Využití JavaScript knihoven při tvorbě interaktivních webových uživatelských rozhraní

Bc. Martin Butko

Diplomová práca
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Martin Butko**
Osobní číslo:	**A18415**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Prezenční**
Téma práce:	**Využití JavaScript knihoven při tvorbě interaktivních webových uživatelských rozhraní**
Téma práce anglicky:	**The Use of JavaScript Libraries to Create Interactive Web User Interfaces**

Zásady pro vypracování

1. Prostudujte a popište některé dostupné JavaScript nástroje vhodné pro návrh moderního interaktivního uživatelského prostředí webové aplikace.
2. Zaměřte se také na nástroje umožňující návrh nestandardního uživatelského prostředí například s využitím grafického formátu SVG, provnejte je a uveďte vhodné způsoby aplikace.
3. Vyberte některý z popsaných nástrojů pro praktickou implementaci interaktivního rozhraní, jako je například rezervační systém kina. Zabývejte se způsobem vizualizace míst k sezení či rezervaci.
4. Teoreticky navrhněte aplikaci dle zvoleného a popsaného případu užití.
5. Dle návrhu proveďte implementaci funkční aplikace, přičemž se zaměřte zejména na provedení uživatelského rozhraní.

Rozsah diplomové práce:
Rozsah příloh:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PEARLMAN, Ellen a Lorien HOUSE. SVG for web developers. Upper Saddle River (New Jersey): Prentice Hall PTR, 2003. ISBN 978-0-13-100499-3.
2. HAVERBEKE, Marijn. Eloquent Javascript: a modern introduction to programming. San Francisco: No Starch Press, 2011. ISBN 978-1-59327-282-1.
3. DUCKETT, Jon. HTML & CSS: design and build websites. Indianapolis: John Wiley, 2011. ISBN 978-1-118-00818-8.
4. KUMAR, Dhananjay. Angular Essentials: The Essential Guide to Learn Angular. New Delhi: BPB Publications, 2019. ISBN 978-93-88511-24-7.
5. YAHIAOUI, Houssem. Firebase Cookbook: Over 70 recipes to help you create real-time web and mobile applications with Firebase. Birmingham: Packt Publishing, 2017. ISBN 978-1-78829-239-9.
6. FREEMAN, Adam. Essential TypeScript: From Beginner to Pro. New York: Apress, 2019. ISBN 978-1-4842-4979-6.
7. LARSEN, Rob. Beginning HTML and CSS. Hoboken (New Jersey): John Wiley, 2013. ISBN 978-1-118-41651-8.
8. EISENBERG, J.David. SVG Essentials: Producing Scalable Vector Graphics with XML. Sebastopol (Kalifornia): O'Reilly Media, 2002. ISBN 978-0-596-00223-7.

Vedoucí diplomové práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **28. listopadu 2019**
Termín odevzdání diplomové práce: **15. května 2020**

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 04.08.2020

Martin Butko, v.r.
podpis diplomanta

ABSTRAKT

Témou diplomovej práce bolo popísať JavaScriptové nástroje pre tvorbu front-endovej časti webovej aplikácie, ako aj nástroje pracujúce s grafickým formátom SVG. Hlavným cieľom bolo vybrať vhodné nástroje a implementovať dynamický rezervačný systém s interaktívnym používateľským rozhraním, ktorý by bol modulárny a použiteľný v už existujúcich projektoch.

Teoretická časť sa zaoberá popisom voľne dostupných JavaScriptových nástrojov určených pre tvorbu front-endu, ako aj popisom návrhového vzoru MVC, ktorý väčšina z nich používa. Taktiež sú tu popísané nástroje určené pre tvorbu a manipuláciu grafických útvarov formátu SVG.

Praktická časť obsahuje návrh aplikácie, porovnanie SVG nástrojov a tiež návrh modulárnej štruktúry a možnosť znovu použitia aplikácie. Ďalej je v rámci praktickej časti popísaný výber vhodných nástrojov, implementácia a publikácia výslednej aplikácie.

Kľúčové slová: SVG, rezervačný systém, webová aplikácia, JavaScript framework

ABSTRACT

The topic of the diploma thesis was to describe JavaScript tools for creating a front-end part of a web application, as well as tools working with the SVG graphic format. The main goal was to select suitable tools and implement a dynamic reservation system with an interactive user interface, that would be modular and usable in existing projects.

The theoretical part deals with the description of freely available JavaScript tools for creating a front-end, as well as a description of the MVC design pattern, which most of them use. Tools for creating and manipulating SVG graphics are also described here.

The practical part contains the design of the application, comparison of SVG tools and also the design of a modular structure and the possibility of reusing the application. Furthermore, the practical part describes the selection of suitable tools, implementation and publication of the final application.

Keywords: SVG, reservation system, web application, JavaScript framework

Rád by som sa poďakoval svojmu vedúcemu Ing. Radkovi Valovi, Ph.D. za cenné rady a pripomienky poskytnuté pri tvorbe tejto práce.

Prehlasujem, že odovzdaná verzia diplomovej práce a verzia elektronická nahratá do IS/STAG sú totožné.

OBSAH

ÚVOD

Vo svete webových aplikácií sa čoraz viac začínajú používať JavaScriptové nástroje a frameworky, ktoré zakladajú najmä na svojej modularite a prehľadnej architektúre aplikácií. Pomocou týchto nástrojov je tiež možné vytvárať moderné aplikácie s interaktívnym používateľským rozhraním, ktoré okamžite reaguje na zmenu dát.

Pri aplikáciách, ktoré obsahujú používateľské rozhrania so zložitejšími vektorovými prvkami grafiky, je možné použiť JavaScriptové nástroje pre prácu s SVG formátom. Takýchto nástrojov je voľne dostupných viac, a každý má svoje výhody, či nevýhody.

S využitím týchto nástrojov je možné vytvárať webové aplikácie s netradičnými a veľmi dynamickými používateľskými rozhraniami. Keďže takto implementované aplikácie sú modulárne, je možné ich jednoducho znovu použiť. Túto myšlienku poníma aj výsledná aplikácia tejto práce. Jedná sa o dynamický interaktívny rezervačný systém, ktorý je zložený z komponentov a je možné ho znovu použiť v inom projekte. Rezervačný systém by mal umožniť vytváranie podujatí, na ktorých sa budú rezervovať miesta. Vytváranie miest by malo byť dynamické a umožniť podporu pre rôzne typy podujatí, ako napríklad kiná. Publikácia takéhoto rezervačného systému ako open source by umožnila komunite používateľov jednoduchšiu správu rezervácií.

Prvá kapitola práce sa venuje popisu JavaScriptových nástrojov pre návrh interaktívneho používateľského prostredia webových aplikácií. Popísaný je aj návrhový vzor MVC, ktorý väčšina týchto nástrojov používa. Ďalej sú priblížené nástroje využívajúce grafický formát SVG, ktoré uľahčujú prácu s vektorovou grafikou.

V práci sa následne nachádza úvod ku rezervačným systémom ako aj návrh samotnej aplikácie. Zhrnuté sú požiadavky na používateľské rozhranie, modulárnosť aplikácie a taktiež sú objasnené funkčné a nefunkčné požiadavky. Následne sú jednotlivé SVG knižnice porovnané podľa rôznych kritérií. V ďalšej kapitole sú vybraté vhodné nástroje a technológie pre implementáciu výslednej aplikácie. Posledná kapitola sa venuje implementácií navrhnutej webovej aplikácie. To zahŕňa súborovú štruktúru projektu, hierarchiu komponentov a dátový model. Následne je vysvetlené uloženie dát v databáze a manipulácia s nimi. Na záver je popísaná implementácia aplikácie podľa požadovaných funkcií a publikácia na platforme GitHub.

I. TEORETICKÁ ČASŤ

1 JAVASCRIPT NÁSTROJE PRE NÁVRH INTERAKTÍVNEHO POUŽÍVATEĽSKÉHO ROZHRANIA WEBOVEJ APLIKÁCIE

JavaScript je v súčasnej modernej dobe jeden z najpoužívanejších a najpopulárnejších programovacích jazykov tvoriaci veľkú komunitu vývojárov. Existuje a inovuje sa mnoho nástrojov, ktoré uľahčujú prácu s týmto programovacím jazykom. Tieto vývojové nástroje poskytujú pre vývojárov veľké množstvo funkcií a jednoduchšiu syntax. [1]

Tieto JavaScriptové nástroje sa delia do dvoch hlavných kategórií a to knižnice a frameworky.

1.1 JavaScript knižnica

Je to kolekcia nejakej konkrétnej funkcionality vo forme súboru s funkciami, ktoré vykonávajú konkrétne úlohy, ako napríklad spracovanie udalostí alebo animácie. [2]

1.2 JavaScript Framework

Je to aplikačný framework napísaný v JavaScripte. Od JavaScriptovej knižnice sa odlišuje tak, že knižnica poskytuje funkcionalitu, ktorá je volaná z rodičovského programu, avšak framework definuje celý aplikačný dizajn. Poskytuje základné vzory pre tvorenie škálovateľných a udržiavateľných aplikácií. [2]

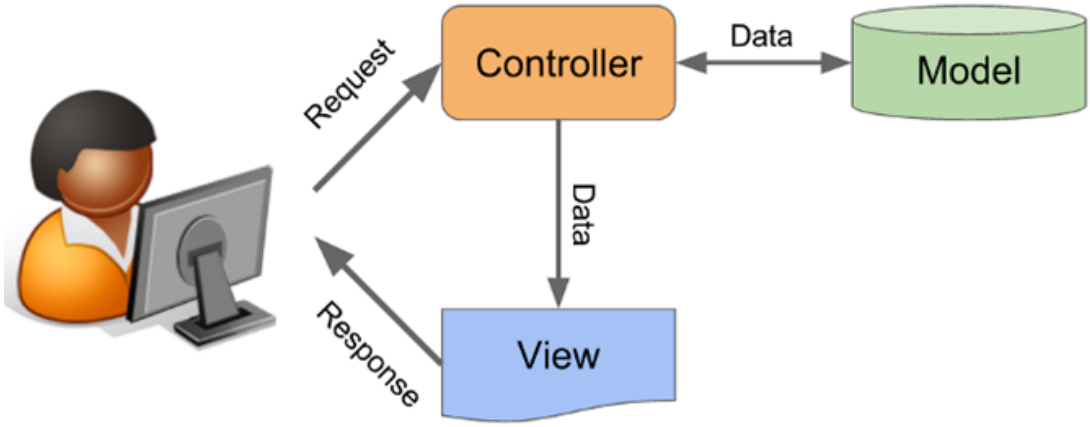
1.3 Návrhový vzor Model-View-Controller

Model-View-Controller je architektonický návrhový vzor, ktorý vznikol na desktopoch, ale súčasne sa najviac používa pri tvorbe webových aplikácií. Je súčasťou webových frameworkov, akými sú napríklad Nette, ASP .NET, Angular. Je určený najmä pre zložitejšie a komplexnejšie aplikácie, ktorých vývoj by bez tohto návrhového vzoru bol náročný. [3]

MVC rozdeľuje aplikáciu na tri hlavné komponenty a to sú:

- Model
- View
- Controller

Jednotlivé komponenty a ich prepojenie je zobrazené na obrázku (Obr. 1).



Obr. 1. Prepojenie jednotlivých komponentov architektúry MVC [4]

Každý z týchto komponentov je navrhnutý tak, aby spravoval špecifickú časť vyvíjanej aplikácie. Ide o oddelenie modelu dát a výstupu na obrazovke, ktorý spravuje časť view. Tieto dva komponenty prepája controller, ktorý obsahuje funkcie pre interakciu medzi nimi. Výhodou je lepšia čitateľnosť kódu aplikácie a oddelenie logiky od výstupu. [5]

1.3.1 Model

Model je komponent, ktorý zahŕňa všetku logiku spojenú s dátami, s ktorými používateľ pracuje. Môže to predstavovať buď údaje, ktoré sa prenášajú medzi komponentami view a controller alebo akékoľvek iné dáta spojené s business logikou aplikácie. Patria sem taktiež výpočty, požiadavky na databázu, validácia a podobne. Model nevie nič o výstupe a jeho funkcia spočíva len v prijatí údajov a ich následnom odoslaní von. Takisto nevie, odkiaľ dáta prišli a ani ako budú výstupné dáta sformátované. [3] [5]

Napríklad, objekt zákazníka by prijímal informácie o zákazníkovi z databázy, manipuloval by s nimi a mohol by ich upravené poslať späť do databázy alebo ich použiť na vykreslenie dát. Pri metóde objektovo-relačného programovania korešpondujú modely priamo s databázovými tabuľkami. [5]

1.3.2 View

View alebo aj pohľad, sa používa pre všetku logiku používateľského rozhrania a stará sa o zobrazenie výstupu používateľovi. Najčastejšie sa jedná o šablónu obsahujúcu HTML stránku a tagy značkovacieho jazyka, ktorý umožňuje do šablóny vkladať premenné, prípadne robiť iterácie, podmienky alebo prepájať premenné pre zobrazenie výstupu s jednotlivými dátami z modelu. Niektoré frameworky umožňujú prepojenie premenných

priamo s HTML elementami, akými sú radio buttony, checboxy a podobné prvky slúžiace pre výstup a vstup od používateľa. [3] [5]

Pohľadov aplikácia obsahuje mnoho, napríklad pohľad pre prihlásenie, pohľad pre registráciu, pohľad pre profil a tak ďalej. V tomto prípade je pohľad pre zobrazenie profilu spoločný pre všetkých používateľov a sú doň posielané rôzne dáta podľa toho, kto je zrovna prihlásený a koho profil zobrazujeme. Tieto dáta sú následne dosadené do HTML elementov šablóny. Šablóny je možné do seba vnárať a používať ich znovu, čím sa predchádza duplicitám. [3] [5]

View nie je len šablóna, ale taktiež zobrazovač výstupu. Obsahuje teda minimálne množstvo logiky, ktorá je pre výpis nutná. To zahŕňa validáciu vstupov od používateľa, kde sa môže kontrolovať dĺžka hesla alebo vyplnenie textu pred odoslaním. View podobne ako model nevie, odkiaľ mu dáta prišli, ale stará sa len o ich zobrazenie používateľovi. [3] [5]

Napríklad, view používateľa by obsahoval všetky UI komponenty, akými sú textové polia, menu a checkboxy určené pre interakciu s používateľom. [5]

1.3.3 Controller

Controller sa správa ako rozhranie medzi modelom a view. Spracováva všetku business logiku a prichádzajúce požiadavky, manipuluje dáta použitím modelu a spolupracuje s view pri vykresľovaní výstupu. Controller sa dá nazvať aj ako prostredník v tomto návrhovom vzore, pomocou ktorého komunikujú zvyšné dva komponenty a používateľ. Drží tak celý systém pohromade a komponenty prepojuje. V komplexnejších aplikáciách by mala mať každá entita svoj vlastný controller. [3] [5]

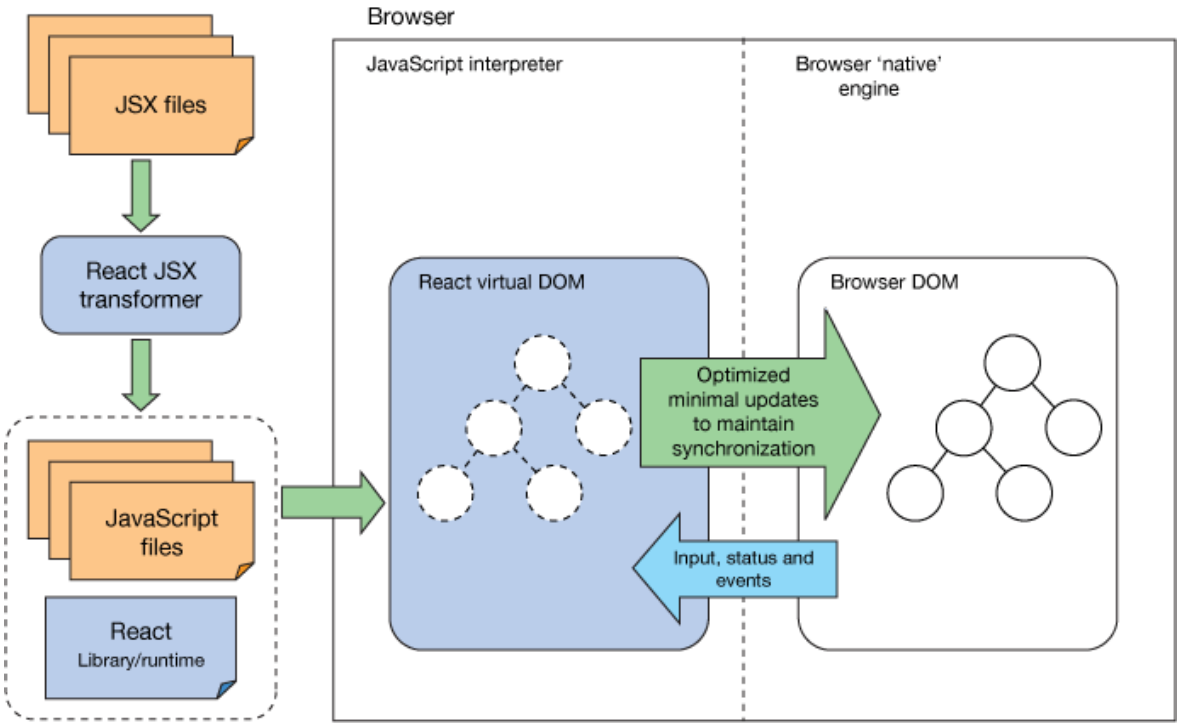
Napríklad, controller pre zákazníka by mal za úlohu všetky interakcie a vstupy z view používateľa a aktualizoval by databázu použitím používateľského modelu. Taktiež by bol použitý pri zobrazovaní dát o používateľovi. [5]

1.4 React

React je JavaScriptová knižnica slúžiaca pre tvorbu používateľského rozhrania webovej aplikácie. Táto knižnica umožňuje deklaratívnym spôsobom vytvoriť jednoduché pohľady - views pre každý stav aplikácie a následne aktualizovať a vykresliť daný komponent, keď nastane zmena dát. Každý komponent sa stará o správu jeho vlastného stavu a všetky komponenty tvoria komplexné používateľské rozhranie. Logika komponentov je písaná v

JavaScripte a je oddelená od HTML šablóny. Pohľady sú deklaratívne, čo robí kód aplikácie viac predvídateľným a jednoduchším na debugovanie. [6]

Framework používa technológiu Virtual DOM (VDOM), čo je vlastne koncept, kde je virtuálna reprezentácia používateľského rozhrania uložená v pamäti a následne je synchronizovaná s reálnym DOM. O túto koncepciu sa stará knižnica Reactu s názvom ReactDOM a je ju možné vidieť na obrázku (Obr. 2). [6]



Obr. 2. Koncept Virtual DOM v knižnici React [7]

React sa dá použiť aj v kombinácií s inými technológiami, čo umožňuje tvorbu nových funkcií bez potreby prepisovania celého kódu aplikácie. React taktiež podporuje vykresľovanie na servery za pomoci technológie Node.js. Vývoj knižnice React je podporovaný firmou Facebook. [6]

Hlavnými výhodami Reactu sú:

- Integrácia do existujúcich projektov
- Znovu použiteľný kód
- Rýchlosť a svižný rendering
- Virtuálny DOM
- Možnosť používať HTML tagy priamo v kóde

- Rozšírenie JSX

1.4.1 React JSX

React ponúka syntaxové rozšírenie JavaScriptu, ktoré sa nazýva JSX. Tento názov vznikol ako skratka zo spojenia slov JavaScript XML. JSX nemá definovanú sémantiku a umožňuje písať jednoduchý a elegantný kód podobný XML, ktorý sa následne transformuje do JavaScriptu. Je odporúčané používať ho s Reactom na opis toho, akoby malo vyzerať používateľské rozhranie aplikácie. Pripomína šablónový jazyk, ale má plnú podporu funkcií JavaScriptu. [6]

JSX produkuje React elementy, ktoré sa následne vykresľujú na DOM. Taktiež zaisťuje lepšiu čitateľnosť jednotlivých komponentov. Ukážka syntaxe JSX je zobrazená na obrázku (Obr. 3). [6]

Obr. 3. Ukážka syntaxe rozšírenia JSX [6]

1.5 Vue

Vue je progresívny JavaScriptový framework, ktorý slúži pre tvorbu používateľského rozhrania. Na rozdiel od iných frameworkov je Vue od základov navrhnutý tak, aby bol postupne prispôsobiteľný. Knižničné jadro tohto frameworku sa zameriava na vrstvu zobrazenia a je jednoduché sem integrovať iné knižnice alebo existujúce projekty. Vue je taktiež dobrý pre tvorbu sofistikovaných single-page aplikácií, keď je použitý v spojení s modernými nástrojmi a podpornými knižnicami. Podobne ako v Reacte, je možné jednotlivé časti aplikácia programovať nezávisle na sebe a teda v rámci už existujúcej

aplikácie nie je nutné programovať všetky stránky odznovu. Vue zaisťuje multiplatformnosť, pretože kód z pôvodnej webovej aplikácie je možné použiť pre vytvorenie aplikácie mobilnej. [8]

Vue tiež podporuje deklaratívne vykresľovanie dát do DOM objektov, využitím priamej šablónovej syntaxe, čo je možno vidieť na obrázku (Obr. 4). [8]

```HTML
<div id="app">
  {{ message }}
</div>
```

```JS
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

Obr. 4. Ukážka priamej šablónovej syntaxe vo Vue [8]

Tvorcovia Vue sa snažia reagovať na aktuálne požiadavky vývojárov a používateľov a podľa toho framework aktualizovať. Framework nie je na scéne tak dlho ako jeho konkurenti, čo môže predstavovať menšiu nevýhodu kvôli absencií niektorých funkcií.

Výhodami Vue sú:

- Možnosť vykresľovania na strane serveru
- Rozdelenie aplikácie na nezávislé časti
- Časté aktualizácie
- Uľahčené použitie animácií a prechodov
- Podpora natívneho vykresľovania
- Rýchlejšie reakcie vďaka virtuálnemu DOM

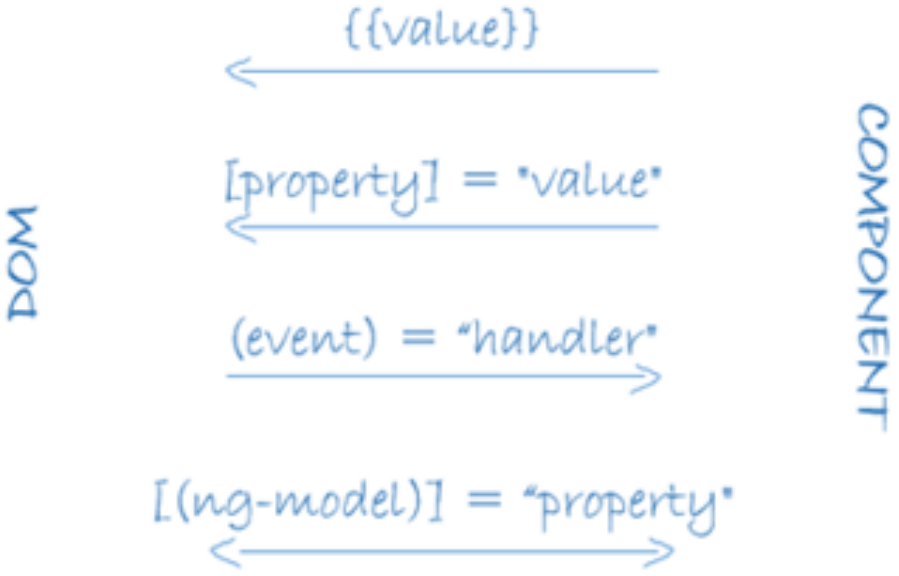
1.6 Angular

Angular je framework slúžiaci pre tvorbu single-page klientskych aplikácií. Pri tvorbe využíva HTML a TypeScript. Angular ponúka základnú a voliteľnú funkcionalitu v podobe

sady TypeScriptových knižníc, ktoré sa môžu použiť v aplikácií. Architektúra aplikácií vytvorených v Angular frameworku sa skladá z niekoľko konceptov. Základným stavebným blokom sú moduly *NgModules*, ktoré poskytujú kompilačný kontext pre komponenty. Tieto moduly tvoria potrebnú sadu funkcií pre aplikáciu. Každá aplikácia musí mať aspoň jeden koreňový modul, ktorý umožňuje samo-zavádzanie aplikácie. Komponenty definujú pohľady, ktoré predstavujú sadu zobrazovaných elementov a môžu ich modifikovať na základe programovej logiky a dát aplikácie. Komponenty taktiež používajú služby (services), čo sú objekty poskytujúce špeciálnu funkcionalitu. Funkcionalita služieb nie je priamo závislá na pohľadoch. Poskytovanie služieb pre komponent sa v Angular frameworku nazýva vstreknutie injekcie závislosti (dependency injection). [9]

Kód, ktorý je napísaný v tomto frameworku sa dá znovu použiť pre tvorbu natívnych aplikácií pre mobil aj desktop. Funkcie aplikácie sa dajú vytvárať jednoducho, použitím deklaratívnych šablón. Vytvorené komponenty je možné znovu použiť v iných aplikáciách a k dispozícií je aj široká škála už existujúcich komponentov. Zdieľanie medzi komponentom a šablónou HTML je zaručené vďaka takzvanému *two-way data bindingu*. [9]

Synchronizácia dát medzi aktuálnym stavom komponentu a jeho vykreslením na obrazovku ide oboma smermi, takže pri zmene stavu komponentu dôjde ku aktualizácií zobrazenia na obrazovke a naopak. Na nasledujúcom obrázku (Obr. 5), je možné vidieť štyri typy prepojenia komponentu so zobrazením. [9]

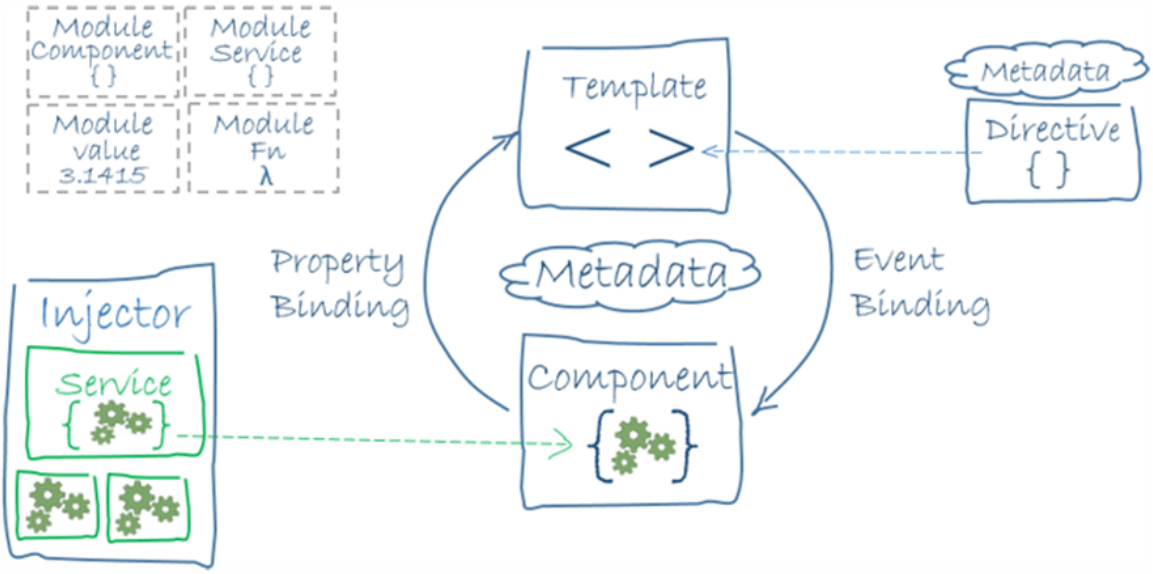


Obr. 5. Typy prepojení komponentu so zobrazením v Angulari [9]

Ďalej Angular poskytuje službu smerovania, ktorá umožňuje definovať navigačnú cestu medzi rôznymi stavmi aplikácie a hierarchiou pohľadov. Služba smerovania je postavená na známych konvenciách webových prehliadačov:

- Presmerovanie na stránku, ktorej URL adresa je zadaná v paneli s adresou.
- Navigácia na novú stránku po kliknutí na odkaz v aplikácií.
- Navigácia tlačidlami späť a dopredu umožňuje navigovať v histórií navštívených stránok. [9]

Dôležitou časťou Angular aplikácie sú aj metadáta. Tie informujú Angular o tom, kde získať hlavné stavebné bloky, ktoré sú potrebné na vytvorenie a prezentáciu komponentu a jeho pohľadu. Metadáta priraďujú šablónu ku komponentu, buď priamo vloženým kódom alebo pomocou referencie. Komponent a jeho šablóna tak spoločne popisujú pohľad (view). Štruktúru, jednotlivé časti a ich vzájomnú interakciu možno vidieť na obrázku (Obr. 6). [9]



Obr. 6. Časti aplikácie a ich prepojenie v Angulari [9]

Angular používa rozhranie príkazového riadku Angular CLI, ktoré je určené pre vytvorenie projektu a jeho komponentov, služieb, direktív a podobne.

Medzi výhody patria:

- Modulárnosť a znovu použitie komponentov
- Poskytovanie funkcionality cez dependency injection
- Nadstavba TypeScript

- Two-way data binding
- Reaktívne formuláre
- Router pre uľahčenie smerovania
- Angular CLI
- Rozšírenie RxJS

1.6.1 TypeScript

TypeScript je nadstavba (superset) jazyka JavaScript, ktorý ho rozširuje o typovanie, triedy, rozhrania a ďalšie veci týkajúce sa objektovo orientovaného programovania. TypeScript využíva *transpiler*, ktorý číta napísaný kód a prekladá ho do JavaScriptu. Taktiež poskytuje podporu pre všetky najnovšie funkcie JavaScriptu, ako napríklad asynchrónne funkcie. [10]

1.6.2 RxJS

RxJS (Reactive Extensions for JavaScript) je knižnica pre reaktívne programovanie, ktorá kladie dôraz na spracovanie asynchrónnych údajov s viacerými udalosťami. Taktiež poskytuje možnosť mapovať hodnoty do rôznych typov alebo iterovať hodnotami v streamoch. Základný prvok tejto knižnice je *observable*, ktorý podporuje posielanie správ medzi časťami aplikácie a tiež zaisťuje správu udalostí, asynchrónne programovanie a spracovanie viacnásobných hodnôt. [9]

1.7 Aurelia

Aurelia je moderný front-end framework určený pre tvorbu webových, mobilných a desktopových aplikácii. Menovaný framework je open source v licencií MIT a je postavený na otvorených webových štandardoch. Aurelia obsahuje kolekciu moderných, funkčne orientovaných, JavaScriptových modulov, ktoré spolu slúžia ako výkonná platforma pre tvorbu aplikácií. Do týchto modulov patria napríklad metadáta, poskytovanie služieb, smerovanie, šablónovanie alebo prepájanie funkčných častí so zobrazovanými dátami. [11]

Každý modul je napísaný pomocou JavaScriptu alebo TypeScriptu. Mnoho z týchto modulov môže byť použitých individuálne v ľubovoľnom projekte typu JavaScript, vrátane Node.js. V šablóne môže byť tok dát nastavený pomocou kľúčových slov:

- **Jedným smerom** – dáta sa prenášajú z modelu do pohľadu.
- **Dvoma smermi** – dáta sa prenášajú z modelu do pohľadu a späť.

- **Jeden raz** – dáta sa vykreslia na pohľade len raz a ďalej sa nesynchronizujú. [11]

Ďalej môžu byť dáta zobrazované pomocou iterácií poľa alebo priamym odkazom na premennú, čo možno vidieť aj na obrázku (Obr. 7). [11]

```
<template>
  <label repeat.for="color of colors">
    <input type="radio" name="clrs" value.bind="color" checked.bind="$parent.favoriteColor">
    ${color}
  </label>
</template>
```

Obr. 7. Ukážka kódu, pre zobrazovanie dát pomocou iterácií v Aurelia [11]

Zatiaľ čo moduly je možné využiť pre rôzne účely, ich najväčšia výhoda spočíva v ich spoločnom využití ako základ front-endovej časti aplikácie. Aurelia taktiež poskytuje bohatý model komponentov, dynamické zloženie používateľského rozhrania a komplexnú sadu funkcií a nástrojov pre tvorbu akýchkoľvek front-end aplikácií. [11]

Jadrom frameworku je vysoko výkonný, reaktívny systém, ktorý je schopný aktualizovať DOM veľmi rýchlo a efektívne. Taktiež konzistentnosť a škálovateľnosť výkonu nezávisia od toho, akú zložitosť má používateľské rozhranie aplikácie. Okrem tohto Aurelia zaisťuje aj množstvo ďalších rozšírení, napríklad pre internacionalizáciu, validáciu, modálne dialógy, virtualizáciu používateľského rozhrania a mnoho ďalších. Súčasťou frameworku je aj CLI, ktoré umožňuje generovanie a vytváranie nových projektov, doplnok pre webový prehliadač a doplnok pre program Visual Studio Code. [11]

Výhodou tohto frameworku je dostupnosť ukážok pre začiatočníkov ako aj voľne dostupná dokumentácia. Dokumentácia je prehľadná a rozdelená podľa jednotlivých modulov.

Hlavnými výhodami Aurelia sú:

- Rýchlosť vykresľovania a nízka spotreba pamäte
- Možnosť two-way bindingu
- Postavenie na štandardoch W3C
- Modularita a škálovanie aplikácií
- Komponenty sa dajú písať čistým JavaScriptom / TypeScriptom
- Jednoduchá integrácia s ostatnými knižnicami a frameworkami
- Podpora aj pre staršie prehliadače

2 NÁSTROJE VYUŽÍVAJÚCE GRAFICKÝ FORMÁT SVG

Scalable Vector Graphics (SVG) je značkovací jazyk z rodiny značkovacích jazykov XML (Extensible Markup Language), ktorý je primárne určený na opis dvojrozmernej, statickej alebo animovanej vektorovej grafiky. Vytvorilo ho konzorcium World Wide Web (W3C). Štandard SVG je kompatibilný so štandardmi HTML5 a inými W3C štandardmi, čo zabezpečuje jeho jednoduché začlenenie do webových aplikácií a pomáha zaistiť kompatibilitu. [12]

Pomocou základných geometrických útvarov, akými sú úsečka, priamka, krivka, bod je definovaná vektorová grafika. Tento štandard je základom pre všetky moderné webové aplikácie. Možnosti animácie a interaktivity zabezpečujú tomuto jazyku budúcnosť pri vývoji grafiky pre web. Keďže SVG je založené na XML, tak to znamená, že každý element je prístupný v rámci SVG DOM. Pre každý takýto element je možné naviazať udalosť (event) a jej správu v JavaScripte. V SVG je každý útvar braný ako objekt, preto keď sa zmenia jeho atribúty, tak webový prehliadač môže automaticky tento útvar prekresliť. [12]

SVG jazyk povoľuje tvorbu textu, tvarov vektorovej grafiky a rastrových obrázkov. Kombinuje spojenie grafiky a programovania pre správu webových obrázkov spôsobom, akým by to pri bitmapových obrázkoch nešlo. Vektorové obrázky vytvorené pomocou SVG sú založené na matematickej rovnici a koordinačnom systéme, vďaka čomu nie sú limitované veľkosťou pixelu ako bitmapové obrázky. Na ďalšom obrázku (Obr. 8) je možné vidieť príklad SVG útvaru napísaného v šablóne HTML. [13]

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
</svg>
```

Obr. 8. SVG grafický útvar, vytvorený v šablóne HTML [12]

Medzi výhody SVG patria:

- Open source licencia
- Oficiálny W3C grafický štandard, napísaný v XML
- Vytváranie obrázkov bez použitia textového editora

- Približovanie, odďaľovanie obrázkov
- Vytlačenie obrázkov vo vysokej kvalite pri rôznom rozlíšení
- Zmena veľkosti grafického komponentu bez straty kvality vzhľadu
- Možnosť jeho animácie
- Aplikovanie transformácií na obrázky
- Možnosť filtrov a masiek
- Podporuje cross-browser
- Dynamickosť a jednoduchá údržba

2.1 Dostupné JavaScriptové SVG knižnice

Tieto knižnice slúžia na prácu a manipuláciu s SVG grafikou. Poskytujú funkcie napísané v JavaScripte, ktoré umožňujú vytvárať grafické objekty a následne s nimi manipulovať, či už formou posunutia, otočenia, zmeny mierky a podobne. Niektoré knižnice poskytujú aj funkcie pre animácie a rôzne efekty, ktoré je možné aplikovať na vytvorenú grafiku. Hlavným dôvodom ich použitia je uľahčenie písania zdĺhavého kódu, keďže knižnice zaobaľujú komplexnú funkcionalitu do svojich funkcií, ktoré stačí zavolať nad útvarom. Každá z týchto knižníc sa zameriava na niečo iné a má svoje výhody aj nevýhody. [14]

Zoznam SVG knižníc pre tvorbu vektorovej grafiky, ktoré sú voľne dostupné a stiahnuteľné na internete:

- SVG.js
- Raphael.js
- Snap.svg
- D3.js
- P5.js
- Two.js

Popis osobitých knižníc SVG je obsiahnutý v nasledovnej časti.

2.1.1 SVG.js

SVG.js je malá knižnica na tvorbu, animovanie a manipuláciu SVG grafiky. Zahŕňa všetky špecifikácie SVG a nie je závislá na knižniciach tretých strán. Je jednou z najmenších knižníc spomedzi SVG, pritom poskytuje rovnakú funkcionalitu ako ostatné. [15]

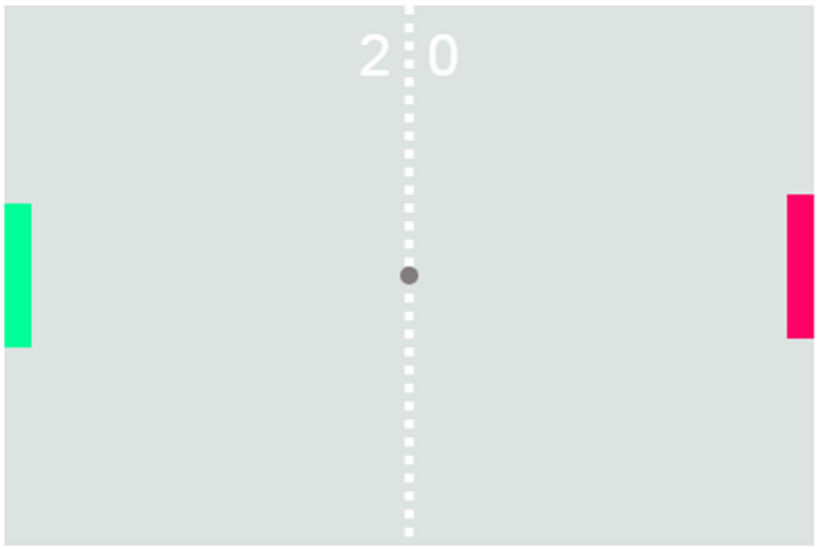
Jej výhodou je rýchle vykresľovanie grafických komponentov aj pri ich veľkom počte. Spomedzi konkurencie by mala byť jedna z najrýchlejších SVG grafických knižníc. [15]

Stručná a ľahko čitateľná syntax je jednou z najväčších výhod tejto knižnice. Syntax umožňuje jednoduché reťazenie príkazov pre prácu s grafickými komponentami. Veľký počet riadkov kódu v čistom JavaScripte dokáže knižnica nahradiť jedným zreťazeným riadkom. [15]

SVG.js tiež podporuje dynamické gradienty a priehľadnosť. Ďalej umožňuje animácie pri zmene veľkosti, pozície, transformácie a farby. Na jednotlivé grafické elementy sa dajú naviazať spúšťače udalostí. Grafické prvky je možné zoskupovať do skupín a text je možné ukotviť na cesty (paths). [15]

Užitočné rozšírenia a niekoľko príkladov použitia je k dispozícii na oficiálnej stránke tejto knižnice. Dokumentácia knižnice je štruktúrovaná a dobre vysvetlená, niekedy aj s príkladmi. Nevýhodou je potreba inštalovať rozšírenia pri potrebe dodatočnej funkcionality. Licencia knižnice je podmienená MIT licenciou. Autor je Wout Fierens. [15]

S pomocou SVG.js sa dajú vytvoriť rôzne kreácie a jedna z nich je zobrazená na obrázku (Obr. 9).



Obr. 9. Ukážka využitia SVG.js [15]

2.1.2 Raphael.js

Raphael je malá, voľne dostupná JavaScriptová knižnica. Pôvodne vznikla pre potrebu zjednotiť prehliadače používajúce vektorovú grafiku. Ako základ pre tvorbu grafiky využíva VML (Vector Markup Language) a SVG štandard od konzorcia World Wide Web. [16]

Raphael umožňuje jednoduchú prácu s vektorovou grafikou, umožňuje tvoriť špecifické grafy, výrezy obrázkov, otáčanie a iné transformácie útvarov. Každý grafický objekt vytvorený knižnicou Raphael je súčasne aj DOM objekt. [16]

Syntax je pomerne jednoduchá a podrobne popísaná v dokumentácií dostupnej na webovej stránke knižnice. Ku jednotlivým funkciám sú popísané aj parametre a ich návratové hodnoty. Občas sa pri popise vyskytne aj príklad pre použitie danej funkcie. [16]

Knižnica zaisťuje kompatibilitu s viacero webovými prehliadačmi. Veľkou výhodou je podpora starších prehliadačov. Aktuálne podporuje: Chrome 5.0+, Opera 9.5+, Firefox 3.0+, Internet Explorer 6.0+, Safari 3.0+. Nevýhodou sú málo časté aktualizácie knižnice, keďže autor začal vyvíjať obdobnú knižnicu Snap.svg. Autor knižnice je Dmitry Baranovskiy. [16]

Zavedenie knižnice na stránku HTML je pomerne jednoduché, stačí zahrnúť súbor *raphael.js* a použitie je uvedené na nasledujúcom obrázku (Obr. 10). [16]

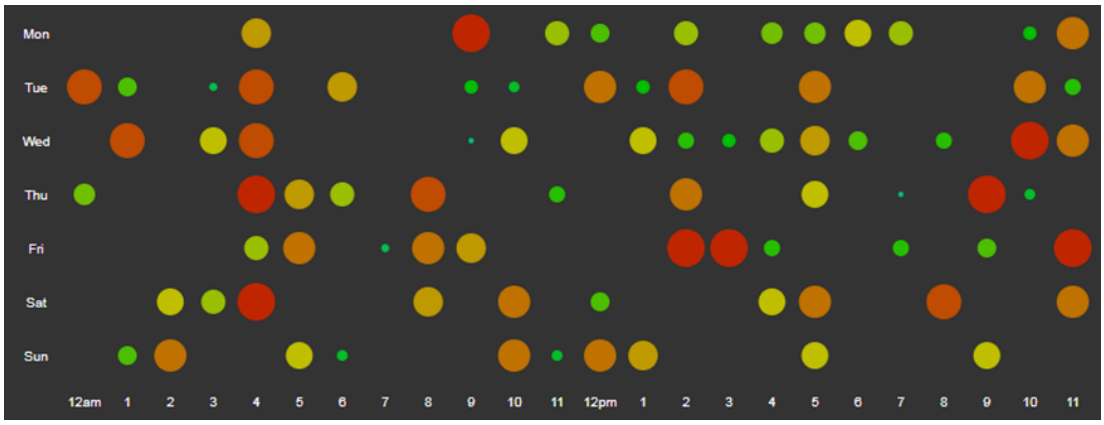
```
// Creates canvas 320 × 200 at 10, 50
var paper = Raphael(10, 50, 320, 200);

// Creates circle at x = 50, y = 40, with radius 10
var circle = paper.circle(50, 40, 10);
// Sets the fill attribute of the circle to red (#f00)
circle.attr("fill", "#f00");

// Sets the stroke attribute of the circle to white
circle.attr("stroke", "#fff");
```

Obr. 10. Použitie knižnice Raphael.js [16]

Názorná ukážka využitia knižnice pre vytvorenie grafiky je zobrazená na obrázku (Obr. 11).



Obr. 11. Ukážka použitia Raphael.js pre tvorbu grafiky [16]

2.1.3 Snap.svg

Snap.svg je JavaScriptová knižnica, ktorá umožňuje vytváranie interaktívnej vektorovej grafiky. Kvalita výslednej grafiky je zachovaná na displejoch s rôznou veľkosťou. Ďalej knižnica zaisťuje jednoduchú manipulácia s grafickými objektami pomocou svojich funkcií. [17]

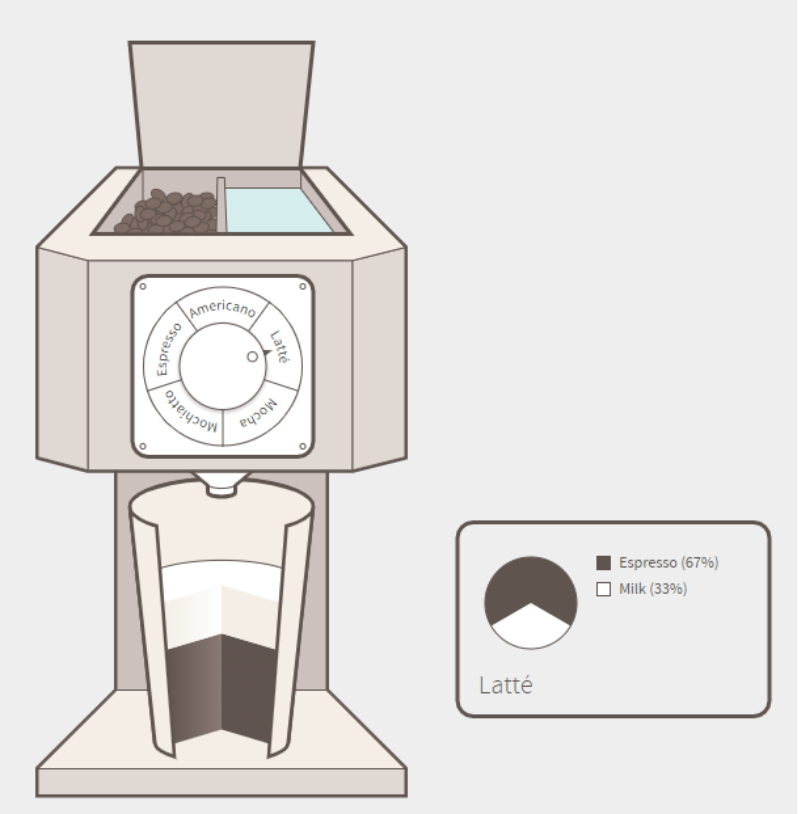
Na rozdiel od knižnice Raphael, Snap.svg je prispôsobená pre moderné prehliadače, a preto podporuje najnovšie funkcie SVG, akými sú orezávanie, maskovanie, vzorkovanie, plné gradienty, skupiny, transformácie a ďalšie. Taktiež umožňuje správu udalostí (eventov) a animácie. [17]

SVG grafiku je možné generovať priamo skrz Snap alebo cez vektorový nástroj ako napríklad Inkscape, Sketch alebo Adobe Illustrator a potom importovať do Snapu. [17]

Taktiež je možné jednotlivé reťazce obsahujúce SVG načítať asynchrónne a následne sa dotazovať, ktoré časti sú potrebné pre zostavenie grafického celku. [17]

Výhodou spomínanej knižnice je jednoduchá inštalácia a ľahko čitateľná syntax. Nevýhodou je nejasná dokumentácia, ktorá nie je kompletná a niektoré dôležité funkcie tam chýbajú. Knižnica je dostupná pod Apache 2 licenciou, čo znamená, že je open source a kompletne zadarmo. Autorom Snap.svg je Dmitry Baranovskiy a knižnica má podporu od firmy Adobe. [17]

Ukážku použitia tejto knižnice je možné vidieť na obrázku (Obr. 12).



Obr. 12. Použitie Snap.svg pri tvorbe SVG grafiky [17]

2.1.4 D3.js

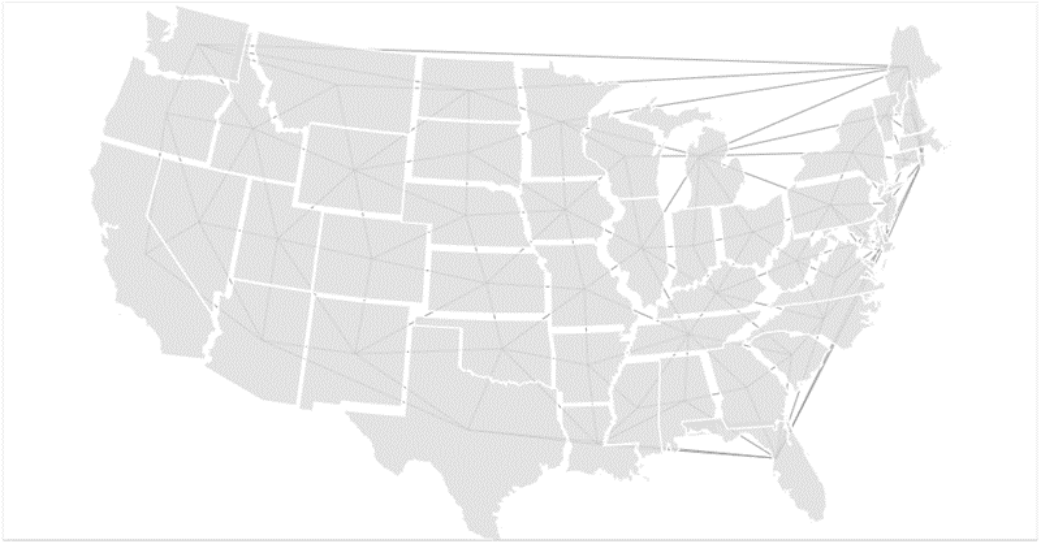
D3.js je ďalšia JavaScriptová knižnica určená na manipuláciu dokumentov, ktoré sú založené na údajoch, ako napríklad mapy, grafy a podobne. Knižnica D3.js pomáha oživiť údaje pomocou SVG, HTML a CSS. Ďalej si dáva záležať na webových štandardoch a poskytuje všetky funkcie novodobých webových prehliadačov bez nutnosti požívania špeciálnych štruktúr. Taktiež podporuje vizualizáciu dát a grafických komponentov. [18]

Využíva sa hlavne na generovanie interaktívnych tabuliek, máp alebo SVG grafov. D3.js je rýchla a zároveň podporuje zobrazovanie pre veľké množiny údajov. Používa

deklaratívny prístup, ktorý pracuje na ľubovoľných množinách uzlov, nazývaných selekcie. [18]

Knižnica ďalej umožňuje dynamické správanie pre animácie a interakciu. Umožňuje aj prepojenie ľubovoľných dát s DOM a následnú aplikáciu transformácií dokumentu založenú na údajoch. Napríklad ju možno použiť na vygenerovanie HTML tabuľky z poľa čísiel. Funkcionálny štýl D3.js dovoľuje opätovné použitie kódu prostredníctvom rozmanitej zbierky oficiálnych a komunitou vytvorených modulov. [18]

Na oficiálnej stránke tejto knižnice je možné nájsť veľké množstvo tutoriálov a rôznych príkladov jej použitia, čo je veľkou výhodou. Jedna z ukážok je zobrazená aj na obrázku (Obr. 13). [18]



Obr. 13. Príklad grafiky vytvorenej pomocou D3.js [18]

Dokumentácia je zrozumiteľná a podrobná a taktiež sa nachádza na webovej stránke knižnice. Knižnica je licencovaná ako BSD a jej autor je Mike Bostock. [18]

2.1.5 P5.js

Je to JavaScriptová knižnica, ktorá sa používa na kreatívne programovanie. Hlavným cieľom je uľahčiť začiatočníkom naučiť sa, ako programovať interaktívne grafické aplikácie a pomocou vizualizácie urobiť programovací jazyk príjemnejším pre používateľa. Bola navrhnutá pre umelcov, učiteľov, dizajnérov a taktiež začiatočníkov. [19]

Pre vykresľovanie grafických komponentov obsahuje plnú funkcionalitu. Používateľ môže použiť celú stránku ako skicu a nie je obmedzený plátnom. Vďaka tomu P5.js obsahuje

prídavné knižnice, ktoré uľahčujú prácu s ostatnými HTML5 objektami. Takýmito objektami sú text, video, webkamera, zvuk a ďalšie. [19]

Menšou nevýhodou tejto knižnice oproti predchádzajúcim je zložitejšia syntax. Výhodou je online editor, ktorý je dostupný na webovej stránke knižnice. P5.js je aktívne vyvíjaná vo verzií open source a jej autor je Lauren McCarthy. Pomocou tejto knižnice sa dajú vytvoriť rôzne grafické kreácie, ako aj príklad na obrázku (Obr. 14). [19]

Obr. 14. Vzor grafickej kreácie pomocou P5.js [19]

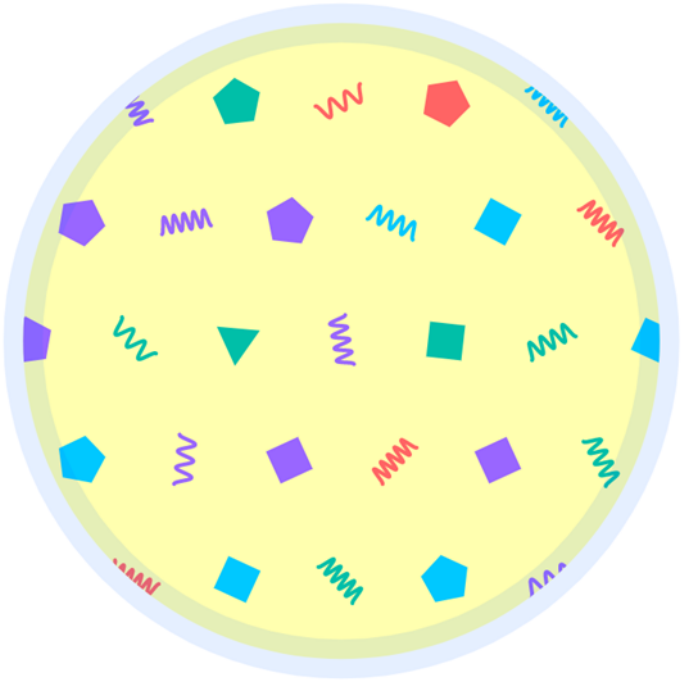
2.1.6 Two.js

Two.js je knižnica zameraná pre tvorbu dvojrozmernej SVG grafiky. Je hlboko inšpirovaná plochou dvojdimenzionálnou grafikou a jej pohybom. Výsledkom je, že two.js má za cieľ uľahčiť a zjednodušiť vytváranie a animovanie plochých tvarov. [20]

Podporované funkcie sú rotácia, animácia, stupnica, preklad a iné. Po vytvorení grafického objektu si ho knižnica uloží a zapamätá. Na takýto objekt je potom možné aplikovať ľubovoľný počet operácií, ako napríklad otočenie, translácia alebo zmena mierky. [20]

Veľmi užitočná funkcia je SVG interpreter, ktorý knižnica poskytuje a vďaka ktorému je možné grafické útvary nakresliť aj v komerčných aplikáciách ako je Adobe Illustrator a následne ich importovať do two.js scény. Výhodou je, že knižnica podporuje všetky moderné webové prehliadače. Nevýhodou je zložitejšie použitie v projekte a nie moc prehľadná dokumentácia na webovej stránke tvorcu. [20]

Ako príklad, čo je možné vytvoriť pomocou tejto knižnice je zobrazenie na obrázku (Obr. 15).



Obr. 15. Ukážka tvorby pomocou knižnice Two.js [20]

II. PRAKTICKÁ ČASŤ

3 NÁVRH APLIKÁCIE REZERVAČNÉHO SYSTÉMU A POROVNANIE SVG KNIŽNÍC

Táto kapitola obsahuje úvod ku online rezervačným systémom, návrh používateľského prostredia a samotnej aplikácie, ktorá by mala byť na záver publikovaná ako open source a tým umožniť komunite jednoduchšie spôsoby rezervácie pre rôzne podujatia, akými sú workshopy, prezentácie alebo divadelné predstavenia. Taktiež tu je popísaný návrh modulárnej štruktúry a možnosť znovu použitia aplikácie. Zhrnuté sú aj funkčné a nefunkčné požiadavky na aplikáciu. V závere kapitoly sú porovnané SVG knižnice podľa rôznych kritérií.

3.1 Online rezervačný systém

Online rezervačný systém je druh informačného systému, ktorého cieľom je evidovať rezervácie a dostupnosť nejakých komodít v reálnom čase. Takouto komoditou môže byť či už služba, výrobok alebo miesto, ako napríklad auto, vstupenka, izba, sedadlo v kine a tak ďalej. Rezervačný systém dodáva informácie o dostupnosti týchto komodít a zabraňuje konfliktom medzi používateľmi pri rezervácií rovnakej komodity. [21]

Online rezervačné systémy pomaly nahrádzajú papierové zápisníky, mobily, či rezervácie cez email. Sú prehľadnejšie, pohodlnejšie a ponúkajú jednoduchý výber požadovanej dostupnej komodity. Používanie rezervačných systémov eliminuje potrebu manuálneho zisťovania, či je daná komodita dostupná. Takéto zisťovanie veľakrát spôsobilo stratu zákazníka, z dôvodu opakovaného zisťovania dostupnosti komodity. Práve preto rezervačný systém umožňuje zákazníkovi vidieť dostupnosť všetkých voľných komodít a šetrí tak čas jemu aj správcovi. [22]

Online rezervačné systémy okrem zobrazovania dostupnosti a informácií o danej komodite, ponúkajú aj možnosť rezervácie danej komodity. Tento proces vykonáva aplikácia a nevyžaduje veľké úsilie zo strany používateľa. Celý proces je digitálny a tak by nemalo dôjsť ku dvojitému objednaniu komodity alebo ku ľudskému pochybeniu, ako to môže byť pri rezervovaní papierovou formou. [21]

Moderné rezervačné systémy umožňujú zákazníkom objednávať aj cez mobilné telefóny. Pracovná doba alebo preťažená pevná linka spoločnosti poskytujúcej komodity, už viac nehrajú rolu. Používatelia si môžu rezervovať komodity odkiaľkoľvek a kedykoľvek počas dňa. Zákazníci budú potešení aj z jednoduchého systému, kde uvidia všetky dostupné

funkcie na prvý pohľad. Takýto rezervačný systém je možné integrovať na webovú stránku spoločnosti, ktorá položky pre rezerváciu poskytuje. [21]

3.2 Používateľské rozhranie aplikácie

Používateľské rozhranie je dôležitá časť aplikácie, ktorú používateľ vidí, počuje a môže s ňou pracovať. Ostatné časti aplikácie sú pre používateľa skryté, ako napríklad databáza, v ktorej sú uložené dáta. [23]

Slúži ako spôsob, ktorým používateľ vykonáva úlohy v aplikácií a ako na to aplikácia reaguje. Používateľské rozhranie nie je len o tlačidlách a menu, ale dôležitá je najmä interakcia medzi používateľom a aplikáciou. Vo veľa prípadoch ide o interakciu viacerých používateľov s rovnakou aplikáciou. To znamená, že používateľské rozhranie by nemalo byť zamerané len na to, ako aplikácia vyzerá, ale aj ako aplikácia funguje. Nejde len o usporiadanie tlačidiel a výber farieb, ale predovšetkým záleží na výbere správnych prostriedkov a elementov pre danú úlohu. [23]

Tvorba používateľského rozhrania je proces prispôsobenia fyzickej reprezentácie rozhrania aplikácie, ktoré používateľ uvidí na obrazovke svojho elektronického zariadenia. Cieľom návrhu dizajnu používateľského rozhrania je komunikačný význam používateľa s aplikáciou. To sa dá dosiahnuť vytvorením vhodných vizuálov, ktoré najlepšie predstavujú to, čo aplikácia robí a ako s ňou možno pracovať. Používateľské rozhranie by malo byť intuitívne a pomáhať používateľovi pochopiť ako daná aplikácia funguje. Aby používateľ pri interakcii nestrácal veľa času a mal jednoduchý prístup ku všetkým funkciám, musí byť používateľské prostredie dobre zrozumiteľné a prehľadné. [23]

3.2.1 Návrh interaktívneho používateľského rozhrania

Používateľské rozhranie výslednej aplikácie by malo byť intuitívne a používateľovi by malo byť jasné, načo slúžia jednotlivé prvky. Dôležité je aj, aby bolo interaktívne a umožňovalo zreteľnú komunikáciu medzi používateľom a aplikáciou. Taktiež by malo byť zobrazované rovnako v rôznych webových prehliadačoch. Pri dizajne používateľského prostredia by mala byť použitá technológia ako Bootstrap alebo podobná.

3.3 Modulárnosť a znovu použitie aplikácie

Navrhovaná aplikácia rezervačného systému by mala byť prispôsobená tak, aby ju bolo možné znovu použiť v už existujúcom projekte. Aplikácia by preto nemala byť

naprogramovaná ako jeden komplexný celok, ale ako súbor komponentov, ktoré komunikujú cez vstupy a výstupy. Takéto rozdelenie do komponentov ponúka väčšina moderných front-end frameworkov pre tvorbu webových aplikácií. Niektoré z nich boli popísané aj v prvej kapitole tejto práce.

Takto modulárne navrhnutú aplikáciu je potom možné vložiť do existujúceho projektu, ako súbor komponentov a po nainštalovaní potrebných dodatočných závislostí, ktoré aplikácia vyžaduje, je možné ju používať v rámci inej existujúcej aplikácie.

Navrhovaná aplikácia by mala byť schopná komunikovať s existujúcou aplikáciou pomocou vstupov a výstupov. Vstupy by mali obsahovať len parametre, ktoré navrhovaná aplikácia nutne vyžaduje, ako napríklad údaje o prihlásenom používateľovi. Výstupy by zas mali ponúkať, len parametre a udalosti, ktoré obsahujú dôležité informácie pre existujúcu aplikáciu.

3.4 Požadovaná funkcionalita aplikácie

Navrhovaná aplikácia by mala predstavovať modul rezervačného systému, ktorý by umožňoval vytvárať podujatia a miesta na nich určené k rezervácií. Miesta by mali byť zobrazované ako SVG grafické útvary na plátne. Ako pozadie plátna by malo byť možné nastaviť ľubovoľný nahratý obrázok, ktorý by predstavoval mapu alebo schému daného podujatia. Rozhranie by malo byť interaktívne a poskytnúť napríklad rezerváciu sedadiel v kine.

Aplikácia rezervačného systému musí byť rozdelená do dvoch častí: prístup administrátora aplikácie a prístup koncového používateľa - zákazníka, ktorý si bude chcieť rezervovať miesto na podujatí.

Takéto rozdelenie umožní administrátorovi vytvárať podujatia (eventy), ktoré bude môcť spravovať. Administrátor by mal mať možnosť pridávať a odoberať nové miesta určené pre rezerváciu. Miesta by mali byť zobrazované ako grafické útvary, vytvorené pomocou zvolenej SVG knižnice. Takéto útvary by mohol pridať priamo na nahratú mapu podujatia. Ďalej by mal mať možnosť útvary rôzne pohybovať a umiestniť ich na ľubovoľné miesto v rámci plátna. Tiež by mal mať možnosť im nastaviť veľkosť a otočenie. Udalostiam by mal mať administrátor umožnené nastaviť dátum a čas, odkedy bude možné si rezervovať miesta na nich, prípadne dátum a čas dokedy.

Koncový používateľ by mal mať možnosť prezerať si detail jednotlivých podujatí a v nich zobrazenie jednotlivých miest. Mal by mať prístup ku detailom dostupných miest, ktoré by si prípadne mohol rezervovať.

Útvary predstavujúce miesta podujatia by mali byť farebne rozlíšené, podľa toho, či sú voľné, obsadené alebo si ich práve niekto prezerá. Koncovému používateľovi musí byť zamedzené otvoriť si detail obsadeného alebo prezeraného miesta. Ku detailom obsadených miest by mal mať prístup len administrátor.

3.4.1 Prístup a funkcie administrátora

Toto zobrazenie musí byť sprístupnené len používateľovi, ktorý je označený ako administrátor. Funkcionalita sprístupnená pre administrátora by mala byť nasledovná:

- **Vytvorenie nového podujatia** – umožní administrátorovi vytvoriť nové podujatie, s možnosťou nastaviť jeho názov, prípadne popis a rozmedzie dátumov pre rezervácie.
- **Nastavenie dátumu a času spustenia, prípadne ukončenia rezervácií** – zaistí administrátorovi nastaviť dátum a čas spustenia rezervácií pre dané podujatie, to znamená, že koncový zákazník bude mať prístup ku danému podujatiu a rezerváciám naň až od nastaveného dátumu a času. Taktiež by mohol byť nastavený dátum a čas ukončenia rezervácií.
- **Prehľad všetkých podujatí** – zabezpečí administrátorovi vidieť všetky podujatia a následne ich spravovať. Prehľad by mal byť stručný a obsahovať len dôležité informácie.
- **Zmazanie podujatia** – funkcia umožní zmazať vybraté podujatie.
- **Úprava podujatia** – táto funkcia zaistí administrátorovi upraviť o danom podujatí informácie akými sú: názov, popis a obdobie rezervácií pre zákazníkov.
- **Pridanie mapy podujatia** – administrátor by mal mať možnosť nahrať ľubovoľný obrázok cez používateľské rozhranie. Tento obrázok by slúžil ako pozadie pre plátno a predstavoval by mapu alebo schému daného podujatia.
- **Umiestnenie vytváraného miesta** – miesto, ktoré bude chcieť administrátor v podujatí vytvoriť, by malo byť možné umiestniť kdekoľvek na plátno.

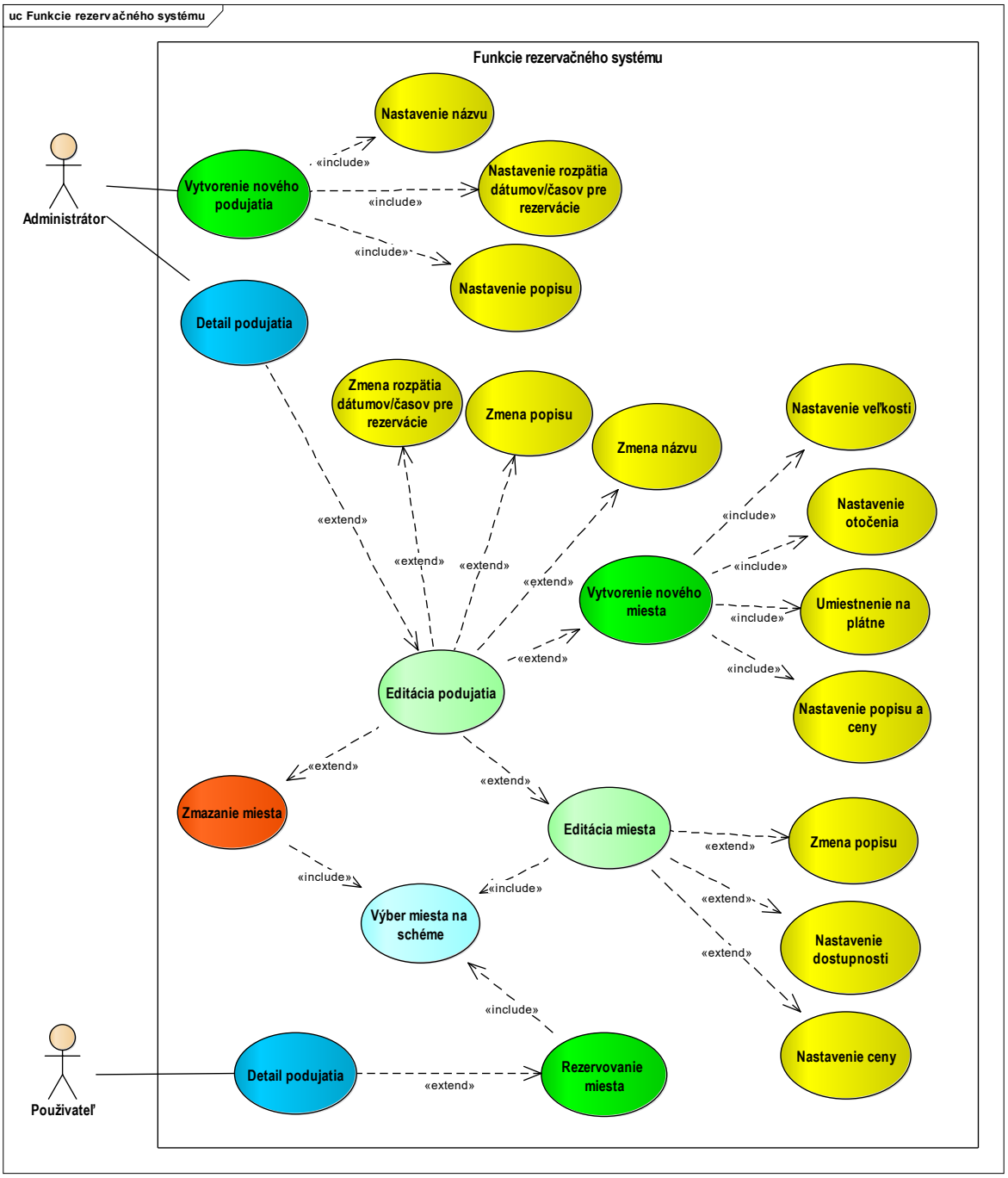
- **Nastavenie veľkosti a otočenia miesta** – vytváranému miestu by taktiež malo byť umožnené nastaviť veľkosť a otočenie.
- **Nastavenie informácií o vytváranom mieste** – administrátor by taktiež mal vedieť vytváranému miestu nastaviť základné informácie ako napríklad názov, cena a podobne.
- **Vytvorenie nového miesta v podujatí** – po nastavení umiestnenia, veľkosti a prípadného otočenia nového miesta by mal mať administrátor možnosť miesto potvrdiť a uložiť.
- **Úprava detailov miesta** – umožní administrátorovi upraviť informácie o vybratom mieste.
- **Zmena stavu obsadenosti miesta** – zabezpečí administrátorovi nastaviť konkrétne miesto ako obsadené, alebo ako znovu dostupné. Ku obsadeniu môže byť pridaná poznámka.
- **Zmazanie miesta** – zaistí zmazanie vybratého miesta.

3.4.2 Prístup koncového používateľa

Toto zobrazenie je viditeľné pre používateľa, ktorý si chce na danom podujatí rezervovať konkrétne miesto, prípadne viac miest. Všetky funkcie, ktoré má k dispozícii koncový používateľ sú taktiež prístupné aj pre administrátora. Koncový používateľ by mal mať k dispozícii nasledujúce funkcie:

- **Prehľad dostupných podujatí** – umožní používateľovi zobraziť všetky podujatia, ktoré sú dostupné pre rezervácie. Dostupnosť podujatia by mala byť určená podľa rozmedzia dátumov nastavených administrátorom.
- **Prehľad všetkých miest v podujatí** – zaistí používateľovi zobraziť všetky miesta, ktoré podujatie obsahuje.
- **Detail dostupného miesta** – zabezpečí používateľovi zobraziť detail miesta, ktoré nie je obsadené.
- **Rezervácia vybratého miesta** – umožní používateľovi rezervovať vybraté dostupné miesto.

Funkcie administrátora a používateľa sú zobrazené aj na diagrame prípadov použitia (Obr. 16).



Obr. 16. Diagram prípadov použitia – požadované funkcie rezervačného systému

Webová aplikácia by okrem vymenovaných funkčných požiadaviek mala spĺňať aj nasledovné nefunkčné požiadavky:

- **Rýchla odozva na zmeny** – navrhovaná aplikácia by mala poskytovať čo najrýchlejšiu odozvu na zmeny vykonávané používateľmi. Zmeny dát by mali byť používateľom zobrazované okamžite v reálnom čase.

- **Pozdržanie miesta pre používateľa** – používateľovi by malo byť na krátky čas napr. desať minút pozdržané miesto, ktoré si označí. Za tento čas bude mať možnosť rozmyslieť si, či si dané miesto chce rezervovať. Pre ostatných používateľov by toto konkrétne miesto malo byť bez možnosti si ho označiť, či rezervovať. Opisovaná funkcionalita by mala taktiež zabrániť duplicitným rezerváciám.
- **Farebné odlíšenie miest podľa vlastností** – miesta v podujatí by mali byť farebne odlíšené podľa ich dostupnosti, či sú obsadené alebo voľné. Taktiež by malo byť odlíšené, či si predmetné miesto práve niekto prezerá a má ho pozdržané.

3.5 Porovnanie jednotlivých SVG knižníc

Porovnanie jednotlivých JavaScriptových SVG knižníc bolo vykonané podľa viacerých kritérií a parametrov. Pomerne veľkú výhodu knižníc predstavovala jednoduchosť ich implementácie vo frameworku Angular, pretože bol vybratý ako technológia pre tvorbu front-endovej časti aplikácie.

Jednotlivé JavaScriptové SVG knižnice sú porovnávané podľa nasledovných vlastností:

- Primárne zameranie danej knižnice
- Implementácia v Angular frameworku
- Čitateľnosť a jednoduchosť syntaxe
- Prehľadnosť a kompletnosť dokumentácie
- Veľkosť GitHub komunity a obľúbenosť knižnice
- Aktuálnosť knižnice a počet problémov na GitHube
- Veľkosť npm balíka
- Počet stiahnutí knižnice za týždeň

V tabuľke (Tab. 1) je ukázané porovnanie jednotlivých knižníc podľa ich primárneho zamerania, pretože každá knižnica zahŕňa špecifickú funkcionalitu, ktorá predstavuje rôzne oblasti práce s SVG grafikou. Je tu porovnaná aj jednoduchosť implementácie vo frameworku Angular.

Tab. 1. Porovnanie SVG knižníc podľa zamerania a implementácie

SVG Knižnica	**Zameranie**	**Implementácia**
SVG.js	Animácie a manipulácia s vektorovou grafikou	Jednoduchá, s potrebou implementovať dodatočné rozšírenia
Raphael.js	Animácie grafiky	Mierne zložitá
Snap.svg	Pokročilé a komplexnejšie animácie	Jednoduchá
D3.js	Mapy, tabuľky, grafické zobrazenie údajov a dát	Jednoduchá s podrobným návodom
P5.js	Určená pre učiteľov, umelcov a dizajnérov	Veľmi zložitá
Two.js	Dvojrozmerná grafika a jej animovanie	Zložitá

V ďalšej tabuľke (Tab. 2) je zobrazené porovnanie vybratých knižníc na základe ich syntaxe pri programovaní. Porovnávaná je takisto dokumentácia, ktorá je u všetkých knižníc dostupná na ich oficiálnych webových stránkach.

Tab. 2. Porovnanie SVG knižníc podľa syntaxe a dokumentácie

SVG Knižnica	**Syntax**	**Dokumentácia**
SVG.js	Veľmi jednoduchá s možnosťou reťaziť príkazy	Prehľadná a detailná
Raphael.js	Jednoduchá	Nie moc prehľadná, ale zato podrobne popísaná
Snap.svg	Nekomplikovaná a jednoduchá	Nie moc prehľadná, popísaná detailne
D3.js	Zložitá a nevhodná pre začiatočníkov	Podrobná a prehľadná
P5.js	Podobná ako pri knižnici D3, kód je písaný do funkcií (draw, setup, ..)	Podrobná, ale neprehľadná
Two.js	Mierne zložitá, nie je možné reťaziť príkazy	Neprehľadná s absenciou príkladov a ukážok

V nasledovnej tabuľke (Tab. 3) sú knižnice porovnávané podľa veľkosti komunity a ich obľúbenosti na platforme GitHub. Veľkosť komunity je možné určovať podľa počtu kontribútorov, sledovateľov a podľa odberateľov, ktorý označili knižnicu symbolom hviezdy. Rovnako sú porovnávané aj podľa aktuálnosti, údržby knižnice a počtu problémov na GitHube. Problémy môžu na GitHube pridávať používatelia, ktorí narazia na chybu alebo problém s knižnicou.

Tab. 3. Porovnanie SVG knižníc podľa GitHub komunity a aktuálnosti

SVG Knižnica	**Komunita a obľúbenosť**	**Aktuálnosť a počet problémov**
SVG.js	77 kontribútorov, 271 sledovateľov a 8.1 tisíc odberateľov	Pravidelne aktualizovaná, 38 aktuálnych problémov
Raphael.js	40 kontribútorov, 405 sledovateľov a 10.7 tisíc odberateľov	Aktualizovaná zriedkavo, 304 aktuálnych problémov
Snap.svg	25 kontribútorov, 429 sledovateľov a 12.8 tisíc odberateľov	Posledná aktualizácia je z roku 2017, 241 aktuálnych problémov
D3.js	126 kontribútorov, 4 tisíc sledovateľov a 92.5 tisíc odberateľov	Aktualizovaná len zriedkavo, ale aktuálna verzia je stabilná s len 6 otvorenými problémami
P5.js	369 kontribútorov, 482 sledovateľov a 13.6 tisíc odberateľov	Aktualizovaná veľmi často, momentálne obsahuje viac ako 8000 potvrdených zmien. Počet aktuálnych problémov je 100
Two.js	32 kontribútorov, 159 sledovateľov a 6.7 tisíc odberateľov	Aktualizovaná pravidelne, 46 aktuálnych problémov

Pre posledné porovnanie boli knižnice hodnotené na základe veľkosti ich npm balíku a taktiež jeho počtu stiahnutí za týždeň. Toto porovnanie je možné vidieť v tabuľke (Tab. 4).

Tab. 4. Porovnanie SVG knižníc podľa veľkosti a počtu stiahnutí za týždeň

SVG Knižnica	**Veľkosť npm balíku**	**Počet stiahnutí za týždeň**
SVG.js	3.72 MB	6 716
Raphael.js	1.11 MB	112 528
Snap.svg	neuvedená	57 231
D3.js	924 kB	1 162 386
P5.js	5.9 MB	4 538
Two.js	1.55 MB	3 167

4 VÝBER VHODNÝCH NÁSTROJOV A TECHNOLÓGIÍ PRE IMPLEMENTÁCIU

Pre navrhnutú aplikáciu bolo potrebné vybrať vhodné nástroje a technológie na implementáciu. Technológie boli vybraté na základe cieľov a požadovanej funkcionality navrhnutej aplikácie. Vyberané boli moderné nástroje, ktoré sú aktívne využívané v oblasti vývoja webových aplikácií.

4.1 Framework Angular

Pre tvorbu front-endovej časti aplikácie spomedzi JavaScriptových frameworkov, bola vybratá technológia Angular. Tento framework bol vybratý práve kvôli jeho modulárnosti a možnosti znovu použitia komponentov. Taktiež je veľmi vhodná architektúra vytvorených aplikácií, ktorá je založená na vzore MVC. Model dát je oddelený od samotného pohľadu, zobrazovaného používateľovi. Veľkou výhodou je aj podpora TypeScriptu, ktorá sprehľadňuje kód aplikácie a umožňuje jednoduchú kontrolu typov.

4.2 NPM

NPM je správca balíkov pre programovací jazyk JavaScript. Je to predvolený správca balíkov pre runtime prostredie Node.js. Skladá sa z klienta príkazového riadku a online databázy verejných balíkov. Je možné ho použiť v prostredí frameworku Angular, kde sa npm balíky taktiež zapisujú ako externé závislosti do súboru: *package.json*. [24]

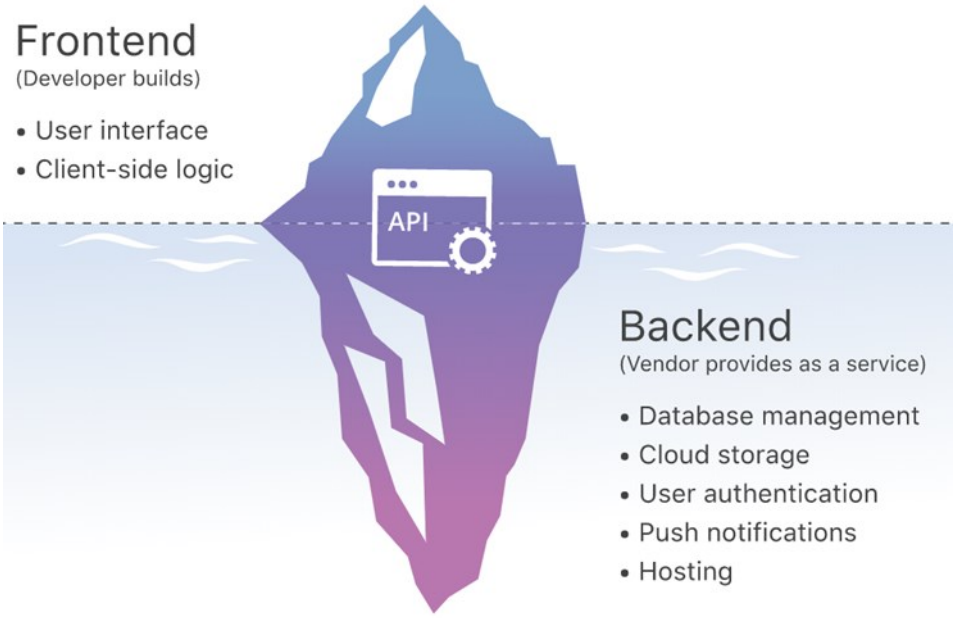
4.3 JavaScriptová knižnica SVG.js

Na základe porovnania bola pre tvorbu interaktívnej vektorovej grafiky zvolená knižnica SVG.js. Najdôležitejším aspektom výberu bolo práve zameranie knižnice, čo je tvorba a manipulácia s SVG objektami. Knižnica taktiež poskytuje jednoduchú implementáciu do prostredia frameworku Angular cez npm balík. Veľkou výhodou je jej jednoduchá syntax s možnosťou príkazy reťaziť. Aj keď knižnica nie je tak populárna ako niektoré konkurenčné knižnice, pracuje na nej veľký počet kontribútorov. Knižnica je pravidelne aktualizovaná a problémy od používateľov sú priebežne riešené.

4.4 Firebase

Firebase je platforma pre vývoj mobilných a webových aplikácií označovaná aj ako Backend-as-a-Service (BaaS). Backend-as-a-Service je model cloudovej služby, ktorá

vývojárom umožňuje zamerať sa na vývoj front-endovej časti aplikácie a využívať back-endové služby bez nutnosti vytvárať ich a udržiavať ich. Tento model je zobrazený aj na obrázku (Obr. 17). [25] [26]



Obr. 17. Zobrazenie modelu BaaS [26]

Firebase je postavený na infraštruktúre spoločnosti Google a dá sa rôzne škálovať. Taktiež ponúka rôzne produkty ako napríklad analýzy a databázy. Tieto produkty majú back-endové komponenty, ktoré sú prevádzkované a udržiavané spoločnosťou Google. Pre nastavenie a interakciu s týmito produktami a službami slúžia klientské knižnice označované Client SDK. Tieto knižnice poskytujú spojenie so službami priamo bez nutnosti nejakého prídavného spojenia medzi aplikáciou a službou. [25]

Knižnice pre správu Firebase služieb a produktov sa inštalujú do projektu pomocou npm balíku. Pre implementovanú aplikáciu bola použitá databáza Cloud Firestore.

4.4.1 Cloud Firestore

Cloud Firestore je dokumentová NoSQL databáza, ktorá umožňuje ukladať, synchronizovať a dotazovať dáta pre webové aplikácie. Funguje na princípe kolekcií, v ktorých sú uložené dokumenty. Kolekcia môže obsahovať dokumenty a nemôže priamo obsahovať ďalšie kolekcie. Hierarchia musí byť zachovaná a to tak, že kolekcia musí byť vytvorená vo vnútri dokumentu. Funkcie pre prácu s Cloud Firestore sú obsiahnuté v nástrojovom balíku, ktorý je možné do projektu nainštalovať cez spomínaný npm balík. Vďaka týmto funkciám je

možné do databázy pristupovať bez nutnosti tvorby vlastného serveru, čo je možné vidieť aj na obrázku (Obr. 18). [25]

Obr. 18. Prístup ku produktom a službám Firebase cez Client SDK [25]

Cloud Firestore zaisťuje automatickú synchronizáciu údajov aplikácie medzi zariadeniami používateľov. Oznamuje zmeny dát hneď, ako k nim dôjde, čo umožňuje spoluprácu medzi zariadeniami v reálnom čase. [25]

4.5 Bootstrap

Bootstrap je jedna z najpopulárnejších HTML, CSS a JavaScriptových knižníc určená pre tvorbu responzívnych mobile-first webových aplikácií. Uľahčuje dizajnovanie a prispôsobenie front-endovej časti webovej aplikácie. Takisto poskytuje použitie premenných, responzívny, mriežkový systém usporiadania elementov, preddefinované komponenty a výkonné JavaScriptové doplnky. Zaistená je taktiež kompatibilita medzi rôznymi prehliadačmi a zariadeniami. [27]

Bootstrap poskytuje aj svoju vlastnú knižnicu pre SVG ikony, ktoré najlepšie fungujú s Bootstrap komponentami. Keďže ikony sú SVG, dajú sa ľahko a rýchlo škálovať, môžu byť implementované niekoľkými spôsobmi a je možné ich upravovať pomocou CSS. [27]

Táto open source knižnica je voľne dostupná pod licenciou MIT a do projektu je možné ju nainštalovať ako npm balík.

4.6 Font Awesome

Font awesome je sada vektorových ikon a log založených na CSS. Táto sada je open source a je možné ju do projektu nainštalovať ako npm balík. Pre zobrazenie ikony v aplikácií je nutné vložiť HTML tag s triedou, ktorá označuje príslušnú ikonu, napríklad:

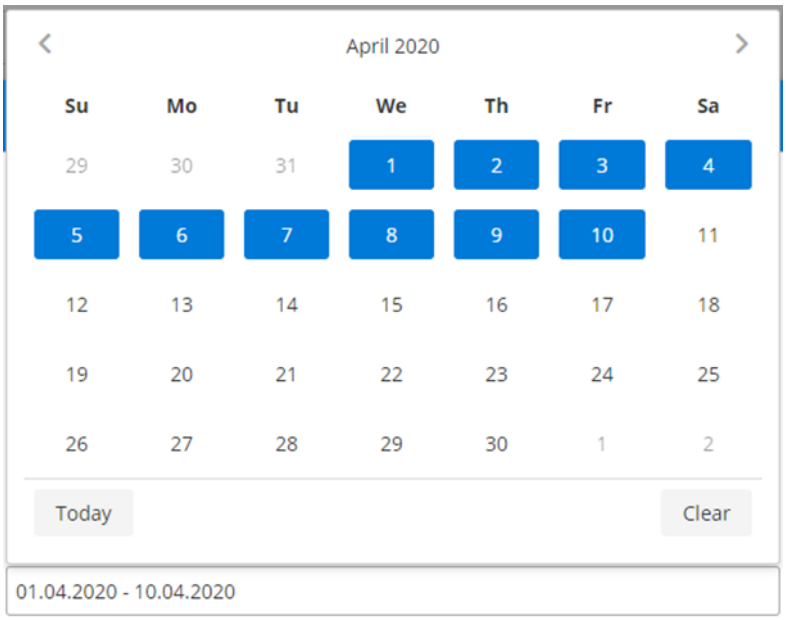
** [28]

4.7 PrimeNG

PrimeNG je kolekcia bohatých komponentov používateľského rozhrania pre Angular, ktorá uľahčuje tvorbu používateľského rozhrania. Na výber je viac ako 80 ľahko použiteľných komponentov, ktoré spĺňajú všetky požiadavky na používateľské rozhranie. Taktiež je možné vybrať si z množstva tém, ktoré ovplyvňujú vzhľad jednotlivých komponentov. [29]

Všetky widgety sú open source a je možné bezplatne ich používať na základe licencie MIT. PrimeNG je vyvíjaný spoločnosťou PrimeTek Informatics, predajcom s dlhoročnými skúsenosťami vo vývoji open source riešení pre používateľské rozhranie. Do aplikácie je možné nainštalovať ho pomocou balíku npm. [29]

V implementovanej aplikácií boli použité PrimeNG komponenty ako napríklad dialógové okno alebo kalendár, ktorý je možno vidieť aj na obrázku (Obr. 19).



Obr. 19. Komponent PrimeNG kalendára

PrimeNG pre svoje komponenty s grafmi používa JavaScriptovú knižnicu Charts.js.

4.8 Charts.js

Charts.js je JavaScriptová open source HTML5 knižnica určená na prácu s grafmi. Slúži pre zobrazenie dát aplikácie na preddefinovaných grafoch. Na výber je viacero typov grafov, ako napríklad stĺpcové, čiarové, koláčové a podobne. Grafy sú taktiež interaktívne a animované. [30]

Táto knižnica bola použitá pre vytvorenie koláčového grafu, ktorý zobrazuje počet dostupných a obsadených miest daného podujatia. Toto grafické zobrazenie je možné vidieť aj na obrázku (Obr. 20).

Obr. 20. Dostupnosť miest pomocou Charts.js

5 IMPLEMENTÁCIA APLIKÁCIE

Kapitola popisuje súborovú štruktúru projektu, hierarchiu komponentov, dátový model a taktiež spôsob uloženia dát v databáze. V kapitole je tiež popísaná implementácia navrhnutej aplikácie na základe funkčných a nefunkčných požiadaviek. Je tu zahrnutá aj publikácia aplikácie na platforme GitHub.

5.1 Súborová štruktúra projektu

Štruktúra súborov sa odvíja od frameworku Angular, pomocou ktorého je aplikácia rezervačného systému programovaná. Projekt a jeho súčasti sú generované pomocou príkazového riadku Angular CLI. Po vygenerovaní projektu sa vygeneruje viacero súborov a adresárov. Koreňový adresár, v ktorom sa nachádzajú hlavné súbory projektu sa nazýva *src*. Ďalej sa tu nachádza adresár *node_modules*, ktorý obsahuje knižnice a nástroje stiahnuté cez npm ako balíky. Tento adresár sa kvôli jeho veľkosti a hĺbke neodporúča presúvať, namiesto toho sa dá ku aplikácií nainštalovať cez príkazový riadok pomocou príkazu: *npm install*.

Adresár *src* obsahuje nasledujúce dôležité adresáre:

- **assets** – tu sa nachádzajú externé obrázky, dokumenty a podobne.
- **enviroments** – obsahuje konfiguračné súbory pre aplikáciu, ako napríklad prihlasovacie údaje pre spojenie s platformou Firebase.
- **app** – tu sú uložené hlavné súbory aplikácie:
    - komponenty – stavebné bloky aplikácie, spojenie všetkých komponentov tvorí celok aplikácie. V komponentoch sa nachádzajú tri typy súborov a to:
        - šablóna – popisuje zobrazenie, ktoré vidí používateľ na obrazovke. Šablóna je typu HTML.
        - logika komponentu – v tomto súbore sú napísané funkcie pre prácu s komponentom. Prepája dáta modelu so zobrazením. Tento súbor je typu TypeScript.
        - štýly komponentu – súbor typu CSS, ktorý obsahuje štýly pre šablónu daného komponentu.

- model_classes – obsahuje modelové triedy, ktoré popisujú dátový model aplikácie, ako napríklad model podujatia alebo model miesta. Súbory sú typu TypeScript.
- services – nachádzajú sa tú súbory, pre triedy služieb. Služby poskytujú funkcie pre externé časti aplikácie, akými je aj databáza Cloud Firestore.

V adresári *src* sa nachádza aj koreňový modul aplikácie *app.module.ts*, do ktorého sa vkladajú moduly pre jednotlivé časti aplikácie. Dôležité moduly v implementovanej aplikácií sú: modul pre formuláre, modul pre databázu Cloud Firestore a moduly pre komponenty knižnice PrimeNG. Dôležitý súbor projektu je aj: *package.json*. Zapisujú sa doň externé závislosti pre aplikáciu, ktoré je možné pri migrácii doinštalovať. Za zmienku stojí aj súbor: *angular.json*, do ktorého sa zapisujú odkazy na štýly a skripty externých knižníc.

Súborovú štruktúru projektu je možné vidieť aj na obrázku (Obr. 21).

Obr. 21. Súborová štruktúra projektu

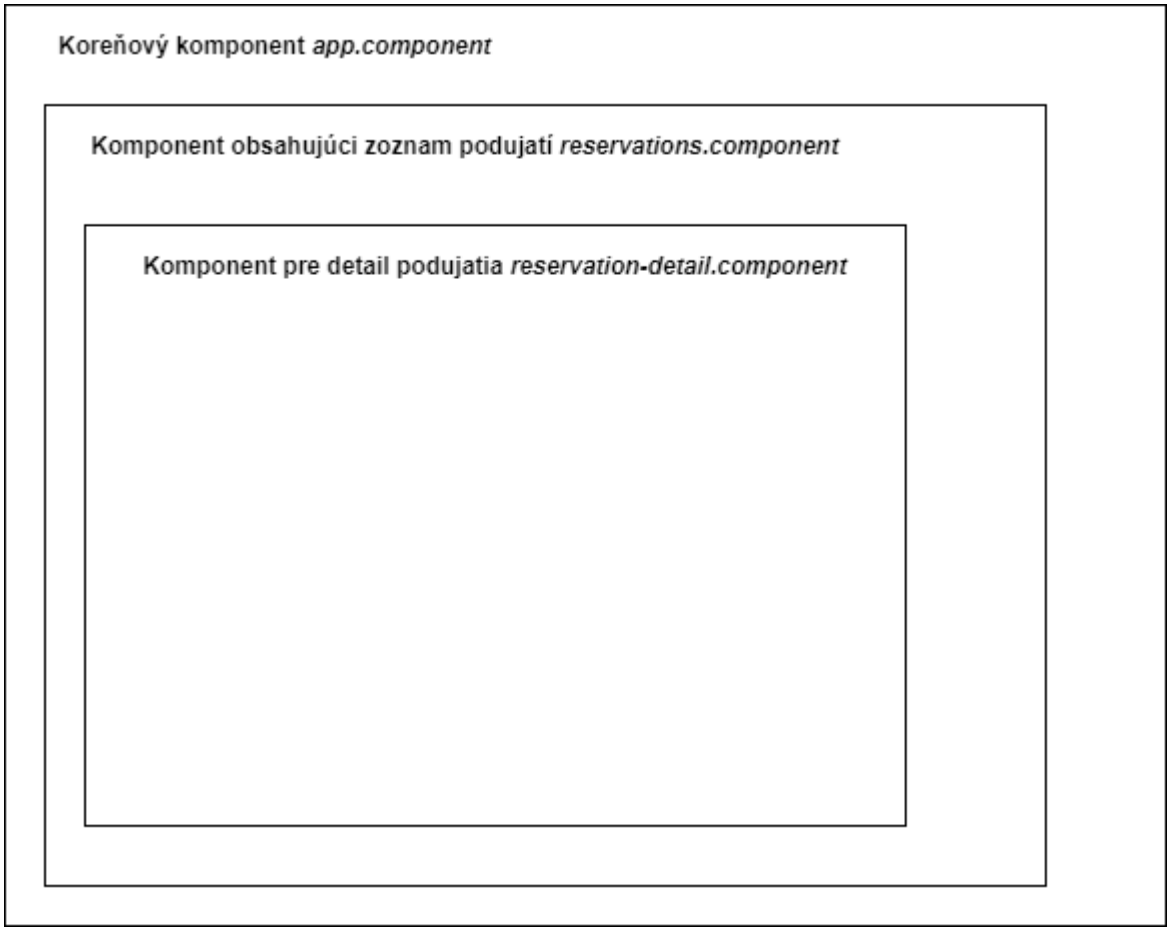
5.2 Hierarchia komponentov

Ako už bolo spomenuté v predchádzajúcej podkapitole, komponenty sú hlavné stavebné bloky Angular aplikácie. Výhodou komponentov je práve ich modularita a možnosť jednoducho ich použiť znovu aj v iných existujúcich aplikáciách. Komponenty sa v aplikácií používajú vložením do HTML šablóny ako tag, zvyčajne *<app-nazov-komponentu></app-nazov-komponentu>*. Komponenty sa tiež dajú vnárať jeden do druhého, pričom vznikne vzťah *parent – child* a takéto komponenty medzi sebou komunikujú pomocou vstupov a výstupov. Každá aplikácia obsahuje koreňový komponent, do ktorého je možné vkladať ďalšie komponenty.

Výsledná aplikácia okrem koreňového komponentu obsahuje ďalšie dva komponenty a to:

- **komponent *reservations*** – komponent obsahuje prehľadný zoznam podujatí, na ktoré je možné si rezervovať miesto. Administrátor navyše vidí funkciu pre zmazanie konkrétneho podujatia a funkciu pre pridanie nového podujatia. Administrátor vidí tiež, či je podujatie dostupné pre používateľov a to pomocou ikony statusu. Po kliknutí na názov podujatia sa zobrazí komponent detailu podujatia.
- **komponent *reservation-detail*** – je vnorený do komponentu *reservations* a predstavuje detail vybratého podujatia. Detail obsahuje informácie o danom podujatí, ktoré môže upravovať len administrátor. Ďalej sa tu nachádza plátno pre vektorovú grafiku, na ktorom sú umiestnené miesta, určené pre rezerváciu. Administrátor má umožnené meniť obrázok mapy podujatia a vytvárať nové miesta. Pod plátnom sa nachádza legenda, ktorá vysvetľuje farebné označenie miest.

Hierarchiu komponentov je možné vidieť aj na obrázku (Obr. 22).



Obr. 22. Hierarchia komponentov aplikácie

5.3 Dátový model aplikácie

Dátový model popisuje formát a štruktúru dát aplikácie a taktiež popisuje vzájomné vzťahy medzi jednotlivými dátovými prvkami. Dátový model sa snaží vytvoriť abstrakciu reálnych objektov. Pri dátovom modeli je dôležité zamerať sa na časť reality, o ktorej chceme uchovávať informácie. Jednotlivé informácie o objektoch sa zapisujú ako ich atribúty, ktoré môžu byť rôznych typov.

Pre popis dátového modelu výslednej aplikácie sú použité nasledujúce triedy:

- **reservation-scheme** – predstavuje podujatie alebo predstavenie, na ktorom si bude môcť používateľ rezervovať svoje miesto. Obsahuje atribúty:
    - key – reťazec, ktorý slúži ako identifikátor daného podujatia.
    - name – reťazec, meno podujatia.
    - mapKey – reťazec, identifikátor obrázku mapy daného podujatia.
    - createdBy – reťazec, identifikátor používateľa, ktorý podujatie vytvoril.
    - shapes – pole, obsahuje miesta daného podujatia.
    - info – reťazec, informácie o podujatí.
    - reservationsStart – reťazec, dátum a čas začiatku spustenia rezervácií.
    - reservationsEnd – reťazec, dátum a čas konca možnosti rezervácií.
- **shape** – predstavuje miesto v podujatí, ktoré je predmetom rezervácie. Obsahuje nasledujúce atribúty:
    - key – reťazec, identifikátor daného miesta.
    - x – číslo, popisuje umiestnenie na osi x plátna.
    - y – číslo, popisuje umiestnenie na osi y plátna.
    - width – číslo, určuje šírku útvaru predstavujúceho miesto.
    - height – číslo, výška útvaru predstavujúceho miesto.
    - transform – reťazec, obsahuje transformačnú maticu útvaru predstavujúceho miesto. Matica obsahuje údaje potrebné ku otočeniu miesta.
    - info – reťazec, popis miesta.

- available – boolean, určuje dostupnosť miesta, či je rezervované alebo voľné.
- onHoldBy – reťazec, obsahuje identifikačný kľúč toho, kto si práve prehliada dané miesto.
- onHoldTill – reťazec, obsahuje dátum a čas dokedy si môže používateľ prehliadať aktuálne vybraté miesto.
- reservedBy – reťazec, obsahuje identifikačný kľúč používateľa, ktorý si miesto rezervoval.
- reservationInfo – reťazec, informácie ku rezervácií daného miesta. Napríklad meno a adresa používateľa, ktorý si miesto rezervoval.
- price – číslo, cena za miesto.

- **map-image** – stvárňuje obrázok mapy podujatia. S podujatím sa prepája pomocou identifikačného kľúča. Atribúty, ktoré obsahuje sú nasledovné:
  - key – reťazec, identifikačný kľúč obrázku mapy.
  - name – reťazec, názov obrázku spolu s príponou.
  - storagePath – reťazec, cesta, popisujúca umiestnenie obrázku vo Firebase úložisku.
  - url – reťazec, odkaz na daný obrázok mapy dostupný online.

5.4 Uloženie dát v databáze a manipulácia s nimi

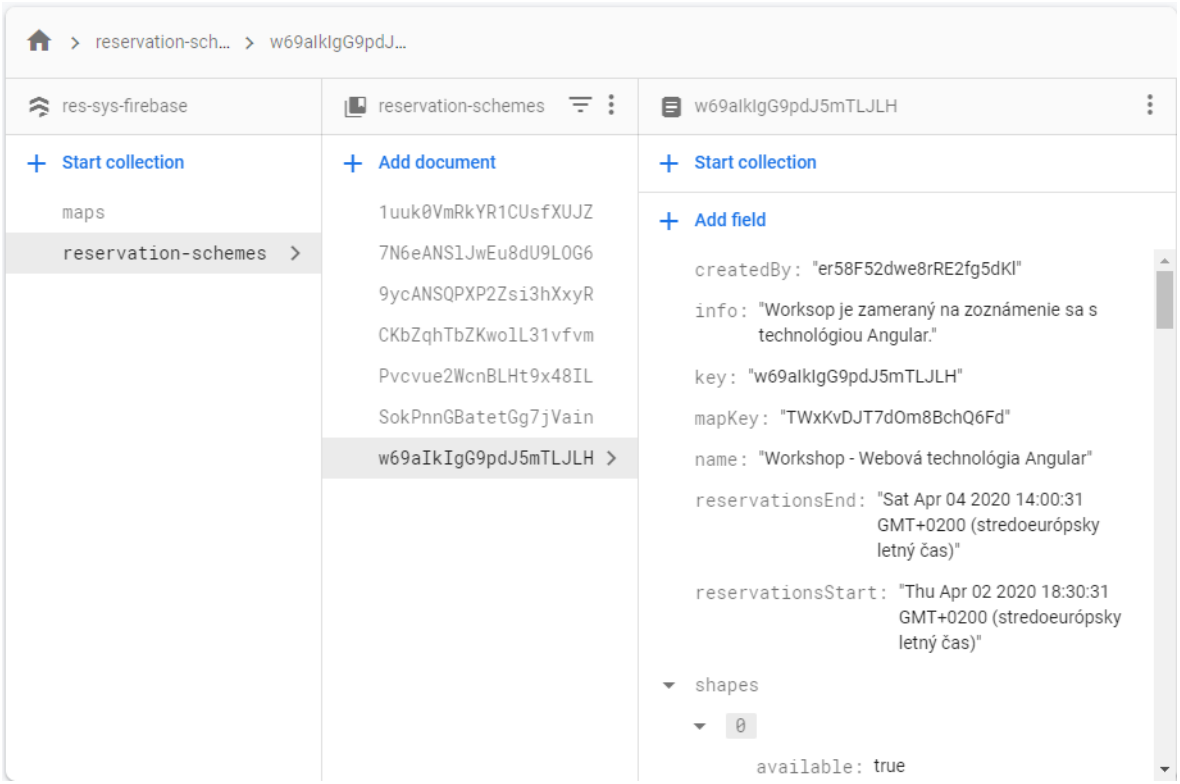
Výsledná aplikácia rezervačného systému by mala spĺňať podmienky modulárnosti a znovu použiteľnosti, preto je navrhnutá tak, aby bolo možné na stranu backendu a databázy použiť ľubovoľnú technológiu. To zabezpečujú služby (services): *reservation-scheme-service.ts* a *file-service.ts*. Metódy služieb aktuálne obsahujú funkcie pre interakciu s databázou Cloud Firestore. Tieto metódy je však možné v prípade použitia inej backendovej technológie prepísať. Zachované musia byť však vstupné parametre a návratové hodnoty metód.

V databáze Cloud Firebase sú dáta uložené ako dve kolekcie a to:

- reservation-schemes
- maps

Kolekcia *reservation-schemes* obsahuje dokumenty predstavujúce jednotlivé podujatia. V prípade, že podujatie má vytvorené nejaké miesta, sú tieto miesta uložené v dokumente podujatia ako pole miest. Druhá kolekcia *maps* obsahuje dokumenty popisujúce jednotlivé obrázky máp pre podujatia. Prepojenie podujatí a obrázkov máp je zabezpečené identifikačným kľúčom, ktorý vlastní každý obrázok mapy.

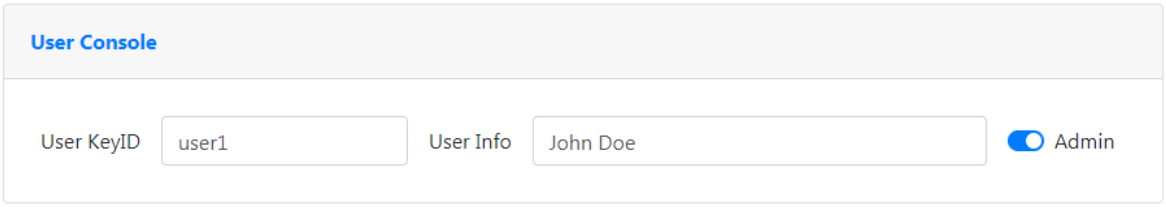
Spomínané uloženie dát v databáze je možné vidieť aj na obrázku (Obr. 23).



Obr. 23. Uloženie dát aplikácie v databáze Cloud Firestore

5.5 Implementácia požadovaných funkcií

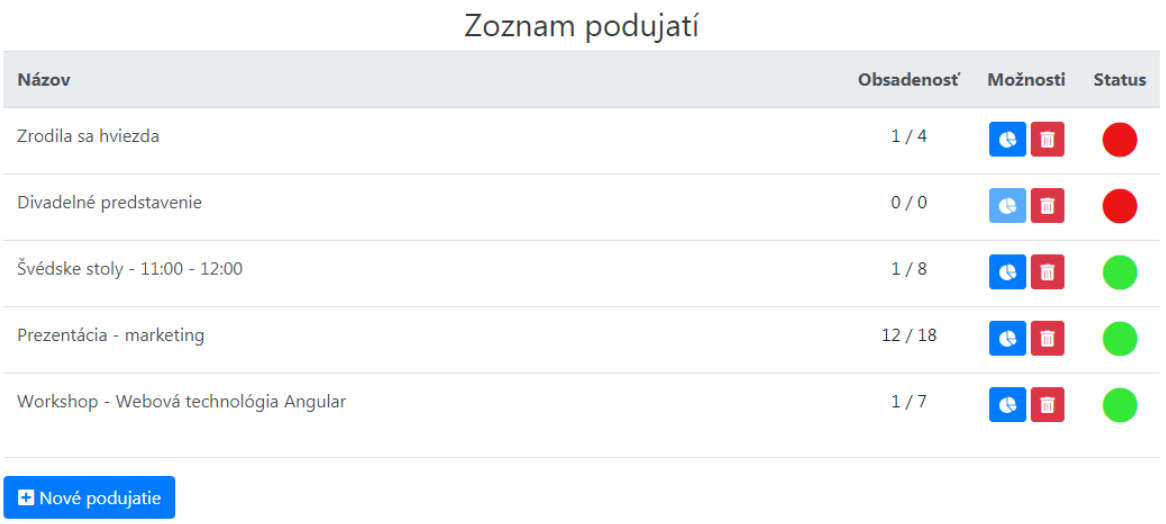
Aplikácia rezervačného systému bola implementovaná postupne podľa funkčných a nefunkčných požiadaviek. Pre komunikáciu komponentov so zvyškom aplikácie boli použité aj vstupy, ktoré požadujú informácie o používateľovi akými sú: identifikačný kľúč používateľa, informácie o používateľovi a binárna informácia o tom, či sa jedná o administrátora. Preto bol pre testovanie aplikácie v koreňovom komponente vytvorený testovací formulár, pomocou ktorého je možné simulovať používateľa. Pri použití v inom, už existujúcom projekte, je možné na pozíciu týchto vstupov napojiť atribúty reálnych používateľov. Obrázok formulára pre simulovanie používateľa je možné vidieť aj na obrázku (Obr. 24).



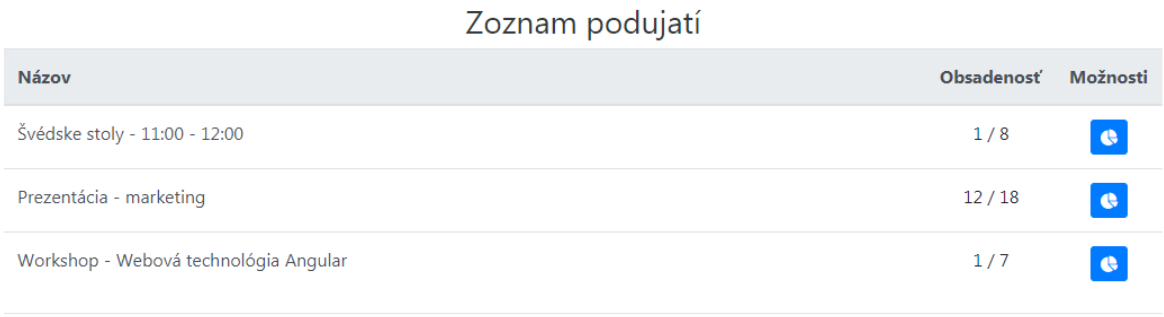
Obr. 24. Testovací formulár pre simuláciu používateľa

5.5.1 Prehľad všetkých podujatí a zmazanie podujatia

Prehľad všetkých podujatí je zobrazovaný ako HTML tabuľka, kde každý riadok tabuľky predstavuje jedno rezervačné podujatie. Riadky tabuľky sú generované pomocou direktívy **ngFor*. Stĺpce tabuľky obsahujú: názov a obsadenosť, tlačidlo pre zobrazenie grafu obsadenosti a pri pohľade administrátora aj tlačidlo pre zmazanie a ikonu statusu daného podujatia. Zmazanie podujatia prebieha na základe jeho identifikačného kľúča, ktorý sa posiela ako parameter pre metódu zmazania. Po kliknutí na názov podujatia sa zobrazí jeho detail. Na obrázkoch (Obr. 25) a (Obr. 26) je možné vidieť ukážku zoznamu podujatí z pohľadu administrátora a z pohľadu používateľa.



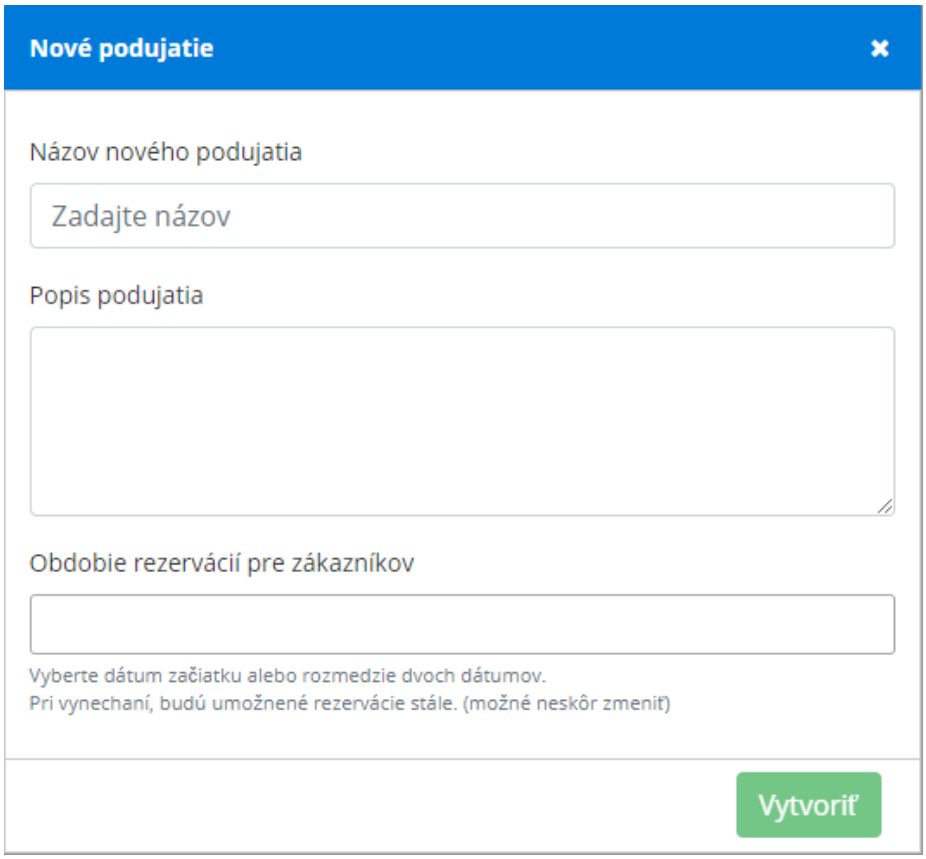
Obr. 25. Zoznam podujatí – pohľad administrátora



Obr. 26. Zoznam podujatí – pohľad používateľa

5.5.2 Vytvorenie nového podujatia

Nové podujatie administrátor pridáva pomocou tlačidla *Nové podujatie*, ktoré je zobrazené na konci prehľadu podujatí. Formulár, do ktorého sa zadávajú informácie o novom podujatí je vložený v PrimeNG dialógovom modálnom okne. Do polí formulára je nutné zadať názov a možné je zadať aj popis a obdobie rezervácií pre zákazníkov.



Obr. 27. Formulár pre vytvorenie nového podujatia

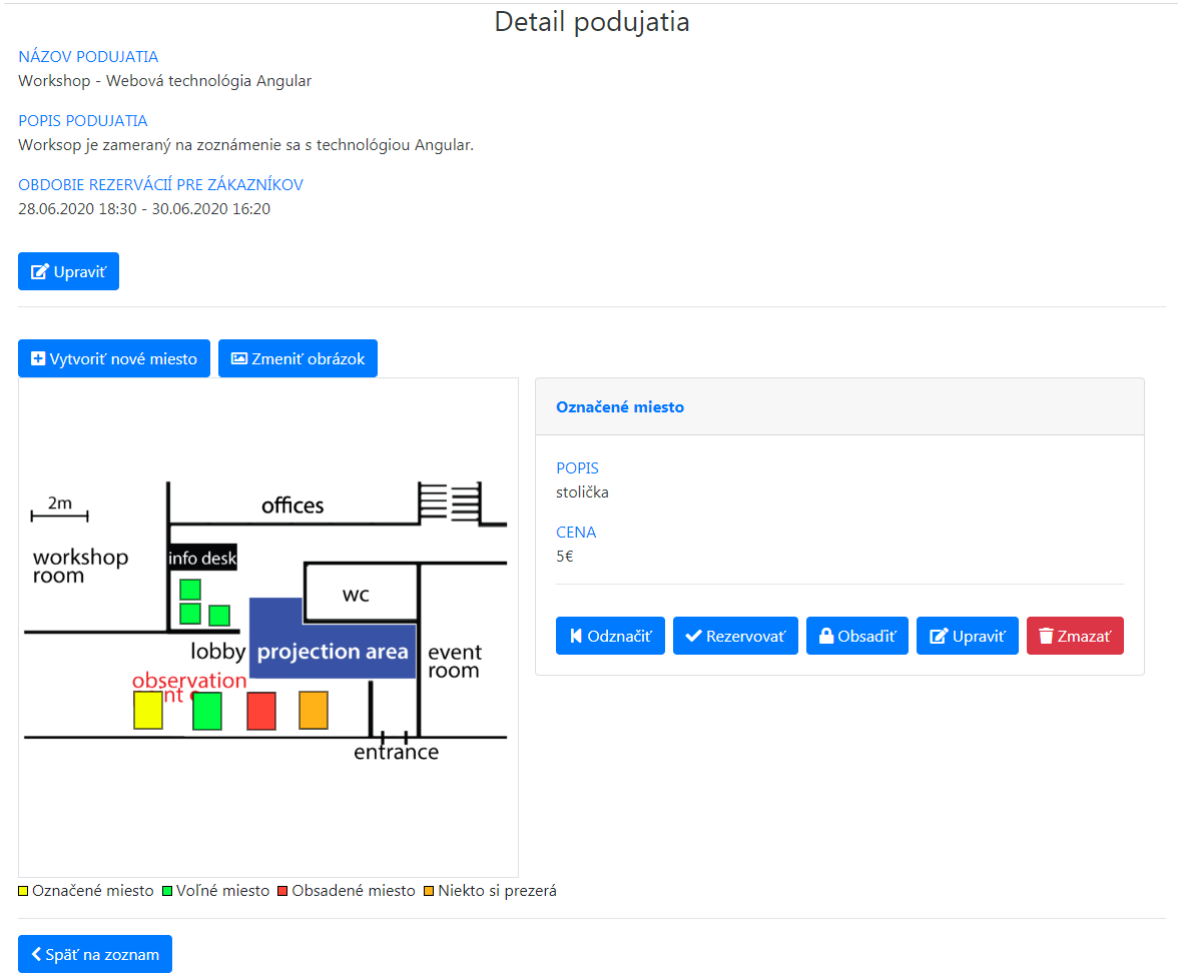
5.5.3 Detail a úprava podujatia

Detail podujatia zobrazuje informácie o vybratom podujatí. Nachádza sa v ňom názov a popis podujatia. Administrátorovi je tiež zobrazené obdobie rezervácií pre zákazníkov a má možnosť všetky tieto údaje upraviť. Aby sa popis podujatia zobrazil správne aj s odsekmi na nový riadok, je elementu *,* v ktorom sa nachádza, nastavený CSS štýl: *"white-space: pre-line;"*. V detaile sa tiež nachádza samotné plátno, na ktorom sú zobrazené miesta podujatia a legenda, ktorá vysvetľuje farebnú odlišnosť miest.

Plátno je v HTML šablóne reprezentované ako *<svg></svg>*. Vo vnútri plátna sa nachádza HTML tag *<image></image>*, ktorý reprezentuje obrázok mapy podujatia. Jednotlivé miesta na plátne sú vykresľované pomocou direktívy **ngFor,* ktorá pre každé miesto vytvorí útvar *<rect/>,* ktorému sú nastavené všetky potrebné atribúty ako výška, šírka, umiestnenie na súradniciach x a y, transformačná matica, farba a ďalšie.

Po kliknutí na dostupné miesto, sa toto miesto sfarbí na žlto a zobrazí sa jeho detail. Administrátor má možnosť pristúpiť aj na detail nedostupného miesta.

Zobrazenie detailu podujatia pre administrátora a pre používateľa je možné vidieť aj na obrázkoch (Obr. 28) a (Obr. 29).



Obr. 28. Detail podujatia – pohľad administrátora

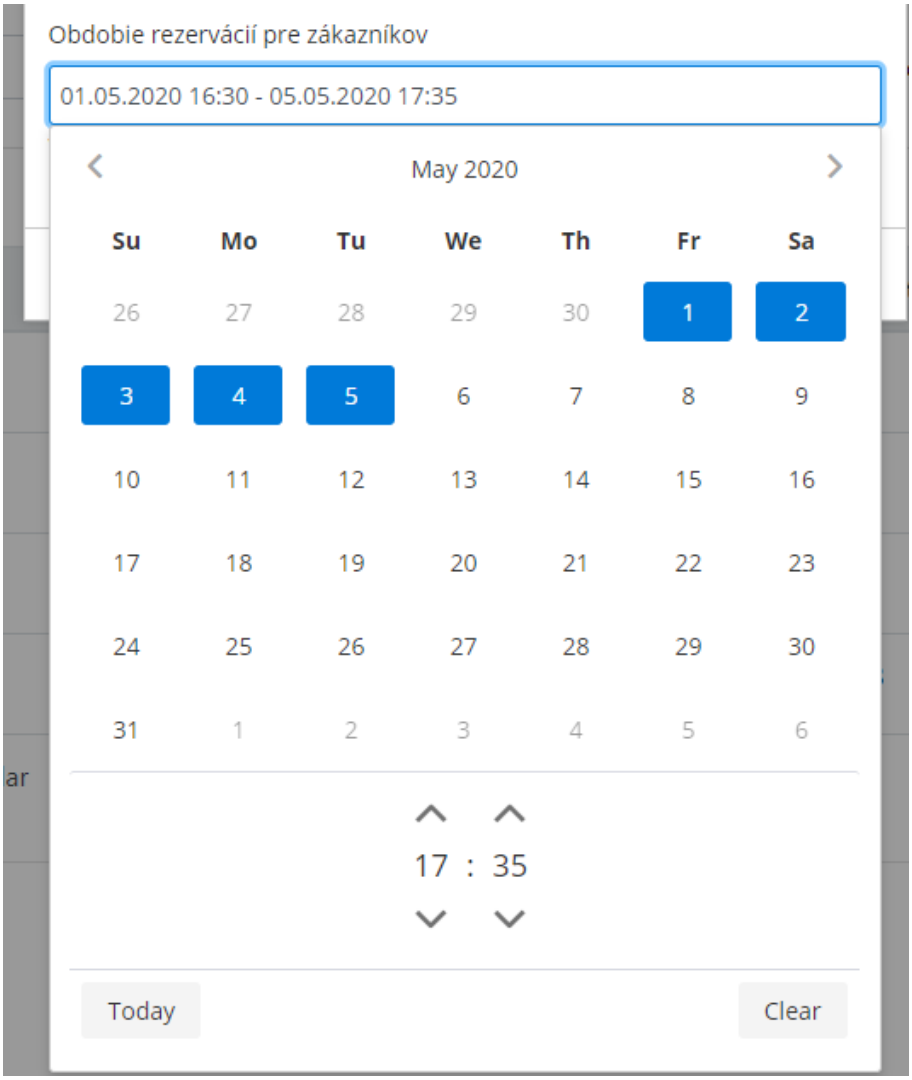
Obr. 29. Detail podujatia – pohľad používateľa

5.5.4 Nastavenie dátumu a času obdobia rezervácií

Dátum a čas obdobia rezervácií je možné nastaviť hneď pri vytváraní podujatia, alebo sa dá neskôr zmeniť v detaile podujatia. Pre výber obdobia slúži PrimeNG widget kalendára, na ktorom je možné nastaviť aj čas.

Ako obdobie rezervácií je možné nastaviť:

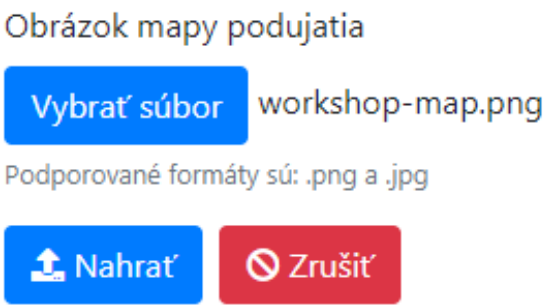
- **Jeden dátum a čas** – rezervácie budú umožnené zákazníkom od začiatku tohto dátumu a času.
- **Rozpätie dátumov a časov** – rezervácie budú umožnené od začiatku prvého dátumu a času po koniec druhého dátumu a času.
- **Žiadne dátumy** - rezervácie budú umožnené zákazníkom stále.



Obr. 30. Nastavenie obdobia rezervácií

5.5.5 Pridanie alebo zmena mapy podujatia

Pridať alebo zmeniť obrázok mapy podujatia má umožnené administrátor. Súbor obrázku musí mať príponu *.jpg* alebo *.png*. Po nahratí obrázku je možné ho potvrdiť a uložiť. Obrázok sa nahrá do súborovej databázy a vytvorí sa preň nová inštancia triedy *map-image,* ktorá sa s podujatím prepojí pomocou jej identifikačného kľúča.

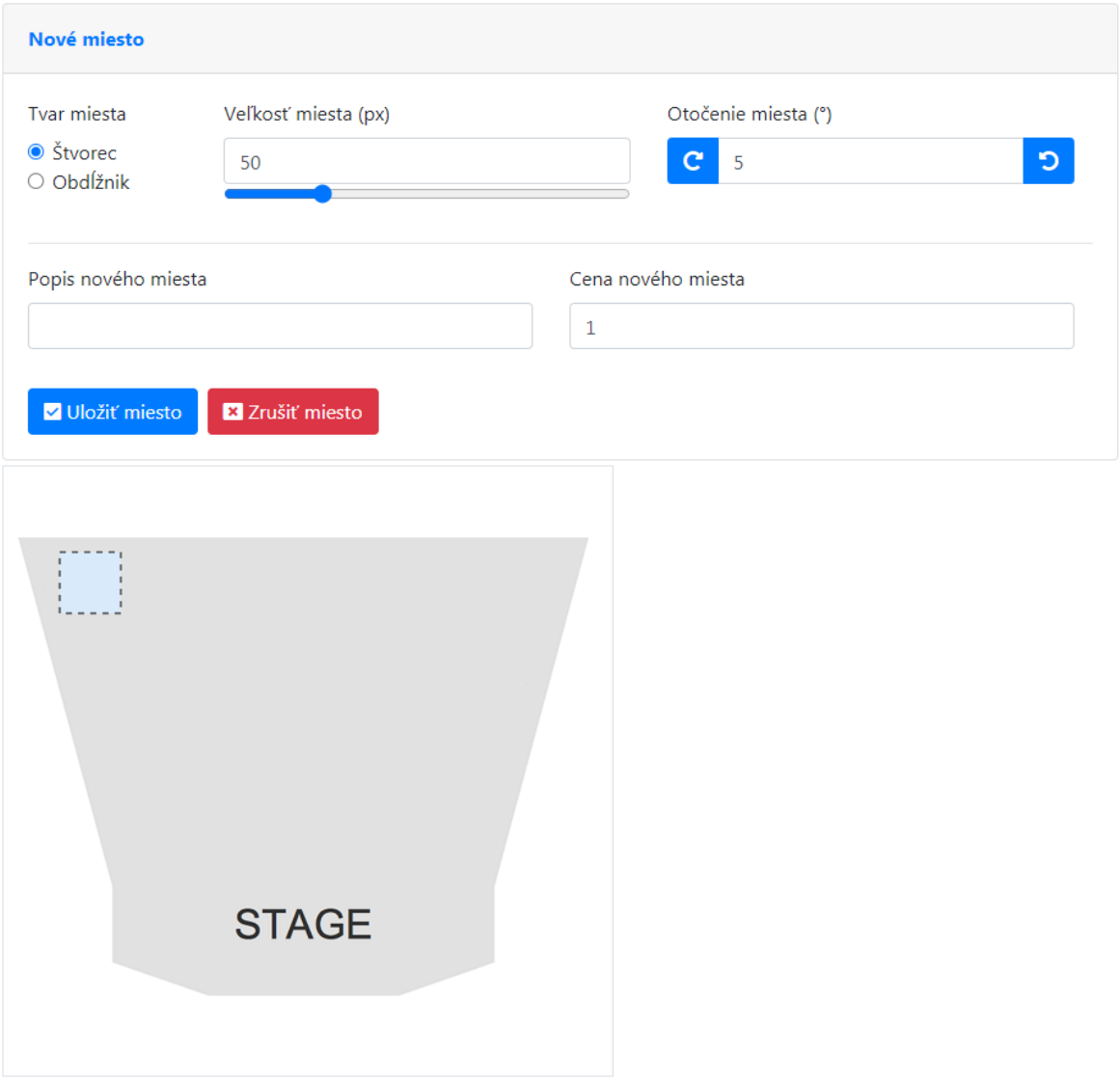


Obr. 31. Pridanie obrázku mapy podujatia

5.5.6 Vytvorenie nového miesta

Nové miesto môže administrátor vytvoriť po kliknutí na tlačidlo *Vytvoriť nové miesto.* Po kliknutí sa zobrazí na plátne ukazovateľ nového miesta. Nad plátnom sa tiež zobrazí formulár, v ktorom môže nastaviť popis a cenu nového miesta. Taktiež je možné vybrať tvar miesta a to štvorec alebo obdĺžnik. Pri tvare obdĺžnika je možné nastaviť šírku aj výšku útvaru, ktorý miesto reprezentuje. Pri štvorci sa nastavuje len veľkosť. Ďalej je možné nastaviť otočenie miesta v stupňoch a následné pomocou tlačidiel je možné ho otáčať smerom doprava alebo doľava. Toto otočenie sa pri ukladaní zapíše ako transformačná matica.

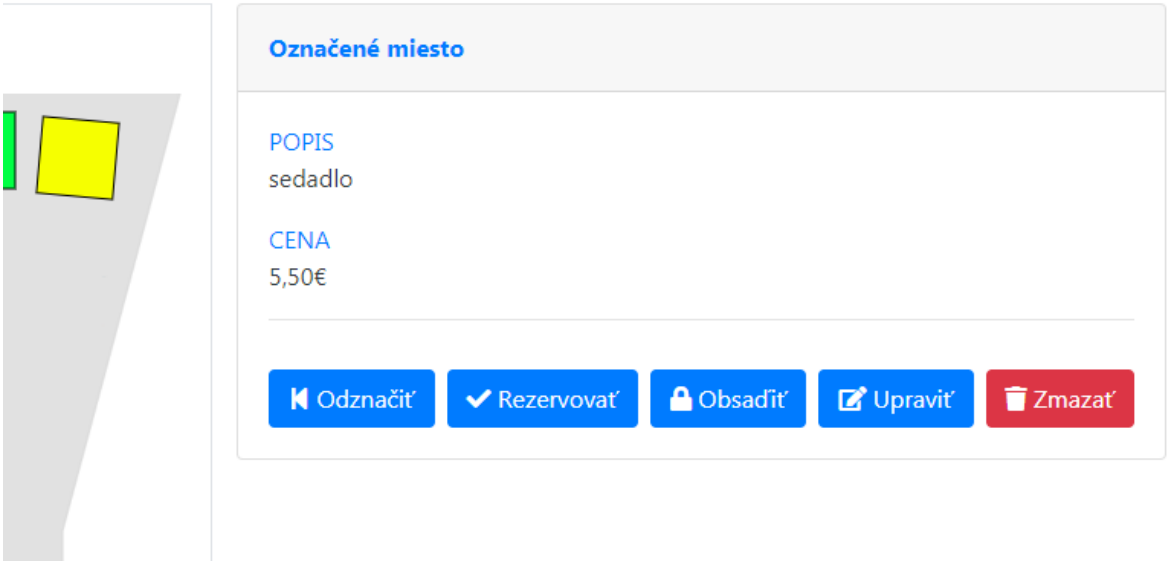
Samotný ukazovateľ nového miesta je zobrazovaný pomocou knižnice SVG.js a je možné ho ľubovoľne premiestňovať v rámci plátna. Zmeny veľkosti sa okamžite premietnu na ukazovateli nového miesta, čo je zabezpečené direktívou *(ngModelChange),* ktorá po zmene dát zavolá metódu *resizeNewShape()*. Po kliknutí na *Ulož miesto* sa všetky parametre ukazovateľa nového miesta nastavia ako atribúty nového miesta, ktoré sa uloží do databázy. To spôsobí aktualizáciu poľa všetkých miest a to sa prejaví tak, že sa vytvorí nový útvar *</rect>* vo vnútri plátna, ktorý okamžite zobrazí nové miesto.



Obr. 32. Vytvorenie nového miesta

5.5.7 Zmena stavu obsadenosti, úprava a zmazanie miesta

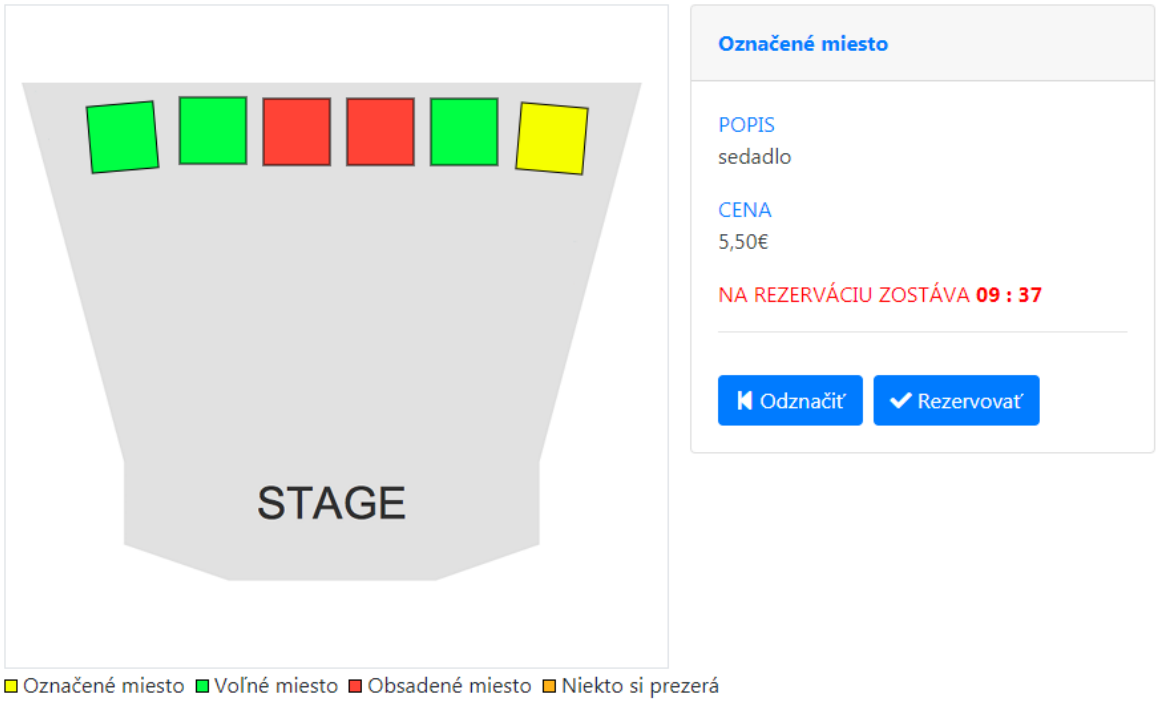
Administrátorovi je po kliknutí na miesto umožnené dané miesto obsadiť alebo uvoľniť. Pri voľbe obsadiť dané miesto, je zobrazený používateľovi formulár, ktorý sa nachádza v PrimeNG dialógovom modálnom okne. Vo formulári má možnosť nastaviť poznámku ku obsadeniu. Administrátor môže miestu upraviť popis a cenu a taktiež môže miesto zmazať. Pri upravovaní miesta sa vytvorí kópia aktuálneho miesta a jej atribúty sa cez direktívu *[(ngModel)]* prepoja s poľami formulára. Pri potvrdení zmien sa atribúty nastavia aj vybratému miestu a to sa uloží do databázy.



Obr. 33. Detail označeného miesta – pohľad administrátora

5.5.8 Rezervácia miesta

Ako používateľ, tak aj administrátor majú k dispozícii tlačidlo pre rezerváciu podujatia. Rezervovať je možné len dostupné miesto. Po rezervovaní miesta, sa do atribútov miesta zapíše identifikačný kľúč používateľa a informácie o používateľovi. Rezervované miesto sa na mape zvýrazní červenou farbou a ostatným používateľom, okrem administrátora, je znemožnené naň kliknúť.



Obr. 34. Rezervácia miesta – pohľad používateľa

5.5.9 Rýchla odozva na zmeny

Rýchlu odozvu na zmeny zabezpečuje databáza Cloud Firestore. Klientská časť aplikácie sa pomocou asynchrónnej metódy *subscribe()* prihlási na odber zmien jednotlivých objektov čerpaných z databázy. Tieto zmeny sa posielajú ako objekt *observable.* Pri hocijakej zmene dát v databáze sú tieto zmeny zaznamenané aj v klientskej časti aplikácie a spravujú sa v tele metódy *subscribe().*

5.5.10 Pozdržanie miesta pre používateľa na určitý čas

Používateľovi je po kliknutí na dostupné miesto, toto miesto pozdržané na čas určený podľa vstupného parametra, štandardne je to desať minút. Takto pozdržané miesto je zvýraznené oranžovou farbou. Po kliknutí na tlačidlo *Odznačiť* alebo po kliknutí na plátno sa miesto odznačí a pozdržanie sa vynuluje.

Pozdržanie je vyriešené pomocou identifikačného kľúča používateľa a dátumu konca pozdržania, ktoré sú priradené danému miestu a upravené miesto je uložené do databázy. Pre dátum a čas pozdržania sa najskôr zoberie aktuálny dátum a čas z databázy, ku ktorému sa pripočíta čas pozdržania.

Používateľovi sa tiež po označení miesta spustí časovač, ktorý môže vidieť aj v detaile daného miesta. Po vypršaní časovača sa miesto automaticky odznačí a pozdržanie sa vynuluje. V prípade, že by používateľ zavrel webový prehliadač alebo by stránku obnovil sa zavolá metóda:

@HostListener('window:beforeunload', ['$event']) beforeUnloadHandler(event), ktorá sa volá vždy pred zavretím stránky prehliadačom. V tele tejto metódy je funkcia *unselectShape()*, ktorá miesto odznačí.

Kontrola takto pozdržaných miest sa vykonáva ešte v tele asynchrónnej metódy, ktorá ja volaná po každej zmene dát a po inicializácii komponentu. V rámci tejto kontroly sa porovnáva čas pozdržania miesta s aktuálnym časom získaným z databázy. Pokiaľ je čas pozdržania menší ako aktuálny čas, tak je pozdržanie miesta vynulované. Táto kontrola prebieha u všetkých používateľov a kontrolujú sa všetky miesta, ktoré ešte nie sú rezervované.

Uvoľnenie takto pozdržaného miesta je teda možné zhrnúť do nasledujúcich bodov:

- Používateľ dané miesto odznačí kliknutím na tlačidlo alebo kliknutím na plátno.

- Používateľovi vyprší čas časovača.
- Používateľ opustí alebo obnoví stránku prehliadača.
- Niektorý z používateľov zruší pozdržanie miesta na základe porovnania času konca pozdržania a aktuálneho času, ktorý je získaný z databázy.

5.5.11 Farebné odlíšenie miest podľa vlastností

Pre farebné odlíšenie miest má každý útvar *<rect/>*, ktorý predstavuje konkrétne miesto, nastavený atribút *fill* nasledovne: *[attr.fill]="getFillOfShape(shape)"*, kde *getFillOfShape()* je metóda, ktorá vracia ako návratovú hodnotu reťazec. Reťazec, ktorý metóda vracia obsahuje hexadecimálny kód predstavujúci farbu miesta. Jednotlivé farby majú nasledovný význam:

- **zelená** – miesto je dostupné a nikto si ho neprehliada.
- **oranžová** – miesto je dostupné, no niekto si ho prehliada a tak je pozdržané.
- **červená –** miesto nie je dostupné, niekto si ho rezervoval.
- **žltá** – miesto je vami označené.

5.6 Publikácia rezervačného systému ako open source

Výsledná webová aplikácia rezervačného systému bola publikovaná ako open source na platforme GitHub, kde je možné ju stiahnuť a použiť. Aplikácia je publikovaná bez adresára *node_modules,* kvôli jeho objemu a je potrebné ho doinštalovať pomocou príkazu: *npm install* cez príkazový riadok. Pri inštalácií je nutné sa v rámci príkazového riadku nachádzať v adresári aplikácie. Taktiež je potrebné do súboru *enviroment.ts* vložiť kód, ktorý obsahuje údaje pre spojenie s platformou Firebase. Platforma Firebase je voľne dostupná a po vytvorení projektu na: *https://console.firebase.google.com/* je kód pre spojenie automaticky vygenerovaný. Možnosť použitia inej backendovej technológie je popísaná v podkapitole *5.4*. Podrobnejšia dokumentácia sa nachádza priamo v GitHub repozitári.

Odkaz na GitHub repozitár: *https://github.com/martin-butko/angular-place-reservation*

5.6.1 Popis vstupov a výstupov komponentov aplikácie

Vstupy a výstupy komponentov slúžia pre komunikáciu medzi sebou a pre komunikáciu so zvyškom aplikácie. Obsahujú prednastavené hodnoty, ktoré je možné zmeniť. Zmena ich

hodnôt je možná v šablóne, v rámci HTML tagu daného komponentu. Na hodnoty výstupov je možné nastaviť aj funkcie. Napríklad:

<app-component [vstup]="hodnotaVstupu" (výstup)="hodnotaVýstupu"></app-component>

Komponent *reservations*

Tento komponent sa do šablóny vkladá tagom *<app-reservations></app-reservations>* a poskytuje vstupy, ktoré sú popísané v tabuľke (Tab. 5).

Tab. 5. Vstupy komponentu reservations

Názov	**Typ**	**Popis**	**Predvolená hodnota**
isAdmin	boolean	Určuje, či používateľ vlastní rolu administrátora	false
userKey	reťazec	Identifikačný kľúč používateľa	"
userInfo	reťazec	Informácie o používateľovi	"

Komponent *reservation-detail*

Do šablóny sa vkladá tagom *<app-reservation-detail></app-reservation-detail>* a poskytuje vstupy a výstupy popísané v tabuľke (Tab. 6).

Tab. 6. Vstupy a výstupy komponentu reservation-detail

Názov	**Typ**	**Popis**	**Predvolená hodnota**
Vstupy			
isAdmin	boolean	Určuje, či používateľ vlastní rolu administrátora	false
userKey	reťazec	Identifikačný kľúč používateľa	"
userInfo	reťazec	Informácie o používateľovi	"
reservationSchemeKey	reťazec	Identifikačný kľúč podujatia, ktorého detail má byť zobrazený	null
canvasWidth	číslo	Šírka SVG plátna v pixeloch	500
canvasHeight	číslo	Výška SVG plátna v pixeloch	500

minShapeSize	číslo	Minimálna veľkosť miesta v pixeloch (šírka aj výška)	20
maxShapeSize	číslo	Maximálna veľkosť miesta v pixeloch (šírka aj výška)	150
minutesToHold	číslo	Počet minút na koľko je miesto pozdržané pre používateľa	10
Výstupy			
onGoBack	EventEmitter	Udalosť, ktorá sa zavolá po kliknutí na tlačidlo *Späť na zoznam*	Metóda onGoBack() z komponentu reservations

ZÁVER

V rámci diplomovej práce boli popísané moderné JavaScriptové nástroje, ktoré uľahčujú vývoj interaktívnych aplikácií a čoraz viac sú používané v rámci vývoja webových aplikácií. Popísané boli aj JavaScriptové knižnice, ktoré pracujú s vektorovým formátom SVG a tak umožňujú tvoriť unikátne používateľské rozhrania. Tieto SVG knižnice boli tiež porovnané na základe kritérií akými sú cieľ knižnice, dokumentácia, syntax, aktuálnosť knižnice a podobne. Následne na základe tohto porovnania bola vybratá vhodná SVG knižnica, ako aj iné užitočné technológie pre implementáciu výslednej webovej aplikácie. Medzi hlavné technológie použité pri implementácií patria: framework Angular, backendová platforma Firebase a knižnica SVG.js, ktorá sa stará o správu a manipuláciu vektorových útvarov v rámci výslednej aplikácie.

Výstupom práce je webová aplikácia predstavujúca dynamický rezervačný systém, ktorý umožňuje vytváranie rezervačných schém pre rôzne typy podujatí, ako napríklad kiná, workshopy a iné. Administrátorovi aplikácie je umožnené pre jednotlivé podujatia vytvárať miesta určené k rezervácií. Predmetné miesta je možné umiestniť kdekoľvek na SVG plátno pomocou funkcie drag and drop. Miesta je tiež možné ľubovoľne otáčať, meniť im veľkosť alebo je možné vybrať ich tvar. Ako pozadie plátna sa dá nastaviť ľubovoľný obrázok predstavujúci mapu alebo priestory podujatia. Spustenie a koniec rezervácií je tiež možné nastaviť na časové obdobie. Pri prezeraní miesta používateľom je vybraté miesto preňho pozdržané na určitý čas, ktorý znázorňuje aj časovač a v tom období ku tomuto miestu nemajú ostatní používatelia prístup.

Táto aplikácia je modulárna a je možné ju použiť aj v iných projektoch využívajúcich front-endový framework Angular. Rezervačný systém je publikovaný ako open source a tak poskytuje komunite používateľov jednoduchšiu správu rezervácií, bez nutnosti implementácie statických riešení. V rámci aplikácie sa podarilo implementovať všetky funkčné aj nefunkčné požiadavky. Veľkou výhodou je aj možnosť použitia ľubovoľnej backendovej technológie, kedy stačí prepísať metódy v súboroch služieb, ktoré tvoria spojenie so serverom.

Tento webový rezervačný systém by bolo v budúcnosti možné obohatiť o nové funkcie, ako napríklad kopírovanie podujatí alebo možnosť usporiadania miest do mriežky. Aplikácia by tiež mohla byť optimalizovaná podľa prípadných požiadaviek zo strany komunity jej používateľov a rozšírená o viacero jazykových lokalizácií.

ZOZNAM POUŽITEJ LITERATÚRY

[1] KIZILOVA, Daria. All You Need to Know About JavaScript Programming Tools - DZone Web Dev. *dzone.com* [online]. 14. únor 2017 [cit. 2020-02-16]. Dostupné z: https://dzone.com/articles/javascript-programming-tools

[2] WARCHOLINSKI, Matt. Top 10 Tools For JavaScript Development. *Blog Brainhub.eu* [online]. [cit. 2020-02-16]. Dostupné z: https://brainhub.eu/blog/top-tools-for-javascript-development/

[3] ČÁPKA, David. *MVC architektura* [online]. [cit. 2020-02-19]. Dostupné z: https://www.itnetwork.cz/mvc-architektura-navrhovy-vzor

[4] HOLLINGWORTH, Dave. The model-view-controller pattern. *Dave Hollingworth* [online]. 6. únor 2016 [cit. 2020-02-19]. Dostupné z: https://davehollingworth.com/blog/the-model-view-controller-pattern/

[5] *MVC Framework - Introduction - Tutorialspoint* [online]. [cit. 2020-02-19]. Dostupné z: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

[6] *React – A JavaScript library for building user interfaces* [online]. 2020 [cit. 2020-02-26]. Dostupné z: https://reactjs.org/

[7] *Web App Development Services | Web App Developers* [online]. [cit. 2020-02-26]. Dostupné z: https://codetoart.com/services/web-app-development

[8] *Vue.js* [online]. [cit. 2020-02-26]. Dostupné z: https://vuejs.org/

[9] *Angular* [online]. [cit. 2020-03-02]. Dostupné z: https://angular.io/

[10] *TypeScript - JavaScript that scales.* [online]. [cit. 2020-03-02]. Dostupné z: https://www.typescriptlang.org/

[11] *Aurelia* [online]. [cit. 2020-03-02]. Dostupné z: https://aurelia.io/

[12] *SVG Tutorial* [online]. [cit. 2020-03-06]. Dostupné z: https://www.w3schools.com/graphics/svg_intro.asp

[13] PEARLMAN, Ellen a Lorien HOUSE. *SVG for web developers*. Upper Saddle River (New Jersey): Prentice Hall PTR, 2003. ISBN 978-0-13-100499-3.

[14] ROCHELEAU, Jake. *8 Best Free Libraries For SVG* [online]. 16. únor 2018 [cit. 2020-03-06]. Dostupné z: https://www.webdesignerdepot.com/2018/02/8-best-free-libraries-for-svg/

[15] *SVG.js* [online]. [cit. 2020-03-07]. Dostupné z: https://svgjs.com/docs/3.0/

[16] *Raphaël - JavaScript Library* [online]. [cit. 2020-03-07]. Dostupné z: https://dmitrybaranovskiy.github.io/raphael/

[17] *Snap.svg* [online]. [cit. 2020-03-09]. Dostupné z: http://snapsvg.io/

[18] BOSTOCK, Mike. *D3.js - Data-Driven Documents* [online]. [cit. 2020-03-15]. Dostupné z: https://d3js.org/

[19] *p5.js* [online]. [cit. 2020-03-15]. Dostupné z: https://p5js.org/

[20] *Two.js* [online]. [cit. 2020-03-15]. Dostupné z: https://two.js.org/

[21] What Is an Online Booking System? (Reservation System Definition). *Amelia WordPress Booking Plugin* [online]. 6. březen 2020 [cit. 2020-03-25]. Dostupné z: https://wpamelia.com/online-booking-system/

[22] HNIK, Ondřej. *Čo je vlastne rezervačný systém* [online]. [cit. 2020-03-25]. Dostupné z: https://blog.reservanto.sk/14/

[23] User Interface Design in Modern Web Applications. *Smashing Magazine* [online]. [cit. 2020-04-01]. Dostupné z: https://www.smashingmagazine.com/user-interface-design-in-modern-web-applications/

[24] *npm - build amazing things* [online]. [cit. 2020-04-01]. Dostupné z: https://www.npmjs.com/

[25] Firebase. *Firebase* [online]. [cit. 2020-04-15]. Dostupné z: https://firebase.google.com/?hl=sk

[26] What is BaaS? | Backend-as-a-Service vs. Serverless. *Cloudflare* [online]. [cit. 2020-05-12]. Dostupné z: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/

[27] *Bootstrap* [online]. [cit. 2020-05-23]. Dostupné z: https://getbootstrap.com/

[28] *Font Awesome* [online]. [cit. 2020-06-05]. Dostupné z: https://fontawesome.com

[29] *PrimeNG - Angular UI Component Library* [online]. [cit. 2020-06-14]. Dostupné z: https://www.primefaces.org/primeng/

[30] *Chart.js - Open source HTML5 Charts for your website* [online]. [cit. 2020-06-15]. Dostupné z: https://www.chartjs.org/

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

BaaS	Backend-as-a-Service
BSD	Berkeley Software Distribution
CLI	Command Line Interface
CSS	Cascading Style Sheet
DOM	Document Object Model
HTML	Hyper Text Markup Language
MIT	Massachusetts Institue of Technology
MVC	Model-view-controller
NPM	Node package manager
SDK	Software development kit
SQL	Structured Query Language
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
VDOM	Virtual Document Object Model
VML	Vector Markup Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

Príloha P I: CD disk

PRÍLOHA P I: CD DISK

Na disku formátu CD sa nachádza:

- elektronická verzia diplomovej práce vo formáte PDF
- zdrojové kódy výstupnej aplikácie