VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

VÝVOJ PODPŮRNÉHO SOFTWARE PRO AUTOŠKOLY
DEVELOPMENT OF SUPPORTING SOFTWARE FOR DRIVING SCHOOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Ondřej Tovaryš

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Aleš Klusák, Ph.D.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Ondřej Tovaryš**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Aleš Klusák, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Vývoj podpůrného software pro autoškoly

Charakteristika problematiky úkolu:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrhy řešení, přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Cílem práce je navrhnout a implementovat do praxe specifický informační systém, který budou moci využívat autoškoly a jejich studenti především pro plánování jízd a závěrečných zkoušek. Systém bude navrhnut jako webová aplikace, aby mohl být nasazen na veřejně dostupném webovém serveru.

Základní literární prameny:

BASL, J. a R. BLAŽÍČEK. Podnikové informační systémy: podnik v informační společnosti. 3., aktualizované a dopl. vyd. Praha: Grada, 2012. ISBN 978-80-247-7594-4.

HOPKINS, C. PHP okamžitě. Brno: Computer Press, Albatros Media a. s., 2016. ISBN 978-80-251-4393-3.

KOCH, M. a B. NEUWIRTH. Datové a funkční modelování. 4. vyd. Brno: Akademické nakladatelství CERM, 2010. ISBN 978-80-214-4125-5.

PROCHÁZKA, D. SEO: cesta k propagaci vlastního webu. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-4222-9.

ŘEPA, V. Analýza a návrh informačních systémů. Praha: EKOPRESS, s.r.o., 1999. ISBN 80-86119-13-0.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17.

V Brně, dne 28. 2. 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

L. S.

B. Půža

doc. RNDr. Bedřich Půža, CSc.
ředitel

doc. Ing. et Ing. Stanislav Škapa, Ph.D.
děkan

ABSTRAKT

Diplomová práce se zaměřuje na návrh a implementaci webové aplikace na základě požadavků a analýzy současného stavu především pomocí technologií PHP a MySQL. Jedná se o specifický systém pro plánování jízd, závěrečných zkoušek a s tím spojených agend v autoškole.

ABSTRACT

This thesis is focused on design and implementation web application it based on the requirements and analysis of the current situation using technologies PHP and MySQL. It is specification system for planning rides, final exams and related functions in driving school.

KLÍČOVÁ SLOVA

Webová aplikace, informační systém, PHP, MySQL, HTML, CSS, Cloud Computing, autoškola.

KEY WORDS

Web application, information system, PHP, MySQL, HTML, CSS, Cloud Computing, driving school.

BIBLIOGRAFICKÁ CITACE

TOVARYŠ, O. *Vývoj podpůrného software pro autoškoly*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2017. 105 s. Vedoucí diplomové práce Ing. Aleš Klusák, Ph.D.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 25. května 2017 ……..

podpis

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé diplomové práce panu Ing. Aleši Klusákovi, Ph.D. za cenné rady, připomínky a vstřícnost při konzultacích.

OBSAH

ÚVOD

Masivní rozšíření globální sítě – internetu, přispívá k pohodlnější komunikaci mezi lidmi, úřady a organizacemi. Webový prostor je v současné době nejsnadnější a nejefektivnější cesta jak dopřát širokému spektru uživatelů příležitost dostat se k potřebným informacím. Lidé, zejména mladší část populace, jsou v dnešní době zvyklí komunikovat a organizovat si svůj čas online.

Tato práce se zabývá analýzou problematiky, návrhem a tvorbou webové aplikace (informačního systému) pro potřeby autoškol a jejich studentů. Výsledný systém bude mimo jiné sloužit k plánování jízd a závěrečných zkoušek pro studenty autoškoly. Dále pak jako přehled uskutečněných jízd, přehledy studentů, správu vozového parku a veškeré související agendy, která je nezbytná pro efektivní chod autoškoly.

Nejefektivnější způsob plánování jízd a závěrečných zkoušek pro studenta autoškoly je využití jejího informačního systému dostupného skrze webový prostor. Tento způsob šetří čas autoškole, protože nezdržuje neustálým vyřizováním telefonátů a hledáním volného termínu jízdy, a i studentům, kteří nemusí osobně přicházet do kanceláře, kde se termíny jízd (popř. závěrečných zkoušek) vedou v papírové podobě. Ušetřený čas používáním plánovače a přidružených agend není jediný benefit, který systém poskytuje. Dalším z mnoha jiných jsou efektivnější využívání vozidel či řízení instruktorů.

CÍL PRÁCE A METODIKA

Cílem této diplomové práce je navrhnout a implementovat do praxe specifický informační systém, který budou moci využívat autoškoly a jejich studenti především pro plánování jízd a závěrečných zkoušek. Systém bude navrhnut jako webová aplikace, aby mohl být nasazen na veřejně dostupném webovém serveru. Bude distribuován jako Platform as a Service. Plánovač jízd, závěrečných zkoušek a veškeré potřebná agenda s tím související bude navržena co nejvíce univerzálně. To znamená, že pro reálné nasazení v jakékoliv autoškole bude potřeba jen minimum programovacích zásahů, ideálně vůbec žádné.

Při návrhu bude kladen důraz na jednoduché a jasné ovládání, aby systém mohl ovládat student jakéhokoliv věku, dovedností a znalostí, bez zdlouhavého školení. Totéž platí pro administraci systému, kterou budou mít na starosti zaměstnanci autoškoly, s tím rozdílem, že tato agenda bude krátké zaškolení obsluhy vyžadovat – pro pochopení systémových souvislostí. Celý systém bude obsáhlý do té míry, aby pokryl veškerou potřebnou agendu a zároveň nebyl „přeplněný“ zbytečnými funkcemi, které se nevyužijí. Systém bude možné rozšířit o novou potřebnou funkčnost pomocí modulů.

Do systému budou mít přístup uživatelé se třemi rozdílnými přístupovými právy. Bez autentizace nebude mít do systému přístup nikdo. Důraz bude kladen na zabezpečení celé aplikace. V případě havárie nebo napadení systému by byl omezen chod autoškoly, k tomu se váží finanční ztráty, které nejsou žádoucí.

V teoretické části budou vysvětleny pojmy, které se vážou k tvorbě webových aplikací. Detailně budou popsány všechny technologie pro tvorbu systému. V analýze bude zhodnocena současná situace, jaké probíhají procesy, požadavky na aplikaci a analýza problémů spojených s návrhem. Praktická část se bude zabývat realizací. Na závěr bude provedeno ekonomické zhodnocení, shrnuty přínosy, a hlavně ohlasy z praxe, protože systém bude nasazen v ostrém provozu. Výstupem celé práce bude plně provozuschopný, snadně ovladatelný systém s administrátorskými i uživatelskými právy pro přístup, který budou moci reálně využívat autoškoly po celé České republice. V případě rozšíření aplikace do jiných států by bylo potřeba program legislativně upravit.

1 TEORETICKÁ VÝCHODISKA PRÁCE

V této části práce jsou shrnuty teoretické poznatky, pojmy a technologie, pro správné pochopení a ucelený pohled, z kterého vychází praktická část práce.

1.1 Internet

Internet je velmi rozsáhlá, celosvětová počítačová síť, která propojuje jednotlivé menší sítě pomocí sady protokolů IP (Internet protocol). V počátku byl internet počítačová síť propojující pouze strategické, vojenské, akademické a vládní počítače. Byla navržena bez hlavního řídícího centra a zájmem jeho tvůrců bylo, aby síť byla co nejméně zranitelná. Tato síť se skládala z řady vzájemně propojených uzlů, které byly rovnocenné z hlediska důležitosti. Přenos dat je navržen tak, že data se po dobu přenosu rozdělí na menší samostatné části, které se nazývají pakety. Každý paket v sobě nese informaci o adresátovi a putuje k cíli samostatně, svou vlastní cestou, nezávisle na ostatních paketech. V případě přerušení jedné přenosové cesty, může paket bez problémů využít alternativní cestu, a to přes zbývající zachované uzly. Toto je základ koncepce internetu, tak jak je známý dodnes. [1]

Počátky internetu se datují do 60. let 20. století a až po současnost docházelo k bouřlivému vývoji a technologickému postupu. Pro ilustraci pár důležitých milníků: v roce 1972 byla vyvinuta první e-mailová aplikace. Rok 1980 představuje experimentální provoz protokolu TCP/IP. V roce 1984 byl vyvinut DNS (Domain Name System) a až v roce 1987 se ujímá název sítě – internet. V roce 1991 byla nasazena služba WWW (World Wide Web). V roce 1994 přechází internet z rukou vědců ke komerčnímu využití a masovému nasazení. V roce 2000 internet využívá cca 250 miliónů uživatelů, o 6 let později v roce 2006 přesáhl hranici 1 miliardy uživatelů. Rok 2008 se spojen s masivním nasazením a rozvojem sociálních sítí (Facebook, Twitter) a v roce 2010 má internet více než 2 miliardy uživatelů. [1]

1.1.1 Princip komunikace

Každé síťové rozhraní, které komunikuje skrze protokol IP, má přiřazen jednoznačný identifikátor – IP adresu. V každém paketu je uvedena IP adresa odesílatele i příjemce. Podle IP adresy příjemce pak každý uzel na trase rozhoduje, jakým směrem paket odeslat dále – provádí tzv. směrování (spíše známé pod pojmem routing). Pro lepší orientaci v IP adresách byl zaveden systém pojmenování domén (DNS). Doménové názvy se v tomto systému překládají na IP adresy v podobě čísel a naopak. DNS má obrovský význam pro uživatele, kteří si tak nemusí pamatovat dlouhá čísla, stačí název domény. [1]

1.1.2 World Wide Web

World Wide Web (známější spíše pod zkratkou WWW) je jednou z mnoha služeb internetu. Označuje se tak systém prohlížení, ukládání a odkazování dokumentů nacházejících se v internetu. Jedná se o nejpoužívanější službu. Založena je na principu klient-server, obě strany spolu komunikují pomocí protokolu HTTP. Klientem je webový prohlížeč, který komunikuje s webovým serverem a zobrazuje požadované dokumenty. Webový server přijímá požadavky od klientů a vrací tyto dokumenty, které jsou navzájem propojeny pomocí hypertextových odkazů zapisovaných ve formě URL. [1]

1.2 Technologie

Tato podkapitola se zabývá technologiemi, které se týkají tvorby webových aplikací a jsou využívány při budování webových systémů.

1.2.1 PHP

PHP je zdaleka nejoblíbenější skriptovací jazyk na straně serveru v oblasti vývoje webových aplikací. Je na něm postaveno okolo 80 % všech webových stránek. Původně okolo roku 1995, kdy tento jazyk vytvořil Rasmus Lerdorf, bylo PHP zkratka pro „Personal Home Page“. Později se ujal název „Hypertext Preprocessor“, který je využíván dodnes. Jazyk PHP je volně dostupný ke stažení na oficiálních webových stránkách. Spravuje ho skupina vývojářů s názvem „The PHP Group“. [2]

Jazyk PHP bývá zpracováván webovým server s nainstalovaným modulem PHP. Jeho výsledky jsou zobrazovány ve webovém prohlížeči. S pomocí jazyka PHP můžeme vytvářet dynamické webové stránky, tj. takové, které se mění na základě různorodých podmínek. Kód jazyka PHP se vkládá mezi značky „**<?php**“ a „**?>**“ přímo do kódu HTML v souborech s příponou *.php*. [2]

```
... HTML kód…

<?php
Kód jazyka PHP;
?>

… HTML kód…
```

Obrovská přednost jazyka PHP spočívá v tom, že je možné ho zdarma zprovoznit na téměř všech operačních systémech (Linux, Windows, Mac OS X, …) a na všech platformách. Nejčastěji běží na serveru pod operačním systémem Linux v běhovém prostředí Apache. V dnešní době se tento jazyk využívá především pro řešení úloh na straně serveru, a ne jako obecný skriptovací jazyk. Jeho hlavní úloha je zpracovávat data tak, aby je bylo možné dynamicky zobrazovat do webových stránek. Provádí se pomocí něj matematické výpočty, převody formátů a spolupracuje s databázemi. [2]

Jazyk PHP je obvyklá volba pro webové vývojáře, protože se specializuje na webový vývoj – webové stránky nebo aplikace. Umožňuje vývojářům vylepšovat statické stránky o reakce na uživatelské podněty nebo ukládat data do databáze a později je uživatelům vypisovat. Poměrně rychle dovoluje i začínajícím programátorům, aby v něm začali vytvářet vlastní aplikace. Zároveň poskytuje obrovské množství funkcí, takže se vývojáři ve většině případů nemusí obracet na jiné programovací jazyky. Jednoduchým příkladem použití může být zobrazení počtu návštěvníků na stránce, ale i složité víceúrovňové navigace webu nebo celé redakční systémy. Jazyk PHP pohání i tak obrovské projekty jako je například Yahoo, Facebook nebo Wikipedia. [2]

PHP je dnes standardně k dispozici na jakémkoli webovém hostingu, ale můžeme ho jednoduše nainstalovat spolu s webovým serverem (nejčastěji Apache HTTP Server) na domácí počítač. Tímto si vytvoříme svůj lokální vývojový server, který můžeme zpřístupnit pomocí internetu celému světu. Skriptovací jazyk podporuje rozhraní pro komunikaci s databázovými systémy, jako například velmi oblíbený MySQL. [2]

1.2.2 SQL

Structured Query Language (spíše známý pod zkratkou „SQL“) je standardním databázovým dotazovacím jazykem pro manipulaci a získávání dat z relačních databází. Je navržen tak, aby sloužil principům relačního modelu databáze. Jeho největší výhoda spočívá v tom, že je použitelný téměř na všech platformách a produktech. Podporuje ho většina systémů pro správu databáze, jako například MySQL nebo Oracle. Pomocí SQL může správce databáze nebo programátor provádět následující činnosti: [3]

- Získávat z databáze libovolnou informaci
- Upravovat databázovou strukturu
- Aktualizovat obsah databáze
- Měnit nastavení zabezpečení systému
- Přidávat přístupová práva k databázím či jednotlivým tabulkám

První verze tohoto jazyka vznikla v polovině 70. let a vyvinula ji firma IBM jako standardní jazyk pro přístup k vůbec první relační databázi, kterou IBM provozovala.

Většinu základních příkazů, jako například „*SELECT*“ pro výběr, „*INSERT*“ pro vkládání, „*DELETE*“ pro mazání nebo „*UPDATE*“ pro upravování, můžeme použít ve všech databázových systémech stejně nebo velice podobně. Příkazy jazyka SQL se dělí na čtyři základní skupiny: [3]

- Příkazy pro definici dat (CREATE, ALTER, DROP, …)
- Příkazy pro manipulaci s daty (SELECT, UPDATE, INSERT, …)
- Příkazy pro řízení přístupových práv (REVOKE, GRANT)
- Příkazy pro řízení transakcí (COMMIT, ROLLBACK, …)

Ukázka použití základních a nejvíce používaných příkazů jazyka SQL (v praxi se dotazy často skládají z více řádků, níže jsou uvedeny pouze ve své nejjednodušší, základní formě):

Příkaz INSERT vloží uživatele do tabulky „*uzivatele*“:

```
INSERT INTO uzivatele (jmeno, prijmeni) VALUES (‘Valentino‘, ‘Rossi‘)
```

Příkaz SELECT vybere z tabulky „uzivatele“ uživatele s příjmením „Rossi“:

```
SELECT prijmeni FROM uzivatele WHERE prijmeni = ‘Rossi‘
```

Příkaz UPDATE změní jméno uživateli s unikátním identifikátorem 8 na „Till“:

```
UPDATE uzivatele SET jmeno = ‘Till‘ WHERE id_uzivatele = 8
```

Příkaz DELETE odstraní uživatele s primárním klíčem 46:

```
DELETE FROM uzivatele WHERE id_uzivatele = 46
```

Jedna z nejpopulárnějších implementací jazyka SQL je MySQL, které se využívá při tvorbě webových aplikací. MySQL se bude věnovat následující podkapitola.

1.2.3 MySQL

MySQL je systém řízení báze dat, který uplatňuje relační databázový model. Snad všechny webové aplikace, které jsou vystavěny na bázi skriptovacího jazyka PHP spolupracují s nějakou databází. Jedním z nejvíce používaných databázových systémů ve webovém prostředí je MySQL. Je to produkt vytvořený švédskou firmou MySQL AB. Pokud MySQL porovnáme s jinými databázovými systémy, patří spíše k těm jednodušším. Nejsou vysoké ani jeho nároky na hardware. MySQL je rychlý, malý, jednoduchý, poměrně spolehlivý, a hlavně nenáročný databázový systém. [4]

MySQL je považován za průkopníka dvojího licencování. K dispozici je jak pod bezplatnou licencí GPL (General Public licence), tak pod licencí placenou. Základním jazykem pro práci s MySQL je jazyk SQL, podobně jako u jiných databází se jedná o jeho rozšíření s některými funkcemi navíc (některé jsou zase ochuzeny). Databáze v MySQL se skládá z databázových tabulek vzájemně propojenými relačními vztahy. Žádná data se nemohou nacházet mimo databázovou tabulku, všechna jsou uložena v tabulkách. Tabulka se, stejně jako u jiných relačních databází, skládá ze schéma relace a těla relace. Schéma relace se skládá z jednotlivých domén, které mají určen svůj datový typ. Tělo tvoří data tabulky – každý řádek je jeden záznam, který má svůj unikátní identifikátor tzv. primární klíč. [4]

Primární klíč by měl být definován nad všemi tabulkami. Výjimkou mohou být např. tabulky s pouze jedním záznamem. Zásada při tvorbě vztahů mezi tabulkami je, že každá tabulka může mít pouze jeden primární klíč. Primární klíč se může vybrat přímo z dat tabulky, tento klíč se nazývá přirozený. Mnohem častější je ovšem jev, kdy se vytváří tzv. klíč umělý, který s daty v tabulce nesouvisí – pouze se na ně odkazuje (nejčastěji sloupec „id“). V obou případech musí primární klíč splňovat dvě základní vlastnosti: [4]

- Jedinečnost v rámci tabulky
- Nesmí obsahovat hodnotu NULL

Databázový systém by správně měl být navrhnut tak, aby se primární klíč daného záznamu nemusel nikdy měnit.

1.2.4 HTML

HyperText Markup Language (zkráceně HTML) je značkovací jazyk určený k vytváření dokumentů, které obsahují hypertextové odkazy a pokročilejší formátování textu (webové stránky). Příklady použití HTML v dokumentu: [5]

- Vložení obrázku a obrázkových map různých tvarů a velikostí, možnost vložení i animovaných GIFů
- Vytváření formulářů, tabulek, rámů
- Nastavení vzhledu a velikosti textu, různé formátování textu
- Strukturování textu do odstavců, vytváření hypertextových odkazů

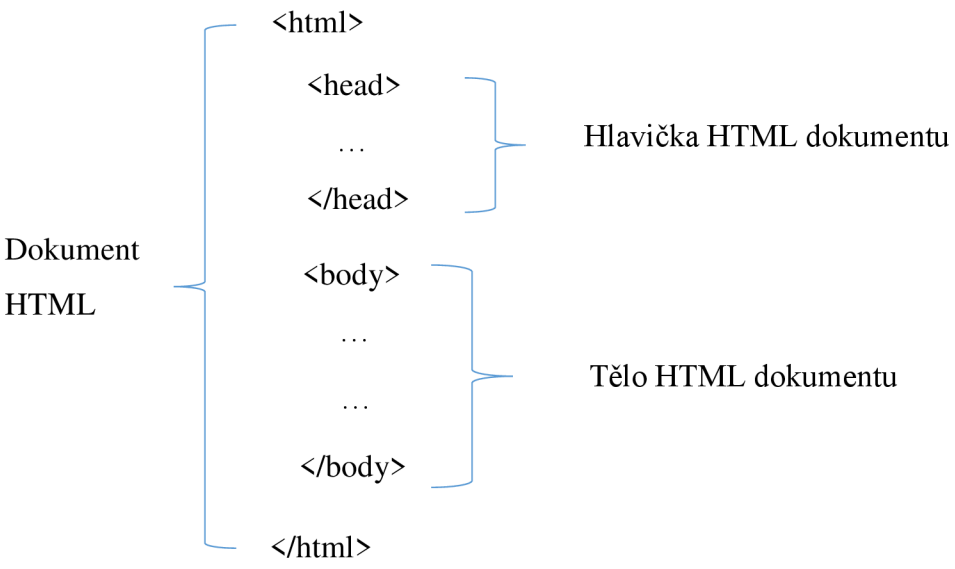
Možnosti využití HTML jsou opravdu široké, nicméně je potřeba si uvědomit, že HTML slouží tvorbě statických webových stránek. Pomocí jazyka HTML například není možné vytvořit ověřování uživatelů, ukládání či vypisování dat z databáze nebo vytvářet dynamicky se měnící nabídky apod. Pro zobrazení výsledné stránky není potřeba žádný kompilátor nebo speciální běhové prostředí, stačí jakýkoliv webový prohlížeč. [5]

HTML dokument se od obyčejného textového dokumentu odlišuje tím, že v obsahu navíc zahrnuje i informace o formátování stránky. Tyto informace (značky) jsou určeny pouze pro prohlížeč vytvořených dokumentů. Uživatel je při prohlížení dokumentu nevidí. Značky (tzv. tagy) se zapisují do ostrých závorek „< >“. Většina těchto značek jsou párové, to znamená, že existuje stejná značka s lomítkem na začátku, která ukončuje platnost tagu. Například: [5]

```
<b>tento text bude tučně</b>
<i>tento text bude kurzívou</i>
```

Samotné používání značek nedělá z obyčejného textu HTML dokument. Každá webová stránka musí mít určitou strukturu. Celý obsah webové stránky je uzavřen mezi značkami *<html>* a *</html>*. Dvěma důležitými částmi dokumentu je hlavička uzavřená mezi

značky *<head>* a *</head>*, kde se nachází např. nadpisek stránky, definice kaskádových stylů, údaje o použitém kódování apod. Druhou nezbytnou částí je tělo dokumentu mezi značkami *<body>* a *</body>*, tady se nachází veškerý obsah webové stránky. [5]



Obrázek č. 1 – Struktura HTML dokumentu (zdroj: vlastní zpracování)

První verze HTML 1.0 vznikla v roce 1990, neustále se vyvíjela a vyvíjí dodnes. Druhá verze vyšla v roce 1994 a stala se oficiálním standardem. Podporu kaskádových stylů a skriptování na straně klienta měla plně až verze 4.0. V současnosti se využívá verze 5.0, jejíž největší přednosti spočívají ve zjednodušeném zápisu některých elementů a možnosti přehrávat multimediální obsah přímo ve webovém prohlížeči. Se značkovacím jazykem HTML úzce souvisí kaskádové styly (CSS). [5]

1.2.5 CSS

Cascade style sheets (spíše známé pod zkratkou „CSS“ nebo česky „kaskádové styly“) slouží k formátování webových dokumentů. Jedná se o samostatný, ale doplňující jazyk k HTML. Vznik CSS se datuje okolo roku 1997, v dřívějších dobách se data na stránce uspořádávala pouze pomocí HTML tabulek. Díky kaskádovým stylům v dnešní době můžeme tvořit moderní a efektivní webové prezentace. CSS například umí: [6]

- Jakkoliv formátovat písmo
- Nastavit pozadí jakýchkoliv objektů nebo písma
- Zvýrazňovat odkazy po přejetí myši
- Jakékoliv pozicování objektů
- Téměř jakoukoliv část dokumentu zneviditelnit, zprůhlednit nebo nezobrazit
- Předefinovat grafický význam běžných značek a mnoho dalšího

V praxi mají CSS nepostradatelný význam. Hlavní přínos pro vývojáře spočívá v tom, že fungují hodně automaticky, vzhled celého webu se může deklarovat jedním souborem. S tímto souvisí trojí možnost použití CSS: [6]

- **Přímý styl** – formátování elementu přímo v textu dokumentu pomocí atributu *style="... "*. Je to nešikovné, ale občas se i tato možnost využívá.
- **Pomocí stylopisu v hlavičce dokumentu** – styly jsou nadefinovány v hlavičce každého dokumentu mezi značkami *<style>* a *</style>*. Nevýhoda spočívá v nutnosti přepisu každého stylopisu v dokumentu zvlášť, pokud se změna týká více dokumentů dohromady.
- **Externí stylopis** – styly jsou nadefinovány v souboru s příponou *.css*, na který se v hlavičce dokumentu odkazujeme značkou *<link>*. Kóderům tato funkce neuvěřitelně usnadňuje práci (na rozdíl od předchozí možnosti). Vzhled webových stránek je možno měnit v jednom souboru a změna proběhne všude, kde je použit.

Příklad kódu CSS:

```
body {
    color: blue;
    background-color: #ffa500;
    font-size: 1.5em;
}
```

Kód uvedený výše definuje pravidlo stylu. Vše, co je uvnitř složených závorek, se nazývá deklarační blok. Část před první složenou závorkou se nazývá selektor a určuje, která část

webové stránky se bude stylovat. Každý řádek deklaračního bloku se označuje jako deklarace a každá deklarace se skládá z vlastnosti (ta část před dvojtečkou) a hodnoty (část za dvojtečkou). Za každou deklarací následuje středník. [7]

1.2.6 JavaScript

JavaScript je skriptovací jazyk vestavěný do webového prohlížeče. Jeho obrovskou výhodou je, že ho lze použít ve většině prohlížečů. Pracuje přímo s prohlížečem, tj. kód se zpracovává na straně uživatele (ne na straně serveru jako je tomu např. u PHP) a je tedy zcela mimo kontrolu vývojáře. Jazyk byl vyvinut Brendanem Eichem ze společnosti NetScape v roce 1996. Z počátku se pomocí JavaScriptu pouze pracovalo s obrázky a ověřovaly se obsahy formulářů na straně klienta. Byla a je to velice užitečná funkce. Kontrola je okamžitá, nemusí se žádná data posílat na server pro ověření. Toto je skvělý způsob, jak zrychlit aplikace a nechat je reagovat na uživatelské podněty, a tím uspořit spoustu času nutného pro komunikaci se vzdáleným serverem. [8]

Pomocí JavaScriptu rychle a jednoduše dosáhneme způsobu, jak do statických stránek přidat dialogy, dynamiku nebo usnadnit uživateli práci s dokumentem. K vývoji skriptů v tomto jazyce je potřeba pouze libovolný textový editor a webový prohlížeč. Přes výhody, které poskytuje, má jazyk i svá omezení: [8]

- **JavaScript nepracuje na straně serveru** – skript nemůže komunikovat s ostatními zařízeními, nemá přístup k proměnným na straně serveru. Pracuje výhradně v prohlížeči uživatele. S tím souvisí problém, že nikdy vývojář nemůže vědět, jaké funkce daný klient podporuje, dokud ji na tomto klientovi nevyzkouší. Pokud při programování na straně serveru není některá z funkcí implementována, server selže a výsledek je znám hned. Server je jeden, klientů je mnoho. Pokud bychom chtěli vytvořit formulář a nashromážděná data uložit na server do databáze, musíme využít služby jiného jazyka.
- **JavaScript nemůže vytvořit grafiku** – složitější jazyky dokáží kreslit obrázky, kód JavaScriptu může pracovat s jen už existujícími soubory (GIF nebo JPEG).
- **JavaScript negarantuje bezpečnost dat** – jak již bylo zmíněno, JavaScript běží na straně klienta a tím vznikají vážné bezpečnostní důsledky. Kdokoli může

provádět kódové úpravy programu předtím, než se výsledek operace odešle (skrze jiný jazyk) zpět serveru. Je potřeba provádět kontrolu vstupů, hned jak se data dostanou na server. Zároveň každý uživatel si podporu JavaScriptu může ve svém prohlížeči vypnout, a tak veškeré bezpečnostní opatření ošetřená JavaScriptem postrádají smysl (zákaz stisku pravého tlačítka apod.)

1.2.7 jQuery

Knihovna jQuery se zakládá na programovacím jazyce JavaScript a je šířena pod licencemi MIT a GPL, tudíž ji je možno volně využívat ve svých projektech. Poprvé byla představena v roce 2006 Johnem Resigem. Knihovna obsahuje opravdu velké množství funkcí a efektů, jak vylepšit webové aplikace. Tato knihovna definuje speciální syntaxi, avšak nijak neomezuje programátora kombinovat javascriptový kód s kódem jQuery. Síla jQuery spočívá hlavně ve výběru jednotlivých elementů v modelu DOM (pozn. DOM je API, které umožňuje přístup nebo modifikaci struktury, obsahu či stylu dokumentu). Tyto elementy vybírá s pomocí selektorového jádra Sizzle – používá stejné selektory jako jazyk CSS, programátor může jednoduše vybírat s čím chce pracovat. Další silnou vlastností jQuery je schopnost modifikace jednotlivých elementů modelu DOM – například lze rychle a jednoduše měnit styly jazyka CSS jednotlivým elementům. [9]

Jednou z nejzajímavějších částí knihovny jQuery je široká škála efektů, jejichž použití je opravdu jednoduché a výsledek skvělý. S využitím této knihovny již není nutné psát složité kusy zdrojového kódu v JavaScriptu a následně je ladit pro správnou funkčnost ve všech prohlížečích. V knihovně jQuery se zavolá požadovaný efekt a ta provede veškerou práci s perfektním výsledkem. Programátor nemusí řešit nekompatibilitu mezi webovými prohlížeči. [9]

Pokud chceme knihovnu využít ve webové aplikaci, můžeme ji buď stáhnout a umístit na stránky nebo na ní můžeme linkovat. Nejčastěji se odkazuje na web Google, kde je tato knihovna nahrána. Tento způsob má obrovskou výhodu v tom, že server Google využívají téměř všichni a již mají tuto knihovnu načtenou v cache paměti prohlížeče. Načítaní webu

bude tedy rychlejší. Pro využití tohoto způsobu načtení knihovny, stačí do hlavičky stránky přidat následující řádek kódu: [9]

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js" type="text/javascript"></script>
```

Knihovna jQuery přímo spolupracuje s technologií AJAX a výrazně s ní zjednodušuje práci. Technologii AJAX se bude věnovat následující podkapitola.

1.2.8 Ajax

Po moderních webových aplikacích se vyžaduje, aby byly interaktivní. Proto, abychom toho dosáhli, obrátíme se na technologii AJAX (Asynchronous JavaScript and XML). První zmínka o Ajaxu se datuje do roku 2005, kdy James Garret publikoval článek Ajax: nový přístup k webovým aplikacím. Na scéně webových aplikací tato nová technologie (nejde tak ani o novou jednotlivou technologii, jako spíše o pojem označující použití více technologií dohromady za určitým cílem) způsobila poprask. Ajax nabídl webovým aplikacím interaktivitu podobající se desktopovým aplikacím. Obsah webových stránek se mění bez nutnosti jejich znovunačtení (Ajax komunikuje se serverem na pozadí). [10]

Technologie Ajax tedy umožňuje vytvářet mnohem lepší a uživatelsky přívětivější webové aplikace. Ajax nevyužívá žádnou novou technologii, dává dohromady staré technologie, které tvoří mocný celek. Díky tomu vývojáři stačí dosavadní znalosti, nemusí se učit nic nového. Podpora funkcionality ve webových prohlížečích je výborná. Nejčastější využití Ajaxu je například při kontrole formulářů při registracích. Uživatel zadává požadované hodnoty a při vyplnění pole se zadaná hodnota automaticky vyhodnocuje na serveru a vrací odpověď. Odpadá nutnost znovunačtení stránky. Uživatel ihned ví, kde se případná chyba nachází. Další časté využití je v podobě „našeptávačů“. Například při vyhledávání zboží, se uživateli okamžitě v roletce pod vyhledávacím políčkem zobrazují možné návrhy podle aktuálně napsaného řetězce. [10]

I přes veškeré nesporné výhody, které Ajax přináší, si musí vývojář při návrhu webové aplikace dát na určité věci pozor nebo se s některými problémy smířit. Například: v některých případech tlačítko zpět nefunguje tak jako u běžných stránek. Vyhledávače nemusí zaznamenat všechny stránky ajaxové aplikace. Zásadní nevýhoda nebo problém může vzniknout tehdy, kdy má uživatel v prohlížeči vypnutou podporu JavaScriptu – v tomto případě se Ajax stává nefunkční. [10]

1.2.9 Cron

Cron je systémový nástroj z prostředí Linuxu/Unixu. Hlavní náplň jeho činností je spouštět různé programy (skripty, části kódu) v předem danou dobu a předem stanoveném intervalu. Obdobou je v operačním systému Windows nástroj „Naplánované úlohy". Naprostá většina vyspělejších webových projektů se bez tohoto systémového nástroje neobejde. Cron nejčastěji slouží k: [11]

- **Získávání aktuálních informací z různých zdrojů na internetu** – například aktuální kurzovní lístek, výsledky sportovních utkání nebo reportáže
- **Generování databázově velmi náročných stránek** – data z databáze se mohou vytahovat jen vždy v určitém intervalu, ukládat do souborů nebo jinam. Z tohoto umístění je možné je číst bez neustálého přístupu k databázi a tím ji zbytečně vytěžovat
- **Mazání nepotřebných dat v databázi**
- **Provádění pravidelného zálohování databází nebo celých disků**

Pokud víme, který skript nebo program chceme s pomocí Cronu pravidelně spouštět, musí se nastavit i čas spouštění. Tento systémový nástroj nelze využít k jednorázovému spouštění aplikací či skriptů, spouští námi definované úlohy v předem daném intervalu. Nejmenší interval, který lze nastavit, je 1 minuta. Cron sám o sobě může spouštět skripty každou hodinu, denně, týdně nebo měsíčně. Pokud bychom chtěli program spouštět například v přesně daný čas 01:15 atp., musí se využít utilita crontab. Tato utilita slouží ke správě seznamů úloh, které systémový démon Cron vykonává. [11]

1.3 SEO

Search Engine Optimalization (SEO), neboli optimalizace pro vyhledávače, je označení pro metodiku vytváření webových stránek tak, aby jejich obsah a forma byly vhodné pro automatizované zpracování roboty internetových vyhledávačů. Cílem této metodiky je získat co možná nejvyšší pozice ve výsledku fulltextového vyhledávání při zadání klíčových slov, která úzce souvisí s obsahem daného webu. Pozice ve vyhledávačích (ideálně první strana) je pro tvůrce webů z hlediska návštěvnosti zásadní věc. Přesto je mnohdy opomíjena. SEO se ve většině případů začne řešit tehdy, kdy je webová stránka hotová, ale nenavštěvuje ji předpokládaný počet návštěvníků, tudíž nevznikají nové zakázky apod. [12]

SEO technik, které se používají k potencionální zvýšení návštěvnosti, je mnoho. Většina z nich je zdarma a závisí jen na znalostech a píli tvůrce webu. Například to mohou být tyto: [12]

- Optimalizace XHTML na požadované úrovni a validní kód (viz kapitola Validátory kódu)
- Správné a důsledné využívání klíčových slov
- Věcná a častá aktualizace obsahu webu
- Využití meta tagů popisující obsah webu
- Přístupný web s „hezkou“ URL
- Správné propojení se sociálními sítěmi
- Zpětné odkazy na spřátelené a podobně zaměřené weby

Do jisté míry je SEO kontroverzní technika. Lze ji totiž použít jak ve prospěch čtenáře webu (správné členění a relevantní informace), tak účelově ve prospěch tvůrce webu (přímočaře tvorba zisku, ale neprospěch čtenáře). Tzv. neetické metody SEO (někdy též podvodné) jsou techniky zakázané a při odhalení jsou tyto stránky penalizovány propadem ve vyhledávačích. Opětovný návrat je velmi těžký, mnohdy nemožný. Nicméně tyto techniky jsou přinejmenším krátkodobě úspěšné. Jedná se např. o: [13]

- **Cloaking** – Podstrčení rozdílného obsahu různým typům klientů. Například se vysoce optimalizovaná stránka (kterou by normální uživatel okamžitě opustil)

podstrčí Googlebotovi i ostatním robotům a jiná (uživatelsky přívětivější) stránka se zobrazí uživateli.

- **Doorway page** – Jediný účel, proč stránka existuje, spočívá v získání vysokého hodnocení ve vyhledávačích. Nejčastěji stránka obsahuje pouze velké množství klíčových slov, která se opakují. Uživateli se zobrazí pro něj nepochopitelná stránka s např. kontextovou reklamou, odkazy na jinou stránku, skrytý iframe nebo se do stránky umístí javascriptový kód pro přesměrování jinam. Tento druh zakázané techniky souvisí s klamnými přesměrováními uživatele a s odkazovými farmami.
- **Odkazové farmy** – Na jednu subdoménu se připraví mnohdy několik tisícovek až milionů stránek, které jsou vzájemně provázané odkazy. Robot stránky najde, zaindexuje a ty díky provázanosti získají vysoké hodnocení. Stránky se zobrazují ve vyhledávání a přesměrovávají návštěvníky na jiné (cílové) stránky. Druhá možnost je, že se na tyto stránky umístí odkazy směřující na stránky klienta, kterým tak uměle zvyšují hodnocení (např. PageRank). Důvodem jedné subdomény je to, aby nebyl problém tuto subdoménu zrušit, kdyby vyhledávač přišel na podvod a rozhodl se ji penalizovat. Takovéto domény se nazývají „Throwaway domains“.
- **Skrytý obsah** – Běžná stránka pro uživatele obsahuje jednu nebo více oblastí, ve které na nacházejí klíčová slova, která mnohdy se stránkou a jejím obsahem nijak nesouvisí. Tyto oblasti bývají uživateli skryty (např. pomocí CSS stylů). Díky těmto oblastem se uživatel dostane na stránku, kde najde jiný obsah, než pro který si přes vyhledávač přišel.
- **Klamné přesměrování** – Jedná se o jistou formu cloakingu. Uživatel je po příchodu na stránku přesměrován na jinou stránku, kterou nehledal. Přesměrování se provádí nejčastěji pomocí JavaScriptu, protože přesměrovávání skrze hlavičku http si robot snadno všimne. Přesměrování pomocí meta tagu *refresh* si snadno všimne parser. JavaScript může (ale taky nemusí) oběma uniknout.
- **Opakovaná a matoucí slova** – Obsah je stránky je uměle doplněn o větší množství nesmyslně použitých klíčových slov. Rozdíl oproti doorway page spočívá v tom, že se jedná o normální stránku, jen lehce „zdokonalenou“.

1.3.1 Analýza návštěvnosti

Vhodná analýza webu a jeho návštěvnosti prozradí, co jsou návštěvníci (a tedy potencionální zákazníci) zač. Vznikne přehled o tom, kde návštěvníci tráví nejvíce času, odkud přišli, co hledají, kolik jich na stránky přišlo, jak se po stránkách pohybují a jestli došli k cíli. Tímto cílem může například být objednání zboží nebo služeb. Vysoká návštěvnost webu nemusí znamenat, že je web úspěšný. Příklad: e-shop, který navštíví denně 10 000 návštěvníků, můžeme vnímat jako úspěšný do té doby, než zjistíme, že z 10 000 návštěvníků si zboží objednalo jen 5. K těmto účelům se využívá ukazatel „Konverzní poměr“, který udává pravděpodobnost, kdy se z návštěvníka stane zákazník. Důležité je rozdělit návštěvníky minimálně na dvě skupiny – náhodné a cílené. [12]

K analýze návštěvnosti se využívají nástroje třetích stran. Velice propracovaným nástrojem k analýze návštěvnosti je např. Google Analytics.

1.3.2 Ranky

Pojem rank v angličtině znamená hodnocení. V odborné komunitě se tento pojem začal chápat jako nějaké číselné ohodnocení stránky, které je nezávislé na hledaném dotazu. Každý vyhledávač má svůj rank a jeho hodnota se určitým způsobem projevuje v řazení výsledků (většinou jen malou měrou, protože do řazení vstupuje dalších mnoho faktorů). Existuje mnoho ranků a ty se počítají různými způsoby. Nejčastěji se ovšem uvádí, že jejich výpočet se provádí na základě odkazové sítě – z toho, z jakých a kolika stránek se na danou stránku odkazuje. Nejznámějším, celosvětově rozšířeným rankem, je PageRank od společnosti Google. [14]

PageRank

PageRank (PR) může nabývat hodnot na stupnici od nuly do jedné (klasické pojetí PageRanku). Google tuto hodnotu vypočítává z PageRanků stránek, které na tuto stránku odkazují. Hodnotu PageRank nemá celý web dohromady, ale každá jednotlivá stránka (URL). Vzorec pro výpočet: [14]

$$PR = (1 - d)/m + d * (PR(T_1) / C(T_1) + ... + PR(T_n) / C(T_n))$$

d – dampening faktor (s největší pravděpodobností nastaven na 0,85)

T_1 až T_n – PageRanky stránek, které na stránku odkazují

m – počet zaindexovaných stránek

C(T) – počet odkazů odkazující ze stránky T

Výše uvedený vzorec se dá zjednodušeně interpretovat tak, že stránka „přeposílá“ určitou část svého PR ostatním stránkám, na které odkazuje. Čím více je těchto odkazů (proměnná C), tím méně předá každé stránce. Pokud chce vývojář stránek z nějakého důvodu zvýšit svým stránkám hodnotu PageRanku, měl by se postarat o to, aby na ni vedlo co nejvíce odkazů z jiných stránek, které mají vysoký PR. [14]

1.4 Validita a validátory kódu

Z programátorského hlediska validita určuje, zda kód splňuje pravidla specifikace. Pomocí validity se dají odhalit případné chyby, a to jak formální, tak logické. Tvorbu pravidel při psaní jazyka XHTML zastřešuje konsorcium W3C. Postupem času, již od prvních verzí jazyka, se pravidla stále precizují a zpřísňují. Pro kontrolu, zda je kód naší či jakékoli jiné stránky validní, z hlediska kódů XHTML a CSS, se využívají tzv. validátory kódu. Validátory jsou volně dostupné nástroje, které dokážou najít chyby v psaném kódu. Existují v mnoha jazykových mutací, ale všechny vycházejí ze stejných pravidel konsorcia W3C. [15]

Celý proces kontroly validity pomocí validátoru je jednoduchou záležitostí. Pouze do příslušného pole formuláře zadáme adresu stránky, kterou má validátor zkontrolovat, on ji prověří a na obrazovku vypíše varování, chybové hlášky nebo v ideálním případě to, že je stránka z hlediska validity kódu v pořádku. Všechny chybové hlášky a varování mají detailní popis problému i s přesným číslem řádku, kde se daný problém nachází. [15]

Validní web má větší pravděpodobnost, že bude zobrazen v různých prohlížečích tak, jak tvůrce předpokládal. Validita také napovídá, že zdrojový kód má určitou kvalitu, což je zejména pro laickou veřejnost dobrý marketingový tahák. Nicméně daleko důležitější, než (někdy až přehnaná) posedlost validitou, je klást důraz na přístupnost webu. [15]

1.5 Bezpečnost webových aplikací

Bezpečnost webových aplikací je široké téma, které by se dalo dobře obsáhnout minimálně v celé knize. Tato podkapitola bude věnována pouze nejčastějším typům útoků a jak se proti nim bránit.

Session Hijacking

Protokol http je bezstavový, nicméně je potřeba nějakým způsobem alespoň určitou formu stavu přenášet. K tomu se využívá mechanizmu session. Při přihlášení je uživateli vytvořen session a jeho identifikace pomocí cookies nebo v menší míře pomocí předávání v URL. Problém Session Hijackingu spočívá v situaci, kdy se útočníkovi podaří získat identifikátor session. Toho lze zneužít a předstírat korektně přihlášeného uživatele. Z tohoto důvodu je nutné session identifikátory generovat opravdu náhodně, a navíc kontrolovat IP adresu uživatele (není to nutné, ale je to bezpečnější). [16]

Pokud se uživatel do systému přihlásí, vykoná, co vykonat chtěl a odhlásí se (opustí stránku, zavře prohlížeč – session vyprší platnost) nehrozí takové nebezpečí jako tehdy, když uživatel nechce zadávat login a heslo při každém přístupu do systému, tzn. chce zůstat přihlášen několik dní nebo týdnů (situace kdy se uživatelova IP adresa bude pravděpodobně měnit). [16]

Obrana proti tomuto typu útoku závisí na tom, zda se pro ukládání identifikace využívají cookies nebo URL. Při využití cookies, je potřeba správně omezit cookies na doménu, kontrolovat IP adresu a dobu platnosti. Nutné je zajistit, aby na serveru nebyl možný Cross Site Scripting. Při využití URL je potřeba zajistit, aby nebylo URL posláno žádnému jinému serveru jako HTTP Referer. V tomto případě se musí provést tyto kroky: [16]

- Zabránit načítání obrázků, stylopisů apod. z nedůvěryhodných serverů
- Při odkazování na jiný server přesměrovat uživatele (nelze využít přesměrování protokolem HTTP – při něm nedojde ke změně refereru)

Injection a SQL Injection

Obecně injection je typ útoku, který využívá špatného ošetření vstupu hodnot od uživatele do systému. Nejčastější typ injection útoku je SQL Injection, neboli špatné ošetření parametrů při tvorbě SQL dotazů. Příklad špatného (nebezpečného) dotazu (proměnná *value* obsahuje hodnotu ze vstupu od uživatele): [16]

```
dotaz = “select * from nazev_tabulky where pole = ‘ “.value.“ ‘ “;
```

Když do tohoto SQL dotazu uživatel do proměnné *value* vloží např. hodnotu:

```
“ ‘ or 1=1 or pole = ‘ “
```

dostane přístup ke všem hodnotám celé tabulky, spojením totiž vznikne dotaz:

```
select * from nazev_tabulky where pole = ‘ ‘ or 1=1 or pole = ‘‘
```

Jedná se o nejjednodušší příklad, v kterém se navíc nachází ještě jedna chyba – při tvorbě dotazů by se nikdy nemělo využívat hvězdičkové konvence, ale vypsat všechny názvy sloupců, které požadujeme. Pomocí chytře vložených subselectů nebo joinů by se daly získat data i z jiných tabulek. [16]

Řešení tohoto problému je poměrně jednoduché – SQL dotaz nesestavovat pomocí prostého skládání řetězců. Každý vývojářský jazyk má na tento problém vytvořené

konstrukce kódu nebo funkce, které se postarají o ošetření vstupních hodnot (zejména správné apostrofy). Například u jazyka PHP je to dobře známá funkce *mysqli_real_escape_string()*. [16]

Cross Site Scripting (XSS)

Cross Site Scripting je typem útoku, kdy se útočník snaží narušit webové stránky využitím bezpečnostních chyb ve skriptech, hlavně neošetřené vstupy. Díky těmto chybám může potencionální útočník do stránek podstrčit např. vlastní javascriptový kód, který může sloužit k poškození vzhledu stránky, obcházení bezpečnostních prvků aplikace nebo je často využíván při phishingu. Pomocí XSS je návštěvníkovi zobrazen jiný obsah na jinak důvěryhodné stránce. [16]

Obrana proti tomuto útoku na straně serveru spočívá v tom, že je nutné zajistit, aby zobrazované vstupy od uživatele nebyly do HTML stránky zapsány přímo, ale jen jako prostý text. Všechny HTML znaky musí být nahrazeny jejich entitními vyjádřeními, např. znak „<“ za „<“ nebo znak „>“ za „>“. Stejně jako u SQL Injection, má každý programovací jazyk funkce, kterými se dají vstupy poměrně jednoduše a účinně ošetřit. Obrana na straně klienta v podstatě neexistuje, zamezení útoku XSS je nutné zajistit na straně serveru. [16]

Cross Site Request Forgery (CSRF)

Cross Site Request Forgery je typ útoku na webové aplikace fungující na bázi vykonání určitého nezamýšleného požadavku v této aplikaci. Požadavek přichází z nelegitimního zdroje a zneužívá akce uživatelů (administrátorů), kteří jsou v daný okamžik v aplikaci přihlášení. Využívá se faktu, že velká část webových aplikací nabízí trvalé přihlášení (cookies má stálou platnost). Tento typ útoku nesměřuje k získání přístupu do systému (nicméně i pro tento účel může být zneužit), ale spíše k napáchání mnohdy nenapravitelných škod nebo získání potencionálně důležitých informací. [16]

Provedený útok může mít následující scénář. Webová aplikace má svou administrační část, kterou spravují administrátoři. Útočník zná nebo si dokáže nějakým způsobem opatřit URL adresy (popřípadě i další informace jako proměnné nebo parametry HTTP

metody GET) skriptů pro spouštění administrátorských akcí (smazání, editace, vložení apod.). HTTP metoda GET předává parametry v URL, což může být pro útočníka ulehčením, nicméně obranou není ani vyžadování příjmu dat pomocí metody POST, protože i tu lze zneužít. Útočník připravený skript nebo stránku, s využitím sociálního inženýrství nebo jiné metody, předá administrátorovi k načtení, čímž provede CSRF útok. [16]

Pokud je v okamžiku načtení této „infikované“ stránky (skriptu) uživatel platně přihlášen do systému, který není proti tomuto zabezpečen, je útok úspěšný. Požadavek na smazání či editaci obsahu se vykoná, protože aplikace není schopná rozlišit z jakého podnětu požadavek přišel. Obrana může spočívat ve dvou či více faktorové autentizaci, např. pomocí CAPTCHA nebo SMS. Operace z prohlížeče potom nelze podvrhnout. Tento způsob je ovšem z administrátorského hlediska značně nepohodlný. Při obraně by se mělo dbát na následující opatření: [16]

- Zajistit, aby na celém serveru nikde nemohl nastat útok typu XSS
- Platnost session mít co nejkratší (nepoužívat trvalé) - pokud není uživatel přihlášen, útok nehrozí
- Využívat autorizační token – náhodně vygenerovaný řetězec, platný pouze pro aktuálního uživatele. Neměl by být od ničeho odvozený, může se jednat o náhodně generovaný, dostatečně dlouhý řetězec. Před zpracováním dat se token vygeneruje a uloží např. do databáze či session. Token se odesílá společně s daty a při jejich zpracování se hodnoty tokenu porovnávají. Administrátor má platný token vždy automaticky předvyplněný, zatímco útočník jej nemá jak zjistit, a tudíž se jeho podvrhnuté požadavky neprovedou.
- Nespoléhat jen na jeden typ obrany, ale kombinovat více opatření dohromady

Clickjacking

Podstata typu útoku Clickjacking spočívá v nepředpokládané činnosti, kterou uživatel sám spustí na jinak zdánlivě neškodné stránce. Např. útočník na stránku vizuálně podstrčí tlačítko z jiné webové aplikace (pomocí iframe) tak, že uživatel o tomto faktu neví a považuje toto tlačítko za součást stránky, kterou vidí. Po stisku tlačítka (např. v domnění,

že jde o odeslání hodnot apod.) dojde k nepředpokládané akci, kterou může být obrovské množství činností, a to od pouhého přesměrování až po instalaci nevyžádaného software. Existuje mnoho variant a možností (mnohdy velice sofistikovaně provedené) využití tohoto typu útoku (postřehové „klikací“ hry apod.). Obrana spočívá především na prohlížečích, které se brání různě – zamezením opakovaného zobrazení vyskakovacích oken, čekací dobou před potvrzením dialogu k instalaci, upozornění uživatele před opuštěním webových stránek při přesměrování apod. [16]

1.6 Datový model

Datové modely definují strukturu a formát dat v informačních systémech a určují vzájemné vztahy objektů. Datový model je výsledkem datového modelování. Datové modelování je proces, při kterém se definují a analyzují požadavky na strukturu dat, s nimiž informační systém pracuje. Při budování informačního systému si analytik nevystačí s jedinou strukturou věty. Každý datový objekt zpravidla potřebuje navrhnout samostatnou strukturu věty. V zásadě jsou tři nejpoužívanější datové modely, a to: lineární, relační a objektový. [17]

1.6.1 Lineární

V lineárních datových modelech neexistují žádné vazby mezi jednotlivými objekty (tabulkami databáze). Mezi objekty není ani žádný vzájemný vztah s výjimkou vztahu „předchůdce“ a „následovníka“. Dobrým příkladem tohoto modelu je kartotéka pacientů u lékaře. Jednotlivé karty pacientů jsou seřazeny v šuplíku podle abecedy. Každá samostatná karta představuje jednu větu databázového souboru a nemá žádnou vazbu na ostatní karty. [17]

1.6.2 Relační

Z daleka nejpoužívanějším datovém modelem je model relační, resp. objektově-relační model. Tento model vzniká spojením dohromady několika lineárních modelů pomocí položek, které nazýváme relační klíč. Spojení modelů není trvalé, ale vzniká v okamžiku, kdy potřebujeme mít data k dispozici ze všech spojených tabulek. Spojení zaniká tehdy,

když práci s modelem ukončíme. Jednotlivé lineární modely lze využívat i samostatně. Relační datové modely umožňují zachytit v modelu nejen data o objektech, ale také vzájemné vztahy těchto objektů, což umožňuje přiblížit se reálnému světu. [17]

1.6.3 Objektový

Z trojice datových modelů je objektový model nejmladší. Vystavěn je na základním prvku – objektu (objektu přibližně odpovídá věta). Objekt má kromě svých atributů definované i metody, které určují chování objektu. Např. pokud je objektem „zaměstnanec“ firmy, atributem tohoto objektu jsou údaje: číslo zaměstnance, pozice, pracovní poměr atd. Objekt „zaměstnanec“ může mít nadefinovány různé metody, jako např. metodu „přiřaď do pracovní skupiny“, kde se zkontroluje, zda již v dané skupině není a jestli zde vůbec patří. Objekty stejného typu tvoří třídu, např. třída všech „zaměstnanců“. Konkrétní záznam o zaměstnanci se nazývá instance objektu. Každý objekt v dané databázi má svůj unikátní identifikátor – OID, pomocí něhož se dají mezi objekty vést přímé vazby. V objektově orientovaném modelu mohou existovat i relační vazby. Hlavní rys objektových modelů je tzv. „zapouzdření objektu“ – jediný způsob, jak s objektem pracovat je volání některé metody objektu (nevidíme „dovnitř“ objektu). [17]

1.7 Funkční model

Funkční modelování se zabývá zkoumáním a algoritmizací procesů a činností, které probíhají v informačním systému. Funkční model je přesný popis procesů, jak zpracovávat data z datového modelu a transformovat je na výstupní data. Při modelování a popisu procesů v informačním systému se provádí hierarchický rozklad funkcí od nejobecnějších až po funkce elementární. Existuje mnoho různých metod, pomocí kterých se znázorňuje funkční model IS. V této podkapitole jsou uvedeny pouze nejvíce používané metody. [17]

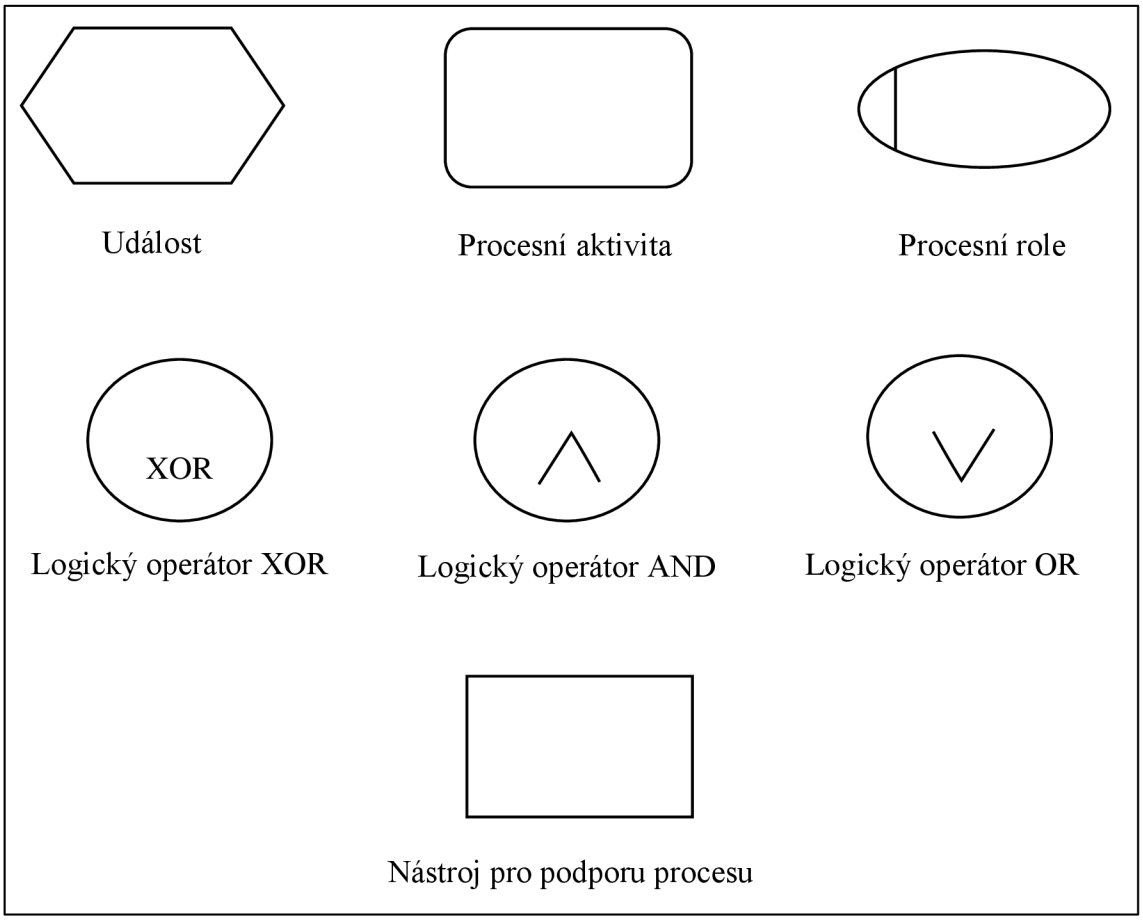
1.7.1 Slovní popis

Slovní popis funkčního modelu je nejpoužívanější metodou při analýze menších projektů. Slouží i pro komunikaci uvnitř analytického týmu. Tato metoda není výhodná pro

dokumentaci celých informačních systémů, protože je poměrně dost nepřehledná. Čtení dlouhých pasáží textu je zdlouhavé a mnohdy nejednoznačné. [17]

1.7.2 EPC diagram

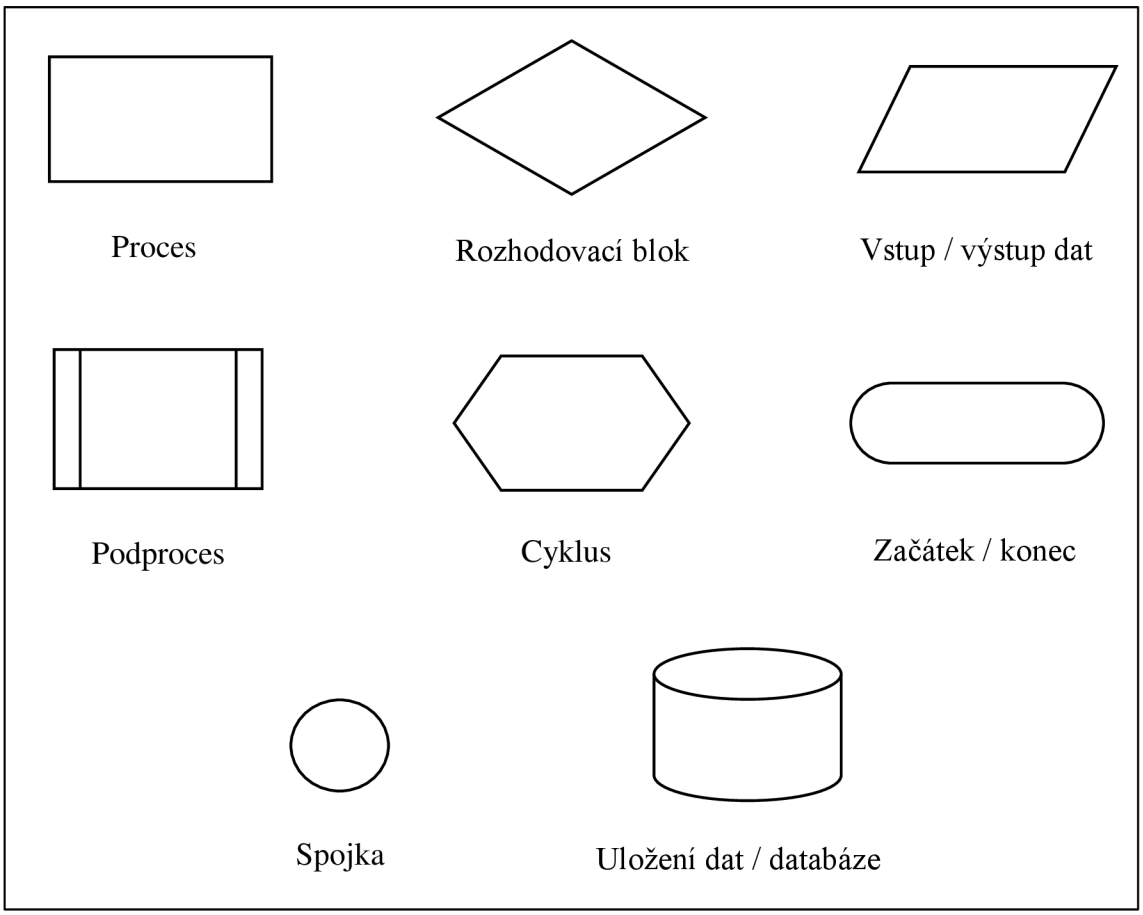
EPC diagram se řadí mezi nejpoužívanější grafické popisy procesu. Využíván bývá i k popisu pracovních postupů. V podnikové praxi se používá především k modelování, analýze a novým návrhům podnikových procesů. Nejčastěji používané značky pro tvorbu EPC diagramu jsou tyto: [17]



Obrázek č. 2 – Značky EPC diagramu (zdroj: vlastní zpracování dle [17])

1.7.3 Vývojový diagram

Vývojový diagram velice dobře zachycuje větvením programu podle splnění či nesplnění požadovaných podmínek. Spolu s DFD diagramem patří k nejpoužívanějším. Při tvorbě vývojového diagramu se pro snadnou orientaci a čtení dodržuje přirozený směr „shora dolů“ a „zleva doprava“. Pokud by se měly větve křížit, využívá se spojek. Základní značky pro tvorbu vývojových diagramů jsou tyto: [17]



Obrázek č. 3 – Značky vývojového diagramu (zdroj: vlastní zpracování dle [17])

1.7.4 Stavový diagram

Pomocí stavového diagramu se zkoumají možné stavy jednotlivých objektů, které mohou nastat. Tato metoda se snaží popsat, co vyvolá změnu přechodu mezi stavy. Stavový diagram musí být vždy tvořen pro určitý objekt (entitu) – např. student, instruktor apod. Diagram je jednoduchý a přehledný. V elipsách se zaznamenávají všechny možné stavy

objektu. Šipky s popisem určují, při jaké podmínce přechází systém do dalšího stavu, přičemž je možné, že stav za dané podmínky přechází do stavu stejného. [17]

1.7.5 DFD diagram (diagram toku dat)

Z diagramu toku dat se dá vyčíst: [17]

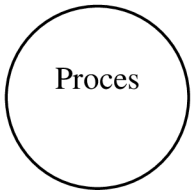
- návaznost jednotlivých činností v rámci úlohy,
- jaké datové vstupy a výstupy se v dané úloze vyskytují (s jakými soubory se pracuje),
- kdo jednotlivé činnosti provádí

Diagramy toku dat se kreslí v různých úrovních. Nejdříve se systém zachycuje jako celek a postupně se rozpracovávají jednotlivé funkce až na úroveň samotné úlohy. Jedná se o jednu z nejpoužívanějších metod funkčního modelování. V diagramu se nedají zachytit rozhodovací procesy, ale velmi přehledně zobrazuje, s jakými datovými soubory se v jednotlivých procesech pracuje. Při tvorbě DFD diagramu se musí dodržovat následující pravidla: [17]

- graf by neměl obsahovat více než 10 procesů
- v grafu se nemůže nacházet proces, který nemá vstupy, ale jen výstupy
- nesmí existovat proces, který má je vstupy a nemá výstupy
- datový tok z entity musí jít přes proces
- datový tok do a z uložení musí procházet přes proces

DFD diagram využívá tyto symboly (notace podle Yourdon and Coad): [17]

1. **Proces** – činnost, která převádí vstupní data na výstupní. Název by měl odpovídat podstatě procesu.



Obrázek č. 4 – Symbol procesu v DFD diagramu (zdroj: vlastní zpracování)

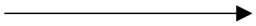
2. **Externí zdroj dat (entita)** – objekt, se kterým proces komunikuje (např. uživatel, zákazník).

Obrázek č. 5 – Symbol externí entity v DFD diagramu (zdroj: vlastní zpracování)

3. **Uložení dat** – místo kam se data ukládají, např. datový soubor.

Obrázek č. 6 – Symbol uložení dat v DFD diagramu (zdroj: vlastní zpracování)

4. **Pojmenovaný datový tok** – skrze datový tok dochází k přesunu dat v systému. Platí zásada, že jeden datový tok nese jeden typ informací.



Obrázek č. 7 – Symbol datového toku v DFD diagramu (zdroj: vlastní zpracování)

1.8 Úloha ICT

ICT (informační a komunikační technologie) sloužily v průběhu minulých dekád k různým podnikovým cílům. Pozornost se časem přesouvala z oblasti technických výpočtů, k podpoře automatizace výroby a hodnocení podnikových procesů. Integraci ICT s podnikovým businessem a manažerskými metodami, přinesla až dekáda po roce 2000. Do této doby bylo IT chápáno spíše jako samostatné oddělení, řešící „své záležitosti“. Integrace přinesla výrazně vyšší potřebu orientovat ICT na podnikové cíle, kterých se má jednodušeji a lépe dosáhnout. ICT se vyvinuly v podpůrný nástroj podnikání, který zvyšuje efektivitu a konkurenceschopnost. [18]

ICT našly v podniku své místo. Přistupovat se k nim musí stejně jako k ostatním výrobním a obchodním prostředkům. Jejich nasazení je zapotřebí strategicky plánovat a dobře zvažovat, jaký dopad budou mít na celou organizaci včetně přínosů a nákladů. Snahu o větší flexibilitu, výkonovou i nákladovou transparentnost reprezentuje směr outsourcingu podporovaný technologiemi typu SaaS (Software as a Service), SOA (Service Oriented Architecture) nebo modely cloud computingu (viz kapitola Cloud computing). K základním trendům v podniku patří využití ICT v e-businessu. Tato oblast je v současné době sledována zejména prostřednictvím ukazatelů on-line obchodování, tj. nákup a placení on-line. [18]

Obrovský potenciál k růstu prosperity, efektivity a konkurenceschopnosti poskytuje ICT všem podnikům bez rozdílu velikosti, speciálně však především malým a středním podnikům, na jejichž růstu a prosperitě závisí úroveň ekonomiky. Zajímavým faktem je v tomto ohledu informace, že limitujícím faktorem jsou nedostatečné znalosti a informace na straně uživatelských organizací. Zejména malé a střední podniky neumějí do ICT efektivně investovat a taktéž neumí ICT správně řídit po celou dobu jejich životního cyklu. [18]

Inovace ICT a jejich bezmyšlenkovité prosazování do života organizací a jednotlivců nemusí být vždy spojeno jen s pozitivy, mohou s sebou přinášet i obavy, zejména o ztrátu soukromí a bezpečnost. S neustále narůstající mírou kyberkriminality budou uživatelé ICT opatrněji využívat on-line služby (např. e-government, e-payment, e-health apod.).

Uživatelé se budou domnívat, že se především na sítě nebudou moci plně spolehnout. V tomto smyslu je potřeba se zabývat otázkou ochrany základních práv v oblasti osobních údajů a soukromí. [18]

1.9 Cloud computing

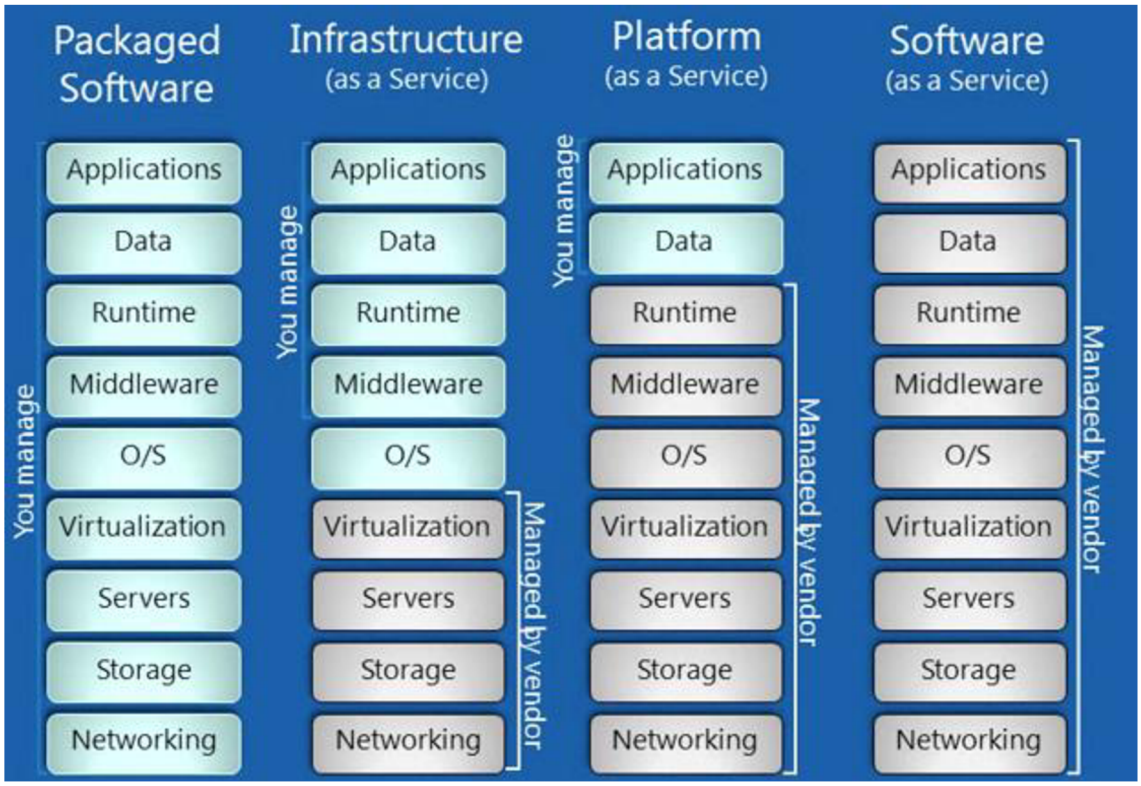
Pojem „cloud computing“ má mnoho definic, některé z nich jsou pojaty příliš úzce, některé příliš rozsáhlé, jiné jsou velmi striktní. Národní institut standardů a technologie (NIST) definuje cloud computing jako síťový přístup na vyžádání ke sdíleným a škálovatelným výpočetním zdrojům, jejichž kapacita může být rychle navýšena nebo naopak uvolněna, a to při minimálním úsilí nebo interakci s poskytovatelem služby. [19] Jinak řečeno, cloud computing je metoda poskytování ICT ve formě služby. Tuto službu poskytuje svým zákazníkům provozovatel cloudu pomocí internetu. Uživatelé tak mohou službu využívat odkudkoliv a téměř na jakémkoliv zařízení. Nejsou potřeba žádné schopnosti a vědomosti k tomu, aby takové služby získali a začali je využívat. Základní vlastnosti cloud computingu: [20]

- **Orientace na služby** – cloud computing poskytuje uživateli nějaké služby (např. datová uložiště, e-mail, funkčnost nějaké aplikace, výpočetní výkon…).
- **Škálovatelnost a elasticita** – poskytované služby se pružně přizpůsobují potřebám uživatele. Patří zde například možnosti navýšení potřebné výpočetní síly, velikosti uložiště a šířky pásma podle aktuálních potřeb uživatele. Odpadá tak proces mnohdy složité aktualizace lokální instalace vlastního hardwaru nebo softwaru.
- **Sdílení** – poskytovaná služba zpravidla pracuje s uloženými informace někde ve vzdáleném datovém centru. Vzhledem k centralizovanosti je snadné tyto informace zpřístupnit více uživatelům současně, a tím jim umožnit vzájemnou spolupráci.
- **Zpoplatnění za užívání** – platí se pouze za dobu užívání, případně objem uložených dat. Odpadají tak náklady na nákup hardwaru a softwaru pro vytvoření vlastních lokálních datových center. Například finanční prostředky za servery, chlazení, zdroje neustálého napájení, IT pracovníci pro správu infrastruktury a v neposlední řadě zálohování.

- **Zaměření na výsledek** – služby by měly fungovat tak, aby poskytovaly co největší efektivnost a snadnost používání (např. mnohdy je možné navýšit výpočetní výkon nebo velikost uložiště dvěma kliky).
- **Používání internetových technologií** – nutná podmínka. Jedinou dnes globálně dostupnou službou pro datové přenosy je internet. Bez připojení k internetu není možné služby cloud computingu využívat.

1.9.1 Distribuční modely

Naprostá většina služeb cloud computingu spadá do tří hlavních kategorií: infrastruktura jako služba (IaaS), platforma jako služba (PaaS) a software jako služba (SaaS). Tyto kategorie jsou postaveny jedna na druhé a jejich správná znalost je výhodná pro plnohodnotné a efektivní využití cloud computingu. Výstižně jsou jednotlivé kategorie, to, co pod ně spadá a kým jsou spravovány, zobrazeny na následujícím obrázku:



Obrázek č. 8 – Distribuční modely cloud computingu (zdroj: Silverlighthack.com - [21])

IaaS (Infrastructure as a Service)

IaaS je nejzákladnější kategorie služeb cloud computingu. Jde o pronájem IT infrastruktury, jako jsou například servery, uložiště, sítě nebo virtuální počítače, které spravuje poskytovatel. Uživatel instaluje a konfiguruje vlastní software, operační systémy a middleware. Uživatel nemá a často ani nemůže mít informaci o tom, kde se pronajímaná infrastruktura fyzicky nachází. Své obvyklé využití IaaS nachází například při: [22]

- **Vývoji a testování** – týmy mohou lehce sestavit a rozebrat prostředí pro vývoj a testování, takže můžou dodat nové aplikace na trh dříve. IaaS umožňuje rychle a ekonomicky škálovat vývojová a testovací prostředí
- **Hostování webů** – může být levnější než tradiční hosting
- **Storage, zálohování a obnovení** – organizace ušetří na počátečních nákladech spojených s nákupem hardware a zároveň se vyhne problémům se složitostí správy uložiště, která vyžaduje zkušený personál. IaaS je výhodná při řešení nepředvídatelných požadavků a postupného růstu uložiště.
- **Webové aplikace –** IaaS poskytuje veškerou infrastrukturu potřebnou pro chod webových aplikací, včetně všech síťových prostředků, webových a aplikačních serverů a uložišť. Webové aplikace se mohou snadno nasadit a je možné škálovat infrastrukturu podle aktuálních potřeb, které jsou mnohdy nepředvídatelné.
- **Analýza velkých objemů dat –** Dolování dat z velkého objemu dat (big data) za účelem získání potencionálně užitečných vzorců, trendů a asociací vyžaduje obrovské množství výpočetního výkonu. IaaS ho může poskytnout za velmi výhodných ekonomických podmínek.

Výhody IaaS spočívají hlavně v eliminaci investičních nákladů a snižování průběžných výdajů. Je to výhodná ekonomická volba nejen pro různé startupy, ale i společnosti testující nové nápady. IaaS obchází počáteční náklady, které jsou nutné k vytvoření a následné správě vlastního datového centra. Organizace může reagovat rychleji na měnící se podmínky. Infrastrukturu lze škálovat podle potřeb tak, aby byly obslouženy výkyvy v určitém období. Opětovný návrat na původní hodnoty výkonu je snadný. V neposlední řadě IaaS poskytuje lepší zabezpečení. Organizace se nemusí starat o bezpečnost vlastními prostředky, o tu se stará poskytovatel a zodpovídá za ni. [22]

PaaS (Platform as a Service)

Platforma jako služba (PaaS) je úplné prostředí (včetně operačního systému) pro vývoj a nasazení v cloudu. PaaS zahrnuje infrastrukturu (stejně jako IaaS) – servery, uložiště a sítě. Navíc v sobě zahrnuje middleware, operační systémy a celá běhová prostředí. To vše si uživatel pronajímá na základě průběžných plateb. Uživatel spravuje pouze vlastní aplikace a data, o vše ostatní se „stará“ poskytovatel služby. [22]

V praxi PaaS umožňuje vyhnout se složitému a mnohdy rozsáhlému nákupu softwarových licencí, podpůrných aplikací infrastruktury a middlewaru nebo dalších podpůrných nástrojů pro vývoj. Vývojář spravuje pouze vyvíjené aplikace a s tím spojené služby a poskytovatel spravuje všechny ostatní potřebné komponenty a nástroje. [22]

Obvyklé použití platformy jako služby: [22]

- **Vývojová architektura** – PaaS poskytuje veškerou architekturu potřebnou pro vývojáře, kteří na ni staví při vývoji nebo úpravách cloudových aplikací. Všechny nástroje a komponenty potřebné pro vývoj jsou vývojářům dostupné během okamžiku a na jednom místě.
- **Analytické funkce a funkce business intelligence** – poskytované nástroje v modelu PaaS umožňují organizacím dolovat a analyzovat data, hledat vzorce v datech a předvídat budoucí vývoj.

PaaS nabízí stejné výhody jako IaaS, ale jeho přidané možnosti a funkce posunou výhody služby jako celku dále. Díky PaaS je možné zkrátit dobu psaní kódu, a to prostřednictvím dostupných předpřipravených aplikačních komponent. Organizace může rozšířit vývojové možnosti bez nabírání lidí, komponenty nabídnou vývojovému týmu nové možnosti bez potřeby dalších pracovníků se zkušenostmi. Díky PaaS může mít organizace snadně a levně geograficky nesourodé vývojové týmy. Na projektu může pracovat kdokoliv, téměř odkudkoliv. Potřeba je jen dostupné internetové připojení a týmy mohou pracovat společně, i když jednotliví členové nesedí na stejném místě. [22]

SaaS (Software as a Service)

Software jako služba (SaaS) je metoda distribuce softwarových aplikací pomocí veřejné sítě internet. Asi nejobvyklejšími příkladem je e-mail nebo kancelářské nástroje, dostupné skrze Microsoft Office 365.

SaaS poskytuje úplné softwarové řešení od poskytovatele služby. Uživatel si pronajímá možnost použití aplikace a přistupuje k ní přes internet, nejčastěji pomocí webového prohlížeče. Veškerá podpůrná infrastruktura, včetně dat aplikace, je umístěna v datovém centru poskytovatele služby. Poskytovatel služby na základě smlouvy o poskytování služeb spravuje hardware i software, včetně zabezpečení aplikace a dat. V rámci smlouvy bývá specifikovaná i dostupnost. SaaS umožňuje organizaci využívat aplikace s minimálními pořizovacími náklady a téměř okamžitě. [22]

Obvyklé využití SaaS v organizacích spočívá v pronájmu kancelářských aplikací, např. e-mail, balík office nebo i propracované obchodní aplikace jako řízení vztahů se zákazníky (CRM) nebo ERP systémy a aplikace na správu dokumentů. Je to obrovská škála aplikací, která se dá pomocí této služby pronajmout. Za použití těchto aplikací se platí na základě předplatného nebo dle úrovně využití. [22]

Přínosy modelu SaaS: [22]

- Nízké počáteční náklady na pořízení
- Náklady na provoz jsou předem známy a nejsou překvapivě navyšovány
- Možnost redukce dodatečných nákladů
- Platba pouze za to, co skutečně organizace využívá
- Součástí služby je průběžné vylepšování softwaru a jeho podpora
- Přístup k datům aplikace odkudkoliv, stačí připojení k internetu

1.9.2 Modely nasazení

Cloudové služby jsou podle institutu NIST dále děleny podle způsobu implementace: [19]

- **Privátní cloud** – infrastruktura privátního cloudu funguje pouze pro účely konkrétní organizace. Vlastní a spravuje ji přímo organizace nebo třetí strana. Při správě infrastruktury třetí stranou, kdy se využívá i datové centrum poskytovatele, hovoříme o „outsourced private cloud“. Outsourcovaný privátní cloud si organizace pronajímá. Přínosy pro organizaci jsou v úspoře na nákladech spojených s nákupem a údržbou hardware. Pokud organizace využívá vlastní datové centrum, používá se termín „on-site private cloud“. Důvody organizace pro jeho vytvoření jsou v efektivním využití firemního hardware a možnosti nabízet software ve formě služby pro své zaměstnance, dodavatele a zákazníky.
- **Komunitní cloud** – Princip komunitního cloudu spočívá ve sdílení infrastruktury mezi několika organizacemi se stejnými zájmy. Cloud může být hostovaný nebo umístěn v rámci organizace.
- **Veřejný cloud** – tento model cloudu charakterizuje, že je k dispozici široké veřejnosti nebo velké průmyslové skupině. Infrastruktura cloudu je umístěna u poskytovatele. Jedná se o nejběžnější a nejvíce rozšířený typ cloudu.
- **Hybridní cloud** – v hybridním cloudu se spojují dva a více modelů nasazení (privátní, komunitní, veřejný). Jednotlivé typy cloudů zůstávají samostatné, ale jsou propojeny standardizovanou technologií, která umožňuje přenos dat.

2 ANALÝZA SOUČASNÉHO STAVU

Tato část práce podrobně popisuje požadavky na aplikaci a účel, ke kterému bude co nejefektivněji využita. Požadavky na systém jsou rozděleny podle přístupových práv. Stručně zanalyzováno je okolí (vnější prostředí) organizace pomocí SLEPT analýzy, vnitřní faktory organizace pomocí analýzy 7S a pro ucelený pohled na analýzu je využita metoda SWOT. Dílčí částí bude zhodnocení situace v autoškole bez této aplikace – online plánování jízd.

2.1 Informace o autoškole

Autoškola byla založena v roce 1997. Segmentově se řadí spíše do menších autoškol. Má dva majitele a 5 stálých instruktorů. V autoškole působí i několik dalších externích spolupracovníků, kteří obstarávají odborná školení a doplňkové činnosti. Celá organizace má mnohaleté zkušenosti ve výuce a výcviku řidičského oprávnění. Počet studentů, kteří za dobu existence úspěšně prošli výcvikem se řádově pohybuje v jednotkách tisíc. Autoškola má pouze jednu pobočku a působí ve středně velkém městě, kde cílí na studenty z celého okresu. Bližší údaje o organizaci jsou uvedeny i v následujících 3 analýzách.

Autoškola, pro kterou je tento systém zpracován, si nepřeje zveřejňovat své jméno, v textu se bude mluvit obecně o „autoškole“, nicméně systém jako takový je navrhnut univerzálně pro využití ve více různých autoškolách. Rizika, procesy a veškeré ostatní údaje jako podklady této práce budou zpracovány pro tuto danou autoškolu. Veškeré tyto údaje jsou ovšem podobné s více autoškolami, protože celkově proces získání řidičského oprávnění společně s nasazením plánovacího systému jízd je více či méně shodný. Systém se dá reálně nasadit v různých autoškolách.

2.2 SLEPT analýza

SLEPT analýza bývá využívána pro posouzení vnějších faktorů podniku – jeho okolí. Posuzováno je především obecné okolí. SLEPT analýza je přehled faktorů z reálného okolí organizace, skládá se z následujících 5 faktorů:

Sociální

Sociální faktory působí nemalým vlivem na jakýkoliv druh podnikání. Jde např. o sociální úroveň zákazníků na které podnik cílí nebo o trendy aktuálně probíhající ve společnosti. Na autoškolu není tento dopad trendů společnosti nijak zásadně velký. Výběr autoškoly není pro žáka příliš náchylný k výkyvům např. v prestiži či módě. Proces získání řidičského oprávnění by měl být v různých autoškolách stejný. Rozhodující je spíše kvalita služeb a cena.

Legislativní

Z legislativního hlediska je autoškola druh živnosti ohlašovací vázaná. Majitelé vykonávají tuto činnost na základě živnostenského oprávnění. Legislativní faktory se tohoto druhu podnikání silně týkají. V nedávné době např. vstoupila v platnost novela zákona týkající se výcviku řidičského oprávnění skupiny A, které se musí všechny autoškoly poskytující tento výcvik pružně přizpůsobit, což s sebou nese nemalé finanční prostředky navíc. Autoškola musí sledovat novinky v legislativě a bezpodmínečně plnit veškerá znění zákona o autoškolách.

Ekonomické

Organizace je při svém rozhodování do jisté míry ovlivněna vývojem makroekonomických trendů. Míra ekonomického růstu ovlivňuje úspěšnost organizace na trhu tím, že přináší určitý rozsah příležitostí, ale ruku v ruce s tím i hrozby, kterým musí čelit. Makroekonomické ukazatele jako HDP, míra inflace nebo průměrné mzdy mají v České republice příznivý trend, který se v čase vyvíjí dobře. Poměrně silné výkyvy na straně poptávky po službách autoškoly se dají předvídat na základě vývoje populace (tzv. silné a slabé ročníky). Nejvíce zájemců o řidičské oprávnění se k výcviku hlásí těsně před dovršením 18 let.

Politické

Politické faktory se do jisté míry v tomto případě váží k faktorům legislativním. Autoškola není příspěvková organizace a ani nedostává dotace od státu nebo města či kraje. Politická rozhodnutí ji skrze legislativu a různé vyhlášky ovlivňují a nastavují mantinely a rozsah pro její činnost. Podmínky pro podnikatelskou činnost jsou v České

republice na dobré úrovni. Polická rozhodnutí by však měla podnikatelským subjektům činnost zjednodušovat.

Technologické

Technologické faktory a trendy ve vývoji technologií jsou významné pro většinu společností. Přesto, že vývoj ani distribuce není předmětem podnikání autoškoly, může díky nim podnikat efektivněji, a tedy snižovat náklady, což může mít i příznivý vliv např. dopad na životní prostředí. Využití IT technologií uspoří čas, energii a může zjednodušit chod autoškoly. Technologický pokrok v dopravě se může projevit na jejích vozech. Organizace si význam technologických faktorů uvědomuje a věnuje mu patřičnou pozornost, čemuž odpovídá i nasazení aplikace, na kterou je primárně zaměřena tato práce.

2.3 Analýza 7S

Metoda 7S se využívá pro analýzu vnitřních faktorů podniku. Podrobně se zaměřuje na sedm faktorů, které navíc rozděluje do dvou skupin – tvrdé a měkké. Tvrdé faktory obsahují informační systémy, strukturu a strategii podniku. Měkké doplňují analýzu o spolupracovníky, styl, sdílené hodnoty a schopnosti. Měkké faktory jsou neméně důležité, avšak častým problémem je jejich popis, který bývá subjektivní a do jisté míry zkreslený (každý analytik je vidí jiným pohledem). V tomto konkrétním případě bude analýza zjednodušena o to, že daná autoškola není velkého rozsahu – jedná se o menší autoškolu.

Organizační struktura

Jak již bylo uvedeno, autoškola se řadí spíše do segmentu malých autoškol, kde majitelé podnikají na živnostenské oprávnění. Organizační struktura je lineární. V čele stojí majitelé, kteří jsou zároveň i zaměstnanci, a mají pod sebou nezbytný počet lidí, tj. 5 zaměstnanců. Největší výhodou je rychlost rozhodování a jasně určené pozice podřízenosti a nadřízenosti.

Informační systém

Informační systém nevyužívají žádný, nasazením IS se zabývá tato diplomová práce. Implementace se tedy bude provádět na „čisté louce“, což je do jisté míry výhodná pozice.

Strategie

Strategií organizace lze rozumět její hlavní cíle, kterých chce dosáhnout. V souladu s těmito cíli provádí veškerá svá rozhodnutí. Na trhu, na kterém se autoškola pohybuje, je velká konkurence. Autoškoly se navzájem předhánějí, co navíc by potencionálním studentům mohly nabídnout. V dnešní době, kdy jsou studenti zvyklí si plánovat svůj čas „online“, může webová aplikace pro plánování jízd znamenat dobrou konkurenční výhodu před ostatními autoškolami, které tento systém zaveden nemají. Díky velké konkurenci je hlavním strategickým cílem udržet si dosavadní postavení na trhu, a s tím související udržování neklesajících tržeb a zisků.

Spolupracovníci

Jak již bylo uvedeno výše, jedná se o malou organizaci. Spolupracovníků není mnoho. Jedná se o instruktory a externí podpůrné pracovníky pro chod organizace. Důležité je zajistit dostatečné kapacity instruktorů pro výkon jízd a výuky. O chod autoškoly (administrativně apod.) se starají majitelé autoškoly. Externí pracovníci, např. pro zdravovědu, doplňkové odborné přednášky apod., se najímají operativně podle potřeby.

Styl řízení

Rozhodovací pravomoc mají majitelé autoškoly, kteří jsou dva. Veškerá zodpovědnost za rozhodnutí leží na jejich bedrech. Při důležitých rozhodnutích se domlouvají společně s odborníky a dalšími zainteresovanými osobami ve svém okolí, které danému tématu odborně rozumí. Jedná se o menší autoškolu, kde hierarchická posloupnost rozhodování není na místě. Styl řízení i kontrola zůstávají na majitelích.

Schopnosti

Povaha podnikání vyžaduje, aby instruktoři byli detailně seznámeni s legislativou ohledně silničního provozu a byli v tomto ohledu skutečnými odborníky. Při přijímání nového pracovníka se samozřejmě dává velká váha na předchozí zkušenosti v oboru. Všichni zaměstnanci prochází pravidelnými školeními a kurzy, aby měli pro své studenty nebo posluchače na odborných školeních aktuální a správné informace. Hodnota zaměstnanců se velice špatně vyčísluje, ale mají klíčový podíl na úspěšném chodu autoškoly.

Sdílené hodnoty

Obor podnikání autoškoly je jedním z těch, kde nemusí být při správném vedení nijak extrémně složité vytvořit sdílené hodnoty mezi zaměstnanci. Instruktor přispívá svou prací k osvojení si správných zásad při řízení motorového vozidla, a tím obecně přispívá k bezpečnosti silničního provozu, jehož součástí je téměř každý člověk. Jednotliví instruktoři jsou spolu denně v kontaktu při přestávkách mezi jízdami nebo výukou.

2.4 SWOT analýza

	Pozitivní	Negativní/Škodlivé
INTERNÍ	**Silné stránky** **S** 1 Dobrá pověst u veřejnosti 2 Poctivost a kvalita ve výcviku 3 Moderně vybavená učebna 4 Dobrá propagace na sociálních sítích 5 Doplňkové služby na úrovni 6 Výborné reference minulých studentů 7 Spolehlivý vozový park	**Slabé stránky** **W** 1 Malé pokrytí skupin ŘO 2 Nízká spolupráce s partnery 3 Zastaralá technologie a vzhled webové prezentace 4 Nízké investice do segmentově cíleného marketingu
EXTERNÍ	**Příležitosti** **O** 1 Vyšší postavení mezi konkurencí v oboru 2 Efektivnější a vyšší využití informačních technologií 3 Vyšší uplatnění v odborných školeních 4 Začít vyučovat další skupinu ŘO	**Hrozby** **T** 1 Rapidní úbytek zájmu o ŘO 2 Změna v legislativě spolené s autoškolami 3 Špatná strategická rozhodnutí 4 Nárůst konkurence 5 Vysoký nárůst cen PHM

2.5 Současný stav a popis změny

Systém (webová aplikace), která slouží jako podklad této práce, bude sloužit pro pohodlnější, efektivnější a jednodušší komunikaci mezi studentem a autoškolou v procesech plánování jízd a závěrečných zkoušek. Dosavadní fungování autoškoly bez této aplikace bylo nesrovnatelně složitější. Možností bylo více, ale nejčastější byly tyto:

- Studenti se museli osobně dostavit do autoškoly, kde si s pomocí instruktora do papírové knihy jízd naplánovali jízdy na několik lekcí dopředu. V případě nemoci, nedostatku času apod. bylo složité telefonicky jízdy hledat, rušit a přesouvat na další volné termíny, které se musely přizpůsobit oběma stranám.
- Student telefonicky nebo v horším případě e-mailem (nebo jiným způsobem) kontaktoval autoškolu a na základě volných termínů a času obou stran se domluvily lekce jízd. Při rušení či přesouvání jízd se celý proces znovu opakoval.

Z popisu výše vyplývá, že pro obě strany (studenta i autoškolu) byl tento proces časově náročný a nepohodlný. Při jakékoliv změně se proces klidně i několikrát opakoval. Informace o jízdách byly často vedeny na více místech zároveň. Některé byly (v tom lepším případě) v papírové podobě v knize jízd a odtud se přesouvaly do „jednotné verze pravdy“ do tabulky v aplikaci MS Excel. Vznikala tak nekonzistentní data a některé jízdy se „ztrácely“. Obdobná situace nastávala při přihlašování na závěrečné zkoušky. Tento proces byl jednodušší v tom, že závěrečnou zkoušku (v ideálním případě) student absolvuje pouze jednou, zatímco výukových jízd je násobně více.

Další problém představovalo předávání informací o termínech jízd studentovi. Termíny se zapisovaly do papírové podoby karty studenta. Papírová karta studenta byl lístek, kde se zapisovaly naplánované jízdy a student měl tedy přehled o tom, kdy se má na jízdu dostavit. Problém byl nejenom v tom, že se tyto lístky často ztrácely a musely se vyplňovat nové, ale také v tom, že problém způsobovala i jakákoliv jiná změna, například změna nástupního místa. Autoškola musela mít pořád někoho na telefonu, kdo tyto drobné informace zpracovával a předával dále.

Z popisu výše je patrné, že vznikla potřeba uchovávat data na jednom místě (v databázi) a spravovat je skrze aplikační prostředí nejlépe odkudkoliv. Vzhledem k tomu, že chytrý

telefon, tablet, notebook, PC nebo obecně jakékoliv zařízení s připojením na internet má v dnešní době téměř každý člověk je ideálním způsobem, jak tuto potřebu uspokojit webová aplikace. Situace je výhodná pro obě strany, studenta i autoškolu. Student má jasný přehled o svém výcviku a může si ho přizpůsobovat doslova online, podle svých potřeb. Majitelé mohou spravovat chod autoškoly např. pouze ze svého chytrého telefonu. Mohou vypisovat jízdy, přidělovat „práci“ jednotlivým instruktorům, plánovat vytěžování jednotlivých vozidel, získávat přehledy o studentech, instruktorech apod.

2.6 Požadavky na aplikaci

V této kapitole budou shrnuty veškeré požadavky, které musí aplikace splnit pro to, aby mohla být úspěšně nasazena pro efektivní fungování autoškoly. Podrobně budou popsány všechny funkce a systémové souvislosti.

2.6.1 Přístupová práva

Přístupových práv jsou 3 druhy:

- **Student** – Má své přístupové údaje. Přihlašuje se na jízdy a závěrečné zkoušky vypsané administrátorem.
- **Instruktor** – Přihlašuje se na základě svého přihlašovacího jména a hesla. Získává přehledy o svých jízdách.
- **Administrátor** – má na starosti správu celého systému, přístup má ke všem částem a může je téměř neomezeně spravovat (např. může rušit již i uzamčené jízdy nebo přihlašovat studenty nad rámec povoleného limitu). Mimo jiné generuje přihlašovací údaje studentům a instruktorům a spadá pod něj veškerá administrace.

Následující schéma ukazuje, do kterých částí systému má kdo přístup:

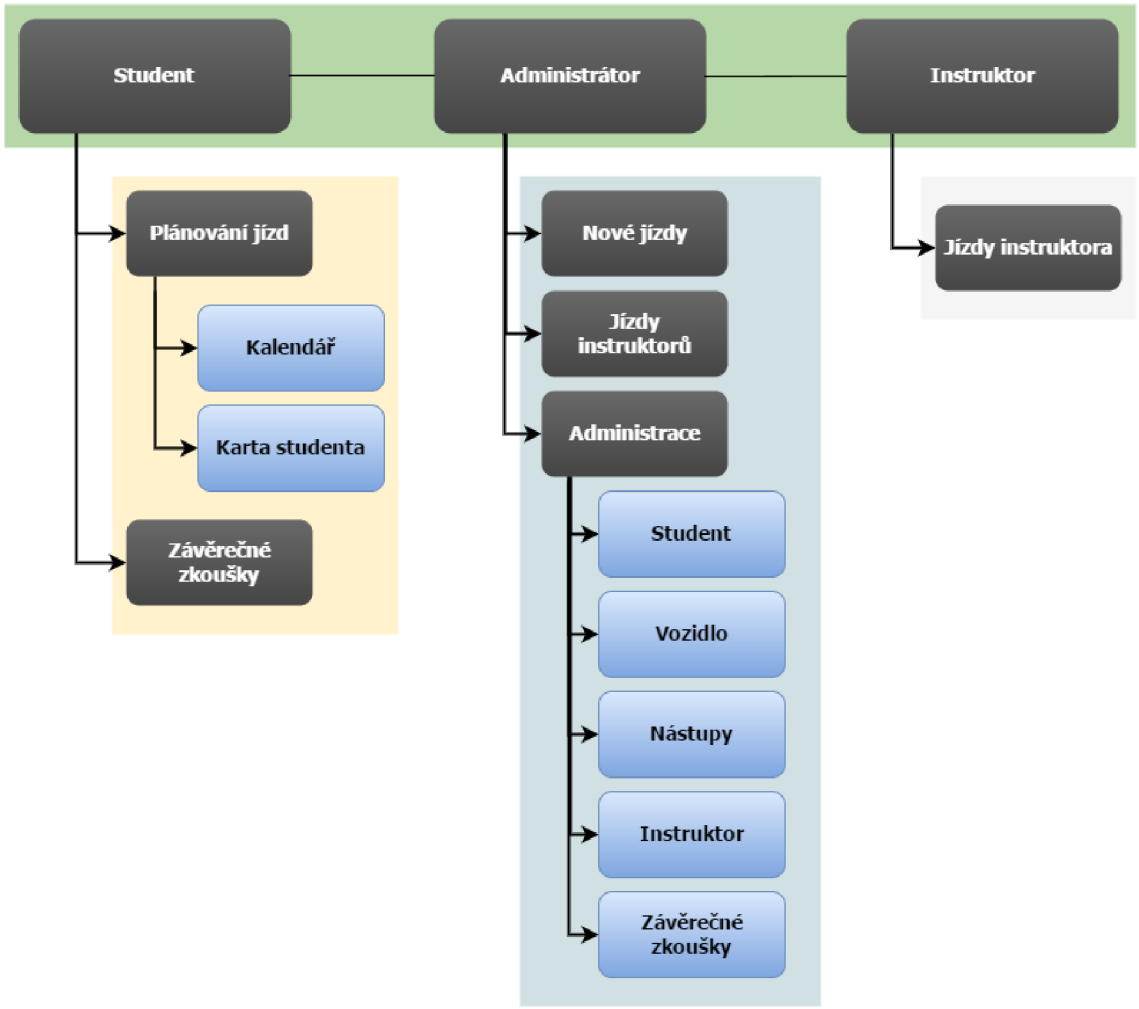


Schéma č. 1 – Přístupová práva (zdroj: vlastní zpracování)

Do aplikace nemá mít přístup nikdo bez přístupových údajů, které spravuje administrátor. Jednotlivé části systému, které jsou uvedeny v grafu výše, budou postupně v následujících podkapitolách detailně analyzovány.

2.6.2 Plánování jízd studentem

Plánování jízd studentem je jednou z hlavních částí systému. I proto, že ji budou využívat studenti s různými znalostmi a schopnostmi, musí být navržena tak, aby její ovládání bylo co nejjednodušší a zároveň naprosto přesné, aby nedocházelo k chybám. Proces plánování jízd musí být detailně zanalyzován.

Souhrn několika základních podmínek pro plánování jízd:

- **V jeden den může mít student pouze jednu jízdu** – tato podmínka platí i v situaci, kdy student absolvuje výcvik na více skupin řidičského oprávnění. Například, provádí-li výcvik na skupiny A+B, v jeden den může absolvovat jízdu pouze buď na motocyklu nebo pouze na automobilu.
- **Jedna jízda trvá 2 vyučovací hodiny** – vyučovací hodina má 45 minut, tzn. jedna jízda trvá 1,5 hodiny.
- **Každá skupina řidičského oprávnění má předepsaný jiný počet hodin výcviku** – student se může přihlásit maximálně na zákonem daný počet jízd. Například při výcviku ŘO skupiny B je zákonem stanovený počet hodin jízd na 28, tj. student se může přihlásit na maximálně 14 jízd (nejsou brány v potaz výjimky, které má k dispozici administrátor systému např. v případě dokoupení kondičních jízd apod., toto téma bude analyzováno v administrátorských funcích systému).
- **Student si může zapsat jízdu jen pro vozidla, na která provádí výcvik dle rozsahu skupin ŘO** – není možné, aby nastala situace, že se student přihlásí na jízdu s vozidlem, na které nedělá řidičské oprávnění. V seznamu vozidel, pro která může zobrazit kalendář, smí být jen ta, na kterých smí absolvovat výcvik dle rozsahu ŘO.
- **Na jeden termín jízdy se může přihlásit pouze jeden student** – systémově nesmí být možné, aby se na jeden termín jízdy přihlásili dva nebo více studentů. Při teoretické možnosti, že se na volný termín v krátkém časovém okamžiku po sobě přihlásí více studentů, bude ostatním (později přihlášeným) termín zamítnut jako obsazený. První vyhrává.
- **Plánování jízd na následující den musí být v určitou stanovenou dobu uzavřeno** – autoškola musí mít na další den jasno, jak budou jednotlivé jízdy probíhat. Z tohoto důvodu se plánování na následující den v určitou hodinu uzamkne a student se již na tento den nemůže přihlásit ani odhlásit z jízdy. Do této doby je možné se libovolně přihlašovat i odhlašovat. V autoškole je tato doba stanovena na 20:00, ale hodina je individuální a každá jednotlivá autoškola ji může mít jinou.

- **Student nemůže mít jízdu ve stejný den jako závěrečnou zkoušku** – poslední jízda může být den před závěrečnou zkouškou

Všechny podmínky týkající se plánování (přihlašování) jízd se musí kontrolovat vícenásobně a naposledy při samotném potvrzování jízdy, těsně před zápisem do databáze. Nesmí nastat situace, že se do ostré databáze dostanou nesprávné informace. Studentovo pracovní rozhraní musí obsahovat minimálně drobné funkčnosti – změnu hesla a odhlášení ze systému.

Kalendář

Jednou z mála změn, které se autoškola musí skrze implementaci podřídit, je nastavení pevného časového rozvrhu dne. Časový rozvrh jízd bude začínat každý den např. v 6:30 ráno (záleží samozřejmě na autoškole, zda v tuto hodinu jízdu vypíše nebo ne). Časový rozvrh se každé autoškole dá přizpůsobit.

Kalendář musí být pro přehlednost uzpůsoben po jednotlivých týdnech. Plánovat jízdy se dá na následujících 30 dní dopředu od aktuálního dne. Posouvat se dopředu i dozadu lze po jednotlivých týdnech. Vzhledově bude kalendář jako školní rozvrh, kdy jedno políčko této matice bude jedna jízda. Na první pohled musí být jasné, která jízda je volná, obsazená anebo „Má jízda“ – jízda aktuálně přihlášeného uživatele. Jiných stavů pole matice nabývat nemůže. Jako obsazená jízda se zobrazí i jízda, která administrátorem nebyla vypsána.

Každé vozidlo bude mít svůj vlastní kalendář jízd. Nad kalendářem bude formulářový prvek rozbalovací nabídka, ze kterého student vybere vozidlo a automaticky se pro něj zobrazí kalendář jízd. To, že u jednoho vozidla je jízda v určitý den a čas obsazená, neznamená, že student nemůže ve stejnou dobu uskutečnit svou jízdu v jiném vozidle. Pokud je jízda volná, může se student přihlásit klikem na toto pole. Po kliku se zobrazí potvrzovací obrazovka s informacemi o jízdě – datum, čas, jméno a příjmení studenta, vozidlo a v rozbalovací nabídce se vybere místo nástupu. Pokud jsou všechny informace v pořádku, po kliku na potvrzovací tlačítko je student na jízdu přihlášen a v kalendáři je

toto pole jednoznačně graficky označeno jako „Má jízda“. Ostatním studentům se toto pole (jízda) bude jevit jako obsazené.
V 20:00, kdy se každý den uzamyká přihlašování (a odhlašování) na jízdy, z kalendáře aktuální den „zmizí“. Skutečnost, zda již není přihlašování na jízdy uzamčeno je nutné kontrolovat i přímo při potvrzování jízdy.

Karta studenta

Karta studenta se pro přehlednost bude nacházet hned vedle kalendáře. Zobrazeny na ní budou základní informace – jméno a skupiny řidičského oprávnění, pro které podstupuje výcvik. První řádek bude pro většinu studentů tím, nejdůležitějším – informace o nejbližší jízdě (skupina ŘO, den, datum, hodina, místo nástupu). Pod tímto oddílem dále následují informace o naplánovaných jízdách. Jízdy budou přehledně řazeny do tabulky podle jednotlivých skupin ŘO. Již odjeté jízdy budou graficky odlišené (např. šedivé), nejbližší jízda v každé skupině bude výrazně probarvená a další, ještě neuskutečněné, jízdy budou normálním stylem písma. Výsledkem bude stav, kdy má student jasný přehled o všech svých jízdách.

2.6.3 Závěrečná zkouška

Během výcviku si student může naplánovat, kdy vykoná (přihlásí se) na závěrečnou zkoušku vypsanou administrátorem. Podmínky pro přihlášení k závěrečné zkoušce:

- **Zkouška již nesmí být plně obsazena** – administrátor vypíše zkoušku pro určitý počet lidí, který nesmí být překročen. Kontrola musí být prováděna opakovaně na více místech a naposledy těsně před zapsáním do databáze.
- **Jeden den = nejvýše jedna závěrečná zkouška** – v jeden den může být student přihlášen pouze na jednu závěrečnou zkoušku
- **Přihlašování na závěrečnou zkoušku musí být v určitý okamžik uzavřeno** – stejně jako u plánování jízd, musí mít autoškola dopředu konečný přehled o následující zkoušce. Rozdíl u závěrečné zkoušky spočívá v delším časovém intervalu, který je nastaven pro uzavření před zkouškou. Nastavení může být jakékoliv, autoškola má standardně 7 dnů před zkouškou již zkoušku „uzavřenou“ a nelze se na ni přihlásit či z ní odhlásit.

- **Nemožnost vícenásobného přihlašování „pro jistotu“ do budoucna** – student se nemůže přihlásit na závěrečnou zkoušku, pokud je již přihlášen na termín, který ještě neproběhl.
- **Do termínu závěrečné zkoušky nutnost odjeté všechny jízdy** – student nemůže mít naplánovanou žádnou jízdu po termínu závěrečné zkoušky. Všechny jízdy musí být do tohoto termínu úspěšně odjety.

Termíny závěrečných zkoušek by se pro přehlednost měly zobrazovat jako jednotlivé „dlaždice“. Její obsah bude obsahovat nejdůležitější informace o zkoušce, detail zkoušky s podrobnějšími informacemi včetně poznámky se zobrazí po otevření. Přihlašování a odhlašování je do uzamčení zkoušky neomezeno. V situaci kdy student otevře podrobnosti o zkoušce, na kterou je přihlášen, musí být nabídka možností jiná než u zkoušky, na kterou přihlášen není. Tzn. postrádá smysl, aby měl student možnost přihlášení se na zkoušku, na kterou již přihlášen je. Po přihlášení na zkoušku bude daná „dlaždice“ graficky odlišena od ostatních, aby bylo na první pohled jasné, že jde o zkoušku aktuálně přihlášeného studenta.

2.6.4 Nové jízdy

Vypisování nových jízd pro studenty je jednou z nejdůležitějších částí celé aplikace. Hlavní požadavek je takový, aby byl tento proces co možná nejjednodušší, přehledně zpracovaný, a přesto přesně splnil veškerou potřebnou funkčnost. Ovládání musí být intuitivní, není v ničím zájmu absolvovat speciální školení pro vypisování jízd. Správné pochopení této funkčnosti zefektivní komunikaci mezi autoškolou a studentem, tudíž zefektivní chod autoškoly. Návrhu celé této funkčnosti je potřeba věnovat náležitou pozornost, protože případná chyba v procesu vypisování jízd, může napáchat velké škody.

Kalendář

Kalendář bude sloužit administrátorovi k vypisování jízd. Vypsání jízdy znamená, že se daný termín objeví studentovi v kalendáři pro plánování jízd jako volný. Stejně jako u části „plánování jízd studentem“ bude mít kalendář maticové uspořádání, kdy jedno pole

= jedna jízda. Jízdy se vypisují pro den a hodinu a zároveň pro dvojici entit – instruktor a vozidlo. Žádná jízda nemůže být vypsána bez toho, aniž by ji „specifikovalo“ vozidlo a instruktor. Každý instruktor nemusí mít oprávnění pro výcvik všech skupin řidičského oprávnění, resp. každý instruktor může jezdit jízdy jen s určitými vozidly na základě skupiny ŘO, které má oprávnění vyučovat.

Při vypisování jízd musí být vybrán instruktor, pro kterého se budou jízdy vypisovat, a vozidlo, se kterým bude jízda odjeta. Vozidla budou filtrována na základě skupin ŘO instruktora. *Příklad: instruktor 1 vyučuje jízdy jen pro skupinu ŘO A (a všechny její podskupiny), tj. všechny motocykly. Při výběru tohoto instruktora se automaticky vyfiltrují všechna vozidla, která může instruktor „využít“ pro jízdy – všechny motocykly.* Při výběru vozidla musí být za jeho názvem zobrazena skupina ŘO, pro kterou je vozidlo primárně určeno. Ve vztahu instruktor – vozidlo platí vazba M:N. Jeden instruktor může využívat různá vozidla, a zároveň jedno vozidlo může být využito různými instruktory (resp. všemi instruktory s danou skupinou ŘO).

V kalendáři musí být na první pohled jasné, kdy je možné jízdu vypsat a kdy již je jízda vypsána (vhodné grafické zpracování). Termín pro vypsání jízdy se v kalendáři jeví jako volný pouze tehdy, je-li pro daný termín volný jak vybraný instruktor, tak vozidlo. Obecně nesmí nastat situace, že by se dala administrátorem vypsat jízda, která by z nějakého důvodu (nedostupnost instruktora nebo vozidla, instruktor nemá požadovanou skupinu ŘO pro výcvik) nemohla být uskutečněna.

Vypisovat jízdy se bude dát na 5 týdnů dopředu včetně aktuálního dne. Kalendář se pro přehlednost bude zobrazovat po jednotlivých týdnech. Pro dny bude zobrazena i jejich zkratka, ne pouze datum. Minulé dny v aktuálním týdnu budou pouze mizet (není logické vypisovat jízdy do minulosti). Při vypsání jízdy na aktuální den se na tuto jízdu již student sám přes své rozhraní přihlásit nemůže (přihlašování na jízdy se pro studenta uzamyká v 20:00 předchozí den). Tuto problematiku řeší administrátorská funkce „Přihlásit studenta na jízdu“, která bude popsána níže.

Pro vybranou dvojici – instruktor a vozidlo – musí být možné (z důvodu uživatelské přívětivosti) v daném týdnu vypsat najednou neomezený počet jízd (tzn. vypisování jízd neprobíhá po 1 jízdě). Všechny informace o vypsaných jízdách musí být v jednom společném kalendáři. Tady nastává situace, kdy v daný termín má vybraný instruktor volno, ale s vybraným vozidlem má již vypsanou jízdu jiný instruktor. V takovém případě se pole v kalendáři „tváří" jako obsazený termín pro vypsání jízdy. Na první pohled by administrátorovi nebylo jasné, z jakého důvodu pro tuto kombinaci instruktor – vozidlo nemůže jízdu vypsat. Tuto problematiku řeší následující podkapitola „Vytíženost instruktorů a vozidel".

Pro každou vypsanou jízdu musí administrátor mít tyto funkce přímo v kalendáři:

- **Přihlásit studenta na jízdu** – pomocí této funkce může administrátor zapsat studenta na libovolnou volnou jízdu (i na jízdu, která má proběhnout aktuální den). Stejně tak pomocí této funkčnosti administrátor může zapsat studenta na jízdu nad rámec povinného počtu jízd stanoveného zákonem, který musí splnit (např. při dokoupení kondičních jízd, pokud studentovi výcvik nestačil apod.). Po vybrání vhodného termínu by se administrátorovi měla zobrazit obrazovka s potřebnými informacemi o dané jízdě (datum, čas, instruktor, vozidlo), kde z rozbalovacích nabídek vybere studenta a místo nástupu.
- **Zrušit jízdu** – vypsaná jízda může být administrátorem zrušena pouze tehdy, pokud na ní není žádný student přihlášen (tzn. je vypsaná, ale neobsazená).

Kalendář musí poskytovat i informace o tom, zda je již daná vypsaná jízda obsazena některým studentem. Pokud ano, bude zde uvedeno jméno studenta. Tyto informace musí být, stejně jako všechny ostatní, vhodně graficky zpracovány. Pro jednotlivé grafické symboly bude na vhodném místě zpracována jednoduchá legenda.

Vytíženost instruktorů a vozidel

Vytíženost vozidel a instruktorů tvoří dvě samostatné tabulky (jedna pro aktuálně vybraného instruktora, druhá pro aktuálně vybrané vozidlo), které budou hned pod kalendářem pro vypisování jízd. Tyto tabulky budou mít stejnou formu jako kalendář (tj.

týdenní rozvrh hodin) s tím rozdílem, že budou pouze co nejpřehledněji poskytovat informace o vytíženosti daného instruktora (vozidla). Na první pohled musí být jasné, kdy v týdnu má instruktor (vozidlo) volno a kdy je již obsazen (tj. vypsán pro jízdu). Pokud se v kalendáři pro vypisování jízd změní instruktor (nebo vozidlo), tato změna se projeví i pro tabulky vytíženost instruktorů a vozidel. Stejná situace nastane v případě, že se v kalendáři posuneme o týden dopředu nebo vzad. Na základě těchto dvou tabulek (časových rozvrhů) bude mít administrátor jasný a jednoduchý přehled o aktuální vytíženosti instruktorů a vozidel v daném týdnu a na jejich základě může vypisovat další jízdy.

2.6.5 Jízdy instruktorů

Funkčnost „Jízdy instruktorů“ bude sloužit pro kompletní přehled o naplánovaných jízdách, které musí instruktor odjet. V rozevíracím seznamu budou všichni instruktoři seřazení dle abecedy. Jako výchozí se zobrazí informace pro instruktora, který je nejblíže začátku abecedy. Při výběru instruktora se pod sebou v tabulkách zobrazí jízdy na dva dny dopředu včetně aktuálního dne. Tabulka bude obsahovat nezbytně nutné informace: čas, jméno a příjmení studenta, místo nástupu, vozidlo včetně uvedené skupiny ŘO.

Vzhledem k tomu, že plánování jízd se pro studenty uzamyká každý den ve 20:00, můžeme od této doby brát jízdy na další den jako závazně přihlášené. Tyto závazně přihlášené jízdy se v tabulce pro následující den vhodně graficky odliší (například symbolem na konci řádku). U každé tabulky bude datum a název instruktora + ikona pro export do formátu PDF pro tisk. V tomto PDF dokumentu, který bude sloužit jako podklad pro jednotlivé instruktory pro informaci kdy mají jaké jízdy odjet, bude navíc číslo telefonu studenta. Po uzamčení jízd na další den, může administrátor tento PDF dokument vytisknout a předat instruktorovi.

Každý instruktor má své přihlašovací údaje do systému a má k těmto informacím (ovšem pouze o svých jízdách) přístup odkudkoliv. Tento PDF dokument si může vytisknout sám. Druhou možností je kontrolovat svůj „časový rozvrh“ v průběhu dne online, např. pomocí chytrého telefonu.

Tabulky s jízdami na další dva dny dopředu (jedná se o dva dny dopředu, kdy má instruktor naplánovány nějaké jízdy, ne kalendářní dny – instruktor nemusí mít jízdy každý den) jsou pouze informativní, protože jízdy ještě nejsou uzamčeny. Student se ještě může z jízdy odhlásit a jízda se tak uvolní pro studenta jiného. Při kliku na jméno daného studenta v tabulce se zobrazí přehled všech jeho naplánovaných jízd (viz část Správa studentů).

Díky této funkcionalitě má administrátor kompletní přehled nad všemi jízdami instruktorů přehledně na jednom místě.

2.6.6 Správa studentů

Správa studentů tvoří část modulu Administrace. Ze všech ostatních částí tohoto modulu se jedná o část nejobsáhlejší a nejvíce využívanou. Správě studentů a jejím jednotlivým funkčnostem je potřeba při analýze věnovat náležitou pozornost.

Přidat nového studenta

Správným přidáním nového studenta do systému celý proces výcviku začíná. Formulář musí mít označeny povinná pole (jméno, příjmení, evidenční číslo a skupiny ŘO pro výcvik). Ostatní pole (jako např. e-mail nebo telefon) budou dobrovolná, ale bude vhodné je, pro větší pohodlí studenta i autoškoly, vyplnit. Při vyplnění e-mailu studentovi dojde po přidání do systému na tuto adresu zpráva s uvítacími informacemi, a hlavně přihlašovacími údaji.

Proces přidání studenta do systému je navržen následovně. Administrátor vyplní formulář (minimálně všechna povinná pole) a systém automaticky pro tohoto studenta vygeneruje přihlašovací údaje. Heslo bude generováno pseudonáhodně. Jako přihlašovací jméno bude možno využít evidenční číslo nebo e-mail. Evidenční číslo musí být v systému unikátní. Vygenerované přihlašovací údaje administrátor předá studentovi. Student si ve svém rozhraní může libovolně heslo změnit.

U ostatních zadaných hodnot se bude kontrolovat pouze délka zadaného řetězce (maximální i minimální) s výjimkou pole e-mail, kde se bude kontrolovat i správný

formát. Určování skupin výcviku bude probíhat skrze rozevírací seznamy – není potřeba kontrolovat formát zadané hodnoty, musí být ovšem splněna podmínka zadání alespoň 1 skupiny řidičského oprávnění.

Seznam studentů

Tato část musí obsahovat všechny informace o studentech v tabulkovém uspořádání. Jeden řádek tabulky = informace o jednom studentovi. Administrátor bude takto mít dostupné všechny informace o studentech na jednom místě. Zobrazováno bude jméno, příjmení, e-mail, telefon, skupiny řidičského oprávnění, datum registrace do systému a poznámka. Z důvodu vyšší přehlednosti a uživatelské přívětivosti bude tabulka vhodně graficky zpracována a doplněna o dodatečné funkce (např. při najetí na daný řádek nastane jeho probarvení). U každého studenta bude tato funkčnost:

- **Smazat studenta** – Vymazání studenta z aktivního používání systému po ukončení výcviku. Po smazání student zmizí ze seznamu studentů, ale není vymazán úplně. Přesune se v databázi do jiné tabulky – bývalých studentů, pro statistické účely. Smazání studenta musí předcházet potvrzovací okno, zda administrátor skutečně chce studenta smazat.
- **Upravit studenta** – Možnost upravit zadané informace o studentovi. Funkčnost je určená pro jakoukoliv změnu či překlep v zadaných hodnotách.
- **Vygenerovat nové heslo** – V případě ztráty či zapomenutí hesla studentem může administrátor vygenerovat heslo nové. Heslo generuje systém pseudonáhodně. V okamžiku vygenerování nového hesla staré heslo okamžitě pozbývá platnosti. Student se do aplikace přihlásí na základě svého loginu a nového hesla. Vygenerování nového hesla musí předcházet potvrzovací okno, zda skutečně administrátor chce heslo vygenerovat.
- **Přehled jízd studenta** – v přehledu jízd studenta se budou zobrazovat všechny naplánované jízdy (odjeté i neodjeté) daného studenta. Jízdy musí být setříděny podle jednotlivých skupin řidičského oprávnění, pro které student vykonává výcvik. O jednotlivých jízdách budou zobrazeny tyto informace: pořadí jízdy, datum, čas, instruktor, vozidlo a místo nástupu. Řazeny budou vzestupně podle čísla jízdy. V tomto přehledu má administrátor důležitou funkci – může kteroukoliv jízdu zrušit (smazat). Toto rozhodnutí může provést kdykoliv i zpětně.

Stejně jako u všech ostatních přehledů a částí systému je důležité vhodné grafické zpracování a uživatelská přívětivost. Zrušit studentovi naplánovanou jízdu může z několika důvodů:

1. Student se na naplánovanou jízdu nedostavil – z důvodu vysoké administrativní zátěže potvrzování jednotlivých odjetých jízd, bylo od tohoto záměru upuštěno. Každá jízda, která není zrušena, se v systému bere jako odjetá. V případě, že student naplánovanou jízdu z jakéhokoliv důvodu neodjede, musí mu ji administrátor v tomto přehledu smazat.
2. Student zjistil, až po uzavření plánování jízd na další den, že se na plánovanou jízdu nemůže dostavit – v tomto případě student kontaktuje administrátora, a ten jízdu v systému v tomto přehledu zruší. Pokud je již na naplánovanou jízdu jakýkoliv student přihlášen, administrátor tuto jízdu v kalendáři pro vypisování nových jízd zrušit nemůže (jízda je v kalendáři uzamčena).
3. Administrátor ve své funkčnosti „přihlášení studenta na jízdu“ omylem vybral špatného studenta. Tomuto studentovi musí omylem přihlášenou jízdu zrušit.

Díky tomuto přehledu, ve spojení s kalendářem pro vypisování nových jízd, je administrátor absolutním správcem veškeré agendy, která se týká jízd.

2.6.7 Evidence vozidel

Evidence vozidel bude sloužit ke správě vozového parku autoškoly. Vozidlu, které bude zaneseno do systému, bude možné vypsat jízdu. Při přidávání nového vozidla je mimo jiné nutné určit, pro kterou skupinu řidičského oprávnění bude vozidlo primárně určeno. To ovšem neznamená, že nemůže být využito pro výcvik jiné skupiny ŘO (typicky vyšší podskupiny). Příkladem může být skupina A, kdy student bez jakékoliv předchozí zkušenosti s motocyklem podstupuje výcvik na nejsilnější, výkonově neomezený motocykl. Vzhledem ke svým nízkým zkušenostem by chtěl první jízdu absolvovat na menším, snadněji ovladatelném motocyklu. V těchto případech musí studentovi jízdu přihlásit administrátor pomocí funkčnosti přihlásit studenta na jízdu. Student se sám na jízdu jiné skupiny ŘO přihlásit nemůže, protože má přístup pouze ke kalendáři vozidel,

pro která absolvuje výcvik. Studentovi se tato jízda objeví v kartě studenta zařazena do správné skupiny ŘO.

Po přidání se vozidlo objeví v seznamu vozidel se všemi potřebnými informacemi jako např. značka, model, skupina řidičského oprávnění, datum zanesení do systému apod. Každé vozidlo je možné vyřadit nebo upravit zadané hodnoty včetně skupiny ŘO. Při vyřazování vozidla ze systému se musí kontrolovat situace, zda dané vozidlo ještě nemá někdy v budoucnu naplánované jízdy nebo není k některému termínu přiřazeno. V takovém případě vozidlo být vyřazeno nemůže a systém na to upozorní.

2.6.8 Evidence nástupních míst

Evidenci nástupních míst bude tvořit číselník v tabulkovém uspořádání. Jediným jeho úkolem bude mít přehledně na jednom místě seznam všech míst nástupu, které si student může vybrat při přihlašování na jízdu. Na tomto místě se potkají s instruktorem a jízda začne. Místo nástupu bude moci administrátor přidat, smazat nebo upravit. Při přidávání je pouze potřeba zkontrolovat zadané hodnoty, především délku řetězce. Místo nástupu by měl vystihovat jasný, krátký, všem srozumitelný název.

2.6.9 Správa instruktorů

Skrze správu instruktorů bude mít administrátor ucelený přehled o instruktorech autoškoly. Každému instruktorovi, který je zaveden v systému, mohou být vypsány jízdy.

Přidat nového instruktora

Při přidávání nového instruktora do systému budou povinná tyto pole: jméno, příjmení, evidenční číslo a skupiny řidičského oprávnění. Je nutné si uvědomit, že instruktor může jezdit jízdy pro více skupin řidičského oprávnění. Ne každý instruktor má oprávnění pouze na jednu skupinu. Při vybírání skupin ŘO tedy musí být na výběr ze všech dostupných skupin. Evidenční číslo musí být v systému unikátní, protože bude zároveň instruktorovi sloužit jako přihlašovací login do aplikace.

Pokud administrátor správně vyplní všechny potřebné údaje, uloží instruktora do databáze a objeví se mu okno s loginem a heslem pro přístup instruktora do systému. Tyto informace předá danému instruktorovi a od této doby bude mít přístup do aplikace s právy instruktora. Heslo je systémem generováno pseudonáhodně, instruktor si ho může kdykoliv po přihlášení změnit.

Seznam instruktorů

Seznam instruktorů bude administrátorovi sloužit pro přehled nad všemi instruktory v systému. V seznamu se budou zobrazovat nejpotřebnější informace pro rychlou orientaci. Přímo zde bude možné každého jednotlivého instruktora smazat, upravit nebo mu vygenerovat nové heslo.

Smazat instruktora ze systému bude možné pouze tehdy, nemá-li naplánovány žádné jízdy nebo není-li uveden u žádného vypsaného termínu. Smazání musí předcházet potvrzovací obrazovka, zda opravdu administrátor chce tento krok provést. V případě zapomenutí hesla instruktorem se zde bude dát vygenerovat heslo nové. Staré heslo pozbývá platnosti okamžitě v okamžiku vygenerování hesla nového. Vygenerování nového hesla musí předcházet potvrzovací obrazovka. U instruktora bude možné, pomocí tlačítka „Editovat“, upravit všechny hodnoty kromě evidenčního čísla. Tato funkčnost je v aplikaci především z důvodu možnosti přidat či ubrat skupiny řidičského oprávnění, které může instruktor vyučovat. Okrajově se ale mohou objevit i případné změny v ostatních hodnotách. Při každé provedené akci je vhodné, aby administrátor dostal informaci o provedené změně (např. úprava, smazání).

2.6.10 Správa závěrečných zkoušek

Správa závěrečných zkoušek bude další z významných funkčností systému. Administrátorovi bude sloužit pro vypisování nových termínů závěrečných zkoušek a kompletní správu již vypsaných termínů.

Vypsání nové závěrečné zkoušky

Nejdůležitější a povinné údaje pro vypsání nového termínu závěrečné zkoušky jsou datum, čas a počet míst (tj. kolik studentů se na termín zkoušky může maximálně

přihlásit). Ostatní pole jako poznámka apod. povinná nejsou a budou sloužit pouze jenom jako doplňkové informace ke zkoušce. Všechny vstupní hodnoty je potřeba kontrolovat a popřípadě převést na správný formát pro uložení do databáze. V okamžiku vypsání termínu zkoušky se termín objeví i studentům v části pro přihlašování na závěrečné zkoušky. V této chvíli se studenti mohou na daný termín přihlásit.

Editace závěrečných zkoušek

Administrátor musí mít jasný přehled o všech termínech a všech přihlášených studentech na tyto termíny závěrečných zkoušek. Musí být na první pohled patrné, jak je který termín obsazen. Při kliku na jméno studenta se zobrazí všechny odjeté a naplánované jízdy (tj. administrátorská funkčnost – Přehled jízd studenta). Každého studenta bude možné kdykoliv z termínu závěrečné zkoušky odhlásit, a to i tehdy, pokud již je pro studenty termín uzavřen. Při tomto kroku by měl student automaticky tuto informaci dostat v podobě e-mailu (za předpokladu, že e-mailová adresa bude u studenta vyplněna). Stejně tak bude možné libovolného studenta kdykoliv a na kterýkoliv termín přihlásit (tj. i tehdy, je-li již přihlašování pro studenty uzamčeno, a dokonce i nad povolený rámec počtu míst daného termínu). Přihlašování studenta na zkoušku administrátorem bude co možná nejsnazší. Pro daný termín se z rozbalovacího seznamu vybere daný student a volba se uloží.

Zrušit vypsaný termín závěrečné zkoušky bude možné jen tehdy, pokud na něj není přihlášen žádný student. Všechny tyto funkčnosti bude, pro uživatelskou přívětivost a rychlost obsluhy, možné využít bez nutnosti otevření podrobností konkrétního termínu. Po otevření detailu zkoušky budou veškeré tyto funkčnosti k dispozici. K dispozici navíc budou i všechny ostatní (méně důležité) podrobnosti o zkoušce.

2.6.11 Statistiky

V rámci administrace bude mít administrátor možnost zobrazit statistiky – dodatečné informace o aplikaci, respektive o autoškole. Vzhledem k tomu, že databáze bude při ostrém provozu aplikace naplněna, z různých úhlů pohledu, mnoha zajímavými daty, byla by škoda, kdyby alespoň některé informace neměl administrátor k dispozici. Jde například o:

- počet studentů, kteří úspěšně ukončili výcvik
- celkové počty naplánovaných jízd pro jednotlivá vozidla a instruktory
- celkové počty odjetých jízd pro jednotlivá vozidla a instruktory
- přehledy bývalých studentů
- přehledy kondičních a doplňkových jízd (jízd nad rámec povinného výcviku)
- úspěšnost u zkoušek, průměrné počty opakování apod.
- ostatní doplňkové informace jako např. průměrná délka trvání výcviku, nejčastější (a naopak nejméně využívané) termíny jízd přihlašované studenty, … atd.

Informace budou moci být zobrazeny dle jednotlivých let a měsíců, a zároveň i souhrnně, tj. za celou dobu fungování aplikace. Patrně budou tyto statistiky dávat smysl až po nějaké době využívání aplikace. V začátcích může autoškola tyto informace srovnávat se svými interními daty. Statistiky mohou majitelé využít pro efektivnější řízení celé autoškoly.

2.7 Uvažovaná hotová řešení

Autoškola vyžaduje systém, který bude splňovat všechny její požadavky, bude mít jednoduché intuitivní ovládání s možností přizpůsobení jejím potřebám. Nabídka těchto systémů pro plánování jízd, závěrečných zkoušek a k tomu potřebné agendy pro správu autoškoly je malá. Jednoznačně byly autoškolou zavrženy volně dostupné obecné rezervační systémy. I přesto, že se tyto rezervační systémy dají do jisté míry přizpůsobit, pro efektivní provoz autoškoly jsou absolutně nevhodné. Pro autoškolu by nasazení podobného systému znamenalo spíše přítěž než zefektivnění svého chodu.
Dále budou představeny dva systémy. Jako uvažované substituční řešení jsou blízko (alespoň v některých parametrech) požadavkům autoškoly. Popsány budou i jejich výhody a nevýhody.

Moderní autoškola

Systém nazvaný Moderní autoškola je určen pro plánování jízd v autoškolách. Nabízí základní agendu s nápovědou pro ovládání. Studentům umožňuje spravovat si své jízdy samostatně a přihlašovat se na administrátorem vypsané termíny jízd. Systém spravuje společnost Weby, s. r. o. Výhodou je poměrně jednoduché a přehledné ovládání bez

zbytečných funkcionalit, které se nevyužijí. Poskytovatel systému garantuje spolehlivost, bezpečnost a možnost off-line provozu při výpadku služeb.

Velká nevýhoda spočívá v absenci více skupin řidičského oprávnění. Aplikace nemá sjednoceny ani dvě základní skupiny (a jejich podskupiny) ŘO, tj. A a B. Tzn. student, který zároveň absolvuje výcvik pro sk. A a B, musí jízdy přihlašovat na dvou místech (každá ze skupin má svůj kalendář a přihlašovací údaje). Další nevýhodou je absence vhodně zpracované agendy závěrečných zkoušek. Pro autoškolu je cena za implementaci a provoz tohoto systému zbytečně vysoká. Veškerá přizpůsobení systému jsou navíc účtována zvlášť.

Moje autoškola

Systém Moje autoškola je nové robustní řešení pro kompletní správu autoškoly. Aplikace pokrývá veškerou agendu od vyplnění evidence teorie, přes správu plateb, vozového parku až po výsledné reporty. Provozovatel systému je společnost NEPLUS, s.r.o. Systém je primárně určen velkým autoškolám s mnoha vozidly a více pobočkami, nicméně může být nasazen i v menší autoškole. Nabízí opravdu širokou škálu funkčností, která uspokojí náročného uživatele. Nasazen může být lokálně nebo i v cloudu. Oproti předchozímu systému se navíc neplatí žádné poplatky za implementaci, pouze za provoz.

Pro autoškolu není systém vhodný z několika důvodů. Autoškola chce především plánování jízd a závěrečných zkoušek pro studenty. Tento systém tuto funkčnost nenabízí ani v základní ani v rozšířené verzi. Musí se doprogramovat na vyžádání. Systém je určen pro kompletní správu autoškoly z hlediska administrátora, popřípadě učitelů (omezeně). Rozhraní pro studenty je potlačeno, čímž nenaplňuje základní podmínku autoškoly. Student může pouze kontrolovat svůj výcvik tak, jak mu ho administrátor zahájil a vede. Jízdy plánuje pouze administrátor. Administrace systému vyžaduje, díky své robustnosti, důkladné školení, ovládání je pro autoškolu poměrně složité. Systém nabízí spoustu funkčností, které autoškola nevyužije (nepotřebuje).

Platby probíhají na základě využívání aplikace, tj. za využívání jednotlivých funkčností dle ceníku. Základní balíčky jsou osekané a teoreticky se dají využít pro opravdu malé

autoškoly. Výběr parametrů závisí na individuálních požadavcích každé autoškoly, nicméně s tím i rapidně roste cena za službu.

2.8 Shrnutí analýzy

Z analýzy vyplývá, že autoškola požaduje systém, který dokáže bezpečně obsloužit její agendu především směrem ke studentům. Zároveň je jedním z požadavků to, aby byl systém poskytován stejně jako cloudová služba PaaS. Jediné, co bude autoškola pro přistup a správu systému potřebovat je jakékoliv zařízení s webovým prohlížečem. Autoškola v současné době nepožaduje implementovat téměř žádnou (z pohledu autoškoly) detailnější pokročilou funkcionalitu (rozšíření jednotlivých modulů), jako např. tankování, platby, teoretický výcvik apod. Nicméně do budoucna bude systém na tyto případné rozšíření připraven. Důraz musí být kladen na intuitivní a přehledně graficky zpracované uživatelské prostředí.

Z uvažovaných hotových řešení požadavky nesplňuje ani jeden ze systémů. Je nutné navrhnout, vytvořit a implementovat do praxe novou aplikaci, která požadované procesy uspokojí a bude přispívat efektivnímu chodu autoškoly. Licence aplikace nebude omezena z hlediska počtu uživatelů, vozidel, instruktorů apod. Po implementaci bude mít administrátor v tomto ohledu volnou ruku a může spravovat tuto agendu autoškoly podle svých potřeb.

Vzhledem k tomu, že do aplikace nebude mít přístup nikdo bez přihlašovacích údajů, bude sloužit poměrně malému počtu lidí (nepředpokládá se návštěvnost v řádu stovek uživatelů denně). I tak by ale měla aplikace (a především databáze) tento nápor v případě potřeby zvládnout.

3 VLASTNÍ NÁVRH ŘEŠENÍ

Tato část se zabývá návrhem webové aplikace pro potřeby autoškoly. Systém bude tvořen na základě analýzy požadavků v předchozí části práce.

3.1 Souhrnný pohled na aplikaci

Vzhledem k požadavkům bude aplikace vyvíjena v jazyce PHP s využitím databázového systému MySQL a přidružených technologií jako jazyk HTML, JavaScript (a jeho knihovna jQuery), CSS, apod. Technologie jsou vybrány vzhledem k dostupnosti a celkové rozšířenosti mezi vývojáři. Na serveru, na kterém bude celá aplikace spuštěna, jsou tyto technologie nainstalované.

Aplikace je navržena pro tři druhy přístupových práv:

- Student
- Administrátor
- Instruktor

Každou agendu spojenou s danými přístupovými právy je nutné řešit zvlášť a z pohledu uživatele, který ji bude obsluhovat nebo využívat. Každá agenda systému a její součásti dle přístupových práv (viz Schéma č. 1 – Přístupová práva) je i s ukázkami řešení nastíněna v následujících kapitolách této části práce. Ukázky řešení jsou vzhledem ke svému velkému rozsahu v „ostrém“ nasazení podstatně zjednodušeny, což ovšem nemá zásadní vliv na jejich funkčnost. Důležitá je ukázka logiky daného řešení, která se následně dá převést do libovolného programovacího jazyka. Případné ukázky kódu (psané v jazyce PHP) jsou očištěny o veškeré bezpečnostní prvky kódu, syntaktické pasáže sloužící jako podpora vývojového jazyka či napojení na jinou část programovaného systému a všechny ostatní kusy kódu, které jsou nutné pouze pro nasazení v reálném provozu. Cílem ukázek je nastínit logiku řešení, ne představit obrovské množství „živého“ kódu.

Vstup do aplikace tvoří přihlašovací okno. Na základě zadaných údajů systém pozná, o jakého uživatele jde a zobrazí agendy, které může obsluhovat (např. u studenta se

automaticky zobrazí kalendář jízd, administrátorovi kalendář pro vypisování jízd). Odkaz na tuto stránku s přihlášením si může autoškola dát na své webové stránky. Odtud se potom studenti mohou se svými přihlašovacími údaji do aplikace přihlásit.

Aplikace pro autoškolu využívá jednu databázi složenou z více databázových tabulek. Vzhledem k vytěžování databáze je nutné vhodně a správně navrhnout datový model systému. Největší a nejvíce vytěžované budou tabulky ohledně plánování termínů jízd a jízd samotných. Ostatní tabulky jsou pro funkčnost systému neméně důležité, ale takto vytěžované nebudou.

Ovládání aplikace je pro studenta naprosto intuitivní. Navíc ho při případných nejasnostech provází nápověda, kde je např. specifikováno co jednotlivé symboly znamenají, kdy se jízdy uzavírají apod. U administrátora je situace obdobná. Při práci ho mj. doprovází výstižná legenda funkčnosti obrázků a symbolů.

3.2 Nastavení serveru a php.ini

Server, na kterém bude aplikace spuštěna má tyto parametry:

- **Operační systém**: Debian Linux 7
- **Procesor**: Intel® Xeon ™ CPU 3.40GHz, 4 jádra
- **RAM paměť**: 8 GB
- **Pevný disk**: 1 TB
- **Verze MySQL**: 5.5.43
- **Verze PHP**: 5.6

Tyto parametry jsou naprosto dostačující pro bezproblémový chod aplikace.

Před započetím programování skriptů aplikace je nutné správně nastavit PHP. K tomuto účelu slouží konfigurační soubor *php.ini*. Můžeme v něm nastavit volby jazyka, zpracování dat, chyb, sessions apod. Tvar zadávaných příkazů je v konfiguračním souboru php.ini stále stejný a to takový:

Parametr = hodnota

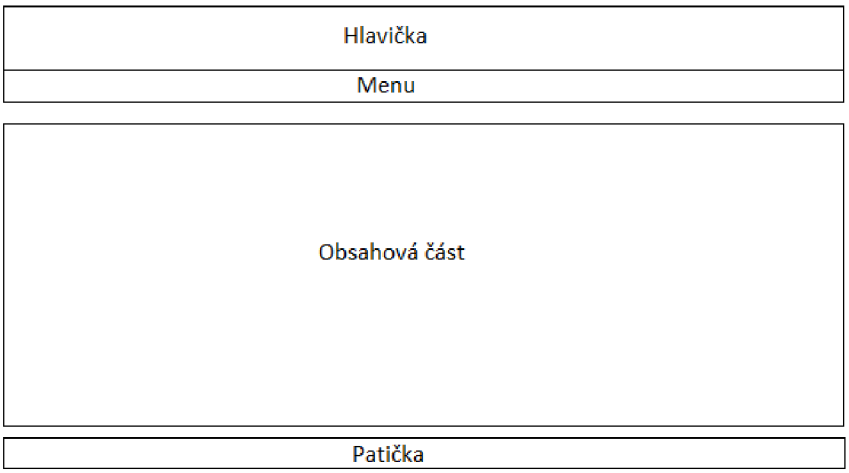
Zde je pro příklad uvedeno pár důležitých voleb nastavení:

- Max_execution_time = 30 – čas v sekundách po který se může skript maximálně vykonávat. Není nutné nastavení vyšších hodnot. V aplikaci se nebudou nahrávat ani zpracovávat extrémně velké soubory či skripty.
- Memory_limit = 64M – určuje kolik paměti smí php skript maximálně zabrat
- Display_errors = 0 – vypisování chybových hlášení na PHP např. při chybách ve skriptu. V tomto případě je volba vypnutá. Výhodné je ji zapnout například při testování. V ostrém provozu může být tato volba zneužita potencionálním útočníkem.
- Short_open_tag = 1 – PHP kód nemusí začínat značkou *<?php* stačí *<?*
- Session.auto_start = 1 – Superglobální proměnné session se nemusí ručně „startovat“

Možností nastavení je samozřejmě podstatně více a záleží na každém vývojáři a jeho zkušenostech, jak si konfigurační soubor nastaví. Obecně lze říci, že všechny hodnoty v nastavení *php.ini* je lepší mírně nadsadit, než je poddimenzovat.

3.3 Grafický návrh

Grafický návrh je při vývoji aplikace jednou z důležitých součástí. Správně navržené grafické rozložení usnadní orientaci uživatele při práci v aplikaci. Rozvržení je pro jednotlivé agendy a uživatelské role (na základě přístupových práv) rozdílné. Nejobecněji by se dalo zobrazit takto:



Obrázek č. 9 – Obecný layout (zdroj: vlastní zpracování)

Obsahová část se v různých agendách podle potřeby dělí na více menších částí (v CSS blokových elementů *div*). Nejlépe to jde vidět na konkrétních příkladech. Zde je například zobrazeno rozhraní studenta pro plánování jízd:

Obrázek č. 10 – Grafický návrh rozložení rozhraní studenta (zdroj: vlastní zpracování)

Položky ve vrchním menu se zvýrazní po najetí myši. Tato funkčnost má omezené využití na jiných zařízeních, která se myší neovládají. Barvy a grafika políček, v kalendáři pro plánování jízd studenta byly zvoleny tak, aby na první pohled evokovaly svůj účel:

- Červená – obsazeno
- Zelená – volno
- Modrá – má jízda

Při grafickém návrhu byl ve všech agendách brán v prvé řadě zřetel na jednoduchost, přehlednost a funkčnost. V celé aplikaci jsou využity dva způsoby rozložení položek:

- **Dlaždice** – využívá se v rozložení administrace, závěrečných zkoušek a obecně tam kde není předpoklad většího počtu položek. Tento způsob rozložení je více přehledný a pro uživatele přívětivější.
- **Seznam** – využívá se v přehledech naplánovaných jízd, studentů, vozidel a obecně tam kde je předpoklad většího počtu položek. V tomto rozložení jsou vždy jednotlivé řádky ob jeden zvýrazněny jinou barvou a při najetí na danou položku se odliší jinou barvou i ona. Využito je vhodné velikosti písma a šířky sloupce. To vše napomáhá přehlednosti, která ani u tohoto typu rozložení není výrazně špatná.

3.4 Přihlašování do aplikace

Do aplikace nemá přístup nikdo bez přihlašovacích údajů. Po správném vyplnění dvou základních formulářových polí – loginu a hesla, se nastaví práva přístupu pro daného uživatele – administrátor, instruktor nebo student.

Proces přihlašování a celkově autentizace v systému funguje na základě tzv. superglobálních proměnných *$_SESSION[]*. Po vložení přihlašovacích údajů skript „ošetří“ zadané vstupní hodnoty a vůči databázi zkontroluje, zda se v ní daný uživatel nachází. Pokud ne, upozorní uživatele na chybu. Pokud ano, z databáze zjistíme jeho unikátní identifikátor a přiřadíme ho do *$_SESSION[id]*. Následně pomocí funkce *header(„Location:…“)* přesměrujeme uživatele, na základě jeho práv, na vhodnou agendu aplikace.

V souboru *func.php* jsou nadefinovány všechny funkce a globální proměnné, které se vkládají ke každému skriptu aplikace. Výhoda spočívá v tom, že se funkce (nebo proměnná) změní na jednom místě a změna se projeví v celé aplikaci. V tomto souboru jsou nadefinovány funkce pro kontrolu, zda se jedná o administrátora, studenta nebo instruktora. Pro kontrolu studenta funkce vypadá takto:

```
function isStudent(){
   if (!isset($_SESSION["id"])) return false;
   $query_log = mysqli_query($mysqli, "select id from uzivatele where
                    opravneni = 0 and id=".$_SESSION["id"]."");
   if(mysqli_num_rows($query_log) == 1) {return true;} else {return false;}
  }
```

Obdobně vypadají i funkce pro ostatní uživatele aplikace. Na začátku každého skriptu se pouze odkontroluje, zavoláním dané funkce (např. *if(isStudent())*), jaký uživatel k němu chce přistupovat.

Tento způsob přihlašování by se dal napadnout útokem typu Session hijacking. Nebudou zde uvedeny všechny bezpečnostní prvky týkající se přihlašování, jen pro příklad:

- Superglobální proměnné Session mají nastavenou určitou dobu platnosti, po které vyprší (doba není zbytečně dlouhá). Po zavření prohlížeče vyprší platnost Session automaticky.
- Heslo se do databáze neukládá v čisté formě. Ukládá se pouze jeho otisk – hash se solí. Při přihlašování se nejprve zjistí hash zadaného hesla (textového řetězce) a následně se oba hashe (z databáze a od uživatele) porovnávají. Výhoda spočívá v tom, že při případném vniknutí útočníka do databáze (nebo zachycení komunikace) není možné získat heslo v čisté formě, ale pouze jeho hash. Nevýhoda tohoto postupu spočívá v situaci, kdy uživatel heslo zapomene. Jeho původní znění se nedá zjistit – musí se v systému generovat heslo nové.

3.5 Rozhraní administrátora

Tato kapitola se věnuje agendám aplikace, které jsou určeny pro správu administrátorem autoškoly. Žádný jiný uživatel bez administrátorských práv nemá k následujícím částem systému přístup.

3.5.1 Vypisování nových jízd

Vypisování nových jízd se skládá ze dvou částí – Kalendáře a Vytíženosti instruktorů a vozidel. Obě části jsou poměrně detailně popsány v analýze. Návrh této agendy je také pro obě části uveden zvlášť, nicméně přímo spolu souvisí.

Kalendář

Vypsat jízdu je možné pouze pro dvojici instruktor a vozidlo. Nad samotným kalendářem jsou formulářové prvky – vysouvací menu (select box). Oba select boxy je nutné z databáze naplnit hodnotami – instruktory a vozidly. Druhý select box pro vozidla je závislý na hodnotě v prvním select boxu. Při výběru instruktora se zobrazí pouze vozidla, se kterými může jezdit jízdy na základě skupin řidičského oprávnění. Při jakékoliv změně select boxu je formulář pomocí javascriptové funkce *onchange* automaticky odeslán. Předávání parametrů formuláře probíhá pomocí metody GET. Formulář obsahuje dvě „skrytá“ pole typu *hidden*, v kterých se ukládají hodnoty ohledně aktuální pozice v kalendáři.

Listování v kalendáři probíhá po týdnech a vypočítává se od aktuálního data. Při přechodu z prvního listu kalendáře se vždy přepočítává o kolik dní se musí jít dopředu, aby byl zobrazen celý následující týden. Stejně tak při přechodu zpět se vždy musí zobrazit správný počet dnů. Všechny parametry se předávají v URL pomocí metody GET.

Kalendář je tvořen jednou velkou maticí formulářových prvků – zaškrtávacích políček (checkbox). Checkboxy jsou pomocí CSS stylů upraveny pro snadnější ovládání i na menších obrazovkách. Každé políčko je jednoznačně identifikovatelné pomocí dvojice hodnot datum aktuálního dne a denní hodina, ukládaných skrze parametr *value* HTML tagu *input*. Hodnota denní hodina je číslo pořadí vyučovací hodiny daného dne. Tato

dvojice hodnot je nesmírně důležitá pro přiřazení plánovaného termínu na správné místo v kalendáři. Problémem je, že obě hodnoty jsou součástí jednoho textového řetězce. Před uložením do databáze se musí textový řetězec pomocí konstrukce *foreach* a funkce *explode()* správně rozdělit a uložit hodnoty odděleně.

Algoritmus, který správně vyplňuje jednotlivá pole kalendáře je v kódovém zápisu dlouhý a náročný na popis (bude popsán pouze zevrubně). Zjednodušeně funguje takto:

- První řádek matice tvoří časy jednotlivých jízd (6:30-8:00 apod.) vytažené z databáze. Další řádky tvoří dny daného týdne, které se procházejí, pomocí několika vnořených cyklů, po buňkách (denních hodinách).
- První buňku řádku (s výjimkou prvního řádku) tvoří zkratka dne a jeho datum. Ostatní buňky daného řádku (dne) jsou určeny pro plánování nebo zobrazení jízd.
- Pomocí funkce *date()* a jejích parametrů se zjistí číslo pořadí aktuálního dne a uloží do proměnné *dnes*. Cyklus *while* s podmínkou na začátku testuje proměnou *dnes* zda je menší nebo rovna 7 (7 dnů v týdnu, 7 = neděle). Při splnění podmínky načteme z databáze (spojením 4 tabulek – planovane_terminy, uzivatele, vozidlo, instruktor) všechny údaje o plánovaných jízdách na tento konkrétní den na základě aktuálně zvoleného instruktora a vozidla (hodnoty zjistíme z asociativního pole *$_GET*). Datum se vypočítává přičtením správného počtu dnů k hodnotě získané z pole *$_GET[]* – hodnota ukládaná v URL při listování v kalendáři. Dotaz do databáze vypadá zjednodušeně takto (nesmírně důležité je správné nastavení podmínek v části za klauzolí WHERE - musí být vybrány jízdy, které v daný termín má buď instruktor nebo vozidlo):

```
SELECT pt.id, pt.id_std, pt.volno, pt.den_hod, pt.datum, u.jmeno as jmeno_std, u.prijmeni as
prijmeni_std, v.znacka, v.model, i.prijmeni
FROM planovane_terminy pt left join uzivatele u ON pt.id_studenta = u.id inner join vozidlo v
ON v.id=pt.vozidlo inner join instruktor i ON i.id=pt.instruktor
WHERE (pt.vozidlo=".$voz." OR pt.instruktor=".$inst." AND pt.datum = '".sqlfor(date("d.m.Y",
strtotime(+$poc_dnu.' day'))).'"
GROUP BY pt.denni_hodina
ORDER BY pt.datum, pt.denni_hodina
```

- Nyní se pomocí cyklu *for* s předem známým počtem opakování (počet denních hodin) prochází jednotlivé buňky s krokem 1 pro daný den. Průchody se počítají. Při každém průchodu jsou načteny aktuální hodnoty z databáze pro plánovaný termín jízdy. Jeden průchod tímto cyklem vypadá typicky takto:
    - Zjistí se, zda je daný den již nějaký plánovaný termín jízdy naplánován. Pokud ne, pomocí cyklu *while* se daný řádek (den) vyplní buňkami s checkboxy a termíny jsou volné pro plánování. Pokud ano, musíme zjistit kterou denní hodinu a před tento termín doplnit vhodný počet checkboxů (volných termínů).
    - V jeden den samozřejmě může být naplánováno libovolný počet termínů jízd. Počet volných termínů se musí dopočítávat po obsazení každé buňky naplánovaným termínem. Při obsazení všech buněk (denních hodin) plánovanými termíny jízd, projde cyklus svůj maximální počet iterací, který může nastat – tj. počet denních hodin.
    - Každý naplánovaný termín se musí kontrolovat, zda již není obsazen studentem, který se na něj přihlásil. Pro nižší zatěžování databáze má každý naplánovaný termín v DB tabulce atribut *volno* a *id_studenta*. Atribut *volno* se při obsazení nastaví na 0, výchozí hodnota je 1. Atribut *id_studenta* se při obsazení přepíše unikátním identifikátorem daného studenta. Kontrola obsazenosti termínu probíhá na základě testování těchto dvou hodnot, zda jsou ve výchozím nastavení. Pokud nejsou, termín je obsazen. Tyto hodnoty se vytěží společně se všemi ostatními na začátku každé iterace cyklu *for*.
    - Termín (buňka), který je pro daného instruktora nebo vozidlo naplánován neobsahuje checkbox, ale obrázek ilustrující vypsanou jízdu s možnostmi přihlásit studenta na jízdu, smazat naplánovaný termín nebo v případě obsazení naplánovaného termínu studentem, informaci o obsazenosti – zámek. Po najetí na zámek se zobrazí jméno studenta a při najetí na obrázek jízdy i jméno instruktora a název vozidla určených pro tuto jízdu.
    - Při projetí cyklu jsou obslouženy všechny možnosti a podmínky plánovaných termínů jízd, které pro daný den mohou nastat. Počítadlo dnů se zvedne o jedna a celý cyklus je spuštěn od začátku.

- Při prvním nesplnění podmínky prvního cyklu *while*, je proces vypisování kalendáře ukončen. Celý týden je pro kombinaci instruktora a vozidla zobrazen.

Kalendář administrátora pro vypisování nových jízd vypadá následovně:

Obrázek č. 11 – Kalendář pro vypisování jízd (zdroj: vlastní zpracování)

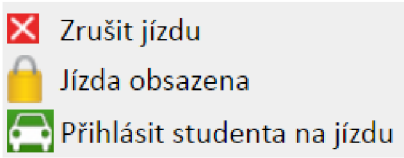
Při stisku tlačítka uložit (uložení termínu jízd do databáze) může administrátor omylem stisknout tlačítko dvakrát nebo i úmyslně opakovaně v situaci, kdy například komunikace

se serverem (databází) trvá delší dobu. V tomto případě se do databáze mohou uložit hodnoty o plánovaných termínech vícenásobně. Této situaci se předchází pomocí jednoduché javascriptové funkce:

```
function Validace(f) {
        f.ulozit.disabled = true;
        return true;
}
```

Při odeslání formuláře tuto funkci zavoláme v parametru *onsubmit="return Validace(this)*. Funkce po dobu odesílání hodnot znepřístupní tlačítko Uložit. Není možné hodnoty odeslat opakovaně. Po odeslání se tlačítko opět zpřístupní.

Symboly v kalendáři označují funkce, které administrátor s naplánovanými jízdami může provádět:



Obrázek č. 12 – Popis symbolů v kalendáři (zdroj: vlastní zpracování)

Funkce **zrušit jízdu** vymaže SQL příkazem DELETE záznam o plánovaném termínu jízdy z databáze, čímž se z termínu stane nevypsaný termín (v políčku kalendáře se objeví checkbox) a je možné ho znovu vypsat. Funkcí **přihlásit studenta na jízdu** se přepneme do samostatného okna kde se, v případě, že termín není obsazen, zobrazí veškeré informace o daném termínu jízdy. Administrátor má na výběr dva select boxy. Jeden, ve kterém jsou z databáze vypsáni všichni studenti (se svým evidenčním číslem a skupinami ŘO) a druhý, ve kterém jsou na výběr všechna místa nástupu. Jediné dvě kontroly, které při tomto přihlašování na jízdu probíhají (na rozdíl od přihlašování studentem na jízdu) jsou tyto:

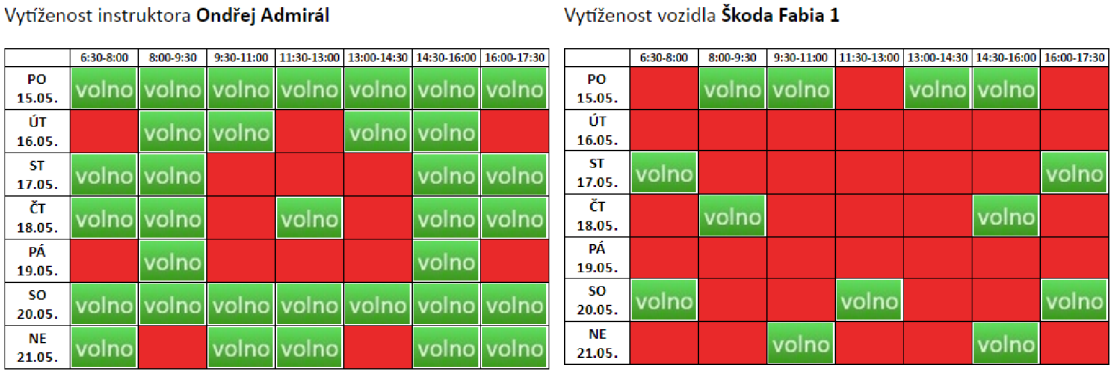
- Kontrola, zda není termín již obsazen – všechny vypsané termíny jsou automaticky viditelné jako volné i pro všechny studenty. Kdokoliv a kdykoliv se

na termín může přihlásit, a tedy je tuto možnost na více místech nutné neustále kontrolovat. Na jeden termín může být přihlášen pouze jeden student.

- Kontrola, zda student, vybraný ze select boxu, již v daný den nemá naplánovanou jízdu s jakýmkoliv vozidlem.

Vytíženost instruktorů a vozidel

Vytíženost instruktorů a vozidel funguje na podobném algoritmu jako kalendář pro vypisování jízd. Obojí jsou stejné matice (kalendáře), u vytíženosti menší a jinak graficky zpracované. Funkčnost je u vytíženosti zjednodušena o několik kontrol. Spokojíme se pouze se dvěma stavy – volno a obsazeno – pro instruktora i vozidlo, kteří jsou aktuálně vybraní v select boxech nad kalendářem (pro tuto dvojici se aktuálně vypisují jízdy).



Obrázek č. 13 – Vytíženost instruktora a vozidla (zdroj: vlastní zpracování)

Z obrázku výše je patrné, že instruktor může být jinak vytížený než vozidlo. S jedním vozidlem může jezdit více instruktorů. V kalendáři pro vypisování nových jízd se jako volné (nevypsané) termíny jeví pouze průnik volných termínů obou entit – dvojice instruktor a vozidlo. Nelze vypsat plánovaný termín jízdy v termínu, kdy má instruktor volno, ale vozidlo nikoliv a naopak.

3.5.2 Jízdy instruktorů

Jednotliví instruktoři, uložení v databázi, jsou seřazeni v select boxu. Při výběru instruktora se automaticky zobrazí jeho jízdy se studenty na nejbližší dny. Hodnoty z tohoto formulářového prvku se předávají pomocí metody GET – jsou viditelné (a

editovatelné) v URL. Stejně jako všude je nutné tyto hodnoty na více místech „ošetřit“ a kontrolovat. Například při vstupu do SQL dotazu se využívá funkce *mysqli_real_escape_string()*. V případě vložení nesprávných hodnot do URL, skript nemůže skončit chybou. Uživatel pouze dostane informaci o neexistenci daného instruktora nebo jiného parametru.

Na základě unikátního identifikátoru, který se ze select boxu předává, získáme z databáze informace o instruktorovi a datech jeho nejbližších jízd. Pokud tyto informace máme, využijeme je v SQL dotazu, kterým zjistíme veškeré podrobnější informace o jízdách. Spojuje se 6 databázových tabulek poiuze s využitím *inner join*, tj. spojení na rovnost. Požadujeme pouze hodnoty, které sobě navzájem odpovídají. Ekvivalentní způsob zápisu je pouze s využitím operátoru =.

```
SELECT
c.hodina, v.znacka, v.model, u.id as id_std, u.jmeno, u.prijmeni, mn.misto, sro.skupina
FROM
casy c, vozidlo v, uzivatele u, misto_nastupu mn, jizda j, skupiny_ro sro
WHERE
j.id_instruktor=".$inst." AND j.denni_hodina = c.denni_hodina AND j.id_vozidlo = v.id AND
u.id = j.id_student AND mn.id = j.misto_nastupu AND j.datum='".$date["datum"]."' AND
sro.id=v.skupina_ro
ORDER BY j.denni_hodina
```

Následně se všechny získané hodnoty vhodně zobrazí. Vyřešit je potřeba „zamykání jízd“. V přehledu se musí zobrazit, kterou jízdu již nelze studentem odregistrovat, a tedy je uzamčena. Tohoto stavu lze docílit vhodnou kombinací podmínek s využitím funkcí *date()* a *strtotime()*. Zámek se u jízdy zobrazí pouze tehdy, nastane-li jedna z možností:

- Aktuální den zvýšený o 1 roven dnu jízdy a zároveň aktuální čas větší než čas uzamčení jízdy
- Aktuální den je roven času jízdy

Přehled jízd pro vybraného instruktora může vypadat takto:

Instruktor: Admirál Ondřej ▾

Ondřej Admirál 4.5.2017

Čas	Jméno a příjmení	Místo nástupu	Vozidlo	
6:30-8:00	Josef Hruška	Autoškola	Škoda Fabia 1 - sk. B	
8:00-9:30	Barbora Vančurová	Hlavní nádraží	Škoda Fabia 1 - sk. B	
9:30-11:00	Slavomír Ordoš	Gymnázium PR	Yamaha R1 - sk. A	
11:30-13:00	Petr Novák	Gymnázium PR	Honda CBR 125 - sk. A1	
14:30-16:00	Pavel Malý	Autoškola	Škoda Fabia 1 - sk. B	
16:00-17:30	Jan Svoboda	Hlavní nádraží	Škoda Fabia 1 - sk. B	

Obrázek č. 14 – Přehled jízd instruktora (zdroj: vlastní zpracování)

Vedle jména instruktora je symbol pro export přehledu jízd do formátu PDF. Export je řešen pomocí PHP knihovny MPDF, jejích tříd a metod. Knihovna musí být někde na serveru uložena, na začátku skriptu se inicializuje. Zjednodušeně lze říci, že s využitím metod knihovny si vytvoříme výsledný PDF dokument pomocí jazyka HTML a kaskádových stylů. Při práci s knihovnou lze samozřejmě využít i jazyk PHP a ostatní přidružené technologie. Potřeba je ovšem dbát na lehce odlišnou syntaxi.

Přehled jízd v exportovaném PDF dokumentu má lehce odlišnou strukturu, resp. nachází se v něm více informací (např. telefon na studenta apod.).

3.5.3 Správa studentů

Správa studentů je jednou z částí administrace, které je věnováno následujících 5 podkapitol. Administrace je nejvíce škálovatelnou součástí aplikace. Záleží na autoškole, do jaké hloubky chce jednotlivé agendy obsáhnout.

Přidat nového studenta

Přidávání nového studenta do systému je řešeno vyplněním formuláře s potřebnými informacemi o studentovi. Všechna pole jsou (v případě) vyplnění kontrolována zejména na délku a formát, který se očekává. Povinná pole jsou označena a bez jejich vyplnění

nelze studenta do systému přidat. Jsou to tyto: jméno, příjmení, evidenční číslo a skupina(y) řidičského oprávnění.

Po správném vyplnění formuláře se zadané hodnoty zkontrolují. Pseudonáhodně se vygeneruje heslo, které se při ukládání do databáze společně s ostatními hodnotami uvede jako jeden z parametrů hashovací funkce. Heslo v čisté formě je uloženo v proměnné, kterou následně po řádném uložení hodnot do databáze společně s přihlašovacím loginem zobrazíme v okně. Tyto informace má možnost administrátor vytisknout a následně předat studentovi. V tuto chvíli může student aplikaci plně využívat. V případě ztráty hesla musí administrátor přes své rozhraní danému studentovi vygenerovat heslo nové. Z databáze ho již nelze vyčíst.

Zároveň při uložení hodnot do databáze je, v případě vyplnění e-mailové adresy, studentovi odeslán e-mail s informací o jeho registraci a adresou, kde se do aplikace přihlásit.

Seznam studentů

Přehled všech studentů je řešen pomocí tabulkového zobrazení. V tabulce jsou základní informace. Při najetí na řádek tabulky je řádek zvýrazněn pomocí vhodného nastylování funkcí *onmouseover()* a *onmouseout()*. Při kliku na studenta se otevře karta s jeho podrobnějšími informacemi. U každého studenta v tomto základním přehledu jsou 4 rychle dostupné funkce:

- **Smazat studenta** – před odstraněním studenta z „živé“ databáze se provede několik kroků. V prvé řadě kontrola (dotazy do databáze), zda student ještě nemá naplánovány nějaké jízdy nebo závěrečnou zkoušku. V případě, že ano, odstranit ho nelze. Následně na základě unikátního identifikátoru studenta, získáme z databáze všechny jeho osobní údaje, které uložíme do tabulky bývalých studentů. Tato tabulka slouží pouze pro statistické účely. V posledním kroku je student, pomocí SQL příkazu DELETE, z tabulky aktivních studentů odstraněn.
- **Upravit** – formulář pro úpravu studenta má stejná pole jako pro přidávání, vyjma pole evidenční číslo, které se upravovat nedá. Všechny hodnoty formulářových polí musíme „vytáhnout“ z databáze a vložit pro úpravu (uživatel musí vědět, co

upravuje). K tomuto slouží u polí typu *text* parametr *value*. U polí typu vysouvací menu k tomu slouží parametr *selected*. U každé hodnoty vysouvacího menu musíme testovat, zda se aktuální hodnota z databáze rovná položce v menu. Pokud ano, parametr *selected* způsobí, že bude vybrána jako výchozí. Po úpravě a uložení do databáze se všechny informace administrátorovi zobrazí v samostatné kartě. Upravené položky jsou zvýrazněny.

- **Vygenerovat heslo** – Z pochopitelných důvodů, zde nebude uveden algoritmus, jakým je heslo generováno. Nové heslo je převedeno na jeho hash a uloženo do databáze. V případě, že má student uvedenu e-mailovou adresu, přijde mu automaticky zpráva (funkce *mail()*). Po úspěšném uložení daného studenta do databáze jsou administrátorovi na obrazovku zobrazeny nové přihlašovací údaje.
- **Přehled jízd** – Nejprve musíme zjistit pro kolik skupin ŘO student koná výcvik. Tuto informaci zjistíme z dekompoziční tabulky *uzivatel/skupina_ro* na základě výskytu počtu ID daného studenta. Podle počtu skupin ŘO se, spojením vhodných tabulek, zobrazí jednotlivé jízdy roztříděné podle těchto skupin a seřazené od nejstarší po nejnovější. U každé jízdy je pro přehlednost uvedeno její pořadové číslo. V SQL dotazech je využito faktu, že všechny podskupiny daných skupin ŘO začínají stejným písmenem jako hlavní skupina (např. hlavní skupina – A, podskupiny – AM, A1, A2). Jízdy do skupin se tedy filtrují propojením tabulek *skupiny_ro* a *uzivatel/skupina_ro* pomocí operátoru *like*. U každé jízdy má administrátor možnost jejího nevratného vymazání. Při vymazání jízdy, která se má uskutečnit v budoucnosti, se tento termín nadále v kalendáři jeví jako volný. Může se na něj přihlásit jakýkoliv jiný (i stejný) student.

3.5.4 Evidence vozidel

U přidávání vozidel do systému jsou povinná tyto tři pole: značka, model a skupina ŘO, pro kterou je dané vozidlo primárně určeno. Student má na výběr pouze ta vozidla, která odpovídají jeho skupině řidičského oprávnění. Administrátor může přihlásit studenta i na jízdu s vozidlem, které neodpovídá přímo jeho skupině ŘO (např. v začátcích výcviku na velký motocykl, může student začít na motocyklu menším).

V přehledu všech vozidel o nich nalezneme podrobnější informace s dvěma možnostmi – Upravit a Smazat vozidlo. Autoškola v současné době nechce agendu vozidel hlouběji rozpracovat, vystačí se základními funkcemi pro chod aplikace. V budoucnosti se počítá s implementací dodatečné, podrobnější funkčnosti. Např. správa tankování, servisu, STK apod.

Před případným odstraněním vozidla musí proběhnout kontrola, zda vozidlo nemá naplánovanou jízdu, která ještě nebyla odjeta:

```
SELECT id
FROM jizda
WHERE id_vozidlo = ".$id_voz." AND datum >= curdate()
```

Pokud dotaz vrátí alespoň jedno ID jízdy, vozidlo nelze smazat. Stejně tak proběhne kontrola, zda vozidlo není přiděleno k nějakému naplánovanému termínu jízdy (ještě neobsazený studentem). Pokud kontrolami projdeme, vozidlo může být ze systému odstraněno.

Při editaci vozidel je nutné všechny položky z databáze vypsat do správných formulářových prvků. Povinná pole musí být, stejně jako při přidávání vozidel, odkontrolována. Všechny změny provedené administrátorem jsou uloženy, s využitím příkazu *UPDATE*, do databáze a okamžitě se v systému projeví. Provedené změny se, společně s informací o úspěchu, administrátorovi zobrazí v aktuálním okně.

3.5.5 Evidence nástupních míst

Místa nástupu tvoří v databázi jednu samostatnou tabulku. V aplikaci se dají jednoduše a neomezeně přidávat, mazat nebo upravovat. V systému jsou uspořádány v tabulkovém zobrazení podle abecedy. Jednotlivé řádky jsou ob jeden řádek barevně odlišeny. Využito je jednoduchého, efektivního algoritmu. Záznamy jsou z databáze vypisovány pomocí cyklu *while*. Před začátkem cyklu je hodnota kontrolní proměnné nastavena na *true*. U

každého vypisovaného řádku se testuje kontrolní proměnná na svou logickou hodnotu, pokud projde, řádek se probarví.

```
<tr <?php if ($control) echo "bgcolor=\"#c0c0c0\" " ?> … ">
```

Na konci každé iterace cyklu se kontrolní proměnná zneguje. Tímto mechanismem se jednoduše dosáhne požadovaného cíle, kdy řádky tabulky střídají svou barvu.

Při případném odstraňování míst nástupu probíhá kontrola, zda zde nemá některý ze studentů naplánovaný začátek své jízdy. Kontrola probíhá stejně jako např. u vozidel, s tím rozdílem, že zde se testují jen naplánované jízdy studenty. Plánované termíny jízd, které nejsou obsazeny, u sebe žádné místo nástupu uvedeno nemají – studentem může být vybráno jakékoliv.

3.5.6 Správa instruktorů

Správa instruktorů se skládá ze dvou důležitých částí – přidávání nových instruktorů a správu stávajících. Při přidávání instruktorů jsou nejdůležitější (a zároveň povinné) tyto hodnoty: jméno a příjmení, evidenční číslo a skupiny ŘO. Evidenční číslo je pro každého instruktora unikátní položka a musí být před uložením kontrolována, zda se již stejná v systému nenachází. Skupin řidičského oprávnění, pro které může instruktor jezdit jízdy, může být více (ne každý instruktor jezdí pouze jednu skupinu ŘO). Tento výběr je řešen pomocí zaškrtávacích políček (check boxů). Jednotlivé skupiny se ukládají do samostatné databázové tabulky společně s unikátním identifikátorem instruktora (dekompozice relační vazby M:N). Resp. do dekompoziční tabulky se ukládají pouze primární klíče obou hodnot. Každá skupina ŘO má svůj unikátní identifikátor (primární klíč), stejně tak instruktor.

Po kontrole všech zadaných hodnot je vygenerováno heslo a jeho otisk společně s ostatními hodnotami uložen do databáze. Skupiny ŘO jsou do samostatné tabulky ukládány s využitím konstrukce *foreach*, která projde všechny hodnoty pole od prvního indexu po poslední. Opakovaně se tak provádí SQL příkaz *INSERT*. Po úspěšném uložení

je tato informace zobrazena administrátorovi společně s přihlašovacími údaji pro daného instruktora. Přihlašovací jméno tvoří evidenční číslo a heslo je naposledy zobrazeno v čisté formě.

V přehledu všech instruktorů jsou zobrazeny základní informace + volby pro správu instruktorů. Po kliku na instruktora se zobrazí karta s podrobnějšími informacemi. Rychlé funkce u instruktora:

- **Smazat** – smazat instruktora (znemožnit jeho zapisování na jízdy) je možné pouze tehdy, nemá-li naplánovanou jízdu nebo není-li přiřazen k některému z naplánovaných termínů do budoucna. Kontroly jsou prováděny stejně jako u Evidence vozidel. Pokud kontrolami neprojde, je důvod vypsán na obrazovku, pokud projde, instruktor je z aktivní databáze odstraněn a nelze ho vypsat na další plánovaný termín jízd.
- **Upravit** – při úpravě je nutné do formulářových prvků správně zasadit hodnoty z databáze. Z tohoto pohledu jsou nejzajímavěji řešeny již zmiňované check boxy. Nejprve si z dekompoziční tabulky *instruktor_skupina_ro* do asociativního pole *$skupiny_instruktora[]*, pomocí cyklu *while*, uložíme všechny skupiny ŘO pro daného instruktora. Následně se pomocí dalšího cyklu *while* vypisují všechny skupiny ŘO z databázové tabulky *skupiny_ro* jako formulářové prvky checkbox. V rámci každého jednotlivého check boxu se, pomocí konstrukce *foreach*, prochází celé pole *$skupiny_instruktora[]* a testuje se, zda se skupina z pole nerovná skupině z databáze. V případě rovnosti se check box jeví jako zaškrtnutý – využívá se vlastnosti *checked*, která určuje výchozí stav políčka. Ukládání upravených hodnot probíhá velice podobně jako při přidávání instruktorů.
- **Vygenerovat nové heslo** – funkčnost je logicky stejná jako u Správy studentů.

3.5.7 Správa závěrečných zkoušek

Vypsání nové zkoušky

Administrátor musí při vypisování nových závěrečných zkoušek vyplnit tři povinná pole: datum, čas a počet míst (kolik studentů se může na zkoušku maximálně přihlásit). Ostatní pole jsou dobrovolná. U data a času probíhají kontroly na správnost formátu pomocí

předem nadefinovaných funkcí. Počet míst musí být numerická hodnota, pro tento účel slouží funkce *is_numeric()*.

Případná poznámka může obsahovat maximálně 300 znaků. Aktuální počet napsaných znaků se zobrazuje přímo vedle pole pomocí javascriptové funkce *DelkaTextu()* s využitím ovladače událostí *onKeyUp*. Funkce vypadá následovně:

```
function DelkaTextu(){
document.getElementById("delkaPoznamka").innerHTML =
document.getElementById("poznamka").value.length;
}
```

Funkce funguje na jednoduchém principu. Při uvolnění klávesy zjistí délku aktuálně napsaného řetězce v elementu *delkaPoznamka*, což je textová oblast pro poznámku.

Po úspěšném uložení všech hodnot do databáze je tato informace zobrazena administrátorovi.

Přehled a editace závěrečných zkoušek

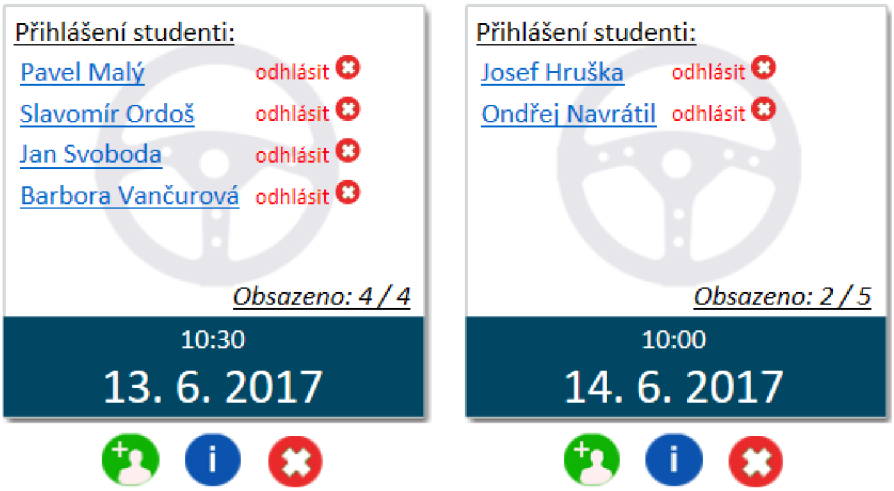
Přehled pro správu závěrečných zkoušek je řešen s využitím dlaždicového zobrazení. Každá dlaždice představuje jednu závěrečnou zkoušku. Administrátor má takto okamžitý přehled o všech zkouškách, jejich obsazenosti a přihlášených studentech. Přehled přihlášených studentů na daný termín zkoušky se nachází přímo uvnitř dlaždice. Při kliku na jméno studenta se zobrazí Přehled jízd studenta. Zároveň je přímo odtud možné kteréhokoliv studenta z termínu odhlásit.

U každé dlaždice (závěrečné zkoušky) má administrátor k dispozici tři možnosti:

- **Přihlásit studenta na termín ZZ** – na rozdíl od přihlašování studentem na závěrečnou zkoušku zde neprobíhá při uložení do databáze tolik kontrol. Administrátor může přihlásit libovolného studenta i nad rámec obsazenosti zkoušky. Nelze přihlásit stejného studenta na 1 termín dvakrát.

- **Zobrazit podrobné informace** – zobrazí se detail závěrečné zkoušky i s poznámkou a dodatečnými informacemi s rychlou nabídkou nejčastějších funkcí.
- **Smazat zkoušku** – odstranit termín závěrečné zkoušky lze jen tehdy, není-li na něj přihlášen žádný student. Kontrola probíhá SQL dotazem do databáze.

Dlaždice se závěrečnými zkouškami se zobrazují vedle sebe 3 na řádku, dále pak pod sebou. Může jich být neomezeně, nicméně se předpokládá, že vypsaných termínů zkoušek moc dopředu nebude. Správa závěrečných zkoušek (dlaždice) vypadá následovně:



Obrázek č. 15 – Správa závěrečných zkoušek – zobrazení (zdroj: vlastní zpracování)

3.5.8 Statistiky

Veškerá data aplikace jsou uložena v databázi. Celá agenda statistik stojí na tvorbě sestav a správných SQL dotazů do databáze. Administrátor si může vybrat, která oblast, jaké informace a za jaké časové období ho zajímají. Např. v přehledech o oblíbenosti časů jízd v rámci dne (na které časy se studenti přihlašují nejčastěji) z databáze vybereme (propojením správných tabulek) pro daný časový interval počet záznamů. Administrátor získá počty v přesných číslech a v procentuálním vyjádření. Zvýrazněny jsou maximální, a naopak i minimální hodnoty apod. Pro efektivitu je u statistik důležité přehledné grafické zpracování jednotlivých reportů. Hodnoty by měly být snadno přístupné pro

export do jiných programů, např. MS Excel, a to z důvodu detailnější analýzy – tvorby grafů apod.

3.6 Rozhraní studenta

Tato kapitola se věnuje agendám systému, které jsou určeny pouze pro studenty. Každý student se do tohoto rozhraní přihlašuje na základě svých přihlašovacích údajů a spravuje si svou výuku samostatně.

3.6.1 Plánování jízd

Agenda plánování jízd studentem se skládá ze dvou částí – Kalendáře pro plánování a Karty studenta. Požadavky na obě části jsou detailně popsány v analýze.

Kalendář

Logika procházení kalendářem je podobná jako v případě vypisování nových jízd administrátorem. Celý kalendář se počítá od aktuálního dne a dny do minulosti nelze zobrazit. Rozdíl oproti kalendáři administrátora spočívá v tom, že se studentovi nezobrazuje aktuální den a pokud je víc jak 20:00, tak ani den následující – přihlašování na jízdy je ukončeno. Tohoto stavu je docíleno následující podmínkou se správnou kombinací funkcí *date()* a *strtotime()*:

```
if ($od == date("d.m.Y") AND date("H:i:s", strtotime($cas_uzamceni)) < date("H:i:s")){
   $od = date("d.m.Y", strtotime("+ 2 day"));
   }
```

Aby se nezobrazoval aktuální den, nastavíme hodnotu proměnné *$od* na den následující:

```
if ($od == date("d.m.Y")) {
$od = date("d.m.Y", strtotime("+ 1 day"));
}
```

Hodnoty se předávají pomocí metody *GET*, tudíž jsou viditelné (editovatelné) v URL. Kdokoliv je může přepsat a potencionálně vnutit skriptu neočekávané hodnoty. Z tohoto důvodu jsou vždy všechny vstupy „ošetřeny“ a více místech kontrolovány. Např. není možné v kalendáři zobrazit dny do minulosti nebo se na některý minulý termín jízdy zpětně přihlásit.

Kalendář dopočítává dny za všech okolností správně. Především při přechodu zpět je nutné spočítat správný počet dnů. Ve všech krocích se bere v potaz čas uzamčení jízd – kdykoliv je aktuální hodnota času větší než čas uzamčení, kalendář zobrazuje správný den a na jízdy se již nedá přihlásit.

Každé vozidlo má svůj vlastní kalendář. Student může zobrazit pouze kalendáře vozidel, pro která podstupuje výcvik na základě skupiny ŘO. Při každé změně vozidla ze selectboxu (nebo přepisem ID vozidla v URL) se provádí kontrola dotazem do dekompoziční tabulky *uživatel_skupina_ro*, zda je aktuálně přihlášený student oprávněn zobrazit jízdy (kalendář) pro dané vozidlo. V již zmiňovaném selectboxu pro výběr vozidla jsou studentovi zobrazeny pouze ta, na kterých může absolvovat výcvik podle skupin řidičského oprávnění.

Při vykreslování kalendáře (matice) mohou u jednotlivých políček nastat 3 stavy:

- **Volná jízda** – administrátor pro dané vozidlo vypsal plánovaný termín jízdy a ten ještě není obsazen jiným studentem. Student se na tento termín jízdy může přihlásit.
- **Má jízda** – termín jízdy, na který je přihlášen aktuálně přihlášený student, tj. student jehož *$_SESSION[“id“]* se rovná ID studenta, který je na danou jízdu přihlášen. Z tohoto termínu se lze, do času uzamčení jízdy,
- **Obsazený termín** – administrátor pro dané vozidlo vypsal plánovaný termín jízdy a již ho obsadil jiný student (*$_SESSION[“id“]* se nerovná ID studenta, který je na danou jízdu přihlášen) nebo pro daný termín nebyl vypsán termín jízdy (studentovi se obě možnosti jeví jako obsazený termín). S tímto termínem nemůže student nijak nakládat.

Každé pole kalendáře je vhodně graficky zpracováno:

Obrázek č. 16 – Pole kalendáře studenta (zdroj: vlastní zpracování)

Při kliku na volnou jízdu se provede několik kontrol:

- zda je termín stále volný
- zda již nebylo přihlašování uzamčeno
- zda již pro danou skupinu ŘO nemá student naplánovány všechny jízdy
- zda v tento den již nemá student naplánovanou jinou jízdu s jiným vozidlem
- zda student může odjet jízdu s tímto vozidlem na základě skupiny ŘO

```
SELECT us.id
FROM uzivatel_skupina_ro us, vozidlo v, planovane_terminy pt
WHERE pt.vozidlo=v.id AND v.skupina_ro=us.skupina AND
us.id_uzivatele=".$_SESSION['id']." AND pt.id=".$_GET['id']."
```

- zda v tento nebo předchozích dnech nemá student naplánovanou závěrečnou zkoušku

```
SELECT zz.id
FROM zaverecna_zkouska zz, student_zavzk sz
WHERE zz.datum <= '".$record["datum"]."' AND sz.id_std= ".$_SESSION["id"]." AND
sz.id_zavzk = zz.id
```

Pokud student všemi kontrolami projde, zobrazí se informace o jízdě s možností výběru místa nástupu a potvrzením. Před uložením do databáze proběhne ještě jedno kolo kontrol (může např. nastat situace, že před potvrzením jízdy se na danou jízdu přihlásí jiný student, a tudíž je již obsazená apod.). Po úspěšném uložení se v kalendáři pro dané vozidlo jízda jeví jako „Má jízda“, stejně tak je zapsaná do karty studenta. Při kliku na pole „Má jízda“ se zobrazí informace o jízdě s možností odhlásit se z jízdy. Po odhlášení (vymazání hodnot z databáze) se jízda opět pro všechny studenty zobrazuje jako „volná jízda“.

Karta studenta

V kartě studenta se zobrazí základní informace o aktuálně přihlášeném studentovi včetně skupin ŘO. Pro přehled je umístěna přímo vedle kalendáře. Na základě počtu skupin se zobrazí počet oddílů, ve kterých budou jednotlivé jízdy podle těchto skupin roztříděny. Student má pohromadě a přehledně zobrazeny všechny jízdy. První oddíl nahoře, hned pod informacemi o studentovi, je věnován aktuálně nejbližší jízdě, která studenta čeká.

Další oddíly jsou věnovány skupinám ŘO. Jízdy jsou z databáze vybírány již seřazené podle data. Přidat se pouze musí pořadové číslo a v rámci každém skupiny ŘO graficky zvýraznit nejbližší jízdu. Tato funkčnost funguje tak, že se u každé jízdy testuje, zda je její datum větší než datum aktuální a pomocná proměnná nastavena na určitou hodnotu. Po prvním průchodu cyklu, kdy se podmínce vyhoví, se pomocná proměnná nastaví na jinou hodnotu, což zapříčiní zvýraznění pouze jedné aktuálně nejbližší jízdy. Již proběhlé jízdy jsou méně výrazné, budoucí jízdy mají normální font a nejsou nijak zvýrazněny.

Místo nástupu dané jízdy se zobrazuje po najetí nebo kliku (klik z důvodu dotykových zařízení) na symbol domečku.

3.6.2 Závěrečné zkoušky

Zobrazení závěrečných zkoušek pro přihlašování studentem je řešeno pomocí dlaždicového uspořádání. Dlaždice poskytují maximální přehlednost. Studentovi se zobrazují pouze zatím neproběhlé termíny zkoušek. Na první pohled má přehled o všech termínech a jejich obsazenosti, na které se může přihlásit. Podmínkou je, aby do termínu závěrečných zkoušek měl odjety všechny jízdy. Přihlásit se může pouze na termín, který není plně obsazen. Po kliku na zkoušku se otevřou podrobné informace o zkoušce, pro přihlášení ji však otevírat nemusí, přihlásit se dá i přímo z dlaždice. Student, na rozdíl od administrátora, nevidí kdo je na zkoušky přihlášen, pouze má přehled o počtu lidí.

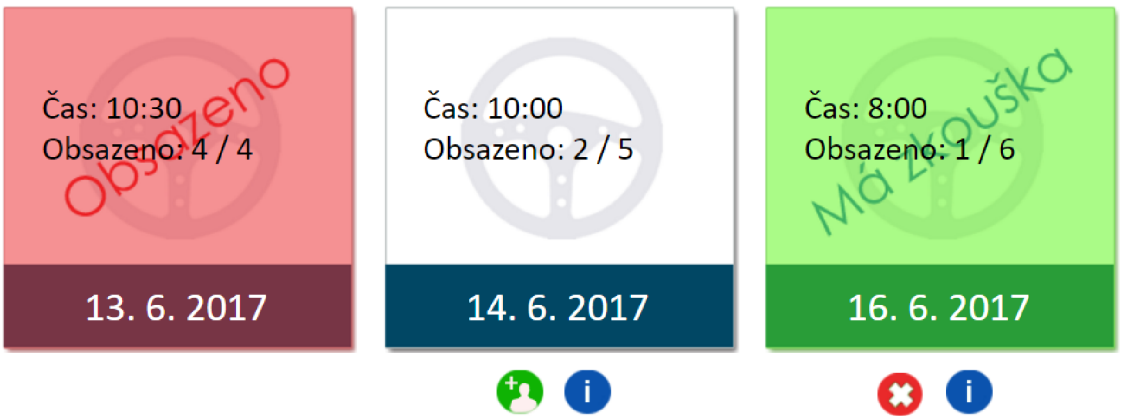
Po kliku na symbol „Přihlásit se na zkoušku“, se provedou potřebné kontroly, např.:

- zda již termín není obsazen
- zda již student není přihlášen na jiný termín ZZ, který ještě neproběhl

- zda již nebylo přihlašování na tento termín ukončeno
- zda v den závěrečné zkoušky nemá naplánovanou jízdu

Po projití kontrolami se zobrazí informace o zkoušce s potvrzením o přihlášení. Po potvrzení se před uložením do databáze znovu spustí všechny kontroly a v případě úspěchu se termín závěrečné zkoušky studentovi graficky zvýrazní jako „Má zkouška“. V administrátorském rozhraní se zároveň student zobrazí jako přihlášený na daný termín zkoušky. Z termínu je možné se kdykoliv před uzamčením odhlásit. Vypsaných zkoušek může být neomezené množství, záleží na administrátorovi.

Jednotlivé zkoušky (dlaždice) pro studenta vypadají takto:



Obrázek č. 17 – Závěrečné zkoušky – přihlašování studentem (zdroj: vlastní zpracování)

3.7 Rozhraní instruktora

V autoškole slouží rozhraní pro instruktora pouze pro přehled naplánovaných jízd. Tato funkčnost je shodná jako administrátorská agenda „Jízdy instruktorů“, s tím rozdílem, že instruktor po přihlášení vidí pouze své naplánované jízdy. Nemá přístup k naplánovaným jízdám ostatních instruktorů. Instruktor si ve svém rozhraní může změnit heslo, vytisknout seznam „uzavřených“ jízd nebo popřípadě zobrazit přehled jízd daného studenta.

Tato část aplikace v současné době tvoří pouze informační podporu pro instruktora. Instruktor má odkudkoliv (s využitím např. chytrého telefonu) přehled o svých naplánovaných jízdách. Autoškola nyní žádnou další funkčnost nevyžaduje. Implementace rozšiřujících agend je plánována do budoucnosti např. o správu tankování PHM, apod.

3.8 Rozšíření do budoucna

Systém je možné rozšířit o velké množství doplňkových agend, které slouží pro ještě vyšší podporu chodu autoškoly. Autoškola v současné době nechce žádná další rozšíření, výhledově do budoucna se ale tato možnost jeví jako pravděpodobná. Aplikace je pro případné dodatečné implementace agend připravena a již od začátku návrhu se s touto možností počítalo.

Budoucí rozšíření pro autoškolu budou zaměřena především na tyto oblasti:

- Evidence plateb
- Správa teoretické výuky, zdravovědy
- Rozšíření správy vozidel – tankování, STK, údržba, nájezdy km, náklady
- Správa zaměstnanců
- Komunikace s úřady

3.9 Ostrý provoz

Před nasazením do reálného provozu autoškoly aplikace prošla finálním testováním a měsíc běžela ve zkušebním provozu. Během této doby se podařily odhalit poslední drobné chyby, které se při testování s testovacími daty nevyskytly. Všechny procesy byly na základě dodatečných požadavků autoškoly dotáhnuty do konce. Aplikaci se podařilo naimplementovat do ostrého provozu a efektivně autoškole slouží k podpoře její činnosti.

Po dobu ostrého provozu dosud s aplikací již dokončily svůj výcvik stovky studentů. Nevyskytl se žádný problém s ovládáním nebo funkčností systému. Aplikaci využívají studenti téměř všech věkových kategorií, od nejmladších po cca ročník narození 1960. Kladné ohlasy přicházejí jak ze strany studentů, tak i od autoškoly. Při prováděném průzkumu mezi studenty, jak aplikaci vylepšit, co v ní chybí a bylo by užitečné dodělat, byla několikrát zmíněna myšlenka vytvořit k stávajícímu rozhraní i mobilní aplikaci pro operační systémy Android a iOS. Žádné jiné smysluplné návrhy nebyly zmíněny. Podle statistiky návštěvnosti v současné době do aplikace přistupuje zhruba 50 % uživatelů z operačního systému Android. Druhou půlku tvoří OS od společnosti Microsoft – Windows. Procento uživatelů, kteří do aplikace přistupují z chytrého telefonu nebo tabletu, v čase neustále roste, a tak se budoucí vývoj bude nejspíš ubírat i tímto směrem.

3.10 Ekonomické zhodnocení a přínosy

Dosavadní vývoj ukazatelů nasvědčuje tomu, že s implementací systému autoškole meziročně vzrostl počet zájemců o výcvik cca o 15 %. Mezi studenty je online komunikace a plánování svého času žádaným benefitem a pro autoškolu je implementace aplikace do jisté míry konkurenční výhoda.

Celková částka za celou aplikaci včetně vývoje, četných konzultací a implementace závisí na počtu člověkohodin a množství agend, které autoškola požaduje. V tomto případě se počet člověkohodin pohybuje okolo 350 – 400. To znamená, že při hodinové sazbě pohybující se okolo 400 Kč / hod. se celková částka pohybuje okolo 150 000 Kč bez DPH. Tyto náklady jsou pro autoškolu vysoké a sama by si implementaci nemohla dovolit. Vzhledem k faktu, že aplikace byla tvořena obecně, s možností využití ve více autoškolách (procesy jsou v autoškolách velice podobné), obchodní model je jiný. Autoškola platí malý měsíční poplatek za provoz systému splatný jednou ročně. Takto si aplikaci (vzhledem k ceně) může dovolit naimplementovat jakákoliv autoškola. S růstem počtu implementací se náklady vložené do vývoje budou vracet rychleji.

ZÁVĚR

Úvodní, teoretická část práce se zabývala technologiemi a metodami pro stavbu aplikace, která slouží pro podporu chodu autoškoly. Při jakémkoliv vývoji je důležitým bodem bezpečnost. V této kapitole byly představeny nejčastější útoky na webové aplikace a způsoby obrany. Závěr teoretické části práce je věnován současné úloze ICT v podniku a možnostem využití a nasazení Cloud computingu.

V analytické části byla představena a zanalyzována současná situace v autoškole pomocí vhodných metod (SLEPT, 7S, SWOT). Z analýzy vyplynula nutnost změny ve fungování a chodu autoškoly. Následně byly detailně popsány požadavky na aplikaci, vycházející z potřeb autoškoly a jejích procesů. Uvažovaná hotová řešení nebyla pro autoškolu uspokojivá, tudíž se přistoupilo k vlastnímu návrhu řešení.

Poslední část práce tvoří vlastní návrh řešení. Zabývá se tvorbou a implementací aplikace do chodu autoškoly. V jednotlivých kapitolách byly ukázkami nastíněny návrhy řešení agend a jejich funkčností na základě provedené analýzy. Velká pozornost byla věnována vhodnému grafickému rozvržení uživatelského rozhraní. Na závěr bylo provedeno ekonomické zhodnocení daného řešení a shrnuty poznatky z reálného ostrého provozu tohoto systému. Systém se podařilo úspěšně naimplementovat do reálného provozu autoškoly.

Takto navrženou a naprogramovanou aplikaci lze reálně nasadit, s minimem úprav, do více autoškol. Aplikaci je možné poměrně snadno rozšířit o další moduly, které autoškola využije při své praxi. Záleží na individuálních požadavcích. Vzhledem k měnící se legislativě a požadavkům autoškoly vývoj této aplikace nekončí.

SEZNAM POUŽITÝCH ZDROJŮ

[1] PROCHÁZKA, D. *SEO: cesta k propagaci vlastního webu*. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-4222-9.

[2] HOPKINS, C. *PHP okamžitě*. Brno: Computer Press, Albatros Media a. s., 2016. ISBN 978-80-251-4393-3.

[3] STEPHENS, R. K., R. R. PLEW a A. JONES. *Naučte se SQL za 28 dní: [stačí hodina denně]*. Brno: Computer Press, Albatros Media a. s., 2016. ISBN 978-80-251-4220-2.

[4] PONKRÁC, M. *PHP a MySQL: bez předchozích znalostí: [průvodce pro samouky]*. Vyd. 1. Brno: Computer Press, 2007. ISBN 978-80-251-1758-3.

[5] PÍSEK, S. *HTML: začínáme programovat*. 4., aktualiz. vyd. Praha: Grada, 2014. Průvodce (Grada). ISBN 978-80-247-5059-0.

[6] JANOVSKÝ D. CSS styly – úvod. *Jakpsatweb.cz* [online]. [cit. 2014-11-10]. Dostupné z: http://www.jakpsatweb.cz/css/css-uvod.html

[7] LAZARIS, L. *CSS okamžitě*. Brno: Computer Press, Albatros Media a. s., 2016. ISBN 978-80-251-4278-3.

[8] SUEHRING, S. *JavaScript: krok za krokem*. Brno: Computer Press, Albatros Media a. s., 2016. ISBN 978-80-251-4069-7.

[9] MARGORÍN, M. *JQuery bez předchozích znalostí*. Brno: Computer Press, 2011. ISBN 978-80-251-3379-8.

[10] PROGRAMUJTE. Ajax – úvod. *Programujte.com* [online]. ©2008-2016 [cit. 2016-12-15]. Dostupné z: http://programujte.com/clanek/2008062101-ajax-uvod/

[11] INTERVAL. Jak na démona Cron. *Interval.cz* [online]. Datum vydání: 21. 4. 2002 [cit. 2015-03-12]. Dostupné z: http://interval.cz/clanky/jak-na-demona-cron/

[12] PROCHÁZKA, D. *SEO: cesta k propagaci vlastního webu*. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-4222-9.

[13] JANOVSKÝ D. Podvodné optimalizační techniky. *Jakpsatweb.cz* [online]. [cit. 2014-12-10]. Dostupné z: https://www.jakpsatweb.cz/clanky/podvodne-seo-techniky.html

[14] JANOVSKÝ D. Google PageRank. *Jakpsatweb.cz* [online]. [cit. 2014-12-10]. Dostupné z: https://www.jakpsatweb.cz/seo/pagerank.html

[15] DRUSKA, P. *CSS a XHTML: tvorba dokonalých webových stránek krok za krokem*. Praha: Grada, 2006. Průvodce (Grada). ISBN 80-247-1382-9.

[16] Ferschmann P. Bezpečnost na webu – přehled útoků na webové aplikace. *Zdrojak.cz* [online]. ©2008-2016 [cit. 2016-12-12]. Dostupné z: https://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/

[17] KOCH, M. a B. NEUWIRTH. *Datové a funkční modelování*. Vyd. 4., rozš. Brno: Akademické nakladatelství CERM, 2010. ISBN 978-80-214-4125-5.

[18] BASL, Josef a Roman BLAŽÍČEK. *Podnikové informační systémy: podnik v informační společnosti*. 3., aktualizované a dopl. vyd. Praha: Grada, 2012. Management v informační společnosti. ISBN 978-80-247-7594-4.

[19] NIST. The NIST Definition of Cloud Computing. *Recommendations of the National Institute of Standards and Technology*. [online] 2011 [cit. 2016-20-12] Dostupné z: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[20] LUPA. *Cloud computing ... je všude okolo nás*. Lupa.cz [online]. Datum vydání: 19. 9. 2010 [cit. 2016-20-12]. Dostupné z: http://www.lupa.cz/clanky/cloud-computing-je-vsude-okolo-nas/

[21] SILVERLIGHT HACK. IaaS, PaaS and SaaS Terms Clearly Explained and Defined. *Silverlight Hack.com* [online]. ©2011 [cit. 2016-12-24]. Dostupné z: http://www.silverlighthack.com/post/2011/02/27/IaaS-PaaS-and-SaaS-Terms-Explained-and-Defined.aspx

[22] MICROSOFT. Co je to Cloud computing? *Azure.microsoft.com* [online]. ©2017 [cit. 2016-12-29]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-cloud-computing/

SEZNAM OBRÁZKŮ

SEZNAM SCHÉMAT

SEZNAM PŘÍLOH

Příloha 1: Základní databázové schéma