FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webová aplikace pro generování obrázkových memů
Student:	Josef Kříž
Vedoucí:	Ing. Petr Šlajchrt
Studijní program:	Informatika
Studijní obor:	Web a multimédia
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Analyzujte existující webové aplikace pro tvorbu internetových memů.
Navrhněte a implementujte vlastní aplikaci umožňující vytvoření memu:
- Uživatel může nahrát vlastní obrázek nebo vybrat již existující z knihovny a na něj vložit text s libovolným fontem a velikostí.
- Výsledek si uživatel může stáhnout, uložit na stránce nebo sdílet na sociálních sítích.
- Uživatelé si mohou navzájem memy hodnotit.

Klienta aplikace vytvořte jako SPA s pomocí frameworku Angular nebo React.
Pro serverovou stranu použijte vhodný jazyk a framework pro snadné škálování dle vlastních preferencí a zkušeností, například NodeJS/Express nebo Scala/Akka-http.
Pro persistenci dat použijte NoSQL databázi či obdobné cloudové řešení (DynamoDB, Datastore, ...).
Pro manipulaci s obrázky využijte vhodné knihovny, například ImageMagick.
Aplikaci vytvořte s ohledem na škálovatelnost, případně přímo provoz v cloudu (AWS, Azure, ...).
Aplikaci otestujte, včetně UI testování pomocí frameworku Protractor.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 29. prosince 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Webová aplikace pro generování obrázkových memů

Josef Kříž

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petr Šlajchrt

14. května 2018

Poděkování

Děkuji svému vedoucímu Ing. Petr Šlajchrtovi za vedení práce a cenné rady. Děkuji také svým kolegům z Jumpshotu za pomoc s testováním aplikace. V neposlední řadě si také cením morální podpory od rodiny a přátel.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 14. května 2018 .....

Odkaz na tuto práci

Kříž, Josef. *Webová aplikace pro generování obrázkových memů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018. Dostupný také z WWW: ⟨`http://memegenerator.josefkriz.com`⟩.

Abstrakt

Práce se zabývá analýzou, návrhem a implementací webové aplikace pro tvorbu internetových memů, což jsou obrázky s doplněným vtipným textem. V práci jsou popsána existující řešení a jejich výhody a nevýhody. Na základě zjištění z této části je pak následně navrhnuto řešení, po kterém následuje samotná implementace. Výsledkem je webová aplikace napsaná v jazyce JavaScript využívající framework Angular. Backend aplikace je kompletně provozován v cloudu. Součástí práce jsou i testy k ověření funkčnosti. Aplikaci lze do budoucna snadno rozšiřovat.

Klíčová slova Webová aplikace, meme generátor, návrh, implementace, JavaScript, Angular, serverless, Amazon AWS

Abstract

The goal of this bachelor thesis is to analyze, design and implement a web application for generating internet memes, which are images with a humorous text. This work contains description of existing solutions and their pros and cons. Based on this analyzis a solution is designed followed by the implementation. The result is a web appliaction in JavaScript using the Angular framework. The backend of the appliaction is entirely run in a cloud. The work includes tests for assuring functionality. The application can be easily extended in the future.

Keywords Web application, meme generator, design, implementation, JavaScript, Angular, serverless, Amazon AWS

Obsah

Seznam obrázků

Seznam tabulek

Seznam kódů

Úvod

Internetové memy jsou stále populárnější a těší se velké oblibě zvláště mezi mladými uživateli internetu. Jsou to povětšinou obrázky s vtipným textem, ačkoli existuje mnoho dalších forem memů. Meme generator umožní lidem vytvořit vlastní mem.

Výsledná aplikace pomůže zejména lidem, kteří rádi vytvářejí memy a hledají k tomu rychlý a jednoduchý nástroj. Ostatním umožní tyto memy procházet a hodnotit.

Jako člověk, který tyto obrázky pro pobavení ve volných chvílích vytváří jsem měl možnost vyzkoušet si několik existujících nástrojů, neexistuje však takový který by splňoval všechny mé požadavky. Proto jsem se rozhodl takový nástroj vytvořit.

Práci zahajuji vysvětlením, co je to vlastně mem, neboť je to nutné k pochopení účelu této práce. Následuje analýza existujících řešení a vyhodnocení požadavků na aplikaci. Následují části o návrhu a implementaci samotné aplikace.

Kapitola 1

Cíl práce

Cílem rešeršní části práce je najít a analyzovat existující řešení internetových generátorů memů. Zjistit jejich chyby a slabé stránky a také jaké technologie jsou u nich použity. Co uživatel od takové aplikace očekává.

Cílem praktické části je analyzovat a navrhnout webovou aplikaci. Vypořádat se s funkčními a nefunkčními požadavky. Na základě návrhu implementovat řešení problému za použití vhodných technologií a knihoven. Nakonec aplikaci otestovat a zhodnotit její budoucí směřování.

Kapitola 2

Co je to mem

Těžko si lze dnes představit někoho, kdo brouzdá po internetu a nenarazil ještě na žádný internetový mem. Memy jsou velkou součástí dnešní moderní kultury a memy tak, jak je známe dnes se vyvinuly zhruba před deseti lety s příchodem sociálních sítí[1].

Samotné slovo mem by se dalo definovat jako kulturní obdoba genu. Informace, kterou nese, se replikuje podobně jako DNA a přenáší se mezi organismy nebo společenstvími organismů. I memy podléhají přirozenému výběru: slabší zanikají, silnější se dále replikují a mohou se i časem přeměňovat a tím se udržet stále na vrcholu.

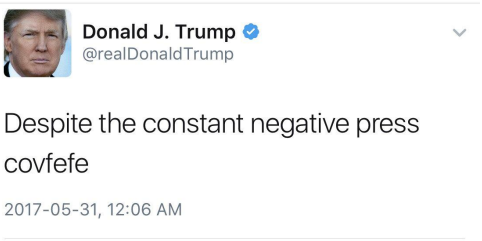
Obrázek 2.1: Ukázka memu „distracted boyfriend“

Ne náhodou totiž se slovem mem[1] přišel evoluční biolog Richard Dawkins ve své knize *The Selfish Gene* z roku 1976, která se zaměřuje na úlohu sebereplikace v evoluci. Předpokládal, že nejen biologická informace může podléhat přirozenému výběru a že cokoli, co se dokáže replikovat podléhá evoluci[2]. Tomuto jevu dal jméno „mimema“, což latinsky znamená imitovat. Později bylo toto slovo zkráceno na mem.

[1]Respektive anglický ekvivalent „meme“

S nástupem internetu v devadesátých letech postupně slovo mem nabývalo významu a dnes již o jiných než internetových memech prakticky neuvažujeme. Zprvu se šířily po internetových fórech, blozích, chatech a emaily, největší boom ovšem nastal s rozšířením sociálních sítí jako Facebook, Youtube nebo Reddit.

Memem se může stát téměř cokoli. Může to být obrázek, video, pouhé slovo nebo dokonce jen chování. Originální příspěvek nebo zdroj může být postupně překrucován, aby byla zachována komičnost. Čím je vyšší, tím rychleji se šíří a může se stát „virální“, tedy že se v krátkém čase stane velmi populární.



Obrázek 2.2: Příklad toho, jak se překlep může stát memem[4]

Memy zprvu bývaly záležitostí mladých lidí. S rostoucí popularitou se ale memy dostaly z uzavřenějších komunit do každodenního života. Sílu tohoto trendu už odhalili i v reklamních agenturách, a tak memy můžeme vidět například i na billboardech nebo jiných částech reálného světa.

Vzhledem k dosavadnímu růstu popularity lze očekávat, že zde memy ještě nějakou chvíli budou a stanou se populární nejen mezi mladými lidmi, ale napříč všemi generacemi. Aplikace vznikající v tomto projektu tento trend reflektuje a umožní i lidem bez znalostí grafických editorů snadno a rychle vytvořit obrázkový mem.

Kapitola 3

Analýza a rešerše

Protože internetové memy jsou stále populárnější[5], vzniklo již mnoho internetových nástrojů, které generování memů umožňují. Jak je běžné, bývají rozličné kvality, ať už z pohledu uživatelského rozhraní nebo počtu funkcí, které generátor umožňuje.

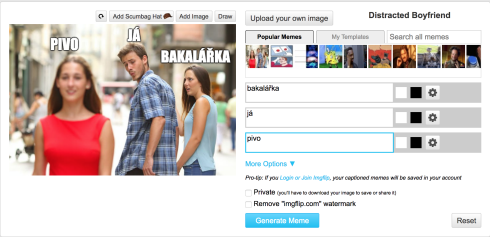
David Solomon na svém blogu uvádí seznam nejlepších online generátorů současnosti[6] a také přátelé a kolegové z mého okolí, kteří memy rádi vytváří, mezi nejlepší současné generátory řadí servery imgflip.com, memegenerator.net a imgur.com.

Ještě před samotnou analýzou těchto webů je třeba podotknout, že k vytvoření obrázkového memu lze využít libovolného grafického editoru, ať už například Malování dostupné v každém operačním systému Windows, popřípadě pokročilejší programy jako GIMP nebo Adobe Photoshop. Online generátory ovšem tuto práci velice usnadňují a umožňují vytvořit mem i méně zkušenějším uživatelům, kteří s grafickým editorem neumí pracovat a nevyužijí tak jeho pokročilejších funkcí.

3.1 imgflip.com

Ze všech analyzovaných generátorů nabízí Imgflip nejvíce funkcí. Nahrání vlastního obrázku je samozřejmostí, nabídka šablon je ovšem velice rozsáhlá a lze v ní vyhledávat. Editor umožňuje hned několik možností úpravy textu, včetně možnosti měnit font a stín. Jako jediný imgflip nabízí vložení dalšího obrázku.

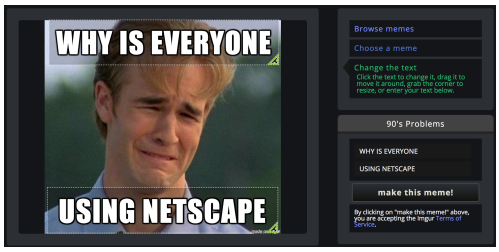
Pokud uživatel nezatrhne možnost *Private*, výsledný obrázek se uloží na stránky serveru a je veřejně dostupný. Lze k němu přidávat komentáře a sdílet na sociálních sítích jako facebook nebo reddit. Velkou nevýhodou tohoto generátoru je, že umístí do obrázku vodotisk, který lze odstranit pouze při měsíčním poplatku 2.95 $ za měsíc.



Obrázek 3.1: Ukázka editoru na serveru imgflip.com

3.2 imgur.com

Dle analytického serveru alexa.com je Imgur v top 50 nejnavštěvovanějších serverech na světě a v top 15 v USA[7], což ho pasuje na nejpoužívanější stránku svého zaměření. Nutno ovšem říci, že hlavním cílem této služby je agregace veškerého populárního a virálního grafického obsahu na internetu, který může kdokoli sdílet.



Obrázek 3.2: Grafické prostředí imgur.com

Funkce tohoto generátoru jsou velmi omezené, svou práci ovšem odvádí skvěle. Do obrázku lze vložit pouze dvě textová pole a tím výčet funkcionalit končí. Vlastnosti textu nelze nijak měnit. Grafické prostředí je ovšem velmi intuitivní a přehledné. Výsledný mem lze opět veřejně sdílet, včetně sociálních sítí. Tento generátor také přidává nesmazatelný vodotisk.

3.3 memegenerator.net

Další z populárních generátorů s atraktivním uživatelským prostředím. Samotný editor je poměrně přímočarý, uživatel má podobně jako na předchozích serverech možnost vyhledávat mezi existujícími šablonami nebo si nahrát obrázek vlastní.

Při testování této aplikace jsem narazil hned na několik chyb, některé byly pouze vizuální a jen rušivé, jedna z nich ovšem zabránila nahrát jakýkoli obrázek. Je možné, že má tato stránka pouze omezenou podporu a není vždy 100% funkční.

Obrázek 3.3: Generátor memů na memegenerator.net

Samotný editor nabízí ze všech porovnávaných serverů nejméně funkcí, ty jsou omezeny na pouhé vložení dvou textových polí do horní a spodní části obrázku, přičemž nelze měnit ani vlastnosti textu, ani pozici polí na obrázku. Také kvalita náhledu je tristní a špatně se z ní odhaduje vzhled výsledku. Zajímavou funkcionalitou je výběr jazyka, ve kterém uživatel mem tvoří, což nejspíše zjednodušuje budoucí vyhledávání na základě jazykových preferencí.

3.4 Funkční a nefunkční požadavky

Cílem této analýzy bylo porozumět procesům, které vedou k vytvoření memu. Na základě tohoto pozorování jsem schopný sestavit seznam požadavků na aplikaci. Zároveň se mohu vyvarovat chyb, které jsem v těchto aplikacích našel a naopak využít v navrhované aplikaci principy, které se zdály jako zajímavé a užitečné.

Generátor memů

- Uživatel může vybrat obrázek z knihovny nebo nahrát vlastní
- Uživatel může měnit velikost a barvu textu na obrázku
- Uživatel může přidat stín textu a jeho barvu
- Uživatel může přidávat a ubírat libovolné množství textových polí
- Uživatel si může mem stáhnout nebo uložit na stránce a tím ho zveřejnit
- Uživatel může výsledek sdílet na sociálních sítích

Prohlížeč memů

- Uživatel může procházet již hotovými memy
- Uživatel může memy hodnotit

Nefunkční požadavky

- Responzivní design
- Provoz v cloudu
- Škálovatelnost, NoSQL databáze pro perzistenci dat,

3.5 Případy užití

Tvorba memu Aplikace zobrazí kreslící plátno, na jehož pozadí si uživatel zvolí obrázek. Uživatel si přidává a odebírá textová pole podle libosti. Vyplní jednotlivá pole a upraví si vzhled textu. Aplikace veškeré změny ihned propaguje na kreslící plátno, čímž má uživatel okamžitý přehled o tom co dělá. Je-li uživatel s výsledkem spokojený, kliknutím tlačítka si obrázek stáhne nebo ho uloží na serveru. Aplikace mu poskytne odkaz na obrázek, který může jednoduše sdílet např. na Facebooku nebo jiných sociálních sítích.

Prohlížení memů Aplikace zobrazí již hotové memy, které dříve byly vytvořeny. Uživatel si může nějaký mem vybrat a ohodnotit ho tak, že mu dá palec nahoru nebo dolu. Aplikace u každého z nich zobrazuje počet takových palců.

Sdílení memů Aplikace po uložení memu zobrazí nabídku několika sociálních sítí, na kterých lze meme ihned sdílet. Nechybí nejpoužívanější sítě jako Facebook, Twitter nebo Google. Po kliknutí je uživatel ihned přesměrován na předvyplněný nový příspěvek.

Kapitola 4

Návrh

Při návrhu aplikace jsem vycházel nejen ze zadání práce, ale také z předchozích zkušeností z předmětů na fakultě a doporučení vedoucího práce. Zvolené technologie a frameworky by měly být vhodné pro tento typ aplikace a proto je třeba jejich výběru věnovat patřičnou pozornost. Použití nevhodných nástrojů by mohlo vést k nízké spolehlivosti komponent, obtížné údržbě aplikace, nemožnosti jejího rozšíření či škálovatelnosti případně i její celkové nefunkčnosti.

4.1 Architektura MVC

MVC, neboli Model–View–Controller je architektura webových aplikací, rozdělující ji na tři části. Jednotlivé komponenty jsou samostatnými logickými celky, které mezi sebou navzájem interagují. Webové frameworky využívající MVC jsou dnes vůbec nejrozšířenější[11].

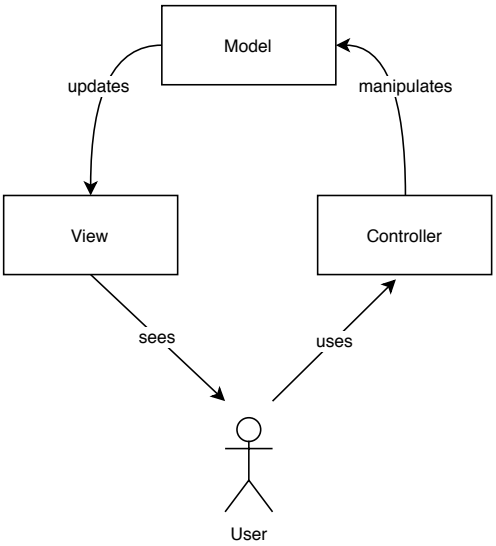
Model tvoří základ aplikace. Obsahuje logickou vrstvu, která se stará o operace s daty a jejich uložení v databázi.

View se stará o reprezentaci dat, kterou je v tomto případě webové rozhraní.

Controller je část aplikace, která má na starosti zpracování požadavků uživatele. Požadavky zpracovává a následně předává controlleru nebo view.

Komunikace probíhá podle obrázku 4.1 v následujících krocích:

1. Uživatel provede akci v uživatelském rozhraní
2. Controller akci zachytí
3. Controller může na základě této akce aktualizovat view
4. Controller změní model
5. View zobrazí změny uživateli



Obrázek 4.1: Schéma architektury MVC[8]

Nutno podotknout, že architektura MVC není striktně třívrstvá, tedy že je rozdělena na databázovou, logickou a prezentační vrstvu. Controller může být navržen velmi minimalisticky a být pouze prostředníkem mezi view a modelem.

4.2 SPA vs MPA

Z dnešního pohledu lze rozdělit webové aplikace na Multi Page Applications (MPA) a Single Page Applications (SPA). MPA jsou aplikace postavené tak, jak jsme na ně byli zvyklí od počátku internetu – tedy že uživatel při své cestě vždy načítá soubory tak, jak je vyžaduje. Navštívení nové stránky tedy znamená přenačtení celého HTML obsahu.

SPA místo toho používají architekturu, která umožňuje, aby uživatel de facto zůstával na jedné stránce a při přechodu na jinou pouze přenačetl ta data, která se změnila. Výsledkem je úspora dat, která nemusí být přenášena stále znovu[9]. Z tohoto faktu těží jak uživatel, tak programátor. Načítání je také rychlejší a plynulejší a rychleji reaguje na akce uživatele[10].

SPA aplikace jsou psány většinou za pomoci JavaScriptu. V malých projektech lze použít jQuery, což je knihovna, která velice usnadňuje psaní client-side aplikací, nicméně se zvětšující se velikostí projektu začíná být jQuery spíše přítěží. Může za to nepřehledná struktura kódu i fakt, že neobsahuje jednoduché a elegantní nástroje pro nakládání s daty.

Naštěstí dnes existuje hned několik frameworků/knihoven pro tvorbu SPA

aplikací. Mezi TOP3 patří Angular (AngularsJS resp. Angular 2/4/5/6)[2], React.js a Vue.js[11][12][13]. V následujících kapitolách se pokusím shrnout jednotlivé výhody a nevýhody těchto frameworků.

Nezodpovězenou otázkou zůstává, proč upřednostnit SPA před MPA. Jak už to bývá, obě možnosti mají své výhody i nevýhody[9][15]. S přihlédnutím k účelu aplikace a výhod SPA jsem se rozhodl pro tento typ aplikace.

4.2.1 Výhody SPA

Rychlost Protože při přechodu na další stránku není potřeba znovu přenášet celý obsah, dochází k úspoře při přenosu dat a tedy i ke zrychlení celkové rychlosti načítání.

Rychlý vývoj Existuje mnoho knihoven, které stačí nainstalovat a použít. Navíc backend a frontend vývojáři mohou pracovat paralelně, protože se jejich práce nepřekrývají.

User experience Používání moderních a atraktivních prvků v SPA je mnohem jednodušší, což má za následek zvýšení uživatelského komfortu a celkového dojmu z používání aplikace.

Trend SPA jsou trendem poslední doby a řada velkých firem na tento typ aplikace přešla. Dá se očekávat, že SPA mají budoucnost.

4.2.2 Nevýhody SPA

JavaScript SPA vyžadují podporu JavaScriptu na straně uživatele.

SEO SPA využívající AJAX mohou trpět na nedostatečnou optimalizaci ze strany internetových vyhledávačů. Nicméně minimálně v případě Google se nejedná o závažný problém[14].

Bezpečnost Javascript může obsahovat bezpečností rizika, která mohou ohrozit bezpečnost celé aplikace. Tomu se lze vyvarovat používáním aktuálních verzí frameworků a obezřetností při psaní kódu, ale i samotné prohlížeče obsahují bezpečnostní mechanizmy jako je například CORS (Cross-Origin Resource Sharing).

4.2.3 Výhody MPA

SEO Architektura MPA umožňuje snadno každou stránku optimalizovat pro internetové vyhledávače.

[2] AngularJS je původní JavaScriptový framework, jehož počátky se datují do roku 2009. V roce 2016 byla vydána nová verze Angular obsahující zásadní změny, jako například možnost použití TypeScriptu namísto JavaScriptu. Vývoj postupně pokračoval až k verzím 4 a 5, resp. 6 vydanou 2. května 2018. Podpora AngularJS přesto nadále pokračuje, není však s ostatními verzemi kompatibilní.

Snadný vývoj Při vývoji MPA není třeba tolik různorodých knihoven a technologií, což usnadňuje a zlevňuje jejich vývoj.

Rozšíření Každý web developer má zkušenosti s tvorbou MPA. Na trhu existuje mnoho hotových řešení a programování zejména statických stránek vyžaduje mnohem méně nákladů.

4.2.4 Nevýhody MPA

Rychlost Každou stránku je nutné celou načítat znova, včetně obsahu který se nemění.

Mobilní verze SPA usnadňuje vytvoření mobilní verze aplikace. Větší část kódu MPA k tomuto účelu nelze využít a tím se prodlužuje doba vývoje.

Frontend a backend Obě strany jsou více propojené, protože spolu úzce komunikují. Práce vývojářů je zde o něco komplikovanější.

4.3 Volba frameworku

V kapitole výše jsem vyjmenoval tři JavaScriptové frameworky, které se v dnešní době používají k tvorbě SPA aplikací nejvíce. Nyní se na každý z nich zaměřím podrobněji a zvážím jejich výhody a nevýhody.

4.3.1 React

Ačkoli mnoho lidí nazývá React frameworkem, ve skutečnosti se jedná o knihovnu, která v klasickém MVC návrhu představuje view. Zobrazuje UI, tedy uživatelské rozhraní, a k tomu využívá komponent – součástek, které jsou samostatné a později se z nich skládá celek. Toto zapouzdření není pouze vlastností Reactu, ale využívají ho i Angular a Vue.

Důležitou součásti Reactu jsou JSX. JSX je rozšíření podobné XML pro ECMAScript, které nemá definovanou žádnou sémantiku. Kód JSX je totiž později transpilován do standardního ECMAScriptu. Toto řešení mi přijde neintuitivní, neboť ostatní frameworky a zaběhlá praxe se snaží oddělit šablony od JavaScriptu.

```
1 class Hello extends React.Component {
2   render() {
3       return <div>Hello {this.props.toWhat}</div>;
4     }
5 }
```

Kód 4.1: Příklad sytaxe JSX[16]

```
1 class Hello extends React.Component {
2   render() {
3       return React.createElement('div', null, `Hello ${this.props
            .toWhat}`);
4     }
5 }
```

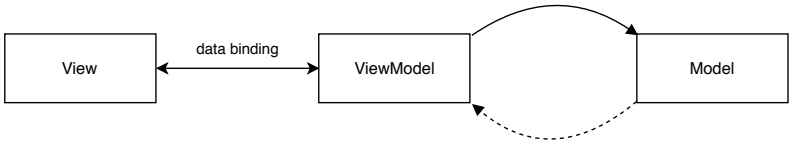
Kód 4.2: Příklad sytaxe v ES6[16]

React jsem se rozhodl nepoužít z toho důvodu, že je to pro mě zcela nová technologie, která se sice lehce pochopí, ale plné proniknutí do fungování této knihovny zabere mnoho času a úsilí, i díky přispění ne příliš kvalitní dokumentace. Dalším důvodem je fakt, že je třeba využít dalších knihoven, protože React je pouze knihovna řešící view vrstvu aplikace.

4.3.2 Vue

Framework Vue si nese několik nej. Z porovnávaných frameworků je nejmladší, je vyvíjen nejmenším týmem a v posledních letech zaznamenal nejstrmější růst[17][18]. Dle [19] je také ze všech tří nejrychlejší a paměťově nejúspornější a to téměř ve všech uskutečněných testech.

Vue sám sebe nazývá MVVM (Model–View–Viewmodel) a prezentační a bussiness logika je tedy kompletně oddělená mezivrstvou zvanou viewmodel. Ta převádí model pro view a stará se o téměř všechny akce na straně view tak, aby se s objekty snáze manipulovalo, či aby je bylo možno správně a snadno zobrazit.



Obrázek 4.2: Schéma architektury MVVM

Také Vue se skládá z komponent, které pracují se šablonami, ty se ale píší v obyčejném HTML (ačkoli lze použít i JSX). V tomto směru je velmi podobný Angularu. Rozdílná je ovšem výsledná velikost aplikace – Vue může mít okolo 20KB, oproti tomu Angular i více jak čtyřnásobek[20].

Ačkoli podíl aplikací ve Vue stále stoupá, jeho zastoupení na trhu je stále málo významné. Je obtížné najít někoho, kdo tomuto frameworku rozumí a pracuje s ním. I když se mi Vue zdá atraktivní, trpí ještě dětskými chybami a pro projekt v této práci preferuji Angular, který je už prověřený.

4.3.3 Angular

Angular je JavaScriptový framework používající TypeScript vytvořený a vyvíjený firmou Google. Postupem času se z MVC přetransformoval v MVW[21].

Takzvaný Model–View–Whatewer měl ukončit debaty o tom, kterému modelu je Angular více podobný, neboť nebylo jasné, zda naplňuje více znaky MVC nebo MVVM.

TypeScript je nadstavbou JavaScriptu, kterou vyvíjí a udržuje Microsoft. Je schválně navržen tak, aby libovolný JavaScriptový kód byl zároveň validní v TypeScriptu. Jazyk přináší několik užitečných věcí a jak název napovídá, nejdůležitější je silné typování.

Angular se skládá z komponent, ke kterým je přidružena šablona. Ta obsahuje HTML a tzv. microsyntax, umožňující modifikovat strukturu dokumentu. Silnou doménou Angularu je two-way data binding, což je synchronizace mezi modelem a view, která veškeré akce z view propaguje do modelu a naopak. Tato funkce obrovsky usnadňuje zobrazení a manipulaci s daty v uživatelském rozhraní aplikace.

```
1 <input [(ngModel)]="name" >
```

Kód 4.3: Příklad two-way bindingu pomocí direktivy ngModel

Angular je v dnešní době již poměrně rozšířený a využívá ho mnoho webů, včetně několika služeb Googlu. Oficiální webové stránky obsahují podrobnou dokumentaci, která značně usnadňuje vstup do této technologie. To je velmi užitečně vzhledem k faktu, že kromě samotného frameworku se musí programátor orientovat i v TypeScriptu.

Z těchto důvodů jsem se rozhodl vytvořit práci za pomoci Angularu, velmi ale pomohl i fakt, že jsem se s ním setkal již během studia na fakultě v předmětech WT1 a WT2 Webové technologie. Práce na projektu tedy pro mě bude alespoň zprvu o něco jednodušší a zároveň vím, že tento framework se na práci tohoto typu hodí a dokáži využít jeho silných stránek.

4.4 Backend aplikace

Po výběru technologií s níž vytvořím frontend se naskýtá logické pokračování a to zvolit, kde se budou ukládat data aplikace a jak se k nim bude přistupovat. Běžný a zaběhlý přístup je mít server zpracující požadavky a relační databázi. Během studia jsem se setkal a také sám vytvořil klasickou implementaci využívající PHP a MySQL, ovšem toto řešení považuji za zastaralé a nevhodné pro moderní aplikace.

V této práci se snažím používat moderní technologie, aby práce byla zajímavá a nejlépe předvedla použití zvolených technologií v praxi. Proto se můj výzkum zaobíral směrem k Node.js, který je stále používanější[22], případně k některým moderním jazykům z rodiny JVM jako Scala nebo Kotlin. S jazykem Scala jsem se již setkal v nepovinném předmětu MI-PSL Programování v jazyku Scala.

Po domluvě s vedoucím práce se ještě nabídla další možnost, a to využít tzv. serverless architektury, která již podle názvu postrádá jakýkoli server.

Toto řešení je novátorské, přináší řadu výhod a zajímavých výzev. Rozhodl jsem se touto možností nadále zabývat.

4.4.1 Server vs Serverless

Serverless nazýváme architekturu, při které využíváme serveru jako služby. Naše aplikace představuje funkci, která když se zrovna nevykonává, nespotřebovává žádné zdroje. Dochází zde k významné úspoře nákladů, protože není potřeba provozovat server a lidi, kteří jej udržují. Není také třeba konfigurovat server, aby bezpečně zvládl přenosové špičky (například neděle večer), i když po zbytek týdne pracuje jen na zlomek své kapacity adochází tak k efektivnějšímu využití kapacit.

Takzvané Backend as a Service (BaaS) popř. Function as a Service (FaaS) umožňuje programátorovi plně se zaměřit na psaní kódu a strávit méně času nad infrastrukturou, kterou jeho aplikace potřebuje. Dochází i k úspoře časové – není třeba nic nastavovat. Platí se pouze za čas, který je spotřebován zpracováním požadavku.

Samozřejmě jsou zde i nevýhody. Vlastním-li server, mám k němu přístup 24/7 a mohu prakticky libovolně měnit jeho konfiguraci, případně si instalovat další software podle své chuti. Vzhledem k tomu, že tento projekt vyžaduje pouze základní funkce se toto omezení nezdá závažné.

Validní otázka se také týká bezpečnosti, kdy provozem databáze v cloudu přenechávám svá data na serverech třetí strany, což je velmi aktuální téma díky zavedení GDPR. Meme generátor ale žádná citlivá data (natož osobní) uchovat potřebovat nebude a tak se touto otázkou nadále nezabývám.

Protože zde neexistují žádné důvody, proč tuto aplikaci neprovozovat v cloudu (a tedy využít serverless architekturu), rozhodl jsem se nejít cestou vlastního serveru. Využiji tedy služeb jedné z korporací, které cloud computing nabízí.

Samozřejmě důležitou otázkou je, zda není velké riziko svěřit celý svůj backend cizí korporaci. Co kdyby jednoho dne prostě přestali své služby poskytovat? Na to se dá odpovědět, že je to extrémně nepravděpodobné. Např. Amazon poskytuje své služby mnoha významným firmám a má tolik zákazníků, že jakákoli změna dostupnosti, rozsahu nebo ceny jeho služeb by vyvolala protesty. A jak se dozvěděli v Rusku, tak velkého hráče nelze jednoduše „vypnout“[23]. A i když to nemusí být jednoduché, vždy lze přejít k některému z konkurentů[24].

4.5 Cloud computingové služby

Dnes trhu s cloudovými řešeními dominují tři hráči: Amazon AWS, Microsoft Azure a Google Cloud Platform. Všichni se nezaměřují pouze na firemní klientelu a tak jejich služby může využívat každý. Nyní se každému budu věnovat podrobněji, na první pohled je ovšem zřejmé, že jednotlivé produkty se příliš

neliší a tak hlavním rozhodovacím kritériem bude cena, resp. rozsah služeb které jsou zdarma.

4.5.1 Microsoft Azure

Výrobce světové nejrozšířenějšího operačního systému se v posledních letech zaměřil na cloudová řešení v podobě produktu Azure. Není sice tak daleko jako Amazon s AWS, nicméně tomuto směru věnuje v poslední době maximum času a Azure je již plně funkční služba ověřená v praxi v soukromém i komerčním sektoru.

Azure má nyní přes 50 datacenter po celém světě a jeho služby jsou tak dostupné ve více jak 140 zemích. To by mělo být údajně více než nabízí konkurence[25]. Z pohledu požadavků mé aplikace ovšem hrají větší roli dostupné služby.

Azure Storage je služba cloudového uložiště, která nabízí vysokou odolnost a dostupnost. Data jsou zálohována (i geograficky) a tak se uživatel nemusí bát výpadku hardwaru. Výhodou je také takřka neomezená škálovatelnost[26].

Azure Cosmos DB je NoSQL databáze slibující nízkou odezvu a globální dostupnost, kde data jsou automaticky replikována do datacenter s největší poptávkou. Samozřejmostí je vysoká spolehlivost[27].

Azure Functions je jednoduchá služba, která umožňuje spouštět kód v cloudu. Lze se pak plně soustředit na kód a starosti o infrastrukturu přenechat jiným. Je podporována celá paleta jazyků včetně C#, Node.js, Java nebo PHP[28].

Storage	5 GB
Cosmos DB	5 GB 400 požadavků
Functions	1 000 000 požadavků/měsíc

Tabulka 4.1: Nabízené služby zdarma od Azure[3][33]

Tímto samozřejmě výčet služeb Azure nekončí, tyto jsou však pro provoz Meme generátoru postačující. Google i Amazon nabízejí prakticky totožné produkty podobných vlastností, proto se jim již nebudu věnovat tak podrobně a raději se zaměřím na rozdíly mezi nimi.

[3]Pro Storage a Cosmos DB platí pouze na 12 měsíců od data registrace

4.5.2 Google Computing Platform

Google nezůstává pozadu a nabízí cloudové služby pod názvem Computing Platform. Ze všech tří porovnávaných společností Google oslovil nejméně firemních zákazníků, zato se ovšem těší poměrně velké oblibě v open-source[29].

Nabídka produktů zde víceméně kopíruje nabídku AWS. Google nabízí cloudové uložiště Cloud Storage, hned několik databází, mezi které patří NoSQL Cloud Datastore a také obdobu AWS Lambda nebo Azure Functions shodně pojmenovanou Functions[4]. Rozdíly oproti konkurenci se odehrávají zejména v ceníku a vlastnosti jednotlivých služeb se téměř neliší a lze mezi nimi libovolně vybírat.

Storage	5 GB 5 000 operací třídy A/měsíc[5] 50 000 operací třídy B/měsíc[6]
Datastore	1 GB 50 000 čtení/den 20 000 zápisů/den 20 000 mazání/den
Functions	2 000 000 požadavků/měsíc 400 000 GB/sekund paměti 200 000 GHz/sekund výpočtů

Tabulka 4.2: Nabízené služby zdarma od GCP[7][32]

4.5.3 Amazon AWS

Amazon dominuje tomuto trhu již od jeho vstupu v roce 2006[30][29]. Za tu dobu vyvinul obsáhlé portfolio služeb zahrnující databáze, computing i umělou inteligenci. Jeho služeb využívá mnoho firemních i soukromých zákazníků.

Opět zde najdeme stejné služby jako u Azure a GCP, jen pod jinými názvy: S3 je název datového uložiště, pod názvem DynamoDB najdeme NoSQL databázi a Lambda je ekvivalentem Azure Functions a Google Functions. Protože nemá cenu opět popisovat totožné vlastnosti těchto služeb, raději se zaměřím na ceníky, které v tomto případě jsou hlavním kritériem výběru cloudového poskytovatele.

Jak lze vidět, Amazon celkově nabízí nejlepší podmínky, a přihlédneme-li k tomu, že služby Azure jsou zdarma pouze 12 měsíců od data registrace, Amazon je zde jasnou volbou. Připočtu-li ještě nejširší nabídku služeb a velmi

[4] V květnu 2018 stále pouze jako beta verze.
[5] Například operace insert nebo update.
[6] Například operace get.
[7] K uvedeným limitům dostane uživatel také 300 $ na první rok zdarma.
[8] Pouze na prvních 12 měsíců od registrace.

S3[8]	5 GB 20 000 GET požadavků/měsíc 2 000 PUT požadavků/měsíc
DynamoDB	25 GB cca 200 milionů požadavků/měsíc
Lambda	1 000 000 požadavků/měsíc 3,2 milionu sekund výpočtů/měsíc

Tabulka 4.3: Nabízené služby zdarma od AWS[31]

podrobnou dokumentaci ke každé službě, rozhodnutí zvolit AWS pro mě bylo velmi jednoduché.

4.5.3.1 S3

Amazon S3 (Simple Storage Service) je internetové uložiště, do kterého lze nahrát libovolné množství dat jakéhokoli typu, které jsou pak přístupné odkudkoli z internetu. Chlubí se 99.99% dostupností[34], jedná se o stejné servery na kterých Amazon provozuje své vlastní služby v oblasti e-commerce. Data jsou pokaždé distribuována do tří různých datacenter, vzdálených minimálně kilometry od sebe.

Data jsou organizována do tzv. buckets, jméno bucketu musí být unikátní napříč všemi datacentry na světě. K datům lze přistupovat hned několika rozhraními (REST, SOAP a BitTorrent, lze využít i webové rozhraní), samozřejmostí je podrobné nastavení práv, kdy lze bucket nastavit kompletně veřejným a naopak.

Platí se pouze za to, co je opravdu používáno – za každou uloženou jednotku GB a to ne více než 0,025 $ za měsíc[35] a dále za každých 1000 požadavků na tato data, jejichž cena se pohybuje mezi 0,0004 $ až 0,005 $ v závislosti na jejich typu. Dokud se ovšem nepřekročí hodnoty uvedené v tabulce 4.3, neplatí se nic.

4.5.3.2 DynamoDB

DynamoDB je NoSQL databáze zaměřená na rychlost a snadnou škálovatelnost. Uživatel je zproštěn veškerých administrátorských záležitostí. Protože se nejedná o relační databázi, není třeba definovat schéma není ani třeba striktně dodržovat datovou strukturu.

S vzrůstajícím počtem dat automaticky dochází k alokaci dalších prostředků a proto je stále zachována nízká latence na straně serveru v řádech milisekund[36]. Je-li zapnutý tzv. auto-scale, vše se řeší samo a automaticky. Bezpečnost a vysoká spolehlivost jsou už samozřejmostí.

Platí se za místo, které data zabírají a také za uskutečněné přenosy. Každý uložený GB dat stojí od 0,25 $ měsíčně a za propustnost se platí dle alokova-

ných jednotek pro čtení nebo zápis[9] a to 0,09 $ respektive 0,47 $ za měsíc[37]. Stejně jako u S3 platí, že vejdu-li se do tzv. free tier (uvedeno v tabulce 4.3), nemusím platit nic.

4.5.3.3 Lambda

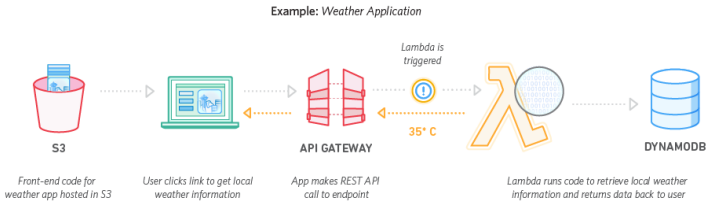
Lambda se nazývá platforma, do které můžeme nahrát svůj kód, který se při námi definovaných událostech spustí a vykoná. Jako jednoduchý příklad lze uvést případ, kdy zavolání API endpointu spustí lambdu, která spustí dotaz do databáze a výsledek vrátí jako odpověď.

Obrovskou výhodou je, že Lambda dokáže plně nahradit server, který by se za běžných okolností staral o zpracovávání příchozích požadavků. Není třeba trávit čas sháněním a konfigurací serveru, veškeré starosti spojené s infrastrukturou jdou na bedra poskytovateli. Lambda také automaticky škáluje podle aktuální zátěže a odpadá tak starost o zajištění dostatečné kapacity jednotlivých komponent[38].

Účtován je pouze opravdový čas, které servery Amazonu stráví vypočítáváním požadavku. Platí se za každých 100 ms a pokud se nic nevykonává, ani se nic neúčtuje. Opět není třeba platit nic, nepřekročí-li se limit služby zdarma (uvedeno v tabulce 4.3), následně se platí 0,0000002 $ za každý požadavek a 0,00001667$ za každou sekundu, kterou funkce běží[39].

4.5.3.4 API Gateway

AWS nabízí mnoho dalších produktů, díky kterým lze jednotlivé produkty propojovat do funkčních celků. Jedním z nich je i API Gateway, jejíž funkce je vytvářet, spravovat a monitorovat API rozhraní. Díky zabudované integraci s ostatními službami je navíc veškeré nastavení věcí doslova pár kliků.



Obrázek 4.3: Příklad propojení služeb AWS[38]

Právě ve spojení s dalšími produkty dává API Gateway smysl, protože samostatně jsou její funkce omezené. Typickým sestavením je například Lambda, která je spouštěna z API Gateway při zavolání určitého endpointu. Lambda dostane parametry požadavku a provede s nimi požadované operace.

[9]Tzv. Read/Write Capacity Unit dokáže obsloužit cca dvě čtení/jeden zápis za sekundu, souhrnně až 5,2 milionu čtení resp. 2,5 milionu zápisů za měsíc[37]

Ať už přichází jeden nebo milión požadavků denně, API Gateway se postará o jejich zpracování. Zákazníkovi je účtován 3,50$ za každý milion požadavků, přičemž první milion je zdarma[40]. Dále se účtuje nejvýše 0,09 $ za každý přenesený GB.

4.5.3.5 CloudWatch, Cloudfront a další

Služby v této kapitole zmiňuji, protože je lze využít vzhledem k rozhodnutí provozovat aplikaci v cloudu. Běh aplikace na nich není závislý, mohly by ovšem být užitečné.

Amazon CloudWatch je monitorovací služba, která sleduje spotřebu zdrojů mnou používaných služeb. Produkt například sleduje a loguje výstupy lambda funkcí, databáze, API a dalších a dokáže je přehledně zobrazit. Užitečnou funkcí je i možnost nastavení upozornění, a to nejen na neobvyklý provoz, ale třeba i na blížící se hranici služeb zdarma. Pokud tato situace nastane, můžu se rozhodnout, zda aplikaci vypnout a nebo doplatit za využité prostředky.

Amazon CloudFront je CDN služba, která distribuuje požadavky po různých centrech Amazonu a zajišťuje nízkou odezvu s vysokou přenosovou rychlostí. U Meme generátoru se nepředpokládá tak velký provoz, aby bylo třeba služeb CDN využít, je ale dobré o této možnosti vědět. V případě nulových nákladů se i nabízí tuto službu vyzkoušet, ač nebude využita.

Jak je vidět, AWS nabízí nepřeberné množství služeb. Za zmínku ještě stojí Amazon EC2, které poskytuje výpočetní kapacitu pro běh například Node.js serveru, na hostování statických webových stránek ovšem není třeba. V případě dalšího rozšiřování aplikace by ale možná nadešel čas o této službě přemýšlet.

4.6 Struktura databáze

Při návrhu běžné relační databáze nastává fáze, při které se definují jednotlivé entity a vztahy mezi nimi. V případě NoSQL ovšem nelze takto přemýšlet, neboť struktura těchto databází nedovoluje zmíněné vztahy implementovat.

Již samotný název evokuje, že zde chybí SQL. Jelikož data nemají struktura, respektive ta se může měnit, postrádalo by dotazování jazykem SQL smysl. Tato vlastnost způsobuje, že nerelační databáze jsou rychlejší a dobře zvládají vysokou zátěž[41].

Rozhodl jsem se implementovat tzv. *Key-value Data Model*, který je v DynamoDB nativně podporován. Jednotlivé záznamy stále mohou mít libovolnou strukturu, musí ovšem obsahovat primární klíč (a v mém případě i klíč sekundární), podle kterého lze jednotlivé objekty jednoznačně identifikovat (zde jedinečnou kombinací klíčů).

```
var cars = [

{ Model: "BMW", Color: "Red", Manufactured: 2016 },

{ Model: "Mercedes", Type: "Coupe", Color: "Black", Manufactured:
    "1-1-2017" }

];
```

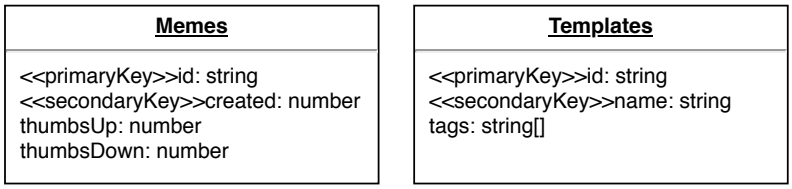
Kód 4.4: Ukázkový dokument v NoSQL databázi[41]

```
SELECT Orders.OrderID, Customers.Name, Orders.Date

FROM Orders

INNER JOIN Customers

ON Orders.CustID = Customers.CustID
```

Kód 4.5: Ukázka dotazu v relační databázi pomocí SQL[41]

V databázi budou dvě nezávislé tabulky: Memes a Templates. Meme má primární klíč název souboru a sekundární klíč timestamp, kdy bylo vytvořeno. To zabrání případné kolizi v názvech souborů, protože je velmi nepravděpodobné, aby dva stejné memy vznikly v tu samou milisekundu.

V druhé tabulce Templates budou uloženy šablony. Šablona je identifikována opět názvem souboru a také jménem. Unikátní kombinace názvu s id nutně nezaručuje, že se v databázi nebudou nacházet dvě stejné šablony, protože lze jeden obrázek nahrát pod dvěma jmény. Jedním z řešení by mohla být manuální kontrola člověkem, který nahrávaný obrázek posoudí, nicméně tato kontrola není součástí návrhu.



Obrázek 4.4: Schéma tabulek databáze

Zbylé hodnoty, tedy thumbsUp a thumbsDown v Meme a pole tags v Template jsou sice nadefinovány, je ale záležitost implementace aplikace, aby byly skutečně vyplněny. To má za následek například to, že se v databázi mohou vyskytovat šablony bez žádných tagů. Tags je pole string řetězců, které definují co je v obrázku zobrazeno. To by mělo usnadnit případné budoucí vyhledávání nebo filtrování. Atributy thumbsUp a thumbsDown budou uchovávat číselnou informaci o hodnocení daného memu.

Záznamy záměrně neobsahují obrázková data. Id je schválně zvoleno tak, aby obsahovalo název souboru, který se nachází na jiném místě. K perzistenci

dat se totiž více vyplatí využívat S3, které je k těmto účelům určené a zároveň nabízí pokročilejší možnosti omezení přístupu.

Kapitola 5

Implementace

Existuje několik metod vývoje softwaru, z kterých jsem si vybral agilní přístup na základě mých předešlých zkušeností a preferencí. Zjednodušeně řečeno se jedná o metodu vývoje po malých kouscích, kdy namísto dodání produktu do určitého data na aplikaci pracuji inkrementálně a dokončuji ji po malých kouscích. Cílem je udělat co nejdříve funkční prototyp, který se následně průběžně plánuje, testuje a vyvíjí i s přihlédnutím k požadavkům zákazníka.

Tento výběr podporuje i fakt, že návrh aplikace není příliš specifický a to i z toho důvodu, aby bylo během vývoje možné přehodnotit předešlá rozhodnutí. Například uživatelské rozhraní se může průběžně měnit tak, jak do něj přibývají nové funkce.

V této kapitole popíši, jak implementace probíhala, jaké přinesla nástrahy a co bylo jejich řešením. Součástí je i oddíl o testování, bez kterého se dnes nelze obejít, a na závěr zhodnotím další možnosti rozvoje aplikace.

5.1 Verzování

Během vývoje aplikace je nutné mít přehled o provedených změnách a také umožnit na projektu pracovat více lidem současně. Nepředpokládá se, že by během psaní této práce na aplikaci pracoval více než jeden člověk, v budoucnu je však vhodné být na tuto možnost připravený.

Možností je hned několik, například SVN, Bit-Keeper nebo Mercurial, dnes je už však jasnou volbou Git, který dnes naprosto dominuje a ukázal se jako mocný a spolehlivý nástroj. Má širokou uživatelskou základnu, což usnadňuje řešení jakéhokoli problému a hlavní předností je, že je zdarma.

Git je distribuovaný VCS (Version Control System), tedy systém pro správu verzí, který sleduje veškeré provedené změny a umožňuje tak se kdykoli vrátit zpět v historii. Výrazně zjednodušuje práci v týmu a slévání změn.

Založení nového repozitáře je velmi jednoduché a doslova záležitostí jednoho příkazu. Stačí mít nainstalovaný Git a spustit `git init [DIRECTORY]`, pří-

padně `git clone URL [DIRECTORY]`[42], chci-li repozitář vytvořit klonováním již existujícího. Této možnosti jsem nevyužil z důvodů uvedených v 5.2.1.

Nyní zmíním pár nejpoužívanějších příkazů, které jsem během implementace používal nejčastěji[10]:

git add přidá soubor nebo soubory do tzv. staging area, kde jsou následně připravené k zahrnutí do revize.

git commit vezme změny ze staging area a vytvoří z nich revizi. Revize je jednotlivý logický kus práce.

git push odešle změny na server, který repozitář spravuje. Ostatní si poté tyto změny mohou stáhnout k sobě pomocí příkazu `git pull`.

Aby případný tým mohl spolupracovat, hodí se využít některých služeb pro správu repozitářů, kterých je hned několik. Jmenuji například Gitlab nebo Github, já se rozhodl pro Bitbucket, se kterým jsem neměl zkušenosti, ale obdržel jsem na něj dobré reference. Výběr zde není nějak podstatný, všechny zmíněné služby jsou zdarma a nabízejí víceméně totéž.

5.2 Frontend

Již v návrhu jsem zvolil Angular jako framework pro vytvoření uživatelského rozhraní aplikace. Protože se liší od AngularJS, se kterým mám zkušenosti z předmětů BI-WT1 a BI-WT2, pomohla mi oficiální dokumentace, která obsahuje i detailně popsanou ukázkovou aplikaci.

Angular je vytvořen jako kolekce několika balíků pro Node.js, což je běhové prostředí pro JavaScriptové aplikace, nicméně to nebude k vývoji této aplikace potřeba. Balíky jsou samostatné programy, které na sobě mohou vzájemně záviset. Pro správu těchto balíků (například instalaci, aktualizaci, mazání a další) se využívá jeden ze správců balíků, jako je yarn nebo npm. Během své krátké praxe jsem měl zkušenost s oběma a yarn se zdá více odladěný, rychlejší a příjemnější na ovládání, a proto jsem se rozhodl využít právě jej.

5.2.1 Angular CLI

Vytvoření a vývoj výrazně usnadňuje nástroj Angular CLI, který obsahuje hned několik funkcí, bez kterých nelze Angular aplikace téměř vyvíjet. Toto rozhraní příkazové řádky mi umožnilo snadno nastartovat nový projekt a vidět v reálném čase změny, které provádím v kódu, rovnou v prohlížeči.

Prvním příkazem, kterým začíná snad každý projekt, je `ng new Meme-Generator` . Tento příkaz vygeneruje strukturu projektu s potřebnými soubory a adresáři[43]:

[10] Ve skutečnosti je nemusím zadávat osobně, ale dělá to za mě IDE, které nabízí líbivější UI.

```
Meme-Generator
├── e2e ..... adresář s end–to–end testy
│   ├── app.e2e-spec.ts
│   ├── app.po.ts
│   └── tsconfig.e2e.json
├── node_modules/ ..... adresář s balíky z package.json
├── src/ ..... zdrojový adresář aplikace
├── .angular-cli.json ..... konfigurační soubor Angular CLI
├── .editorconfig ..... soubor s nastaveními editoru
├── .gitignore ..... seznam ignorovaných souborů pro Git
├── karma.conf.js ..... konfigurační soubor unit testů
├── package.json ..... seznam balíků, na kterých aplikace závisí
├── protractor.conf.js ..... konfigurační soubor e2e testů
├── README.md ..... soubor se základní dokumentací
├── tsconfig.json ..... nastavení TypeScript kompilátoru
└── tslint.json ..... konfigurační soubor pro TSLint
```

```
src
├── app ..... app komponenta
│   ├── app.component.css ..... CSS styly
│   ├── app.component.html ..... HTML šablona
│   ├── app.component.spec.ts ..... unit testy
│   ├── app.component.ts ..... definice komponenty
│   └── app.module.ts ..... definice kořenového modulu AppModule
├── assets ..... adresář s médii
│   └── .gitkeep ..... soubor Git pro verzování prázdného adresáře
├── environments ..... adresář s definicemi prostředí
│   ├── environment.prod.ts
│   └── environment.ts
├── favicon.ico ..... ikonka stránky v prohlížeči
├── index.html ..... hlavní HTML stránka
├── main.ts ..... hlavní skript, který kompiluje aplikaci
├── polyfills.ts ..... skript pro normalizaci standardů v prohlížečích
├── styles.css ..... globální soubor se styly
├── test.ts ..... skript pro kompilaci unit testů
├── tsconfig.app.json ..... nastavení TS kompilátoru pro aplikaci
└── tsconfig.spec.json ..... nastavení TS kompilátoru pro unit testy
```

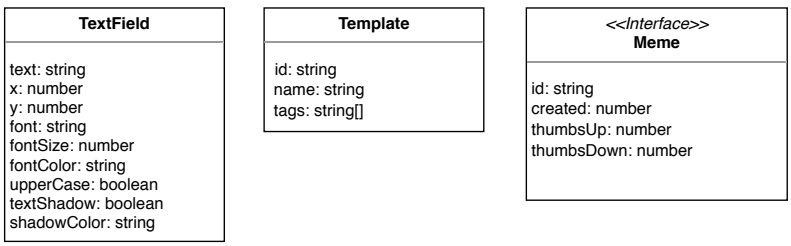
Aplikace je po vygenerování rovnou připravena ke spuštění (ačkoli neobsahuje žádný zajímavý obsah), stačí použít další z příkazů: `ng serve`. Příkaz spustí server a sleduje zdrojové soubory. Zároveň otevře okno prohlížeče[11], ve kterém lze aplikaci otestovat. V okamžiku, kdy dojde k nějaké změně, se automaticky aplikace přestaví a náhled v prohlížeči zaktualizuje.

Mezi další užitečné příkazy patří `ng generate`, usnadňující vytváření nových komponent, modulů, services a dalších. `ng test` sestaví aplikaci a spustí unit testy s frameworkem Karma. Stejně jako `ng serve` sleduje veškeré změny a ihned aktualizuje výsledky testů. `ng e2e` slouží ke spouštění end–to–end testů, testy implicitně probíhají v prohlížeči Chrome a po skončení vypíše výsledky do konzole.

Angular CLI umožňuje rychlý start, kde stačí upravit komponentu `app`, případně zavolat příkaz `ng generate component meme-editor` a ihned začít stavět aplikaci. Celý projekt je také rovnou připraven k verzování, protože se chová jako Git repozitář. První revize je zde vygenerovaná a nese jméno `chore: initial commit from @angular/cli`.

5.2.2 Objekty

K uložení informací o memech a dalších entitách jsem vytvořil proprietární objekty a rozhraní, které s nimi usnadní manipulaci. Jmenovitě jde o objekt textového pole, kterých může být na obrázku libovolné množství (včetně žádného) a je příhodné všechny vlastnosti pole držet „pod jednou střechou“. Dalším objektem je šablona, která se striktně drží formy ukázané ve schéma 4.4. Uchovává si unikátní dvojici id a name a může také obsahovat nepovinné pole tagů.



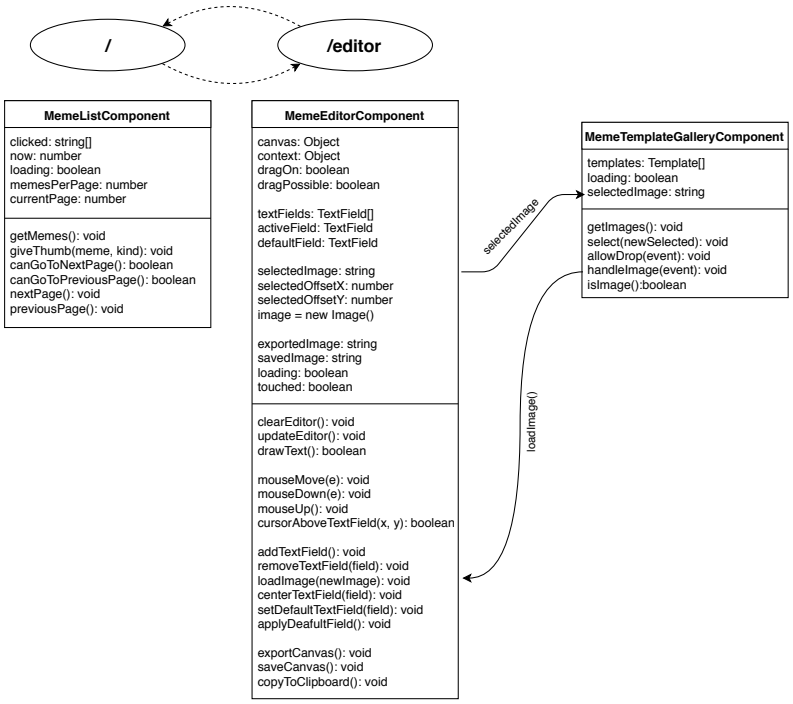
Obrázek 5.1: Modely objektů použitých v Meme generátoru

Poslední je interface Meme, které definuje objekty s vlastnostmi memu. V aplikaci samotné se nikdy Meme neinstancují, neboť jejich vytváření je doménou výhradní lambda funkcí, které je po vytvoření uloží do databáze a aplikace je pouze přijímá. Meme obsahuje atributy id a created, které jsou primárním resp. sekundárním klíčem a tak musí být každá dvojice jedinečná.

[11] S přepínačem `--open`, server poslouchá na adrese `http://localhost:4200/`

5.2.3 Komponenty

Komponenty jsou základním stavebním kamenem Angularu. Komponenta většinou obsahuje samostatnou funkcionalitu, které pak jejich skládáním dávají dohromady celek. Kdyby se jednotlivé závislosti mezi komponentami namalovaly, obrázek by připomínal stromovou strukturu. Součástí každé komponenty je TypeScript soubor a HTML šablona.



Obrázek 5.2: Schéma komponent aplikace s routami, které obsluhují

Na obrázku 5.2 jsou znázorněny komponenty a jejich atributy s metodami. Jak je naznačeno v horní části, komponenta *MemeList* má na starosti obsah kořenové stránky a *MemeEditor* zase část se samotným kreslicím plátnem pro tvorbu memu. Mezi těmito dvěma umístěními lze libovolně přecházet pomocí tlačítek, přičemž Angular přenechává obsluhu správným komponentám v závislosti na cestě.

Součástí editoru memů je i sekce pro výběr některé z existujících šablon, což bylo vyčleněno do samostatné komponenty *MemeTemplateGallery*. O její vykreslení se stará nadřazená *MemeEditor* a jak lze poznat ze schéma, jsou na sobě navzájem závislé.

Galerie si bere jako parametr obrázek, který se právě nachází v editoru a může podle toho upravit zobrazení galerie. Naopak vybere-li si uživatel nějaký jiný obrázek (nebo ho nahraje ze svého počítače), funkce zavolá metodu `loadImage()` z nadřazené komponenty.

Každá komponenta je implementována jako běžná třída s tzv. dekorátorem `@Component`. Ten obsahuje metadata, která určují jak má být komponenta zpracována, instanciována a použita. Třída také implementuje rozhraní `onInit` a obsahuje metodu `ngOnInit()`, která je důležitou součásti životního cyklu komponenty a volá se krátce po vytvoření objektu.

```
1  @Component({
2    selector: 'meme-templates',
3    templateUrl: './meme-template-gallery.component.html',
4    styleUrls: ['./meme-template-gallery.component.scss']
5  })
6  export class MemeTemplateGalleryComponent implements OnInit {
7    @Input('selectedImage') selected: string;
8    @Output() loadImage = new EventEmitter<string>();
```

Kód 5.1: Ukázka dekorátorů `@Component`, `@Input` a `@Output` v komponentě *MemeTemplateGalleryComponent*

```
1  <meme-templates [selectedImage]="selectedImage" (loadImage)="
       loadImage($event)"></meme-templates>
```

Kód 5.2: Příklad vložení komponenty selektorem v šabloně *MemeEditorComponent*

Na příkladech 5.1 a 5.2 lze názorně vidět, jak lehká je implementace vztahů zmíněných v předchozích odstavcích. V HTML šabloně nadřazené komponenty stačí doplnit zvolený selektor a jako atributy mu předat vybraný obrázek a funkci pro změnu šablony. Angular se postará o zbytek, tedy o správné předání argumentů a propagaci událostí do nadřazené komponenty.

Velmi snadné je i zobrazení dat z komponenty uživateli. Nejzákladnějším nástrojem je interpolace, kterou lze v šabloně zobrazit libovolnou proměnnou z komponenty pomocí dvojitých chlupatých závorek (`{{variable_name}}`). Použití lze vidět v kódu 5.3 u proměnné `textField.text`.

```
1  <ul>
2    <li *ngFor="let textField of textFields">
3      <h3>{{textField.text}}</h3>
4      <form>
5          ...
6              <input [(ngModel)]="textField.text" (keyup)="
                   updateEditor()" placeholder="Insert text">
```

Kód 5.3: Příklad použití interpolace a direktiv v šabloně *MemeEditorComponent*

V příkladu 5.3 je také použita direktiva `ngFor`, která iteruje skrze pole *textFields* obsahující textové pole v memu. Pro každý prvek pole v šabloně vytvoří položku seznamu s formulářem pro editací vlastností jako velikost nebo barva textu. Další podobnou direktivou je `ngIf`, která nejprve vyhodnotí podmínku a poté přidá respektive nepřidá element.

Neméně zajímavé funkce byly použity pro zpracování uživatelského vstupu. Na řadu přichází jedna z hlavních předností Angularu: Two-way data binding.

`[(ngModel)]="textField.text"` u tagu `<input>` zajistí, že text objektu je zobrazen jako hodnota vstupního políčka a zároveň veškeré změny provedené uživatelem jsou ihned zpracovávány komponentou v reálném čase. Lze tak pozorovat měnící se text v obrázku zároveň s tím, jak vkládám znaky. Aby to celé fungovalo, je ještě při každé změně nutné překreslit plátno, o což se stará funkce `updateEditor`, která je volána po každém stisknutí znaku na klávesnici.

5.2.4 Routování

Protože se jedná o SPA aplikaci, uživatel nepřechází mezi jednotlivými soubory a adresa v prohlížeči se nemění. Toto není žádoucí hned z několika důvodů, zejména z pohledu uživatele by mohlo být matoucí, že nedochází ke změně a chtěl-li by si uložit stránku do oblíbených, nemusela by to být ta samá stránka na které se nachází.

Angular k tomuto účelu nabízí routování, kde je konvencí svěřit tuto problematiku separátnímu modulu `app-routing`. Do HTML kódu pak jen stačí přidat tag `<router-outlet></router-outlet>`, který v závislosti na URL a nastavení v modulu zobrazí odpovídající komponentu. Angular k tomu využívá HTML5 History API, které umožňuje manipulovat s historií procházení prohlížeče.

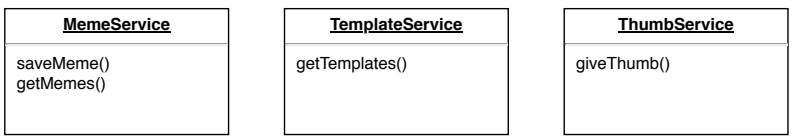
```
1 const routes: Routes = [
2   { path: 'editor', component: MemeEditorComponent },
3   { path: '', component: MemeListComponent},
4   { path: '**', redirectTo: '' } //fallback to root
5 ];
```

Kód 5.4: Část souboru `app-routing-module.ts`, která má na starosti routování

Routování v Meme generátoru je velmi jednoduché, neboť se skládá pouze ze dvou stránek. V kořenu je zobrazen seznam memů a přejde-li uživatel na cestu `/editor` vykreslí se plátno pro editaci. Speciálním případem je neexistující cesta, v takovém případě je uživatel přesměrován zpět na hlavní stránku.

5.2.5 Services

Aby byla aplikace schopná přijímat a odesílat zobrazovaná data, existují speciální třídy zvané *services*. Svěřit komunikaci se serverem komponentám by bylo nepřehledné i špatné z hlediska návrhu. Services se přidávají jako závislost do komponent, které volají jejich metody. Metoda typicky zvolá některý z endpointů serveru a vrátí objekt *Observable*.



Obrázek 5.3: Services a jejich metody

Na obrázku 5.3 lze vidět services, které Meme generátor potřebuje ke svému fungování. Nejdůležitější je *MemeService*, která má metody pro uložení hotového memu do databáze a uložiště a metodu pro získání již existující memů, která je potřeba pro správné fungování hlavní stránky. Pro uložení memu je využito metody POST a jediným parametrem je obrázek zakódovaný v base64. Druhá metoda využívá GET a vrátí pole objektů.

Service *TemplateService* má jedinou metodu, která vrátí seznam šablon. *ThumbService* má nastarosti hodnocení memů, jako parametr si bere mem, který je hodnocen a zda je hodnocen kladně nebo záporně.

```
1 public getMemes(): Observable<Meme[]> {
2     return this.http
3       .get(`${API_URL}/meme`)
4       .map(response => response['Items'])
5       .catch((error: Response | any) => {
6         console.error('API Service call failed: ', error);
7         return Observable.throw(error);
8       });
```

Kód 5.5: Příklad metody, která vrací seznam hotových memů

Services využívají modul *HttpClient*, který je standardní součástí Angularu. Ten obsahuje metody *get*, *post* a další a vrací objekt *Observable*. Od Angularu verze 5 již není potřeba deserializovat příchozí odpovědi z formátu JSON. Na ukázce 5.5 je vidět i ošetření chyb v případech, kdy se ze serveru nevrátí odpověď se statusem 200 OK.

5.2.6 Stylování

Aby aplikace vypadala atraktivně a pohodlně se ovládala, je nutné jí dát zajímavý vzhled. Nejpřímočařejší možnost by byla si napsat vlastní CSS styly, takovou možnost jsem nevyužil, protože nejsem příliš zdatný webdesigner a proto jsem se rozhodl použít jednu z knihoven usnadňující tvorbu uživatelských rozhraní.

Z předchozích zkušeností a krátkého vyhledávání na internetu se do užšího výběru dostaly tyto knihovny: Bootstrap, Semantic a UIKit. S Bootstrapem jako jediným už jsem se seznámil během studia, a ačkoli nebyl špatný, chtěl jsem vyzkoušet něco nového. Z trojice zmíněných knihoven mě nejvíce zaujal UIKit, který nabízel vše, co bylo k Meme generátoru potřeba, zejména responzivní design, mřížky a styly pro vše od nadpisů po tlačítka.

```
1 <div class="uk-grid-small uk-child-width-auto uk-text-center"
2   uk-grid>
```

Kód 5.6: Takto jednoduše lze pomocí UIKitu vytvořit responzivní mřížku

Použití je velice jednoduché, stačí přidat předdefinované třídy do HTML tagu *class*, případně přidat speciální atribut a vše funguje jak má. Bohatá dokumentace s mnoha příklady je velmi přínosná a byla radost ji projít[12]. Vše

[12]Dokumentace se nachází na `https://getuikit.com/docs/introduction`

co jsem potřeboval jsem v ní našel. Až na malé výjimky jsem takto dokázal nastylovat celou aplikaci a potřeboval k tomu minimum vlastních definic. Důkazem je, že žádný z SCSS souborů nemá více jak 15 řádek.

Výsledek záměrně vypadá jednoduše a neobsahuje nic zbytečného navíc. Stránka je responzivní až zhruba do šířky 700 pixelů, což je šířka plátna pro tvorbu memů. Bohužel použitý HTML5 Canvas nelze používat na mobilních zařízeních[13] a tak nemělo smysl pro ně stránku optimalizovat.

5.3 Integrace služeb AWS

Aby bylo možné využívat služby AWS, je nutné se nejprve registrovat. Registrace je rychlá a jednoduchá, jediný zádrhel spočívá v nastavení kreditní karty, ze které si Amazon strhne testovací částku aby si potvrdil že je platná. V budoucnu z této karty strhává měsíční poplatky za využívání služeb.

5.3.1 S3

Nejprve jsem se rozhodl zprovoznit hostování aplikace na S3. V nastavení jsem vytvořil dva buckety – jeden je určen pouze pro samotnou aplikaci a to proto, aby se usnadnilo nasazení a bucket bylo možno celý smazat a přehrát novou verzí aplikace. Druhý bucket slouží k ukládání hotových memů a šablon.

Zprovoznění bylo velmi jednoduché, stačilo nejprve sestavit produkční verzi aplikace (k tomu má Angular CLI příkaz `ng build --prod`) a nahrát ji na S3 pomocí webového prostředí. V nastavení bucketu je pak možnost *Static website hosting*, jejímž zaškrtnutím je hotovo. Aplikace je nyní veřejně přístupná na internetu, pouze má dlouhou a nepraktickou adresu[14].

Amazon má v současné době desítky datacenter a mohl jsem si vybrat, na kterém z nich chci buckety vytvořit. Ačkoli je několik datacenter v Evropě (zejména západní), rozhodl jsem se využít oblast US-EAST 1 v severní Virginii ve Spojených Státech, protože poplatky za používání S3 se v jednotlivých datacenterch liší a zmíněná lokalita byla jedna z nejlevnějších. S umístěním není problém, protože s využitím CloudFront bude obsah stejně automaticky rozdistribuovaný po více lokalitách.

5.3.2 DynamoDB

Také zprovoznění a nastavení databáze je velmi jednoduché. Amazon nabízí ke stáhnutí spustitelnou verzi pro lokální použití, na které jsem si mohl v průběhu vývoje testovat vytváření tabulek, přidávání a mazání záznamů a použití AWS SDK pro JavaScript resp. Node.js.

[13] Na všech dotykových zařízeních nelze posunovat text po plátně což Meme generátor dělá téměř nepoužitelným.

[14] `http://aws-website-memegenerator-jijv4.s3-website-us-east-1.amazonaws.com`

K vytvoření produkční verze stačilo navštívit webové rozhraní a pomocí jednoduchého UI vytvořit obě tabulky *Memes* i *Templates* představené ve schématu 4.4 a přiřadit jim primární a sekundární klíče.

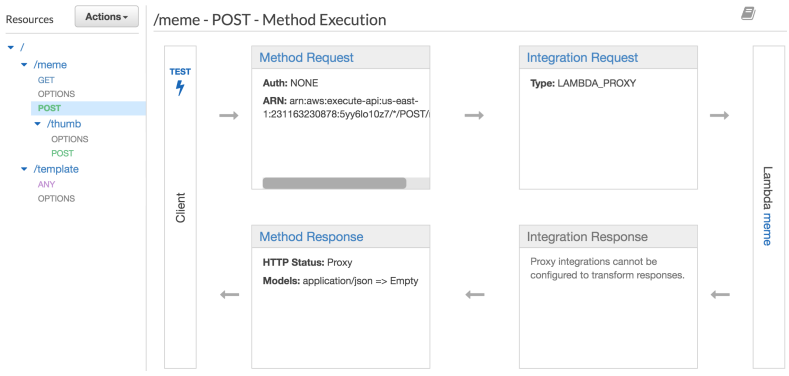
Tím by nastavení databáze mohlo skončit, ještě jsem ovšem vypnul funkci auto-scaling, která se stará o dynamické přiřazování kapacitních jednotek v závislosti na aktuální zátěži. V úrovni, ve které je použití DynamoDB zdarma, je k dispozici přibližně 15 kapacitních jednotek pro čtení a zápis a tato funkce by mohla alokovat více prostředků. Další jednotky by byly už účtovány podle standardního ceníku a tomu se snažím předejít, ačkoli by to mohlo zpomalit aplikaci při velkém počtu požadavků.

5.3.3 API Gateway a Lambda

Nyní přichází část, ve které dojde k propojení všech služeb do funkčního celku podle schématu 4.3, tedy že jednotlivé služby mezi sebou budou komunikovat a předávat si zpracování požadavků uživatelů.

Nastavení API Gateway není obtížné, jde-li o přenesení stávajícího API nabízí nástroje k snadnému přechodu. V mém případě jsem ovšem budoval API od začátku a tak jsem tuto možnost nezvolil. V uživatelském prostředí jsem postupně vytvořil všechny endpointy, které aplikace využije, tedy `/template` , `/meme` a `/meme/thumb`. Zatím ovšem nespouštějí žádné akce, neboť je nejprve nutné vytvořit lambda funkce, které budou požadavky zpracovávat.

Při vytváření funkce lze využít možnosti předpřipraveného kódu a nastavení podle některého z existujících scénářů. Nicméně pro můj případ neexistovala žádná vhodná kombinace jazyka, prostředí a služeb, které využívá, a vytvořil jsem tedy čistou novou funkci.



Obrázek 5.4: Propojení API Gateway a Lambdy

Zvláštní pozornost je třeba věnovat výběru uživatelské role, protože ovlivňuje bezpečnost aplikace. Já vytvořil speciální roli *lambdaUser* speciálně pro tyto funkce, abych mohl nastavit přístup jen k těm službám, které opravdu potřebují. Tato role má přístup k S3, do databáze DynamoDB a také má právo

k zapisování logů do CloudWatch. Jednotlivá práva lze ještě dále upřesňovat, např. na read-only pro S3 apod. a tím ještě více zamezit případnému zneužití.

Nyní lze již finálně nastavit API Gateway. Pro každý endpoint je přiřazena lambda, které API Gateway předá přijatý požadavek, počká na jeho zpracování a výsledek vrátí uživateli. V mé konfiguraci se tedy chová jako proxy, protože požadavky pouze přeposílá. Lambdy vždy přijímají požadavky GET i POST, je-li to třeba, je ale možné přiřadit lambdy na úrovni jednotlivých metod.

5.4 Nasazení

Při agilním způsobu vývoje je běžné nasazovat často i několikrát během týdne a tak jsem této oblasti věnoval více pozornosti. Při správném nastavení CI/CD lze rychle a efektivně vydávat nové verze, bugfixy a další aniž by aplikace měla větší downtime.

Nasazování řeším zatím jednoduše, pro vydání nové verze aplikace mažu obsah bucketu na S3 a následně ho nahradím novější verzí. To znamená až minutový výpadek a je také nemožné tuto akci zkoordinovat s nasazením lambda funkcí, došlo by-li k nějaké změně API.

V budoucnu by se hodilo vytvořit skript na automatické nahrávání aplikace a lambda funkcí. Jak S3 tak Lambda vystavují REST API, přes které lze soubory nahrát, případně lambdy lze mít uložené na S3 (třeba ve stejném bucketu jako aplikaci). Nejlepším řešením by bylo použít nějaký specializovaný software jako například Ansible.

5.4.1 Přesměrování

Pro produkční prostředí jsem použil doménu kterou vlastním, ačkoli to pro projekt nebylo potřebné. Adresa, na které je aplikace dostupná, se však pro praktické využití nehodí, neboť je příliš dlouhá a nelze si ji zapamatovat[15]. V této krátké sekci zrekapituluji, co vše bylo potřebné ke správnému spuštění aplikace na memegenerator.josefkriz.com.

Nastavení služby CloudFront je velmi jednoduché, dokonce při hostování stránek na S3 lze vše nastavit automaticky bez jediného potřebného zásahu. Toho jsem využil a tak vznikla nová vstupní brána do aplikace. CloudFront cachuje statická data a funguje i jako CDN, kdyby aplikace byla opravdu úspěšná a bylo ji potřeba rozmístit po více lokalitách.

Pro správné přesměrování na CloudFront jsem upravil DNS a přidal CNAME záznam, který slouží jako alias. Podobné nastavení bylo potřeba přidat i v administraci CloudFront. Posledním nutným krokem bylo všechny požadavky

[15] `http://aws-website-memegenerator-jijv4.s3-website-us-east-1.amazonaws.com` nebo `http://d1es5ig79kyymo.cloudfront.net`

nemířící do kořenového adresáře přesměrovat, protože jinak se uživatel domáhá přístupu do adresáře na S3, k čemuž není oprávněn a dostává odpověď 403 Forbidden. Také by nefungovalo routování v aplikaci, protože veškeré cesty jsou pouze virtuální. V nastavení CloudFront bylo potřeba vytvořit speciální pravidlo, které všechny 403 přesměruje na index.html v kořenovém adresáři a změní odpověď na 200 OK.

5.5 Testování

Testy jsou neodmyslitelnou součástí projektu, neboť zaručují funkčnost kódu a splnění funkčních požadavků. To bylo zohledněno i v zadání této práce, která obsahuje unit testy a také end–to–end testy, které prověří funkčnost aplikace jako celku. Součástí této kapitoly je i sekce o uživatelském testování, které probíhalo s betaverzí aplikace.

5.5.1 Unit testy

Angular obsahuje nativní nástroje pro tvorbu unit testů a používá-li se k vytváření komponent, services a dalších příkaz Angular CLI `ng generate`, jsou rovnou automaticky vygenerovány i soubory s veškerou přípravou ke psaní testů. Pro spuštění a hodnocení se používá frameworků Karma a Jasmine.

5.5.1.1 Testování services

Při testování services se zaměřuji na to, aby service správně odesílala požadavky na server a aby vrácená data správně propadávala nazpět. V každém testu je vytvořen takzvaný spy, který jako agent ukradne požadavek a vrátí nastrčenou odpověď. Tímto způsobem nedochází k volání opravdového serveru, který by musel být spuštěn, nebo v mém případě nedochází ke spotřebovávání prostředků v cloudu. Také si mohu nastrčit vhodná data, která obsahují dobře testovatelné informace.

```
1 describe ('Meme service: get memes', () => {
2   beforeEach(() => {
3     httpClientSpy = jasmine.createSpyObj('HttpClient', ['get']);
4     memeService = new MemeService(<any> httpClientSpy);
5   });
```

Kód 5.7: Příklad nasazení „špiona“ na *MemeService*, který vrací podvrhnuté odpovědi

Takto testuji GET i POST požadavky včetně chybových stavů, kdy například server vrátí odpověď 500. Testy stačí spustit jednou pomocí příkazu `ng test`. Karma následně sleduje veškeré změny a případně testy spouští znovu. Výsledky testů lze sledovat v reálném čase v okně prohlížeče díky Jasmine HTML Reporter.

5.5.1.2 Testování komponent

Testování komponent je o něco náročnější než testování services, protože komponenta obsahuje i sdruženou HTML šablonu. Testuje se tedy nejen správné volání a odpovědi funkcí, ale i odpovídající změny v DOMu.

Každá komponenta obsahuje několik takových testů. Testy jsou vybrány podle významu jednotlivých funkcionalit, protože nelze unit testy pokrýt úplně vše. Opět dochází k nahrazování funkcí falešnými, protože ty zde nejsou předmětem testování. Jedná se například o metody nadřazených komponent nebo services. V testech jsou připraveny falešná data, která tyto services vrací.

Příklad takového testu je stisk tlačítka, který by měl vyvolat zavolání funkce. Konkrétně kliknutím na šablonu by se měla stát pozadím pro okno editoru. V testu je nasazen špion na metodu pro výběr šablony, odchytí se HTML element s šablonou a zavolá se událost „click“. Poté se zkontroluje zda byl metodě předán správný parametr (v tomto případě se musí shodovat kliknutá šablona s tou, která byla parametrem).

5.5.2 End–to–end testování

E2E testování probíhá pomocí frameworku Protractor, který je určený přímo pro aplikace napsané v Angularu. Je také vyvíjen Googlem a při vytvoření aplikace pomocí Angular CLI již nainstaluje potřebné balíčky a vytvoří první testovací scénář.

Protractor využívá Selenium WebDriver který zpřístupní ovládání internetového prohlížeče automatizovaným testům. Výhodou tohoto frameworku je, že implicitně čeká na dokončení všech vnitřních procesů Angularu (jako třeba HTTP požadavky) a testovací scénář je prováděn na připravené stránce.

Tyto testy zkoušejí funkčnost aplikace jako celku a testovací scénář se dá srovnávat s opravdovým člověkem, který by scénář prošel. Při testu se otevře okno prohlížeče a tak lze v reálném čase sledovat, jak test probíhá. Pro přehlednost se i zde používá framework Jasmine.

Testy pokrývají běžné akce jako přechod na další stránku nebo správné zobrazení seznamu memů. Bohužel aktuální verze Angularu stále na rozdíl od AngularJS neobsahuje nativní podporu falešných services[16] a tak jsem nepokryl testy funkcionality zapisující trvale objekty do databáze (hodnocení palci a uložení memu). Každé spuštění testů by totiž vytvořilo nové memy, které momentálně nelze odstranit jinak než vymazáním z databáze. Taktéž testovací prostředí zatím není dostupné.

Testy jsou rozděleny podle stránek, které obsluhují a vždy se skládají ze dvou souborů. První obsahuje testovací scénáře pro danou stránku, přičemž manipuluje s objekty ze stránky. Ty jsou importovány z druhého souboru, kterému se přezdívá page object a ten obsahuje selektory pro všechny prvky potřebné ve scénáři.

[16] `https://github.com/angular/protractor/issues/3092`

5.5.3 Uživatelské testování

Testování uživateli probíhá od nasazení aplikace na produkční prostředí. Jedná se přibližně o deset lidí z mého okolí, kteří aplikaci používají ke generování memů a dříve používali jeden z generátorů zmíněných v analýze. Již během prvních hodin testování mi bylo několikrát potvrzeno, že dokáže tuto konkurenci nahradit.

Cílem uživatelského testování bylo prověřit, zda aplikace splňuje funkční požadavky a je použitelná pro koncového uživatele. Také se můžou odchytit chyby, které jsem během implementace přehlédnul. Testeři se mohou vyjádřit k uživatelskému prostředí, zda jim připadá intuitivní a přehledné.

Po přibližně dvoudenním průběžném testování jsem zapojeným lidem rozeslal dotazník ohledně dosavadních poznatků z používání generátoru[17]. Byly v něm čtyři otevřené otázky na dosavadní dojem z aplikace, jako například zda člověk nenarazil na něco nelogického nebo jestli ho naopak na aplikaci něco zajímavého zaujalo a líbilo se. V druhé části dotazníku byly otázky s výběrem odpovědí na důvěryhodnost a srozumitelnost aplikace a zda se v ní snadno orientuje.

Zpětná vazba byla veskrze pozitivní. Jediná otázka, se kterou třetina dotázaných nesouhlasila, se týkala srozumitelnosti ovládacích prvků (tlačítek) na stránce. Na základě těchto výsledků jsem aplikaci prošel a případné nejasné navigační prvky upravil.

Největší a také jediná opravdová chyba, kterou tyto testy odhalily, spočívala v možnosti ohodnotit palcem jeden mem vícekrát, ač by to mělo být zakázané. Jak jsem později odhalil, označení memu jako ohodnoceného se provádělo příliš pozdě – v okamžiku, kdy se vrátil požadavek ze serveru – což mohlo trvat i několik sekund a po tuto dobu bylo možné každým kliknutím měnit hodnocení. V aktuální verzi je toto tedy již opraveno.

Souhrnně uživatelské testování dopadlo výborně, ač na malém vzorku lidí (cca 5–10) a prokázalo, že aplikace obstojí i v ostrém provozu. Pomohlo odhalit jednu větší chybu a zjistit reakce testerů na funkce a design aplikace, které byly až na výjimky pozitivní.

5.6 Budoucnost aplikace

Tento projekt vznikl na podnět reálných lidí a proto má reálnou naději úspěšné pokračovat i po obhájení této práce. Vývoj aplikace se nemusí zastavit, ať už bych se nadále aktivně podílel na projektu já nebo někdo z lidí, kdo memy generuje a rád by se podílel na zlepšování této aplikace.

Bude-li to možné, projekt by nadále pokračoval jako open-source s veřejně přístupným kódem na BitBucket, kde si kdokoli bude moci vytvořit pull request[18]. Takto by kolem generátoru postupně mohla vzniknout komunita,

[17] Dotazník je dostupný na `https://goo.gl/forms/4EzGenKqIjZhWlq23`

[18] Neboli žádost o zhodnocení provedných změn v kódu.

která by se o aplikaci starala a udržovala ji aktuální.

Infrastruktura, na které v generátor běží, je účelově připravená jak na další rozšiřování aplikace, tak na vysokou zátěž. Závislost na cloudových službách znamená libovolné škálování prostředků a tak ať už stránku navštěvují jednotky nebo miliony lidí denně, není třeba kvůli tomu cokoli měnit.

Je se však třeba připravit, že případný vyšší zájem o aplikaci by znamenal vyšší náklady na provoz a to přímou úměrou. V tom případě by bylo třeba shánět nějaký druh financování, ať už komunitou (příspěvky) nebo zobrazením reklam.

Velké rezervy má ještě aplikace v chybějících funkcích. Zmíním jen například možnost pohodlně přidávat a upravovat šablony skrze UI nebo atraktivnější hlavní stránku s lepším zobrazením vytvořených memů. Mezery by se našly i po grafické stránce.

V každém případě aplikace je plně připravena na další rozvoj a neobsahuje žádné bariéry, které by jí v tom bránily. Infrastruktura je plně schopna vydržet velký provoz a uživatelské rozhraní lze snadno rozšiřovat o další komponenty.

Závěr

V práci jsem se zabýval analýzou, návrhem a implementací webové aplikace umožňující tvorbu a hodnocení internetových obrázkových memů.

Aplikace umožňuje uživateli přidat libovolný text na vlastní obrázek či na obrázek z vlastní galerie aplikace. Lze měnit různé vlastnosti textu a tak si mem přizpůsobit vlastním představám. Výsledek si lze stáhnout k sobě či ho uložit do aplikace, čímž ho lze sdílet, hodnotit a objeví se na hlavní stránce aplikace.

Aplikace použila moderní způsoby vytváření webových aplikací a je koncipována jako SPA a napsána ve frameworku Angular. Backend aplikace je inovativně plně provozován v cloudu.

Aplikace je otestována, nasazena v produkčním prostředí a prověřena ostrým provozem s prvními uživateli z mého okolí. Do budoucna může kdokoli aplikaci dále rozšiřovat.

Literatura

[1] GIL, Paul. *What is a Meme?: The more you know about memes, the cooler you are* [online]. In: . [cit. 2018-05-09]. Dostupné z: https://www.lifewire.com/what-is-a-meme-2483702

[2] DAWKINS, Richard. *The selfish gene*. 2. vydání. New York: Oxford University Press, 1989. ISBN 01-928-6092-5

[3] *Memes* [online]. [cit. 2018-05-10]. Dostupné z: http://knowyourmeme.com/memes/memes

[4] REUTERS. *Trump's 'covfefe' tweet leaves Internet guessing* [online]. [cit. 2018-05-10]. Dostupné z: https://www.reuters.com/article/us-usa-trump-tweet/trumps-covfefe-tweet-leaves-internet-guessing-idUSKBN18R24E

[5] Trendy Google: *meme* [online]. [cit. 2018-05-06]. Dostupné z: https://trends.google.com/trends/explore?date=all&q=meme

[6] SOLOMON, David. *12 Best Online Meme Generator Websites in 2018* [online]. [cit. 2018-05-06]. Dostupné z: https://thedavidweb.com/best-online-meme-generators/

[7] Alexa Traffic Rank: *How popular is imgur.com?* [online]. [cit. 2018-05-06]. Dostupné z: https://www.alexa.com/siteinfo/imgur.com

[8] FOWLER, Martin. *GUI Architectures* [online]. [cit. 2018-05-06]. Dostupné z: https://martinfowler.com/eaaDev/uiArchs.html

[9] MELNIK, Ian. *Single Page Application (SPA) vs Multi Page Application (MPA): Pros and Cons* [online]. [cit. 2018-05-06]. Dostupné z: http://merehead.com/blog/single-page-application-vs-multi-page-application/

[10] PUKHA, Daryna. *Single-Page Apps vs Multiple-Page Web Apps: What to Choose for Web Development* [online]. [cit. 2018-05-11]. Dostupné z: https://yalantis.com/blog/single-page-apps-vs-multiple-page-apps/

[11] *Web framework rankings* [online]. [cit. 2018-05-06]. Dostupné z: http://hotframeworks.com/

[12] KOROTYA, Eugeniya. *5 Best JavaScript Frameworks in 2017* [online]. [cit. 2018-05-06]. Dostupné z: https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282

[13] YUSOV, Kirill. *Top 10 Best JavaScript Frameworks List* [online]. [cit. 2018-05-06]. Dostupné z: https://jelvix.com/blog/top-10-best-javascript-frameworks-list-in-2017

[14] MUGNAINI, Luca. *SPA and SEO: Google (Googlebot) properly renders Single Page Application and execute Ajax calls* [online]. [cit. 2018-05-06]. Dostupné z: https://medium.com/@l.mugnaini/spa-and-seo-is-googlebot-able-to-render-a-single-page-application-1f74e706ab11

[15] CARVALHO, Rafael. *Single Page Applications: When and Why You Should Use Them* [online]. [cit. 2018-05-06]. Dostupné z: https://www.scalablepath.com/blog/single-page-applications/

[16] PETROSYAN, Michael. *Angular 5 vs. React vs. Vue* [online]. [cit. 2018-05-06]. Dostupné z: https://itnext.io/angular-5-vs-react-vs-vue-6b976a3f9172

[17] NEUHAUS, Jens. *Angular vs. React vs. Vue: A 2017 comparison* [online]. [cit. 2018-05-06]. Dostupné z: https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176

[18] QIAN, Tim. *Star history: facebook/react vs vuejs/vue vs angular/angular* [online]. [cit. 2018-05-08]. Dostupné z: http://www.timqian.com/star-history/#facebook/react&angular/angular&vuejs/vue

[19] KRAUSE, Stefan. *Results for js web frameworks benchmark – round 6* [online]. [cit. 2018-05-08]. Dostupné z: http://www.stefankrause.net/js-frameworks-benchmark6/webdriver-ts-results/table.html

[20] *Vue.js - Comparison with Other Frameworks* [online]. [cit. 2018-05-08]. Dostupné z: https://vuejs.org/v2/guide/comparison.html#Size

[21] MINAR, Igor. *MVC vs MVVM vs MVP.* Google+ [online]. [cit. 2018-05-08]. Dostupné z: https://plus.google.com/+AngularJS/posts/aZNVhj355G2

[22] *Node by Numbers: 2017 Edition* [online]. In: . [cit. 2018-05-11]. Dostupné z: https://nodesource.com/node-by-numbers

[23] VŠETEČKA, Roman. *V Rusku se bojuje o komunikační síť Telegram. Odnáší to Google i Amazon.* In: Idnes.cz [online]. [cit. 2018-05-11]. Dostupné z: https://technet.idnes.cz/telegram-a-internet-v-rusku-omezen-dte-/sw_internet.aspx?c=A180417_211918_tec-kratke-zpravy_vse

[24] ROBERTS, Mike. *Serverless Architectures* [online]. [cit. 2018-05-11]. Dostupné z: https://martinfowler.com/articles/serverless.html

[25] *Globální infrastruktura Azure.* Microsoft Azure [online]. [cit. 2018-05-09]. Dostupné z: https://azure.microsoft.com/cs-cz/global-infrastructure/

[26] *Introduction to Azure Storage.* Microsoft Azure [online]. [cit. 2018-05-09]. Dostupné z: https://docs.microsoft.com/en-us/azure/storage/common/storage-introduction

[27] *Introduction to Azure Cosmos DB.* Microsoft Azure [online]. [cit. 2018-05-09]. Dostupné z: https://docs.microsoft.com/en-us/azure/cosmos-db/introduction

[28] *An introduction to Azure Functions.* Microsoft Azure [online]. [cit. 2018-05-09]. Dostupné z: https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview

[29] FINNEGAN, Matthew a Scott CAREY. *AWS vs Azure vs Google: What's the best cloud platform for the enterprise?* [online]. [cit. 2018-05-09]. Dostupné z: https://www.computerworlduk.com/it-vendors/microsoft-azure-vs-amazon-aws-public-cloud-comparison-which-cloud-is-best-for-enterprise-3624848/

[30] *About AWS* [online]. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/about-aws/

[31] *AWS Free Tier* [online]. AMAZON. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/free/

[32] *Google Cloud Platform Free Tier* [online]. GOOGLE. [cit. 2018-05-09]. Dostupné z: https://cloud.google.com/free/

[33] *Vytvořte si bezplatný účet Azure ještě dnes* [online]. MICROSOFT. [cit. 2018-05-09]. Dostupné z: https://azure.microsoft.com/cs-cz/free/

[34] *Amazon S3* [online]. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/s3/

[35] *Amazon S3 Pricing* [online]. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/s3/pricing/

[36] *Amazon DynamoDB* [online]. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/dynamodb/

[37] *Amazon DynamoDB Pricing* [online]. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/dynamodb/pricing/

[38] *AWS Lambda* [online]. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/lambda/

[39] *AWS Lambda Pricing* [online]. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/lambda/pricing/

[40] *Amazon API Gateway Pricing* [online]. [cit. 2018-05-09]. Dostupné z: https://aws.amazon.com/api-gateway/pricing/

[41] BRODY, Alon. *SQL vs NoSQL: The Differences Explained* [online]. In: . [cit. 2018-05-09]. Dostupné z: https://blog.panoply.io/sql-or-nosql-that-is-the-question

[42] OBŮRKA, Robin a Petr PULC. Přednášky předmětu BI-GIT, Zimní semestr 2017/2018: *Git, the information manager from hell.* In: EDUX FIT ČVUT [online]. 11. prosince 2017 [cit. 2018-05-11]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-GIT/_media/lectures/gitzs2017.pdf

[43] *Angular - QuickStart* [online]. In: . [cit. 2018-05-11]. Dostupné z: https://v5.angular.io/guide/quickstart

Příloha **A**

Seznam použitých zkratek

AJAX Asynchronous JavaScript And XML

API Application Programming Interface

AWS Amazon Web Services

BaaS Backend as a Service

CD Compact Disc

CD Continuous Delivery

CDN Content Delivery Network

CI Continuous Integration

CLI Command-line Interface

CORS Cross-origin resource sharing

CSS Cascading Style Sheets

DB Database

DNA Deoxyribonukleová kyselina

DNS Domain Name System

DOM Document Object Model

E2E End-to-end

EC2 Elastic Compute Cloud

ECMA European Computer Manufacturers Association

FaaS Fucntion as a Service

GCP Google Cloud Platform

GDPR General Data Protection Regulation

GIMP GNU Image Manipulation Program

HTML Hypertext Markup Language

IDE Integrated development environment

JS JavaScript

JSON JavaScript Object Notation

JSX JavaScript XML

JVM Java Virtual Machine

MPA Multi Page Application

MVC Model-View-Controller

MVVM Model-View-Viewmodel

MVW Model-View-Whatever

PHP Hypertext Preprocessor

REST Representational State Transfer

S3 Simple Storage Service

SDK Software development kit

SOAP Simple Object Access Protocol

SPA Single Page Application

SQL Structured Query Language

SVN Apache Subversion

UI User Interface

URL Uniform Resource Locator

USA United States of America

VCS Version Control System

WWW World Wide Web

XML Extensible Markup Language

Příloha B

Návod k instalaci a spuštění

Ke stažení repozitáře a spuštění aplikace je zapotřeba mít nainstalované programy git[19] a yarn[20].

1. Naklonujte git repozitář
   `git clone https://josefkriz@bitbucket.org/josefkriz/meme-generator.git`
2. Vstupte do adresáře projektu
   `cd meme-generator`
3. Nainstalujte potřebné balíčky
   `yarn`
4. Spusťte aplikaci
   `yarn start`
5. Spusťte unit testy
   `yarn test`
6. Spusťte e2e testy
   `yarn e2e`

[19] https://git-scm.com/
[20] https://yarnpkg.com/lang/en/

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `app` ..... adresář se zdrojovým kódem aplikace
  - `dist` ..... sestavená aplikace
  - `e2e` ..... e2e testy
  - `lambda` ..... lambda funkce
- `thesis` ..... adresář se zdrojovým kódem a přílohami práce
- `BP_Kriz_Josef_2018.pdf` ..... text práce