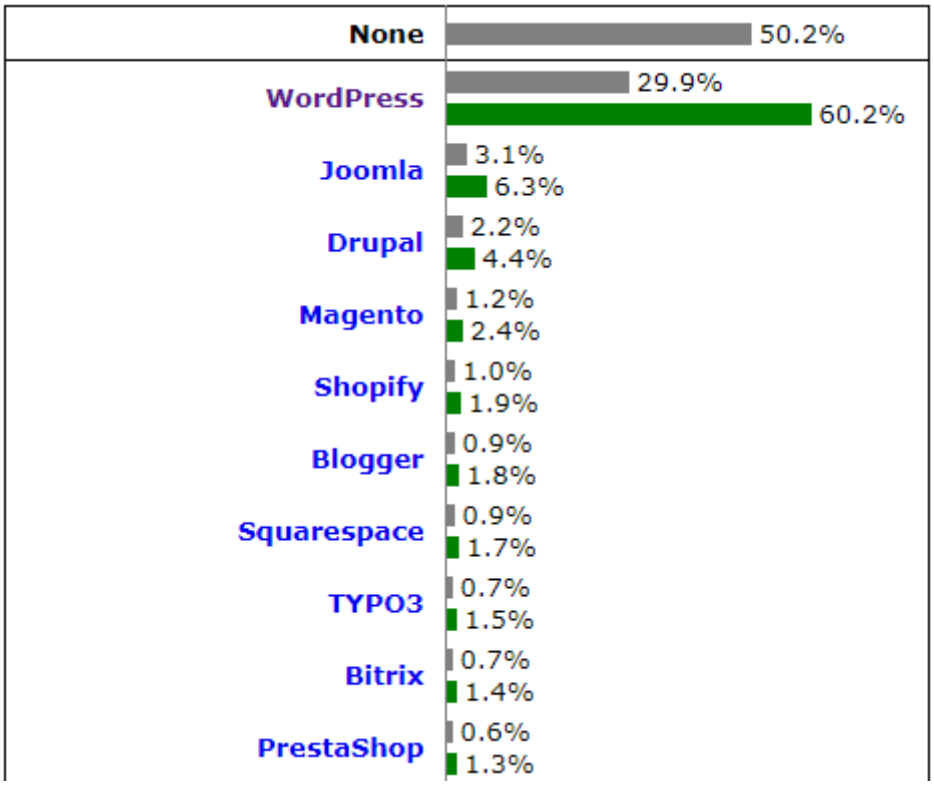
1.4.5 Kompletní řešení

Návrh webového portálu lze pojmout i jiným způsobem než vývojem vlastního řešení. Jednou z možných cest je využití hotových systémů pro správu obsahu (často označované jako „CMS“ z anglického „content management systém“). Tyto systémy jsou někdy

nazývány také jako redakční či publikační systémy. Co přesně si pod těmito pojmy představit? Jak již z názvu vyplývá, jedná se o systémy, které umožňují běžnému uživateli spravovat (přidávat i upravovat) obsah webového portálu.

Pokud se uživatel rozhodne využít tohoto řešení, znamená to pro něj velice rychlou a relativně levnou realizaci webového portálu. Samotný postup je velmi snadný a intuitivní. Uživatel si nejprve vybere konkrétní CMS systém. Na výběr má opravdu nespočet možností, avšak většina uživatelů zřejmě sáhne po nějakém osvědčeném systému, který se těší vysoké popularitě. Rozdíly mezi jednotlivými systémy mohou být v ceně, nabízených službách, funkcích, intuitivním prostředí a podobně. Volit lze samozřejmě i dle přechozích zkušeností nebo recenzí a doporučení ostatních uživatelů těchto systémů. Dalším krokem je pak výběr celkového vzhledu v závislosti na zvolené šabloně. Tyto šablony mohou být placené či volně šiřitelné, uživatel má tak opět na výběr spoustu možností. Posledním a zároveň tím možná nejdůležitějším krokem je vytvoření požadovaného obsahu tak, aby web vyhovoval původním požadavkům.

Absolutně nejpopulárnější redakční systém v současné době je WordPress s více než 60% podílem na trhu těchto systémů. Mezi další velmi oblíbené například Joomla, Drupal či Magento.



Obrázek 4: Graf využití CMS mezi webovými stránkami. 50,2% Webových stránek nepoužívá žádný z monitorovaných redakčních systémů. Zeleně je vyznačen podíl na trhu CMS, šedě pak celkový podíl mezi všemi monitorovanými weby.
Zdroj: https://w3techs.com/technologies/overview/content_management/all/ [vid. 4.3.2018]

WordPress již několik let působí na pomyslné vedoucí pozici na poli redakčních systémů. Je nutné podotknout, že toto prvenství vychází z hlediska oblíbenosti nikoli z objektivního srovnání s konkurenčními systémy. Důvodem této popularity je pravděpodobně přívětivá cenová politika a nyní samozřejmě již dosti silná pozice na trhu. Samotná základní služba je zdarma, přičemž placené jsou některé šablony a přídavné funkce. WordPress je vhodný pro široké spektrum webů od malých blogů až po středně velké webové portály a zpravodajské online servery. Co se technologického řešení týče, je postaven na PHP a MySQL.

„Joomla!“ je v současnosti druhým nejoblíbenějším redakčním systémem. Jedná se o open-source projekt, který je distribuován zdarma pod licencí GNU GPL. Během posledních několika let se systému Joomla podařilo získat i několik ocenění jako například první místo v kategorii „Best Free CMS“ v letech 2015, 2016 a 2017. Tato kategorie je vyhlašována v rámci soutěže „CMS Critic People“s Choice Awards“. Ve stejné kategorii byl nominován i konkurenční WordPress. „Joomla!“ se tak může pyšnit vítězstvím nad nejpopulárnějším konkurenčním CMS. Systém „Joomla!“ je stejně jako WordPress napsán

rovněž v PHP, avšak využívá objektové programování a architekturu MVC (model-view-controller). K ukládání datové struktury může posloužit buď standardní MySQL, ale také MS SQL či databáze PostgreSQL. K popularitě tohoto systému vede opět cenová politika a velká komunita vývojářů po celém světě, jež usiluje o neustálé zlepšování celé platformy. Možnosti využití tohoto CMS jsou vcelku široké. Lze jej běžně využít pro tvorbu e-shopu, školního portálu, online magazínu nebo webových stránek pro malé podniky.

Drupal je rovněž volně dostupným Open-source CMS frameworkem. Konkrétně je distribuován pod volnou licencí GNU GPL. V současné době využívá framework Drupal okolo 2,2 % všech webových stránek na celém světě a v popularitě mezi CMS je tak na třetím místě. Weby, které Drupal využívají, pokrývají opět širší spektrum od osobních blogů přes internetová fóra až po korporátní či vládní portály. Často se se systémem Drupal lze setkat také jako s nástrojem pro management znalostí (angl. knowledge management), což je pojem vyjadřující proces vytváření, sdílení a užívání znalostí a informací v organizaci. Na vývoji služby Drupal se aktivně podílí více než 100 000 uživatelů. K dispozici jsou tak stovky volně dostupných šablon, distribucí, desetitisíce modulů a pluginů. Standardní verze Drupal core obsahuje základní funkcionality systému pro správu obsahu. Jmenovitě jsou to funkce pro registraci uživatele, správu menu, RSS kanály, úpravu uspořádání a stylu stránku a systémovou administraci.

Z výše uvedených informací o těchto redakčních systémech nelze příliš snadno vyvozovat obecné závěry o kvalitách těchto systémů. Stejně tak nelze jednoznačně říci, který z nich je pro řešení vybraného projektu ten nejvhodnější. Nabízí se varianta jít ověřenou cestou, a tedy použít WordPress tak jako majoritní zastoupení webů s redakčním systémem. Avšak ani toto řešení nelze s jistotou doporučit. Pokud uživatel požaduje, aby mu vybraný CMS skutečně vyhovoval v každém aspektu, je pro něho nezbytně nutné, aby si jednotlivé redakční systémy vyzkoušel, porovnal je mezi sebou a až následně se rozhodl, který z nich nakonec použije pro svůj projekt. [52]

2. Postupy při návrhu databáze

Proces vývoje databáze zahrnuje tři hlavní fáze: analýzu požadavků, návrh komponent a implementaci.

2.1.1 Analýza požadavků

V první fázi analýzy požadavků (někdy také nazývané pouze jako fáze požadavků) je nutné od uživatelů systému získat jejich představu o výsledném řešení. S využitím získaných informací a požadavků, je možné započít tvorbu datového modelu. Datový model reprezentuje obsah, vztahy a omezení dat, aby bylo možné splnit příslušné požadavky. Posledním volitelným krokem v této fázi je vytvoření jakéhosi prototypu vybraných částí budoucího systému. Tyto prototypy slouží například k získání zpětné vazby od budoucích uživatelů daného systému. [2]

2.1.2 Datový model

Datové modely lze vytvářet několika způsoby. Nejčastěji se však k tomuto účelu používá E-R model (z angl. entity-relationship model – tedy model entit a vztahů). Poprvé tento model představil Peter Chen již v roce 1976, avšak později byl zdokonalen a vznikl tak rozšířený E-R model. Pro jednoznačné rozlišení jednotlivých modelů bude v tomto textu původní E-R model z roku 1976 nazýván jako Chenův model a všechny současné modely převezmou označení E-R model. V současnosti se používá vícero druhů E-R modelů. My se prozatím zaměříme na ten tradiční (neplést s původním Chenovým). E-R model má tyto základní prvky: entity, atributy, identifikátory a vztahy.

2.1.2.1 Entity

Pro začátek lze říci, že entitou může být cokoliv, co chce uživatel v systému sledovat. Jedna z mnoha definic entity říká, že entita představuje jakýkoliv objekt reálného světa a to buď fyzický (člověk, dům, auto,…) nebo abstraktní (jev či událost), jenž je součástí datového modelu. Entita přitom musí splňovat tato pravidla: je zřetelně odlišitelná od ostatních entit a existuje nezávisle na ostatních entitách.

V souvislosti s entitou se lze setkat také s třídou entity (někdy také nazývána jako typ entity). Dále je nutné zmínit, že entita je pouze abstraktní pojem. Konkrétním ztělesněním entity je její instance. Příkladem instance může být třeba obor „Manažerská Informatika“. Samotný „obor“ je pak v tomto případě třídou takové entity. „Definice říká, že třída entit je kolekcí entit, kterou popisuje struktura entit v dané třídě. Třída entit obvykle obsahuje mnoho instancí entit.“

2.1.2.2 Atributy entity

Třídy entit se od sebe odlišují atributy, jež popisují vlastnosti dané entity. Zároveň platí, že všechny instance entity v jedné třídě spolu sdílí společné atributy. Atributy výše uvédené entity mohou být například id, jméno oboru, garant oboru, fakulta apod. Každý atribut má určen datový typ a vlastnosti. Vlastnosti atributu upřesňují, zda je povinný, zda má atribut nějakou výchozí hodnotu a zda je hodnota atributu omezena například intervalem či jinými omezeními.

2.1.2.3 Identifikátory entity

„Instance entit mají identifikátory. Identifikátory jsou atributy, které pojmenovávají čili identifikují jednotlivé instance.“ (sic) [2] Identifikátory mohou být buď jednoduché, tedy takové, které zahrnují pouze jeden atribut entity, nebo složené, které se skládají ze dvou či více atributů. Kromě toho můžeme identifikátory dělit na jedinečné (unikátní) a nejedinečné. Jak již z názvu vyplývá, hodnota jedinečného identifikátoru přímo určuje pouze jednu konkrétní instanci. Oproti tomu hodnota identifikátoru nejedinečného identifikuje instancí více. Příkladem jedinečného identifikátoru může být například číslo platební karty. Zástupcem nejedinečného identifikátoru může být například datum narození, jméno, věk apod. V tuto chvíli je nutné poznamenat, že ačkoliv identifikátory mohou připomínat klíče v relačním modelu, odlišují se od nich dvěma rozdíly. Za prvé identifikátory jsou pouze logickou koncepcí a představují pouze jakýsi uživatelem navržený název dané entity. Za druhé jak je zmíněno již výše, identifikátory mohou být i nejedinečné, což ovšem neplatí pro primární a kandidátní klíče, které musí být vždy unikátní.

2.1.2.4 Vztahy

Entity v diagramu mohou být mezi sebou sdružovány pomocí vztahů. Podobně jako je rozlišována třída entity a instance entity, existují třídy vztahů mezi třídami entit a naopak instance vztahů mezi instancemi entit. U každého vztahu se označuje jeho stupeň. Stupeň určuje počet entit, kterých se vztah týká. Nejnižší je tedy stupeň binární (nebo také druhý stupeň), který definuje vztah mezi dvěma entitami. Binární vztah se může vyskytovat ve třech následujících podobách.

- vztah 1:1
- vztah 1:N
- vztah N:M

Vztah 1:1 je definován tak, že jediná instance daného typu souvisí s jedinou instancí jiného typu. Příkladem lze uvést třeba vztah mezi entitou zaměstnanec a šatní skříňka. Každý zaměstnanec má pouze jednu šatní skříňku a zároveň platí, že žádná skříňka není přidělena více než jednomu zaměstnanci.

U vztahu 1:N stále platí, že jedna ze zúčastněných entit může být ve vztahu pouze s jednou entitou druhého typu. Liší se tím, že druhá z nich může být ve vztahu s více entitami prvního typu. Celá situace může být opět vysvětlena na příkladu se skříňkami. Pokud zaměstnanec bude ve vztahu 1:N na pozici 1, znamená to, že jeden zaměstnanec může mít více (N) skříněk. V takovém případě ovšem nikdy nemůžou skříňku sdílet dva nebo více zaměstnanců. Jinými slovy entita na pozici N (skříňka) nemůže být ve vztahu s více než jednou entitou na pozici 1 (v tomto případě zaměstnancem). Naopak v situaci, kdy mezi zaměstnanci a skříňkami panuje vztah N:1, by mohlo jednu skříňku vlastnit až N lidí, avšak žádný ze zaměstnanců by nemohl vlastnit skříňky dvě. Z těchto dvou příkladů vyplývá, že u vztahů typu 1:N je velmi důležité, na které straně vztahu dané entity leží. Pro lepší orientaci lze při popisu tohoto typu vztahu užívat označení předek a potomek. Předek je označení pro entitu nadřazenou potomku a leží na jednotkové straně vztahu (může mít více potomků). Naopak potomek je entita podřízená a má vždy pouze jednoho předka.

Posledním typem binárního vztahu je vztah N:M. Vztah N:M je v podstatě kombinací vztahů 1:N a N:1. Zároveň je vztah N:M také nejméně restriktivní ze třech typů binárního vztahu a umožňuje realizovat největší množství kombinací. Za předpokladu, že mezi skříňkami a zaměstnanci existuje vztah M:N, znamená to, že jednomu zaměstnanci může

být přiděleno N skříněk a zároveň platí, že jedna skříňka může být přiřazena M zaměstnancům. Zaměstnanci by tedy mohli vlastnit skříněk několik a zároveň by jedna skříňka mohla být majetkem více zaměstnanců. V reálném světě by toto řešení zřejmě nebylo zcela optimální, avšak pro názornou představu by měl tento příklad dobře posloužit.

Tímto způsobem tedy fungují tři typy binárních vztahů. Tyto vztahy jsou nazývány (a klasifikovány) dle své maximální kardinality. Maximální kardinalita určuje nejvyšší možný počet instancí entity, které se mohou účastnit na instanci vztahu. Vztah tedy nemusí být popsán pouze obecně 1:N, ale například 1:5. Kromě maximální kardinality lze u vztahu určit i minimální kardinalitu. Ta se ovšem nezapisuje do kosočtverce mezi entity a ani nijak neovlivňuje, o jaký typ vztahu se jedná. Zakresluje se na úsečku vztahu, jež entity spojuje. Ovál znamená, že kardinalita je nulová, tedy účast entity ve vztahu je volitelná. Svislá čára značí kardinalitu rovnou jedné, a tedy povinou účast entity ve vztahu. [2][53]

2.1.2.5 Datový model – souhrn

Na závěr této subkapitoly o datovém modelu by bylo vhodné shrnout některá fakta, která mohou být zejména začínajícím programátorům nejasná. Po prvním seznámení s datovým modelem mohou vyvstat některé obligátní otázky. Typicky například k čemu vlastně datový model slouží a jaký je rozdíl mezi entitou a tabulkou? Datový model se používá proto, neboť je jeho návrh snazší než návrh databáze, který následuje až po pečlivém vyladění datového modelu. Za druhé rozdíl mezi entitou a tabulkou je dán zejména tím, že entity jsou mezi sebou provázány vztahy, jež jsou specifikované velice jednoduše pouhou čarou. Oproti tomu vztahy mezi tabulkami jsou definované pomocí cizích klíčů. V datovém modelu tak vývojáři odpadá hned několik starostí, jako jsou primární a cizí klíče, integritní omezení a podobně. Z toho opět vyplývá závěr, který je zřejmý již z odpovědi na první položenou otázku. S entitami, potažmo s celým modelem, se v počáteční fázi projektu pracuje o mnoho lépe. Důvodem jsou zejména nejasné představy o vztazích mezi entitami, jež se v průběhu návrhu mohou měnit. Základním pravidlem při návrhu datového modelu je postup od obecného ke konkrétnímu. Začíná se návrhem entity, poté se definují vztahy mezi nimi a nakonec se specifikují jednotlivé atributy.

2.1.3 Návrh komponent

Ve fázi návrhu komponent (někdy také nazývané jako fáze návrhu systému či fáze návrhu) dochází k převedení navrženého datového modelu na databázový návrh. Takový návrh již na rozdíl od modelu obsahuje tabulky, vztahy a omezení. Součástí návrhu jsou jednotlivé názvy tabulek a všech jejich sloupců. Poté následuje určení datových typů a vlastností jednotlivých sloupců a zvolení primárních a cizích klíčů. Posledním krokem databázového návrhu potažmo celé této fáze je určení omezení. Omezení dat jsou tvořena limity datových hodnot omezeními referenční integrity a obchodními pravidly. Limity datových hodnot definují, jakých hodnot můžou data ve sloupci nabývat. Obvykle se jedná o definice intervalů, tedy maximální a minimální hodnoty, počet znaků či cifer apod. Referenční integrita zajišťuje, aby všechny cizí klíče byly platné. Ve sloupci s cizím klíčem, tak může být buď hodnota „NULL“ nebo cizí klíč, který odpovídá primárnímu klíči v nadřízené tabulce. Díky tomuto omezení tak například nelze z nadřízené tabulky odstranit záznam, jehož primární klíč je cizím klíčem v podřízené tabulce. Obchodní pravidla mohou omezovat kardinalitu ve vztazích. Takové pravidlo může například říkat, že na každý zakoupený díl musí existovat cenová nabídka alespoň od dvou dodavatelů. [2]

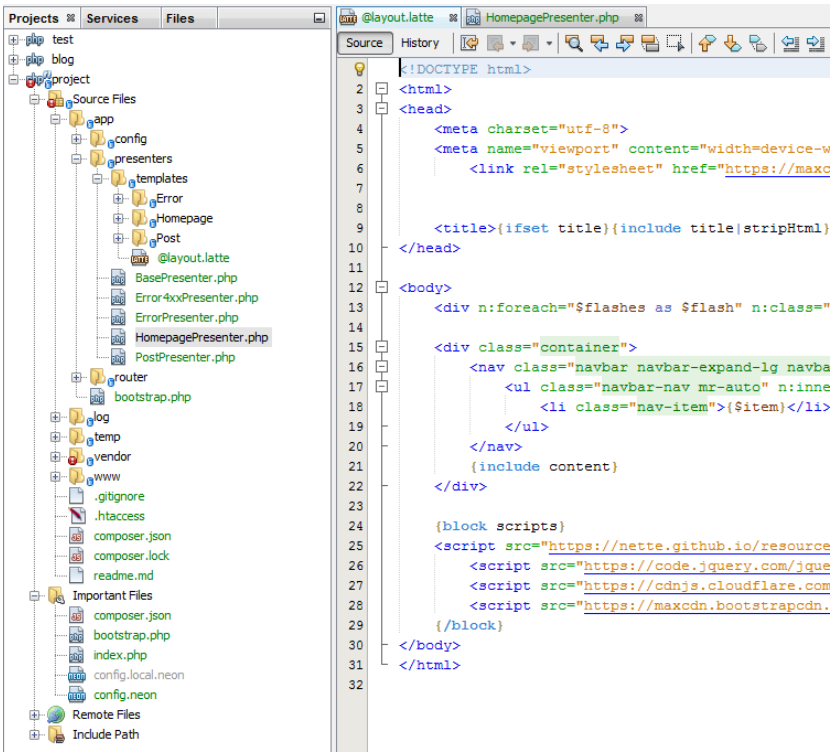
2.1.4 Implementace

„V poslední fázi vývoje konstruujeme databázi v databázovém systému, vkládáme do ní data, vytváříme dotazy, formuláře a sestavy, píšeme aplikační programy a všechny uvedené komponenty testujeme. Ve stejné fázi nakonec probíhá i školení uživatelů, psaní dokumentace a zprovoznění nového systému. “ [2]

3. Doprovodné technologie

3.1 Vývojové prostředí

K zefektivnění celého procesu programování lze použít celou řadu nástrojů. V první řadě je to nějaké vhodné vývojové prostředí, jež zkráceně nazýváme IDE (z angl.. „Integrated Development Kit“). Někteří programátoři využívají vývojová prostředí pro konkrétní jazyk, jiní nějaká univerzální jako například NetBeans. Nelze zřejmě objektivně doporučit, které konkrétní prostředí si má začínající programátor vybrat. Vše závisí spíše na osobním vkusu, či případných návycích, pokud má programátor již nějaké zkušenosti. V každém případě je však dobré si nějaká prostředí vyzkoušet a navyknout si na ně. Používat namísto toho například poznámkový blok nebo jiný obyčejný textový editor je v podstatě nemyslitelné. IDE vývojáři práci výrazně zpříjemní, zrychlí a především učiní celý proces programování efektivnějším.



Obr. č. 1: Výstřižek z pracovní plochy IDE NetBeans
Zdroj: Vlastní

Na obrázku je vidět vlevo stromová struktura projektu, díky čemuž lze vcelku rychle hledat a otevírat různé soubory. Vpravo jsou zobrazeny aktuálně otevřené soubory, mezi nimiž

lze jednoduše přepínat. Samozřejmostí je barevné odlišení různých tagů, syntaxí apod. Mezi další velice příjemné a užitečné funkcionality patří například „hypertextové odkazy“ v napsaném kódu. Ty vývojáři například umožňují přesunout se na řádek s funkcí na niž klikl a to jak uvnitř, tak i vně daného souboru. Dále zde figurují také našeptávače, které doplňují jak známé tagy, tak i již existující názvy funkcí, proměnných apod. V neposlední řadě IDE značí také syntaktické chyby nebo varuje při nesprávných praktikách. Na závěr lze říci, že vývojová prostředí nabízí tolik funkcí, že na některé z nich programátor ani nenarazí nebo je nevyužije. To by však nemělo být důvodem k tomu vývojová prostředí ignorovat a nepoužívat. Čím dříve programátor začne pracovat s nějakým vývojovým prostředím, tím dříve si jej osvojí a může tak naplno využít jeho potenciál.

3.2 Nástroj Git

Obrázek 5: Logo nástroje Git
Zdroj: https://git-scm.com/downloads/logos

Git je aktuálně nejrozšířenější moderní systém pro správu verzí. Distribuován je jako open-source a hodí se ke všem druhům projektů od malých až po velké. Git je vyspělý, aktivně udržovaný open-source projekt vyvíjený již od roku 2005. Jeho tvůrcem je Linus Torvalds, který stojí také za vznikem jádra operačního systému Linux. Zajímavostí je, že Git byl původně vyvinut právě kvůli vývoji Linuxu, neboť systém Bitkeeper, který používali vývojáři Linuxu ke správě verzí původně, se výše zmíněná společnost rozhodla zpoplatnit. Na Git jakožto na systém pro správu verzí se spoléhá obrovské množství softwarových projektů a to jak Open-source, tak i komerčních. V pracovním prostředí je práce s Gitem nepostradatelná a zkušenosti s tímto nástrojem jsou k nezaplacení.

Co je to vlastně ten systém pro správu verzí, jak se používá a k čemu slouží? V první řadě uživateli poskytuje přístup ke kompletní historii projektu.

„Můžeme se libovolně v historii přesouvat a máme přehled o tom, kdo vytvořil jakou část kódu a co se konkrétně změnilo. Díky větvení (viz další kapitoly) můžeme vytvářet novou funkcionalitu, aniž by to mělo jakýkoliv dopad na zbytek projektu. Jestliže selže, není nic jednoduššího, než tento pokus nezahrnout do projektu, vrátit se na původní místo v historii a začít znovu. Projekt zůstane po celou dobu ve stejném stavu.“ [54]

Hierarchie uložených dat z pohledu uživatele vypadá zhruba následovně. Nejníže jsou takzvané repositáře. V každém repositáři může být jeden, ale i více projektů, pokud to uživateli tohoto nástroje z nějakého důvodu vyhovuje. Každý repositář má hlavní větev (angl. master) a dále pak větve, které si uživatel tvoří sám. Změny, které provede, může buď vrátit nebo potvrdit takzvaným „commitem“. Historie dané větve je pak složena právě z těchto „commitů“. Každý „commit“ obsahuje změny oproti „commitu“ předchozímu a zároveň je opatřen základními informacemi o autorovi, datu a čase, přičemž zde nechybí prostor ani pro krátký popis změn, který autor před uložením sepíše. V praxi vedlejší větve slouží například k vytvoření nové funkcionality, která se po zhotovení otestuje a následně se v případě uspěšné kontroly sloučí s větví hlavní. Tento proces v podstatě vytvoří novou verzi daného repositáře a potažmo projektu, od kterého se bude projekt dále vyvíjet. Díky tomuto nástroji je při tvorbě projektu spousta prostoru na kontrolní činnost a i v případě nepozornosti lze změny vracet a to i při navázání na vadný „commit“, který je později nutné zrušit. [5][6]

3.3 Nástroj pro správu databáze

Pro správu databáze je vždy vhodné využít nějaký nástroj. Celosvětovým standardem pro správu MySQL databází se stal software phpMyAdmin. Pro účely tohoto projektu byl ovšem zvolen nástroj Adminer. Adminer pochází od českého tvůrce a vznikl jako alternativa ke konkurenčnímu phpMyAdmin. Výhody, které autor Admineru uvádí, jsou například menší velikost a omezení se na pouze jeden PHP soubor. Dále lze s autorem souhlasit například v tom, že Adminer nabízí snazší a pokročilejší manipulaci s cizími klíči. Z ryze subjektivního hlediska, může být výhodou přehlednější interface, avšak to už záleží na posouzení každého jedince. Nutno podotknout, že z uživatelského hlediska, se oba popisované nástroje liší pouze drobnými rozdíly. Výběr zde ovlivňují zejména osobní

sympatie nebo zvyklosti. Bez zaujetí lze v tomto případě doporučit oba výše zmíněné nástroje, které práci s databází značně ulehčí. [7]

4. Volba technologické platformy

4.1 Volba datové platformy

Při výběru platformy pro datovou vrstvu webového portálu byla volba víceméně už předem jasná. Kromě objektivních hledisek zohledňujících obecnou vhodnost platformy, nelze vynechat ani subjektivní hlediska, kterým jsou například vlastní zkušenosti. V teoretické části tohoto textu jsou zmíněny tři dlouhodobě nejpoužívanější databázové systémy. Autor tohoto textu má přitom nejvíce zkušeností se druhým jmenovaným systémem, kterým je MySQL. Nutno podotknout, že zkušenosti jsou to zejména pozitivní. Druhým, tentokrát již objektivním hlediskem je rozsah projektu. Rozsah projektu je v tomto případě zcela běžný a pro potřeby tohoto portálu není MySQL nikterak limitující či nedostačující. Pokud tedy přihlédneme k výše zmíněným hlediskům a také ke zcela volným podmínkám distribuce MySQL, potom by tato volba neměla být nijak překvapivá.

4.2 Výběr platformy pro aplikační vrstvu

V teoretické části této práce bylo představeno hned několik dostupných moderních technologií pro vývoj backendové části webové aplikace. Každá z těchto technologií má své využití na různých projektech. Pro náš projekt, kterým je webový portál pro ZŠ a MŠ ve Stráži pod Ralskem, byly z pohledu vývoje backendu nejdůležitější tyto prvky: uživatelská přívětivost (jak pro návštěvníky, tak pro uživatele spravující obsah portálu), bezpečnost, spolehlivost a v neposlední řadě také rychlost vývoje.

Podobně jako při výběru vhodné platformy pro datovou vrstvu, i zde hrály roli faktory, jakými jsou zkušenosti s platformou či celková vhodnost platformy pro projekt jako takový. Tyto faktory opět velmi ovlivnily celkový výběr a je pravděpodobné, že různí vývojáři by ve stejné situaci použili různé platformy. Je opravdu obtížné obecně říci, která platforma je pro daný projekt nejvhodnější. V našem případě byla zvolena platforma Nette, což je PHP framework využívající MVC architekturu. K výběru této platformy nás vedly hned dva poměrně zásadní důvody. Prvním důvodem jsou naše již nabyté zkušenosti s tímto frameworkem. Druhým důvodem, který by při rozhodování rozhodně neměl být

opomenut, bylo naše přesvědčení o tom, že Nette skutečně splňuje všechny výše zmíněné požadavky, které jsme od vhodné platformy očekávali.

4.2.1 Představení Nette

V teoretické části této práce jsme si kromě jiných jazyků představili také PHP, přičemž zde nechyběla ani zmínka o existenci PHP frameworků. Na popis většího množství frameworků však v dané části nebylo tolik prostoru, tudíž si námi zvolený framework Nette představíme až zde.

Nette původně skutečně vzniklo jako kompletní Framework. V průběhu let však došlo k jeho rozdělení do samostatných PHP 7 komponent. Tyto komponenty lze mezi sebou libovolně kombinovat. Vhodnou kombinací si přitom každý může vytvořit ucelený framework na míru. Nette se na svých webových stránkách pyšní titulem 3. nejpopulárnější PHP framework na světě (v anketě „Best PHP Framework for 2015“). Na druhou stranu v jiných anketách se Nette nedostalo ani do výběru deseti nejlepších PHP frameworků. Objektivně lze skutečné kvality různých frameworků porovnat jen stěží a ankety svědčí pouze o jeho rozšířenosti mezi vývojáři. Nette je populární obzvláště mezi českými vývojáři, kteří dohromady tvoří vcelku početnou a aktivní online komunitu. Nespornou výhodou je také fakt, že užívání Nette není zpoplatněno a prakticky vůbec není omezeno.

> „Nette Framework je šířen jako svobodný software, aby ho kdokoliv mohl používat. Můžete si vybrat, zda vám lépe vyhovuje licence New BSD nebo GNU General Public License (GPL) ve verzi 2 nebo 3.
>
> Licence BSD je doporučena pro většinu projektů, jelikož je snadné ji pochopit a neklade téměř žádná omezení na to, co můžete s frameworkem dělat. Můžete Nette používat i v komerčních projektech. Pokud se však GPL hodí pro váš projekt lépe, zvolte ji. Přičemž není potřeba nikoho informovat, jak jste se rozhodli. Vždy zachovejte původní copyrighty.“ [8]

Popsat kompletně veškerou funkcionalitu Nette by vydalo na samostatnou práci, přičemž by to bylo i zbytečné, neboť vše podstatné lze nalézt v online dokumentaci. Společně se tedy podíváme pouze na ty nejdůležitější komponenty a celkovou strukturu Nette

frameworku tak, abychom si přiblížili, jakým způsobem se v Nette programuje. V další kapitole se pak podíváme i na konkrétní příklady z programování v Nette.

4.2.2 Výbava Nette

Součástí Nette je i debugger Tracy, česky zvaný jako „laděnka“. Tracy je skutečně nepostradatelným pomocníkem při hledání chyb, které se často při programování v PHP podaří napáchat. Nalézt chybu bez jakéhokoliv nástroje může být zdlouhavé a náročné. Tracy, jež je špičkou mezi diagnostickými nástroji tak představuje velice mocný nástroj, který programátorovi ušetří spoustu času a výrazně přispěje k vyšší produktivitě práce.

4.2.3 Popis architektury MVC

Nette vychází z architektury MVC (z angl.: model, view, controller). MVC je základní třívrstvá architektura, která má v současnosti již více implementací. Konkrétně Nette nahradilo „controller“ za takzvaný „presenter“ a někdy se celá architektura Nette nazývá spíše jako MVP.

4.2.3.1 Model

Model tvoří datový a funkční základ celé aplikace. Je v něm obsažena aplikační logika. Model přímo pracuje s daty v databázi. Model neví o existenci ostatních vrstev aplikace. O existenci modelu ví pouze presenter, který volá jeho funkce.

4.2.3.2 View (pohled)

Pohledová vrstva aplikace má na starost vykreslení požadavku uživatele. View většinou používá šablonovací systém a ví jak zobrazit požadované komponenty. Nette má svůj vlastní šablonovací systém, který nese název Latte. Latte umožňuje psát do šablony různá makra, jejichž syntaxe přitom vychází z PHP a je tedy velmi přívětivá. Skrze tato makra lze psát různé podmínky, cykly, odkazy, vykreslovat formuláře a v neposlední řadě také vnořovat šablony do sebe.

4.2.3.3 Controller (Presenter)

Presenter zpracovává požadavky uživatele a volá příslušné funkce modelu. Výsledek poté předá do pohledové vrstvy, která vše vykreslí. Presenter tedy působí jako prostředník, který komunikuje s oběma krajními vrstvami celé aplikace.

4.2.4 Závěr o Nette

Výhody, které pro vývojáře webových stránek framework Nette představuje, lze shrnout následovně. Na prvním místě Nette napomáhá k vyšší produktivitě práce při psaní kódu. Dalšími přínosy jsou například důraz na bezpečnost a také čistší a přehlednější kód, ke kterému Nette programátora vede.

5. Návrh a realizace webových stránek

Projektem, který jsme si s kolegou Bc. Martinem Šourkem zvolili, je tvorba moderního webového portálu pro Základní a mateřskou školu ve Stráži pod Ralskem. Práci jsme si rozdělili dle klasického schématu na backend a frontend. Mým úkolem bylo postarat se o backendovou část celé aplikace. Zde by bylo zřejmě vhodné připomenout, co se pod těmito dvěma pojmy skrývá a jak je od sebe odlišit. V prvé řadě je nezbytné říci, že obě tyto části dohromady tvoří celou aplikaci a jsou prakticky nedělitelné. Jinými slovy jedna bez druhé nemůže fungovat.

Frontend je ta část aplikace, kterou uživatel vidí a s níž interaguje. Frontend zahrnuje obrázky, tlačítka, menu, formuláře apod. Je ovšem nutné si uvědomit, že jako frontend jsou považovány pouze vizuální prvky jako takové – tedy bez funkční vrstvy. Frontend je pouze povrchovou vrstvou, která v hierarchii stojí nad backendem.

Backend je ta část webové aplikace, která je uživateli skryta, avšak zpracovává jeho pokyny. Backend má tři základní úkoly: 1) přistupovat k informacím, které uživatel požaduje skrze prostředí frontendu, 2) slučovat a transformovat tyto informace, 3) vracet tyto informace v požadovaném formátu zpět uživateli. Zjednodušeně řečeno backend zajišťuje veškerou funkcionalitu, kterou uživatel od aplikace očekává a takzvaně dává frontendu „život“. [9]

5.1 Pracovní postup a stanovení požadavků

Na počátku celého projektu bylo zapotřebí navrhnout uživatelské rozhraní a celkový vzhled stránek. V této fázi jsme vycházeli z potřeb uživatelů a také z původních webových stránek. Modernizace webového portálu probíhala v duchu vytvoření přehlednější a vizuálně nápaditější verze původního portálu. Jedním z požadavků bylo ponechání informační hodnoty, a tudíž jsme nemohli opomenout žádné již existující prvky. Strukturu portálu jsme upravili tak, že jsme přesunuli některé důležité informace na hlavní stránky. Zároveň jsme přidali funkcionality nové, kterými jsou například aktuality, avšak základní struktura webových stránek se obešla bez větších změn. Co se změnilo, bylo prostředí webu, responzivita stránek, databázová struktura a administrační prostředí.

Prvotní fáze návrhu uživatelského prostředí webu spadá zejména do frontendové části vývoje a tudíž byla téměř výhradně v kompetenci mého kolegy. V této fázi návrhu jsem s kolegou konzultoval navržená řešení a sděloval mu případné připomínky.

Ve chvíli, kdy bylo uživatelské prostředí webu alespoň zevrubně navrženo, započal proces návrhu backendu celé aplikace. V prvé řadě jsme se potýkali s tím, jakou bude mít portál datovou strukturu. Dále bylo nutné navrhnout vhodnou databázi tak, aby vyhovovala požadavkům, které jsme si stanovili v předchozí fázi. Zde ovšem vývoj backendu ještě nekončí, neboť data, která jsou uložena v databázi je nutné vhodným způsobem získávat a předávat do prezentační vrstvy. Posledním úkolem tedy bylo navrhnout aplikační vrstvu tak, abychom docílili výsledku v podobě plnohodnotného školního portálu se vším, co jsme v prvotní fázi navrhli.

5.2 Návrh datové vrstvy

Jak už jistě víte, backend lze dále dělit na dva dílčí celky, kterými jsou datová a aplikační vrstva tak, jak byly představeny již v teoretické části této práce. Datová vrstva by již z principu měla být navržena dříve než vrstva aplikační, neboť aplikační vrstva přistupuje k datům v této vrstvě a ta dále zpracovává či modifikuje. Z tohoto důvodu bude nejprve představen průběh návrhu a realizace datové vrstvy.

5.2.1 Stanovení požadavků na funkcionalitu

Předtím než bylo možné se pustit do návrhu databázové struktury, bylo potřeba ujasnit si veškeré požadavky. Konkrétně šlo zejména o to, jaký typ dat bude potřeba v databázi uchovávat a jaký obsah budou moci správci webu přidávat a upravovat. Z těchto dvou zásadních otázek dále vyplývají ještě další, a to a jakým způsobem budou veškerá data v databázi strukturována a jakým způsobem bude možné daný obsah webu z administračního prostředí upravovat a přidávat.

Při návrhu jsme dbali hlavně na to, aby se uživatelé na webu co nejlépe orientovali. Z tohoto důvodu jsme do hlavního menu, které dále vyjíždí, zabudovali sekce hlavních stránek. Sekce představuje v HTML stránce oddíl, jehož začátek je označen námi

zvoleným unikátním id. Je vhodné dodat, že id je HTML atribut, který slouží k identifikaci kteréhokoliv prvku na stránce. Zbývající část tohoto řešení je dále rozebrána v datovém modelu, konkrétně v kapitole „sekce hlavních stránek“.

Některé prvky, které bychom zprvu očekávaly, nakonec vůbec nebylo nutné do databázové struktury zahrnout. Příkladem budiž rozvrhy hodin, předměty a učebny. K tvorbě rozvrhů totiž výborně slouží systém Bakaláři, který škola již dlouhodobě používá.

Zároveň jsou v tomto systému údaje o žácích, učitelích a dalších zaměstnancích školy. Některé z těchto údajů škola doposud prezentovala na svých dosavadních webových stránkách a i my jsme tedy požadovali tyto údaje nadále zobrazovat. V tuto chvíli však již nedávalo smysl, abychom nutili zaměstnance školy ukládat všechna data nejprve do systému Bakaláři a poté znovu do našeho administračního systému. Na výběr jsme měli ze dvou možností. První variantou byla poměrně komplikovaná cesta, která spočívala v napojení systému na druhou databázi systému Bakaláři. My jsme pro jednoduchost implementace zvolili variantu druhou. Vybraná varianta spočívá v automatizovaném sběru dat ze systému Bakaláři a importu vybraných dat do naší databáze.

Co se týče uživatelů administračního prostředí, ze zkušenosti víme, že obsah webu spravují pouze někteří zaměstnanci školy, kteří jsou tímto úkolem pověřeni. Pro tyto uživatele byly zřízeny uživatelské účty s plnými administračními právy. Uživatelské účty mohou v budoucnu přibývat a to zcela jednoduše skrze administrační prostředí. Mysleli jsme i na to, že některým uživatelům by mohla být přidělena práva pouze do některé části administračního prostředí. Z tohoto důvodu bylo nutné přiřadit uživatelům role, přičemž každá role představuje konkrétní skupinu oprávnění. Poslední záležitostí, na kterou je třeba v případě většího počtu uživatelů myslet, je odpovědnost za úkony, které v administraci provede. Je tedy žádoucí zaznamenávat autora veškerých změn, které v systému byly provedeny. Díky tomu je následně možné zpětně zjistit, kdo co přidal, smazal nebo upravil. Spravující uživatelé tak za své činy nesou odpovědnost a nemohou změny provádět anonymně.

5.2.2 Návrh datového modelu

Nyní, když už máme určitou představu o tom, co vše bude nutné do databáze zahrnout, můžeme se přesunout k návrhu konceptuálního datového modelu. V této kapitole jsou detailně rozebrány jednotlivé entity datového modelu. Vlastnosti entit, které se nazývají atributy, budou pro přehlednost v následujícím textu odlišeny *kurzívou*. Jednotlivé entity jsou zde představeny z pohledu administrátora webového portálu.

5.2.2.1 Uživatel

První entitou v datovém modelu, je uživatel. Uživatel představuje zaměstnance školy s přístupem do administračního prostředí webu. Jako každá jiná entita musí mít uživatel své *id*, které představuje identifikátor neboli primární klíč v tabulce. Pokud nebude v textu uvedeno jinak, pro *id* vždy používáme celočíselný datový typ s vlastností automatického přírůstku. Dalším nezbytným atributem je údaj sloužící k přihlášení. My jsme pro tyto potřeby zvolili *e-mail*. E-mail je textový řetězec o maximální délce 128 znaků, což by mělo být více než dostačující. Díky tomuto řešení máme informaci o e-mailu uživatele, což může být v budoucnu užitečné v některé části systému (může sloužit například k odesílání automatických upozornění apod.). Pro uživatele je toto řešení také přívětivé, neboť si nemusí vymýšlet originální uživatelské jméno nebo si pamatovat nějaké náhodně vygenerované. Posledním nezbytným atributem uživatele je *heslo*, které rovněž slouží k přihlášení. Pro *heslo* jsme zvolili stejný datový typ jako pro *e-mail*, tedy textový řetězec, avšak o délce pouze 64 znaků. *Heslo* se samozřejmě neukládá tak, jak jej uživatel napíše, nýbrž jako hash (výstup hašovací funkce), který je před vložením do databáze vytvořen. V našich požadavcích jsme již zmínili také uživatelské *role*. *Role* slouží k tomu, aby bylo možné přidělovat různým uživatelům různý rozsah oprávnění. Pro tyto potřeby jsme zvolili atribut tzv. výčtového typu. Výčtový typ je datový typ tvořený konečnou omezenou množinou pojmenovaných hodnot. V seznamu máme prozatím pouze jedinou položku, kterou je „admin“, avšak do budoucna lze počítat s rozšířením. Mezi atributy nechybí ani skutečné *jméno* uživatele. *Jméno* uživatele je zde z důvodu, pokud bychom na některém místě chtěli zobrazovat údaj, kdo daný obsah přidal nebo upravil. Datový typ *jména* je opět běžným textovým řetězcem. Délka řetězce je přitom stanovena na maximálně 64 znaků.

5.2.2.2 Stránka

Stránka je zřejmě nejpodstatnějším prvkem celého portálu. Primárním klíčem stránky je opět *id*. Dále má stránka *jméno*, které se zobrazuje hned na dvou místech. Prvním takovým místem je navigace, ze které na ni vede odkaz a druhým je samotný nadpis stránky. Volitelným atributem je *titulek* stránky, který se zobrazuje v prohlížeči. Oba atributy jsou textové řetězce o maximální délce 64 znaků. Následující tři atributy slouží k identifikaci stránky ze strany aplikační vrstvy. Jedná se o *název* presenteru, *akci* presenteru a *URL* adresu. Datovým typem těchto atributů je opět textový řetězec. Všechny tři atributy jsou přitom generovány v aplikační vrstvě a uživatel je nevyplňuje. Posledním atributem je tzv. *rodičovská stránka* (nebo také nadstránka), která představuje stránku umístěnou v hierarchii nad danou stránkou. Datovým typem tohoto atributu je celé číslo a představuje cizí klíč ze stejné entity.

5.2.2.3 Sekce hlavních stránek

Pět hlavních stránek obsahuje na jedné stránce více prvků, které nazýváme jako sekce. Tyto sekce jsou v HTML kódu pojmenovány v patřičném elementu skrze HTML atribut id (neplést s atributem enitity *id*!). Tuto entitu nemůžou administrátoři webu spravovat, avšak bylo ji nutné vytvořit, aby se uživatelé mohli na poměrně dlouhé stránce lépe orientovat. Prvním atributem sekce je *id*. Dále je zde cizí klíč v podobě *id stránky*, k níž se sekce vztahuje. Dalším atributem je pochopitelně *název sekce*, což je textový řetězec. Tento řetězec odpovídá názvu (id) elementu v HTML kódu a jeho maximální délka je stanovena na 64 znaků. Posledním atributem této entity je *titulek*, který slouží jako nadpis daného odkazu na tuto sekci. *Titulek* je opět textový řetězec se stejnou maximální délkou.

5.2.2.4 Obsah stránek

Tato entita uchovává obsah jednotlivých stránek. Celočíselnými atributy jsou *id* obsahu a *id stránky*. *Id* obsahu je opět primárním klíčem. *Id stránky* je cizím klíčem odkazující na konkrétní stránku, které obsah náleží. Dále je zde atribut *typ obsahu*, který je výčtového typu. Ve výčtu jsou prozatím pouze dvě hodnoty, které odlišují obsah primární a sekundární. Posledním a zároveň nejdůležitějším atributem je samotný *obsah*. *Obsah* je typu textové pole a slouží k uložení HTML kódu.

5.2.2.5 Článek

Článek je v podstatě totéž jako stránka, avšak nabízí administrátorovi více možností. Články mohou tvořit na webu buďto aktualitu (takový článek informuje o události, která již proběhla) nebo nadcházející událost. K tomu, aby bylo možné rozlišit, do které kategorie článek zařadit, je zapotřebí znát *datum* konání události, který je v databázi uložen standardně jako datový typ datum a čas. Dále u aktualit rozlišujeme, zda se týkají mateřské či základní školy. K tomuto účelu slouží atribut *kategorie*, který je cizím klíčem odkazujícím se na entitu stránky. Tento atribut může nabývat i hodnoty „NULL“ a to v případě, že je článek obecného rázu a týká se jak mateřské, tak základní školy. Článek má i dva atributy typu textový řetězec. Jsou jimi *titulek* a *URL adresa*. Při vytvoření článku se zaznamenává i *datum vytvoření* a *autor článku*. *Autor článku* je cizím klíčem uživatele. Ke článku je možné přiřadit i *obrázek*, který slouží jako miniatura v náhledu aktuality. *Obrázek* je cizím klíčem entity soubor. Posledním atributem je atribut booleanského typu s názvem *publikováno*. Díky tomu je možné články před publikováním (nebo i poté) skrýt.

5.2.2.6 Blok článku

Celý obsah článku je uložen v blocích. Bloky na rozdíl od entity, která ukládá obsah stránek, dovolují více customizace. Blok článku má v prvé řadě své *id* a *id článku*. Tedy primární a cizí klíč. Dále obsahuje dva atributy, které určují obsah konkrétního bloku. První z nich je *typ*. *Typ* bloku je výčtového typu a může nabývat pouze dvou hodnot. *Typ* bloku určuje, zda je v bloku uložen obrázek nebo formátovaný HTML text. V případě, že se jedná o obrázek, je v dalším atributu s názvem *hodnota,* uloženo id souboru. Jelikož tento atribut sdílí s bloky HTML textu, je id souboru uloženo rovněž ve formě textového řetězce. O zbytek se postará aplikační vrstva. To že v databázi není tato hodnota uložena jako cizí klíč i když jím fakticky je, nepředstavuje v praxi žádné potíže a jedná se o již ověřené funkční řešení. Poslední dva atributy určují vertikální a horizontální pozici daného bloku. Tyto atributy jsou nazývány ř*ádek* a *sloupec*. Oba atributy jsou celočíselného typu a nemohou nabývat hodnoty „NULL“.

5.2.2.7 Soubor

Soubor je entitou, která zajišťuje bezpečné ukládání a správné zobrazování všech souborů na serveru. Soubor je klíčovou entitou v našem systému, neboť jej využívá mnoho jiných entit jako například výše uvedený článek a blok článku, ale i další entity, které jsou

představeny dále v textu. Nejdůležitějšími atributy jsou *hash* a *přípona*. Tyto atributy v aplikační vrstvě slouží k identifikaci souboru a jedná se o běžné textové řetězce. Nahraný soubor dále popisují atributy, jako jsou *velikost, jméno* a *typ* souboru. *Velikost* souboru je vyjádřena celým číslem. *Jméno* a *typ* souboru jsou opět textové řetězce. Atribut *typ* slouží pouze jako vnitřní identifikátor. Tento údaj nám sděluje, z jaké části administrace byl soubor nahrán. Posledními atributy souboru jsou údaje o nahrání souboru, tedy *datum a čas nahrání* souboru a *id uživatele*, který soubor nahrál.

5.2.2.8 Dokumenty

Soubory ke stažení uchováváme v entitě dokumenty. Jedná se o entitu, která rozšiřuje entitu soubor. Entita dokument obsahuje tři celočíselné atributy a pouze jeden textový. *Název dokumentu* je textovým řetězcem o délce maximálně 64 znaků. Atributy celočíselné jsou vlastní *id* dokumentu (primární klíč), *id souboru* a *id stránky* (oba cizí klíče). *Id stránky* určuje, na které stránce bude dokument ke stažení.

5.2.2.9 Varianty obrázku

Zbylé soubory jsou obrázky a fotografie. Ty se automaticky ukládají do několika velikostních variant. Kromě vlastního *id* má tato entita i atribut odkazující na *id souboru*. Dále obsahuje dva atributy popisující *výšku* a *šířku* obrázku a to jako celé nenulové číslo. Posledním atributem je *šířka zobrazení*, která rozhoduje, kdy má být která varianta použita. *Šířka zobrazení* je opět celé nenulové číslo.

5.2.2.10 Fotogalerie obrázků

Obrázky a fotografie mohou být sdružovány do fotogalerií. Ta má své *id, jméno, čas vytvoření, čas poslední modifikace* a *id článku*, ke kterému má být přiřazena. Jednotlivé fotografie jsou do fotogalerie přiřazeny skrze spojovací tabulku. Spojovací tabulka obsahuje pouze dva atributy, které nesou *id souboru* a *id fotogalerie*.

5.2.2.11 Jídelníček

V této entitě jsou zaznamenána jídla pro každý den. Každé jídlo má své jedinečné *id*. Atribut *datum* slouží k přiřazení jídla ke konkrétnímu dni a je ukládán jako obyčejný datum. Atribut *typ* rozlišuje (v současnosti) tři druhy jídel: svačinu, oběd a přesnídávku. Atribut *typ* je tedy výčet o těchto třech položkách. Poslední dva atributy jsou textové řetězce. První z nich je celý *název/popis* jídla a druhý je *výčet alergenů* v jídle.

5.2.2.12 Ceník jídel

Vzhledem k tomu, že u ceníku jídel lze do budoucna předpokládat změny, zavedli jsme jej také jako entitu v databázi. Ceny se neliší dle jídla, nýbrž dle strávníka. Přesněji řečeno každý strávník spadá do určité kategorie, která platí jednotnou cenu za stravování. Každá kategorie má své *id*, *název/popisek*, *cenu jídla*, *poznámku* a symbolický *obrázek*. *Titulek* a *poznámka* jsou textové řetězce, *cena* je vyjádřena celým číslem a *obrázek* je cizím klíčem souboru.

5.2.2.13 Třída

Entita třída slučuje žáky a třídního učitele. Každá třída má své *id* a *označení* („1.A“ apod.). *Označení* (název) třídy je uloženo jako běžný textový řetězec. Podstatný je i další atribut, který rozlišuje, zda se jedná o třídu mateřské nebo základní školy. Tento atribut nazýváme jako *typ třídy* a jedná o výčet, který obsahuje dvě hodnoty: základní a mateřská.

5.2.2.14 Žák

Žák je jednoduchá entita, která má pouze své *id, křestní jméno, příjmení* a *třídu*, do které žák náleží. *Křestní jméno* a *příjmení* jsou textové řetězce a *třída* je cizím klíčem entity třída.

5.2.2.15 Učitel

Učitel má naprosto identické atributy jako žák. Rozdílem je, že *třída* učitele může nabývat hodnoty „NULL“. Ne každý učitel totiž musí být třídním učitelem nějaké třídy. Oproti žákovi je entita učitel ještě rozšířena o atribut *titul*, který je zaznamenáván jako textový řetězec.

6. Realizace aplikační vrstvy webových stránek

V tuto chvíli, kdy už jsme měli vcelku jasnou představu o tom, jak by měl výsledný produkt vypadat, zbývalo jen vše naprogramovat. Zde se ovšem nabízí otázka, kde vlastně začít? Obecná pravidla, která by nám dala jasnou odpověď na tuto otázku, prakticky neexistují. Za předpokladu, že je v roli vývojáře celé aplikace pouze jeden člověk, je možné začít v podstatě kdekoliv. Takový vývojář si může nejprve napsat metody (takto se nazývají funkce v OOP) pro získávání dat z databáze, ovšem stejně tak může začít detailním návrhem šablon.

Pokud je vývoj rozdělen na frontend a backend tak, jak je to běžné, může být situace poněkud komplikovanější. Frontendista může navrhnout vizuální podobu stránek, avšak pokud není vývoj backendu započat, nemůže přímo vytvářet dílčí Latte šablony, jež dohromady utváří vizuální podobu stránek. Toto byla první překážka, na kterou jsme s kolegou narazili. Řešením k tomuto problému bylo vytvoření hlavní statické webové stránky frontendistou, odkud jsme později převedli veškeré potřebné prvky do Latte šablon, které postupně vznikaly tak, jak se utvářela celá aplikace.

6.1 Stručný popis návrhu aplikační vrstvy

Při návrhu aplikační vrstvy jsme vycházeli z grafického návrhu. Z tohoto návrhu bylo zřejmé, že domovská stránka a pět hlavních stran budou mít každá svůj vlastní specifický vzhled. Připravili jsme si tedy nejprve hlavní šablonu a k ní šest různých šablon (pro každou z hlavních stránek), které plní místo obsahu (prostor mezi navigací a patičkou webu). Dále se vytvořily univerzální šablony také pro všechny ostatní stránky a články. Při tomto procesu se také některé prvky, které se používají ve více šablonách, umístily do zvláštních šablon a na požadovaném místě načetly odtud.

V šablonách jsme potřebovali zobrazit spoustu dat dynamicky z databáze a tak bylo nutné napsat patřičné metody v presenterech a službách, jež jsou součástí modelu. Před tímto krokem ještě proběhlo vytvoření tzv. entit a repositářů v modelu. Ty odpovídají entitám, které byly představeny v kapitole o datové vrstvě.

Takto ve zkratce probíhal celý vývoj aplikační vrstvy webových stránek. V následujících podkapitolách jsou jednotlivé vrstvy architektury MVC/MVP detailněji popsány a to včetně příkladů z praxe.

Na konci této kapitoly je pro lepší představu uveden jednoduchý příklad, na kterém je vyobrazen celý proces od získání dat z databáze až po jejich zobrazení na stránce. Takovýmto způsobem přitom bylo potřeba obstarat mnoho dat od položek v navigaci, přes názvy stránek, obrázky, textové dokumenty až po obyčejné texty a další prvky. Nutno podotknout, že identickým způsobem probíhal také vývoj aplikační vrstvy pro administrační část celé aplikace. Zde bylo navíc nutné zhotovení formulářů, metod pro ukládání dat do databáze a dalších funkcionalit, které jsou blíže představeny v kapitole o administračním prostředí.

6.2 Dělení aplikační vrstvy (model MVC/MVP prakticky)

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

Obrázek 6: Doporučená adresářová struktura Nette
Zdroj: https://doc.nette.org/cs/2.4/presenters

Z obrázku je patrné, že adresářová struktura Nette projektu odpovídá architektuře MVP. Konfigurace projektu není obtížná a návodů, které tuto problematiku popisují, je i v českém prostředí internetu nespočet. Z tohoto důvodu se následující kapitola bude věnovat pouze té zajímavější části celé aplikace. To nejdůležitější se nachází v adresářích „*model*“ a „*modules*“ (v obrázku výše „modules“ odpovídá „presenters“).

6.2.1 Model

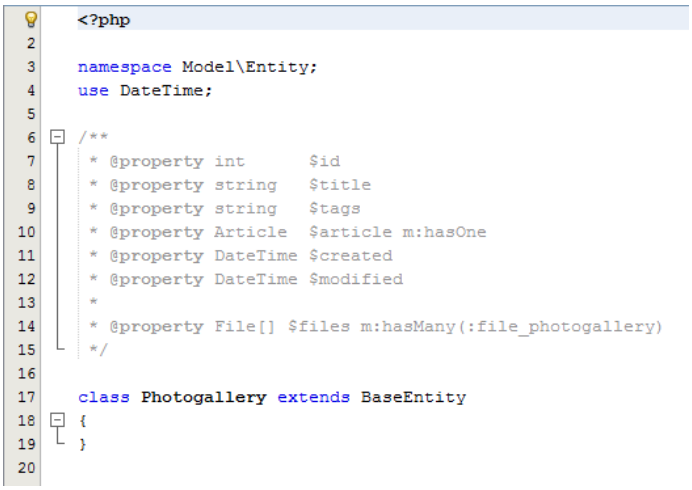
Adresář Model pochopitelně představuje modelovou vrstvu celé aplikace. Tento adresář je dále rozdělen do dalších podadresářů s názvy „Entity“, „Repository“ a „Service“.

6.2.1.1 Entita

V tomto adresáři se nachází všechny entity, s nimiž je potřeba nějakým způsobem pracovat. Každá entita je uložena v samostatném souboru typu PHP. Struktura každého souboru je obecně následující. Soubor začíná značkou označující začátek PHP kódu: „<?php“. Poté je nejprve uveden tzv. jmenný prostor třídy. Ten umožňuje používat stejně nazvané třídy z různých knihoven, aniž by docházelo ke kolizi. Následují atributy entity, které ovšem zapisujeme tak, že se tváří jako pouhý komentář. V další části souboru již začíná definice třídy, která je pojmenovaná podle entity, již představuje. Tato třída může obsahovat vlastní konstanty a metody. Často se setkáme také s „prázdnými“ třídami, které si vystačí s metodami, jež dědí od základní entity. Tyto třídy nám umožňují pracovat s entitami jako s objekty a přistupovat tak k jejím atributům.

```
20  class Page extends BaseEntity
21  {
22      /** @var LeanMapper\Connection */
23      public $conn;
24
25      public function __construct($arg = null, Connection $conn = NULL)
26      {
27          parent::__construct($arg);
28          $this->conn = $conn;
29      }
30
31      public function getContent($type)
32      {
33          foreach ($this->contents as $c) {
34              if ($type == $c->type) {
35                  return $c;
36              }
37          }
38          return NULL;
39      }
40
41      public function getChildPages()
42      {
43          return $this->conn->query(
44              'SELECT `page`.*
45              FROM `page`
46              WHERE `page`.`parent_page` = %i
47              ', $this->id);
48      }
49
50      public function getSections()
51      {
52          return $this->conn->query(
53              'SELECT *
54              FROM `page_section` s
55              WHERE s.`page_id` = %i
56              ', $this->id)->fetchPairs('section', 'title');
57      }
58  }
59
```

Obrázek 7: Třída entity stránka obsahující tři vlastní metody a konstruktor
Zdroj: vlastní



```php
<?php

namespace Model\Entity;
use DateTime;

/**
 * @property int      $id
 * @property string   $title
 * @property string   $tags
 * @property Article  $article m:hasOne
 * @property DateTime $created
 * @property DateTime $modified
 *
 * @property File[] $files m:hasMany(:file_photogallery)
 */

class Photogallery extends BaseEntity
{
}

```

Obrázek 8:Třída entity fotogalerie a její atributy
Zdroj: vlastní

6.2.1.2 Repositář

Repositáře jsou obdobně jako entity reprezentovány jednotlivými PHP soubory a slouží k práci s entitami v databázi. Repositář představuje tabulku v databázi. Skrze repositáře lze snadněji pracovat s daty uloženými v databázi. Název entity i repositáře odpovídá názvu tabulky v databázi. Soubory s repositáři a uvnitř definované třídy tedy obsahují tento název doplněný o označení „Repository“. Samotná třída repositáře může také obsahovat vlastní metody, avšak nebývá to tak časté. Základní repositář, od kterého ostatní dědí, nabízí metody pro přidávání a editaci řádku, získání řádku, vrácení nalezených řádků a také pro napsání vlastního dotazu do databáze.

6.2.1.3 Služba

Služby jsou rovněž rozděleny do samostatných PHP souborů. Za běžných okolností služby pro přehlednost dělíme dle obdobného principu jako repositáře a entity. Jejich počet však může být i menší než počet tříd entit nebo repositářů. Každá služba totiž může přistupovat k neomezenému počtu repositářů a tak v případě, že spolu entity přímo souvisejí, je vhodné vytvořit jednu službu, jež repositáře společně sdílí.

Obsahem těchto tříd jsou metody, které jsme si pro své potřeby napsali. Tyto metody obvykle obstarávají veškerou práci s daty v databázi jako je jejich čtení, přidávání, editace a mazání. Metody služeb k datům přistupují skrze repositáře, ze kterých volají patřičné metody. Metody služeb lze přitom volat přímo z presenteru, případně také z třídy jiné služby nebo entity. Všechny služby a repositáře musí být registrované v konfiguračním souboru config.neon.

6.3 Moduly

Zde se náš projekt mírně liší od výše uvedené doporučené adresářové struktury. Stále jde ovšem o vcelku běžnou strukturu. Jediným rozdílem je, že namísto adresáře „presenters“, je použit adresář „Modules“, který dále dělí presentery do dvou podadresářů: „admin“ a „front“. Každý z těchto dvou tzv. modulů přitom představuje odlišnou část aplikace (administrační a veřejnou).

6.3.1 Presentery

Presentery slouží hned k několika účelům. V prvé řadě v presenteru jsou napsány akce, které uživatel volá skrze URL adresu, jež je přeložena do konkrétního požadavku.

6.3.1.1 Render metody

Za nejdůležitější můžeme považovat metody, které stojí za samotným vykreslením stránky. K tomu jsou určeny tzv. render (vykreslovací) metody. Metoda *renderDefault* vykresluje do výchozí šablony s názvem „default.latte“. Pro vykreslování do jiných šablon máme obvykle napsané render metody nesoucí shodné jméno se jménem šablony.

Úkolem těchto render metod je v prvé řadě vykreslení stránky s požadovanou šablonou. Při tomto procesu je často nutné dynamicky získat z databáze data, která mají být vypsána v šabloně. V metodě render je možné tato data získat skrze zavolání požadované metody patřičné služby, uložit je do proměnné a tu poté použít v šabloně k jejich vykreslení. Celý proces je přitom detailněji popsán v kapitole s konkrétním příkladem.

6.3.1.2 Before render metody

Volají se ještě před render metodami a lze je využít, pokud se například nabízí předat data společná pro více pohledů.

6.3.1.3 Action metody

Slouží jako alternativa k render metodám, pokud není potřeba nic vykreslit a metoda má například přesměrovat uživatele po přihlášení.

6.3.1.4 Handle metody (signály)

Na tyto metody je možné se odkazovat přímo z šablony, kde mohou například představovat úkon smazání řádku v databázi. Na stejné úrovni jsou i metody obsluhující zpracování formuláře.

6.3.1.5 Tvorba komponent

V presenterech lze vytvářet i formuláře jako takzvané komponenty, které lze snadno vykreslit v pohledových šablonách.

6.4 Šablony

V naší aplikaci se nachází dva adresáře se šablonami „templates“, každý pro jinou část aplikace (administrační a veřejnou). Z hlediska architektury MVP šablony představují pohled neboli „view“. Nejdůležitější šablonou je hlavní šablona celého modulu. Ta nese název „@layout.latte“ a nachází se přímo v adresáři „templates“. V našem případě máme opět dvě tyto šablony, neboť administrační prostředí nesdílí grafický základ s veřejnou částí webu.

Tato šablona tvoří jakýsi obal pro všechny stránky, které se v daném modulu generují. Hlavní šablona je velmi podobná obyčejnému HTML dokumentu. Rozdílem jsou některé použité syntaxe, které běžné HTML nezná. Důležité je například makro *{include content}*, které je umístěné na místě, na kterém se dynamicky mění zobrazovaný obsah stránky. Na toto místo je posléze vložena již konkrétní Latte šablona podle toho, na jaké adrese se uživatel nachází.

Ostatní šablony, které představují obsah stránky, se nacházejí v podadresářích, jež sdílejí název s názvy jednotlivých presenterů. V každém takovém adresáři je obvykle alespoň jedna šablona s názvem „default.latte“, která představuje „vnitřní“ šablonu (umístěnou do obalující šablony „layout.latte“) pro výchozí akci presenteru. Tyto šablony začínají značkou *{block content}*, která značí, kde začíná obsah. Na rozdíl od hlavní šablony neobsahují tagy pro začátek HTML dokumentu, hlavičku apod., neboť tato část dokumentu už je obsažena zde.

6.5 Praktický příklad vykreslení stránky

Tato kapitola demonstruje, jak probíhá celý proces od získání dat z databáze až po jejich konečné vykreslení na stránce.

Řekněme, že chceme na domovské stránce vykreslit počet všech (publikovaných) aktualit. Začneme vytvořením služby „Articles“, kde si napíšeme metodu pro získávání počtu těchto článků. Tato metoda bude přistupovat ke článkům skrze repositář se články.

Ještě před definováním samotné třídy „Articles“ si na začátku souboru definujeme jmenný prostor dané třídy:

```
namespace Model\Service;
```

Poté můžeme definovat jmenné prostory, které budeme chtít použít:

```
use Model\Repository;
```

Tím zajistíme, že již dále v tomto souboru nemusíme uvádět celý jmenný prostor všech tříd v prostoru „Model\Repository“ a stačí namísto něho uvádět pouze „Repository“ (tento krok je vhodný zejména pokud daný jmenný prostor použijeme alespoň dvakrát).

Třídu „Articles“ definujeme následujícím způsobem (tečky zde nahrazují obsah třídy):

```
class Articles extends \Nette\Object {…}
```

V této třídě budeme potřebovat získávat data z repositáře „articleRepository, který si musíme nejdříve předat do konstruktoru následujícím způsobem:

```
/** @var Repository\ArticleRepository */

private $articleRepository;

public function __construct(Repository\ArticleRepository $articleRepository)

{
```

```
        $this->articleRepository = $articleRepository;

    }
```

Nyní již můžeme přistoupit ke psaní konkrétní metody. Pojmenujeme si ji třeba „getPublishedArticlesCount“. Tato metoda musí počet článků vrátit na místo, odkud je volána, tudíž obsahuje jazykový konstrukt „return“. V argumentu konstruktu „return“ můžeme vidět, jak přistupujeme do repositáře článků a voláme metodu „query“. Metoda „query“ je jednou ze základních metod abstraktního repositáře, od kterého ostatní repositáře dědí. Skrze tuto metodu můžeme vytvářet dotazy, které jsou následně přetvořeny na SQL dotazy. Celá metoda může vypadat zhruba následovně:

```
    public function getPublishedArticlesCount()

    {

        return $this->articleRepository->query()

            ->where('@status', 'publish')

            ->count();

    }
```

Volání metody „query“ vede vždy k vygenerování SQL dotazu. V našem případě se dotaz přetvoří do této podoby:

```
    SELECT COUNT(*)

    FROM (

    SELECT `article`. *

    FROM `article`

    WHERE (`article`.`status` = 'publish')) `data`
```

Nyní, když už dokážeme získat počet publikovaných článků, musíme tuto metodu odněkud zavolat a také ji někam uložit. Oba tyto úkony provedeme v render metodě presenteru. Nejdříve je nutné službu „Articles“ uložit do proměnné tak, abychom ji mohli v presenteru používat.

```
class PagePresenter extends BasePresenter

{

    /** @var \Model\Service\Articles @inject */

    public $articles;

}
```

Poté již můžeme metodu z presenteru zavolat následujícím způsobem:

```
$this->articles-> getPublishedArticlesCount()
```

My ovšem chceme toto číslo předat dále do šablony a tudíž si jej uložíme do proměnné „$this->template“. Výsledek může vypadat třeba takto.

```
public function renderHome()

{

    $this->template->numberOfArticles =

    $this->articles->getPublishedArticlesCount();

}
```

Nakonec už pouze do šablony s názvem „home.latte“ (umístěné v adresáři „PagePresenter“), vložíme na vhodné místo proměnou, do které jsme si získané číslo uložili:

```
{$numberOfArticles}
```

Tato proměnná je dostupná pouze v šabloně „home.latte“, neboť jsme si ji uložili do proměnné „$this->template“ ve vykreslovací metodě „renderHome“. Výsledkem je skutečně vypsání počtu publikovaných článků v databázi na požadovaném místě domovské stránky.

7. Zhodnocení výsledného řešení

Celou námi navrženou aplikaci lze rozdělit na dvě části a to podle toho, komu je umožněno do dané části aplikace přistupovat. Obě části jsou v této kapitole zevrubně představeny včetně různých funkcionalit a jejich technologického řešení.

První část aplikace je veřejně přístupná část webových stránek. Do této části přistupují všichni uživatelé, které lze souhrnně označovat jako návštěvníky webu. Návštěvníky webu je možné dále dělit také do dalších podskupin, avšak pro potřeby této práce to není nezbytně nutné.

Druhá část aplikace slouží k administraci veřejné části webu. Tato část webu je přístupná pouze registrovaným uživatelům, kteří představují správce neboli administrátory webu. Správcem webu se stává kdokoliv, komu je udělen přístup do této části systému. Zpravidla se jedná zejména o zaměstnance školy, avšak pověřena může být i jiná osoba. Nejčastěji se v roli správce vyskytují učitelé či ředitel školy. Dílčí části webu mohou být svěřeny například vedoucímu jídelny nebo dalším zaměstnancům.

7.1 Prostředí webu

Prostředí webu je detailně rozebráno v diplomové práci mého kolegy Bc. Martina Šourka. Z tohoto důvodu je zde tato část vynechána a následující kapitola se zaměřuje spíše na popis administračního prostředí včetně veškerých funkcionalit, které uživateli nabízí. Avšak jelikož administrační prostředí s prostředím webu přímo souvisí, objeví se v následujícím textu i částečný popis toho, jak je daná funkcionalita využita v prostředí webu.

7.2 Administrační prostředí

Od webových stránek školy se očekává, aby byl jejich obsah dynamický a mohli jej spravovat uživatelé, kteří nemají žádné zkušenosti s programováním ani se značkovacím jazykem HTML. Jedním z prvních požadavků tedy bylo vytvoření plnohodnotné administrační části webu pro správu obsahu. Návrh a realizace této

administrační části webových stránek byla kompletně svěřena do mých rukou. Následující kapitola se věnuje administračnímu prostředí a funkcionalitám, které uživateli nabízí. Mimojiné zde nechybí ani detailní popis realizace těchto funkcionalit, jejich vzhledu a skutečného účelu v prostředí webu.

7.2.1 Vzhled a pracovní plocha

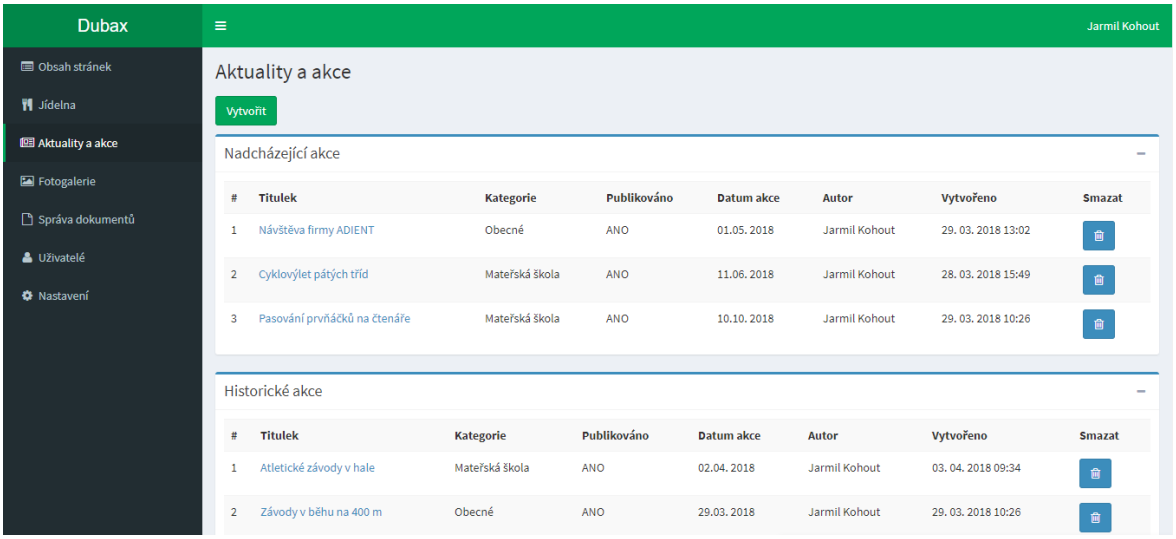
Při návrhu designu administračního prostředí jsem neměl v úmyslu experimentovat a zvolil jsem již osvědčené řešení. Inspirací mi byly jiné firemní projekty s administračním prostředím, na kterých jsem se rovněž podílel. Pro účely administračního prostředí totiž používáme téměř výhradně jediný design, ze kterého jsem vycházel. Jelikož se jedná o administrační část aplikace, nemělo by být překvapivé, že celkový vzhled je spíše prostšího rázu. Na poměry administračních prostředí však design působí relativně moderně a vyváženě. Při návrhu přitom bylo podobně jako v případě uživatelské části webu, využito prvků z Bootstrapu. Bootstrap je frontendový framework, který kodérovi umožňuje vytvářet vcelku rychle fungující rozvržení celé stránky. Rozložení prvků na stránce, které bylo zvoleno, je přitom pro administrační účely velmi typické.

Na horní liště je pouze hlavička s firemním logem, jménem přihlášeného uživatele a tlačítko pro odhlášení. Hlavní prvky menu jsou umístěny v navigaci na levém okraji. Důvodem k tomuto rozložení je v prvé řadě praktičnost, která je v případě administračního prostředí postavena nad formu.

Vertikální menu, které je umístěné na boku má oproti vrchnímu horizontálnímu menu několik předností. Za prvé je možné do něho vložit v podstatě neomezený počet položek. Za druhé i větší množství položek zůstává přehledné, pokud jsou rozumně seřazeny. Nakonec i formátování je snazší, neboť text se zpravidla vejde do požadované šíře a není třeba jej zalamovat, zmenšovat či jinak upravovat. Aby i naše navigace splňovala druhou zmíněnou vlastnost – přehlednost, bylo potřeba zvolit vhodné řazení položek v navigaci. Obvykle se používají následující varianty. První možnou variantou je řazení dle abecedy, dále je možné řazení dle důležitosti položek, potažmo dle frekvence jejich používání nebo dle logického uspořádání. My jsme zvolili poslední z výše zmíněných variant, neboť se

nám takto menu jeví nejpřehlednější a položek zde není tolik, aby se v nich uživatel ztrácel. Textové položky menu jsou navíc doplněny výstižnými ikonami.

Ve zbytku pracovní plochy se již nachází konkrétní nástroje určené pro správu obsahu. Použity jsou zejména formuláře pro přidávání a editování obsahu, ale také seznamy jednotlivých záznamů. Záznamem je myšlen libovolný objekt, který se pomocí administrace vkládá do databáze. Příkladem takového objektu je například článek, fotografie, fotogalerie, jídlo, stránka apod. V seznamech jsou vypsány základní údaje o těchto objektech. Tyto seznamy slouží hned k několika účelům. Za prvé seznamy poskytují uživateli přehled o tom, jaká data jsou uložena v databázi. Za druhé skrze tyto seznamy se uživatel může dostat i na detail záznamu, kde jej lze dále editovat. Třetím a zároveň posledním účelem, ke kterému lze tyto seznamy a náhledy využít, je vyhledávání a mazání nevalidních či již jinak nevyhovujících záznamů.



Obrázek 9: Seznam článků, který je rozdělen na nadcházející a historické události
Zdroj: vlastní

7.2.2 Představení formulářů a jejich prvků

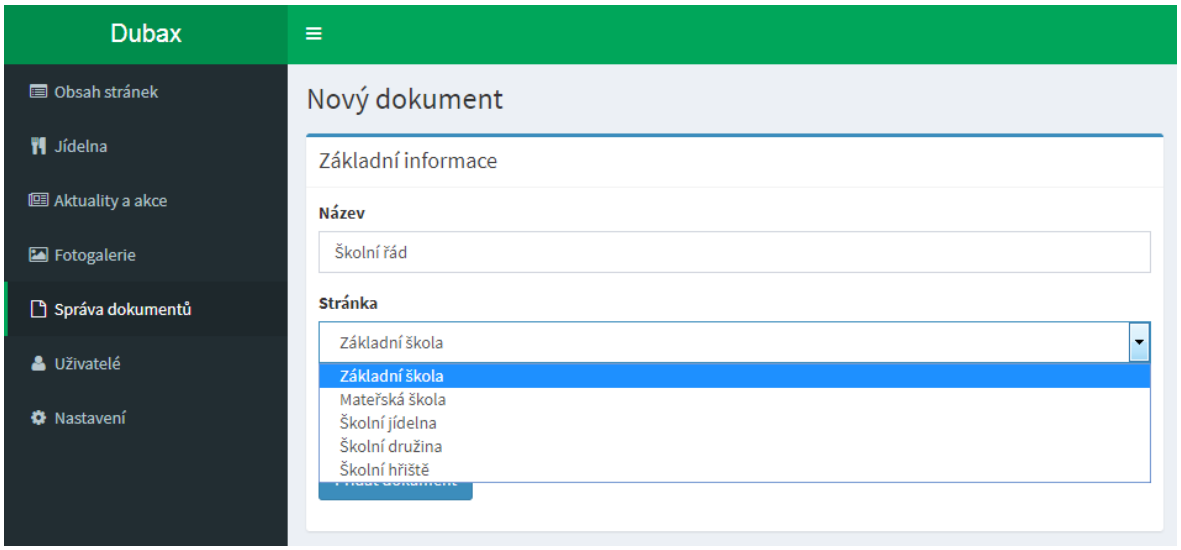
K přidávání i editaci jednotlivých záznamů slouží formuláře. Nette Framework oplývá vlastními formuláři Nette/Forms, ale lze využít i jiné doplňky. Nejčastěji používanou doplňkovou knihovnou pro formulářové prvky je knihovna Nextras/Forms. Nette formuláře poskytují základní formulářovou výbavu. Kromě běžných formulářových prvků nabízí také nastavení validačních pravidel, povinného vyplnění a výchozí hodnoty pro každý prvek. My v tomto projektu používáme klasické Nette formuláře rozšířené o výše

zmíněnou knihovnu Nextras/Forms. Důvodem je, že některé užitečné formulářové prvky formuláře Nette nenabízí. V následující části jsou uvedeny jednotlivé formulářové prvky, které v administrační části našeho portálu využíváme.

Základním a pravděpodobně nejčastěji užívaným prvkem je textové pole. Textové může pojmout veškeré znaky, které uživatel zadá a pracuje s nimi jako s textovým řetězcem. Tento prvek může mít i další vlastnosti jako jsou omezení na počet znaků. Nette dále umožňuje také použití validačních pravidel, která uživateli povolí vkládat pouze celá nebo desetinná čísla.

Další důvěrně známý formulářový prvek, který i zde často využíváme je výběrové pole neboli „select box“. „Select box“ slouží k výběru položky z předem zvoleného seznamu položek. „Select box“ využíváme ve chvíli, kdy potřebujeme mít úplnou kontrolu nad tím, co uživatel do systému zadá. Z pohledu databázové struktury, je nutné tento prvek použít, například při práci s cizími klíči.

Jako názorná ukázka může posloužit tento příklad z administračního prostředí. Uživateli je umožněno nahárt do systému nový PDF dokument, kterým může být například školní řád. Pro potřeby webových stránek je po uživateli požadována stránka, na které se posléze zobrazí odkaz ke stažení daného dokumentu. Ve formuláři pro nahrání dokumentu je tedy umístěn „select box“, ve kterém jsou vylistovány všechny stránky, které jsou uložené v databázi. Hodnoty, z kterých uživatel vybírá, je nutné nejprve nějakým způsobem získat. Stránky získané z databáze je potřeba následně uložit do jednoduchého pole, kde klíč tvoří „id“ stránky a hodnotu její název. Toto pole bude tvořit hodnoty, z nichž může uživatel vybírat. Uživateli se v seznamu zobrazí názvy stránek a formulář získá požadované „id“, které tvoří klíč pole. S tímto klíčem lze dále pracovat po zpracování formuláře a přitom jasně odkazuje na konkrétní stránku jakožto cizí klíč.



Obrázek 10: Formulářový prvek výběrové pole (angl. select box)
Zdroj: vlastní

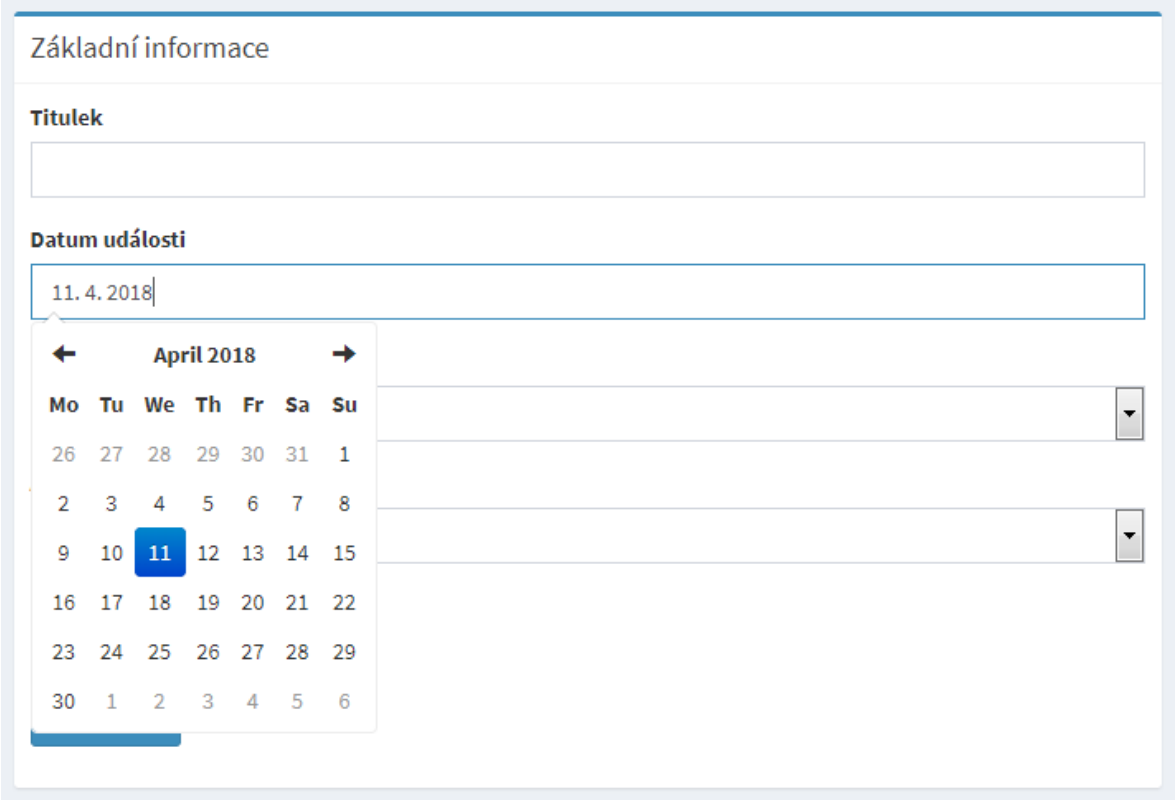
Vsuvka: Pole, které tvoří hodnoty „select boxu“ je možné vyplnit zcela libovolným způsobem. Na výše zobrazeném obrázku je například vidět, že uživatel má na výběr pouze z pěti hlavních stran.

Dalším prvkem je zaškrtávací pole neboli „check box“, který podobně jako „select box“ zajišťuje, že uživatel zadá přesně to, co je po něm požadováno. V případě zaškrtávacího pole je ovšem situace podstatně jednodušší. Políčko může být buďto zaškrtnuté, tedy „true“ (ano/pravda/logická 1) nebo nezaškrtnuté, tedy „false“ (ne/nepravda/logická 0). Jak z výše uvedeného vysvětlení vyplývá, zaškrtávací políčko se využívá na místech, kde se rozlišují pouze dva stavy (typicky ano/ne).

Nepostradatelným formulářovým prvkem moderního administračního prostředí je také prvek umožňující nahrávání souborů. Dle potřeby lze vybírat ze dvou variant tohoto prvku. Jednoduchá varianta umožňuje nahrát pouze jeden soubor (například jeden dokument nebo náhledový obrázek článku). Mnohonásobná varianta tohoto prvku, umožňuje nahrát více souborů zároveň (ideální například pro přidávání fotogalerie apod.). Dále lze nad tímto prvkem nastavit požadovaný formát souboru a maximální (či minimální) velikost souboru.

To by bylo z těch běžných prvků, které využíváme v naší aplikaci vše. Mimo ně však používáme ještě další rozšíření, která jsou nadstavbou ke klasickým formulářům. Obvykle lze tato rozšíření nalézt v různých variacích od různých vývojářů a téměř výhradně bývají napsána v JavaScriptu. My jsme využili „datetime-picker“ využívající „Nextras/Forms“,

dále „Dependent SelectBox“ a „Token-input“. První jmenovaný prvek slouží k výběru data a času. Klasické formuláře toto bohužel neumožňují, a tak pokud je potřeba, aby uživatel zadal datum, čas nebo obojí ve správném formátu, je nejlepší zvolit tuto osvědčenou a bezproblémovou knihovnu. Uživateli je po kliknutí do pole nabídnut grafický kalendář, ve kterém může postupně datum (případně i čas) zvolit. Poté je datum a čas v poli zobrazen v obvyklém tvaru (např. 1. 1. 2000) a lze jej kopírovat i editovat. Po odeslání formuláře je z tohoto textového řetězce vytvořen objekt typu datum a čas a lze s ním dále pracovat.



Obrázek 11: Formulářový prvek pro výběr data
Zdroj: vlastní

Druhým prvkem, je takzvaný „token input“. Ten se využívá zejména na místech, kdeje potřeba použít výběrové pole s velkým množstvím prvků. V takovém případě by pro uživatele bylo nepohodlné a zdlouhavé hledat v seznamu například stovek či tisíců uživatelů. „token input“ umožňuje uživateli vyhledávání v reálném čase. V praxi to vypadá tak, že se výběr s tím, jak uživatel píše jednotlivé znaky, postupně zužuje.

Snadné vyhledávání ovšem není jediný důvod, proč je na některých místech vhodné toto rozšíření použít. V případě vyššího množství záznamů (v řádu tisíců apod.) by si obyčejný „select box“ s takovým množstvím nemusel umět poradit a mohlo by dojít k chybě

z důvodu nedokončeného požadavku. „Token input“ nabízí řešení tím, že umožňuje nastavit minimální počet znaků pro vyhledávání. Tím je dosaženo toho, že se nejdříve zúží množství záznamů, z kterých lze vybírat a až poté se zobrazí. Díky tomu nedojde k vytvoření „select boxu“ s několika desetitisíci záznamů.

Posledním dodatečným formulářovým prvkem je závislé výběrové pole. Zpravidla je používáno v kombinaci s obyčejným výběrovým polem nebo s dalším závislým výběrovým polem. Principem závislého výběrového pole je, že hodnoty, kterými je pole naplněno, závisí na jiném prvku ve formuláři (typicky právě na předchozím výběrovém poli). Pro lepší představu nejlépe poslouží příklad z praxe. Uživateli je pro zvolení bydliště nabídnuto namísto textového pole výběrové pole. V prvním poli uživatel vybírá stát. V dalším poli je nutné uživateli poskytnout pouze kraje ze státu, který uživatel vybral (a to vše před odesláním formuláře). Přesně k tomuto účelu je možné využít některé z existujících rozšíření představující závislý select box. My jsme zvolili knihovnu Dependent Select Box od Nas Extensions.

7.2.3 Obecné představení požadavků

Při návrhu funkcionalit bylo nejdříve potřeba zvážit dva důležité body. Prvním bodem jsou požadavky uživatelů, kteří budou toto prostředí využívat. Druhým bodem, kolem kterého se celý backend navrhoval, je datová struktura. Konkrétně tedy bylo nutné vymyslet, jakým způsobem se budou jednotlivé prvky obsahu webu ukládat, zobrazovat a spravovat.

Prvky, které lze považovat za upravovatelný obsah webu, je možné rozdělit do několika kategorií. První kategorií jsou prvky, které mají své konkrétní místo na stránce, a lze u nich editovat pouze konkrétní hodnoty. Příkladem takového prvku může být například jídelníček nebo ceník jídel.

Do druhé kategorie spadá multimediální obsah. Tím jsou myšleny různé PDF a jiné dokumenty.

Další kategorií jsou fotografie a obrázky. Ty lze přidávat na více místech, kde slouží k různým účelům.

Vcelku standardním prvkem je i obsah stránky, který uživatel může vkládat jako HTML kód. K úpravě takovéhoto obsahu je využito WYSIWYG editoru (z angl. what you see is what you get – tedy co vidíš, to dostaneš). Toto řešení je použito výhradně na místech, kde je vhodné uživateli umožnit formátování textu a kde by z nějakého důvodu nedávalo smysl data formátovat univerzálně, dělit je do sloupců a vytvářet pro ně extra editační formuláře.

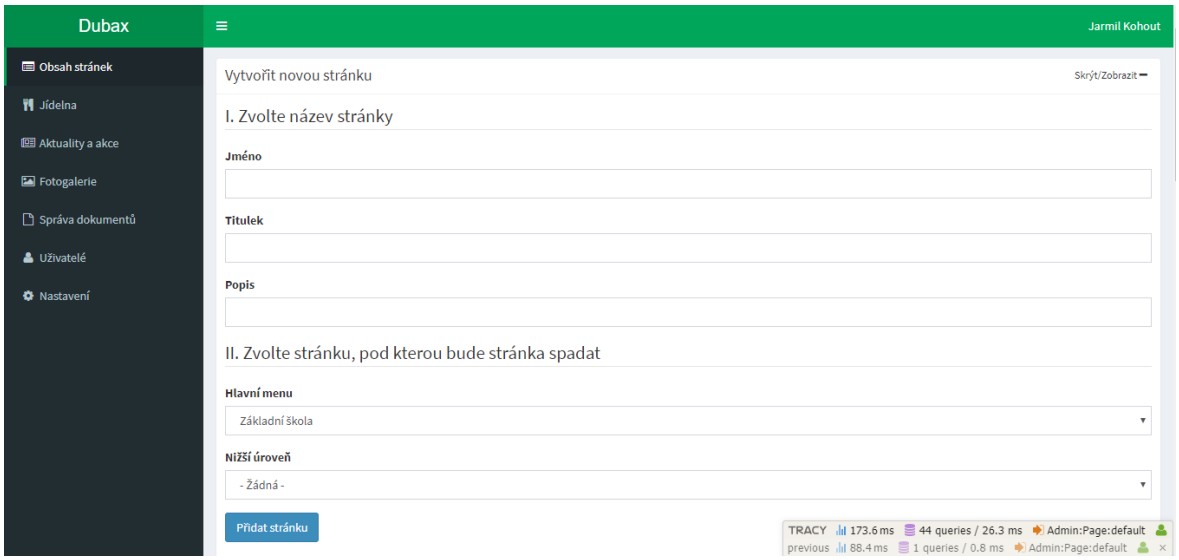
Posledním trochu specifickým typem obsahu, který je umožněn uživateli vkládat, je článek. Článek ve své podstatě není nic jiného než kombinace textových polí s HTML kódem (tedy formátovaného textu) a obrázků v libovolném rozmístění a pořadí. Článek tedy umožňuje uživateli vkládat obsah zcela dle jeho představ a poskytuje mu nad obsahem nejvyšší úroveň kontroly ze všech výše zmíněných kategorií.

7.2.4 Konkrétní funkcionality

Návrh funkcionalit probíhal souběžně s návrhem datové struktury, čímž bylo dosaženo toho, že data, která spolu souvisejí, jsou takto zanesena v databázi a jejich vztahy lze řešit i skrze nástroje, jež nabízí administrační část celé aplikace.

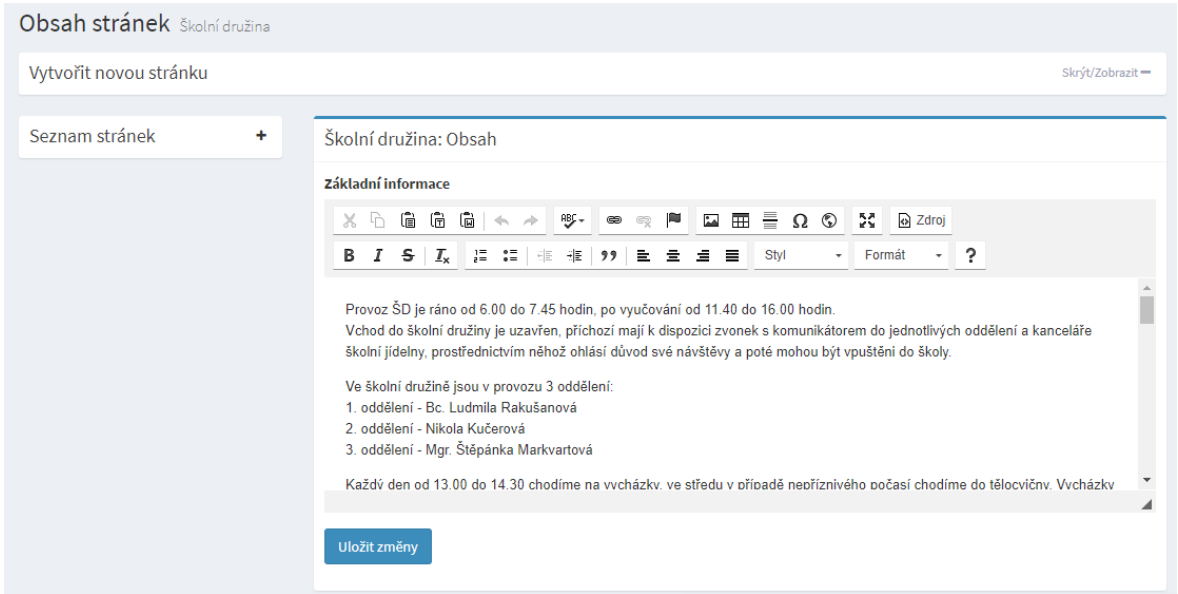
7.2.4.1 Stránka

Základním stavebním prvkem celého webu je stránka. Uživatelé mohou vytvářet stránky nové a navázat je na některou z již existujících. Tímto způsobem lze na sebe stránky vrstvit prakticky neomezeně. Každá stránka má pouze jednu stránku, která je jejím „rodičem“, což znamená, že na každou stránku lze přistoupit pouze jednou cestou. Z logiky věci vyplývá, že každá stránka musí mít svého rodiče, aby se na stránku dalo odněkud dostat. Výjimkou je pouze domovská stránka, která rodiče nemá a lze na ni přistupovat buď skrze hlavičku webu, nebo přímým vstupem na výchozí URL webové stránky. Domovská stránka je přitom rodičem pěti základních stránek v hlavní navigaci.



Obrázek 12: Formulář pro tvorbu nové stránky
Zdroj: vlastní

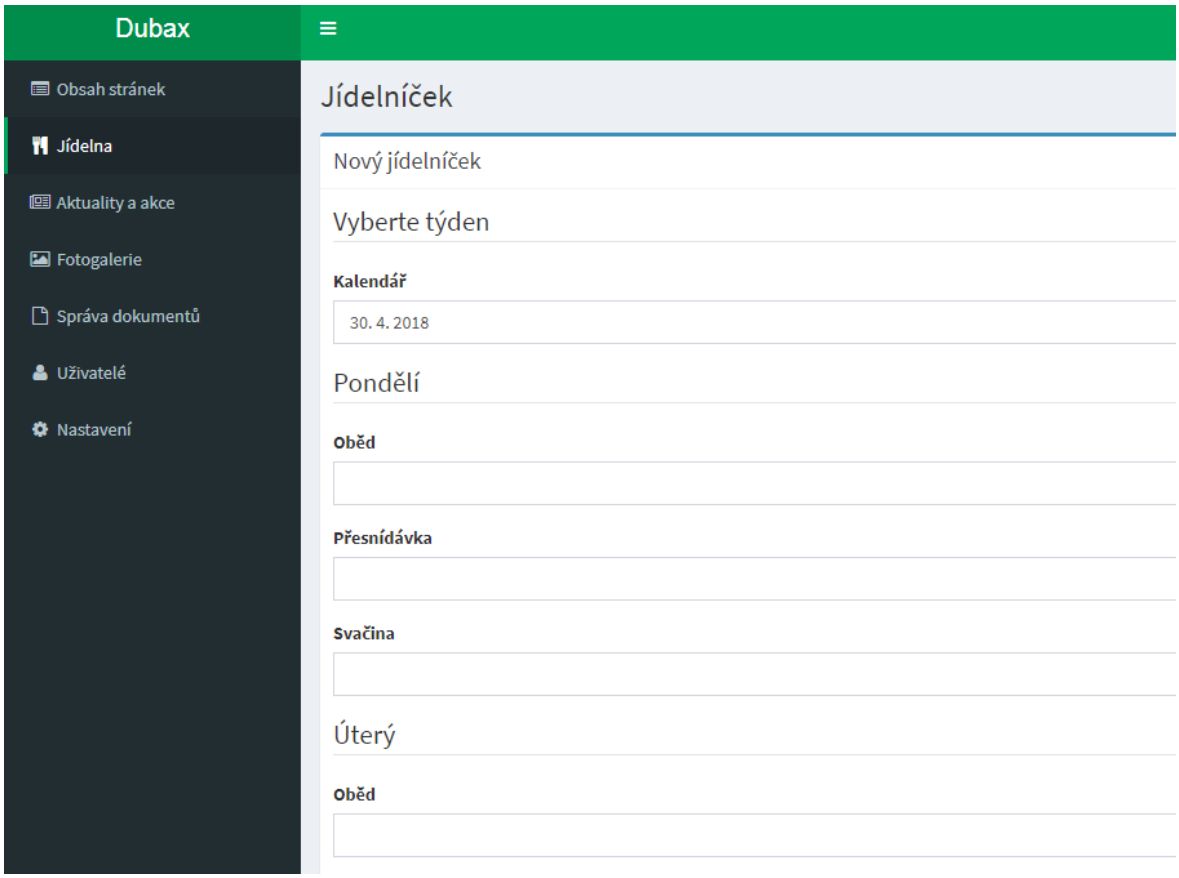
Každá stránka má svůj obsah, který lze editovat ve WYSIWYG editoru. My jsme pro tyto účely zvolili editor s názvem CKEditor. Tento editor umožňuje uživatelům naformátovat text do HTML formátu a to bez potřeby znalosti HTML. K editaci textu je uživateli poskytnuto grafické rozhraní (podobně jako například v textovém editoru Word). Vzniklý HTML kód se poté uloží do databáze a na požadovaném místě šablony se opět vyvolá. Kromě toho má každá stránka ještě titulek, který slouží jako nadpis celého obsahu. Hlavními důvody k oddělení nadpisu a obsahu je docílení konzistentního formátování na celých webových stránkách, ale také získání titulku stránky pro další potřeby.



Obrázek 13: Použití CK editoru pro správu obsahu stránky
Zdroj: vlastní

7.2.4.2 Jídelníček

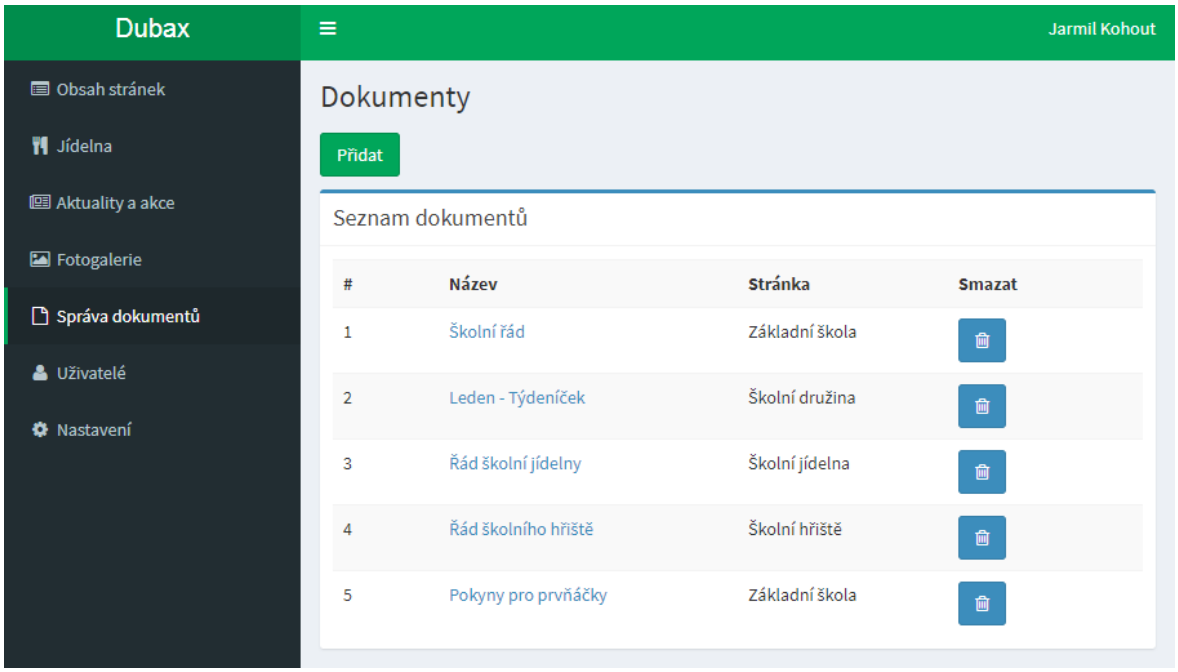
Formulář pro vytvoření jídelníčku je navržen tak, že na počátku uživatel vybere datum začátku týdne a dále již jen vyplní konkrétní jídla pro každý den do daných polí formuláře. Pokud je jeden nebo více dnů například volno, stačí, když jídla pro tento den uživatel nevyplní a o zbytek se již postará aplikace (a to včetně řádného uložení dat a jejich formátování).



Obrázek 14: Ukázka tvorby jídelníčku
Zdroj: vlastní

7.2.4.3 Správa dokumentů

Uživateli je umožněno na stránky vkládat také dokumenty ke stažení. Tato problematika byla v administraci vyřešena následujícím způsobem. Součástí administrace je záložka „Správa dokumentů“, která spravujícím uživatelům nabízí kompletní přehled všech dokumentů, které si mohou návštěvníci webu stáhnout. Zde je umožněno dokumenty přejmenovat, mazat, přesouvat na jiné místo na webu, ale i vytvářet dokumenty nové.



Obrázek 15: Seznam nahraných dokumentů
Zdroj: vlastní

Vytvořením nového dokumentu ke stažení provází administrátora jednoduchý formulář. Formulář obsahuje pouze tři formulářové prvky, přičemž jsou všechny tři prvky povinné. Prvním prvkem je název dokumentu. Název dokumentu uživatel vyplňuje z toho důvodu, aby nemusel být odvozován například z názvu souboru. Soubor by mohl mít v názvu nežádoucí znaky, jako jsou pomlčky, podtržítka, čísla označující verzi apod. Takto je docíleno toho, že název může být upraven do patřičné formy a lze jej libovolně měnit i po nahrání souboru. Druhý prvek je výběrové pole, ve kterém je uživateli nabídnut výčet stránek. Zde si uživatel zvolí, na které stránce se má dokument ke stažení nacházet. Do výčtu, ze kterého lze vybírat, jsme zahrnuli pouze hlavní stránky webu, aby dokumenty ke stažení zůstaly vždy pohromadě na jednom místě a nebyly tak rozeseté po spoustě podstránek. Posledním prvkem je samozřejmě prvek umožňující nahrání souboru. Zde je důležité zmínit jen to, že je umožněno nahrát pouze jeden soubor, který zároveň musí být v jednom z povolených formátů (pdf, docx, xlsx, pptx, apod.). Po stisknutí tlačítka jsou data zanesena do tabulky v databázi a soubor je uložen na server.

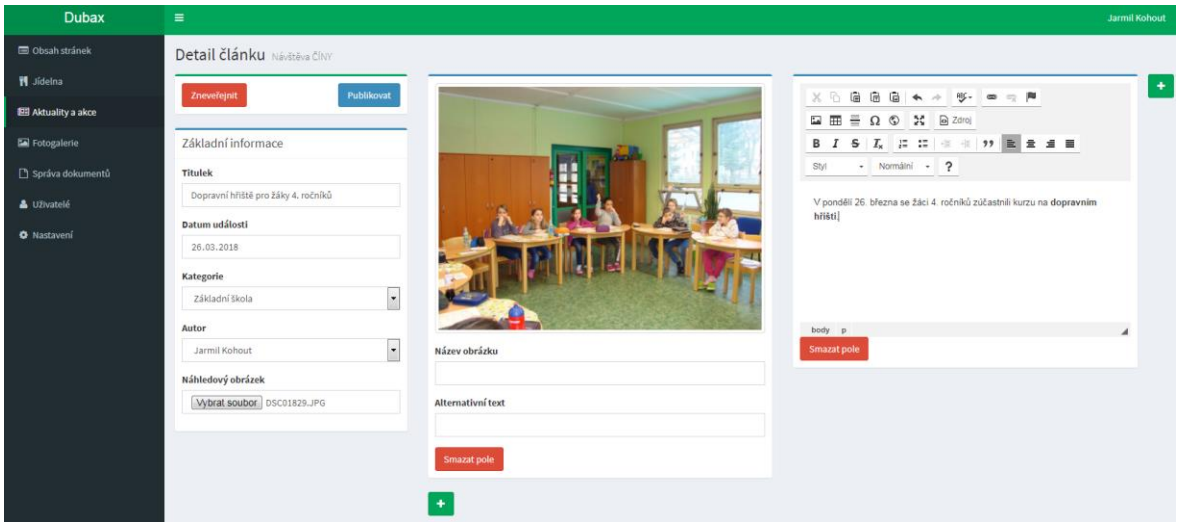
7.2.4.4 Články

Dále je v administraci záložka „Aktuality a nadcházející akce“. Obě položky se spravují naprosto identicky a souhrnně je nazýváme jako články. Při návrhu jsme vycházeli z toho, že aktualita se liší od nadcházející akce pouze v jediné vlastnosti. Tou vlastností je datum,

kdy daná událost proběhla (nebo v případě nadcházející akce – má proběhnout). Dle aktuálního data jsou tedy události rozděleny do těchto dvou kategorií, avšak sdílí spolu jak tabulku v databázi, tak i formulář a ostatní prvky jako metody pro zpracování apod.

V administraci v přehledu všech akcí jsou pro přehlednost události rozděleny dle tohoto kritéria a dále jsou seřazena dle data. Samotný proces tvorby nové události má dvě části. V první fázi je nutné zvolit datum konání akce, název akce, náhledový obrázek, krátký popis a také kategorii do, které událost spadá. Kategorie jsou na výběr pouze tři: Základní škola, Mateřská škola a obecná. Jak již bylo ve zkratce zmíněno, z data události je odvozeno, zda se jedná o něco, co již proběhlo nebo daná událost teprve nastane. Nutno podotknout, že zařazení do jedné z těchto kategorií není trvalé. Ve chvíli, kdy je aktuální datum roven nebo větší než datum události, událost je automaticky považována za aktualitu.

Druhá fáze poskytuje administrátorovi volnou ruku v návrhu detailu dané události. Detail může být zhotoven z obrázků a formátovaného HTML textu, přičemž oba prvky jsou řazeny do takzvaných bloků. Na čisté pracovní ploše není žádný blok. Uživatel vytvoří nový blok kliknutím na tlačítko označené znakem „+“, kde si dále zvolí buď textový blok, nebo obrázek. Vytvořený blok představuje de-facto jednu buňku. Dále se může rozhodnout, zda další blok umístí jako další sloupec nebo další řádek. Takto je možné pokračovat, co se sloupců týče až do maximálního čísla 12. Řádků může být neomezeně. Na každém řádku přitom může být libovolný počet „sloupců“ (v rozmezí 0-12) nehledě na to, kolik sloupců je v předchozím či následujícím řádku. Díky této volné koncepci struktury článku je umožněno uživateli vytvářet velice pestré články.

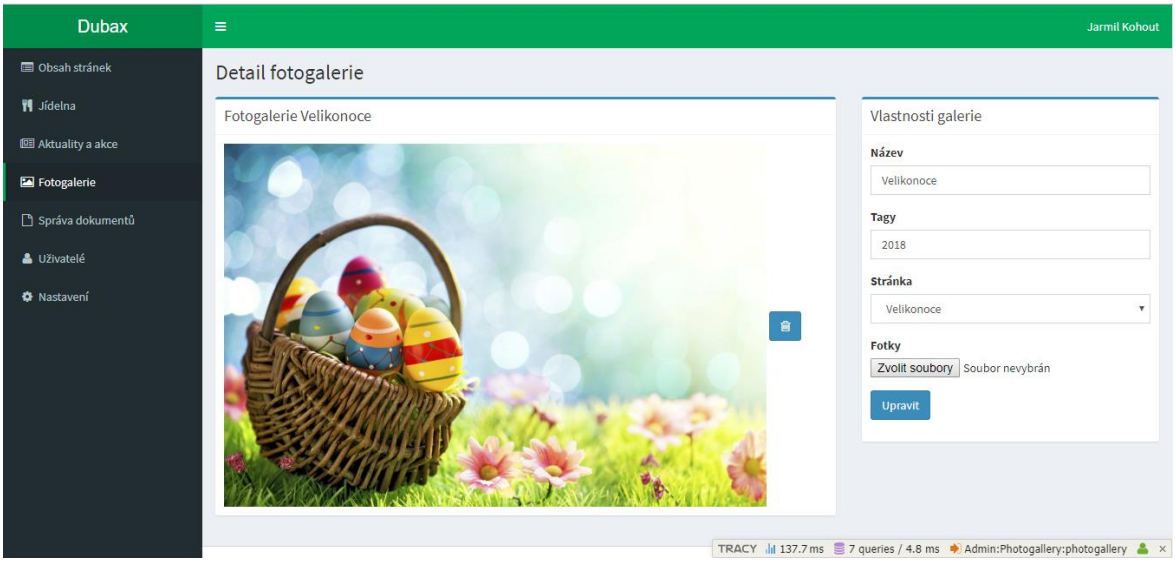


Obrázek 16: Editace článku v blocích
Zdroj: vlastní

Obrázky, které se vkládají do bloků článku, jsou automaticky ukládány v několika velikostních variantách. Hodnoty šířky a výšky těchto variant jsou rovněž uloženy do databáze. Pro následné zobrazování těchto obrázků ve článku používáme skript s názvem LazyLoad. Ten zajistí načtení obrázku vhodné velikosti dle rozlišení zobrazovacího zařízení.

7.2.4.5 Fotogalerie

Obrázky, které jsou přímou součástí článku, však slouží pouze ke zpestření obsahu a nejedná se o jediný způsob, jak nahrát například fotografie z nějaké školní akce na web. K tomuto účelu je vyhrazena jiná část administrativy, která je nazvána fotogalerie. Prostředí fotogalerie se opět příliš neliší od ostatních částí administrativy. Je zde přehled všech fotogalerií, kde je možné je upravovat a také úplně mazat. Samozřejmě zde nechybí ani tlačítko pro přidání nové fotogalerie. Proces vytváření nové fotogalerie je opět velice intuitivní. K dokončení celého procesu je zapotřebí vyplnit pouze tři formulářové prvky. Prvním prvkem je textové pole, do nějž se vyplňuje název fotogalerie. Druhým prvkem formuláře je výběrové pole, které fotogalerii přiřazuje k aktualitě. Na výběr jsou veškeré vytvořené aktuality. Posledním prvkem je „multi upload“ umožňující nahrání několika obrázků najednou. Množství fotografií není nijak omezené, avšak formát obrázku musí odpovídat některému z běžných formátů jako je jpg, png apod.



Obrázek 17: Detail fotogalerie, odkud lze editovat
Zdroj: vlastní

Ve veřejné části webu je poté fotogalerie vložena na konec článků, k němuž je přiřazena. Pro zobrazení fotografií jsme použili JavaScriptovou knihovnu s názvem halkaBox. Tímto způsobem zobrazujeme pouze několik prvních fotek jako miniatury a po rozkliknutí miniatury lze listovat všemi fotografiemi v dané fotogalerii. Fotografie se zobrazují přes celé okno prohlížeče s černým pozadím a navigačními tlačítky vpřed, vzad a zavřít.

8. Testování

Nette disponuje laděnkou Tracy, která byla představena již v kapitole o výbavě Nette. Tato laděnka umí zachytávat chyby jak při samotném vývoji, tak i za běhu aplikace na produkčním serveru. Samotné testování přitom probíhalo ve třech fázích.

V první fázi vždy každou funkci řádně otestuji sám, když danou funkcionalitu programuji. Ve chvíli, kdy je tato fáze u konce, je šance na výskyt chyby díky Tracy již podstatně menší. Přes toto síto projde pouze minimum chyb, které mohou vzniknout nějakou drobnou nepozorností například při dodatečných úpravách na již existující funkcionalitě apod. Pokud se vše udělá důkladně, je možné už v tuto chvíli získat aplikaci bez zjevných závad. Hraje zde ovšem velkou roli lidský faktor a tak nelze zajistit, aby testování proběhlo vždy správně a nic neuniklo naší pozornosti.

Ve druhé fázi přichází na řadu kontrola kódu některým z kolegů. Jedná se o běžnou praxi, kterou uplatňujeme u všech projektů. Optimální je odesílat změny po menších částech, tak aby bylo možné je důkladně zkontrolovat. Zde může kód úspěšně „projít“ hned na poprvé nebo se může vrátit s připomínkami, které je potřeba opravit. Po schválení kódu, je možné přejít dále.

V poslední fázi je na řadě testování přímo na serveru, kde jej používají běžní uživatelé. Obezřetní vývojáři mohou nejdříve použít testovací server namísto serveru, který je v ostrém provozu. Takový server velmi dobře poslouží k testování nových funkcionalit. Pokud se v této fázi skutečně nějaká chyba v kódu objeví a uživatel na ni narazí, vývojař je o tomto ihned informován a přesně ví, kde se problém nachází. Zbývá pouze chybu odladit, a projít celý tento proces znovu.

Laděnka při výskytu chyby prezentuje zdrojový kód aplikace s označeným řádkem chyby. Je zřejmé, že není žádoucí, aby se uživateli zobrazoval jakýkoliv kód a tak je mu namísto toho prezentována některá ze stránek informující pouze o kódu chyby (404, 500 apod.). Stránka, kterou generuje laděnka a která se nám při vývoji běžně zobrazuje, je v případě chyby na produkčním serveru uložena do zvláštní složky s logy. Opakující se chyby se ukládají pouze jako text do textového souboru s datem a časem vzniku. Díky tomuto nástroji je tak velmi snadné vzniklé chyby odhalit i jim částečně předcházet.

Závěr

Teoretická část této práce je zaměřená na současné trendy při vývoji backendové části webových aplikací. Detailněji se věnuje technologiím pro datovou a aplikační vrstvu aplikace, které dohromady tvoří celý backend. Autor se rovněž zmiňuje i o trendech spočívajících v užívání již kompletních řešení v podobě takzvaných redakčních systémů. Ve druhé části jsou představeny obecně platné postupy při návrhu databáze, ze kterých autor čerpá také v praktické části této práce. Poslední část teoretické části práce se týká doporučených doprovodných technologií, bez kterých si současný vývoj lze jen těžko představit.

Druhá polovina práce je již ryze prakticky zaměřená. V textu je uvedené, jaké platformy autor zvolil pro jednotlivé vrstvy projektu a to včetně odůvodnění své volby. Čtenářům je poskytnut pohled na celý proces vývoje od tvorby databázové struktury až po konečný průběh testování. Nemalý prostor je věnován také popisu jednotlivých funkcionalit, jimiž disponuje administrační část celého portálu. Nechybí zde ani zevrubná charakteristika architektury frameworku Nette a vlastního projektu, která je doplněna o praktické příklady a řešení.

Hlavním přínosem celé práce je vytvoření zcela nového webového portálu na míru pro Základní školu a mateřskou školu ve Stáži pod Ralskem. Podstatnou součástí celého portálu je i pro návštěvníky nepřístupný administrační systém, se kterým přicházejí do styku zaměstnanci školy spravující obsah portálu. Tento systém byl navržen s ohledem zejména na funkčnost, přehlednost, efektivitu a moderní a responzivní design.

Tato diplomová práce přitom může posloužit jako vodítko, které čtenáři pomůže vytvořit si představu o tom, jak je potřeba k návrhu takovéhoto projektu přistupovat. Vedlejším přínosem je také vysvětlení běžně užívaných postupů při návrhu a realizaci konkrétního webového portálu. V práci jsou rovněž představeny současné trendy při vývoji aplikační a datové vrstvy aplikací a nabízí tak obecné srovnání aktuálních technologických řešení

Seznam použité literatury

Citace

[1] ŘEZÁČ, Jan. 2014. Web ostrý jako břitva: návrh fungujícího webu pro webdesignery a zadavatele projektů. Jihlava: Baroque Partners. ISBN 978-80-87923-01-6.

[2] KROENKE, David a David J. AUER. Databáze. Brno: Computer Press, 2015. ISBN 978-80-251-4352-0.

[3] Frequently Asked Questions (FAQ) - The Go Programming Language. The Go Programming Language [online]. Dostupné z: https://golang.org/doc/faq

[4] GoUsers · golang/go Wiki · GitHub. The world‘s leading software development platform · GitHub [online]. Copyright © 2018 [cit. 11.03.2018]. Dostupné z: https://github.com/golang/go/wiki/GoUsers

[5] What is version control | Atlassian Git Tutorial. Atlassian | Software Development and Collaboration Tools [online]. Dostupné z: https://www.atlassian.com/git/tutorials/what-is-version-control

[6] Git - git-commit Documentation. Git [online]. Dostupné z: https://git-scm.com/docs/git-commit

[7] phpMyAdmin VS Adminer - Zdroják. Zdroják - o tvorbě webových stránek a aplikací [online]. Dostupné z: https://www.zdrojak.cz/clanky/phpmyadmin-vs-adminer/

[8] Licenční politika | Nette Framework. Quick 'n' Comfortable Web Development in PHP | Nette Framework [online]. Copyright © 2008, 2018 Nette Foundation. All rights reserved. [cit. 30.04.2018]. Dostupné z: https://nette.org/cs/license

[9] Back-End Development: A Guide to the Basics. App Development & Design Company - Wearables, Mobile & Web Development – ThinkApps.com [online]. Copyright © 2014. All rights reserved. [cit. 30.04.2018]. Dostupné z: http://thinkapps.com/blog/development/basics-back-end-development/

[10] Web portal - Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/Web_portal

[11] What is a hybrid website? Web Design Company India | Website Design India | Web Designing Company [online]. Copyright © 2004 [cit. 30.04.2018]. Dostupné z: http://www.xmediasolution.com/what-is-a-hybrid-website.html

[12] Website: Static vs Dynamic - javatpoint. Tutorials - Javatpoint [online]. Copyright © Copyright 2011 [cit. 30.04.2018]. Dostupné z: https://www.javatpoint.com/website-static-vs-dynamic

[13] Static Vs Dynamic websites - what's the difference?. Website design Salford, digital media from EDinteractive Salford University [online]. Copyright © 2018 [cit. 30.04.2018]. Dostupné z: http://edinteractive.co.uk/static-vs-dynamic-websites-difference/

[14] Usage Statistics and Market Share of Server-side Programming Languages for Websites, April 2018. W3Techs - extensive and reliable web technology surveys [online]. Copyright © 2009 [cit. 30.04.2018]. Dostupné z: https://w3techs.com/technologies/overview/programming_language/all

[15] Upwork - Hire Freelancers & Get Freelance Jobs Online [online]. Dostupné z: https://www.upwork.com/hiring/development/how-scripting-languages-work/

[16] Usage Statistics and Market Share of JavaScript Libraries for Websites, April 2018. W3Techs - extensive and reliable web technology surveys [online]. Copyright © 2009 [cit. 30.04.2018]. Dostupné z: https://w3techs.com/technologies/overview/javascript_library/all

[17] Episode 8: Interview with Ryan Dahl, Creator of Node.js - Mapping The Journey. Home - Mapping The Journey [online]. Copyright © 2017 Pramod Shashidhara All Rights Reserved [cit. 02.05.2018]. Dostupné z: https://www.mappingthejourney.com/single-post/2017/08/31/episode-8-interview-with-ryan-dahl-creator-of-nodejs/

[18] Vícevrstvá architektura – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Vícevrstvá_architektura

[19] Jak uplatnit principy třívrstvé architektury v rámci web integračního projektu | Webová integrace. [online]. Copyright © 2012 [cit. 30.04.2018]. Dostupné z: http://www.web-integration.info/cs/blog/jak-uplatnit-principy-trivrstve-architektury-v-ramci-web-integracniho-projektu/

[20] What is Oracle Database (Oracle DB)? - Definition from Techopedia. Techopedia - Where Information Technology and Business Meet [online]. Copyright © 2018 Techopedia Inc. [cit. 30.04.2018]. Dostupné z: https://www.techopedia.com/definition/8711/oracle-database

[21] What is MySQL? - Definition from Techopedia. Techopedia - Where Information Technology and Business Meet [online]. Copyright © 2018 Techopedia Inc. [cit. 30.04.2018]. Dostupné z: https://www.techopedia.com/definition/3498/mysql

[22] What is Microsoft SQL Server? - Definition from WhatIs.com. SQL Server: Covering today's SQL Server topics [online]. Dostupné z: https://searchsqlserver.techtarget.com/definition/SQL-Server

[23] What is Three-Tier Architecture? - Definition from Techopedia. Techopedia - Where Information Technology and Business Meet [online]. Copyright © 2018 Techopedia Inc. [cit. 30.04.2018]. Dostupné z: https://www.techopedia.com/definition/24649/three-tier-architecture

[24] Software Architecture: One-Tier, Two-Tier, Three Tier, N Tier. Software Testing Material [online]. Copyright © 2018 [cit. 30.04.2018]. Dostupné z: https://www.softwaretestingmaterial.com/software-architecture/

[25] 5 Benefits of a 3-Tier Architecture [online]. Dostupné z: https://www.izenda.com/blog/5-benefits-3-tier-architecture/

[26] 10 nejlepších PHP frameworků pro vývojáře | Interval.cz. Interval.cz | Svět Internetu, Technologií a Bezpečnosti [online]. Copyright © [cit. 30.04.2018]. Dostupné z: https://www.interval.cz/clanky/10-nejlepsich-php-frameworku-pro-vyvojare/

[27] Top 7 Best PHP Frameworks for 2017 | Archer Software. [online]. Dostupné z: http://www.archer-soft.com/en/blog/top-7-best-php-frameworks-2017

[28] An Overview of PHP Framework Guides for Developers. Onextrapixel - Web Design and Development Online Magazine [online]. Copyright © 2017 Onextrapixel. All

Rights Reserved. [cit. 30.04.2018]. Dostupné z: https://onextrapixel.com/an-overview-of-php-framework-guides-for-developers/

[29] What is Java? - Definition from WhatIs.com. TheServerSide.com: your Java Community discussing server side development [online]. Dostupné z: https://www.theserverside.com/definition/Java

[30] Where is Java used in Real World? [online]. Dostupné z: http://javarevisited.blogspot.cz/2014/12/where-does-java-used-in-real-world.html

[31] Comparison of Java and PHP for Web Applications - codecentric AG Blog. codecentric AG Blog - Expertenwissen rund um agile Softwareentwicklung, Java und Performance Solutions. [online]. Dostupné z: https://blog.codecentric.de/en/2008/07/comparison-of-java-and-php-for-web-applications/

[32] The 10 Best Java Web Frameworks for 2018 [online]. Dostupné z: http://www.dailyrazor.com/blog/best-java-web-frameworks/

[33] What is C++?. Computer Hope's Free Computer Help [online]. Copyright © 2018 Computer Hope [cit. 30.04.2018]. Dostupné z: https://www.computerhope.com/jargon/c/cplus.htm

[34] What is the C++ Programming Language? - Definition from Techopedia. Techopedia - Where Information Technology and Business Meet [online]. Copyright © 2018 Techopedia Inc. [cit. 30.04.2018]. Dostupné z: https://www.techopedia.com/definition/26184/c-programming-language

[35] Facebook Chat [online]. Dostupné z: https://www.facebook.com/notes/facebook-engineering/facebook-chat/14218138919/

[36] Learn Python (Programming Tutorial for Beginners). Learn Programming: Tutorials and Examples from Programiz [online]. Copyright © by Programiz [cit. 30.04.2018]. Dostupné z: https://www.programiz.com/python-programming

[37] What is the Python programming language? Everything you need to know | InfoWorld. InfoWorld - Technology insight for the enterprise [online]. Copyright © 2018 IDG Communications, Inc. [cit. 30.04.2018]. Dostupné z: https://www.infoworld.com/article/3204016/python/what-is-python.html?page=2

[38] Python at Google [online]. Dostupné z: http://quintagroup.com/cms/python/google

[39] Why Learn Ruby - Best Programming Language. Best Programming Language For Me in 2018 [online]. Dostupné z: http://www.bestprogramminglanguagefor.me/why-learn-ruby

[40] 6 Reasons Why JavaScript's Async/Await Blows Promises Away (Tutorial) [online]. Dostupné z: https://hackernoon.com/6-reasons-why-javascripts-async-await-blows-promises-away-tutorial-c7ec10518dd9

[41] Node.js as Backend: Best Use Cases, Tools & Limitations - Snipcart. Shopping Cart Platform for Any Website - Snipcart [online]. Copyright © All rights reserved, [cit. 30.04.2018]. Dostupné z: https://snipcart.com/blog/javascript-nodejs-backend-development

[42] Why Go? [online]. Dostupné z: https://hackernoon.com/why-go-ef8850dc5f3c

[43] Why you should learn Go | Pluralsight. Unlimited Online Developer, IT and Cyber Security Training | Pluralsight [online]. Copyright © 2004 [cit. 30.04.2018]. Dostupné z: https://www.pluralsight.com/blog/it-ops/go-programming-language

[44] GitHub lists the 15 most popular programming languages - Business Insider. Business Insider [online]. Copyright © 2018 Insider Inc. All rights reserved. [cit. 30.04.2018]. Dostupné z: http://www.businessinsider.com/the-9-most-popular-programming-languages-according-to-the-facebook-for-programmers-2017-10/#for-bonus-points-heres-the-chart-showing-these-languages-relative-popularity-16

[45] Programming languages used in most popular websites - Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites

[46] What is HTML? | HyperText Markup Language explained. HTML Source: HTML Tutorials [online]. Copyright © 2000 [cit. 30.04.2018]. Dostupné z: http://www.yourhtmlsource.com/starthere/whatishtml.html

[47] Cascading Style Sheets - Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[48] HTML5 – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/HTML5

[49] Upwork - Hire Freelancers & Get Freelance Jobs Online [online]. Dostupné z: https://www.upwork.com/hiring/development/how-scripting-languages-work/

[50] What Is jQuery?. Web Design and Development Tutorials, Articles and Forums | Elated.com [online]. Copyright © Elated Communications 1996 [cit. 30.04.2018]. Dostupné z: https://www.elated.com/articles/what-is-jquery/

[51] Vícevrstvá architektura: popis vrstev - CleverAndSmart. CleverAndSmart - ICT management [online]. Copyright © 2008 [cit. 30.04.2018]. Dostupné z: https://www.cleverandsmart.cz/vicevrstva-architektura-popis-vrstev/

[52] About Joomla! [online]. Dostupné z: https://www.joomla.org/about-joomla.html

[53] Entita [online]. Dostupné z: https://www.it-slovnik.cz/pojem/entita

[54] 1. díl - Git - Historie a principy. [online]. Copyright © 2018 itnetwork.cz.Veškerý obsah webu [cit. 03.05.2018]. Dostupné z: https://www.itnetwork.cz/software/git/git-tutorial-historie-a-principy

Bibliografie

MELONI, Julie C. PHP, MySQL & JavaScript All in One, Sams Teach Yourself. Sams Publishing, 2017. ISBN 978-0672337703

ZANDSTRA, Matt. PHP Objects, Patterns, and Practice. s.l. : Springer Verlag, 2016. ISBN 978-1484219959

ULLMAN, Larry. Php and mysql for dynamic web sites: visual quickpro guide. Peachpit Press, 2016. ISBN 978-0134301846

DUBOIS, Paul. Mysql Cookbook: Solutions for Database Developers and Administrators. Sebastopol, CA : Oreilly & Associates Inc, 2014. ISBN 978-1449374020

Log In - ProQuest [online]. Dostupné z: https://search.proquest.com/docview/28905726/EEB49E38DE054771PQ/2?accountid=17116