## Въведение

Приложението представлява уеб страница (Single Page Application), в която потребителите могат по потърсен адрес да намерят точното му местоположение върху карта. За целта е изградена “търсачка”, както и карта с подложка на Google Maps. Процесът по атрибутиране на адрес към координати се нарича “Geocoding”. За успешното геокодиране е използван сървис на ArcGis.

Начин на стартиране на приложението:  
1. yarn install, за да се инсталират всички “dependencies”

2. Yarn start, за да се стартира дев сървър.

3. By default, приложението се намира на localhost, порт 3000

## Технологии

Основните технологии, които са използвани са JavaScript и CSS.

Беше взето решение да се използва ReactJS като библиотека, заради удобството, което тя дава, а именно реактивния начин, по който интерфейсът реагира на промени в данните.

За надграждане на code base-a беше взето решение да се използва TypeScript, поради причината, че има данни които се преизползват на различни места, както и предимството всички данни да имат ясно обособени типове.

Второстепенни технологии, които са използвани:

* Zod - библиотека, която се използва за валидационни схеми, чрез които да се валидират данните, върнати от ендпойнтите;
* Immer - библиотека, която се използва за “immutable” ъпдейтване на “state”;
* React-google-maps - библиотека, която се използва за визуализация на google maps

## Решения

След разглеждане на документацията на сървиса, беше взето решение да се използват два ендпойнта от сървиса - suggest и findAddressCandidates. Като, suggest ендпойнтът е използван за търсене на адреси, а findAddressCandidates е използван, за да се геокодира избрания от потребителя адрес. В данните върнати от suggest, за всеки адрес присъства поле magicKey, чрез което, извиквайки findAddressCandidates и подавайки го, се оптимизира процесът по геокодиране.

Имплементацията на търсенето на адреси е енкапсулирана в компонент, който като “деца” рендерира полето за търсене, както и резултатите. Взето е решение да се имплементира кеш за потърсените резултати, който е non-persistent (при презареждане на приложението, кешът се изчиства). Поради причината, че един адрес може да участва в няколко различни търсения, беше взето решение структурата на резултатите да е обект, в който се пазят адресите по ID. За ID се използва text полето на върнатия резултат, което може би не е най-оптималното, тъй като не може да се гарантира, че два еднакви стринга винаги сочат към един адрес. В отделен обект се пазят резултатите за направените търсения, като резултатите се пазят в масив, съдържащ ID-тата на резултатите. Кешът съдържа резултатите за всички предходни търсения.   
  
 Имплементацията на избиране на адрес и геокодирането е извършена, използвайки React Context. При селектиране на адрес от потребителя, се извършва заявка за геокодиране (findAddressCandidates) на адреса, като се подава самия адрес и magicKey поле. Решението да се използва контекст, е базирано на факта, че двата компонента, които се интересуват от геокодирането са разкачени един от друг. След като мине заявката и има данни, компонентът, в който е имплементирана картата, реагира на промяната в контекста и след това “премества” камерата на въпросното място.

## Алтернативни решения, подходи и технологии

Функционалността за търсене може да се имплементира и чрез използване на [Redux](https://redux.js.org/). Предимството за използването му е, че “стейтът” на резултатите би бил изолиран от самите компоненти, както и заявките могат да се изпълняват извън контекста на компонентите, използвайки “middleware” какъвто е [Redux Saga](https://redux-saga.js.org/). Решението **да не се използва** Redux e базирано на факта, че приложението е достатъчно малко с малко на брой компоненти и използването на такъв тип global state management solution би било overkill.   
  
 Имплементацията на кешът, може да бъде пропусната, ако се използва библиотеката [react-query](https://tanstack.com/query/v3/), която по подразбиране кешира резултатите за еднакви заявки. В комбинация с Redux, може да се използва и Redux RTK. Решението **да не се използва** react-query е, че кешът е сравнително изолиран, както и няма нужда от уголемяване на bundle size-a за нещо толкова изолирано и малко.  
  
 Търсенето е имплементирано със suggest ендпойнтът поради факта, че този ендпойнт има семантиката на “търсачка”. Потенциално, findAddressCandidates също може да се използва за “търсачка”, но семантично е грешно, тъй като неговата основна идея е да геокодира. Също така, спрямо документацията, комбинацията от двата ендпойнта работи оптимизирано (описаното по-горе с magicKey). От своя страна, използвайки само findAddressCandidates, би оптимизирало user experience-a тъй като се спестява асинхронното геокодиране при клик на адрес. От друга страна, обаче, не е нужно да се пазят данни в паметта, които не са нужни (координатите на адресите).