Gymnázium Jana Keplera

Maturitní práce z informatiky

Aplikace ke komunikaci – Ez chat

autor: Michal Kubiče, 4.C

vedoucí práce: Karel Jílek

**Anotace:**

Aplikace má sloužit k jednoduché a přehledné komunikace mezi více lidmi. Uživatel si založí vlastní účet a má možnost vytvořit soukromou místnost, kterou následně může sdílet s dalšími lidmi.

Instalace

- Aplikaci jsem nahrál na platformu heroku, na které aplikace běží zdarma

- lokálně:

1. Naklonování repozitáře ze služby Github:

<https://github.com/MichalKubice/ez-chat>

1. Je třeba mít nainstalovaný NodeJS - <https://nodejs.org/en/>
2. Následně je třeba nainstalovat moduly jak pro backend, tak pro frontend

– ve složce client – npm install

- v složce s aplikací – npm install

1. Poté ve složce s aplikací – npm run dev - spustí frontend i backend zárověň
2. <http://localhost:3000>

Úvod

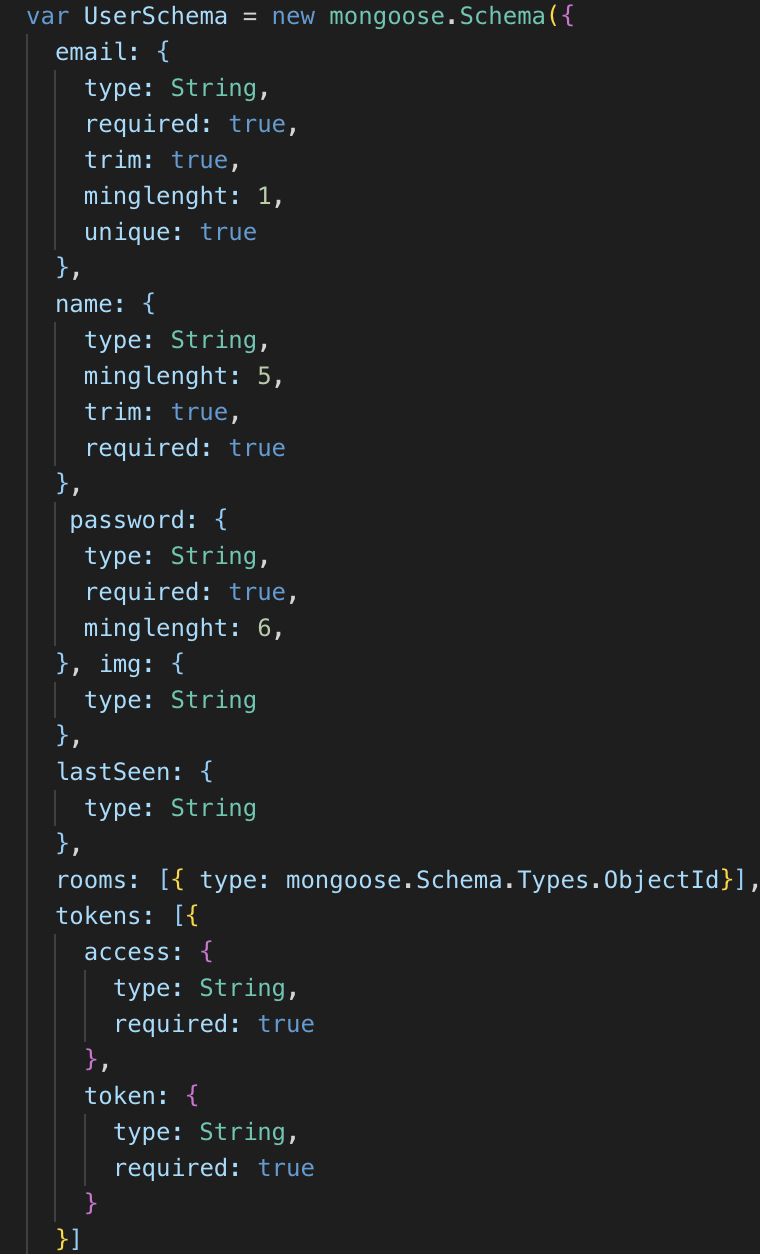
Rozhodl jsem se vytvořit jednoduchou aplikaci, ve které si uživatel bude moci vytvořit privátní místnost a komunikovat zde s přáteli. Tento koncept najdeme na internetu často, nicméně málokdy je možnost si zde založit účet a místnost vytvořit pernamentně. Uživatel se tedy zaregistruje, založí privátní místnost, která vygeneruje unikátní kód, který následně uživatel sdílí s ostatními – ke kódu jim také musí sdělit heslo, které zadá při vytváření místnosti. Můj cíl byl zanechat aplikaci jednoduchou a přehlednou – žádné nadbytečné funkce, aby bylo velmi jednoduché aplikaci používat. Ostatní uživatelé mohou vidět pouze přezdívku, tudíž je možné zůstat zcela anonymní.

Technologie

Jelikož jsem si zvolil aplikaci, ve které je potřeba jak front-end, tak back-end, tak mi přišlo výhodné využít javascriptovou platformu NodeJs, která má spoustu knihoven, čímž nám ulehčuje práci, v kombinaci s ExpressJs, která dobře funguje ve spojení s front-endovou knihovnou React-js, tudíž front-end tak back-end je napsán pomocí javascriptu. React, jak již název říká, umožnuje aplikaci fungovat bez potřeby refreshovat stránku, jelikož aplikaci rozkládá na komponenty a vytváří interaktivní uživatelské rozhraní – login page je jeden komponent, main page je další komponent, zpráva další atd. – a react dle potřeby vyrenderuje právě ten komponent, který potřebujeme. Využil jsem také redux, který vytváří store – globální state. Ten nám říká, v jaké fázi vlastně jsme a umožnuje komponentům předávat state mezi sebou. 

Server komunikuje s klientem díky REST - klient posíla http requesty na mnou vytvořené API endpointy na serveru(GET, POST,DELETE,PUT), a server mu zašle zpátky response – když bylo vše tak jak mělo být – succes, pokud něco bylo špatně, např. špatný formát dat, tak error.

Další bylo zapotřebí vybrat databázi, ve které budou uloženi uživatelé, místnosti a zprávy. Vybral jsem si NoSql MongoDB, která používá JSON formát a je velmi jednoduchá k použití spolu s NodeJS díky knihovně mongoose, která nám snadno umožnuje pracovat s daty pomocí schémat v databázi. Navíc mi vyhovovalo, že je možné jednoduše propojit kolekce navzájem kvůli tzv. referencím – např. každý User ma atribut rooms, ve kterém jsou unikátní ID odkazující na každou místnost, ve které uživatel je. Při vývoji jsem databázi vytvářel lokálně, později jsem využil služby mongoDB atlas, která zdarma poskytuje až 512 MB uložistě, což je pro naše účely dostačující.

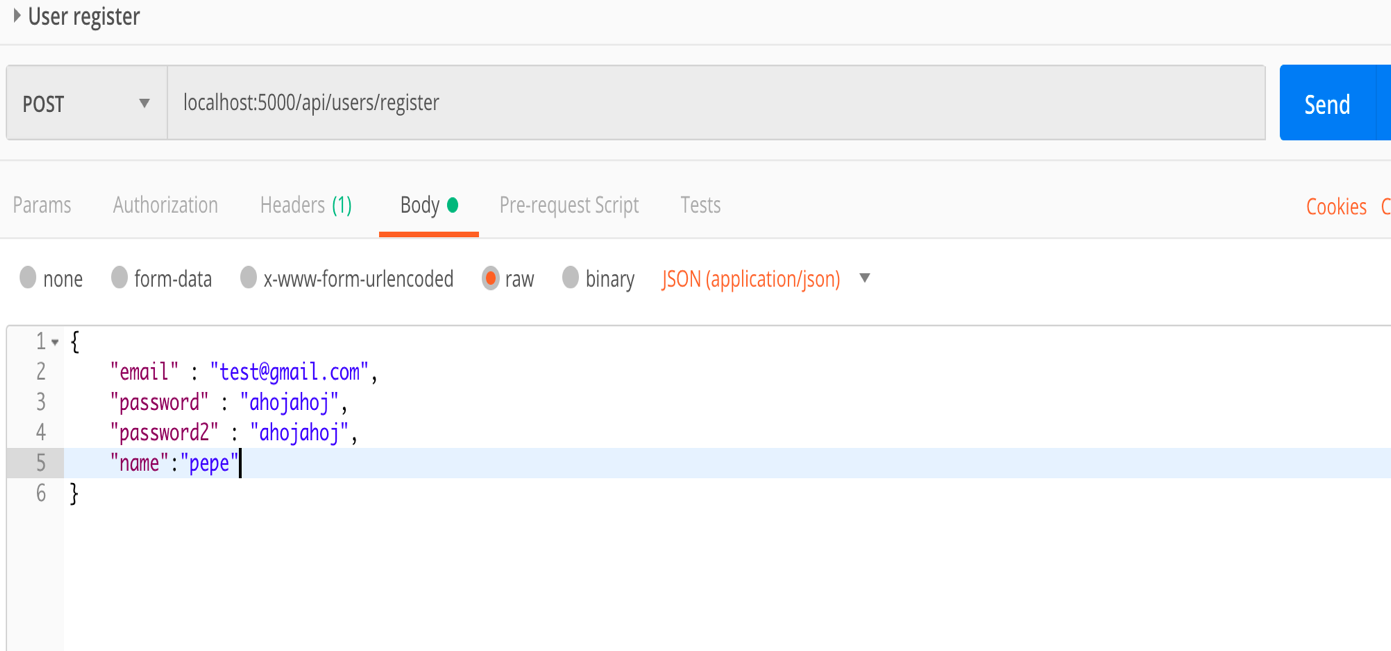




Následně bylo ještě zapotřebí vytvořit bezpečný přihlašovací systém, a to mi umožnila celkem jednoduše knihovna JWT. Uživatel při úspěšném přihlášení obdrží token, který je následně posílán s každým requestem na server. Token je uložen jako header u uživatele a zároveň v databázi, a při každém requestu docházi k porovnání. Token je při každém přihlášení vytvořen nový a přepisuje ten starý, tudíž v jednu chvíli může mít pouze jeden token.

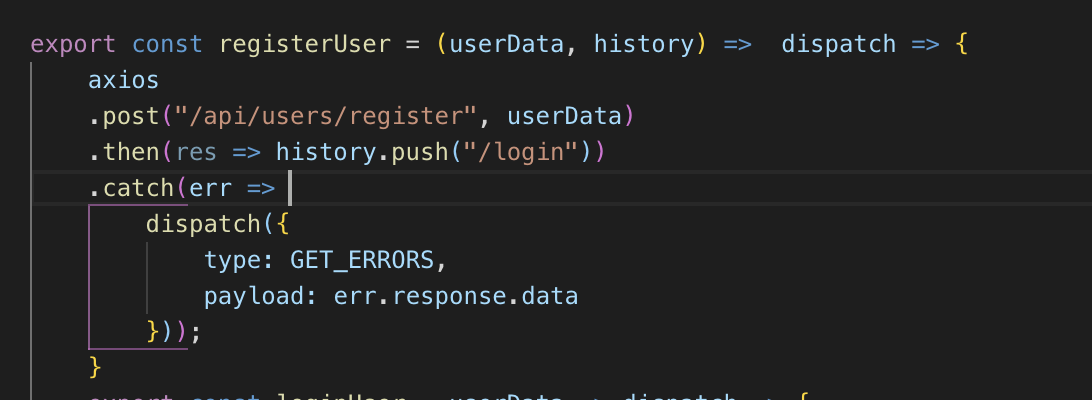
Postup

Nejdříve jsem začal pracovat na back-endu. Tato část mi přišla náročnější, jelikož nic z toho co vytvoříme zatím moc nevidíme. Používal jsem aplikaci Postman, která umožnuje „hitovat“ API endpointy, aniž bychom měli vytvořené uživatelské rozhraní.

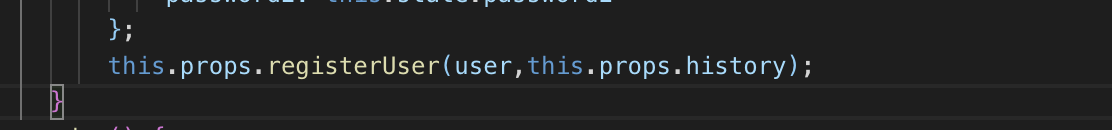


Pokud je endpoint napsán správně a funguje tak jak má, po kliknutí na send nám postman pošle response. Endpointy se dělí na 2 skupiny – pro users a pro rooms. Např. user má endpoint na registraci, login, getUserRooms(zobrazí místnosti, ve které je daný user). Room má zase např. getRoomMessages(zobrazí zprávy příšlušné místnosti), joinRoom,createRoom, sendMessage atd.

Když jsem měl všechny potřebné endpointy hotové a funkční, bylo už „jen“ zapotřebí je propojit s frontendem. Bylo také potřeba převést API requesty na akce, a to kvůli reduxu. Tudíž bylo možné pomocí knihovny axios posílat requesty v jednotlivých komponentech. Response ze serveru následně nastavil globální state – zase buď úspešně, nebo error.

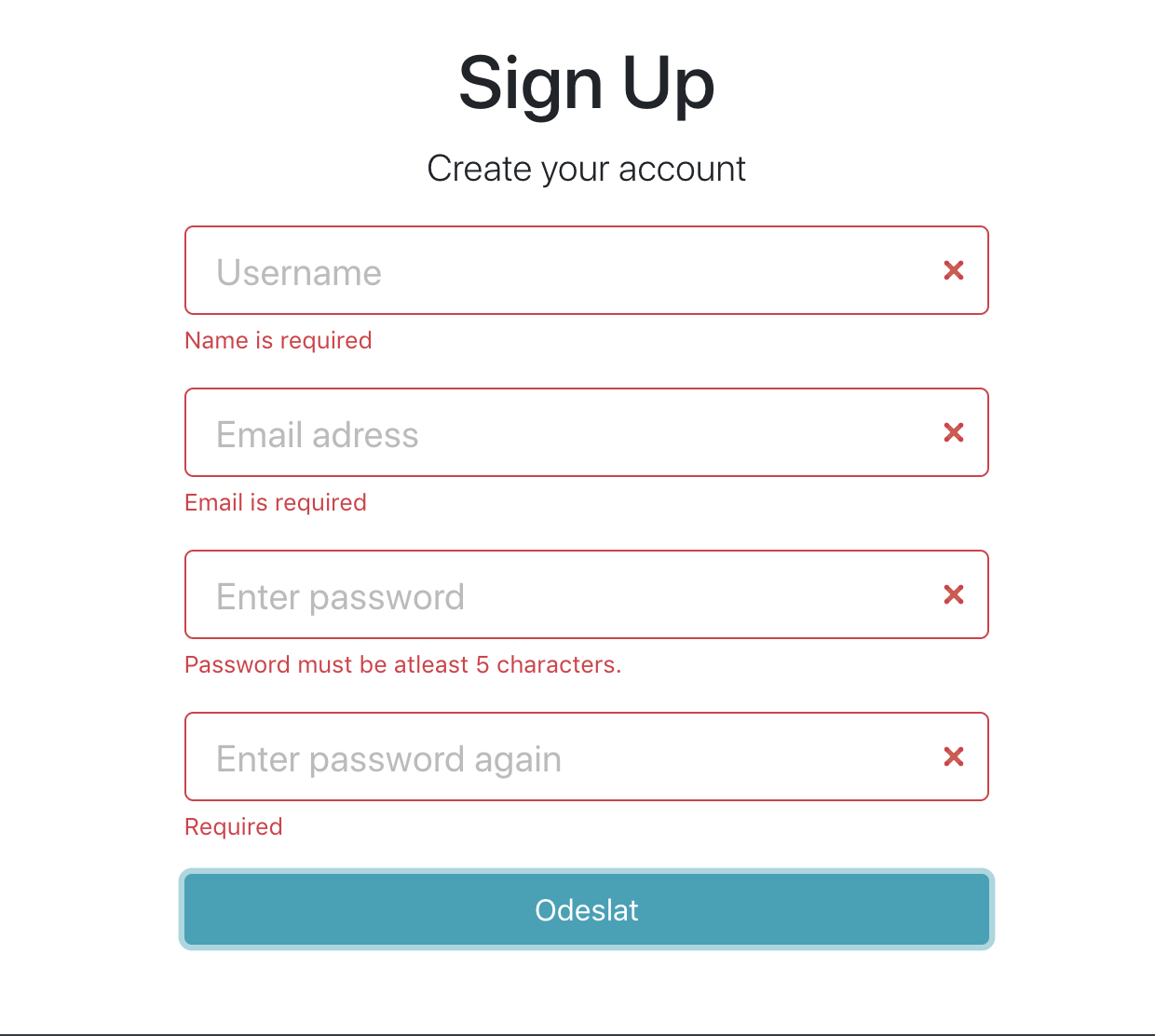


převedení api na akci



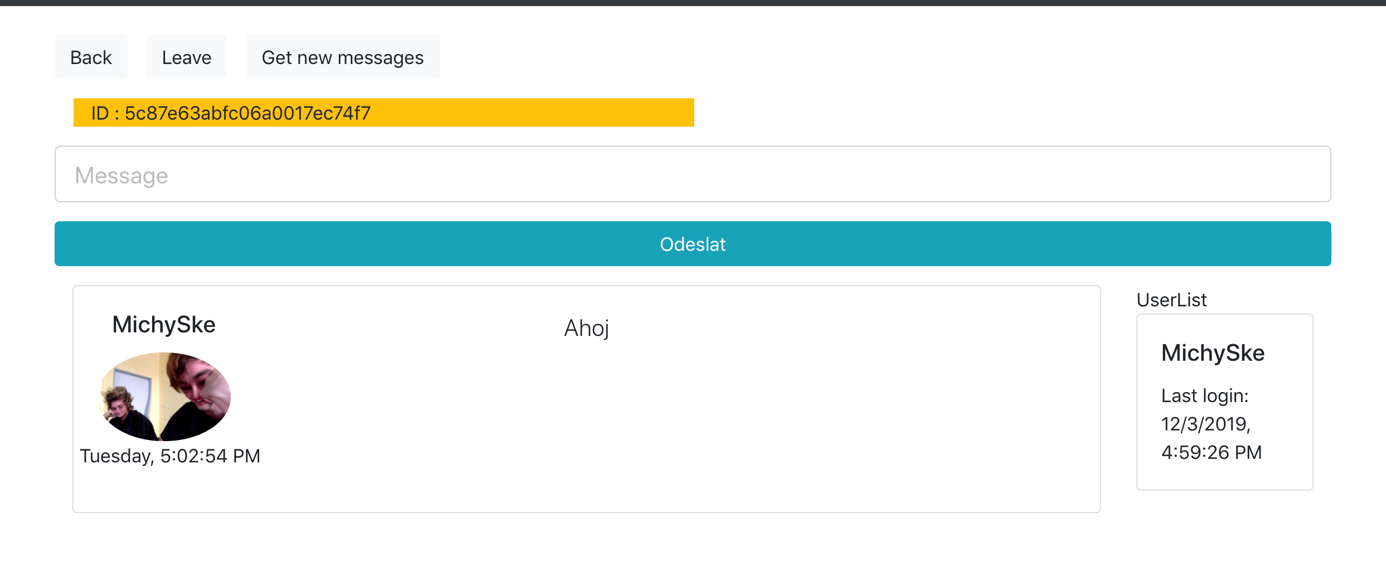
následné vyvolání akce v komponentu

Nejdříve jsem vytvořil komponenty Landing, Register a Login. Bylo třeba také vytvořit validaci – když response ze serveru házel nějaký error, musel to dát vědět klientovi – např. když se snažil registrovat a nezadal potřebné údaje. Tuto část považuji z frontendu za nejtěžší, jelikož jsem s ní strávil nejvíce času. Když se mi tato část podařila zprovoznit, dále to šlo již celkem snadno, jelikož postup byl podobný.



Když již bylo možné se zaregistrovat a přihlásit, vytvořil jsem dashboard – hlavní stránku. Tam měl uživatel své místnosti, možnost místnost vytvořit nebo se připojit již do existující místnosti. Uživatel má také možnost nastavit si obrázek. Při vývoji lokálně jsem pro obrázky využíval lokální uložiště, nicméně při umístění aplikace na platformu heroku jsem zjistil, že takové řešení nefunguje. Tudíž jsem musel využít služby cloudinary, která poskytuje zdarma až 100MB uložiště na obrázky, což pro účel avatarů bohatě stačí. 

Po vytvoření místnosti již můžeme posílat zprávy. Nahoře máme ID, kterou můžeme poslat kamarádům společně s heslem, a ti mají možnost se do místnosti připojit.



Závěr

Jelikož jsem dříve nic s takhle velkým rozsahem nědelal, nebylo to pro mě vůbec jednoduché. S front-endem jsem měl zkušenosti téměř nulové a s backendem minimální, a díky této práci jsem získal nespočet zkušeností a pochopil jsem, jak vlastně funguje server-side a client-side. Nejvíce jsem se naučil hledat informace na internetu, např. v dokumentacích nebo na stackoverflow, což bylo ze začátku hodně náročné, ale časem jsem již věděl co vlastně hledat a postupem času se mi práce zjednodušovala. Občas se stalo, že jsem se zasekl na zcela triviální věci, u které jsem seděl několik hodin, ale o to víc jsem byl pak spokojenější, když se mi to podařilo zprovoznit.

Práci jsem si na začátku snažil navrhnout – funkce, technologie a databázi, což mi následně ulehčilo práci. Když byl backend funkční, zdálo se už jednoduché to propojit s frontendem, ale rozhodně tomu tak nebylo. Kolikrát jsem si říkal, že to musím celé předělat, ale naštěstí jsem vždycky našel řešení.

Dále jsem si procvičil práci se službou Github, což považuju za užitečné, jelikož se v dnešní době používá často a práce s ním se mi v budoucnu bude hodit. Navíc mi několikrát pomohl, když se něco pokazilo a já nevěděl co, tak jsem jednoduše stáhl předchozí funkční verzi.

Ačkoliv jsem si vědom, že by aplikace mohla být obohacena dalšími funkcemi, jsem s výsledkem spokojený, jelikož hlavní účel splňuje. Rozhodně mi dala ponaučení do budoucích projektů, např. v přehlednosti v kódu, vyhledávání informací, technologiích a hlavně orientace v kódu.

Zdroje:

<https://stackoverflow.com/>

<https://www.google.com/>

<https://github.com>

https://www.codecademy.com/

Využité packages v Backendu:

Databáze: <https://www.mongodb.com/cloud/atlas>

Cloud platforma:

<https://www.heroku.com/>

Postman:

<https://www.getpostman.com/>

Cloudinary:

<https://cloudinary.com>



Využité packages ve frontendu:

