Gymnázium Jana Keplera

obor vzdělání:

7941K41 Čtyřleté gymnázium pro absolventy ZŠ

Maturitní práce z informatiky

Sociální síť - Same

autor: Vít Paulík

vedoucí práce: Karel Jílek

Praha 2018

*Prohlašuji, že jsem jediným autorem této maturitní práce a všechny citace, použitá literatura a další zdroje jsou v práci uvedené.*

*Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium Jana Keplera, Praha 6, Parléřova 2 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu*

**Anotace:**

Maturitní práce má podobu sociální sítě s názvem “Same”. Jedná se o webovou aplikaci, na které se mohou vyjadřovat registrovaní uživatelé pomocí textových a obrázkových příspěvků, které se vkládají na jejich vlastní stránku. Uživatelé se také mohou navzájem sledovat, čímž se nové příspěvky sledovaných sledujícím zobrazí na “news feedu” a na příspěvky reagovat pomocí tzv. “same” (cítím se stejně).

**Abstract:**

The graduation project is a social network called “Same”. It’s a web application, where users can express themselves using text and image posts, which are posted to their own user page. Users can also follow each other, and watch posts of their followed users on the “news feed”, and react to posts with “same” (meaning “I feel same”).

**Obsah:**

[**Instalace**](#_trt803kdjjvk) **4**

[**Úvod**](#_4lpvdsanrua0) **5**

[**Hlavní část**](#_7e9tr7cssxo) **6**

[Instalace](#_dmzyqg3smn7a) 6

[Funkce a design](#_z9vv9us6f4rl) 6

[Technologie](#_mx0hkst62n5o) 15

[Databáze](#_rkxcooc8cljc) 16

[Frontend](#_3xtxyj2i1rtb) 18

[Backend](#_z4q7o5rjnanx) 20

[Komunikace mezi serverem a frontendem](#_wjycz4rdarms) 21

[**Závěr**](#_o9kdcycvd88i) **22**

[**Zdroje**](#_mky2tc80b29w) **23**

[Technologie](#_zgxwjhujd9m) 23

[Využitý kód](#_kxeejyqq4j3p) 23

# Instalace

1. Naklonování repozitáře z githubu:  
 <https://github.com/fadexmusic/sa.me>

2. Instalace NodeJS

<https://nodejs.org/en/>

3. Instalace Angular

<https://angular.io/guide/quickstart>

4. Instalance MongoDB

<https://www.mongodb.com/download-center?jmp=nav#community>

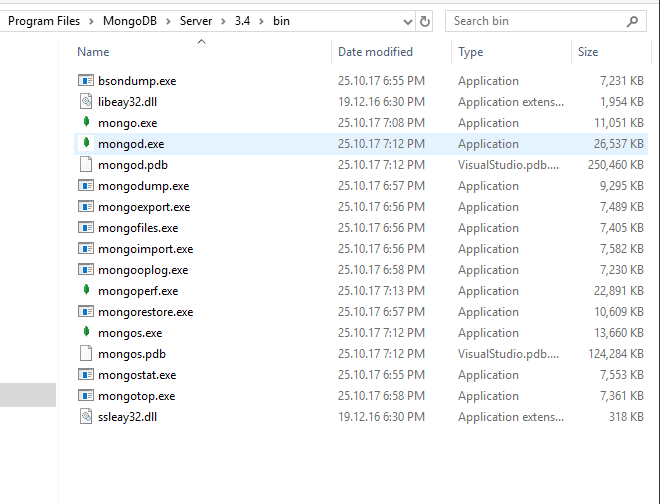
5. instalace modulů frontendu (ve složce same-fe)



6. instalace modulů backendu



7. Spuštění databáze (mongod.exe v místě instalace)



8. spuštění frontendu (ve složce same-fe)



9. spuštění backandu (ve složce same-be)



# 

# 

# Úvod

Rozhodl jsem se, že jako svůj maturitní projekt vytvořím sociální síť, jejíž koncept jsem už měl v hlavě nějakou dobu. Nápad je založen na podobných existujících sociálních síťích - například si bere hodně inspirace z instagramu a jeho jednoduchosti. Každý uživatel má svou vlastní stránku, kam může přidávat textové a obrázkové příspěvky. Ostatní uživatelé mohou sledovat stránky jednotlivých uživatelů, čímž se jim nové příspěvky všech sledovaných uživatelů zobrazí na “news feedu”. Projekt by se také dal přirovnat k Twitteru, ale je Same je o něco jednodušší, což ubírá na náročnosti pro uživatele, v čemž vidím výhodu. Hlavní myšlenka je založená na reakcích na příspěvky. Místo klasického “liku” uživatel reaguje pomocí “same”, kterým dává najevo, že se cítí stejně/smýšlí stejně jako to vyjádřil autor v příspěvku. V dnešní době jsou všude po internetu populární právě různé obrázky/texty, se kterými se člověk může ztotožnit, a přesně na tom je stránka postavena. Autor může sledovat, kolik lidí se ztotožňuje s jeho názory, problémy, vtipy atd.. Jako úkol jsem tedy před sebou měl vypracovat sociální síť a to od front-endu, po back-end.

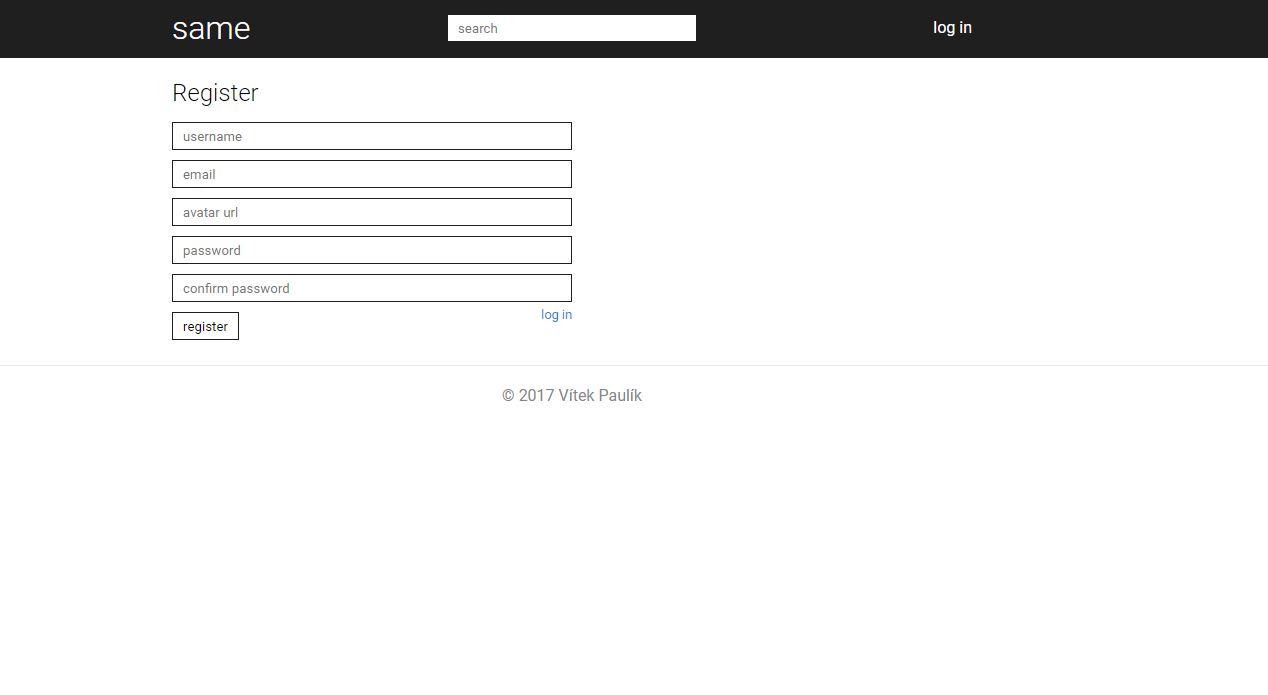
V první řadě jsem tedy měl nejdříve promyslet všechny funkce, na základě kterých jsem vybral všechny technologie, dále udělat design, následně vytvořit design a schémata databáze a komunikaci se serverem a frontendem, a na konec vše převést do kódu.

# Hlavní část

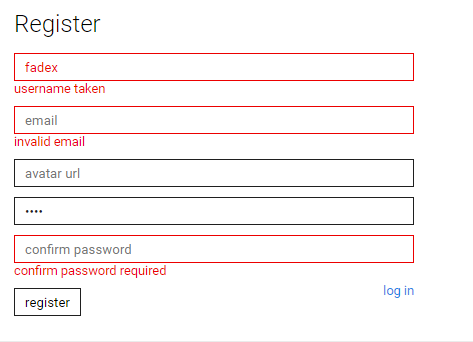
## Funkce a design

Nejdříve jsem si musel definovat funkce, které od sociální sítě chci a jak je zpracuji. Rozhodl jsem se udělat minimalsitický design, který uživatele zaměřuje na obsah.

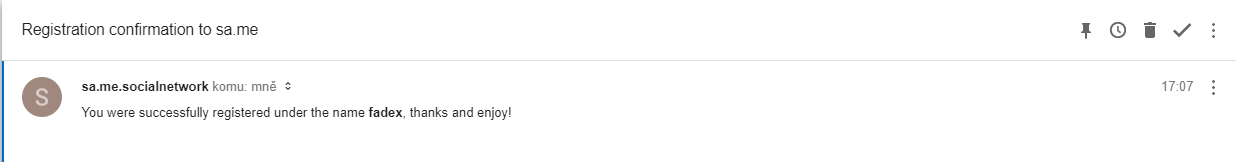
První skupina funkcí je určitá práce s uživatelskými účty. Uživatel pro registraci účtu vyplní formulář, do kterého vyplní základní informace.



Všechny formuláře v aplikaci jsou rekativní a upozorňují uživatele na jakékoliv chyby (email/uživatelské jméno je zabrané, políčko je povinné, hesla se neshodují atd.)



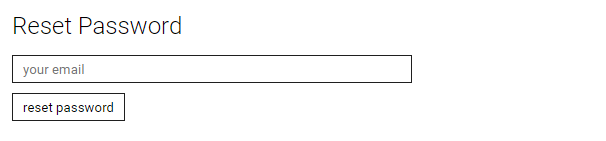
Když se uživatel registruje, přijde mu email, který registraci potvrzuje.

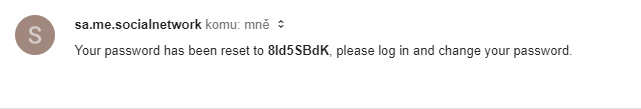


Dále je přístupný přihlašovací formulář.

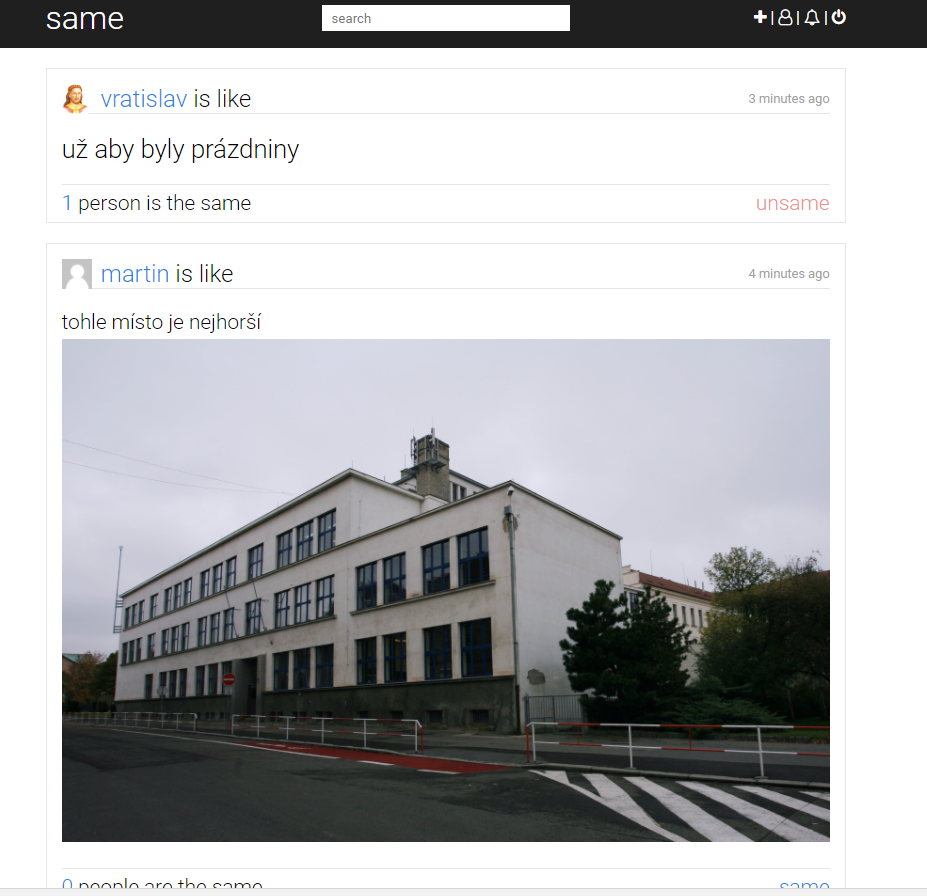
## 

Když uživatel zapomene heslo, může zadat svůj email a přijde mu nové vygenerované heslo.

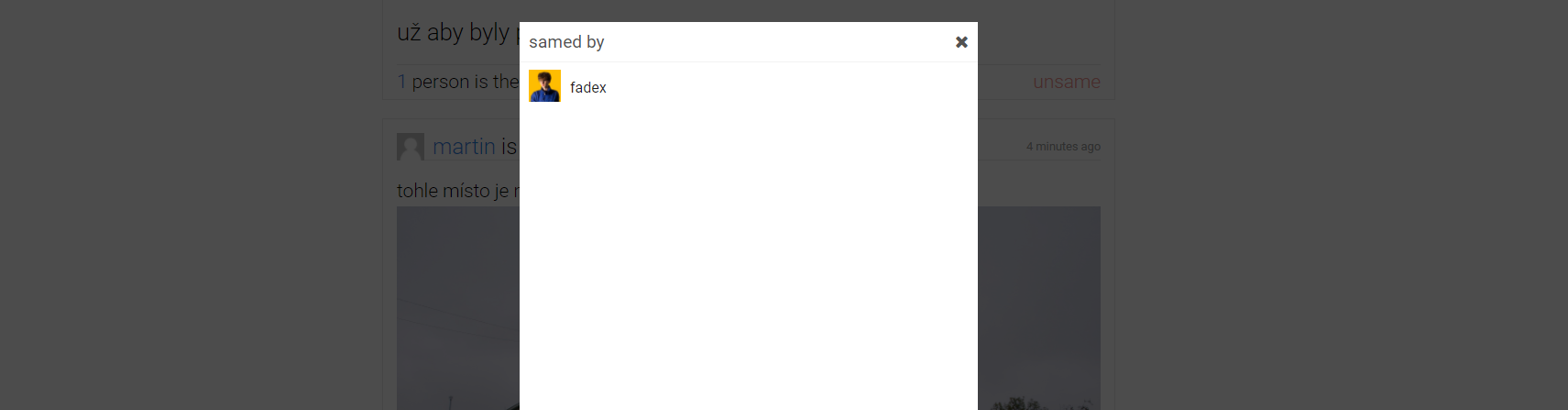




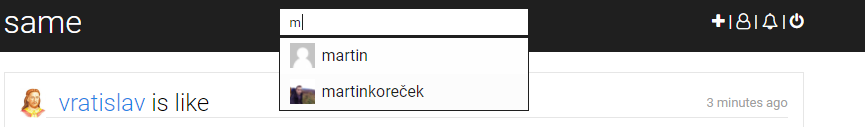
Po přihlášení se uživateli zobrazí “news feed”, kde vidí příspěvky lidí, které sleduje.



Na příspěvky může reagovat a prohlížet si kdo na ně reagoval.

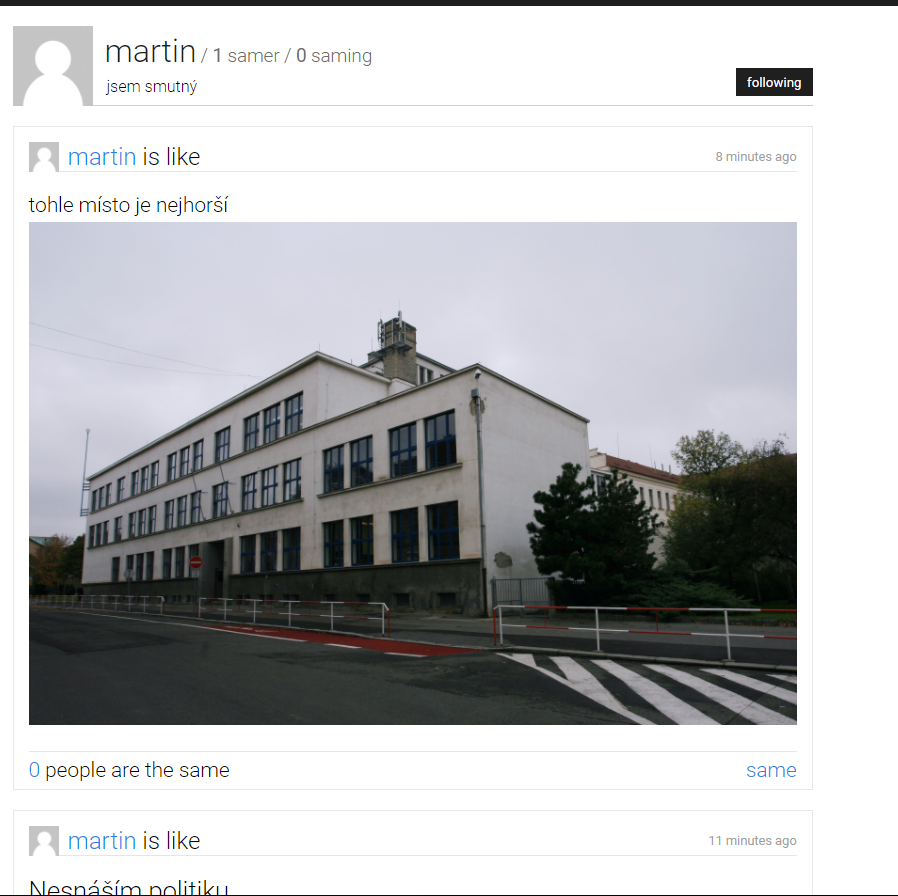


Uživatel si může vyhledávat mezi existujícími uživateli.

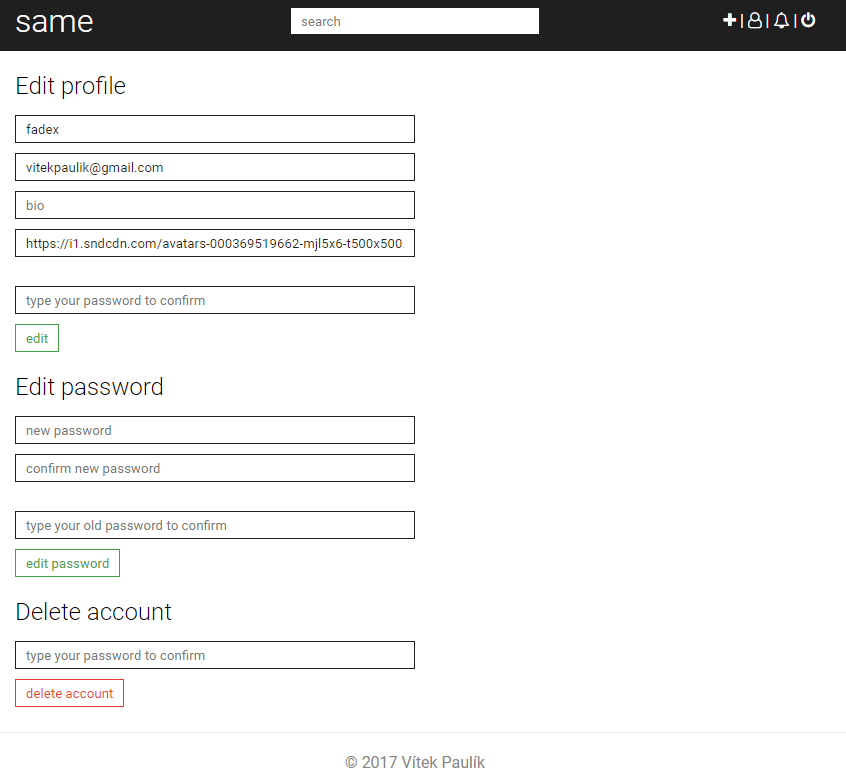


Na stránce uživatele si může prohlížet příspěvky uživatele,

kdo uživatele sleduje, koho sleduje.

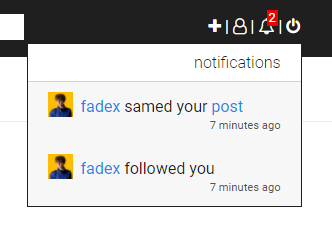


Když uživatel zobrazuje svůj účet, má možnost ho upravovat pomocí formuláře (měnit informace, heslo, smazat účet). Všechny změny musí být potvrzeny stávajícím heslem.



Přihlášenému uživateli se zobrazuje v pravém horním rohu menu, ve kterém může navigovat na svou stránku, zobrazit notifikace, odhlásit se a přidat nový příspěvek.

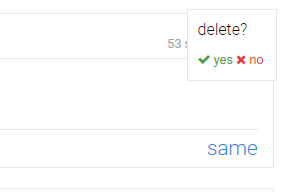
notifikace



přidávání příspěvku.



Uživatel také může mazat své přípěvky.



Všechny akce na stránce mají také vizuální odezvu - když uživatel přidá příspěvek, přihlásí se, tak se v pravém dolním rohu zobrazí

popup.

## 

## 

## Technologie

Jelikož se jedná o sociální síť, výběr použitých technologií byl založen na tom, že forma aplikace je webová.

Vzhledem k tomu, že se ve frontendu pohybuji velice dobře, tak jsem měl v technologii jasno. Použil jsem systém vytvoření pro one-page aplikace *Angular 5*. Tuto technologii jsem zvolil, protože je vyvynuta pro dynamické webové aplikace, reaktivní formuláře a velmi dobře se v ní implementuje právě komunikace se serverem.

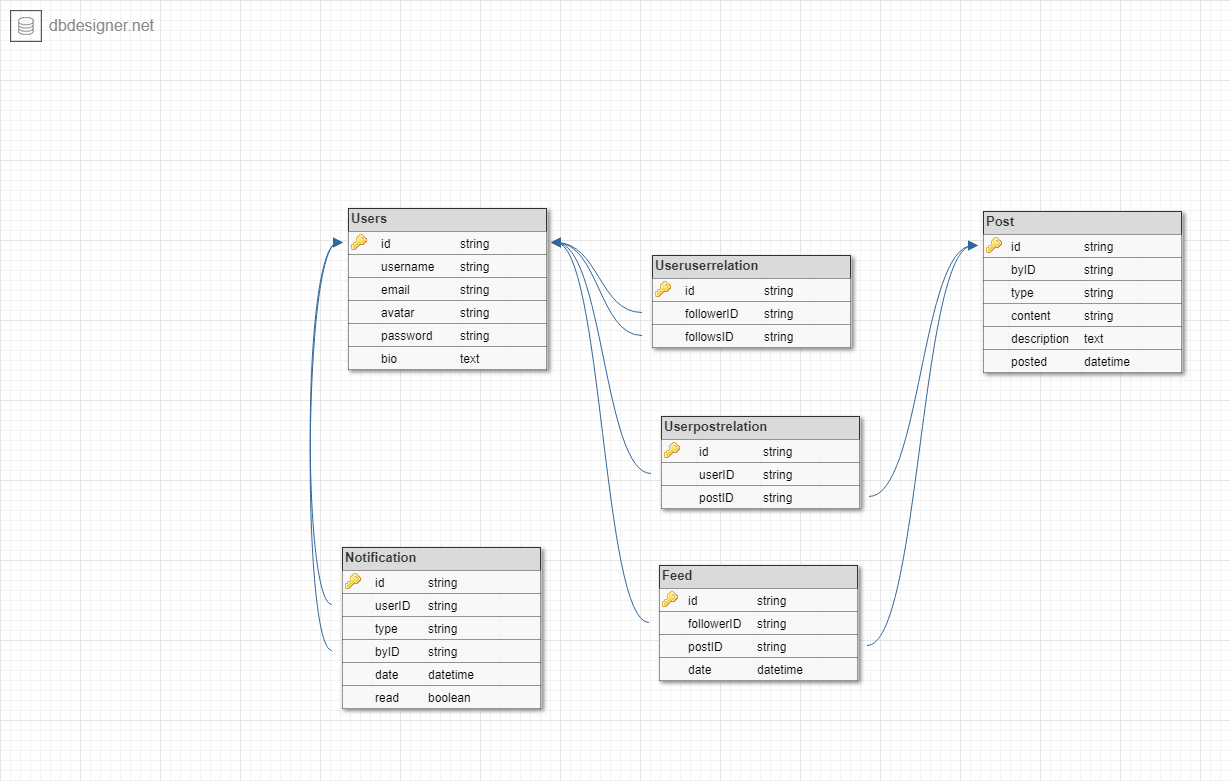
Další na řadě byl výběr systému, který zařídí komunikaci mezi frontendem a serverem a zvolil jsem klasický REST API, který je založený na jednorázových http requestech GET, POST, PUT, DELETE. Tento systém je vhodný, jelikož na stránce nejsou žádné “kontinuální” requesty, ke kterým bych použil například websockety a navíc velmi dobře funguje s *Angularem*.

Vzhledem k tomu, že jsem na back-endu nikdy moc nepracoval, zvolil jsem pro něj front endu blízký node js, v kombinaci s express js, které jsou založeny na frontendovém jazyku Javascriptu. Express je navíc vytvořený právě pro REST API, které jsem zvolil pro komunikaci mezi serverem a frontendem. Dále jsem ze serverových technologií v podobě pluginů do node js použil *nodemailer*, který zařizuje posílání emailů, *bcrypt*, který je využit pro enkryptování hesel a následné přihlašování. Následuje JWT, využitý pro generování json web tokenů, pomocí kterých zařizuji bezpečnou komunikaci mezi přihlášeným uživatelem a serverem. Dále cors, který řeší problém s cross-origin hlavičkami a v neposlední řadě async, který zjednodušuje asynchronní programování v nodu.

Jako poslední jsem musel zvolit databázovou technologii a rozhodl jsem se pro MongoDB, které je opět přívětivé pro lidi, co nemají takovou zkušenost s backendovým programováním. K pracování s MongoDB jsem ještě použil nodový plugin mongoose, který zjednoduššuje organizaci dokumentů v databázi, pomocí schémat (které známe například z SQL).

Když jsem měl mezi technologiemi tak nějak jasno, mohl jsem se vrhnout na design databáze.

## Databáze

Design databáze nebyl nijak náročný a nakonec jsem potřeboval 6 tabulek.   


První tabulka “Users” v sobě uchovává informace o uživatelech.

Každý uživatel má unikátní identifikační číslo, uživatelské jméno, email, profilový obrázek, heslo a bio (krátký text, který uživatel může o sobě napsat).

Hned vedle uživatele je tabulka Useruserrelation, která propojuje dvě uživatelské řádky. Dřží v sobě identifikační číslo sledujícího a sledovaného.

Další důležitá tabulka je tabulka “Post”, která v sobě dřží informace o příspěvkách. Má opět unikátní identifikační číslo, typ (který určuje, jestli se jedná o obrázek, nebo text), content (buďto odkaz na obrázek, nebo samotný text), description (popis, který může, nebo nemusí uživatel napsat k obrázku) a datum, kdy byl příspěvek přidán.

Další tabulka váže uživatele a příspěvek. Když uživatel vytvoří příspěvek, vytvoří se řádek v tabulce Userpostrelation, který nese identifikační číslo uživatele a příspěvuk.

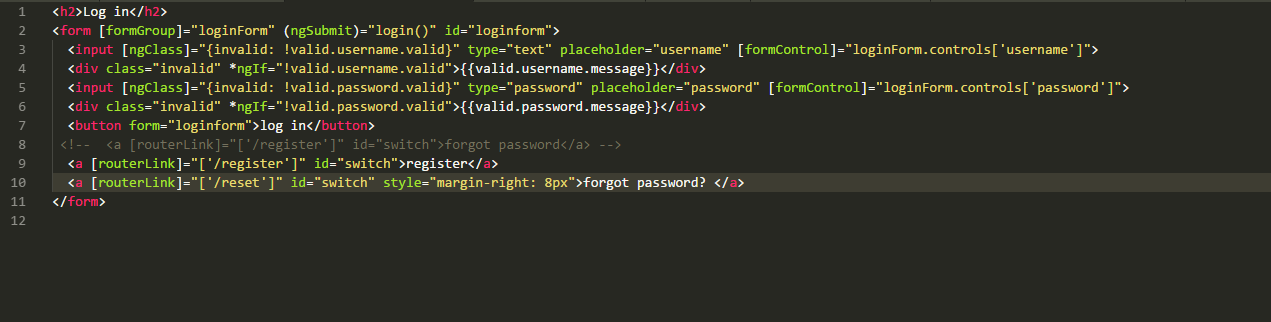
Dále je v databázi tabulka “Feed”, která v sobě drží identifikační číslo sledujícího, identifikační číslo příspěvku a datum vytvoření feedu. Tato tabulka je využita pro vytváření feedů z příspěvků sledovaných uživatelů pro sledující. U této tabulky jsem narazil na dilema, jak se vlastně samotný feed bude skládat. Zjistil jsem, že jsou dvě zažité metody - tzv. “push” a “pull”. Jde o to, kdy se má feed skládat. V metodě push, kterou jsem nakonec zvolil, se vytváří řádky do tabulky “Feed” pokaždé, když uživatel vytvoří příspěvek. Vytvoří se zvláštní řádek pro každého sledujícího uživatele. U metody pull se feed vytváří pokaždé, když ho sledující načte, tak, že se vytáhnou všechny řádky z tabulky Userpostrelation uživatelů, které sleduje, srovnají se podle data a následně se podle identifikačních čísel vezmou samotné příspěvky a vykreslí se na stránku. Metoda pull mi přišla mnohem náročnější jak na server, tak časově na generování stránky feedu, proto jsem se nakonec rozhodl pro metodu push, u které nevadí, že vytváření feed tabulek trvá déle, protože na ně nikdo přímo nečeká.

Poslední tabulka je tabulka “Notifications”, která je využita pro upozorňování uživatelů na určité akce (když je někdo začne sledovat, nebo jim dá “same” na příspěvek). Drží v sobě tedy identifikační číslo uživatele, pro kterého je určena, typ (jestli se jedná o sledování, nebo same), dále identifikační číslo uživatele, kterým byla vyvolána (sledujícím, nebo reagujícím), datum, kdy byla vytvořena a boolean, který zobrazuje zda byla uživatelem zobrazena, nebo ne.

## Frontend

Frontend je postavený na frameworku Angular 5. Vzhledem k tomu, že jde o systém postavený pro single-page aplikace, hlavní stavební buňkou je tzv router. Pokaždé, když kliknu na nějaký odkaz, router mě přesměruje na daný modul (login, feed, user..).

Samotný layout je napsaný v html.

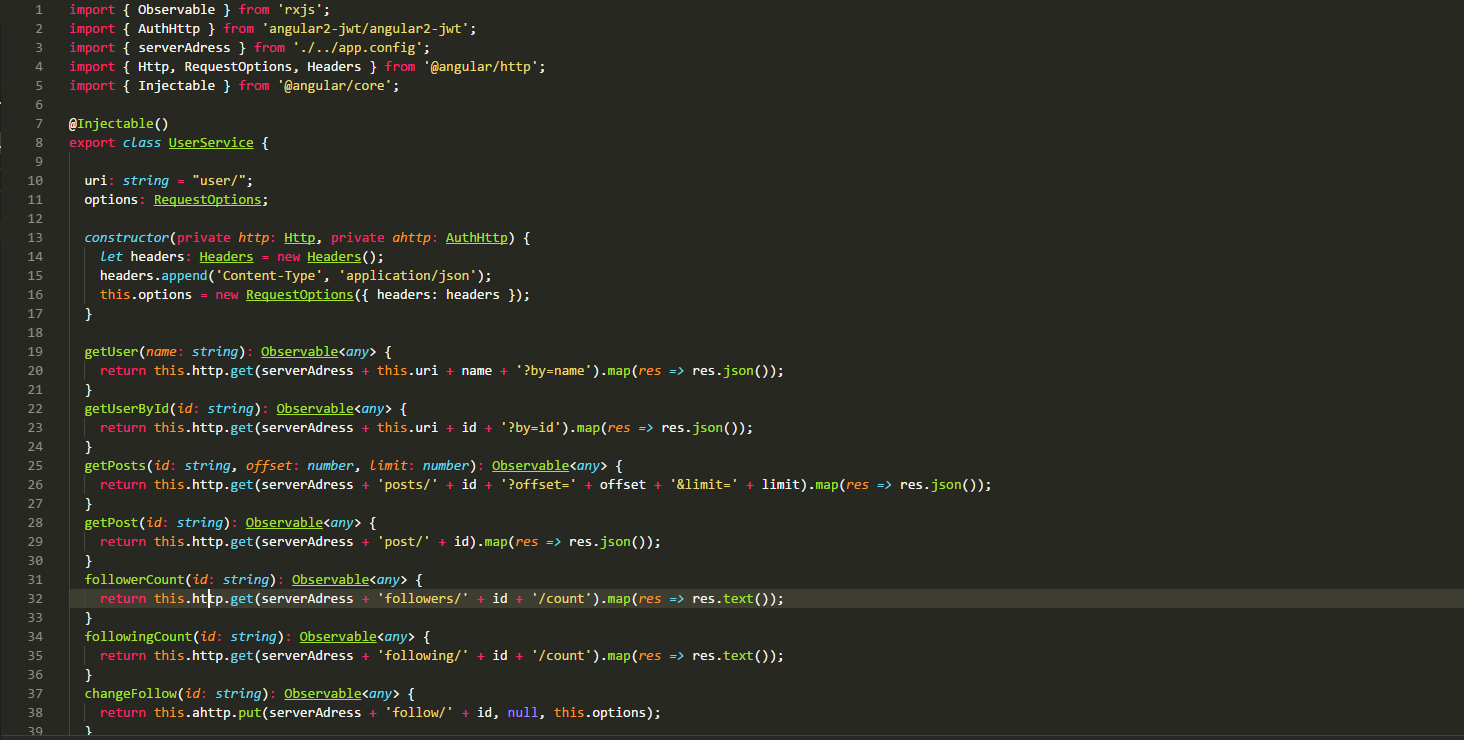


a jeho nastylování pomocí SCSS.



Následné requesty jsou vytvořeny pomocí tzv. servisů, které v sobě drží jejich logiku a následné tázání se serveru.

příklad servisu na uživatelskou stránku.



Logika modulů se nepíše v čistém *Javascriptu*, ale v jeho vylepšené verzi *Typescriptu*.

ukázka modulu pro vytváření příspěvku



Velmi dobře se mi pracovalo s formuláři, které má skvěle vyřešený Angular. Na políčka jdou přidávat tzv. Validatory, které můžou kontrolovat různé vstupy (je text moc dlouhý? je políčko vyplněne? je zadaný email správně?).

Ve frontendu jsem také použil pár externích modulů. AngularJWT na práci s json web tokeny na frontendu. Ng-click-outside na uživatelský vstup.

## 

## Backend

Backend má za úkol přijímat požadavky z frontendu a vytahovat data z databáze. Frontend se dotáže na nějaké informace, nebo iniciuje změnu. Backend vyhledá relevantní info v databázi a posílá chyby, nebo získané data, pomocí objektů.

získávání notifikací



Backend je napsaný pomocí frameworku NodeJS, ve kterém je dostupných několik pluginů, které usnadňují práci (viz. technologie).

## 

## Komunikace mezi serverem a frontendem

Aby byla komunikace mezi přihlášeným uživatelem bezpečná, rozhodl jsem se využít tzv. json web tokenů. Pokaždé, když se uživatl přihlásí, na serveru se vygeneruje tento token a pošle se uživateli, kterému se uloží do lokální paměti. Pokaždé když udělá nějaký request spojený se změnou jeho účtu, pošle se i token, který se na serveru ověří a následně se provede akce.

Komunikace je udělána pomocí tzv. endpointů, což je adresa určité akce na kterou se uživatel táže. Například když chci získat seznam lidí co mě sleduje, udělám GET request na endpoint /follows/:id, kde id je uživatelské identifikační číslo.

# 

# Závěr

Tento projekt byl pro mě něco úplně nového a náročného. Vzhledem k tomu, že jsem do této doby nikdy s backendem nepracoval, naučil jsem se spoustu velice užitečných informací ohledně jeho logiky, práce s databází a komunikace mezi uživatelem a server. Tyto informace se do budoucna určitě budou hodit.

K práci jsem přistupoval velmi systematicky, což mi velmi pomohlo - projekt jsem si rozvrhnul, chronologicky jsem zpracoval všechny důležité části (design, databáze, technologie) a pak jsem projekt naprogramoval.

Dále jsem se naučil pracovat s verzovacím systémem GitHub, který je v dnešní době velmi důležitý a určitě se s ním setkám.

Všechny funkce, které jsem chtěl udělat a popsal na začátku, jsem splnil a výsledkem je funkční aplikace, kterou může používat spousta uživatelů.

# 

# 

# 

# Zdroje

## Technologie

NodeJS: <https://nodejs.org/en/>

ExpressJS: <https://expressjs.com/>

bcrypt: <https://www.npmjs.com/package/bcrypt>

jwt-simple: <https://www.npmjs.com/package/jwt-simple>

cors: <https://www.npmjs.com/package/cors>

nodemailer: <https://nodemailer.com/about/>

async: <https://caolan.github.io/async/>

bodyParser: <https://www.npmjs.com/package/body-parser-json>

Angular 5: <https://angular.io/>

AngularJWT: <https://github.com/auth0/angular-jwt/>

ng-click-outside: <https://www.npmjs.com/package/ng-click-outside>

MongoDB: <https://www.mongodb.com/>

Mongoose: <http://mongoosejs.com/>

## Využitý kód

generoví náhodného stringu: <https://stackoverflow.com/questions/1349404/generate-random-string-characters-in-javascript>