Západočeská Univerzita v Plzni
Fakulta Aplikovaných Věd

Katedra Informatiky a Výpočetní Techniky
Pokročilé Softwarové Inženýrství

Webová aplikace pro zpracování geografických údajů v novoasyrských textech - dokumentace pro frontend

Jakub Šmíd
Václav Honzík
Michal Schwob
Viktorie Pavlíčková

1. Úvod

Tento soubor obsahuje dokumentaci k frontendu webové aplikace pro zpracování geografických údajů v novoasyrských textech v rámci předmětu Pokročilé softwarové inženýrství na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni. Cílem bylo vytvořit webovou stránku užitečnou pro výzkumníky klínopisce v dané oblasti. Poskytne informace a nástroj, který výzkumníkům a dalším zájemcům pomůže lépe pochopit historické události Novoasyrské říše.

2. Technologie a architektura

Tato sekce obsahuje nejdůležitější použité technologie použité na frontendu aplikace.

2.1 Technologie

Asi nejdůležitější použitou technologií je JavaScript framework React, který umožňuje aplikaci strukturovat do komponent obsahujících mix HTML definujícího vzhled a JS, který obsahuje logiku. Protože JS je často velmi nepředvídatelný, použili jsme jeho více udržitelnou variantu TypeScript, který do JS přidává typování. React je poměrně volný framework a ze základu nenabízí velké množství funkcionality pro sestavení komplexnejších aplikací, proto bylo potřeba použít dodatečné third-party knihovny, které ale mají typicky velkou uživatelskou základnu.

Typickým problémem u single page aplikací je správa stavu aplikace (state management) a předáváním stavu mezi komponentami. K tomu je v aplikaci využitý framework Redux spolu s knihovnami Redux Toolkit (značné snížení boiler-plate kódu Reduxu) a Redux Persist (uložení stavu do paměti prohlížeče).

Pro komunikaci s backendem byla použitá knihovna Axios. Další velkou část aplikace obsahovala práce s mapou, pro kterou bylo nutné najít knihovnu, kterou lze s Reactem použít a zároveň bude pracovat s mapami, které lze volně používat bez větších omezení. Pro tento účel jsme použili knihovnu Leaflet a její wrapper React Leaflet.

Co se týče UI a UX, zde bylo použito knihoven několik - Material UI (velké množství reusable komponent), Formik, React Quill, React Virtuoso a React Beautiful DND (Drag and Drop funkcionalita). Veškeré použité knihovny a jejich verze jsou vidět v tabulce č. 1. Pro sestavení aplikace byl použit nástroj yarn.

Jméno	**Verze**
axios	0.27.2
dompurify	2.3.6
dotenv	16.0.0
formik	2.2.9
jwt-decode	3.1.2
leaflet	1.8.0
react	17.0.2
react-beatiful-dnd	13.1.0
react-csv	2.2.2
react-html-parser	2.0.2

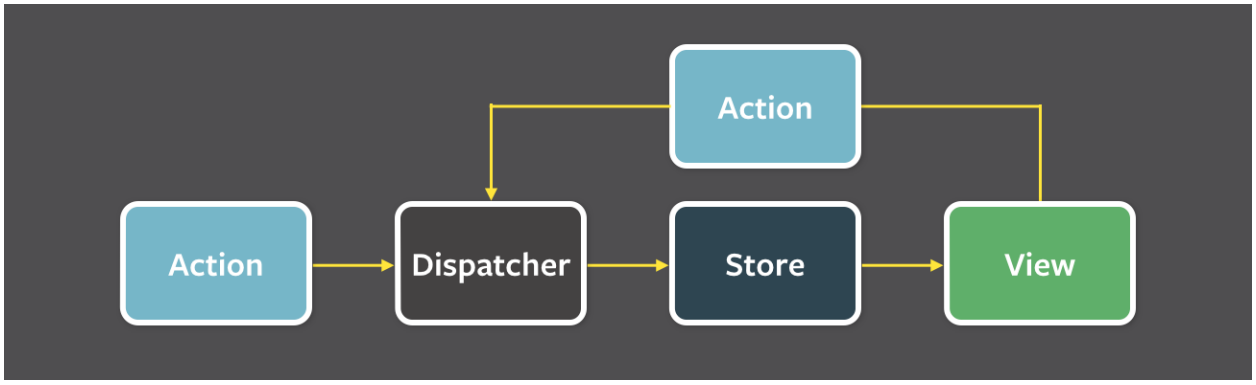
react-leaflet	3.2.5
react-leaflet-textpath	2.1.1
react-quill	1.3.5
react-redux	7.2.6
react-router-dom	6.2.2
react-scripts	5.0.0
react-virtuoso	2.11.2
redux	4.1.2
redux-persist	6.0.0
redux-thunk	2.4.1
swagger-typescript-api	9.3.1
ts-node	10.7.0
typescript	4.4.2
uuid	8.3.2
web-vitals	2.1.0
yup	0.32.11

Tab. 1 - Přehled všech dependencies

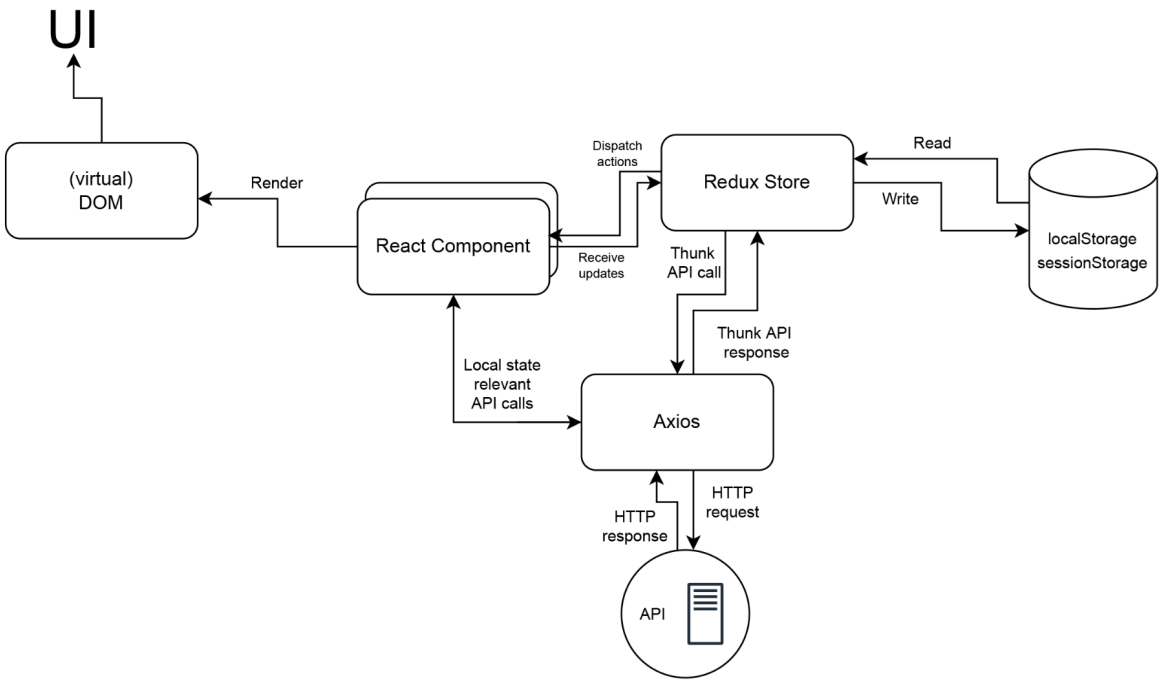
2.2 Architektura

Jak již bylo zmíněno, frontend je Single Page Aplikace (SPA) - tzn. běží na samostatném serveru a při odeslání GET požadavku na server dostane uživatel celou funkční aplikaci, která poté následně komunikuje s backend API.

Protože aplikace využívá Redux, architektura aplikace se řídí architekturou FLUX (viz. https://facebook.github.io/flux/docs/in-depth-overview/). Schéma architektury je vidět na obr. č. 1. Základem této architektury je store, který drží veškerý sdílený stav v aplikaci - tzn. data, které čte a upravuje typicky více než jedna komponenta. Komponenty mění stav pomocí akcí, které se doručují dispatcheru - což lze chápat jako objekt, který má na starost předání a provedení akce ve správné části storu. View v tomto případě lze chápat jako typickou React komponentu. Celkovou architekturu frontendu zobrazuje obr. č. 2.



Obr. 1 - FLUX architektura (zdroj: https://facebook.github.io/flux/docs/in-depth-overview/)



Obr. 2 - Celková architektura aplikace

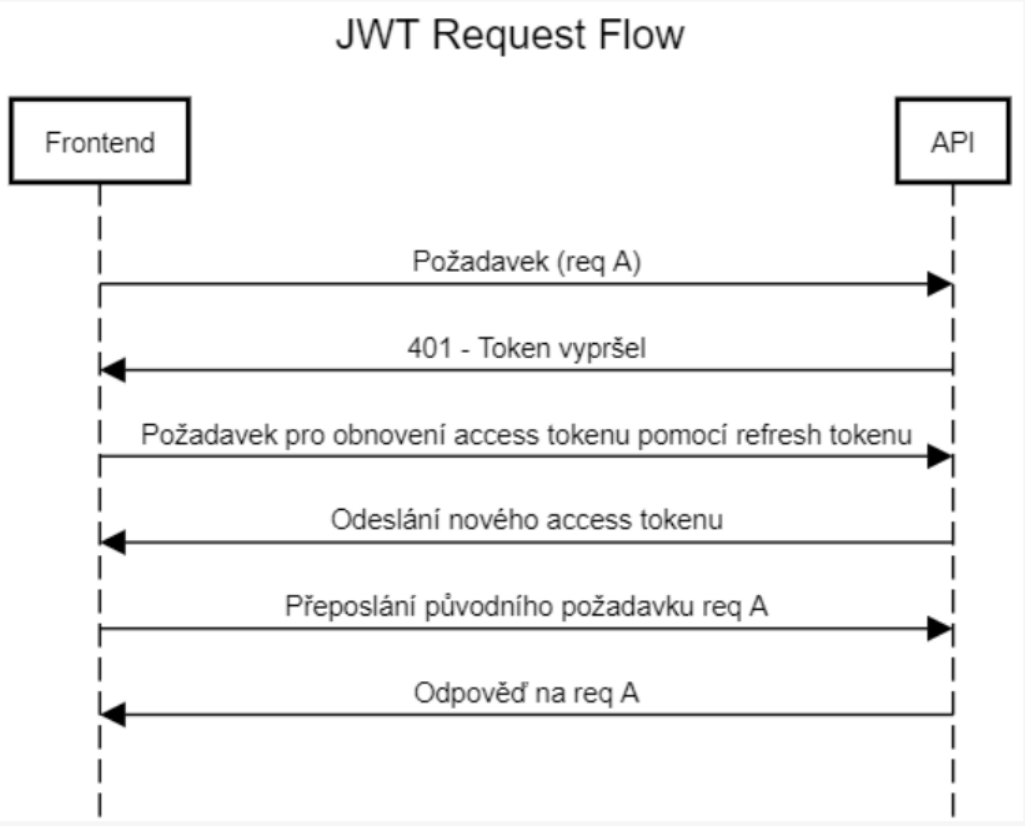
2.2.1 Komunikace s backendem

Ve velkém množství případů je potřeba pro komunikaci s backendem v požadavku dodat autorizaci pomocí tokenu (JWT), který vrací API po úspěšném přihlášení uživatele. Pro bezpečnost jsou tyto tokeny dva - tzv. access token, který má malou dobu platnosti kvůli bezpečnosti, a refresh token, který slouží pro získání nového access tokenu, aby se uživatel nemusel přihlašovat každých 15 minut.

Při odesílání požadavků musíme tedy navíc kontrolovat, zda-li jsme dostali response kód 401, který indikuje, že nám token vypršel, a pokud se tak stane musíme získat nový, uložit si ho a znovu odeslat předchozí požadavek. Celý průběh této operace je zobrazen na obrázku 3.

2.2.2 Routing

Přestože aplikace je SPA, typicky chceme aby se uživatel mohl při specifickém URL dostat na danou komponentu - např. pro */catalog* bychom chtěli zobrazit katalog, apod. Pro tuto funkcionalitu se využívá knihovna react-router-dom, která obaluje komponentu aplikace a kontroluje URL v prohlížeči, pro které podle definovaných pravidel (viz. soubor App.tsx) vyrenderuje danou stránku (komponentu). Pokud nelze žádnou stránku pro dané URL najít, uživateli se zobrazí generická stránka s hláškou “Page Not Found”.



Obr. 3 - Sequence diagram znázorňující obnovu access tokenu při vypršení a přeposlání původního požadavku.

2.2.3 Struktura projektu

Frontend celé aplikace se nachází ve složce **frontend.** V kořeni této složky jsou dva Python skripty pro nastavení vývojového a produčkního prostředí a Dockerfile pro nasazení na platformu Kubernetes, ve které celá aplikace běží. Dále jsou ještě přítomné soubory na stažení závislostí (package.json), nastavení TypeScriptu (tsconfig.json) a .env soubor s konfigurací, která se načte při startu aplikace. Zdrojový kód se nachází ve složce **src** a projekt je

strukturovaný **per-feature** - tzn. jednotlivé logické celky aplikace se nachází ve stejné složce - např. složka Catalog obsahuje veškerou funkcionalitu pro katalog, apod.

3. Přehled nejdůležitějších komponent

Tato sekce obsahuje přehled všech komponent / stránek, které se v aplikaci nachází. Kód všech komponent se nachází ve složce **frontend/src/features** a veškeré cesty jsou vztažené vůči této cestě.

3.1 Úvodní obrazovka

Komponenty pro úvodní obrazovku jsou uloženy ve složce **Home**. Obsah komponenty je při otevření asynchronně získán požadavkem na backend a následně zobrazen jako formátované HTML (Home.tsx). Uživatelům s právem administrátora je umožněno upravit úvodní obrazovku (EditHome.tsx) využitím rich-text editoru z knihovny react-quill.

3.2 Katalog

Všechny komponenty týkající se katalogu jsou uložené ve složce **Catalog**. Katalog se skládá z tabulky (CatalogTable.tsx) nalezených toponym a filtru (CatalogFilter.tsx) pro zpřesnění vyhledávání. Při načtení stránky se na backend automaticky odešle požadavek pro získání všech toponym, které se asynchronně uloží do tabulky, případně se zobrazí hláška s chybou, pokud k nějaké došlo. Při změně filtru se navíc do vyhledávání přidají patřičné parametry.

V tabulce lze navíc na dané toponymum kliknout, což uživatele přesune do detailu (CatalogItemDetail.tsx) pro daný záznam. Pokud má uživatel patřičná práva, může informace i upravit (EditCatalogItem.tsx) a nebo záznam smazat.

3.3 Navigace

Složka **Navigation** obsahuje veškerý kód týkající se navigačního menu v aplikaci. Soubor Navigation.tsx obsahuje HTML pro navigační menu, které obsahuje komponentu NavigationMenu.tsx s jednotlivými tlačítky na dané stránky v aplikaci - ty se namapují z modulu navigationMenuItems.ts.

3.4 Nástroj pro trasování (Tracking Tool)

Kódově největší část aplikace tvoří nástroj pro trasovaní, jehož kód je umístěn ve složce **TrackingTool**. Tracking tool obsahuje velké množství komponent, které jsou umístěné do hlavní komponenty TrackingTool.tsx.

První důležitou komponentou je Mapa, která je ve složce **TrackingTool/Map**. Na mapě se může zobrazit několik typů bodů a cest, podle vstupu od uživatele:

- Cesta a lokality z textového vstupu od uživatele - ProcessedTextPath.tsx
- Lokality zadané souřadnicemi - CoordinatesMapMarkers.tsx
- Lokality nalezené z lokálního a externího katalogu - CatalogMapMarkers.tsx
- Cesta a lokality importované pomocí JSONu získaného z předchozího exportu - ExternalMapPaths.tsx

Mapa navíc obsahuje kontextové menu, které lze spustit pravým kliknutím a uživateli zobrazí dvě tlačítka pro vytvoření nové lokality ze souřadnic a nebo import z lokálního / externího katalogu - viz. složka **TrackingTool/Import**.

Kromě mapy se při načtení textu musí zobrazit také jak byl samotný text rozpoznán a ovládání pro případnou změnu trasy nebo přidání / odstranění bodů z trasy. Pro ovládání lokalit v cestě se v pravém dolním rohu zobrazí seznam s přetahovatelnými řádky - ReorderableMapPointList.tsx. Protože byl tento seznam překvapivě výkonově náročný na prohlížeč, byl virtualizován pomocí knihovny Virtuoso.

Poslední důležitou částí je import dříve exportované cesty, pro které bylo potřeba vytvořit dialog s výběrem souboru (viz soubory ExternalPathImportDialog.tsx a ExternalPathExportButton.tsx) a také menu pro ovládání, kde uživatel může danou importovanou cestu odstranit nebo schovat (ManageFilePaths.tsx).

3.5 Správa uživatelů

Všechny komponenty využity pro správu uživatelů jsou uloženy ve složce Administration. Zobrazení hlavní komponenty (Administration.tsx) se liší pro různé role uživatelů. Administrátor vidí seznam všech uživatelů a detail vybraného uživatele (UserDetail.tsx) s možností editace práv, změny hesla a smazání uživatele. Seznam uživatelů se asynchronně načte a uloží při otevření stránky. Jiné role uživatelů mají zobrazenou pouze možnost změny svého vlastního hesla.

3.6 Správa externích zdrojů

V trasovacím nástroji je možnost importovat místa z externích zdrojů. Data z těchto zdrojů nejsou automaticky aktualizována. K aktualizaci slouží tato komponenta (ExternalSources.tsx). Komponenta obsahuje popis souboru, který je potřeba nahrát k aktualizaci externích zdrojů, a formulář k nahrání souboru.

3.7 Bibliografie

Komponenty pro zobrazení a úpravu bibliografie je uložena ve složce **Bibliography**. Pracuje na stejném principu jako úvodní obrazovka, asynchronně získá data požadavkem a zobrazí

formátovaný HTML kód (Bibliography.tsx). Uživatelé s rolí administrátora mohou obsah stránky upravovat komponentou EditBibliography.tsx po kliknutí na tlačítko upravit.

4 Závěr

Tento dokument obsahuje dokumentaci k architektuře aplikace a také programátorskou dokumentaci. Kód projektu by měl být dostatečně okomentovaný a přehledně strukturovaný, aby při případném rozšíření nebo údržbě šla aplikace snadno vylepšit. Aplikace splňuje zadání od zákazníka.