Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

jQuery - vytvoření pluginu pro rezervační systém

Plzeň, 2013 Petr Kukrál

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 5. května 2013 Petr Kukrál

Poděkování

Rád bych poděkoval panu Ing. Martinu Dostalovi za odborné konzultace a za přístup vedení bakalářské práce, díky kterému jsem vše včas stihnul. Také bych rád poděkoval své přítelkyni Veronice Švecové za pomoc s korekturami této práce.

Abstract

This thesis deals with the creation of plugin for reservation system in programming language JavaScript, using the jQuery library. Plugin can be used for every type of a reservation system. The illustration of this thesis is the application of plugin for a reservation system of cinemas and application of plugin for reservation system of squash court. Emphasis is placed on graphic design of the plugin. Graphic design should be attention getting, that's why the plugin disposes of easy expansion of graphical function and change of using pictures. Plugin can be also easily integrated into website. The only condition is connection of the jQuery library. Plugin can load input from the file in XLS, CSV and JSON format. Format JSON is in illustration used to realize loading data from the database.

Abstrakt

Tato práce se zabývá tvorbou pluginu pro rezervační systém v programovacím jazyce JavaScript s použitím knihovny jQuery. Plugin má být obecně použitelný pro jakýkoliv rezervační systém. Ukázkou použití této práce je použití pluginu pro rezervační systém kina a použití pluginu pro rezervační systém squashového kurtu. Důraz je kladen na grafický vzhled pluginu. Grafický vzhled má být co nejpoutavější, proto plugin disponuje snadným rozšířením grafických funkcí a změnou používaných obrázků. Plugin je také možné snadno zakomponovat do webové stránky. Jedinou podmínkou je připojení knihovny jQuery. Plugin dokáže načíst vstup ze souboru ve formátu XLS, CSV a JSON. Pomocí formátu JSON je pak v ukázce realizované načítání dat z databáze.

Obsah

1 Úvod

JQuery je moderní knihovna napsaná v jazyce JavaScript. Je malá a rychlá. Poskytuje základní metody pro práci s DOM (Document Object Model). Celý plugin je postavený právě na této knihovně. Touto prací vytvoříme plugin, který rozšíří knihovnu jQuery o funkci rezervačního systému.

Při hledání podobných open source pluginů, které by měly splňovat alespoň rámcově zadání této práce, jsem žádné vyhovující nenašel. V kapitole Relevantní práce představím podobnou práci. Nebude se ale jednat o plugin a tato práce má jen omezené možnosti použití. Proto bych chtěl tímto pluginem vývojářům nabídnout open source systém pro rezervace, který by mohli ve svých projektech využívat.

Zvláště chci upozornit na kapitolu Tvorba pluginu jQuery. V kapitole se budu věnovat nejen návrhu pluginu pro rezervační systém, ale také best practices, které se používají při tvorbě pluginu. Dále se také například chci zmínit o ošetření vytvářeného pluginu při použití více pluginů na jedné stránce apod.

Cílem této práce je vytvořit plugin pro rezervační systém. Tento rezervační systém by měl být univerzální. Půjde použít například pro rezervační systém kina, nebo rezervaci squashového kurtu. Proto bude systém disponovat velkým množstvím nastavení, aby bylo dosaženo co největší univerzálnosti. Také musí mít dobře navrženou strukturu výstupu, aby se k jednotlivým elementům dalo snadno přistupovat pomocí scriptů. Plugin dokáže pracovat s následujícími formáty vstupu:

- JSON
- CSV
- XLS

Ukázka použití pluginu by měla obsahovat nejen načítání dat z konfiguračního souboru, ale také načítání dat z databáze. Velký důraz bude kladen na grafické provedení ukázky použití pluginu. Plugin by mělo jít rozšířit o další grafické funkce a to nejlépe odkazem na funkci v nastavení pluginu.

2 Použité technologie

V této kapitole si popíšeme technologie, které byly použity nejen při tvorbě samotného pluginu, ale také při tvorbě ukázky použití. Nejdříve se budeme zabývat HTML 5 a CSS 3. Probereme, co tyto nové technologie přinesly, a co bylo využito při tvorbě pluginu a ukázky. Poté stručně popíšeme jazyk PHP a databázový systém MySQL. Dále se seznámíme s normálními formami. Projdeme si základní informace o jazyku JavaScript a moderním formátu JSON, se kterým plugin umí pracovat. Zjištěné informace porovnáme s knihovnou jQuery. Pak se budeme zabývat hlavními výhodami, které knihovna přináší, a proč je dobré ji používat. Po celou dobu práce na projektu byl používán systém pro správu verzí GIT, představíme si tedy i další systémy pro správu verzí a porovnáme jejich vlastnosti. Nakonec si popíšeme jednotlivé frameworky a pluginy, které byly při práci použity.

2.1 HTML 5 a CSS 3

V práci jsou obsaženy frameworky a pluginy, které používají HTML 5 a CSS 3. V této podkapitole si představíme některé důležité vlastnosti HTML 5 a CSS 3, ale hlavně se zaměříme na prvky, které se vyskytují přímo v ukázce pluginu rezervačního systému.

Nové HTML 5 přineslo do vývoje webových aplikací řadu změn. Mezi ně patří práce se soubory pomocí nového FILE API. To umožňuje nahrávat na server soubory o větší velikosti, než server povoluje. Do té doby uživatel nemohl pomocí PHP skriptu větší soubory nahrát [1]. Dalšími přínosy jsou podpora vektorové grafiky ve formátu SVG, podpora režimu celé obrazovky a podpora přehrávání videí bez nutnosti používat FLASH.

HTML 5 také zavedlo používání nových značek [2]. V kódu této práce jsou použity značky:

- header
- section
- nav

Pomocí těchto značek rozdělujeme dokument do těchto sekcí. Podle [2], *header* představuje hlavičku své sekce, nebo hlavičku celé stránky. V ukázce použití pluginu obsahuje *header* hlavní nadpis celé stránky a horní navigační lištu. Do elementu *section* patří obsah, který se ze stránky nedá vyjmout bez toho, aby se porušila její logika. V ukázce použití pluginu obsahuje *section* například rezervační systém. Element *nav* představuje hlavní navigaci. V této práci obsahuje *nav* celou horní navigační lištu.

Při psaní pluginu byl velkým přínosem atribut *data-**, který je podporován právě HTML 5 [3]. To nám umožňuje vytvářet si vlastní atributy jednotlivých elementů. Tento atribut nám pomůže hlavně při psaní aplikace v jazyce javascript. Nemusíme data uchovávat například v atributu *rel*, ale vytvoříme si vlastní atribut. Ten musí začínat řetězcem „data-“ a dále může pokračovat libovolným řetězcem. Při vhodném pojmenování atributu se tak už z jeho názvu dozvíme, o jaká data se jedná. V práci se setkáme s atributem *data-status*, ve kterém je uchováno, zda je element zarezervovaný.

CSS 3 přinesly spoustu nových změn. Podporují animaci ve stránce. Pomocí CSS 3 lze nyní vytvořit téměř jakoukoliv grafiku, kromě obrázků a fotek samotných [4]. Dále podporují například stíny, zaoblené rohy, nebo barevné přechody. V práci CSS 3 velmi často využíváme právě pro tyto prvky.

V zadání práce byl kladen důraz na grafické provedení, proto byly CSS 3 při vývoji často používány. V ukázkové aplikaci je používá hlavně framework bootstrap a to například pro vykreslování grafiky tlačítek. V ukázce ho také můžeme najít při použití stínu pod rezervačním systémem kina.

2.2 PHP a MySQL

PHP [5] je široce používaný interpretovaný jazyk. Zpracovává se na straně serveru a výstup se pak posílá klientovi. PHP podporuje objektový návrh aplikace, ale můžeme aplikaci programovat i strukturálně. Díky tomu můžeme do stránky napsat malý krátký skript, ale větší aplikace tak zůstanou díky použití objektů přehledné a dobře upravovatelné. V jazyce PHP je naprogramována řada frameworků. Mezi ně patří například Zend framework a Nette framework.

MySQL je Open Source databázový systém [6], který pro ukládání dat používá tabulky propojené pomocí relací. Používá se v LAMP (Linux, Apache, MySQL, PHP / Python / Perl). V tomto projektu používáme MySQL pro uložení veškerých dat.

Databázi můžeme převést do první, druhé a třetí normální formy. Podle [7], první normální forma musí splňovat tyto podmínky:

- Musí se odstranit všechny sloupce, které mají stejný obsah.
- Pro každou skupinu navzájem vázaných dat se musí vytvořit vlastní tabulka.
- Každý záznam se identifikuje pomocí tzv. primárního klíče.

Podmínky druhé normální formy jsou následující:

- Pokaždé, když se začne v sloupcích jejich hodnota opakovat, je nutné vytvořit více menších tabulek.
- Tabulky se musí spojovat pomocí cizích klíčů.

Třetí normální forma má podle [7] jediné pravidlo. Sloupce, které nejsou bezprostředně závislé na primárním klíči, je nutno umístit do vlastní tabulky.

2.3 JavaScript

Javascript je objektový multiplatformní skriptovací jazyk. Je určen pro snadné zabudování do jiných produktů a aplikací, jako jsou např. webové prohlížeče. Javascript můžeme použít jak na straně klienta, tak na straně serveru [8]. Dále si budeme popisovat pouze použití Javascriptu na straně klienta.

Jazyk JavaScript nám umožňuje vyhledávat, měnit a kontrolovat elementy v modelu DOM (Document Object Model). Například na straně klienta nám umožňuje reagovat na kliknutí myší, validování formuláře před jeho odesláním a mnoho dalšího. To vše bez opětovného načtení stránky. Web se tak za použití tohoto jazyka stává dynamickým. Programování v jazyce JavaScprit je bohužel obtížné, protože zdrojový kód není přehledný. Navíc nenormalizuje rozdíly mezi prohlížeči. Může se tedy stát, že se skript bude v každém prohlížeči chovat jinak. Proto je dobré použít některou z knihoven, která programování značně usnadní. Kód je pak čitelnější a lépe upravovatelný.

JSON (JavaScript Object Notation) je textový formát určený k uložení a výměně dat [9]. Jak už z názvu vyplývá, tento formát je často využíván právě při programování v jazyce JavaScript. JSON není ovšem na jazyce JavaScript nijak závislý a je dobře použitelný i v jiných programovacích jazycích. Formát JSON je velmi podporovaný ve webových technologiích, a proto najdeme funkce pro práci s ním např. i v jazyce PHP. V jazyce JavaScript si můžeme data z JSON převést do objektu.

2.4 JQuery

JQuery je open source knihovna, napsaná v jazyce Javascript. Normalizuje rozdíly mezi prohlížeči a nabízí velkou skupinu metod, které zjednodušují vývoj aplikací [10]. JQuery nenabízí žádnou novou funkcionalitu, ale tu stávající obaluje do vlastních metod, které jsou v kódu čitelnější a lépe se s nimi pracuje [11]. Nacházejí se v ní jen nejpoužívanější metody využívané při vývoji aplikace. Proto knihovna zůstává malá.

V minimalizované verzi má okolo 91 kB (platí pro verzi 1.9.0). Ke knihovně lze ale najít velké množství pluginů a zásuvných modulů. Programátor si tak sám může zvolit, které pluginy bude ke knihovně využívat. Tento přístup má velkou výhodu v tom, že nestahujeme ze stránky zbytečné skripty, které se pak nebudeme používat. Stránka se tak rychleji načítá. Knihovna je dobře zdokumentovaná a programátor si tak snadno dohledá vlastnosti, které potřebuje využívat, a to včetně názorných příkladů.

Podle [10] lze rozdělit knihovnu jQuery do tří logických částí:

- Manipulace s elementy a jejich vyhledávání.
- Řetězená volání metod jQuery na skupinu elementů.
- Implicitní iterace a používání obalu jQuery.

Hledání elementů probíhá pomocí selektorů jazyka CSS [12]. Tedy prohledáváme model DOM (Document Object Model) a nad ním spouštíme různé operace. Zde knihovna jQuery sjednocuje funkce pro vyhledávání. Uvedeme si příklad. Pomocí jazyka Javascript pro hledání elementu na stránce s určitým id použijeme metodu *document.getElementById("id")*. Pro hledání elementu image slouží funkce *document.getElementsByTagName("img")*. Jak je vidět, tento zápis má několik nevýhod. Funkce mají příliš dlouhý zápis. Jelikož je vyhledávání v modelu DOM velmi časté, bylo by lepší, kdyby funkce měly kratší a výstižnější zápis. Druhou nevýhodou je, že pro hledání různých elementů používáme různé funkce. Nyní si ukážeme hledání elementů s použitím knihovny jQuery. Pro názornější ukázku použijeme zkrácený zápis $, který představuje proměnnou obsahující objekt jQuery. Pro hledání id bude zápis vypadat *$("#id")*. Pro hledání obrázku použijeme zápis *$("img")*. Knihovna jQuery tedy sjednocuje zápis obou funkcí pro vyhledávání pod jeden název. Zápis je také kratší a kód je díky tomu přehlednější. Navíc do sebe můžeme selektory různě zanořovat. Například zápis *$("#nav li > a")* použijeme pro vyhledání všech odkazů umístěných přímo pod složkami odkazů (tedy jejich přímých potomků) v elementu s identifikátorem *nav*. Nad takto vybranými elementy můžeme přímo spouštět metody pro změnu elementů (například elementy skrýt).

Knihovna jQuery je vytvořena tak, aby umožnila řetězit její metody. Každá metoda tak vždy vrací elementy, aby se s nimi mohlo dále pracovat. Díky tomu je samozřejmě kód kratší a přehlednější. Tento přístup má ale i další výhodu. Elementy vybíráme z modelu DOM pouze jednou, a pak na ně aplikujeme metody knihovny jQuery [10]. To snižuje dobu zpracování, protože nemusíme model DOM procházet stále znovu. Tento prvek je důležitý pro optimalizaci výkonu celé aplikace.

Při vyhledávání elementů *div,* pomocí knihovny jQuery, dostaneme jako výsledek všechny elementy div, které se na stránce nacházející. Nikoli jen první nalezený element *div*. Výsledek vyhledávání pak dostaneme jako skupinu elementů *div* obalenou „funkčností“ knihovny jQuery. První výhodou je, že nemusíme ošetřovat stav, kdy používáme metodu na výsledek vyhledávání, který neobsahuje žádný element. Metoda jQuery pak tiše selže a my se tak vyhneme použití příkazu *if*. Druhou výhodou je automatická iterace prvků. Například metoda sloužící pro skrytí prvku na stránce *hide* se bude volat pro každý prvek skupiny. Nemusíme tak psát cyklus procházející celou skupinu elementů a aplikující metodu na každý element zvlášť.

2.5 Systémy pro správu verzí

Systémy pro správu verzí slouží jako úložiště zdrojových kódů a dalších důležitých dat vztahujících se k danému projektu. Uchovávají verze souborů a usnadňují práci více vývojářů na jednom projektu [13]. Většina systémů pro správu verzí používá tzv. větve (branches). Větve rozdělují projekt do více částí, na kterých vývojáři mohou pracovat samostatně.

Systémů pro správu verzí je celá řada. V této práci si uvedeme jen tři, které jsou podle mého názoru nejzákladnější:

- CVS
- SVN
- GIT

Nejstarším systémem je CVS (Concurrent Version System). Jde o pomyslné rozšíření RCS (Revision Control System) [14]. CVS verzuje jednotlivé soubory zvlášť, bez ohledu na ostatní. Soubory mohou být uloženy v různých adresářích. Adresář v tomto systému není brán jako soubor a to má špatný vliv na celý systém. V CVS má adresář speciální postavení, soubory se ukládají a organizují podle toho, v jakém adresáři se nacházejí. To znamená, že adresář nejde dost dobře smazat, vznikají zbytečné problémy při přesouvání souborů apod. Proto by měl adresář, pokud možno, co nejvíce vystupovat jako obyčejný soubor, u kterého si pouze poznamenáme, že zároveň obsahuje i nějaké jiné soubory. Měli bychom pak možnost adresáře přesouvat, přejmenovávat, vytvářet a mazat, stejně jako všechny ostatní soubory. Další funkcí CVS je spravování a slučování větví. Provádí to ale neefektivním programem *diff3,* který po sloučení větví zapomíná informace o vedlejších větvích [15].

SVN (Subversion) nabízí již lepší funkčnost. Hlavní změnou, jak uvádí [16], je nakládání s adresářem. SVN na rozdíl od CVS nerozlišuje rozdíl mezi adresářem

a souborem. Také lépe zvládá větvení projektu. Jeho nevýhodou, oproti systému GIT, je rychlost práci s daty.

Jako třetí systém pro správu verzí představíme GIT. Tento program je napsán v jazyce C. Podle [17] byla při vývoji GITu rychlost běhu hlavním cílem. V tabulce 2.1 převzaté z [17] je uvedeno porovnání rychlosti běhu GIT a SVN. Z tabulky je zřejmé, že GIT pracuje ve všech testovaných případech rychleji. Porovnání je pouze přibližné, jak sám autor testů uvádí. Také GIT umí dobře pracovat s větvemi, dokáže je vytvářet i slučovat. Po sloučení větví nedochází ke ztrátě dat o vedlejší větvi.

V této práci používáme systém GIT pomocí služby github. GITHUB je služba založená na systému GIT, kterou je možné zdarma používat pro veřejné projekty. Celý projekt je pak veřejně přístupný na webu github.com i se zdrojovými kódy.

Operation		Git [s]	SVN [s]	
Commit Files (A)	Add, commit and push 113 modified files (2164+, 2259-)	0.64	2.60	4x
Commit Images (B)	Add, commit and push 1000 1k images	1.53	24.70	16x
Diff Current	Diff 187 changed files (1664+, 4859-) against last commit	0.25	1.09	4x
Diff Recent	Diff against 4 commits back (269 changed/3609+,6898-)	0.25	3.99	16x
Diff Tags	Diff two tags against each other (v1.9.1.0/v1.9.3.0)	1.17	83.57	71x
Log (50)	Log of the last 50 commits (19k of output)	0.01	0.38	31x
Log (All)	Log of all commits (26,056 commits - 9.4M of output)	0.52	169.20	325x
Log (File)	Log of the history of a single file (array.c - 483 revs)	0.60	82.84	138x
Update	Pull of Commit A scenario (113 files changed, 2164+, 2259-)	0.90	2.82	3x
Blame	Line annotation of a single file (array.c)	1.91	3.04	1x

Tabulka 2.1: porovnání rychlostí systémů Git a SVN [17]

2.6 Knihovny a frameworky

V ukázce použití práce byly využívány následující frameworky a knihovny:

- framework Bootstrap
- PHP Form Builder Class
- jQuery Tokeninput
- jQuery Nivo Slider

Bootstrap [18] je front-end framework využívající HTML 5, CSS 3 a JavaScript od společnosti Twitter. S jeho využitím můžeme vytvářet velmi poutavé weby ve velmi krátkém čase. Bootstrap přestylovává všechny základní HTML elementy a přidává mnoho dalších předstylovaných tříd. Velmi rychle se přidává do stránky. Pouze stačí do stránky přidat CSS a JavaScriptový dokument. Je známý, a proto se můžeme setkat s řadou programů, které tento framework podporují a to od jednoduchých generátorů používaných barev až po např. automatický generátor formulářů ve frameworku Nette. Další velkou výhodou Bootstrapu je rozsáhlá a podrobná dokumentace. Ta je rozdělena do těchto kategorií:

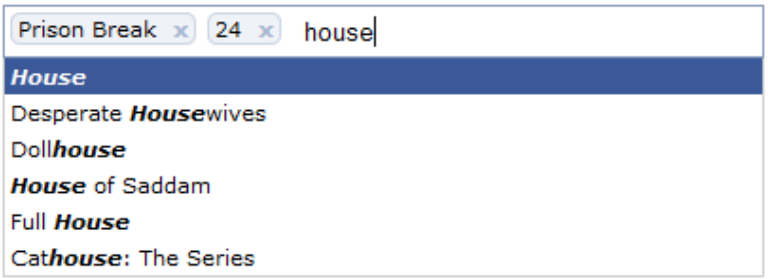
- Základní seznámení s frameworkem a ukázky webových stránek, které v něm byly vytvořeny.
- Jak si framework stáhnout a nainstalovat.
- Dokumentace bootstrapu, která popisuje základní a rozšířené stylování a vkládání předpřipravených ikon až po využití předpřipravených JavaScriptů.
- Nastavení, které části frameworku chceme stáhnout, aby se tak docílilo co nejmenší velikosti stažených souborů.

Jelikož je dokumentace takto detailně zpracovaná, je výhodné framework používat na veřejných projektech (open source projektech), protože se v něm každý rychle zorientuje, popřípadě rychle vyhledá, co potřebuje.

Bootstrap nabízí kompletní systém pro vytvoření grafiky stránek od layoutu až po drobná tlačítka. Podporuje také zobrazení pro mobilní zařízení a pro tablety. Často se používá v kombinaci s vlastními styly, kdy použijeme Bootstrap jako základ stránky a pak stylujeme jen jednotlivé části. Také nám nabízí 140 ikon, které můžeme v projektech volně využívat. Kromě HTML a CSS můžeme použít také předpřipravené JavaScripty. Můžeme tak využívat nastylovaná vyskakovací okna, varovné hlášky a mnoho dalšího. Bootstrap nemusíme stahovat celý, ale můžeme stáhnout jenom části, které zrovna potřebujeme využívat. Tím se zmenšuje jeho velikost a to nám zajistí rychlejší načítání stránky.

PFBC (PHP Form Builder Class) [19] je PHP framework, sloužící pro vytváření formulářů. Využívá technologií HTML 5. Pro vzhled pak využívá frameworku Bootstrap. Dále se dá ve formulářích využít například jQuery UI, TinyMCE a mnoho dalších. PFBC je napsán objektově orientovaným přístupem. Vytvoříme si tedy objekt Form a pomocí metod k němu přidáváme formulářové elementy. Ty pak mohou mít vlastní validační podmínky, jako je například validace e-mailu. Jelikož formuláře lze propojit s jQuery UI, najdeme ve formuláři i element kalendář.

Tokeninput [20] je jQuery plugin, který umožňuje uživatelům vybrat více položek z předem definovaného seznamu. Váže se k formulářovému poli *input* typu *text*, ke kterému vytváří nápovědu možných hledaných výrazů, v závislosti na již vyplněném řetězci. Uživatelé mohou využívat automatické dokončování hledaného výrazu. Podobnost uživatelem hledaného výrazu s databází slov plugin neporovnává jen od počátku řetězce, ale hledá i podobnosti s jeho částmi. Uživatel tak může zadat jen prostřední část řetězce a plugin mu správně nabídne jeho doplnění. Plugin také podporuje vybírání více hledaných položek v seznamu. Uživatel tak může hledat najednou například město Praha a město České Budějovice. Všechny hledané položky bude mít přehledně vložené přímo ve vyhledávacím poli, ze kterého si je pomocí myši snadno odstraní. Do nápovědy hledaných výrazů jdou také vkládat ikony a malé obrázky. Uživatel se tak může v hledaných výsledcích rychleji zorientovat. Podobný typ zadávání textu můžeme najít při vyplňování seznamu příjemců při odesílání zpráv na Facebooku. Na obrázku 2.1 vidíme ukázku pluginu Tokeninput pro více položek vyhledávání.



Obrázek 2.1: použití pluginu Tokeninput pro více položek [20]

Nivo Slider [21] je jQuery plugin, který vytváří obrázkové galerie. Pro snadné listování mezi obrázky obsahuje nejen šipky, ale také lištu, ve které jsou zobrazené body znázorňující jednotlivé obrázky. Uživatel se tak může jedním kliknutím vrátit o několik

obrázků zpět, popřípadě se může díky liště v obrázcích dobře orientovat. Lišta zobrazuje, kolikátý obrázek si uživatel prohlíží a umožňuje se k němu snadno vrátit. Nivo Slider obsahuje 16 grafických přechodů mezi obrázky. Mezi přechody můžeme najít například i změnu obrázku po jednotlivých dlaždicích, které se mění každá samostatně. Také nabízí několik grafických provedení, které můžeme využít při zobrazení na naší stránce. Je poměrně malý. V komprimované verzi má okolo 15 KB. To je výhodné kvůli rychlosti načítání webové stránky.

3 Relevantní práce

Relevantní prací, kterou jsem našel, je rezervační systém pro autobus [22]. Systém je napsán v jazyce javascript s použití knihovny jQuery. Tato práce není psaná jako plugin, ale jde pouze o script s použitím knihovny jQuery. Systém může sloužit i pro rezervaci v kině, divadle, restauraci apod. Má pouze omezené nastavení vygenerovaných sedadel. V systému si můžeme nastavit jen počet řad sedadel a počet sedadel v každé řadě. Z konfiguračního souboru tak například nemůžeme vložit uličku, nebo vynechat nějaké sedadlo. Struktura sedadel tedy bude mít vždy obdélníkový charakter.

Sedadla mají tři stavy: volné, vybrané a zarezervované sedadlo. K těmto sedadlům lze přiřadit vlastní CSS třídy v nastavení. Tím můžeme měnit vzhled jednotlivých sedadel. Systém dokáže zobrazit všechna zarezervovaná sedadla, nebo právě vybraná. Zobrazuje je ve formátu čísel sedadel oddělených čárkou.

Tento systém můžeme použít pro vytvoření velmi jednoduchého rezervačního systému. Při požadavcích na složitější rozložení sedadel, tvorbu uliček, či dvojmístná sedadla, však nebude stačit. Jelikož nejde o plugin, zdrojový kód se může v každé stránce lišit podle toho, jak moc ho programátor upraví. To způsobí, že na více projektech budou různé verze systému a nebudou vzájemně kompatibilní. To může být problém hlavně při udržování systému různými programátory na více webových serverech, nebo při snaze použít jednotný systém pro více webů. Na obrázku 3.1 vidíme použití systému pro rezervační systém autobusu.

Obrázek 3.1: ukázka použití rezervačního systému pro autobus [22]

4 Tvorba pluginu jQuery

V této kapitole se budeme zabývat nejdůležitějšími problémy, se kterými jsem se při psaní pluginu setkal. Zároveň zde popíšeme, jak jsem je v práci řešil. V této kapitole se zaměříme na nejlepší techniky (tzv. best practices) při tvorbě jQuery pluginu. Představíme si největší a nejčastější problémy, které při tvorbě pluginu vznikají a jejich řešení.

4.1 Vytvoření pluginu a připojení ke knihovně

Knihovna jQuery nabízí velmi jednoduchý způsob, jak k ní připojovat další funkce. Stačí rozšířit objekt *jQuery.fn*. Tedy například u pluginu pro rezervační systém ukládáme do objektu *jQuery.fn.booking_system* všechny potřebné vlastnosti pluginu. Ukázku vidíme ve zdrojovém kódu 4.1.

```
$.fn.booking_system = function(options){
    /* zde je vše potřebné pro běh pluginu */
}
```

Kód 4.1: připojení pluginu ke knihovně

K vytvořenému pluginu pak přistoupíme jednoduše tak, jako by to byla jakákoliv jiná funkce jQuery. V našem případě to bude *jQuery.booking_system*.

4.2 Použití obalové funkce

Při používání knihovny jQuery [10], se často využívá u přístupu k objektu jQuery jeho zkrácená verze zápisu $. Tento přístup, který používáme i v této práci, je výhodný z důvodů:

- Kód je kratší.
- Kód je čitelnější.

Kód je díky tomuto zápisu kratší. Jelikož se javascript stahuje spolu s načtením stránky a teprve potom je spuštěn v prohlížeči, je kladen velký důraz na to, aby byl kód co nejkratší. Dle [23] je díky tomu zajištěna rychlejší odezva stránky mezi začátkem stahování stránky a mezi tím, než stránka začne plně reagovat.

Kód je čitelnější. Jelikož se objekt jQuery používá v pluginu opravdu často, zkrácený zápis je v mnoha případech čitelnější.

Protože je používání zkráceného zápisu oblíbené, může přinést řadu problémů. Například knihovna, která je přidána ke stránce před tímto pluginem, může zkrácený zápis $ využívat jako ukazatel na něco jiného, než objekt jQuery. Z tohoto důvodu použijeme v práci obalovou funkci. Zdrojový kód obalové funkce je uveden v ukázce zdrojového kódu 4.2.

```
(function($) {
   /* celý kód pluginu */
})(jQuery);
```

Kód 4.2: obalová funkce

Je to velmi jednoduché řešení, kdy definujeme obalovou funkci, ve které se nachází celý plugin. Její parametr bude $. Přímo za dodefinováním funkce ji zavoláme, a jako parametr vložíme objekt jQuery. Jazyk JavaScript pak vymezí rozsah platnosti této proměnné na celou funkci. Ve funkci potom můžeme $ libovolně používat bez vzniku kolize. Další výhodou tohoto postupu je, že definováním obalové funkce vytvoříme uzávěr. Definici uzávěru uvádí [10]. Uzávěr nám pomůže s možnou kolizí názvů metod tohoto a ostatních používaných pluginů.

Jazyk JavaScript si dokáže poradit se zapomenutým středníkem na konci řádku [10]. K problému dochází v případě minimalizace kódu. Konce řádků se vypouští a může tak docházet k chybám. Proto je důležité na začátku pluginu napsat středník. Tím ošetříme případ, kdy programátor, který psal předcházející kód, na středník zapomněl.

4.3 Defaultní parametry pluginu

Nejjednodušším řešením je přidávat nastavení pluginu, jako parametr funkce. Toto řešení ovšem nebude stačit v případě, chceme-li správně nabídnout uživateli pluginu defaultní možnosti.

V tomto případě je dobré definovat si vlastní defaultní hodnoty a připojit je k pluginu. Pak jednoduše pomocí metody *extend* porovnáme defaultní nastavení a nastavení od uživatele. Údaje, které uživatel neuvedl, metoda sama doplní defaultními údaji.

```
/* doplneni nevyplnenych nastaveni o deafultni nastaveni */
var opts = $.extend({}, $.fn.booking_system.defaults, options);
```

Kód 4.3: použití funkce extend

Výhodou tohoto přístupu je čitelnější kód. Pluginy většinou mívají hodně proměnných, které je potřeba nastavit. Tímto přístupem se vyhneme kontrolování, zda došlo k nastavení všech proměnných. Jednoduše kontrolujeme jen to, co potřebujeme. V ukázce zdrojového kódu 4.3 je vidět, jak se pomocí funkce extend načítá defaultní nastavení a nastavení, které si zvolil uživatel.

```
$.fn.booking_system.defaults = {
    /* formát vstupu */
    format: "cvs"
    /* další nastavení ... */
};
```

Kód 4.4: připojení defaultního nastavení k pluginu

V druhé ukázce zdrojového kódu 4.4 je samotné připojení defaultního nastavení k pluginu. Ve skutečnosti je nastavení obsáhlejší. Zde si pro zjednodušení ukazujeme pouze jedno konkrétní nastavení a to vstupní formát.

4.4 Private funkce a načítání dat ze souboru

Při psaní pluginu potřebujeme napsat hodně funkcí, které nebudou viditelné pro uživatele pluginu. Kdybychom je připojili k pluginu, mohl by uživatel tyto funkce překrýt. Tento problém řeší již zmíněná obalová funkce. Uvnitř obalové funkce si definujeme funkci, kterou jsme nepřipojili k objektu *jQuery.fn*. Tím zůstane funkce navenek skrytá, ale můžeme ji využívat uvnitř pluginu.

Při načítání dat ze souboru je možné použít Javascript a knihovnu jQuery. Slouží k tomu funkce *get*, která se volá nad objektem jQuery. V případě tohoto pluginu je použitá funkce *file_get_contents*, která je v jazyce PHP. Toto řešení jsme zvolili proto, že načtení souboru vytvořeného uživatelem může být zdrojem mnoha chyb. Například daný soubor neexistuje, nebo data byla načtena ve špatném formátu. Zvolili jsme řešení načítat soubor do stránky, kde se dá jednoduše v náhledu zdrojového kódu stránky ověřit, zda se data opravdu data načetla a jsou správná.

4.5 Formáty vstupu

Pro vstupní data musíme zvolit vhodný formát, ve kterém se data dají uložit do souboru. Tento formát by měl být známý, aby si uživatel mohl snadno vyhledat, jak do formátu svá data převést.

Pro načtení dat ze souboru jsme si vybrali formát CVS. Uživatel si například navrhne data v MS Excelu a jednoduše je uloží v tomto formátu. Při práci s tímto tabulkovým procesorem byl doplněn ještě jeden podporovaný formát dat. Je jím XLS. Zjistil jsem, že při ukládání dat přímo ve formátu XLS, (nikoliv v XLSX) jsou data přehledná. Uživateli, který používá MS Excel, se tak bude s daty lépe pracovat.

Podporovaným formátem je i JSON. Jde o velmi používaný formát, a proto jsme ho také začlenili do podporovaných formátů. V ukázce načítáme data pro tento formát z databáze, a pak je převádíme do formátu JSON. Ukázka tak naznačuje, jak se dá pracovat s daty z databáze. Převod je totiž, díky dobré podpoře formátu jazykem PHP, snadný. S daty v databázi se lépe pracuje a myslím si, že načítání dat z databáze by ukázka určitě měla obsahovat.

4.6 Standardy pro tvorbu komentářů

Komentáře jsou důležitou součástí každého softwaru. Autorovi i dalším programátorům, kteří budou kód používat, umožňují lepší orientaci. V této práci jsou použity různé

druhy dokumentačních postupů. Postupy jsou závislé na tom, ve kterém jazyce jsou komentáře napsané.

Při psaní Modelu databáze jsou použity komentáře PhpDoc (celým názvem PhpDocumentor). Tento způsob je popsán ve [24]. Díky PhpDoc komentářům je možné vygenerovat přehlednou dokumentaci celého kódu. Používat lze nejen jednoduché texty, ale i tagy, jako je například seznam. Dokumentaci potom můžeme vygenerovat v různých formátech, jako je HTML, PDF či CHM. Další velkou výhodou těchto komentářů je, že se dají propojit s některými vývojovými prostředími (například s vývojovým prostředím Eclipse). Při použití zdokumentovaného kódu může vývojové prostředí našeptávat komentáře, jako nápovědu.

U samotného pluginu se snažíme co nejvíce přiblížit komentářům Jsdoc, které jsou popsané ve zdroji [25]. Z komentářů, napsaných tímto způsobem, se dá rovněž vygenerovat dokumentace ve formátu HTML.

4.7 Použitelnost

Při psaní pluginu bylo nutné dbát na možnost jeho použití u různých rezervačních systémů. Je to zřejmé na příkladech kina a squashového kurtu, které jsou vytvořeny, jako ukázky používání pluginu.

Důležité bylo vyhnout se přímému stylování objektů. V pluginu generujeme celý systém přímo do stránky a nesmí zde být uveden žádný přímý styl. Přímé stylování by znemožnilo použitelnost pluginu na jakýkoliv jiný případ, protože má nejvyšší prioritu vytváření vzhledu objektu a nešlo by překrýt. Všechny styly jsme tedy přesunuli do vnějšího souboru, kde si je uživatel pluginu může lehce změnit.

Dalším důležitým prvkem u stylů bylo nechat názvy tříd a identifikátorů volitelné. Mohlo by totiž dojít k tomu, že by kolidovaly s již použitými názvy na stránkách uživatele. Ten by pak musel své stránky, nebo plugin ručně předělat. Aby uživatel nemusel nastavovat všechny třídy ručně, při přidávání pluginu do své stránky, jsme ponechali všem třídám a identifikátorům výchozí hodnoty. Uživatel si tak změní jen ty názvy, které mu na stránkách kolidují.

Jelikož plugin potřebuje pro svůj správný běh uvést cestu k obrázkům, které používá (jako je třeba obrázek sedačky v kině), i tato cesta zůstala v pluginu nastavitelná pro všechny obrázky.

Při psaní pluginu bylo nutné dbát na jeho vizuální stránku. Při najetí kurzorem myši nad objekt, či kliknutí na objekt, se spouští různé akce. To objekt graficky mění. Například ho zprůhledňuje. Uživatel může s objektem spojit libovolnou jinou akci. Stačí v nastavení pluginu zadat jméno funkce, která defaultní funkci v pluginu překryje. Více o této problematice je zařazeno v kapitole Obecná použitelnost, protože uživatel si může napsat funkce nakládající s objektem, jak potřebuje. Jedinou podmínkou pro uživatele je, přijímat objekt *events*, jako parametr funkce, přes který se dostane k objektu, se kterým je manipulováno.

4.8 Výstup pluginu

Výstup z pluginu se přímo vypisuje do HTML elementu, podle daného identifikátoru id. Výpis je možný s použitím elementů table (tabulka) nebo div.

Výstup pomocí tabulky je výhodný v tom, že data se sama řadí do mřížky. Nemusíme je jakkoliv ovlivňovat styly. Projeví se to obzvláště u generování více dat, která se pak řadí do řádků a sloupců. Toho můžeme využít i pro řazení sedadel do řady. K sedadlům stačí přidat patřičné značky tabulky označující začátek řádku (v případě kina řady sedadel) a dalšího prvku (sedadla). Nevýhoda tohoto přístupu je v tom, že elementy nejde rozmístit nepravidelně. Tedy například chce-li uživatel rozmístit elementy do geometrického obrazce (například kruhu), musí jednotlivé elementy vyhledávat a stylovat každý zvlášť. Z pohledu struktury HTML dokumentu by v tabulce měly být pouze tabulková data, nikoliv celé elementy. Proto je tento přístup nevhodný.

Druhá možnost je ovlivnit rozmístění objektů pomocí značek div. V tomto případě každý element div obsahuje vlastní objekt (v případě kina sedadlo) a uživatel může objekt přesunout, kam potřebuje. Nevýhoda tohoto přístupu je zřejmá. Elementy div se automaticky neřadí do mřížky. Proto je nutné si pro ně vytvořit vlastní kaskádové styly.

V pluginu používáme značky div. Vytváříme skupiny objektů. V nich se jednotlivé objekty nachází. V případě kina si je můžeme představit, jako řady sedadel. V kulečníkové herně by to byly stoly. V této skupině se pak objekty mohou stylovat libovolně. Například u kulečníku je můžeme umístit okolo stolu do oválu. Tento případ by se využitím tabulky řešil velmi špatně.

5 Realizace pluginu pro rezervační systém

V této části se již budeme zabývat samotným provedením pluginu. Oproti předchozí kapitole budeme více konkrétní a zaměříme se právě na vytvářený plugin. Dozvíme se zde více o realizaci pluginu i jeho ukázky. Na závěr si ukážeme použití rezervačního systému na dvou konkrétních příkladech.

5.1 Použití frameworku bootstrap

Bootstrap je front-end framework od Twitteru. Po vložení do kódu stránky přestyluje všechny základní HTML elementy. Také obsahuje mnoho předpřipravených tříd. Vývojář díky tomu může psát pouze HTML kód. Bootstrap také nabízí řadu JavaScriptových funkcí, jako je například vytváření stylovaných vyskakovacích oken, nebo různé práce s textem. Více informací o Bootstrapu je uvedeno v kapitole Použité technologie.

Při vytváření ukázky bylo nutné rozhodnout, zda použít pro stylování některý z dostupných frameworků, nebo zda si vytvořit vlastní styly. Jedním z hlavních kritérií rozhodování byla dobrá dokumentace. Ukázka pluginu má být intuitivní a uživatel by neměl přemýšlet nad ukázkou samotnou. Měl by se zaměřit hlavně na použití pluginu, aby ho mohl co nejdříve využívat na svém webu. To jsou důvody, proč bylo vhodné zvolit bootstrap framework. Má velmi kvalitní dokumentaci všech elementů. V dokumentaci je nejen popis stylování, či javascriptu, ale také ukázka jejich použití. Uživatel si tedy může velmi rychle vyhledat informace o tom, s čím právě pracuje.

Z frameworku jsou v práci využity převážně předstylované HTML tagy. Ty jsou použity na celkový vzhled ukázky a ve většině náhledů. Na stylování samotné stránky, kde je použit rezervační systém, jsme museli přidat i vlastní styly. Bylo nutné ovlivnit konkrétní věci, jako je třeba řazení a pozice sedaček v kině. Díky použití vlastních stylů jen na konkrétní výjimečné situace jsme dosáhli toho, že vlastní styly jsou velmi krátké a uživatel se v nich lépe vyzná.

5.2 Databázová vrstva – realizace s využitím PHP

Při vytváření ukázky použití pluginu bylo důležité dobře realizovat přístup k databázi. Přístup je realizován přes model databáze. Ten obsahuje metody psané přímo pro konkrétní výběry dat, či jejich ukládání. Model disponuje obranou proti útoku SQL injection. Obsahuje konfigurační soubor pro centralizaci dat, která jsou potřebná při práci s databází.

Celý přístup do databáze probíhá přes jediný objekt *BookingSystemDatabase*. Z pohledu MVC (Model View Controller) architektury je připojení k databázi považováno právě za model. Ten zajišťuje veškerou komunikaci s databází. Objekt *BookingSystemDatabase* se musí před prvním použitím vytvořit a inicializovat.

Při tvorbě ukázky byl celý objekt nejdříve vytvořen podle návrhového vzoru jedináček. Tedy konstruktor byl soukromý a měl privátní statickou metodu, která vracela novou, nebo již vytvořenou instanci. Výhodou bylo, že stačilo přes metodu vrátit objekt. Uživatel se nemusel starat o to, zda již připojení někde v kódu použil a je tudíž vytvořené. Uživatel tak měl vždy jeden objekt, ke kterému mohl přistupovat. Toto řešení není vhodné z důvodu, že ukázka má být v první řadě jednoduchá a dobře čitelná. Při použití modelu jedináček taková nebyla. Uživatel se totiž musel seznámit s napsáním modelu, aby pochopil, jak přistupovat k objektu. Objekt si v ukázce vytvořit nemohl, protože konstruktor byl podle návrhového vzoru privátní. Proto vytváříme objekt přes konstrukci *new*. Vytvoření a inicializaci objektu jsme nechali ve zvláštním souboru *booking_system.class.php*, který se musí pro používání databáze do stránky přidat. Tím jsme se snažili dosáhnout také toho, aby uživatel vytvořil objekt právě jednou.

SQL dotazy se skládají přímo v objektu, nikoliv v kódu stránky. K tomu slouží metody, které přímo vykonají SQL dotaz a popřípadě vrátí výsledek. Metody jsou konkrétní a každý nový druh přístupu k databázi znamená novou metodu. Například pro přihlášení má objekt metodu *signIn* s parametry email a heslo. Ukázka zdrojového kódu metody je v ukázce 5.1. Metody jsou chráněny před SQL injection. To znamená, že uživatel nemá možnost modifikovat SQL dotaz pomocí předaných dat. Ochrana spočívá v tom, že všechny parametry jsou escapovány funkcí *addslashes*. Ta se používá pouze tehdy, není-li zaplá direktiva *magic_quotes_gpc*, která by escapovala řetězec podruhé. Tento druh ochrany je vhodný hlavně pro hostingy a servery používající starší verze PHP. Metoda je napsaná podle [26].

```php
public function signIn($email, $password)
{
    $email = $this->gpc_addslashes($email);
    $password = $this->gpc_addslashes($password);

    return $this->query(
        "SELECT id FROM ". TABLE_USERS . " WHERE email = '" .
        $email . "' AND password = '" . md5($password) . "'")
            ->fetch();
}
```

Kód 5.1: ukázka funkce pro přihlášení

Objekt obsahuje metody na přihlášení a odhlášení k databázi. Tyto metody nemají žádné parametry. Potřebné hodnoty se inicializují už při vytváření objektu. Metody jsou veřejné, aby si je uživatel mohl volat přímo v kódu. Další metody jsou *query* a *fetch*. Tyto metody jen obalují funkce *mysql_query* a *mysql_fetch_object* a řeší případné výjimky.

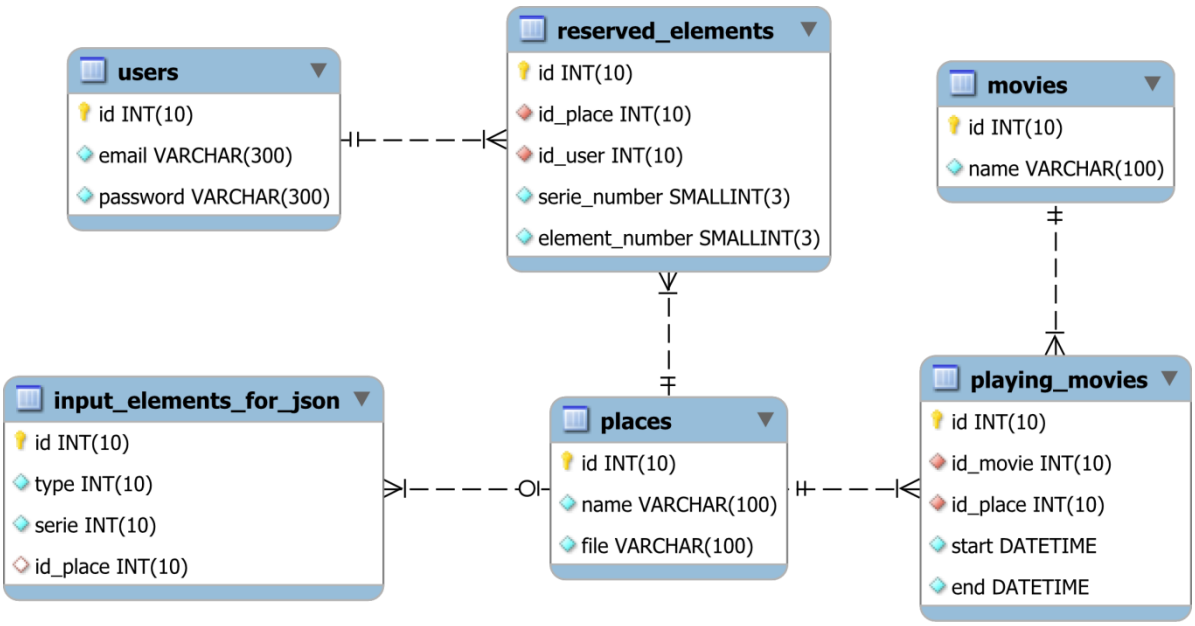
```php
public function getJSON()
{
    $rs = array();
    while($rs[] = mysql_fetch_assoc($this->rows)) {
       // zde se skutečně nemá nic dělat
    }
    return json_encode($rs);
}
```

Kód 5.2: metoda modelu pro vracení výsledků z databáze ve formátu JSON

Metoda, kterou zde podrobněji popíšeme, je metoda *getJSON*. Tato metoda slouží pro výběr dat z databáze a uložení do formátu JSON. Využívá funkci *json_encode*, která slouží k převádění dat do formátu JSON. Tato metoda slouží převážně k ukázce, že plugin může přijímat i data v moderním formátu JSON. Jak můžeme vidět na ukázce 5.2, metoda má velmi zajímavou implementaci. V těle cyklu totiž není vůbec žádný

kód. Cyklus slouží pouze k tomu, aby se uložily výsledky z databáze do pole *$rs*. To cyklus vykoná již v podmínce. Když funkce *mysql_fetch_assoc* nemá další data, která by mohla převést do asociativního pole a uložit do proměnné *$rs*, vrátí hodnotu FALSE a cyklus se ukončí.

5.2.1 Diagram databáze



Obrázek 5.1: diagram databáze rezervačního systému

Na obrázku 5.1 vidíme diagram databáze používané v ukázce pluginu. Diagram obsahuje pouze vazby N:1. Databáze je ve třetí normální formě. Všechny tabulky obsahují primární klíče, které jsou zastoupeny maximálně desetimístným číslem. Číslo je každému záznamu přiřazeno automaticky. V následujících odstavcích si popíšeme jednotlivé tabulky a vztahy mezi nimi.

playing_movies – Představuje filmy, které se budou promítat v kině. Sloupce *start* a *end* představují začátek a konec promítání. Sloupec *id_movie* je identifikátorem promítaného filmu. Sloupec *id_place* je identifikátorem sálu, kde se film promítá. Sloupec *id_place* je v ukázce důležitý hlavně proto, aby se podle sálu správně vybralo rozložení sedadel. Tabulka *movies* je propojená s tabulkou *places* pomocí cizího klíče *id_place* vazbou N:1. Dále je propojená s tabulkou *movies* pomocí cizího klíče *id_movie* také vazbou N:1. Tabulka má význam pouze v ukázce použití pluginu pro kino.

V ukázce pro squash se tato tabulka nepoužívá.

movies – Znázorňuje samotné filmy. Jelikož nejde o opravdový systém kina, tabulka kromě svého identifikátoru obsahuje pouze název filmu. Ten je uložen ve sloupci *name*. V případě uvedení této ukázky do praxe, by tato tabulka jistě musela být doplněna o další hodnoty, jako je například délka filmu, jestli se jedná o přístupný film do 18 let a jiné. Je propojená s tabulkou *playing_movies* přes primární klíč.

places – Tabulka obsahuje místa. V ukázce kina tyto místa můžeme chápat jako jednotlivé sály. Tato tabulka se používá hlavně v případě, kdy je nastaven vstup ze souboru. Ve sloupci *file* je uvedeno jméno souboru, ze kterého se má vstup načítat. Každé místo má v ukázce právě jeden vlastní soubor, proto tabulka nebyla rozdělena na dvě. Ve sloupci *name* nalezneme jména jednotlivých míst např. „sál 1“.

input_element_for_json – Slouží pro ukázku načtení dat ve formátu JSON. Data ukládáme do této tabulky a pak je pomocí metody modelu databáze převádíme do formátu JSON. Data, která tato tabulka obsahuje, jsou jednotlivé rezervovatelné elementy. Pro příklad kina to jsou sedačky. Sloupec *type* označuje, zda jde o jednomístnou sedačku, nebo dvojmístnou sedačku. Sloupec *serie* označuje řadu, ve které se sedačka nachází. Ve sloupci *id_place* je uvedeno, kterému sálu daná sedačka náleží. Tato tabulka je přes klíč *id_place* spojená s tabulkou *places*.

reserved_elements – Používá se pro zarezervování elementu. V případě kina si můžeme element představit, jako sedadlo. Kdyby se data načítala pouze z formátu JSON, obsahovala by tabulka pouze identifikátor elementu z tabulky *input_element_for_json*. Protože ale musí existovat i možnost načíst vstup ze souboru, v tabulce musí být informace pro jednoznačné určení elementu. Při načítání dat ze souboru se data neukládají do databáze, ale generují se přímo do stránky. Ve sloupci *serie_number* je informace o sérii prvku. U kina to bude řada sedadel. Sloupec *element_number* obsahuje číslo elementu. Sloupec *id_user* představuje identifikátor uživatele, který si daný element zarezervoval. Spojuje tuto tabulku s tabulkou *users* vazbou N:1. Poslední sloupec označuje místo, ve kterém si uživatel element zarezervoval. Pomocí tohoto sloupce je tato tabulka spojena s tabulkou *places*. Vazba je N:1.

users – Obsahuje informace o uživatelích. Používáme ji nejen při registraci a přihlášení, ale také k identifikaci uživatele, který si daný element zarezervoval. Tabulka je spojena s tabulkou *reserved_elements* pomocí primárního klíče. Ve sloupci *email* je uveden email uživatele. Ve sloupci *password* je hashované heslo uživatele.

5.2.2 Konfigurační soubor pro databázi

V konfiguračním souboru jsou uloženy veškeré konstanty používané v ukázce. Byl vytvořen proto, aby bylo možné vše dohledat a měnit na jednom místě. Nachází se v něm názvy všech tabulek, uložené jako konstanty. To má speciální význam. Díky tomu se dají jednoduše měnit názvy tabulek a program zůstane stabilní. Uživatel chce například změnit název tabulky *movies* na *filmy*. Změnu stačí udělat pouze v tomto konfiguračním souboru a změny se projeví v celé ukázce. Kromě názvů tabulek soubor obsahuje i údaje potřebné pro přihlášení k databázi. V ukázce se jedná o soubor *config.inc.php*.

5.3 Funkce rezervačního systému

V této krátké podkapitole si podrobněji popíšeme základní funkce pluginu pro rezervační systém. Nalezneme zde rozpracovaný popis funkcí spolu s ukázkou jejich provedení.

5.3.1 Vygenerování všech elementů

Po načtení vstupních dat a jejich převedení z různých formátů do dvourozměrného pole, se spustí generování všech elementů do stránky.

Nejdřív musíme projít obě pole. K tomuto průchodu slouží konstrukce *$.each* [27]. Tuto metodu obsahuje přímo knihovna jQuery. Funguje podobně, jako konstrukce *foreach* z jazyka PHP. Prochází pole a vrací jeho jednotlivé prvky. Metoda má dva parametry. Prvním parametrem je pole, které má procházet. Druhým je funkce, která bude pracovat s prvkem pole. Funkci pak můžeme předat prvek pole buď pomocí parametrů, nebo pomocí *this*. Pro procházení druhého rozměru pole tuto funkci používáme ještě jednou. Zde si musíme dávat pozor na zanoření obou funkcí. *This* bude před zanořením do druhé funkce obsahovat jiná data, než po zanoření. Při průchodu druhého rozměru pole už metodě jako parametr předáme *this*.

Metoda *$.each* není zaměnitelná s metodou *each*, která se volá nad selektorem objektu *jQuery*. Obě metody prochází skupinu prvků, ale metoda volaná nad selektorem jQuery má pouze jeden parametr. Tím je funkce zpětného volání. Oproti tomu, metoda *each* volaná nad objektem jQuery, má parametry dva.

V ukázce zdrojového kódu 5.3 vidíme použití konstrukce $.each při procházení dvourozměrného pole.

```
$.each(array_series_elements, function(index2, value){
    // zde bude příprava proměnných pro druhý průchod polem
    $.each(this, function(index1, value){
        // zde bude volání funkce pro vytvoření elementu
    });
});
```

Kód 5.3: použití funkce $.each pro dvourozměrné pole

Při každém průchodu prvního rozměru pole vytváříme novou sérii elementů (v případě kina si sérii elementů můžeme představit, jako řadu sedadel). Do té pak následující elementy umísťujeme. Každá série má vlastní identifikátor id, aby se mohla jednoznačně určit. Série obaluje všechny prvky v ní obsažené pomocí značek DIV.

Průchodem druhého rozměru pole získáme samotné elementy. Při jejich procházení nad každým z nich voláme funkci createElement. Ta podle parametrů rozhodne, o jaký element se jedná (v případě kina o jednomístné, či dvoumístné sedadlo). Element může být prázdný (například ulička). Funkce dále přiřadí elementu pořadové číslo a číslo elementu. Pořadové číslo slouží hlavně pro ukázku kina, kde potřebujeme zobrazit čísla sedadel. Důležité je vědět, že číslo sedadla není číslo elementu. V případě, že se vykresluje ulička, se nesmí pořadové číslo zvýšit. Naproti tomu číslo elementu označuje každý vložený element, včetně uličky. Po vytvoření elementu ho přiřadíme poslední vložené sérii. Poslední sérii najdeme pomocí filtru knihovny jQuery. Pro hledání použijeme jméno třídy série s parametrem *:last*. Nakonec pomocí metody *append* element vložíme do série.

Druhý rozměr pole může obsahovat mimo elementy i popisky jednotlivých sérií. Práce s popisky je podobná, jako práce s elementem. Rozdíl spočívá v tom, že popisek se nesmí číslovat a přiřadí se mu jiná třída. Popisek také nese své jméno, např. v kině to bude řada sedadel „C“.

5.3.2 Přidání a odebrání elementů z vybraných položek a rezervace

Elementy se mohou přidávat a odebírat z vybraných. Vybrané elementy si uživatel může zarezervovat. Pro vybrání elementu na něj stačí klepnout myší. Všechny vybrané elementy jsou označeny třídou *select*. Když uživatel vybere element bez třídy *select*, spustí se funkce pro vybrání elementu. V opačném případě se spustí funkce

pro odstranění elementu z vybraných. Informace o vybraných elementech se ukládají do formuláře pro rezervaci sedadel. Po jeho odeslání se sedadla zarezervují.

Funkce *addToSelected* slouží k označení, že je element vybraný. Spustí se nad obrázkem daného objektu. S obrázkem ale pracovat nechceme, proto se pomocí metody *parent* dostaneme k jeho rodiči. Tím je element DIV, který celý prvek obaluje. V tomto elementu jsou uloženy informace, o jaký element se jedná (číslo elementu a série). Díky nim lze element jednoznačně určit. Tyto informace se pak zapíší do formuláře rezervace sedadel, jako skrytý prvek typu *hidden*. Nakonec se přidá třída elementu *select* a změní se obrázek elementu.

Ke zrušení vybraného elementu slouží funkce *removeFromSelected*. Vrací element do stavu, ve kterém byl před jeho označením. Odstraní třídu *select* a odebere daný element z formuláře elementů připravených k rezervaci.

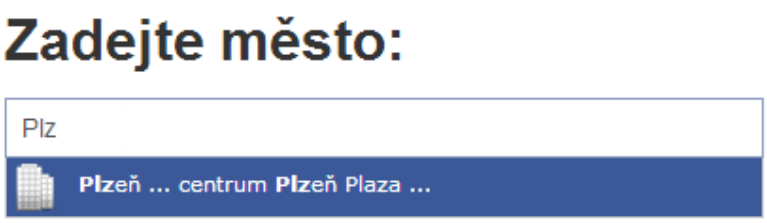
Od zarezervovaného elementu se odpojí události kliknutí myši, přesunutí a odsunutí kurzoru myši z elementu. Takový element pak nejde zarezervovat. Má také jiný obrázek a obsahuje v atributu *data-status* informaci *reserved*. Zarezervované elementy se načítají z databáze a ukládají se do seznamu přímo do HTML stránky. Z ní si pak plugin pomocí funkce *loadReservedElement* všechny data načte. Funkce prochází prvek po prvku a hledá dané elementy. Těm pak přidá informaci o zarezervování do atributu *data-status* a změní jejich obrázek. Změnou *data-status* na *reserved* se odpojí všechny tři výše zmíněné události.

5.4 Popis ukázky

V této podkapitole se již podrobněji zaměříme na ukázku použití pluginu. Ukázka má několik částí, kterými se zde budeme konkrétně zabývat.

5.4.1 Výběr města

Nejdřív musí uživatel v ukázce zvolit město. Možnost volby města je k dispozici pouze v ukázce použití pluginu pro kina. Na této stránce jsme použili plugin *Tokeninput*. Ten reaguje na formulářové pole s textem a nabízí různé nápovědy, jak text doplnit. *Tokeninput* potom pošle proměnnou s názvem města další stránce. Stránka ho použije pro výběr kina v daném městě. Zajímavou vlastností je i to, že *Tokeninput* používá obrázky vedle své nápovědy pro lepší orientaci. Více se o tomto pluginu dozvíme v kapitole Použité technologie. Na obrázku 5.2 vidíme ukázku našeptávání.



Obrázek 5.2: ukázka pluginu Tokeninput

5.4.2 Výběr filmu a sálu

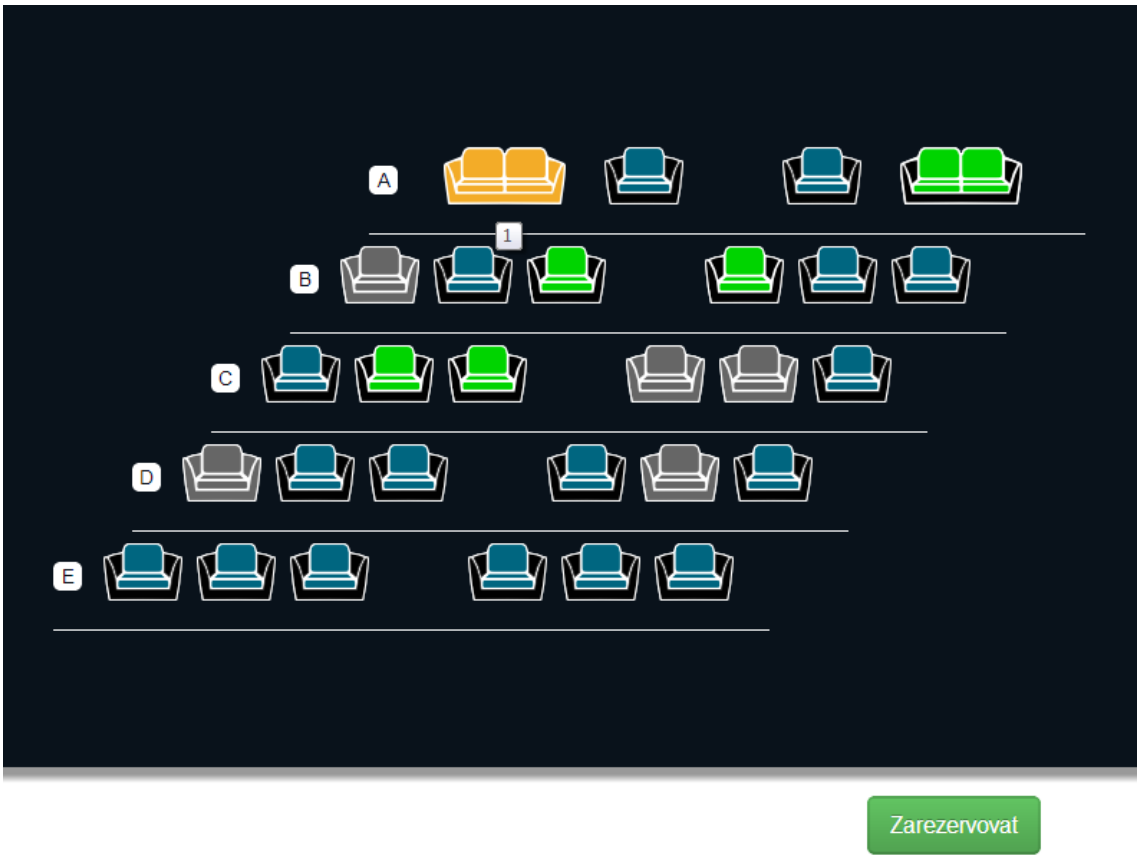
Na této stránce je použita galerie obrázků. Tu zajišťuje plugin *Nivo Slider*, který umožňuje jednotlivé obrázky procházet a automaticky galerii promítá. Více o pluginu *Nivo Slider* je uvedeno v kapitole Použité technologie. Dále na stránce najdeme seznam všech právě promítaných filmů. U jednotlivých filmů jsou pak uvedeny informace o tom, kdy se začínají promítat a čísla sálů, ve kterých se promítají.

Výběr sálu probíhá spolu s výběrem filmu. Pro lepší ukázku možností pluginu je výběr sálu umístěn i přímo do stránky, kde se plugin pro rezervační systém používá. Změnu sálu pak zajišťuje formulář s výběrem sálu, pomocí konstrukce *select*. Pro snazší práci je formulář generován pomocí *PHP Form Builder Class*. Jde o nástroj sloužící k objektovému návrhu formuláře. Využívá HTML 5 a Ajax. Více se o *PHP Form Builder Class* dozvíme v kapitole Použité technologie.

5.4.3 Výběr místa

Výběr místa uživatel najde na stránce rezervace. Výběr místa zajišťuje plugin pro rezervační systém. Plugin při vygenerování všech elementů naváže na každý element několik událostí. První je událost *mouse_over***,** tedy přejetí myší nad elementem. Podle zadání má být kladen důraz na grafické provedení. Proto při přejetí kurzorem myši element plynule mění barvu. Druhá událost je klepnutí myší na element. Ten po této akci opět vyvolá změnu barvy a zároveň zavolá funkci pluginu *addToSelected***.** Funkce má jediný parametr *e* značící událost (event), která nastala. Více o této funkci nalezneme v podkapitole Funkce rezervačního systému. Při kliknutí na označený element se pak spustí funkce *removeFromSelected* se stejným parametrem, který daný element odebere z výběru. Poslední událostí je *mouse_out*. Zavolá funkci, která vrací element do původního stavu po události *mouse_over*. Uživatel si může vybrat tolik míst, kolik potřebuje. Všechny informace rezervační systém ukládá do předpřipraveného

formuláře. Formulář při dokončení výběru odešle. Elementy, které jsou již zarezervované, mají zvláštní obrázek. Není na ně navázaná žádná z výše uvedených událostí a je tak jednoduše ošetřeno, že nejdou rezervovat. Na obrázku 5.3 je zobrazeno rozložení sedadel v kině. Zelené sedačky zobrazují vybraná místa. Modrá barva pak značí místa, která jsou volná a je možné je zarezervovat. Šedá sedadla jsou již zarezervovaná a není možné je nyní rezervovat. Při najetí kurzorem myši nad sedadlo se zobrazí žlutě. Také se u tohoto sedadla zobrazí popisek, ve kterém je uvedeno jeho pořadové číslo. Z obrázku je patrné, že sedačka zobrazuje své číslo. Podrobnější popis a více názorných příkladů najdeme v kapitole Ukázka pluginu.



Obrázek 5.3: zobrazení sedaček v kině

5.4.4 Rušení rezervace

K rušení rezervace slouží vlastní stránka s názvem *Shrnutí*. Na této stránce uživatel najde své rezervace. Ke každé rezervaci se dozví (v případě kina) o jaký sál se jedná a jaké sedadlo si zarezervoval. Ke zrušení rezervace stačí zmáčknout tlačítko *zrušit rezervaci*. Tím se nad daným elementem spustí skript, který vymaže rezervaci z databáze.

Rušení rezervace má pouze ilustrativní charakter. Programátor, který daný plugin použije, může z ukázky čerpat a dále jí rozvíjet podle svých představ. Pravděpodobněji z ní ale použije jen plugin samotný a stránku na zrušení rezervace si napíše sám ve frameworku, který při práci používá.

5.5 Návrh HTML dokumentu pro vložení pluginu

Aby plugin mohl správně fungovat, musí mít stránka určitou strukturu. Tato struktura zahrnuje:

- prostor pro vygenerování grafického rozhraní pluginu
- element pro uložení zarezervovaných sedadel
- formulář pro ukládání a zarezervovaní vybraných elementů
- element sloužící k načítání vstupu pluginu

Nejdůležitější vlastností dokumentu je prostor pro vygenerování grafického rozhraní pluginu. Pro vytvoření tohoto prostoru si zvolíme HTML element (například element DIV) tak, že mu přiřadíme patřičný identifikátor id. Ten samý identifikátor pak musíme pluginu předat v jeho nastavení. Tento element pak ponecháme prázdný a plugin si ho sám naplní.

Pro uložení zarezervovaných sedadel budeme potřebovat další element. V ukázce používáme element DIV. Tomu musíme přiřadit identifikátor *reserved*. Do elementu DIV vložíme HTML značky seznamu. Pak do elementu vygenerujeme všechna zarezervovaná sedadla v daném sále (v případě použití systému pro kino). Jednotlivá sedadla musí být položky seznamu a nést informaci k jednoznačnému určení sedadla. Přesný popis a ukázky jsou uvedeny v programátorské příručce.

Data o vybraných elementech ukládáme do formuláře pro zarezervování elementů pluginu. Formulář vybereme tak, že ho obalíme elementem DIV a označíme ho identifikátorem *selected*. Plugin pak bude do tohoto formuláře generovat vybrané elementy sám.

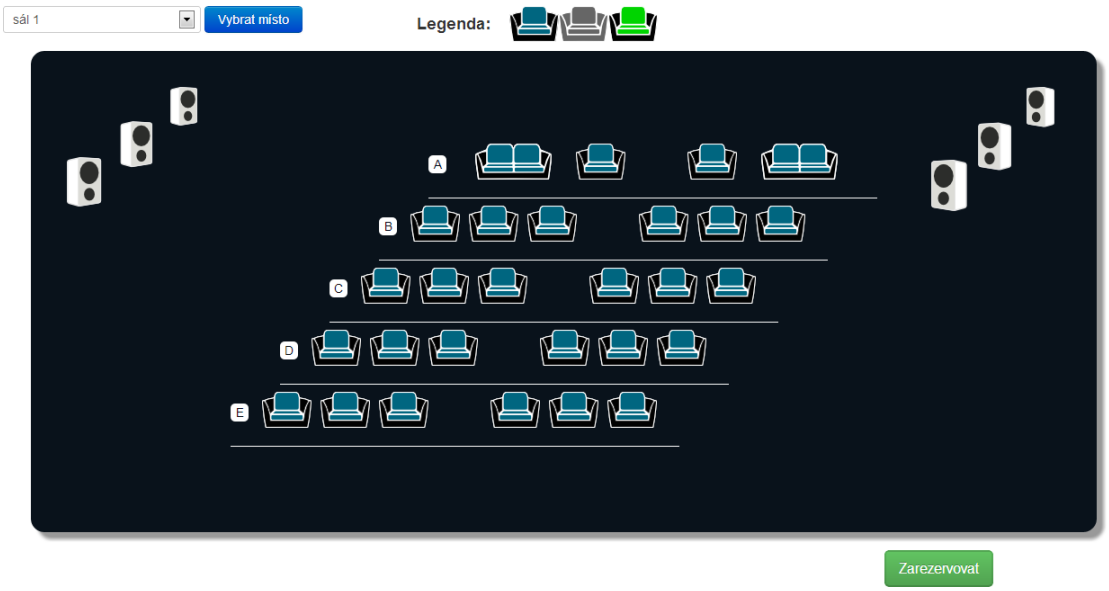
Vstup do pluginu máme uložen ve zvláštním elementu. Element označíme identifikátorem, který si sami zvolíme. V ukázce je element označen identifikátorem *xls* (pro ukázku načtení dat ze souboru) a *json* (pro načtení dat z databáze ve formátu JSON). Pro vstup si musíme vybrat pouze jeden element. Pouze v případě ukázky jsem do práce zařadil dva elementy. Při použití načítání dat ze souboru, do tohoto elementu, přímo načteme obsah souboru. Nemusíme data nijak upravovat, plugin dokáže pracovat přímo s formátem dat obsaženým v souboru. Při použití načtení dat z databáze a uložení ve formátu JSON data upravujeme. V ukázce k tomu slouží metoda modelu databáze. Jde o funkci *loadAllSeatsFromDatabaseJSON*.

5.6 Ukázka vzhledu pluginu pro různé použití

Plugin je navržen tak, aby se dal univerzálně použít pro rezervační systém. Konkrétní rezervační systém se z něj stane až použitím obrázků a různých doplňujících stránek (např. výběr filmu). V následujících podkapitolách si ukážeme použití pluginu pro rezervační systém kina a rezervační systém squashových kurtů.

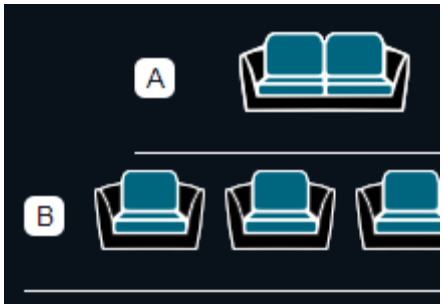
5.6.1 Použití pro kino

V této podkapitole se budeme zabývat vzhledem rezervačním systému kina. Tento vzhled slouží pouze jako ukázka jedné z mnoha možností implementace pluginu pro kino.



Obrázek 5.4: náhled celého sálu

Na obrázku 5.4 je vidět celkový vzhled kina. V náhledu sálu vidíme jednotlivé řady sedaček. Řady sedadel jsou posunuté doprava, pro vytvoření jednoduchého prostorového efektu. Je na programátorovi, zda se rozhodne pro posun řad. Jde pouze o grafický efekt. V levé a pravé části kina jsou umístěny dva obrázky s reprobednami. Jejich použití je opět na programátorovi. V případě, že budeme potřebovat rychle implementovat tento plugin, si stačí stáhnout demo a tyto obrázky vyměnit za jiné.



Obrázek 5.5: popisky jednotlivých řad

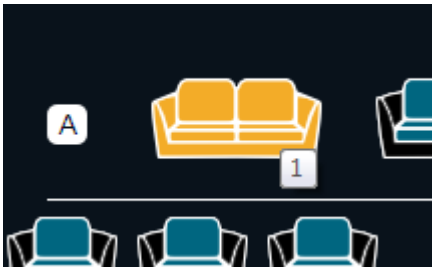
Na obrázku 5.5 vidíme, že každá řada má svůj popisek. Tento popisek nejde v nastavení pluginu vypnout. Pokud si nepřejeme popisek zobrazit, stačí ho nezadat do vstupu a plugin bude vytvářet řady bez popisku.

Obrázek 5.6: zobrazení vygenerované legendy

Obrázek 5.6 ukazuje používání legendy v pluginu. Výhodou používání legendy je, že plugin sám zobrazí obrázky, které používá. Programátorovi tak šetří čas. Generování legendy je možné v nastavení pluginu vypnout.

Obrázek 5.7: ukázka popisků v legendě

Obrázek 5.7 znázorňuje zobrazení popisku. Ten se ukáže při najetí kurzorem myši nad sedačku v legendě.

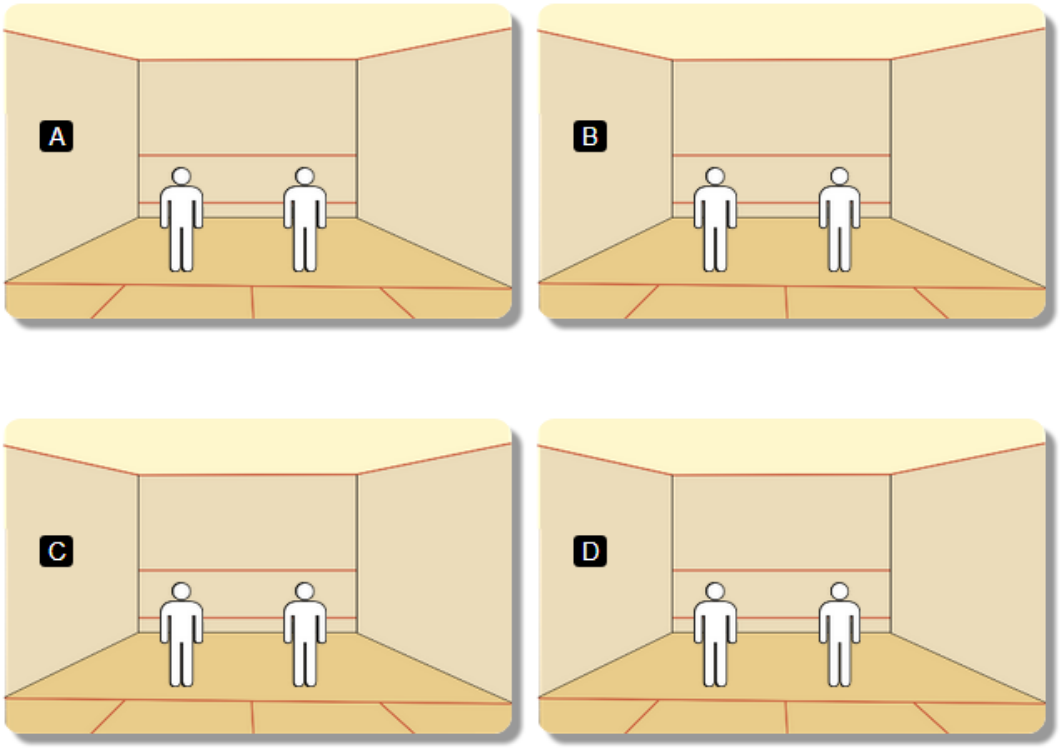


Obrázek 5.8: Ukázka sedadla při najetí kurzorem myši

Obrázek 5.8 ukazuje, co se stane při najetí kurzorem myši nad sedačku v kině. Sedačka plynule změní barvu z modré na žlutou. Žlutá barva sedačky pak zůstane tak dlouho, dokud nepřesuneme kurzor myši mimo sedačku. Dále se objeví pořadové číslo sedačky.

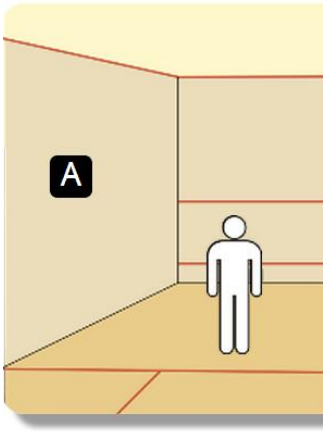
5.6.2 Použití pro squash

Další ukázkou použití pluginu je rezervační systém squashe. Na obrázku 5.9 vidíme, že systém lze použít i pro více kurtů na jedné stránce. Při použití rezervačního systému pro kino představovala sérii řada sedadel. Při použití systému pro squash představuje každá série právě jeden kurt. Každý kurt pak obsahuje vlastní elementy. Elementy jsou hráči na jednotlivých kurtech. Oproti ukázce systému kina zde není vygenerována legenda. Nebyla potřeba, protože obrázek je intuitivní. V nastavení pluginu byla vypnuta.



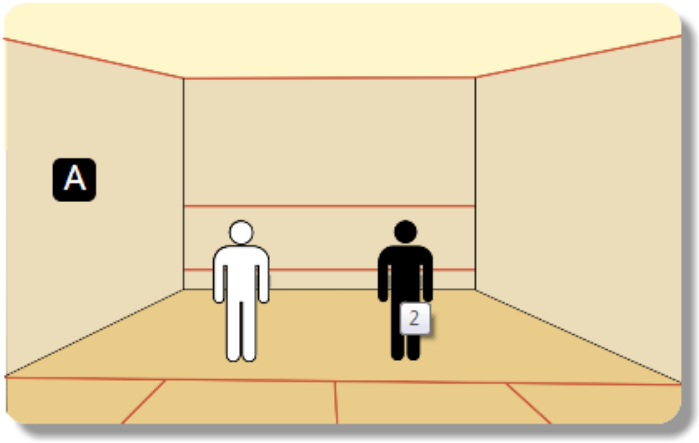
Obrázek 5.9: ukázka zobrazení několika kurtů na jedné stránce

Na obrázku 5.10 vidíme zobrazení popisku každého kurtu. Popisek představuje písmeno „A“ v černém obdélníku. Jde o zvláštní značku zadanou přímo ve vstupních datech pluginu. Pokud si značku nepřejeme zobrazit, stačí ji opět nezadat do vstupních dat.



Obrázek 5.10: ukázka zobrazení popisku kurtu

Na obrázku 5.11 vidíme, jak se změní obrázek hráče, po najetí kurzorem myši (pravý hráč). Hráč změní barvu a zobrazí se u něj popisek.



Obrázek 5.11: ukázka změny hráče při najetí kurzorem myši

6 Testování

Plugin se spouští v prohlížeči na straně klienta. Proto bylo nutné testovat aplikaci na různých prohlížečích. Knihovna jQuery podle [28] podporuje prohlížeče Internet Explorer od verze 6 a vyšší. To platí pro knihovny 1.x. V projektu využíváme knihovnu této verze, můžeme tedy očekávat, že plugin alespoň v omezených možnostech bude v prohlížeči fungovat správně. Testování jsem provedl v následujících prohlížečích:

- Google Chrome verze 26.0.1410.64 m
- Fire Fox verze 19.0.2
- Internet Explorer verze 10.0.9200.16540

Prohlížeče Google Chrome a Fire Fox spustili plugin dle očekávání. Rezervační systém na nich běží bez problémů. Vzhled pluginu a celé ukázky je až na minimální rozdíly stejný. Pro uživatele, který chce rezervační systém použít, nebudou tyto rozdíly viditelné.

Při testování pluginu v prohlížeči Internet Explorer 10 byl plugin správně zobrazen i spuštěn a rezervační systém funguje správně. V Internet Explorer v režimu pro vývojáře jsem při přepnutí prohlížeče do režimu odpovídajícího verzi 9 narazil na rozdíly v zobrazení. Jelikož Bootstrap používá zaoblení rohů pomocí CSS, mohou se na nepodporujících prohlížečích zobrazit rohy hranaté. To se stalo i v tomto případě. Plugin si ale stále zachoval plnou funkčnost a ukázka vypadala pořád přehledně.

Při snížení režimu prohlížeče Internet Explorer na verzi 7 a 8 plugin přestává fungovat. Vygeneruje pouze popisek první řady sedadel a poté selže. Horní menu v ukázce bootstrap změní vzhled a odkazy, které se zobrazovaly jako tlačítka, se nyní zobrazují v seznamu. Odkazy jsou ale stále funkční.

7 Závěr

Při práci na pluginu jsem se naučil programovat v jazyce JavaScript a pracovat s knihovnou jQuery. Také jsem si uvědomil, proč je knihovna právě v jazyce JavaScript tolik důležitá. Přišel jsem také na spoustu dalších oblastí, kde se dá práce s tímto jazykem a knihovnou uplatnit. Jejich použití je velmi široké a určitě budu v budoucnosti využívat získaných zkušeností. Dále jsem se seznámil s problematikou vývoje pluginů a vyzkoušel jsem si, jak takový plugin v praxi vytvořit. Velmi zajímavá byla zkušenost práce s různými formáty.

Při vývoji pluginu jsem se zaměřil hlavně na použitelnost pro různé rezervační systémy. Výsledkem mé práce jsou ukázky pro dva různé rezervační systémy. Přednastavil jsem všechny možnosti nastavení pluginu, aby vývojář mohl při použití v praxi z této ukázky čerpat. Při vývoji pluginu a jeho ukázky jsem se snažil používat již existující technologie. Snažil jsem se volit ty nejznámější (jako je např. bootstrap), aby se o nich dalo snadno najít dostatek informací a vývojář se tak s ukázkou rychleji seznámil.

Žádný podobný plugin pro rezervační systém neexistuje, nebo není snadno dostupný, proto je tato práce velkým přínosem. Plugin jsem se rozhodl zveřejnit pod většinou licencí, umožňující volné použití, jako je např. GPL a Creative Commons. Pro zálohování a verzování celé práce jsem použil systém pro správu verzí GIT. K tomuto účelu jsem využil služby GITHub, díky které je tato celá práce dobře dostupná z internetu.

Plugin jsem otestoval na většině moderních prohlížečů. Při testování se nevyskytly žádné problémy. Při využití vývojářského režimu aplikace Internet Explorer verze 7 a 8 se plugin nespustil správně. To může být z důvodu nedostatečné podpory nových technologií prohlížeče (např. HTML 5). V moderních prohlížečích plugin funguje správně. Při použití pluginu rezervačního systému v praxi by měl být uživatel upozorněn, že se nachází na nepodporovaném prohlížeči a měla by mu být nabídnuta vhodná alternativa.

Cílem práce bylo vytvořit plugin pro rezervační systém tak, aby byl co možná neuniverzálnější. Podle mého názoru jsem zadání splnil a naučil se využívat nové technologie. Vylepšit by bylo možné podporu prohlížeče Explorer 8. Při vytváření tohoto pluginu jsem dbal na to, aby se mohl dále používat v praxi. Při vytvoření obou ukázek jsem si ověřil, že plugin je použitelný i pro náročnější požadavky. Také jsem nechal prostor pro dopsání několika funkcí, které pracují s grafickými efekty, aby si každý vývojář mohl upravit plugin podle svého. Práce splňuje všechny body zadání. S touto prací jsem spokojen a doufám, že se bude používat.

Použitá literatura

[1] M. Smola, „HTML5: co přináší a proč se o něj zajímat,“ 29 srpen 2012. [Online]. Available: http://www.root.cz/clanky/html5-co-prinasi-a-proc-se-o-nej-zajimat/. [Přístup získán 7 duben 2013].

[2] M. Šimeček, „Programujte,“ 11 červenec 2011. [Online]. Available: http://programujte.com/clanek/2010082200-html5-nove-vlastnosti/. [Přístup získán 7 duben 2013].

[3] C. Bewick, „HTML5 Custom Data Attributes,“ 27 květen 2010. [Online]. Available: http://html5doctor.com/html5-custom-data-attributes/. [Přístup získán 7 duben 2013].

[4] M. Smola, „Novinky v CSS3: animace,“ 5 září 2012. [Online]. Available: http://www.root.cz/clanky/novinky-v-css3-animace/. [Přístup získán 7 duben 2013].

[5] neznámý. [Online]. Available: http://php.net/. [Přístup získán 7 duben 2013].

[6] neznámý, „What is MySQL?,“ [Online]. Available: http://dev.mysql.com/doc/refman/5.5/en/index.html. [Přístup získán 7 duben 2013].

[7] M. Kofler a B. Öggl, PHP 5 a MySQL 5, Brno: Computer Press a.s., 2007.

[8] neznámý, „MOZILLA DEVELOPER NETWORK,“ 23 říjen 2012. [Online]. Available: https://developer.mozilla.org/en-US/docs/JavaScript/Guide/JavaScript_Overview. [Přístup získán 13 duben 2013].

[9] L. Lhotka, „Nové standardy pro JSON,“ 11 duben 2013. [Online]. Available: http://www.root.cz/clanky/nove-standardy-pro-json/. [Přístup získán 13 duben 2013].

[10] J. Resig, jQuery Kuchařka programátora, Brno: Computer Press a.s., 2010.

[11] J. Rutter, „Smashing jQuery - Google Books,“ 2011. [Online]. Available: http://www.google.cz/books?hl=cs&lr=&id=owKwLfAll6QC&oi=fnd&pg=PP2&

dq=jQuery+best+practices+options+jQuery&ots=8qp75w6bRG&sig=kYlbOG0y my2uhq5Mf2EvDm97yf4&redir_esc=y. [Přístup získán 4 květen 2013].

[12] J. Chaffer a K. Swedberg, „Learning Jquery - Google Books,“ září 2011. [Online]. Available: http://www.google.cz/books?id=d_6bPyc5QVcC&printsec=frontcover&hl=cs#v=onepage&q&f=false. [Přístup získán 4 květen 2013].

[13] F. Kučera, „Distribuované verzovací systémy,“ 25 leden 2011. [Online]. Available: http://www.abclinuxu.cz/clanky/distribuovane-verzovaci-systemy-uvod-1. [Přístup získán 13 duben 2013].

[14] P. Baudiš, „Výlet do říše verzí: CVS,“ 15 duben 2003. [Online]. Available: http://www.root.cz/clanky/vylet-do-rise-verzi-cvs/. [Přístup získán 11 březen 2013].

[15] P. Baudiš, „Výlet do říše verzí: ideální verzovací systém?,“ 3 květen 2004. [Online]. Available: http://www.root.cz/clanky/vylet-do-rise-verzi-idealni-verzovaci-system/. [Přístup získán 11 březen 2013].

[16] J. Faigl, „Systémy pro správu verzí,“ [Online]. Available: http://lynx1.felk.cvut.cz/pte/doc/vcs.pdf. [Přístup získán 11 březen 2013].

[17] S. Chacon, „Small and Fast,“ 29 červen 2009. [Online]. Available: http://git-scm.com/about/small-and-fast. [Přístup získán 11 březen 2013].

[18] a. neznámý, „Bootstrap,“ [Online]. Available: http://twitter.github.com/bootstrap/index.html. [Přístup získán 15 březen 2013].

[19] neznámý, „PHP Form Builder Class,“ [Online]. Available: http://www.imavex.com/pfbc3.x-php5/index.php. [Přístup získán 13 duben 2013].

[20] J. Smith, „jQuery Tokeninput,“ [Online]. Available: http://loopj.com/jquery-tokeninput/. [Přístup získán 14 duben 2013].

[21] neznámý, „dev7studios,“ [Online]. Available: http://dev7studios.com/nivo-slider/. [Přístup získán 3 květen 2013].

[22] B. M. Dammani, „Seat Reservation with jQuery,“ 30 říjen 2011. [Online]. Available: http://techbrij.com/seat-reservation-with-jquery. [Přístup získán 13

duben 2013].

[23] D. Odell, JavaScript Průvodce programováním ajaxových aplikací, Brno: Computer Press a.s., 2010.

[24] M. Teutsch, „Introduction to PhpDoc,“ 9 leden 2012. [Online]. Available: http://phpmaster.com/introduction-to-phpdoc/. [Přístup získán 11 březen 2013].

[25] a. neznámý, „Jsdoc toolkit,“ [Online]. Available: https://code.google.com/p/jsdoc-toolkit/. [Přístup získán 11 březen 2013].

[26] J. Vrána, „Obrana proti SQL Injection,“ 2 březen 2005. [Online]. Available: http://php.vrana.cz/obrana-proti-sql-injection.php. [Přístup získán 15 březen 2013].

[27] R. York, „Beginning JavaScript and CSS Development with jQuery - Google Books,“ únor 2011. [Online]. Available: http://www.google.cz/books?hl=cs&lr=&id=L9otyT4crSQC&oi=fnd&pg=PT9&dq=jQuery+best+practices+options+jQuery&ots=qkIortf08D&sig=OUz4QVokGvdQhLZk7BBLfxuSKw8&redir_esc=y#v=onepage&q&f=false. [Přístup získán 4 květen 2013].

[28] neznámý, „jQuery - Browser Support,“ [Online]. Available: http://jquery.com/browser-support/. [Přístup získán 14 duben 2013].