# Team seeker

## Dokumentace k projektu

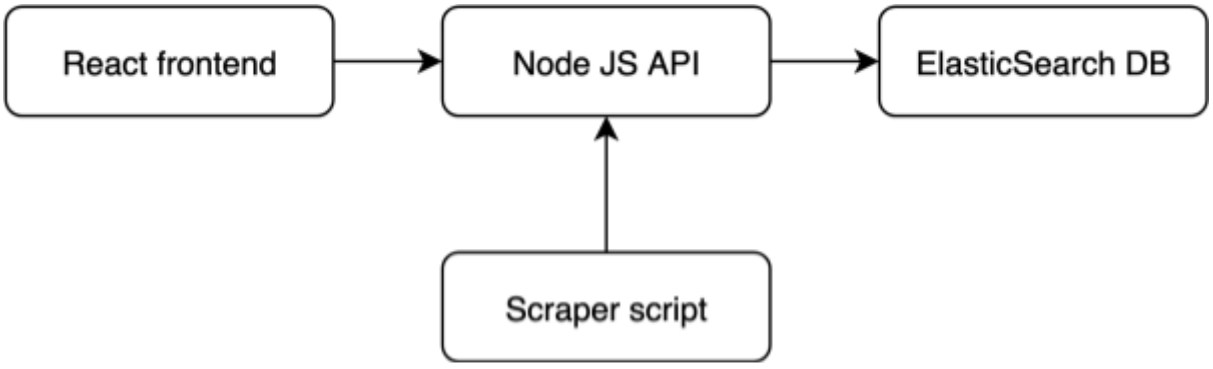
Lukáš Kúšik, Gabriela Chmelařová, Petr Kohout

### Motivace

IGEM je mezinárodní soutěž vědeckých různých věkových kategorií z celého světa. Letošního ročníku se účastní i brněnský tým s členy z řad našeho řešitelského týmu. V rámci soutěže se získávají body z velkého množství kategorií, mezi které patří i přínos komunitě iGEM týmů. Při každoročním výběru témat jednotlivých družstev dochází k vyhledávání a případnému kontaktování již dříve registrovaných týmů a látek, které tyto týmy vytvořily. Cílem tohoto projektu je ulehčit vyhledávání mezi velkým objemem dat a umožnit tak novým týmům prohledat a identifikovat možnosti výzkumu.

## Architektura

Projekt se skládá ze tří hlavních částí, které jsou zobrazeny na obrázku 1. První část tvoří *Scraper script,* který je zodpovědný za extrakci dat ze stránek iGEMu (seznam stránek, z kterých se sbírají informace lze nalézt v souboru *README.md*). Skript po extrakci dat zasílá tato data na server, který je po kontrole na absenci potřebných atributů ukládá do databáze ElasticSearch. Server data nadále poskytuje webové aplikace skrze API rozhraní. Webová aplikace zobrazuje informace uživateli a umožňuje mu zadávat slova pro vyhledávání, která jsou odeslána na server, který ten požadavek zpracuje a dotáže se na databázi. Výsledek dotazování je odeslán zpět aplikaci, která tyto výsledky zobrazí.




Obrázek 1: architektura projektu

## React frontend

Webová aplikace byla sestavena pomocí JavaScript knihovny React, která byla zvolena především díky jejím možnostem použití takzvaných komponent, které umožňují snadnější manipulaci s často se opakujícími prvky. Základní znalosti potřebné k práci s touto knihovnou byly získány formou tutoriálu Learn React JS.

Aplikace komunikuje s API pomocí zasílání požadavků GET a POST. Samotná aplikace je založena na třech hlavních komponentách. První z nich je komponenta App, která nejprve pomocí metody fetch vytvoří požadavek GET na data o týmech a biobricks a následně je připraví k zobrazení. Dále vykreslí jednu ze dvou podob další komponenty nazvané SearchBar, která umožňuje vyhledávání a zobrazování výsledků. Je možné volit mezi vyhledáváním týmů a biobricks. Dle příslušného zvoleného vyhledávače se následně načtou získaná data, jejichž formát je upraven pomocí komponent, které jsou pro každý vyhledávaný objekt vytvořeny.

Vyhledávání funguje následujícím způsobem. Zadaný výraz je vyhledáván s předem specifikovanými parametry, které je možné změnit pomocí přepínačů pod vyhledávacím polem. Je možné zvolit, v které kategorii se má prohledávat a zda mají být zvoleny výsledky, co výraz obsahují, nebo právě ty, co ho neobsahují.

Způsob získávání výsledků vyhledávání je založen na zasílání požadavků POST, jejichž obsah je postupně vystavěn na základě zvolených parametrů. Nejprve je získán název právě zvolené kategorie, ve které bude zadaný text vyhledáván. Dále se získá příznak, zda má nebo nesmí výsledný text tento výraz obsahovat a nakonec je jako hodnota uložen zadaný text z vyhledávacího pole. V případě přímého zadávání textu bez předchozí volby kategorií jsou k sestavení požadavku použity defaultně zvolená nastavení.

## Node JS API

### Použité knihovny

- bcrypt - API - vytváření hash hodnoty hesla, která se ukládá do databáze
- body-parser - API - parsování požadavků směřujících na server
- elasticsearch - API - klient databáze ElasticSearch
- express -API - framework pro Node.js
- jsonwebtoken -API - autentizace uživatelů (jwt)
- morgan -API - logování serveru
- nodemailer -API - zasílání e-mailové notifikace uživateli (vytvoření účtu)
- randomstring -API - generování unikátního a náhodného odkazu pro ověření uživatele při vytváření účtu

Pro realizaci aplikačního API byl využit Node.js. Toto řešení bylo zvoleno na základě jeho vysoké obliby a rozsáhlé podpory v podobě učebních materiálů a existujících knihoven.

Pro získání základních znalostí z oblasti API technologií jsem využil sérii videí o vytváření REST API na YouTube na kanále Academind.

Server přijímá požadavky typu GET, POST a DELETE a zprostředkovává veškerou komunikaci s databází ElasticSearch. Obslužná rutina požadavků se odehrává následujícím způsobem. Požadavky jsou přijaty serverem a distribuovány zodpovědným zpracovatelům dotazu. V případě GET požadavků dochází k okamžité komunikaci s databází a vrácení výsledku. V případě POST požadavků dochází v některých případech k ověření uživatele, který tento požadavek zadal, a k následnému ověření korektní struktury dat. Při zpracování požadavku typu DELETE dochází k ověření uživatele a následnému provedení akce. V případě jakékoliv chyby je vráceno příslušné chybové hlášení.

V aplikaci se zpracovávají hlavní tři druhy dat: uživatelská, biologická (tzv. biobricks) a informace o týmech. Pro každou kategorii byl vytvořen vlastní manipulátor (handler). Struktura požadavků směřujících na API je popsána níže.

**GET**
V základním nastavení běží server na portu 3001.

`localhost:3001/teams`
Vrací seznam týmů, které jsou obsaženy v databázi.

`localhost:3001/teams/structure`
Vrátí seznam atributů obsažených v záznamech týmu.

`localhost:3001/biobricks`
Vrací seznam biologických struktur, které jsou obsaženy v databázi a byly registrovány v rámci iGEMu.

`localhost:3001/biobricks/structure`
Vrátí seznam atributů obsažených v záznamech biologických struktur.

**POST**

`localhost:3001/teams/match`
`localhost:3001/biobricks/match`
Vrací data získaná pomocí dotazu, který je specifikován v těle požadavku se strukturou a logikou popsanou níže.

```
{
      NAME:[{contain: Bool, value: String}]
      .
      .
      .
}
```

Pozice name obsahuje jméno atributu, který se vyhledává. Hodnotou tohoto atributu je seznam, kterých má výsledek nabývat. Položka `contain` značí příznak zda hodnota atributu `value` by měla nebo nesmí být obsažena ve výsledku. Jednotlivé požadavky jsou následně proiterovány a je pomocí nich sestaven dotaz, který se přepošle na databázi.

Níže je popsán požadavek na týmy, které se registrovaly v roce 2020 a jejich jméno neobsahuje číslo 2, ale obsahuje hodnotu `team`.

```
POST: localhost:3001/teams/match
{
      "title":[{"contain":false, "value":"2"},
               {"contain":true, "value":"team"}],
      "year":[{"contain":true, "value":"2020"}]
}
```

z této struktury je sestaven dotaz, který využívá seznamu hodnot, které by měly, či nesmí být pravdivé. Výše uvedený příklad bude převeden do následující interpretace.

```
{
      "query":{"bool":{
                  "should": [{"match":{"title":"team"}},
                             {"match": {"year": 2020}}],
                  "must_not": [{"match": {"title": "2"}}]
               }}
}
```

**Vkládání**
Pro vkládání dat do aplikace lze využít POST požadavků na následujících adresách.

`localhost:3001/teams`
`localhost:3001/biobricks`
Po obdržení dat serverem a ověřením uživatele je v případě chyby zaslána uživateli příslušná chybová hláška.

Pro možnost budoucího přidávání aktuálnějších záznamů byla vytvořena možnost vkládání záznamů pomocí skriptu, který tyto informace extrahuje ze stránek iGEMu. Pro kontrolu vkládání byl zaveden proces autentizace uživatelů, který se využije především při vkládání nových dat. Běžný uživatel k plnohodnotnému použití aplikace ji nebude potřebovat.

`localhost:3001/users/signup` - pošle data uživatele, který požaduje registraci
`localhost:3001/users/login` - provede přihlášení
Při přihlášení i registraci je nutné poslat v rámci POST požadavku e-mailovou adresu účtu a heslo. Při vytváření uživatelského účtu se ověří, zda účet s danou adresou je již zřízen. V případě již existujícího účtu je uživatel informován. Poté je heslo převedeno na hash pomocí knihovny *bcrypt* a účet je vytvořen účet v dočasném úložišti, ve kterém čeká na potvrzení. Poté je uživateli zaslán e-mail s odkazem a po jeho prokliknutí je uživateli vytvořen skutečný

účet a je možné se přihlásit. Níže je znázorněna struktura těla požadavku pro registraci a přihlášení i s odpovědí při úspěšném přihlášení.

```
POST: localhost:3001/user/signup
{
      "email": "test@email.com",
      "password": "testPassword"
}


POST: localhost:3001/user/login
{
      "email": "test@email.com",
      "password": "testPassword"
}

response = {
                  "message": "Auth successful",
                  "code": 200,
                  "token": "eyJhbGciOiJIUzI1..."
            }
```

V odpovědi je obsažen *token,* který se používá k autentizaci uživatele. Požadavek je nutné posílat v hlavičce v atributu Authorization ve formátu Bearer s následujícím tokenem.

```
{
      Authorization: “Bearer eyJhbGciOiJIUzI1…”
}
```

Pro smazání všech dočasných účtů je nutné jako autentizovaný uživatel poslat na url
`DELETE: localhost:3001/user/confirm`

`DELETE: localhost:3001/teams/:teamCode`
pro smazání týmu pomocí položky *teamCode*

`DELETE: localhost:3001/biobricks/:title`
pro smazání biobricks pomocí položky *title*

`DELETE: localhost:3001/user`
pro smazání uživatele (sebe sama) pomocí položky. V těle požadavku musí být zaslán token a přihlašovací údaje.

## ElasticSearch DB

Jako místo pro ukládání dat o iGEM týmech a biobricks jsme zvolili databázi ElasticSearch, díky její specializaci na fulltextové vyhledávání.

ElasticSearch umožňuje tvorbu textových indexů nad jednotlivými poli, které umožňují v textu vyhledávat jen základ slova, skloňované výrazy nebo synonyma. Je proto vhodná pro naše využití, kde uživatel z frontend webové stránky zadává (nepřesný) vyhledávaný řetězec a backend pracuje vlastně jako vyhledávač.

## Scraper script

Data o iGEM týmech a biobricks nejsou v současné době dostupné ve snadno stažitelné formě nebo prostřednictvím API. Nacházejí se na webových stránkách organizace iGEM ve formě "wiki" stránek. Abychom tato data získali k následné indexaci, využili jsme techniku scraping.

Při scrapingu iGEM týmů si skript nejprve stáhne tabulkový soubor CSV se seznamem týmů, ve kterém jsou částečné informace o historicky všech týmech zúčastněných v soutěži iGEM. Každý tým má (zvlášť pro každý rok) svou detailnější stránku, na níž se nacházejí další informace, jako například název školy, seznam členů týmu nebo divize. Pro nás je nejdůležitější pro vyhledávání název a abstrakt projektu daného týmu.

Biologických částí zvaných biobricks je na stránkách iGEM velké množství a jejich kompletní seznam není k dispozici. Nacházejí se v různých kolekcích podle kategorie. Skript prohlédne množství takových kategorií a najde v nich stránky s popisem jednotlivých biobricks. Popis vlastních biobricks je ve formě wiki stránky. Ta má často členitou formu (odstavce, obrázky). Skript však z ní získává pouze textový obsah všech odstavců.

Získaná data jsou odesílána prostřednictvím POST requestu do Node JS backendu. Při posílání se skript autentizuje pomocí bearer tokenu, který získá, při spuštění skriptu, z Node JS backend, s použitím e-mailové adresy a hesla získaných z argumentů skriptu.

Skript je napsán v jazyce Kotlin a využívá knihovnu skrape.it, která je přehledná a vhodná pro snadnou a rychlou implementaci do tohoto projektu.

## Budoucí využití

Využití našeho projektu shledáváme především v praktickém využití týmy soutěže iGEM. Nápad na tuto webovou aplikaci byl vytvořen především díky požadavkům ze strany účastníků soutěže, kteří využívají data o minulých tématech týmů k inspiraci a případnému navázání na ně. Vyhledávání jsme rozšířili o možnost vyhledávání vytvořených látek zvaných biobricks, které jsou pro týmy esenciální a informace o nich se nacházejí na několika místech. Současné nástroje umožňující vyhledávání podobných dat jsou podle uživatelů nepřehledné a nebo obsahují pouze neaktuální data. Z toho důvodu jsme vytvořili náš nástroj s možností budoucí aktualizace dat.

## Spuštění

npm run start

K dispozici je také docker-compose soubor, který spustí všechny složky projektu (frontend, backend, databáze) v nástroji Docker.

## Rozdělení práce v týmu

Lukáš Kúšik: scraper script, konfigurace Elasticsearch, nasazování na server, Docker
Gabriela Chmelařová: webová aplikace, modifikace dat zadaných uživatelem na API dotazy, zpracování přijatých dat
Petr Kohout: API, komunikace s databází, dotazovací logika, autentizace uživatele