Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Virtuální automobilové muzeum

Bc. Tomáš Kalina

Diplomová práce

2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš Kalina**
Osobní číslo:	**I16223**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Automobilové virtuální muzeum**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem diplomové práce je vytvořit pomocí nejnovějších javascriptových a PHP technologií webovou prezentaci na téma automobilové virtuální muzeum.
V teoretické části práce bude provedena rešerše stávajících javascriptových a PHP technologií a popis frameworku Vue.js, která budou implementovány v aplikaci. Detailněji bude zaměřeno na framework Symfony a jeho nadstavbu Sonata.
V praktické části bude naprogramováno virtuální automobilové muzeum na základě popsaných technologií, které bude umožňovat prezentaci jednotlivých automobilů a popis technických parametrů vozů. Dále bude obsahovat foto dokumentaci jak aktuálního tak dřívějšího stavu. K naplnění těchto dat bude sloužit implementovaný administrační systém.

Rozsah grafických prací: **10**

Rozsah pracovní zprávy: **60**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

ULLMAN, Larry E. Modern JavaScript: develop and design. Berkeley, CA: Peachpit Press, c2012. Develop and design. ISBN 9780321812520.
FRANÇOIS ZANINOTTO AND FABIEN POTENCIER. The definitive guide to symfony. New ed. Berkeley, Calif: Apress, 2007. ISBN 9781590597866.
GASSTON, Peter. The modern Web: multi-device Web development with HTML5, CSS3, and JavaScript. San Francisco: No Starch Press, 2013. ISBN 1593274874.

Vedoucí diplomové práce: **Ing. Zdeněk Šilar, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **30. října 2017**
Termín odevzdání diplomové práce: **18. května 2018**

Ing. Zdeněk Němec, Ph.D.
děkan

Univerzita Pardubice
Česká republika
L.S.

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2017

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 17. 5. 2018

Bc. Tomáš Kalina

PODĚKOVÁNÍ

Chtěl bych poděkovat panu Ing. Ph.D Zdeňku Šilarovi za vedení diplomové práce.

ANOTACE

Diplomová práce se věnuje problematice vývoje webů a nových technologií. Zejména pak technologii SPA v programovacím jazyce JavaScript ve framevorku Vue.js. Je zde také rozebrána řada technologií pro ulehčení tvorby vizuálu webu. Dále také rozebírám možnosti tvorby klientské části a její jednoduchou stavbu nad jazykem Symfony. V praktické části byla vytvořena aplikace na propagaci automobilového muzea.

KLÍČOVÁ SLOVA

SPA, Bulma, JavaScript, Vue, PHP, Symfony, Sonata

TITLE

Automobile Virtual Museum

ANNOTATION

The diploma thesis deals with the development of web sites in new technologies. In particular, the SPA technology in the JavaScript programming language in the Vue.js. There are also numbers of technologies to help visualize the Web. We also explore the possibilities of creating a client part and its simple construction over the language of the Symphonies. In the practical part, an application for the promotion of the automobile museum was created.

KEYWORDS

SPA, Bulma, JavaScript, Vue, PHP, Symfony, Sonata

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

PDF	Portable Document Format
PHP	PHP Hypertext Preprocessor
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
JS	JavaScript
SPA	Single-page applicatoin
MPA	Multiple-page application
DOM	Document Object Modelb
JSON	JavaScript Object Notation

ÚVOD

Diplomová práce se zabývá tématem vytvoření prezentace virtuálního automobilového muzea pomocí moderních webových technologiích. Práce je rozdělena na dvě části, teoretickou a praktickou.

V teoretické části bude provedena rešerše technologiích, kterými je možné vyvíjet webové aplikace. Vzhledem k tomu, že moderní webové aplikace postupně nahrazují ty tradiční okenní, budou rozebrány možnosti architektury, kterou můžeme implementovat. Dále bude proveden rozbor klasických JavaScriptových technologií, které jsou léty prověřené, tak i těch moderních, které začali vznikat v nedávné době. Ke každé technologii budou rozebrány možnosti, jak vytvářet konzistentní rozhraní pro tyto aplikace a použití daných standardů pro dobrou uživatelskou přívětivost. Dále budou specifikovány možnosti jak tvořit konzistentní, udržitelnou a bezpečnou aplikaci v jazyce PHP. Nad touto technologií budou prověřeny možnosti, jak nejlépe dělat administrační rozhraní, které bude komunikovat se serverovou databází.

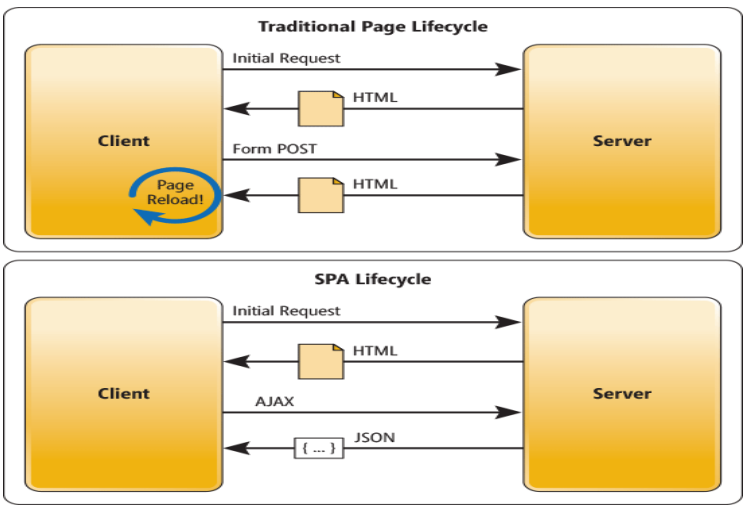
V praktické části bude popsána implementace reálné aplikace. Ta bude obsahovat jednostránkovou prezentační aplikaci, která bude zobrazovat zdrojová data, která nám budou poskytována ze serveru. Tato aplikace bude realizována technologií Vue.js a bude splňovat nejvyšší požadavky na výkon aplikace. Aplikace bude stylována metodami popsanými v teoretické části a nejmodernějšími technologiemi. Dále bude provedena realizace administrační části, která bude komunikovat s databází a bude snadno rozšiřitelná pro potřeby dalšího vývoje a bude obsahovat všechny operace potřebné pro editaci dat a fotografií. Také bude potřeba, aby byla aplikace schopná poskytovat data pomocí webové služby do prezentační části aplikace.

1 WEBOVÁ ARCHITEKTURA

V dnešní době jsou čím dál tím větší požadavky na to, aby naše aplikace fungovala stejně dobře na velkých monitorech stolních počítačů i na mobilních zařízeních. Toho jsme v minulosti mohli docílit jenom tím způsobem, že jsme vytvořili aplikaci pro daný operační systém na stolních počítačích a jinou aplikaci pro mobilní zařízení. To bohužel vedlo k velkým problémům jako je špatná synchronizace týmů a tím pádem obě aplikace mnohdy uměly něco jiného. V moderním světě vytváření webových aplikací existují dvě varianty vývoje aplikace. První z nich je multi-page application (MPA) neboli více stránková aplikace. Druhý způsob je single-page applicatoin (SPA) v překladu jednostránková aplikace.

Předtím ne začneme začneme jednotlivé způsoby vývoje popisovat je potřeba zmínit, že je i tyto přístupy možné kombinovat. Tím pádem si nemusíme uzavřít cestu v danném řešení.

Dále také můžeme rozdělovat styl vývoje mezi uživatelskou část, tak zvaný front-end, nebo serverovou část, tak zvaný back-end [1].



Obrázek 1: Rozdíl architektur. Zdroj:[6]

1.1 Více stránková aplikace (MPA)

Vícestránková aplikace je pojem, který nám do nedávna přišel naprosto běžný a bylo to synonimem klasického webu. Jak nám obrázek 1 napovídá, hlavní funkcionalitou bylo na základě každého požadavku znovu sestavit stránku na webu a odeslat jí zpět do klientova prohlížeče. Historicky byl na dynamicky renderovaných stránkách tento přístup naprosto běžný, ale postupem času jsme začaly přicházet na to, že není tak interaktivní jako běžné desktopové aplikace. Znovupřekreslování stránek mělo za negativní následek to, že se zbytečně vytěžoval server různými stejnými sestavováními dotazů. Jedním z řešení těchto problému jsou různé cachovací nástroje, které toto sestavování eliminují, ale přináší další řadu úskalí.

Naopak výhodou takových stránek je jejich rychlost vývoje. Jelikož má tento přístup velké historické kořeny, tak se řada nástrojů o tento princip opírá napříč všemi programovacímy jazyky a frameworky. Tím pádem získáváme vypiplané nástroje pro vývoj a snadné administrování webových projektů.

1.2 Jedno stránková aplikace (SPA)

Díky dnešním JavaScriptovím technologiím přicházíme do styku s tímto typem archytekturování aplikací. Díky potřebám firmy Facebook a jejich obrovskými požadavky na náročnosti aplikace se firma rozhodla o postupný přesun ze striktní vícestránkové aplikace na migraci směrem k volání webových servis a tudíž na jedno stránkovou aplikaci. Hlavní výhoda je ta, že se při každém dotazu na stránku nepřekresluje celý její obsah ale pouze část která je zapotřebí. To nutně nemusí znamenat dotaz na webovou službu, ale může jít jenom o drobné akce na úrovni klienta. S jejich rostoucími potřebami vznikly speciální požadavky na JavaScriptové frameworky. Díky obrovské programátorké síle, kterou Facebook disponuje se jí podařilo díky experimentům vytvořit React.js, který zahýbal celým světem vývojářů webových stránek.

Striktní jednostránková aplikace je tedy taková, která se nerenderuje na serveru, ale pomocí dotazování na webové služby si dotahuje data sama. Tím padem přesuneme obrovskou část výpočetního výkonu na stranu klienta. Což odlehčí hostitelskému serveru, ale dosáhneme tím také daleko vyšší rychlosti než u klasicky renderovaných stránek. Mezi další

výhody patří možnost ovlivnění stavu webové prezentace při načítání. Díky těmto možnostem může mít uživatel daleko příjemnější pocit z používání.

2 PREZENTAČNÍ ČÁST

V dřívější dobách nebylo nutné aplikaci rozdělovat na více částí, stačil nám defakto jeden programátor který nám udělal komplet vše. V dnešní době se ale každý člověk specializuje většinou na úzkou skupinu technologií, protože je velmi těžké udržet tempo s vývojem všech nástrojů, které se dneska na webu používají. S tohoto důvodu vznikla skupina, která se specializuje pouze na uživatelskou část, říkáme jim frond-end vývojáři.

Rozsah prací, za které jsou tito lidé zodpovědní, je v každé firmě jiný. Někde mají na starost malování návrhů a kódovaní HTML a CSS, jinde zase dělají vše co může člověk vidět a na co může klikat, takže mají přesah i do konkrétních technlogií, v kterých je daná stránka napsaná, a jinde se věnují i psaní celých aplikací v JavaScriptových frameworcích. Dále si proto popíšeme standardní skupinu technologií, kterou by měli programátoři zabývající se klienskou částí znát.

2.1 Značkovací jazyk HTML

HTML je zkratkou pro Hyper Text Markup Language, je to zkratka pro značkovací jazyk, který se používá pro kódování webových stránek. V tomto jazyce můžeme navrhovat základní strukturu stránek a rozložení daných prvků. Bez znalosti toho jazyka se žádný webový vývojář neobejde.

Vznik jazyka se datuje na rok 1990 kdy byl vyvinut společně s protokolem HTTP neboli HyperText Transfer Protokol který umožňuje komunikovat serveru a prohlížeči.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
    <title>Todo</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Obrázek 2: Ukázka HTML kódu

Základním stavebním kamenem je takzvaný tag neboli značka. Značky mají svou jedinečnou funkcionalitu a můžeme je dělit na párové a nepárové. Nepárové HTML tagy jsou takové, které mají pouze začínající značku která za zároveň ukončovací. Mezi tyto tagy patří například mezera „
“. Dále zde máme i tagy párové, ty mají tu vlastnost, že ohraničují text, na který má toto nastavení vliv. Například „<h1>“ který značí nadpis první úrovně.

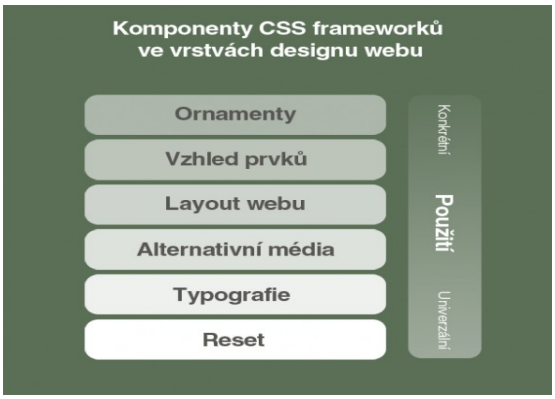
2.2 Kaskádové styly

Pro kaskádové styly neboli Cascading Style Sheets se používá zkratka CSS. Využívají se pro účely obohacení elementů HTML a další informace týkající se stylů a rozložení stránek. Jazyk podléhá standardům organizace W3C, která zastřešuje jeho vývoj. Mezi první autory se řadí Hakon Winum Lie.

Syntaxe jazyka je sestavená ze selektoru a bloku deklarací. Selektor udává pro co platí deklarovaná pravidla, přesněji pro jaký HTML element nebo třídu. Deklarovaný blok následně definuje hodnoty pro daný blok jako jsou například barva, pozadí, velikost a spoustu dalších.

2.2.1 Frameworky

Od začátku vzniku webu jsme byli odkázání pouze sami na sebe a vše na webu jsem si museli kódovat sami. S příchodem CSS frameworků máme ale k dispozici řadu nástrojů, díky kterým nemusíme stavět weby na zelené louce a můžeme použít předpřipravená řešení. To nám urychlí práci a vyřeší velké množství požadavků, se kterými se tvůrci těchto nástrojů už poprali před námi.



Obrázek 3: Komponenty CSS frameworků zdroj:[3]

Strukturu můžeme rozdělit do několik částí viz obrázek 3, kde na vrchních patrech stojí konkrétní prvky, které můžeme upravovat pro konkrétní řešení a na nižších patrech stojí univerzální komponenty, které používají všichni prakticky stejně a které definují uživatelská rozhraní. Lépe řečeno by je všichni stejně používat měli, protože to ulehčuje uživatelům práci s webem, protože komponenty už znají odjinud.

Mezi ornamenty patří sady ikonek, vzhled tlačítek a ostatní pravidla CSS. Některé frameworky obsahují i negativní témata, která upravují vzhled stránky opačné barvy z pravidla z bílého pozadí a černého textu na černý text a bílé pozadí. Vzhledy prvků obsahují definice formulářů. To nám umožňuje dělat rychle standardně nastylované formuláře. Na dalším patře je layout webu, to je definice rozvržení stránky. Zde můžeme řešit i responzivitu webu tak, aby byl vidět na mobilních zařízeních rozumně, ale to trochu zasahuje už do dalšího patra alternativní media. Řešíme zde například jak stránka bude vypadat v případě tisknutí. Typografie se řadí mezi ty základní kameny, které by měli být měněny pouze pokud víme co děláme, protože je to velice sofistikované téma. Řadí se zde především systematizace a fonty písem a nadpisů. A reset značí přepsání výchozích hodnot prohlížeče tak, abychom zaručili, že web se bude všude zobrazovat konzistentně [3].

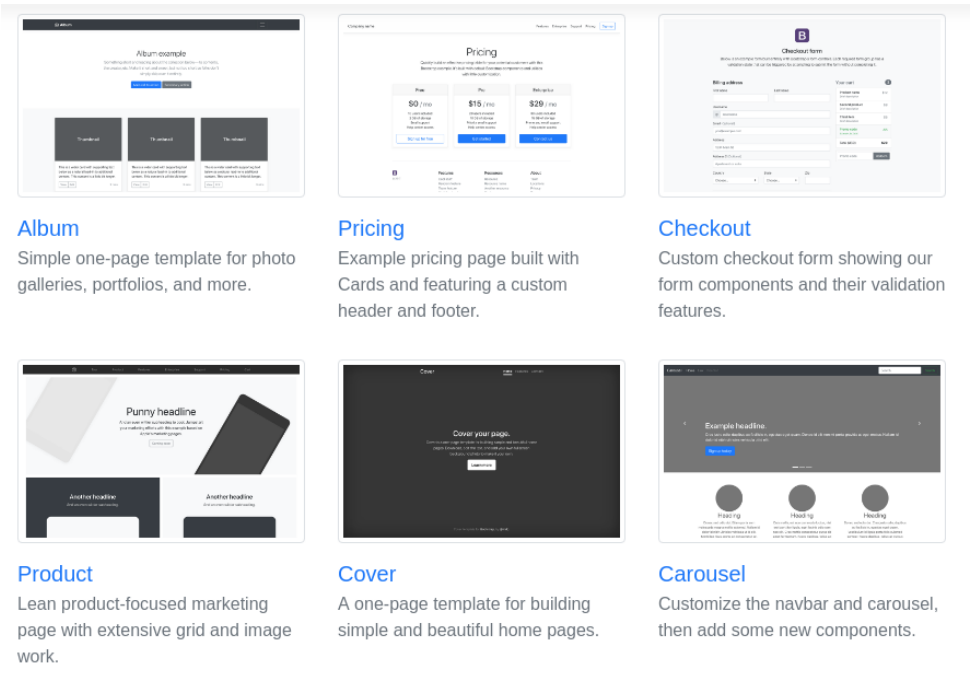
2.2.2 Bootstrap

Jednu z prvních nejpopulárnějších knihoven vytvořili v roce 2011 ve firmě Twitter. Momentálně se řadí mezi nejpoužívanější na celém světě.

Důvodem ke vzniku tohoto projektu byla hlavně nekonzistence všech aplikací ve firmě. Také bylo potřeba znát každý systém zvlášť a podle něj upravovat vzhledy. Napadla je tedy myšlenka vše sjednotit do jedné knihovny, kterou budou používat všichni stejně. Firma nakonec celý projekt uveřejnila jako open-source. To nám umožňuje jeho volné používání na libovolných projektech i ke komerčním účelům. Za dobu jeho existence se stal zárukou rychlého a kvalitního webu pro všemožné projekty. Obrovskou výhodou jsou i vzhledy, které už jsou předpřipravené na této platformě, takže nemusíme web stavět od nuly a zrychlí se nám tím práce.

Obrázek 4: Logo, Zdroj:[4]

Bootstrap staví na těch nejmodernější základech webdesingu. Vlastnost, která už je dnes defakto základ je responzivita. Styly, které jsou vytvořeny jsou dokonale přizpůsbeny ovládání na mobilních zařízeních. To je dnes velmi důležité, protože přístupy z těchto zařízení pomalu přesahují ty z klasických PC. Další důležitý fakt je ten, že dnešní prohlížeče upřednostňují weby s mobilním rozhraním proto, aby donutily vývojáře vyladit web. Mobile-first je další moderní technika pomocí které navrhujeme vzhled primárně na mobilní zařízení tak, aby se dal pohodlně obsluhovat, a až poté na velké rozlišení. To nám pomůže dělat z pohledu UX daleko ovladatelnější weby napříč všemi rozlišeními. Jednou z praktik na které také staví je Flat desing, což je trend kterým se weby dnes ubírají a to znamená to, že se veškerou grafiku snaží zjednodušit a udělat hodně kontrastní. Dále disponuje grid systémem. To nám zajišťuje 12 sloupců, které se nám sestavují podle rozlišení obrazovky a při správném použití dosáhneme velice jednoduše konzistentního vzhledu všude [4].



Obrázek 5: Repozitář Bootstrapu Zdroj:[20]

2.2.3 Bulma

Obdobně jako Bootstrap i Bulma slouží k jednodušší tvorbě webů. Na rozdíl od něj ale nemusí nést řadu zpětně kompatibilních věcí, které s sebou Bootstrap díky své dlouhé historii nese. Vznikla hlavně kvůli nezávislosti na jQuery, která je zásadní pro použitelnost v moderních JavaScriptových stránkách. Ona totiž vlastně nepoužívá JavaScript vůbec, protože využívá nejmodernějších kaskádových stylů, které už řeší mnohé použití. Díky tomuto konceptu, ale přichází o zpětnou kompatibilitu a proto podporuje pouze 90% funkcionalit IE 11.

Bulma díky svému minimalismu úplně odstranila gridové komponenty, protože díky flaxboxu, což je vlastnost CSS3, už není potřeba a orientace na stránce se řídí pomocí něj. Flexbox je defakto synonymem pro bootrap grid protože plní stejnou funkci.

Máme zde i obrovskou podporu ikonek díky podpoře Awesome Font ve verzi 5 a máme proto jednoduše k dispozici sadu 2 481 ikonek, které jsou okamžitě připravené pro použití [5].

2.2.4 Kaskádové styly v JavaScriptu

Posledních pár let, kdy se vyvíjí a postupně se zdokonalují technologie moderního JavaSriptu řešíme. jestli nemůžeme nahradit kaskádové preprocesory jakou jsou LESS nebo SASS. Ty nám napomáhají v jednoduchém stylovacím jazyce používat proměnné, podmínky nebo cykly. Díky tomu můžeme dosáhnout efektivnějších výsledků s lepší udržitelností stylů.

Snahou docílit jednoduchosti se v práci uchycuje myšlenka přesunout kaskádové styly do JavaScriptu. Této metodě říkáme CSS in JS. Tato myšlenka má kořeny ale daleko hlubší než je dnešní moderní web. A to v roce 1996, kdy ještě před vznikem značkovacího jazyka CSS vznikl ve společnosti Netscape jazyk s názvem JavaScript Style Sheets (JSSS). Ten uměl psát styly v JavaSriptu, ale převážně díky tomu, že se většina webů psala bez velkého množství tohoto jazyka a také díky tomu, že mnoho prohlížečů hojně JavaScript vypínala se tato metoda moc neujala.

Obrázek 6 HTML, JS, CSS Zdroj: [7]

Dnešní svět ale opět zatoužil po technologii která by nám umožňovala elegantně psát styly v kódu programu a proto vznikla řada knihoven, které nám tento postup ulehčí. Je ale nutné si specifikovat jaké jsou základní rozdíly ve využívání rozdílných technologií [8].

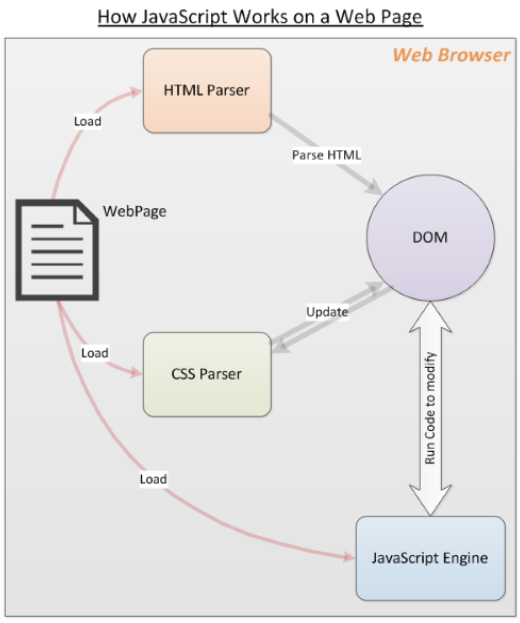
2.3 JavaScript

Rozvoj jazyka JavaScript začal v roce 1995 v Netscape Communications tvůrci prohlížeče Netscape. Uvědomili si, že přidání "jazykového lepidla", které by zlepšilo uživatelské zkušenosti, by zvýšilo příjemnost používání pro uživatele. Takže přivedli Brendana Eicha, aby vytvořil programovací jazyk Scheme. Jelikož Java byla v té době novým jazykem webu, rozhodli se, že jazyk se bude inspirovat syntakticky Javě. Výsledkem byl JavaScript s funkcemi Scheme, orientace objektu SmallTalk a syntaxe jazyka Java. První verze tohoto jazyka se jmenovala Mocha. Poté došlo k jejimu přejmenování na LiveScriptu a to ve stejném roce a ve stejném roce znovu přejmenována na JavaScript. V roce 1996 byl JavaScript předložen společnosti ECMA International pro standardizaci specifikace. V červnu 1997 byla vydána první oficiální specifikace pro jazyk jako ECMA-262. Nejnovější verzí jazyka je ECMAScript 2017, který byl vydán v červnu roku 2017.

Základ programovacího jazyka JavaScript tkví v tom, že běží na straně klienta, což je mnohdy z hlediska bezpečnosti dost limitující například v tom, že neumí pracovat se soubory. Základní jazyk umožňuje deklaraci proměnných, uložení a načtení hodnot, definování a volání funkcí, definici tříd, vkládání a využívání externích modulů, psát obslužné rutiny událostí, které odpovídají uživatelským a dalším událostem a mnohem víc.

Kompilace webu prohlížečem funguje následujícím způsobem. Webový prohlížeč načte webovou stránku, zpracovává HTML a vytvoří z obsahu obsah Document Object Model (DOM). DOM zobrazuje živý pohled na webovou stránku na kód JavaScript. Váš kód pak může aktualizovat DOM a nechat ho okamžitě předložit uživateli. Prohlížeč také umožňuje zaregistrovat kód, který chcete upozorňovat na události v uživatelském rozhraní, jako je pohyb myši, kliknutí na tlačítko atd. Díky všem těmto vlastnostem je možné vytvářet skvělé aplikace, které slouží jakémukoliv účelu, který si můžete vybrat.

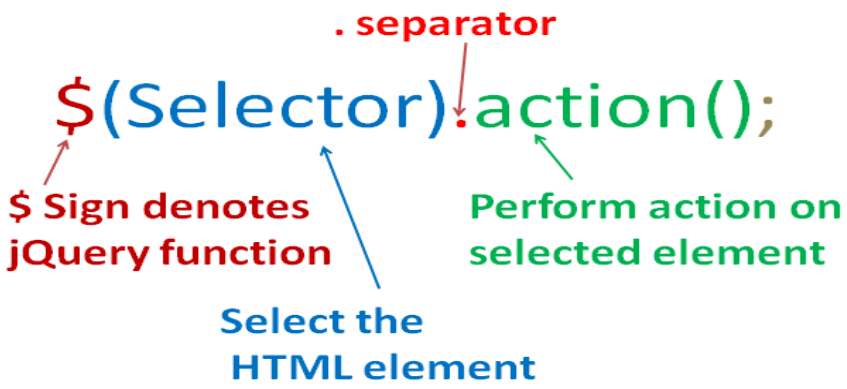
Když webový prohlížeč načte webovou stránku, analyzátor HTML začne analyzovat kód HTML a vytvářet DOM. Kdykoli analyzátor narazí na kód CSS nebo JavaScript, předává se podle potřeby analyzátoru CSS nebo JavaScriptu. JavaScriptový modul načte externí soubory JavaScript a vložený kód, ale kód neprovede okamžitě. Čeká na dokončení analýzy HTML a CSS. Jakmile to bude provedeno, bude JavaScript spuštěn v pořadí, v jakém byly nalezeny na webové stránce: jsou definovány proměnné a funkce, jsou spuštěny funkce, jsou spuštěny obsluhy událostí apod. Tyto činnosti vedou k aktualizaci jazyka DOM pomocí JavaScript a je vykreslen okamžitě prohlížečem [10].



Obrázek 7: Fungování jazyka JavaScript
Zdroj: 10

2.3.1 jQuery

Populární JavaScript knihovna známá jako jQuery dělá vše od rozbalovacích menu až po pokročilé efekty mnohem jednodušeji, než kdy předtím. Používají ji některé z největších a nejmenších jmen na webu Google, Dell, Netflix a NBC, a nepřeberné množství všech ostatních. Všichni tito velcí hráči tvořili pomocí jQuery docela úžasný obsah. Jedna z nejdůležitějších věcí proč toho docílily je, protože je zdarma, je to open-source a umožňuje vývojářům dělat více za méně času. Bohužel v době mnoha prohlížečů nám mnohdy standarty, které by zabezpečily stejné vykreslování mnoha prvků dost váznou. Díky tomu se vývoj běžného JavaScriptu dost prodražoval. Bylo to zapříčiněno tím, že se jakékoli maličkosti museli ladit pro velké množství prohlížečů a neexistoval způsob jak tuto práci urychlit. A přesně proto přišlo jQuery. To díky své dlouhé historii má za sebou řadu vyřešených funkcionalit a proto se používá na obrovském množství internetových stránek.



Obrázek 8: Syntaxe jQuery Zdroj: [11]

Funguje na základě manipulace s HTML Document Object Model (DOM) a je navržen pro zjednodušení skriptování HTML na straně klienta, jQuery obsahuje části HTML a CSS. Vyvíjení s jQuery je intuitivní a snadno se učí. Tato knihovna je postavena na kratším, jednodušším kódu pomocí jednoduché syntaxe a otevřených standardů kódování. Díky těmto vlastnostem mohou vývojáři zkrátit dobu potřebnou k nasazení aplikace nebo webu a odlaďování zpětné kompatibility.

Kromě toho vývojáři nemusí být experti v oblasti programování nebo tvorby webových stránek, aby vytvořili skvělé styly pro vaše stránky. Každý vývojář, který strávil hodiny kódováním a testováním souborů CSS jistě ocení jednoduchou implementaci, kterou jQuery přináší. K dispozici je také sada robustních komponent jQuery UI, které mohou vývojáři připojit na své webové stránky. Například bootstrap je naprosto svázaný.

V dnešní době jsme ale už technologicky jQuery překonali, protože nemusíme řešit každý prohlížeč zvlášť, ale díky technologii virtual-dom můžeme použít daleko rychlejší řešení pro manipulaci se stránkou [13].

2.3.2 React

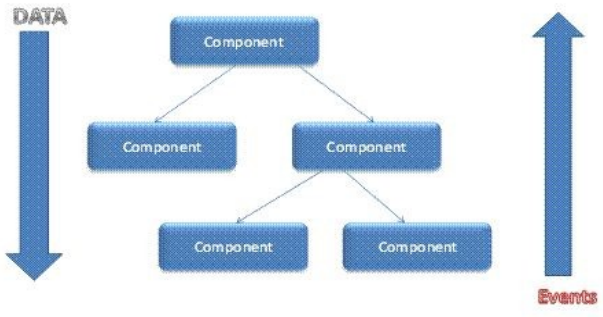
ReactJS je otevřená JavaScriptová knihovna , která se používá pro vytváření uživatelských rozhraní speciálně pro SPA aplikace. Používá se pro vykreslení zobrazovací vrstvy pro webové a mobilní aplikace. React také umožňuje vytvářet opakovaně použitelné komponenty rozhraní. React byl poprvé vytvořen Jordanem Walkeem, softwarovým inženýrem pracujícím pro Facebook. První nasazení bylo na Facebooku v roce 2011 a na Instagramu v roce 2012.

Obrázek 9: Logo Reactu, Zdroj:[15]

React umožňuje vývojářům vytvářet velké webové aplikace, které mohou měnit data bez nutnosti opětovného načtení stránky. Hlavním účelem Reactu je aby aplikace byla rychlá, škálovatelná a jednoduchá. Funguje pouze v uživatelských rozhraních v aplikaci. Nejčastěji se připodobňuje view vrstvě v MVC paternu, což neodpovídá skutečnosti, ale pro lepší představu postačí. Může být použita s kombinací dalších knihoven nebo frameworků jazyka JavaScript.

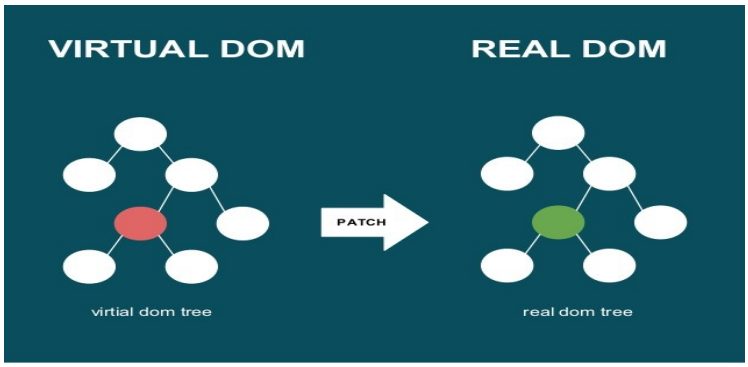
React má několik vlastností díky kterým nám pomáhá vytvářet kvalitnější aplikace, první z nich je JSX. Ten máme k dispozici místo běžného jazyka JavaScript pro vytváření šablon. JSX je jednoduchý JavaScript, který umožňuje používat HTML značky k parametrizaci a vykreslování dílčích komponent. Syntaxe HTML je zpracovávána do volání jazyka React Frameworku. Můžeme také psát čistě starý JavaScript.

V Reactu pro tok dat používáme takzvaný jednosměrný datový tok, to v praxi znamená, že komponenty mohou obsluhovat pouze svůj vnitřní stav a stavy podřazené. Tím zabalíme funkcionalitu do komponent a nebude se nám stávat, že se nám stav bude negativně ovlivňovat s ostatními aplikačními částmi [16].



Obrázek 10: Jednosměrný datový tok, Zdroj:[16]

Jako další a nejsilnější zbraň Reactu je virtuální DOM. Celý tento koncept vznikl právě v dílnách Reactu, kdy vývojáři zjistili jako první že v okamžiku, kdy upravujeme data ve virtuálním domu a pak je pomocí záplat přenášíme do toho reálného, je to mnohem efektivnější než vždy přistupovat k tomu reálnému a na základě jeho momentálního nastavení ho měnit nebo ne. Po této implementaci se vyrojila řada klonů, které tento koncept kopírují a jsou podobně efektivní [17].

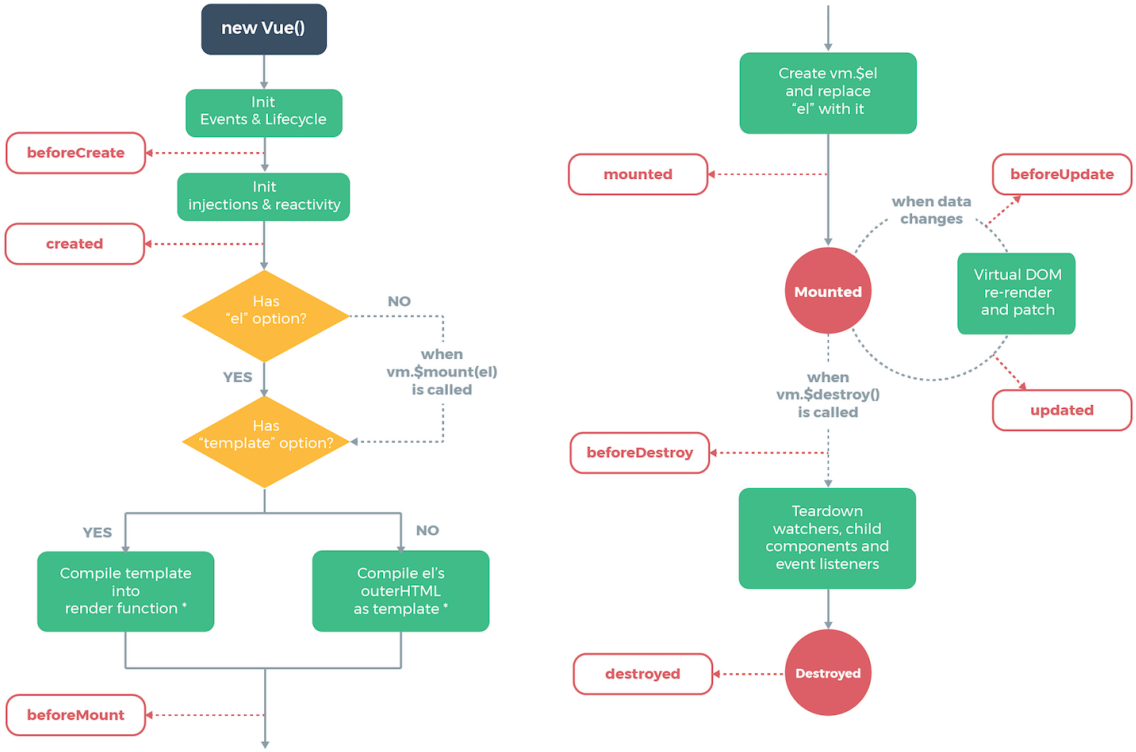


Obrázek 11: Virtual-dom. Zdroj: [17]

2.3.3 Vue.js

Vue je progresivní JavaScriptový framework pro vytváření uživatelských rozhraní. Na rozdíl od jiných monolitických, je Vue navržen od základů, aby byl postupně integrovatelný. Knihovna jádra je zaměřena pouze na vrstvu zobrazovací stejně jako React. Veliká výhoda je ve snadné integraci s jinými knihovnami nebo existujícími projekty. Na druhou stranu je Vue dokonale schopno využívat i SPA technologii, když je používán v kombinaci s moderními nástroji a dalšími knihovnami.

Největší výhoda Vue tkví v rozdělení šablonovacího systému mimo JavaScriptovou část. To umožní integraci se všemi dalšími jazyky daleko jednodušeji a sofistikovaněji. Další z nesporných výhod je jeho velikost. Protože je novější a nemusí nést následky z předchozích verzí. Díky pár výhodám se rozhodli používat ho jako primární jazyk například v technologii postavené na Laravelu.

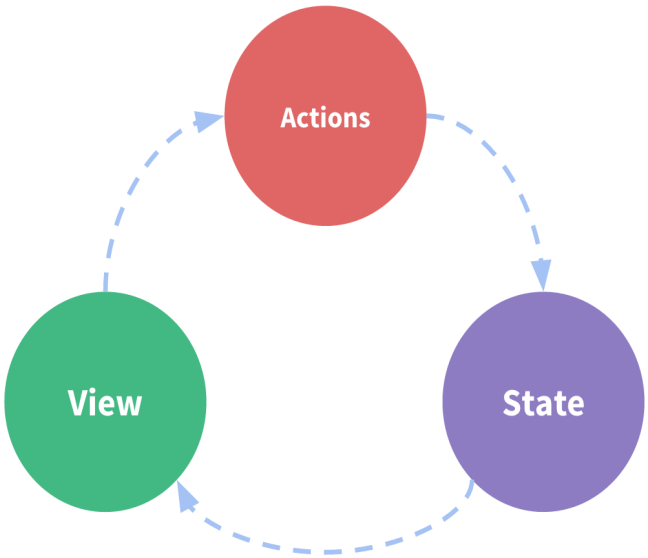


Obrázek 12: životní cyklus Vue.js Zdroj: [18]

Základním stavebním prvkem každé moderní technologie je životní cyklus. Ve Vue je životní cyklus podobný jako v Reactu a protože je velmi důležitý pro pochopení fungování je nutné ho zmínit [18] [19].

2.3.4 Vuex

Ve světě moderního JavaScriptu je velkým problémem uchovávání dat. Pří výběru můžeme použít několik typů. Můžeme si vytvořit vlastní statový kontejner nebo můžeme použít už navrhnuté koncepty. Mezi těmi, které jsou normou patří Redux nebo Mobx. Ve frameworku je právě Vuex obdoubou knihovny Redux. Je to knihovna pro řízení stavu pro aplikace Vue.js. Slouží jako centrální sklad pro všechny součásti v aplikaci. Podle pravidel zajišťuje, že stav může být změněn pouze imutabilním způsobem. Což je způsob, kde se nemění hodnota dat ale předává se kopie dat se změněnou hodnotou. Je také integrován v oficiálním rozšířením Vue devtools, který poskytuje pokročilé funkce, jako je ladění při procházení aplikace s nulovým konfigurováním.



Obrázek 13: Průbeh měnění stavů aplikace. Zdroj: [19]

Stavový kontejner funguje tak, že můžeme měnit stav aplikace tedy State pouze přes akce. Kontejner pak dá vědět prezentační vrstvě označované jako View, aby díky immutabilnímu stavu změnila jenom ty hodnoty, které byly opravdu změněny. Tím ušetříme hodně časově náročných operací při překreslování aplikace [19].

2.3.5 Node.js a NPM

Node je asynchronní událostmi řízený JavaScript. Je navržen tak, aby vytvářel škálovatelné síťové aplikace. Proto může pracovat souběžně s mnoha spojeními, což šetří mnoho zdrojů a může být rychlejší než ostatní systémy. Při každém připojení je spuštěno zpětné volání, jinak je ale vlákno uspané pro budoucí potřeby využití.

To je na rozdíl od dnešního běžnějšího modelu aplikací, které pracují sériově, kde jsou používány závity OS. Sítě založené na vláknech jsou poměrně neúčinné a velmi obtížně použitelné. Navíc uživatelé procesy jsou bez blokování procesů, protože neexistují zámky. Téměř žádná funkce nevykonává přímo čtení nebo zápis, takže proces nikdy nezablokuje. Vzhledem k tomu, že nic nesmí blokovat, jsou v systému Node velmi jednoduše implementovatelné škálovatelné systémy.

Stejně jako všechny moderní technologie i JavaScriptový Node má svůj balíčkovací systém. V tomto případě jde o NPM. Ten řeší závislosti v projektu a zároveň spravuje repozitář, kde jsou uloženy všechny balíčky, kterých je něco přes půl miliónu. Tento repozitář nabízí i privátní uskladnění balíčků pro práci v teamu. Tento nástroj zpravuje jak balíčky ve formě knihoven, tak již hotové programy nebo kompilátory. Neslouží ale zdaleka jenom jako hloupý repozitář, ale můžeme díky němu provádět i takzvané npm scripty. Ty slouží na spuštění všech možných procesů a částečně nahrazují makefile [21].

2.3.6 Webpack

Webpack je sada nástrojů, díky které můžeme pohodlně skládat všechny naše používané knihovny do jednoho celku, to nám umožní jednodušeji aplikaci distribuovat, ale také provádět různé operace nad ní. Můžeme například mimifikovat kód, který jsme složili z více částí. Tato vlastnost nám ušetří velkou režiji na přenos aplikace a díky ní vyšší uživatelskou přívětivost. Dále můžeme nyní dělat „tree shaking“, tedy třesení stromem. To umožňuje to, že nám do výsledného souboru projdou pouze třídy, které nejsou uvnitř kódu použité což opět sníží velikost aplikace. Umí ale velké množství dalších věcí. Asi nejlepší věc ze světa JavaScriptového vývoje je Hot Module Replacement. To je vlastnost, která nám umožní vyměnit ve výsledné aplikaci pouze upravené části kódu. Díky tomu, můžeme vyvíjet aplikaci a průběžně jí ukládat a ona sama na základě hlídaní změn v souborech určí, která část stránky se má znovu načíst a překreslit.

2.3.7 Babel

Babel je překladač novější verze JavaScriptu do starší verze. Původně byla hlavně vyvinuta na překlad z verze JavaScriptu ES6 do ES5. Bylo to díky obrovskému skoku, který tento jazyk učinil za velmi krátkou dobu. Tato funkcionalita je nezbytná pro vývoj moderních aplikací. Je tomu tak, protože dlouhou dobu byl tento jazyk nevyvíjen a momentální podpora prohlížečů nových verzí je dost žalostná. Díky tomu tedy můžeme používat vlastnosti moderního JavaScriptu už dnes a v budoucnu pouze přepnout finální kompilaci.

Verze ES6 nám přináší ohromné množství nových funkcionalit, které je potřeba používat z důvodu udržitelnosti kódu a většího pohodlí při programování. Největší novinkou jsou třídy. Vlastně upravena je hlavně dědičnost, kterou už nemusíme řešit přes prototyp funkce, ale máme jí implementovanou stejně jako v ostatních jazycích i když nejde o klasickou OOP dědičnost. Další novinka je zrušení univerzální proměnné vat a nahrazení nové let a const. První z páru definuje proměnné které mohou v čase měnit svojí hodnotu. Oproti tomu const je neměnná a stále stejná po celou dobu existence. Jako další máme moduly, a to pomocí klíčového slova export. Díky němu můžeme vystavovat funkce nebo třídy ven z balíčku pro použití i jinde. Pro načtení pak máme klíčové slovo import. Díky tomu můžeme pak pohodlně používat nástroj Webpack se všemi jeho výhodami [22].

3 ADMINISTRAČNÍ ČÁST

Administrační část se nejčastěji v programátorském světě říká backend. Jedná se o část aplikace, která může například spolupracovat s databází nebo jinak poskytovat data. Volba technohlogií, které použijeme je jen na nás a nemusíme brát ohledy na fungování SPA aplikace na klientské části. Díky tomu si můžeme vybrat jazyk který nám nejvíce vyhovuje. Na to působí mnoho aspektů. Jeden z nich je například požadovaná rychlost aplikace kde se rozhodujeme na základě toho, aby uživatelská část byla co nejpříjemnější na obsluhu. Další argument může být bezpečnost například v bankovním sektoru. A nebo celková cena vývoje a provozních nákladů.

3.1 PHP

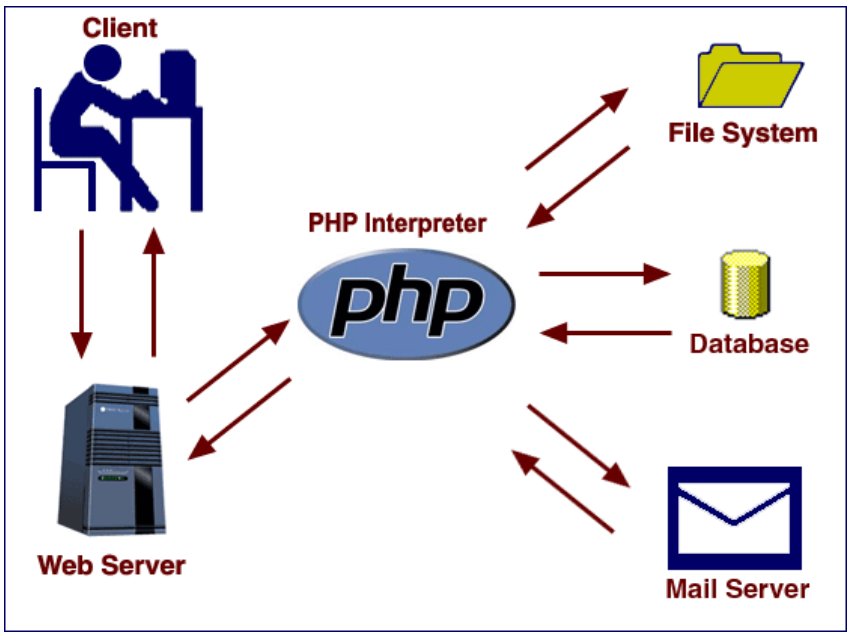
Zkratka PHP je rekurzivní zkratkou pro PHP Hypertext Preprocessor, kde je právě první slovo rekurzí názvu. Jedná se o open-source scriptovací jazyk, který je primárně vyvinut pro vývoj webových aplikací a vkládání do HTML. PHP se zaměřuje hlavně na skriptování na straně serveru. To nám umožňuje dělat vlastně vše, co je potřeba. Například sbírat data ve formulářích, vytvářet dynamické obsahy stránek nebo odesílat a přijímat soubory. PHP však dokáže daleko víc. Podporuje procedurální programování nebo objektově orientované programování nebo jejich kombinaci. Existují tři hlavní oblasti, kde se používají PHP skripty.

Skriptování na straně serveru je nejtradičnější a hlavní cílem. K tomu je potřeba tři věci. Analyzátor PHP, webový server a webový prohlížeč. Potřebujete spustit webový server s připojenou instalací PHP. K výstupu programu PHP můžete přistupovat pomocí webového prohlížeče, který zobrazuje stránku PHP prostřednictvím serveru.

Další využití může sloužit jako skriptování příkazového řádku. Můžete si vytvořit PHP skript, který bude spuštěn bez jakéhokoli serveru nebo prohlížeče. Tento typ použití je ideální pro skripty, které jsou pravidelně prováděny pomocí cron v Linuxu nebo Plánovač úloh ve systému Windows. Tyto skripty lze také použít pro jednoduché úlohy zpracování textu.

Psaní desktopových aplikací v PHP není nejlepším způsobem k vytvoření okení aplikace s grafickým uživatelským rozhraním, ale pokud znáte PHP velmi dobře a chcete využít některé pokročilé funkce PHP ve vašich aplikacích na straně klienta, můžete také použít PHP-GTK psát takové programy. Máte také možnost psát aplikace na více platforem najednou, protože interpret PHP je možné spustit všude.

PHP lze použít na všech hlavních operačních systémech, včetně Linuxu, mnoha variant Unixu (včetně HP-UX, Solaris a OpenBSD), Microsoft Windows, Mac OS X, RISC OS a dalších. PHP také podporuje naprostou většinu webových serverů. PHP funguje buď jako modul nebo jako procesor CGI, který spolupracuje se serverem. U tohoto jazyka se nejedná pouze o výstupní HTML. Možnosti PHP zahrnují výstup obrázků, souborů PDF generovaných za běhu. Můžete také snadno vytisknout jakýkoli text, například XHTML a jiný soubor XML. Další silnou zbraní je obrovská podpora databáz která zahrnuje všechny možné implementace. Podporuje také komunikaci s dalšími službami protokolů jako jsou LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM. Podporuje také protokoly systému Windows a mnohé dalších. Existuje mnoho dalších zajímavých rozšíření, které lze jednoduše implementovat. Pro vývoj se také naprosto nezbytný Xdebug, který umožňuje procházet aplikace a ladit je stejně jako tomu je v ostatních objektových jazycích [23].



Obrázek 14: Fungování PHP Zdroj:[24]

3.1.1 Composer

Composer je multiplatformní nástroj pro spravování závislostí v PHP . Můžeme skrze něj instalovat balíčky a definovat jakou verzi máme nainstalovat a můžeme použít nebo jakou máme navýšit. Všechny závislosti, které v projektu máme se nám ukládají do složky vendor v projektu. Ale můžeme instalovat závislosti i globálně. Také můžeme nastavit cesty pro snadné automatické načítání projektu. Composer je open-source programem. Standardní uložiště pro balíčky, které používá je Packagist. To je velké skladiště všech balíčků a jejich verzí, kde mohou programátoři vystavovat své projekty. Na na webu si můžeme i jednoduše zjistit další závislosti určité knihovny. Závislosti jsou zapisovány do souboru composer.json. Zároveň máme soubor composer.lock do kterého se zapisují aktuální nainstalované verze které se zapíší automaticky po nainstalování závislostí. Composer není v PHP od začátku. Inspiroval se od své konkurence v jiných jazycích například od NPM a vznikl roku 2011. A stejně jako jeho vzor obsahuje i možnost tvorby vlastní definice skriptů [25].

3.1.2 Symfony

Jedná se o PHP Framework. Obdobně jako velké množství technologií kolem PHP a JavaScriptu je i Symfony open-source. Slouží pro vývoj webových aplikací, které jsou renderované na serveru. Na tomto řešení momentálně fungují tisíce webů po celém světě a je použit v mnoha dalších projektech jako je Drupal, phpBB, Laravel nebo Stormm. Od roku 1998 má podporu společnost SensionLabs, která stojí za tímto projektem.

Symfony je momentálně ve verzi 4 a představuje mnoho svých nápadů a vlastností od základů tak, aby odpovídaly praktickým postupům. Nemá balíčky, které odstranila z předchozí verze. Konfigurační parametry jsou nyní v proměnných prostředí. Struktura adresářů aplikací je snadnější v navigaci a mnoha dalších.

Symfony se snaží standardizovat vývoj aplikací, aby nebyly tak nepřehledné a inspiruje se u ostatních jazyků. Mezi základní výhody patří jednodušší učení, snadnější konfigurace, snadnější instalace a nasazení a snadnější ovládání.

Symfony klade důraz na bezproblémovou integraci pomocí nového nástroje Flex tak, aby automatizoval nejčastější úkoly prováděné v aplikacích, jako jsou normální nastavení všech závislých konfigurací. Automatizace využívá Symfony Recepty, které obsahují pokyny pro integraci stovek balíčků třetích stran.

Nově splňuje Symfony definici mikro frameworku, to znamená že je malý a postupně do něj můžeme doinstalovávat další potřebné rozšíření. Díky tomu se hodí na jakékoli použití jako jsou mikro stránky, API, monolitické webové aplikace, konzolové aplikace nebo jako backend pro SPA aplikace v JavaScriptu [26].

3.1.3 Sonata

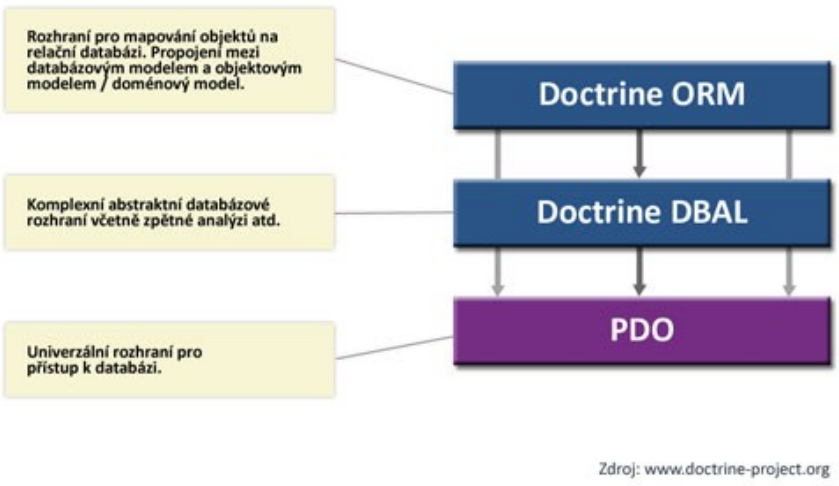
Projekt Sonata je open-source sada balíčků postavených na nejnovější verzi Symfony. Je založené na komunitní práci po celém světě a používá nejznámější Symfony balíčky. Projekt Sonata byl založen společností Thomas Rabaix a jeho cílem bylo vybudovat jiný způsob e-commerce řešení. První balíček, který byl vydán komunitně, je AdminBundle, který má dosud přes 300K stažení. Během posledních 3 let pomohl tým projektu Sonata a přispěvatelů k vydání více než 20 balíčků. Každý konkrétní balíček reaguje na konkrétní potřebu nebo problém a může pracovat nezávisle na druhých. Jsou využívány v projektech jako CMF nebo Sylius.

Sonáta především řeší základní problémy s nimiž se vždy potýkali programátoři aplikací. Díky ní můžeme psát efektivně aplikace a nemusíme znovu vytvářet věci, které budeme potřebovat [27].

3.1.4 Doctrine 2

Je object-relational mapper (ORM), tedy objektově relační mapovač pro PHP v minimální verzi 5.4. Zajišťuje persistentní stav v PHP. To je takový stav, kde obsah v entitách zrcadlí obsah v databázi. Využívá model mapování, který je obsažen v jádře této knihovny. Na rozdíl od své první verze, která uměla hodně dalších věcí, umí Doctrine 2 pouze mapovat objekty z databáze. První verze toho uměla daleko víc a díky tomu se stal její vývoj neudržitelný. Doctrina podporuje veškeré populární typy jak relačních, tak dokumentových databází, které umí efektivně.

Entity jsou objekty PHP, které obsahu právě data z databáze, které lze identifikovat na základě mnoha požadavků pomocí jedinečného identifikátoru nebo primárního klíče. Tyto třídy nepotřebují implementovat žádnou abstraktní základní třídu nebo rozhraní. Třída entity nesmí být final a nesmí obsahovat final metody.



Obrázek 15: Doctrine ORM

Doctrina disponuje také posluchači na dané události. Takže můžeme zachycovat upravování entit v čase. Můžeme například odchytávat entitu ještě před uložením a pracovat s ní. To nám umožní úplně odstranit databázové trigry, které nám mnohdy můžou přivodit velké problémy při provozu aplikace [28].

3.2 Databáze

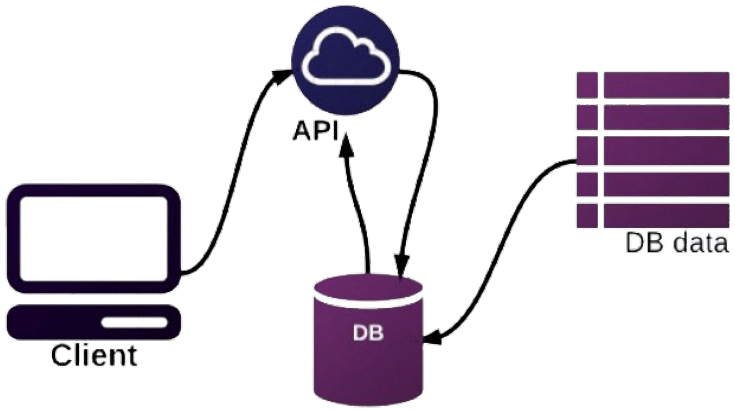
Databáze obsluhují souhrn informací, které jsou organizovány tak, aby se data snadno vkládala, vypisovala a upravovala. Databáze se od svého založení v šedesátých letech minulého století začaly rozvíjet, hierarchické a síťové databáze, v osmdesátých letech objektově orientované databáze a dnes v databáze SQL, NoSQL a cloudové. Databáze můžeme dělit podle základních principů uspořádání dat na relační a dokumentové. Základní a nejvíce využívané jsou relační. Data v nich jsou uspořádaná do řádků, sloupců a tabulek. Jsou indexována tak, aby usnadnily nalezení relevantních informací. Data se aktualizují, rozšiřují a odstraňují při přidávání nových informací. Databáze řeší optimalizace při zatížení při němž se vytvářejí, aktualizují a vyhledávaní data. Některé databáze nabízejí prostředky pro zachování atomicity, konzistence, izolace a trvanlivosti. Tím zařídí, že data budou konzistentní a konkurenceschopné napříč dotazy. Dokumentové databáze nám zase přináší daleko vyšší rychlost díky tomu, že postrádají relace mezi objekty. Proto se nejvíce hodí na velké objemy dat takzvané big data. Můžou se využívat efektivně na vytváření historie objektů, kde se data jenom otiskují a už se nad nimi nemusí provádět žádné další operace [29].

3.2.1 MariaDB

Server MariaDB je momentálně jedním z nejoblíbenějších databázových serverů na světě. Tento projek byl vytvořen vývojáři MySQL klonem této databáze po akvizici společnosti Oracle. Je plně open-source a nadále by tomu tak mělo zůstat. Mezi nejvýznamnější uživatele patří Wikipedia, WordPress a Google. MariaDB zpracovává data na strukturované informace v široké škále aplikací, od bankovních systémů po internetové stránky. Jedná se o lepší náhradu za MySQL který zůstává zpětně kompatibilní. Používá se, protože je rychlá, škálovatelná a robustní, s bohatým ekosystémem paměťových motorů, rozšíření a mnoha dalších nástrojů, díky nimž je velmi univerzální pro širokou škálu případů použití [30].

3.3 Webová služba API

Webové služby jsou specifické koncové body, které nám poskytují strojově čitelná data. Původně sloužila pouze na zefektivnění obchodních procesů a sdílení dat v rámci organizací, e-shopů a podniků mezi sebou. Dnes se ale může jednat o mezivrstvu při komunikaci se SPA aplikací. Tyto webové služby slouží na vystavování dat tak, aby nemuselo záležet na konkrétní jazykové implementaci. Proto můžeme pomocí tohoto rozhraní propojit různé programovací jazyky. Služby mohou provádět jakékoli operace na serveru a to například různé obchodní výpočty, matematické výpočty, čtení z databáze, ukládání či upravování dat.



Obrázek 16: Webová služba Zdroj: [31]

Webové služby můžeme rozdělit do datových, a procedurálních. Mezi ty procedurální patří starší XML-RPC nebo SOAP. Mezi datové patří REST nebo QraphQL [31].

3.3.1 REST

REST pochází z anglického spojení Representational State Transfer. A jedná se o architekturu rozhraní navrženou pro distribuci dat. Byla vytvořena v roce 2000 vývojářem Royem Fieldingem, který stojí také za základy protokolu HTTP a má s ním tedy hodně společného. První verze vznikla zároveň s verzí HTTP1/1 a od té doby je pomocí tohoto protokolu distribuována. Jazyk popisuje možnost upravování dat nebo provádění akcí nad zdroji. Tyto zdroje jsou unikání URL adresy nad kterými jsou prováděné akce pro vytváření, čtení, upravování nebo mazání. Díky voláním zdrojů následně aplikace mění stav, to se může například projevovat nejčastěji čtením nebo zápisem do databáze. Formáty dat poskytovaných technologií REST můžou být různé. Dříve byli velmi populární ATOM/RSS protokoly, které popisovali data podobně jako XML. V novodobé sféře internetu jsou ale tato data prakticky bez vyjímky poskytována jako JSON. REST definuje čtyři základní principy pro práci nad daty, které se všudypřítomně označuje jako CRUD. To znamená vytváření dat, získávání dat, upravování dat a mazání. Každá z těchto metod má odpovídající akci v HTTP.

Pro získávání dat se používá GET. To je metoda, která by měla obsahovat pouze URL. Zdroj by měl základně fungovat tak, že po zavolání metodou GET nám vrátí všechny data pro přístupný zdroj. Taková operace by měla být neměnná, vracet pokaždé stejná data abychom mohli využít silné zbraně cachování, které nám poskytuje HTTP. Tato akce by neměla obsahovat tělo požadavku.

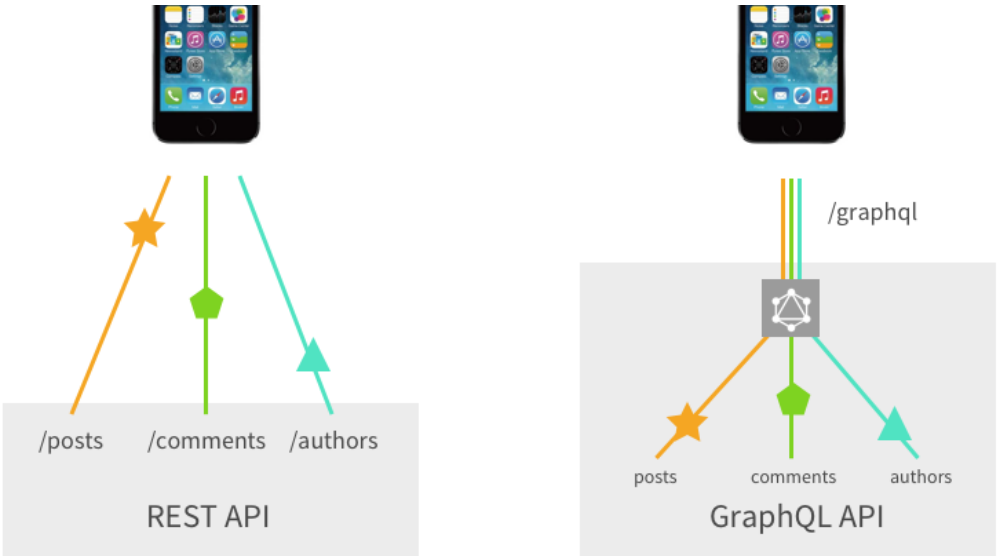
Pro vytváření dat máme k dispozici metodu POST. Ta má za úkol cokoli co jde na vstup zpracovat a pokud jsou informace v požadovaném formátu, uloží jej. Data se přidávají do těla požadavku, kde si je server následně vytáhne a zpracuje.

Na mazání dat nám poslouží metoda DELETE, tato metoda by měla vypadat podobně jako metoda GET. Dost často je i jím nahrazována buď z nedostatečného pochopení služby nebo s technologických limitů, které dříve obsahovala.

A poslední metoda slouží k upravování dat a je to metoda PUT. Ta je také mnohdy nahrazována služnou POST ale při použití této metody vzniká akorát zmatek v kódu a vede to k neudržitelnosti dat.

3.3.2 GraphQL

GraphQL je dotazovací jazyk vytvořený společností Facebook, pro tvorbu webových služeb. Vznikl na základě potřeb na vysokou rychlost přenosu dat a minimalistickou velikost. Má podobu formátu JSON, ale lehce upravenou o parametry, aby se dal lehce používat v jazyce JavaScript. Jedná se o jazyk, který je prováděný na straně serveru. Má za úkol získávání a manipulování s daty. Na základě specifikace v jazyce server provádí dotazy do databáze nebo jiného uložiště. Specifikace jazyka v GraphQL službách je tvořena dle definice typů. Pro vývoj obsahuje prostředí, díky kterému můžeme pěkně skládat dotazy a ladit je. Oproti technologii REST máme jenom jeden koncový bod na který provádíme dotazy podle toho, jak potřebujeme. Dotazy řadíme do několika kategorií.



Obrázek 17: GraphQL vs REST Zdroj: [32]

První je Field. Ten slouží pro definici objektů a jejich parametrů, které se mají vracet. Na základě toho můžeme definovat jaké informace budou dostupné pro získávání a v jaké podobě.

Na úpravu dat slouží metoda Mutations. Touto metodou definujeme možnost upravení nebo vkládání dat. Je to na základě parametrů, ale funguje podobně jako definice Field [33].

3.3.3 Autentizace

Dost často, obzvlášť v SPA aplikacích, potřebujeme přihlášení uživatele. To lze naaplikovat několika způsoby. Nejjednodušší cesta pro autentizaci na webových službách je pomocí Basic Authentication. Základním principem je zakódování jména a hesla v hlavičce ve formátu base64. Proto je nutné používat tohoto zabezpečení pouze na spojení HTTPS kvůli šifrování spojení. Jako další nevýhoda je potřeba mít heslo v hlavičce při každém dotazu na server. Vylepšení které řeší tento problém je HMAC. To nám umožňuje odesílané heslo nejprve zašifrovat. Problém je, že i když je jméno zakódováno, tak jsou způsoby na rozšífrování.

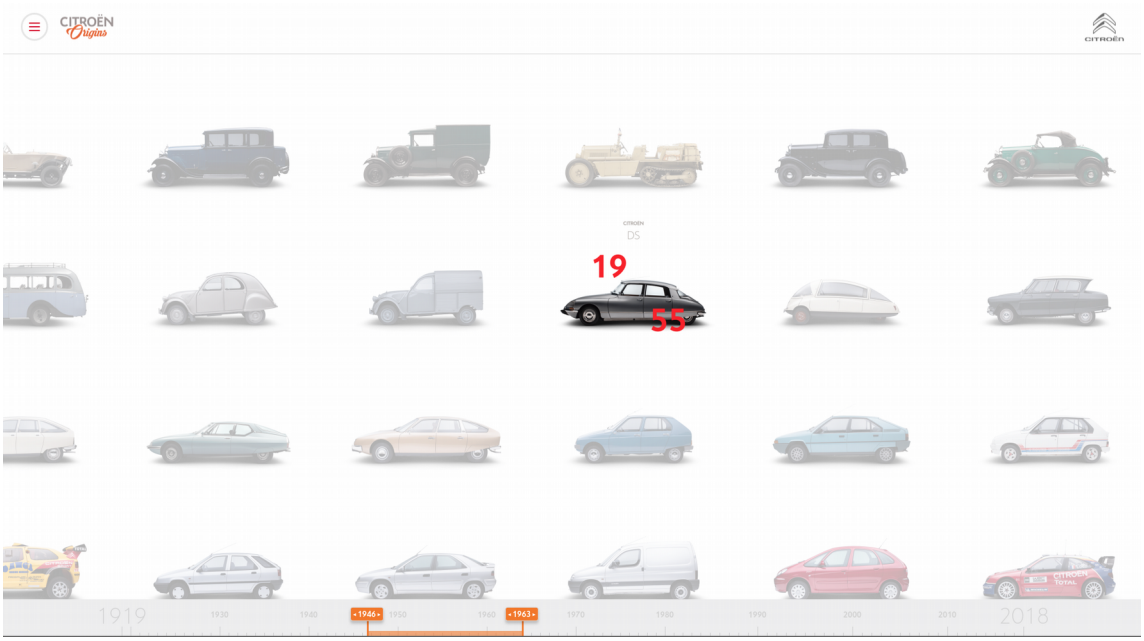
Další způsob je OAuth. Verze 1.0 vznikla v roce 2007. Jedná se o implementaci frameworku pro digitální podpisy. Byl velmi prosazován společnostmi jako je Google nebo Twitter. V roce 2012 ale přišel nástupce ve verzi OAuth 2.0. Tato verze je mnohem jednoduší na implementaci, ale díky tlaku větších společností, bylo uděláno mnoho kompromisů ohledné bezpečnosti. Každopádně se jedná o bezpečný způsob autentizace API [34].

4 REALIZACE VIRTUÁLNÍHO MUZEA

Zadáním diplomové práce bylo vytvořit virtuální automobilové muzeum. Po rozboru technologií jsem se tedy rozhodl použít popsané technologie v rešerši a to přesně SPA aplikaci ve Vue.js a PHP framework Symfony na pozadí.

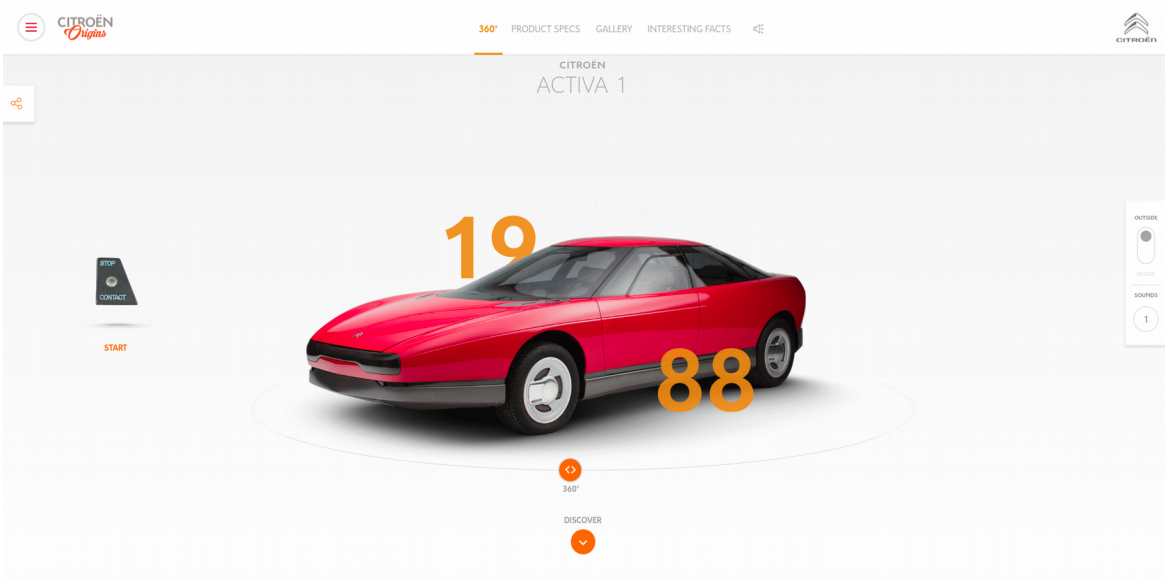
4.1 Analýza

Při analýze bylo potřeba prozkoumat okolní trh pro případnou inspiraci a vytvoření představy, jak by takové muzeum mělo vypadat. Asi jako jediný kvalitní projekt bylo nalezeno muzeum automobilky Citroen. Po rozboru technologií se přišlo na to, že je web postavený na technologii Drupal 7 a jQuery. Tyto technologie jsou vlastně předchůdci mnou vybraných technologií. To znamená, že realizovat podobný projekt by vlastně neměl být velký problém. Jediným problémem, který před realizací vyplynul, je skenování automobilů pro realizaci virtualizace, která je implementovaná v detailu vozů.



Obrázek 18: Automobilové muzeum citrone

V průběhu analýzy jsem se také zaměřil a vizuální stránku věci a uživatelskou přívětivost. Ta byla moc pěkná a interaktivní, protože obsahovala jak pěkné ovládací prvky, tak i zvukové efekty a proto člověka dokázala víc vtáhnout do děje. Při první návštěvě webu si můžeme vybrat i rozsah výrobních dat automobilů a podle toho jednodušeji vybrat období, o které nám jde. Tato funkcionalita je vytvořená moc hezkým způsobem, ale v mém konkrétním řešení jsem se rozhodl jí nepoužít.



Obrázek 19: Detail vozu

Co se týče detailu vozu, je poskytnut pomocí 3D vizualizace. Poskytuje rotaci kolem celého obejktu. Auto je ale ručně modelované a nejedná se o oskenovaný reálný objekt. To by mohl být problém při napodobování, protože vytváření takovýchto objektů je velmi pracné a nejde automatizovat. Na obrázku výše vidíme i ovládací prvky na nastartování auta, které nám přehrají reálný zvuk nastartovaného vozu.

4.2 Nefunkční požadavky

Systémy klientské části a administrační části by měly na sobě fungovat nezávisle, tak aby bylo možné aplikační data použít i na jiné platformy. Kvůli tomu je potřeba zabezpečit aplikaci i proti velkému náporu uživatelů tím, že klientská část je oddělená a v případě výpadku hlavního serveru bude vše dobře fungovat. Návrh aplikace by měl počítat se škálovatelností, aby se v případě potřeb mohla aplikace přesunout na jiné servery.

Aplikace musí být naprogramována v otevřených technologiích proto, abychom docílili co nejnižších provozních nákladů. Je tedy nutné aplikaci koncipovat na platformu LAMP. To znamená, že musí být Aplikace kompatibilní se systém Linux, musí být postavená a spustitelná na aplikačním serveru Apache. Musí také umět komunikovat s otevřenou databází MariaDB a běžet na systému PHP.

4.3 Funkční požadavky

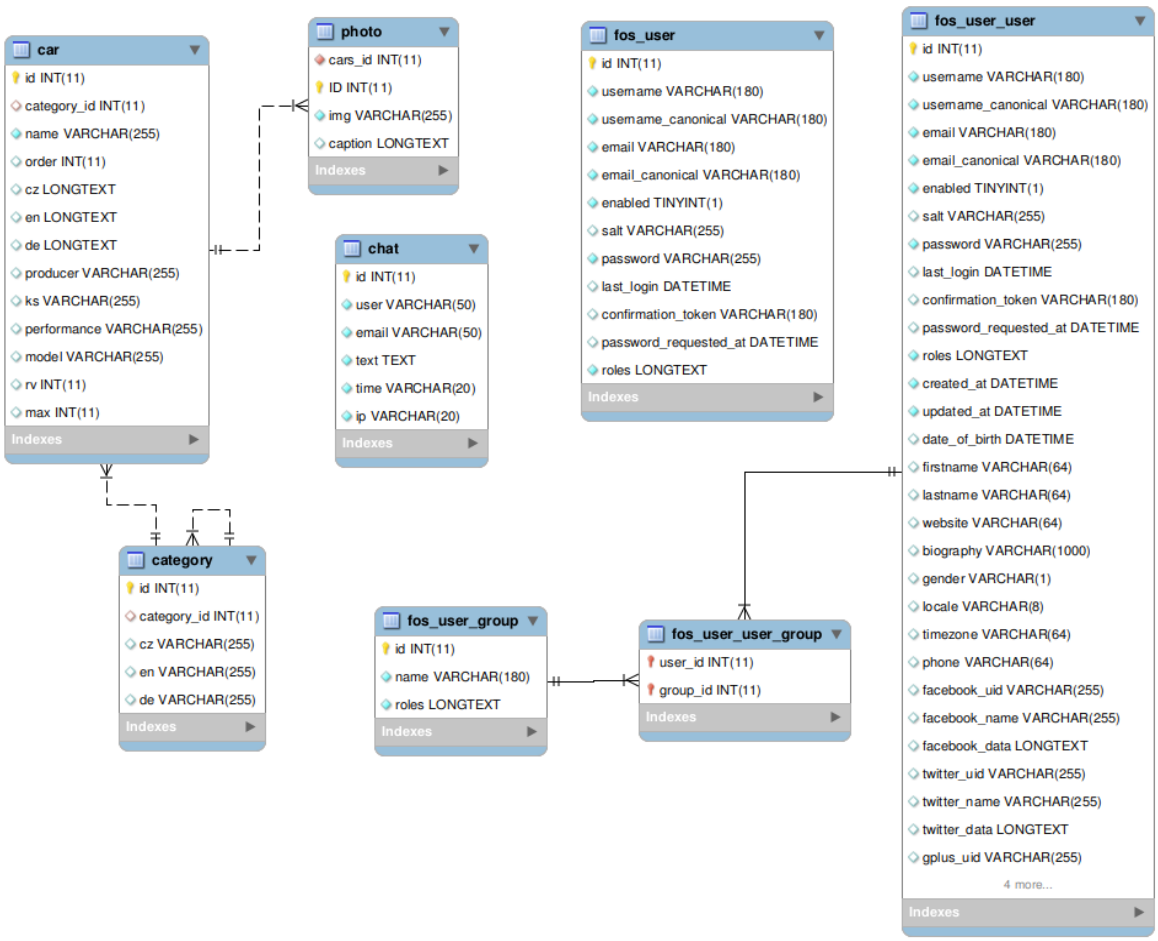
Návrh funkčních požadavků.

- Možnost pracovat s uživateli v administraci – v aplikaci je potřeba umět pracovat s uživateli. To znamená že se ze uživatelé budou moci registrovat, upravovat si své informace, přihlašovat a odhlašovat se.
- Vkládání automobilů – do aplikace musíme umět vkládat mobilní automobily s jejich technickým popisem a krátkou historií. Musíme umět nejen automobily vkládat ale také upravovat a to všechny informace, které o nich máme. Automobily musíme také umět mazat.
- Vkládání fotografií – součástí prezentace bude i fotografická prezentace automobilu. Proto aplikace musí umět pracovat s fotografiemi. Musí je umět vkládat a mazat.
- Zpráva kategorií vozů – vozy musíme umět strukturovat. Proto je nutné umět vkládat kategorie a editovat je. Také musíme umět tyto kategorie přiřazovat vozům.

- Upravování menu – pro lepší orientaci musíme umět pracovat s dynamicky tvořeným menu. To znamená, že podle kategorie vozu se bude generovat menu a odesílat uživateli pro zobrazení.
- Vývojářská konzole – je nutné, aby aplikace obsahovala i konzoly pro přístup k datům přes webové rozhraní. To je nutné zejména pro to, aby se mohla ověřovat data, která jsou k dispozici a zobrazit je vývojářům dalších aplikací, kteří by mohli naše otevřená data používat.
- Náhled vozu – jednotlivé vozy musí mít náhled. Je to z důvodů rychlé orientace na webu pro přejití na detail vozu, kde bude větší množství informací. Náhled by měl obsahovat fotografii a základní výčet parametrů vozu.
- Detail vozu – v detailu, budou všechny informace které jsou o vozu k dispozici. Zároveň bude detail obsahovat interaktivní galerii. Ta bude mít fotografie s popiskem pro příslušný automobil.
- Přizpůsobitelný vzhled – vzhled stránky by měl být lehce přizpůsobitelný všem zařízením tak, aby se jednoduše prohlížely napříč všemi mobilními zařízeními, tablety i osobními počítači.
- Interaktivní galerie – galerie bude provedena tak, aby bylo jednoduché jí ovládat jak na mobilní zařízeních, tak na velkých počítačích a přizpůsobila se rozměru celé obrazovky. Bude možné se v ní přesouvat na další fotografie a bude obsahovat i popisek. Měla by být rychlá a znovupoužitelná.
- Přidávání stránek – bude také možné přidávat další stránky které nebudou obsahovat automobily, ale budou sloužit pro různé akce nebo popisy stránek, jakou jsou uvítací stránky a podobně.

4.4 Návrh databáze

Návrh databáze byl zhotoven v nástroji MySQL Workbench. Jedná se o základní nástroj pro práci s MySQL databází a protože je kompatibilní i s dalšími dal využít i pro připojení k MaridDB. Při návrhu databáze se musely dělat některé ústupky vzhledem k tomu, že databáze pochází ze straší verze aplikace a z pohledu zpětné kompatibility musela zůstat zachována.



Obrázek 20: Návrh databáze

4.5 Implementace

Při řešení konkrétní aplikace byla zvolená zvláštní kombinace pro jednostránkovou a více stránkovou prezentaci. Tyto technologie byli rozděleny podle použitelnosti a rychlosti vývoje.

První část je prezentační. Ta byla navržena formou jednostránkové aplikace. Ta dovolí běžnému návštěvníkovy procházet prezentaci a jednoduše procházet mezi daty. Tato část je plně responzivní, a proto by se měla dobře zobrazovat na všech zařízeních od malých zařízení po velké displeje.

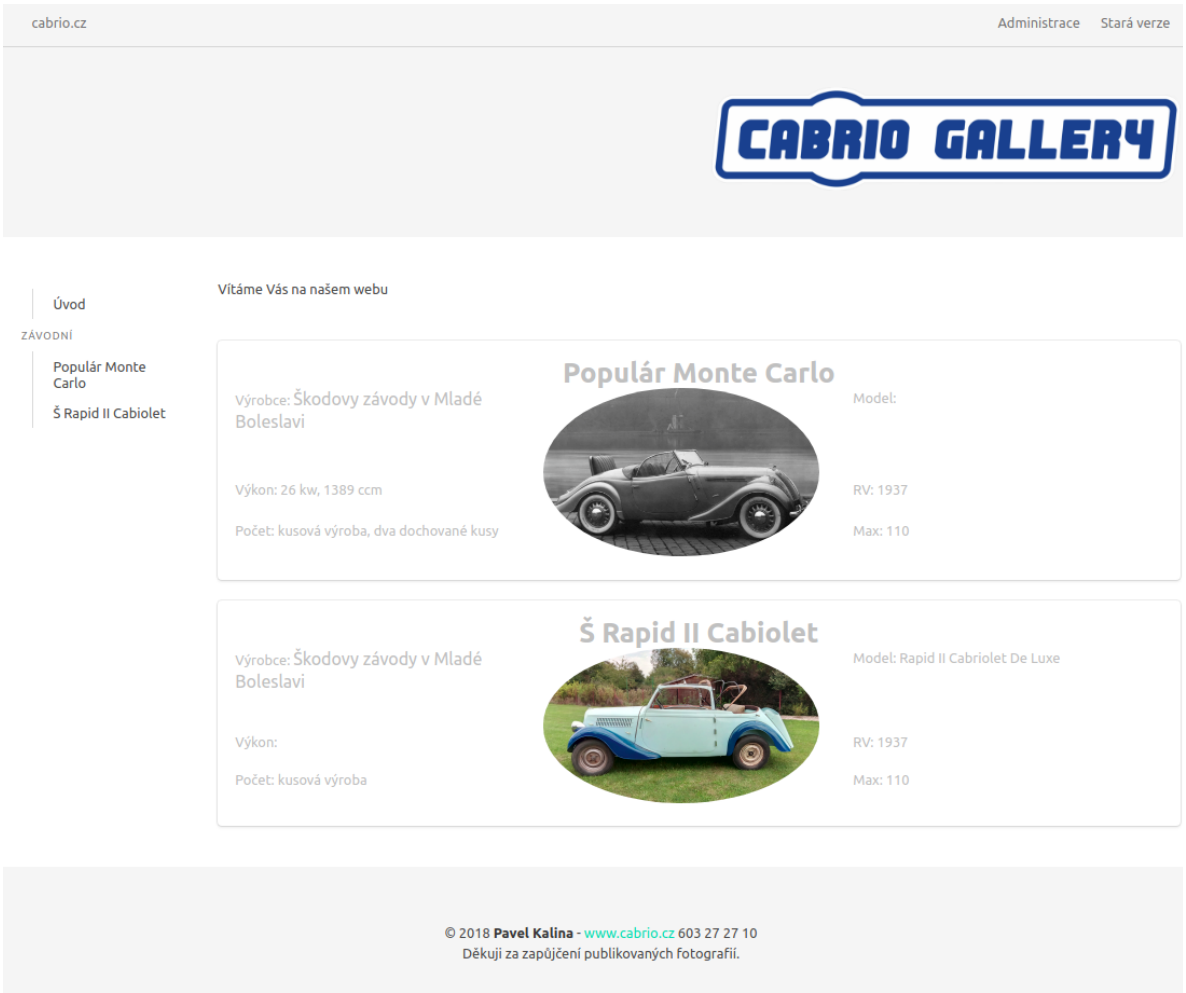
Druhá část se označuje jako administrační. V této části byla zhotovena komplet celá administrace databáze a fotografií. Také se zde implementovalo rozhraní pro komunikaci mezi serverem a prezentační aplikací. Nachází se zde kompletní administrace ostatních uživatelů popřípadě administrátorů a jejich dat. Dále se uživatelé mohou jednoduše přihlašovat a následně upravovat data. Ty mají k dispozici jak v přehledných výpisech které mohou řadit, filtrovat, ale také i v detailech, které mohou upravovat. Mohou je také přidávat. Dají se zde upravovat jak data v databázi, tak fotografie které jsou uloženy na serveru.

4.5.1 Metodika vývoje.

Protože při návrhu aplikace bylo vycházeno z předešlé špatně navržené a zastaralé aplikace, bylo nutné udělat několik ústupků. Protože jsme na začátku návrhu aplikace nemohli vědět, co nám technologie dovolí, byli jsme se nucení zvolit cestu agilního vývoje. To v našem případě spočívalo v tom, že jsme vytvářeli aplikaci přírůstkově a to tak, aby při každé provedené práci byl jistý kus vidět. Proto jsme každou dílčí část aplikace probíraly s klientem a tudíž jí vyvíjet tak, abychom docílili co největšího úspěchu. Protože velké množství aplikací při metodikách typů vodopád ztroskotá na špatném navržení aplikace, my jsme se rozhodli tento návrh dělat agilně a při každé dílčí činnosti.

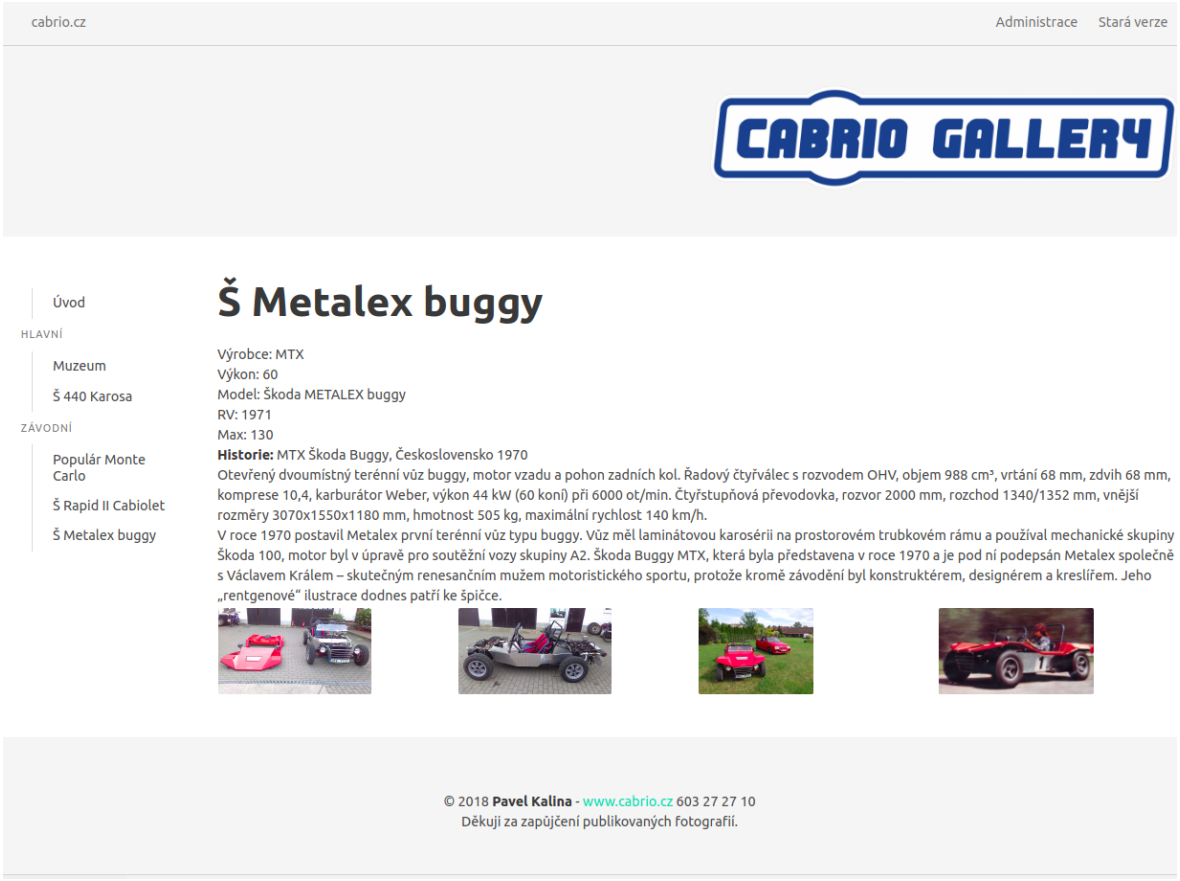
4.5.2 Prezentační část

Při vstupu na hlavní stránku (Obrázek 21) nás čeká základní vstupní bod jednostránkové aplikace. Jedná se o JavaScriptovou aplikaci napsanou v jazyce Vue.js. Jako CSS framework zde byla použita Bulma. Ta je velmi odlehčená a dobře kompatibilní s Vue. Je postavená na flexboxu, což je pozicovací nástroj implementovaný v CSS. Na vstupní stránce vidíme miniatury vozů. Ty jsou distribuovány přes GraphQL jedním dotazem na server což, šetří komunikační prostředky. Zároveň máme k dispozici menu pro přepínání mezi vozy nebo můžeme jednoduše rozkliknout detail vozu přes obrázek.



Obrázek 21: Uživatelská část s miniaturami

Při přejití na vůz získáme jeho detail. Ten nám poskytuje rozšířené informace o vozu. Informace jsou nastavitelné v administraci a podle toho, které jsou nastavené se zobrazí. Dále máme k dispozici popis vozu, který je adminitrován pomocí wysivig editoru. Tento text se pak zobrazuje jako surová HTML data a tudíž má uživatel možnost provádět všechny pokročilé práce s textem.

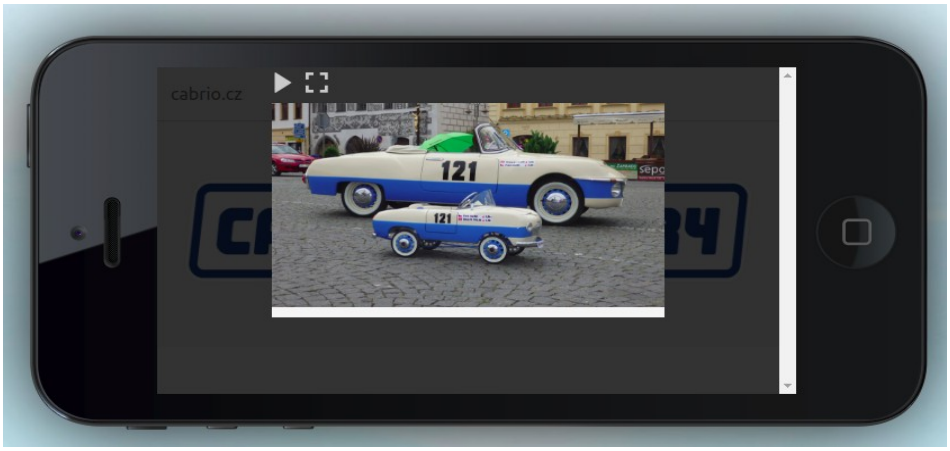


Obrázek 22: Detail vozu

V detailu vozu máme k dispozici veškeré podrobnosti z databáze. Můžeme zde vidět i popis vozů, který může být klientsky implementován i pomocí HTML. Dále zde máme i responzivní fotografii. Ta umí pomocí ovládacích prvků vytvořit prezentaci a posouvat fotky automaticky nebo roztáhnout fotografii přes celou stránku.

Obrázek 23: Galerie

Velkou výhodou při používání CSS frameworků je to, že při jejich správném použití můžeme jednoduše dosáhnout kvalitní responzivity. Toho dosáhneme tak, že dobře použijeme kvalitní koncept návrhu. V tomto konkrétním případě byla použita Bulma. Ta umí základní prvky přizpůsobit na stránku tak aby vypadaly dobře.



Obrázek 24: Náhled na responzivní fotogalerii

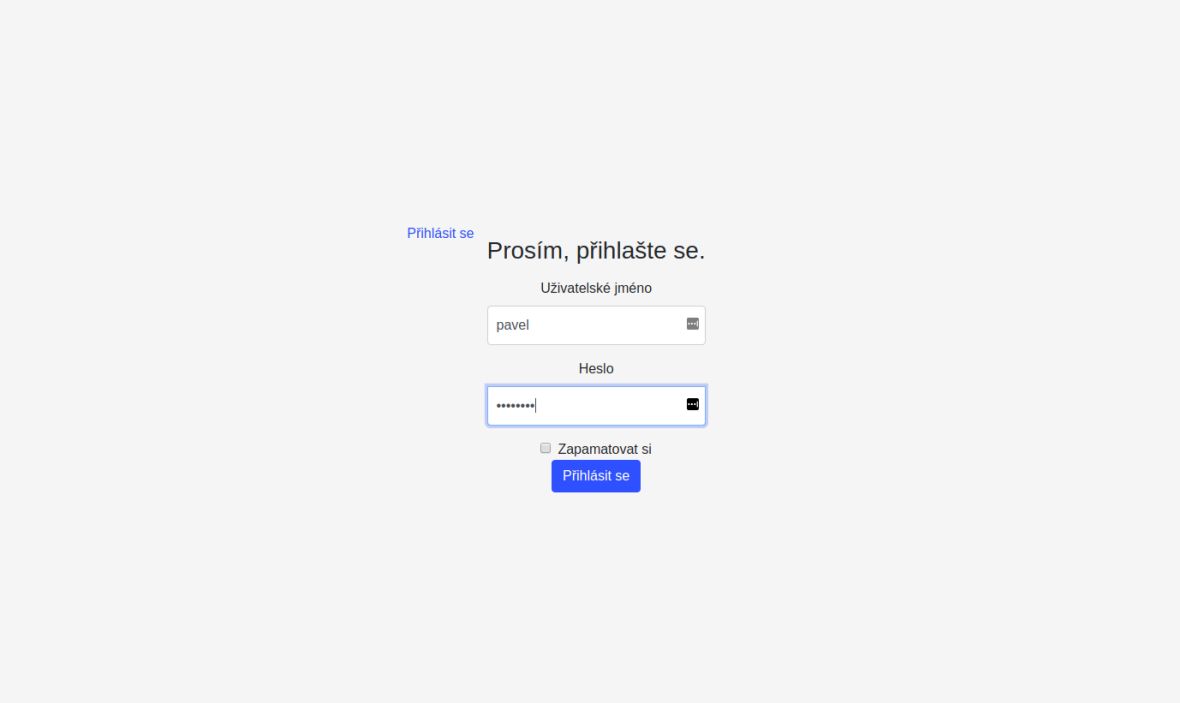
Proto, abychom docílili dobrého pořadí skládání velkých celků jako je obsah nebo menu, je použit Flexbox, který je novou funkcí v kaskádových stylech. Na obrázku č. 25 vidíme základní ukázku kódu který toto zajišťuje. Další podrobnosti nalezneme v příloze.

```
<template>
  <div class="home">
    <h1>{{ msg }}</h1>
    <br><br>
    <carsWidget></carsWidget>
  </div>
</template>

<script>
  import carsWidget from './CarsWidget'

  export default {
    components: {
      carsWidget: carsWidget
    },
    name: 'home',
    data () {
      return {
        msg: ''
      }
    }
  }
</script>
```

Obrázek 25: Ukázka zdrojového kódu

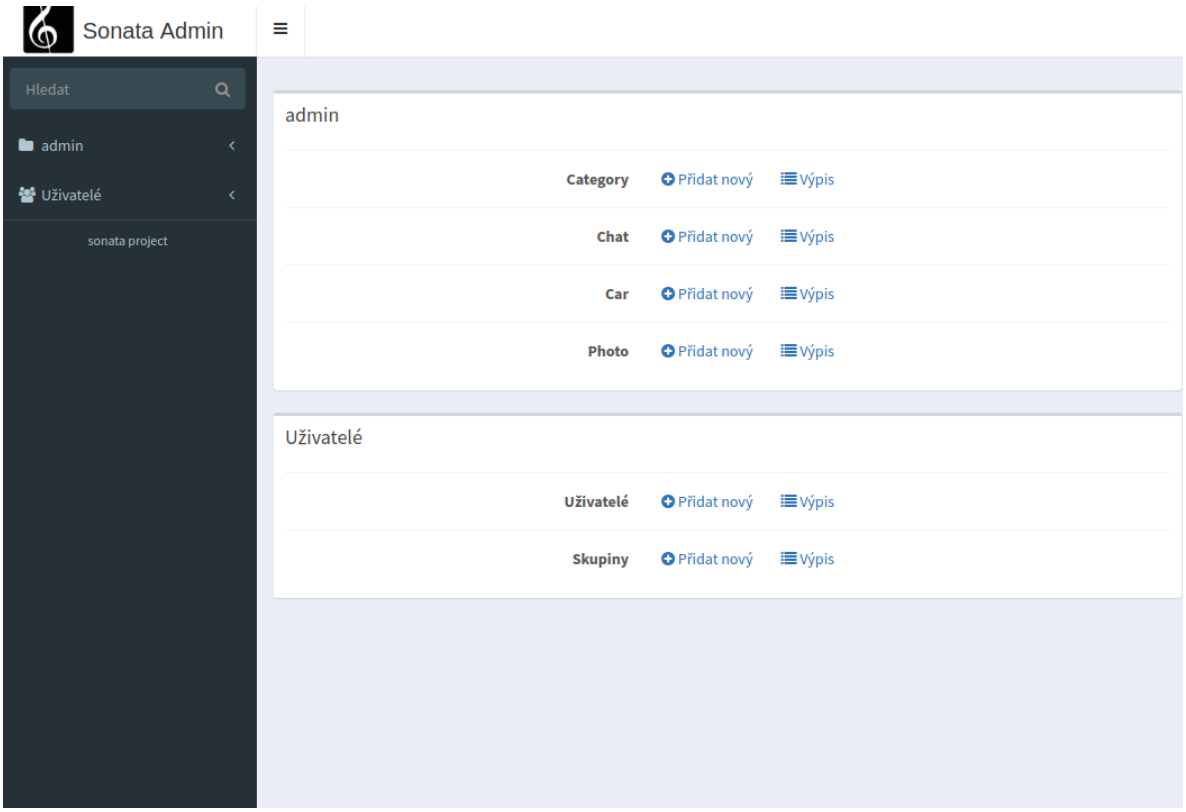
4.5.3 Administrační část

Administrační část byla naprogramována v programovacím jazyce PHP s Frameworkem Symfony. První věc, co uživatel uvidí před přihlášením do administrace, je přihlašovací obrazovka. Ta je implementována přes Symfony FOS User Bundle. To je balíček, který obsahuje všechny potřebné operace které můžeme s uživatelem vytvářet. Obsahuje i základní vzhled přihlašovací obrazovky a přihlašování. Přihlašovací obrazovka je ještě dostylována pomocí základního stylovacího frameworku Bootstrap.



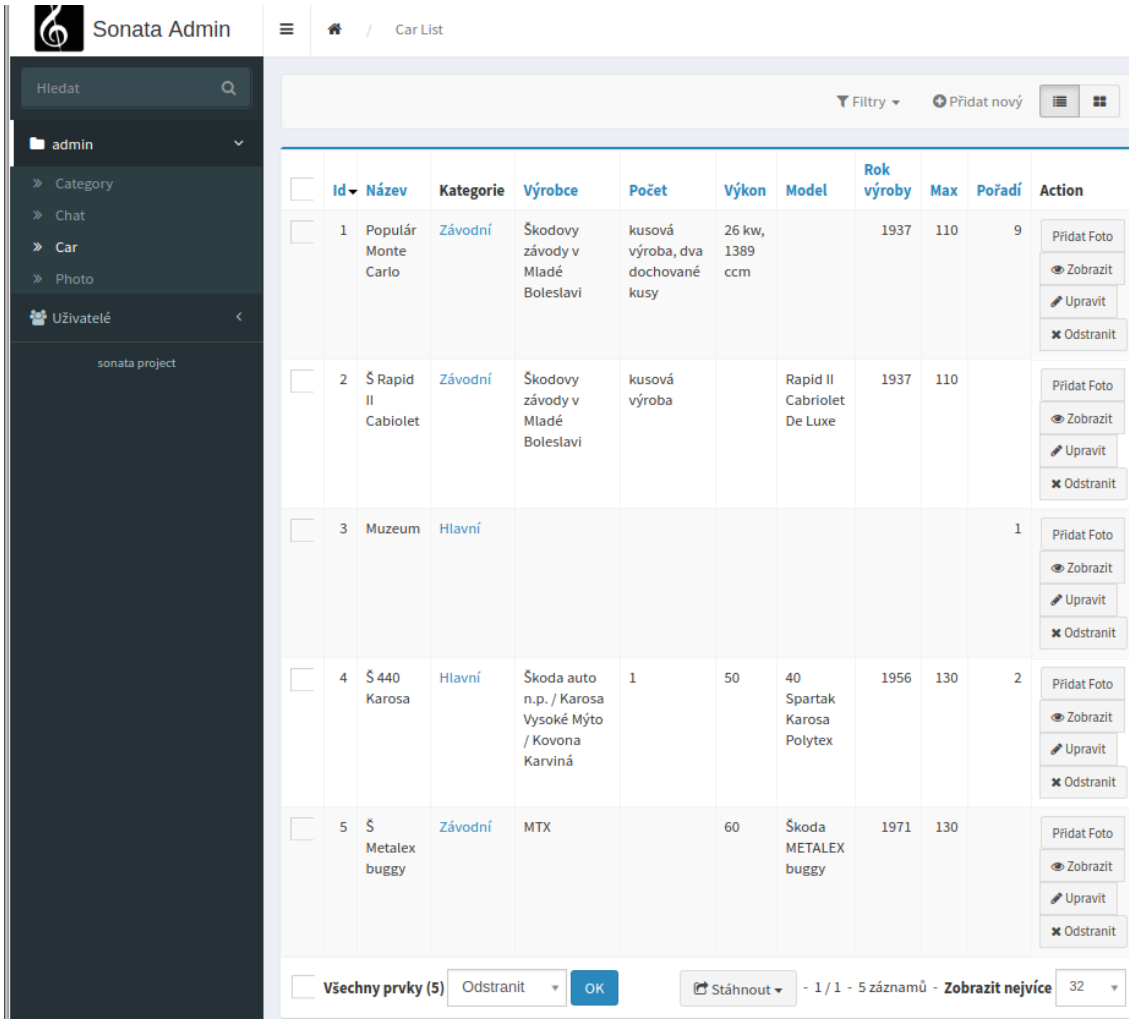
Obrázek 26: Přihlašovací obrazovka aplikace

Po přihlášení se dostaneme do uživatelského rozhraní pro základní administraci. Ta je implementovaná v rozšíření Sonata Bundle. Ten zjednodušuje implementaci na základní operace nad entitami. Díky tomu můžeme tvořit rychlé administrace se znovupoužitelným kódem, který vzniká na základní kostře těchto projektů. Proto je možné jednoduše pracovat v teamu a rozšiřovat aplikace synchronně a organizovaně. Ve standardním nastavení obsahuje výpis, který vypisuje seznam všech entit podle nastavení. Dále můžeme jednoduše přidávat do databáze pomocí jednoduchého formuláře. Můžeme použít i wysiwyg editor pro to, aby uživatel mohl vkládat bezpečné kód, který není zabezpečen proti všem formám hackingu.



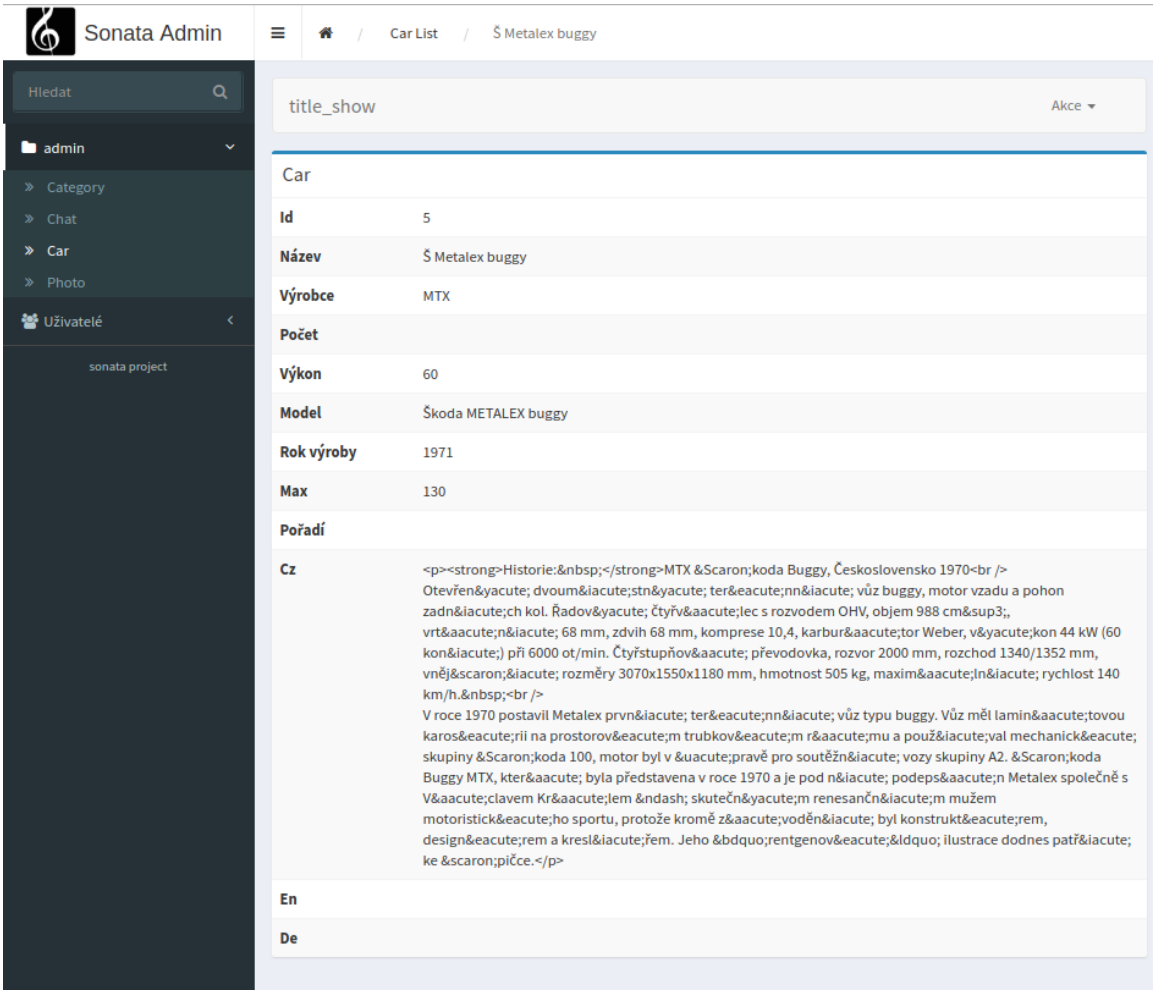
Obrázek 27: Základní vzhled Sonata Bundle

Při přechodu na libovolný odkaz pro výpis se nám vypíše konzistentní výpisová tabulka. Ta zajišťuje, že se všechny výpisy budou chovat standardně na základě předdefinovaných pravidel. Základem tohoto listu jsou všechny odkazy na všechny operace pro editaci řádku. Tyto operace můžeme provádět i hromadně díky akčnímu výběru. Tabulka také v základu obsahuje stránkování s nastavitelným počtem výskytů. Základem je také katalogový výpis, kterým je možné vypsat karty vozů pro lepší orientaci. Nad celou tabulkou je také možné filtrovat. Tento filtr je možné nastavit programově nezávisle na zobrazovaných parametrech, takže je možné nastavit filtr i ve schovaných atributech. Lze také bez problému řadit formulář podle všech atributů a na základě toho ho procházet.



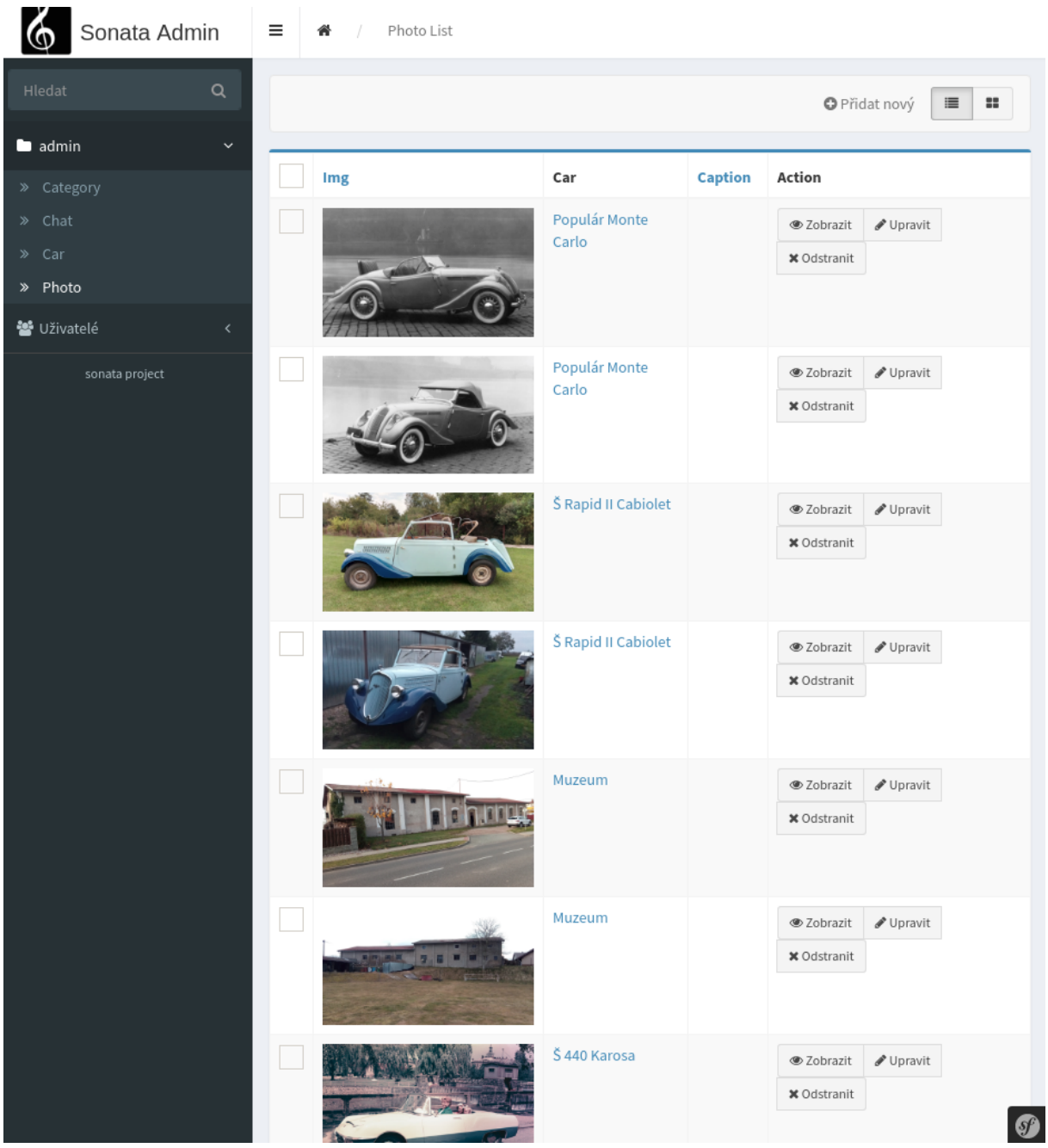
Obrázek 28: List vozů

Vedle listu s přehledem máme k dispozici i detail. V něm máme k dispozici na nahlédnutí všechny prvky entity. Tyto přehledy jsou přizpůsobitelné a je možné je upravit pro každou entitu zvlášť. To nám poskytuje velkou variabilitu při pohledu na jakoukoli entitu požadujeme.



Obrázek 29: Detail vozu v administraci

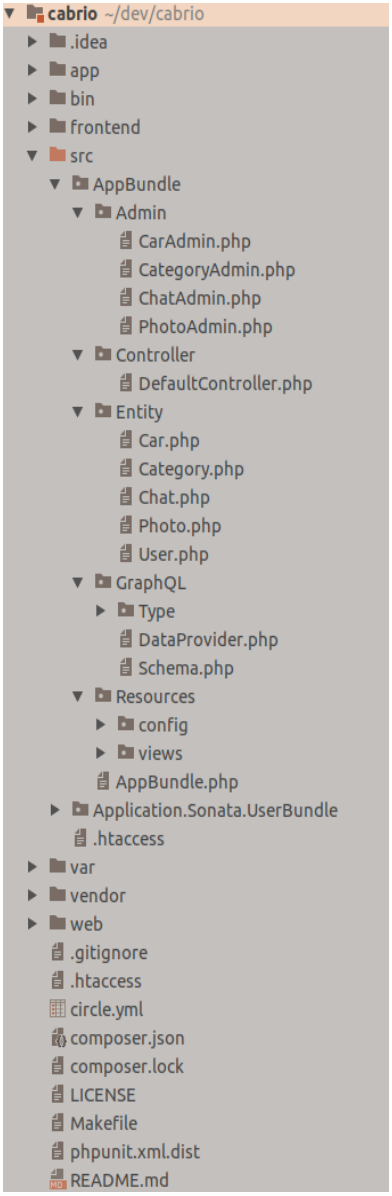
V administraci máme také možnost upravovat jak všechny fotografie, tak fotografie řazené podle zvoleného vozu. Je to díky tomu, že zde byla administrace fotografií tvořena formou potomka, to nám umožňuje jednoduše dělat stejné formuláře s předfiltrovanými daty pro danou kategorii.



Obrázek 30: Administrace fotografií

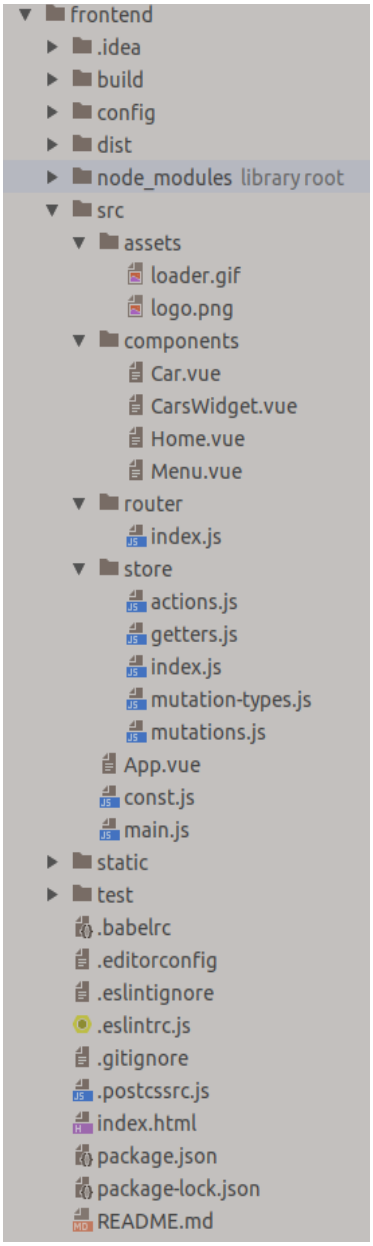
4.6 Struktura projektu

Na začátku vytváření aplikace byl projekt rozdělen na dva samostatné projekty. Toto řešení se následné ukázalo jako špatné. Příčinou bylo špatné přepínání kontextu při vývoji. Projekty byli vyvíjeny v GIT repozitáři a tak bylo nutné je sloučit. Po sloučení vznikl v hlavním PHP projektu nový podadresář fronend který obsahuje celý projekt SPA aplikace. Technologie jsou stále oddělené jen se jednodušeji dělají spojené části.



Obrázek 31: Struktura PHP projektu

V JavaScriptové části je ponecháno standardní rozložení Vue aplikace, kde byly akorát doplněny potřebné třídy pro zobrazení dat ze serverové části.

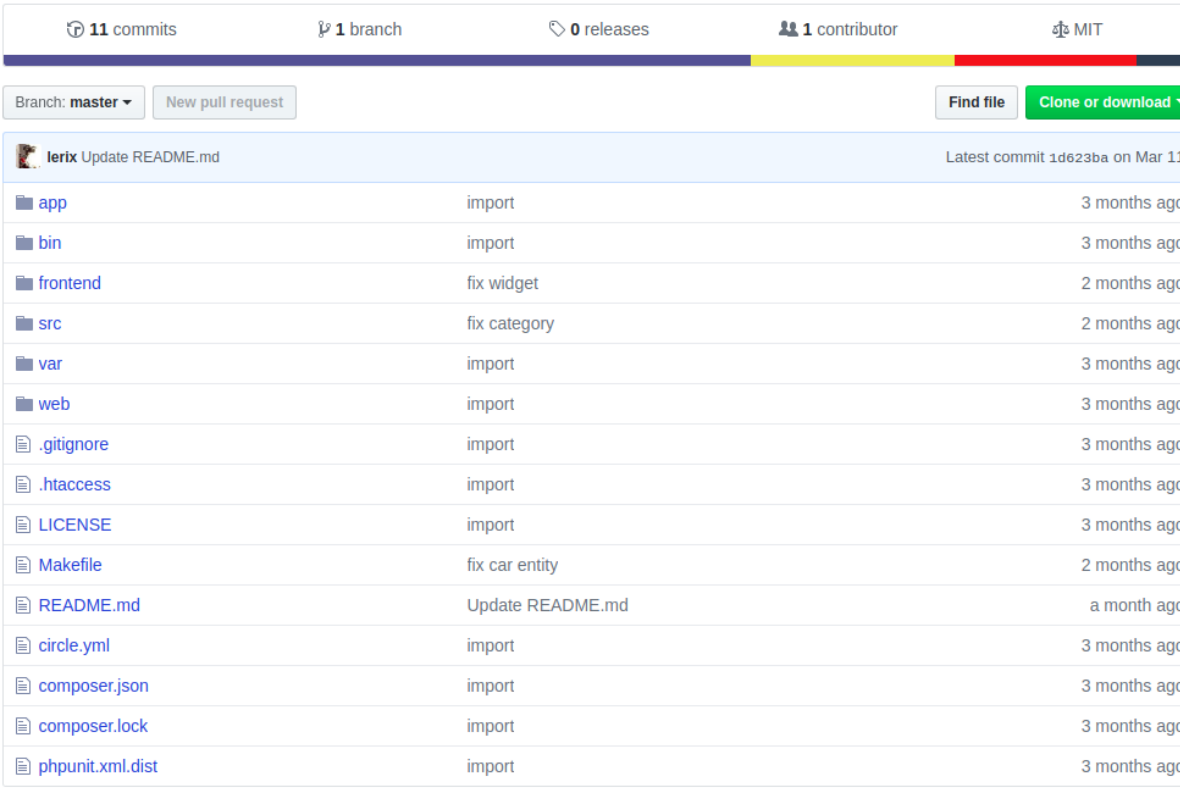


Obrázek 32: Struktura JS projektu

4.7 Technologie

4.7.1 Git

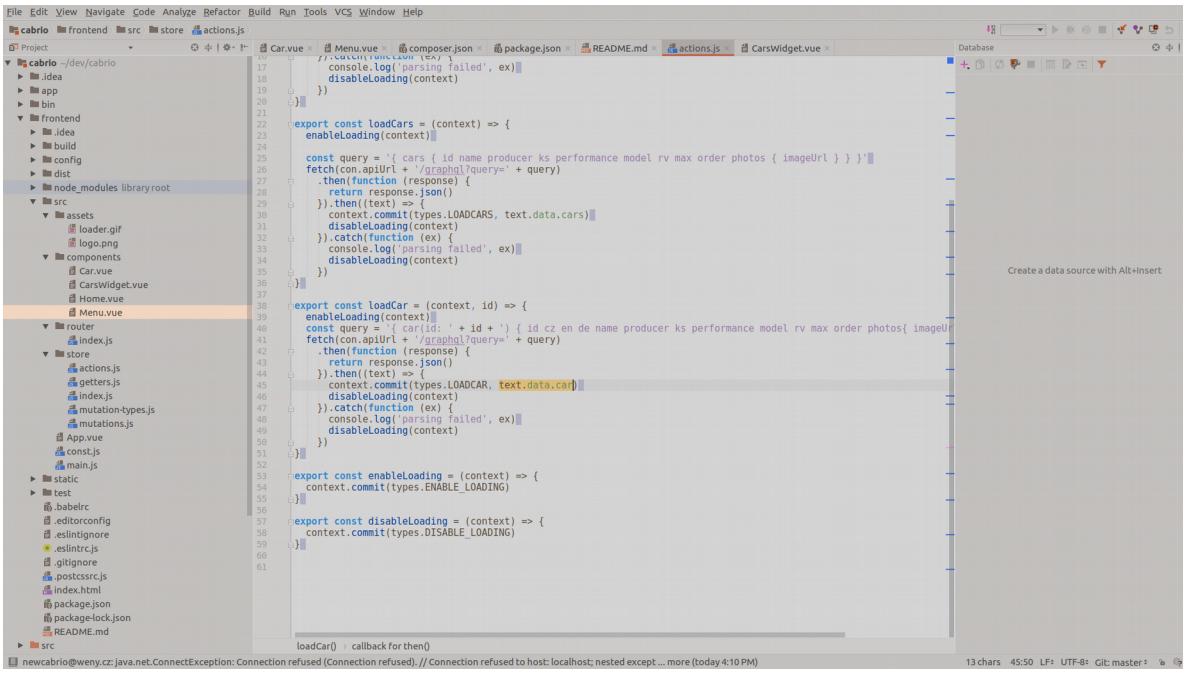
Nástroj Git je svobodný nástroj, který byl původně vyvinut pro potřeby systému Linux. Slouží na verzování projektů. Je dobrým společníkem pro vývoj software, ale dá se také použít pro dokumenty. Při práci z více zařízeními zajišťuje propojení a zálohu. Pro potřeby diplomové práce byly využíváný služby dvou poskytovatelů Git repository. Na praktickou část bylo využito nejdříve služeb společnosti Atlassinan a to Bitbucket. Ten poskytuje zdarma soukromé uložiště. Později však proběhla migrace na Github, protože se jedná o vývojářskou sociální síť, kde byl projekt poskytnut veřejně. Bitbucket byl však použit na psaní dokumentové části, kde docházelo k přechodu mezi několika zařízenímy.



Obrázek 33: Projekt na Githubu

4.7.2 IDEA

Pro vývoj byly používány produkty od společnosti IntelliJ. Jedná se o jediný placený produkt, který byl pro vývoj použit. Konkrétně bylo testováno prostředí Phpstorm a Idea. Tyto verze vývojového nástroje se od sebe liší pouze doplňky. Prostředí Idea totiž umí pracovat se všemi programovacími jazyky. Nevýhodou ale je jeho velká cena. Pro naše konkrétní potřeby by nám stačila verze Phpstorm která umí pracovat jak s PHP, tak s JavaScriptem. Tato verze je díky menšímu počtu funkcí zhruba o půlku levnější. Oba tyto nástroje vznikají od vývojářů pro vývojáře. To znamená, že má všechny vestavené funkce, které využijete, na dosah. S tímto nástrojem roste taky velká produktivita práce díky dobrému zpracování. Jedna z nejsilnějších zbraní, které oproti ostatním prostředí má, je indexace projektu. To nám zajišťuje to, že můžeme jednoduše procházet zdrojové kódy a vyhledávat s velkou rychlostí nad celým projektem. Dále umí bezvadně pracovat s verzovacím systémem Git a to v takové míře, že obsahuje všechny jeho funkce a nepotřebujeme tudíž používat konzolové ovládání. Dále stojí také za zmínku velká podpora klávesových zkratek a přizpůsobitelnost jak lokální historie souborů, tak historie z Gitu. Dobře funguje také interní nástroj pro zprávu databáze, kde máme po ruce všechny možné ovladače pro připojení. Nástroje od firmy IntelliJ ušetří při programování spoustu nepříjemností a určitě stojí za používání.



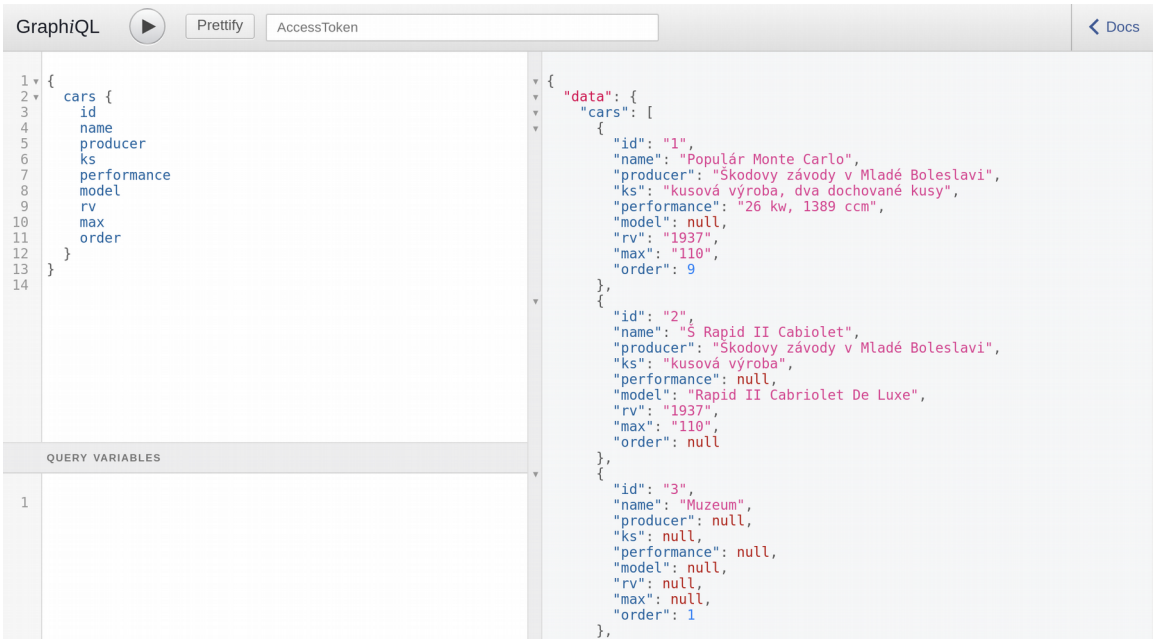
Obrázek 34: IntelliJ IDEA

4.7.3 Vue-images

Jedná se implementaci fotografické galerie. Knihovna je šířena jako open-source na síti Github. Byla použita ve verzi 1.1. V rámci diplomové práce byla zduplikována a upravována pro potřeby projektu. Původně totiž knihovna nepracovala správně s proměnnými v jazyce Vue a při změně kontextu se nepřekreslovala. Toto bylo upraveno, ale je potřeba jí nadále zdokonalovat. Jelikož se jedná o otevřený projekt, tak toto není problém. Nevýhodou tohoto řešení je ale tristní dokumentace, která je mnohdy s asijskými znaky. Platí proto více než jinde, že nejlepší dokumentace je zdrojový kód. Tato knihovna je responzivní a proto poskytuje dobré zobrazení fotografií jak na mobilním telefonu, tak na klasickém osobním počítači.

4.7.4 Symfony GraphQl Bundle

Pro vývoj webové služby byla vybrána knihovna Symfony GraphQL Bundle ve verzi 1.2. Jak název napovídá umožňuje nám jednoduší vývoj webové služby v jazyce GraphQL. Obsahuje také konzoly pro vývojáře, která nám pomáhá s vývojem a sestavováním dotazů. Umožňuje nám psát jednoduché datové typy podle reálných entit v projektu. Tento dotaz se potom odešle na server a zobrazí v pěkné podobě. Díky této konzoly si vždy můžeme ověřit funkčnost požadavku, tedy zda-li vrací požadovaná data.



Obrázek 35: Vývojářská konzole GraphQL

4.7.5 Verze a prostředí

Verze a závislosti pro php prostředí ze souboru composer.json.

php	>=7.0
doctrine/doctrine-bundle	^1.6
doctrine/orm	^2.5
incenteev/composer-parameter-handler	^2.0
sensio/distribution-bundle	^5.0.19
sensio/framework-extra-bundle	^3.0.2
sonata-project/admin-bundle	^3.20
sonata-project/doctrine-orm-admin-bundle	^3.1
sonata-project/easy-extends-bundle	^2.2
sonata-project/formatter-bundle	^3.2
sonata-project/user-bundle	dev-master
symfony/monolog-bundle	^3.1.0
symfony/polyfill-apcu	^1.0
symfony/swiftmailer-bundle	^2.3.10
symfony/symfony	3.3.*
twig/twig	^1.0||^2.0
youshido/graphql-bundle	^1.2

Tabulka 1: Závislosti PHP aplikace

Verze závislostí na JavaScript prostředí, ze souboru package.json

node	>= 4.0.0
bulma	^0.5.1
es6-promise	^4.2.2
vue	^2.3.3
vue-images	^1.0.10
vue-router	^2.6.0
vuex	^2.3.1
npm	>= 3.0.0
whatwg-fetch	^2.0.3

Tabulka 2: Závislosti JavaScript aplikace

ZÁVĚR

Cílem diplomové práce bylo rozebrat možnosti vytvoření moderní webové aplikace v technologii JavaScript. Byli porovnány jak technologie dřívějších let, tak ty současné. Společně s technologiemi byli rozebrány i možnosti vytváření konzistentních vzhledů, a to k jednotlivým technologiím jak je nejlepší je používat. Také byly popsány základní technologie pro programování na straně serveru s různými technologiemi, které napomáhají chodu a vývoji v jazyce PHP. Součástí práce bylo prozkoumat možnosti moderních webových služeb.

V praktické části byly splněny všechny požadavky ze zadání. Byla provedena analýza konkurenčního produktu a vytvořeny funkční a nefunkční požadavky pro potřebu vytvoření aplikace. Podle těchto výsledků a na základě rešerše byla vytvořena moderní jednostránková aplikace pro prezentační vrstvu a klasická vícestránková administrace pro zprávu dat. Prezentační část obsahuje rozhraní pro uživatelsky přívětivé prohlížení aplikace. V administraci je možné obsluhovat data na základě přihlášení uživatele. Tato administrace je navržena tak, aby bylo možné ji jednoduše rozšiřovat a byla uživatelsky přívětivá. Máme v ní možnost kompletně editovat data a provádět s nimi všechny možné operace. Pro komunikaci mezi administrační částí a serverem musela být také navržena webová služba, pomocí které můžeme distribuovat data i do dalších aplikací.

Už dnes můžeme říct, že moderní JavaScript je kompilovaným jazykem a nejspíše to tak zůstane. Ale máme dnes už takové nástroje, které nám s ním dovolí lehce pracovat. Díky Reactu a jeho technologii virtuální DOM, která je průlomovým objevem, už nejsme odkázání na čistý kód a nebo jQuery, ale můžeme používat daleko kvalitnější a čistší řešení. Každopádně to vypadá na to, že už se technologie založené na JavaScriptu pomalu stabilizují, jako ostatní jazyky. Z pohledu administrační části se poslední dobou nic nezměnilo a tyto technologie jsou velmi vyladěny pro produkční nasazení s veškerou dokumentací. Oproti tomu můžeme říct, že novinkou ve světe webových služeb je jazyk GraphQL. S tímto dotazovacím jazykem nyní můžeme provádět na vrstvě API velmi pokročilé práce bez větších omezení.

POUŽITÁ LITERATURA

[1] NEOTERIC. *Single-page application vs. multiple-page application* [online]. 2016 [cit. 2018-05-01]. Dostupné z: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

[2] ULLMAN, Larry E. Modern JavaScript: develop and design. Berkeley, CA: Peachpit Press, c2012. Develop and design. ISBN 9780321812520.

[3] MICHÁLEK, Martin. *CSS frameworky pro masy*[online]. 2009 [cit. 2018-05-01]. Dostupné z: https://www.zdrojak.cz/serialy/css-frameworky/

[4] *Bootstrap* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://getbootstrap.com/

[5] THOMAS, Jeremy . *Bulma* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://bulma.io/alternative-to-bootstrap/

[6] *Developer Graphic* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://dotcms.com/blog/images/Microsoft%20Developer%20Network%20Graphic.png

[7] JUICY MEDIA LIMITED. *HTML, CSS & JavaScript* [online]. [cit. 2018-05-01]. Dostupné z:] https://www.juicymedia.co.uk/services/development/html-css-js

[8] LASN, Indrek. *All You Need To Know About CSS-in-JS* [online]. 2017 [cit. 2018-05-01]. Dostupné z: https://hackernoon.com/all-you-need-to-know-about-css-in-js-984a72d48ebc

[9] ULLMAN, Larry E. Modern JavaScript: develop and design. Berkeley, CA: Peachpit Press, c2012. Develop and design. ISBN 9780321812520.

[10] FRANÇOIS ZANINOTTO AND FABIEN POTENCIER. The definitive guide to symfony. New ed. Berkeley, Calif: Apress, 2007. ISBN 9781590597866.

[11] *JQuery Syntax - action on HTML elements* [online]. 2009 [cit. 2018-05-01]. Dostupné z: http://referencedesigner.com/tutorials/jquery/jq_4.php

[12] SRIDHAR, Jay. *What Is JavaScript and How Does It Work?* [online]. 2017 [cit. 2018-05-01]. Dostupné z: https://www.makeuseof.com/tag/what-is-javascript/

[13] HEIN, Rich. 6 Things You Need to Know About jQuery [online]. 2012 [cit. 2018-05-01]. Dostupné z: https://www.cio.com/article/2394876/java/6-things-you-need-to-know-about-jquery.html

[14] GASSTON, Peter. The modern Web: multi-device Web development with HTML5, CSS3, and JavaScript. San Francisco: No Starch Press, 2013. ISBN 1593274874.

[15] CORCOS, Sam. *Where to Hold React Component Data: state, store, static, and this* [online]. 2016 [cit. 2018-05-01]. Dostupné z: https://medium.freecodecamp.org/where-do-i-belong-a-guide-to-saving-react-component-data-in-state-store-static-and-this-c49b335e2a00

[16] CORCOS, Nitin. *What Is ReactJS and Why Should We Use It?* [online]. 2017 [cit. 2018-05-01]. Dostupné z: https://www.c-sharpcorner.com/article/what-and-why-reactjs/

[17] DEVACRON, Nitin. *What Is ReactJS and Why Should We Use It?* [online]. 2017 [cit. 2018-05-01]. Dostupné z: http://www.devacron.com/write-your-own-virtual-dom/

[18] SAFARI BOOKS. *Lifecycle hooks* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://www.safaribooksonline.com/library/view/full-stack-vuejs-2/9781788299589/c3bb3fdf-3850-4518-99d7-a281b4732c46.xhtml

[19] YOU, Evan. *The Progressive JavaScript Framework*[online]. 2018 [cit. 2018-05-01]. Dostupné z: https://vuejs.org/

[20] ČÁPKA, David. *Úvod do CSS frameworku Bootstrap* [online]. 2017 [cit. 2018-05-01]. Dostupné z: https://www.itnetwork.cz/html-css/bootstrap/uvod-do-css-frameworku-bootstrap

[21] *About Node.js* [online]. 2017 [cit. 2018-05-01]. Dostupné z: https://nodejs.org/en/about/

[22] MIKSU, Vojtech. *About Node.js* [online]. 2017 [cit. 2018-05-01]. Dostupné z: https://www.dze-jes.cz/babel.html

[23] THE PHP GROUP. *What can PHP do?* [online]. 2018 [cit. 2018-05-01]. Dostupné z: http://php.net/manual/en/intro-whatcando.php

[24] *How PHP Works* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://www.webucator.com/tutorial/learnphp/php-basics/how-php-works-reading.cfm

[25] ADERMANN, Nils a , Jordi BOGGIANO. *Dependency Manager for PHP* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://getcomposer.org

[26] POTENCIER, Fabien. *Symfony is a set of reusable PHP components...* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://symfony.com/

[27] SONATA CONTRIBUTORS. *ABOUT SONATA*[online]. 2018 [cit. 2018-05-01]. Dostupné z: https://sonata-project.org/about

[28] *Getting Started with Doctrine* [online]. 2018 [cit. 2018-05-01]. Dostupné z: http://doctrine2.readthedocs.io/en/latest/tutorials/getting-started.html

[29] *Database (DB)* [online]. 2014 [cit. 2018-05-01]. Dostupné z: https://searchsqlserver.techtarget.com/definition/database

[30] MARIADB FOUNDATION. *About MariaDB* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://mariadb.org/about

[31] CODEXPERTISE. *WEB SERVICES* [online]. 2018 [cit. 2018-05-01]. Dostupné z: http://codexpertise.com/web-services

[32] STUBAILO, Sashko. *GraphQL vs. REST* [online]. 2017 [cit. 2018-05-01]. Dostupné z: https://dev-blog.apollodata.com/graphql-vs-rest-5d425123e34b

[33] FACEBOOK INC. *A query language for your API*[online]. 2018 [cit. 2018-05-01]. Dostupné z: https://graphql.org/

[34] LEVIN, Guy. *RESTful API Authentication Basics*[online]. 2016 [cit. 2018-05-01]. Dostupné z: https://blog.restcase.com/restful-api-authentication-basics/

PŘÍLOHY

PŘÍLOHA A – ZDROJOVÉ KÓDY NÁHLEDU VOZU

```
<template>
  <div class="carsWidget">
    <div class="carWidget box " v-for="car in getCars">
      <table class="carWidgetTable">
        <tr>
          <td colspan="3" style="text-align: center">
            <h2 class="atribut title">{{ car.name }}</h2>
          </td>
        </tr>
        <tr>
          <td width="33%">
            Výrobce: <span class="atribut subtitle">{{ car.producer }}</span>
          </td>
          <td rowspan="3" width="33%">
            <router-link :to="{ path: '/car', query: { carId: car.id }}">
              <img class="miniatur" v-if="car.photos.length!=0" :src="car.photos[0].imageUrl">
            </router-link>
          </td>
          <td width="33%">
            <span class="atribut">Model: {{ car.model }}</span>
          </td>
        </tr>
        <tr>
          <td>
            <span class="atribut">Výkon: {{ car.performance }}</span>
          </td>
          <td>
            <span class="atribut">RV: {{ car.rv }}</span>
          </td>
        </tr>
        <tr>
```

```
          <td>
            <span class="atribut">Počet: {{ car.ks }}</span>
          </td>
          <td>
            <span class="atribut">Max: {{ car.max }}</span>
          </td>
        </tr>
      </table>
    </div>
  </div>


</template>

<script>
  import { mapGetters, mapActions } from 'vuex'
  import * as types from '../store/mutation-types'

  export default {
    name: 'carsWidget',
    computed: mapGetters({
      getCars: 'getCars'
    }),
    methods: mapActions({
      loadCars: types.LOADCARS
    }),
    created () {
      this.loadCars()
    }
  }

</script>
```

```
<style scoped>
  tr{
    width: 33%;
  }
  .carWidgetTable {
    width: 100%;
  }
  .atribut {

    color: #c1c1c1;
    flex: 3;
  }
  .atribut:hover {
    color: black;
  }
  .miniatur {
    border-radius: 100%;
    width: 20em;
  }

  .carsWidget {
    /*display: flex;*/
    /*flex-wrap: nowrap;*/
  }
  .carWidget {
    color: #c1c1c1;
  }

</style>
```