STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Tempora - Stránka pro tvorbu interaktivních časových os

Jiří Petřík

Pražský kraj

Praha 2025

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Tempora - Stránka pro tvorbu interaktivních časových os
Tempora - Webside for creation of interactive timelines

Autor: Jiří Petřík

Škola: Gymnázium, Praha 6, Arabská 14, Arabská 682/14, 160 00, Praha 6 - Vokovice

Kraj: Pražský kraj

Konzultant: Mgr. Jan Lána

Praha 2025

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne ....

Poděkování

Chtěl bych poděkovat Vítězslavu Procházkovi za pomoc s výběrem technologií a přínosné konzultace a Ing. Ivu Petříkovi za zpětnou vazbu a návrhy na zlepšení projektu.

Anotace

Ve své práci SOČ jsem se zabýval vývojem webové stránky sloužící jako nástroj pro vytváření a vizualizaci interaktivních časových os. Cílem práce bylo umožnit uživatelům tvorbu, procházení a sdílení časových os s možností jejich rozkliknutí pro podrobnější informace o jednotlivých obdobích a autorech. Tyto osy se vyznačují důrazem na zasazení událostí do historického kontextu a umožňují srovnání dvou tématicky odlišných období.

Projekt byl naprogramován s využitím open-source frameworku Nuxt.js a databázové platformy Supabase.

Klíčová slova

Časová osa; webová stránka; učební pomůcka; historický kontext

Annotation

In my SPA work, I have been developing a website used as a tool for creating and visualizing interactive timelines. The aim of the work was to allow users to create, browse, and share timelines with the ability to click through for more detailed information about each period and author. These timelines are specific in that they emphasize placing events in a historical context and provide the ability to compare two thematically distinct periods.

The project was programmed using the open-source framework Nuxt.js and the database platform Supabase.

Keywords

Timeline; website; learning tool; historical context

Obsah

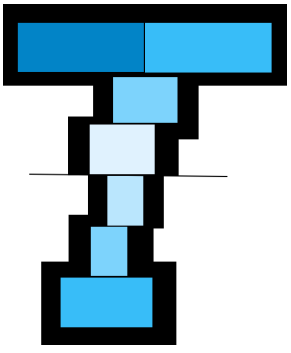
1 Úvod

Orientace v dějinách je často komplikovaná, zejména pokud se snažíme pochopit souvislosti mezi různými historickými událostmi, osobnostmi a směry. Setkáváme se s velkým objemem historických informací, letopočtů, významných událostí, konfliktů, ale i nových objevů a vynálezů, a proto je velmi obtížné získat ucelený přehled o zasazení jednotlivých událostí do dějin. Tradiční přístupy k výuce historie často pracují s oddělenými časovými liniemi napříč několika studijními předměty, což ztěžuje porovnání událostí probíhajících ve stejném období, ale v odlišných kontextech. V mém projektu se proto snažím vytvořit nástroj, který by tento problém pomohl uživatelům vyřešit.

Jméno Tempora pochází z množného čísla latinského slova tempus = čas. Logo projektu jsem sám vytvořil – znázorňuje jednotlivé úseky na časové ose a používám ho jako favicon projektu [1].

1.1 Cíl projektu

Cílem projektu Tempora je poskytnout uživatelům nástroj pro tvorbu a vizualizaci interaktivních časových os, který umožní jednoduché porovnávání různých témat v daném období, jejich zasazení do historického kontextu a přehledné zobrazení dat. Důležitou součástí projektu je vytvořit tuto osu interaktivní a uživatelsky příjemnou, s možností rozkliknutí jednotlivých období, ke kterým jsou vytvořeny podrobnější popisy, vysvětlivky nebo případně odkazy na další informace.



Obrázek 1: Logo Tempora

2 Použité technologie

Projekt je postaven na frameworku Nuxt [2], který je určený pro tvorbu webových aplikací a stránek pomocí Vue.js [3]. Nuxt byl vybrán díky své flexibilitě a funkcím, jako je automatické routování stránek, asynchronní data, lazy-loading, automatická importace modulů [4] a široký ekosystém modulů a podporovaných knihoven.

O databázi se stará služba Supabase [5], která využívá PostgreSQL (více v kapitole 4 *Backend*).

Pro design webové stránky jsem použil Tailwind CSS [6], který se řídí principem utility-first, používá intuitivní syntaxi a konzistentní barvy [7, 8]. Celý kód mého projektu jsem průběžně zálohoval pomocí Gitu na službu GitHub do repozitáře Tempora organizace gyarab [9].

3 Frontend

Webová aplikace je z velké části napsána ve Vue.js. Tato JavaScriptová knihovna pro tvorbu uživatelských rozhraní využívá komponentový přístup, který umožňuje rozdělit uživatelské rozhraní do logických a znovupoužitelných částí. To pomáhá udržet dobrou organizaci a architekturu projektu.

3.1 Vlastní komponenty

Jak již bylo zmíněno, Nuxt a Vue.js jsou založeny na znovupoužívání komponent pro zachování jednoduchosti na hlavních stránkách. Vytvořil jsem devět komponent, z nichž nejdůležitější je ***TimelineComp.vue***. Tento komponent v sobě ukrývá veškeré prvky týkající se přímo časové osy, jako jsou například CSS styly nebo ovládání osy.

Na stránce, kde se nachází časová osa, je také postranní menu vytvořené v ***SidebarComp.vue***. Obsahuje uživatelské funkce, jako je přepínání barevného režimu časové osy nebo sdílení odkazu na danou osu.

Dále existují komponenty, které se přepínají podle potřeby. Například když uživatel klikne na událost, zobrazí se ***ItemInfoComp.vue***, který poskytne informace. Pokud je však uživatel v editačním režimu, objeví se ***ItemEditComp.vue***, který umožní změnit informace o události. Podobným způsobem funguje i ***InfoComp.vue***, zobrazující informace o časové ose, a ***SettingsComp.vue***, který umožňuje upravovat hodnoty časové osy. Komponent ***CreateTimeline.vue*** se zobrazí pouze tehdy, když chce uživatel vytvořit novou časovou osu, a slouží k zadání potřebných informací.

Komponent ***LineCard.vue*** slouží jako šablona karty jednotlivých časových os, do níž se vkládají informace o autorovi nebo letech, které osa pokrývá.

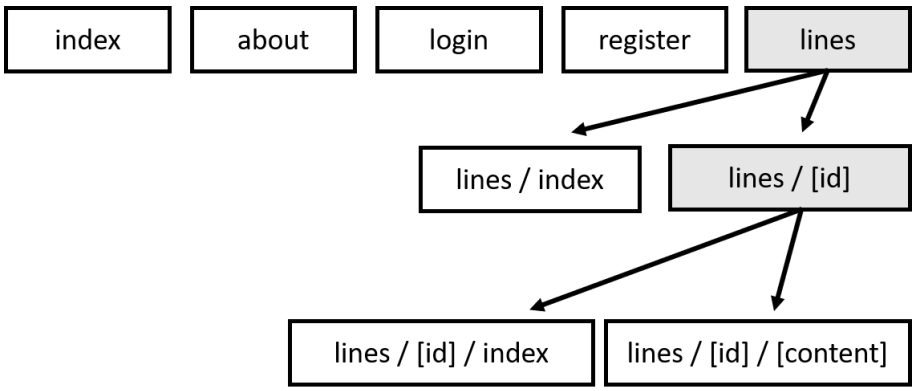
Poslední komponent je ***QuillEditor.vue*** z knihovny Quill [10], který umožňuje uživateli používat různé textové prvky.

3.2 Rozdělení stránek

V Nuxt jsou stránky definovány v adresáři *pages/*, kde název .vue souboru a složky/složek, ve kterých se nachází, určuje jeho URL adresu. Názvy souborů mohou být buď statické (např. pages/about, pages/login), nebo dynamické, které pracují s proměnnými, přičemž jejich jméno je v hranatých závorkách (např. pages/[id]: pokud zadáme adresu `lines/6`, zobrazí se stránka „[id].vue“ složky lines s proměnnou 6, se kterou lze dále pracovat pomocí Vue Routeru [2]). Základní stránka každé složky se vždy jmenuje *index.vue* a její název se nezobrazuje v URL. Nuxt také ve výchozím nastavení implementuje plně přizpůsobitelnou stránku *error.vue*, na kterou jste přesměrováni, pokud nastane nějaká fatální chyba.

Strukturu mé webové stránky lze rozdělit do tří vrstev. První vrstva zahrnuje přihlášení, registraci, domovskou stránku uživatele a stránku „O projektu“. Ve druhé vrstvě se nachází „Přehled časových os“ (*lines/index*), kde mohou uživatelé vyhledávat vytvořené osy, zobrazit si o nich informace, aniž by museli osu načíst, nebo vytvořit nové časové osy. Poslední, třetí vrstva obsahuje samotnou časovou osu (*lines/id/index*) a podrobnosti o jednotlivých událostech (*lines/id/content*).

Funkci dynamických stránek jsem ve svém projektu použil dvakrát – při výběru ID dané osy a při podrobném zobrazení údajů o události. Pokud by tedy uživatel chtěl zobrazit podrobnosti o události 17 v ose s ID 123456, výsledná URL adresa bude `/lines/123456/17`.



Obrázek 2: Rozdělení stránek ([ ] = dynamická stránka, šedá = složka)

Nuxt také používá funkci lazy-loading, která přednačítá stránky dosažitelné ze stránky aktuální pomocí speciálního odkazu NuxtLink. Tento přístup výrazně zrychluje načítání, například při rozkliknutí podrobností o události ze stránky s časovou osou.

3.3 Moduly a knihovny

Nuxt s Vue.js nabízí velkou škálu různých modulů, které přidávají novou funkcionalitu a rozšiřují možnosti. Jelikož jsem nikdy předtím s ekosystémem Nuxt nepracoval, na některá z těchto vylepšení jsem přicházel postupně v průběhu projektu [11, 12].

Jeden z nejužitečnějších modulů je NuxtUI [13], který přidává nejrůznější frontendové komponenty, jako jsou např. modal a toggle, ale zároveň poskytuje i vylepšené základní komponenty, například UButton, u kterého lze nastavit parametry pro ikonu, barvu či velikost. NuxtUI přidává také animovanou postranní lištu, ale tu jsem vytvořil ještě před instalací tohoto rozšíření [14].

Jednou z funkcí mé aplikace je, že v zobrazení detailu můžete vidět názvy sekcí, které následují a předcházejí zobrazené sekci, což pomáhá s orientací v období. Toto zajišťuje state management framework Pinia [15].

3.4 Modul a komponent časové osy

Nejdůležitější knihovna, která byla použita, je Vue Timeline Chart [16]. Hlavním důvodem, proč jsem si vybral právě tuto knihovnu, byla její flexibilita a dobře zpracovaná dokumentace. Základem této osy jsou řádky (timelineGroup) a události (timelineItem). Události mají několik údajů, jako je jméno, unikátní ID, start a end, udávané v milisekundách Unixového času (od 1. ledna 1970, 00:00:00 UTC – epoch). Základní časová osa není nijak komplikovaná, ale pokud chceme změnit její vzhled, musíme správně použít CSS a přepsat výchozí styly.

Mojí vizí bylo vytvořit dvě strany uprostřed rozdělené časovou osou, přičemž každá strana bude mít čtyři řádky s velikostí podle jejich důležitosti. Nahoře a dole bude kontext k období, dále „Hlavní skupina“, která obsahuje nadpis období, „Vedlejší skupina“, obsahující převážně autory z daného období, a nakonec „Detail skupina“, která slouží k zobrazení dalších informací, například děl daných autorů (tyto kategorie závisí na autorovi osy a může si je přizpůsobit).

Aby osa vypadala tak, jak jsem si představoval, hrálo klíčovou roli nacházení již zavedených CSS tříd, které knihovna používá, a přepisování předem nastavených stylů pomocí syntaxe !important [17], viz ukázka: 1.

Ukázka 1: TimelineComp.vue, Timestamps style

```
356 .timestamps {
357   transform: translateY(calc(var(--group-height) + var(--primaryGH) + var(--secondaryGH) +
          var(--detailGH)));
358   position: absolute;
359   width: 100%;
360   background-color: transparent !important;
361 }
362
363 .timestamps:before {
364   content: "";
365   position: absolute;
366   left: 0;
367   right: 0;
368   top: 50%;
369   border-top: 2px solid v-bind('timelineStyles.timestamp.color');
370   transform: translateY(-50%);
371 }
372
373 .timestamp {
374   color: v-bind('timelineStyles.timestamp.color');
375   position: relative;
376   background-color: v-bind('timelineStyles.timestamp.backgroundColor');
377   padding: 0 10px;
378   border-left: 0px !important;
379 }
```

4 Backend

V rámci projektu Tempora jsem zvolil Supabase jako backendové řešení pro správu databáze a autentizaci uživatelů. Supabase nabízí funkcionalitu podobnou Firebase s plnou podporou PostgreSQL, což mi umožňuje efektivně pracovat se strukturovanými daty. Supabase jsem si vybral převážně kvůli jednoduché implementaci do Nuxt frameworku a automatické údržbě databázového backendu, která eliminuje nutnost správy vlastních serverů, čímž výrazně urychlila vývoj projektu.

4.1 Logické soubory - JavaScript

I když JavaScript není úplně backendový jazyk, zmíním ho právě v této kapitole, protože až na výjimky obsluhuje operace, které nejsou uživatelem vidět v grafickém rozhraní. Veškeré JavaScriptové soubory mám ve složce composables.

UseSupabase.js - soubor, ve kterém jsou požadavky na databázi týkající se uživatelských dat, jako je jméno, uložené osy a také časové osy, jako je vytvoření časové osy, její aktualizace nebo smazání.

Druhým souborem pro obsluhu databáze je ***supabaseItem.js***. Zde jsou veškeré databázové dotazy týkající se jednotlivých událostí, od jejich načítání až po smazání. Tyto soubory využívají supabaseClient pro komunikaci s databází.

Aby se jednotlivé události správně ukládaly do databáze, je nejprve nutné je upravit do správného tvaru, převést roky na milisekundy atd. ***ItemManipulation.js*** zařizuje právě tyto úkony a rozhoduje podle parametrů, které dostane, do jakých řádků se musí jaké údaje uložit, aby časová osa vypadala tak, jak uživatel potřebuje.

U mé komponenty časové osy je nedílnou součástí ***timelineFeatures.js***. Tento soubor slouží k poskytnutí a aktualizaci správně zobrazených dat na časové ose a funkcionalitě jednotlivých ovládacích prvků, jako je přibližování, posouvání nebo zobrazovaný rok při najetí kurzoru nad časovou osu.

Posledním souborem je ***state.js***. Tento soubor byl původně vytvořen pouze k uchování stavu postranního menu, ale později do něj bylo začleněno i přepínání režimů pro nastavení, zobrazení a úpravy časové osy.

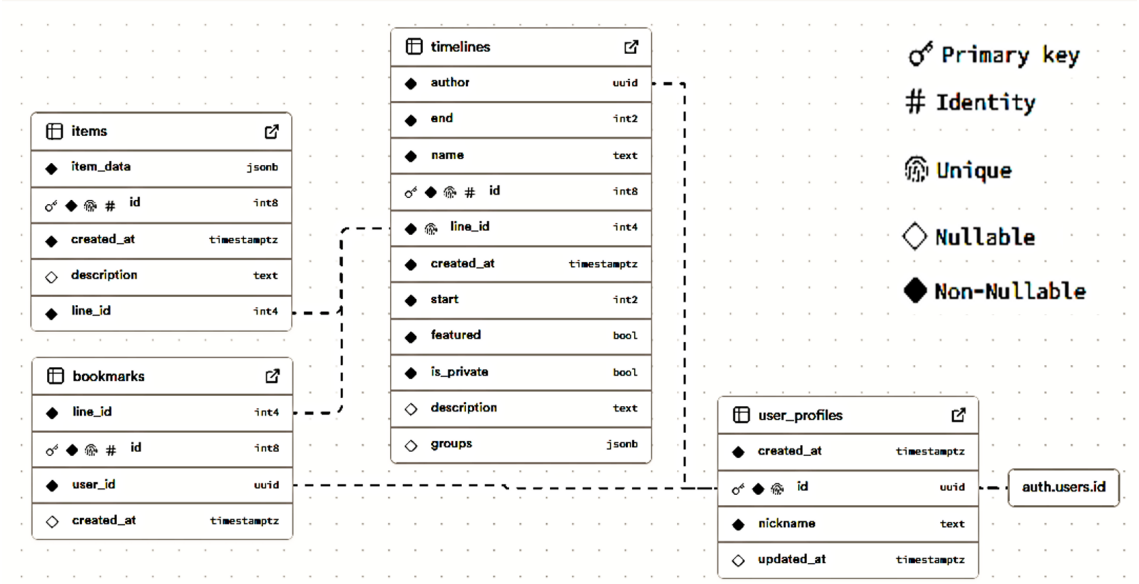
4.2 Ověření uživatele

Supabase používá k autentikaci základní tabulku auth.users, kterou automaticky spravuje. Do této tabulky se vždy po registraci s e-mailem a heslem přidá nový uživatel a na jeho e-mailovou adresu se odešle automaticky generovaný odkaz pro potvrzení.

V aplikaci jsem chtěl mít registraci a přihlášení nepovinné. Uživatel tedy nepotřebuje účet, aby procházel veřejné osy, pouze mu chybí možnost tvorby nových os a ukládání svých oblíbených časových os. [18, 19, 20]

4.3 Databázová struktura

Pro práci se všemi daty využívám čtyři SQL tabulky: timelines, items, user_profiles, bookmarks a jednu předgenerovanou auth.users. Tyto tabulky jsou vzájemně propojené a interagují s webovou stránkou, viz obrázek 3.



Obrázek 3: Struktura tabulek z vizualizace v Supabase, legenda značek

4.3.1 Tabulka – timelines

V centru celého systému stojí tabulka timelines, ve které jsou uložena všechna data, která se samotné časové osy týkají. Hlavními částmi jsou id, name – jméno dané časové osy, start a end, určující počátek a konec zobrazovaného období, line_id, obsahující šestimístné číslo, které určuje odkaz na danou osu a zároveň slouží k propojení s dalšími tabulkami.

Featured je booleanová proměnná, která určuje, zda se daná osa zobrazí pro všechny uživatele na stránce v sekci „Vybrané“. Is_private si nastavuje každý uživatel sám – určuje možnost návštěv autorovi osy jinými uživateli, více v sekci 4.4 *Tabulková pravidla (policies)*. Uživatel si v časové ose také může nastavit jména jednotlivých řádků, což se ukládá ve formátu jsonb [21] do groups, a vlastní popis osy, který je uložen v description.

4.3.2 Tabulka – items

Tabulka items zabezpečuje správné uložení jednotlivých událostí. Je propojená s tabulkou timelines pomocí cizího klíče line_id. Obsahuje údaj o vytvoření a detailnější popis dané události, který má v databázi formát obyčejného textu, i když jsou v něm uloženy prvky v HTML syntaxi (`<h3>Nadpis </h3>`).

Nejsložitější strukturou je pak určitě jsonb item_data, který obsahuje všechny potřebné informace vyžadované knihovnou vue-timeline-chart ke správnému umístění na osu, viz ukázka 2. Události jsem rozdělil na dva typy – kontextové a normální. Normální událost se skládá ze tří řádků: hlavní části, vedlejší a detailu (korespondující s označením řádků v kapitole 3.4). Bohužel knihovna, kterou jsem využíval, neumožňuje přidat jednu událost na více řádků, proto jsem normální událost musel rozdělit do tří částí. Každý předmět má své unikátní id, začátek a konec, jméno, které se zobrazuje v časové ose, barvu nastavenou jako cssVariable, group označující řádek, kam bude umístěn, a poslední tag, podle kterého se sjednocují zmíněné tři části normální události. Kontextová událost je velice intuitivní, skládá se jen z jedné hlavní části a má všechny zmíněné proměnné stejné, jen tag je u ní vždy roven id.

Ukázka 2: item_data, Supabase items table

```
{
  "id": 35,
  "end": -473385600000,
  "tag": 35,
  "name": "Osvicenstvi v literature",
  "group": 5,
  "start": -631152000000,
  "cssVariables": {
    "--item-background": "#073E5A"
  }
}
```

4.3.3 Tabulka – user_profile

Tato tabulka je propojená se základní tabulkou auth.users.id a stará se o ukládání uživatelského nastavení. Protože jsem nechtěl, aby si jiní uživatelé mohli zobrazit váš e-mail, vytvořil jsem možnost nastavit si vlastní uživatelské jméno (nickname).

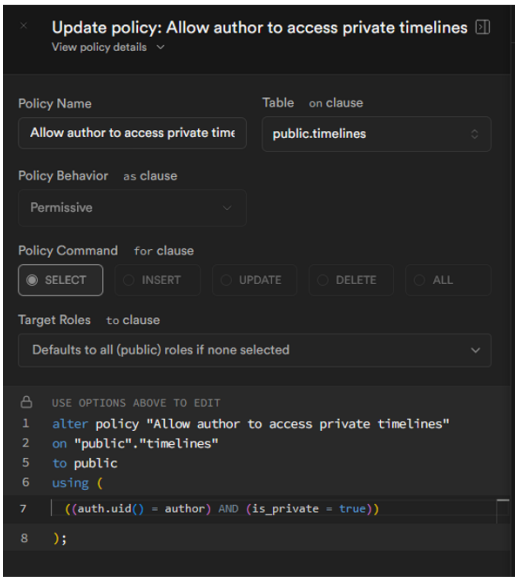
4.3.4 Tabulka – bookmarks

Pro zlepšení uživatelského prostředí byla vytvořena tabulka bookmarks, která přidává uživateli funkci uložit si časové osy, které se mu líbily, a umožní mu k nim snadný přístup v sekci „Uložené“. Jsou v ní uložena ID časových os a UUID uživatelů.

4.4 Tabulková pravidla (policies)

U všech svých tabulek mám zapnuté RLS neboli row-level security, které slouží k řízení přístupu k datům na úrovni jednotlivých řádků v tabulkách. Při použití RLS je důležité pečlivě nastavit příslušná pravidla (policies), která definují, kteří uživatelé nebo role mají jaká oprávnění pro čtení, vkládání, aktualizaci a mazání dat. Tato pravidla jsou definována pomocí SQL funkcí, které vracejí logickou hodnotu true nebo false v závislosti na kontextu aktuálního uživatele. V Supabase lze tato pravidla nastavovat přímo v uživatelském rozhraní.

Například pro tabulku timeline mám nastavená pravidla: přidat novou osu může jen přihlášený uživatel, aktualizovat a mazat data může pouze autor dané časové osy. Pro zobrazení dat mám dvě pravidla – pokud je osa veřejná, nemusím kontrolovat nic. Pokud je ale osa soukromá, je nutné zjistit, zda je daný uživatel autorem, a na základě toho data buď zobrazit, nebo ne.



Obrázek 4: Ukázka nastavení pravidla tabulky v Supabase

Zprvu jsem měl tuto ochranu jen na tabulku timelines, takže pokud by znal nepřihlášený uživatel id nějaké události v soukromé ose, mohl si ji zobrazit, i když k ose jako takové přístup neměl. Tento problém jsem samozřejmě opravil.

4.5 Práce s daty v databázi

Díky ekosystému Nuxt je práce mezi programem a databází víceméně přímočará. Každá funkce v *useSupabase.js* a *supabaseItem.js* používá pro komunikaci s databází asynchronní funkci useSupabaseClient(), do které se zadají požadované SQL příkazy. Každá funkce má také návratovou hodnotu – ta může obsahovat samotná data při načítací funkci nebo odpověď databáze, buď o provedení, nebo o chybě, která nastala.

4.6 Ukládání událostí

Ukládání událostí je asi nejtěžší operace, při které záleží na různých parametrech, aby fungovala správně. Celý tento cyklus začíná v *itemEditComp.vue*, kde se po zmáčknutí tlačítka „Uložit změny“ zavolá funkce saveChanges, která zkontroluje správnost zadaných roků, a pokud je vše správně, zavolá funkci handleItemUpdate z *itemManipulation.js* se všemi parametry. Tato funkce se skládá ze tří částí: ukládání nově vytvořených částí, aktualizace již vytvořených a smazání nepotřebných událostí. Funkce se rozhoduje na základě parametrů od uživatele v porovnání s daty načtenými z databáze. Například pokud normální událost již měla v databázi uloženou hlavní a sekundární část a uživatel sekundární smazal, funkce zavolá *updateItem*, aktualizuje hlavní část a sekundární smaže pomocí *removeItem* z *supabaseItem.js*.

Při tvorbě nové části události je důležité podívat se na poslední použité ID a vytvořit další tak, aby se žádné neopakovalo. Dalším úskalím při tvorbě nové události je její uložení do správného řádku na základě parametrů contextType a isBottom, což řeší tato část kódu – ukázka 3.

Ukázka 3: Handle update, itemManipulation.js

```
154 const mainGroup = contextType
155   ? isBottom
156     ? 8 : 1
157   : isBottom
158     ? 5 : 2;
159
160   const secondaryGroup = isBottom ? 6 : 3;
161   const detailGroup = isBottom ? 7 : 4;
```

Při tvorbě úplně nové události z postranního menu v *SidebarComb.vue* používám stejnou funkci saveChanges, pouze je nastaven parametr pomocí query z přesměrování: `const creatingNew = ref(route.query.creatingNew === 'true');`.

5 Řešení problémů

V projektu jsem se několikrát zasekl kvůli často zbytečným chybám, ale i různým faktorům mimo moji kontrolu, a tak bych tady chtěl rozebrat pár problémů, se kterými jsem se setkal.

5.1 Převádění času – Unixový čas

5.1.1 Data před rokem 1970

Na začátku projektu, když jsem se teprve učil pracovat s *vue-timeline-chart*, jsem nastavil rozpětí let časové osy na 1950 až 2000. Po několika dnech od implementace jsem si ale všiml, že zobrazovaná část je pouze od roku 1970 do 2000. Bohužel jsem ve stejnou dobu pracoval na ovládání této osy, a tak jsem se snažil objevit chybu v posouvání zobrazené části a přibližování, která by mohla za useknutí dvaceti let. Asi týden jsem hledal tuto neexistující chybu, než jsem přišel na to, že rok 1950 je před začátkem unixového času. To znamená, že počet milisekund by měl být záporný.

5.1.2 Data mezi lety 0 až 100

Pro přeměnu milisekund na pro lidi příjemnější roky jsem využíval knihovní funkci *Date.UTC(year, 0, 1)*. Tato funkce se zdála být více než dostačující pro převody let na milisekundy a naopak – až do chvíle, kdy jsem se snažil změnit roky před Kristem z formátu -50 až 50 na 50 BC až 50. Všiml jsem si totiž, že místo očekávaného roku 50 byl načten rok 1950.

Pro kontrolu jsem používal online nástroj *Epoch Converter* [22], který ovšem nejspíše využívá stejnou metodu. Dlouho jsem přemýšlel, kde nastala chyba, než jsem otevřel oficiální dokumentaci [23] a zjistil, že právě pro roky 0 až 100 jsou jejich hodnoty nastaveny jako roky 1900 až 2000. V dokumentaci jsem také hledal maximální a minimální hodnoty, které lze zadat do parametru roku, ale ty jsem nenašel. Metodou pokus-omyl jsem tedy dospěl k minimální hodnotě roku -271 820 a maximální 275 760. Tyto hodnoty ale nejsou plně podporovány komponentou časové osy, a tak jsou pro uživatelské zadávání povoleny hodnoty v rozmezí -12 000 až +12 000 let.

Finální funkce pro převod z roku na milisekundy od epochy je uvedena v ukázce 4 [24].

Ukázka 4: itemManipulation.js, convertYearToMs

```
export function convertYearToMs (year) {
  if (year >= 100 || year < 0) {
    return Date.UTC(year, 0, 1);
  } else {
    // For years before 100, we need to use a different approach
    const baseYear = year >= 0 ? 1900 : 2000;
    const yearDiff = Math.abs(year - baseYear);
    return -(-Date.UTC(baseYear, 0, 1) + (yearDiff * 31536000000));
  }
};
```

5.1.3 Načtení dat roku 1970

Poslední chyba, na kterou jsem narazil a která se vztahuje k převodům času, byl samotný rok 1970. Tento rok je totiž v milisekundách nula. Chyba nastala při načítání dat z databáze s nulovou milisekundovou hodnotou, kterou si můj program vyložil jako základní hodnotu nezadané položky – NaN (Not a Number) – a při převodu použil základní hodnotu 0 místo 1970. Oprava této chyby netrvala dlouho, stačilo přidat podmínku: pokud je hodnota NaN, nastav ji na 1970, viz [ukázka: 5].

Ukázka 5: itemManipulation.js, loadItemData

```
// ...
start: new Date(contextItem?.start || regularItem?.start).getFullYear() || 1970
// ...
```

5.2 Předání informace nadřazenému souboru

Jelikož ve svém projektu používám podmíněné renderování, které umožňuje zobrazit nebo skrýt různé komponenty, často mám nastavené základní rozvržení pozadí mimo danou komponentu. Někdy ale potřebuji předat informaci nadřazenému *.vue* souboru, aby změnil stav nebo své vlastnosti. Přesně toto jsem potřeboval pro dynamické nastavení barvy z komponenty *ItemEditComp.vue* na pozadí nadřazené stránky. Proto jsem použil funkci *emit*.

Jako první musím nastavit konstantu *emit* s daným jménem funkce, kterou chci provádět v nadřazeném komponentu. Poté nastavím pozorovatele (*watcher*), který při zaznamenání změny hodnoty *selectedColor* vyšle signál *emit* s novou barvou.

Nadřazený komponent poslouchá *event handler* (@update-background=”updateBackgroundColor”), a pokud takovou změnu objeví, provede se část kódu se jménem této změny. Funkce *updateBackgroundColor* pak změní barvu pozadí v `div` díky reaktivnímu atributu `:style`.

Ukázka 6: Update background; itemEditComp.vue, content.vue

```
1
2  // 1. Define emit
3  const emit = defineEmits(['update-background']);
4
5  // 2. Watch for color changes and emit them
6  watch(selectedColor, (newColor) => {
7    emit('update-background', newColor);
8  });
9
10
11 ////// Parent component ///////
12
13 <template>
14     <div class="content\_box h-full w-full mx-10 px-4 md:px-6 relative "
15             :style="{ backgroundColor: backgroundColor }">
16   <!-- Listen for emitted event -->
17     <ItemEditComp v-if="inEdit" @update-background="updateBackgroundColor" />
18     <ItemInfoComp v-if="!inEdit" @colorSelected="updateBackgroundColor" />
19 </template>
20
21 <script setup>
22 const backgroundColor = ref('#BAE6FD'); // Default color
23
24 // Handle the emitted event
25 function updateBackgroundColor(newColor) {
26   backgroundColor.value = newColor;
27 }
28 </script>
```

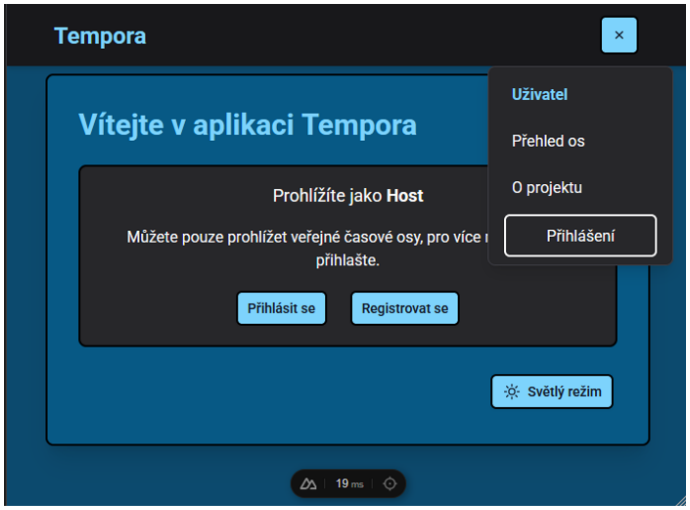
Tento přístup zajišťuje konzistentní komunikaci mezi nadřazeným a podřazeným prvkem a umožňuje psát znovu použitelný kód. Ve zmíněném souboru je totiž ještě druhý *event handler*, který místo na uživatelem vybranou barvu čeká na barvu načtenou z databáze skrze komponentu *ItemInfoComp.vue*.

6 Uživatelské rozhraní

V této kapitole se pokusím vysvětlit, jak používat aplikaci, jak se ovládá a co se s ní dá dělat. Dále uvedu příklady mnou vytvořených časových os.

6.1 Navigace na stránce

Stránka používá k navigaci horní lištu se záložkami jednotlivých sekcí stránek (tento prvek je uložen jako *default.vue* ve složce *layout*). Na menších zařízeních se místo záložek objeví rozklikávací menu pro výběr jednotlivých stránek.



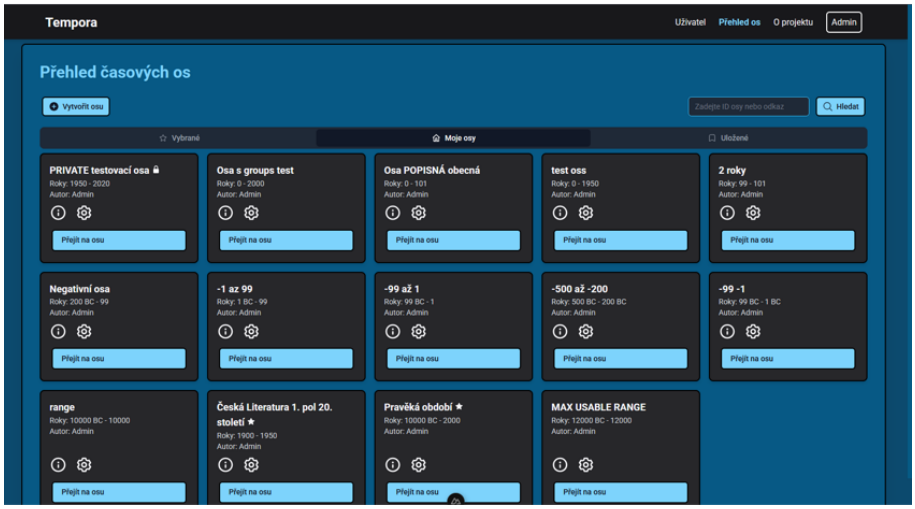
Obrázek 5: Mobilní rozhraní ve tmavém režimu

6.2 Tvorba osy

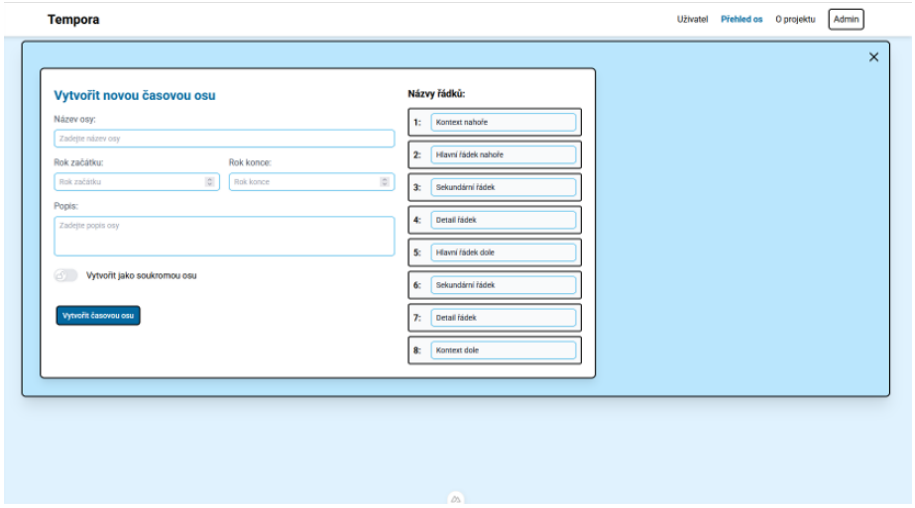
Pokud je uživatel registrován a přihlášen, může vytvořit vlastní časovou osu. Nejprve se musí přesunout do sekce „Přehled os“, kde se nachází vybrané osy, osy vytvořené uživatelem a uložené osy, viz obrázek 6. V této sekci lze také vyhledávat osy podle jejich ID nebo celého odkazu.

Po kliknutí na tlačítko „Vytvořit osu“ se otevře nová nabídka, kam uživatel zadává jméno časové osy, rok začátku a konce popisovaného období (pro zadání let před naším letopočtem se používá znak mínus před počtem let) a krátký popis dané osy, který může obsahovat například zdroje, odkud čerpal při tvorbě.

Na pravé straně je možnost pojmenovat jednotlivé řádky. Také lze nastavit osu jako soukromou, což znemožní komukoliv kromě autora si tuto osu zobrazit.



Obrázek 6: Stránka Přehled os (tmavý režim)



Obrázek 7: Tvorba nové osy (světlý režim)

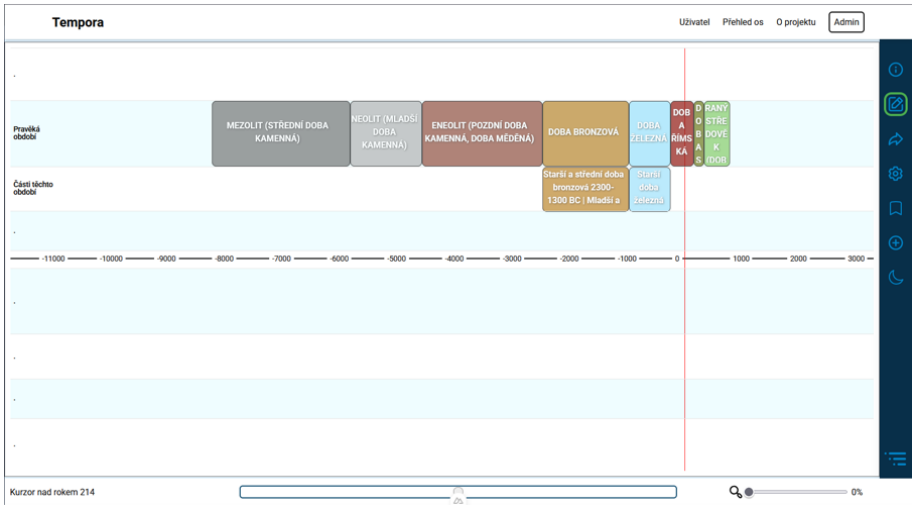
6.3 Ovládání osy

Po vytvoření osy je uživatel přesunut přímo na stránku, kde se daná osa zobrazuje. Zde se nachází časová osa, uprostřed níž jsou uváděny letopočty. Pod osou je ovládací panel, pomocí kterého se osa posouvá, přibližuje nebo oddaluje. Nachází se zde také prvek, který ukazuje letopočet odpovídající pozici červené čáry a kurzoru umístěného na časové ose pro lepší orientaci.

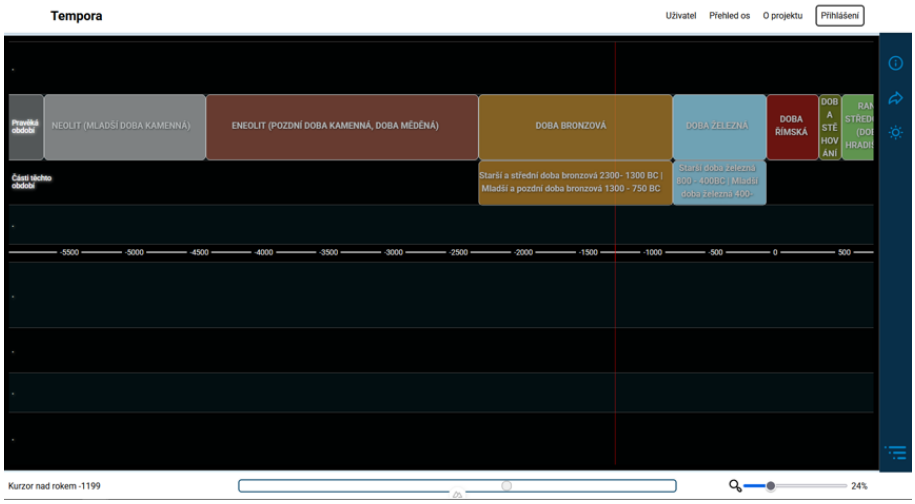
Klíčovou roli v ovládání má postranní lišta napravo, obsahující tlačítka: informace o ose, zapnutí editačního režimu, sdílení, nastavení, uložení, přidání nové události a přepnutí světlého/tmavého režimu.

Na obrázku 8 je zapnutý editační režim, což znamená, že po kliknutí na jednotlivé události je můžete upravovat. Zároveň se objeví nové tlačítko pro přidání události (více v sekci 6.4 Úprava událostí). Tlačítko pro přepnutí časové osy do tmavého/světlého režimu funguje nezávisle na režimu celé aplikace a umožňuje uživateli zachovat jeho preferovaný režim, přičemž osu zobrazí v režimu, který se mu bude zdát příjemnější.

Celou boční lištu lze také sbalit, aby nepřekážela. Pokud si prohlížíte osu, jejímž autorem nejste, uvidíte boční lištu tak, jak je zobrazena na obrázku 9.



Obrázek 8: Zobrazení postranní lišty - autor (světlí režim osy)

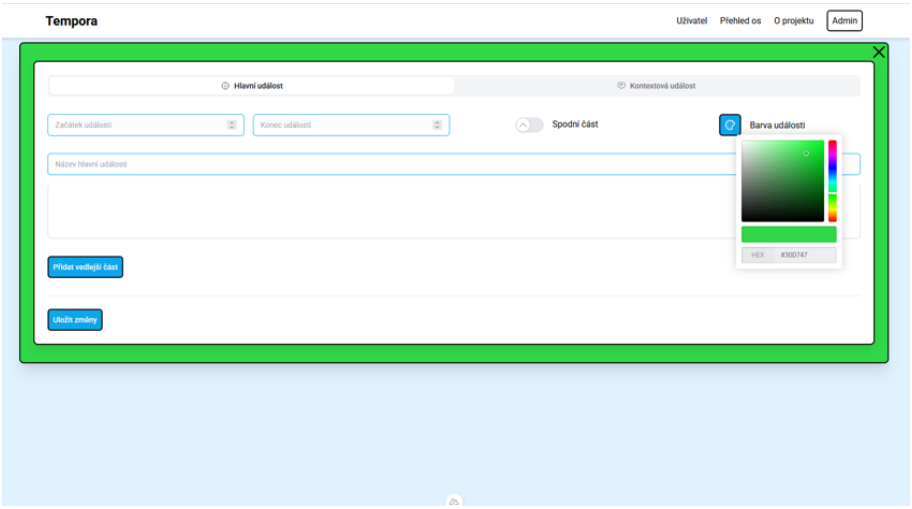


Obrázek 9: Zobrazení postranní lišty - návštěvník (tmavý režim osy)

6.4 Přidání a úprava události

Pro přidání nové události musí být autor dané osy v editačním režimu a kliknout na tlačítko „Přidat událost“ (+), což ho přesměruje na URL nové události, kterou může začít upravovat, viz obrázek 10.

V horní části si může vybrat, jestli chce vytvořit kontextovou nebo hlavní událost. Obě tyto části potřebují zadat roky trvání, aby se mohly na časovou osu umístit, a jejich zobrazované jméno, případně popis. Dále si také může vybrat, jakou barvu bude událost mít na časové ose – tato barva dynamicky mění okraj události. Pokud si uživatel zvolí hlavní událost, dostane možnost přidání vedlejší části a části detailu. Každá část umožňuje použít panel nástrojů pro editaci podrobností textu, odkazů, vzorců nebo zarovnání paragrafu.



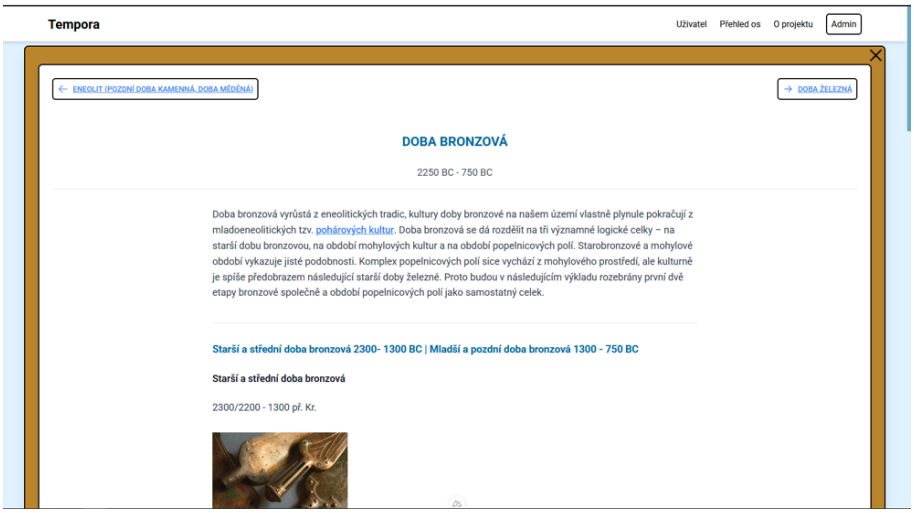
Obrázek 10: Tvorba nové události

Pokud chce uživatel upravit již vytvořenou událost, musí mít zapnutý editační režim a kliknout na událost na časové ose, kterou chce změnit. Otevře se mu stejné menu jako při tvorbě nové události, ale s načteným obsahem z databáze. Jedinou změnou je, že si nevybírá mezi hlavní a kontextovou událostí a může zahodit změny, které provedl, nebo smazat celou událost.

6.5 Zobrazení události

Pokud si chce uživatel prohlédnout podrobnosti o události, musí vypnout editační režim a kliknout na danou událost na časové ose. Poté se zobrazí stránka, na které jsou části rozděleny přesně podle toho, jak byly vytvořeny – s hlavním nadpisem a roky trvání. Pokud je vytvořeno více hlavních událostí v tomto řádku, objeví se nahoře také šipky pro přesměrování, ukazující, které období předcházelo a které následuje.

Obrázek 11: Úprava již vytvořené události



Obrázek 12: Ukázka zobrazení podrobností o události

6.6 Vylepšení uživatelské zkušenosti

Jedním z mých hlavních cílů bylo vytvořit uživatelsky příjemné a intuitivní prostředí. Proto jsem se již od začátku snažil tvořit prvky pro zpětnou vazbu – tlačítka změní barvu, když na ně najedete kurzorem, při načítání dat z databáze je vypsán aktuální stav toho, co se právě děje, a smazání údajů se nestane omylem díky potvrzovacímu dialogu. Celá webová aplikace podporuje světlý i tmavý režim (pomocí *Tailwind* `dark:`), a časová osa má svůj vlastní přepínač, aby se zabránilo špatně vypadajícím barvám v jednom z režimů.

Aplikace často využívá ikony z knihovny *Iconify* [25] pro zlepšení přehlednosti. Pokud jsou tyto ikony použity místo tlačítek, vždy se při najetí kurzorem zobrazí popisek vysvětlující jejich funkci.

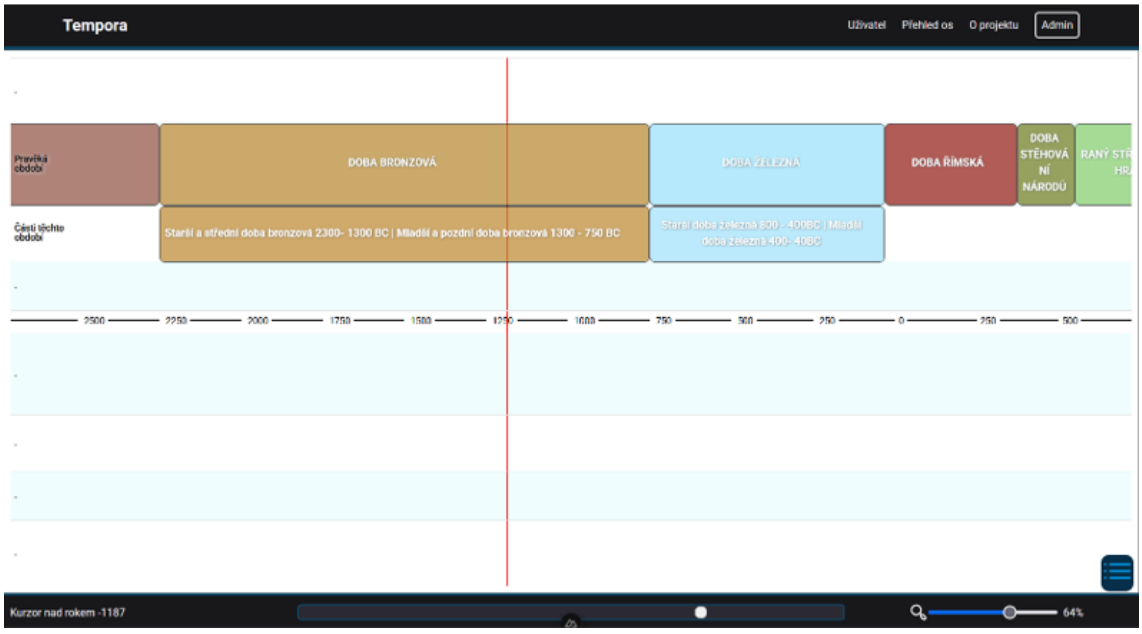
Uživatel si při tvorbě událostí a podrobností může sám zvolit jednotlivé barvy událostí díky komponentě [26], a při tvorbě textu může využít prvky jako odrážky, tučné písmo, podtržení, vložení odkazu a mnoho dalších díky „HTML RichText“ z knihovny *Quill* [10]. V celém projektu používám font *Google Roboto* [27].

Aplikace by měla být plně responzivní a použitelná i na mobilu díky *Tailwind* prefixům `md:`, `lg:`, i když hlavním záměrem byla verze pro počítač.

6.7 Ukázka hotových os

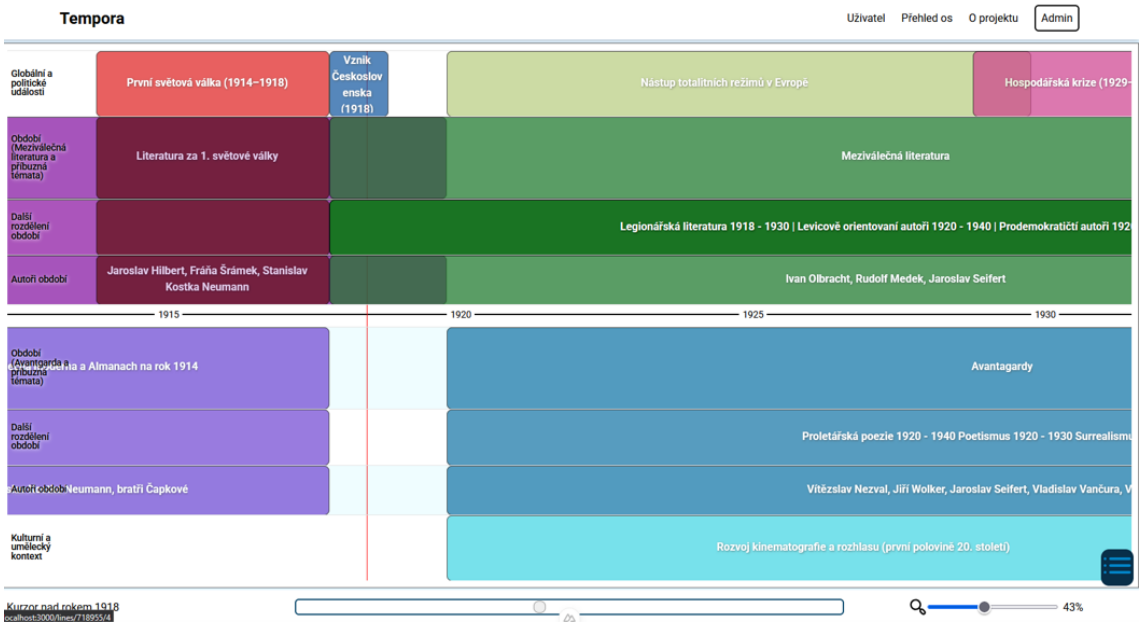
Pro tento projekt jsem vytvořil dvě ukázkové osy. Pro ukázku širokého rozpětí osy „Pravěká období“, která začíná 8000 let před naším letopočtem a končí rokem 1000, a pro ukázku použití kontextu a událostí v kratší časové době jsem vytvořil osu s názvem „Česká literatura první poloviny dvacátého století“.

Při tvorbě ukázkových časových os jsem použil stránky Národního muzea [28] a Wikipedie [29]. Tyto časové osy nemusí být stoprocentně přesné a slouží pouze k demonstraci použití mé webové aplikace.

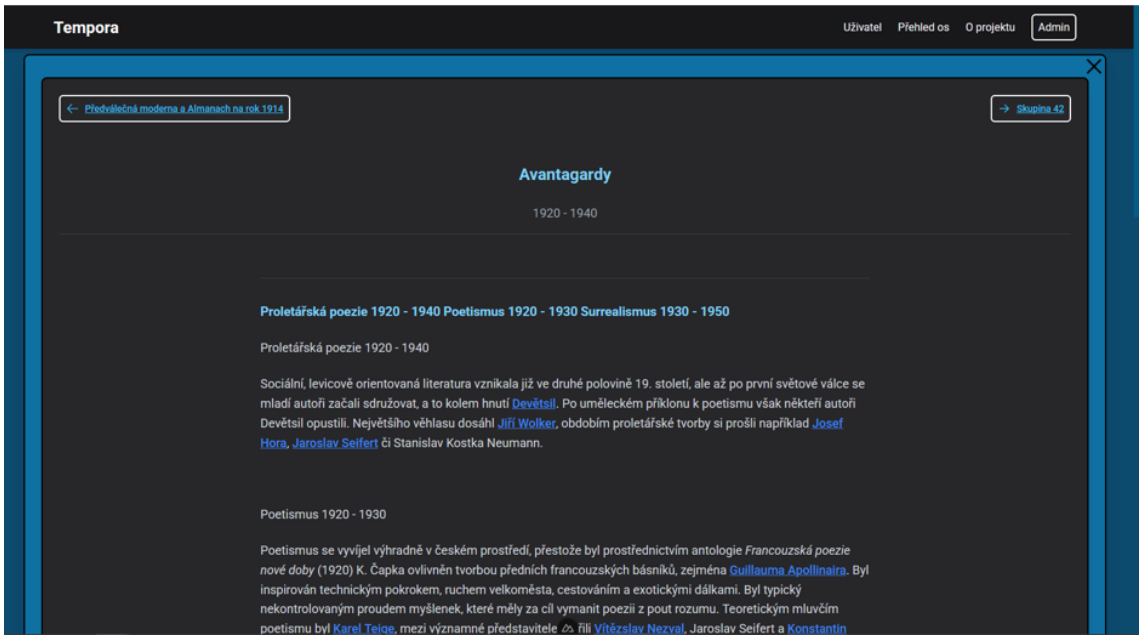


Obrázek 13: Ukázka zobrazení přiblížené Pravěké osy

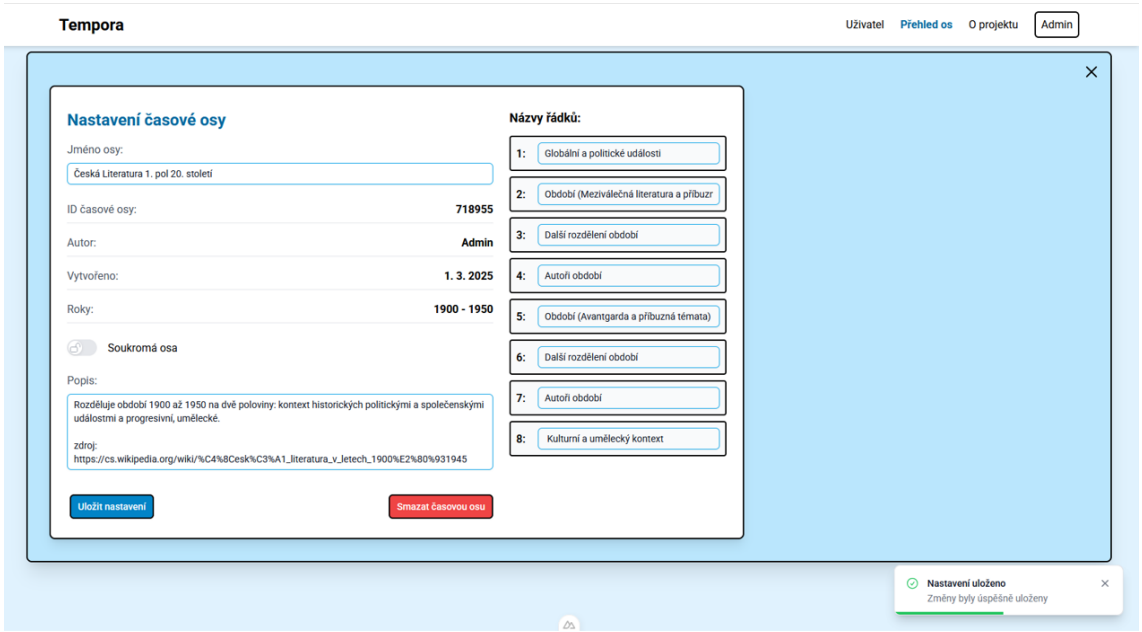
Obrázek 14: Ukázka oddálené časové osy Literatura 20. století



Obrázek 15: Ukázka přiblížené časové osy Literatura 20. století



Obrázek 16: Ukázka zobrazení podrobností Avantgardy (tmavý režim)



Obrázek 17: Ukázka možností nastavení časové osy s notifikací o uložení

7 Možná budoucí vylepšení

Jako téměř jakýkoliv projekt ani tento není naprosto perfektní a vždy je možné něco zlepšovat. Proto bych tuto kapitolu chtěl věnovat možným budoucím vylepšením, která nejsou implementována, ale mohla by projekt posunout dále.

7.1 Uživatelská vylepšení

7.1.1 Kategorie

Jedna z věcí, která je v mé stávající verzi obtížná, je možnost objevovat nové osy jiných uživatelů. Nyní je sice možné sdílet osy pomocí odkazu nebo ID, ale při nasazení aplikace pro veřejnost nastává problém s tím, jak tyto osy efektivně vyhledávat podle toho, co mě zajímá nebo co se mi líbí. Nebylo by tedy špatné implementovat ověřený systém kategorií nebo hashtagů s různými tématy, podle kterých by se daly jednotlivé osy vyhledávat.

7.1.2 Hodnocení

Dále, pokud bych přidal vyhledávání veřejných os, nabízí se vytvořit uživatelské hodnocení, které by mohlo fungovat podobně jako např. na platformě Reddit, kde uživatelé hodnotí buď kladně („up-vote“), nebo záporně („down-vote“).

7.1.3 Admin role

Zlepšit by se dala i funkce vybraných os. Tuto funkci jsem přidal hlavně z důvodu zobrazení alespoň nějakého úvodního obsahu pro nepřihlášené uživatele. Tento koncept by se ale mohl posunout dále a vytvořit roli admina, který může ocenit správně udělané časové osy, jež mají velké množství kladných bodů, a napevno je umístit do sekce „Vybrané“ přímo z UI aplikace, a ne přes editaci tabulky skrze SQL příkaz.

7.2 Další návrhy

7.2.1 Překlad

Když jsem s projektem začínal a ještě nevěděl, jak se co bude jmenovat, náhodně jsem používal česko-anglické názvy. Později, když už jsem získal představu, jak chci, aby aplikace vypadala, jsem si musel vybrat, jestli ji nechat celou v angličtině, nebo celou v češtině. Zvolil jsem češtinu, ale nebylo by špatné, pokud by se aplikace více rozšířila, nechat uživatele vybrat si svůj jazyk.

7.2.2 Shrnutí období

Zajímavá funkce, která mě napadla, by bylo shrnutí roku. Tato funkce by umožnila uživateli vybrat nějaké období a poté by z dat vybraných os vrátila vše, co se v daném roce stalo. Tato možnost je ale pouze návrh, protože by velice záleželo na implementaci, např. z jakých os by se data brala, kolik dat vzít nebo jak zajistit, že uživatel dostane relevantní informace.

8 Závěr

Když jsem s prací začínal, nevěděl jsem o frameworku Nuxt téměř nic, ale postupnou prací na projektu jsem se mnoho naučil. Kupříkladu, jak pracovat s databází, vytvořit přehlednou strukturu projektu nebo jak vytvářet uživatelsky příjemné prostředí.

Důležité při projektu tohoto rozsahu bylo dobře si rozdělit čas, pracovat na projektu průběžně a organizovat plán, jak implementovat jednotlivé funkce, sepisovat si nápady a zaznamenat si, pokud nějaká část aplikace nefunguje správně.

Povedlo se mi tedy vytvořit interaktivní webovou aplikaci pro vlastní tvorbu a prohlížení časových os, která dokáže přehledně vizualizovat daná období a zobrazit o nich podrobnější informace, umožnit porovnání tematicky odlišných událostí v jedné době nebo přidat potřebný historický kontext.

Práce tedy splnila mé očekávání, i když je vždy co dodělávat, a jak jsem zmínil, existují další nápady na nové funkce a vylepšení, na kterých bych určitě chtěl pracovat i v budoucnu.

9 Použité zdroje

[1] *How to add a Favicon to your Nuxt 3 project - Favicon.im blog*. 13. srp. 2024. URL: https://favicon.im/blog/add-favicon-to-nuxt3-project.

[2] *NUXT: The Intuitive Vue Framework*. URL: https://nuxt.com/.

[3] *Vue.js*. URL: https://vuejs.org/guide/essentials/application.html.

[4] *How do I fix "the requested module does not provide an export named 'default'"?* URL: https://stackoverflow.com/questions/70475635/how-do-i-fix-the-requested-module-does-not-provide-an-export-named-default.

[5] *Supabase — the open source Firebase alternative*. URL: https://supabase.com/.

[6] *Tailwind CSS*. URL: https://tailwindcss.com/docs/.

[7] *Colors - Core concepts*. URL: https://tailwindcss.com/docs/colors.

[8] *Tailwind Color Palette*. URL: https://tailwindcolor.com/.

[9] Gyarab Jiří Petřík. *GitHub - gyarab/2024-4e-petrik-Tempora: Tempora - nástroj pro tvorbu a vizualizaci interaktivních časových os*. URL: https://github.com/gyarab/2024-4e-petrik-Tempora.

[10] *Formats - Quill Rich Text Editor*. URL: https://quilljs.com/docs/formats.

[11] LearnVue. *8 libraries I use on EVERY project*. YouTube video. 10. říj. 2024. URL: https://www.youtube.com/watch?v=_GrdYoO3h0g.

[12] Net Ninja. *Nuxt 3 Crash Course 1 - What is Nuxt?* YouTube video. 31. říj. 2022. URL: https://www.youtube.com/watch?v=GBdO5myZNsQ.

[13] *Notification - Nuxt UI*. URL: https://ui.nuxt.com/components/notification.

[14] Justin Brooks. *Animated Navbar using CSS, Vue and Vue Router*. YouTube video. 31. květ. 2021. URL: https://www.youtube.com/watch?v=CfTvye31Ad0.

[15] *Pinia — The intuitive store for Vue.js*. URL: https://pinia.vuejs.org/.

[16] *Types — Vue Timeline chart*. URL: https://laurens94.github.io/vue-timeline-chart/reference/types.html.

[17] *How to overwrite css style*. URL: https://stackoverflow.com/questions/13117126/how-to-overwrite-css-style.

[18] John Komarnicki. *Easily add authentication with NUXT 3 + Supabase*. YouTube video. 14. čvn. 2023. URL: https://www.youtube.com/watch?v=yO-JMkjc4oo.

[19] AmritKhalsa. *Supabase redirecting all requests to /login Issue 16551 supabase/supabase*. URL: https://github.com/supabase/supabase/issues/16551.

[20] *ChatGPT - guest and logged-in access*. URL: https://chatgpt.com/share/6748872c-7bf8-800f-a04f-bde2dd7dbda8.

[21] Antonello Zanini. *JSON vs JSONB - A complete comparison*. 3. ún. 2025. URL: https://www.dbvis.com/thetable/json-vs-jsonb-in-postgresql-a-complete-comparison/.

[22] *Epoch Converter*. URL: https://www.epochconverter.com/.

[23] *Date.UTC() - JavaScript — MDN*. 11. ún. 2025. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/UTC..

[24] *Date.UTC() for years between 0-100*. URL: https://stackoverflow.com/questions/7766122/date-utc-for-years-between-0-100.

[25] *Icônes*. URL: https://icones.js.org/collection/heroicons.

[26] *Nuxt-Color-Picker · Nuxt Modules*. URL: https://nuxt.com/modules/nuxt-color-picker.

[27] *Roboto - Google Fonts*. URL: https://fonts.google.com/specimen/Roboto.

[28] Národní muzeum. *Přehled pravěkých období na našem území*. URL: https://www.archeologienadosah.cz/o-archeologii/chronologie/prehled-pravekych-obdobi-na-nasem-uzemi/.

[29] Přispěvatelé projektů Wikimedia. *Česká literatura v letech 1900–1945*. 7. led. 2024. URL: https://cs.wikipedia.org/wiki/%C4%8Cesk%C3%A1_literatura_v_letech_1900%E2%80%931945.

Seznam obrázků

Seznam ukázek kódu