Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Návrh a implementace webové aplikace pro rezervaci jízd v autoškole

Adam Valenta

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Adam Valenta

Systémové inženýrství a informatika
Informatika

Název práce

Návrh a implementace webové aplikace pro rezervaci jízd v autoškole

Název anglicky

Design and Implementation of Web Application for Reservation of Driving in Driving School

---

Cíle práce

Cílem této práce je navržení a implementace webové aplikace pro žáky autoškoly a jejich instruktory. Hlavní funkcí bude umožnit rezervace jízd pro žáky, kdy tyto rezervace budou moci vytvářet instruktoři podle jejich časových možností.

Metodika

1. Vypracujte literární rešerši problematiky vývoje webových aplikací na základě relevantních zdrojů

2. Popište problémovou doménu a analyzujte uživatelské požadavky na webovou aplikaci. Uživatelské požadavky znázorněte pomocí vhodných nástrojů (Use case)

3. V souladu s osvědčenými metodami softwarového návrhu navrhněte vlastní webovou aplikaci

4. Návrh aplikace implementujte ve vhodném prostředí, řádně otestujte a následně zhodnoťte její další možný vývoj

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

webová aplikace, uživatelské požadavky, autoškola

Doporučené zdroje informací

GÁLA, Libor, Jan POUR a Zuzana ŠEDIVÁ. Podniková informatika: počítačové aplikace v podnikové a mezipodnikové praxi. 3., aktualizované vydání. Praha: Grada Publishing, 2015. Management v informační společnosti. ISBN 978-80-247-5457-4.

MCNEIL, Patrick. Inspirativní webdesign: průvodce nejlepšími tématy, trendy a styly. Brno: Computer Press, 2011. ISBN 9788025135174.

WELLING, Luke a Laura THOMSON. Mistrovství PHP a MySQL. Přeložil Ondřej BAŠE. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.

Předběžný termín obhajoby

2021/22 ZS – PEF

Vedoucí práce

Ing. David Buchtela, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 25. 11. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 27. 11. 2021

Čestné prohlášení

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

V Praze dne 30. 11. 2021 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Davidovi Buchtelovi, Ph.D. za rady a odborné vedení při zpracování této práce.

Návrh a implementace webové aplikace pro rezervaci jízd v autoškole

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací webové aplikace pro rezervaci jízd v autoškole. V teoretické části se řeší problematika technologií a životní cyklus aplikací při jejich vývoji. Především zde probíhá seznámení s kódovacími jazyky HTML a CSS, programovacími jazyky PHP a Javascript a dotazovacím SQL, které jsou následně využity při implementaci již zmíněné webové aplikace. Implementaci předchází analýza, kde se hodnotí současný stav a požadavky na aplikaci. Návrh sestává z Use Case a wireframe (drátěných) modelů doplněný o návrh databáze. Výsledná aplikace umožňuje žákům rezervovat jízdy, které vytvářejí instruktoři podle svých časových možností. Aplikace dále umožňuje registraci, editaci a smazání žáků a instruktorů, na což mají práva pouze instruktoři (respektive admini).

Klíčová slova: webová aplikace, uživatelské požadavky, autoškola

Design and Implementation of Web Application for Reservation of Driving in Driving School

Abstract

This bachelor thesis deals with the design and implementation of web application for driving reservations at a driving school. The theoretical part addresses the issue of technology and the life cycle of applications in their development. Above all, there is an introduction to HTML and CSS coding language, PHP and Javascript programming languages and SQL query, which are then used in the implementation of the aforementioned web application. Implementation is preceded by an analysis that evaluates the curent state and requirements of application. The design consists of Use Case and wireframe models supplemented by a database design. The resulting application allows students to book rides created by instructors according to their time possibilities. The application also allows registration, editing and deleting of students and instructors, to which only instructors (respectively admins) have rights.

Keywords: web application, use cases, driving school

Obsah

Seznam obrázků

1 Úvod

Dlouho jsem přemýšlel, na jaké téma psát svoji bakalářskou práci. Až když jsem před nedávnem dělal autoškolu, uvědomil jsem si, kolik času zabere plánování jízd nejen pro učitele, ale i pro žáky. Vždy se strávila minimálně třetina teoretické hodiny tím, že se plánovaly jízdy, a i během výuky náš školitel často dostával spoustu telefonátů od žáků s žádostmi o naplánování svých praktických hodin a tím se naše hodina zúžila na ještě méně času.

Proto jsem si zvolil práci na téma vytvoření webové aplikace pro rezervaci jízd v autoškole, která by měla výše zmíněnou ztrátu času eliminovat a to tak, že si jízdy budou školitelé plánovat mimo teoretickou výuku a stejně tak si je budou rezervovat žáci autoškoly mimo teoretických hodin. Základem aplikace bude přihlášení žáka do aplikace a jeho následná rezervace termínu své praktické výuky, kde si tyto termíny budou plánovat přímo ti instruktoři, kteří budou uchazeče školit při jízdě.

Webová aplikace je ideálním řešením pro tuto práci, protože technologie internetu je dnes všude kolem nás a každý se s ní musel někdy setkat a také jediné, co k ní účastníci potřebují je zařízení připojené k internetu. Výhodou je také to, že se nikde nic nemusí instalovat a veškeré aktualizace softwaru se ihned promítnou na obrazovkách uživatelů aplikace.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem této práce je popsat problematiku vývoje webových aplikací, s čímž bude souviset následné navržení a implementace webové aplikace pro žáky autoškoly a jejich instruktory pomocí programovacího jazyka PHP s využitím značkovacího jazyka HTML a stylovacího CSS. Hlavní funkcí bude umožnit rezervace jízd pro žáky, kdy tyto rezervace budou moci vytvářet instruktoři podle jejich časových možností. Vedlejší funkce bude možnost přidávat nové žáky a instruktory, dále možnost je editovat a mazat.

Dalším cílem bude popsat vytvoření této aplikace a snaha o to, aby byla aplikace co nejjednodušší pro její uživatele.

2.2 Metodika

Bakalářská práce bude sestávat ze dvou částí, kdy teoretická část bude vytvářena na základě studia odborných tištěných a elektronických zdrojů.

Druhá, praktická, část práce bude tvořena na základě poznatků z první části. Práce bude sestávat z návrhu samotné webové aplikace a její následné tvorbě. Front-end se vytvoří prostřednictvím jazyků HTML a CSS a back-end s využitím jazyka PHP. Databázová část potom z jazyka SQL prostřednictvím databázového systému MySQL.

3 Teoretická východiska

3.1 Internet

Internet je systém propojených počítačových sítí, které propojují mezi sebou jednotlivá zařízení (nejčastěji počítače, ale dnes již také mobily a tablety) komunikující mezi sebou pomocí protokolů TCP/IP (Transport Control Protocol/ Internet Protocol). Jedná se o sadu protokolů potřebných v síti, aby výměna dat mezi počítači probíhala bez chyb. Skládá se ze čtyř vrstev – aplikační, transportní, síťové a vrstvy síťového rozhraní. Posledně jmenovaná vrstva je nejnižší a zajišťuje vysílání a příjem dat, což závisí na konkrétní přenosové technologii (např. Ethernet, Wi-fi). [1, 2]

Síťová vrstva přenáší data mezi jednotlivými uzly (počítači) v síti. Aby tento úkol zvládla, musí být přenášená data opatřena IP hlavičkou, která obsahuje příjemce dat. Ovšem pokud při přenosu dojde k poškození nebo ztrátě dat, tak to ani nepostřehne a automaticky vysílá další data. [2]

To, zda je potřeba vysílat ihned další data, nebo ztrátu dat nahradit zajišťuje transportní vrstva, konkrétně TCP. Protože např. při online přenosu je potřeba, aby se vysílalo ihned dále a ztracená data se již neobnovila oproti např. načítání webové stránky, kterou je potřeba mít celou a je tedy nutné poslat ztracená data znovu. [2]

Poslední, aplikační vrstva, standardizuje přístup a poskytuje podporu aplikacím v počítači. Příkladem můžou být protokoly POP3 a IMAP, které se týkají příjmu a odesílání e-mailů, mezi další známe protokoly patří FTP pro přenos souborů nebo HTTP (Hyper Text Transfer Protokol) pro komunikaci WWW stránek. [2]

3.1.1 WWW (World Wide Web)

Jedná se o jednu ze služeb poskytovaných globální počítačovou sítí internet, která umožňuje přenos a prohlížení webových aplikací (www nerovná se internet!). Jak již bylo řečeno, ke komunikaci na internetu v rámci „celosvětové sítě“ slouží protokol HTTP, nebo dnes již na většině webů HTTPS (Hypertext Transfer Protocol Secure), což jak název

napovídá, znamená, že přenos dat probíhá zabezpečeně, a to pomocí protokolu TLS (dříve SSL). Komunikace probíhá mezi webovým serverem, kde jsou data (webové aplikace) uložena a prohlížečem (klientem) nainstalovaným na počítači uživatele, který si je vyžádá pomocí hypertextového odkazu zapsaného ve formě URL a které mu jsou následně vygenerovány technologií HTML. [5, 6]

HTTP dále obsahuje sadu metod, které se volají podle toho, co chce uživatel provést. Mezi ty základní patří:

- GET – metoda odesílá data v adrese (URL) webové stránky, používá se ve chvíli, kdy klient vyžaduje data ze serveru.
- POST – metoda zajišťující odeslání dat od klienta na server (např. skrze nějaký formulář). Odesílá data v těle požadavku.
- DELETE – metoda smaže daný údaj ze serveru. [5, 6]

3.1.2 Historie internetu

Hlavním impulsem ve vývoji telekomunikačních technologií ze strany západních zemí bylo vypuštění první družice do vesmíru Sovětským svazem v roce 1957. Ministerstvo obrany USA proto na počátku 60. let zakládá organizaci ARPA (Advanced Research Projects Agency), které se 29. října 1969 povedlo spustit experimentální počítačovou síť ARPANET. Prvně propojovala pouze čtyři americké univerzity a byla pouze pro vojenské a vládní účely. Pro komerční využití tehdy nikdo neviděl potenciál, jelikož tehdejší využití bylo omezené, většinou jen na posílání e-mailů (v roce 1984 propojeno pouze 1000 počítačů). [3, 4]

Přelom přichází až na konci 80. let, kdy se v laboratořích ve švýcarském CERNU Tim-Bernes Lee a Robert Cailliau ujímají vývoje hypertextových dokumentů, navržených již dříve a první webové stránky (napsané v jazyce HTML) tak spatřují světlo světa 6. srpna 1991. O rok později je připojen Bílý dům a internet přichází také do České republiky, také je vyvinut první webový prohlížeč, který se jmenoval WorldWideWeb a tento rok je také datován jako počátek komerčního využití internetu. Tomu také odpovídají čísla, protože v roce 1996 je připojeno již 55 milionu zařízení. [3, 4]

3.2 Webová aplikace

Je to aplikace běžící na vzdáleném webovém serveru (v podstatě jde o počítač, na němž je uložena webová aplikace včetně databáze), ke kterému se uživatel připojí pomocí právě již zmíněného internetu a následně s ním komunikuje přes prohlížeč (např. Google Chrome, Mozzila Firefox, Opera), jež musí mít nainstalovaný na svém zařízení. Obecně největší výhodou webových aplikací je jejich dostupnost, jelikož uživateli stačí mít pouze internetové připojení, a kromě prohlížeče nemusí na svůj počítač již instalovat další programy. Právě spouštění přes webový prohlížeč je další výhodou webů, protože se nemusí řešit hardware uživatele, aplikace jde spustit na kterémkoliv zařízení, kde je nainstalován prohlížeč. Naopak za nevýhodu se dá označit napadnutelnost webových aplikací, jelikož na internetu se může pohybovat kdokoliv. [7]

Webové aplikace se dělí na jednodušší statické webové stránky a sofistikovanější dynamické webové stránky (označované právě jako webové aplikace). Rozdíl je v tom, že první zmíněné ze serveru generují pouze jednoduché HTML stránky, takže jejich využití je omezené v podstatě pouze na psaní článků nebo jednoduchých informativních webů. Zatím co druhé zmíněné obsahují programovou logiku, dosaženou programovacími jazyky jak na straně serveru (Back-end), tak na straně klienta (Front-end) s databází pro ukládání dat. Umožňují tedy např. autentizaci uživatelů, přidávání příspěvků od uživatelů na diskusních fórech, vytváření různých rezervací, e-shopů atd. [7]

3.3 Front-end

Jedná se o tu část webové aplikace, která je viditelná uživatelům. Front-end daného webu by tedy měl být přehledný, zajímavý pro uživatele a jednoduše ovladatelný, tedy jinak řečeno user-friendly (uživatelsky přívětivý), aby se uživatelé na web často a rádi vraceli. [8]

Mezi jazyky, pomocí kterých se vytváří front-end patří HTML, CSS a Javascript, které jsou popsány níže. Jinak řečeno, jedná se o jazyky běžící na straně klienta. [8]

3.3.1 HTML

HTML neboli „Hyper Text Markup Language“ je značkovací jazyk používaný pro tvorbu webových stránek. Značkovací jazyk znamená, že vlastní text, obrázek, video atd. obsahuje dodatečné informace o tom, jak má být daný text zpracován – především upravují obsah, strukturu a vzhled (vzhled však dnes již přebral jazyk CSS, o kterém je napsáno níže). To se děje pomocí tzv. tagů (značek). Příkladem HTML tagu pro vypsání odstavce je Hello world , kde je značka pro začátek a konec odstavce, vše mezi značkami se pak vypíše jako odstavec. [5]

Pro tvorbu HTML stránek se dá použít v podstatě jakýkoliv textový editor, i když existují i sofistikované programy, tzv. „WYSIWYG“ editory což znamená „What You See Is What You Get“ (Co vidíš, to dostaneš). Jedná se např. o Adobe Dreamweaver, kde uživatel pouze nadesignuje stránku podle jeho požadavků a následně se mu vygeneruje HTML kód. Pro vygenerování obsahu je poté potřeba následně použít webový prohlížeč, který interpretuje napsaný zdrojový kód. [5]

Jeho zakladatelem je Timothy Berns Lee, který v roce 1991 vymyslel tento jazyk jako následník jazyka SGML. V současné době se používá verze HTML 5 vydaná v roce 2015, která se více přimyká k CSS a odstraňuje některé tagy, ovlivňující vzhled stránek – tedy HTML se soustředí na obsah a strukturu, zatím co CSS, který je popsán níže se zaměřuje na úpravu vzhledu webových stránek. [5]

3.3.2 CSS

CSS (Cascading Style Sheets) je jazyk sloužící k úpravě vzhledu a rozložení HTML elementů na stránce. CSS dokument je většinou oddělen od HTML dokumentu a je k němu připojen pomocí odkazu. Jeho výhodou je jednak oddělení obsahu od vzhledu, což zvyšuje přehlednost kódu a tím zjednodušuje údržbu webových stránek. Dalším plusem je možnost odkázat jeden selektor na více HTML elementů. [10, 11]

Počátek CSS se datuje na rok 1996 jako snaha oddělit vzhled a formátování od obsahu. Tehdy vzniklo CSS1, které však umožňovalo pouze úpravu malého množství prvků (barvu, velikost a pozadí textu). V dnešní době se od roku 2014 využívá verze CSS3, pomocí které lze vzhledově upravit celou webovou stránku. [10, 11]

3.3.3 Bootstrap

Bootstrap je framework, což znamená, že se jedná o ucelenou knihovnu poskytující řešení k různým (často se opakujícím) problematikám. Konkrétně Bootstrap poskytuje sadu nástrojů pro CSS a upravuje často vyskytující se komponenty webových aplikacích jako jsou texty, nadpisy, obrázky, formuláře, tlačítka, tabulky, navigace, atd. Dále nabízí velmi jednoduchý tzv. grid systém (mřížkový systém), který pomyslně rozděluje webovou stránku na 12 stejně širokých sloupců (to z toho důvodu, že číslo 12 je dobře dělitelné více čísly). Díky tomu se dá hezky naplánovat (a následně jednoduše rozmístit), kde co na stránce bude. Bootstrap je navíc open-source, což znamená, že ho lze použít zdarma, a to i pro komerční účely. [9]

Jeho vznik je datován rokem 2011 kdy ho představili zaměstnanci Twitteru (Mark Otto a Mark Thornton), jež trápil různý vzhled aplikací ve firmě. Již příští rok se stal nejoblíbenějším frameworkem kaskádových stylů a toto prvenství si drží dodnes. [9, 12]

3.3.4 Javascript

Dalším jazykem běžícím na straně klienta je Javascript. Jedná se o scriptovací jazyk spadající do rodiny programovacích jazyků C. Jeho využití je především na webu, kde umožňuje vytvářet dynamiku stránky, čímž se rozumí různé roletky, přidávání smajlíků, zaslání upozornění uživateli ještě před odesláním požadavku na server, pokud nejsou např. vyplněna všechna pole (nebo jsou vyplněna chybně) a dalších praktických, ale i „zkrášlujících“ prvků. Menší nevýhodou může být možnost vypnutí v prohlížeči, jelikož běží na straně uživatele. Nicméně používá se také při vývoji desktop aplikací a

počítačových her a dnes ho lze použít i jako jazyk na straně serveru u webových aplikací. [6]

Autorem je Brendan Eich ze společnosti Netscape. Jazyk byl vytvořen v roce 1995 z důvodu větší interakce webu, jelikož v té době existovalo pouze HTML. Ačkoliv název může připomínat jazyk Java, tak s ním má pramálo společného, i když z něho částečně vychází. [6]

3.4 Back-end

Slovy back-end (volně přeloženo jako „zadní vrátka“) je myšlena ta část aplikace, kterou obyčejný uživatel nevidí. Je mu skryta a v ideálním případě k ní vůbec nemá přístup (pokud ano, aplikace není dostatečně zabezpečena). Jinak řečeno jde o programovací jazyky pracující na straně serveru (tzv. komunikují se serverem), pomocí nichž lze vytvářet dynamické webové stránky. Jedná se o logickou část aplikace, která pracuje v pozadí a ovládá frontendovou část aplikace podle příkazů uživatele. Příkladem back-end části webové aplikace může být zpracování požadavku uživatele na serveru. [8]

Mezi programovací jazyky pracující na straně serveru můžeme zařadit dnes nejvyužívanější PHP, dále ASP.NET a Python. [8]

3.4.1 PHP

PHP je programovací opensource (to znamená, že zdrojový kód je volně přístupný) skriptovací jazyk specializující se čistě pro backendové účely webových aplikací. Původní zkratka PHP byla Personal Home Page, později ale došlo ke změně na rekurzivní zkratku Hypertext Preprocessor. Syntaxí spadá do rodiny C jazyků, ale zpravidla je lehčí na pochopení než ostatní programovací jazyky v této skupině, takže je vhodný pro nováčky v programování. Mezi další jeho přednosti patří výkon (běží na něm i velké webové aplikace jako je Facebook), vysoká podpora databázových systémů (MySQL, PostgreSQL, Oracle, MongoDB) a velké množství vestavěných funkcí hodících se při vývoji webových aplikací.

Dnes (ale také v minulosti) patří jednoznačně mezi špičku back-end jazyků, jelikož okolo 80 % veškerých webů stojí právě na něm. Jeho skripty lze zahrnout do kódu HTML, od kterého se odděluje na začátku pomocí tagu „<?php“ a na konci „?>“. Např. výpis textu v PHP vypadá následovně:

<?php echo “Hello world“; ?> [14]

První verze jazyka vyšla roku 1995 a jejím tvůrcem je dánský programátor Rasmus Lerdorf. Nejnovější verzí je PHP 8, nicméně jde o sedmou verzi jazyka, protože PHP 6 nikdy nevyšla. [14]

3.4.2 Databázové technologie

Databáze a vše s ní související nepochybně pracují na straně serveru, proto jsou zařazeny pod back-end.

3.4.2.1 MySQL

MySQL je systém pro řízení báze dat (SŘBD), pomocí kterého jsou data (DB) v databázi řízena podle požadavků uživatele, dohromady pak tvoří databázový systém (DBS). Konkrétně MySQL je pak relačním SŘBD, což znamená, že databáze je založena na tabulkách (relace = tabulka). Není to tedy pouze uložiště dat, ale řeší spoustu dalších problémů jako je zabezpečení dat nebo tzv. ACID, což je zkratka pro slova Atomicity (nedělitelnost), Consistency (validita), Isolation (izolace) a Durability (trvanlivost):

- Atomicity = operace s daty v databázi se provádějí jako nedělitelné operace, tedy pokud část operace selže, operace se vůbec neprovede.
- Consistency = stav databáze po provedení operace je vždy konzistentní podle pravidel a omezení. Nikdy nemůže nastat, že by se databáze nacházela v nekonzistentním stavu.
- Isolation = operace se navzájem neovlivňují, jsou izolované.

- Durability = všechna upravovaná data jsou po provedení operace okamžitě zapsána na trvanlivé uložiště. [14, 15, 16]

Mezi výhody oproti jiným SŘBD patří zabezpečení dat, vysoký výkon díky dobré optimalizaci, možnost použití na platformách Windows a Linux, dále je zdarma pro aplikace s otevřeným zdrojovým kódem, pro komerční aplikace je pak za menší poplatek. Další výhodou je nepochybně to, že ho vlastní společnost Oracle specializující se na databázové systémy, která nabízí různé konzultace a školení. Jako určitou podporu lze vidět i v tom, že MySQL je dnes nejpoužívanější webový databázový systém a na internetu lze díky tomu nalézt spoustu návodů. [14, 15, 16]

3.4.2.2 SQL

Pro komunikaci s MySQL se používá dotazovací jazyk SQL (Structured Query Language), pomocí něhož lze vykonávat všemožné operace s daty uvnitř databáze. Operace s daty probíhá pomocí příkazů. Mezi ty základní příkazy lze zařadit ukládání (INSERT), upravování (UPDATE), výběr (SELECT) a mazání (DELETE) dat, u tabulek jsou to obdobné příkazy – vytvoření (CREATE), upravení (ALTER) a smazání (DROP) tabulky. Je využíván ke komunikaci s mnoha dalšími SŘBD jako je Oracle, PostgreSQL, Microsoft SQL atd. Jeho použití je velmi jednoduché, protože syntaxe jazyka se tvoří v podstatě jako reálné věty. Např. příkaz pro výběr všech dat z tabulky vypadá následovně:

```
SELECT * FROM [nazev tabulky];
```

V překladu tento příkaz znamená: vyber všechna data (* označuje všechna data) z dané tabulky. [15, 16, 17]

3.4.2.3 PhpMyAdmin

PhpMyAdmin neboli ve zkratce PMA je dnes nejpoužívanější prostředí pro obsluhu MySQL (program je přeložen do 72 jazyků). Je to webová aplikace ulehčující uživatelům spravovat své databáze prostřednictvím grafického rozhraní, tedy to znamená, že k

vytváření a správě relací není potřeba psát kód, ale lze využít formulářů a tlačítek, což správu databáze ještě velmi ulehčuje. [15, 16, 17]

3.5 OOP

Objektově orientované programování neboli OOP je moderní metodika pro vývoj softwaru. Klade důraz na přehlednost a znovupoužitelnost kódu, který je poskládán z komponent – tzv. objektů. Těmi jsou myšleny většinou jako objekty z reálného světa (např. dům, auto). Objekt má nějaké své atributy (jinak řečeno vlastnosti, např. auto je červené) a metody (např. auto umí jezdit). Dále jsou součástí třídy, pomocí kterých se vytvářejí instance, což jsou objekty předělané podle vzoru dané třídy, tedy jinak řečeno, třída definuje, jaké bude mít instance atributy a metody. Instance se však liší hodnotami, např. každé auto má svoji značku, jenže jedno auto má značku Škoda a druhé zase Volkswagen. [13, 14]

Další pilíře OOP jsou polymorfismus, dědičnost a zapouzdření. První z pojmů znamená, že různé objekty reprezentující např. dopravní prostředek se můžou chovat jinak při volání stejných metod (např. objekty auto a kolo zavoláme metodu „jeď“, oba se sice začnou pohybovat, ale každý jiným způsobem). Dědičnost lze použít při vytváření dceřiných tříd (podtříd), které dědí atributy a metody ze své rodičovské třídy (nadtřídy) a zároveň je lze rozšířit o další. Takže například auto a kolo by mohli být dceřinými třídami rodičovské třídy dopravní prostředek. Zapouzdření umožňuje skrýt dané metody, aby byly použitelné pouze pro třídu zevnitř, ty mají přístupovou hodnotu nastavenou na „private“. Naopak veřejně přístupné metody mají přístupovou hodnotu „public“. [13, 14]

3.6 Životní cyklus aplikace

Životní cyklus webové aplikace je několik etap, kterými aplikace postupně prochází. Jedná se o doporučené postupy při vývoji nejen webových aplikací, takže vývojáři se jimi sice řídit nemusí, nicméně vytváření programů pak může trvat déle a práce

na nich nemusí být tak efektivní. Tyto doporučené postupy se nazývají metodikami. Většina firem v praxi tyto doporučené metodiky upravuje podle svých potřeb. Životní cyklus aplikace má většinou tyto fáze:

- Plánování a příprava
- Analýza a návrh
- Implementace
- Zavedení do provozu
- Provoz a užívání
- Rozvoj a optimalizace

Po dokončení cyklu aplikace často končí opět na počátku, kdy přechází na vyšší úroveň (např. nové požadavky uživatelů, nutnost přejít na nové technologie). [18]

3.6.1 Plánování a příprava

V této fázi se řeší, jestli má vůbec smysl se aplikací zabývat. Pokud ano, řeší se plánování projektu (řeší se cíle, důvody, harmonogram projektu, atd.) a úvodní studie (studie proveditelnosti). Druhá jmenovaná studie konkrétněji stanovuje cíle aplikace, ustavuje nároky na projekt – personální, finanční a technologické. Také určuje řídící výbor projektu a pracovní skupiny, komunikaci mezi skupinami, atd. [18]

3.6.2 Analýza a návrh

V dalším kroku se v rámci analýzy hodnotí aktuální stav ve firmě a požadavky podniku na aplikaci. Dále se analyzuje stav a kvalita databáze. Hodnotí se také stávající aplikace, které se již provozují. [18]

Z analýzy následně vychází návrh aplikace, podle kterého by mělo dojít k výsledné implementaci aplikace. Dělí se na logickou a fyzickou (technologickou) úroveň. Logická úroveň specifikuje obsah, jednotlivé funkce, vstupní a výstupní data, interní vazby,

případně vazeb na ostatní aplikace. Technologická úroveň definuje potřebné technologie k implementaci programu. [18]

3.6.3 Implementace

Samotné implementaci neboli vytvoření aplikace předchází vytvoření prototypů, které důkladněji skutečné potřeby uživatelů a sníží rizika omylů při formulaci funkcionalit aplikace. Samotná implementace poté samozřejmě probíhá podle návrhu a platí, že čím lepší návrh je, tím efektivnější je práce vývojářů. Do vytvoření programu následně spadá také testování (často se aplikace před nasazením nejprve nahraje na testovací server, kde je následně otestována). Celý proces by měl být dokumentován pro případné změny v aplikaci. [18]

3.6.4 Zavedení do provozu

V této fázi dochází k migraci dat na server a zavedení aplikace do provozu. Migrace je pracovně a organizačně velmi náročná činnost, proto je nutné si nejprve stanovit plán pro nasazení celého programu. Ihned po nasazení je potřeba vyškolit zaměstnance, kteří budou aplikaci provozovat. [18]

3.6.5 Provoz a užívání

Po zavedení aplikace do provozu dochází k běžným údržbovým operacím, provoznímu servisu a tzv. Help desku (pomoc uživatelům v orientaci v aplikaci). Nepochybně důležitou úlohou během provozu je údržba počítačové sítě, aby nedocházelo k výpadkům aplikace. Dále může docházet k vytváření profilů uživatelům (pokud si je nevytvářejí uživatelé samy) a monitorování aplikace (poskytuje informace o případných vzniklých chybách, vytížení a dalších provozních statistikách). [18]

3.6.6 Rozvoj a optimalizace

Další rozvoj probíhá většinou ve stylu menších úprav. Pokud je však rozsah požadovaných změn, ať už z důvodu různých inovací nebo složité údržby příliš velký, dochází k zadání nového projektu a tím začíná cyklus od znovu. [18]

4 Vlastní práce

Vlastní práce se soustředí na vytvoření rezervačního systému Drive pro rezervaci jízd (praktických hodin) pro autoškolu. Aplikace bude tvořena od počátku analýzou, kde se zhodnotí aktuální stav a stanoví požadavky, dále návrhem aplikace a databáze, implementací, a nakonec otestováním vytvořeného systému.

4.1 Analýza

Analýza se zabývá popisem a zhodnocením nynějšího stavu a shrnuje, co by měla aplikace obsahovat. Autor bohužel neměl možnost porovnat jiné rezervační systémy zaměřující se na tuto problematiku.

4.1.1 Aktuální stav

Jelikož byl autor nedávno žákem autoškoly, může autenticky zhodnotit nynější stav. V dnešní době není součástí autoškoly žádný sofistikovaný systém pro rezervaci praktických hodin. Rezervace jízd probíhala buď na teoretických hodinách nebo telefonicky. Tím se na teoretických hodinách ztratilo spoustu času, navíc instruktor během hodiny často přijímal telefonáty od dalších žáků autoškoly s prosbou o zarezervování jízdy, a to hodinu dále zkracovalo. Tyto problémy by měla vyřešit nově vytvořená aplikace.

4.1.2 Požadavky na aplikaci

Požadavky (cíle) na aplikaci jsou následující:

- Autentizace všech uživatelů (aplikace bude řešena formou intranetu, přístup do ní budou mít tedy pouze žáci a instruktoři, kteří mají přihlašovací údaje).
- Tři typy účtů – žák, instruktor, admin.
- Vytvoření účtu instruktora, jeho editace a smazání adminem.

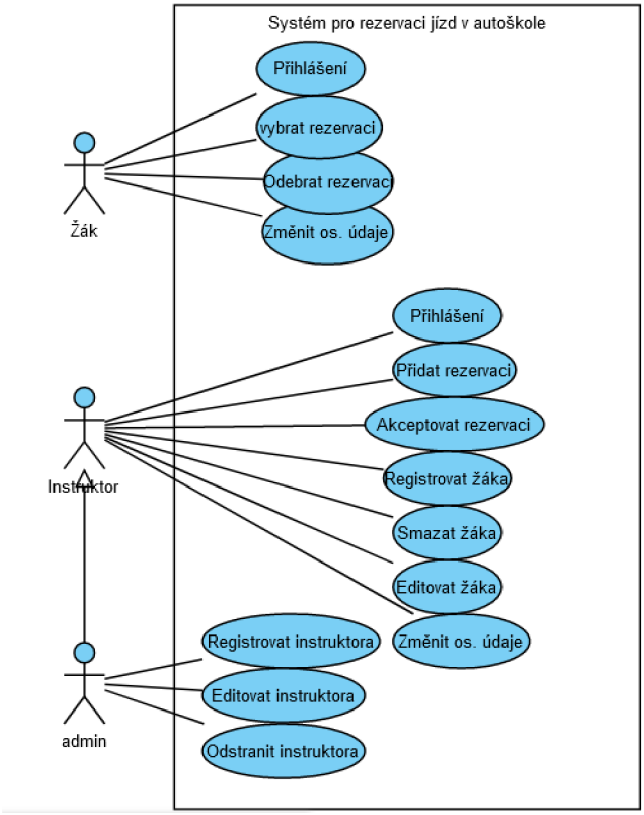
- Vytvoření účtu žáka, jeho editace a smazání adminem nebo instruktorem.
- Vytváření rezervací instruktorem nebo adminem.
- Výběr rezervací žákem.
- Žák bude moci vybírat pouze jízdy shodné se skupinou řidičského oprávnění, kterou si vybral na počátku (respektive, kterou si vybral v přihlášce do autoškoly).
- Přehled žáků na straně instruktora a admina.
- Přehled instruktorů na straně admina.
- Možnost editace vlastních údajů všemi uživateli aplikace.

4.2 Návrh

Návrh aplikace sestává z Use Case (případ užití) diagramu, což je diagram znázorňující funkcionality systému a zároveň ukazuje, kým mohou být spuštěny a z Use Case specifikace, která jednotlivé části aplikace popíše. Dále je součástí Wireframe (drátěný model) webu, který znázorňuje zjednodušený náhled programu. Nakonec je součástí databázový návrh, jež je představen pomocí ERD (Entity Relationship Diagram).

4.2.1 Use Case diagram

Aplikace má celkem tři vrstvy pravomocí, z nichž dědí pouze dvě, jak lze vidět na obrázku níže. Žák má možnost přihlásit se, vybrat rezervaci, odebrat rezervaci nebo změnit své osobní údaje. Tvoří však samostatnou vrstvu, nikdo z něho nedědí. Další je instruktor, který má možnost přihlásit se, přidat rezervaci, akceptovat rezervaci, registrovat žáka, editovat žáka, smazat žáka a změnit své osobní údaje. Z instruktora dědí admin, který má navíc pravomoc registrovat instruktora, editovat instruktora nebo ho odstranit.



Obrázek 1 - Use case diagram

4.2.2 Use Case specifikace

Use case specifikace se používá pro upřesnění jednotlivých funkcí aplikace. Součástí je popis funkcionalit, aktéři účastnící se případu užití, základní a alternativní tok, z nichž první předpokládá bezproblémový průběh akce, zatím co druhý předpokládá nesprávný průběh (např. uživatel zadá špatné heslo). Nicméně uživatel by měl být upozorněn k chybě a zpět naveden k základnímu toku. Součástí mohou být také podmínky pro spuštění a dokončení případu užití.

4.2.2.1 Přihlášení

Popis: umožňuje přihlášení uživatelů do systému.

Aktéři: všichni nepřihlášení uživatelé, systém.

Podmínky pro spuštění: uživatel není přihlášený.

Základní tok: uživatel zadá své přihlašovací údaje (pokud je instruktorem či administrátorem, zaškrtne ještě checkbox) a následně je pomocí autentizace vpuštěn do systému.

Alternativní tok: uživatel zadá špatné přihlašovací údaje (heslo nebo email), na to je upozorněn zprávou a zadává údaje znovu.

Podmínky pro dokončení: uživatel je autentizován a stává se přihlášeným.

4.2.2.2 Vytvoření rezervace

Popis: vytvoření nové rezervace v systému

Aktéři: instruktor (admin), systém

Podmínky pro spuštění: uživatel je přihlášen

Základní tok: zadání povinných dat – času, místa, skupiny řidičského oprávnění a případně poznámky (ta není povinná). Následně dojde k vytvoření nové rezervace.

Alternativní tok: autor rezervace nezadá jednu z povinných kolonek nebo zadá nepovolené znaky, na což je upozorněn.

Podmínky pro dokončení: uložení rezervace do databáze.

4.2.2.3 Úprava osobních údajů

Popis: umožňuje upravit své osobní údaje v systému.

Aktéři: všichni přihlášení uživatelé, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: uživatel úspěšně upraví své osobní údaje.

Alternativní tok: uživatel zadá nepovolené znaky nebo neplatný email. U změny hesla zadá neplatné původní heslo nebo se neshodující se nová hesla. Na tyto chyby je uživatel upozorněn a je mu umožněno zadat údaje znovu.

Podmínky pro dokončení: pozměněné údaje jsou systémem úspěšně uloženy v databázi.

4.2.2.4 Přehled rezervací (žák)

Popis: poskytuje přehled nevybraných rezervací na levé straně a přehled rezervací uživatelem vybraných na pravé straně.

Aktéři: žák, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: systém zobrazí dostupné rezervace a žákem vybrané rezervace.

4.2.2.5 Výběr rezervace

Popis: umožní uživateli vybrat rezervaci.

Aktéři: žák, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: uživatel vybere rezervaci z nabídky rezervací a potvrdí výběr.

Alternativní tok: v nabídce nejsou k dispozici žádné rezervace, nebo nedojde k potvrzení výběru.

Podmínky pro dokončení: rezervace je systémem přidělena danému uživateli.

4.2.2.6 Odebrání rezervace

Popis: umožní uživateli odebrat rezervaci.

Aktéři: žák, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: jízda začíná za více než dva dny – uživatel odebere rezervaci.

Alternativní tok: jízda začíná za dva dny a méně – uživatel již nemá možnost odebrat rezervaci.

Podmínky pro dokončení: systém v databázi uživateli odebere rezervaci.

4.2.2.7 Akceptace rezervace

Popis: uživatel potvrdí, zda rezervace (jízda) skutečně proběhla.

Aktéři: instruktor, admin, systém.

Podmínky pro spuštění: uživatel musí být přihlášený.

Základní tok: po ukončení jízdy uživatel potvrdí, zda jízda skutečně proběhla – v případě, že klikne na „ano“, přičte se jízda žákovi v databázi a zároveň se rezervace smaže, pokud klikne na „ne“, rezervace se pouze smaže.

Alternativní tok: uživatel nepotvrdí, zda jízda skutečně proběhla (což je chyba).

Podmínky pro dokončení: systém do databáze přičte odjetou jízdu k žákovi a smaže rezervaci, respektive pouze smaže jízdu (pokud k jízdě nedošlo).

4.2.2.8 Registrace nového žáka

Popis: umožňuje registrovat nového žáka do systému.

Aktéři: instruktor, admin, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: uživatel úspěšně registruje nového žáka.

Alternativní tok: uživatel nevyplní nějaké pole nebo zadá neplatný email – bude upozorněn a následně bude mít možnost to napravit.

Podmínky pro dokončení: systém nového žáka úspěšně uloží do databáze.

4.2.2.9 Editace žáka

Popis: umožní editaci žáka v systému.

Aktéři: instruktor, admin, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: uživatel úspěšně edituje žáka.

Alternativní tok: uživatel zadá neplatný email a telefonní číslo.

Podmínky pro dokončení: systém editaci úspěšně uloží do databáze.

4.2.2.10 Smazání žáka

Popis: umožní smazání žáka ze systému.

Aktéři: instruktor, admin, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: uživatel smaže žáka.

Podmínky pro dokončení: systém smaže žáka z databáze.

4.2.2.11 Registrace nového instruktora

Popis: umožňuje registraci nového instruktora.

Aktéři: admin, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: uživatel úspěšně registruje nového instruktora.

Alternativní tok: uživatel uvede neplatný email nebo telefonní číslo nebo zapomene vyplnit kolonku nutnou k registraci instruktora.

Podmínky pro dokončení: systém úspěšně uloží do databáze nového instruktora.

4.2.2.12 Editace instruktora (admina)

Popis: umožňuje uživateli editaci instruktora (admina).

Aktéři: administrátor, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: uživatel úspěšně edituje instruktora.

Alternativní tok: uživatel edituje neplatný email nebo telefonní číslo nebo nevyplní jedno z polí – systém uživatele napomene a uživatel může chybu napravit.

Podmínky pro dokončení: systém úspěšně uloží editaci uživatele do databáze.

4.2.2.13 Smazání instruktora (admina)

Popis: umožňuje smazat instruktora (admina).

Aktéři: administrátor, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

Základní tok: uživatel úspěšně smaže instruktora.

Podmínky pro dokončení: systém úspěšně smaže instruktora z databáze.

4.2.2.14 Odhlásit se

Popis: umožňuje odhlášení ze systému.

Aktéři: všichni přihlášení uživatelé, systém.

Podmínky pro spuštění: uživatel musí být přihlášen.

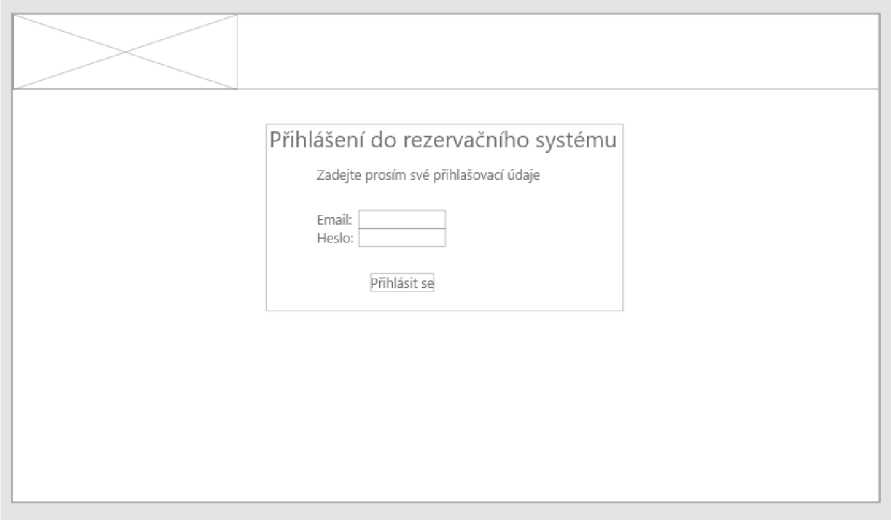
Základní tok: uživatel klikne na tlačítko signalizující odhlášení.

Podmínky pro dokončení: systém odhlásí uživatele.

4.3.1 Wireframe model

Wireframe (drátěný) model, tedy grafický návrh aplikace, byl vytvořen v programu Adobe XD, který je k používání zdarma.

4.3.1.1 Přihlášení



Obrázek 2 - Drátěný model přihlášení

U přihlášení bude systém od uživatele požadovat email a heslo. Tyto údaje (zejména heslo) uživatel získá při návštěvě prvních teoretických hodin v autoškole (heslo si pak bude moci změnit podle sebe dále v aplikaci). Po kliknutí na tlačítko „Přihlásit se“ se uživatel dostane do systému. Pokud zadá špatné údaje nebo nevyplní nějaké pole, bude na tuto skutečnost upozorněn.

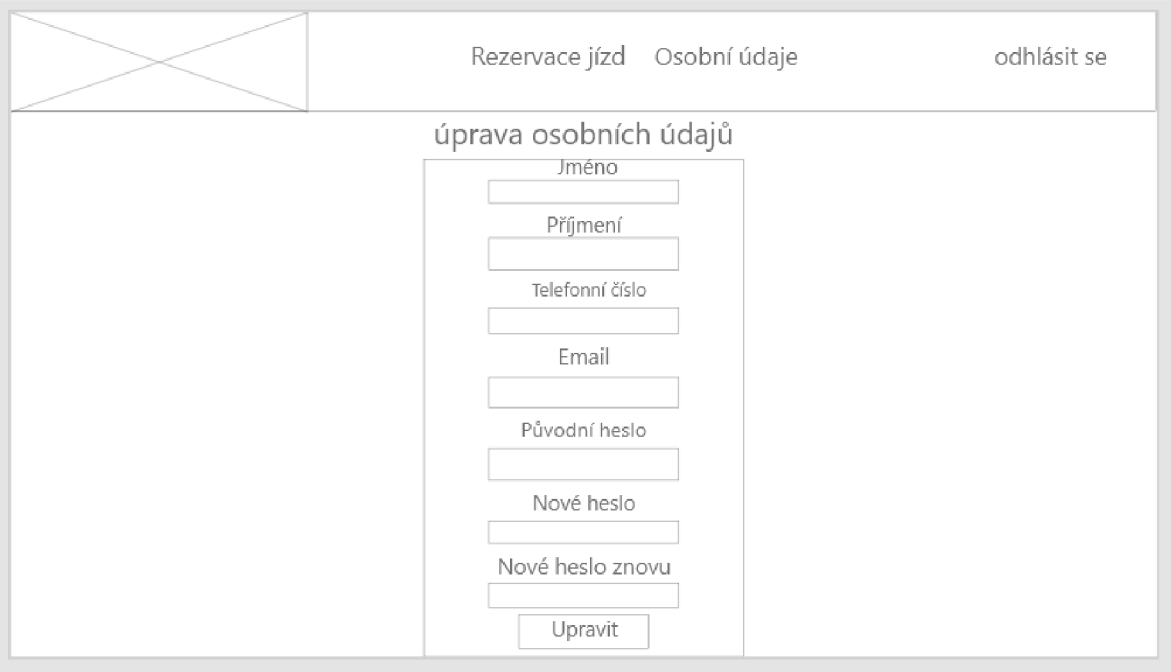
4.3.1.2 Výběr rezervací

Obrázek 3 - Drátěný model výběru rezervací

Takto vypadá aplikace po přihlášení z pohledu žáka autoškoly. Jde o stránku Rezervace jízd. Nahoře si lze všimnout navigace. V její levé části se bude nacházet logo, uprostřed přístup na stránky Rezervace jízd (kde se nyní nacházíme) a Osobní údaje. Vpravo bude tlačítko „Odhlásit se“.

Níže vlevo si bude moci žák rezervovat jízdu podle svých možností, jelikož každá rezervace musí mít zadaný den, čas, místo. Na pravé straně jsou pak jízdy, které má žák rezervovány a které může odebrat zpět, nicméně pouze více než dva dny před uskutečněním jízdy, protože poté již tlačítko „Odebrat rezervaci“ zmizí a jízdu tak musí uskutečnit nebo se s instruktorem domluvit jinak přes telefon, jelikož součástí rezervace je i jméno a telefon na příslušného instruktora. Nutno dodat, že žákovi se zobrazují pouze jízdy příslušné skupiny, jinak řečeno, pokud dělá řidičský průkaz skupiny B (na auto), budou se mu zobrazovat pouze rezervace této skupiny.

4.3.1.3 Úprava osobních údajů



Obrázek 4 - Drátěný model úpravy osobních údajů

Úprava osobních údajů zahrnuje možnost upravit jméno, příjmení, telefonní číslo, email a heslo. Po kliknutí na tlačítko „Upravit“ dojde k potvrzení změny údajů. Tato stránka je přístupná všem uživatelům bez ohledu, jestli jde o žáka, instruktora, či administrátora.

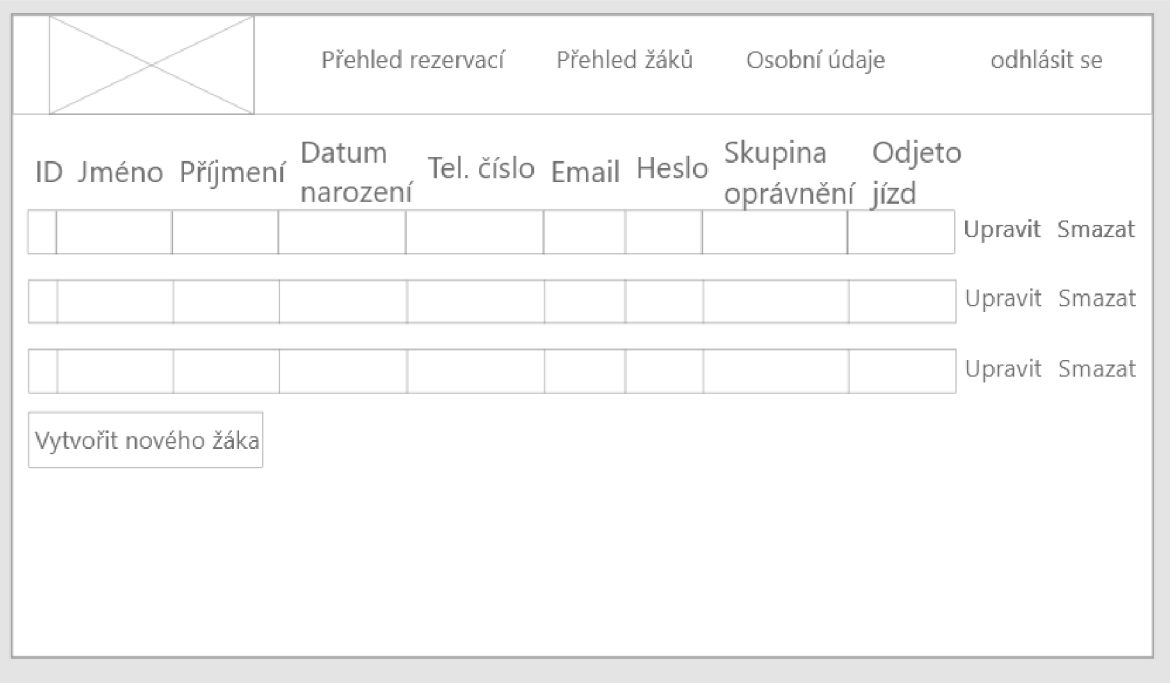
4.3.1.4 Přehled rezervací

Obrázek 5 - Drátěný model přehledu rezervací

Na této stránce Přehledu rezervací, která se promítne instruktorovi bezprostředně po přihlášení si lze všimnout, že v navigaci přibyl Přehled žáků, který vidí pouze instruktor (respektive admin).

Vlevo dole se nachází tlačítko „Přidat rezervaci“, pomocí kterého instruktor může přidat novou rezervaci (odkáže ho na stránku s formulářem pro přidání rezervace). V levém sloupci jsou pak rezervace, které instruktor vytvořil, ale ještě je žádný z žáků nevybral, proto je tu možnost zrušit je (smazat z databáze). Na pravé straně jsou rezervace žákem vybrané a také již proběhlé rezervace. Obsahují navíc také potřebné informace o žákovi. Rezervace na prvním místě ještě neproběhla, zatím co druhá již ano a instruktor musí potvrdit, zda se tak opravdu stalo. Pokud klikne na „Ano“, danému žákovi se přičte odjetá jízda v databázi a rezervace se smaže, pokud klikne na „Ne“, rezervace se pouze smaže z databáze.

4.3.1.5 Přehled žáků / Přehled instruktorů



Obrázek 6 - Drátěný model přehledu žáků

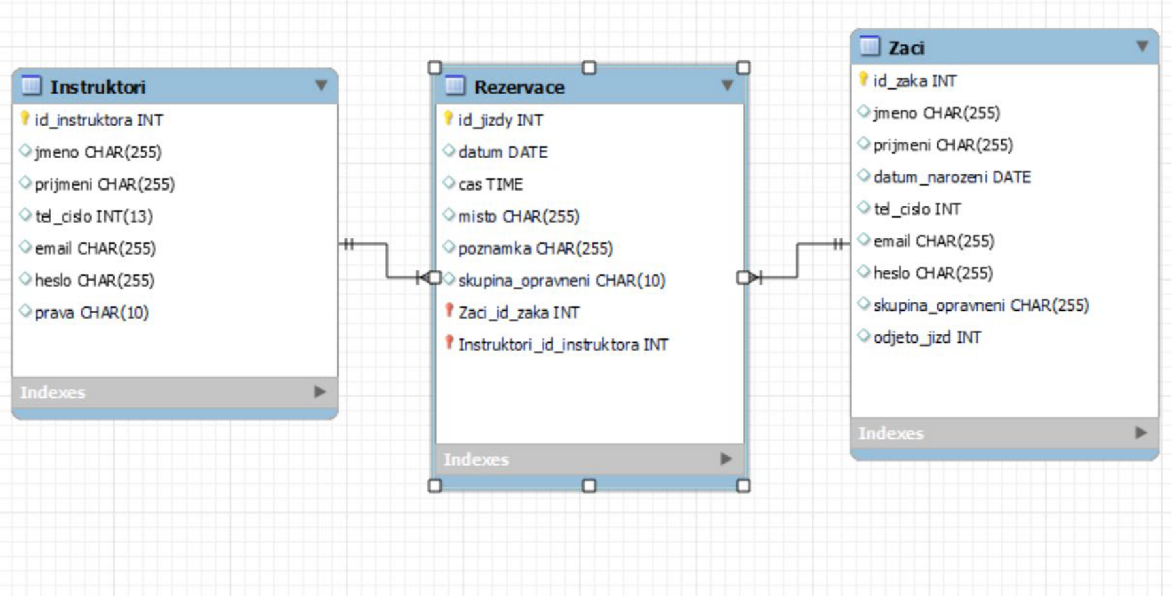
Další stránkou, kterou má instruktor k dispozici, je Přehled žáků. Měla by obsahovat veškeré informace o žákovi, které jsou k dispozici. Napravo se nachází tlačítko „Upravit“, pomocí kterého dojde k editaci žáka a „Smazat“, jež smaže žáka ze systému. Tlačítko „Vytvořit nového žáka“ otevře formulář, kde bude možné registrovat nového žáka. Tento formulář se bude podobat tomu z Úpravy osobních údajů, proto zde ani nebude uveden. Na konec nutno dodat, že stejný přehled bude mít k dispozici také administrátor o instruktorech včetně zmíněných tlačítek.

4.3.2 Návrh databáze

Návrh databáze byl vytvořen v programu MySQL Workbench jako ER diagram. Data jsou uloženy ve třech tabulkách. Základem je tabulka Rezervace, ve které je uloženo id_rezervace, které je primárním klíčem, a tedy specifikuje konkrétní rezervaci od ostatních. Dále jsou zde datum, čas a místo, tedy kdy a kde proběhne počátek jízdy. Poznamka slouží k dodatečnému obsahu pro žáka a skupina_opravneni (skupina řidičského oprávnění) definuje, jestli to bude jízda na motorce (skupina A) nebo autem (skupina B). Zaci_id_zaka a Instruktori_id_instruktora jsou cizí klíče propojující tabulky Instruktori a Zaci pomocí vazby 1: N, to znamená, že žák s instruktorem může mít více rezervací, zatím co rezervace může mít pouze jednoho instruktora anebo žáka.

Tabulka Instruktori zaznamenávající údaje o instruktorech opět obsahuje primární klíč id_instruktora. Dále obsahuje položky jmeno, prijmeni, tel_cislo, email, heslo a prava instruktora.

Poslední tabulka Zaci obsahuje informace o žácích autoškoly. Na počátku má primární klíč id_zaka. Dále položky jmeno, prijmeni, datum_narozeni, tel_cislo, email, heslo, skupina_opravneni a odjeto_jizd.



Obrázek 7 - Návrh databáze

4.4 Implementace

Pro implementaci aplikace bylo zvoleno vývojové prostředí PhpStorm. Nejdříve však musel být nainstalován XAMPP, což je softwarový balíček obsahující webový server Apache, programovací jazyk PHP a databázi MariaDB (databázový systém od tvůrců MySQL). Databáze je vytvořena v prostředí PhpMyAdmin.

Nejdříve byla implementována kostra aplikace v podobě formulářů podle drátěného modelu pomocí jazyka HTML. Jako první byl implementován formulář pro přihlášení, který lze vidět níže:

```
<label for="email"> Přihlašovací email:  </label><br />
<input name="email" id="email" type="text"><br />
<br />
<label for="heslo">Heslo:</label><br />
<input name="heslo" id="heslo" type="password"> <br />
<br />
<label for="vyber">Zaškrtněte, jste-li instruktor: </label>
<input name="vyber" id="vyber" class="form-check-input" type="checkbox"> <br />
<br />
<input name="odeslat" class="btn btn-primary" id="prihlasit" type="submit"
value="Přihlásit se"> <br />
```

Obrázek 8 – Přihlašovací HTML formulář

Labely slouží pro popis daného inputu. Input s typem „text“ je pak samotné formulářové pole. Typ „password“ skryje text a jeho použití je u hesel, dále typ „checkbox“ zobrazuje zaškrtávací pole. Nakonec tlačítko s textem „Přihlásit se“ má typ „submit“. Na podobný způsob byly vytvořeny všechny formuláře.

4.4.1 Implementace databáze

	Tabulka	Operace	Řádků	Typ	Porovnávání	Velikost	Navíc
☐	instruktori	Projít Struktura Vyhledávání Vložit Vyprázdnit Odstranit	8	InnoDB	utf32_czech_ci	80,0 KiB	-
☐	rezervace	Projít Struktura Vyhledávání Vložit Vyprázdnit Odstranit	6	InnoDB	utf32_czech_ci	96,0 KiB	-
☐	zaci1	Projít Struktura Vyhledávání Vložit Vyprázdnit Odstranit	8	InnoDB	utf32_czech_ci	96,0 KiB	-
	3 tabulky	Celkem	22	InnoDB	utf32_czech_ci	272,0 KiB	0 B

Obrázek 9 - Tabulky databáze

Po vytvoření formulářů autor vytvořil databázi v PhpMyAdmin. Název databáze je Autoškola a skládá se ze tří tabulek, jejichž přehled je na obrázku výše.

Na obrázku níže je pro příklad struktura tabulky „rezervace“. Vlevo lze vidět názvy jednotlivých položek. Ikonka zlatého klíče znázorňuje primární klíč, stříbrné ikonky klíčů jsou zase cizí klíče propojující tabulky instruktora a žáka. Dále je zde zajímavý „Typ“, který určuje datový typ položky, popřípadě jeho délku.

#	Název	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Komentáře	Další	Operace
1	id_jizdy	int(10)			Ne	Žádná		AUTO_INCREMENT	Změnit Odstranit Více
2	datum	date			Ne	Žádná			Změnit Odstranit Více
3	cas	time			Ne	Žádná			Změnit Odstranit Více
4	misto	char(255)	utf32_czech_ci		Ne	Žádná			Změnit Odstranit Více
5	poznamka	char(255)	utf32_czech_ci		Ne	Žádná			Změnit Odstranit Více
6	id_instruktor	int(11)			Ne	Žádná			Změnit Odstranit Více
7	id_zak	int(11)			Ano	NULL			Změnit Odstranit Více
8	skupina_opravneni	varchar(1)	utf32_czech_ci		Ne	Žádná			Změnit Odstranit Více

Obrázek 10 – Struktura databázové tabulky rezervace

4.4.2 Implementace PHP

Pro implementaci aplikace byl zvolen objektově orientovaný přístup. Nejprve autor vytvořil všechny potřebné třídy, což je celkem šest tříd (pro každou třídu jeden soubor): instruktor, zak, rezervace, instruktorDB, zakDB a rezervaceDB. První tři třídy slouží především k nastavení a kontrole hodnot, popřípadě vytváření formulářů v html. Pro ukázku je zde kód nastavující hodnoty žáka, které se následně ukládají do databáze, a proto je třeba tyto hodnoty ošetřit proti možným útokům zvenčí, zejména proti XSS (Cross-Site Scripting) útoku (snaha poškodit PHP kód) a SQL injection (snaha poškodit databázi). Proti prvnímu typu útoku se data ošetřují pomocí funkce preg_match, které nejprve zadáme regulární výraz povolující pouze určité znaky a následně ověříme vkládaný výraz. Pokud tedy uživatel zadá nepovolený znak, vyskočí na něho hláška implementovaná Javascriptem, tento způsob je ale využit především u editace žáků nebo instruktorů. Funkce htmlspecialchars zase pomocí ENT_QUOTES převede html znaky na text, čímž znemožní zadat php skript do formuláře. K vidění je zde ještě funkce password_hash, která pomocí PASSWORD_DEFAULT zahashuje zadané heslo. Výsledkem je otisk hesla, který je uložen v databázi a který se vždy verifikuje s heslem zadaným uživatelem při autentizaci (pomocí PASSWORD_VERIFY).

```
public function nastavHodnoty($id_zaka, $jmeno, $prijmeni, $datum_narozeni, $tel_cislo, $email, $skupina_opravneni, $odjeto_jizd, $heslo)
{
   $this->id_zaka = $id_zaka;
   $redexp = "/[A-Za-z0-9ĚŠČŘŽÝÁÍÉŇŤěščřžýáíéňť]+$/";
   if(preg_match($redexp,$jmeno))
   {
      $this->jmeno = htmlspecialchars($jmeno, ENT_QUOTES);
   }
   else {
      echo "<script>
      alert('Neplatné jméno!');
      </script>" ;
      return false;
   }
   $redexp = "/[A-Za-z0-9ĚŠČŘŽÝÁÍÉŇŤěščřžýáíéňť]+$/";
   if(preg_match($redexp,$prijmeni))
   {
      $this->prijmeni = htmlspecialchars($prijmeni, ENT_QUOTES);
   }
   else {
      echo "<script>
      alert('Neplatné příjmení!');
      </script>" ;
      return false;
   }
   $this->datum_narozeni = htmlspecialchars($datum_narozeni, ENT_QUOTES);
   $redexp = "/[+]?[0-9. -]{9,}$/";
   if(preg_match($redexp,$tel_cislo))
   {
      $this->tel_cislo = htmlspecialchars($tel_cislo, ENT_QUOTES);
   }
   else {
     echo "<script>
     alert('Neplatné telefonní číslo!');
     </script>" ;
     return false;
   }

   if (filter_var($email, FILTER_VALIDATE_EMAIL))
   {
      $this->email = filter_var($email, FILTER_VALIDATE_EMAIL);
   }
   else
   {
      echo "<script>
     alert('Neplatný email!');
     </script>" ;
      return false;
   }
   $this->heslo = password_hash($heslo, PASSWORD_DEFAULT);
   $redexp = "/[aA]|[bB]{1}$/";
   if(preg_match($redexp,$skupina_opravneni))
   {
      $this->skupina_opravneni = htmlspecialchars($skupina_opravneni, ENT_QUOTES);
   }
   else {
      echo "<script>
      alert('Platná je pouze skupina řidičského oprávnění A nebo B!');
      </script>" ;
      return false;
   }
   $this->odjeto_jizd = $odjeto_jizd;
   return true;
}
```

Obrázek 11 - Implementace metody nastavHodnoty

Poslední tři třídy pak slouží čistě pro komunikaci s databází. Pomocí kódu níže se třída pomocí konstruktoru připojí k databázi. Autor konkrétně zvolil připojení přes třídu PDO, která je asi nejvyužívanější, ale je samozřejmě více možných způsobů (např. také existuje hodně využívaná funkce mysqli()).

```
class RezervaceDB
{
   private $spojeni;
   private $databaze = "autoskola";
   private $jmeno = "root";
   private $heslo = "";

   public function __construct()
   {
      $this->spojeni = new PDO("mysql:dbname=$this->databaze;charset=UTF8;host=localhost",
         $this->jmeno, $this->heslo);
   }
```

Obrázek 12 - Připojení k databázi

Dále je již možné databázi pokládat příkazy podle libosti. Například na obrázku níže můžeme vidět metodu umožňující uložení nové rezervace do databáze. Metoda v parametru získá objekt rezervace pomocí proměnné $rezervace. Následuje dotaz, který je položen v SQL jazyce, pro bezpečnost ale nejsou hodnoty vloženy přímo do dotazu, ale jsou vloženy pomocí funkce bindParam (do dotazu jsou vloženy pouze parametry, které tato funkce nahradí již skutečnými proměnnými). Ještě předtím však dojde ke spojení s databází a přípravě dotazu pomocí funkce prepare. Nakonec pomocí funkce execute dojde k provedení dotazu. Na tomto principu jsou založeny všechny funkce v těchto třídách.

```
public function ulozeniRezervace($rezervace)
  {
    $dotaz = "INSERT INTO rezervace (id_jizdy, datum, cas, misto, poznamka, id_instruktor, skupina_opravneni)
          VALUES (NULL,:datum,:cas,:misto,:poznamka, :id_instruktor, :skupina_opravneni
                )";

    $sql = $this->spojeni->prepare($dotaz);
    $sql->bindParam(":datum", $rezervace->datum);
    $sql->bindParam(":cas", $rezervace->cas);
    $sql->bindParam(":misto", $rezervace->misto);
    $sql->bindParam(":poznamka", $rezervace->poznamka);
    $sql->bindParam(":id_instruktor", $_SESSION["id_instruktora"]);
    $sql->bindParam("skupina_opravneni", $rezervace->skupina_opravneni);

    if ($sql->execute()) {
      return $this->spojeni->lastInsertId();
    } else {
      return false;
    }
  }
```

Obrázek 13 - Implementace metody ulozeniRezervace

Těchto metod byla vytvořena celá řada. Jak už bylo řečeno, na stejném způsobu bylo implementováno také mazání rezervací, vkládání nových žáků, jejich editace a smazání, to samé u instruktorů. Dále autor podobnou technikou implementoval také výpisy žáků, instruktorů a rezervací, doplněných o různé podmínky. Například metoda na obrázku níže vypíše všechny rezervace, kde se shoduje cizí klíč „id_zak“ rezervace s primárním klíčem žáka, což je využito při výpisu rezervací vybraných daným žákem. Funkce setFetchMode potom načítá jednotlivé rezervace podle třídy „rezervace“.

```
public function vyberRezervaciProZaka()
   {
      $dotaz = "SELECT jmeno, prijmeni,tel_cislo, id_jizdy, id_instruktor,datum, cas, misto, poznamka,
     rezervace.skupina_opravneni FROM rezervace, instruktori WHERE rezervace.id_instruktor =
instruktori.id_instruktora
     AND id_zak IS NOT NULL AND id_zak = :id_zak";
      $sql = $this->spojeni->prepare($dotaz);
      $sql->bindParam(":id_zak", $_SESSION['id_zaka']);
      $sql->execute();
      $sql->setFetchMode(PDO::FETCH_CLASS, "rezervace");
      return $sql->fetchAll();
   }
```

Obrázek 14 - Implementace metody vyberRezervaciProZaka

Zbytek aplikace obsahuje soubory s jednotlivými stránkami, kam se tyto metody volají, což ukazuje kód níže. Do proměnné $rezervace uložíme metodu popsanou výše (metoda se volá z objektu $db, o tom je pojednáno níže) a následně přes příkaz foreach vypíšeme jednotlivé rezervace do html kódu, jež je součástí foreach. Zde je vidět ještě jedna funkce „dnydorezervace“, která udává, že žák může zrušit rezervaci nejpozději dva dny před začátkem jízdy, poté tlačítko „Zrušit rezervaci“ zmizí.

```
$rezervace = $db->vyberRezervaciProZaka();
foreach($rezervace as $rezervaci)
{
   $rez = new rezervace();
if ($rez->dnydorezervace($rezervaci->datum, $rezervaci->cas)<2)
{
   echo "

<div class='list-group-item list-group-item-action'>
          <li class='list-unstyled' >Datum jízdy: $rezervaci->datum </li>
          <li class='list-unstyled' >Čas: $rezervaci->cas </li>
          <li class='list-unstyled' >Místo: $rezervaci->misto </li>
          <li class='list-unstyled' >Poznámka: $rezervaci->poznamka </li>
          <li class='list-unstyled' >Jméno instruktora: $rezervaci->jmeno $rezervaci->prijmeni</li>
          <li class='list-unstyled' >Telefonní číslo instruktora: $rezervaci->tel_cislo </li>
          </div>
";
}
else
{
   echo "
<div class='list-group-item list-group-item-action'>
          <li class='list-unstyled'>Datum jízdy: $rezervaci->datum </li>
          <li class='list-unstyled' >Čas: $rezervaci->cas </li>
          <li class='list-unstyled' >Místo: $rezervaci->misto </li>
          <li class='list-unstyled' >Poznámka: $rezervaci->poznamka </li>
          <li class='list-unstyled' >Jméno instruktora: $rezervaci->jmeno $rezervaci->prijmeni</li>
          <li class='list-unstyled' >Telefonní číslo instruktora: $rezervaci->tel_cislo </li>
          <a href='odebrat_rezervaci.php?id=$rezervaci->id_jizdy' class='btn btn-secondary'> Zrušit rezervaci</a>
          </div>
";
}
}
```

Obrázek 15 - Ukázka výpisu rezervací

Pro využívání těchto metod je však nejdříve nutné připojit třídní soubory k souborům, kde se tvoří jednotlivé stránky, což lze pomocí funkce „include“ („include_once“ pak zajistí, že se soubor připojí pouze jednou, aby se zabránilo chybám, což zachycuje následující obrázek). Pomocí posledních dvou řádků na dalším obrázku se následně objekty daných tříd a na ty se volají jednotlivé funkce z tříd.

```php
<?php
session_start();
if(!isset($_SESSION["id_instruktora"]))
{
   header("Location:index.php ");
}
include_once "instruktorDB.php";
include_once "instruktor.php";
$db= new InstruktorDB();
$instruktor = new Instruktor();
```

Obrázek 16 - Připojení tříd a vytvoření objektů

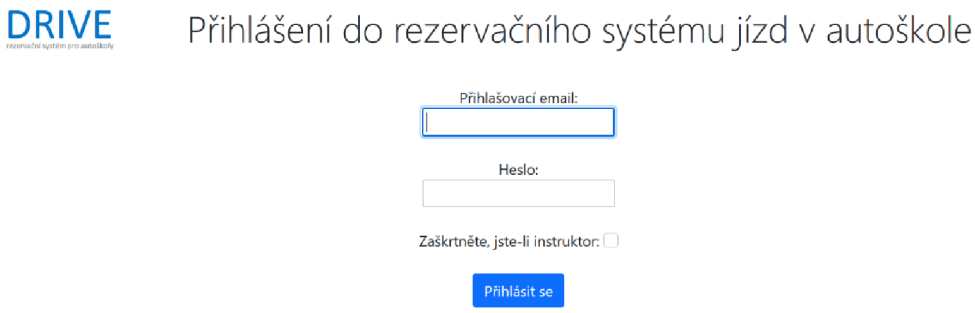
Na tomto obrázku je zajímavá ještě jedna věc a to sessiony. Při přihlášení totiž dochází k načtení „id_instruktora“, respektive „id_zaka“ do superglobální proměnné $_SESSION, kterou lze použít kdekoliv v rámci aplikace, kde je na počátku souboru zavolána funkce „session_start()“. Jedním z případů, kde se toto pak dále využívá je právě ten na obrázku kde, pokud není nastavena $_SESSION instruktora (uživatel tedy není přihlášen), je uživatel přesunut k přihlášení (to je soubor index.php). Tím se zabrání přístupu neoprávněným uživatelům do aplikace skrze url adresu.

4.4.3 Testování

Testování probíhalo vždy po implementaci určité části programu (většinou nějaké metody nebo funkce). Testy zahrnovaly pouze manuální procházení aplikace, přitom byla snaha projít všechny možné scénáře, které mohou nastat, aby se předešlo co největšímu množství chyb. Pokud daná část aplikace nefungovala, přistoupilo se k její opravě, následně se opět testovala. Tento cyklus se opakoval vždy do doby, než otestování proběhlo úspěšně.

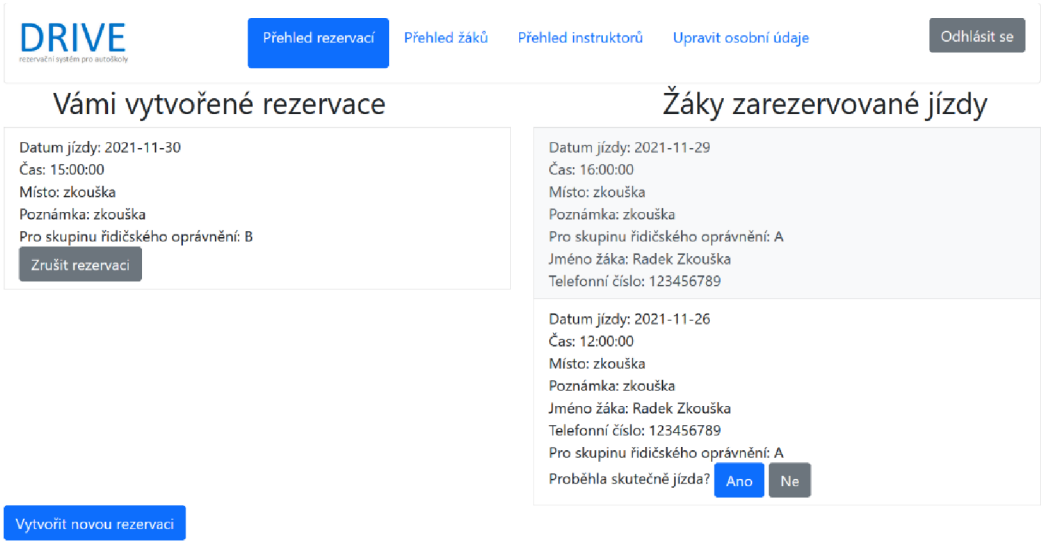
4.4.4 Zhotovená aplikace

Tato kapitola obsahuje popis funkcí a přehled finálních stránek.



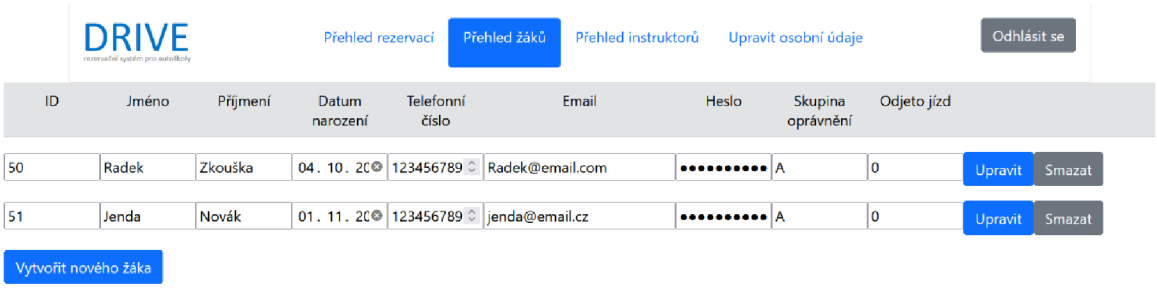
Obrázek 17 - Přihlášení do aplikace

Výše můžeme vidět přihlášení do aplikace. Přístup mají pouze uživatelé s platnými přihlašovacími údaji. Po zadání neplatných přihlašovacích údajů vyskočí uživateli pod formulářem chyba.



Obrázek 18 - Přehled rezervací z náhledu admina

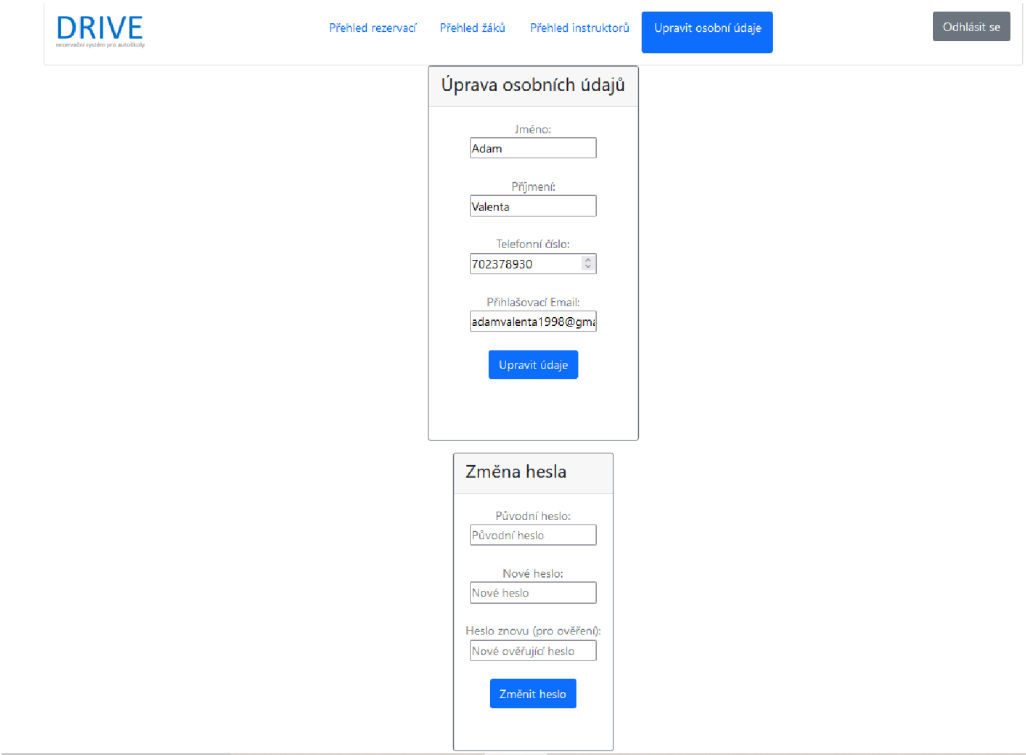
Výše vidíme stránku, která se načte instruktorovi po úspěšném přihlášení. Nahoře se nachází navigační lišta, pomocí které je schopen se přesunout na další stránky, úplně vpravo je možnost odhlásit se z aplikace. Níže se nachází samotné rezervace. Vlevo jsou rezervace, které instruktor vytvořil, ale ještě nejsou nikým rezervované (lze je také tlačítkem „Zrušit rezervaci“ smazat). Vpravo se nachází rezervace, které jsou již vybrané žáky, obsahují navíc jméno žáka a jeho telefonní číslo. První rezervace ještě neproběhla, zatím co druhá již ano, a proto instruktor musí ještě potvrdit, zda skutečně proběhla (pokud klikne na „Ano“, v databázi se přičte počet odjetých jízd o jedna, pokud klikne na „Ne“, jízda se pouze smaže). Pomocí tlačítka vlevo dole lze vytvořit novou rezervaci.



Obrázek 19 - Přehled žáků z náhledu admina

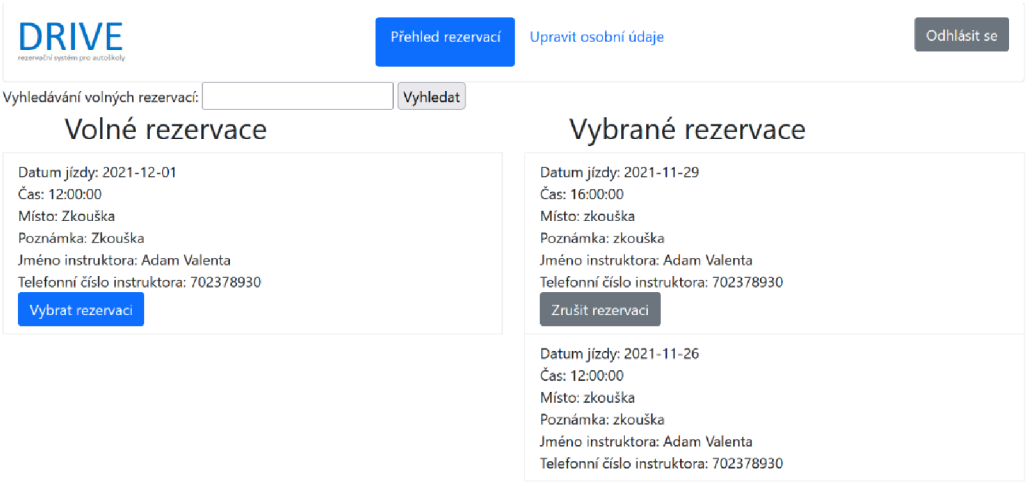
Na dalším obrázku je přehled žáků. Žáky lze upravovat přímo v tabulce, jelikož jde o formulářová pole, stačí tedy upravit údaj a potvrdit kliknutím na tlačítko „Upravit“. Pouze ID nelze upravovat, z důvodu možného poškození databáze. Hned vedle je tlačítko „smazat“ pomocí kterého dojde ke smazání celého záznamu. Podobně jako v přehledu rezervací, také zde je vlevo dole tlačítko pro vytvoření nového žáka. Kliknutím na toto tlačítko se zobrazí stránka s formulářem pro registraci nového žáka.

Jelikož jde o účet admina, vedle přehledu žáků je také přehled instruktorů (včetně adminů), který je obdobný.



Obrázek 20 - Úprava osobních údajů z náhledu admina

Dále je výše k vidění stránka umožňující úpravu osobních údajů včetně hesla. Stejné formuláře pro úpravu údajů má k dispozici také žák.



Obrázek 21 - Přehled rezervací z náhledu žáka

Jako poslední ukázku autor vybral přehled rezervací ze strany žáka. Ten má k dispozici také vyhledávání, jelikož se při reálném nasazení předpokládá větší vytíženost. Jak si lze všimnout, opět existují tři druhy rezervací. Volné rezervace představují všechny volné rezervace (tzv. zatím jim nebylo přiřazeno Id_zaka), které se shodují s řidičským oprávněním žáka (tedy žákovi, který dělá řidičský průkaz na auto se zobrazí pouze volné rezervace s řidičským oprávněním skupiny B). Dále je implementována metoda, která odstraní z volných rezervací (smaže je) všechny ty, které jsou datem zastaralé. Vpravo se nachází rezervace, které má žák vybrané. Ty, které obsahují tlačítko „Zrušit rezervaci“, jsou od datumu uskutečnění vzdáleny více než dva dny a lze je tedy ještě zrušit.

5 Výsledky a diskuse

5.1 Výsledek práce

Výsledkem vlastní práce je aplikace Drive. Tento rezervační systém pro rezervaci jízd v autoškole splňuje všechny cíle předeslané v analýze a zadání. Jsem přesvědčen, že po menších úpravách a spíše konkrétních požadavcích konkrétních instruktorů (aplikace je implementována způsobem, který takové úpravy umožňuje) je aplikace schopna jít do provozu nějaké malé autoškoly (autoškola do cca tří instruktorů). Avšak je zde určitě velké množství způsobů, jak aplikaci ještě zlepšit.

Jedna z menších žádoucích úprav by určitě bylo přidat stránkování (paginaci) k výběru rezervací, přehledu žáků a přehledu instruktorů, aby se předešlo případnému zahlcení webu. Dále by se hodilo sofistikovanější vyhledávání rezervací ze strany žáka, protože je nutné si uvědomit, že v reálné situaci, kdy každý z instruktorů bude mít naplánovány jízdy např. na týden dopředu budou v přehledu rezervací žáka desítky rezervací. Systém v současné době povoluje pouze nejrozšířenější skupiny řidičských oprávnění, a to skupiny A (na motorku) a B (na auto), nicméně rozšíření o další skupiny opět není žádný velký problém. Další, trochu větší změnou, by byla možnost přidat adminovi (respektive i instruktorovi) přehled všech rezervací, tato změna by byla opravdu žádoucí u už trochu větších autoškol, které mají řekněme více než tři instruktory a více, aby majitel autoškoly (admin) měl přehled o tom, jestli jsou počty jízd na daný týden konzistentní. Již opravdu velkým upgradem by bylo udělat z této aplikace nejen rezervační systém, ale celé stránky pro autoškolu včetně materiálů k teoretické výuce a zavedení otázek k teoretickým zkouškám.

6 Závěr

Nejprve jsem nastudoval potřebné zdroje k napsání teoretické práce. V úvodu jsem sdělil o čem tato práce bude, následně napsal teoretickou část, která zahrnovala potřebné technologie pro implementaci a zprovoznění webové aplikace jako je internet a služba WWW. Následně byly popsány technologie front endu, kam patří jazyky HTML, CSS (včetně frameworku Bootstrap) a Javascript. Poté následoval back-end, který zahrnuje programovací jazyk PHP a databázové technologie MySQL, SQL a PhpMyAdmin. Poté byla popsána problematika programovací techniky OOP následovaná životním cyklem aplikace.

V praktické části došlo k převedení teoretických znalostí do praxe, a to v podobě vytvoření aplikace Drive, která splňuje podobu rezervačního systému pro rezervaci jízd v autoškole. Nejprve však došlo k analýze, kde se stanovily cíle aplikace a zhodnotil aktuální stav. Následně byl vytvořen návrh aplikace pomocí Use Case, drátěného modelu a návrhu databáze. Poté se přistoupilo k samotné implementaci systému. Nejdříve byly vytvořeny formuláře pomocí jazyka HTML, poté databáze v prostředí PhpMyAdmin. Následně došlo k implementaci PHP kódu, a nakonec byl dodán konečný design pomocí CSS frameworku Bootstrap. Současně během (a také po dokončení) implementace programu docházelo k testování jednotlivých funkcí aplikace.

7 Seznam použitých zdrojů

1. Struktura Internetu. *Jaknainternet.cz* [online]. [cit. 2021-11-08]. Dostupné z: https://www.jaknainternet.cz/page/1795/struktura-internetu/
2. Model TCP/IP. *Ijs2.8u.cz* [online]. [cit. 2021-11-08]. Dostupné z: http://ijs2.8u.cz/index.php?option=com_content&view=article&id=14:model-tcp-ip&catid=9&Itemid=120
3. Historie Internetu. *Ijs2.8u.cz* [online]. [cit. 2021-11-08]. Dostupné z: http://ijs2.8u.cz/index.php?option=com_content&view=article&id=32&Itemid=134
4. Na počátku byl ARPANET .... *Earchiv* [online]. [cit. 2021-11-08]. Dostupné z: http://www.earchiv.cz/a95/a504c502.php3
5. HTML5 od A do Z - Online kurz. *itnetwork* [online]. [cit. 2021-11-08]. Dostupné z: https://www.itnetwork.cz/html-css/html5
6. Lekce 1 - Úvod do JavaScriptu. *Itnetwork* [online]. [cit. 2021-11-08]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk
7. Difference between Website and Web Application (Web App). *Guru99* [online]. [cit. 2021-11-08]. Dostupné z: https://www.guru99.com/difference-web-application-website.html
8. Front end vs. Back end - jaký je mezi nimi rozdíl? *Apitree* [online]. [cit. 2021-11-08]. Dostupné z: https://www.apitree.cz/blog/front-end-vs-back-end-jaky-je-mezi-nimi-rozdil
9. Lekce 1 - Úvod do CSS frameworku Bootstrap. *Itnetwork* [online]. [cit. 2021-11-08]. Dostupné z: https://www.itnetwork.cz/html-css/bootstrap/kurz/uvod-do-css-frameworku-bootstrap
10. CSS. *Jakpsatweb* [online]. [cit. 2021-11-08]. Dostupné z: https://www.jakpsatweb.cz/css/
11. Webové stránky krok za krokem - Online kurz. *Itnetwork* [online]. [cit. 2021-11-08]. Dostupné z: https://www.itnetwork.cz/html-css/webove-stranky
12. Building Twitter Bootstrap. *Alistapart* [online]. [cit. 2021-11-08]. Dostupné z: https://alistapart.com/article/building-twitter-bootstrap/
13. Objektově orientované programování a evoluce vývoje softwaru. *Itnetwork* [online]. [cit. 2021-11-08]. Dostupné z: https://www.itnetwork.cz/navrh/objektove-orientovane-programovani-a-evoluece-vyvoje-softwaru
14. WELLING, Luke a Laura THOMSON. *Mistrovství PHP a MySQL*. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.
15. VOSTROVSKÝ, Václav. *Vytváření databází v ORACLE*. Praha: Česká zemědělská univerzita v Praze, Provozně ekonomická fakulta, 2004. ISBN 978-80-213-1191-6.

16. *Lekce 1 - MySQL krok za krokem: Úvod do MySQL a příprava prostředí* [online]. [cit. 2021-11-09]. Dostupné z: https://www.itnetwork.cz/mysql/mysql-tutorial-uvod-a-priprava-prostredi
17. *Lekce 2 - MySQL krok za krokem: Vytvoření databáze a tabulky* [online]. [cit. 2021-11-09]. Dostupné z: https://www.itnetwork.cz/mysql/mysql-tutorial-vytvoreni-databaze-a-tabulky
18. POUR, Jan. *Podniková informatika: Počítačové aplikace v podnikové a mezipodnikové praxi - 3., aktualizované vydání* [online]. Praha: Grada, 2015 [cit. 2021-11-09]. ISBN 978-802-4799-186.

8 Přílohy

Odkazovaný seznam příloh