



Pedagogická  
fakulta  
Faculty  
of Education

Jihočeská univerzita  
v Českých Budějovicích  
University of South Bohemia  
in České Budějovice

## Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

## Tvorba webu s využitím technologie JAM Stack Web development using the JAM Stack technology

Bakalářská práce

**Vypracoval:** Jakub Minka

**Vedoucí práce:** PaedDr. Petr Pexa, Ph.D.

České Budějovice 2019

**JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH**  
**Pedagogická fakulta**  
**Akademický rok: 2017/2018**

**ZADÁNÍ BAKALÁŘSKÉ PRÁCE**  
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jakub MINKA**  
Osobní číslo: **P15672**  
Studijní program: **B7507 Specializace v pedagogice**  
Studijní obor: **Informační technologie a e-learning**  
Název tématu: **Tvorba webu s využitím technologie JAM Stack**  
Zadávající katedra: **Katedra informatiky**

**Z á s a d y p r o v y p r a c o v á n í :**

Cílem bakalářské práce je zpracovat aktuální problematiku vývoje webu s využitím technologie JAM Stack. Jedná se o způsob tvorby front-endu webu pomocí klientského JavaScriptu, API a Markupu. Díky absenci komunikace se serverem jsou weby vytvořené pomocí této technologie rychlé, díky absenci databází zcela bezpečné. Veškerý kód od JavaScriptu až po Markup je uložen v repozitáři Git a tak je aktualizace stránek velice rychlá. Mezi jeho další nesporné výhody patří využití CDN, která JAM Stacku přináší spolehlivost a škálovatelnost. V teoretické části bude popsána celá technologie JAM Stack, budou porovnány různé metody použití JavaScriptu (knihovny JS nebo frameworky) a API, stejně jako některé druhy Markup open source nástrojů (Grunt, Gulp) i statických generátorů (Jekyll, Hugo). Součástí teoretické části bude i srovnání s dalšími metodami vývoje webu, výhody a nevýhody použití JAM Stacku a způsoby využití v praxi. V praktické části bude tato technologie aplikována na příkladovém webu.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. JAMstack — JavaScript, APIs, and Markup [online]. [cit. 2018-04-03]. Dostupné z: <https://jamstack.org>
2. Netlify: All-in-one platform for automating modern web projects. [online]. [cit. 2018-04-03]. Dostupné z: <https://www.netlify.com/docs/>
3. Cosmic JS — Cloud-Hosted Content Management Platform , API-first CMS [online]. [cit. 2018-04-03]. Dostupné z: <https://cosmicjs.com>
4. Jekyll o Simple, blog-aware, static sites [online]. [cit. 2018-04-03]. Dostupné z: <https://jekyllrb.com>
5. The world's fastest framework for building websites — Hugo [online]. [cit. 2018-04-03]. Dostupné z: <http://gohugo.io>
6. Git [online]. [cit. 2018-04-03]. Dostupné z: <https://git-scm.com>
7. GitHub [online]. [cit. 2018-04-03]. Dostupné z: <https://github.com>

Vedoucí bakalářské práce: **PaedDr. Petr Pexa, Ph.D.**

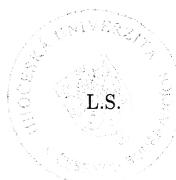
Katedra informatiky

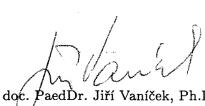
Datum zadání bakalářské práce: **3. května 2018**

Termín odevzdání bakalářské práce: **30. dubna 2019**

  
Mgr. Michal Vančura, Ph.D.

děkan



  
doc. PaedDr. Jiří Vaníček, Ph.D.  
vedoucí katedry

V Českých Budějovicích dne 3. května 2018

## **Prohlášení**

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě - v úpravě vzniklé vypuštěním vyznačených částí archivovaných pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích interaktivních stránkách, a to se zachováním mého autorského práva k odevzdánému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 20. dubna 2019.

Jakub Minka

## **Abstrakt/anotace**

Cílem bakalářské práce je představit moderní alternativu v oblasti tvorby webu pomocí technologie JAM Stack. Webové technologie se vyvíjí velice rychlým tempem a v současné době je na trhu množství webových technologií, které lze použít k tvorbě webu. Každá má své plusy a minusy a právě technologie JAM Stack se nachází někde na rozhraní. Staví totiž na rozdíl od klasického frontend + backend na čistě statickém konceptu bez použití serveru a databází. Umožňuje tak tvořit bezpečné a rychlé weby. JAM Stack je název pro techniku tvorby webu, která sestává ze tří elementů. JavaScriptu, který zpracovává a vykonává požadavky klienta, množství API (rozhraní pro programování aplikací), které zprostředkovávají server-side a databázové operace a značkovacího jazyka (Markup). Autor se ve své práci zaměří na různé formy JAM Stacku, vyzkouší a popíše některé použitelné API a nástroje pro zápis jazyka, prozkoumá možnosti využití této technologie a porovná jí s ostatními technologiemi. V praktické části vytvoří funkční webovou prezentaci, která bude celá tvořená čistě technologií JAM Stack. K publikaci webu na internetu využije některý z hostingů statických webových stránek a web bude distribuován prostřednictvím CDN.

## **Klíčová slova**

JAM Stack, frontend, tvorba webu, JavaScript, API, Markup, CDN, Git, Statické generátory

## **Abstract**

The aim of the thesis is to introduce a modern alternative in the field of web development using JAM Stack technology. Web technologies are developing at a very fast pace and there are currently a lot of web technologies available on the market to develop websites. Each has its own pluses and minuses and the JAM Stack is located somewhere between. In contrast to the classical frontend + backend, JAM Stack is built on a purely static concept with no need to use servers and databases. It allows web creators to build secure and fast websites. JAM Stack is a method, which consists of three elements – JavaScript that handles and performs client requests, number of API's (application programming interfaces) that mediate server-side and database operations and Markup language. The author will focus on various forms of JAM Stack, test and describe some useful API's and markup language tools, explore the possibilities of using this technology and compare it with the other technologies. In the practical part he will create a functional website entirely built with JAM Stack technology. The website will be published using some of the static hosting providers and the website will be distributed over CDN.

## **Keywords**

JAM Stack, frontend, web development, JavaScript, API, Markup, CDN, Git, Static site generators

## **Poděkování**

Rád bych poděkoval panu PaedDr. Petru Pexovi, Ph.D. především za odborné vedení a věcné rady, které mi pomohly k úspěšnému dokončení mé práce. Dále bych rád poděkoval své rodině a přítelkyni Janě, kteří mě po celou dobu studia podporovali.

# **Obsah**

<b>1</b>	<b>Úvod</b>	<b>11</b>
1.1	Cíle . . . . .	11
1.2	Metody . . . . .	12
1.3	Východiska . . . . .	12
<b>2</b>	<b>Způsoby generování webů</b>	<b>13</b>
2.1	Statické webové stránky . . . . .	13
2.2	Dynamické webové stránky . . . . .	14
2.3	Staticky generované stránky . . . . .	16
<b>3</b>	<b>Srovnání současných metod vývoje webu</b>	<b>17</b>
3.1	LAMP Stack . . . . .	17
3.2	MEAN Stack . . . . .	17
3.3	.NET . . . . .	18
<b>4</b>	<b>JAM Stack</b>	<b>19</b>
4.1	Výhody JAM Stacku . . . . .	21
4.2	Slabé stránky JAM Stacku . . . . .	22
4.3	Využití JAM Stacku . . . . .	23
4.4	JavaScript . . . . .	23
4.5	API . . . . .	24
4.5.1	Vytváření HTTP požadavků . . . . .	25
4.5.2	Seznam vybraných API: . . . . .	26
4.6	Markup . . . . .	28
4.7	Technologie využívané v JAM Stacku . . . . .	28

4.7.1	CDN . . . . .	28
4.7.2	Git . . . . .	29
4.8	Workflow práce v JAM Stacku . . . . .	29
4.8.1	Více rolí v jednom vývojovém workflow . . . . .	33
4.8.2	Koncept CI/CD . . . . .	34
<b>5</b>	<b>Statické generátory stránek</b>	<b>35</b>
5.1	Jekyll . . . . .	35
5.1.1	Instalace a vytvoření první stránky . . . . .	36
5.1.2	Základní struktura webu . . . . .	37
5.1.3	Tvorba webu v Jekyllu . . . . .	38
5.2	Hugo . . . . .	40
5.2.1	Instalace a spuštění první stránky . . . . .	40
5.2.2	Základní struktura webu . . . . .	41
5.3	GatsbyJS . . . . .	42
5.3.1	Instalace a vytvoření první stránky . . . . .	42
5.3.2	Základní struktura webu . . . . .	43
<b>6</b>	<b>Hosting statických stránek</b>	<b>45</b>
6.1	Netlify . . . . .	45
6.2	GitHub Pages . . . . .	46
6.3	GitLab Pages . . . . .	47
6.4	Bitbucket pages . . . . .	48
<b>7</b>	<b>Headless CMS</b>	<b>49</b>
<b>8</b>	<b>Praktická část</b>	<b>51</b>

8.1	Struktura webu . . . . .	51
8.2	API . . . . .	53
8.2.1	PouchDB . . . . .	53
8.2.2	Google Maps API . . . . .	54
8.2.3	Netlify Forms API . . . . .	54
8.3	Blog . . . . .	56
8.4	Vyhledávání . . . . .	58
8.5	Eshop . . . . .	58
<b>9</b>	<b>Závěr</b>	<b>60</b>
<b>Seznam použité literatury a zdrojů</b>		<b>62</b>
<b>Seznam obrázků</b>		<b>66</b>
<b>Seznam tabulek</b>		<b>67</b>
<b>Seznam příkladů</b>		<b>68</b>
<b>A</b>	<b>Příloha</b>	<b>70</b>
<b>B</b>	<b>Příloha</b>	<b>71</b>
<b>C</b>	<b>Příloha</b>	<b>72</b>

# 1 Úvod

Moderní doba jde kupředu velice rychlým tempem a s ní i dnešní technologie. V oblasti webu se za posledních několik let objevilo množství nových technologií, které od počátku internetu značně ovlivnily jeho vývoj. Jednou z těchto technologií je i JAM Stack - poměrně obecné pojmenování pro moderní způsob tvorby webu. JAM Stack se neváže na konkrétní technologie - výběr programovacího jazyka a nástrojů, kterými lze tvořit a publikovat web, je zcela na vývojáři.

## 1.1 Cíle

Cílem bakalářské práce je zpracovat aktuální problematiku vývoje webu s využitím technologie JAM Stack. Jedná se o způsob tvorby webu pomocí klientského JavaScriptu, API a Markupu. Díky absenci backendových operací a komunikace s databázovými a aplikačními servery jsou weby tvořené pomocí této technologie rychlé a zcela bezpečné. Veškerý kód od JavaScriptu až po Markup je uložen ve vzdáleném repozitáři Git, následně je doručován pomocí CDN a proto je načítání stránek velice rychlé. Mezi jeho další nesporné výhody patří využití verzovacích systémů, které JAM Stacku přináší spolehlivost a škálovatelnost, a také hostingu statických stránek, který je ve spoustě případů zdarma. V teoretické části bude popsána technologie JAM Stack včetně workflow práce, budou porovnány různé nástroje vývoje, prozkoumáno použití API, nejpoužívanějších statických generátorů, hostinových služeb a tzv. headless CMS. Součástí teoretické části bude i srovnání s dalšími metodami vývoje webu, výhody a nevýhody použití JAM Stacku

a způsoby využití v praxi. V praktické části bude tato technologie aplikována na příkladovém webu.

## **1.2 Metody**

V teoretické části se hned v úvodu zaměřím na problematiku statického generování webu, srovnání s ostatními metodami a popíši technologii JAM Stack, včetně různých nástrojů, které se v této oblasti používají. Následně dle využití v praxi popíši workflow práce a zhodnotím JAM Stack z hlediska rychlosti, bezpečnosti a škálovatelnosti. V další části prozkoumám jednotlivé nástroje pro statické generování stránek a vyzkouším některé API.

V praktické části vytvořím webové stránky čistě pomocí technologie JAM Stack.

## **1.3 Východiska**

Od počátků tvorby webu pokročila doba velice rychlým tempem. Současná doba je hektická, internetem koluje stále větší množství dat, zvyšuje se množství uživatelů a počet webů a tím i nároky na vývoj. Moderní uživatel se už nespokojí se statickým webem. Požaduje rychlé načítání, přehlednost a funkčnost, a to na kterékoliv platformě. V současnosti se také stává stále diskutovanější téma bezpečnosti internetu. I těmto trendům se musel vývoj webu přizpůsobit. Vznikla tak technologie JAM Stack, jejíž cílem je co nejvíce vytěsnit problémy při využití současných technik tvorby webu. Hlavními přínosy JAM Stacku jsou rychlosť, snadný vývoj a škálovatelnost, bezpečnost, menší nároky na vývojový tým a tím i nižší náklady na vývoj.

## 2 Způsoby generování webů

Počátky tvorby webových stránek sahají do konce 90. let, kdy vznikla dodnes stále funkční webová prezentace<sup>1</sup> Evropské organizace pro jaderný výzkum (CERN). Tato stránka a další dokumenty CERN byly přístupné v rámci této organizace pomocí URL a propojené mezi sebou hypertextovými odkazy. Šlo tak o první statickou stránku v historii. S požadavky na větší weby, častější aktualizace a rozsáhlejší funkčnost webových stránek přišly následně i první dynamické stránky, jejichž idea ovšem vznikala už na počátcích internetu.

### 2.1 Statické webové stránky

Statické stránky zobrazují stejný obsah všem uživatelům nezávisle na kontextu. Jsou to většinou HTML dokumenty formátované CSS styly s trochu JavaScriptu uložené v souborovém systému dostupné webovým serverem přes HTTP (klient/prohlížeč zašle požadavek a webový server dle tohoto požadavku vrátí odpověď ve formě statické stránky). Vhodné jsou tedy spíš pro obsah, který je neměnný a nepotřebuje být aktualizován nebo je potřebná pouze jeho občasná aktualizace. Statické stránky se tedy ukázaly vhodnější pro jednoduché weby s malým množstvím obsahu.[7]

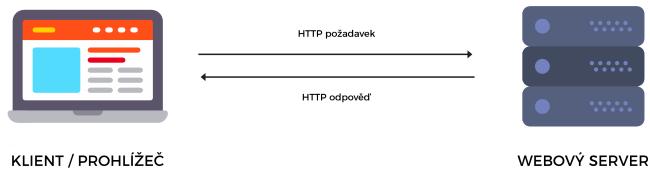
Výhoda spočívá především v jednoduchosti úprav, bezpečnosti, rychlosti načítání koncovým uživatelům, finančních nákladech a možnosti zobrazení přímo webovým prohlížečem bez potřeby webového nebo aplikacního serveru.[7]

Nevýhodou je především nemožnost automatizace stránek a tím pádem

---

<sup>1</sup>Webová stránka je přístupná na adresě <http://info.cern.ch>

nutnost změn přímo v kódu pro každou aktualizaci obsahu a jeho opětovné publikování. V případě větších webů tak nastává problém s ruční aktualizací velkého množství stránek a následnou nekonzistencí dat v případě nahrávání obsahu na server přes FTP. Dynamická interakce s uživatelem v případě statických webů zcela chybí.[7]



Obrázek 1: Schéma statického webu - vlastní tvorba

### **2.2 Dynamické webové stránky**

Dynamické webové stránky řeší hlavní problémy statického konceptu. Obsah každé webové stránky je v tomto případě automaticky generován s aktuálními informacemi pro každé individuální zobrazení – mění se v závislosti na čase, uživateli, uživatelské interakci, kontextu nebo jakékoli kombinaci předchozího. Při každé změně obsahu tak probíhá komunikace s aplikačním nebo webovým serverem a databázovým serverem, který požadavek klienta zpracuje a jako výsledek vrátí webovou stránku v HTML. Skriptování a vytváření obsahu probíhá buď na straně klienta (JavaScript nebo ActionScript) nebo na straně serveru (využívá se program běžící na serveru, který produkuje dy-

namické webové stránky pomocí jazyků používajících CGI<sup>2</sup>) a v neposlední řadě spočívá v kombinaci předchozích způsobů (klient a server) - AJAX<sup>3</sup>.[6]

Dynamický způsob generování stránek tak umožňuje tvorbu větších webů, snadnou správu pro editory bez nutnosti zasahování do kódu, „živější“ interakci s uživatelem a v případě již hotových webů minimální nebo žádné finanční nároky na správu. Výhodou může být i skrytí zdrojového kódu aplikací koncovému uživateli, což přináší zvýšení bezpečnosti, ale i přesto je právě bezpečnost v současné době redakčních systémů ohrožena. Použitelnost dynamického generování stránek nalezneme u blogů, různých publikačních systémů, e-shopů, atd.[9]

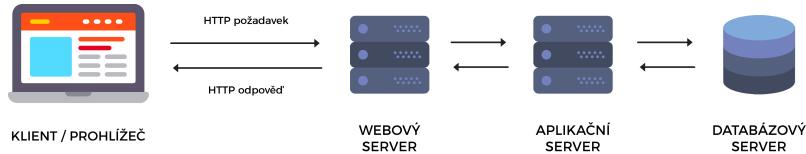
Mezi hlavní nevýhody patří bezpečnost. Podle konzervativního odhadu je dnes zranitelných více než 70% instalací redakčního systému WordPress (dle statistik používá tento redakční systém více než 23% webových stránek na internetu) a ohrožení se týká také 12 milionů uživatelů redakčního systému Drupal.[10] Mezi další nevýhody pak patří mimo jiné rychlost načítání (pomalý TTFB<sup>4</sup>), vysoké pořizovací náklady, náklady na škálování a jakékoli další programování webu a cachování.[6]

---

<sup>2</sup>Common Gateway Interface - protokol pro propojení externích aplikací s webovým serverem.

<sup>3</sup>Asynchronous JavaScript and XML - označení technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich kompletního znovačítání za pomoci asynchronního zpracování webových stránek pomocí knihovny napsané v JavaScriptu

<sup>4</sup>Time to first byte - jednotka měření odezvy webových serverů nebo webů



Obrázek 2: Schéma dynamického webu - vlastní tvorba

### 2.3 Staticky generované stránky

O počátcích staticky generovaných stránek se mluví v souvislosti s příchodem desktopových aplikací Dreamweaver a FrontPage, které nabídly řešení pro budování webových stránek, řízené přes WYSIWYG editory, kde stránky mohly být rozdeleny do částí, jako je navigace, záhlaví a zápatí atd. V některých ohledech to byly původní statické webové generátory: stavební webové stránky ze šablon, partials, knihovny médií a někdy dokonce i SQL databáze a jejich zveřejňování přes FTP jako statické soubory. Dnešní statické generátory jsou programy, které pomocí šablonovacího systému skládají zdrojové soubory do formy statických webových stránek. Zdrojovými soubory mohou být dokumenty HTML, textové soubory v Markdownu, CSS styly, skripty, konfigurační soubory, média a další.[10]

## 3 Srovnání současných metod vývoje webu

### 3.1 LAMP Stack

Webová architektura LAMP Stack vznikla ze čtyř open-source komponent, které vývojáři na počátku devadesátých let využili k vytváření webových stránek: operačního systému Linux, Apache HTTP Serveru, databáze MySQL a jazyka PHP (JavaScript). Způsob, jakým fungují webové servery postavené na LAMP, spočívá v tom, že při každé žádosti uživatele o stránku nejprve server dotazuje databázi a následně kombinuje výsledek s daty markupu stránky a pluginů a vygeneruje v prohlížeči dokument v HTML. Výhody použití LAMP Stacku spočívají ve vysoké bezpečnosti, open-source licenci a rozsáhlé komunitě vývojářů. Nevýhodou se může jevit jeho křivka učení. K získání všech dovedností pro správnou funkčnost, výkon a bezpečnost webu je třeba se v problematice podrobně orientovat.[4]

### 3.2 MEAN Stack

MEAN Stack je označení pro 4 open-source komponenty, které dohromady tvoří metodu pro vytváření dynamických JavaScriptových aplikací. Skládá se z MongoDB databáze, Express.js (aplikační backend framework), Angular.js (aplikační frontend framework) a běhového prostředí jazyka JavaScript Node.js. Tato metoda je známá především v oblasti SPA (Single Page Applications). Je určena primárně k vývoji aplikací v JavaScriptu a ten se může jevit jako ta nejzásadnější výhoda, protože všechny komponenty běží

### ***3 SROVNÁNÍ SOUČASNÝCH METOD VÝVOJE WEBU***

---

ve stejném programovacím jazyce. Základem je architektura MVC<sup>5</sup>.[4]

#### **3.3 .NET**

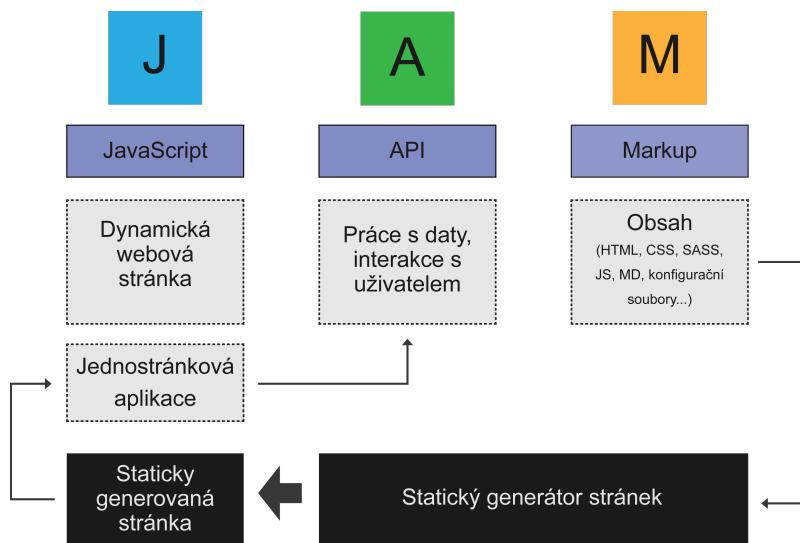
Za označením .NET stojí firma Microsoft, která tuto metodu vyvinula na počátku dvacátých let. Jedná se o termín, který zastřešuje několik vývojových nástrojů a technologií. Zahrnuje v sobě aplikace pro vývoj podnikových, mobilních a web aplikací - APS.NET, WCF, WebAPI, Microsoft SQL Server a Windows hosting pod webovým serverem MS IIS. Po nedávném představení .NET Core je nyní možné k hostování .NET webových aplikací používat Linux i macOS s různými databázovými a webovými servery. .NET používá programovací jazyky C# a JavaScript. Výhodou je například možnost použití v kombinaci s JAM Stackem - MS SQL server jako databáze, MS IIS nebo ASP.NET WebAPI jako server a Angular, React nebo Vue.js ve frontendu. Nevýhodou může být nutná znalost objektově orientovaného programování.[4]

---

<sup>5</sup>Softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní.

## 4 JAM Stack

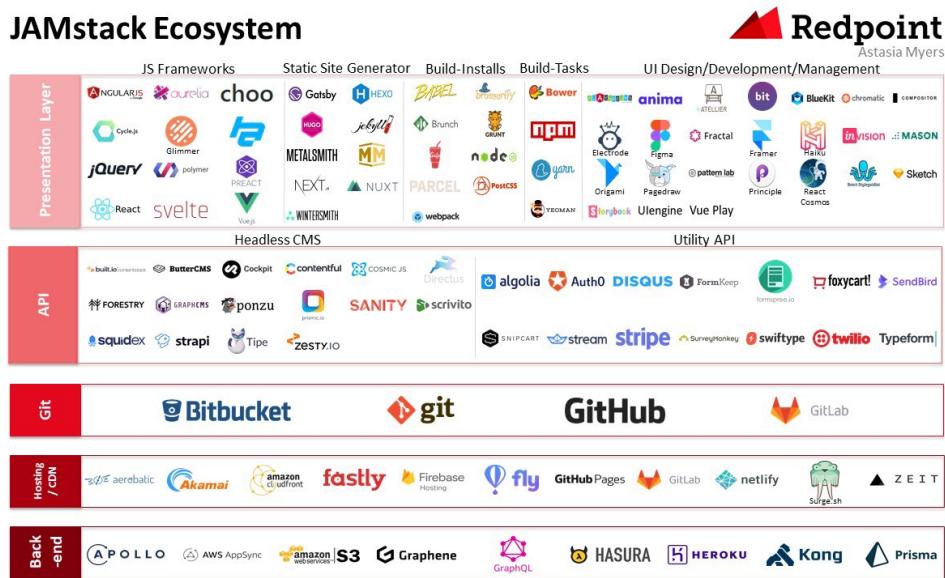
JAM Stack je pojmenování moderní technologie pro vývoj webu a webových aplikací založené na klientském JavaScriptu, API a Markupu. Weby vytvořené touto technologií využívají výhod statických i dynamických stránek a odstraňují některé jejich nedostatky. V souvislosti s touto technologií tak již nemluvíme o operačních systémech, webových serverech, programovacích backend jazycích či databázích. Jedná se o moderní způsob vývoje webu a aplikací, který přináší vyšší výkon, lepší zabezpečení, menší finanční nároky a škálovatelnost, a také kvalitnější optimalizaci SEO.[1]



Obrázek 3: Schéma JAM Stacku - vlastní tvorba

Více než pojmenování jedné konkrétní technologie se jedná spíš o souhrnný název pro širokou škálu nástrojů a jazyků, které společně pod určitými pravidly vytvářejí statické weby. K tomu může také využít kombinace s některou z výše zmíněných metod (MEAN, LAMP nebo .NET). Jedná se tedy

způsob tvorby statických stránek, které dokážou nabídnout funkcionalitu běžných dynamických stránek s využitím modelů SaaS<sup>6</sup> nebo FaaS<sup>7</sup>. Tyto stránky tak běžně pracují se šablonami, interagují s uživatelem, zpracovávají data a formuláře a v reálném čase reagují na požadavky uživatele.[1]



Obrázek 4: Široká paleta technologií a nástrojů JAM Stacku<sup>8</sup>

<sup>6</sup>Software as a Service (v překladu software jako služba) je způsob poskytování licencí softwaru formou služby - prostřednictvím webového rozhraní.

<sup>7</sup>Function as a Service (v překladu funkce jako služba) - je kategorie Cloudových služeb, která umožňuje fungování webů bez přítomnosti serverů - tzv. serverless, umožňuje využívat pouze funkce softwaru běžícího v Cloudu.

<sup>8</sup>Zdroj: [www.medium.com](http://www.medium.com)

## 4.1 Výhody JAM Stacku

- **Široký výběr jazyků a nástrojů** - mezi největší výhody JAM Stacku patří obrovský výběr jazyků a nástrojů, takže si každý vývojář může vybrat dle svých vlastních preferencí
- **Vyšší výkon** – weby budované pomocí JAM Stacku jsou velice rychlé díky statickému generování obsahu pomocí statických generátorů, které prohlížeči skrze CDN<sup>1</sup> předkládají již hotové, čistě vygenerované HTML (mluví se až o 6x rychlejším TTBF ve srovnání i s vysoce optimalizovanými dynamickými weby)[5]
- **Snazší cachování** – všechny URL vrací stejnou HTML všem uživatelům a všechny změny se projevují ihned - ná rozdíl od dynamických CMS, v nichž mohou URL vracet po nějakou dobu již neaktuální obsah[5]
- **SEO** - JAM Stack dokonale zjednodušuje URL stránek a umožňuje lepší organizaci, filtrování a stránkování webu[5]
- **Atomické nasazení** - projekty v JAM Stacku se mohou rozrůstat vysokou rychlostí. Každá změna znamená nasazení množství souborů, což může způsobit nekonzistentní stav. JAM Stack proto využívá verzovací systém, který umožňuje provádět "atomické nasazení", kde se žádné změny neprojeví, dokud nebudou zkонтrolované nebo nebudou nahrány všechny upravené soubory[5]
- **Vyšší bezpečnost** – díky absenci databází, pluginů a aplikací běžících na straně serveru zde nehrozí riziko útoků nebo např. SQL injekce

---

<sup>1</sup>Content Delivery Network - síť pro doručování obsahu

- všechny dynamické funkce jsou vykonávány robustními API a klientským JavaScriptem[5]
- **Lepší developer experience** – veškeré komponenty, konfigurační soubory, kaskádové styly, skripty a další obsah jsou uloženy na jednom centrálním místě - ve vzdáleném repozitáři Git[5]
- **Škálovatelnost** – webové stránky mohou být kdykoliv verzovány jakýmkoliv uživatelem a vyvíjeny jako samostatné větve a to jak v rámci jednotlivců, tak i celým vývojovým týmem díky systému Git[5]
- **Nižší náklady** – díky statickému obsahu jsou veškeré hostingové služby v rámci JAM Stacku ZDARMA a také není třeba backend programátorů a údržby[5]

## 4.2 Slabé stránky JAM Stacku

Stejně jako ostatní technologie - i JAM Stack má své slabé stránky. Ty se mohou objevit například při vývoji velkých nebo složitých webů (generování obrovského množství stránek), u projektů s mnoha interaktivními funkcemi nebo např. u stránek vyžadujících časté aktualizace. U takových webů je lepší sáhnout po jiných technologiích. Jednou z hlavních nevýhod je i dlouhý proces učení a orientace v použití jednotlivých nástrojů a jazyků. Pole nástrojů JAM Stacku je opravdu široké a pro začátečníka v této oblasti může být snazší výběr tradičních uživatelsky přívětivých CMS. V neposlední řadě se jako nevýhoda může jevit právě dynamická stránka této technologie. Existuje již celá řada API, které spolehlivě nahradí funkce tradičních server-side webů. Některé dynamické funkce jsou ale stále nenahraditelné (JAM Stack zatím

nenabízí ani zdaleka tolik použitelných API jako např. WordPress obsahuje pluginů).[5]

Tradiční CMS	JAM Stack
Rychlejší vývoj	Pomalejší vývoj
Pomalejší načítání	Rychlejší načítání
Dražší, méně flexibilní hosting	Levnější, flexibilnější hosting
Využití u větších webů	Použití na menší stránky

Tabulka 1: Srovnání tradičních CMS vs. JAM Stack - vlastní tvorba

### 4.3 Využití JAM Stacku

Využití JAM Stacku najdeme především u blogů, menších eshopů, osobních stránek, webů společností a organizací. JAM Stack nenahrazuje současné CMS systémy, ale je jakousi jejich alternativou. Každopádně má velký potenciál stát se jednou z převažujících technologií na poli vývoje webu.[1]

### 4.4 JavaScript

O veškerou dynamickou funkčnost webu (požadavky ze strany klienta a vrácení odpovědí) se stará JavaScript a to kompletně na straně klienta - ať už ve formě frontendového frameworku, knihovny (Vue.js, Angular, React) nebo tzv. vanilla JavaScriptu. Využívá se především ke komunikaci s API, ale také pro běžné skriptování na stránce.[8]

```
1 <script
2   src="https://maps.googleapis.com/maps/api/js?key=
3     AIzaSyCTjhUHTwh5cEeZziPWxcD2NRt3BEfZPQk&callback=initMap"
4   async
5   defer
5 ></script>
```

Příklad 1: Ukázka integrace API Google Maps JavaScriptem

## 4.5 API

API (Application Programming Interface) neboli česky aplikační programové rozhraní je sada rutin, protokolů a nástrojů pro vytváření softwarových aplikací. Lze jej definovat také jako soubor metod pro komunikaci mezi různými softwarovými komponentami. Jinými slovy API umožňuje softwaru komunikovat s jiným softwarem. V zásadě rozhraní API určuje, jak by měly softwarové komponenty interagovat. API se navíc používají při programování komponent grafického uživatelského rozhraní (GUI). Dobrý API usnadňuje vytváření programu tím, že poskytuje všechny stavební bloky. Programátor pak sestaví bloky dohromady. V případě JAM Stacku se API využívají jako nahrazení za backend - umí nahradit eshop, komentáře, obsluhu formulářů a jiné. Právě API stojí za bezpečností JAM Stacku, která je dána samostatným vývojem (čím kvalitněji napsaná API, tím menší riziko ohrožení bezpečnosti).[18]

Webové rozhraní API umožňuje interakci serveru se softwarem třetí strany. Webový server používá HTTP požadavky pro komunikaci s veřejně dostupnou.

stupným koncovým adresářem URL, který obsahuje JSON data<sup>9</sup>. Ke komunikaci se v případě webových API používá koncept CRUD (Create, Read, Update, Delete).[18]

Některé API již umožňují snadnou integraci v kódu bez nutnosti složitého skriptování. Např. aplikace Netlify Forms API poskytuje již zabudovanou funkcionality pro přístup k datům formulářů prostřednictvím vlastních atribut, které se volají v tagu `<form>`.

```
1 <form name="odeslat" action="/dekujeme.html" netlify>
```

Příklad 2: Volání Netlify Forms API zabudovanými atributy

Akce	HTTP metoda	Definice
Create	POST	Vytvoří nový zdroj
Read	GET	Načte zdroj
Update	PUT/PATCH	Aktualizuje existující zdroj
Delete	DELETE	Odstraní zdroj

Tabulka 2: Tabulka HTTP požadavků a akcí [19]

#### 4.5.1 Vytváření HTTP požadavků

V příkladu 1 je ukázána jednoduchá komunikace s API Google Maps, přičemž tento script načítá z URL skript, kterému JavaScript přes HTTP předává API klíč a volá funkci pro inicializaci mapy.

Následující příklad ukazuje vytvoření HTTP požadavků. Nejprve je do proměnné `request` přiřazen objekt `XMLHttpRequest`, který umožňuje

<sup>9</sup>Způsob zápisu dat nezávislý na platformě

zasílat požadavky serveru. Následně je otevřeno nové připojení a požadavkem GET je přiřazen požadavek URL koncového adresáře. Po vytvoření připojení je možné pracovat s daty JSON. Nakonec je nutné požadavek odeslat.[19]

```
1 var request = new XMLHttpRequest();
2 request.open('GET', url, true);
3 request.onload = function() {};
4 request.send();
```

Příklad 3: Práce s API - vytvoření GET požadavku

V dalším kroku server odešle požadovaná data (ve formátu JSON) a poté je možné s daty pracovat. Na začátku je nutné převést obdržená data do pole objektů JavaScriptu. To provedeme příkazem `JSON.parse()`. Následně je možné vypsat do konzole všechny objekty (v tomto případě produkty) z pole data. Příkazy je nutné obalit do podmínky, která zabrání neočekávaným problémům, které mohou na zadané adrese nastat.[19]

```
1 var data = JSON.parse(this.response);
2 if (request.status >= 200 && request.status < 400) {
3   data.forEach(object => {
4     console.log(product.title);
5   });
6 } else { console.log('error'); }
```

Příklad 4: Práce s obdrženými daty API

#### 4.5.2 Seznam vybraných API:

- **Disqus** - globální systém komentářů, který zlepšuje diskusi na webových stránkách a spojuje konverzace napříč webem [15]

- **FormKeep** - jedná se o jednoduchou clouдовou databázi, která umožňuje zachytávat, ukládat a sdílet data z webových formulářů[12]
- **Snipcart** - API, která umožňuje vytvářet e-shopy a obsluhovat jejich funkce[12]
- **Algolia** - Algolia poskytuje snadný způsob integrace AJAX vyhledávání do webových stránek[12]
- **Google Maps API** - rozhraní API služby Mapy Google patří do skupiny API od společnosti Google, které umožňuje vývojářům vkládat Mapy Google do webových stránek pomocí rozhraní JavaScript nebo Flash. Aplikace Google Maps API je navržena tak, aby fungovala jak na mobilních zařízeních, tak v desktop prohlížečích.[26]
- **Words API** - Words API je API pro anglický jazyk, které má širokou škálu funkcí, jako je například extrahování definic slov nebo synonym, načítání příkladů slov nebo frází, získávání rýmů slov a mnoho dalšího.[26]
- **YouTube API** - Youtube API společnosti Google umožňují vývojářům integrovat videa a funkce YouTube na webové stránky nebo do aplikací. Rozhraní API služby YouTube zahrnují API služby YouTube Analytics API, YouTube Data API, YouTube Live Streaming API, YouTube Player APIs a další.[26]
- **Facebook Comments API** - API od společnosti Facebook umožňuje přidávat komentáře použitím účtu Facebooku[26]

- **Webtask a Serverless** - tzv. serverless služby, které dle současných trendů FaaS umožňují backendovou funkcionalitu[26]

## 4.6 Markup

Pod pojmem Markup neboli značkovací jazyk se skrývá veškerý textový obsah webu - klasické HTML nebo CSS, ale je možné použít i další značkovací jazyky. Nejpoužívanějšími jsou Markdown, AsciiDoc nebo Textile.[17]

Markup tvoří prezentační vrstvu webu v závislosti na použitém způsobu generování stránek (statické generátory vs. buildovací nástroje) a šablonovacím systému (Liquid v případě generátoru Jekyll, ReactJS v případě Gatsby). [17]

## 4.7 Technologie využívané v JAM Stacku

### 4.7.1 CDN

**CDN (Content Delivery Network)** neboli síť pro doručování obsahu je síť vzájemně propojených počítačů skrze Internet, které zajišťují rychlé doručení obsahu. CDN umožňuje rychlý přenos dat potřebných pro načítání obsahu včetně stránek HTML, souborů JavaScriptu, stylů CSS, obrázků a videí. Popularita služeb CDN neustále roste a dnes je již většina webového obsahu zprostředkována prostřednictvím CDN, včetně obsahu webů Facebooku, Netflixu či Amazonu.[27]

Na rozdíl od klasických webových hostingů CDN nehostuje obsah. Nemůže tak nahradit výhody běžného hostingu, ale pouze pomáhá cachovat obsah na okraji sítě, což zlepšuje výkon webových stránek. Využíváním ukládání

do mezipaměti k omezení hostování šírky pásma pomáhá předcházet přerušením provozu a zvyšuje bezpečnost.[27]

#### 4.7.2 Git

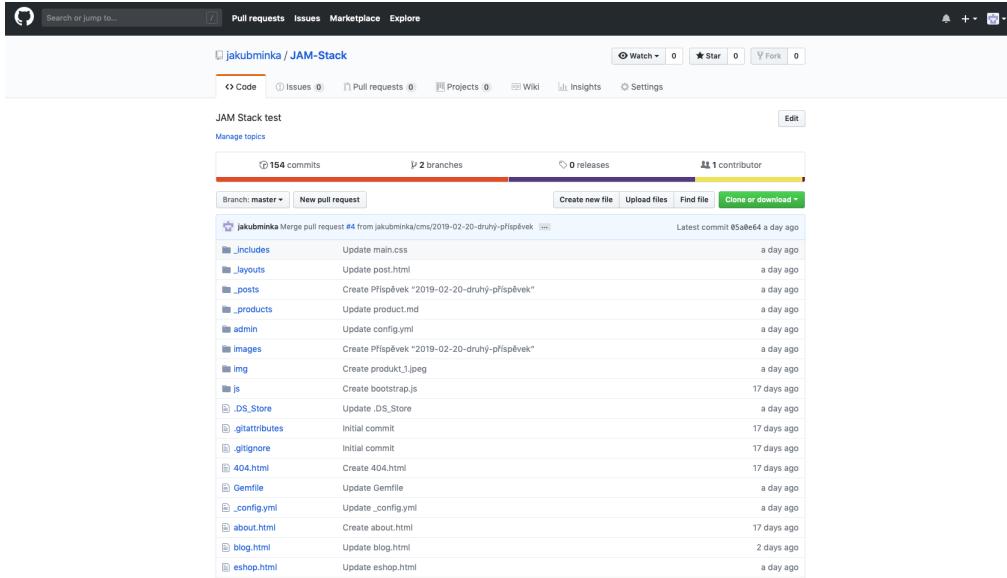
**Git** je v dnešní době nejpoužívanější systém pro řízení verzí. Jedná se o open-source projekt vyvinutý v roce 2005 Linusem Torvaldem. Git umožňuje snadnou správu všech verzí webových stránek, větvení vývoje, nelineární vývoj, škálování a další pro snadný vývoj webových stránek i velkými vývojovými týmy.[28]

V případě JAM Stacku nalezneme jeho využití především u vzdálených repozitářů **GitHub**, **Bitbucket** a **GitLab**. Tyto repozitáře umožňují snadné sdílení kódu i mezi ostatními uživateli, kteří jej mohou dále klonovat a vyvíjet jej jako samostatnou větev či verzovat.[28]

### 4.8 Workflow práce v JAM Stacku

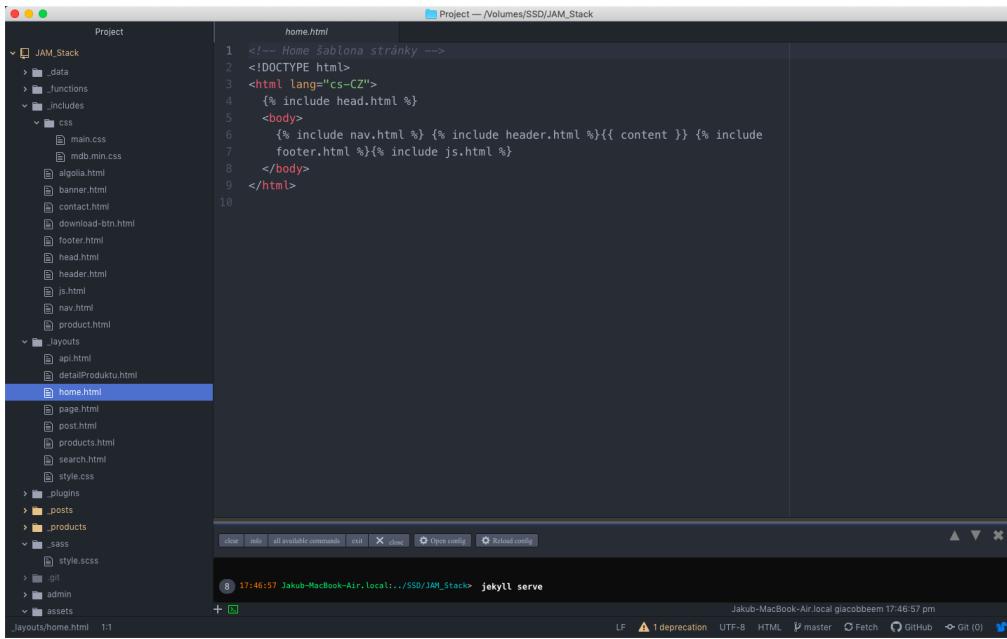
1. Proces tvorby webu v JAM Stacku závisí na vybraných nástrojích a začíná právě jejich výběrem a také výběrem programovacích jazyků. Základem je výběr statického generátoru nebo buildovacího nástroje, jenž ovlivní celou další tvorbu, ale také textového editoru, v němž budeme celý web psát. Vytvoření základní struktury webu probíhá nejlépe za pomoci příkazové řádky (Terminálu), kdy se jednoduchými příkazy (dle vybraného statického generátoru) automaticky vytvoří všechny potřebné soubory a adresáře, včetně závislostí, pluginů, konfiguračních souborů, atp. Následné kódování je již čistě na vývojáři. Vybírat může

## 4 JAM STACK



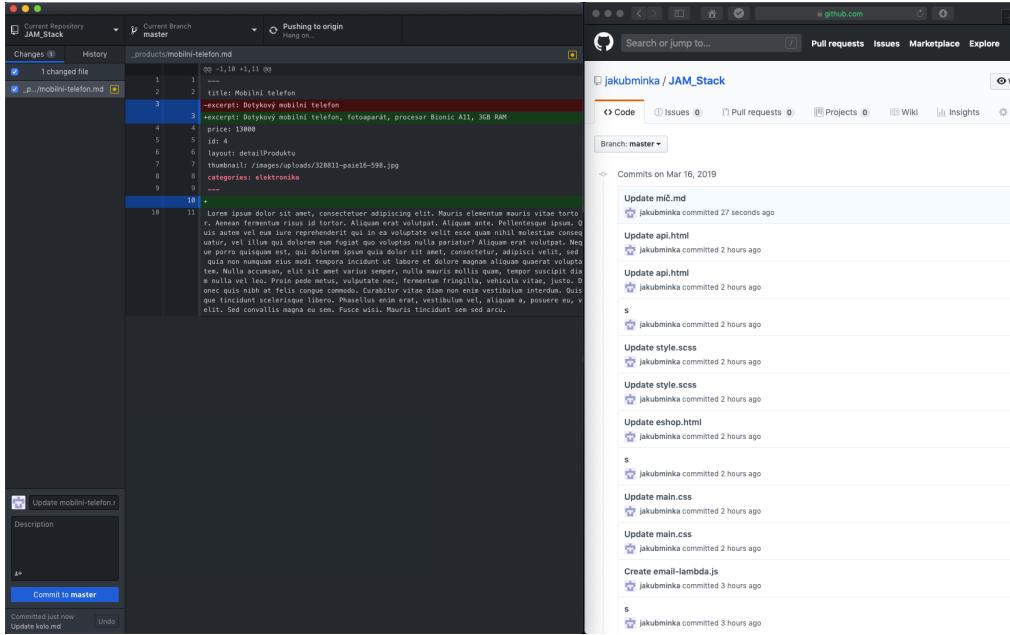
Obrázek 5: Ukázka repozitáře GitHub

z nepřeberného množství frameworků a frontendových jazyků. Vývoj může probíhat jak offline v desktopu, tak rovnou na serveru. V tomto kroku probíhá i připojení požadovaných API a skriptování.[9]



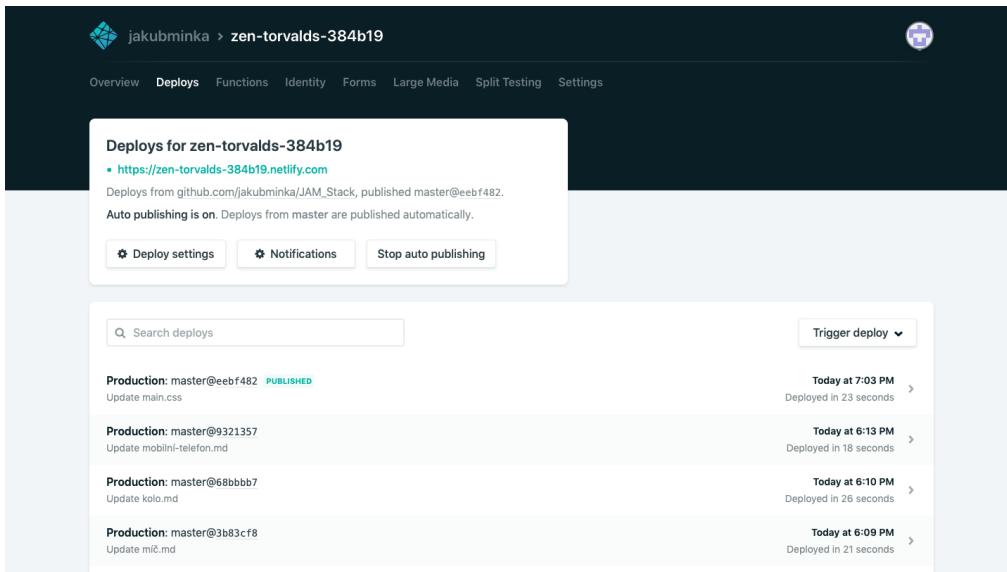
Obrázek 6: Ukázka lokálního vývoje v textovém editoru Atom s využitím Terminálu

2. Ve druhém kroku je v případě běhu na serveru vhodný výběr služby vzdáleného repozitáře, kde budou uložené všechny pracovní soubory webu a kam bude vybraný statický generátor automaticky generovat statickou verzi webu. Inicializace vzdáleného repozitáře probíhá opět pomocí příkazů v Terminálu nebo pomocí desktopových aplikací, které nabízejí grafické rozhraní a jednoduché ovládání. Tyto aplikace automaticky sledují změny ve struktuře místního adresáře a porovnávají jej s aktuální verzí (větví) ve vzdáleném repozitáři. Jedním kliknutím pak lze vytvořit novou vývojovou větev, slučovat větve, komentovat provedené změny a také provádět v případě použití gitu běžné operace push, pull a fetch.[9]



Obrázek 7: Publikování změn do vzdáleného repozitáře GitHub prostřednictvím desktopové aplikace GitHub Desktop

3. Dalším krokem je výběr vhodného hostingu, nastavení cesty ke vzdálenému repozitáři (dle výběru v předchozím kroku), určení spouštěcích a buildovacích příkazů (dle vybraného statického generátoru) a další úvodní nastavení. Hostingové služby jako je Netlify nebo GitHub Pages automaticky kontrolují změny ve vzdáleném repozitáři a při každé změně ve zvolené vývojové větvi automaticky spustí aktuální (funkční) verzi webu.[9]



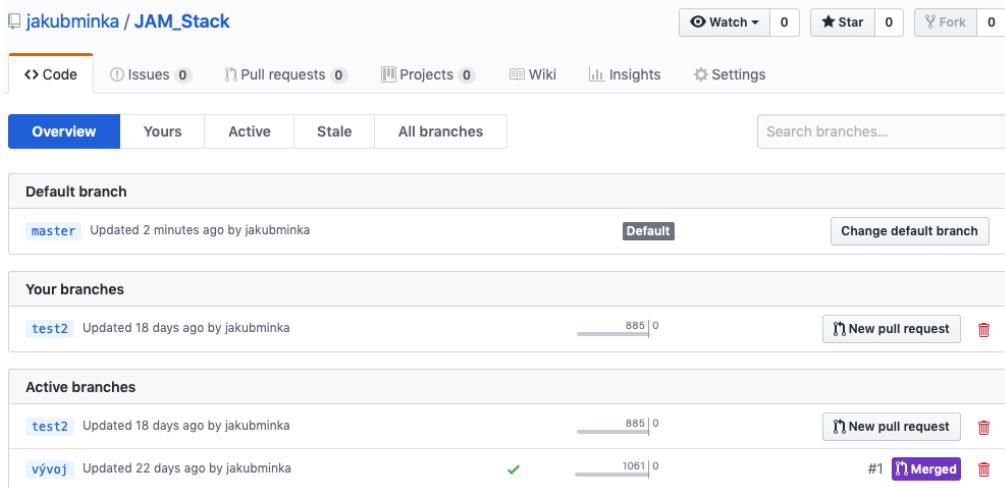
Obrázek 8: Hlavní nabídka správy webu v Netlify

4. Volitelným krokem je pak napojení headless CMS pro dynamickou funkčnost webu.[9]

#### 4.8.1 Více rolí v jednom vývojovém workflow

Technologie JAM Stack nabízí možnost paralelního vývoje webových stránek. Díky CDN, verzovacímu systému a možnosti oddělení vývojových větví může v tu samou chvíli více vývojářů (nebo i více týmů) pracovat na frontendu a vytvářet tak novou strukturu webu, editor přispívat do blogu a zároveň může např. návštěvník objednávat z eshopu.

To vše v reálném čase, bez čekání na nahrání na server, atp.[23]



Obrázek 9: Správa větví ve vzdáleném repozitáři GitHub

#### 4.8.2 Koncept CI/CD

Kouzlo JAM Stacku tkví v automatizaci procesů. Využívá se zde koncept CI/CD neboli Continous Integration and Continuous Deployment (v českém překladu průběžná integrace a nasazení). Jedná se o vývojový koncept, který cílí na rychlý a bezpečný vývoj. CI/CD znamená umístění veškerého obsahu webu na jednom centrálním místě a tím pádem snazší kódování a sledování změn v souborech. I sebemenší změna může být okamžitě publikována, kontrolována a testována (usnadněna práce ve větších vývojových týmech). Vývojář se tak může soustředit pouze na psaní kódu - všechny procesy kontroly a spuštění kódu jsou zajišťovány službami vzdáleného repozitáře a hostingu. Velkou výhodou tohoto konceptu je právě kontrola aktuálnosti a stavu souborů.[11]

## 5 Statické generátory stránek

Jak už bylo výše řečeno, statické generátory stránek se zaměřují na generování kompletně statických HTML stránek. Tyto generátory vytváří statické stránky při každé změně nebo aktualizaci obsahu a ty jsou ihned k dispozici uživateli. Nespornou výhodou statických generátorů je rychlosť a bezpečnost. Přehled těch nejpoužívanějších je k dispozici na stránce [www.staticgen.com](http://www.staticgen.com). Těmi nejznámějšími jsou Jekyll, Hugo a Gatsby.[20]

### 5.1 Jekyll

Jekyll je zřejmě nejpopulárnější statický webový generátor, jehož základem je dynamický open source programovací jazyk Ruby. Jeho zakladatelem byl v roce 2008 Tom Preston-Werner - americký softwarový vývojář, který stojí také za webovou službou GitHub podporující vývoj softwaru za pomoci verzovacího nástroje Git. Jekyll je distribuován pod licencí MIT a veškerá jeho dokumentace je přístupná na webové adrese [www.jekyllrb.com](http://www.jekyllrb.com).[2]

Namísto použití databází Jekyll převezme obsah (HTML, CSS), šablony v Markdown, Textille nebo Liquid a vytvoří kompletní statickou stránku připravenou k publikaci Apache HTTP serverem, Nginxem nebo jiným webovým serverem.[2]

Využití tohoto generátoru nalezneme především u osobních stránek, stránek projektů a organizací. Velmi užitečné je jeho propojení s hostingovou službou statických webových stránek GitHub Pages (dostupné na adrese [www.pages.github.com](http://www.pages.github.com)), která přináší možnost hostování stránek s minimálními náklady a také možnost použítí v kombinaci se zabudovaným pre-

procesorem CSS Sass, frontendovými frameworky Bootstrap, Semantic UI a dalšími. Jekyll může být dále propojen s cloudovými CMS systémy jako jsou CloudCannon, Forestry, Netlify nebo Siteleaf, které umožní editaci obsahu i běžným uživatelům bez znalosti programování, takže se funkcionalitou přiblíží běžným CMS systémům jako jsou WordPress, Drupal a další.[2]

### 5.1.1 Instalace a vytvoření první stránky

Pro začátek je nutná instalace vývojového prostředí Ruby. V operačním systému Windows stačí stáhnout a nainstalovat instalátor Ruby. Na UNIX-like systémech je nejjednodušší použítí správců balíčků.[2]

Instalaci Ruby provedeme pomocí Homebrew následujícím příkazem:[2]

```
1 $ brew install ruby
```

Příklad 5: Instalace jazyka Ruby

Dále je potřeba instalace frameworku RubyGems pro správu balíčků Ruby a také sadu překladačů GNU Compiler Collection (zkráceně GCC) a program GNU Make pro automatizaci překladu. Odkazy je možné najít v dokumentaci Jekyllu.[2]

V dalším kroku nainstalujeme samotný Jekyll:

```
1 $ gem install jekyll bundler
```

Příklad 6: Instalace statického generátoru Jekyll

Následně vytvoříme novou stránku (ve výchozím nastavení se vytvoří v domovském adresáři aktuálního uživatele), přesuneme se do adresáře stránky a necháme Jekyll naši první stránku spustit na lokálním serveru (defaultně se spustí na adrese <http://localhost:4000>):[2]

```
1 $ jekyll new stranka  
2 $ cd stranka  
3 $ bundle exec jekyll serve
```

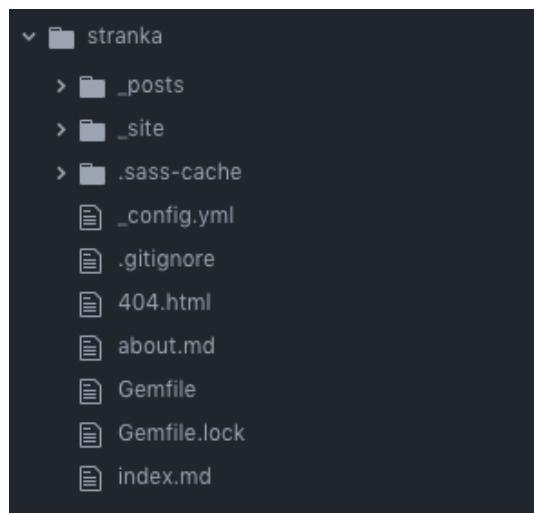
Příklad 7: Inicializace nového adresáře v Jekyllu a spuštění na lokálním serveru

### 5.1.2 Základní struktura webu

Po instalaci dostaneme základní adresář stránky. Struktura webu v Jekyllu je znázorněna na obrázku 10. V kořenovém souboru najdeme konfigurační soubory (základní konfigurační soubor **\_config.yml**), organizovaný obsah stránky ve složkách a HTML, Markdown nebo textile stránky (přípony .html, .markdown, .md nebo .textile). Označení a význam jednotlivých složek je následující:[2]

- **/\_layouts** - obsahuje šablony
- **/\_includes** - tato složka obsahuje znovupoužitelné části stránky, které je možné vkládat liquid tagy do dalších stránek
- **/\_posts** - obsahuje publikované příspěvky v přísně definovaném formátu (RRRR-MM-DD-nazev.MARKUP) pro řazení příspěvků dle data
- **/\_drafts** - označuje dosud nepublikované příspěvky (bez data)
- **/\_data** - zde jsou utříděny data stránky v podporovaných formátech .ymml, .yaml, .jsou, .csv, .tsv
- **/\_sass** - obsahuje soubory preprocesoru sass

- **/\_site** - do této složky Jekyll generuje výsledné statické soubory stránky a to přehledně do předem dané struktury
- **/config.yml** - hlavní konfigurační soubor Jekyllu



Obrázek 10: Adresářová struktura stránky v Jekyll

Vývojový server, který zajistí automatické sledování změn ve struktuře lokálního repozitáře a jejich okamžité nasazení je možné spustit příkazem:

```
1 $ jekyll serve
```

Příklad 8: Spuštění lokálního vývojového serveru v Jekyllu

### 5.1.3 Tvorba webu v Jekyllu

**Liquid** Je šablonovací jazyk, který umožňuje jednoduše vytvářet znovu-použitelné komponenty webu - zápisem přímo v HTML. Hlavními komponentami Liquid jsou objekty, tagy a filtry.[2]

- **Objekty** - objekty v Liquidu slouží k dynamickému vkládání obsahu na určené místo přímo v HTML stránce nebo například k označení konkrétního prvku na stránce a všech jeho atributů.[2]

```
1  {{ product.title }}  
2  {{ content }}
```

Příklad 9: Výběr titulku produktu a výpis obsahu stránky

- **Tagy** - vytvářejí logiku a řízení toku šablon - je možné používat běžné operace v programování jako jsou cykly, podmínky, atp.[2]
- **Filtry** - jednoduché metody, které modifikují označený výstup - např. kód `{{ product.title | uppercase }}` styluje titulek produktu napsaný velkými písmeny[2]

### Výhody:

- Propojení s GitHub Pages umožňuje snadný hosting webových stránek
- Jekyll je snadný k naučení i díky obsáhlé a jasně vysvětlené dokumentaci
- Kolem Jekyllu se sdružuje velká komunita vývojářů, kteří posunují vývoj dopředu
- Standartní funkcionality je možné rozšířit množstvím dostupných pluginů

### Nevýhody:

- Jednou z hlavních nevýhod se může jevit právě jazyk Ruby, protože je před použitím generátoru Jekyll nejprve třeba instalace vývojového prostředí Ruby
- Ruby nepatří v roce 2019 mezi nejpopulárnější programovací jazyky pro vývoj webových aplikací a to může mít vliv na aktuálnost Jekyllu
- Pomalejší generování webu proti ostatním generátorům

## 5.2 Hugo

Dalším neméně populárním statickým generátorem je Hugo, který je psaný v programovacím jazyce Go. Byl vyvinut norským vývojářem Bjørnem Erikkem Pedersenem a jeho týmem. Hugo je open-source projekt vydaný pod licencí Apache License 2.0.[16]

### 5.2.1 Instalace a spuštění první stránky

Instalace je opět jednoduchá a je možná hned několika způsoby, v závislosti na používaném operačním systému.

Celý proces instalace a vytvoření adresáře tvorí následující příkazy:[16]

```
1 $ brew install hugo  
2 $ hugo new site stranka
```

Příklad 10: Instalace Huga a vytvoření nové stránky

Hugo pro začátek nabízí možnost instalace široké nabídky témat. Nejprve je nutné přejít do adresáře webu, následně vytvořit repozitář Git a poté přidat

vybrané téma:[16]

```
1 $ git init  
2 $ git clone tema
```

Příklad 11: Výběr tématu Hugo

Nyní stačí spustit server Hugo, který poběží v localhostu na defaultním portu 1313 příkazem:

```
1 $ hugo server
```

Příklad 12: Spuštění lokálního serveru Hugo

### 5.2.2 Základní struktura webu

Hugo stejně jako Jekyll nabízí organizovanou strukturu souborů a složek:

- **/archetypes** - složka, která obsahuje šablony obsahu s přednastavenými informacemi front matter
- **/assets** - zde jsou uloženy všechny soubory pro následné zpracování Hugo Pipes
- **/data** - obsahuje konfigurační soubory
- **/content** - v této složce je obsažen veškerý obsah webu a v podsložkách jsou pak uloženy např. jednotlivé sekce webu, atp.
- **/layouts** - obsahuje šablony ve formátu .html
- **/static** - tato složka obsahuje veškerý statický obsah, který Hugo vygeneruje (obrázky, CSS, JavaScript, atd.)
- **/config.toml** - hlavní konfigurační soubor Hugo

**Výhody:**

- Velice rychlý vývojový proces (téměř žádná počáteční konfigurace)
- Rozsáhlá nabídka volně dostupných šablon, přehledná dokumentace
- Šablonovací systém napsaný také v jazyce Go

**Neýhody:**

- Chybí pluginy

### 5.3 GatsbyJS

Gatsby je rychlý statický generátor webových stránek, který k vytváření statického obsahu využívá JavaScriptové knihovny React.js a pro přístup k datům moderní dotazovací jazyk GraphQL. Je vydáván pod licencí MIT.[16]

#### 5.3.1 Instalace a vytvoření první stránky

Pro začátek je nutná instalace JavaScriptového prostředí Node.js, Gitu a nástroje příkazové řádky Gatsby CLI. Vytvoření adresáře webu je stejně jednoduché jako u předchozích generátorů. Nejprve je nutná instalace Gatsby CLI, následuje vytvoření stránky a spuštění vývojového serveru:[16]

```
1 $ npm install -g gatsby-cli  
2 $ gatsby new stranka  
3 $ gatsby develop
```

Příklad 13: Instalace, vytvoření stránky a spuštění vývojového serveru Gatsby

Kompletní dokumentace je k dispozici na adrese: <https://www.gatsbyjs.org/docs/>

### 5.3.2 Základní struktura webu

- **/.cache** - složka cache obsahuje automaticky generovaný interní cache GatsbyJS
- **/public** - tato složka obsahuje finálně vygenerované soubory webu
- **/plugins** - složka plugins obsahuje pluginy specifické pro daný projekt
- **/src** - veškterý zdrojový kód webu je uložen ve složce src - šablony, komponenty a další
- **/src/pages** - komponenty v této složce se po generování automaticky stanou stránkami s cestami dle názvu souboru
- **/src/templates** - složka templates obsahuje šablony pro automatické generování stránek
- **/src/components** - obsahuje komponenty pro následnou snazší automatizaci
- **/static** - soubory v této složce nebudou zpracovány a rovnou se publikují do složky public
- **gatsby-config.js** - hlavní konfigurační soubor

**Výhody:**

- Základ v Reactu
- Data jsou spravována pomocí GraphQL
- Dostupné rozšiřující pluginy
- Produkujе velice rychlé statické HTML stránky

**Neýhody:**

- Pro začátek nutná znalost JavaScriptu, Reactu i GraphQL

## 6 Hosting statických stránek

Hosting webu vytvořeného technologií JAM Stack je velice snadný, rychlý a především zdarma. Díky staticky generovaným stránkám není potřeba databázových a aplikačních serverů (a tím pádem drahých hostingových služeb) a vygenerované statické stránky je možné kdykoliv jednoduše přemístit na jiný hosting. Většina hostingů statických webových stránek navíc podporuje širokou škálu statických generátorů a podporují koncept CI/CD.[12]

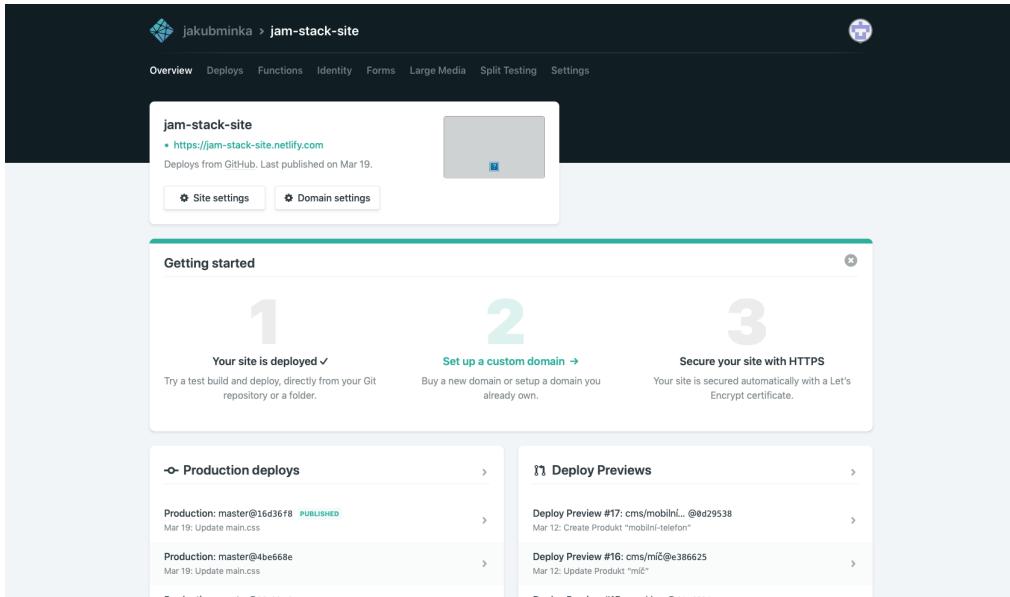
### 6.1 Netlify

Hostingová služba Netlify nabízí All-in-one řešení pro hosting statických webových stránek. Nabízí vlastní CLI<sup>10</sup> a rychlou CDN, protokol zabezpečení HTTPS na jedno kliknutí, hosting zdarma pro neomezený počet stránek (pouze veřejné repozitáře), nastavení vlastního názvu domény nebo připojení na vlastní doménu a také již připravené API služby pro obsluhu formulářů a obsluhu Webhooků<sup>11</sup>. Netlify navíc umožňuje automatické sledování změn v repozitáři, atomické nasazení webu a jeho verzování (možnost nastavení verzí ve formě subdomén) přímo v repozitáři.[14]

---

<sup>10</sup>Command Line Interface - příkazový řádek

<sup>11</sup>Uživatelsky definované HTTP callbacky, které umožňují provést vývojářem definovanou akci (kliknutí na tlačítko odešle mail, atp.)



Obrázek 11: Ukázka základní obrazovky hostingu Netlify

## 6.2 GitHub Pages

Hosting statických stránek, který umožňuje publikovat stránky přímo v nastavení repozitáře na GitHubu. Hosting je i v tomto případě zdarma (v základním plánu je omezený počet domén a jsou povolené pouze veřejné repozitáře). GitHub Pages navíc nabízí desktopovou aplikaci GitHub Desktop, která umožňuje snadnou kontrolu změn v lokálním repozitáři a nejdůležitější operace vzdále-ného repozitáře, včetně publikování změn, větvení, spojování, atp. GitHub Pages umožňují nastavení vlastního názvu domény i podporu HTTPS.[28]

### GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

The screenshot shows the GitHub Pages settings interface. At the top, a green banner indicates that the site is published at <https://jakubminka.github.io/jamstack/>. Below this, the 'Source' section shows the site is built from the master branch. A dropdown menu is set to 'master branch'. The 'Theme Chooser' section shows the site uses the Minima theme, with a 'Change theme' button. The 'Custom domain' section allows users to enter a custom domain and save changes. Finally, the 'Enforce HTTPS' section has a checked checkbox and a note explaining it's required for the default domain.

✓ Your site is published at <https://jakubminka.github.io/jamstack/>

**Source**  
Your GitHub Pages site is currently being built from the `master` branch. [Learn more](#).

`master branch` ▾

**Theme Chooser**  
Select a theme to publish your site with a Jekyll theme. [Learn more](#).  
Your site is currently using the `Minima` theme.

[Change theme](#)

**Custom domain**  
Custom domains allow you to serve your site from a domain other than `jakubminka.github.io`. [Learn more](#).

Save

**Enforce HTTPS**  
— Required for your site because you are using the default domain (`jakubminka.github.io`)

HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site.  
When HTTPS is enforced, your site will only be served over HTTPS. [Learn more](#).

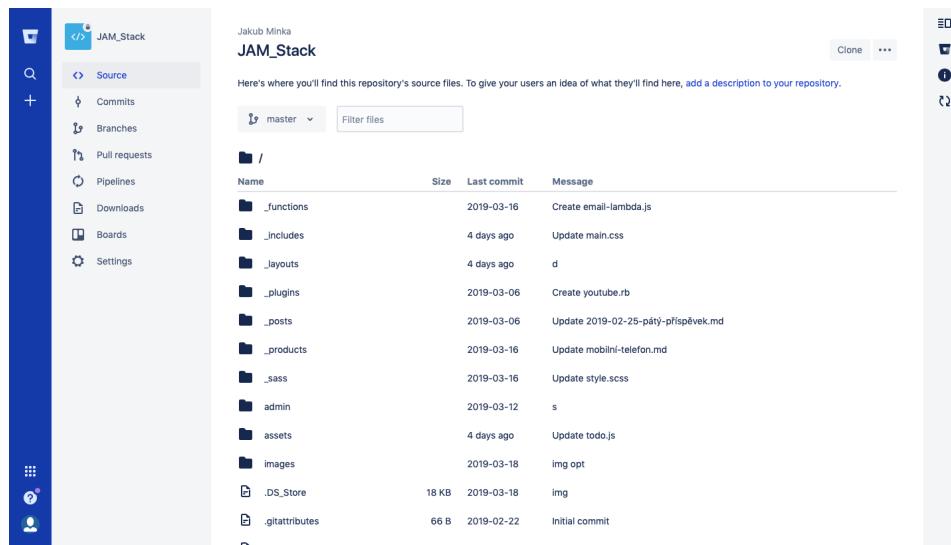
Obrázek 12: Ukázka nastavení GitHub Pages přímo v možnostech repozitáře

### 6.3 GitLab Pages

GitLab Pages fungují na podobném principu jako předchozí hostingová služba a opět tak nabízí hosting zdarma (v základním plánu). Integrace je v tomto případě spojená se službami repozitáře GitLab. Velkou výhodou oproti předchozím hostingovým službám je možnost hostingu neomezeného počtu soukromých repozitářů zdarma.[14]

## 6.4 Bitbucket pages

Bitbucket pages je poskytovatelem hostingových služeb, který je založen na verzovacím systému Bitbucket. V základní verzi nabízí hosting neomezeného počtu soukromých repozitářů zdarma, vlastního desktopového klienta i podporu CI/CD. Hostování stránek je zajištěno shodností názvu repozitáře s uživatelským jménem.[14]



Obrázek 13: Ukázka prostředí Bitbucket

## 7 Headless CMS

Tradiční CMS (Content Management System) jsou systémy, které se běžně používají k vytváření, spravování a ukládání obsahu webu spolu se všemi daty v backendu a zároveň jsou těsně spojené s frontendem. Této tradiční architektury, při které backend a databáze jsou spojené se stejným systémem, který zobrazuje obsah koncovým zařízením a uživatelům (frontend) využívají tradiční blogovací platformy jako např. WordPress, Squarespace nebo Wix.

**Headless CMS** jsou systémy pro správu obsahu, které při použití statického generování webu nahrazují klasické CMS. Rozdílem je pouze absence frontendové vrstvy, přitom stále poskytuje administrátorské prostředí, databázi a API. Tyto CMS umožňují pouhé vytváření a spravování obsahu, který následně čeká na zavolání API a následně může být distribuován dle požadavků jakýmkoliv způsobem. Seznam nejpoužívanějších headless CMS je volně dostupný na adrese <https://headlesscms.org>.[21]

### Rozdělení headless CMS

**Git-based CMS** Git-based CMS jsou spravovací systémy úzce spjaté s nástrojem Git. Stejně jako je veškerý vývoj verzován Gitem, i headless CMS založené na tomto systému jej plně využívají. Nevýhodou může být v tomto případě škálování obsahu na různých platformách. Výhodou je několik open source Git-based CMS.[21]

**API-driven CMS** Zde se veškerý obsah doručuje prostřednictvím rozhraní API. Není tak možné sledovat změny obsahu v Gitu, ale je možné jed-

noduše škálovat pomocí flexibilních rozhraní API. Nevýhodou je vyšší cena pořízení.[21]

Název	Typ	Licence	Poznámky
Netlify CMS	Git-based	MIT	Podpora všech statických generátorů
Contentful	API-driven	MIT	Cloud based CMS
Jekyll Admin	Git-based	Open-source	Plugin Jekyllu
Cosmic JS	API-driven	MIT	Multiplatformní CMS
Strapi	API-driven	Open-source	Založeno na node.js
Forestry	Git-based	Open-source	WYSIWYG editor

Tabulka 3: Srovnání headless CMS[21]

## **8 Praktická část**

V rámci praktické části bakalářské práce jsem vytvořil web čistě pomocí technologie JAM Stack. Ke kódování webu jsem využil editor Atom, který nabízí širokou škálu dostupných pluginů (Prettier pro automatické stylování nebo např. integrovaný Terminal), které usnadňují zápis kódu. Jako statický generátor jsem použil Jekyll ve verzi 3.8.5, šablony jsou vytvářeny pomocí šablonovacího jazyka Liquid. Stylování webu je řešeno převážně frameworkm Bootstrap a designem Material Design. K následnému verzování jsem použil desktopovou verzi GitHub Desktop, která automaticky kontroluje změny provedené v dokumentu a jednoduchým kliknutím lze následně publikovat změny a nahrát je do vzdáleného repozitáře, kterým je v tomto případě GitHub. K hostingu využívám bezplatných hostingových služeb Netlify, které nabízejí už v základu podporu zabezpečení HTTPS. Obsah je možné spravovat pomocí headless CMS Netlify CMS. Stránka je k dispozici na adrese [www.jam-stack-site.netlify.com](http://www.jam-stack-site.netlify.com).

### **8.1 Struktura webu**

Jedná se o jednoduchý několikastránkový web s blogem a eshopem, na kterém je ukázána funkčnost JAM Stacku a také popis této technologie. Jednotlivé stránky jsou generovány automaticky pomocí statického generátoru Jekyll, včetně odkazů v navigaci, blogových příspěvků nebo např. souboru `sitemap.xml`.

Vzhled jednotlivých stránek webu je daný šablonami ve složce `_layouts`. Tyto šablony obsahují jak kód Liquid, tak i HTML, Markdown a vložené

inline styly. Kostra webu staví na frameworku Bootstrap, který je přímo podporovaný Jekylllem a který je doplněn designem Material Design Bootstrap<sup>12</sup>. Struktura webu je znázorněna na obrázku 16.

```

1 <!DOCTYPE html>
2 <html lang="cs-CZ">
3   {% include head.html %}
4   <body>
5     {% include nav.html %}
6     {% include header.html %}
7     {{ content }}
8     {% include footer.html %}
9     {% include js.html %}
10    </body>
11 </html>
```

Příklad 14: Zápis automatického vložení částí stránky home v systému Liquid

```

<div class="collapse navbar-collapse" id="collapseNow">
  <ul class="navbar-nav mr-auto">
    {% assign navigation_pages = site.html_pages | sort: 'navigation_weight' %}
    {% for p in navigation_pages %}
      {% if p.navigation_weight %}
        <li {% if p.url == page.url %}
          class="nav-item active"
        {% else %}
          class="nav-item"
        {% endif %}>
          <a href="{{ p.url }}" class="nav-link text-wrap">
            <span data-hover="{{p.title}}">{{ p.title }}</span>
          </a></li>
        <li class="hidden"><a href="#page-top"></a></li>
      {% endif %}
    {% endfor %}
```

Obrázek 14: Automatické generování odkazů v navigační liště

---

<sup>12</sup>Dostupné z webu <https://mdbootstrap.com>

## 8.2 API

Stránka API ukazuje přehled některých použitelných API včetně popisu použití.

### 8.2.1 PouchDB

Ukázka použití API PouchDB jakožto lokální databáze v prohlížeči formou jednoduché aplikace pro přidávání úkolů. PouchDB využívá asynchronní API, podporuje callbacky, promises i asynchronní funkce a většina funkcí je defaultně volána prostřednictvím zápisu:

```
1 db.doSomething( args... , [ options ] , [ callback ] )
```

Příklad 15: Defaultní volání funkcí v PouchDB

```
1 function(error, result) {}
```

Příklad 16: Využití callbacků v PouchDB

```
1 var db = new PouchDB('todos');
2 function addTodo(text) {
3     var todo = {
4         _id: new Date().toISOString(),
5         title: text,
6         completed: false
7     };
8     db.put(todo, function callback(err, result) {
9         if (!err) {
10             console.log('Successfully posted a todo!');
11         }
12     });
13 }
```

```
14 function newTodoKeyPressHandler(event) {
15     if (event.keyCode === ENTER_KEY) {
16         addTodo(newTodoDom.value);
17         newTodoDom.value = "";
18     }
19 }
```

Příklad 17: Přidávání úkolů do lokální databáze v prohlížeči prostřednictvím API PouchDB

### 8.2.2 Google Maps API

API Google Maps přináší jednoduchou integraci voláním přes HTTP - viz. příklad 1. Mapy je následně možné libovolně stylovat v přiloženém .json souboru. Voláním API je spuštěna funkce `initMap`, ve které je možné nastavit chování API.

```
1 var map;
2 function initMap() {
3     map = new google.maps.Map(document.getElementById("map") , {
4         center: { lat: 48.973911, lng: 14.47502 },
5         zoom: 8
6     });
7 }
```

Příklad 18: Nastavení API Google Maps

### 8.2.3 Netlify Forms API

Na kontaktním formuláři je ukázáno obsluhování formuláře pomocí zábudovaných funkcí Netlify API. Integrace API je možná díky předpřipravené

vlastnosti `data-netlify="true"`. Hosting Netlify pak při každém dalším buildu kontroluje data odeslaná formulářem prostřednictvím HTTP a vypisuje je do administrátorského prostředí pro správu formulářů. Netlify nabízí také snadnou integraci zabezpečení ReCaptcha prostřednictvím vlastnosti `data-netlify-recaptcha="true"`.

```
1 <form name="contact" method="POST" data-netlify="true" />
2   <input type="text" name="jmeno" placeholder="Jmeno:" />
3   <input type="email" name="email" placeholder="Email:>/>
4   <div data-netlify-recaptcha="true"></div>
5   <input type="submit" value="Odeslat zpravu" />
6 </form>
```

Příklad 19: Nastavení API Netlify Forms

Prostřednictvím API Zapier pak lze dále pracovat s obdrženými daty. Pro testování obdržených dat z formuláře jsem využil Webhooků, které ukládají data do tabulky Google Spreadsheets. Jak je vidět na příkladu 20, Netlify Forms API sbírá kompletní data o vyplnění formuláře včetně přesného času odeslání, jedinečného identifikátoru ID, IP adresy a dalších uživatelsky definovaných dat.

```
1 2019-03-22T22:09:35.021Z | contact | 5c955d1f647266fad0f34483 |
2   176.97.9.154 | Jmeno | Prijmeni | email@email.cz | 123456789
3   | Obsah zpravy .
```

Příklad 20: Data z formuláře obdržená prostřednictvím Webhooků Zapier

### 8.3 Blog

Stránka blogu ukazuje využití šablony k automatickému generování dynamického obsahu. Jednotlivé příspěvky, které jsou umístěny ve složce `_posts`, jsou automaticky řazeny do Bootstrapových komponentů card. Příspěvky do blogu ve formátu `.md` je možné přidávat přímo do adresáře `_posts`, a to s příslušným formátováním dle data přidání nebo prostřednictvím administrátorského prostředí Netlify CMS. Každý jednotlivý příspěvek je založen na šabloně `post` a má své nastavení ve Front Matteru<sup>13</sup>. Nastavení polí příspěvků v uživatelském rozhraní Netlify CMS se nachází v konfiguračním souboru `config.yml` ve složce `admin`. V detailech jednotlivých příspěvků je možné přidávat komentáře integrací API Disqus.

```

1   {%- for post in site.posts %}

2     <div class="card-body">

3       <h5>{{ post.categories }}</h5>

4       <a href="{{ post.url | prepend: site.baseurl }}>

5         <h4>{{ post.title }}</h4>

6       </a>

7       <p>{{ post.excerpt }}</p>

8       <a href="{{ post.url | prepend: site.baseurl }}>

9         <button>Zobrazit více</button>

10        </a>

11    {%- endfor %}

```

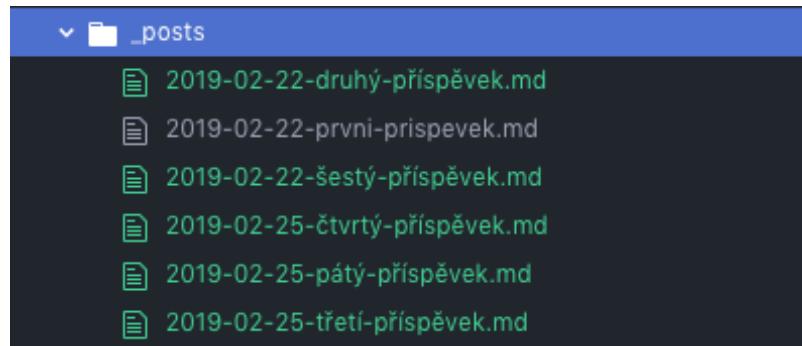
Příklad 21: Výpis příspěvků blogu

---

<sup>13</sup>Blok textu, který bude následně zpracovaný Jekylllem jako nastavení stránky/příspěvku

```
1  {%- if page.comments %}            
2      <div id="disqus_thread"></div>            
3      <script>            
4          (function() {            
5              var d = document,            
6                  s = d.createElement("script");            
7              s.src = "https://jam-stack.disqus.com/embed.js";            
8              s.setAttribute("data-timestamp", +new Date());            
9              (d.head || d.body).appendChild(s);            
10         });            
11     </script>            
12     <noscript>            
13         Please enable JavaScript to view the            
14         <a href="https://disqus.com/?ref_noscript">            
15             comments powered by Disqus.</a>            
16     </noscript>            
17 {%- endif %}
```

Příklad 22: Integrace API Disqus



Obrázek 15: Struktura složky s jednotlivými příspěvky

## 8.4 Vyhledávání

Na stránce vyhledávání je implementována Algolia API pro AJAX vyhledávání v příspěvcích blogu.

```

1 <div id="search-searchbar"></div>
2 <div id="search-hits">
3 {%- for post in site.posts %}
4     <div class="post-item">
5         <strong class="post-meta">{{ post.date }}</strong>
6         <h3>
7             <a href="{{ post.url }}">{{ post.title }}</a>
8         </h3>
9         <div class="post-snippet">{{ post.excerpt }}</div>
10    </div>
11 {%- endfor %}
12 </div>
13 {% include algolia.html %}
```

Příklad 23: Automatický výpis výsledků vyhledávání

Soubor `algolia.html` obsahuje veškerý kód aplikace. Ta využívá JavaScriptové knihovny `InstantSearch.js`, která umožňuje třívrstvé programní použití. v tomto případě je využito dvou vrstev - předdefinovaného widgetu a jeho rozšíření.

## 8.5 Eshop

Na stránce eshop je ukázána funkcionality staticky generovaného eshopu a integrace dynamických funkcí prostřednictvím API Snipcart. Stránka s produkty je opět generovaná pomocí šablony do Bootstrapových komponent

card. Na každém tlačítku pro přidání do košíku je pak volána Snipcart API.

```

1  {% for product in site.products %}
2 <div class="card">
3     <a href="{{ site.baseurl }}{{ product.url }}>
4         
6     </a>
7     <div class="card-body">
8         <a href="{{ site.baseurl }}{{ product.url }}>
9             <h5>{{ product.categories }}</h5>
10            </a>
11            <h5>{{ product.title }}</h5>
12            <p>{{ product.excerpt }}</p>
13            <h4>{{ product.price }}</h4>
14            <button
15                class="snipcart-add-item"
16                data-item-name="{{ product.title }}"
17                data-item-id="{{ product.id }}"
18                data-item-image="{{ product.thumbnail }}"
19                data-item-url="{{ product.url }}"
20                data-item-price="{{ product.price }}"
21                >
22                Koupit za {{ product.price }} Kc
23            </button>
24        </div>
25    </div>
26  {% endfor %}
```

Příklad 24: Výpis produktů v eshopu

## **9 Závěr**

Bakalářská práce se zabývala moderní technologií v oblasti vývoje webu s názvem JAM Stack, která je bezpochyby jednou z nejzajímavějších technologií, které se za poslední dobu v oblasti tvorby webu objevily. Vzhledem ke stále aktuálnější problematice bezpečnosti internetu se statické stránky jeví jako skvělá alternativa dynamickým systémům, které představují velké riziko ohrožení.

V teoretické části jsem prozkoumal současné způsoby generování webu a metody vývoje a porovnal je s technologií JAM Stack. Následně jsem rozbral celou technologii včetně jejích hlavních součástí (použití JavaScriptu, napojení a otestování některých API a Markup), technologií využívaných při vývoji v JAM Stacku a workflow práce. V dalších částech práce jsou podrobně popsány nejdůležitější nástroje této technologie, včetně statických generátorů stránek, hostingů statických stránek a tzv. headlessCMS pro správu obsahu webů v JAM Stacku.

V praktické části jsem vytvořil web čistě pomocí technologie JAM Stack a popsal jeho specifika. Web jsem tvořil v lokálním repozitáři prostřednictvím praktického vývojářského editoru Atom a příkazů v Terminálu. Jako statický generátor jsem si zvolil Jekyll včetně šablonovacího systému Liquid. Šablonu webu tvorí framework Bootstrap (vzhled jsem styloval pomocí knihovny Material Design Bootstrap). Celý web jsem pak pomocí desktopové aplikace GitHub Desktop přesunul do veřejného vzdáleného repozitáře GitHub na adresu [www.github.com/jakubminka/JAM\\_Stack](https://www.github.com/jakubminka/JAM_Stack) a publikoval online zdarma pomocí hostingu Netlify se zabezpečením HTTPS. Pro demonstraci dynamické funkčnosti jsem využil výhod statického generátoru, který generuje

jak příspěvky do blogu, tak i produkty do eshopu a použil některé API, jako např. Disqus pro komentování příspěvků blogu, Algolia API pro AJAX vyhledávání na stránce nebo Snipcart pro obsluhu eshopu. Web je dostupný na adrese <https://jam-stack-site.netlify.com> a jeho TTFB je uctitelných 27ms.

JAM Stack nabízí opravdu nepřeberné množství nástrojů a technologií, v nichž jsem se na počátku tvorby bakalářské práce ztrácel. Postupem času jsem se ovšem dostal této technologii pod povrch a neuvěřitelně mě pohltila. V JAM Stacku se podle mě skrývá obrovský potenciál.

## **Seznam použité literatury a zdrojů**

- [1] JAM Stack. *JAM Stack - JavaScript, APIs, and Markup* [online]. [cit. 2018-12-18]. Dostupné z WWW: <[www.jamstack.org/](http://www.jamstack.org/)>.
- [2] Jekyll. *Jekyll - Simple, blog-aware, static sites — Transform your plain text into static websites and blogs.* [online]. [cit. 2018-12-18]. Dostupné z WWW: <[www.jekyllrb.com](http://www.jekyllrb.com)>.
- [3] PRSKAVEC, Ladislav. *JAM Stack (Javascript + Api + Markup) - Zdrojak.cz* [online]. 2017-07-04 [cit. 2018-12-18]. Dostupné z WWW: <[www.zdrojak.cz/clanky/jam-stack-javascript-api-markup/](http://www.zdrojak.cz/clanky/jam-stack-javascript-api-markup/)>.
- [4] CMS Wire. *JAMstack vs. LAMP Stack vs. MEAN vs .NET: Tech Stacks Compared* [online]. 2019-01-18 [cit. 2019-02-26]. Dostupné z WWW: <[www.cmswire.com/digital-experience/jamstack-vs-lamp-stack-vs-mean-vs-net-tech-stacks-compared/](http://www.cmswire.com/digital-experience/jamstack-vs-lamp-stack-vs-mean-vs-net-tech-stacks-compared/)>.
- [5] The New Stack. *The Sweetness of JAMstack: JavaScript, APIs and Markup* [online]. 2017-12-26 [cit. 2019-01-11]. Dostupné z WWW: <[www.thenewstack.io/the-sweetness-of-jamstack-javascript-apis-and-markup/](http://www.thenewstack.io/the-sweetness-of-jamstack-javascript-apis-and-markup/)>.
- [6] Wikipedia. *Dynamic web page* [online]. [cit. 2019-02-26]. Dostupné z WWW: <[www.en.wikipedia.org/wiki/Dynamic\\_web\\_page](http://www.en.wikipedia.org/wiki/Dynamic_web_page)>.
- [7] Wikipedia. *Static web page* [online]. [cit. 2019-02-26]. Dostupné z WWW: <[www.en.wikipedia.org/wiki/Static\\_web\\_page](http://www.en.wikipedia.org/wiki/Static_web_page)>.

## SEZNAM POUŽITÉ LITERATURY A ZDROJŮ

- [8] Concepta. *What You Should Know About JAMstack (REVIEW)* [online]. 2017-09-13 [cit. 2019-02-26]. Dostupné z WWW: <[www.conceptainc.com/blog/what-you-should-know-about-jamstack/](http://www.conceptainc.com/blog/what-you-should-know-about-jamstack/)>.
- [9] POLYAKOV, Vasiliy. K\$C Blog. *JAM Stack is the new face of static sites* [online]. 2018-08-14 [cit. 2019-02-26] Dostupné z WWW: <[www.kruscheccompany.com/blog/post/jamstack-is-the-new-face-of-static-sites](http://www.kruscheccompany.com/blog/post/jamstack-is-the-new-face-of-static-sites)>.
- [10] Rychlí kodéři. *Statické generátory webových stránek* [online]. [cit. 2019-02-26] Dostupné z WWW: <[www.rychliskoderi.cz/staticke-generatory-webovych-stranek](http://www.rychliskoderi.cz/staticke-generatory-webovych-stranek)>.
- [11] MYERS, Astasia. *The JAMstack: It's Pretty Sweet* [online]. 2018-09-05 [cit. 2019-01-15]. Dostupné z WWW <[www.medium.com/memory-leak/the-jamstack-its-pretty-sweet-e0834e4e6bb7](http://www.medium.com/memory-leak/the-jamstack-its-pretty-sweet-e0834e4e6bb7)>.
- [12] DIONNE, Mathieu. Snipcart.com. *The JAMstack in 2019: Why (and How) to Get Started - Snipcart* [online]. 2019-01-16 [cit. 2019-02-26]. Dostupné z WWW: <[www.snipcart.com/blog/jamstack](http://www.snipcart.com/blog/jamstack)>.
- [13] Netlify CMS. *Netlify CMS: Open-Source Content Management System* [online]. [cit. 2019-02-26]. Dostupné z WWW: <[www.netlifycms.org](http://www.netlifycms.org)>.
- [14] Netlify. *Netlify: All-in-one platform for automating modern web projects* [online]. [cit. 2019-02-26]. Dostupné z WWW: <[www.netlify.com](http://www.netlify.com)>.
- [15] Disqus. *Disqus - The #1 way to build your audience* [online]. [cit. 2019-02-26]. Dostupné z WWW <[www.disqus.com](http://www.disqus.com)>.

## *SEZNAM POUŽITÉ LITERATURY A ZDROJŮ*

---

- [16] Medium. *Top Static Site Generators For 2019* [online]. [cit. 2019-02-26]. Dostupné z WWW: <[www.medium.com/codingthesmartway-com-blog/top-static-site-generators-for-2019-26a4c8afcc05](http://www.medium.com/codingthesmartway-com-blog/top-static-site-generators-for-2019-26a4c8afcc05)>.
- [17] KYRNIN, Jennifer. Lifewire *What are Markup Languages?* [online]. 2018-03-03 [cit. 2019-02-26]. Dostupné z WWW <[www.lifewire.com/what-are-markup-languages-3468655](http://www.lifewire.com/what-are-markup-languages-3468655)>.
- [18] RASCIA, Tania. *How to Connect to an API with JavaScript* [online]. 2017-12-07 [cit. 2019-02-26]. Dostupné z WWW <[www.taniarascia.com/how-to-connect-to-an-api-with-javascript/](http://www.taniarascia.com/how-to-connect-to-an-api-with-javascript/)>.
- [19] PouchDB. *About PouchDB* [online]. [cit. 2019-02-26]. Dostupné z WWW <[www.pouchdb.com/learn.html](http://www.pouchdb.com/learn.html)>.
- [20] StaticGen. *StaticGen — Top Open Source Static Site Generators* [online]. [cit. 2019-02-26]. Dostupné z WWW <[www.staticgen.com](http://www.staticgen.com)>.
- [21] headlessCMS. *headlessCMS — Top Content Management Systems for JAMstack sites* [online]. [cit. 2019-02-26]. Dostupné z WWW <[www.headlesscms.org](http://www.headlesscms.org)>.
- [22] HAWKSWORTH, Phil. Hawksworths.com. *Adding search to a JAM Stack site* [online]. 2018-10-18 [cit. 2019-02-26]. Dostupné z WWW <[www.hawksworths.com/blog/adding-search-to-a-jamstack-site/](http://www.hawksworths.com/blog/adding-search-to-a-jamstack-site/)>.
- [23] MANSOUR, Ahmed. *JAMstack - What, why and how - Ahmed Mansour* [online]. 2019-02-05 [cit. 2019-02-26]. Dostupné z WWW <[www.mansour.blog/jamstack-what-why-and-how/](http://www.mansour.blog/jamstack-what-why-and-how/)>.

## *SEZNAM POUŽITÉ LITERATURY A ZDROJŮ*

---

- [24] SHANKAR. Opensense Labs. *Drupal + JAMstack: A Paradigm Shift — Opensense Labs* [online]. 2018-11-19 [cit. 2019-02-26]. Dostupné z WWW <[www.opensenselabs.com/blog/articles/drupal-jamstack](http://www.opensenselabs.com/blog/articles/drupal-jamstack)>.
- [25] KOSTRZEWA, Denis. Bejamas.io. *JAMstack: The Cornerstone of Modern-day Web Development - Bejamas Blog* [online]. 2018-12-20 [cit. 2019-02-26]. Dostupné z WWW <[www.bejamas.io/blog/jamstack-modern-web-development](http://www.bejamas.io/blog/jamstack-modern-web-development)>.
- [26] Academind. *What is the JAM Stack?* In *Youtube* [online]. Zveřejněno 19.12.2018 [vid. 2019-01-15]. Dostupné z WWW <[www.youtube.com/watch?v=Y8PXMbr0Kqo](http://www.youtube.com/watch?v=Y8PXMbr0Kqo)>.
- [27] TIWARI, Abhishek. *Content Management Systems of the Future: Headless, JAMstack, ADN and Functions at the Edge* [online]. 2018-11-03 [cit. 2019-01-15]. Dostupné z WWW <<https://www.abhishek-tiwari.com/cms-of-the-future-headless-jamstack-adn-and-functions-at-the-edge>>.
- [28] VISCONTI, Amanda. *Building a static website with Jekyll and GitHub Pages* [online]. 2016-04-18 [cit. 2019-01-15]. Dostupné z WWW <[www.programminghistorian.org/en/lessons/building-static-sites-with-jekyll-github-pages](http://www.programminghistorian.org/en/lessons/building-static-sites-with-jekyll-github-pages)>.

## **Seznam obrázků**

1	Schéma statického webu - vlastní tvorba . . . . .	14
2	Schéma dynamického webu - vlastní tvorba . . . . .	16
3	Schéma JAM Stacku - vlastní tvorba . . . . .	19
4	Široká paleta technologií a nástrojů JAM Stacku . . . . .	20
5	Ukázka repozitáře GitHub . . . . .	30
6	Ukázka lokálního vývoje v textovém editoru Atom s využitím Terminálu . . . . .	31
7	Publikování změn do vzdáleného repozitáře GitHub prostřednictvím desktopové aplikace GitHub Desktop . . . . .	32
8	Hlavní nabídka správy webu v Netlify . . . . .	33
9	Správa větví ve vzdáleném repozitáři GitHub . . . . .	34
10	Adresářová struktura stránky v Jekyll . . . . .	38
11	Ukázka základní obrazovky hostingu Netlify . . . . .	46
12	Ukázka nastavení GitHub Pages přímo v možnostech repozitáře	47
13	Ukázka prostředí Bitbucket . . . . .	48
14	Automatické generování odkazů v navigační liště . . . . .	52
15	Struktura složky s jednotlivými příspěvky . . . . .	57
16	Struktura webu praktické části bakalářské práce . . . . .	72

## **Seznam tabulek**

1	Srovnání tradičních CMS vs. JAM Stack - vlastní tvorba . . . . .	23
2	Tabulka HTTP požadavků a akcí [19] . . . . .	25
3	Srovnání headless CMS[21] . . . . .	50

## Seznam příkladů

1	Ukázka integrace API Google Maps JavaScriptem . . . . .	24
2	Volání Netlify Forms API zabudovanými atributy . . . . .	25
3	Práce s API - vytvoření GET požadavku . . . . .	26
4	Práce s obdrženými daty API . . . . .	26
5	Instalace jazyka Ruby . . . . .	36
6	Instalace statického generátoru Jekyll . . . . .	36
7	Inicializace nového adresáře v Jekyllu a spuštění na lokálním serveru . . . . .	37
8	Spuštění lokálního vývojového serveru v Jekyllu . . . . .	38
9	Výběr titulku produktu a výpis obsahu stránky . . . . .	39
10	Instalace Hugo a vytvoření nové stránky . . . . .	40
11	Výběr tématu Hugo . . . . .	41
12	Spuštění lokálního serveru Hugo . . . . .	41
13	Instalace, vytvoření stránky a spuštění vývojového serveru Gatsby . . . . .	42
14	Zápis automatického vložení částí stránky home v systému Liquid . . . . .	52
15	Defaultní volání funkcí v PouchDB . . . . .	53
16	Využití callbacků v PouchDB . . . . .	53
17	Přidávání úkolů do lokální databáze v prohlížeči prostřednictvím API PouchDB . . . . .	53
18	Nastavení API Google Maps . . . . .	54
19	Nastavení API Netlify Forms . . . . .	55
20	Data z formuláře obdržená prostřednictvím Webhooků Zapier	55

## *SEZNAM PŘÍKLADŮ*

---

21	Výpis příspěvků blogu . . . . .	56
22	Integrace API Disqus . . . . .	57
23	Automatický výpis výsledků vyhledávání . . . . .	58
24	Výpis produktů v eshopu . . . . .	59

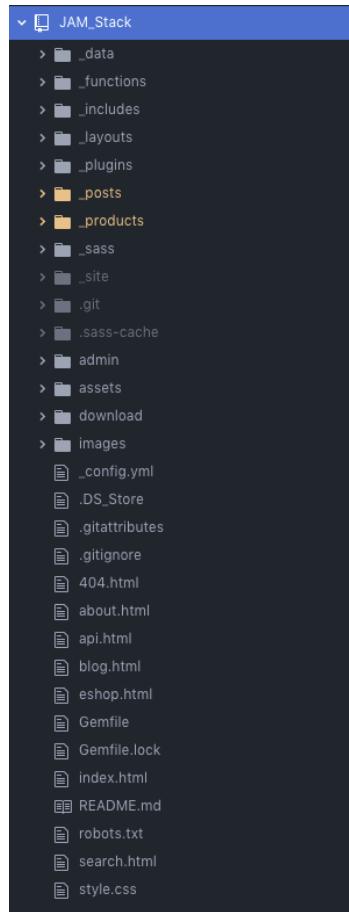
## **A Příloha**

CD se zdrojovými kódy celého webu a plné znění BP v PDF.

## **B** Příloha

Web: <https://jam-stack-site.netlify.com/> - webová stránka vytvořená čistě technologií JAM Stack.

## C Příloha



Obrázek 16: Struktura webu praktické části bakalářské práce