Mendelova univerzita v Brně
Provozně ekonomická fakulta

Informační systém pro základní školu

Diplomová práce

Vedoucí práce:
Ing. Jiří Lýsek, Ph.D.

Bc. David Mateašák

Brno 2015

TADY VLOŽIT ZADÁNÍ DP

V první řadě bych tímto způsobem velmi rád a srdečně poděkoval základní škole, která si mne vybrala a pro kterou má diplomová práce mohla být přínosem a také vedoucímu své diplomové práce Ing. Jiřímu Lýskovi, Ph.D. za poskytnutí kvalitních a účelných rad a za čas, který mi věnoval. Dále bych také rád poděkoval ostatním osobám, bez kterých bych se nedostal nebo nedokončil tuto diplomovou práci. A to mému kamarádovi Lukáši Vaňáčkovi, který byl spojovacím článkem mezi mnou a základní školou a převážně své rodině za podporu a trpělivost, kterou museli občas vykazovat z důvodu mého časového vytížení. Velké díky všem.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Informační systém pro základní školu**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.* Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 27. prosince 2015 _____

Abstract
Mateašák D. The information system for primary school.
Thesis. Brno, 2015.
The thesis describes an information system developing which is built on customer's needs and requests. The customer in this case is an elementary school. The whole information system fully correspond to modern developer techniques and standards with using MVC architecture and responsive design.

Keywords

Information system, responsive design, MVC architecture, nette framework, bootstrap, wireframe, latte template, presenter

Abstrakt

Mateašák D. Informační systém pro základní školu.
Diplomová práce. Brno, 2015.
Diplomová práce se zaobírá tvorbou informačního systému, který je postaven na základě požadavků od zákazníka, kterým je základní škola. Informační systém plně splňuje moderní vývojářské technologie a principy, za použití MVC architektury a responzivního designu.

Klíčová slova

Informační systém, responzivní design, MVC architektura, Nette framework, bootstrap, wireframe, latte šablona, presenter

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod a cíl práce

1.1 Úvod

Aktuální potřeba, řekl bych až hladová, dnešní společnosti zůstat stále informována neodmyslitelně spadá na každodenní pořádek. Je kladen velký důraz na rychlosti a přesnost získaných informací a pokud možno, abychom je měli už včera. S těmito myšlenkami se pojí i tato diplomová práce, která se bude snažit tento trend udržet pokud možno aktivní.

Když se podíváme kolem sebe, tak si nemůžeme nevšimnout, neustále se zdokonalující hardwarové technologie a velkých možností, které se nabízejí. S tím jde ruku v ruce vývoj a zdokonalování developerských dovedností a technik, které úspěšně dohání tento mechanický vývoj. V posledních letech se přešlo u zobrazování webových projektů z desktopového rozlišení na menší mobilní displeje mobilních telefonů, tabletů a hybridních zařízení (kombinující tablet a notebook). Z tohoto pohledu mne velmi lákal vytvořit systém, který by byl responzivní a dokázal se přizpůsobit právě prohlíženému zařízení, bez všelijakých omezení na kráse grafickému designu.

1.2 Cíl práce

Hlavním cílem práce je návrh a implementace informačního systému pro základní školu. Bude provedena detailní analýza požadavků na základě ústní domluvy se zákazníka, ze které se získají potřebné informace, na kterých se poté vybuduje návrh IS. Bude vytvořen ERD diagram pro databázi a za neustálé komunikace se zákazníkem budou systému navrženy požadované funkce (v podobě modulů), které budou naimplementovány a to i včetně grafického designu na přání zákazníka.

Sekundárním cílem je snaha držet krok s dobou a navrhnout grafické design v responzivní podobě, aby se systém dokázal přizpůsobit dnešním zařízením s různou velikostí displeje (mobil, tablet, desktop).

Cílem systému, který bude sloužit pro plnění a správu dat webových prezentací s prvky informačního systému, bude snaha udržet jeho uživatele, kterými jsou převážně rodiče ze strany žáků základní školy a další zainteresované osoby, vždy pokud možno, co nejvíce informované. To se bude dít na základě rozesílání hromadných oznámení novinek vznikajících na školním území.

2 Návrh informačního systému

Z důvodu správného a zřejmého chápání informačních systémů si v následující kapitole nejprve přiblížíme pár pojmů, které se blízce dotýkají informačních systémů a softwarových procesů, které s vývojem a návrhem těchto systémů souvisí. Dále se podíváme na samotný návrh informačního systému a v závěru kapitoly bude příkladově ukázán jeden systémový proces.

2.1 Softwarové procesy

Softwarové procesy jsou užitečnou sadou, možná přesněji řečeno sekvencí aktivit, které nám pomáhají smysluplně a efektivně zhotovit zamýšlený softwarový produkt.

V dnešní době se můžeme setkat s velkou škálou různých softwarových procesů. Od typu procesů, které mohou vést vývoj aplikace od úplného začátku, přes další typ, který může být užitečný pro vývoj již zaběhlé aplikace pomocí jejího rozšiřování a úprav. Až k typu softwarových procesů, pomocí kterých můžeme integrovat či konfigurovat softwarové produkty, které jsou běžné k dostání. Obecně lze říci, že všechny softwarové procesy by měly zahrnovat následující čtyři aktivity, které hrají v softwarovém inženýrství důležitou roli.

- *Specifikace požadavků*
  Je základním krokem ve vymezení softwarových funkcí a omezeních na jeho činnosti.
- *Návrh a implementace softwaru*
  Vyhotovený software by měl vycházet z návrhu a ten by měl splňovat vytyčené specifikace.
- *Validace softwaru*
  Před finálním vydáním je nutné software validovat (zkontrolovat). To proto, abychom zjistili a případně se zajistilo, aby odpovídal vlastnostem a požadavkům zákazníka.
- *Další vývoj softwaru*
  Software by se měl do budoucna, pokud možno, neustále vyvíjet a udržovat aktuální dle vyžadovaných potřeb zákazníka.

Dále kromě aktivit mohou být také procesy popsány následujícími způsoby:

- Produktem, což bývá výsledek nějaké aktivity procesu. Např.: výsledkem aktivity návrhu architektury může být model softwarové architektury.
- Rolí, která reflektuje odpovídající chování osoby, která se procesu účastní. Např.: role administrátora.

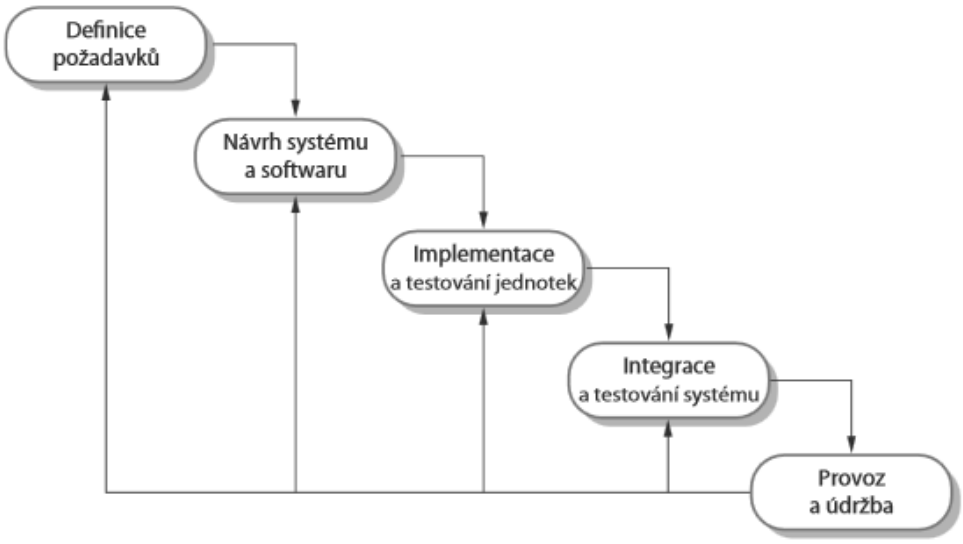
- Podmínkou, kterou většinou zastupuje nebo různá tvrzení týkající se provedení aktivity procesu. V mém případě např. tak, že před zahájením aktivity návrh a implementace softwaru musí všechny specifikace schválit zákazník.

Protože jsou softwarové procesy složité a tak, jako většina kreativních a psychických procesů jsou také závislé na lidech. Téměř neexistuje obecně ideální a platný proces, který by se dal aplikovat na všechny typy vývoje softwaru, a proto si většina společností vytváří procesy vývoje softwaru vlastní. Takové procesy lze samozřejmě zlepšovat a zjednodušovat například pomocí standardizace, kdy se patřičně omezuje jejich rozmanitost.

Anebo pomocí další pomocné berličky, kterou může být využití modelů, které ve zjednodušené formě prezentují příslušný softwarový proces. Každý takový model zobrazuje proces z určitého úhlu pohledu a částečně jej popisuje. Jinými slovy by se dalo říct, že se jedná o popisy abstrakce procesů, pomocí kterých můžeme vysvětlit různé přístupy k vývoji softwaru.

2.1.1 Vodopádový model

Model pracuje se základními aktivitami procesu (specifikace, návrh, validace a další vývoj), které zde prezentují samostatné kroky, ty můžeme vidět na obr. 1. Všechny kroky procesu je vhodné naplánovat ještě před tím, než se začnou zpracovávat.



Obr. 1 Vodopádový model (Softwarové inženýrství, 2013)

Jednotlivé kroky přesně zrcadlí základní aktivity softwarového procesu.

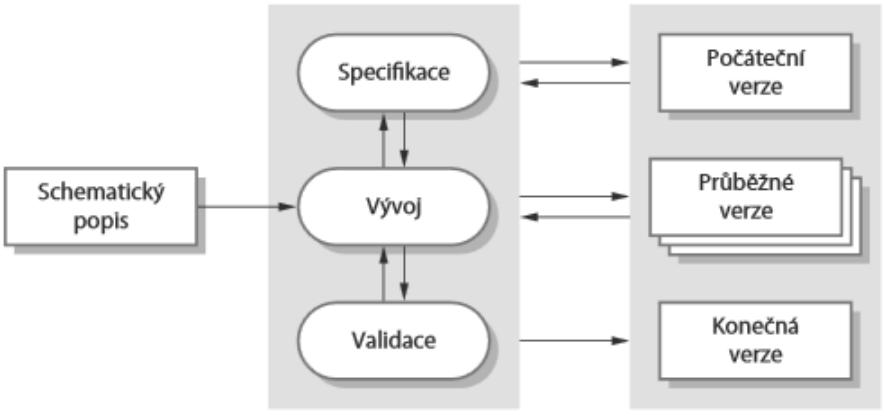
- *Definice požadavků*
  Probíhá na základě vzájemné komunikace se zákazníkem a uživateli systému. Zde se tyčí služby, omezení a jednotlivé cíle systému.

- *Návrh systému a softwaru*
  V tomto kroku se přidělí jednotlivé požadavky na hardware nebo software systému pomocí systémové architektury. Obsahuje rozpoznání a popis základních abstrakcí systému a jejich vztahů.
- *Implementace a testování jednotek*
  V počáteční fázi se realizuje návrh systému, jako dílčí programové jednotky a v části následující se každá jednotka testuje, zdali splňuje svou specifikaci.
- *Integrace a testování systému*
  Provádí zapojování a testování každé jednotky systému jako kompletní celek. A po testovací fázi se výsledný softwarový produkt předá zákazníkovi.
- *Provoz a údržba*
  Obecně nemusí platit vždy, ale běžně se jedná o časově nejdelší fázi vývoje. Zhotovený systém se instaluje, předává zákazníkovi a v neposlední řadě i zaškoluje. Dále se časem provádí nezbytné opravy chyb, které nebyly nalezeny během vývoje, vylepšují se systémové jednotky a služby systému dle nových požadavků zákazníka.

Často mezi jednotlivými kroky dochází k jejich překrývání, to proto, že svou komplexností pokrývají velmi široké oblasti. A také je to dáno přenosem informací mezi jednotlivými kroky a tím pádem i možným zvýšením efektivity práce.

2.1.2 Model inkrementálního vývoje

Tento přístup mezi sebou prolíná jednotlivé aktivity softwarového procesu specifikace, vývoj a validaci, kdy mezi nimi probíhá rychlá zpětná vazba. Samotný systém se pak vyvíjí jako řada dílčích verzí, kde každá následující verze přidává další funkcionalitu k předchozí verzi a tím ho vylepšuje.



Obr. 2 Model inkrementálního vývoje (Softwarové inženýrství, 2013)

Princip tohoto modelu spočívá ve vytvoření počáteční implementace systému, která se nechá validovat uživatelem. Poté se dále vyvíjí několik verzí, do doby než vznikne původně plánovaný systém.

Tento model také vytváří základní kámen agilního vývoje softwaru, kdy je většinou vhodnější než vodopádový přístup, protože lépe odráží a dokáže se lépe přizpůsobit principům lidského myšlení. Kdy ne tak často umíme zpracovat kompletní řešení od začátku do konce, krok po kroku bez chyby. Inkrementální způsob vývoje umožňuje snadnější návrat k předchozím krokům, kdy můžeme spoustu problémů ihned opravit anebo něco vylepšit.

Inkrementálního vývoj v kombinaci s vodopádovým modelem nesou značné výhody při vývoji softwaru:

- Kdy se snižují režijní nároky a také náklady na provedení změn vyvolaných u měnících se požadavků zákazníka.
- Díky dílčím vývojovým verzím lze získat rychlejší zpětnou vazbu od zákazníka a tím pádem i provádět rychlejší úpravy na systému.
- Možnost rychlejšího nasazení systému u zákazníka, i když nebude systém zcela kompletní. To je možné opět díky dílčím verzím systému. Například samotný vodopádový model tuhle možnost moc dobře neumožňuje.

Podstatné je zmínit fakt, že výše zmíněné modely se navzájem nevylučují a často se používají zároveň, zejména při vývoji komplexnějších systémů. (Sommerville, 2014)

V mém případě vhodně kombinuji benefity vodopádového modelu a modelu inkrementálního vývoje. Kdy části systému, které můžu dobře předem zanalyzovat, specifikovat a vyvíjet, jako například návrh systému postavený na zákaznících požadavcích, jednotlivé funkce systému apod. jsou postaveny na vodopádovém modelu. A ty části, které jsou těžce předem specifikovatelné, jako například uživatelské rozhraním, různé detaily jednotlivých funkcionalit systému apod., je optimální vyvíjet pomocí inkrementálního modelu.

2.2 Specifikace požadavků

Za specifikací požadavků se skrývá první a zároveň nejdůležitější aktivita vývoje softwarového procesu, kdy zjišťujeme, jak má vyvíjený systém pracovat. Tento krok také představuje jaké funkce a služby bude poskytovat, dále pak vymezuje různé omezení funkcí a vývoje systému. Z tohoto pohledu můžeme rozdělit specifikaci požadavků na dvě skupiny a to na funkční požadavky a nefunkční požadavky. Dobré je také zmínit to, že je vhodné využívat obě skupiny požadavků, protože nám poskytují různé informace o chování systému a teprve dohromady utváří efektivní komplet.

Co se týče získávání požadavků, dělo se vše na základě ústní komunikace a během několika opakování, za neustálého upřesňování a vymezování požadavků. Osobně si myslím,že tento krok zabral nejvíce času.

2.2.1 Funkční požadavky

Tyto požadavky přesněji vymezují a popisují jednotlivé systémové funkce, které závisí od typu vyvíjeného softwaru a uživatelů, kteří k němu budou přistupovat a využívat jej. Požadavky by měly být úplné a tedy vymezovat všechny potřebné systémové funkce a možnosti. A také kompatibilní a neměly by si zároveň mezi sebou odporovat v činnosti.

Všeobecné požadavky znějí „r*ádi bychom, kdybychom si mohli měnit veškerý textový obsah na webu sami. Tzn. měnit pěvně ukotvený titulek na liště v prohlížeči, akreditaci školy, vizi školy, text v patičce, také obrázek na pozadí střední veřejné části.*“

Modul stránek: „Systém *bude mít možnost vytvářet stránky pod jednotlivé instituce (základní škola, školní družina a mateřská školka) a hlavní informační bloky školy (pro rodiče a žáky, školní akce a mezinárodní spolupráce). U stránek si přejeme měnit jejich textový obsah a také mít možnost jeho vlastního formátování. Dále vyžadujeme možnost přikládat obrázkové i souborové přílohy. Také chceme mít možnost stránku skrýt nebo zobrazit po jejím vytvoření anebo před jejím vydáním.*“

Modul slider „*bude umožňovat vkládání, editování, mazání jednotlivých obrázků a také jejich hromadné mazání. Dále chceme mít možnost zobrazovat nebo skrývat jednotlivé obrázky a celý slider na veřejné části webu. Také si přejeme, aby se obrázky zobrazovaly, jakoby z prostoru a ne, aby přejížděly zprava doleva a naopak, mohlo by to být nepříjemné pro některé návštěvníky.*“

Modul partnerů školy „*zobrazení partnerů bude na hlavní stránce a bude umožňovat stejné operace pro jednotlivé partnery, jako část předchozí. Celý blok partnerů na veřejné části webu se bude moci dát zobrazit nebo skrýt. V případě neposkytnutí loga od partnera se místo něj zobrazí pouze odkaz na jeho web.*“

Modul newsletter „*bude umožňovat našim návštěvníkům přihlášení k odběru našich novinek (články a školní oznámení) pomocí emailu. Ten návštěvníci zadají na hlavní stránce webu. Odkaz pro odhlášení z newsletteru bude umístěný v samotných emailech, anebo nás emailově kontaktují a mi si odběratele vyřadíme sami. Chceme mít možnost v administraci přidávat, editovat i mazat jednotlivé odběratele.*“

Modul článků „*bude umět vytvářet články dle našich představ, tzn. s vlastním formátováním textu. Bude zde umožněno vkládání příloh, ty budeme chtít na veřejné části webu mít rozřazené a to tak, že obrázky budou zobrazené v části galerie článku a soubory v části příloh ve článku. Dále chceme vytvářet články rozdělené podle typů na školní články, články patřící školní družině, mateřské školce a školní akce. A také si přejeme rozlišovat články na normální články a články s video tématikou. Dále články, které nebudeme chtít vidět na hlavní stránce, budeme chtít uschovat v archivu.*“

Modul školních oznámení „*bude informovat veřejnost krátkými školními oznámeními. Ty budou zobrazeny ve vlastním panelu na hlavní stránce a také v kalendáři, pro lepší vizualizaci. Dále chceme jejich celkovou správu (přidávání, editace, mazání, zobrazování nebo skrývání oznámení na stránce a i v kalendáři nezávisle na sobě). Také bychom měli rádi rozlišená oznámení podle typu, jako u modulu článků.*“

Modul požadavků školy je popsán v kapitole 2.4.1 funkční požadavky.

Modulu zaměstnanců školy *„si přejeme vytvářet zaměstnance s těmito informacemi (titul, jméno, příjmení, profilová fotka, role zaměstnance, email, telefon a krátký popisek). Dále jejich běžnou editaci, mazání, skrytí nebo zviditelnění a také jejich hromadné mazání. Dále požadujeme vytváření zaměstnaneckých rolí a jejich přiřazování jednotlivým zaměstnancům.“*

2.2.2 Nefunkční požadavky

Jak může z názvu vyplývat, tak se nefunkční požadavky přímo nevztahují na konkrétní služby systému, ale spíše na jeho vlastnosti, jako celku. Tyto požadavky bývají často kritičtější než funkční požadavky z důvodu vyšší náchylnosti na chod systému. Například se může stát, že při vývoji opomene respektovat některý z nefunkčních požadavků je možné, že to způsobí pád nebo, že nebude funkční celý systém.

Softwarové požadavky:
„V první řadě bychom chtěli mít administrační část oddělenou od veřejné části systému a pod heslem. Veřejnou část bychom rádi uvítali v barvité a hravé grafice. Požadujeme responzivní design celého systému, aby naši návštěvníci neměli problém s prohlížením, jak na mobilu, tabletu tak počítači. Také by mělo být jasné rozdělení školních institucí a dalších hlavních informačních bloků.“

Modul zaměstnanci školy *„bude vykreslován hierarchicky a to tak, že ředitelka školy bude umístěná nejvýše nad ostatními zaměstnanci, učitelé pod ní a ostatní personál školy pod učiteli.“*

Modul požadavků školy je popsán v kapitole 2.4.1 nefunkční požadavky.

Hardwarové požadavky:
„Nemáme jasnější představu o tom, co tohle všechno obnáší. Ale obecně si přejeme, abychom měli dostatečnou kapacitu na serveru pro všechny naše soubory a vytvářený obsah. A dále, aby chod systému nebyl nijak omezován a rušen ze strany poskytovatele webhostingu.“

Požadavky na použitelnost:
„Vyžadujeme, co nejjednodušší manipulaci se systémem. Vše přehledně uspořádáno, aby bylo možné systém během pár minut intuitivně používat.“

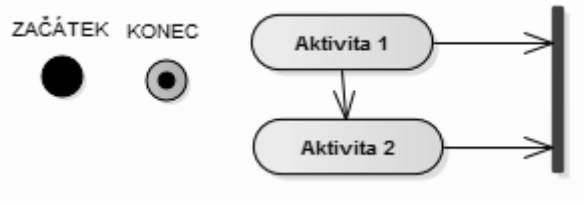
2.3 Návrh IS

Proces návrhu informačního systému nám umožňuje vytvářet abstraktní pohledy (modely) nad systémem, kde každý takový pohled ukazuje odlišný způsob zobrazení daného systému nebo jeho části. To se děje pomocí diagramů, které jsou definované a vytvářené v jazyce UML (Unified Modeling Language), což je v dnešní době etablovaný standartní modelovací jazyk pro objektově orientované modelování.

Samotné modelování všech diagramů bylo provedeno v modelovacím nástroji Enterprise Architect.

2.3.1 Diagramy aktivity

Aktivity, ze kterých se skládá systémový proces (výpočetní proces), a jak si předávají řízení mezi sebou, jsou znázorněny pomocí diagramů aktivit. Ty dále zobrazují postup jednotlivých aktivit v čase a také, kde začíná a kde končí celý proces. (mpavus.wz.cz, 2013)



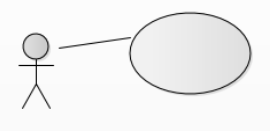
Obr. 3 Diagram aktivity

Popis diagramů aktivit:
Každý diagram představuje směsici různých objektů (symbolů, čar a šipek) s různou funkcionalitou. Protože je diagram chápán s časovou osou, má tím pádem začátek, který bývá značený vyplněným kruhem a konec, který je značený kruhem s dalším vyplněným kruhem uvnitř.

Dále se diagram skládá z obdélníků se zaoblenými rohy, které symbolizují jednotlivé aktivity, které je třeba provést. Jednotlivé aktivity diagramu bývají mezi sebou a s ostatními objekty diagramu propojeny pomocí šipek, ty značí směr toku informací a prezentaci časové posloupnosti činností.
Z plných čar (bez šipek na koncích), které znázorňují sjednocení nebo koordinaci aktivit. Je-li na plnou čáru napojeno více toků informací z různých aktivit (více šipek), aby se přešlo k dalšímu kroku, musí se vyčkat, dokud nebudou všechny aktivity provedeny. Směřuje-li více toků informací z plné čáry, mohou být jednotlivé aktivity prováděny paralelně.

2.3.2 Diagramy případu užití

Tyto diagramy se nejčastěji využívají pro popis scénáře, který zachycuje chování uživatelů, jejich typů a také jaké činnosti se systémem provádějí. Jednotlivé činnosti systému se potom nazývají případy užití, které plní nějaký cíl. Jinými slovy tyto diagramy umožňují znázornit funkční požadavky systémového procesu tak, že popisují interakci mezi ním a uživateli. (uml.czweb.org, 2010)



Obr. 4 Diagram případu užití

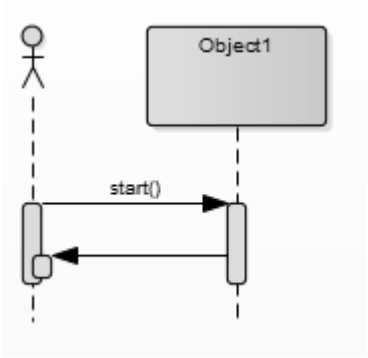
2.3.3 Popis diagramů případu užití:

Protože tyto diagramy zobrazují uživatele, kteří jsou v kontaktu se systémem, bývají nejčastěji znázorněni pomocí figurek z jednoduchých čar.

Jednotlivé případy užití se značí nejčastěji formou elipsy. A jednotlivé interakce mezi objekty, potom pomocí čar znázorňující tok informace.

2.3.4 Sekvenční diagramy

V jazyce UML jsou sekvenční diagramy nejrozšířenějšími diagramy sloužící k zobrazování aktivit mezi aktéry a objekty systému. Znázorňují sekvenci (pořadí) jednotlivých interakcí, ke kterým dochází během určitého případu užití.



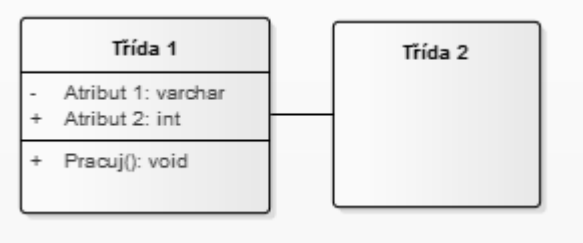
Obr. 5 Sekvenční diagram

Popis sekvenčních diagramů:
Opět se zde setkáváme s aktéry, kteří jsou znázorněni pomocí figurek a jsou umístěni v horní části diagramu. Zde jsou umístěny také objekty, se kterými aktéři komunikují. Komunikace mezi nimi je symbolizována pomocí šipek s popisky, které popisují volání objektů, jejich parametry a návratové hodnoty. Od každého objektu potom vede čárkovaná čára tzv. lifeline (životní čáru), která nám říká, jakým způsobem se instance daného objektu interakce účastní.

2.3.5 Diagramy tříd

Při vývoji objektově orientovaného programování se používají v diagramech tříd objekty, které jsou prezentací reálných objektů, a jejich vazby. Každá třída si nese určité informace. Mezi ně spadají parametry třídy a metody, které mohou s těmito parametry pracovat. Díky vazbě mezi jednotlivými třídami je možné sdílet mezi nimi informace.

Protože je navrhovaný systém implementovaný pomocí architektury MVP (více v kapitole 4.3.1) bude diagram tříd lechce odlišný od běžného zobrazení, protože se zde nepracuje pouze se třídou, která prezentuje reálný objekt. Ale s vrstvou, která obsluhuje a obstarává práci s tímto objektem, který je prezentován databázovou tabulkou v ERD, pro více informací přejděte na obr. 14.

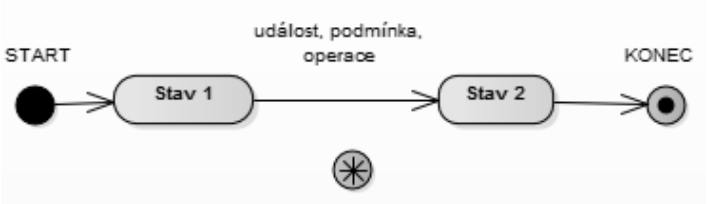


Obr. 6 Diagram tříd

Popis diagramu tříd:
Informace o dané třídě často obsahují název objektové třídy, který bývá umístěn v horní části třídy. Dále atributy nebo parametry třídy s jejich názvy (mohou zde být obsaženy i jejich datové typy), které bývají umístěné ve střední části. A také metody, které s atributy třídy pracují. V dalších krocích potom můžeme přimodelovat přidružené třídy s jejich popisy a přidat je do asociace s ostatními třídami, podle vztahů, jaký mezi sebou mají, to se značí plnou čarou.

2.3.6 Stavové diagramy

Předpokládá se, že systém má omezený počet stavů, do kterých se může dostat a do kterých může přecházet. Stavy si můžeme představit, jako situaci v době trvání objektu, která splňuje vhodné podmínky pro udržení této situace. Anebo jako nějakou operaci nebo čekání na událost.



Obr. 7 Stavový diagram

Popis stavových diagramů:
Stavové diagramy se popisují obdobně, jako diagramy aktivit (kapitola 2.3.1.). Tzn., že mají začátek a konec stavu, zaoblené obdélníky prezentující jednotlivé stavy systému a jednotlivé přechody zobrazené šipkami znázorňující informační toky mezi jednotlivými stavy. Informační toky je možné pro přehlednost v diagramu sjednotit do jednoho toku, to se děje prostřednictvím kruhem s hvězdou.

2.4 Systémový proces – školní potřeba

Je proces, který slouží k obstarávání školního daru ze strany návštěvníků systému. Když se návštěvník dostane na příslušnou stránku, může podat žádost o poskytnutí daru pro školu. Následně se tato žádost rozešle na příslušné strany (zájemci a pro škole) a administrátor poté provede závěrečné kroky a přiřadí dárce k příslušnému daru.

2.4.1 Specifikace požadavků

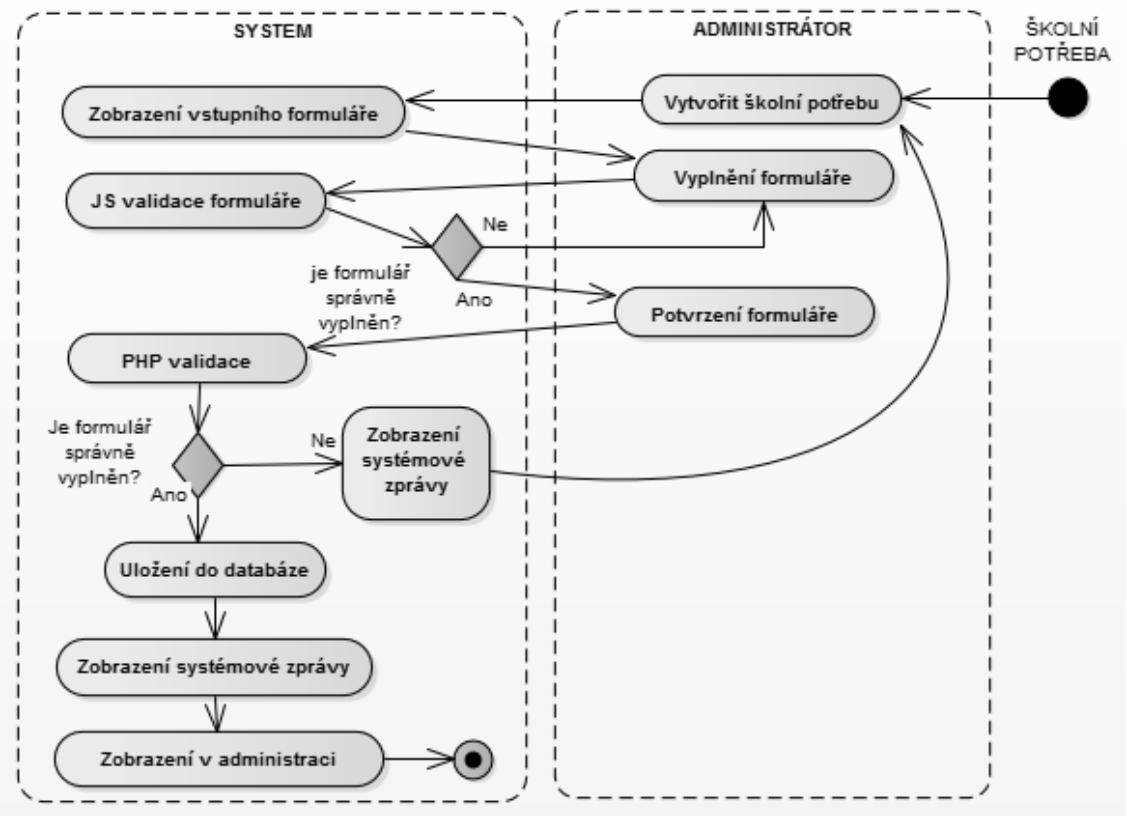
Funkční požadavky:
Mezi hlavní požadavky spadají jednoznačně *vytvoření* školní potřeby a *podání žádosti* o příslušnou školní potřebu ze strany návštěvníka. Sekundárně potom správa školních potřeb, jako je jejich *editace* a *mazání* z databáze. Dalším přáním zákazníka byla možnost vytvářet různé typy potřeb, které se ke školním potřebám mohou přiřadit a také možnost mít přehled všech školních potřeb pohromadě a mít možnost jejich *hromadného mazání*.

Nefunkční požadavky:
Tady byly získány informace, aby se školní požadavky chovaly podle toho, kolik obsahují kusů a jestli je na nich už někdo přihlášen. V případě, že obsahují více kusů než nula anebo nekonečně mnoho kusů a není na školní potřebu přihlášen žádný dobrovolník, bude tato šk. potřeba vykazovat otevřený stav. Když počet kusů zůstane rozdílný od nuly a bude mít vyplněného zájemce, bude stále možné se ke školní potřebě přihlásit, ale bude viditelně rozdílný stav, oproti předchozímu stavu. Poslední možností je signál, kdy je školní potřeba uzavřena a není možné se na ni přihlásit. To se děje, když má alespoň přihlášeného jednoho dobrovolníka a počet nula kusů.

2.4.2 Diagram aktivit

Na obr. 8 můžeme vidět, jakým způsobem probíhá první část systémového procesu a tou je vytvoření školní potřeby. Nejdříve je potřeba vyvolat požadavek na vytvoření školní potřeby, o kterou administrátor v systému požádá prostřednictvím volby v*ytvořit školní potřebu*. To se děje prostřednictvím tlačítka, které pak nabídne administrátorovi formulář pro její vytvoření.

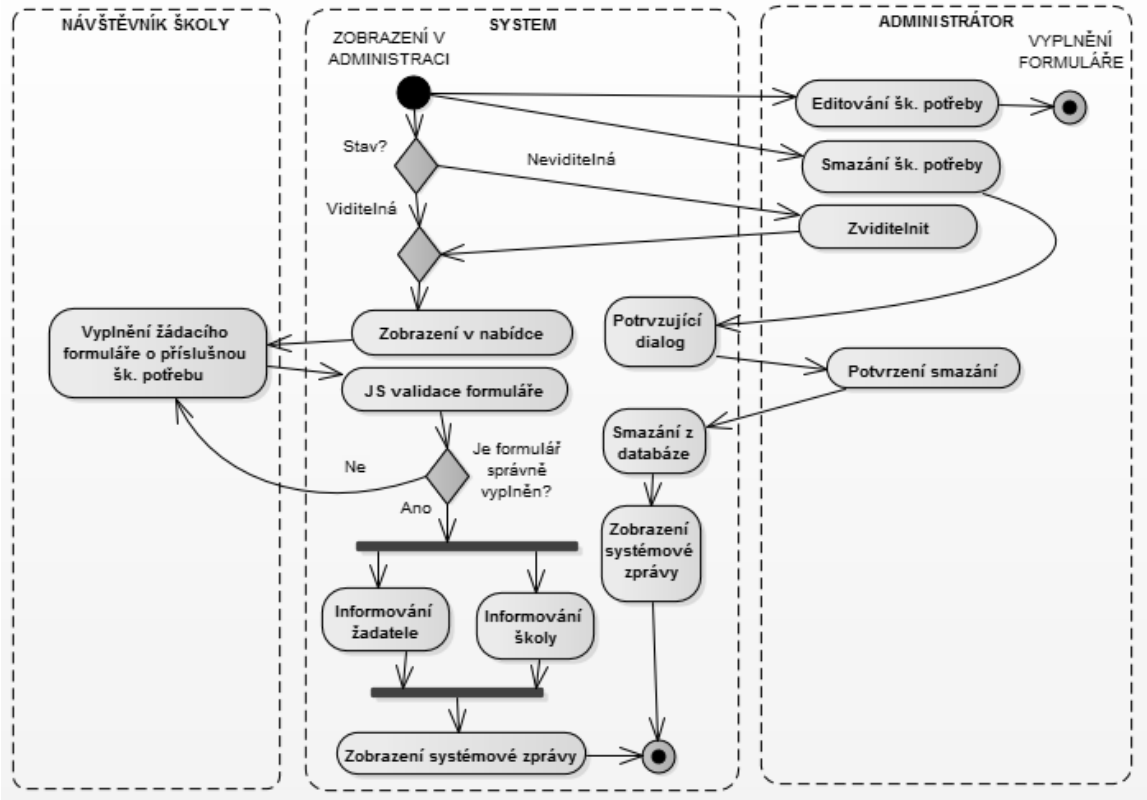
Po vyplnění formuláře se zadané hodnoty zkontrolují JS a po správném odeslání formuláře se vše zvaliduje na straně server a poté uloží do databáze. Závěrem této tvořící etapy je zobrazení systémové zprávy, která oznamuje úspěšné vytvoření šk. potřeby.



Obr. 8 Diagram aktivity procesu – školní potřeby: vytváření

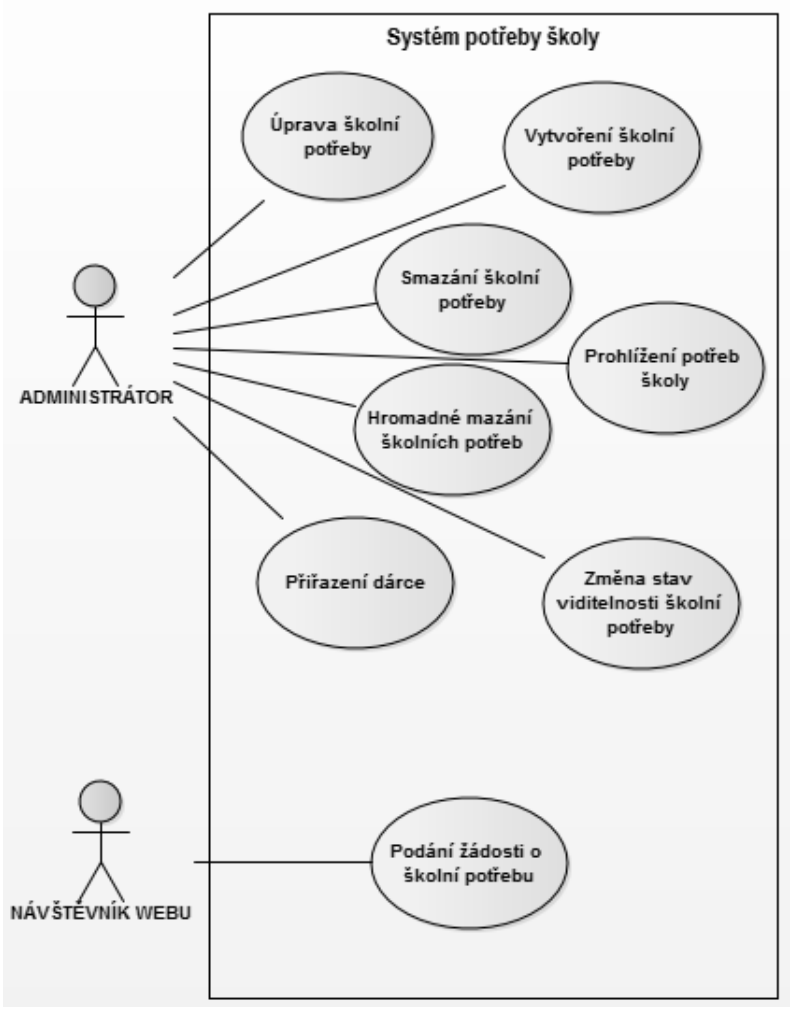
Na obr. 9 můžeme sledovat, co se dále děje se školní potřebou. V administraci se položka zobrazí se všemi důležitými hodnotami (jakého typu je školní potřeba, kolik čítá kusů, její název, podrobný popisek, dárce a zobrazovaný stav položky). Zde je možné šk. potřebu spravovat dle potřeby. Nabízí se zde také možnosti k její editaci a mazání. U volby mazání je nutné tuto možnost potvrdit v dialogovém okně, které ošetřuje neopatrné zacházení. Dále je možné u školní potřeby měnit její viditelnost, což se projevuje zobrazením nebo skrytím v nabídce školních potřeb na veřejné části webu, ke které přistupují návštěvníci.

Hlavní příčinou vzniku školních potřeb je jejich plnění ze strany návštěvníků. Návštěvníci zde proto mají možnost požádat prostřednictvím žádacího formuláře, o kterou školní potřebu mají zájem a tu poté darovat škole. Tady se opět vyplňující formulář kontroluje pomocí JS, aby se předešlo nechtěnému odeslání špatně vyplněných hodnot. Po odeslání a správné validaci formuláře systém automaticky rozešle emaily, jak pro návštěvníka školy, tak na školní email. Tyto emaily nesou informace o žádané školní potřebě, kontaktní údaje dárce z formuláře a kontakt školu, kdyby bylo cokoliv potřeba dále řešit. Závěrem celého procesu je manuální editační krok, pomocí které administrátor v systému přiřadí požadovaného nebo požadované dárce k příslušné školní potřebě.



Obr. 9 Diagram aktivity procesu – školní potřeby: zobrazení v administraci, podání žádosti

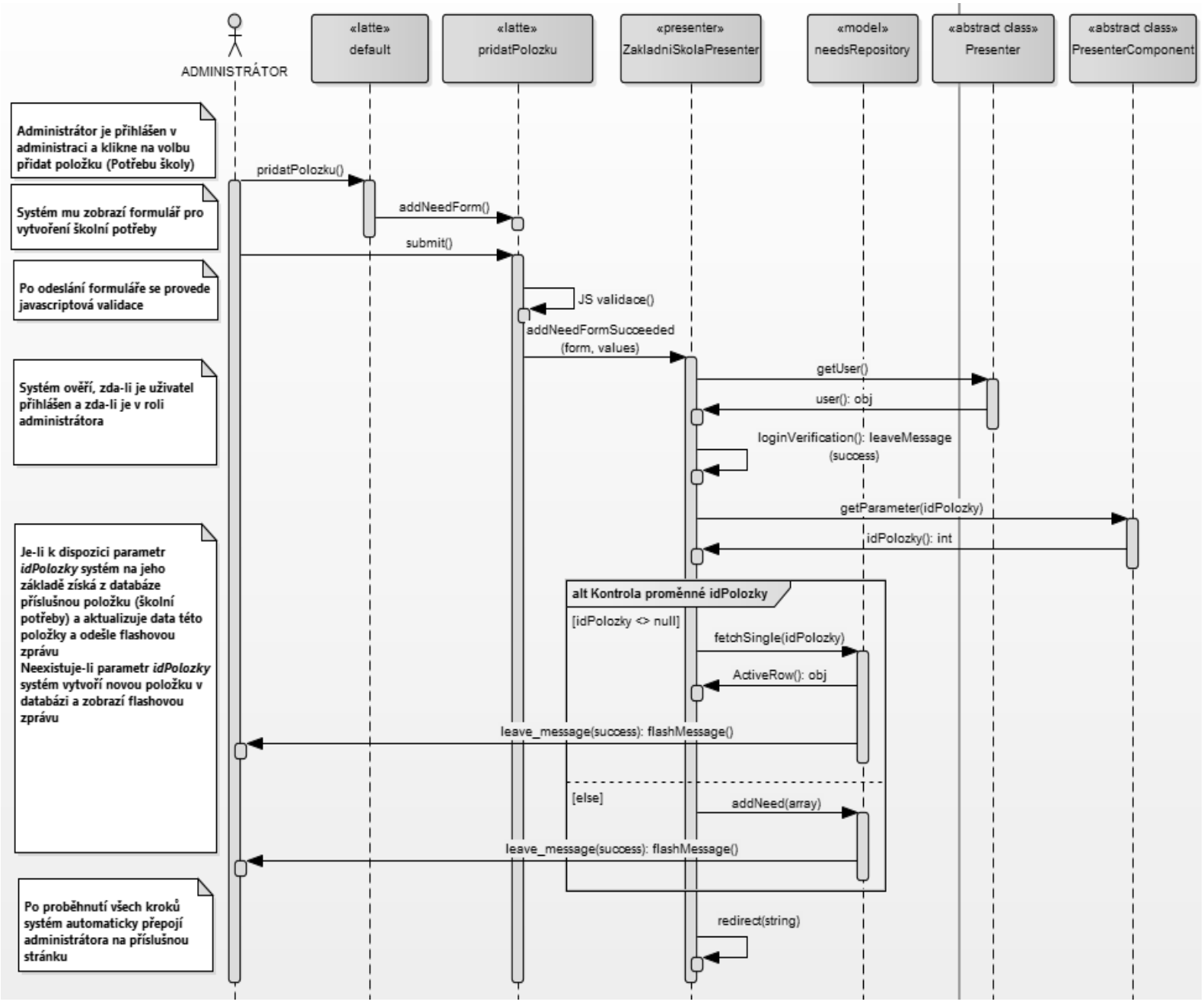
2.4.3 Diagram případu užití



Obr. 10 Diagram případu užití pro systémový proces – školní potřeba

V diagramu případu užití vystupují jednotlivý aktéři, kteří s daným systémovým procesem interagují. Mezi aktéry patří *administrátor*. Ten se musí nejprve do systému přihlásit a poté může se zobrazenými činnostmi operovat, vyjma možnosti přiřazení dárce, která se děje manuálně pomocí úpravy školní potřeby. Další aktérem je *návštěvník webu*, který podá žádost o příslušnou školní potřebu. Vše je zaštítěno poslední aktérem, kterým je samotný *systém potřeby školy*.

2.4.4 Sekvenční diagram pro případ užití – Vytvoření školní potřeby



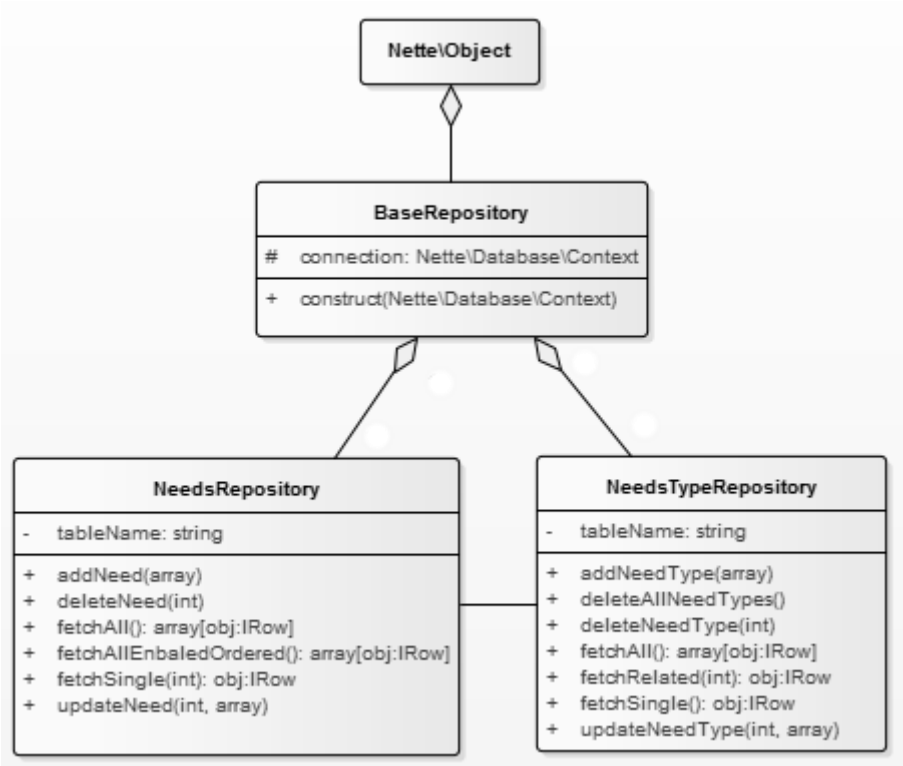
Obr. 11 Sekvenční diagram pro případ užití – Vytvoření školní potřeby

2.4.5 Diagram tříd

Diagram na obr. 12 zobrazuje vrstvu modelu architektury MVP (více v kapitole 4.3.1). Můžeme si všimnout, že zde není zobrazena pouze jedna třída *NeedsRepository*, která se pojí s databázovou třídou školní potřeba. Ale je zde spousta dalších tříd, které jsou s ní spojeny. Základní třídou je třída *BaseRepository*, která pomocí konstruktoru otvírá nad databází spojení. Od této třídy pak dědí všechny ostatní, které potřebují přístup k databázi.

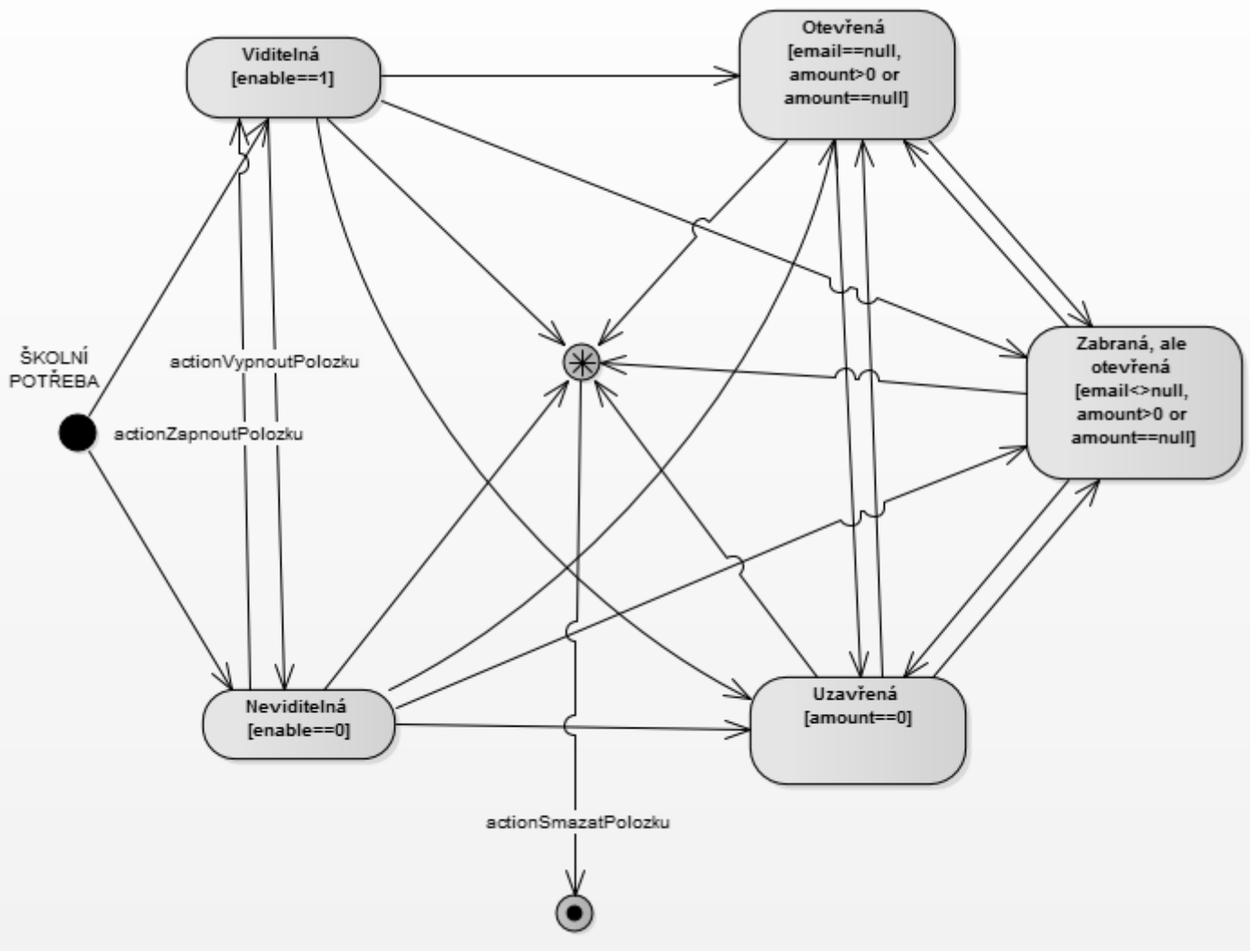
Při volání libovolných metod třídy *NeedsRepository*. Ať je to vytvoření nové školní potřeby, její editace či smazání apod. Vždy její konstruktor udržuje spojení nad databází. Nette vytvoří vlastní ovladač, který detekuje nastavení databázového serveru podle DNS jména a podle toho zvolí své chování. V tomto případě je to MySQL ovladač. Podle nastavení parametru *lazy* v souboru config.local.neon se určí, zda připojení k databázi bude navázáno jen tehdy, když bude skutečně potřeba anebo hned při vytvoření instance.

Dále pro školní potřebu můžeme v systému přiřadit její typ, který představuje třída *NeedsTypeRepository*. Vazba mezi třídami je typu asociace, protože zde objekty obou tříd mohou existovat nezávisle na sobě.



Obr. 12 Model systémového procesu – školní potřeba – zobrazení pomocí diagramu tříd

2.4.6 Stavový diagram



Obr. 13 Stavový diagram systémového procesu – školní potřeba

Systémový proces školní potřeby, začíná svou stavovou činnost ihned po vytvoření a může skončit přímo v některých ze dvou možných stavů *viditelná* a *neviditelná*. Tohle rozhodnutí závisí na administrátorovi, který při vytváření školní potřeby volí možnost, aby byla šk. potřeba viditelná již po vytvoření. Jinak je šk. potřeba neviditelná. Dále je možné mezi těmito stavy libovolně přepínat pomocí funkcí v administraci.

Zbylé stavy vznikají a přepínají se mezi sebou zcela automaticky, a to v závislosti počtu kusů a vyplněných emailech dárců pro danou školní potřebu. Celý cyklus stavů končí po smazání školní potřeby z databáze.

2.4.7 Validace procesu

Vývoj systémového procesu školní potřeba byl testován za použití ladícího a testovacího nástroje Tracy (detailně popsán v kapitole 4.3.4), který je poskytován ze strany frameworku. Běh jednotlivých fází systémového procesu byl otestován na základě sad reálných dat vkládaných do systému.

2.4.8 Další vývoj systémového procesu

Dle uvedených specifikací požadavků (v kapitole 2.4.1.) zákazník prozatím neplánuje další možný budoucí vývoj tohoto systémového procesu.

2.5 Schéma databáze

Databázové schéma (obr. 12) je navrženo na základě požadavků získaných od zákazníka. Protože systém pracuje převážně se stránkami a články, považuji z tohoto pohledu za podstatné uvést tabulku *pages*, jako první. Při vytváření stránek nebo článků musí být zadán název (title. V případě vytváření stránky se musí vybrat typ stránky, kde se jednotlivé typy berou z tabuly *const_pages_type* a jsou neměnné (základní škola, školní družina, mateřská školka, pro rodiče a žáky, školní akce, mezinárodní spolupráce). A v případě vytváření článku se musí zvolit typů článku, ty jsou uloženy v tabulce *const_notifications_type* (základní škola, školní družina, mateřská školka, školní akce). A poté je nutné napsat něco málo do obsahu (content). Nepovinnými údaji jsou datum (created_at), které, když není vloženo tak se automaticky vloží aktuální datum při vytvoření. Dále nadpisek (header), který může nahradit hlavní nadpis, který se původně přebírá z názvu, pak titulní obrázek (image), přílohy (files), kdy u obrázkových příloh se může zvolit, kde přesněji budou zobrazeny. Toto umístění doplňuje tabulka *const_images_positions*. A parametry u článku, pro funkci odeslání newsletteru (has_newsletter) nebo zobrazení (enable) stránky/článku na veřejné části ihned po vytvoření. Další tabulkou, která je v relaci s výše zmíněnou tabulkou (pro typy článku) je tabulka *notifications*. Ta uchovává data o školních oznámení, u kterých je nutno zadat název (title), jejich datum ukončení (final_date) a také jejich typ, o ten se stará výše zmíněná tabulka *const_notifications_type*. Vložení obsahu (content) je nepovinné, protože se zde jedná o doplňující popisek ke školnímu oznámení. Nepovinné také jsou parametry určující odeslání newsletteru (has_newsletetr) nebo zobrazení ve výpisu školních oznámení (enable) nebo v kalendáři (enable_calendar).

Tabulka *users* se využívá pro uchovávání dat o uživatelích systému, kterými jsou administrátor a návštěvník. To o jakou roli se jedná má na starost tabulka *users_role*. V případě návštěvníka evidovaná emailová adresa slouží k rozesílání newsletteru, kdy odeslání newseletteru se povede, pokud je návštěvník v systému povolený. Jednotlivé emaily návštěvníků jsou zde unikátní, aby nedošlo k duplicitám při odesílání newsletteru.

Kdyby nebylo zaměstnanců, tak by nebylo ani školy a proto další popisovanou tabulkou, je tabulka zaměstnanců *employees*. V ní jsou uchovávány záznamy o všech

zaměstnancích vyskytujících se v budově školy a mezi povinné údaje patří jméno (name), příjmení (surname), pohlaví, které je získáváno z tabulky *const_genders_type* a ze kterého jsou odvozovány rody pro jednotlivé role, a pak typ zaměstnance, který je uložen v tabulce *employees_type*. Doplňujícími daty k zaměstnancům jsou telefon (phone), email a detailní popisek (description), který může napovědět více o příslušném zaměstnanci.

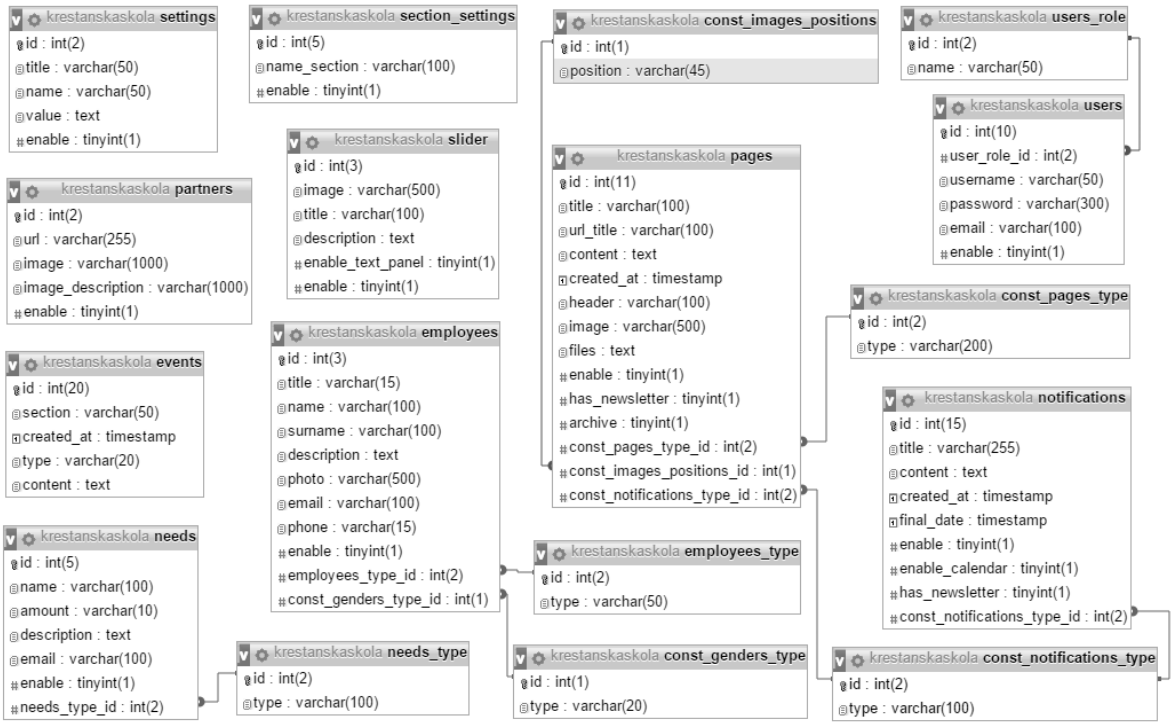
Dále systém eviduje školní potřeby v tabulce *needs*. Při jejich vytváření je nutné vložit jméno (name), jejich množství (amount) a typ, který se uchovává v tabulce *needs_type*. Emaily jsou vkládány do systému manuálně administrátorem, pro lepší manipulaci s daty. A to na základě kopie emailu s žádostí o poskytnutí daru ze strany návštěvníka systému.

V tabulce *slider* jsou uloženy obrázky, které zobrazuje slider na veřejné části systému. U jednotlivých obrázků se může zobrazit textový panel (enable_text_panel), kterým můžeme detailně popsat obrázek. Popis je možný titulkem (title) a popiskem (description).

Serverové události a různé systémové zprávy jsou evidovány v tabulce events. Rozlišují se podle sekce, ve které byly spuštěny a podle typu (úspěšné, varovné, neúspěšné).

Sponzoři a partneři školy jsou ukládáni do tabulky *partners*. Povinnou hodnotou je url adresa, podle které se lze na partnery odkázat. Nepovinnými poli je logo partnera, protože když není dodáno, bude partner zobrazen jen pomocí url adresy. A dále popisek loga, který slouží pro detailní popisek loga.

Nastavení systému se uchovává v tabulce settings, kde jsou pod unikátními názvy (title) uloženy texty (akreditace, vize, email pro newsletter, email pro školní potřeby apod.). S nastavením souvisí i tabulka *section_settings*, kde jsou uložena nastavení viditelnosti, která se váží na různé systémové moduly, které toto nastavení vyžadují z požadavků zákazníka.



Obr. 14 ERD – Schéma databáze

3 Grafický design IS

Při navrhování grafické podoby softwaru je žádoucí zdržet se subjektivního přemýšlení a pokud možno svůj osobní názor na uzdě. To z toho důvodu, že zde již vidíme vizuální podobu systému, u které můžeme velmi jednoduše říci, jestli se nám líbí anebo nelíbí. Vhodné tedy je se snažit navrhovat takový design, který bude přínosný (užitečný, smysluplný) pro daný projekt či nikoliv. Návrh grafického designu by se dal proložit do následujících kroků.

- Tvorba wireframu
- Grafický návrh systému

3.1 Wireframe

Patří mezi první kroky návrhu grafického designu a jedná se o prototyp nebo jinými slovy drátěný model systému, který se potom potáhne navrhovanou grafikou. Z tohoto modelu by se mělo jasně poznat:

- Kde a jaký obsah bude na dané stránce umístěn
- Obsahová vizuální priorita prvků na stránce
- Vztahy mezi jednotlivými části obsahu

Cílem wireframu je rozvrhnout obsah jednotlivých stránek systému tak, aby se návštěvník neztrácel, byl rychle zorientovaný a mohl provést pokud možno, co nejrychleji a nejpřirozeněji akci, které má pro něj přínos např.: přečíst si článek, zjistit informaci, co se kde ve škole děje apod. (Řezáč J, 2014)

3.1.1 Popis

Dle obr. 15 můžeme vidět, že dle zvyklostí, tak i v tomto případě je logo umístěno v hlavičce webu po levé straně. Napravo od něj je umístěna akreditace (text opravňující k provozu činnosti) školy.

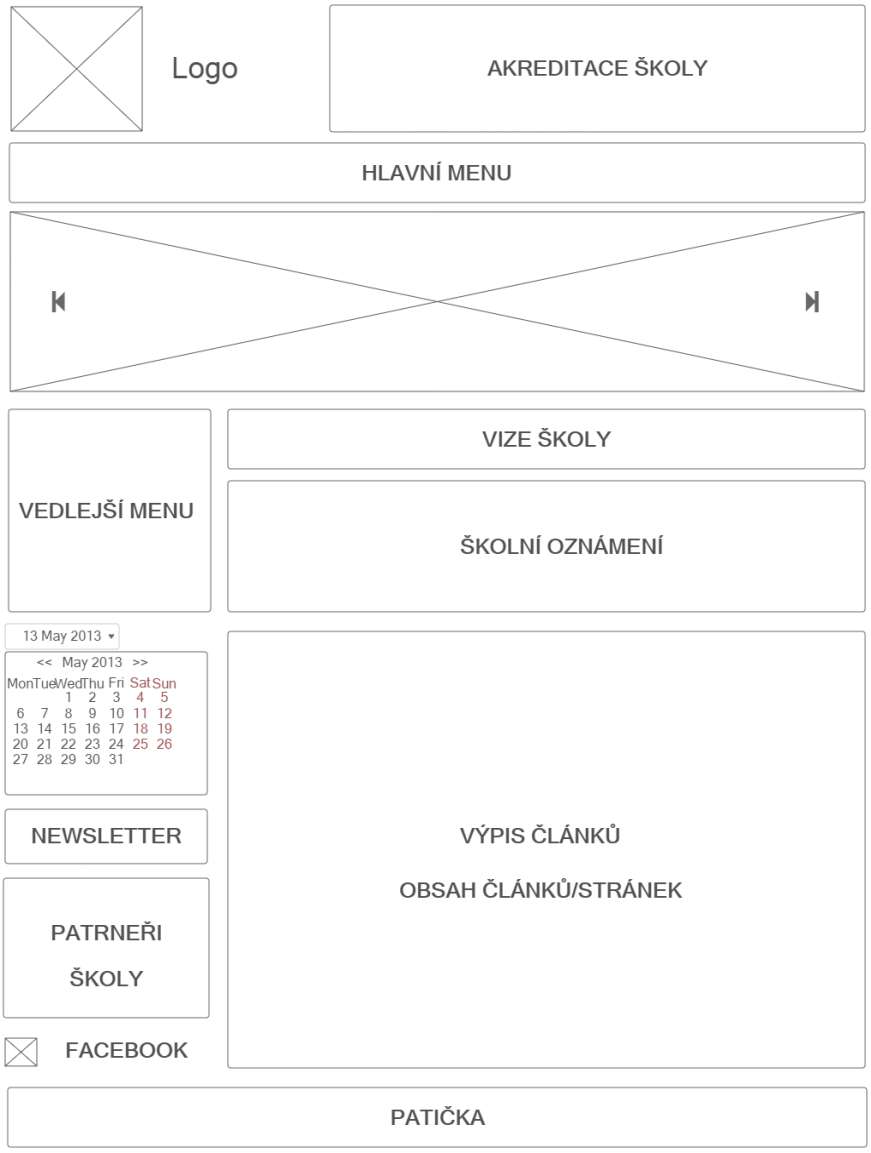
Hlavní menu, které bude ležet pod hlavičkou menu, bude zabírat celou šířku bloku a bude nabízet návštěvníkům přehled nejdůležitějších možností a to naše potřeby, ke stažení, kde bude přehled nejdůležitějších školních dokumentů ke stažení, kontakty a archiv se staršími články.

Slider s prezentačními obrázky školy bude umístěn ihned pod hlavním menu, aby měli návštěvníci povědomí o tom, že se škola nevěnuje pouze standartní výuce. Ihned pod sliderem se bude nacházet hlavní blok systému, který je rozdělený na levý a pravý panel.

Levý panel bude obsahovat vedlejší menu se stránek, které budou zařazeny pod nabídky institucí školy jako základní škola, školní družina a mateřská školka a dále hlavních informačních bloků, které budou pro rodiče a žáky, školní akce a mezinárodní spolupráce. Pod vedlejším menu se bude nacházet kalendář s barevně označenými školními akcemi. Dále zde bude panel s možností odběru newsletteru

pro návštěvníky systému. Ihned pod ním bude umístěn panel s předními partnery a sponzory školy a nakonec odkaz pro sociální síť `facebook.com`.
V pravém panelu bude jako první zobrazena vize školy, která bude informovat návštěvníky o tom, jakým způsobem škola směřuje styl výuky. Dále se zde bude nacházet panel se školními oznámeními informující veřejnost krátkými zprávami ze školního prostředí. Tyto dva panely budou viditelné pouze na hlavní straně systému z toho důvodu, aby nerušili návštěvníka při čtení stránky nebo článku. Pod těmito panely se bude na hlavní straně zobrazovat výpis článků. A po otevření stránky nebo článku se zde zobrazí jejich obsah.

Vše bude uzavřené patičkou, ve které bude copyright text a skrytý odkaz pro přihlášení do administrace.



Obr. 15 Wireframe systému

3.2 Grafický návrh systému

Grafická podoba celého systému byla navrhována v širším barevném spektru. To proto, aby byla vyjádřena dětská hravost a také, aby byla vyjádřena neustálá změna v chování dítěte.

3.2.1 Veřejná část

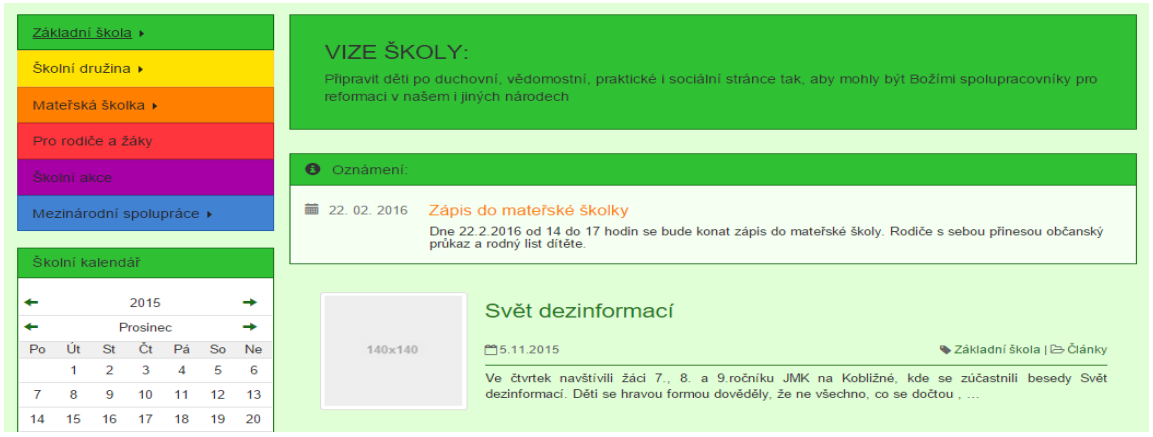
Veřejná část systému přístupná převážně pro návštěvníky by se dala rozložit do několika grafických bloků.

Prvním blokem, se kterým návštěvník systému přijde do kontaktu, je hlavička. Tato část se nese v tmavších odstínech modré barvy, které nepřitahují tolik pozornosti, s cihlovitou texturou vyjadřující školní zeď. To proto, že ve školách převážně na zdech s nástěnkami bývají umístěné podstatné informace. A také proto, že zde není umístěn, tak informačně důležitý obsah, jako v následujícím bloku.

Obr. 16 Grafický návrh – hlavička systému

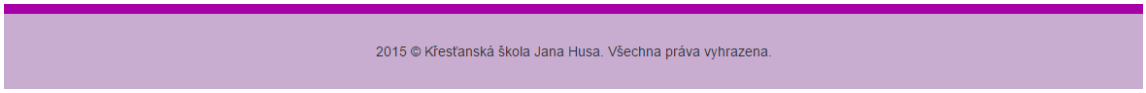
Po první bloku dále následuje obsahová část systému, která je vykreslována v odstínech zelených barev. A to převážně proto, že když se zde uživatelova pozornost zastaví na mnohem delší dobu, bývá následkem možný únava očí. A dalším důvodem je to, že zelené barvy mají uklidňující účinek na mysl pozorovatele.

V úvodu kapitoly byla zmíněna dětská hravost, která je zde podpořena změnou barev u jednotlivých pod-stránek. Když se zobrazí pod-stránka z příslušné nabídky z vedlejšího menu, jsou vždy pod-stránky zabarveny do odpovídajícího odstínu příslušné sekce. Tyto odstíny byly vybírány na základě základních barevných složek duhy, která často vyjadřuje radost a hravost prostoru během deště.



Obr. 17 Grafický návrh – tělo systému

Posledním grafickým blokem systému je patička, jenž je navržena v odstínech růžovo-fialové barvy. Barvy jsou opět zvoleny tak, aby nepoutaly příliš návštěvníkovi pozornosti, protože se nejedná o oddíl s důležitými informacemi. Přitom se, ale klade důraz na význam této části díky proužku a rozložení přes celou šířku stránky. To proto, že tento blok graficky celý systém uzamyká do jednoho celku.

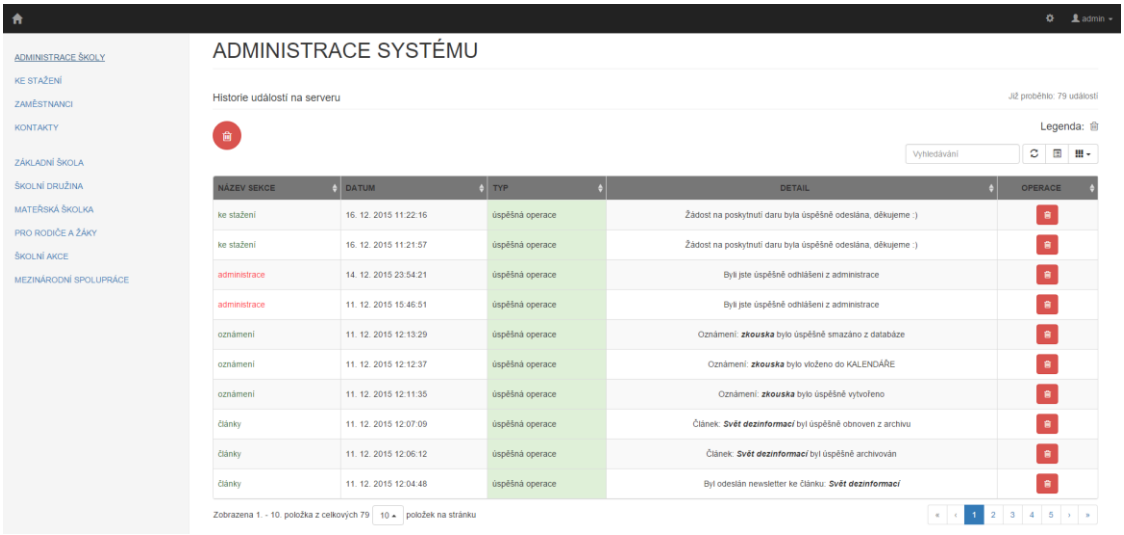


Obr. 18 Grafický návrh – patička systému

3.2.2 Administrace

Grafický návrh administrace systému se nese v jednodušším a přehlednějším podání než veřejná část. To proto, že sem má přístup pouze administrátor systému a ne návštěvníci. Dále je jednodušší z toho důvodu, že tato vlastnost vyzvedá intuitivní ovládání systému, což se projeví na pohodlné manipulaci. A dalším důvodem je přehlednost, kterou systém díky jednoduššímu designu získá s přibývajícím množství dat.

Grafické rozložení administrace je jednodušší než u veřejné části. Podstatná část je horní hlavní menu, které nabízí možnosti pro návrat na veřejnou část systému, zobrazení administrace systému a dále zobrazení jména přihlášeného uživatele systému s příslušnými operacemi (změna hesla, odhlášení ze systému). Administrace se dále rozkládá na levý panel, který je strukturován podle hlavních obsahových celků, které jsou viditelné na veřejné části systému. A na pravý panel, kde se zobrazuje obsah pro položky z levého panelu.



Obr. 19 Grafický návrh – administrace systému

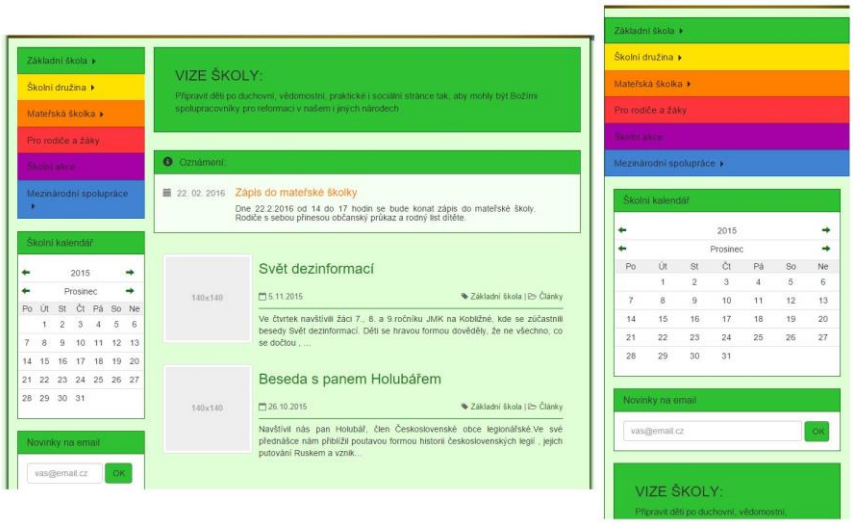
3.2.3 Bootstrap

Bootstrap se podle informací na své internetové stránce (`getbootstrap.com`) řadí mezi světově nejpoužívanější CSS, HTML a JS framework pro vývoj responzivního designu webových projektů s důrazem na „mobile first" metodiku. Používání této metodiky vede k efektivní optimalizaci stránek pro mobilní zařízení. Převážně v dnešní době je to podstatné a velmi žádané z pohledu SEO. Kdy například vyhledávač google od roku 2014 upřednostňuje weby, které mají grafickou optimalizaci pro mobilní zařízení.

Tento framework se řídí se heslem „*one framework, every device*", které můžeme volně přeložit a pochopit jako jednoduchou a účinnou změnu rozlišení (od mobilu, přes tablet až k desktopu) stránek a jednotlivých prvků použitých na stránce za použití jednoho typu kódu. Hlavní předností tohoto frameworku je ve využívání předdefinovaných css tříd a převážně pak responzivního mřížkového sytému („grid system"), který se používá pro vytváření layoutů stránek, což vede k velké efektivitě grafického kódování a šetření času. Mřížkový systém pracuje s řádky, které slouží pro vytváření horizontálních skupin sloupců pomocí třídy *.row*. Pak pracuje se sloupci, kterých může být v jednom řádku až 12. Do nich se poté vkládá obsah. Sloupce využívají třídu *.col-rozlišení-počet sloupců*. Rozlišení se dělí do následujících tříd:

- *col-xs* pro extra malá zařízení (mobilní telefony) s rozlišením < 768px
- *col-sm* pro malá zařízení (tablety) s rozlišení ≥ 768px
- *col-md* pro středně velká zařízení (notebook, desktopy) s rozlišením ≥ 992px
- *col-lg* pro velká zařízení (desktopy) s rozlišením ≥ 1200px

(getbootstrap.com, 2015)



Obr. 20 Ukázka bootstrap frameworku – vlevo náhled z desktopu, vpravo náhled z mobilního zařízení

4 Využití jazyka PHP pro tvorbu dynamických webových stránek s využitím MVC frameworků

Protože se zde budeme pouštět do nové kapitoly, opět si vysvětlíme, jak se věci mají, a ukážeme si pár důvodů, které nám zodpoví otázku, proč se v dnešní době začínají čím dál více používat PHP frameworky. Dále je proveden průzkum, který ukazuje dneska nejvíce používané frameworky. U těch nejvíce používaných si představíme jejich přednosti a odlišnosti od ostatních.

4.1 Proč používat PHP Frameworky

Pojďme se podívat na pěknou řadu praktických důvodů, proč se při vývoji webových projektů v dnešní době čím dál více používají PHP frameworky.

4.1.1 Organizace kódu a souborů

Běžně po dokončení instalaci PHP frameworku, se v adresáři s projektem vytvoří určitá adresářová struktura, která tímto způsobem očekává, že se bude dále dodržovat. Ale ne vždy, je to nutností, tady záleží na vývojáři.

Přidávání nových modulů roztříděných do jednotlivých souborů se tímto způsobem stává efektivním. A při použití kvalitního IDE vývojového prostředí s full textovým vyhledáváním je psaní organizovaného kódu jistým způsobem (každý framework má vše organizované odlišným způsobem) velmi výkonné a svižné.

4.1.2 Knihovny a addony

PHP je obecně velmi rozšířený programovací jazyk používaný pro vývoj webových projektů s bezpočtem doplňkových nástrojů a knihoven. Nicméně pokud se někdo snaží vyvíjet webový projekt sám od začátku. Většinou se potýká s vybíráním mezi spoustou knihoven třetích stran nebo si je píše sám. Z tohoto důvodu jsou zde PHP frameworky, které nabízí spoustu užitečných knihoven pomocných tříd. Mezi jejichž nejčastější funkcionalitu patří:

- Validace formulářů
- Filtrování vstupů a výstupů
- Přístup k databázové vrstvě
- Správa sessions a coockies
- Email systém, správa kalendáře, stránkování obsahu atd.

4.1.3 Architektura MVC

PHP samo osobě pracuje, jako šablonovací systém, který nešťastně vede k nepřehlednému a neudržitelnému kódu u rozsáhlejších projektů. Proto byla navržena architektura MVC, které rozděluje projekt na tři části:

- *Modely* (Models) jsou vstupní datovou vrstvou, která prezentuje datové struktury v projektu, které se pojí a pracují s databází.
- *Pohledy* (Views) obsahuje jednotlivé šablony a v nich vykreslený obsah. Působí jako výstupní front-end vrstva architektury.
- *Kontroléry* (Controllers) jsou mezi vrstvou, která spojuje dvě výše zmíněné vrstvy. Zpracovává data z databáze a přeposílá je na výstupní vrstvu.

Další výhodou architektury MVC je možnost rozdělení tvorby projektu do jednotlivých částí, tak aby každá byla zpracovávána jiným týmem a zároveň si týmy nijak nepřekáželi.

4.1.4 Bezpečnost

Jazyk PHP běžně pracuje s mnoha vstupními a výstupním filtrovacími funkcemi, které slouží k zabezpečení webových projektů před různými útoky. V případě ručního ošetřování vstupů a výstupů velmi často dochází z důvodu jejich velkého množství k chybovosti nebo opomenutí ošetření, některé části. V případě PHP frameworků se tohle děje zcela automaticky na základě zabudovaného ošetření vstupních a výstupních prvků do systému.

4.1.5 Méně kódu a rychlejší vývoj

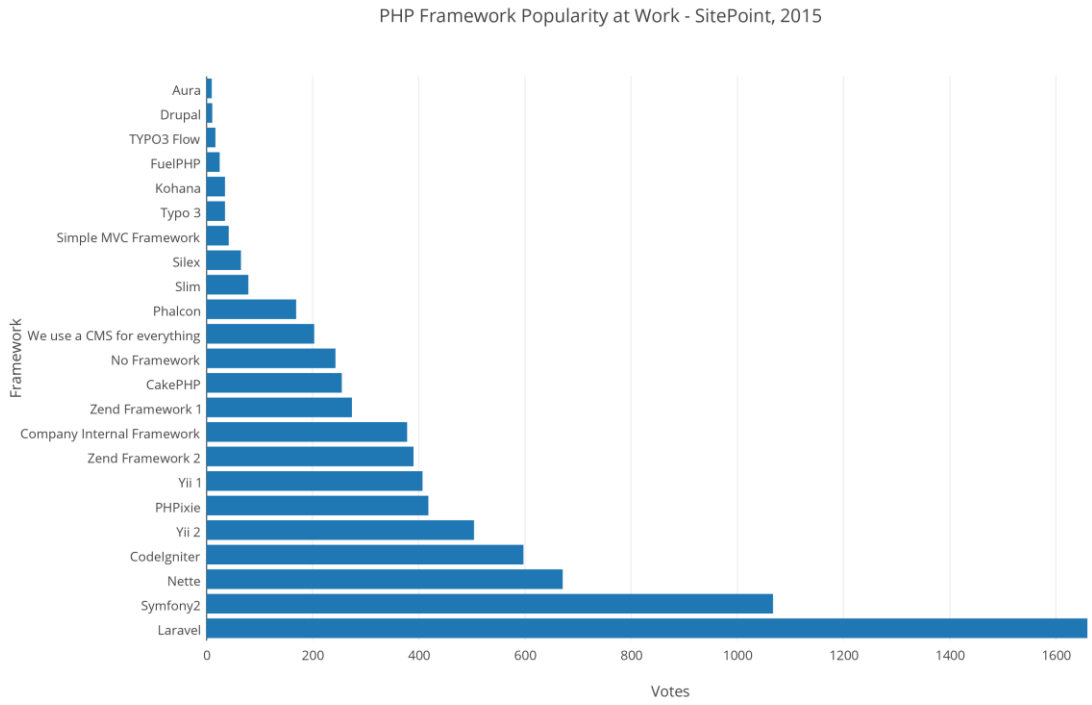
Učit se od začátku jakýkoliv PHP framework zabere spoustu času, než se člověk dostane přes základní funkcionalitu a objeví výhody, které framework nabízí. Ale po této fázi přichází velmi zrychlený vývoj aplikace, protože spousta funkcí je u frameworků již vyřešena. K tomu přispívá i adresářové organizace a čistě psaný kód, což vede k velké efektivitě vývoje a časové úlevě.

4.1.6 Podpora ze strany komunity

Protože většina nejznámějších PHP frameworků má kolem sebe velkou aktivní komunitu lidí, kteří s frameworkem pracují, vývojářů atd. Můžete se přímo jich zeptat na diskuzních fórech nebo požádat o pomoc či rady a vyřešit tak rychle spoustu svých problémů. (phpandstuff.com, 2015)

4.2 Nejpoužívanější PHP frameworky

Podle internetové ankety Best PHP Framework 2015 pořádané magazínem sitepoint.com se do pěti nejpoužívanějších PHP frameworků, které se využívají ve firemní sféře, dostaly frameworky, které můžeme vidět na obr. 20.



Obr. 21 Přehled nejvíce používaných PHP frameworků (sitepoint.com, 2015)

4.2.1 Laravel

Dle ankety nejpoužívanější framework v roce 2015. Je to open-source poskytovaný zdarma pod licencí MIT. Laravel se vyznačuje hlavně optimalizací pro reálný svět okolo nás, čímž velmi usnadňuje práci vývojářům, která nabízí často používané metody, které jsou nutné při vývoji webových aplikací. Mezi základní funkční moduly frameworku Laravel patří zejména:

- *Autentifikace*: kontrolování přístupu uživatelů do systému
- *Routování*: správa a editace přesměrování a zpracovávání dotazů
- *Databáze*: výkonné nástroje pro správu databáze s využitím Eloquent ORM
- *Mailing systém*
- *Sessions modul*
- *Caching*: kešování obsahu stránek

Podstatné je také zmínit, že Laravel obsahuje mimo výše zmíněné další spoustu vestavěných modulů a také užitečnou komponentu Composer (PHP Dependencies Manager), který zajišťuje vztahy addonů v JSON souborech.

Základní požadavky kladené na tento framework je minimální verze PHP 5.4, MCrypt, JSON a nejlépe databáze MySQL (není podmínkou, protože se Laravel umí vypořádat i s jinými typy databází). (laravel.com, 2015)

4.2.2 Symfony2

Je jeden z mála open-source frameworků, který má kolem sebe tak velkou aktivní komunitu vývojářů. Je vyvíjen společností SesionLabs, která postavila hodnoty frameworku na filozofii „*Symfony is a set of PHP Components, a Web Application framework, a Philosophy, and a Community — all working together in harmony*“. Z toho vyplývá i síla Symfony2 jenž je založena právě na komponentách, které umožňují i dílčí používání těchto komponent. Je jednoduše možné vyvíjet projekt postavený z části na Symfony2 a z časti na zcela jiném frameworku.

Mezi základní a nejvíce používané moduly patří obdobně, jako u Laravelu modul poskytující sofistikovanou autorizaci uživatelů (Security modul) vylepšený o jednodušší správu a vytváření komplexních autentizačních systémů (Guard modul). Modul pro mapování http požadavků na sadu konfiguračních proměnných (Routing modul). Dále mezi velmi užitečné moduly spadá modul poskytující nástroje pro validaci tříd v projektu (Validator modul).

Minimální požadavky vyžadované pro instalaci Symfony2 jsou PHP ve verzi 5.5.9, MySQL databáze, dále je nutné mít povolený JSON a CType. (symfony.com, 2015)

4.2.3 Nette

V dnešní době je nette nejvíce rozšířený open-source framework v ČR, kde je vyvíjen českým vývojářem Davidem Grudlem (od roku 2008) společně s organizací Nette Foundation. Praktickou výhodou je velká aktivní komunita vývojářů, která se také částečně podílí na jeho vývoji a převážně na poskytování a vytváření tzv. teoretického backgroundu frameworku (tvorba dokumentace, návodů, obsahu fóra apod.). Framework nette má své přednosti převážně v zabezpečení aplikace během jejího vývoje. Oproti zmíněným frameworkům poskytuje velmi užitečný ladící nástroj Tracy přezdívaný „Laděnka“ (detailní popis v kapitole 4.3), kvalitní formulářové zabezpečení proti různým typům útoků (XSS, CSRF, session hijacking, session fixation atd.) a dál nabízí podporu HTML5, AJAX, Dependency Injection, SEO a znovu použitelnosti kódu podpořenou pomocí přístupů KISS a DRY.

Požadavek pro instalaci nette ze strany PHP je verze 5.3 a vyšší a MySQL databáze. V případě neznalosti běhu prostředí je možné využít oficiální nástroj pro kontrolu požadavků Requirements Checker. (nette.org, 2015)

4.2.4 Codeigniter

Další v pořadí je další open-source malý a výkonný framework (instalace zabírá zhruba 2MB a obejde se téměř bez nutnosti cokoliv nastavovat) disponuje jednoduchým rozhraní pro práci s knihovnami, které řeší časté úkoly vývojářů, jako jsou běžně správa sessions, mail systém, stránkování obsahu, formulářová validace atd. Tohle, ale není moc podstatný rozdíl a tedy i výhoda od výše zmíněných frameworků. Hlavním důvodem, proč se dostal framework v roce 2015 na scénu mezi pětici nejpoužívanější, je díky kvalitně rozsáhlé a přehledně zpracované dokumentaci, která tímto šetří spoustu času začátečníkům při učení se s frameworkem. Výhodou je, že se dokumentace velmi rychle aktualizuje a pokrývá vždy změny z poslední vývojové verze frameworku. Dalším důvodem je velmi jednoduché používání frameworku, díky zabudovanému šablonovacímu systému a v neposlední řadě taky rozsáhlá aktivní komunita vývojářů, která čítá řádově desetitisíce členů.

Jedinými požadavky pro zavedení frameworku je PHP v minimální verzi 5.2.4 a MySQL databáze. (codeigniter.com, 2015)

4.2.5 Yii 2

Je další z řady oblíbených open-source, objektově orientovaných s podporou MVC architektury frameworků. Popularitu získává na základě své jednoduchosti a přehlednosti, kterou se snaží udržovat na základě zkratky v jeho názvu, která znamená „Yes it is“, což v překladu je „Ano je“. Tímto se snaží odpovídat na spoustu základních otázek vývojářů před samostatným vývojem, který framework při vývoji použít. Jako např.: Je rychlý? Je zabezpečený? Je profesionální? Je správný pro můj další projekt? Apod.

Framework Yii je výkonově optimalizovaný pro vývoj aplikací jakýchkoliv rozsahů a poskytuje možnosti konfigurace frameworku od „hlavy až k patě“. Dále v sobě nad rámec obecně užívaných modulů oproti výše zmíněným frameworkům, umožňuje využít automatické generování komplexních WSDL specifikací a správu webových služeb.

Současná verze frameworku vyžaduje PHP ve verzi 5.4 a MySQL. (yiiframework.com, 2015)

4.2.6 Volba frameworku pro implementaci systému

Mou pozornost nejvíce upoutal framework *Codeigniter* díky přehledné a kvalitní dokumentaci, ovšem to u vývoje nebylo vše, co jsem požadoval. Hlavním požadavkem bylo kvalitní zabezpečení celého systému proti různým typům útoků, formulářové zabezpečené a jejich správa. Dále pak efektivní práce s databází a celkový výkon frameworku. Srovnávací testy některých z výše zmíněných frameworků můžete nalézt na adrese `root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/`. Z toho všeho jasně vychází *Nette* ještě podpořený ladícím nástrojem Tracy a také spoustou zkušeností z okruhu mých kamarádů.

4.3 Nette framework

Pro pochopení implementace systému bude vhodné si popsat Nette framework, na kterém je celý systém vyvíjen, jeho uspořádání adresářů ve struktuře, princip činnosti, zabezpečení vstupů a výstupů a různé funkce, které využívá.

Proniknout do podstaty tohoto frameworku není příliš složité jednak proto, že na jeho stránkách (`nette.org`) můžeme nalézt spoustu užitečných rad a návodů, jak začít, praktickou API dokumentaci, přehlednou příručku s nejběžněji používanými funkcemi Nette frameworku a také fórum, kde se řeší a je vyřešena spousta problémů.

Užitečným balíčkem pro začínající projekt, který urychlí jeho start, je využití tzv. „sandboxu“, který framework nabízí. Jedná se o předem vytvořený skeleton nového projektu s připravenou strukturou adresářů a souborů, kde je vše nachystané ihned pro používání. První spuštění aplikace se projeví zobrazením úvodní stránky Nette frameworku.

4.3.1 Struktura adresářů

Struktura adresářů slouží primárně pro lepší přehlednost zdrojových kódů v celé aplikaci. Všechny adresáře nacházející se v projektu mají jistou logiku a každý má svůj účel. Z hlavních adresářů ve struktuře stojí za popsání *app*, *temp*, *log*, *vendor* a *www*.

- **app**
  V tomto adresáři se nachází hlavní logika celé aplikace. Nette si zachovává best-practicies a odděluje tento adresář od přístupu zvenčí, aby nebylo možné soubory napadnout.

  V samotných stejnojmenných podadresářích se nacházejí konfigurační soubory pro celou aplikaci, komponenty ve formě formulářů, které je možné sdílet skrze celou aplikaci, soubor s routami. Také je zde adresář, ve kterém se nachází modelová vrstva aplikace. Dále je zde zavádějící soubor bootstrap.

  Dále se zde nachází složka presenters a templates, zde jsou uloženy, jak už názvy napovídají presentery a šablony, které pracují na úrovni celé aplikace. Proto nejsou zanořeny v žádném dlaším zvláštním adresáři.

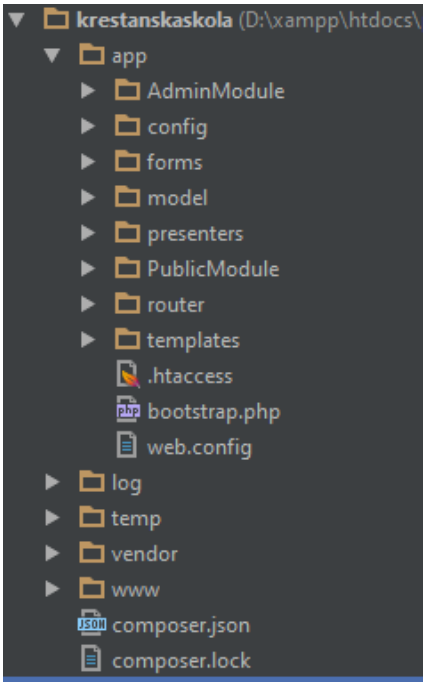
  Samotný sytém je pak rozdělený na dva moduly. *AdminModule*, ve kterém jsou opět adresáře s příslušnými presentery a šablonami. A *PublicModul*, ve kterém jsou uloženy soubory pro veřejnou část systému. Celé rozložení tak působí organizovaně a přehledně.

- **log**
  Zde si Nette loguje chybové hlášky a výpisy, když systém běží na produkčním serveru, aby nebyl rušen návštěvníka webu.

- **temp**
  Adresář je téměř bez-údržbový, protože si sem Nette sám ukládá dočasná data (cash). Když, ale potřebuje většinou na produkčním serveru projevit libovol-

nou změnu například v šablonách. Často se neprojeví z toho důvodu, že má obsah uložený zde, který je třeba nejprve promazat.

- **vendor**
  Zde se nacházejí veškeré soubory nutné pro běh frameworku, knihovny třetích stran apod.
- **www**
  Tento adresář je, jako jediný je dostupný zvenčí. Nachází se na něm CSS soubor, JS soubory, obrázky, uživatelské soubory apod.



Obr. 22 Struktura adresářů

4.3.2 Architektura MVP

Jak již bylo výše zmíněno (v kapitole 4.1.3) frameworky využívají architekturu MVC, ale v případě Nette se pouze jmenovitě jedná o MVP architekturu (Model, View, Presenter). Pojďme si ji tedy blíže popsat.

4.3.3 Modely

Využívání modelové vrstvy v Nette frameworku, která si připojuje a získává data z databáze (ale nemusí jen z ní), je poměrně snadná ale zároveň otevřená záležitost. Pro připojení k databázi se využívá vestavěná Nette třída `Nette\Database\Connection`, která obaluje PDO databázové rozhraní. Pokročilé funkce pro práci s tabulkami zajišťuje třída `Nette\Database\Table`. Další velmi užitečnou třídou, která umožňuje výběr řádků z databázových tabulek a jejich filtrování, je `Nette\Database\Table\Selection`. Jak může vypadat takové připojení

k databázi a výběr všech hodnot z tabulky *settings* můžeme vidět na následujícím kódu.

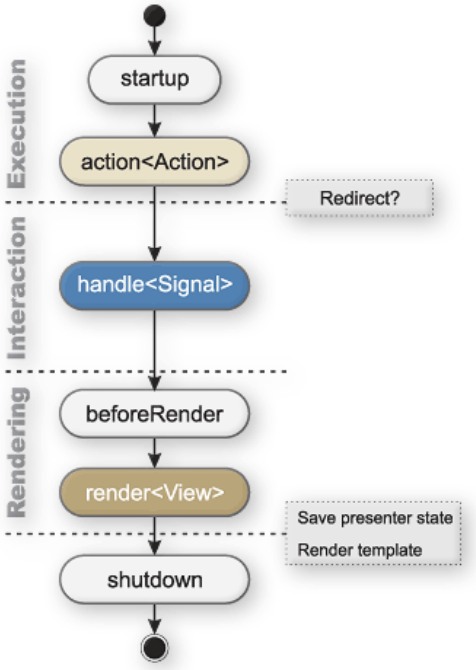
```
protected $connection;
private $tableName = 'settings';

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

public function fetchAll() {
   return $this->connection->table($this->tableName);
}
```

4.3.4 Presentery

Každý HTTP požadavek se dostává přes soubory index.php a bootstrap.php (zavádějící soubor frameworku) až do objektu $application. Ten si s nimi, ale moc neví rady a tak žádá vestavěný modul router, aby požadavky přeložil. Překlad se děje tak, že router zodpoví, pro který presenter je HTTP požadavek určený a jakou akci s ním chce vykonat. Poté se vytvoří objekt daného presenteru a zavolá se příslušná akce (metoda), která může odpovídat například HTML stránkou, obrázkem, souborem na disku, přesměrováním nebo čímkoliv je zapotřebí.

Presentery mají svůj životní cyklus (obr. 23), během kterého můžeme vytvářet různé metody, kterou jsou volány v konkrétních fází tohoto cyklu.

- *startup*
  Je metoda volána ihned po vytvoření presenteru, kdy inicializuje proměnné a ověřuje uživatelská oprávnění (přihlášení do systému apod.).
- *action<Action>*
  Metoda je obdobná, jako render<View>, ale která nic nevykresluje na front-end. Vykonává se dříve než vykreslovací metoda a provádí jen potřebné operace a nejčastěji poté přesměruje jinam.
- *handle<Signal>*
  Metoda zpracovává signály určené zejména pro komponenty a zpracování AJAX požadavků.
- *beforeRender*
  Tato metoda nejčastěji obsahuje nastavení šablony, předává společné proměnné pro více view apod.
- *render<View>*
  Je metoda, která předává potřebná data z modelů do šablon.
- *Shutdown*
  Poslední metoda v životním cyklu presenteru vyvolává jeho ukončení.



Obr. 23 Životní cyklus presenteru (nette.org, 2015)

Dále je možné presentery hierarchicky dělit. Kdy mezi základní presenter se obecně řadí BasePresenter, ve kterém často bývají funkce, ošetření uživatelských přihlášení a oprávnění, vytváření komponent, které jsou společné pro celou aplikaci nebo její dílčí části. Od tohoto presenteru potom ostatní presentery dědí a přebírají tím i jeho metody a vlastnosti.

4.3.5 Šablony

Šablony jsou velkou předností Nette frameworku a neslouží jen pro vykreslování obsahu na stránkách, ale převážně usnadňují práci a zabezpečují výstup před útokem XSS. Příkladem lze uvést následující kód zapsaný v jazyce PHP.

```
<?php if ($items): ?>
  <?php $counter = 1 ?>
  <ul>
  <?php foreach ($items as $item): ?>
          <li id="item-<?php echo $counter++ ?>"><?php
          echo htmlSpecialChars(mb_convert_case($item,
          MB_CASE_TITLE)) ?>
          </li>
  <?php endforeach ?>
  </ul>
<?php endif?>
```

Tento zápis je poměrně nepřehledný a nesmíme zapomenout na „eskejpování“ proměnných, což znamená převedení speciálních HTML znaků na jinou odpovídající sekvenci. Díky tomu vznikají v PHP nejrůznější šablonovací systémy např. Latte. Porovnejme výše uvedený kód s kódem zapsaným v Latte.

```
<ul n:if="$items">
{foreach $items as $item}
  <li id="item-{$iterator->counter}">{$item|capitalize}</li>
{/foreach}
</ul>
```

Jak můžeme vidět zápis je značně zkrácený a více přehledný. U kódu zapsaného v Latte si můžeme všimnout dvojího zvláštního zápisu.

- **makra** ve složených závorkách {foreach}{/foreach},
- **n:makra** například n:if="…"

Každé makro, které je v páru jako výše zmíněný foreach cyklus, a které operuje nad jedním HTML elementem. Můžeme zapsat v podobě n:makra. Výše zmíněný zápis můžeme ještě zkrátit do následující podoby.

```
<ul n:if="$items">
  <li n:foreach="$items as $item">{$item|capitalize}</li>
</ul>
```

Další velkou výhodou šablonovacího systému Latte je eskejpování všech proměnných zcela automaticky. A to i nezávisle na typu dokumentu a na různých umístění v kódu, kdy jsou vyžadovány odlišné eskejpovací funkce. To se děje díky technologii CAE (více v kapitole 4.3.6).

Také si můžeme všimnout u výpisu proměnné `$item|capitalize`, že je rourou oddělena funkce capitalize. Těmto funkcím se v latte šablonách říká filtry a slouží k úpravě nebo přeformátování dat do požadované podoby.

V šablonovacím systému Latte existuje jeden typ šablony, který je hlavní a označuje se jako *layout*. Ten vykresluje obsah pro celou stránku v podobě hlavičky, patičky, nalinkovaných css stylů a javascriptových knihoven. Ve výchozím stavu bývá tato šablona umístěna v adresáři *templates/@layout.latte* mezi ostatními adresáři presenterů (je nadřazená ostatním šablonám). Tato šablona obsahuje makro block.

```
{include content}
```

Tento zápis vkládá do šablony blok kódu s názvem content, který se definuje v ostatních šablonách. Tím se zajistí změna obsahu na hlavním layoutu.

4.3.6 Tracy (debugger)

Jedna z nejvyužívanějších knihoven mezi vývojáři při vývoji aplikací na Nette frameworku je zcela jistě Tracy, pod hovorovým názvem „Laděnka“ pomáhá:

- Odhalovat a opravovat chyby
- Logovat chyby
- Vypisovat proměnné
- Měřit čas

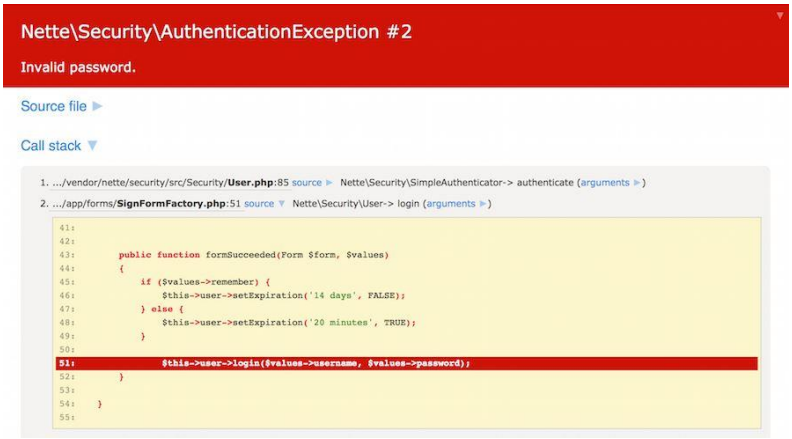
Aktivovat Laděnku můžeme snadno přidáním těchto dvou řádků kódu za místo, kde se načte framework, do souboru bootstrap.php.

```
use Tracy\Debugger;
Debugger::enable();
```

První, co se stane je zobrazení plovoucího debugger panelu na vývojovém prostředí, který pomocí widgetů zobrazuje čas načtení stránky, její velikost, aktuálně zobrazený soubor, počet dotazů nad databází a jejich čas vykonání. Dle potřeby je možné si napsat widgety vlastní.

Obr. 24 Tracy – debugger bar

Když bylo zmíněno vývojové prostředí, tak se s tím pojí i výpis chybových hlášení, které PHP běžně vypisuje do zdrojového kódu stránky a občas se může stát, že vše splývá do jednoho kusu kódu a je špatně čitelné. Z tohoto důvodu Tracy přichází se zcela novým výpisem chyb.



Obr. 25 Tracy – grafická ukázka výpisu chyby

Mimo jiné Laděnka automaticky rozeznává vývojové prostředí od produkčního serveru a veškerý výpis chyb loguje do souborů, aby návštěvník nebyl rušen chybovými výpisy a viděl jen srozumitelnou hlášku, která se dá předdefinovat pomocí latte šablon.

4.3.7 Formuláře a jejich zabezpečení

Formuláře používané v Nette frameworku toho umí opravdu spoustu. Níže jsou popsány základní vlastnosti:

- Validace odeslaných dat na straně serveru i pomocí JS
- Zabezpečení proti zranitelnosti
- Několik režimů vykreslování

- Vícejazyčnost

Velkou výhodou je kvalitní zabezpečení formulářů proti různým typům útoků Cross Site Scripting (XSS) i Cross-Site Request Forgery (CSRF), které jsou popsány v kapitole 4.3.6. Dále odfiltrování kontrolních znaků ze vstupů, ověření UTF-8 kódování apod. Vyhneme se tím řadě problémů a rutinních úkolů, jako třeba psaní dvojí validace (na straně klienta i serveru).

Vytváření formulářů pomocí Nette je opravdu velmi jednoduché. Níže je ukázka kódu pro vytvoření přihlašovacího formuláře do systému. (nette.org, 2015)

```
protected function createComponentSignForm() {
   $form = new Form;

   $form->addText('username')
      ->setRequired('Napiště prosím vaše přihlašovací jméno');
   $form->addPassword('password')
      ->setRequired('Napište prosím heslo, na jehož základě budete roz-
poznáni')
      ->addRule(Form::MIN_LENGTH, 'Heslo musí být minimálně %d znaky
dlouhé', 4);
   $form->addPassword('passwordVerify')
      ->setRequired('Pro kontrolu napište prosím znovu vaše heslo')
      ->addRule(Form::EQUAL, 'Hesla se neshodují. Zkuste to znovu',
$form['password']);

   $form->addSubmit('cancel');
   $form->addSubmit('send');

   $form->addProtection('Vypršel ochranný časový limit, odešlete prosím
formulář ještě jednou');

   $form->onSuccess[] = array($this, 'signFormSucceeded');
   return $form;
}
```

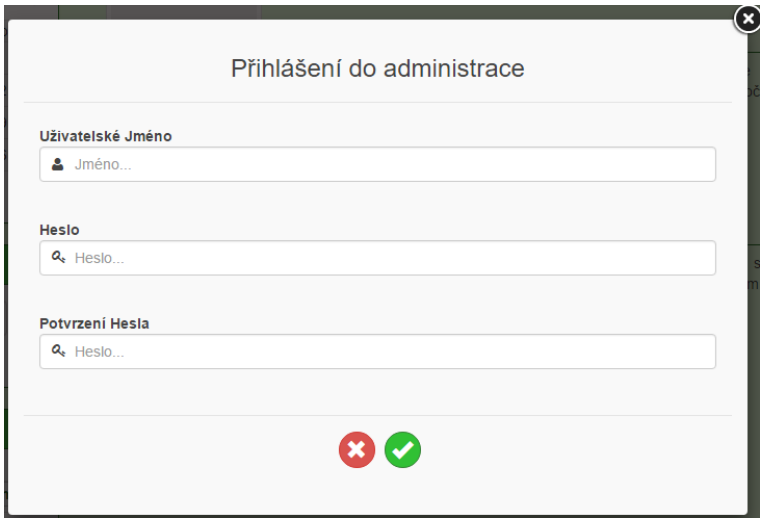
V metodě si můžeme všimnout použitých různých validačních pravidel. Ta nám umožňují ošetřovat formulářové prvky, které můžou být povinné nebo musí obsahovat pouze celá čísla, čísla v určitém rozsahu apod. Výpis chybových hlášení může být nastavení s původními hodnotami nebo vlastní.

Na konci metody si můžeme všimnout metody *signFormSucceeded*, která se zavolá po odeslání a úspěšné validaci formuláře. Jako parametr dostává tato metoda samotný formulář s jeho hodnotami. A pokud si přejeme získat místo objektu pole, zavoláme funkci `$values = $form->values`.

```
public function signFormSucceeded($form) {
   $values = $form->values;

   try {
      $this->getUser()->login($values->username, $values->password);
      $this->redirect(':Admin:Admin:');
      $this->leaveMessage('success', 'glyphicon glyphicon-ok', 'přihlá-
šení do administrace',
         'Vítejte v administraci <strong><em>' . $values->username .
'</em></strong>!');
   } catch (Nette\Security\AuthenticationException $e) {
      $this->leaveMessage('danger', 'glyphicon glyphicon-info-sign',
```

```
                'přihlášení do administrace', $e->getMessage());
        }
    }
```

Samotný formulář se poté v šabloně vykreslí pomocí následujícího makra `{form signForm class => "form-horizontal", autocomplete => "off"}`. A protože se v práci nevyužívá pro vykreslování formulářů jejich výchozí vzhled, ale je vše založeno na bootstrap frameworku. Děje se jejich vykreslování manuálně s větší kontrolou nad samotným kódem (ukázka kódu v příloze xxx).



Obr. 26 Vykreslení formuláře postaveném na bootstrapu

4.3.8 Bezpečnost

Jak již bylo zmíněno, tak Nette se řadí mezi nejvíce zabezpečený framework vůbec. Rozeznává hned několik typů externích útoků na webové aplikace.

Cross-Site Scripting (XSS)
Mezi nejčastější typ útoku na webových stránkách bývá napadení přes jejich neošetřené výstupní řetězce. Útočník skrze tento typ útoku může podstrčit svůj vlastní škodlivý kód, pomocí kterého může získávat citlivá data ze serveru nebo data o návštěvnících webu apod. K tomuto útoku může dojít, i když není chráněná byť jen jediný výstupní řetězec, proto je důležité, aby vše bylo ošetřeno. Toho lze velmi snadno dosáhnout pomocí zabudované technologie Context-Aware-Escaping přímo v Nette, která automaticky ošetřuje všechny výstupy z aplikace. A dokonce automaticky rozpozná, kterou eskejpovací funkci je potřeba právě na daném místě v kódu použít.

Cross-Site Request Forgery (CSRF)
Tento typ útoku se nejčastěji vyskytuje ve spojitosti s formulářovými prvky na stránkách. Kdy útok přiměje například přihlášeného návštěvníka navštívit stránku nebo odeslat formulář se škodlivým kódem, který skrytě provede útok. Tím útokem může například být smazání nebo úprava dat v databázi apod.

Chránit se v Nette proti tomuto typu útoku je velice snadné, stačí přidat přidat di komponenty, který vytváří formulář příkaz `$form->addProtection()`.

URL attack, control codes, invalid UTF-8

Většina útoků začíná podstrčením škodlivých kódů na vstupech do aplikace a s tím následky spojené mohou být různé (poškozený výstupu, změna informací v databázi apod.). Spolehlivou obranou je řádné ošetření všech vstupů do aplikace na úrovni bajtů. A i o tento způsob ochrany se Nette stará zcela automaticky.

Session hijacking, session stealing, session fixation

S používáním sessions ve webových aplikacích se pojí hned několik závažných typů útoků, jako například krádež anebo podstrčení jiné ID session uživateli, který se chystá přihlásit. A díky tomu lze získat přístup do aplikace, aniž by útočník musel znát heslo. Obrana proti tomuhle typu útoku tkví ve správné konfiguraci PHP a serveru. Nette zde opět boduje a konfiguraci provádí automaticky. (nette.org, 2015)

Pro ochranu je nutné mít povolenou funkci `ini_set(String $varname, string $newvalue)` u PHP, která dovoluje měnit pomocí parametrů hodnoty v konfiguraci. (php.net, 2015)

5 Systémová implementace

V následující kapitole je popsána implementace celého systému. Pro přehlednost se začátek kapitoly člení kvůli její přehlednosti na veřejnou a administrační část systému. A dále jsou potom popsány jednotlivé systémové moduly. Vhodné by bylo ze začátku říci, že implementace systému proběhla ve vývojovém prostředí PhpStorm 9.0, které je vyvíjeno za účelem vytváření webových aplikací. Má v sobě integrovanou podporu nejrůznějších verzovacích systémů. Já jsem ve svém projektu použil například velmi populární nástroj GIT, za podpory webového repositáře `bibucket.org`. Co například tohle IDE neobsahovalo, tak byla podpora šablonovacích souborů typu .latte, která se musela dodatečně ošetřit pomocí stažení příslušného pluginu.

5.1 Veřejná část systému

Jak již bylo zmíněno v kapitole 4.3.4 tak i zde je využit, jako základní společný presenter, na kterém vše stojí *BasePresenter*. Zde se odehrávají veškeré důležité úkony a jsou zde obsaženy metody, které jsou společné pro celou veřejnou část systému. Vytváří se zde komponenta formuláře pro přihlášení do administrace (podrobnosti v kapitole 4.3.7), komponenty pro jednotlivé horní i levé boční menu. Je zde obsažena spousta obslužných metod pro chod systému, které jsou využívány napříč celou touto veřejnou částí. Dále se zde injektuje systémové nastavení z databáze (např.: newsletter školní email, počet položek pro vykreslování článků, důležité textové popisky na stránce atd.).

5.1.1 Hlavní menu

Ukázka kódu pro vytvoření komponenty hlavního menu, které leží v horní liště, nepředstavuje nic složitého. O vše se stará plugin *SmfMenu*, který je jen potřeba přidat do presenteru pomocí továrničky *injectMenuFactory*. Jednotlivé položky jsou vkládány pomocí metody *addChild*, kde prvním parametrem je název položky menu a poté pole, které sdružuje informace typu, jaký bude popisek položky, na jaký presenter nebo jeho šablonu se bude odkazovat apod. Ukázka kódu, metody pro vytvoření komponenty levého bočního menu s vlastní správou pod-stránek, je vložena do přílohy B.

```
public function injectMenuFactory(Menu\Control\Factory $factory) {
   $this->menuFactory = $factory;
}

protected function createComponentTopMenu() {
   $menu = $this->menuFactory->createControl();
   $root = $menu->getRoot();

   $root->setChildrenAttribute('class', 'navbar-nav');

   $root->addChild('ke stazeni', array('label' => 'KE STAŽENÍ', 'link'
```

```
=> 'KeStazeni:default'));
   $root->addChild('zamestnanci', array('label' => 'ZAMĚSTNANCI', 'link'
=> 'Zamestnanci:default'));
   $root->addChild('kontakty', array('label' => 'KONTAKTY', 'link'  =>
'Kontakty:default' ));
   $root->addChild('archiv', array('label' => 'ARCHIV', 'link'  => 'Ar-
chiv:default'));

   return $menu;
}
```

5.2 Administrace systému

Pro administrační část systému je také společný presenter s názvem *BasePresenter*. To, že mají presentery stejné názvy v mém případě vůbec nevadí, protože jsou zcela modulárně odděleny a tím pádem na sebe vůbec nevidí. Tento presenter zastává principiálně stejnou funkcionalitu, jako v části veřejné. Jen s odlišnými metodami a jinými komponentami pro vytvoření menu apod.

5.2.1 Vstup do administrace

Při vstupu do administrace systému se musí uživatel prokázat uživatelským jménem a heslem (kód pro formulář je k nahlédnutí v kapitole 4.3.7). Poté se porovnává hash vloženého hesla s heslem uloženým u příslušného uživatele v databázi. Po shodě obou hashů a zdali splňuje uživatel oprávnění pro vstup do administrace (tzn. je v roli administrátora), je poté uživatel vpuštěn do administrace.

Na následující ukázce kódu můžeme vidět, jak je zpracována kontrola uživatele při vstupu do administrace. Metoda ověřuje, jestlipak je uživatel přihlášený. Když ano, tak jestli je i v roli administrátora. Touto metodou se poté ošetřuje jakékoliv místo v administraci, do kterého je vyžadován autentizovaný přístup.

```
protected function loginVerification($user, $type, $icon, $historyType,
$content, $contentHistory = null) {
   if (!$user->isLoggedIn()) {
      $this->leaveMessage($type, $icon, $historyType, $content . $con-
tentHistory);
      $this->redirect(':Public:Homepage:');
   } elseif (!$user->isInRole(1)) {
      $this->leaveMessage($type, $icon, $historyType, $content . $con-
tentHistory);
      $this->redirect(':Public:Homepage:');
   }
}
```

5.2.2 Obsah v administraci

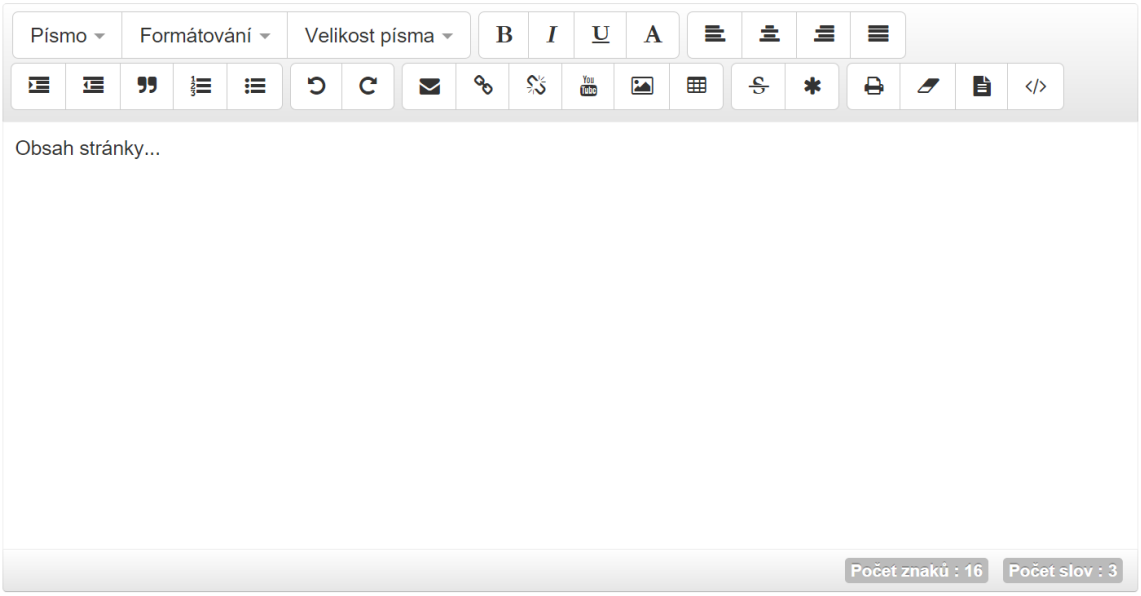
Veškerý obsah získaný z databáze, který je zobrazovaný v administraci se vkládá do tabulek. Ty jsou velmi užitečně sestaveny a poskytují velmi kvalitní nástroj pro manipulaci se zobrazenými daty (detail na obr. 19). Tabulky umožňují vzestupné i sestupné řazení všech hodnot ve sloupcích, je možné zobrazovat a skrývat jednot-

livé sloupce dle potřeb uživatele a také je zde velmi užitečná možnost full-textové vyhledávání napříč celou tabulkou.

5.3 Modul stránek

Klíčovou funkcionalitu splňuje právě tento modul, pomocí kterého může administrátor vytvářet jednotlivé pod-stránky systému. Ty se dají rozdělit dle požadavků do několika podsekcí: *základní škola, školní družina, mateřská školka, pro rodiče a žáky, školní akce a mezinárodní spolupráce*. Tyto sekce jsou prezentovány, jako konstanty a nelze s nimi manipulovat. To proto, že budou stálé a neměnné (ani v budoucnu).

Vytvoření stránky probíhá, na základě běžného formuláře, který ale poskytuje velmi užitečný nástroj pro formátování obsahu stránky. Inline zabudovaný WISIWIG editor do vstupního prvku formuláře textarea. Ten je zprostředkován pomocí jQuery knihovny *LineControl*, doplněné mnou o další funkce (vkládání embeded youtube videí, vkládání emailové adresy) a vykreslovací úpravy, pro příjemnější ovládání a rozpoznání, co který prvek dělá.



Obr. 27 Ukázka chování WISIWIG editoru

U stránek je dále možné vložit titulní obrázek, zobrazovaný v hlavičce stránky a je umožněno vkládat přílohy v podobě souborů a obrázků. Systém disponuje filtrovacími funkcemi, které detekují, zdali se jedná o soubory nebo obrázky a vykreslí je na stránce do odpovídajících oddílů.

Galerii (oddíl s obrázky) se může dle potřeby měnit pozice na stránce. Může být například zobrazována nad/pod obsahem stránky a nebo nad/pod oddílu se souborovými přílohami.

U stránek není umožněno odesílání newsletteru, protože se jedná o pevně ukotvené informační části, které zůstanou na svém místě patřičně dlouhou dobu.
Protože stránky zabírají komplexnější řešení na více místech (presenterech) v souborech. Je jejich komponenta pro vytvoření formuláře vytvářena na základě vlastní továrničky. Následující kód je její prezentací.

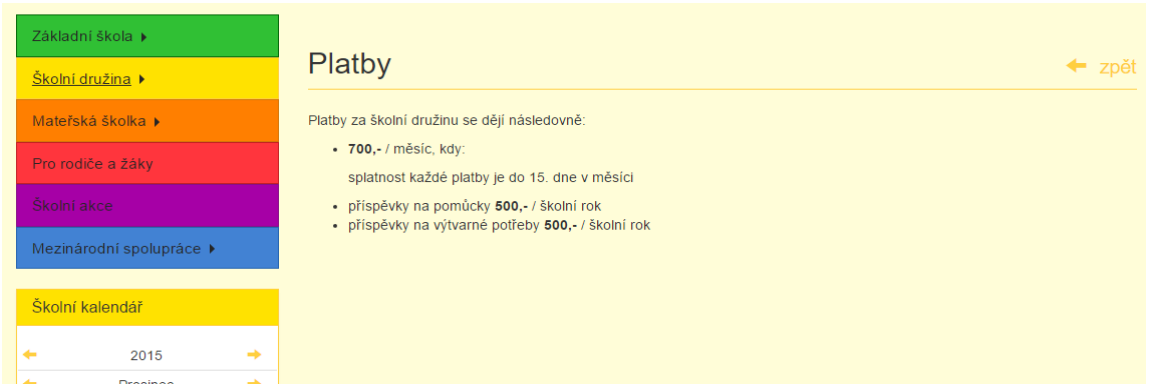
```
use Nette\Application\UI\Form;

class AddPageFormFactory {

   public function create() {
      $form = new Form();

      $form->addText('title')
         ->setRequired('Napište prosím název stránky')
         ->setAttribute('placeholder', 'Nový název stránky');

      $form->addTextArea('content')
         ->setRequired('Napište prosím alespoň pár slov do obsah')
         ->setAttribute('placeholder', 'Obsah...');

      $form->addUpload('image')
         ->addCondition(Form::FILLED)
         ->addRule(Form::IMAGE, 'Úvodní obrázek musí být formátu JPEG,
PNG nebo GIF.')
         ->addRule(Form::MAX_FILE_SIZE, 'Maximální velikost úvodního ob-
rázku je 7 MB.', 7000 * 1024);
      $form->addCheckbox('deleteImage')
         ->setDefaultValue(false);

      $form->addMultiUpload('files', TRUE)
         ->addRule(Form::MAX_LENGTH, 'Zmenšete počet souborů, maximum je
25', 25);
      $form->addCheckbox('deleteFiles')
         ->setDefaultValue(false);

      $form->addCheckbox('enable')
         ->setDefaultValue(false);

      $form->addSubmit('cancel');
      $form->addSubmit('send');

      $form->addProtection('Vypršel ochranný časový limit, odešlete pro-
sím formulář ještě jednou');

      return $form;

   }
}
```

Zpracování formuláře se děje v metodě *addPageFormSucceeded*. A pro tak, ani ne velký formulář, ale komplexní s velkou řadou ošetřujících kroků uvedu jen v textové podobě.

Po odeslání formuláře se v první řadě zkontroluje, zdali je uživatel oprávněn jej editovat (toto opatření obsahují veškeré metody v administrační části). Dále se převede název stránky do url odpovídajícího tvaru pomocí funkce *webalize*. Dále se

zjišťuje, jestli daná stránka existuje, pokud ano přistoupí se na její editaci, pokud ne pokračuje se ve vytvářecím procesu. Dále jsou vytvořeny adresáře s příslušnými jmény (název článku v url podobně) a v příslušných sekcích. Poté přichází filtrace příloh a jejich uložení na server, kdy se obrázky upraví do rozlišení 840x840 se zachováním poměru stran originálního obrázku. Na závěr se vše uloží do databáze a vypíše se příslušné systémové hlášení.

Při editační části se veškeré změny kontrolují oproti původním hodnotám, souborům a adresářům. Porovnávají se názvy stávajícího adresáře a nového adresáře, pokud přichází změna v názvu článku apod.

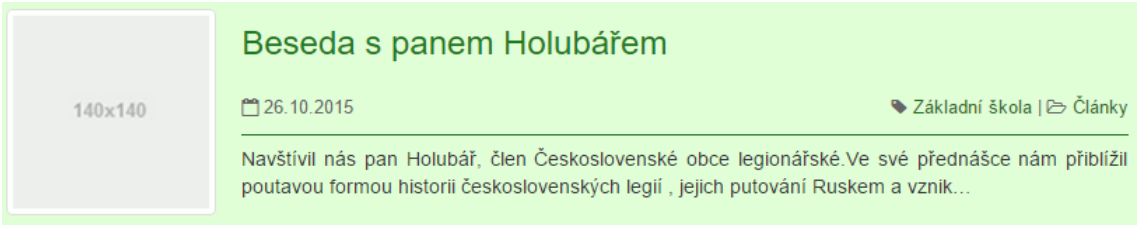


Obr. 28 Ukázka zobrazení pod-stránky v sekci Školní družina

5.4 Modul článků

Principiálně pracuje totožně, jako výše zmíněný modul pro stránky. S tím rozdílem, že je umožněno vytvářet články s jiným typem (základní škola, školní družina, mateřská školka, školní akce) a také je rozlišováno, jestli se jedná o článek nebo článek s videem. Další možností odesílat newsletter, protože se jedná o více proměnlivé informace a pro návštěvníky systému zajímavější.

Dále tento modul disponuje funkcí řadit zastaralé články do sekce archiv (a zpět), která je součástí hlavního horního menu. Pro příjemnější manipulaci s články je dle výše zmíněných typů umožněná jejich filtrace.



Obr. 29 Ukázka zobrazení článku

5.5 Systémová hlášení

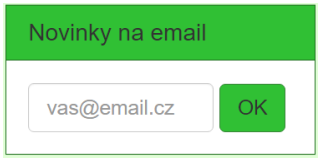
Celý systém je vybaven velmi užitečným mechanismem chybových hlášení, který je zprostředkován pomocí vestavěné Nette metody *flashMessage.* Na výstup tato metoda vypisuje systémová hlášení. Po modifikaci vypisuje zprávy zabarvené dle míry bezpečnost (zelené, žluté, červené). Dále tento systém vypisuje události nebo úkony, které se právě provedly. Ty jsou dále uloženy v databázi a vypisovány v administraci, aby měl administrátor přehled, co se na serveru děje. Obslužný kód metody *leaveMessage* můžeme vidět níže. Metoda přebere odpovídající parametry, uloží zprávu do databáze a poté vypíše na výstup odpovídající zprávu.

```
public function leaveMessage($type, $icon, $historyType, $content, $con-
tentHistory = null) {
   $this->eventsRepository->addEvent($this->createHistory($historyType,
$type, $content . $contentHistory));
   $this->flashMessage($content, $type, $icon);
}
```

Obr. 30 Ukázka výpisu systémového hlášení

5.6 Newsletter modul

Při příchodu návštěvníka na stránky je zde známá možnost odběru novinek, které se zde zobrazují v podobě článků a oznámení, které škola vydává. To se děje pomocí velmi jednoduchého formuláře, který žádá vložení emailové adresy.



Obr. 31 Newsletter formulář

Ta se při odeslání formuláře ověří, zda je ve správném tvaru a poté se formulář odešle. Kód, pro vytvoření formuláře k odběru novinek vypadá následovně.

```
protected function createComponentNewsletterForm() {
   $form = new Form;

   $form->addText('email')
      ->setAttribute('placeholder', 'vas@email.cz')
      ->addCondition(Form::FILLED)
         ->addRule(Form::EMAIL, 'Email má špatný formát');

   $form->addSubmit('send');

   $form->onSuccess[] = array($this, 'newsletterFormSucceeded');
```

```
    return $form;
}
```

Následuje kód pro zpracování odeslaného formuláře, kde se v první řadě zkontroluje, zda je vyplněna emailová adresa. Proč se kontroluje vyplněná emailová adresa až zde, je z toho důvodu, že JS ošetření prázdného vstupu před odesláním zbytečně vypisuje chybová hlášení už jen při kliknutí a odklinutí ukazatele myši z textového inputu formuláře, což může působit zbytečně rušivým dojmem. Dále se kontroluje, zdali pak vložená emailová adresa již neodebírá školní novinky. V případě neshod se uživateli zobrazí titulní stránka s příslušným systémových hlášením.

```
public function newsletterFormSucceeded($form) {
   $values = $form->values;

   if($values['email'] == null) {
      $this->leaveMessage('warning', 'glyphicon glyphicon-info-sign',
'newsletter', 'Nevyplnili jste žádný email, zkuste to prosím znovu, dě-
kujeme');
      $this->redirect('Homepage:');
   }

   if($this->settingsRepository->fetchSingleName('newsletter_email')){
      $sender = $this->settingsRepository-
>fetchSingleName('newsletter_email')->value;
   }else{
      $sender = 'zs@krestanskaskola.cz';
   }

   $newsletter = $this->userManager->fetchSingleEmail($values['email']);

   if ($newsletter) {
      $this->leaveMessage('warning', 'glyphicon glyphicon-info-sign',
'newsletter', 'Váš email: <strong><em>' . $newsletter->email .
'</em></strong> již odebírá novinky naší školy, děkujeme');
      $this->redirect('Homepage:');
   }else{

      if($values['email']) {

         $email = new Message;
         $email->setFrom($sender, 'ZŠ Křesťanská škola')
            ->addCc($sender)
            ->addTo($values['email'])
            ->setSubject('Přihlášení k odběru novinek Křesťanské Školy
Jana Husa');

         $template = $this->createTemplate();
         $template->setFile(__DIR__ .
'/../templates/Email/newsletterEmail.latte');
         $template->notifications = $this->notificationsRepository-
>fetchAllEnbaledWeekOld();

         $template->articles = $this->pagesRepository-
>fetchAllArticlesVideosEnabledOneWeekOld();
         $template->title = 'Přihlášení k odběru novinek Křesťanské Ško-
ly Jana Husa';
         $template->schoolEmail = $sender;
         $template->values = $values;
```

```
            $email->setHtmlBody($template);

            $mailer = new SendmailMailer;
            $mailer->send($email);

            $this->leaveMessage('success', 'glyphicon glyphicon-ok', 'ne-
wsletter', 'Váš email: <strong><em>' . $values['email'] .
'</em></strong> byl přihlášen k odběru novinek naší školy, děkujeme');
            $values['enable'] = 1;
            $this->userManager->addUserEmail($values);
            $this->redirect('Homepage:');

        } else {
            $this->leaveMessage('danger', 'glyphicon glyphicon-exclamation-
sign', 'newsletter', 'Nedostali jsme k odběru novinek žádnou emailovou
adresu');
            $this->redirect('Homepage:');
        }
    }
}
```

Po úspěšném odeslání formuláře dojde odběrateli na uvedený email poděkování s přehledem článků a školních oznámení za poslední týden. V případě nezájmu o další odběr novinek je na konci každého emailu možnost odhlásit se z odběru a také kontakt na školu, pro případ potřeby další komunikace.

Dále v administraci systému je samozřejmou možností manuální správa všech odběratelů. S funkcemi přidávání nového odběratele, editace stávajícího nebo jeho smazání. Je zde také možné mít uloženy odběratele v databázi a nemít u nich povoleno odesílání novinek. To se může hodit například ze statistických důvodů.

5.6.1 Odeslání newsletteru

V ukázce kódu hromadného odesílání newsletteru si můžeme všimnout, že se nejdříve v první řadě vybere školní email pro newsletter, když není tento email vyplněn, vloží se jako email pro odesílatele školní informační email. Dále se vyberou všichni přihlášení odběratelé, kterým se odešle email s příslušnými daty oznamující nové školní oznámení, názvem oznámení apod.

```
public function sendNewsletter($newsletters, $value , $emailTemplate) {
    $email = new Message;

    if($this->settingsRepository->fetchSingleName('newsletter_email')){
        $sender = $this->settingsRepository-
>fetchSingleName('newsletter_email')->value;
    }else{
        $sender = 'zs@krestanskaskola.cz';
    }

    foreach($newsletters as $newsletter) {

        $email->setFrom($sender, 'ZŠ Křesťanská škola')
            ->setSubject('Nové školní oznámení: '. $value->title)
            ->addBcc($newsletter->email);

        $template = $this->createTemplate();
        $template->setFile(__DIR__ . '/../templates/Email/' . $emailTem-
```

```
plate);
        $template->title = 'Nové školní oznámení - ' . $value->title;

        $template->unsubscribeEmail = $newsletter->email;
        $template->schoolEmail = $sender;
        $template->values = $value;

        $email->setHtmlBody($template);

        $mailer = new SendmailMailer;
        $mailer->send($email);
    }
}
```

5.7 Modul partneři školy

Celá funkcionalita modulu je zpracovávána v administraci systému. Kdy prvním krokem je vytváření partnera školy. Tady je povinným parametrem ve vstupním formuláři pouze vyplnění url adresy jeho webu. Mezi volitelné prvky formuláře spadá možnost vložit logo partnera. Při výběru loga pomocí vstupního prvku formuláře uživatel dostane zobrazení svých souborů pouze v podobě obrázků. Obrázek je možné vložit pouze ve formátu typu .jpg, .png, .gif, aby se předešlo různým komplikacím apod. Dále pak je možné přidat textový popisek loga, který by uživateli napovídal, o co přesněji se jedná.

Obr. 32 Ukázka zobrazení partnerů školy

Následuje ukázka kódu, který zpracovává vkládaný obrázek přes odeslaný formulář. V první řadě se zkontroluje, zdali existuje nějaká souborová příloha, když ne do databáze se vloží *null* hodnota, která zapříčiní, že se výstupu bude vykreslovat pouze url adresa partnera. Poté se vkládanému obrázku do názvu přidá prefix „logo_“, aby bylo zřejmé při FTP manipulaci se soubory na serveru, že se jedná o logo. Dále následuje formátování loga na maximální rozměry 300x300 pixelů při zachování originálního poměru stran, aby nedocházelo k nechtěné deformaci vkládaného loga.

```
if($values->image->isOk()) {

    $values->image = 'logo_'. Strings::toAscii($values->image-
>getName());

    $file = $form['image']->getValue();
    $imgUrl = $this->context->expand('%wwwDir%/' . $directory . $values-
```

```
>image);

    $file->move($imgUrl);
    $img = $file->toImage();
    $img->resize(300, 300, Image::SHRINK_ONLY);
    $img->sharpen();
    $img->save(Strings::toAscii($imgUrl));
} else {
    $values->image = null;
}
```

5.8 Modul slider

Slider byl se dal označit, jako velmi krátká posouvací galerie s obrázky. Je zprovozněn za JS podpory bootstrap frameworku. Slider se vytváří se automaticky na základě vkládaných obrázků. Vkládání se děje přes formulář, pro vytváření obrázků. Zde je povinné pole pouze pro obrázek. Formulář opět dovoluje vkládání pouze obrázkových příloh v příslušných formátech, aby se zamezilo vložení nesprávného souboru. Formulář také obsahuje nepovinná pole, kterými jsou popisky pro obrázek. Ty jsou rozděleny na titulek a detailní popisek obrázku a jsou uloženy v tzv. textovém panelu, který je možný dle potřeby celý zobrazit nebo skrýt.

Po odeslání formuláře, se obrázek naformátuje do požadovaného rozměru 1600x1060. Tentokrát nezůstává pevný poměr stran u originální obrázku, takže se obrázek dle potřeby roztáhne nebo zúží. To se děje díky využití širokoúhlému vykreslování obrázků ve slideru.

V administraci jsou další možnosti, jak pracovat se sliderem. Je možné jej celý zobrazit/skrýt na veřejné části. Dále je zde možnost zobrazit/skrýt jednotlivé obrázky samostatně, nezávisle na sobě.

Obr. 33 Ukázka slideru

5.9 Modul školních oznámení

Patří mezi hlavní nástroje systému, díky kterému zůstává veřejnost informována okolo dění na školní půdě. Při vytváření školního oznámení je možné vybrat, do jaké skupiny bude toto oznámení zařazeno.

5.9.1 Skupiny školních oznámení

Tyto skupiny jsou uloženy v databázi a není možné s nimi jakkoliv operovat. Řadí se tímto mezi konstanty. Je to z toho důvodu, že se v systému nebudou dále tyto skupiny měnit.

- Základní škola
- Školní družina
- Mateřská školka
- Školní akce

Přiřazení typů šk. oznámení do formuláře znázorňuje následující kód.

```
$not_types = $this->notificationsTypeRepository->fetchAll();
foreach ($not_types as $type) {
   $not_types_array[$type->id] = $type->type;
}
$form->addSelect('const_notifications_type_id')
   ->setItems($not_types_array)
   ->setPrompt('-- Zvolte typ oznámení --')
   ->setRequired('Prosím zvolte typ oznámení');
```

Po přiřazení typu se školní oznámení při vykreslování obarví příslušnou barvou. Je to z toho důvodu, aby už při pohledu bylo zřejmé, do jaké skupiny oznámení spadá.

Během vytváření školního oznámení je možné, ale ne povinné, ihned po jeho vytvoření odeslat newsletter, který bude vznik nového oznámení oznamovat. Tato možnost je v administraci pro pozdější použití možná, pro veškerá školní oznámení. Je zde také možnost rozesílat newsletter opakovaně.

Dále je možné školní oznámení zobrazovat/skrývat a to, jak v panelu školních oznámení, tak v kalendáři zcela nezávisle. Třeba pro organizační možnosti.

5.10 Modul zaměstnanců školy

Modul nabízí veškerou správu všech zaměstnanců ve školním zázemí. Při vytvoření zaměstnance je důležité a povinné vybrat typ zaměstnance, protože bez toho není možné jej vytvořit. Typy zaměstnanců systém umožňuje spravovat dle potřeb administrátora. Velkou výhodou je vytváření oslovení na základě pohlaví zaměstnance, které přidává k typu zaměstnance, které musí být v mužském rodě, postfix „ka“ např.: ředitel – ředitelka. A také automatické hierarchické řazení při vykreslování zaměstnanců, kdy vedoucí školy se vykreslují vždy nad učiteli. A ti se vykreslují nad ostatním personálem školy.

Obr. 34 Ukázka výpisu zaměstnanců školy

5.11 Výběr webhostingu

Toto rozhodnutí nenechávalo příliš prostoru pro rozhodování, protože už několik let využívám pro své menší webové projekty služby od společnost Wedos. S jejich nabídkou služeb jsem byl vždy plně spokojen. A s častými akcemi, které poskytují, až 50% slevy nešlo ani v tomto případě říci ne. V tabulce uvádím krátký přehled poskytovaných služeb, který si vyžadovala má práce (pro více informací doporučuji navštívit jejich web `hosting.wedos.com/cs/`).

Tab. 1 Specifikace webhostingu – NoLimit

Parametry	Hodnota
Diskový prostor pro web	neomezeně
Množství přenesených dat	neomezeně
PHP 5.3 – 5.6(vysoké parametry s možností změn)	ano
Počet databází MySQL	neomezeně
FTP účet, FTPS, zamykání FTP, webFTP	ano
Omezování přístupu podle IP adresa	ano
On-line správa přes zákaznické centrum	ano
Provoz webu s www i bez www	ano
Podpora IPv6	ano
Podpora HTTPS se sdíleným i vlastním certifikátem	ano
DDoS ochrana	ano
Vedení DNS s možností plné editace zákazníkem	ano

Zdroj: hosting.wedos.com/cs/webhosting.html, 2015.

6 Závěr

Závěrem bych chtěl velmi krátce shrnout, jak se zadařilo plnění cíle této diplomové práce a přínosy, které mi tvorba systému přinesla.

Osobně si myslím, že se mi velmi dobře zadařilo splnit vytyčené cíle a hlavně se podařilo splnit veškerá přání a požadavky zákazníka. Slova zákazníka a nadšení, které vyvolávala naše spolupráce myslím, dostatečně potvrzují výsledek mé práce.

Samotná práce mě velmi obohatila zkušenostmi získanými s tvorbou webové aplikace postavené na Nette frameworku. Nikdy mě nenapadlo, že se do této oblasti dostanu, protože jsem s používáním Nette začal již před tvorbou této práce. Začátky byly velmi časově náročné a díky za ně, protože jen tak jsem mohl sestavit něco dle slov zákazníka „*úžasné Davide*“. Dále jsem získal mnoho zkušeností při kódování responzivní designu, které mi již teď obstarávají částečnou pracovní náplň. A také mnoho zkušeností s nasazováním js knihoven na zpříjemnění uživatelského ovládání nejrůznějších prvků v systému.

6.1 Další možná vylepšení a rozšíření

Dnem 1. 12. 2015, kdy byl systém plně sestaven dle požadavků zákazníka, uvedených v kapitole 2.2 a předveden. Byla po prezentaci získána spousta dalších budoucích rozšíření, se kterými zákazník přišel.

- Moduly slider, školní oznámení, vize školy apod. zobrazovali pouze na hlavní úvodní stránce systému.
- Změna barev loga, dle stávajících konvencí grafického loga organizace Slovo života.
- V sekci archiv seskupovat články do skupin dle jednotlivých let, ve kterých byly články napsány.
- Detailnější rozřazení školních zaměstnanců, při jejím vykreslování na veřejné části.
- Oddělit pod-stránku školní potřeby od sekce základní škola a přidělit ji hlavnímu menu.
- Přidat do levého bloku panel s biblickými citacemi a umožnit frekvenci jejich změny.
- Z šablony pro email oddělat kompletně formátování, aby uživatelé, kteří nevyužívají podporovaného emailového klienta, nebyli zmateni.

Systém je v dnešních dnech v provozu na dočasné doméně webhostingu, dokud nebude plně zhotoven dle aktualizovaných požadavků a naplněn odpovídajícím obsahem ze strany zákazníka.

7 Literatura

ACTIVITY DIAGRAM: DIAGRAM AKTIVIT. MPAVUS [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://MPAVUS.WZ.CZ/UML/UML-B-ACTIVITY-3-2-3.PHP

CODEIGNITER WEB FRAMEWORK: CODEIGNITER ROCKS. CODEIGNITER [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTPS://WWW.CODEIGNITER.COM/4.2.4

ENTERPRISE ARCHITECT - UML: ULTIMATE MODELING POWER. SPARXSYSTEMS [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://WWW.SPARXSYSTEMS.COM.AU/PRODUCTS/EA/INDEX.HTML

HOPKINS, CALLUM. PHP OKAMŽITĚ. 1. VYD. BRNO: COMPUTER PRESS, 2014, 134 S. ISBN 978-80-251-4196-0

HUDSON, PAUL. PRACTICAL PHP: HACKING WITH PHP [ONLINE]. 2015 [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://WWW.HACKINGWITHPHP.COM/

LARAVEL: THE PHP FRAMEWORK FOR WEB ARTISANS. LARAVEL [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://LARAVEL.COM/

NETTE FRAMEWORK: RYCHLÝ A POHODLNÝ VÝVOJ WEBOVÝCH APLIKACÍ V PHP [ONLINE]. 2008 [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTPS://NETTE.ORG/

PHP AND STUFF: TOP 10 REASONS WHY YOU SHOULD USE A PHP FRAMEWORK. PHPANDSTUFF [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://WWW.PHPANDSTUFF.COM/ARTICLES/TOP-10-REASONS-WHY-YOU-SHOULD-USE-A-PHP-FRAMEWORK

PHP: INI_SET - MANUAL: INI_SET. PHP [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://PHP.NET/MANUAL/EN/FUNCTION.INI-SET.PHP

PŘÍKLADY POUŽITÍ DIAGRAMŮ UML 2.0: DIAGRAM PŘÍPADŮ UŽITÍ. UML [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://UML.CZWEB.ORG/PRIPAD_UZITI.HTM

ŘEZÁČ, JAN. WEB OSTRÝ JAKO BŘITVA: NÁVRH FUNGUJÍCÍHO WEBU PRO WEBDESIGNERY A ZADAVATELE PROJEKTŮ. VYD. 1. JIHLAVA: BAROQUE PARTNERS, 2014, 211 S. ISBN 978-80-87923-01-6

SCHAFER, STEVEN M. HTML, XHTML A CSS: BIBLE [PRO TVORBU WWW STRÁNEK] : 4. VYDÁNÍ. 1. VYD. PRAHA: GRADA, 2009, 647 S. PRŮVODCE (GRADA). ISBN 978-80-247-2850-6

SOMMERVILLE, IAN. SOFTWAROVÉ INŽENÝRSTVÍ. 1. VYD. BRNO: COMPUTER PRESS, 2013, 680 S. ISBN 978-80-251-3826-7.

SYMFONY: HIGH PERFORMANCE PHP FRAMEWORK FOR WEB DEVELOPMENT. SYMFONY [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://SYMFONY.COM/4.2.2

THE BEST PHP FRAMEWORK FOR 2015: SITEPOINT SURVEY RESULTS. SITEPOINT [ONLINE]. [CIT. 2015-12-27]. DOSTUPNÉ Z: HTTP://WWW.SITEPOINT.COM/BEST-PHP-FRAMEWORK-2015-SITEPOINT-SURVEY-RESULTS/

Přílohy

A Kód vykreslující formulář pro přihlášení do administrace

```
{form signForm class => "form-horizontal", autocomplete => "off"}
   <div class="modal-body">
      <fieldset>
         <div class="col-xs-12">
            <div class="form-group">
               <label for="name" class="control-label">Uživatelské Jmé-
no</label>
               <div class="row">
                  <div class="col-xs-12 left-inner-addon">
                     <span class="fa fa-user"></span>
                     <input n:name="username" id="name" type="text"
class="form-control">
                  </div>
               </div>
            </div>
         </div>
         <div class="col-xs-12">
            <div class="form-group">
               <label for="name" class="control-label">Heslo</label>
               <div class="row">
                  <div class="col-xs-12 left-inner-addon">
                     <span class="fa fa-key"></span>
                     <input n:name="password" id="password" type="password"
class="form-control">
                  </div>
               </div>
            </div>
         </div>
         <div class="col-xs-12">
            <div class="form-group">
               <label for="name" class="control-label">Potvrzení Hesla</label>
               <div class="row">
                  <div class="col-xs-12 left-inner-addon">
                     <span class="fa fa-key"></span>
                     <input n:name="passwordVerify" id="passwordVerify" ty-
pe="password" class="form-control">
                      </div>
               </div>
            </div>
         </div>
       </fieldset>
   </div>

   <div class="modal-footer">
      <div class="form-group">
         <div class="col-xs-6 col-xs-offset-1 sign">
            <button n:name="cancel" type="reset" class="btn btn-circle-sm btn-
danger"> <span class="glyphicon glyphicon-remove"></span></button>
            <button n:name="send" type="submit" class="btn btn-circle-sm btn-
success"> <span class="glyphicon glyphicon-ok"></span></button>
         </div>
      </div>
   </div>
{/form}
```

B Ukázka metoda pro vytvoření komponenty levého menu

```
public function createComponentLeftMenu() {
   $menu = $this->menuFactory->createControl();
   $root = $menu->getRoot();

   $root->setChildrenAttribute('class', 'nav-pills nav-stacked');

   $root->addChild('skola', array('label' => 'Základní škola','link'  =>
'ZakladniSkola:default'))
      ->setLinkAttribute('class', 'green')
      ->setChildrenAttribute('class', 'light-green bullet pull-middle
pull-right');

      $pages = $this->pagesRepository->fetchAllTypeEnabled(1);
      foreach ($pages as $page) {
         if($page->url_title <> 'kontakty') {
            $root['skola']->addChild($page->url_title, array('label' =>
$page->title, 'link' => array('ZakladniSkola:', $page->url_title) ));
         }
      }
      $root['skola']->addChild('potreby-skoly', array('label' => 'Potřeby
školy', 'link' => array('ZakladniSkola:', 'potreby-skoly') ));


/*
 * …
 */

   return $menu;
}
```