**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**

# Bakalářská práce

## Kadeřnický informační systém na webu s rezervačním systémem

**Marek Musil**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Marek Musil

Informatika

Název práce

**Kadeřnický informační systém na webu s rezervačním systémem**

Název anglicky

**Hairdressing information system on the web with a reservation system**

---

**Cíle práce**

Cílem bakalářské práce je vytvoření webu pro kadeřnický salon s informačním systémem umožňující vytvoření rezervace. Mimo vytvoření rezervace bude umožněno také její upravení či smazání.

**Metodika**

V teoretické části bude část o jazyku PHP a část o použitých značkovacích jazycích HTML, CSS, JavaScript.

V praktické části bude provedena analýza již existujících systémů umožňující online rezervaci a nalezeny požadavky na systém. Dále bude navržení základního designu a navržení funkčnosti systému pomocí UML diagramů. Systém bude vytvořen za pomoci programovacích jazyků HTML, CSS, PHP a JavaScript.

**Doporučený rozsah práce**

30-40 stran

**Klíčová slova**

Kadeřnický informační systém, Rezervační systém, PHP, HTML

---

**Doporučené zdroje informací**

SCHAFER, S M. *HTML, XHTML a CSS : bible [pro tvorbu WWW stránek] : 4. vydání.* Praha: Grada, 2009. ISBN 978-80-247-2850-6.

SCHMULLER, J. *Myslíme v jazyku UML : knihovna programátora.* Praha: Grada, 2001. ISBN 80-247-0029-8.

SUEHRING, S. *JavaScript : krok za krokem.* Brno: Computer Press, 2008. ISBN 978-80-251-2241-9.

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

---

**Předběžný termín obhajoby**

2022/23 LS – PEF

**Vedoucí práce**

Ing. Marek Pícka, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 24. 11. 2022

**doc. Ing. Tomáš Šubrt, Ph.D.**

Děkan

V Praze dne 15. 03. 2023

**Čestné prohlášení**

Prohlašuji, že svou bakalářskou práci "Kadeřnický informační systém na webu s rezervačním systémem" jsem vypracoval 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 15.03.2023 _____

**Poděkování**

Rád bych touto cestou poděkoval vedoucímu této bakalářské práce Ing. Markovi Píckovy, Ph.D. za jeho vynaložený čas pro konzultování problémů ohledně bakalářské práce a za poskytnuté rady.

# Kadeřnický informační systém na webu s rezervačním systémem

**Abstrakt**

Tato bakalářská práce se zabývá návrhem a realizací rezervačního systému na webu pro kadeřnický salon, který umožní zákazníkům salonu jednoduchou a pohodlnou možnost vytvoření rezervace. Systém bude vytvořen na základě stanovených požadavků od klienta. Výsledkem této práce by měl být rezervační systém, který bude připraven na nasazení do provozu. Salon poskytuje kadeřnické služby, kosmetické služby, manikúru a pedikúru. Rezervační systém byl vytvořen pomocí jazyků HTML, CSS, PHP a JavaScript. Teoretická část se zaměřuje na použité technologie a praktická část se zaměřuje na stanovení požadavků na systém, analýzu existujících rezervačních systémů, návrh pomocí UML diagramů a ER diagramu, vytvoření wireframů a mockupů a implementaci systému.

**Klíčová slova:**

kadeřnický informační systém, rezervační systém, HTML, CSS, Tailwind, PHP, JavaScript

# Hairdressing information system on the website with a reservation system

**Abstract**

This bachelor's thesis deals with the design and implementation of an information system on the website for hair salons and the creation of a reservation system that will allow salon customers a simple and convenient option for booking. The system will be created based on the established requirements of the client. The result of this work should be a reservation system that will be ready for deployment. The salon provides hairdressing services, beauty services, manicures, and pedicures. The reservation system was created using HTML, CSS, PHP and JavaScript. The theoretical part focuses on the used technologies, and the practical part focuses on the determination of system requirements, analysis of existing reservation systems, design using UML diagrams and ER diagrams, creation of wireframes and mock-ups, and system implementation.

**Keywords**:

hairdressing information system, reservation system, HTML, CSS, Tailwind, PHP, JavaScript

# Obsah

# 1 Úvod

V dnešní době je připojení k internetu takřka povinnost a internet se stal nedílnou součástí lidského života. Ať už kvůli zábavě a trávení volného času, nebo z praktických důvodů. A stejně jako internet se stal nedílnou součástí života, tak i informační systémy. Informační systémy jsou totiž všude kolem nás.

Jedním z druhů informačního systému je rezervační systém, kde jsou data tvořena rezervacemi, s kterými následně systém pracuje. Rezervační systémy zlepšují kvalitu života díky úspoře času a nenutnosti osobní komunikace. Existuje několik profesionálních rezervačních systémů, avšak někdy je lepší vytvořit vlastní systém přímo na míru danému klientovy, aby obsahoval přesně ty funkce, které klient požaduje.

Důvodem tvorby tohoto rezervačního systému je nabídka vytvořit pro kadeřnický salon „Kadeřnictví Hrdlořezy“ (pro který byly vytvořeny také celé webové stránky) rezervační systém, který je jednoduchý, funkční a efektivní. I přesto, že byly vytvořeny celé webové stránky, tato práce se zabývá pouze samotným rezervačním systémem.

Práce obsahuje proces tvorby rezervačního systému od analýzy až po výslednou implementaci. První kapitola je zaměřena na teorii ohledně použitých technologií k tvorbě systému, převážně na základní popis a syntaxi jednotlivých jazyků a popis CSS frameworků. Následně práce obsahuje část zaměřenou na stanovení požadavků pro systém, analýzu existujících řešení, návrh pomocí UML diagramů, konkrétně za pomoci Use Case diagramu, diagramu tříd a stavového diagramu, vytvoření ER diagramu a návrh UI systému pomocí wireframů a mockupů. Poslední část je věnovaná implementaci systému a zhodnocení vytvořeného systému. Tato část je zaměřená především na výběr HTML editoru a ukázku vytvořených funkcí a některých částí kódu, které jsou pro systém stěžejní. Zhodnocení systému obsahuje osobní pohled na výsledný systém a navržení několika funkcionalit, který by systém mohl obsahovat a být tak více univerzální.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Hlavním cílem práce je vytvoření rezervačního systému pro existující kadeřnický salon Kadeřnictví Hrdlořezy. Nabídka na vytvoření samotného systému přišla přímo od samotného salonu, který požaduje vytvoření jednoduchého a efektivního systémů, který by umožňoval pohodlně vytvořit rezervaci ze strany zákazníka a ze strany zaměstnance rezervaci upravit.

Mezi další cíle se řádí provedení analýzy existujících řešení, a srovnání jejich funkcionalit na základě požadavků na systém od klienta. Navržení systému pomocí UML diagramů, vytvoření wireframů a mockupů pro navržení UI systému a následně samotná implementace systému a nahrání na testovací doménu. Dalším cílem je zhodnocení vytvořeného systému a návrh na přidání některých funkcionalit, po kterých by byl rezervační systém více univerzální

Výsledkem práce bude plně funkční rezervační systém vytvořený na míru klientovy, který bude připravený na nasazení do ostrého provozu. Vytvořený systém bude klientem zkontrolovaný a uznaný za vyhovující.

## 2.2 Metodika

Požadavky na systém byly stanoveny při setkání s klientem, který je vlastníkem salonu Kadeřnictví Hrdlořezy. Na základě setkání budou všechny požadavky od klienta detailně sepsány jako požadavky na systém. Požadavky na systém budou rozděleny podle typu uživatele pro zákazníky, zaměstnance a administrátora systému. Na základě stanovených požadavků na systém bude provedena analýza existující řešení a porovnání funkcionalit existujících rezervačních systému se zadanými požadavky na systém. Po analýze existujících řešení budou vytvořeny diagramy pro systém, konkrétně UML diagramy Use Case diagram, diagram tříd a stavový diagram a ER diagram. Dále bude navrženo UI systému pomocí wireframů a mockupů, které budou vytvořené jak pro počítače, tak pro telefony.

V kapitole implementace bude popsán samotný kód. Bude definován zvolený HTML editor, popsána struktura kódu a propojení s databází, příklady vytvořených funkcí a tvorby časových slotů pro rezervace. Následně bude popsáno nahraní systému na doménu, výpis jednotlivých uživatelských účtů pro přihlášení do administrativní části systému a příklad

výsledné podoby systému. Závěrečná část práce bude věnována zhodnocení výsledné práce, která bude obsahovat můj jakožto autorův názor na vytvořený systém.

# 3 Použité technologie a metody

Pro vytvoření rezervačního systém jsem zvolil následující technologie:

## 3.1 HTML (hypertextový značkovací jazyk)

HTML se řadí mezi značkovací jazyky, což jsou jazyky, které obohacují základní text o nové informace pomocí instrukcí, které se píšou do speciálních znaků. Umožňují jednoduché instrukce jako podtržení či zvýraznění textu až po složitější instrukce, které umožňují vytvořit celou strukturu dokumentu. Výstupem značkovací jazyků jsou textové soubory a díky tomu jsou značkovací jazyky velmi lehce upravitelné i pomocí základních textových editorů. Mezi další značkovací jazyky patří například XHTML nebo XML.

HTML soubor se vytváří s příponou *.html* a je tvořen tzv. elementy. Element je složený z otevíracího tagu, textu a uzavíracího tagu. Otevírací tag může dále obsahovat hodnotu, která dále upřesňuje chování daného tagu a tato hodnota se nazývá atribut. Nejzákladnější atributy jsou atribut *id*, kterým lze elementu přiřadit unikátní název a atribut *class*, kterým lze přířadit stejné označení více elementům. Tyto atributy jsou využívané při stylování dokumentu (viz podkapitola 3.2 CSS (kaskádovité styly)). (M. Schafer, 2009)

### 3.1.1 Základní HTML tagy

Základní HTML tagy jsou: (M. Schafer, 2009)

- ** ** - Tag využívaný kodérem jako nástroj, který umožňuje vytvořit blok obsahující několik elementů. Jedná se o element typu *block* (viz podkapitola 3.2.4 Základní CSS vlastnosti*)*.
- **** - Obdoba tagu <*div*>, avšak tento element je typu *inline* (viz podkapitola 3.2.4 Základní CSS vlastnosti).
- **<h1, h2, h3, h4, h5, h6>** - Označení pro nadpis. Tag <*h1*> označuje hlavní nadpis a na stránce by se měl vyskytovat pouze jednou. Vyšší číslo v tomto tagu představuje méně významný nadpis.
- **<p>** - Odstavec pro text.
- **<br>** - Vytvoření nového řádku
- **<table>** -Vytvoření tabulky. Uvnitř elementu je zapotřebí použít element <*tr*> označující řádek a element <*td*> označující buňku tabulky (popřípadě <*th*> označující název sloupce).

- **<**table>

```
<tr>
    <th>název_sloupce_1 </th>
    <th>název_sloupce_2</th>
</tr>
<tr>
    <td>hodnota_sloupce_1</td>
    <td>hodnota_sloupce_2</td>
</tr>
<table>
```

- **<form> -** Tag označující formulář. V tagu je potřeba atribut *action*, který specifikuje, kam se posílají data při odeslání a atribut *method*, který specifikuje, jak se formulář chová. Metoda muže být *post* nebo *get* a jak názvy napovídají, *post* metoda data odesílá, zatímco *get* metoda data získává.
  - **<input> -** Tag označující prvek formuláře. Dále se dělí podle hodnoty atributu *type* například na textové pole (*text)*, odesílací tlačítko (*submit*), označovací pole (*radio, checkbox).* Existuje celkem 22 hodnot pro atribut *type.* Dále lze nastavit atributy *id*, *name* a *value.*
  - **<label> -** Označuje popisek pro daný <*input*>. Pomocí atributu *for* se určuje, jakému tagu <*input*> se na základě jeho hodnoty atributu *id* daný <*label*> přiřazuje.
  - **<select> -** Pole pro výběr z přednastavených hodnot. Hodnoty se zapisují pomocí tagu <*option*>
  - Příklad formuláře:

```
<form>
    <label for="input_id"></label>
    <input id="input_id" type="text"></input>
    <label for="select_id"></label>
    <select id="select_id">
        <option>možnost_1</option>
        <option>možnost_2</option>
    </select>
</form>
```

- **<a>** Tag pro vytvoření odkazu. Pomocí atributu *href* se nastavuje cesta, kam daný odkaz odkazuje. Lze odkazovat na jiný HTML soubor v adresáři, nebo na jakoukoliv webovou stránku. Atribut *target* specifikuje, jak se daný odkaz zobrazí.
    - Příklad pro odkaz vedoucí na webovou stránku a otevírající se v novém okně:
      `<a href="cesta_k_souboru" target="_blank">text</a>`

### 3.1.2 HTML 5 tagy

Od roku 2014 je používána verze HTML 5, která přinesla například podporu přehrávání multimédií v prohlížeči, jednoduší a rychlejší zápis značek, možnost vytvořit aplikaci, která v prohlížeči funguje i bez připojení k internetu a přidala několik nových tagů, které specifikují části stránky v rámci přístupnosti. (HTML5 - Overview, 2023)

Před příchodem HTML 5, bylo zvykem celý dokument konstruovat do elementů <*div*>. Tento element slouží pro kodéra stránky a umožnuje uzavírat elementy do samostatných menších bloků, díky čemuž je jednoduší stylování stránky. *av*šak jak již bylo zmíněno víše, HTML 5 přineslo několik nových tagů umožňujících lepší přístupnost. Tyto tagy více popisují strukturu celého dokumentu a v dnešní době je žádoucí dané tagy využívat. (HTML5 - Syntax, 2023)

- **<header>** - Záhlaví stránky (hlavička). Obsahuje převážně hlavní nadpis, popřípadě hlavní menu.
- **<footer>** - Zápatí stránky (patička). Obsahuje převážně souhrn informací, odkaz na kontakty, adresu, popřípadě pracovní nabídky.
- **<nav>** - Označuje menu.
- **<article>** - Označení pro blok informací, které spolu souvisí. Informace vepsané v tomto elementu by měly tvořit ucelených celek.
- **<section>** - Využívá se pro další rozdělení do bloků v rámci elementu <*article*>.
- **<aside> -** Označení bloku, který nesouvisí s hlavním obsahem stránky, ale poskytuje určitou informaci navíc

### 3.1.3 Struktura HTML

Na začátku HTML dokumentu musíme předat informaci webovému prohlížeči, že dokument, který je právě zobrazován je HTML. Tato informace se předá pomocí zápisu <*!DOCTYPE html*>. I přesto, že zápis vypadá jako zápis elementu, je důležité si uvědomit, že

se o HTML element nejedná. Základní struktura HTML začíná elementem <*html*> a dále se dělí na elementy <*head*> a <*body*>. (M. Schafer, 2009)

Základní struktura elementu <*html*> vypadá následovně:

```
<html>
    <head> … </head>
    <body> … </body>
</html>
```

#### 3.1.3.1 Element <head>

V elementu <*head*> se uchovávají data o HTML dokumentu (tzv. metadata). Informace zapsané v tomto elementu se na stránce nezobrazují (kromě elementu *title*) a využívá se převážně pro definování titulku stránky, použité znakové sady a stylování stránky. Pro nastavení titulku stránky se využívá element <*title*>, pro definování znakové sady element <*meta*> s atributem *charset="znaková_sada"* a pro následně je možné použití kaskádových stylů interním zápisem pomocí elementu <*style*> (viz podkapitola 3.2.2 CSS interně). (M. Schafer, 2009)

Základní struktura elementu <*head*> vypadá následovně:

```
<head>
    <meta charset="UTF-8">
    <title>Titulek stránky</title>
    <style> … </style>
</head>
```

#### 3.1.3.2 Element <body>

Element <body> následně obsahuje již samotný kód, který webový prohlížeč zobrazí. Tedy vše, co v tomto elementu bude napsáno, bude také zobrazeno prohlížečem přímo na webové stránce. (M. Schafer, 2009)

Základní struktura elementu <*body*> může vypadat například následovně:

```
<body>
    <header>
        <h1>Nadpis stránky<h1>
        <img src="cesta_k_souboru" alt="popis">
    </header >
```

```
        <article>
            <h2>nadpis_článku</h2>
            <section>
                <p>text_odstavce_2</p>
            </section>
        </article>
        <footer>
            <a href="cesta_k_souboru">Odkaz</a>
        </footer>
    </body>
```

## 3.2 CSS (kaskádovité styly)

Po HTML se jedná o jednu z nejzákladnějších částí webu. Na rozdíl od HTML, který upravuje strukturu webu, CSS upravuje grafickou stránku webu. CSS funguje na principu určování vlastností a hodnot daných vlastností jednotlivým selektorům, popřípadě zvoleným třídám či id. CSS může být zapisováno externě, interně, nebo jako vložené CSS. (Terberová, 2023)

### 3.2.1 CSS externě

CSS soubor je uložen zvlášť jako *nazev_souboru.css* a následně v HTML soboru je na daný CSS soubor odkazováno v hlavičce pomocí vlastnosti: *<link rel='stylesheet' href="odkaz_na_css_soubor">*. Tento způsob zápisu je v praxi nejvíce využívaný z důvodu přehlednosti a možnosti používat stylování na více HTML souborech a následně jednoduché úpravě stylů. (Terberová, 2023)

### 3.2.2 CSS interně

Stylování je zapisováno v hlavičce HTML souboru do sekce *<style>*. Nevýhoda tohoto stylu zápisu spočívá v tom, že pokud je potřeba změnit styl elementu, je potřeba stylování upravit v každém souboru zvlášť. (Terberová, 2023)

### 3.2.3 Vložené CSS

Stylování používané pouze pro jeden daný element. Stylování je zapisované jako atribut elementu přímo v HTML kódu. (Terberová, 2023)

Příklad pro nastavení červené barvy hlavnímu nadpisu*:*

```
<h1 style="color: red;"> Nadpis </h1>
```

### 3.2.4 Základní CSS vlastnosti

Základní CSS vlastnosti jsou: (CSS Tutorial, c1999-2023)

- **color** – Barva textu. Dnešní prohlížeče rozeznávají až 140 typů barev, pomocí použití klíčového slova pro danou barvu.
    - Příklad:
      ```
      color: indigo;
      ```
- **font-family** – Změna fontu textu. Udává se několik druhů fontů pro případ, pokud by daný font nebyl daným zařízením podporován. První font v pořadí má nejvyšší prioritu.
    - Příklad:
      ```
      font-family: arial, sans-serif, serif ;
      ```
- **font-size** – Velikost textu. Může být udávaná v absolutních nebo relativních jednotkách.
    - Příklad:
      ```
      font-size: 1.4rem;
      ```
- **display** – Vlastnost display je nejdůležitější vlastností CSS pro nastavení rozložení elementů na stránce. Každý HTML element má předem nastavenou hodnotu této vlastnosti podle toho, o jaký typ elementu se jedná. Většina elementů má hodnotu vlastnosti *block* nebo *inline*. Pokud je element typu *block,* element vždy začíná na novém řádku a šířku má nastavenou na sto procent. Elementy typu *inline* jsou přesným opakem, tedy nezačínají na novém řádku a šířku mají nastavenou na co nejmenší je potřeba, podle velikosti obsahu v elementu. Mimo tyto hodnoty lze také nastavit například hodnotu *none,* která element skryje z celého rozvržení stránky.
    - Příklad:
      ```
      display: block;
      ```
- **position –** Stanovuje způsob, jak se element na stránce pohybuje a zobrazuje. Dva základní typy pozicování jsou *relative* a *absolute*. Relativní pozicování element posouvá vůči jeho pozici na stránce, zatímco absolutní pozicování element posouvá vůči okrajům

celé stránky. Pokud je však element s absolutním pozicováním umístěn v elementu s relativním pozicováním, posouvá se vůči tomuto elementu.

  - Příklad:

    ```
    position: absolute;
    ```

- **border –** Ohraničení elementu. Ohraničení lze nastavit tloušťku, styl a barvu do jedné hodnoty vlastnosti anebo lze všechny tyto hodnoty nastavit zvlášť pomocí vlastností *border-width*, *border-style* a *border-color*.
  - Příklad:

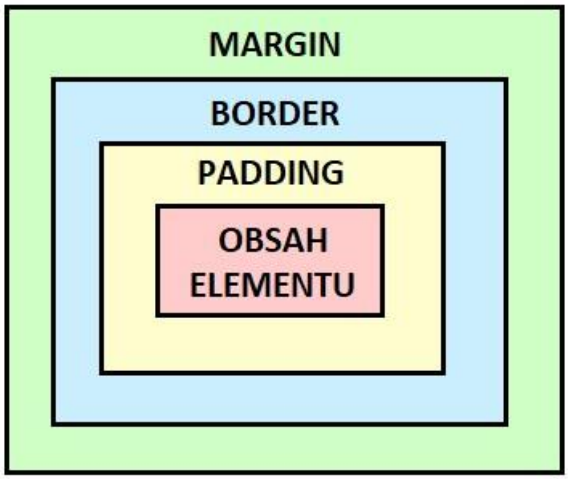
    ```
    border: 3px solid blue;
    border-width: 3px
    border-style: solid;
    border-color: blue;
    ```

- **margin** – Nastavení vnějšího odsazení. Lze specifikovat pouze pro jednu stranu nebo nastavení stejného odsazení na všechny strany nebo nastavení pro jednotlivé strany do jednoho zápisu pomocí vypsání čtyř hodnot vlastností, přičemž první hodnota přestavuje horní odsazení, druhá pravé, třetí spodní a čtvrtá levé.
  - Příklad:

    ```
    margin: 1rem;
    margin-left: 2rem;
    margin: 1rem 1rem 1rem 2rem;
    ```

- **padding –** Nastavení vnitřního odsazení v elementu. Možnosti zápisu jsou totožné jako v případě vlastnosti margin.
  - Příklad:

    ```
    padding: 1rem;
    padding-left: 2rem;
    padding: 1rem 1rem 1rem 2rem;
    ```




*Obrázek 1 - CSS padding, margin, border (zdroj: vlastní)*

### 3.2.5 Media Queries

V dnešní době, kdy má každý mobilní telefon, je potřeba mít stránky správně nastylované jak na PC, tak na mobilní telefony, popřípadě tablety. A právě o to se stará CSS modul Media Queries. Media Queries se nastavují v CSS souboru následovně: *@media only screen and (min-width: 1200px) { ... }*. Stylování napsané v takto nastaveném Media Queries bude aktivní pouze u obrazovek s minimální šířkou 1200 pixelů. (CSS Tutorial, c1999-2023)

### 3.2.6 CSS frameworky

Frameworky si lze představit jako knihovnu přednastavených stylů pro určité třídy, které mohou obsahovat určité části JavaScriptu a které zjednodušují a zrychlují práci při stylování elementů. Existuje mnoho frameworku, které se od sebe rozlišují hlavně formou syntaxe. Přes rozdíly v syntaxi je však jejich cíl stejný. V dnešní době téměř většina frameworků zjednodušuje práci s Media Queries a díky tomu lze efektivně přizpůsobovat stránku danému zařízení podle velikosti obrazovky. Mezi nejznámější CSS frameworky se řadí Bootstrap, Foundation či Tailwind CSS, přičemž právě Tailwind CSS je používaný v tomto projektu. (Guide to CSS Framework, 2022)

#### 3.2.6.1 Bootstrap

Bootstrap byl vydán v roce 2011 a jedná se o jeden z vůbec prvních vytvořených CSS frameworků. Důvodem jeho vzniku bylo standardizovat používání daných stylů ve společnosti Twitter. Bootstrap se využívá pro kódování typu „mobile-first“, kdy se stránka tvoří prvně pro mobilní zařízení a od něho se následně upravuje design pro větší obrazovky. Mimo CSS

knihovnu obsahuje také rozsáhlou JavaScript knihovnu obsahující nejrůznější pluginy. Tento framework používají společnosti jako Spotify, Udemy nebo právě zmiňovaný Twitter. (Spurlock, 2013)

Základním stavebním kamenem Bootstrapu je práce v takzvaných „gridech“. Grid rozděluje část webové stránky do mřížky která obsahuje 12 sloupců, které jsou v základu 60 pixelů široké a mají nastavenou vzdálenost od ostatního elementu z leva 20 pixelů.

Pro vytvoření responzivního designu má Bootstrap definované velikosti, které lze použít při zápisu dané třídy. Lze tak nastavit počet buněk v řádku rozdílně pro různé velikosti obrazovky pomocí třídy *col-xs-12 col-ms-10 col-md-8 col-lg-6*. Část *col* nastavuje roztažení buňky na danou hodnotu, která je uvedená na konci zápisu, tedy že daná buňka bude široká jako daný počet buněk a hodnoty *xs, ms, md, lg* přestavují právě dané velikosti obrazovek. (Bootstrap Grid System, c1999-2023)

| **Typ obrazovky** | **Velikost obrazovky** | **Prefix** |
|---|---|---|
| Telefony | < 798px | -xs- |
| Tablety | >= 768px | -sm- |
| Malé monitory | >= 992px | -md- |
| Velké monitory | >=1200px | -lg- |

*Tabulka 1 - Bootstrap velikosti obrazovek (zdroj: (Bootstrap Grid System, c1999-2023))*

Nevýhodou daného frameworku je však daná šablonovitost. Díky tomu, že Bootstrap používá již nastylované třídy, stránky, které využívají Bootstrap mnohdy vypadají podobně. Například vytvořené menu nebo tlačítka. Pokud je tlačítku nastavena hodnota atributu *class=“btn“*, tlačítko má již přednastavené zaokrouhlení rohů, přednastavený padding a velikost textu. Pokud třídu pro tlačítko nastavíme jako *class=“btn btn-large“*, tlačítko bude mít větší text a padding. Bootstrap má také předem nastavené barvené varianty, které lze použít u různých elementů. Barvy jsou pojmenované podle typu použití. Například *success* (zelená), *warning* (oranžová), *danger* (červená), *default* (šedá), *primary* (modrá). Barvu lze nastavit tlačítku pomocí hodnoty atributu *class=“btn btn-primary*“. Právě z důvodu jisté podobnosti stránek využívající Bootstrap stále více a více kodérů přechází na framework Tailwind CSS. (Spurlock, 2013)

#### 3.2.6.2 Tailwind CSS

Tailwind vznikl v roce 2017 a je využíván například společností Netflix. Ve své podstatě je podobný frameworku Bootstrap. Největší rozdíl však spočívá v tom, že Tailwind CSS neobsahuje přednastavené komponenty a zaměřuje se především na rychlejší stylování, díky kterému je možné element upravovat pomocí hodnoty atributu *class*. Díky tomu Tailwind CSS umožňuje nastavit vlastní design danému elementu.

Na rozdíl od frameworku Bootstrap, kde je možné použít přednastavené třídy k vytvoření předem definovaného komponentu, v Tailwind CSS žádnou takovou třídu nenajdeme a je potřeba vytvořit celý komponent od základu. Tailwind CSS obsahuje pouze jakési zkratky klasických CSS vlastností. (Murphy, 2022)

Například pokud bychom chtěli elementu nastavit vnitřní odsazení, stačí pouze přidat třídu *p-2*, kde *p* přestavuje padding a číslo za ním velikost odsazení.

Lze také definovat odsazení pouze pro dané strany.

- *pr-2* (odsazení napravo)
- *pl-2* (odsazení nalevo)
- *pt-2* (odsazení nahoře)
- *pb-2* (odsazení dole)
- *px-2* (odsazení na ose x)
- *py-2* (odsazení na ose y)

Tailwind CSS má přednastavené velikosti pro různé obrazovky stejně jako Bootstrap, avšak zápis třídy je zde mnohem přehlednější a lze jej použít na jakýkoliv typ vlastnosti. Lze nastavit vnější odsazení elementu rozdílně pro různé obrazovky, přímo v zápisu třídy následovně: *sm:m-2 md:m-4 lg:m-6 xl:m-8 2xl:m-10*. (Responsive Design - Tailwind CSS, 2023)

| **Typ obrazovky** | **Velikost obrazovky** | **Prefix** |
|---|---|---|
| Malé tablety | >= 640px | sm: |
| Velké tablety | >= 768px | md: |
| Malé monitory | >= 1024px | lg: |
| Středí monitory | >= 1280px | xl: |

| | | |
|---|---|---|
| Velké monitory | >= 1536px | 2xl: |

*Tabulka 2 - Tailwind CSS velikosti obrazovek (zdroj: (Responsive Design - Tailwind CSS, 2023))*

Tailwind CSS funguje stejně jako Bootstrap na principu „mobile-first“. Tím pádem, pokud při zápis třídy není specifikovaná velikost obrazovky, je daný efekt použitý na všechny obrazovky a pro rozdílné zobrazení na větších obrazovkách je zapotřebí použít potřebný prefix. (Responsive Design - Tailwind CSS, 2023)

## 3.3 JavaScript

V dnešní době je JavaScript využíván téměř na každé webové stránce. Jedná se o skriptovací jazyk, který umožňuje vytvářet stránky, které vykazují určité chování v průběhu času, tedy tzv. dynamické weby. Spolu s HTML a CSS se jedná o základní trojici jazyků tvořící drtivou většinu dnešních stránek. JavaScript přidává do webových stránek možnost vytvářet animace a tím pádem zpříjemnit uživatelské rozhraní a zlepšit UX. I přesto, že dnes drtivá většina prohlížečů podporuje JavaScirpt, je stále potřeba psát HTML soubor tak, aby mohl fungovat i bez použitých skriptů. (Koďousková, 2022)

JavaScript lze označit jako samostatný malý program, který běží na webové stránce. Tento program se označuje výrazem skript a zapisuje se přímo do HTML souboru pomocí tagu *<script>*. Mimo zápis přímo v HTML lze na soubor pouze odkazovat a skript psát v separátním souboru označeným příponou *.js*. JS soubor se následně načte pomocí atributu *src* v tagu *script*, který se z pravidla zapisuje jako poslední element v elementu *body*.

Zápis pro načtení skriptu do HTML souboru ze separátního JS souboru:

```
<script src="cesta_k_JS_souboru"></script>
```

I pro tento jazyk existuje mnoho frameworků, které zrychlují a zjednodušují zápis příkazů pomocí zápisu skriptů přímo do HTML souborů. Nejpoužívanější JS frameworky jsou React, Vue a Angular, avšak pro tento projekt byl zvolen pouze čistý JavaScript. (Kaur Arora, 2022)

## 3.4 PHP

PHP se řadí se mezi programovací jazyky, které fungují na straně serveru. Na rozdíl od HTML stránek jsou PHP stránky dynamicky generované, tedy umožňují měnit obsah stránky v průběhu času, bez nutnosti znovunačtení stránky. PHP se používá především pro práci

s externími daty v databázi. První verze PHP byla vytvořena v roce 1995 Rasmusem Lerdorfem a od té doby se PHP stále vyvíjí a vycházejí stále nové verze.

Pro vytvoření PHP souboru se využívá přípona *.php* a celý PHP kód je uzavřen pomocí otevíracího tagu *<?php>* a uzavíracího tagu *<?>*. PHP se zpravidla píše na začátek souboru a následně soubor obsahuje HTML část. Lze však použít PHP i přímo v HTML části souboru. (Nixon, 2021)

### 3.4.1 PHP operátory

Výpis PHP operátoru, rozdělených podle typu operací. (PHP Operators, c1999-2023)

- **Aritmetické –** Operátory představující matematické operace

| **Operace** | **Značka** |
|---|---|
| Sčítání | + |
| Odčítání | - |
| Násobení | * |
| Dělení | / |
| Modulo (zbytek po dělení) | % |
| Přičtení hodnoty 1 | ++ |
| Odečtení hodnoty 1 | -- |

*Tabulka 3 - PHP aritmetické operátory (zdroj: (PHP Operators, c1999-2023))*

- **Logické –** Operátory používané pro logické provázání několika operací.

| **Operace** | **Značka** |
|---|---|
| Logický součin (Je pravda pokud platí všechny podmínky) | &&, AND |
| Logický součet (Je pravda pokud platí alespoň jedna podmínka) | \|\|, OR |
| Logické negace (Je pravda pokud podmínka neplatí) | ! |

*Tabulka 4 - PHP logické operátory (zdroj: (PHP Operators, c1999-2023))*

- **Relační –** Operátory používané při vytváření podmínek

| Operace | Značka |
|---|---|
| Menší než | < |
| Větší než | > |
| Menší nebo rovno | <= |
| Větší nebo rovno | >= |
| Je rovno | == |
| Není rovno | != |

*Tabulka 5 - PHP relační operátory (zdroj: (PHP Operators, c1999-2023))*

### 3.4.2 Základní PHP syntaxe a funkce

PHP jazyk nerozlišuje velká a mála písmena v běžných příkazech, avšak v rámci názvů proměnných a funkcí je potřeba velká a malá písmena dodržovat. Dalším pravidlem, které se musí dodržovat je používání středníku na konci každého příkazu. Pro zápis komentáře do kódu lze požadovanou část textu uzavřít do sekvence /* ... */ nebo v případě komentáře pro celý řádek lze využít //. (Nixon, 2021)

Příklad základních PHP příkazů: (Nixon, 2021)

- **$** - Vytvoření proměnné.
    - Příklad vytvoření proměnné:

```
$promenna = 0;
```

- **echo** – Výpis hodnoty na stránku.
    - Příklad vypsání textu „Ahoj světe!“:

```
echo „Ahoj světe!“;
```

    - Příklad vypsání proměnné *$promenna*:

```
echo $promenna;
```

- **if** – Vytvoření podmínky (větvení). Podmínky jsou jedním ze základních pilířů téměř všech programovacích jazyků. Umožňují rozdělit část programu podle toho, zda je splněna určitá podmínka či nikoliv. V kulatých závorkách se zapisuje samotná podmínka a ve složených závorkách příkaz který se vykoná, pokud daná podmínka platí. Po použití příkazu *if* lze použít příkaz *else*, který stanovuje, co se vykoná, pokud daná podmínka neplatí. Pokud je požadováno více podmínek, je možné použít příkaz *elseif*, v kterém lze specifikovat další podmínky. Příkaz *elseif* je možné použít pouze po příkazu *if*.

- **if** – Pokud je proměnná rovna 1, nastavit proměnnou na hodnotu 2. Příklad zápisu:

```
If ($promenna == 1)
     {$promenna = 2;}
```

- **if / else** –Pokud je proměnná rovna 1, nastavit proměnnou na hodnotu 2, pokud ne, nastavit na hodnotu 3. Příklad zápisu:

```
if ($promenna == 1)
     {$promenna = 2;}
else
     {$promenna = 3;}
```

- **if / elseif / else** – Pokud je proměnná rovna 1, nastavit proměnnou na hodnotu 2. Pokud je proměnná rovna 2, nastavit proměnnou na hodnotu 1. Pokud není splněná žádná podmínka, nastavit na hodnotu 3. Příklad zápisu:

```
if ($promenna == 1)
     {$promenna = 2;}
elseif ($promenna == 2)
     {$promenna = 1;}
else
     {$promenna = 3;}
```

- **switch** – Jedná se o obdobu příkazu *elseif*. Požívá se, pokud je potřeba vyhodnotit větší množství podmínek. Na rozdíl od *elseif* je příkaz *switch* přehlednější, avšak funkčně jsou příkazy totožné. Do kulaté závorky se nezapisuje podmínka, ale pouze proměnná, která se bude vyhodnocovat. Následně ve složených závorkách se vypisují příkazy *case,* které určují hodnotu, která se porovnává s určenou proměnnou a příkaz, který se vykoná, bude-li splněna podmínka. Pokud je hodnota rovna proměnné, vykoná se daná větev. Příkaz *case* musí obsahovat příkaz *break,* který určuje konec dané větve.
  - Příklad zápisu (Totožný příklad jako pro užití *elseif*):

```
switch ($promenna)
{
     case 2:
          $promenna = 2;
          break;
```

```
        case 20:
            $promenna = 20;
            break;
        Default:
            $promenna = 100;
            break;
    }
```

- **while** – Vytvoření cyklu. který se využívá především pokud není předem určený počet opakování. Cykly v kódu tvoří část programu, která vykazuje totožné chování v rámci jedné podmínky. Je potřeba dbát na správné zapsání podmínky, aby nedošlo k nekonečnému zacyklení. Do kulatých závorek se zapisuje podmínka a následně ve složených závorkách je příkaz, který se v každém cyklu vykoná.
    - Pokud je proměnná menší než 5, vypíše se proměnná a přičte se k proměnné hodnota 1. Příklad zápisu:

```
$promenna = 1;
While ( $promenna < 5 ) {
    echo $promenna;
    $promenna++;
}
```

- **for –** Vytvoření cyklu, který se využívá, pokud je předem určený počet opakování. V kulatých závorkách se určuje proměnná, podmínka a operace která se vykoná na konci každého cyklu, rozdělené pomocí *;*.
    - Příklad zápisu (Totožný příklad jako pro užití *while*):

```
For($promenna=1; $promenna<5; $promenna++){
    echo $promenna;
}
```

- **function** – Funkce umožňuji vytvořit část programu, kterou lze v kódu využít vícekrát na několika různých místech, a tím pádem zamezit duplicitě kódu a vytvořit tak přehlednější kód. Název funkce nesmí obsahovat mezery a je pravidlem psát stylem tzv. „camelCase“, tedy začít název malým písmenem a následně každé nové slovo začít velkým písmenem. Funkce může být deklarována bez parametrů nebo s parametry. Volání funkce následně probíhá v kódu pouze pomocí vypsání názvu funkce, popřípadě přidáním parametrů pro funkci. (Nixon, 2021)

- **Funkce bez parametrů** – nejjednodušší typ funkce, která má stanovené pouze jedno chování, které se nemění. Příklad pro deklaraci funkce vypisující „Ahoj světe!“ a následné volání dané funkce:

```
function nazevFunkce() {echo "Ahoj světe!";}
nazevFunkce();
```

- **Funkce s parametry** – funkce, která pracuje s určenými parametry, a lze tedy ovlivnit její chování. Příklad pro deklaraci funkce vypisující součet svou čísel a následné volání dané funkce:

```
function nazevFunkce($x, $y) {echo ($x+$y);}
nazevFunkce(5, 3);
```

- **Funkce s návratovou hodnotou** – funkce, která při zavolání vrací hodnotu, kterou lze uložit do proměnné a dále s ní v kódu pracovat. Hodnota, kterou funkce vrací se nazývá návratová hodnota a zapisuje se pomocí příkazu *return*. Příklad pro deklaraci funkce součtu tvou čísel s návratovou hodnotou a následné volání funkce:

```
function nazevFunkce($x, $y) {return ($x+$y);}
nazevFunkce(5, 3);
```

## 3.5 UML

Pod zkratkou UML se skrývá název Unified Modeling Language neboli sjednocený modelovací jazyk. Jedná se o sadu standardizovaných postupů pro navržení softwarových systémů umožňující objektově orientovaný přístup. Cílem UML není popsat, jak přesně vytvořit daný systém, ale pouze popsat jakým stylem zapisovat tvorbu návrhu systému.

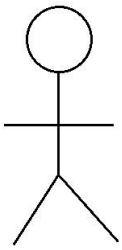
UML se skládá z několika diagramu jako například Use Case diagram, diagram aktivit, diagram tříd nebo diagram objektů. Hlavní výhodou používání UML je právě zmiňovaná standartizace. Pokud tedy ovládáte UML, jste schopní porozumět všem návrhům softwarových systémů, které byly navrženy pomocí UML. V dnešní době se UML používá téměř v každém návrhu systému. (Eriksson, 2004)

### 3.5.1 Use Case Diagram (Diagram případu užití)

Use Case diagram specifikuje, co přesně systém dělá v závislosti na uživatelských akcích. Je zapisován jednoduchými symboly, které představují samotné elementy daného diagramu. Skládá se z aktérů a případu užití. (Eriksson, 2004)

#### 3.5.1.1 Actors (aktéři)

Aktér posílá, získává nebo si vyměňuje informace se systémem. Může se jednat jak o živého člověka, tak o další systém, který s daným systémem komunikuje. Jednoduše by se dalo říct, že aktér je cokoliv, co interaguje s daným systémem. Aktér nepředstavuje skutečného uživatele ale pouze roli, kterou může zastávat více skutečných uživatelů. Například role admin, zaměstnanec nebo zákazník. Aktér se v Use Case diagramu značí symbolem jednoduché lidské postavy. (Eriksson, 2004)



*Obrázek 2 - Aktér v Use Case diagramu (zdroj: vlastní)*

#### 3.5.1.2 Use Cases (případy užití)

Případy užití představují samotné akce, které systém po vyžádání aktéra vykonává. Samotné případy užití musí být vždy spuštěné aktérem, musí aktérovy předávat určitou hodnotu a musí vždy představovat ucelenou akci s daným výstupem. Případy užití se v Use Case diagramu zapisují do elipsy a jsou spojené s aktéry pomocí jednoduché čáry. (Eriksson, 2004)

Případy užití mohou být mezi sebou spojené vazbami a vazby rozdělujeme následovně: (Eriksson, 2004)

- **základní** – Vyjadřují vztah, kdy tzv. dětský případ užití dědí z tzv. rodičovského případu užití veškeré jeho akce. Je označen plnou čárou s šipkou směřující směrem od dětského k rodičovskému případu užití
- **rozšiřující (extend) –** Vyjadřuje vztah, kdy případ užití přidává novou akci k jinému případu užití. Tento vztah je nepovinný. Je označen přerušovanou čárou s šipkou a s textem <<*extend*>> směřujícím směrem od rozšiřujícího k rozšiřovanému případu užití.
- **zahrnující (include) –** Vyjadřuje vztah, kdy případ užití odkazuje na akci jiného případu užití. Případ užití, na který je pomocí vazby include odkazováno je povinný, a proběhne při každém spuštění vyvolávacího případu užití. Je označen přerušovanou

čárou s šipkou a s textem <<*include*>> směřujícím směrem od původního k zahrnujícímu případ užití.

### 3.5.2 Stavový diagram

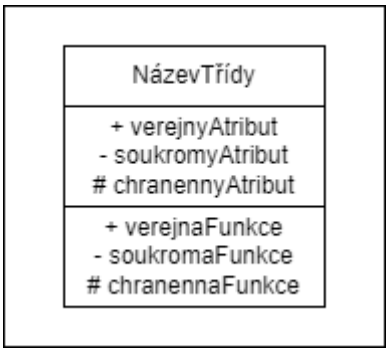
Stavový diagram specifikuje stavy jednotlivých objektů. Popisuje, jak se objekt do daného stavu dostal, jak se z něj dostane do jiného stavu a jak se objekt ve stavu chová neboli jaké aktivity v daném stavu vykonává. Jednotlivé stavy jsou zapisovány do obdélníku se zaoblenými rohy a události, které spouštějí stavy jsou označeny šipkou s názvem události. Každý stav může obsahovat několik aktivit, které se dělí podle času jejich vykonání na aktivity *entry* (vykonané při vstupu do stavu), *do* (vykonané během pobytu ve stavu) a *exit* (vykonané při výstupu ze stavu). Každý stavový diagram musí mít vyznačený začátek a konec. Začátek se značí plným kruhem a konec plným kruhem s obrysem. (Eriksson, 2004)

### 3.5.3 Diagram tříd

Diagram tříd vychází z ER diagramu, avšak na rozdíl od ER diagramu se zde entity nazývají třídy a je v něm navíc popsané chování jednotlivých tříd. Třídy se zapisují do obdélníku, který je rozdělen na 3 části. První část obsahuje název třídy, druhá část obsahuje jednotlivé atributy třídy a třetí část obsahuje operace (chování), které daná třída vykonává. (Eriksson, 2004)

#### 3.5.3.1 Viditelnost atributů a operací

U atributů a operací se definuje jejich viditelnost, která značí, zda jsou atributy dosažitelné mimo jejich vlastní třídu. Viditelnost se označuje znaménky +, - a #. Znaménko + označuje viditelnost jako veřejnou, tedy atribut je dostupný i mimo svoji danou třídu, známko – označuje viditelnost jako soukromou, tedy atribut je dostupný pouze ve své dané třídě a znaménko # označuje viditelnost jako chráněnou, tedy že atribut je dostupný i v podtřídách své dané třídy. (Eriksson, 2004)




*Obrázek 3 - Zápis třídy v diagramu třídy (zdroj: vlastní)*

# 4 Analýza

Po schůzce s klientem byly předány základní informace o salonu a stanoveny základní požadavky na systém. Jedná se o malý salon, který vykonává služby kadeřnictví, kosmetiky, manikúry a pedikúry. Kvůli kapacitním omezením salonu vždy každou službu vykonává pouze 1 zaměstnanec. V momentální situaci salonu není možné, aby pro 1 službu bylo více zaměstnanců. Zaměstnanci mají pevné pracovní doby od 08:00 do 16:00.

Mezi základní požadavky patří možnost vytvoření a upravení rezervací. Systém nemá obsahovat možnost vytvořit účet pro zákazníky a nemá je ani evidovat v databázi. Jediné účty, které systém bude evidovat budou účty zaměstnanců pro přihlášení do administrace. Zaměstnanci salonu budou po přihlášení schopni rezervace mazat a upravovat. Dalším požadavkem je odesílání informativních emailů zákazníkům při vytvoření nebo upravení rezervace a nastavení práv pro úpravu rezervace podle typu služby uživatele. Systém by měl dále obsahovat jeden účet, který může upravovat jakékoliv rezervace a také upravovat uživatelské účty.

Časové sloty pro rezervace mají být vytvářeny po 30 minutách a každá služba má svůj specifikovaný čas trvání. Velký důraz byl kladen na jednoduchost a efektivnost při vytváření a upravování rezervace, možnost příjemně obsluhovat systém z telefonu a českou lokalizaci.

## 4.1 Požadavky na systém

Základním krokem pro sestavení rezervačního systému bylo vymezení požadavků pro systém neboli stanovení funkcí, které od systému vyžadujeme. Požadavky se dělí do dvou základních skupin. Požadavky funkční a nefunkční. (Gorbachenko, 2023)

### 4.1.1 Funkční požadavky

Funkční požadavky vyjadřují, co musí systém umožnit vykonávat. Popisují, co musí systém obsahovat, pro uspokojení uživatelských potřeb (uživatelské požadavky). Funkční požadavky lze chápat jako vyjádření toho, co vše daný systém bude dělat. (Gorbachenko, 2023)

#### 4.1.1.1 Stanovení funkčních požadavků (zákazník)

- **Vybrání typu služby pro rezervaci –** Salon nabízí služby kadeřnictví, kosmetika, manikúra a pedikúra a z tohoto důvodu systém musí umožnit výběr, na kterou danou

službu a na jaký konkrétní typ práce (střih, barvení, melír, …) se chce zákazník zarezervovat.

- **Vybrání datumu rezervace –** Zobrazení kalendáře, který nabídne zvolení dne, na který si uživatel přeje zarezervovat termín. Vytvořit rezervace je nutné nejméně 1 den předem.
- **Vybrání času rezervace –** Podle dostupnosti časových slotů, systém zobrazí volné časy, na které se lze objednat. Pokud se na daný časový slot není možné objednat, systém daný časových slot nezobrazí. Pokud není volný žádný časový slot, systém uživatele upozorní, že není volný žádný časový slot. Časové sloty jsou vytvářeny po 30 minutách.
- **Zobrazení souhrnu rezervace –** Po vybrání časového slotu systém zobrazí souhrn rezervace, který bude obsahovat typ služby, název služby, datum a čas.
- **Zadání informací o zákazníkovy –** Přímo pod souhrnem rezervace systém zobrazí formulář pro zadání informací o zákazníkovy. Formulář bude obsahovat Jméno, email a telefon a všechny tyto informace musejí být povinné.
- **Potvrzení rezervace –** Po zobrazení souhrnu rezervace a zadání informací o zákazníkovy systém zobrazí tlačítko pro potvrzení rezervace. Po odeslání systém informuje zákazníka o úspěšném zarezervování prostřednictvím výpisu na dané stránce a odesláním emailu zákazníkovi.

#### 4.1.1.2 Stanovení funkčních požadavků (uživatel / zaměstnanec)

- **Zobrazení všech rezervací pro danou službu –** Systém zobrazí všechny rezervace podle typů jednotlivých služeb. Systém bude umožňovat třídění rezervací podle datumu nebo zadaného textu. Rezervace se musí automaticky řadit podle času.
- **Zobrazení dnešních rezervací –** Na samostatné stránce systém zobrazí všechny rezervace, které jsou zarezervované na dnešní datum, rozdělené podle typu služby a seřazené podle času
- **Upravení rezervace –** Při výpisu rezervace systém zobrazí tlačítko umožňující upravení rezervace. Po upravení rezervace se odešle informativní email zákazníkovi.
- **Smazání rezervace –** Stejně jako při zobrazení tlačítka pro upravení rezervace se zobrazí tlačítko pro zrušení rezervace. Po smazání rezervace se také odešle informativní email zákazníkovi.
- **Správa rezervací podle typu služby –** Pro upravení, čí smazání rezervace je potřeba, aby daný uživatel vykonával službu, pro kterou je daná rezervace zaevidována.

Zaměstnanec, který vykonává kadeřnické služby tak například nemůže upravit rezervaci vytvořenou pro kosmetické služby. Jedinou výjimku tvoří účet pro administrátora.

#### 4.1.1.3 Stanovení funkčních požadavků (admin)

- **Přidání a odebrání uživatelů –** Admin bude mít přístup ke všem funkcím jako uživatel, ale navíc mu bude umožněno přidání či odstranění uživatelů.
- **Správa rezervací nehledě na typ služby –** Uživatel s přístupem admin musí být schopný upravit jakékoliv rezervace, nehledě na typ služby.

### 4.1.2 Nefunkční požadavky

Nefunkční požadavky specifikují, jak daný systém bude fungovat. Zaměřují se na UX a efektivnost daného systému. Nesplnění požadavků neohrožuje chování systému, avšak má za následek znepříjemnění uživatelské přívětivosti. Zaměřují se na specifikování rychlosti, přístupnosti, kapacity, bezpečnosti a použitelnosti systému. (Gorbachenko, 2023)

#### 4.1.2.1 Stanovení nefunkčních požadavků

- Vytvoření rezervace by nemělo zabrat více než pár jednotek sekund
- Systém by měl být co nejjednodušší a nejpřehlednější
- Uživatelská hesla budou v systému zašifrována
- Časové sloty pro rezervaci budou tvořeny po 30 minutách
- Systém bude obsahovat českou lokalizace
- Design systému bude responzivní

## 4.2 Analýza existujících řešení

Následující část se zaměřuje na analýzu existujících rezervačních systému a jejich porovnání s danými požadavky na systém. Existuje nespočet rezervačních systému, které lze používat a drtivá většina těchto systému se rozděluje na bezplatnou a placenou verzi, která nabízí více funkcionalit. Jedním z požadavků na systém je nenutnost placení předplatného a z tohoto důvod se analýza zaměřuje pouze na bezplatné varianty systémů. Níže jsou vypsány některé rezervační systémy s jejich klady a zápory.

- **Reservatic** (Reservatic, c2015-2023)

| **Klady** | **Zápory** |
|---|---|
| Neomezený počet uživatelů | Nelze posílat notifikace uživatelům |
| Neomezený počet rezervací | Reklamy v UI |
| Aktualizace | Nelze nastavit vlastní vzhled |
| Česká lokalizace | |

*Tabulka 6 - Klady a zápory rezervačního systému Reservatic (zdroj: vlastní)*

- **Reservanto** *(Reservanto, 2023)*

| **Klady** | **Zápory** |
|---|---|
| SMS notifikace | Omezený počet uživatelů |
| Neomezený počet rezervací | Nelze upravovat rezervace na telefonu |
| Neomezený počet služeb | Nelze nastavit uživatelská oprávnění |
| Česká lokalizace | Nelze nastavit vlastní vzhled |

*Tabulka 7 - Klady a zápory rezervačního systému Reservanto (zdroj: vlastní)*

- **TeamUp** *(Teamup, 2023)*

| **Klady** | **Zápory** |
|---|---|
| Posílání notifikací uživatelům | Není česká lokalizace |
| Neomezený počet rezervací | Nelze nastavit uživatelská oprávnění |
| Mobilní aplikace | |

*Tabulka 8 - Klady a zápory rezervačního systému TeamUp (zdroj: vlastní)*

- **Reservio** *(Reservio, c2012–2023)*

| **Klady** | **Zápory** |
|---|---|
| Připomínky rezervací přes e-mail | Vytvoření pouze 40 rezervací měsíčně |
| Mobilní aplikace | Nelze nastavit vlastní vzhled |
| Česká lokalizace | Reklamy v UI |

*Tabulka 9 - Klady a zápory rezervačního systému Reservio (zdroj: vlastní)*

- **SimplyBook.me** *(SimplyBook.me, 2021)*

| **Klady** | **Zápory** |
|---|---|
| Mobilní aplikace | Vytvoření pouze 50 rezervací měsíčně |
| | Omezený počet uživatelů |
| | Není česká lokalizace |

*Tabulka 10 - Klady a zápory rezervačního systému SimplyBook.me (zdroj: vlastní)*

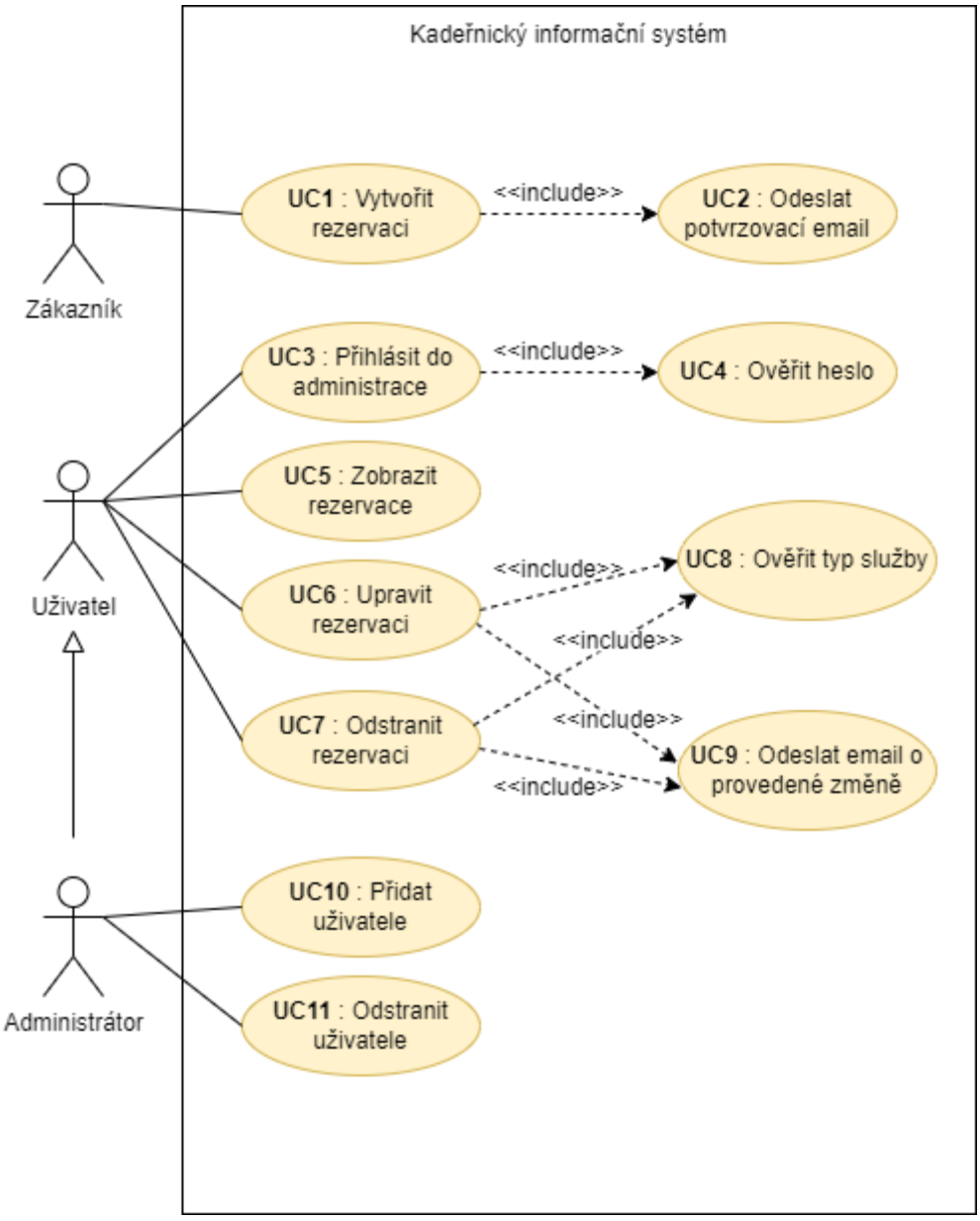
Klient pro svůj rezervační systém považuje jako klíčové vlastnosti českou lokalizaci, neomezený počet uživatelů, neomezený počet rezervací za měsíc, nastavení vlastního vzhledu systému a čistotu stránky bez jakýchkoliv reklam. Žádný z těchto systému neobsahuje všechny tyto funkcionality v bezplatné verzi a z tohoto důvodu není žádný z těchto systému vyhovující.

## 4.3 Vytvořený Use Case diagram

Pro tento projekt jsem použil aktéry *zákazník, uživatel* a *administrátor*. Aktér *zákazník* může pouze vytvářet nové rezervace a následně mu bude poslán potvrzovací email o vytvoření rezervace. Vazba mezi případy užití *vytvořit rezervaci* a *odeslat potvrzovací email* je typu *include*, jelikož se email odešle vždy po vytvoření rezervace.

Aktérovy *uživatel* je umožněno přihlásit se do administrativního systému, v kterém si může následně zobrazit rezervace nebo je upravit či smazat. Při úpravě a odstranění rezervace se odešle email informující o změnách. Tento případ je v diagramu zobrazen jako nový případ užití, který je jako v případě potvrzovacího emailu označen vazbou *include.*

Aktér *administrátor* dědí od aktéra *uživatel.* Na rozdíl od předchozího aktéra však může přidávat a mazat uživatele v systému.




*Obrázek 4 - Vytvořený Use Case diagram (zdroj: vlastní)*

### 4.3.1 Scénáře

- **UC1 : Vytvořit rezervaci**
  - Uživatel zvolí možnost vytvořit rezervaci.
  - Systém zobrazí nabídku výběru typu služby a výběru konkrétní služby.
  - Uživatel zvolí typ služby a konkrétní typ služby.
  - Systém zobrazí nabídku výběrů datumu.
  - Uživatel zvolí datum.
  - Systém zobrazí nabídku výběru času rezervace po 30minutových časových slotech. Systém zobrazí pouze ty termíny, na které je možné rezervaci vytvořit.

Pokud se není možné na daný termín rezervovat, systém daný termín nezobrazí.

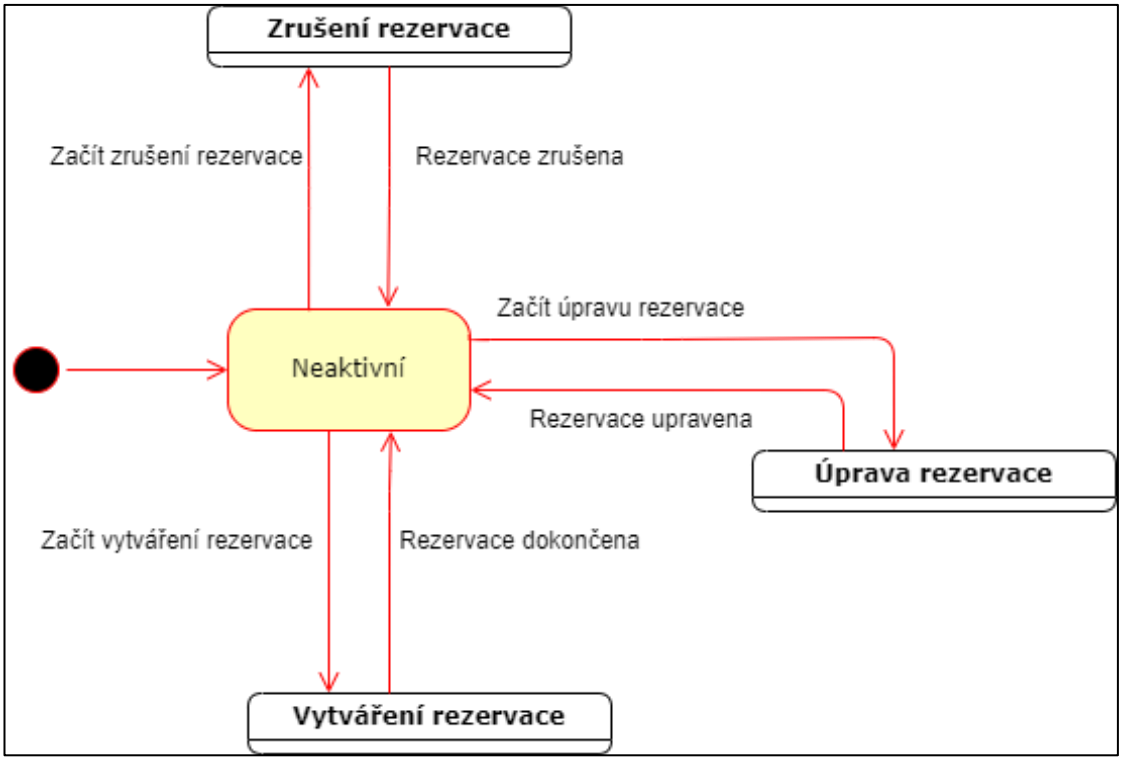
- Uživatel zvolí čas rezervace.
- Systém zobrazí formulář pro zadání informací o zákazníkovy a souhrn rezervace.
- Uživatel zadá informace do formuláře a potvrdí rezervaci
- Systém vytvoří rezervaci.

- **UC2 : Odeslat potvrzovací email**
  - Po vytvoření rezervace systém odešle zákazníkovy email o úspěšném vytvoření rezervace a souhrn dané rezervace
- **UC3 : Přihlásit do administrace**
  - Systém zobrazí formulář pro zadání uživatelského jména a hesla
  - Uživatel zadá přihlašovací údaje
- **UC4 : Ověřit heslo**
  - Pokud jsou údaje validní systém zobrazí stránku pro správu rezervací. Pokud údaje nejsou validní, systém zobrazí příslušnou chybovou hlášku.
- **UC5 : Zobrazit rezervace**
  - Systém zobrazí rezervace podle zvolené možnosti. Systém může zobrazit pouze dnešní rezervace nebo rezervace roztříděné podle typu služby.
- **UC6 : Upravit rezervaci**
  - Pokud uživatel vykonává stejný typ služby jako daná rezervace, systém zobrazí tlačítko pro úpravu rezervace. Pokud ne, systém žádné tlačítko nezobrazí.
  - Uživatel klikne na tlačítko pro úpravu rezervace
  - Systém zobrazí formulář pro úpravu rezervace
  - Uživatel změní údaje ve formuláři.
  - Pokud chce uživatel změnit čas, systém zobrazí nové časové sloty, na které je možné rezervaci vytvořit.
  - Uživatel klikne na tlačítko pro potvrzení změn.
  - Systém provede změny v dané rezervaci.
- **UC7 : Odstranit rezervaci**
  1) Pokud uživatel vykonává stejný typ služby jako daná rezervace, systém zobrazí tlačítko pro odstranění rezervace. Pokud ne, systém žádné tlačítko nezobrazí.
  2) Uživatel klikne na tlačítko odstranit rezervaci.

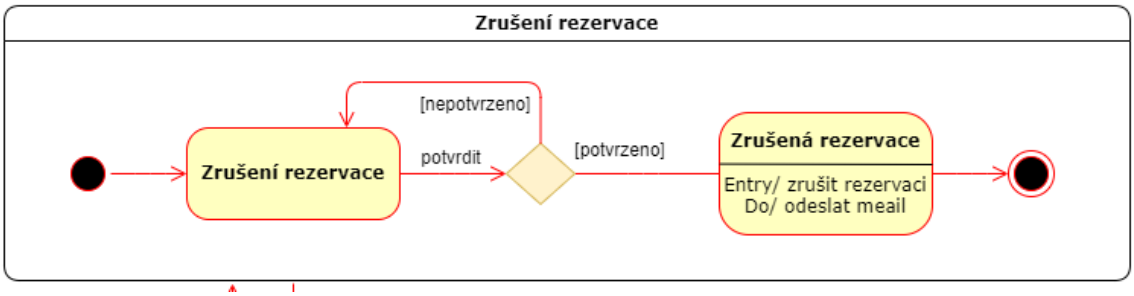
3) Systém zobrazí potvrzovací okno, zda chce uživatel vážně rezervaci smazat.
4) Pokud uživatel zvolí, že chce rezervaci vážně smazat, systém rezervaci smaže. Pokud zvolí, že rezervaci nechce smazat, systém rezervaci nesmaže.

- **UC8 : Ověřit typ služby**
  - Ověření, zda uživatel má evidovaný stejný typ služby jako daná rezervace.
  - Pokud je typ služby totožný, systém umožní provést změnu dané rezervace.
  - Pokud je typ služby rozdílný, systém neumožní provést změnu dané rezervace.
- **UC9 : Odeslat email o provedené změně**
  - Po upravení nebo smazání rezervace systém odešle email o provedených změnách
- **UC10 : Přidat uživatele**
  - Administrátor klikne na tlačítko pro správu uživatelů.
  - Systém zobrazí formulář pro přidání uživatele.
  - Administrátor zadá údaje o uživateli.
  - Pokud jsou údaje validní, systém uživatele přidá do systému. Pokud údaje nejsou validní, systém zobrazí příslušnou chybovou hlášku.
- **UC11 : Odstranit uživatele**
  - Administrátor klikne na tlačítko pro správu uživatelů.
  - Systém zobrazí formulář pro odstranění uživatele.
  - Administrátor klikne na tlačítko pro odstranění příslušeného uživatele
  - Systém zobrazí potvrzovací okno, zda chce administrátor vážně uživatele smazat.
  - Pokud administrátor zvolí, že chce uživatele vážně smazat, systém uživatele smaže. Pokud zvolí, že uživatele nechce smazat, systém uživatele nesmaže.

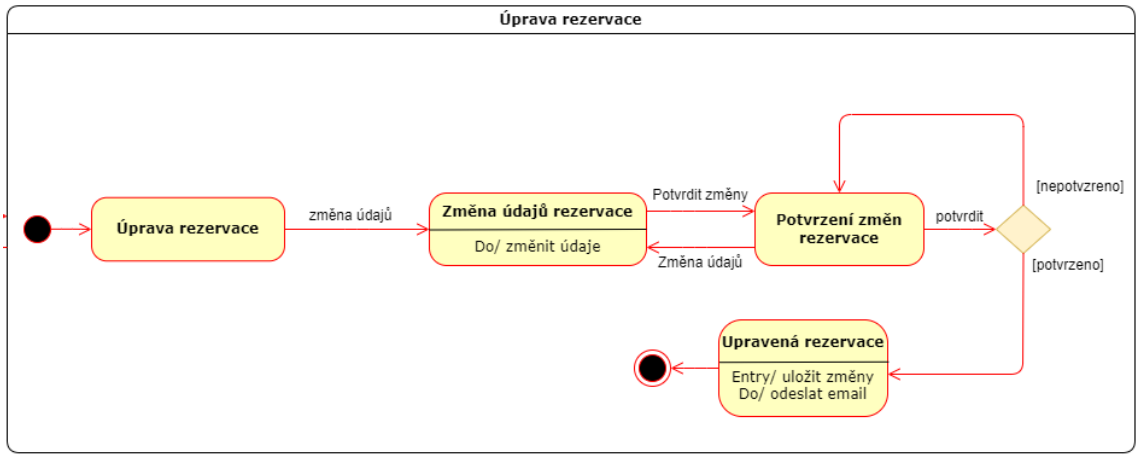
## 4.4 Vytvořený stavový diagram



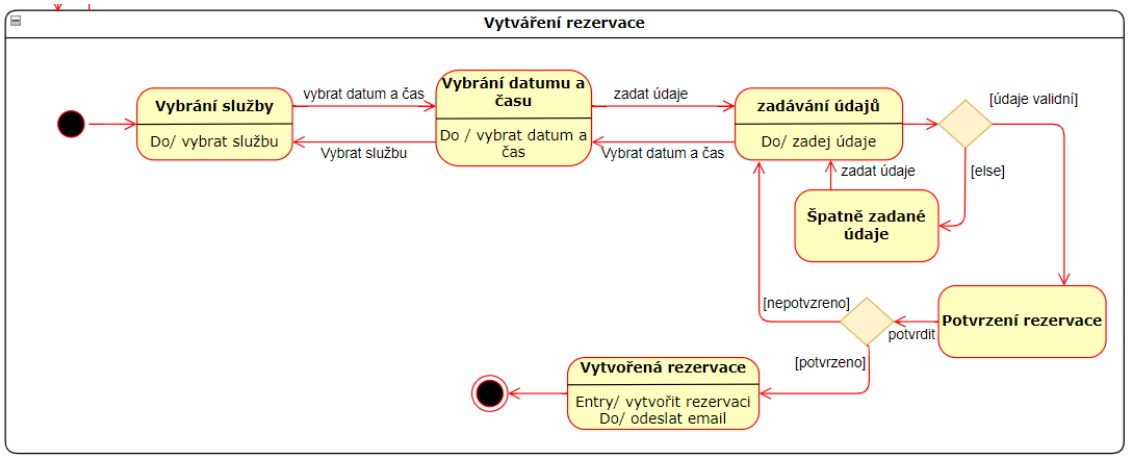

*Obrázek 5 - Vytvořený stavový diagram pro rezervace (zdroj: vlastní)*




*Obrázek 6 - Vytvořený stavový diagram pro rezervace – zrušení rezervace (zdroj: vlastní)*




*Obrázek 7 - Vytvořený stavový diagram pro rezervace – úprava rezervace (zdroj: vlastní)*



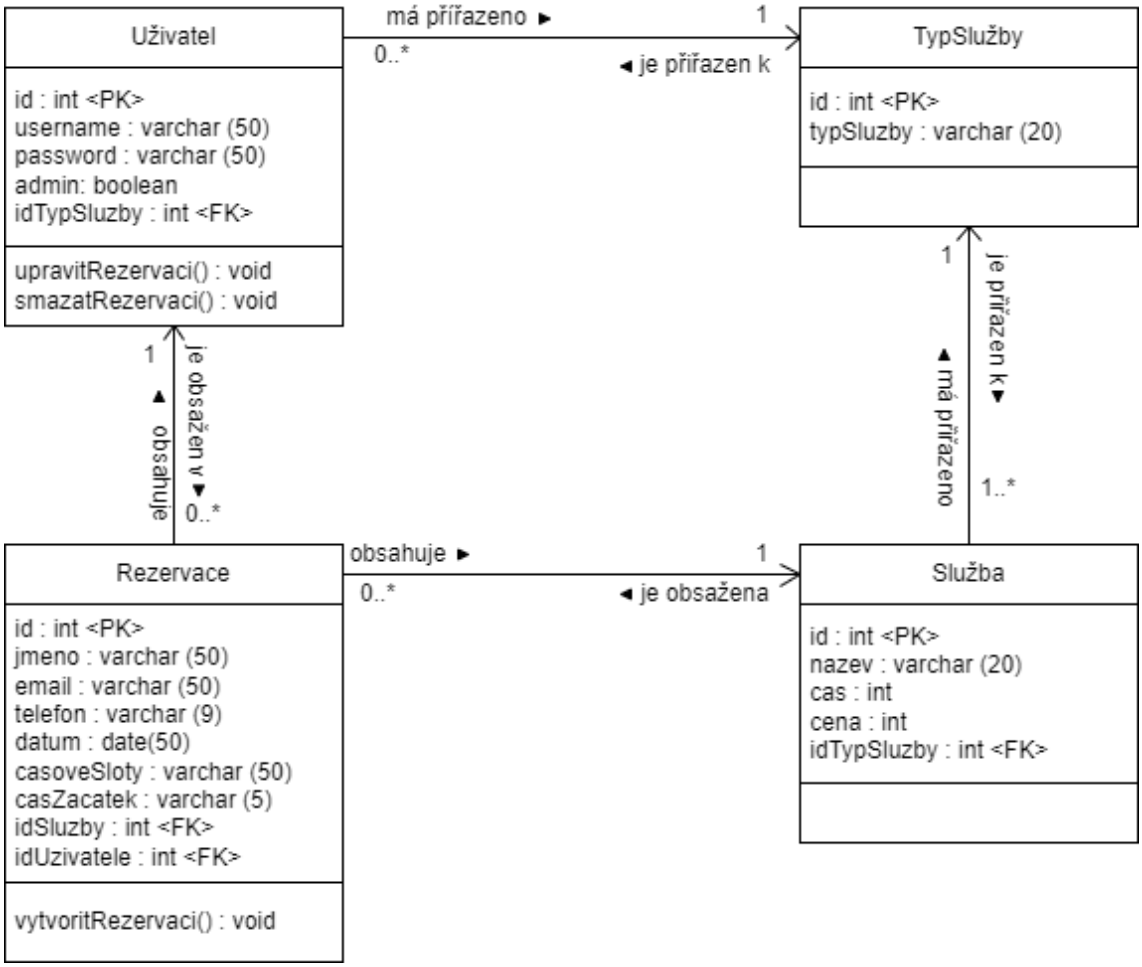

*Obrázek 8 - Vytvořený stavový diagram pro rezervace – vytváření rezervace (zdroj: vlastní)*

# 5 Návrh systému

Před implementací samotného systému je žádoucí nejprve vytvořit návrh daného systému. Návrh systému slouží k definování funkčnosti a vzhledu systému a ke stanovení jasného cíle kterého je potřeba dosáhnout. Pokud by nebyl předem vytvořený návrh, mohlo by dojít k neustálému předělávání v průběhu tvorby, a tak by mohl stoupnout čas tvorby systému a spolu s tím také náklady na vytvoření systému.

## 5.1 Vytvořený diagram tříd

Ve vytvořeném diagramu tříd jsem požil třídy *rezervace, služba, typSlužby a uživatel*. Třída *služba* představuje jednotlivé práce jako například střih, barvení nebo melír. Třída *typSlužby* označuje, zda se jedná o kadeřnictví, kosmetiku, manikúru nebo pedikúru. Třída *uživatel* představuje zaměstnance, kteří se přihlašují do administrativní části systému a třída *rezervace* obsahuje informace o rezervaci



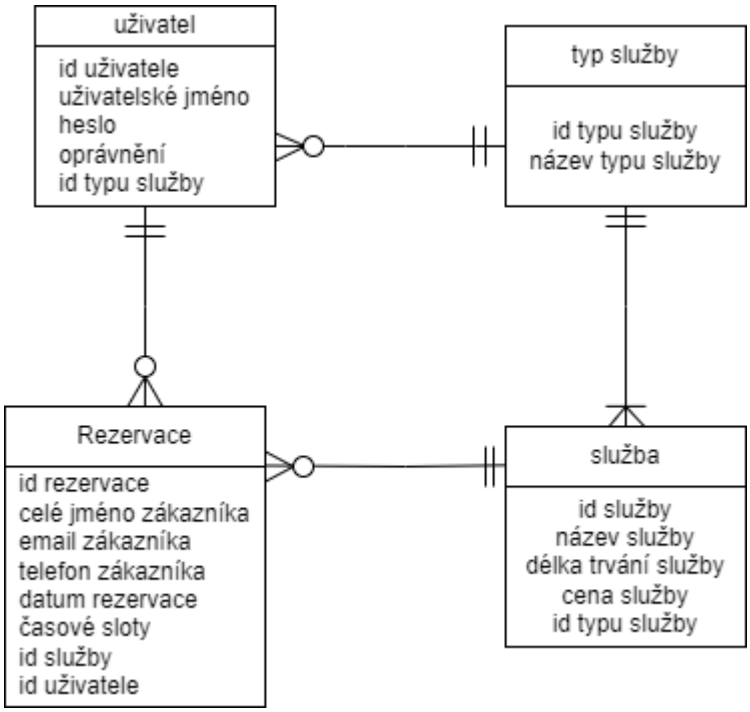

*Obrázek 9 - Vytvořený diagram tříd*

Popis vztahů mezi třídami:

- Rezervace obsahuje právě jednu službu a jedna služba je obsažena v žádné nebo ve více rezervacích.
- Služba má přiřazen právě jeden typ služby a typ služby je přiřazen k jedné nebo k více službám.
- Typ služby je přiřazen k žádnému nebo více uživatelům a jeden uživatel má přiřazenou právě jednu službu. I přesto, že salon v momentální situaci neumožňuje zaměstnávat více zaměstnanců ke stejnému typu služby, systém je připraven na možné rozšíření salonu a s tím spojený větší počet zaměstnanců pro jeden typ služby.
- Uživatel je evidován pro žádnou nebo více rezervací a rezervace má evidovaného právě jednoho uživatele. Třída *rezervace* eviduje údaj o uživateli (zaměstnanci), i přesto, že v momentální situaci salonu není tento atribut nezbytně potřebný. Avšak díky možnému budoucímu rozšíření salonu je potřeba myslet na možnost volby zaměstnance pro rezervaci.

## 5.2 Vytvořený ER diagram

Vytvořený ER diagram vychází z navrženého diagramu tříd.




*Obrázek 10 - Vytvořený ER diagram*

## 5.3 Návrh UI

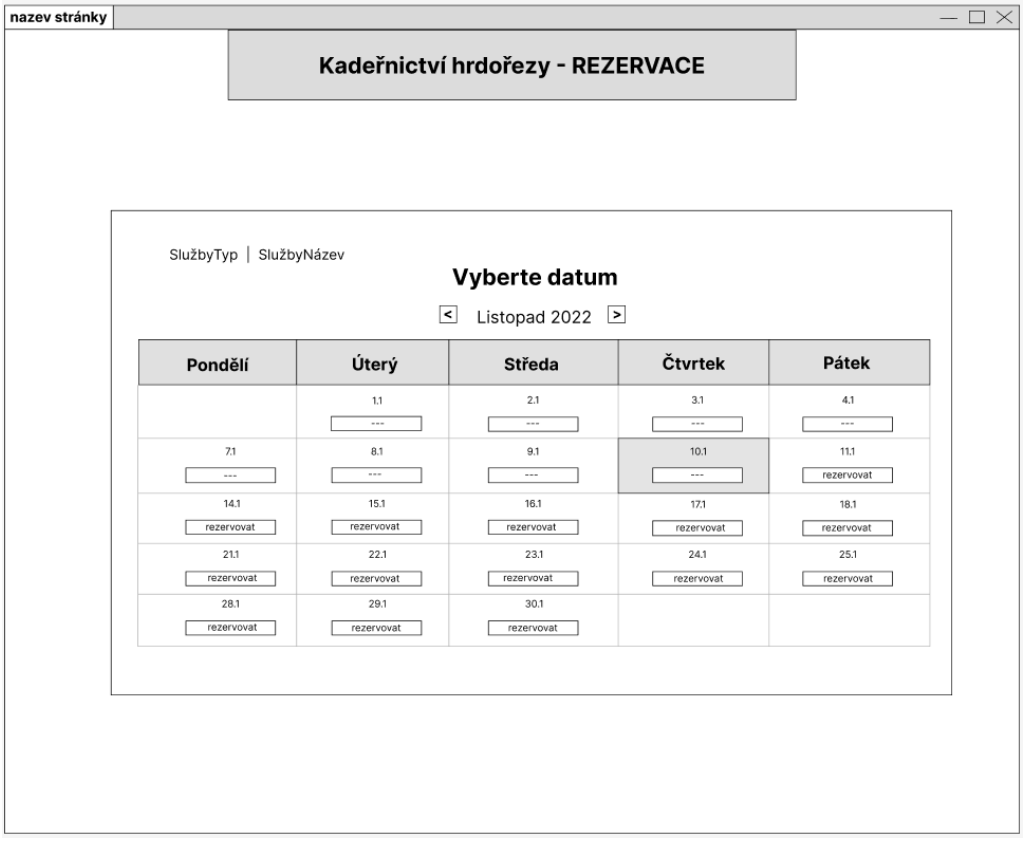
Návrh a zpracování UI je neméně důležitou součástí celého projektu. UI představuje uživatelské rozhraní. Je důležité rozlišovat pojem UX a UI. UX se zaměřuje na spokojenost uživatele a snaží se zpříjemnit uživatelský zážitek z používání celého systému. UI se zaměřuje na design jako takový. Tedy na styl písma, použité barvy nebo rozvržení rozhraní jako takového. Přesto je však potřeba se při navrhování UI zaměřovat také na UX.

Návrh UI je tvořen prototypy. Prototyp představuje jakousi základní kostru celého systému, po jeho vizuální stránce. Jednou z největších výhod při tvorbě prototypů jsou téměř nulové náklady na jeho tvorbu a možnost změnit návrh designu bez větších potíží. Prototypy se dělí na dvě základní skupiny, a to na lo-fi (low-fidelity) a hi-fi (high-fidelity). Lo-fi prototypy přestavují nejjednodušší typ prototypů. Jedná se o takové prototypy, které lze navrhnout pouze pomocí tužky a papíru. Hi-fi prototypy zaberou více času na tvorbu, avšak obsahují více detailů o systému. Proto je také jejich předělání více časově náročné. Návrh designu by měl začínat tvorbou lo-fi prototypů a až poté tvorbou hi-fi prototypů. Pro návrh tohoto systému jsem zvolil lo-fi prototypy wireframe a mockup.

### 5.3.1 Wireframe

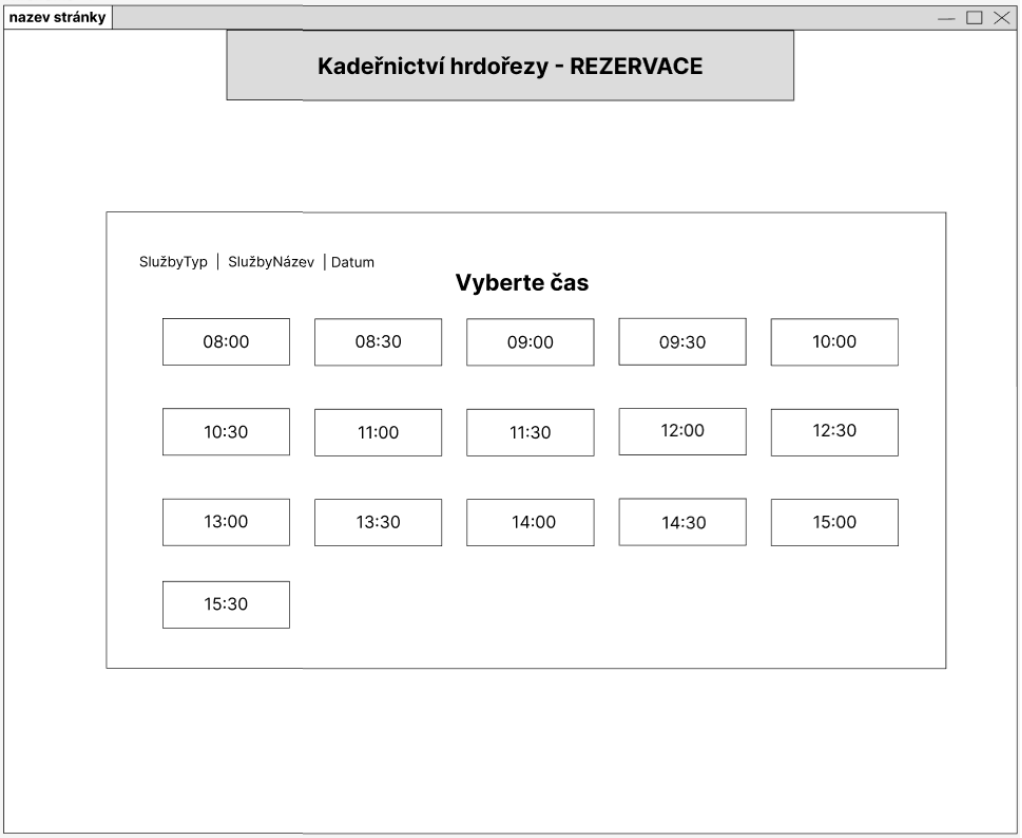
Wireframe je základním a nejjednodušším typem prototypu, který vzniká jako první při návrhu UI. Jedná se o základní navržení struktury systému. Z pravidla je pouze černobílý, a jeho hlavním cílem je pouze navrhnout rozvržení celkového designu, a ne jeho grafickou stránku. Wireframe je časově nenáročný na vytvoření a díky tomu umožňuje experimentovat a zkoušet nejrůznější návrhy systému. (Hartson, 2018)

Pro tento systém jsem vytvořil 2 typy wireframů. Wireframy pro telefony a pro počítače. Níže jsou zobrazeny příklady vytvořených wireframu pro výběr datumu rezervace, výběr času rezervace a zobrazení rezervace. Všechny vytvořené wireframy jsou obsažené v příloze.

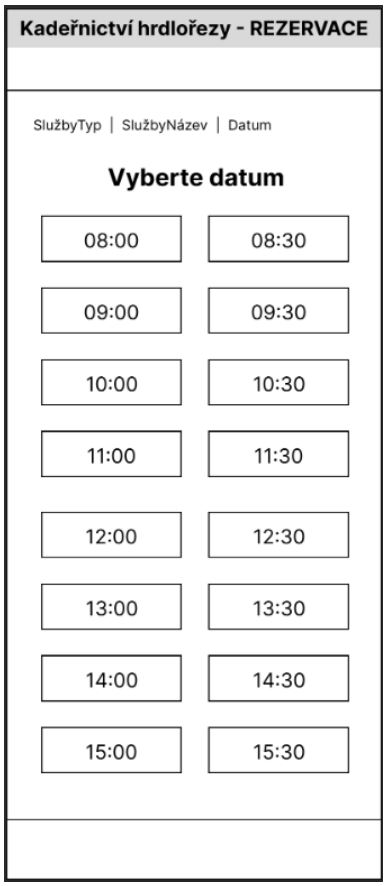



*Obrázek 11 - Vytvořený wireframe pro výběr datumu rezervace na PC – pohled zákazníka (zdroj: vlastní)*

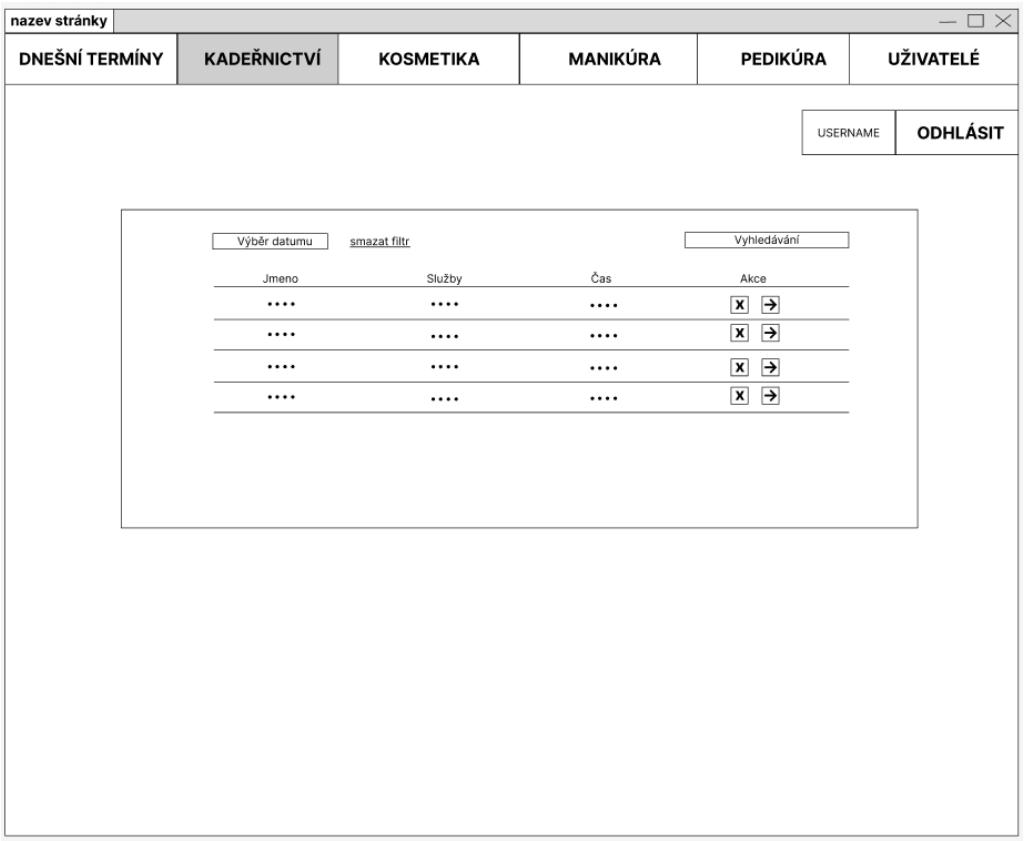

*Obrázek 12 - Vytvořený wireframe pro výběr datumu rezervace na telefonu – pohled zákazníka (zdroj: vlastní)*




*Obrázek 13 - Vytvořený wireframe pro výběr času rezervace na PC – pohled zákazníka (zdroj: vlastní)*




*Obrázek 14 - Vytvořený wireframe pro výběr času rezervace na telefonu – pohled zákazníka (zdroj: vlastní)*



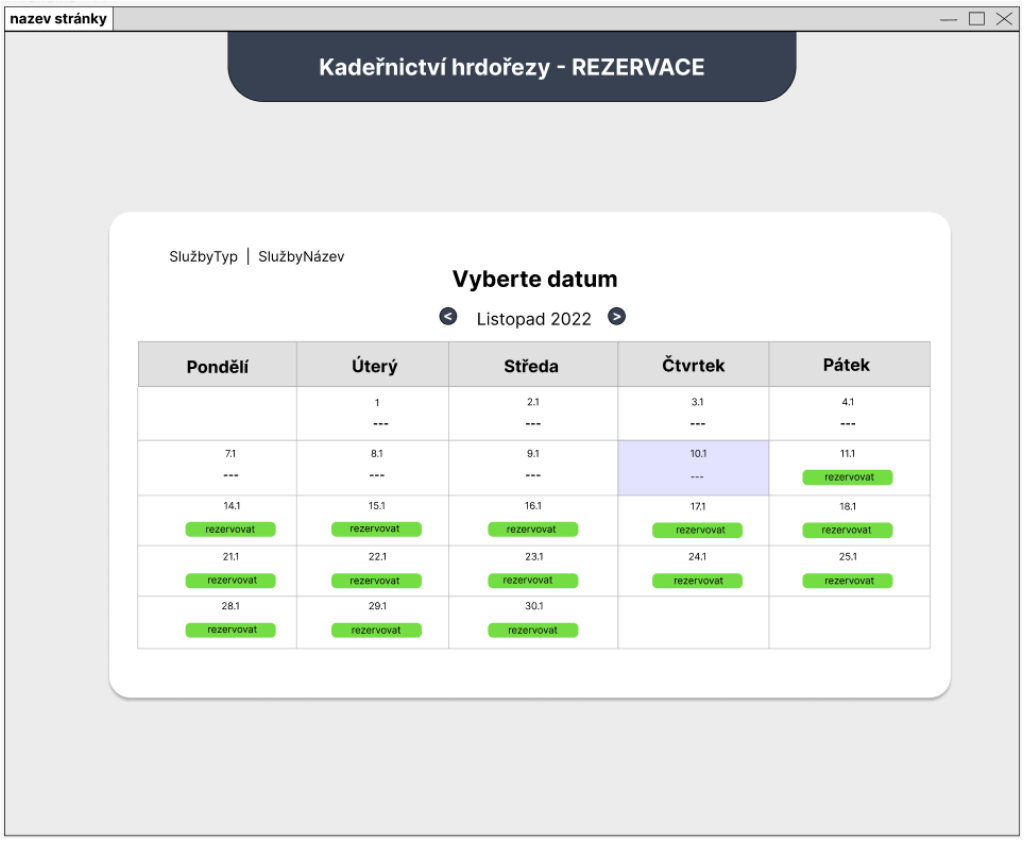

*Obrázek 15 - Vytvořený wireframe pro výpis rezervací na PC – pohled uživatele (zdroj: vlastní)*


*Obrázek 16 - Vytvořený wireframe pro výpis rezervací na telefonu – pohled uživatele (zdroj: vlastní)*

### 5.3.2 Mockup

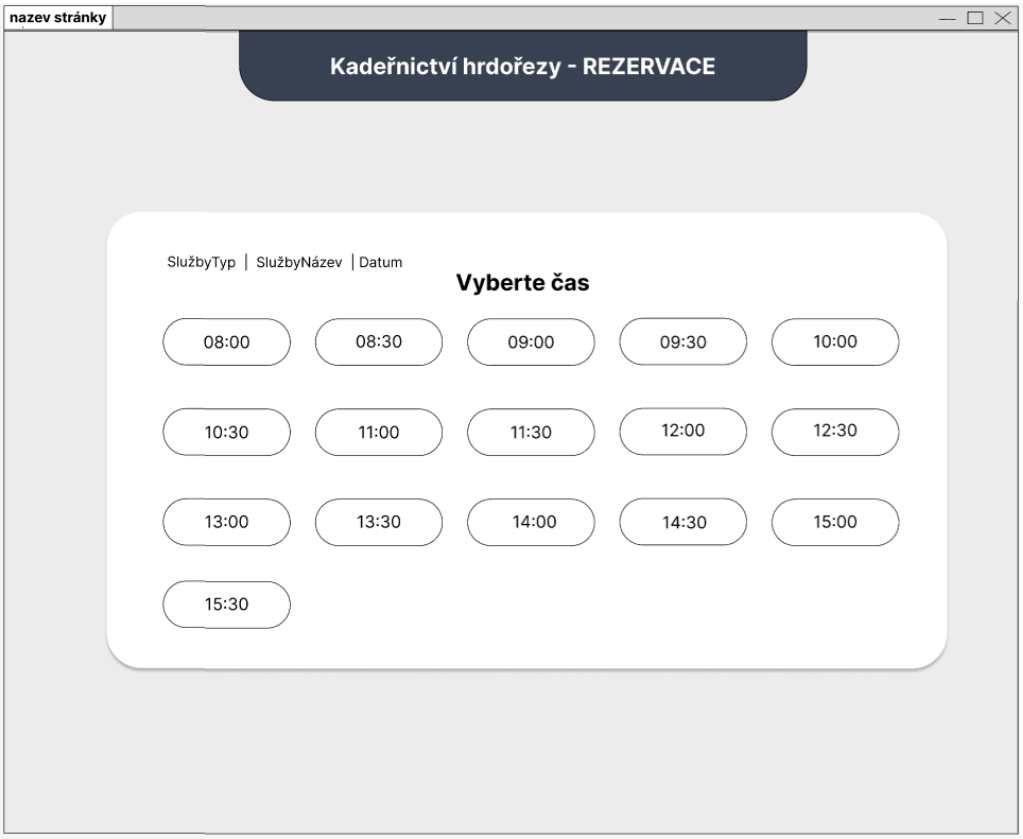
Mockup vychází z již vytvořených wireframů a doplňuje je grafikou. Mockup by tedy na rozdíl od wireframů měl představovat kompletní výslednou podobu daného systému.

Stejně jako v případě wireframů jsem pro tento systém jsem vytvořil 2 typy mockupů. Mockupy pro telefony a pro počítače. Níže jsou zobrazeny příklady vytvořených mockupů pro výběr datumu rezervace, výběr času rezervace a zobrazení rezervace. Všechny vytvořené mockupy jsou obsaženy v příloze.

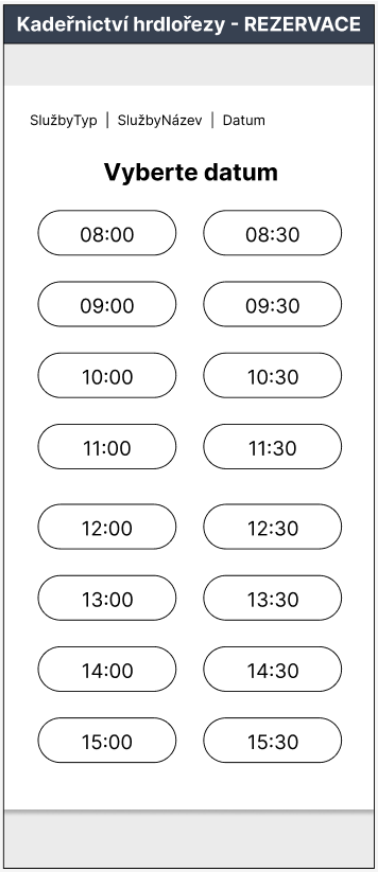



*Obrázek 17 - Vytvořený mockup pro výběr datumu rezervace na PC – pohled zákazníka (zdroj: vlastní)*

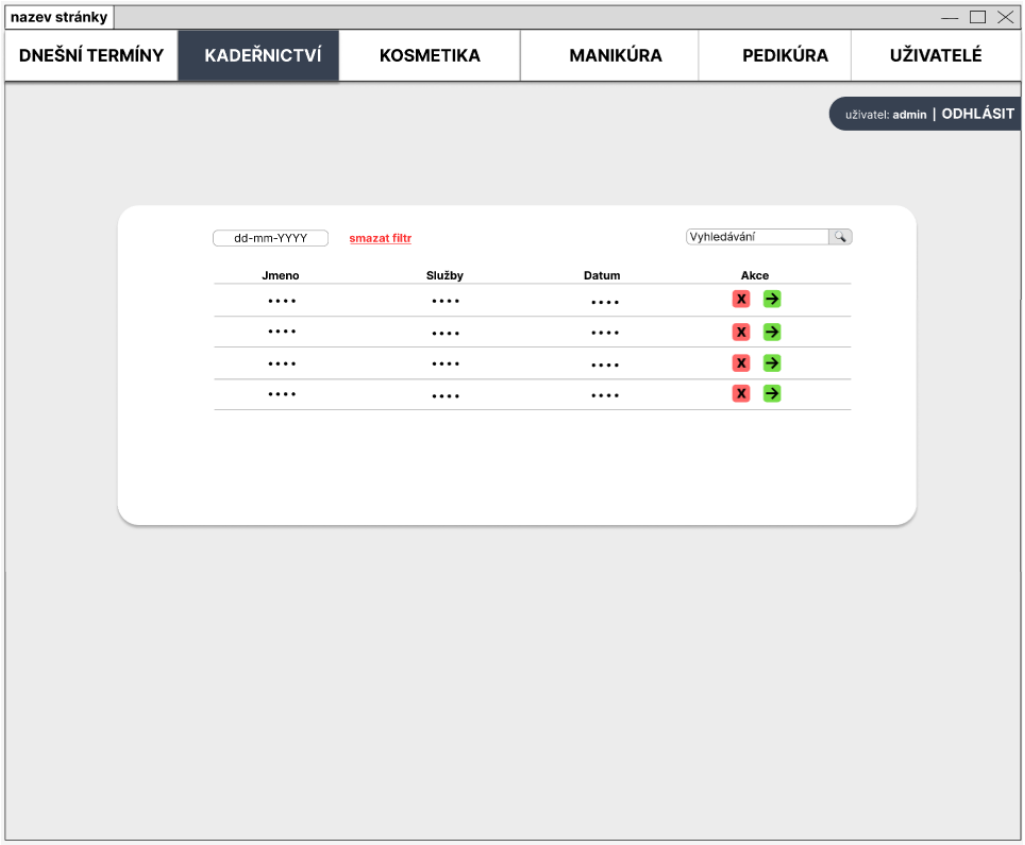

*Obrázek 18 - Vytvořený mockup pro výběr datumu rezervace na telefonu – pohled zákazníka (zdroj: vlastní)*



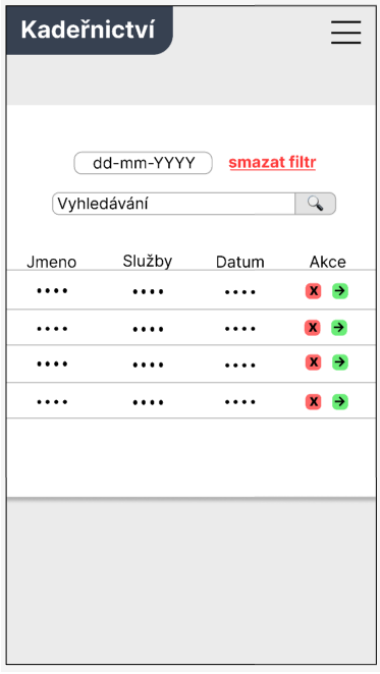

*Obrázek 19 - Vytvořený mockup pro výběr času rezervace na PC – pohled zákazníka (zdroj: vlastní)*




*Obrázek 20 - Vytvořený mockup pro výběr času rezervace na telefonu – pohled zákazníka (zdroj: vlastní)*




*Obrázek 21 - Vytvořený mockup pro výpis rezervací na PC – pohled uživatele (zdroj: vlastní)*




*Obrázek 22 - Vytvořený mockup pro výpis rezervací na telefonu – pohled uživatele (zdroj: vlastní)*

# 6 Implementace

Finální část této práce je zaměřena na implementaci daného systému. Celý rezervační systém tvoří 22 souborů. 2 CSS soubory *styleAdmin.css* a *styleRezervace.css*, 2 JS soubory *scriptAdmin.css* a *scriptRezervace.css* a 18 PHP souborů, které představují jednotlivé stránky

rezervačního systémů. Z důvodu počtu souboru jsou v následující části vypsány pouze funkce, které systém využívá a části kódů, které jsou pro systém stěžejní. Celý kód je obsažený v příloze.

## 6.1 HTML editor

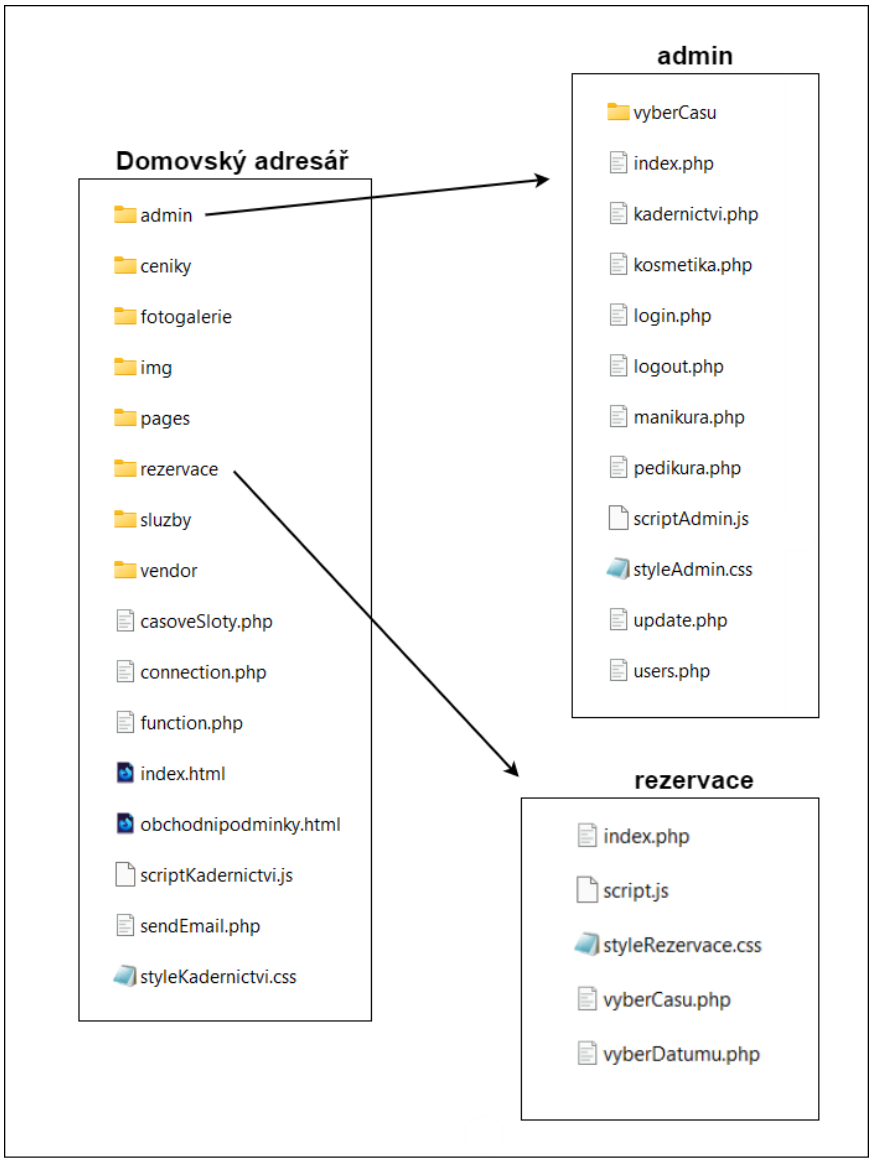
Prvním krokem implementace je zvolit vhodný program, ve kterém se bude daný systém vytvářet. Pro tuto práci jsem zvolil použití jazyků HTML,CSS, JS, a PHP, stačilo by tedy použít obyčejný textový editor. Práce v textovém editoru je ovšem velmi nepříjemná a využívají se speciálně upravené editory určené pro psaní webových stánek, tzv. HTML editory.

HTML editory značně zpříjemňují psaní webových stránek díky několika funkcionalitám, které nabízí. HTML editory například poznají klíčová slova a podle toho obarvují text, při vypisování klíčových slov napomáhají s psaním díky tzv. našeptávání, nebo usnadňují práci při procházení souborů v rámci celého adresáře souboru. Na trhu existuje mnoho HTML editoru. Některé jsou použitelné zadarmo, některé jsou placené, avšak funkčně jsou si velmi podobné. Mezi nejpoužívanější HTML editory patří například Atom, Sublime Text, Brackets, Notepad++ nebo Visual Studio Code, který byl také zvolen pro tvorbu tohoto projektu.

## 6.2 Struktura systému

Struktura celé webové stránky je členěna do několika adresářů. Kód související s rezervačním systém je obsažený v adresářích *rezervace* a *admin*. Výjimku tvoří soubory *casoveSloty.php*, *connection.php* a *function.php*, které jsou uložené přímo v domovském adresáři. Soubor *function.php* obsahuje používané funkce napříč celým dokumentem, soubor *connection.php* obsahuje propojení s databází a soubor *casoveSloty.php* obsahuje část kódu, která vytváří časové sloty, podle stanovených kritérií.

V adresáři *rezervace* jsou sobory, které souvisí s vytvářením nových rezervací a v adresáři *admin*, jsou soubory, které souvisí s částí systému, určeným pro uživatele systému, tedy zaměstnance salonu. Ostatní soubory v projektu nesouvisí s rezervačním systém ale pouze s webovou stránku pro salon.




*Obrázek 23 - Struktura webové stránky (zdroj: vlastní)*

## 6.3 Propojení s databází

Pro rezervační systém je zapotřebí databáze a technologie ke správě databáze. Pro tento rezervační systému jsem zvolil systém PHPMyAdmin. Jedná se bezplatný software, který je psaný v jazyce PHP a umožňuje spravovat databázi pomocí webového rozhraní.

Vždy když je v kódu zapotřebí pracovat s daty z databáze, je nutné navázat spojení s danou databází. Pro navázání spojení se využívá uživatelské jméno, heslo, jméno dané databáze a název serveru, na kterém je databáze nahraná. Všechny tato údaje se vypíšou do příkazu *new mysqli*. Jelikož se s databází v kódu pracuje v několika souborech, je žádoucí vytvořit separátní soubor, který obsahuje nastavení pro přístup k databázi. Soubor je nazván *connection.php*. Tento souboru se následně volá ve všech souborech, které potřebují pracovat s databází pomocí příkazu *include "cesta_k_souboru"*.

```php
$server = 'a046um.forpsi.com';
$username = 'f157313';
$password = '[redacted]';
$dbname = 'f157313';

$conn = new mysqli($server,$username,$password,$dbname);
```

*Obrázek 24 - Navázání spojení s databází (zdroj: vlastní)*

V souboru *connection.php* jsou dále vytvořeny proměnné, obsahující názvy tabulek, pro usnadnění práce s tabulkami.

```php
$rezervaceTable = 'rezervace';
$sluzbyTable = 'sluzby';
$typSluzbyTable = 'typsluzby';
$usersTable = 'uzivatele';
```

*Obrázek 25 - Nastavení názvů tabulek do proměnných (zdroj: vlastní)*

## 6.4 Funkce

Systém obsahuje 6 funkcí, které jsou všechny uložené v souboru *function.php*. Jedná se o funkce *getSluzba*, *getSluzbaTyp*, *GetTypSluzbaId*, *resultWithPagination*, *createPaginationMenu* a *sendMail*.

### 6.4.1 Funkce get

Funkce začínající na *get* jsou funkce zaměřené na získávání hodnot, předávaných pomocí url. Tyto funkce neobsahují parametry a vracejí jednu hodnotu.

```php
function getSluzba(){
    if (isset($_GET["sluzba"])){
        return $_GET["sluzba"];
    }
    return "Není předaná hodnota";
}
```

*Obrázek 26 - Funkce getSluzba (zdroj: vlastní)*

### 6.4.2 Funkce resultWithPagination

Funkce *resultWithPagination()* nastavuje při výpisu záznamů z tabulky stránkování. Obsahuje parametr *pocetZaznamu*, který určuje počet záznamů na jednu stránku a parametr *sql*, který obsahuje SQL příkaz pro výpis záznamů z dané tabulky. Tato funkce vrací hodnotu v podobě výsledku vykonání příkazu *mysqli_qeury*.

```php
function resultWithPagination($pocetZaznamu, $sql){
    include "connection.php";

    if (isset($_GET["page"])) {
        $page=$_GET["page"];
    }
    else{
        $page=1;
    }
    $result = mysqli_query($conn, $sql);
    $limitStart=($page-1)*$pocetZaznamu;
    $sql = $sql." LIMIT ".$limitStart.','.$pocetZaznamu;
    $result = mysqli_query($conn, $sql);
    return $result;
}
```

*Obrázek 27 - Funkce resultWithPagination (zdroj: vlastní)*

### 6.4.3 Funkce createPaginationMenu

Tato funkce vytváří menu pro listování mezi stránkami při výpisu z tabulky, pokud je nastaveno stránkování pomocí předchozí funkce. Funkce obsahuje parametr *pocetZaznamu*, který má totožný účel jako u funkce *resultWithPagination*, parametr *hrefLocation*, který určuje soubor, v kterém vytváříme dané menu pro daný výpis a parametr *sql*, taktéž totožný jako pro funkci *resultWithPagination*.

```php
function createPaginationMenu($pocetZaznamu, $hrefLocation, $sql){
    include "connection.php";
    if (isset($_GET["page"])) {
        $page=$_GET["page"];
    }
    else{
        $page=1;
    }
    $result = mysqli_query($conn, $sql);
    $num=mysqli_num_rows($result);
    $totalPage=ceil($num/$pocetZaznamu);
    echo "<div class='text-center mt-10'>";
    for ($i=1; $i <= $totalPage; $i++) {
        if ($page==$i) {
            echo "<a href='".$hrefLocation."?page=".$i."' class='border border-2
            border-gray-600 bg-gray-600 text-white px-2 py-1 mx-1 rounded-lg'>".$i.
            "</a>";
        }
        else{
            echo "<a href='".$hrefLocation."?page=".$i."' class='border border-2
            border-black px-2 py-1 mx-1 rounded-lg'>".$i."</a>";
        }
    }
    echo "</div>";
}
```

*Obrázek 28 - Funkce createPaginationMenu (zdroj: vlastní)*

### 6.4.4 Funkce sendMail

Funkce zajišťující odesílání emailů informujících o vytvoření, úpravě nebo smazání rezervace. Pro odesílání emailu byla využita PHP knihovna PHPMailer, která znatelně usnadňuje nastavení odesílání emailů. Funkce obsahuje parametry *jmeno. datum, cas, sluzba a typSluzby*, které se využívají pro výpis souhrnu rezervace. Dále obsahuje parametr *prijemce*, který nastavuje emailovou adresu, na který se daný email odešle a parametr *status*, který určuje, zda se jedná o email informující o potvrzení, upravení nebo smazání rezervace.

```php
function sendMail($jmeno, $datum, $cas, $sluzba, $typSluzby, $prijemce, $status){
    $email = "kadernictvihrdlorezy@gmail.com";
    $name = "Kadeřnictví Hrdlořezy";
    if ($status == "confirm") {
        $subject = "Potvrzení rezervace";
        $message =
        "Vaše rezervace byla vytvořena.

        Souhrn rezervace
        Jméno: ".$jmeno."
        Den konání rezervace: ".$datum."
        Čas rezervace: ".$cas."
        Služba: ".$sluzba." - ".$typSluzby;
        }
    elseif ($status == "change") {···
    elseif ($status == "delete") {···
    }
    $mail = new PHPMailer(true);
    $mail->isSMTP();
    $mail->CharSet = "UTF-8";
    $mail->SMTPAuth = true;
    $mail->SMTPSecure = "ssl";
    $mail->Host = "smtp.gmail.com";
    $mail->Port = 465;
    $mail->Username = "kadernictvihrdlorezy@gmail.com";
    $mail->Password = [redacted];
    $mail->setFrom($email, $name);
    $mail->addAddress($prijemce);
    $mail->Subject = $subject;
    $mail->Body = $message;
    $mail->send();
}
```

*Obrázek 29 - Funkce sendMail (zdroj: vlastní)*

## 6.5 Tvorba časových slotů

Jednou z nejpodstatnějších částí kódu je tvorba časových slotů. Časové sloty jsou vytvářené po 30 minutách. Tato hodnota je nastavena v proměnné *dobaTrvani*. V proměnné *zacatek* a *konec* jsou uložené hodnoty začátku a konce pracovní doby. Tato část kódu je uložená v separátním souboru *casoveSloty.php*. Součástí soubor je také funkce *casy*, po jejímž zavolání se vrací pole, které obsahuje všechny časové sloty.

Každá služba má specifikovanou časovou náročnost na provedení služby v minutách. Podle časové náročnosti a hodnoty určující dobu trvání jednoho časového slotu se automaticky vypočítává počet časových slotů pro jednu službu. Při vytváření rezervace jsou u výběru času následně zobrazeny pouze ty časy, na které je možné vytvořit rezervaci.

```php
<?php
    $dobaTrvani = 30;
    $vymazani = 0;
    $zacatek = "08:00";
    $konec = "16:00";


    function casy($dobaTrvani, $vymazani, $zacatek, $konec){
        $zacatek = new DateTime($zacatek);
        $konec = new DateTime($konec);
        $interval = new DateInterval("PT".$dobaTrvani."M");
        $vymazatInterval = new DateInterval("PT".$vymazani."M");
        $slots = array();

        for ($i=$zacatek; $i < $konec ; $i->add($interval)->add($vymazatInterval)) {
            $konecSlotu = clone $i;
            $konecSlotu->add($interval);
            if ($konecSlotu>$konec) {
                break;
            }
            $slots[] = $i->format("H:i")/* ." - ".$konecSlotu->format("H:i") */;
        }
        return $slots;

    }
?>
```

*Obrázek 30 - Vytvoření časových slotů (zdroj: vlastní)*

## 6.6 Nahrání systému na doménu

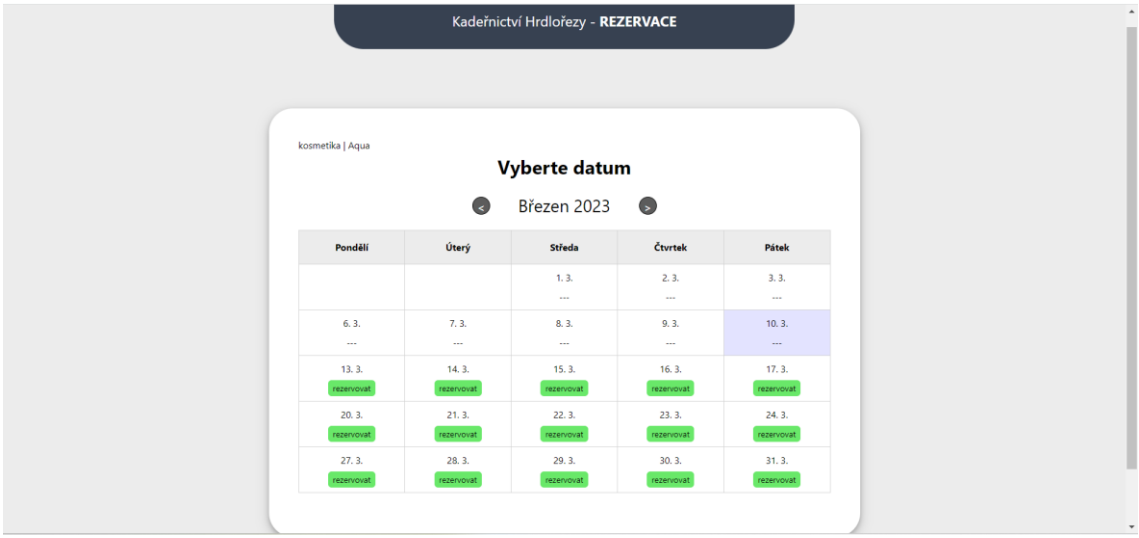
Systém byl nahrán na testovací doménu www.marekmusilweb.cz/kadernictvi. Část rezervačního systému pro správu rezervací je dostupná skrze doménu www.marekmusilweb.cz/kadernictvi/admin. Doména byl pořízena prostřednictvím společnosti Forpsi. Pro přístup do časti pro správu rezervací lze použít následující účty:

- Účet pro kadeřnictví
    - Přihlašovací jméno: kadernictvi
    - Heslo: kadernictvi
- Účet pro kosmetiku
    - Přihlašovací jméno: kosmetika
    - Heslo: kosmetika
- Účet pro manikúru
    - Přihlašovací jméno: manikura
    - Heslo: manikura
- Účet pro pedikúru
    - Přihlašovací jméno: pedikura

- Heslo: pedikura
- Účet pro administrátora
  - Přihlašovací jméno: admin
  - Heslo: admin

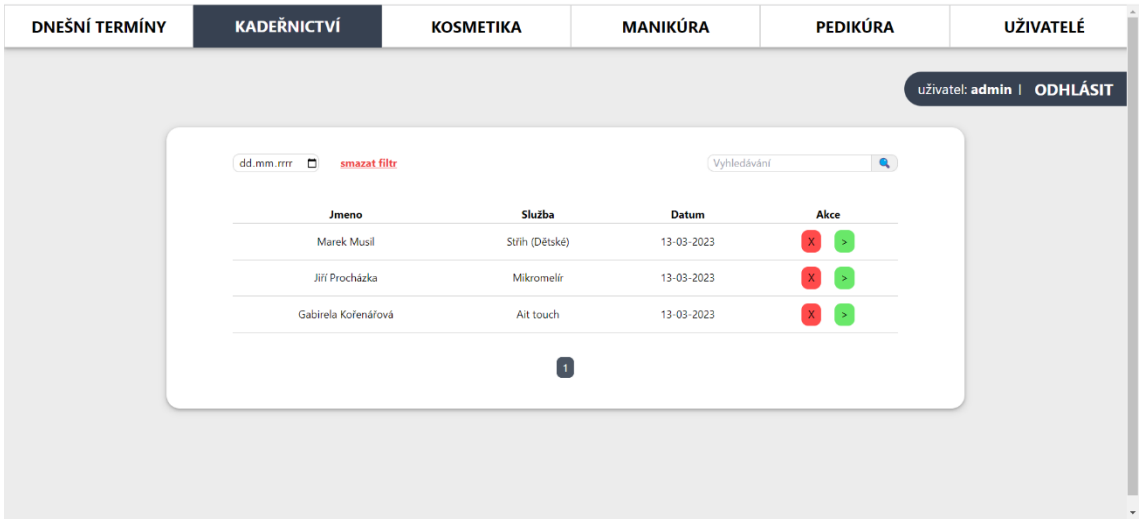
## 6.7 Výsledná podoba systému

Níže jsou zobrazeny výsledné stránky systému pro výběr datumu rezervace, výběr času rezervace a zobrazení rezervací pro kadeřnictví. Celý systém lze zobrazit na doméně www.marekmusilweb.cz/kadernictvi a část pro správu rezervací na doméně www.marekmusilweb.cz/kadernictvi/admin.




*Obrázek 31 - Výběr datumu rezervace (zdroj: vlastní)*


*Obrázek 32 - Výběr času rezervace (zdroj: vlastní)*




*Obrázek 33 - Výpis rezervací pro kadeřnictví (zdroj: vlastní)*

# 7 Zhodnocení vytvořeného systému

Systém byl vytvořen na míru klientovy, a splňuje klientovu představu. I přesto, že je systém pro klienta vyhovující, existuje několik funkcí, které by bylo možné k systému přidat a daný systému udělat více univerzální. Z mého pohledu mezi nejvíce limitující faktory patří omezené množství zaměstnanců salonu a s tím spojená pouze volba typu služby, a ne daného zaměstnance a nemožnost vytvoření zákaznických účtů.

Systém byl vytvořen velmi specificky pro konkrétní salon, který má kapacitu obsluhovat pouze 1 zaměstnance pro 1 službu a zaměstnanci jsou zaměstnaní na plný úvazek s pevnou pracovní dobou. Z tohoto důvodu je při vytváření rezervace zbytečné vybírat jméno zaměstnance ke kterému se chce rezervace vytvořit a stačí specifikovat pouze typ služby a danou službu. I přesto, že salon může zaměstnávat pouze jednoho zaměstnance pro jednu službu, byl systém navržen tak, aby v případě budoucího rozšíření salonu bylo možné evidovat více zaměstnanců ke stejnému typu služby. Systém uživatelské účty eviduje s již přiřazeným typem služby a je ho tedy možné jednoduše upravit pro možnost volby zaměstnance k rezervaci.

Klient se rozhodl neevidovat zákazníky a neumožnit jim vytvoření profilu. Zavedení teto funkcionality by umožňovalo lepší přehled o svých rezervacích pro zákazníky, možnost zapamatování zákaznických údajů a s tím spojené usnadnění vypisování údajů při vytváření rezervace a také snadnější zrušení rezervace. Klientovým přání však je, aby klient pro zrušení rezervace musel kontaktovat salon napřímo a ostatní výhody, které by tato funkoncalita přinesla zákazníkům nejsou pro klienta podstatné. Pro implementaci této funkcionality by bylo zapotřebí pouze vytvoření nové tabulky, nové stránky pro administraci zákaznického účtu a lehkou úpravu kódu. Tato funkcionalita je tedy taktéž lehce implementovatelná.

# 8 Závěr

Výsledkem této bakalářské práce je vytvořený rezervační systém pro kadeřnický salon. Práce obsahuje analýzu existujících řešení, návrh systému a implementaci systému. Rezervační systém je plně funkční, umožňuje rychle a jednoduše vytvořit rezervaci a ze strany zaměstnance také rezervaci upravovat. Všechny požadavky na systém byly splněny. Funkčnost rezervačního systému je jednoduchá a efektivní. Byl vytvořen Use Case diagram, stavový diagram, diagram tříd a ER diagram pro návrh funkčnosti systému a wireframy a mockupy pro návrh UI systému.

Celý systém je vytvořen v HTML editoru VS Code a je nahraný na doméně www.marekmusilweb.cz/kadernictvi. Rezervační systém byl klientem schválen a je tak uznán za vyhovující. Rezervační systém prozatím není nahraný na doméně kadeřnického salonu, avšak je na to připravený, a po rozhodnutí klienta o uvedení do ostrého provozu je rezervační systém schopen být plně nasazen. Závěrečná část práce obsahuje zhodnocení vytvořeného systému a návrhy na úpravu systému, aby byl systém více univerzální.

V této bakalářské práci byly splněny všechny cíle. Ať už cíle ve formě požadavků na systém od klienta, tak všechny cíle stanové mnou, jakožto autorem této práce.

# 9 Seznam použitých zdrojů

Bootstrap Grid System, c1999-2023. In: *W3Schools* [online]. Hommersåk: Refsnes Data [cit. 2023-03-11]. Dostupné z: https://www.w3schools.com/bootstrap/bootstrap_grid_system.asp

Co je to CSS – kaskádové styly, 2023. In: *Artster* [online]. Artster.cz [cit. 2023-03-11]. Dostupné z: https://artster.cz/co-je-to-css/

CSS Tutorial, c1999-2023. In: *W3Schools* [online]. Hommersåk: Refsnes Data [cit. 2023-03-11]. Dostupné z: https://www.w3schools.com/css/

ERIKSSON, Hans-Erik, Magnus PENKER, Brian LYONS a David FADO, 2004. *UML 2 Toolkit*. 1. vydání. Indianapolis: Wiley Publishnig. ISBN 0-471-46361-2.

GORBACHENKO, Pavel, 2023. Functional vs Non-Functional Requirements. In: *Enkonix* [online]. Oud Blaricummerweg: Enkonix [cit. 2023-03-13]. Dostupné z: https://enkonix.com/blog/functional-requirements-vs-non-functional/

Guide to CSS Framework, 2022. In: *Simplilearn* [online]. San Francisco: Simplilearn [cit. 2023-03-13]. Dostupné z: https://www.simplilearn.com/tutorials/css-tutorial/css-framework

HARTSON, Rex a Pardha S. PYLA, 2018. *The UX Book: Agile UX Design for a Quality User Experience*. 2nd ed. Cambridge: Morgan Kaufmann Publishers. ISBN 978-0-12-805342-3.

HTML5 - Overview, 2023. In: *Tutorials Point* [online]. Kavuri Hills: Tutorials Point [cit. 2023-03-11]. Dostupné z: https://www.tutorialspoint.com/html5/html5_overview.htm

HTML5 - Syntax, 2023. In: *Tutorials Point* [online]. Kavuri Hills: Tutorials Point [cit. 2023-03-11]. Dostupné z: https://www.tutorialspoint.com/html5/html5_syntax.htm
JavaScript pro začátečníky, 2023. In: *Rascasone* [online]. Praha: Rascasone [cit. 2023-03-11]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky

KAUR ARORA, Simran, 2022. Best JavaScript Frameworks. In: *Hackr* [online]. Miami: Hackr [cit. 2023-03-14]. Dostupné z: https://hackr.io/blog/best-javascript-frameworks

KOĎOUSKOVÁ, Barbora, 2022. JavaScript pro začátečníky. In: *Rascasone* [online]. Praha: Rascasone [cit. 2023-03-14]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky

M. SCHAFER, Steven, 2009. *HTML, XHTML a CSS: Bible pro tvorbu www stránek*. 1. vyd. Praha: Grada. ISBN 978-80-247-2850-6.

MURPHY, Coner, 2022. Tailwind CSS vs. Bootstrap. In: *Prismic* [online]. San Francisco: Prismic [cit. 2023-03-14]. Dostupné z: https://prismic.io/blog/tailwind-vs-bootstrap

NIXON, Robin, 2021. *Learning PHP, MySQL & JavaScript*. 6th ed. Sebastopol: O'Reilly Media. ISBN 978-1-492-09382-4.

PHP Operators, c1999-2023. In: *W3Schools* [online]. Hommersåk: Refsnes Data [cit. 2023-03-11]. Dostupné z: https://www.w3schools.com/php/php_operators.asp

Reservanto [online], 2023. Liberec: Reservanto [cit. 2023-03-13]. Dostupné z: https://www.reservanto.cz/

Reservatic [online], c2015-2023. Ostrava: Reservatic [cit. 2023-03-13]. Dostupné z: https://reservatic.com/

Reservio [online], c2012–2023. Brno: Reservio [cit. 2023-03-13]. Dostupné z: https://www.reservio.com/cs

Responsive Design - Tailwind CSS, 2023. In: *Tailwind CSS* [online]. Tailwind Labs [cit. 2023-03-11]. Dostupné z: https://tailwindcss.com/docs/responsive-design

SimplyBook.me [online], 2021. Lemesos: SimplyBook.me [cit. 2023-03-13]. Dostupné z: https://simplybook.me/en/

SPURLOCK, Jake, 2013. *Bootstrap: Responsive Web Development*. 1st ed. Sebastopol: O'Reilly Media. ISBN 9781449344597.

Tailwind CSS vs. Bootstrap, 2022. In: *Prismic* [online]. San Francisco: Prismic [cit. 2023-03-11]. Dostupné z: https://prismic.io/blog/tailwind-vs-bootstrap

Teamup [online], 2023. Teamup Solutions AG [cit. 2023-03-13]. Dostupné z: https://www.teamup.com/

TERBEROVÁ, Hana, 2023. Co je to CSS – kaskádové styly. In: *Artster* [online]. Artster [cit. 2023-03-14]. Dostupné z: https://artster.cz/co-je-to-css/

# 10 Seznam obrázků, tabulek, grafů a zkratek

## 10.1 Seznam obrázků

## 10.2 Seznam tabulek