

Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií



Bakalářská práce

Návrh a implementace webové aplikace pro základní školu

Jan Velebný

© 2020 ČZU v Praze

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Velebný

Systémové inženýrství a informatika
Informatika

Název práce

Návrh a implementace webové aplikace pro základní školu

Název anglicky

Design and Implementation of Web Application for Primary School

Cíle práce

Cílem praktické části bakalářské práce je návrh a implementace webové aplikace pro podporu procesů ve zvolené základní škole.

Metodika

- Na základě studia odborných zdrojů popište formou literární rešerše doménu vývoje webových aplikací dle osvědčených metod softwarového návrhu
- Proveďte analýzu uživatelských požadavků na webovou aplikaci a popište je vhodnými nástroji (Use Case)
- V souladu s doporučenými postupy softwarového inženýrství vytvořte návrh vlastní webové aplikace (konceptuální model)
- Návrh aplikace implementujte ve vhodném prostředí a řádně otestujte
- Zhodnoťte dopady nasazení navržené aplikace oproti stávajícímu stavu

Doporučený rozsah práce

30-40 stran

Klíčová slova

Webová aplikace, vývoj software, uživatelské požadavky

Doporučené zdroje informací

SOMMERVILLE, I. Softwarové Inženýrství. Praha: Computer Press, 2013. ISBN 978-80-251-3826-7

ULLMAN, L. *PHP a MySQL: názorný průvodce tvorbou dynamických www stránek*. Brno: CP, 2004. ISBN 80-251-0063-4.

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. David Buchtela, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 14. 3. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 14. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 10. 08. 2020

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Návrh a implementace webové aplikace pro základní školu" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.11.

Poděkování

Chtěl bych touto cestou poděkovat vedoucímu této práce, panu Ing. Davidovi Buchtelovi, Ph.D. za ochotu a trpělivost při vedení práce. Dále bych rád poděkoval Bc. Petru Mouchovi za duševní a technickou podporu a také Bc. Alžbětě Pokorné a Bc. Matěji Němcovi za cenné rady. V neposlední řadě patří můj dík Mgr. Jiřímu Blahoutovi a Mgr. Editě Doubravské spolu s paní Mikešovou a Bohumilem Pocem, jejichž pomoc byla zásadní.

Návrh a implementace webové aplikace pro základní školu

Abstrakt

Bakalářská práce je zaměřena na metodiky vývoje webových aplikací a metody softwarového návrhu a následné využití těchto znalostí pro návrh a implementaci webové aplikace pro podporu procesů na základní škole. Aplikace umožňuje správu známek, rozvrhů a evidence učitelů a žáků. Zvolenou technologií je programovací jazyk PHP a framework Symfony. Teoretická část práce se zabývá teorií softwarového návrhu a popisu užitých technologií. Praktická část pojednává o návrhu webové aplikace na bázi systémových požadavků a její implementaci. Součástí je detailní vysvětlení kódu aplikace spolu s ukázkami konečného řešení.

Klíčová slova: Webová aplikace, vývoj software, uživatelské požadavky

Design and implementation of web application for primary school

Abstract

This bachelor thesis is focused on methodology of web application development and methods of software design and usage of this knowledge for design and implementation of web application for support of processes on primary school. Application allows for management of grades, timetables and evidence of teachers and students. The technology of choice is the PHP programming language and the Symfony framework. The theoretical part of this thesis is focused on the theory of software design and a description of used technologies. The practical part is focused on the design of the web application on the basis of system requirements and its implementation. Detailed explanation of the code of the application as well as samples of the finished project are a part of this thesis.

Keywords: Web application, software development, use cases

Obsah

| | |
|--------------------------------------|-----------|
| 1 Úvod..... | 8 |
| Cíl práce a metodika..... | 9 |
| 1.1 Cíl práce | 9 |
| 1.2 Metodika | 10 |
| 2 Teoretická východiska | 11 |
| 2.1 Webové aplikace | 11 |
| 2.2 Vývoj aplikací | 13 |
| 2.2.1 Jazyk UML | 15 |
| 2.2.2 Diagram užití | 15 |
| 2.3 Informační systém..... | 15 |
| 2.4 HTML | 17 |
| 2.4.1 Historie HTML | 17 |
| 2.4.2 HTML 5 | 18 |
| 2.5 CSS..... | 18 |
| 2.5.1 CSS3 | 19 |
| 2.5.2 Bootstrap..... | 20 |
| 2.6 JavaScript | 20 |
| 2.7 PHP | 21 |
| 2.7.1 Historie PHP | 21 |
| 2.7.2 Symfony Framework | 22 |
| 2.7.3 Composer | 24 |
| 2.8 MySQL..... | 24 |
| 2.8.1 Historie MySQL | 25 |
| 2.8.2 SQL | 25 |
| 2.9 Apache Web Server..... | 25 |
| 2.9.1 Laragon | 26 |
| 3 Vlastní práce..... | 27 |
| 3.1 Počáteční Analýza..... | 27 |
| 3.1.1 Funkční požadavky | 28 |
| 3.1.2 Metodika vývoje | 30 |
| 3.1.3 Případy užití..... | 30 |
| 3.1.4 Scénář 1 – Přihlášení..... | 37 |
| 3.1.5 Scénář 2 – Znamky | 38 |
| 3.2 Vývoj webové aplikace | 39 |
| 3.2.1 Databáze..... | 40 |
| 3.2.2 Zvolené technologie..... | 44 |

| | | |
|--|---|-----------|
| 3.2.3 | Implementace | 44 |
| 3.2.4 | Hlavní Obrazovka | 45 |
| 3.2.5 | Zobrazení známek | 49 |
| 3.2.6 | Zabezpečení a dodatečné informace | 53 |
| Výsledky a diskuse | | 56 |
| 3.3 | Testování | 56 |
| 3.4 | Přínosy aplikace | 57 |
| 3.5 | Budoucí Vývoj | 58 |
| Závěr | | 59 |
| 4 Seznam použitých zdrojů | | 60 |
| 5 Přílohy | | 63 |
| Příloha A – CD se zdrojovým kódem aplikace..... | | 64 |

Seznam obrázků

| | |
|---|----|
| Obrázek 1 - Znázornění dynamické webové stránky s databází [1] | 12 |
| Obrázek 2 - Příklad HTML [2] | 17 |
| Obrázek 3 - Příklad CSS [3] | 19 |
| Obrázek 4 - Znázornění fungování MVC modelu [4] | 23 |
| Obrázek 5 - Use Case Diagram z pohledu Administrátora..... | 31 |
| Obrázek 6 - Wireframe přihlašovacího formuláře | 37 |
| Obrázek 7 - Wireframe domovské obrazovky | 38 |
| Obrázek 8 - Wireframe stránky se známkami | 39 |
| Obrázek 9 - Wireframe formuláře pro přidání známky | 39 |
| Obrázek 10 - ER Diagram databáze | 41 |
| Obrázek 11 - Seznam Atributů v Entitách v Databázi | 42 |
| Obrázek 12 - Seznam Atributů v Entitě Žák..... | 43 |
| Obrázek 13 - Příklad kódu v šabloně Twig pro rozlišení rolí..... | 46 |
| Obrázek 14 - Ukázka hotové domácí obrazovky | 47 |
| Obrázek 15 - Ukázka controlleru pro domovskou obrazovku | 48 |
| Obrázek 16 - Ukázka funkce index controlleru pro známky | 50 |
| Obrázek 17 - Ukázka cyklů v šabloně Twig pro výpis známek | 52 |
| Obrázek 18 - Ukázka hotové stránky pro výpis známek | 53 |
| Obrázek 19 - Ukázka ukládání do databáze z controlleru pro registraci | 54 |
| Obrázek 20 - Ukázka omezení přístupu ze security.yaml | 55 |

Seznam tabulek

| | |
|---|----|
| Tabulka 1 - Scénář Přihlášení | 37 |
| Tabulka 2 - Scénář přidání známky | 38 |

1 Úvod

V současné době probíhá spousta debat o tom, zdali vedeme naše školství optimálně a jestli bychom ho neměli reformovat či změnit. Existují i názory, že je naše školství stále zaseklé v 17.století. Já se do debatování ohledně didaktiky nebo optimální délky vyučovacích hodin a přestávek pouštět nebudu. Musím však uznat, že by školství v jistých oblastech trochu inovace opravdu neuškodilo. V Některých konkrétních školách méně a v jiných více, nicméně v této práci se chci zaměřit pouze na základní školy a na oblast organizace. I přesto, že informační technologie již dávno nejsou žádnou novinkou a stávají se nezbytnou pomůckou v mnoha odvětvích lidské činnosti. Do mnoha škol se tato inovace dostává stále značně pomalu, a to i přesto, že právě ve školním prostředí by podobný systém mohl výrazně zlepšit organizaci, produktivitu a efektivitu výuky.

Obečně podnikové informační systémy pracují s informacemi. Umějí je sbírat, zpracovávat a šířit, a to za účelem dalšího plánování, rozhodování a řízení. Což jsou činnosti, ze kterých bude základní škola doajista benefitovat, stejně jako každý jiný, dostatečně velký, podnik. Informační systém je většinou softwarové řešení, které běží na serveru a data ukládá do databáze, která běží jako oddělená součást na pozadí. V tomto případě se jedná o aplikaci webovou, k systému se tedy přistupuje pomocí internetového prohlížeče a je obsluhována pomocí uživatelského rozhraní, daném webovou stránkou.

Cíl práce a metodika

V této kapitole popíši veškeré nutné teoretické základy, které stojí za celou webovou aplikací. Popíši všechny dílčí základy a technologie, na kterých je tato webová aplikace založena a předám nutné informace, které stály za jejím vytvořením.

1.1 Cíl práce

Cílem bakalářské práce je návrh a implementace webové aplikace pro podporu procesů na základních školách. Autor bude spolupracovat s učiteli a odborníky na více základních školách a na bázi konzultace s nimi navrhne a vytvoří Webovou aplikaci. Aplikace má obsluhovat některé důležité součásti školního provozu, jako evidování informací o žácích a učitelích a regulování přístupu k těmto informacím, nebo zobrazení a tvorbu rozvrhů, známek, absencí a suplování za použití přehledného uživatelského rozhraní. Teoretická část má za cíl analyzování onoho školního prostředí a procesů, tam probíhajících, za účelem návrhu odpovídající funkcionality, za využití korektních a ověřených postupů softwarového inženýrství. Dále také popis problematiky celého vývoje webových aplikací, na základě jejich studia. Taktéž budou vysvětleny jednotlivé kroky a prostředky, které byly využity pro vývoj webové aplikace. V praktické části této práce bude mým cílem tuto aplikaci navrhnout a následně vytvořit. K návrhu budu využívat diagramů pro znázornění struktury systému a ER diagramů pro znázornění struktury databáze pro systém, kterou také vytvořím. Výsledkem bude webová aplikace, která žákům a učitelům umožní bezpečné přihlašování a přístup k důležitým informacím, známkování, rozvrhům, suplování a docházce. Dále bude umožňovat správcům systému manipulovat s funkcionalitou aplikace a s databází na hlubší úrovni. V závěru proběhne otestování aplikace a zhodnocení objektivní funkcionality a zjištěných připomínek při testování.

1.2 Metodika

Na základě studia odborné literatury a článků z oboru softwarového inženýrství, vývoje softwaru a programování, systémové analýzy provedu analýzu a navrhnu webovou aplikaci.

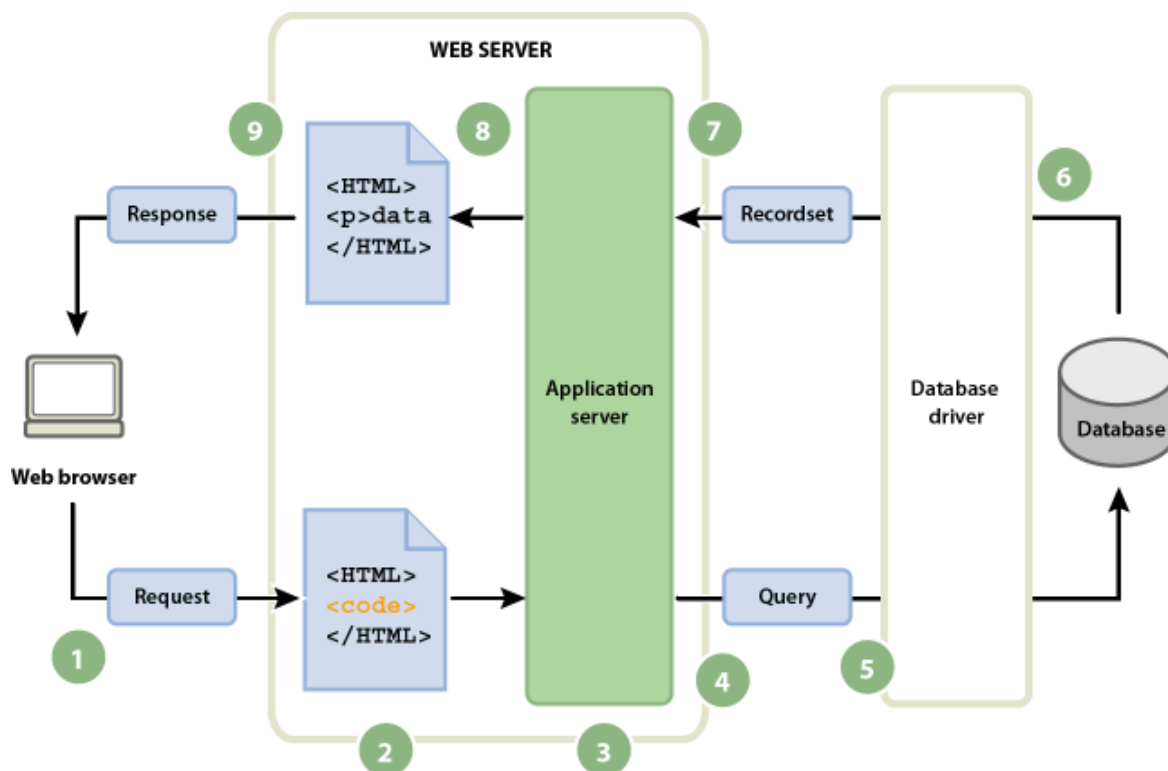
V teoretické části se budu zabývat převážně hlavními tématy této práce, tedy webovými aplikacemi a informačními systémy a vývojem. Vysvětlím všechny důležité náležitosti a základní pojmy. V neposlední řadě se budu zabývat všemi důležitými použitými technologiemi, které byly použity pro vývoj jako jsou HTML a CSS, PHP a MySQL. Je nezbytné zmínit Framework Symfony s architekturou MVP, která bude taktéž vysvětlena spolu s komponentami a nástroji, které s frameworkem úzce spolupracují jako Twig, ORM Doctrine a Composer.

Primární náplní praktické části poté jest samotná tvorba aplikace v jazyce PHP a také tvorba podrobného use case diagramu, jenž bude popisovat jednotlivé scénáře případů užití aplikace. Také bude vyhotoven ER diagram pro zobrazení struktury databáze webové aplikace. Samotná implementace pak proběhne za využití popsaných technologií a k ilustraci technického řešení budou přidány ukázkové části kódu.

2 Teoretická východiska

2.1 Webové aplikace

Hlavní rozdíl mezi běžnou webovou stránkou a webovou aplikací spočívá v tom, že webová stránka pouze staticky zobrazuje obsah, uložený v HTML stránkách na webovém serveru (což je software schopný odeslat správnou webovou stránku na základě daného požadavku). Kdežto webová aplikace disponuje možností změnit svůj obsah nebo výstup díky aplikační logice na webovém serveru na bázi zadaných vstupů, je tedy dynamická. Obsah, který se zobrazí, se určí až poté, co uživatel požádá o danou stránku z webového serveru, kdežto finální obsah statické webové stránky se nijak nemění, když je stránka vyžádána. Samozřejmě, webová aplikace nemusí nutně být pouze dynamická, obecně se skládá z obou, statických i dynamických webových stránek. Tím vším to ale nekončí, do procesu zobrazení dat se totiž dá zapojit i databáze, z které může webový server číst data. Je k tomu zapotřebí databázová vrstva na straně aplikačního serveru, neboť data z databáze jsou pro aplikační server nečitelná a musí být přeložena. Tímto způsobem je možno vyjmout data z databáze a vložit je do kódu HTML na stránce. Používají se k tomu databázové dotazy v jazyce SQL, které se umisťují do tagů nebo skriptů na straně serveru, databáze potom aplikačnímu serveru vrátí sadu záznamů, tedy přeložená data z jedné nebo více tabulek v databázi. Využití databáze ve webové aplikaci s sebou nese tu výhodu, že je pak možné mít data oddělená od návrhu a logiky webu. Je pak možné namísto mnoha HTML stránek s různými daty, určenými pro různé typy požadavků mít pouze jednu HTML stránku či pouze šablonu a její obsah měnit podle zobrazovaných informací. Následující obrázek popisuje celý proces zpracování požadavku na zobrazení webové stránky. Jeho odeslání do aplikačního serveru, který si podle vnitřní logiky vyžádá data ve formě dotazu, přes databázovou vrstvu z databáze. Tato data se mu vrátí jako sada záznamů, kterou aplikační server pak vloží do HTML stránky nebo šablony a vrátí jako odpověď uživateli.[1]



Obrázek 1 - Znáznornění dynamické webové stránky s databází [1]

Na rozdíl od tradičních aplikací, které běží přímo v prostředí operačního systému, přístup k webové aplikaci uživateli zprostředkovává webový prohlížeč. To je právě jedna z hlavních výhod webových aplikací. Vzhledem k tomu, že je spouštěna přes prohlížeč, není nutné vyvíjet aplikaci pro více druhů systémů. Také to znamená, že grafické rozhraní záleží pouze na prohlížeči, a ne na operačním systému, takže je jednodušší vyvinout konzistentní prostředí pro všechny platformy. Díky tomu, že aplikace může běžet v prohlížeči jako je například Mozilla Firefox, je zajištěné, že bude fungovat na operačním systému Windows, MacOS nebo i většině distribucí Linuxu. Také není nutné vydávat update pro jednotlivé platformy, stačí jednoduše updatovat aplikaci na serveru. Také data, která jsou zadávána do aplikace, se ukládají na server a je tudíž možné, mít přístup ke stejným datům z více zařízení.[5]

2.2 Vývoj aplikací

Celý proces vývoje softwaru se člení do takzvaných cyklů. Jsou to jednotlivé etapy procesu vývoje, které na sebe navazují a mají definovanou strukturu. Nelze započít novou etapu dřív, než skončí ta předcházející a bez ukončení jedné, nelze pokračovat v další etapě. Své životní cykly mají také jednotlivé komponenty, objekty a procesy a jejich vymezení se určuje pomocí modelování.[6]

Asi nejznámější sled etap v životním cyklu vývoje sestává z:

- **Analýzy a požadavků** – jde hlavně o zhodnocení, zda je možné software vyvinout v daném čase a za daných podmínek. Dále jde o získání požadavků na systém, což umožňuje vymyslet strukturu a rozhodnout o rozsahu funkcionalit, přičemž se nejedná pouze o funkční požadavky, ale také se může jednat o systémové, provozní, legislativní nebo požadavky na rozhraní či vývojového procesu.
- **Analýzy systému** – jde o podrobné zhodnocení požadavků z předchozího kroku a tvorby konceptuálního modelu systému. Tato fáze je vcelku důležitá, neboť, pokud se zde nepodaří odhalit nějaké nedostatky nebo pokud se v této fázi udělá nějaká chyba, již velmi obtížně se pak bude odstraňovat.
- **Projektové studie** – je výsledkem analýzy systému. Jedná se o podrobný návrh konkrétní implementace systému, tvoří se zde implementační model a také logický a fyzický datový model. V této fázi se také připravují veškerá smluvní ujednání o vývoji softwaru a všech jeho náležitostech.
- **Implementace a zavedení** – jedná se o vlastní tvorbu a programování softwaru. Jako podklad slouží implementační model a veškeré informace, které byly shromážděny během předchozích etap. Na základě těchto informací se definují vstupy a výstupy jednotlivých operací, naprogramují se všechny funkce a jejich propojení a připraví se testovací data.
- **Testování** – fáze, ve které se provádí veškeré testování s připravenými daty. Provádí se v testovacím prostředí, a ne v ostrém provozu systému. Pokud jsou objeveny jakékoliv nedostatky, je nutné je okamžitě odstranit.
- **Provoz systému a udržování** – Jedná se o závěrečnou fázi projektu, kdy se jedná o ostrý, rutinní provoz a používání. Také musí být zajištěna podpora aplikace tak, aby byl zajištěn optimální běh aplikace dle nových i stávajících požadavků.

- **Stažení systému z užívání** – Konečná etapa životního cyklu, ke které dojde, když aplikaci dojde podpora nebo se dojde k rozhodnutí, že už se nebude dále vyvíjet.

[7]

K tvorbě softwaru se dá ale přistupovat také více způsoby. Existuje mnoho modelů pro vývoj softwaru spadajících do dvou kategorií – iterativní, kdy se software vyvíjí v iteracích, přičemž, v každé proběhlé iteraci dojde k vytvoření nějakého výsledku a ten se potom porovnává s požadavky. Druhá možnost je inkrementální přístup, kdy se software vyvíjí přidáváním jednotlivých komponent, jako přírůstků, přičemž každá z těchto komponent může být samostatnou vyvíjenou součástí software. Velmi známý zástupce z inkrementální kategorie je například model Vodopád, kde jsou jednotlivé etapy v posloupnosti za sebou a nejsou zde žádné cyklické návraty na předchozí etapy. Další model je pak kupříkladu Prototyp, kdy se vytvoří prototyp se základní funkcionalitou, který pak testují uživatelé a na základě jejich zpětné vazby se upravuje až do formy výsledného produktu. Nejznámější model vycházející z iterativního přístupu je potom Spirálový model. [6]

Další metodikou trochu odlišného charakteru jak pak například metodika UP neboli Unified Process, tedy unifikovaný/sjednocený proces. Je úzce spojená s níže zmíněným jazykem UML a jedná se o velmi otevřenou a základní metodiku, kterou je ale možno přizpůsobit jakémukoliv rozsahu projektu. Musí se však přizpůsobit jeho cílům a podmínkám. Proces UP se dělí na jednotlivé Iterace. Každá z nich prochází pěti fázemi, které jsou velmi podobné výše popsaným etapám životního cyklu. Jedná se o stanovení požadavků, analýzu, návrh, implementaci, testování a nakonec iteraci. Přičemž Iterace, které zde mohou probíhat i paralelně, jsou pro UP velmi podstatný prvek, neboť každá má svoje vlastní etapy, které jsou dále tvořeny dílčími aktivitami. Podstatný rozdíl u této metodiky v porovnání s ostatními spočívá v tom, že každá iterace tvoří tak zvaný baseline, tedy jakýsi soubor schválených a přezkoumaných prvků. Jedná se o základ k dalšímu vývoji. Přírůstky jsou zde pak rozdíly mezi dvěma následnými baselinami. Každá etapa může mít jednu nebo více iterací a je zakončená milníkem. [6]

Za zmínku pak ještě stojí Agilní metodiky vývoje software, což je další přístup. Používá se zejména pro software s menším měřítkem, jako jsou webové aplikace nebo malé systémy. Snaží se upřednostnit rychlost vývoje a zbavuje se náročnějších postupů. Princip fungování je postaven na tom, že se primárně tvoří základní a nejpodstatnější

funkcionalita a po otestování se přidávají další funkce. Iterativní vývoj se také vyznačuje velmi krátkými cykly. [6]

2.2.1 Jazyk UML

Nedílnou součástí tvorby softwaru a prototypování je grafické modelování a diagramy. Právě proto vznikl jazyk UML, tedy Unified Modelling Language. Jedná se o způsob vizualizace softwaru anebo jakéhokoliv jiného systému, pomocí skupiny diagramů. UML jich definuje 13 základních. Je to objektově orientovaný modelovací nástroj, používaný na znázornění širokého spektra projektů softwarového inženýrství. Je přijímán skupinou Object Management group jako standard pro modelování vývoje softwaru. Při vytváření UML diagramu se propojují tvary, reprezentující třídu nebo objekt s jinými tvary, které představují vztahy a tok informací a dat. Typy UML diagramů se řadí do tří kategorií, strukturální, diagramy chování a diagramy interakce. [8]

2.2.2 Diagram užití

Jedná se o diagram, popisující chování modelu. Vymezuje hranice modelovaného systému, umožňuje naň pohlédnout zvnějšku. Tento systém svým způsobem ukazuje chování systému z pohledu uživatele. Vyjadřuje, co se od systému očekává. Zobrazuje sled interakcí uživatele, jenž má přidělenou roli systémem. Diagram se skládá z případů užití, aktérů a vztahů mezi nimi. Přičemž Případ užití je množina akcí, která vede k dosažení určitého cíle. Používá se k definování jedné funkcionality systému. Aktér je potom role, která komunikuje s výše zmíněnými případy užití. Aktér může být aktivní nebo pasivní podle toho, zda on inicializuje případ užití, anebo je naopak iniciován. [9]

2.3 Informační systém

Informační systém je soustava, která obsahuje prvky jako subjekty, technické prostředky a metody. Tyto prvky spolupracují a zajišťují dohromady funkce systému, za využití technologií, s cílem zprostředkovat uživateli chtěný výstup. Lze také říci, že informační systém můžeme chápat jako celek, který se skládá z hardwaru a softwaru, ale také i lidí, kteří tento hardware a software obsluhují a používají a činnosti a procesy, které při tom vykonávají, za účelem šíření nebo získávání potřebných informací k plánování, řízení a rozhodování. Ve výsledku jsou informační systémy schopny pomoci v realizaci cílů v

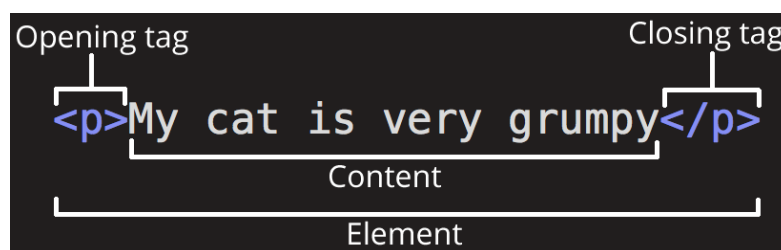
mnoha oblastech. Informační systémy můžeme dělit podle toho, jaký typ úkonu vykonávají a jaké informace zpracovávají, respektive, s jak kritickým stupněm rozhodování pomáhají. Informační systémy rozlišujeme do těchto kategorií: [10]

- **Transakční systémy** – Pomáhají zjednodušit nebo zautomatizovat ukládání dat a úlohy agendy jako například účetnictví nebo skladové zásoby, evidence. Jedná se asi o nejrozšířenější typ informačních systémů.
- **Informační systémy pro řízení** – Tyto systémy obsluhují naopak zobrazování informací a dávají je do nového kontextu, pomáhají se v datech lépe zorientovat. Jedná se například o přehledy zisku z nějaké období nebo přehled výkonnosti nějakého oddělení.
- **Systémy pro podporu rozhodování** – Tyto systémy slouží hlavně řídicím pracovníkům a jsou v principu jinak velmi podobné Informačním systémům pro řízení. Umožňují důležité analýzy, aby řídicí pracovníci mohli uskutečňovat správná rozhodnutí.
- **Informační systémy pro vrcholové řízení** – Jsou určené, jak z názvu vyplývá, pro vrcholové řízení, pomáhají tedy s rozhodnutími, které ovlivňují budoucnost podniku a poskytují k tomu informace na diagnostické úrovni.
- **Strategické informační systémy** – Systémy, jejichž hlavním úkolem je zlepšit konkurenceschopnost podniku. Jsou spojeny přímo s výrobky nebo výrobou podniku.
- **Prognostické systémy** – Vytvářejí složité předpovědi a analýzy a zvažují různé scénáře, které by mohly nastat. Vytvářejí prognózy a pomáhají tak utvářet vhodná rozhodnutí pro budoucnost firmy.

[10]

2.4 HTML

HTML je jeden z nejzákladnějších prvků, na kterých stojí internet, konkrétně webové stránky. HTML znamená HyperText Markup Language, tedy hypertextový značkovací jazyk. Určuje a definuje význam a strukturu obsahu na stránce. Kromě HTML jsou k definování vzhledu stránky využívány jiné prostředky, konkrétně CSS a k definování logiky zase JavaScript. O obou prostředcích se zmíním v dalších kapitolách. Samotný pojem Hypertext pojednává o odkazech, které propojují webové stránky. Odkazy jsou základní stavební prvky internetu. HTML používá značky k tomu, aby určil, jak se má obsah zobrazovat v prohlížeči, existuje v něm proto mnoho elementů pro různé účely zobrazování různého obsahu. Elementy se liší od ostatního obsahu značkami, které jsou uvnitř `< >` závorek, elementy samotné pak mohou mít ještě navíc atributy, které mají název a hodnotu a které ještě více specifikují funkcionalitu elementu. [11]



Obrázek 2 - Příklad HTML [2]

2.4.1 Historie HTML

HTML poprvé vzniklo v roce 1990 jako výsledek SGML neboli Standard Generalized Markup Language, což je standardní obecný značkovací jazyk. Je to složitá technická specifikace, která popisuje značkovací jazyky. Obzvlášť ty, které se používaly pro elektronickou výměnu dokumentů, správu dokumentů anebo publikování dokumentů. HTML původně vzniklo, aby umožnilo odborníkům, kteří nebyli znalí SGML, aby i tak mohli snadno publikovat a vydávat dokumenty vědeckého nebo technického charakteru. HTML bylo v tomto ohledu velmi užitečné, neboť umožňovalo propojení jednotlivých dokumentů pomocí hypertextových odkazů. Brzy si potenciálu HTML všimli i lidé mimo akademickou sféru a uvědomili si, že HTML je samostatné a dá se snadno naučit, takže se zanedlouho dočkalo mnoha dalších využití. Jak postupovala evoluce World wide webu, z HTML se brzy stal základní stavební prvek. [12]

2.4.2 HTML 5

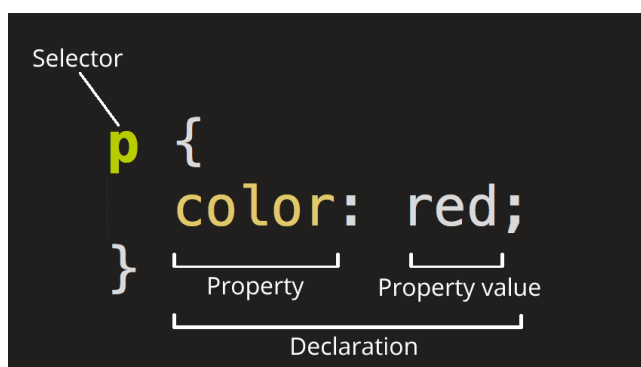
HTML5 je nejnovější verze standartu, který definuje HTML. Zahrnuje novou verzi jazyka HTML s novými elementy, atributy a chováním a s větší základnou technologií, které umožňují vytvářet složitější a bohatější webové stránky, obsah a aplikace. Tato verze byla vytvořena, aby jí mohly využít všechny internetové prohlížeče. Hlavní výhody oproti starším verzím HTML spočívají v:

- **Čistším kódu** – Umožňuje psát přehlednější kód, který lépe vyjádří, co je jeho význam, díky větší diverzitě elementů, které se liší svým účelem.
- **Offline možnostech** – Umožňuje webovým stránkám ukládat data lokálně na straně klienta a pracovat efektivněji, když jsou offline.
- **Multimédiích** – V HTML5 je nativní podpora pro mnohé audiovizuální formáty a již není potřeba využívat flash. Také 2D a 3D grafika je zde na mnohem lepší úrovni, umožňující lepší prezentaci obsahu.
- **Lepším přístupem** – HTML5 zlepšuje přístup z více typů zařízení. Také nabízí lepší optimalizaci a dokáže lépe využít hardwarových prostředků.
- **Stylování** – Nyní je možné využít lepší vzhled některých prvků, jako jsou ovládací prvky a jiné formulářové elementy na zadávání dat. Také došlo ke zlepšení popisu obsahu, takže je web dostupnější i pro lidi s nějakým postižením. [13]

2.5 CSS

CSS neboli Cascading Style Sheets, což znamená Kaskádové styly, je jazyk, pomocí kterého můžeme určit výsledný vzhled a prezentaci dokumentu napsaného v HTML nebo případně XML. Poprvé vznikl v roce 1994, vytvořil jej Håkon Wium Lie, který chtěl najít jednotný způsob tvorby vzhledu pro webové stránky. CSS jednoduše popisuje, jak se mají jednotlivé elementy renderovat na obrazovce, umožňuje aplikovat specifické vlastnosti na vybrané elementy. Jedná se o jeden ze základních jazyků internetu a je standardizován napříč všemi prohlížeči. CSS se vyvíjí v modulech, přičemž verze CSS1 je dnes už zastaralá, verze CSS2 je doporučena a CSS3, která je rozdělena do menších modulů se pomalu stává novým standardem. I přes to je však CSS3 na internetu hojně využíván. [14]

Kaskádové styly umožňují definovat, jakým způsobem se budou jednotlivé elementy zobrazovat, a to u každého elementu na stránce. Nemusí ale být součástí textu stránky, čímž jí dělá přehlednější a nenarušuje její strukturu. Styly také umožňují definovat jednotný vzhled určitého elementu v celém dokumentu bez nutnosti zápis opakovat pro každý. Do CSS se zapisují jednotlivé třídy, které mají dvě části: selektor a deklaraci. Navíc všechny deklarace se pak ještě skládají z dalších dvou částí: vlastnosti a její hodnoty. Tyto pravidla a deklarace je možné navzájem spojovat. Pomocí selektoru se styl naváže na odpovídající HTML element, u kterého se chce dosáhnout kýženého efektu. Je možné použít všechny elementy HTML. Vlastností, které můžeme pro každý element použít je nepřeberné množství. Kaskádové styly můžeme na stránce použít třemi způsoby. Buď můžeme pomocí elementu link připojit ke stránce oddělený soubor s CSS, nebo můžeme už přímo kód kaskádových stylů psát mezi značky style ve webové stránce, anebo je možné využít přímo atributu style u konkrétního elementu. [15]



Obrázek 3 - Příklad CSS [3]

2.5.1 CSS3

Narozdíl od CSS2 a CSS1, CSS3 již nedefinuje všechny své funkce a vlastnosti v jedné velké specifikaci, ale je rozdělena do několika oddělených modulů. Každý modul přidává nové vlastnosti anebo rozšiřuje ty, které byly definovány v předchozích verzích a to tak, že je zachována zpětná kompatibilita. První návrhy verze 3 se objevily už v roce 1999, krátce po vydání verze druhé a stále se jedná o verzi nejnovější. Modulární struktura také zajišťuje, že nové prvky jsou takto přidávány a jazyk se neustále vyvíjí spíše než, aby by se pracovalo na úplně nové verzi. V nové verzi CSS můžeme také nalézt mnoho nových funkcí a vylepšení, jako například zlepšená podpora pro zařízení s různou velikostí

obrazovky, lepší podpora napříč všemi webovými prohlížeči, lepší pozadí stránek, které již nemusí být jen jedno a statické, vylepšená podpora ohraničení a efektů hran a zaoblené rohy a stíny, propracovanější efekty pro text nebo například efekty a animace pro obrázky s vylepšenými přechody. [16] [17]

2.5.2 Bootstrap

Bootstrap je open source framework zaměřený na frontend webu, tedy na jeho vizuální podobu. Byl vyvinout dvěma designéry a vývojáři z firmy Twitter v roce 2010. Nejprve byl znám jako Twitter blueprint a byl původně zamýšlen pro účely firmy, aby došlo k sjednocení designu napříč jejími aplikacemi. Bootstrap je tedy vlastně taková souhrná knihovna již napsaných tříd, připravených pro použití. Obsahuje nepřehledné množství komponent a usnadňuje vývoj vizuální stránky webu. Zjednodušuje například tvorbu responzivního designu, tedy přizpůsobení velikosti obsahu pro různé velikosti obrazovky nebo nabízí pokročilý grid layout a systém flexboxů, čímž výrazně zjednodušuje tvorbu rozvržení webové stránky. [18] [19]

2.6 JavaScript

JavaScript je programovací jazyk, který umožňuje přidat komplexní logiku do webových stránek, poprvé vyšel v roce 1995 pod názvem LiveScript jako součást prohlížeče Netscape 2.0. Umožňuje tvorbu dynamicky se měnícího obsahu podle interakce s uživatelem. Je možné ho přidávat přímo do HTML stránky. Buď do značky `<script>`, anebo jako odkaz na oddělený soubor. Jedná se o kód, který se spouští na straně klienta, tedy před tím, než se odešle požadavek pro zobrazení HTML stránky. To znamená, že stránka může být statická HTML stránka, ale může obsahovat skripty, s kterým interaguje uživatel a které ovládají prohlížeč, který pak ve výsledku dokáže změnit obsah HTML stránky, což jí dělá dynamickou. Výhody JavaScriptu spočívají v nízké závislosti na serveru, neboť se vykonává na straně klienta a tím se tolik nezatěžuje. Také není nutné čekat na obnovení stránky, aby se docílilo kýžené akce a zároveň je možné vytvořit bohatší ovládací prvky na stránce, jako drag and drop komponenty a posuvníky. [20]

2.7 PHP

PHP neboli Hypertext Preprocessor nebo Personal Homepage je open-source skriptovací jazyk, který běží na straně serveru. To znamená, že server vyhodnotí žádost a php interpreter přeloží veškerý kód, který potom vloží do html stránky a tu potom vrátí. Narozdíl od Javascriptu, jehož logika se provádí na straně klienta. Výhoda, která z tohoto faktu pramení je, že stačí mít nainstalované PHP na serveru a klienti se už nemusejí o nic starat. Jedná se tedy o ideální multiplatformní řešení. PHP se většinou používá pro vývoj dynamických webů a webových aplikací. Je to skriptovací jazyk, což znamená, že k tomu, aby se mohl spustit není potřeba překlad, jedná se pouze o sled instrukcí. PHP je možné vložit přímo do HTML stránek, což umožňuje tvořit dynamický obsah. Právě proto je na něm založeno mnoho template systémů a frameworků pro tvorbu webových stránek. Pokud chceme do HTML stránky vložit PHP skript, vložíme jej do značky `<?php ?>`. Další předností PHP je zabudovaná podpora databází jako je mySQL a jiné. Jedná se tedy o velmi vhodný programovací jazyk pro tvorbu webových aplikací. [21]

2.7.1 Historie PHP

PHP vzniklo zprvu roku 1994 jako systém pro evidenci návštěvnosti webových stránek. Těšil se však velké oblibě, a proto ho jeho autor Rasmus Lerdorf začal doplňovat o novou funkcionalitu. Vytvořil k němu dokonce i dokumentaci, a nakonec ho vydal jako Personal Home Page Tools. Kromě toho vyvinul Rasmus Lerdorf i systém, který umožňoval začlenění SQL příkazů do stránek, uměl i vytvářet webové formuláře a zobrazovat výsledky dotazů. Tento program pro přístup k databázi se jmenoval Form Interpreter. Ve své druhé verzi si získal masivní podporu a byl včleněn do jazyka PHP. Jednalo se pak o jednoduchý programovací jazyk, který se zadával přímo do HTML stránek. V roce 1998 vyšel PHP 3 na kterém spolupracoval s Andi Gutmansem a Zeev Suraskim. Nová verze běžela i na operačním systému Windows a nabízela velkou škálu rozšiřitelnosti. PHP 4 vyšlo pak v roce 1999 a přinášelo s sebou nový Zend engine, který výrazně zvyšoval výkon a přinesl mnoho nových funkcí jako podporu velkého množství systémů databází a API prostředí. PHP 5 pak vyšlo v roce 2004 a představilo Zend engine 2 s novým objektovým modelem a s ještě větším množstvím nových funkcí. [22]

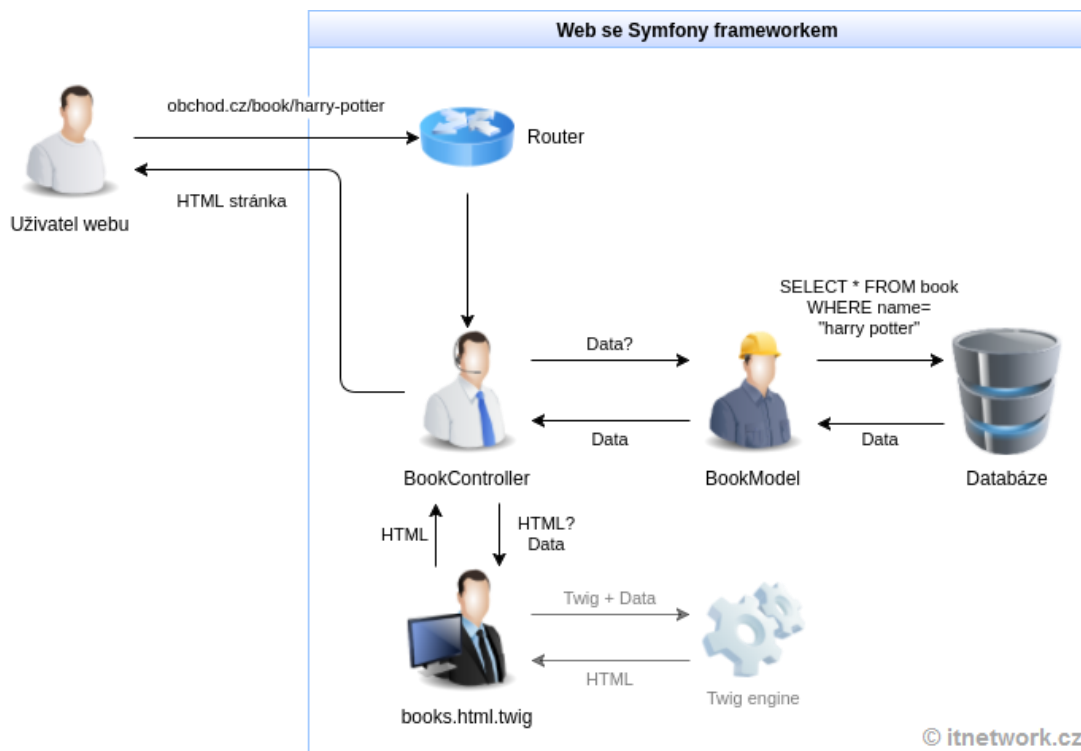
2.7.2 Symfony Framework

Symfony je Open Source PHP framework, vycházející ze struktury MVC. Unikátní je tím, že to vlastně nemusí být kompletní framework, pokud to není potřeba. Skládá se totiž z mnoha oddělených komponent, které se dají nezávisle na sobě do projektu přidávat. Je tedy možné použít kompletní framework nebo začít s odlehčenou verzí, obsahující pouze to nejnútnejší. To je výhodné například, pokud chceme tvořit pouze webové API. Je ale možné tuto verzi využít i pro jiné projekty a podle potřeby moduly přidávat. Každý z nich má i vlastní git a podporu. Co je to ale Framework? Jedná se o knihovnu již předem vytvořených funkcí a tříd, které stačí pouze aplikovat a není tak nutné znovu vytvářet známé principy. MVC je známý, osvědčený a častý model fungování webových frameworků. Zkratka MVC znamená Controllers – Models – Views a míní se tím, že logika a fungování aplikace se programuje na 3 spolupracující celky, tedy:

[4] [23]

- **Controllery neboli řízení** – Jedná se o části kódu, které fungují jako prostředník mezi uživatelem a zbytkem kódu. Zpravidla rozpoznávají URL, které uživatel zadává. Potřebné informace předá dalším modulům a získá z nich adekvátní data a konkrétní výsledek předá do pohledu, který se pak uživateli zobrazí.
- **Modely neboli logika** – Jedná se o logiku aplikace nebo o databázovou vrstvu, tedy třídy, jež se starají o získávání dat z databáze, kdy každá entita v databázi má většinou svoji třídu se všemi potřebnými vlastnostmi.
- **Views neboli výstup** – Jedná se o html stránku, do které jsou vloženy data controllerem a jež je vrácena uživateli. Ve většině frameworku se jedná o nějakou šablonu. Symfony pro tento účel používá svůj vlastní template engine Twig. Je to nástroj na šablony, který umožňuje do html kódu vkládat logiku a data z PHP.

[4]



Obrázek 4 - Znáznornění fungování MVC modelu [4]

Symfony bylo poprvé vytvořeno v roce 2005 francouzskou společností Sensio Labs pro vývoj webových aplikací, pro její klientelu. Postupně se framework však rozrostl a získal obrovskou popularitu. Dnes se nachází již v páté verzi. [24]

Obsahuje nepřeberné množství modulů a funkcionality, které je možné přidávat do konkrétní instance Symfony z oficiální stránky, jako například a hlavně:

- **HttpFoundation** – jež funguje jako objektově orientovaná nástavba pro PHP na tradiční webové úlohy.
- **Security** – tedy třída pro bezpečné registrování nových uživatelů, přihlašování, šifrování hesel a bezpečné rozlišování uživatelů a jejich rolí.
- **Doctrine ORM** – jedná se o Symfony PHP modul, který je určen pro komunikaci s databází. Udrží pro každou tabulku v databázi třídu s vlastnostmi, které odpovídají atributům a umožňuje tak s daty z databáze nakládat jako s objekty.
- **Routing** – neboli směrování – jedná se o jednoduchý, ale důležitý modul, který podle uživatelem zadaného URL zavolá příslušný Controller, který pak vyřídí příslušný dotaz.

[25]

2.7.3 Composer

Composer je konzolový nástroj pro PHP, inspirovaný Npm od nodu a Bundlerem od ruby, který umožňuje správu PHP komponent. Je to PHP dependency manager, tedy správce závislostí. Umožňuje stažení jednotlivých PHP balíčků a všech nutných závislostí, tedy jiné balíčky, které jsou vyžadovány k tomu, aby celek fungoval korektně. Jeho další výhodou oproti například podobnému staršímu správci PEAR nebo Yum a Apt je, že nestahuje balíčky do systému globálně, nýbrž jenom v rámci projektu. Což je praktické, pokud vytváříme více projektů a nějaký z nich vyžaduje starší verzi některého balíčku. Composer tedy umí instalovat balíčky i globálně, ale není to jeho primární účel. Podporuje velké množství PHP frameworků, jako právě Symfony a mnoho dalších. Composer tedy umožňuje vcelku jednoduše instalovat balíčky z jednotlivých frameworků, bez toho, aniž bychom je museli celé stahovat, a umí je kombinovat do jednoho projektu, pokud je to něco, co potřebujeme. Je však samozřejmě možné pomocí něj přidávat jednotlivé balíčky dle potřeby do frameworku, který právě využíváme, jako například přidávání jednotlivých modulů do Symfony. [26] [27]

2.8 MySQL

MySQL je systém řízení databáze od společnosti Oracle. Umožňuje přistupovat k datům a zpracovávat a přidávat je do databáze. Přičemž databáze je strukturovaná kolekce dat. V případě MySQL se jedná o relační databázi, což znamená, že data jsou uložena v tabulkách. Model ukládání dat do databáze funguje tak, že každá tabulka se nazývá entita a shromažďuje jednotlivé vlastnosti, které se nazývají atributy. v databázi se nastavují pravidla mezi tabulkami, které určují kardinalitu, tedy pravidla o tom, kolik relací může být mezi tabulkami a každý řádek neboli záznam je jednoznačně označen primárním klíčem. Dále se ukládají informace o unikátnosti nebo o jiných omezeních a systém řízení databáze, zajišťuje, že všechna tato pravidla budou dodržena. [28]

2.8.1 Historie MySQL

MySQL byl poprvé vytvořen Švédskou společností MySQL AB v roce 1995 třemi vývojáři – Michaellem Wideniusem, Davidem Axmarkem a Allanem Larssonem. Hlavní účel byl zajistit efektivní a spolehlivý systém pro ukládání dat. V roce 2000 bylo MySQL vydáno jako Open-Source pro veřejnost. V roce 2005, kdy už bylo MySQL velký a úspěšný projekt jej zakoupila firma Oracle což vyústilo, krom jiného, ve vydání MySQL 5, který přinesl mnoho nových funkcí. V roce 2008 byl MySQL zakoupen firmou Sun Microsystems, ale protože to nakonec nebyl úplně vhodný krok vpřed, v roce 2009 Oracle zakoupil rovnou celé Sun Microsystems a získal tak MySQL zpět. Z MySQL také vzešel MariaDB, vytvořil jej Michael Widenius, bývalý zaměstnanec Sun Microsystems. [29]

2.8.2 SQL

SQL neboli Structured Query Language, což v češtině znamená strukturovaný dotazovací jazyk jest standardní dotazovací jazyk určený ke komunikaci s databází. Užívá poměrně jednoduché příkazy podobné mluvené angličtině a je proto relativně jednoduchý na pochopení. První prototyp vyvinula firma IBM podle návrhu Dr. E.F. Codda. V roce 1979 vyšel první SQL produkt, pojmenovaný Oracle, který vydala firma Relational Software, Inc. Která se později stala Oracle Corporation. SQL je také považován za standard, uznaným americkým ANSI od roku 1986 v podobě SQL od firmy IBM a v roce 1987 přijala SQL jako standard i mezinárodní organizace ISO, nejnovější verze se jmenuje SQL-99. [30]

2.9 Apache Web Server

Jak název napovídá, Apache je webový server, je velmi populární a open source a funguje na všech hlavních platformách. Umožňuje uživatelům přistupovat na webové stránky, uložené na fyzickém serveru tím, že odpovídá na jejich žádost zadanou jako url. Webový server poté načte požadovanou stránku. Chová se tedy jako prostředník mezi fyzickým serverem a uživateli, kteří se snaží zobrazit webové stránky na něm uložené. Když si uživatel přeje zobrazit stránku, prohlížeč vyšle žádost na server a ten vrátí zpět všechny vyžádané stránky ve formě statické HTML stránky. Webové servery jsou schopné zpracovávat soubory napsané v různých programovacích jazycích a jsou schopny je vrátit

jako HTML. Server a klient komunikují pomocí protokolu HTTP. Apache je velmi konfigurovatelný server, má modulární strukturu a obsahuje moduly pro bezpečnost, cachování nebo přihlašování a jiné. [31]

Jméno Apache pochází z přezdívky "a patchy server" neboť dříve, když chtěli vývojáři tento server použít, bylo nutné ho vlastnoručně upravit, tedy aplikovat několik vlastních "patchů". [32]

Apache byl poprvé vytvořen Robertem McCoolem ve Spojených Státech. Vyšel v roce 1995 a velmi rychle získal na popularitě. Na Apache server běží více než 50 % internetu. Původně se jmenoval NCSA HTTPd Web server a McCool ho vytvořil, když studoval na vysoké škole University of Illinois. Apache webový server nyní spravuje a vyvíjí Apache Software Foundation. [33]

2.9.1 Laragon

Laragon je přenositelné, izolované a univerzální vývojové prostředí pro PHP, Python nebo Javu a mnoho dalších. Jedná se o modifikovatelnou serverovou aplikaci, která obsahuje konkrétní serverové řešení pro vývoj webových aplikací pro operační systém Windows. Je možné na něj nainstalovat MySQL server a Apache server a mnoho dalších. Obsahuje HeidiSQL program pro správu databáze a umožňuje doinstalování například phpMyAdmin. Umožňuje přístup na stránku pomocí zkrácených odkazů a je přenositelný. Je tedy možné jej bez obav kopírovat v systému nebo mezi počítači a paměťovými médii. Je izolovaný od služeb operačního systému a všechny servery, které jsou přes něj nainstalovány jsou automaticky nakonfigurovány, aby spolupracovaly. [34]

3 Vlastní práce

V této kapitole popíši průběh návrhu a vývoje webové aplikace, představím její výsledný vzhled a funkcionalitu a představím ukázky z kódu a shrnu výsledky testování. Ve všech fázích vývoje i návrhu aplikace jsou používány technologie a poznatky popsané v předchozí kapitole.

3.1 Počáteční Analýza

Aplikace je určena pro základní školu, při jejím návrhu jsem spolupracoval s více školami a učiteli, aplikace tedy není vyvíjena specificky pro jednu konkrétní základní školu. Cílem této webové aplikace je podpora procesů na základní škole a chceme-li toto učinit efektivně, musíme se podívat, na potenciální typy uživatelů, pro které by tato aplikace mohla být určena. Na základní škole jsou vesměs 4 typy osob, kterých se týká rozsah zájmu aplikace. Jsou to žáci, kteří budou aplikaci používat, převážně, aby získali nutné informace pro své studium, a také jejich rodiče, kteří si budou chtít zjišťovat informace o studiu svých ratolestí. Dále jsou to učitelé, kteří aplikaci využijí taktéž pro získání praktických informací o svých žácích. Též budou moci získat informace pro podporu výuky. Poslední zainteresovaná skupina jsou správci systému, kteří se budou starat o plynulý běh aplikace, opravování chyb a o vkládání osob do databáze a jinou administrativní činnost.

Když se na celou situaci zahledíme, není těžké se dopracovat k celkem očividné skutečnosti, že ve webové aplikaci budeme nakládat s celkem citlivými daty, a to s daty určenými pouze pro vnitřní účely základní školy. Je proto žádoucí, aby byl systém zajištěn přihlašованиеm, aby se do něj dostaly jen povolané osoby. Nadále bude samozřejmě nutné korektní rozdělení práv pro jednotlivé uživatele v aplikaci, neboť je nežádoucí, aby si například studenti přidávali známky sami. Po bližším posouzení fungování procesů na základní škole a konzultacích s administrátory a učiteli, autor zvážil, že ty nejpodstatnější funkce a zároveň ty, které by mohly nejvíce benefitovat z toho, že budou přístupné v aplikaci, budou evidence studentů a učitelů, informace o nich pro účely matriky a možnost vedení absencí a suplování a poté hlavně tvorba a zobrazování rozvrhů a známek. Při rozhovorech s učiteli a adminy se autor ptal, co je pro ně v systému důležité a jaké funkce

by chtěli mít v systému novém. Odpovědi by se dali shrnout asi tak, že systém existuje hlavně kvůli matrice a usnadnění tvorby výkazů, a tedy XML souboru, který školy odevzdávají na stránky ministerstva každé sběrné období. Dále by všichni chtěli, aby bylo dobré propojení modulů a aby byla aplikace přehledná. Jako nejdůležitější funkce se většinou uvádí docházka, suplování a známkování nebo rozvrh a již zmíněná matrika.

Aplikace má tedy uživatelům umožňovat:

- **Přihlašování a Registraci**
- **Evidovat žáky a jejich rodiče a uživatele**
- **Zobrazení podrobností o žákovi nebo učiteli či rodiči, známce či předmětu**
- **Možnost přidání a editace žáků a učitelů a rodičů**
- **Přidávání a zobrazování rozvrhů a známek**
- **Možnost evidence docházky a nastavení suplování a omluvenek**
- **Zasílání zpráv mezi uživateli**
- **Export dat z matriky do souboru XML**
- **Další drobná vylepšení**

3.1.1 Funkční požadavky

Funkční požadavky jsou v případě této práce diktovány povahou prostředí, pro které se webová aplikace má vyvíjet. Jedná se o zadání, které bylo vytvořeno na základě konzultací s učiteli a administrátory, pracujícími pro základní školy. Požadavky, které plynou ze situace jsou následující:

- **Separace uživatelů podle rolí a možnost přihlášení** – V aplikaci musí být integrován systém přihlašování, aby se do ní dostaly jenom pověřené osoby, tedy žáci a zaměstnanci školy. Jednotliví uživatelé musí mít také patřičná práva podle jejich role.
- **Aplikace musí umožňovat přidávat nové uživatele** – Aplikace musí umět komunikovat s databází a umožňovat vkládat záznamy i do evidence žáků a učitelů a uživatelů. Přičemž do databází bude mít právo vkládat nové údaje pouze administrátor.
- **Informace o osobách je možné zobrazit, editovat a mazat** – Možnost prohlížet, editovat a mazat záznamy je nutná pro případ změny aktuálních dat nebo pro případ odchodu žáka či učitele. Nebo pouze pro vyhledání informací. Přičemž

administrátor bude jediný, kdo bude mít možnost záznamy mazat a editovat v plné míře. Učitel bude moci editovat pouze méně důležité informace u žáků anebo na vlastním profilu. Žák bude moci pouze zobrazovat informace buď o sobě nebo informace o všech učitelích, které jsou relevantní pro studium.

- **Aplikace musí umožňovat přidávání nových známek a rozvrhů** – Tyto operace bude schopen vykonávat pouze administrátor anebo učitel. V patřičných formulářích jim bude umožněno filtrovat podle tříd a v případě známek i podle předmětu, pro snazší administrativu.
- **Aplikace musí umožnit zobrazení známek a rozvrhu** – V případě žáka, aplikace musí být schopna zobrazit pouze jeho vlastní známky. Bude však umožňovat zobrazovat také známky z minulých let, přičemž bude zobrazovat pouze relevantní známky a předměty podle ročníku. Administrátor a učitel si budou moci zvolit konkrétní třídu, pro kterou chtějí informace zobrazit.
- **Aplikace musí umožňovat správu docházky** – Vedení docházky je na škole důležitý úkon, aplikace bude umět zobrazovat žáky podle třídy, data a hodiny s možností nastavení absence. Tato funkce bude umožněna pouze učiteli nebo správci, žák pouze uvidí souhrn své vlastní absence a jeho rodič bude moci přidávat omluvenky.
- **Aplikace musí umožňovat správu suplování** – Administrátor bude moci nastavovat podle data a hodiny absence i pro učitele a přidělit náhradního vyučujícího do hodiny. Samotný učitel potom bude moci zobrazit osobní rozvrh, kde uvidí své běžné hodiny a suplování.
- **Komunikace mezi uživateli** – I přesto, že se nejedná o zásadní funkci, je výhodou mít základní funkci posílání zpráv mezi všemi uživateli.
- **Export dat z matriky** – Školní systém existuje hlavně pro účely evidence a usnadnění zpracování a vykazování dat z ní. Aplikace proto musí mít správně nastavenou databázi, zejména pro evidenci žáků a bude z ní umět exportovat soubor XML pro ministerstvo školství. Tento úkon bude moci vykonávat pouze administrátor.

3.1.2 Metodika vývoje

Jelikož projekt vyvíjí autor práce sám a nejedná se o velký projekt v porovnání s korporátními systémy na zakázku, volba metodiky vývoje software nemá takový dopad. Tato skutečnost dělá ale také celou situaci poměrně jednoduchou. V tomto rozsahu se jeví jako nejvýhodnější model Prototyp, kdy se nejprve vytvoří jakési jádro s nejdůležitější funkcionalitou, to se pak testuje a podle toho, zdali splňuje požadavky a cíle je upravováno a doplňováno. Webová aplikace zprvu obdrží jenom to nejpodstatnější jako přihlášení a postupně se bude ladit a budou se k ní přidávat nové funkce dle míry spokojenosti autora a zpětné vazby od uživatelů při testování.

3.1.3 Případy užití

Nyní rozeberu aplikaci podle use casů. Následující diagram znázorňuje obecnou funkcionalitu celé aplikace, je tvořen z pohledu administrátora, který je tedy aktérem diagramu, protože ten má přístup ke všem funkcím a plní tak nejlepší ilustrativní účel. Jednotlivé případy užití značí veškeré funkce, které je možné v systému provádět ze strany administrátora po přihlášení. Ostatní dvě role se liší jenom v tom, že mají omezenější práva, nevidí některé ovládací prvky nebo nemohou přejít na některé stránky aplikace. Učitel například může prohlížet a měnit rozvrhy a známky, nemůže však přidávat nebo zásadně měnit záznamy v evidenci učitelů a žáků. Může měnit data menší důležitosti na svém profilu nebo u jakéhokoliv žáka. Žák potom nemůže měnit vůbec nic. Může informace pouze zobrazovat. Nemá ani žádný přístup k seznamu ostatních žáků. Má přístup pouze k seznamu učitelů, kde vidí jenom méně citlivé informace. Vidí pouze svůj rozvrh a své známky. Rodič potom vidí vše, co vidí jeho děti a může zobrazovat informace pro každé, má-li jich v systému více. Navíc pro ně může přidávat omluvenky, ale jinak má stejná práva jako žák.

Případy užití:

1. Zobrazit Studenty

Pokud uživatel není student nebo rodič, může si zobrazit seznam žáků a jejich profily. Když uživatel klikne na zobrazení studentů, bude přesměrován na stránku, kde se zobrazí tabulka všech studentů v databázi. Krom toho je zde tlačítko pro přidání nového studenta, které uživatel vidí pouze, pokud je administrátor. Dále je zde karta pro zobrazení seznamu studentů, členěného podle tříd. Pokud uživatel zvolí možnost přidání nového studenta, je přesměrován na formulář, kam je nutno vyplnit všechny údaje. Formulář nedovolí uložit data, pokud nejsou vyplněna povinná pole. Formulář je možné potvrdit tlačítkem uložit. Po jeho stisknutí je uživatel přesměrován zpět na seznam studentů. Při kliku na kartu pro výpis studentů podle tříd, se zobrazí velmi podobná tabulka. Nyní jsou zde však minimalizované posuvné prvky odpovídající každé jedné třídě, ukrývající tabulky se žáky z oné třídy. Tato se zobrazí po rozkliknutí. Nad samotnou tabulkou je navíc řada s tlačítky s názvy všech tříd, které umožňují rychle přejít na pozici vybrané třídy v seznamu, bez nutnosti posouvat stránku a hledat ji v seznamu. Položky v tabulce samotné jsou interaktivní a když na nějakou uživatel klikne, je přesměrován na detailní profil vybraného žáka. Zde se zobrazí všechny podstatné informace, pokud je uživatel učitel nebo administrátor, vidí všechny údaje žáka, pokud ne, vidí pouze zjednodušený výpis. Popřípadě se může zobrazit také jeho profilový obrázek, pokud má nějaký uložený a také poznámky k jeho osobě, pokud mu byly nějaké přidány. Naspodu stránky se vygeneruje trojice tlačítek pro smazání, úpravy informativních dat a úpravy citlivých dat. Uživatel může použít a vidí funkci úpravy citlivých dat a mazání jenom pokud je administrátor. Pokud je žák, nevidí a nemůže využít tlačítko žádné a ani nemůže přistoupit na jiný profil než ten vlastní. Po kliknutí na tlačítko pro smazání se záznam smaže a uživatel je přesměrován na výpis studentů. Po kliknutí na tlačítko pro úpravu citlivých dat se zobrazí předvyplněný formulář pro přidání nového žáka, umožňující aktualizovat informace. Po kliknutí na úpravu informativních dat se zobrazí okénko s předvyplněnými informativními daty umožňující je aktualizovat nebo přidat. Naspodu formuláře se objeví tlačítko uložit. Formulář nedovolí uložení, pokud nejsou vyplněná povinná data. Po uložení je uživatel přesměrován zpět na profil Žáka.

V této sekci nebudu popisovat funkci zobrazení všech učitelů, neboť princip je naprosto totožný a liší se pouze v maličkostech, jako v typu polí ve formuláři pro přidání nového učitele, neboť má trochu jiné atributy v databázi. Seznam všech učitelů a jejich profil si ale může zobrazit každý. Mazat a přidávat nebo upravovat citlivá data může opět jenom administrátor. Učitel sám si může upravit informativní data na svém profilu, ale ne jiným učitelům. Taktéž na jeho profilu se zobrazují odlišné informace než na profilu žáka, konkrétně, jaké třídy je třídním učitelem, jaký je jeho kabinet, jeho telefon, email a webové stránky. Žák samozřejmě na profilu učitele nevidí žádná tlačítka a ani nemůže využít jejich funkce.

2. Zobrazit Rozvrhy

Každý uživatel má možnost zobrazit stránku s rozvrhy. Pokud na ní uživatel klikne a je učitel nebo administrátor, uvidí tabulku se dny v týdnu jako řádky a časy začátků vyučovacích hodin jako sloupce. Pod tabulkou bude taktéž vysouvací nabídka pro výběr, pro jakou třídu si přeje uživatel zobrazit rozvrh a tlačítko pro potvrzení. Pokud je uživatel žák, tyto dva prvky neuvidí a automaticky se mu zobrazí rozvrh pro jeho třídu. V tabulce budou barevně oddělená políčka s předměty. Na místech, kde žádný předmět nebude, bude odkaz s textem "přidat". Tento text uživatel neuvidí, pokud je žákem nebo rodičem. Pokud na odkaz uživatel klikne, zobrazí se formulář pro přidání nového předmětu do rozvrhu, jež bude mít předvyplněnou třídu a čas, a bude možné doplnit učebnu, předmět a vyučujícího z vysouvací nabídky. Naspodu formuláře se objeví tlačítko uložit, pokud na něj uživatel klikne, zobrazí se opět tabulka s rozvrhem. Je možné kliknout také na políčko s již zadaným předmětem, což zobrazí informační stránku o předmětu. Zobrazí se na ní jméno předmětu, v jaké učebně se bude konat, jaký vyučující ho bude učit, v kolik hodin začíná a pro jakou třídu. Pokud je uživatel učitel či administrátor, naspodu stránky taktéž uvidí dvojici tlačítek – pro úpravu a pro mazání. Tlačítko pro mazání smaže položku v rozvrhu a přesměruje učitele zpět na stránku s rozvrhem. Tlačítko pro úpravu zobrazí předvyplněný formulář pro zadávání nového předmětu do rozvrhu a umožní tak uživateli upravit informace v něm, poté bude přesměrován zpět na stránku s rozvrhem.

3. Zobrazit známky

Každý uživatel má možnost si zobrazit známky. Žákovi a rodičovi se však zobrazí jiná stránka než ostatním dvěma rolím. Pokud je uživatel žák či rodič, načte se tabulka, kde řádky jsou jednotlivé předměty pro jeho třídu v aktuálním roce a sloupce číslice určující počet sloupců, přičemž poslední sloupec je vždy průměr. Nad tabulkou se objeví výsuvná nabídka pro výběr ročníku, kterým si žák již prošel a tlačítko pro potvrzení. Pokud na něj uživatel klikne, stránka se obnoví a objeví se předměty, které měl žák ve zvoleném roce a známky taktéž. Uživatel má také možnost klikat na jednotlivé známky, což načte stránku detailu známky s doplňujícími informacemi, jako hodnotu známky, kdo jí udělil, z jakého je předmětu, kdy byla zapsána a poznámka.

Pokud je uživatel administrátor nebo učitel a zobrazí si stránku se známkami, uvidí výsuvnou nabídku pro zvolení roku, která je povinná a její výchozí hodnota je aktuální rok. Poté se pod ní načtou výsuvné nabídky pro výběr třídy a pro výběr předmětu, tyto nabídky povinné nejsou a je možné vybrat hodnotu v obou, pouze jedné, nebo žádné. Dále se pod nimi objeví tlačítko pro potvrzení výběru zvolených omezení. Pokud na něj uživatel klikne, stránka se obnoví a pod nabídkami se objeví tabulka se seznamem žáků jako řádky a počtem známek jako sloupce, předposlední sloupec je pak vždy průměr a poslední sloupec obsahuje odkaz "přidat", pokud na něj uživatel klikne, zobrazí se formulář pro přidání známky. Kolonky pro učitele, žáka a datum jsou předvyplněné, přičemž datum je neměnné, vždy je možné zadat jenom aktuální datum. Naspodu stránky bude tlačítko pro uložení. Pokud na něj uživatel klikne, nová známka se uloží a dojde k přesměrování na stránku se známkami. Pokud uživatel klikne na již vyplněnou známku, zobrazí se stránka s detaily o známce, stejné, jako je tomu, pokud je uživatel žák. Ovšem pokud je uživatel učitel nebo administrátor, vidí naspodu stránky ještě další dvě tlačítka pro úpravu a smazání. Po kliknutí na tlačítko pro mazání se známka smaže a uživatel bude přesměrován zpět na stránku se známkami. Po kliknutí na tlačítko pro úpravu se zobrazí předvyplněný formulář pro přidání nové známky, umožňující tak údaje ve známce aktualizovat nebo doplnit.

4. Registrovat

Pokud je uživatel administrátor, má možnost registrovat nové uživatele. Zobrazí se jednoduchý formulář s možností vyplnění uživatelského jména a hesla, přičemž heslo vyžaduje potvrzení ve formě ověřovacího druhého pole. Dále je zde možnost zaškrtnout roli, jež bude účet mít, pomocí zaškrťovacích políček. V neposlední řadě také dvě výsuvná menu, kde bude možné přiřadit k novému účtu konkrétního učitele nebo žáka z databáze. Naspodu stránky bude tlačítko pro registraci a po kliknutí na něj bude nový účet zaregistrován a uživatel přesměrován na domovskou stránku. Formulář nedovoluje uložení bez zadání všech povinných údajů.

5. Zadat docházku

Pokud je uživatel administrátor nebo učitel, po kliknutí na možnost Absence je přesměrován na stránku s tabulkou s žáky. Tabulka zobrazuje jméno žáka, jeho celkový počet zameškaných hodin, datum, název předmětu, odkaz pro více informací, který uživatele přesměruje na seznam všech absencí daného žáka, a nakonec údaj, zda je žák přítomen s možností jej změnit. Po kliknutí na odkaz změnit, je uživatel přesměrován na formulář s předvyplněnými údaji předmětu, učitele, data a hodiny a s možností zaškrtnutí absence. Po uložení je uživatel přesměrován zpět na tabulku s docházkou. Uživatel má možnost zvolit konkrétní datum a čas začátku hodiny a třídu pro kterou chce docházku zobrazit. Tyto údaje jsou předvyplněny podle času a rozvrhu, pokud daná hodina probíhá. Pokud uživatel klikne na jméno žáka v tabulce, je přesměrován na jeho profil s podrobnými údaji.

Pokud je uživatel Žák, po kliknutí na možnost Absence je přesměrován na stránku s výpisem všech jeho absencí. Je zde uveden také počet zameškaných hodin a možnost filtrovat i absenci z minulých let. Ve výpise je uveden předmět, datum, čas a bližší informace o absenci. Pokud je uživatel rodič žáka, vidí to samé, co on, ale má navíc i možnost přidat novou omluvenku. Po kliknutí na tlačítko je přesměrován na formulář s možností zadání nové omluvenky s možností výběru data, předmětu, hodiny a zadání textu omluvenky. Po uložení je přesměrován zpět

na výpis absencí žáka, kde má poté i možnost kliknut na neomluvenou absenci a omluvit jí, i když už existuje.

6. Zadat Suplování

Pokud je uživatel administrátor, po kliknutí na tlačítko Suplování je přesměrován na tabulku se seznamem učitelů. V seznamu je uvedeno jméno učitele, na které když uživatel klikne, je přesměrován na jeho profil, datum, hodina a předmět a případný suplující učitel, pokud tento učitel chybí. Dále se zde nachází informace, zda je učitel ve škole nebo ne a možnost jí změnit. Po kliknutí na tento odkaz je uživatel přesměrován na formulář s možností výběru učitele, zda je přítomen, data a hodiny, tyto údaje budou předvyplněné. Také je zde možnost zvolit učitele, který bude suplovat zvolenou hodinu, pokud původní učitel v tento čas chybí. Po kliknutí na tlačítko uložit, je uživatel přesměrován zpět na seznam učitelů. Krom seznamu je zde možnost filtrovat i podle hodiny a podle data. Aplikace prochází rozvrhy učitelů a zobrazuje pouze relevantní seznam učitelů podle těchto nastavení. Taktéž se zde nachází tlačítko pro zobrazení seznamu všech suplování. Zde má uživatel možnost nastavit datum od a do kterého data se má suplování zobrazovat. Po kliknutí na zobrazit se načte seznam všech zadaných suplovaných hodin s datem, časem, předmětem, suplujícím učitelem a informací o chybějícím učiteli. Uživatel má také možnost suplování smazat anebo upravit.

Pokud je uživatel učitel, po kliknutí na tlačítko suplování se mu zobrazí jeho osobní stálý rozvrh s přidanými hodinami, které supluje. Má možnost zvolit pro jaký týden chce rozvrh zobrazit pomocí nastavení data od a data do. Po kliknutí na předmět, se mu zobrazí bližší informace, jako třída nebo učebna.

7. Zobrazit zprávy

Tuto funkci si ve stejné míře může zobrazit každý uživatel. Po kliknutí na tlačítko Zprávy se zobrazí seznam přijatých zpráv, označených jejich předmětem. V okně je také záložka pro zobrazení odeslaných zpráv. Po rozkliknutí zprávy se zobrazí informace o odesílateli, data odeslání a samozřejmě samotný text zprávy. Uživatel má možnost odpovědět anebo vytvořit novou zprávu. Tyto možnosti otevrou nové okno, kde může uživatel vybrat příjemce, zadat předmět a text zprávy a zprávu odeslat. Po odeslání je přesměrován do záložky odeslaných zpráv, kde se nachází seznam odeslaných zpráv.

V následujících dvou podsekcích uvedu dva příklady možných scénářů využití aplikace a funkcí, popsaných výše jako příklad, uvedu i příklady grafického návrhu rozvržení webové aplikace.

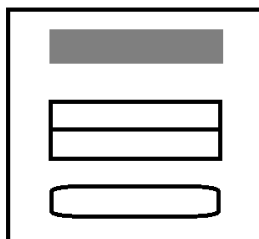
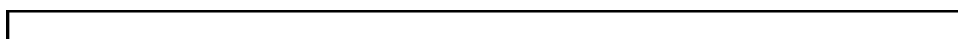
3.1.4 Scénář 1 – Přihlášení

Počítá se, že uživatel zná své přihlašovací údaje, je tedy již registrovaný administrátorem, který je jediný, kdo tuto akci může provést. Samotný Administrátor musí být do databáze přidán manuálně, před vpuštěním aplikace do provozu.

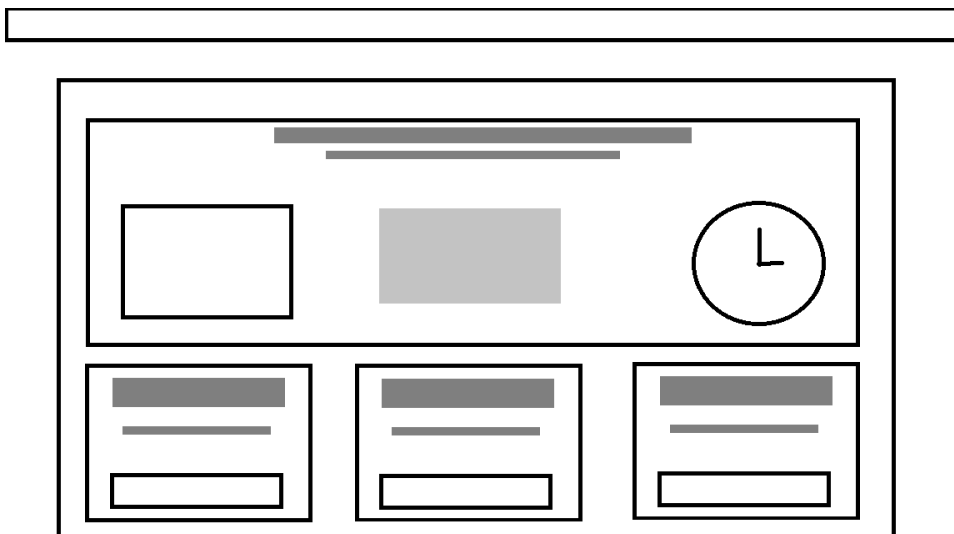
Tabulka 1 - Scénář Přihlášení

| | Aktér | System |
|-----|--|--|
| 1 | Nepřihlášený uživatel kliknul na přihlásit na domovské obrazovce | |
| 2 | | System zobrazí přihlašovací formulář |
| 3 | Uživatel zadá přihlašovací údaje a odešle je | |
| 4 | | System údaje ověří |
| 4.1 | | System uživatele přesměruje na přizpůsobenou domovskou obrazovku |
| 4.2 | | Když údaje nejsou korektní, system zobrazí chybovou hlášku |

Na následujících obrázcích bude znázorněna přibližná podoba budoucího formuláře pro přihlašování a pro domovskou obrazovku dle scénáře, ve formě wireframu.



Obrázek 6 - Wireframe přihlašovacího formuláře



Obrázek 7 - Wireframe domovské obrazovky

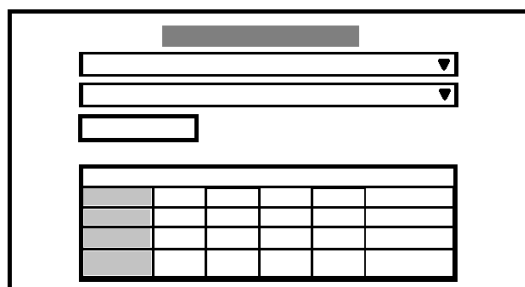
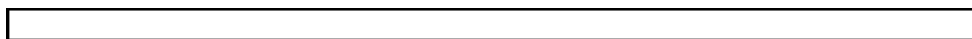
3.1.5 Scénář 2 – Znamky

Tento scénář popisuje proces zobrazení známek učitelem pro konkrétní třídu a předmět a následné přidání nové známky do systému.

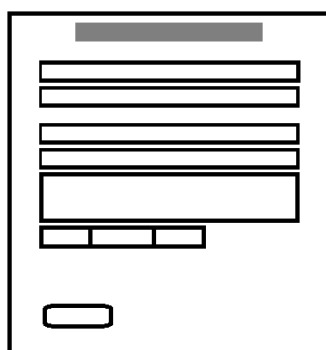
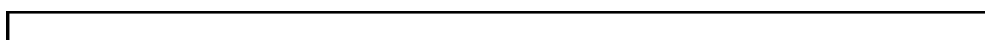
Tabulka 2 - Scénář přidání známky

| | Aktér | Systém |
|-----|---|--|
| 1 | Uživatel s rolí učitel zvolí možnost Znamky na domovské stránce | |
| 2 | | Systém zobrazí stránku s tabulkou známek ve výchozí konfiguraci |
| 3 | Uživatel ponechá rok a vybere třídu a předmět, výběr potvrdí | |
| 4 | | Systém zobrazí aktualizovanou tabulku dle zadaných požadavků |
| 5 | Uživatel klikne na "přidat" u konkrétního žáka | |
| 6 | | Systém zobrazí formulář pro zadání známky |
| 7 | Uživatel zadá všechny potřebné údaje, klikne na "uložit" | |
| 8 | | Systém uloží údaje do databáze |
| 8.1 | | Systém uživatele přesměruje na výpis známek ve výchozí konfiguraci |
| 8.2 | | Když nejsou do formuláře zadány povinné hodnoty, systém uživatele upozorní |

Jako i u předchozího scénáře, i u tohoto můžete níže vidět příklad přibližné podoby okna pro zobrazení známek a formuláře pro přidání nové známky, tak jak bylo popsáno ve scénáři. Obrázky zobrazují návrh ve formě wireframu.



Obrázek 8 - Wireframe stránky se známkami



Obrázek 9 - Wireframe formuláře pro přidání známky

3.2 Vývoj webové aplikace

Aplikaci jsem vyvíjel ve vývojovém prostředí PHPStorm od firmy JetBrains v licenci poskytnuté školou. PHPStorm jsem zvolil, neboť jsem v něm již v minulosti tvořil projekty z oblasti tvorby webu a dle mých zkušeností se jedná o velmi spolehlivé a intuitivní

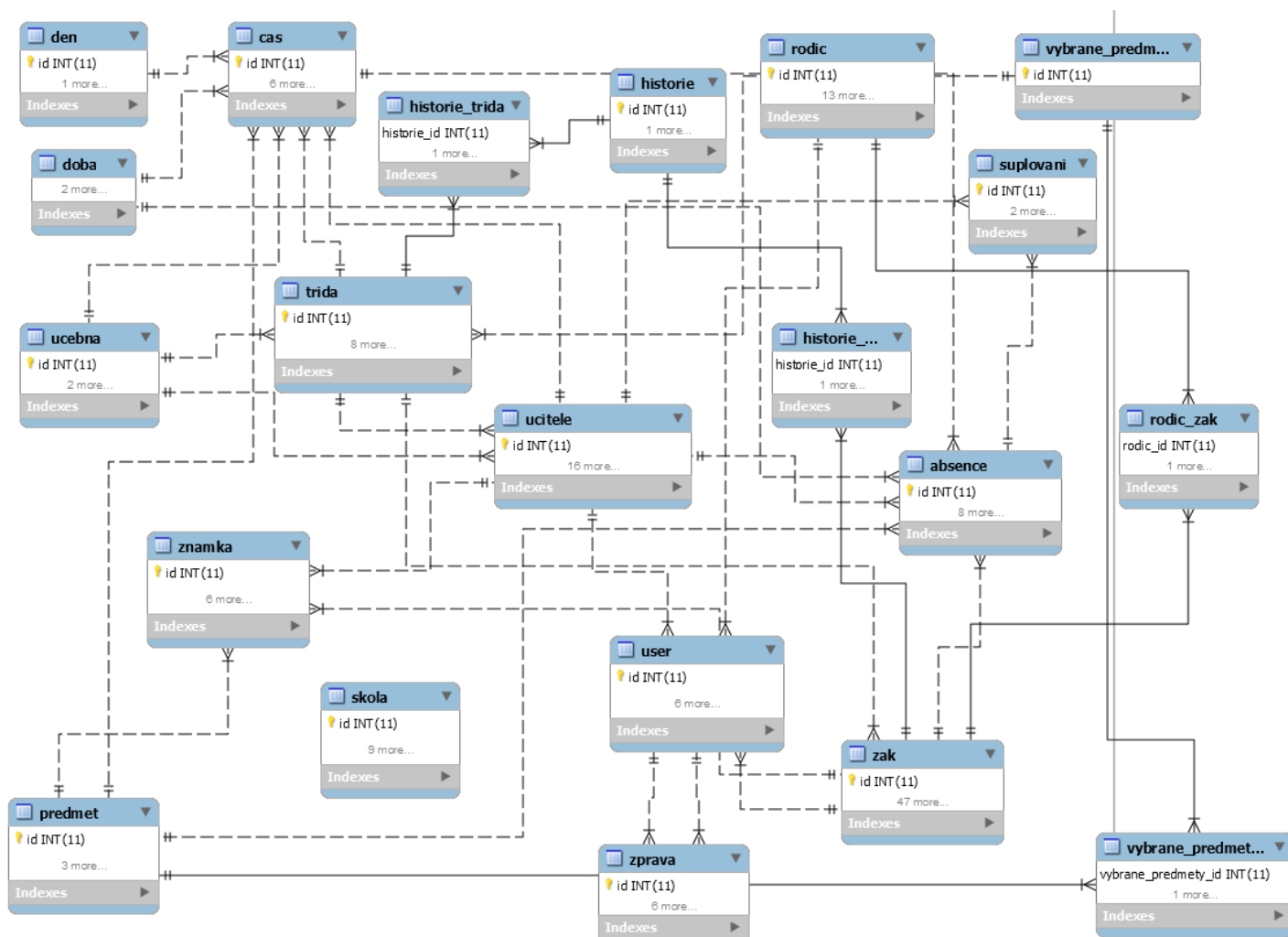
vývojové prostředí. Práci jsem prováděl ve Frameworku Symfony, který je volně dostupný. Symfony využívá programovací jazyk PHP a engine na HTML šablony Twig. Symfony jsem si vybral, protože je to jeden z nejoblíbenějších PHP frameworků a umožňuje pohodlný vývoj rozsáhlých webových aplikací, také mi byl doporučen. Samotné soubory frameworku jsem přidával do projektu pomocí nástroje PHP Composer a k testování a spouštění aplikace jsem využil volně dostupný server Apache, na ukládání dat jsem zvolil volně dostupný databázový server MySQL. Obě tyto serverové řešení zastřešoval program Laragon, taktéž volně dostupný, který zajistil, že spolupracují a jsou přenositelné a všechny soubory jsou na jednom místě.

3.2.1 Databáze

Jeden z prvních kroků, které bylo nutno udělat po vytvoření funkčního základu aplikace, bylo vytvořit funkční databázi. K tvorbě databáze jsem využil hlavně vlastnosti frameworku a PHP Composer, který mi je zprostředkoval. Pomocí Composeru jsem stáhnul nutné moduly, které chyběly ve frameworku, jako ORM Doctrine pro práci s databází. Poté jsem pomocí konzolových příkazů pro PHP tvořil PHP třídy v projektu mé aplikace. Jedná se o tak zvané Entity třídy, které obsahují veškeré vlastnosti, které poté budou odpovídat atributům entitě v databázi. Zaznamenávají se zde i primární a cizí klíče a vazby a nulovost atributů. Všechny hodnoty jsou také zapouzdřené. Díky tomuto řešení je pak možné v rámci frameworku pracovat s daty z databáze jako s objekty v rámci PHP. Tyto data potom načítá z databáze vrstva ORM Doctrine, když jsou potřeba pro funkčnost kódu. Když jsem byl spokojený se svým návrhem a stavem Entity tříd v projektu, provedl jsem tak zvanou migraci. Jedná se o příkaz pro PHP, který podle Entity tříd v aktuálním projektu provede zrcadlově vytvoření databáze se všemi potřebnými tabulkami a vlastnostmi, tak aby databáze spolehlivě fungovala s Aplikací.

Samotný návrh databáze jsem prováděl postupně, podle toho, jak se aplikace rozšiřovala. Nejprve jsem měl tabulky pro žáky a učitele, třídy a učebny. Později jsem přidal tabulku user pro přihlašovací údaje a tabulky předmět a známka spolu s tabulkami čas, den a doba pro účely ukládání rozvrhů a známek, později pak ještě tabulku pro rodiče, zprávy a absence pro účely funkcí suplování a docházky pro přidání rodičovského účtu. Strukturu databáze předvedu v následujícím obrázku pomocí ER Diagramu neboli Entitně-

relačního diagramu. Obdélníkovými rámečky jsou zobrazeny jednotlivé Entity. Čáry znázorňují vztahy mezi těmito entitami, doplněné o parcialitu a kardinalitu. Entity se seznamem atributů budou uvedeny na obrázcích níže.



Obrázek 10 - ER Diagram databáze

| | | | | |
|---|--|--|---|---|
| den id INT(11) den VARCHAR(25) Indexes | cas id INT(11) Indexes | historie id INT(11) rok SMALLINT ... Indexes | rodic id INT(11) jmeno VARCHAR(255) prijmeni VARCHAR(255) titul VARCHAR(50) ulice VARCHAR(255) obec VARCHAR(255) m_cast VARCHAR(255) stat VARCHAR(255) cis_pop VARCHAR(255) psc INT(11) telefon VARCHAR(13) email VARCHAR(50) dat_nar DATE image VARCHAR(255) Indexes | ucitele id INT(11) jmeno VARCHAR(255) image VARCHAR(100) prijmeni VARCHAR(255) dat_nar DATE stat VARCHAR(255) mesto VARCHAR(255) ulice VARCHAR(255) cis_pop VARCHAR(25) psc SMALLINT(6) poznamka LONGTEXT telefon VARCHAR(15) email VARCHAR(50) web VARCHAR(50) asistent_pedagoga TINYINT(...) Indexes |
| doba id INT(11) doba TIME Indexes | trida id INT(11) nazev VARCHAR(255) typ_tridy VARCHAR(255) tt VARCHAR(1) rocnik INT(11) obor VARCHAR(7) jaz_o VARCHAR(20) Indexes | suplovani id INT(11) Indexes | | |
| ucebna id INT(11) nazev VARCHAR(2...) patro SMALLINT(6) Indexes | | | | |
| znamka id INT(11) hodnota SMALLINT(6) poznamka VARCHAR(2...) datum DATE Indexes | absence id INT(11) pritomen TINYINT(1) datum DATE poznamka VARCHAR(2...) Indexes | skola id INT(11) nazev VARCHAR(2...) stat VARCHAR(255) mesto VARCHAR(2...) ulice VARCHAR(255) cis_pop VARCHAR... izo INT(11) red_izo VARCHAR(...) izo_spz VARCHAR(...) delka INT(11) Indexes | zprava id INT(11) text LONGTEXT precteno TINYINT(1) predmet VARCHAR(...) dat_odesl DATE Indexes | user id INT(11) username VARCHAR(180) roles LONGTEXT password VARCHAR(255) Indexes |
| predmet id INT(11) nazev VARCHAR(255) zkratka VARCHAR(2...) jazyk VARCHAR(25) Indexes | vybrane_predm... id INT(11) Indexes | | | |

Obrázek 11 - Seznam Atributů v Entitách v Databázi



Obrázek 12 - Seznam Atributů v Entitě Žák

3.2.2 Zvolené technologie

Zvolený framework využívá principu MVC, má tedy Controllery, třídy, které obsahují logiku pro odpovídání na dotazy a zobrazení příslušné šablony s již připravenými daty z databáze. Postupoval jsem tedy tak, že první krok před přidáním nové funkcionality, byl přidání nové třídy typu controller. U ní jsem si vytvořil všechny potřebné funkce, které bude onen controller a potažmo i příslušná část webové stránky schopná vykonávat. K celé třídě a k funkcím jsem potom vytvořil notace pro routing neboli směrování. tedy přidal jsem text, který spustí příslušnou funkci, pokud jej uživatel zadá do adresního řádku v prohlížeči. Do funkcí jsem přidal veškerou potřebnou logiku a jako návratovou hodnotu funkce jsem většinou použil response v podobě načtení příslušné šablony a předal jsem jí veškeré nutné parametry. Tuto šablonu jsem potom i vytvořil, tedy většinou má každá funkce controlleru i svoji šablonu. U šablon jsem využil možnosti mít jednu šablonu jako vzor, který pak mohou dědit všechny ostatní šablony a do té jsem umístil neměnné prvky grafického prostředí webové aplikace a jiná data. Je to například fixní hlavní lišta s nabídkou funkcí na horní straně obrazovky, nebo pravidla pro načítání hlavního obsahu jako typ Card z css frameworku Bootstrap, vždy uprostřed obrazovky. Pro jednotlivé šablony jsem vytvořil převážně klasický HTML kód s výjimkou míst, kam jsem vkládal data z databáze, předaná controllerem nebo míst s podmínkami či cykly. V následující části se pozastavím nad částmi kódu, které vystupují z řady a myslím, že bych měl vysvětlit, jak jsem pomocí nich dospěl ke kýženému výsledku a jak jsem aplikaci vytvořil z hlediska programování.

3.2.3 Implementace

V Aplikaci je 17 Entit, každá z nich odpovídá příslušné tabulce v databázi. Tyto třídy nejsou zase až tak zajímavé, v podstatě se jedná pouze o kolekci vlastností a o notaci pro ORM Doctrine spolu s funkcemi get a set pro každou vlastnost neboli zapouzdření. V těchto metodách jsem z klasického šablonového vzhledu nic neměnil, pouze jsem doplnil funkci toString, kde to bylo nutné, tedy schopnost převést objekt na textovou hodnotu. Nebo jsem přidal jednoduchou funkci na vracení specifičtější kolekce vlastností. Krom toho má každá Entity třída ještě zrcadlově Repository třídu, což je vlastně podobný případ, jedná se o automaticky vygenerované třídy pro práci s databází. Liší se od Entit tím, že

místo toho, aby jako objekt uchovávaly informace o jednom konkrétním záznamu z databáze, Repository je kolekce, která uchovává informace o všech záznamech příslušné tabulky. Protože tyto typy tříd, nejsou tolik zásadní, co se týče samotného programového chodu aplikace a programového řešení, nebudu se zde jimi zabývat do hloubky.

Nadále se v projektu aplikace nachází 12 controller tříd a 33 HTML Twig šablon. Je tomu proto, že zpravidla každá funkce v controlleru má svou šablonu, kterou po provedení logiky zobrazí. Hlavní části aplikace, tedy zobrazení žáků, učitelů a známek, mají každá 4 šablony a suplování a absence jsou šablony z jednoho controlleru a mají jich dohromady 6, zobrazení rozvrhů hodin a zprávy mají pak oba 3. Stránky pro zobrazení registračního a přihlašovacího formuláře a domovská obrazovka mají každá po jedné šabloně, plus je tu ještě šablona base, která uchovává všechny obecná a základní data, která pak dědí všechny ostatní šablony, aby nebylo nutné je zadávat vícekrát.

3.2.4 Hlavní Obrazovka

Mám-li vzít popis a prohlídku aplikace postupně, řekněme, že začneme u hlavní obrazovky. Je to stránka, která se zobrazí, když uživatel přejde na stránku webové aplikace ve svém prohlížeči. Tato stránka se zobrazí, i pokud uživatel není přihlášen. Domovská stránka totiž přizpůsobuje svůj obsah, podle role uživatele, který na ní vstoupí. V tomto případě, pokud uživatel není přihlášen, zobrazí se na stránce pouze nadpis školního systému a tlačítko pro přihlášení. Každá role pak vidí na domovské stránce trochu jiný obsah. Administrátor zde vidí v hlavní části tabulku se všemi uživatelskými jmény a může je zde upravovat, žák zde vidí informace o sobě a informace o příští vyučovací hodině, zatímco učitel vidí také informace o sobě a informace o hodině, kterou bude vyučovat příště, rodič potom vidí své nepřečtené zprávy a všechny jeho děti. Toto zobrazování rozdílného obsahu je možná díky podmíněnému příkazu v šablonách twig a identifikaci role uživatele. V obrázku níže předvedu konkrétní ukázkou z kódu.

```

{% elseif is_granted('ROLE_TEACHER') %}

<h1>Používáte systém jako Vyučující</h1>
<p>Jako vyučující můžete spravovat žáky a přidávat jim známky nebo tvořit rozvrhy</p>

<hr>

<div class="d-flex justify-content-center">
    <ul class="list-group list-group shadow-sm" style="...">
        <li class="list-group-item">Jste přihlášen jako: {{ app.user.ucitel.jmeno }}
            {{ app.user.ucitel.prijmeni }}</li>

        {% if app.user.ucitel.trida == null %}

            <li class="list-group-item">Nejste Třídním učitelem</li>

        {% else %}

            <li class="list-group-item">Třídním učitelem pro: {{ app.user.ucitel.trida }} </li>

        {% endif %}

        <li class="list-group-item">S Kabinetem: {{ app.user.ucitel.ucebna }}</li>

        <li class="list-group-item">Počet nových zpráv : {{ zpravy }}</li>

    </ul>

{% if rozvrh == null %}

    <h5 style="...">Příští hodina už není, vyučování skončilo</h5>

{% else %}

    <ul class="list-group list-group shadow-sm" style="...">
        <li class="list-group-item">Příští hodina je {{ rozvrh.predmet.nazev }}</li>
        <li class="list-group-item">V učebně {{ rozvrh.ucebna }}</li>
        <li class="list-group-item">Se třídou {{ rozvrh.trida }}</li>
    </ul>

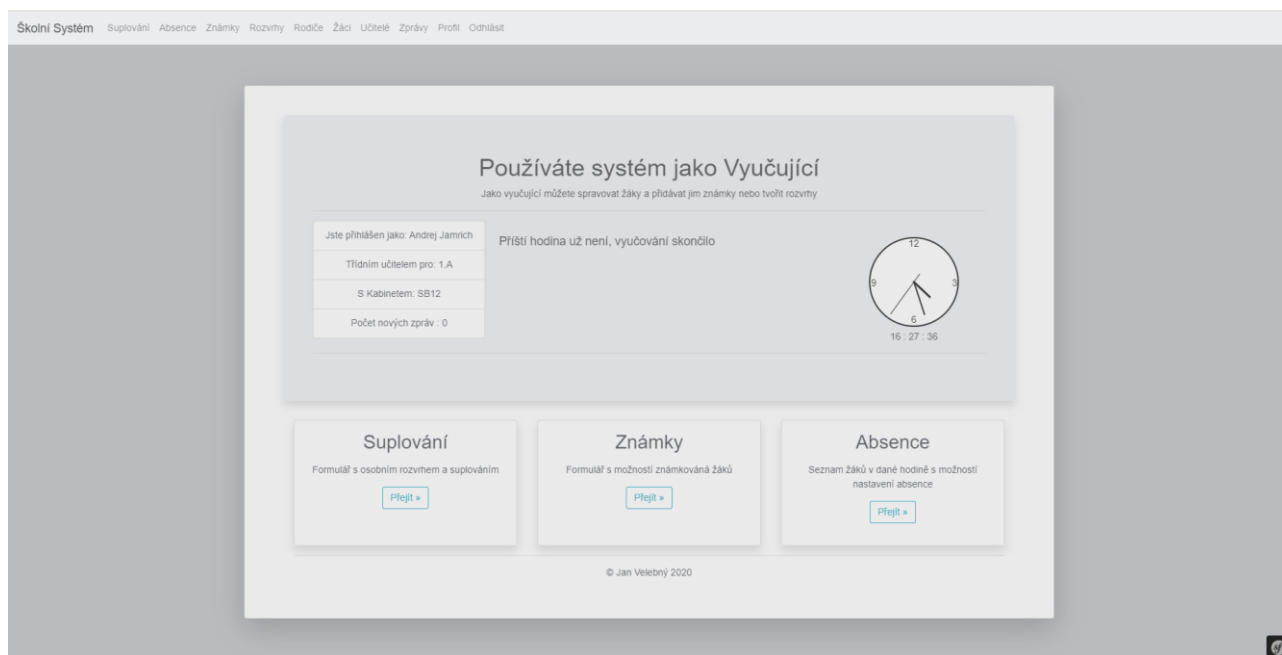
{% endif %}

```

Obrázek 13 - Příklad kódu v šabloně Twig pro rozlišení rolí

Z obrázku je patrné, jak velké množství možností nám engine pro šablony Twig dává. Na začátku příkladu je příkaz `elseif`, který kontroluje, zda je uživatel učitel a pokud ano, provede kód pod ním, tedy nadpis, nějaké informace a jeho jméno, které získá z příkazu `app.user`, což je příkaz pro zjištění uživatele pro Twig. User potom má informace o objektu učitele, s kterým je spojen a `jmeno` a `prijmeni` je pouze vlastnost onoho učitele. V další podmínce se kontroluje, jestli spojení objektu učitel s objektem třída je `null` nebo není, tedy zda konkrétní učitel, který je přihlášený má v databázi přidělenou nějakou třídu,

tedy, zda je třídním učitelem nějaké třídy. Pokud je, tato třída se vypíše, pokud není, vypíše se, že není třídním učitelem. Další podmínka pak kontroluje, podle hodnot, které nám dodal controller podle aktuálního času, jestli právě probíhá nějaká vyučovací hodina v rozvrhu, které by měl být konkrétní uživatel součástí. Pokud není, vypíše se, že vyučování skončilo, pokud je, uvedou se details o této hodině. Níže umístím screenshot toho, jak taková domovská obrazovka vypadá v hotové praxi právě pro přihlášeného učitele.



Obrázek 14 - Ukázka hotové domácí obrazovky

Aby však twig mohl stránku přivést do tohoto grafického stavu, musí mu controller dodat všechna potřebná data, podle své vnitřní logiky. Proto na následujícím obrázku ještě přiložím obrázek hlavní funkce pro controller pro domácí obrazovku. Je v něm dobře vidět základní práce s načítáním dat z databáze. Na obrázku si můžeme všimnout, že jsem v rámci lepší čitelnosti nepřidal notaci pro routing. Také zde používám funkci `findTime`, kterou jsem implementoval v kontroleru také, ale jediné, co dělá je, že projde databází hodin v rozvrhu pro nějakého uživatele a porovná ji s aktuálním časem a vrátí konkrétní hodinu, která právě probíhá. Avšak abych popsal kód, funkce má repository všech uživatelů, příkaz `findAll` je právě všechny načte. Příkaz `this -> getUser` slouží v PHP logice frameworku pro zjištění právě přihlášeného uživatele. Takže kontroluji v podmínce, zdali není null a poté si zjistím jeho uživatelské jméno. Poté projdu všechny žáky v databázi a pro každého z nich zjišťuji, jestli se uživatelské jméno přihlášeného uživatele shoduje s tím jeho. Pokud ano a

je tedy přihlášeným uživatelem, vložíím do proměnné rozvrh, rozvrh pro jeho třídu. Analogicky to samé provádím i s učitelem, ale u něj, pokud se uživatelské jméno shoduje, ještě musím projít databází všech rozvrhů, protože učitel nemá informaci o hodinách, ve kterých učí, to hodiny mají informaci o učitelích. Poté, po kontrolních podmínkách, projdu všechny hodiny v nalezeném rozvrhu a vrátím tu, která bude probíhat příště, podle aktuálního času. Další příkaz slouží k nalezení nepřečtených zpráv uživatele, zjistím tedy příkazem getUser o jakého uživatele jde, nahraji si do proměnné všechny zprávy z databáze a poté je všechny projdu a když se příjemce zprávy shoduje s uživatelem a zpráva není přečtená, přičtu do počítací proměnné, kterou potom vrátím spolu s dalšími údaji do zobrazené šablony příkazem render.

```
public function index(UserRepository $userRepository, ZakRepository $zakRepository,
    UcitelRepository $ucitelRepository, CasRepository $casRepository, ZpravaRepository $zpravaRepository)
{
    $users = $userRepository->findAll();
    $result = null;
    $checkU = false;
    $vysledek = 0;
    if($this -> getUser() != null) {
        $user = $this->getUser()->getUsername();
        $rozvrh = null;
        foreach ($zakRepository -> findAll() as $z) {
            if ($z->getUser() != null && $z->getUser()->getUsername() == $user) {
                $rozvrh = $z->getTrida()->getCas();
            }
        }
        foreach ($ucitelRepository -> findAll() as $u) {
            if ($u->getUser() != null && $u->getUser()->getUsername() == $user) {
                foreach ($casRepository -> findAll() as $t)
                {
                    if($t -> getUcitel() == $u)
                    {
                        $checkU = true;
                        $rozvrh = $t;
                    }
                }
            }
        }
        if($rozvrh != null)
        {
            if($checkU) $result = $this -> findTime($rozvrh);
            foreach ($rozvrh as $r)
            {
                $r -> getId();
                $result = $this -> findTime($r);
            }
        }
        $us = $this -> getUser();
        $zpravy = $zpravaRepository -> findAll();
        foreach ($zpravy as $zprava)
        {
            if($zprava -> getPrijemce() -> getId() == $us -> getId() && $zprava -> getPrecteno() == false)
            {
                $vysledek++;
            }
        }
    }
    return $this->render( view: 'home/index', ['users' => $users, 'rozvrh' => $result, 'zpravy' => $vysledek]);
}
```

Obrázek 15 - Ukázka controlleru pro domovskou obrazovku

3.2.5 Zobrazení známek

Co se týče hlavních funkcionalit v aplikaci, většina z nich funguje na dosti obdobném principu, jen s lehčími obměnami. Téměř vždy se jedná o zobrazení jakési tabulky, na jejíž řádky, reprezentující jednotlivé položky, lze klikat pro zobrazení detailní stránky o vybrané položce. Tam je pak možnost ji nějak upravit nebo mazat. Také většinou krom funkce zobrazení takřka vždy existuje možnost přidání nové položky. Občas je do toho přidáno filtrování podle kritérií ve vysouvacích nabídkách, ale to je většinou řešeno jenom jako formulářové znovunačtení stránky s aktualizovanými parametry. Díky těmto společným rysům však většina controllerů v aplikaci obsahuje většinou funkce: index pro zobrazení, show pro načtení detailu, remove pro smazání, create pro načtení formuláře pro vytvoření nového záznamu a edit pro úpravu. Využívá se pak formulář pro vytvoření nového záznamu, jen se mu předá již vytvořený objekt místo nového. Informace se pak zobrazí korektně a nedojde k založení duplikátu ale k aktualizaci. Díky těmto společným vlastnostem je celkem zbytečné procházet všechny controllery a funkce webové aplikace a dopodrobna je popisovat. Dovolil bych si demonstrovat obecný princip na té složitější z nich, a to je controller pro známky. Je to i vhodný příklad, neboť zrovna tato funkce byla zmíněna i výše ve scénáři užití. Řekl bych, že nejvhodnější na ukázkou z tohoto controlleru bude funkce pro načtení známek, protože je asi nejkomplexnější v porovnání s ostatními. Funkce remove a edit, jsou celkem jednoduché, a ne až tolik zajímavé. Na následujícím obrázku tedy ukáži funkci index pro načtení všech známek v controlleru pro zobrazení známek.

```

* @Route("/{tid}/{pid}", name="index")
* @ParamConverter("predmet", options={"id" = "pid"})
* @ParamConverter("trida", options={"id" = "tid"})
* @param Trida $trida
* @param Predmet $predmet
* @param ZakRepository $zakRepository
* @param ZnamkaRepository $znamkaRepository
* @param Request $request
* @return Response
*/
public function index(Trida $trida, Predmet $predmet, ZakRepository $zakRepository,
                    ZnamkaRepository $znamkaRepository, Request $request)
{
    $user = $this -> getUser();
    if($user -> getRoles()[0] != 'ROLE_TEACHER' && $user -> getRoles()[0] != 'ROLE_ADMIN')
        throw $this -> createAccessDeniedException("Přístup odepřen");

    $zak = $zakRepository -> findAll();
    $znamka = $znamkaRepository -> findAll();
    $form = $this -> createFormBuilder()
        -> add( child: 'trida', type: EntityType::class,
            ['class' => Trida::class, 'label' => 'Známky pro třídu:', 'required' => false])
        -> add( child: 'predmet', type: EntityType::class,
            ['class' => Predmet::class, 'label' => 'Známky z předmětu:', 'required' => false])
        -> add( child: 'rok', type: ChoiceType::class,
            ['required' => true, 'choices' => $this->roky()])
        -> add( child: 'vyber', type: SubmitType::class,
            ['label' => 'Zobrazit známky pro: ', 'attr' => ['class' => 'btn btn-outline-primary btn-group']])
        -> getForm();

    $form->handleRequest($request);

    if($form -> isSubmitted())
    {
        return $this -> render( view: 'grades/index', ['zak' => $zak, 'znamka' => $znamka,
            'trida' => $form -> getData()['trida'], 'predmet' => $form -> getData()['predmet'],
            'rok' => $form->getData()['rok'], 'form' => $form->createView()]);
    }
    return $this -> render( view: 'grades/index', ['zak' => $zak, 'znamka' => $znamka,
        'trida' => $trida, 'predmet' => $predmet, 'rok' => date( format: "Y"),
        'form' => $form->createView()]);
}

```

Obrázek 16 - Ukázka funkce index controlleru pro známky

Na začátku kódu je patrná notace pro Routing, která udává pro framework počet parametrů a návratový typ a v neposlední řadě URL, jež je nutné zadat, aby byla stránka zobrazena. Závorky {} v url značí, že se zde očekává jakási hodnota, která bude doplněna. Konkrétně id třídy a předmětu, pro které se mají známky načíst. Protože jsou tyto hodnoty dvě, ORM Doctrine neví, která z nich je která a v jaké databázi má hledat, proto je nutné použít paramconverter, kde specifikujeme, o jakou entitu se jedná. Poté ve funkci samotné, první podmínka kontroluje, zda je uživatel učitel nebo administrátor a pokud není, vyhodí chybu odmítnutí přístupu. Dále probíhá načítání všech žáků a známek z databáze a tvorba nového formuláře se třemi vysouvacími nabídkami, které načítají své možnosti z databáze z

tabulek třída a předmět a jedním tlačítkem na potvrzení. Je k tomu využita funkce `createFormBuilder`, která tvoří formulář přímo v controlleru a je celkem užitečná pro menší množství ovládacích prvků, jako je tomu zde. Pro větší formuláře je výhodnější využít oddělenou třídu a v controlleru používat jenom její instanci. Potom, co je formulář vyplněn a odeslán tlačítkem, se jednoduše vrátí šablona a odešlou se jí všechny potřebné parametry. To samé se udělá i pokud form nebyl odeslán, tedy většinou pokud se jedná o první načtení nebo obnovení stránky. Rozdíl je jenom v datech, rok se liší a některé hodnoty mohou být null. Také si čtenář může všimnout, že využívám funkci `roky`. Je to pouze jednoduchá funkce, která vrací rozsah 10 let před aktuálním rokem.

V šabloně pro zobrazení známky se potom vykreslí jednotlivé prvky formuláře pomocí php příkazů v systému Twig a je i možné je kombinovat s ostatními HTML prvky, takže lze vytvořit ničím nerušené uživatelské prostředí. To nejsložitější ovšem v této šabloně je vytvořit dynamickou tabulku pro zobrazení známek, neboť nikdy není dopředu jisté, kolik žáků bude nutné zobrazit. A kolik bude v jednom řádku známek a tudíž, jak má být celá tabulka dlouhá. A také průměr bude pro každý řádek rozdílný. Řešení je využití cyklů, které Twig nabízí. V následujícím obrázku demonstruji na praktické ukázce, jak vypadají v praxi.

```
<div class="shadow" style="...">
  <table id="table" data-toggle="table" data-height="500"
    class="table table-bordered table-borderless">
    <thead class="thead-light">
      <tr>
        <th data-field="Jméno žáka">Jméno žáka</th>

        {% set Max = 0 %}

        {% for z in zak %}
          {% set cnt = 0 %}
          {% for zn in z.znamka %}
            {% if zn.predmet == predmet and z.trida == trida and zn.datum|date("Y") == rok %}
              {% set cnt = cnt + 1 %}
            {% endif %}

            {% if predmet == null and trida == null and zn.datum|date("Y") == rok %}
              {% set cnt = cnt + 1 %}
            {% endif %}

            {% if predmet == null and trida == z.trida and zn.datum|date("Y") == rok %}
              {% set cnt = cnt + 1 %}
            {% endif %}

            {% if predmet == zn.predmet and trida == null and zn.datum|date("Y") == rok %}
              {% set cnt = cnt + 1 %}
            {% endif %}

          {% endfor %}

          {% if cnt > Max %}
            {% set Max = cnt %}
          {% endif %}
          {% set cnt = 0 %}
        {% endfor %}

        {% for i in 0..Max - 1 %}
          <th> {{ i + 1 }} </th>

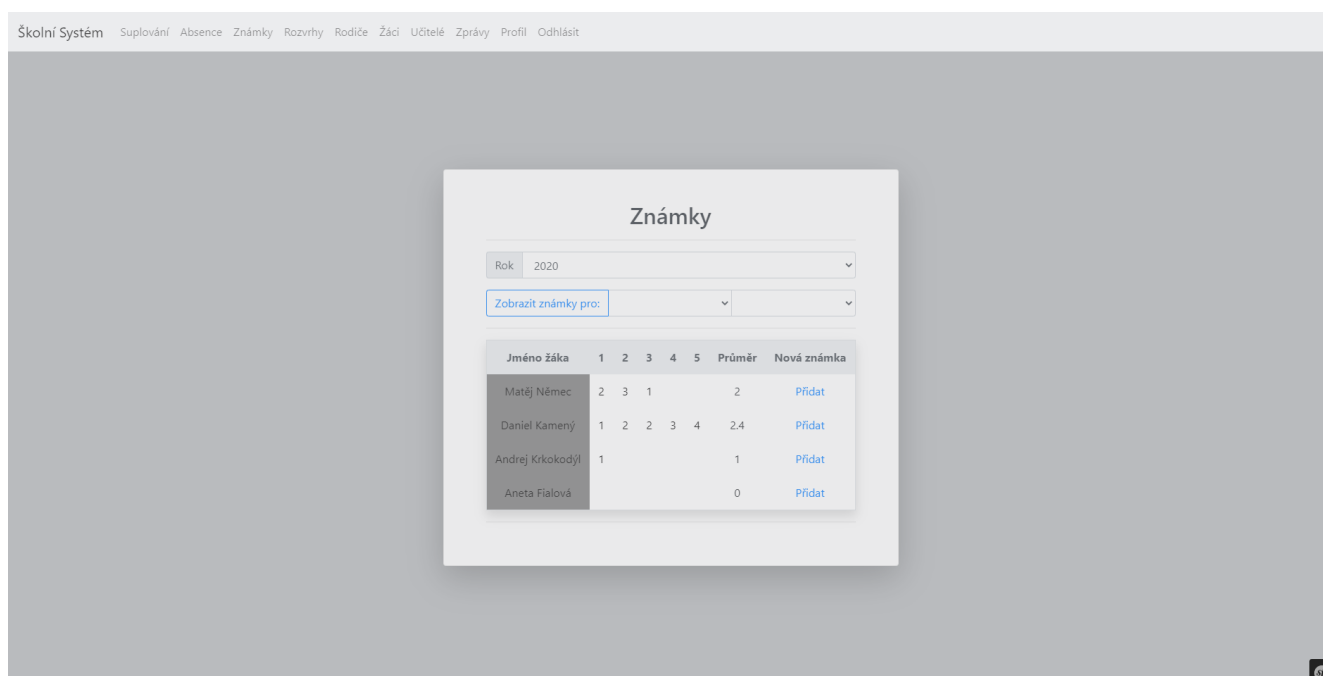
          {% endfor %}
        </tr>
      </thead>
    </table>
  </div>
```

Obrázek 17 - Ukázka cyklů v šabloně Twig pro výpis známek

Na obrázku jest vyobrazena tabulka využívající mimo jiné třídy kaskádových stylů frameworku Bootstrap. Ukázka je pouze část kódu tabulky pro tvorbu záhlaví. Jednotlivé řádky v záhlaví jsou tvořeny čísly od jedné do maximálního počtu známek v jednom řádku a algoritmus na obrázku zjišťuje přesně to. Prochází se databáze žáků a pro každého z nich se ještě prochází databáze všech jeho známek. Pro každou známku se pak kontroluje, jestli odpovídá nějaké z možných variant omezení, které mohl uživatel zadat formulářem. Tedy jestli zvolil rok nebo předmět nebo třídu, pro které chce hodnoty zobrazit. Pokud nějakou

hodnotu nezvolil, bude právě ta, null. Na začátku cyklu se nastavuje proměnná cnt, do které se přičte vždy hodnota, pokud jsou vybraná kritéria splněna. Toto se provádí pro každou známku. Když se takto projdou všechny známky jednoho žáka, kontroluje se, jestli počet jeho známek, které odpovídaly kritériu, překročil dosavadní proměnou Max pro maximum. Pokud ano, nastaví se hodnota v proměnné cnt jako nové maximum a takto se dá efektivně zjistit nutný počet sloupců pro vypsání tabulky. Poté již jenom proběhne cyklus, který se opakuje od nuly do maxima - 1 a vypisují se hodnoty maxima po jedné. Každá iterace cyklu znamená jednu novou položku HTML značky <th>.

No a aby ilustrace byla kompletní, na následujícím obrázku ukáži reálný vzhled webové aplikace, tak, jak vypadá v prohlížeči z pohledu učitele nebo administrátora.



Obrázek 18 - Ukázka hotové stránky pro výpis známek

3.2.6 Zabezpečení a dodatečné informace

Tyto příklady zastřešují principy té nejdůležitější funkcionality. Ještě bych však chtěl ukázat několik menších příkladů a dovysvětlit některé věci ohledně dodatečné funkcionality. Jedna z věcí, kterou bych chtěl ještě demonstrovat, je ukládání dat do databáze přes controller, neboť prozatím se z ní ve všech příkladech jenom četlo. Rozhodl jsem se k tomu využít controller pro registraci nových uživatelů. Na následujícím obrázku tedy předvedu část kódu, kde se data ukládají do databáze.

```

if ($form -> isSubmitted()) {
    $data = $form->getData();

    $user = new User();
    $user -> setUsername($data['username']);
    $user -> setPassword($passwordEncoder -> encodePassword($user, $data['password']));
    $user -> setRoles($data['roles']);
    $user -> setUcitel($data['ucitel']);
    $user -> setZak($data['zak']);
    $user -> setRodic($data['rodic']);

    $em = $this -> getDoctrine() -> getManager();

    $em -> persist($user);
    $em -> flush();

    return $this -> redirect($this -> generateUrl( route: '/' ));
}

return $this -> render( view: 'registration/index', ['form' => $form -> createView()] );

```

Obrázek 19 - Ukázka ukládání do databáze z controlleru pro registraci

Na příkladu můžeme vidět, že poté, co je formulář odeslán se z něj do proměnné uloží data a vytvoří se nový objekt uživatele. Tomu se pomocí funkcí zapouzdření, tedy set nastaví potřebné vlastnosti podle dat, získaných z formuláře. Potom pouze dojde k založení nového entity managera části frameworku ORM Doctrine, která manipuluje s databází. Tento manažer je potom využit pro uložení nového uživatele do databáze pomocí příkazů persist a flush. Nakonec funkce přesměruje uživatele na domovskou obrazovku. Druhý return se vykoná při prvním načtení, a tedy načítá HTML šablonu pro registraci uživatelů.

Příklad o registraci je vlastně celkem příhodný, neboť mi umožňuje plynule přejít k bezpečnosti. Aplikace má být pro základní školy a je tedy nezbytné, aby měl systém přihlašování a dostatečné zabezpečení. V aplikaci je k tomuto účelu využit Security modul frameworku Symfony, právě pro bezpečnost. Funkce zjišťování uživatele aplikace, které se objevovaly v kódu v předchozích příkladech, jsou právě součástí tohoto modulu. V příkladu výše si čtenář také mohl povšimnout, že k ukládání hesla je použita funkce encodePassword, která heslo zašifruje. Do databáze se tedy neukládá přímo heslo, ale pouze jeho zašifrovaná podoba. Framework pro funkce přihlašování a autentizaci využívá třídu MainAuthenticator z modulu Security. Ta právě obsahuje funkce pro přihlašování, umí obsluhovat csrf token, ukládá cestu pro přihlášení nebo odhlášení a umí kontrolovat správnost údajů.

Další vrstva zabezpečení je omezení přístupu pro jednotlivé role. V aplikaci je realizována na třech stupních. V enginu Twig pro HTML šablony je ošetřeno, aby každý uživatel viděl jenom ty prvky, které může využívat a nemohl se dostat nikam, kam nemá přístup pomocí funkce `isGranted`, která byla součástí příkladů výše. Další stupeň zabezpečení je konfigurační soubor modulu pro zabezpečení `security.yaml`, který obsahuje základní nastavení bezpečnosti. Například udržuje informaci o cestách pro přihlášení a odhlášení anebo umožňuje nastavení typu šifrování pro ukládání hesel. V aplikaci je zde také definována hierarchie uživatelských rolí. Krom toho však nabízí volbu `access_control`, tedy omezení přístupu, kam se mohou přidávat jednotlivé url cesty, ve formě regulérního výrazu, a role, které je mohou využít. Aplikace pak nedovolí uživateli přejít na stránku, pro kterou nemá dostatečná práva. Třetí způsob je pak kontrolovat roli právě přihlášeného uživatele manuálně, pomocí funkce `symfony` přímo v controlleru a poté zamítnout přístup vyhozením chyby o odmítnutí přístupu. V aplikaci je tak učiněno v každé funkci v každém controlleru, který má omezený přístup pro nějaké uživatele. V následujícím příkladu ještě předvedu ukázkou kódu z konfiguračního souboru `security.yaml`.

access_control:

- { **path:** (/|/login)\$, **roles:** IS_AUTHENTICATED_ANONYMOUSLY }
- { **path:** ^/, **roles:** ROLE_USER }

Obrázek 20 - Ukázka omezení přístupu ze `security.yaml`

V aplikaci je pomocí tohoto souboru dosaženo dvou podmínek. Určují, že pokud uživatel není přihlášený, jediné místo, kam mu aplikace dovolí jít, je domácí obrazovka (která obsahuje jenom název systému a tlačítko přihlásit pro nepřihlášené uživatele) a stránka pro přihlášení. Druhá podmínka pak stanovuje, že jakýkoliv přihlášený uživatel má přístup na všechny stránky. Je to tedy jednoduché rozlišení přihlášených a nepřihlášených uživatelů, a je to využito jako nejhrubší síto v hierarchii zabezpečení.

Výsledky a diskuse

3.3 Testování

Aplikace byla spouštěna a testována autorem v průběhu vývoje a jednotlivé nedostatky a chyby byly odstraněny procházením jednotlivých funkcí a kódu. Autor stav vývoje konzultoval se školním IT expertem, což vedlo ke zdokonalení funkcí a k lepšímu návrhu rozšíření aplikace. Díky tomu byly do aplikace přidány funkce suplování a docházky a došlo k rozšíření množství důležitých dat v databázi. Také bylo díky tomu možné implementovat funkci generování XML souboru pro ministerstvo školství ze školní matriky. Během testování docházelo k zjištění, že databáze neodpovídá požadované funkcionalitě a byla do ní vícekrát přidána funkcionalita. Následně docházelo k chybám grafického charakteru. Tabulky pro načítání známek se nezobrazovaly správně, počet polí byl příliš malý. Rozvrhy měly tendence se zobrazovat několikrát vedle sebe, bylo zjištěno, že položky v rozvrhu mají tendenci se řadit hned za sebou, i když mají mít časový odstup. S odchylováním chyb pomáhal ladící nástroj frameworku Symfony, spouštěný přímo v prohlížeči. Další nedostatek, byl bezpečnostního charakteru, tedy, že uživatelé sice nemohli přejít na stránky, na které neměli oprávnění pomocí grafických navigačních prvků, avšak pokud kupříkladu správně pozměnili URL, bylo to možné. Všechny zásadní nedostatky byly však odstraněny, chyby v grafické reprezentaci byly opraveny opravami v příslušném kódu a bezpečnostní nedostatek byl opraven přidáním kontroly rolí do všech funkcí controllerů.

Ve finální fázi vývoje byla aplikace představena několika učitelům k otestování. Většina z nich souhlasila s tím, že aplikace je funkční a svižná a na základní administrativní funkce na škole dostačuje. Hodně uživatelů mělo ale trochu problém zvykat si na práci v novém grafickém prostředí, které mnozí označili za neintuitivní. Taktéž si mnoho z nich všimlo absence tlačítka zpět, které by se tedy hodilo přidat. Vesměs by se testování dalo zhodnotit jako, že se aplikace učitelům líbila a chválili, že je svižná, někteří také chválili grafický vzhled. Našlo se ale hodně připomínek na zlepšení rozložení stránky, přehlednosti nebo na intuitivnost ovládacích prvků. Některým uživatelům také chyběly některé pokročilé funkce jako například vybírání data z kalendáře nebo možnost zadávání známky více žákům najednou či absence váženého průměru ve známkách. Testování na studentech bohužel nebylo možné kvůli epidemiologické situaci v době psaní této práce.

3.4 Přínosy aplikace

Webová aplikace funguje vcelku dobře a pokud by se zavedla na menší základní škole, která stále používá pouze staré dobré a papírové třídní knihy, mohla by výrazně vylepšit procesy na škole a zlepšit efektivitu výuky. A to jak pro studenty, tak pro učitele. Umožňuje zasílání zpráv, vyhledávání informací o učitelích i žácích, správu známek, rozvrhů, docházky a suplování. Jedná se o webovou aplikaci, takže tyto informace jsou dostupné odkudkoliv, kde je internet a zařízení, které k němu má přístup.

Mnoho základních školách ale již nějaký informační systém používá. Nejčastější systém na velkých školách je systém Bakaláři. Jedná se o robustní systém s mnoha funkcemi, kterému se dá jen těžko konkurovat. Další takový systém je například Škola Online, který podobně jako aplikace z této bakalářské práce je pouze webový. Autor práce provedl jako součást návrhu aplikace několik rozhovorů s učiteli a zástupci ředitele na několika základních školách a jednou z otázek byly zkušenosti na systém Bakaláři. Některé názory se trochu rozcházejí a bylo vidět, že je rozdíl mezi tím, jestli aplikaci někdo používá jako administrátor anebo jako pouhý uživatel. Uživatelé si stěžovali, že aplikace je občas pomalá a má technické problémy. V zásadě všichni chválí funkcionalitu aplikace, ale všichni se shodují s tím, že je graficky velmi nepřehledná. Funkce jsou v rozhraní schované a mají nevýstižné názvy. V aplikaci chybí jakákoliv nápověda. Také je zde až mnoho modulů a jsou od sebe oddělené, neprovázené. Navíc systém Bakaláři existuje ve verzi pro počítač a pro web a pro telefon a každá z nich obsahuje trochu jiné funkce. I přesto, že aplikace vyvinutá autorem je jenom studentský projekt a nemá ani zdaleka takový počet funkcí, má oproti systému Bakaláři taky nějaké výhody.

Aplikace je to minimalistická a velmi přehledná, není příliš náročná, takže funguje svižně, všechny funkce jsou na jednom místě. Je to webová aplikace, takže nabízí na všech zařízeních stejný počet funkcí. Při nasazení by si také mohla aplikaci spravovat škola sama, nemusela by tedy spoléhat na externí firmu a nemusela by platit neustálé poplatky za provoz.

3.5 Budoucí Vývoj

Jak bylo zjištěno z testování, aplikace stále trpí lehčími nedostatky, které by bylo dobré odstranit pro lepší uživatelský komfort. V současné podobě aplikace potřebuje v podstatě hlavně kosmetické úpravy. Je nutné vylepšit chování některých tlačítek a rozložení ovládacích prvků nebo podání toho, jak pracují některé funkce. Samozřejmě už jenom z pohledu na složitější systémy je jasné, že se dá rozšiřovat paleta funkcí. Ať už se jedná jen o možnost vážených průměrů při zadávání známek, nebo zadávání více známek najednou anebo úplně nové funkční moduly jako je například archiv dokumentů nebo plánování akcí či rozšíření systému do e-learningové platformy a nabídnout možnost zadávání testů a domácích úkolů, tyto funkce by se hodily například za epidemiologického stavu v době psaní bakalářské práce, kdy probíhá výuka z domova.

Závěr

V rámci bakalářské práce jsem vyvinul webovou aplikaci pro základní školy. Předmětem této práce bylo vyvinout webovou aplikaci pro podporu procesů pro základní školy. Hlavní zvolená technologie byl programovací jazyk PHP s frameworkem Symfony. Cílem práce bylo tuto aplikaci navrhnout a implementovat.

V první části práce je rozebrána metodika vývoje webových aplikací a základy softwarového návrhu, tedy princip životního cyklu vývoje software a jednotlivých modelů pro vývoj. Dále jsou zde popsány veškeré teoretické poznatky a technologie, nutné pro pochopení praktické části práce, které při ní byly také použity. Praktická část práce pojednává o návrhu aplikace, tvorbě prvotní analýzy a funkčních požadavků, jež vycházely z povahy prostředí, v němž má být aplikace nasazena, tedy základní školy. Dále jsou v této části podrobně popsány případy užití, které rozdělují aplikaci na tři druhy typů přístupu podle uživatelských rolí. V praktické části práce se pak hovoří o samotné implementaci webové aplikace, její databázi, programové logice a je zde dopodrobna rozebrán princip fungování nejdůležitějších aspektů aplikace spolu s ukázkami kódu, tedy princip MVC a implementace systému pro odesílání zpráv, zobrazení a evidenci známek, rozvrhů, žáků, docházky, suplování, učitelů a rodičů.

Práce byla pomocí metodik, popsaných v teoretické části, vytvořena a otestována na několika učitelích a IT expertech ze základní školy. Při testování bylo nalezeno mnoho nedostatků, všechny byly ale odstraněny, jediné, které přetrvávají jsou převážně kosmetické nedostatky a nutnost zlepšení uživatelského komfortu a některých funkcí. Z toho plyne jedna část vývoje do budoucna, tedy oprava těchto chyb. Nadále je pro aplikaci možné prohloubit existující funkcionalitu a přidat nové funkce jako archiv dokumentů nebo plánování akcí či posílání úkolů a testů v systému.

I přes menší nedostatky aplikace však můžu bezpečně říci, že funguje svižně a spolehlivě a může být přínosem pro základní školy. Cíle práce tedy byly naplněny.

4 Seznam použitých zdrojů

1. ADOBE INC. *Informace o webových aplikacích* [online]. 2017 [cit. 2020-03-19]. Dostupné z: <https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html>
2. CHRISTENSSON, Per. *Web Application* [online]. 2014 [cit. 2020-03-19]. Dostupné z: https://techterms.com/definition/web_application
3. MARTINŮ, Jiří; ČERMÁK, Petr. *Metodiky vývoje software, studijní opora pro kombinované* [online]. Moravská vysoká škola Olomouc, Olomouc: Moravská vysoká škola Olomouc, o.p.s., 2018 [cit. 2020-03-20]. Dostupné z: <https://mvso.cz/wp-content/uploads/2018/02/Metodiky-vývoje-software-studijní-text.pdf>
4. ŠMÍD, Vladimír. *Životní cyklus informačního systému* [online]. 2002 [cit. 2020-03-20]. Dostupné z: <https://www.fi.muni.cz/~smid/mis-ziveyk.htm>
5. SMARTDRAW, LLC. *UML Diagram – Everything You Need to Know About UML Diagrams* [online]. c1994-2020 [cit. 2020-03-20]. Dostupné z: <https://www.smartdraw.com/uml-diagram/>
6. ČÁPKA, David. *Lekce 2 - UML – Use Case Diagram* [online]. 2018 [cit. 2020-03-20]. Dostupné z: <https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram/>
7. PAVLÍK, Lukáš. *Informační systémy, studijní opora pro kombinované* [online]. Moravská vysoká škola Olomouc, Olomouc: Moravská vysoká škola Olomouc, o.p.s., 2018 [cit. 2020-03-20]. Dostupné z: <https://mvso.cz/wp-content/uploads/2018/02/Informační-systémy-studijní-text.pdf>
8. MOZILLA FOUNDATION. *HTML: Hypertext Markup Language* [online]. 2018 [cit. 2020-03-20]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTML>
9. MOZILLA FOUNDATION. *So what is HTML?* [online]. 2019 [cit. 2020-03-20]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTML>
10. DARRELL, Robert. *The History of HTML* [online]. c2004-2011 [cit. 2020-03-20]. Dostupné z: <https://www.ironspider.ca/webdesign101/htmlhistory.htm>
11. MOZILLA FOUNDATION. *HTML5* [online]. 2019 [cit. 2020-03-21]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>
12. MOZILLA FOUNDATION. *CSS: Cascading Style Sheets* [online]. 2020 [cit. 2020-03-21]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/CSS>

13. KOSEK, Jiří. *Kaskádové styly* [online]. 1998 [cit. 2020-03-21]. Dostupné z: <https://www.kosek.cz/clanky/dhtml/styly.html>
14. MOZILLA FOUNDATION. *CSS basics* [online]. 2019 [cit. 2020-03-21]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics
15. HISSOM, Amy. *History of HTML and CSS* [online]. c2011 [cit. 2020-03-21]. Dostupné z: <http://www.amyhissom.com/HTML5-CSS3/history.html>
16. PEHAM, Thomas. *10 powerful things you didn't know about CSS3* [online]. c2020 [cit. 2020-03-21]. Dostupné z: <https://usersnap.com/blog/css3-facts-web-development/>
17. ČÁPKA, David. *Lekce 1 - Úvod do CSS frameworku Bootstrap* [online]. 2019 [cit. 2020-03-21]. Dostupné z: <https://www.itnetwork.cz/htmlcss/bootstrap/kurz/uvod-do-css-frameworku-bootstrap>
18. BOOTSTRAP TEAM. *About* [online] [cit. 2020-03-21]. Dostupné z: <https://getbootstrap.com/docs/4.0/about/overview/>
19. TUTORIALS POINT. *JavaScript - Overview* [online]. c2020 [cit. 2020-03-22]. Dostupné z: https://www.tutorialspoint.com/javascript/javascript_overview.htm
20. GURU99. *What is PHP? Write your first PHP Program* [online]. c2020 [cit. 2020-03-21]. Dostupné z: <https://www.guru99.com/what-is-phpfirst-php-program.html>
21. THE PHP GROUP. *History of PHP* [online]. c2001-2020 [cit. 2020-03-21]. Dostupné z: <https://www.php.net/manual/en/history.php.php>
22. MÁČA, Jindřich. *Lekce 1 - Úvod do Symfony frameworku pro PHP* [online]. 2018 [cit. 2020-03-21]. Dostupné z: <https://www.itnetwork.cz/php/symfony/zaklady/uvod-do-symfony-frameworku-pro-php/>
23. POTENCIER, Fabien. *What is Symfony2?* [online]. 2011 [cit. 2020-03-21]. Dostupné z: <http://fabien.potencier.org/what-is-symfony2.html>
24. SYMFONY SAS. *About Symfony Project* [online] [cit. 2020-03-21]. Dostupné z: <https://symfony.com/about>
25. HANSEN, Stevonn. *7 Good Reasons to Use Symfony Framework for Your Project* [online]. 2017 [cit. 2020-03-21]. Dostupné z: <https://hackernoon.com/7-good-reasons-to-use-symfony-framework-for-your-project-265f96dcf759>
26. BROWN, Philip. *What is PHP Composer?* [online]. 2013 [cit. 2020-03-21]. Dostupné z: <https://cultht.com/2013/01/07/what-is-php-composer/>

27. COMPOSER. *Introduction* [online] [cit. 2020-03-21]. Dostupné z: <https://getcomposer.org/doc/00-intro.md>
28. ORACLE CORPORATION. *What is MySQL?* [online]. c2020 [cit. 2020-03-22]. Dostupné z: <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>
29. RIEUF, Emmanuelle. *History of MySQL* [online]. 2016 [cit. 2020-03-22]. Dostupné z: <https://www.datasciencecentral.com/profiles/blogs/history-of-mysql>
30. PLEW, Ron; STEPHENS, Ryan. *Welcome to the World of SQL* [online]. 2002 [cit. 2020-03-22]. Dostupné z: <https://www.informit.com/articles/article.aspx?p=29583>
31. G., Domantas. *What is Apache? An In-Depth Overview of Apache Web Server* [online]. 2020 [cit. 2020-03-22]. Dostupné z: <https://www.hostinger.com/tutorials/what-is-apache>
32. ADAPTIC, S.R.O. *Apache Server* [online]. c2005–2020 [cit. 2020-03-22]. Dostupné z: <https://www.adaptic.cz/znalosti/slovnicek/apache-server/>
33. THE EDITORS OF ENCYCLOPAEDIA BRITANNICA. *Apache Web Server* [online]. 2017 [cit. 2020-03-22]. Dostupné z: <https://www.britannica.com/technology/Apache-Web-server>
34. KHOA, Leo. *Documentation* [online]. c2020 [cit. 2020-03-22]. Dostupné z: <https://laragon.org/docs/>

5 Přílohy

Příloha A – CD se zdrojovým kódem aplikace

Příloha A – CD se zdrojovým kódem aplikace

BakalarskaPrace.7z