Gymnázium, Praha 6, Arabská 14

Programování

ROČNÍKOVÝ PROJEKT

2024

Matěj Kratochvíl

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

ROČNÍKOVÝ PROJEKT

Předmět: Programování
Téma: Sociální Síť

Autor: Matěj Kratochvíl
Třída: 4.E
Školní rok: 2024/25
Vedoucí práce: Mgr. Jan Lána
Třídní učitel: Mgr. Blanka Hniličková

Čestné prohlášení:

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené.

Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne ....

Anotace

Tato práce představuje vývoj webové aplikace PrachSocial, která funguje jako plnohodnotná sociální síť. Aplikace umožňuje uživatelům vytvářet vlastní profily, publikovat příspěvky s textem a multimediálním obsahem, sledovat ostatní uživatele, komentovat a hodnotit příspěvky, a komunikovat prostřednictvím systému zasílání zpráv. Uživatelé mohou také vyhledávat obsah, ukládat oblíbené příspěvky a dostávat oznámení o aktivitách spojených s jejich účtem. Aplikace implementuje moderní přístup k vývoji webových systémů s využitím frameworku Next.js, Prisma ORM pro práci s databází a Stream Chat pro implementaci chatovacího systému.

Abstract

This work presents the development of PrachSocial, a web application functioning as a comprehensive social network. The application allows users to create profiles, publish posts with text and multimedia content, follow other users, comment on and like posts, and communicate through a messaging system. Users can also search for content, save favorite posts, and receive notifications about activities related to their account. The application implements a modern approach to web system development using the Next.js framework, Prisma ORM for database operations, and Stream Chat for implementing the chat system.

Zadání projektu

Zadáním projektu bylo vytvořit sociální síť se všemi moderními webovými technologiemi a prvky. Bude umožňovat uživatelům sdílet příspěvky, lajkovat, komentovat a komunikovat přes zprávy. Bude obsahovat nekonečný feed, notifikace a správu uživatelského účtu s veřejným profilem, včetně autentizace pomocí hesla nebo Google účtu. Data budou zpracovávány přes Postgres databázi pomocí knihovny Prisma, s trending hashtagy, zmínkami a vyhledáváním jak příspěvků tak uživatelů. Bude postavena na frameworku Next.js s Tailwind CSS. Vše bude plně responzivní s možností přepínání mezi motivy. Všechny data a příspěvky budou optimalizované pro rychlé načítání, cachování a revalidaci.

Obsah

1 Úvod

V rámci ročníkového projektu jsem se rozhodl vytvořit vlastní sociální síť PrachSocial, která kombinuje klíčové funkce známých platforem jako jsou Twitter, Instagram a další.

Cílem bylo vytvořit moderní webovou aplikaci, která bude nejen funkční, ale také dobře navržená z hlediska uživatelského rozhraní a architektury softwaru. Práce klade důraz na použití současných technologií a postupů ve vývoji webových aplikací, jako jsou React, Next.js, TypeScript a práce s různými API pro zajištění funkčnosti jako je ukládání souborů či chatování.

Síť nabízí uživatelům možnost registrace buď pomocí Googlu nebo klasicky pomocí uživatelského jména a hesla. Po přihlášení mohou vytvářet příspěvky s textem, obrázky nebo videy, sledovat ostatní uživatele, komentovat a hodnotit příspěvky, a také komunikovat prostřednictvím přímých zpráv. Aplikace zahrnuje systém oznámení, který uživatele informuje o nových sledováních, komentářích a hodnoceních jejich příspěvků.

Během vývoje jsem se snažil uplatnit moderní postupy v oblasti webového vývoje, jako je serverové vykreslování, které zlepšuje výkon a SEO, nebo použití typového jazyka TypeScript pro minimalizaci chyb. Aplikace také implementuje responzivní design, který zajišťuje optimální zobrazení na různých zařízeních od mobilních telefonů po velké monitory.

2 Postup instalace

Pro správnou instalaci a spuštění aplikace je třeba provést několik kroků, které jsou podrobně popsány níže.

2.1 Instalace závislostí

Prvním krokem je instalace všech potřebných závislostí. Vzhledem k specifickým požadavkům některých knihoven je nutné použít parametr `-legacy-peer-deps`:

```
npm i --legacy-peer-deps
```

2.2 Konfigurace externích služeb

Aplikace využívá několik externích služeb, které je potřeba nastavit:

2.2.1 UploadThing

Pro ukládání a zpracování souborů je využívána služba UploadThing:

1. Vytvoření projektu na uploadthing.com
2. Získání Secret a App ID v sekci Legacy
3. Zadání těchto hodnot do .env souboru

2.2.2 Stream

Pro funkčnost chatu je použita služba Stream:

1. Vytvoření aplikace na getstream.io
2. Získání Key a Secret z App Access Keys
3. Vypnutí Threads & Replies v nastavení Chat Messaging → Channel Types → messaging
4. Zadání těchto hodnot do .env souboru

2.2.3 Google Cloud

Pro autentizaci přes Google:

1. Vytvoření projektu v Google Cloud Console
2. Nastavení OAuth Consent Screen:
    - Volba External
    - Vyplnění povinných polí
    - Přidání scope `.../auth/userinfo.email` a `.../auth/userinfo.profile`
    - Publikování projektu
3. Vytvoření OAuth client ID:
    - Typ Web Application
    - Přidání `http://localhost:3000/api/auth/callback/google` do Authorized redirect URIs
    - Získání Client ID a Client secret
    - Zadání těchto hodnot do .env souboru

2.3 Vytvoření souboru prostředí

Je třeba vytvořit soubor `.env` s následujícími proměnnými:

```
POSTGRES_USER="username"
POSTGRES_HOST="host"
POSTGRES_PASSWORD="password"
POSTGRES_DATABASE="database"

UPLOADTHING_SECRET='UploadThing Secret Key'
NEXT_PUBLIC_UPLOADTHING_APP_ID='UploadThing App ID'

NEXT_PUBLIC_STREAM_KEY='Stream Key'
STREAM_SECRET='Stream Secret'

GOOGLE_CLIENT_ID='CLIENT_ID.apps.googleusercontent.com'
GOOGLE_CLIENT_SECRET='CLIENT_SECRET'

CRON_SECRET='Random String'
NEXT_PUBLIC_BASE_URL='http://localhost:3000'
```

2.4 Nastavení databáze s Prisma

Pro inicializaci databáze je potřeba spustit:

```
npx prisma generate
npx prisma db push
```

2.5 Nastavení Cron Job pro čištění nepoužívaných souborů

Nevyžadováno

Pro pravidelné čištění nepoužívaných uploadů je třeba nastavit cron úlohu, která bude volat API endpoint aplikace pro odstranění souborů, které nejsou spojeny s žádným příspěvkem nebo uživatelským avatarem. Tato funkce pomáhá udržovat čistotu úložiště a předchází hromadění nepotřebných dat.

```
crontab -e
```

A přidat řádek:

```
0 2 * * * curl -X GET "http://localhost:3000/api/clear"
-H "Authorization: Bearer [CRON_SECRET]"
```

Kde `[CRON_SECRET]` je hodnota z .env souboru.

2.6 Spuštění vývojového serveru

Po dokončení všech předchozích kroků lze spustit vývojový server:

```
npm run dev
```

Aplikace bude dostupná na adrese `http://localhost:3000`.

3 Využité technologie

3.1 Základní technologie

3.1.1 Next.js

Next.js je React framework, který umožňuje vytvářet moderní webové aplikace s podporou serverového vykreslování, statické generace a klientského vykreslování. V projektu využívám App Router, který je součástí Next.js 13+, nabízející intuitivní způsob organizace souborové struktury a routování. Next.js usnadňuje optimalizaci výkonu, SEO a nabízí jednoduché API pro implementaci serverových funkcí.

3.1.2 React

React je knihovna pro tvorbu uživatelských rozhraní, která umožňuje vytvářet znovupoužitelné komponenty. V projektu využívám nejnovější funkce Reactu jako jsou Hooks pro správu stavu a side-effectů. React je základním stavebním kamenem celé aplikace.

3.1.3 TypeScript

TypeScript je nadstavba JavaScriptu, která přidává statické typování. Díky němu jsem mohl odhalit mnoho chyb již během vývoje, zlepšit dokumentaci kódu a usnadnit refaktoring. TypeScript výrazně zlepšuje spolehlivost a udržitelnost kódu.

3.2 Styling a UI

3.2.1 Tailwind CSS

Tailwind CSS je utility-first CSS framework, který umožňuje rychlé vytváření uživatelských rozhraní pomocí předdefinovaných tříd. Díky Tailwindu jsem mohl vytvořit konzistentní design napříč celou aplikací bez psaní vlastního CSS.

3.2.2 Shadcn/UI

Shadcn/UI je knihovna komponent pro React, která staví na Tailwind CSS. Poskytuje sadu vysoce přizpůsobitelných komponent, které jsem využil pro vytvoření formulářů, tlačítek, modálních oken a dalších UI prvků. Knihovna je založena na Radix UI, což zajišťuje dobrou přístupnost.

3.3 Správa dat

3.3.1 Prisma

Prisma je moderní ORM (Object-Relational Mapping) pro Node.js a TypeScript. Umožňuje typově bezpečnou práci s databází a generuje typové definice na základě schématu databáze. V projektu používám Prisma pro všechny databázové operace, což značně zjednodušuje práci s daty.

3.3.2 PostgreSQL

PostgreSQL je výkonný open-source relační databázový systém. Zvolil jsem ho pro jeho robustnost, škálovatelnost a podporu fulltextového vyhledávání, které využívám pro implementaci vyhledávání v aplikaci.

3.3.3 TanStack Query

TanStack Query je knihovna pro správu stavů a práci s asynchronními daty v React aplikacích. Usnadňuje cachování dat, invalidaci cache, paginaci a další běžné úkoly při práci s API. V projektu je využívána především pro získávání dat ze serveru a implementaci nekonečného scrollování.

3.4 Autentizace a autorizace

3.4.1 Lucia

Lucia je autentizační knihovna pro Next.js, která umožňuje implementovat bezpečnou správu uživatelských účtů a sessions. V projektu ji využívám pro autentizaci uživatelů jak pomocí hesla, tak i OAuth (Google).

3.4.2 Argon2

Argon2 je moderní hashovací algoritmus, který je odolný vůči útokům pomocí specializovaných hardwarových zařízení. Používám ho pro bezpečné ukládání hesel uživatelů.

3.5 Externí služby

3.5.1 Stream Chat

Stream Chat je API a SDK pro implementaci real-time chatu. V projektu ji využívám pro implementaci systému zasílání přímých zpráv mezi uživateli. Služba zajišťuje real-time komunikaci,

ukládání historie zpráv a správu chatovacích kanálů.

3.5.2 UploadThing

UploadThing je služba pro ukládání souborů, která je optimalizovaná pro použití s Next.js. Využívám ji pro nahrávání obrázků a videí do příspěvků a pro profilové obrázky uživatelů.

3.5.3 Google OAuth

Pro umožnění přihlášení pomocí Google účtu využívám Google OAuth 2.0. Díky tomu se uživatelé mohou do aplikace přihlásit jedním kliknutím bez nutnosti registrace.

3.6 Další použité knihovny

- **react-hook-form** - Knihovna pro práci s formuláři v Reactu
- **zod** - Knihovna pro validaci schémat
- **Tiptap** - WYSIWYG editor pro vytváření příspěvků
- **react-cropper** - Implementace ořezávání obrázků
- **react-intersection-observer** - Detekce viditelnosti elementů pro implementaci nekonečného scrollování

4 Struktura projektu

Projekt je organizován podle moderních konvencí vývoje webových aplikací s využitím frameworku Next.js a jeho App Router architektury. Tato struktura umožňuje přehlednou organizaci kódu, jednodušší navigaci v projektu a efektivní oddělení zodpovědností jednotlivých komponent a modulů (1). V této sekci podrobně popíšu strukturu projektu a vysvětlím účel jednotlivých souborů a adresářů.

4.1 Kořenový adresář projektu

V kořenovém adresáři projektu se nachází několik klíčových souborů a adresářů:

- **src/** - Hlavní adresář zdrojového kódu aplikace
- **public/** - Statické soubory, které jsou přímo dostupné přes webový server
- **prisma/** - Obsahuje definice databázového schématu a migrační soubory
- **next.config.mjs** - Konfigurační soubor pro Next.js
- **package.json** - Seznam závislostí a skriptů pro správu projektu
- **tsconfig.json** - Konfigurační soubor pro TypeScript
- **tailwind.config.ts** - Konfigurační soubor pro Tailwind CSS
- **components.json** - Konfigurační soubor pro komponenty z knihovny shadcn/ui
- **.env** - Soubor s proměnnými prostředí (není součástí repozitáře)

4.2 Adresář /src

Adresář `/src` obsahuje veškerý zdrojový kód aplikace. Je strukturován podle doporučení Next.js a dále rozdělen na několik klíčových podadresářů:

4.2.1 /app

Adresář `app` je jádrem aplikace v Next.js App Router architektuře. Obsahuje definice stránek, layoutů a API routes. Struktura tohoto adresáře přímo odpovídá URL struktuře aplikace (2).

- **layout.tsx** - Základní layout aplikace, který definuje společné části pro všechny stránky, jako jsou meta tagy, hlavičky a patičky. Obsahuje také globální providery, jako jsou `NextSSRPlugin`, `ReactQueryProvider` a `ThemeProvider`.

- **globals.css** - Globální CSS styly, včetně definice CSS proměnných pro Tailwind CSS a speciálních stylů pro stream-chat.
- **loading.tsx** - Komponenta zobrazovaná během načítání stránek.
- **not-found.tsx** - Stránka zobrazovaná při nenalezení požadované URL.
- **ReactQueryProvider.tsx** - Poskytuje kontext pro React Query v celé aplikaci.

4.2.2 /(auth)

Tento adresář obsahuje komponenty a stránky spojené s autentizací. V Next.js je možné použít závorky v názvu adresáře pro seskupení souvisejících stránek bez ovlivnění URL cesty (3).

- **layout.tsx** - Definuje layout pro všechny stránky autentizace. Obsahuje logiku pro přesměrování přihlášených uživatelů.
- **/login/** - Adresář obsahující přihlašovací stránku:
  - **page.tsx** - Definuje UI pro přihlášení.
  - **LoginForm.tsx** - Formulářová komponenta pro přihlášení.
  - **actions.ts** - Obsahuje serverové akce pro zpracování přihlášení.
  - **/google/** - Podadresář pro autentizaci přes Google:
    - **GoogleSignInButton.tsx** - Komponent tlačítka pro Google přihlášení.
    - **route.ts** - API route pro Google OAuth flow.
- **/signup/** - Adresář obsahující registrační stránku s podobnou strukturou jako přihlašovací.
- **actions.ts** - Sdílené serverové akce pro autentizaci, jako je funkce pro odhlášení.

4.2.3 /(main)

Tento adresář obsahuje hlavní část aplikace, která je přístupná pouze přihlášeným uživatelům.

- **layout.tsx** - Hlavní layout pro přihlášené uživatele. Obsahuje navigační prvky, menu a ověření přihlášení.
- **navbar.tsx** - Komponenta horní navigační lišty.
- **MenuBar.tsx** - Komponenta bočního menu.
- **SessionProvider.tsx** - Poskytuje kontext pro informace o aktuálním uživateli.
- **page.tsx** - Hlavní stránka aplikace zobrazující feed příspěvků.
- **ForYouFeed.tsx** a **FollowingFeed.tsx** - Komponenty pro zobrazení feedů příspěvků.

- **/messages/** - Adresář obsahující funkcionalitu pro systém zpráv:
  - **page.tsx** - Hlavní stránka zpráv.
  - **Chat.tsx** - Hlavní komponenta pro chat.
  - **ChatSidebar.tsx** - Boční panel pro seznam chatů.
  - **ChatChannel.tsx** - Komponenta pro zobrazení konkrétního chatu.
  - **NewChatDialog.tsx** - Dialog pro vytvoření nového chatu.
  - **useInitializeChatClient.ts** - Hook pro inicializaci Stream Chat klienta.
- **/notifications/** - Adresář pro funkcionalitu oznámení:
  - **page.tsx** - Stránka s oznámeními.
  - **Notifications.tsx** - Komponenta pro zobrazení seznamu oznámení.
  - **Notification.tsx** - Komponenta pro jednotlivé oznámení.
- **/posts/[postId]/** - Dynamická cesta pro zobrazení konkrétního příspěvku (4):
  - **page.tsx** - Stránka detailu příspěvku s podporou pro SSR a metadaty.
- **/search/** - Adresář pro vyhledávací funkcionalitu:
  - **page.tsx** - Stránka s výsledky vyhledávání.
  - **SearchResults.tsx** - Komponenta pro zobrazení výsledků vyhledávání.
- **/users/[username]/** - Dynamická cesta pro profily uživatelů:
  - **page.tsx** - Stránka profilu uživatele.
  - **UserPosts.tsx** - Komponenta pro zobrazení příspěvků uživatele.
  - **EditProfileButton.tsx** a **EditProfileDialog.tsx** - Komponenty pro úpravu profilu.
  - **actions.ts** - Serverové akce pro profil uživatele.
  - **mutations.ts** - Definice mutací pro React Query.
- **/saved/** - Adresář pro uložené příspěvky:
  - **page.tsx** - Stránka s uloženými příspěvky.
  - **Saved.tsx** - Komponenta pro zobrazení uložených příspěvků.

4.2.4 /api

Adresář `api` obsahuje API routes, které definují backend funkcionalitu aplikace. V Next.js je každý soubor `route.ts` automaticky zpřístupněn jako API endpoint podle své cesty v adresářové struktuře (5).

- **/auth/** - Autentizační API endpointy:
  - **/callback/google/route.ts** - Callback endpoint pro Google OAuth.
- **/posts/** - API endpointy pro práci s příspěvky:
  - **for-you/route.ts** - Endpoint pro získání feedu "Pro vás".
  - **following/route.ts** - Endpoint pro získání feedu ze sledovaných uživatelů.
  - **saved/route.ts** - Endpoint pro získání uložených příspěvků.
  - **[postId]/** - Dynamická cesta pro operace nad konkrétním příspěvkem:
    - **likes/route.ts** - Operace pro likes příspěvku.
    - **saved/route.ts** - Operace pro uložení příspěvku.
    - **comments/route.ts** - Operace pro komentáře příspěvku.
- **/users/** - API endpointy pro uživatele:
  - **[userId]/** - Dynamická cesta pro operace nad konkrétním uživatelem:
    - **followers/route.ts** - Operace pro sledování uživatelů.
    - **posts/route.ts** - Získání příspěvků konkrétního uživatele.
  - **username/[username]/route.ts** - Získání informací o uživateli podle username.
- **/notifications/** - API endpointy pro oznámení:
  - **route.ts** - Získání seznamu oznámení.
  - **unread-count/route.ts** - Získání počtu nepřečtených oznámení.
  - **mark-read/route.ts** - Označení oznámení jako přečtených.
- **/messages/** - API endpointy pro zprávy:
  - **unread-count/route.ts** - Získání počtu nepřečtených zpráv.
- **/search/route.ts** - Endpoint pro vyhledávání.
- **/get-token/route.ts** - Endpoint pro získání Stream Chat tokenu.
- **/clear/route.ts** - Endpoint pro cron job, který čistí nepoužívané soubory.
- **/uploadthing/** - Endpointy pro UploadThing službu:
  - **core.ts** - Definice uploadovacích routerů.
  - **route.ts** - API endpointy pro UploadThing.

4.2.5 /components

Adresář `components` obsahuje znovupoužitelné React komponenty, které jsou využívány napříč aplikací. Jsou organizovány tematicky do podadresářů dle doporučených postupů v React (8).

- **/ui/** - Předdefinované UI komponenty poskytnuté v základu
- **/posts/** - Komponenty pro práci s příspěvky:
  - **Post.tsx** - Komponenta jednotlivého příspěvku.
  - **PostsLoadingSkeleton.tsx** - Komponenta pro načítací stav příspěvků.
  - **PostMoreButton.tsx** - Tlačítko s dalšími akcemi pro příspěvek.
  - **DeletePostDialog.tsx** - Dialog pro smazání příspěvku.
  - **actions.ts** - Serverové akce pro příspěvky.
  - **mutations.ts** - Definice mutací pro React Query.
  - **/editor/** - Adresář s komponentami pro editor příspěvků:
    - **PostEditor.tsx** - Hlavní komponenta editoru příspěvků.
    - **useMediaUpload.ts** - Hook pro nahrávání médií.
    - **mutations.ts** - Mutace specifické pro editor.
    - **actions.ts** - Serverové akce pro editor.
    - **styles.css** - Speciální styly pro editor.
- **/comments/** - Komponenty pro komentáře:
  - **Comments.tsx** - Komponenta pro zobrazení všech komentářů.
  - **Comment.tsx** - Komponenta jednotlivého komentáře.
  - **CommentInput.tsx** - Komponenta pro zadávání nových komentářů.
  - **CommentMoreButton.tsx** - Tlačítko s dalšími akcemi pro komentář.
  - **DeleteCommentDialog.tsx** - Dialog pro smazání komentáře.
  - **actions.ts** - Serverové akce pro komentáře.
  - **mutations.ts** - Definice mutací pro komentáře.
- **UserAvatar.tsx** - Komponenta pro zobrazení avataru uživatele.
- **UserButton.tsx** - Komponenta tlačítka uživatele v navigaci.
- **UserTooltip.tsx** - Komponenta pro tooltip s informacemi o uživateli.
- **UserLinkWithTooltip.tsx** - Komponenta odkazu na uživatele s tooltipem.
- **FollowButton.tsx** - Komponenta tlačítka pro sledování uživatele.

- **FollowerCount.tsx** - Komponenta pro zobrazení počtu sledujících.
- **SearchField.tsx** - Komponenta vyhledávacího pole.
- **PasswordInput.tsx** - Specializovaná komponenta pro zadávání hesla.
- **Linkify.tsx** - Komponenta pro detekci a zvýraznění odkazů a zmínek v textu.
- **LoadingButton.tsx** - Tlačítko s indikátorem načítání.
- **InfiniteScrollContainer.tsx** - Komponenta pro nekonečné scrollování.
- **TrendsSidebar.tsx** - Komponenta s trendy a doporučenými uživateli.
- **CropImageDialog.tsx** - Dialog pro ořezávání obrázků.

4.2.6 /lib

Adresář `lib` obsahuje pomocné funkce, utility a definice typů, které jsou používány napříč aplikací. Tento vzor je doporučován v Next.js dokumentaci pro oddělení pomocných funkcí od komponent (6).

- **types.ts** - Definice TypeScript typů a rozhraní používaných v aplikaci.
- **utils.ts** - Pomocné utility jako formátování data, čísel, a další.
- **validation.ts** - Schémata pro validaci formulářů pomocí Zod.
- **prisma.ts** - Singleton instance Prisma klienta. Implementuje návrhový vzor Singleton pro zabránění vytváření více instancí Prisma klienta (10).
- **ky.ts** - Konfigurovaná instance HTTP klienta Ky.
- **stream.ts** - Konfigurovaná instance Stream Chat klienta pro server.
- **uploadthing.ts** - Utility pro UploadThing službu.

4.2.7 /hooks

Adresář `hooks` obsahuje vlastní React hooks, které zapouzdřují znovupoužitelnou logiku dle doporučených postupů Reactu (9).

- **useDebounce.ts** - Hook pro debouncing vstupu, inspirovaný řešením na Stack Overflow (13).
- **useFollowerInfo.ts** - Hook pro získání informací o sledujících.

4.2.8 auth.ts

Soubor `/src/auth.ts` obsahuje konfiguraci autentizační knihovny Lucia, definice adaptérů pro Prisma a utility pro validaci požadavků (12).

4.3 /prisma

Adresář `prisma` obsahuje definice databázového schématu a migrace.

- **schema.prisma** - Definice databázového schématu v Prisma formátu (11), včetně modelů pro:
  - Uživatele (User)
  - Relace (Session)
  - Příspěvky (Post)
  - Média (Media)
  - Komentáře (Comment)
  - Lajky (Like)
  - Uložené příspěvky (Saved)
  - Sledování (Follow)
  - Oznámení (Notification)

4.4 Typy souborů a jejich význam

V Next.js architektuře mají různé typy souborů specifický význam (7):

- **page.tsx** - Definuje UI komponenty pro stránku na dané URL cestě. Tyto soubory jsou automaticky dostupné jako route v aplikaci.
- **layout.tsx** - Definuje sdílený layout pro stránky nebo skupinu stránek. Layouts se vnořují a můžou být definovány na různých úrovních.
- **route.ts** - Definuje API endpoint na dané URL cestě s možností implementace různých HTTP metod (GET, POST, PUT, DELETE).
- **actions.ts** - Obsahuje serverové akce, které mohou být volány z klientského kódu pomocí Server Actions API v Next.js.
- **mutations.ts** - Definuje mutace pro React Query, které zapouzdřují logiku pro změnu dat na serveru.

- **loading.tsx** - Definuje UI komponent, který se zobrazí během načítání stránky. Je součástí Suspense API v Next.js.
- **not-found.tsx** - Definuje UI komponent, který se zobrazí, když není nalezena odpovídající stránka. Nahrazuje stránku Error kódu 404.

4.5 Speciální adresářové konvence

Next.js používá několik speciálních konvencí pro adresáře (3):

- **(skupiny)** - Adresáře v závorkách jsou používány pro seskupení souvisejících stránek nebo layoutů bez ovlivnění URL cesty.
- **[dynamické-parametry]** - Adresáře v hranatých závorkách definují dynamické segmenty URL cesty, které mohou být přístupné jako parametry v komponentách.

5 Implementace klíčových funkcí

5.1 Autentizace a autorizace

Systém autentizace je implementován pomocí knihovny Lucia (12), která poskytuje moderní a bezpečný způsob správy uživatelských sesí. Systém podporuje dva způsoby přihlášení:

- Klasické přihlášení pomocí uživatelského jména a hesla
- OAuth přihlášení přes Google účet

Hesla uživatelů jsou hashována pomocí algoritmu Argon2 (16), který je považován za jeden z nejbezpečnějších hashovacích algoritmů v současnosti. Pro implementaci Google OAuth je využita knihovna Arctic (17).

5.1.1 Konfigurace Lucia

Základní konfigurace Lucia je umístěna v souboru `src/auth.ts`:

```
export const lucia = new Lucia(adapter, {
  sessionCookie: {
    expires: false,
    attributes: {
      secure: process.env.NODE_ENV === "production",
    },
  },
  getUserAttributes(databaseUserAttributes) {
    return {
      id: databaseUserAttributes.id,
      username: databaseUserAttributes.username,
      displayName: databaseUserAttributes.displayName,
      avatarUrl: databaseUserAttributes.avatarUrl,
      googleId: databaseUserAttributes.googleId,
    };
  },
});
```

Tento kód vytváří instanci Lucia s použitím PrismaAdapter, který propojuje Lucia s naší databází. Konfigurace také specifikuje, jaké atributy uživatele budou dostupné v objektu User.

5.1.2 Implementace přihlašovacího formuláře

Přihlašovací formulář je implementován s využitím knihovny React Hook Form (14) a Zod (15) pro validaci. Formulář automaticky validuje vstupní údaje a zobrazuje chybové zprávy uživateli v případě neplatných údajů.

Samotný proces přihlášení je realizován pomocí serverové akce, která ověří přihlašovací údaje proti databázi, vytvoří novou session pomocí Lucia a nastaví session cookie pro uživatele. V případě úspěšného přihlášení je uživatel přesměrován na hlavní stránku aplikace, v případě neúspěchu je zobrazena chybová zpráva.

5.1.3 Implementace Google OAuth

Pro implementaci přihlášení přes Google jsou využity dva hlavní endpointy, z nichž jeden inicializuje proces autentizace a druhý zpracovává callback od Google OAuth serveru. V callback endpointu je implementováno zpracování OAuth tokenu, získání informací o uživateli z Google API a vytvoření nebo aktualizace uživatelského účtu v databázi.

Tímto způsobem mohou uživatelé využít své existující Google účty pro rychlé přihlášení bez nutnosti registrace a pamatování dalšího hesla.

5.2 Práce s příspěvky

Příspěvky jsou centrálním prvkem aplikace. Implementace zahrnuje vytváření, zobrazování, upravování a mazání příspěvků. Zásadní je také efektivní načítání příspěvků v nekonečném scrollu.

5.2.1 Editor příspěvků

Editor příspěvků využívá knihovnu Tiptap (18) pro formátování textu a specializovaný hook `useMediaUpload` pro nahrávání médií. Tento přístup umožňuje uživatelům vytvářet bohaté příspěvky s formátovaným textem a multimediálním obsahem.

```
const editor = useEditor({
  extensions: [
    StarterKit.configure({
      bold: false,
      italic: false,
    }),
    Placeholder.configure({
      placeholder: "Write something...",
    }),
  ],
});
```

5.2.2 Nahrávání médií

Nahrávání médií je implementováno pomocí služby UploadThing (27). Na straně serveru jsou definovány uploadovací routery, které zpracovávají nahrané soubory, provádějí validaci a vytvářejí záznamy v databázi. Implementace poskytuje uživatelům možnost nahrávat obrázky a videa, sledovat průběh nahrávání a zobrazuje náhledy nahraných médií.

Pro použití v příspěvcích byla vytvořena komponenta MediaPreviews, která efektivně zobrazuje nahraná média v různých konfiguracích a podporuje responzivní design.

5.3 Implementace vyhledávání

Vyhledávací funkce umožňuje uživatelům hledat příspěvky a uživatele podle klíčových slov. Implementace využívá fulltextové vyhledávání v PostgreSQL (24).

5.3.1 Vyhledávací pole

Komponenta `SearchField.tsx` poskytuje uživatelské rozhraní pro zadávání vyhledávacích dotazů a předává je do URL parametrů pro zpracování na stránce výsledků vyhledávání:

```
1  function handleSubmit(e: React.FormEvent<HTMLFormElement>) {
2    e.preventDefault();
3    const form = e.currentTarget;
4    const q = (form.q as HTMLInputElement).value.trim();
5    if (!q) return;
6    router.push(`/search?q=${encodeURIComponent(q)}`);
7  }
```

5.3.2 Backend implementace vyhledávání

Na backendu je vyhledávání implementováno s využitím fulltextového vyhledávání PostgreSQL přes Prisma ORM. Dotaz je rozdělen na jednotlivá slova a převeden do formátu, který PostgreSQL používá pro fulltextové vyhledávání. Výsledky jsou následně seřazeny podle relevance a vráceny klientovi.

Vyhledávání bere v úvahu obsah příspěvků, uživatelská jména a zobrazovaná jména uživatelů, čímž poskytuje komplexní výsledky.

5.4 Hashtagy a zmínky v příspěvcích

Důležitou součástí sociální sítě je možnost označit uživatele pomocí zmínek (@) a vytvářet hashtagy (#). Implementace této funkcionality zahrnuje detekci a zvýraznění těchto prvků v textu a

jejich propojení s příslušnými stránkami.

5.4.1 Detekce a zvýraznění v textu

Pro detekci a zvýraznění odkazů, hashtagů a zmínek je implementována komponenta `Linkify`, která využívá regulární výrazy a knihovnu `react-linkify-it` pro transformaci prostého textu na interaktivní odkazy:

```
function LinkifyUsername({ children }: LinkifyProps) {
  return (
    <LinkIt
      regex={/(@[a-zA-Z0-9_-]+)/}
      component={(match, key) => (
        <UserLinkWithTooltip key={key} username={match.slice(1)
   }>
          {match}
        </UserLinkWithTooltip>
      )}
    >
      {children}
    </LinkIt>
  );
}
```

5.4.2 Implementace trendů

Pro zobrazení populárních hashtagů je implementována komponenta `TrendingSidebar`, která pomocí SQL dotazu extrahuje a agreguje hashtagy z příspěvků. Výsledky jsou cachovány pomocí funkce `unstable_cache` pro lepší výkon.

5.5 Systém oznámení

Systém oznámení informuje uživatele o aktivitách souvisejících s jejich účtem, jako jsou nové komentáře, lajky a sledování.

5.5.1 Typy oznámení

V databázovém schématu jsou definovány tři typy oznámení pomocí enum:

```
enum NotificationType {
  LIKE
  FOLLOW
  COMMENT
}
```

5.5.2 Vytváření oznámení

Oznámení jsou vytvářena současně s akcemi, které je spouští. Například při přidání lajku k příspěvku je v rámci jedné transakce vytvořen jak samotný lajk, tak i oznámení pro autora příspěvku (pokud autor není stejný uživatel, který dal lajk). Tento přístup zajišťuje atomicitu operací a konzistenci dat.

5.5.3 Zobrazení oznámení

Komponenta `Notifications.tsx` zobrazuje seznam oznámení s nekonečným scrollováním a automaticky označuje oznámení jako přečtená při jejich zobrazení. Jednotlivá oznámení jsou renderována pomocí komponenty `Notification.tsx`, která na základě typu oznámení zobrazuje odpovídající ikonu, text a odkaz.

5.5.4 Počítání nepřečtených oznámení

Pro zobrazení počtu nepřečtených oznámení je implementován API endpoint a komponenta `Notifications-Button.tsx`, která tento počet zobrazuje v navigačním menu. Počet nepřečtených oznámení je pravidelně aktualizován pomocí React Query.

6 Uživatelské rozhraní

Uživatelské rozhraní aplikace bylo navrženo s důrazem na přehlednost, použitelnost a responzivitu. Vzhledem k tomu, že se jedná o sociální síť, bylo klíčové vytvořit intuitivní prostředí, které umožní uživatelům snadno konzumovat obsah a interagovat s ním.

6.1 Návrh a implementace UI

Pro návrh uživatelského rozhraní jsem zvolil kombinaci moderních přístupů a nástrojů. Základem je knihovna Tailwind CSS (26), která umožňuje rychlý a konzistentní vývoj UI pomocí utility tříd. Na ní je postavena knihovna komponent Shadcn UI (29), která poskytuje přístupné a esteticky příjemné UI komponenty.

6.1.1 Systém komponent

Aplikace využívá komponentovou architekturu, kde každá UI komponenta je zodpovědná za specifickou část rozhraní:

- **Navigační komponenty** - Navbar, MenuBar, podpora pro mobilní i desktopová zařízení
- **Komponenty příspěvků** - Post, PostEditor, PostMoreButton
- **Formulářové komponenty** - Input, Textarea, PasswordInput, SearchField
- **Dialogová okna** - EditProfileDialog, DeletePostDialog, NewChatDialog
- **Interaktivní prvky** - FollowButton, LikeButton, SaveButton, UserTooltip
- **Stránkování a načítání** - InfiniteScrollContainer, PostsLoadingSkeleton

Všechny komponenty jsou navrženy tak, aby byly znovupoužitelné a konzistentní v celé aplikaci.

6.1.2 Designový systém

Designový systém je založen na proměnných v CSS, které definují barvy, zaoblení, stíny a další vizuální aspekty aplikace. Díky tomu je možné snadno změnit vzhled aplikace a implementovat různé barevné motivy.

Proměnné jsou definovány v souboru `globals.css`:

```
@layer base {
  :root {
  // Proměnné základních barev a zvýraznění
  }

  .dark {
  // Proměnné pro tmavý režim
  }
}
```

6.2 Responzivní design

Aplikace je plně responzivní a přizpůsobuje se různým velikostem obrazovek od mobilních telefonů po velké desktopové monitory. Implementace využívá breakpointy Tailwind CSS (26) a speciální komponenty pro různé velikosti obrazovek.

6.2.1 Mobilní zobrazení

Na mobilních zařízeních je menu zobrazeno jako spodní navigační lišta a obsah zabírá celou šířku obrazovky. Některé méně důležité prvky jsou skryty nebo zmenšeny, aby se maximalizoval prostor pro obsah.

6.2.2 Tablet a desktop

Na větších obrazovkách se menu přesouvá na levou stranu jako boční panel, což umožňuje lepší přehled a navigaci. Obsah je zobrazen ve středu s dostatečnými okraji pro lepší čitelnost a na pravé straně se zobrazuje panel s trendy a doporučenými uživateli.

6.2.3 Implementace breakpointů

Pro zajištění správného zobrazení na různých zařízeních jsou využity CSS třídy Tailwind s breakpointy:

```
<div className="hidden sm:block lg:w-64">
  {/* Obsah viditelný pouze na obrazovkách větších než sm */}
</div>

<div className="block sm:hidden">
  {/* Obsah viditelný pouze na mobilních zařízeních */}
</div>
```

6.3 Podpora světlého a tmavého režimu

Aplikace podporuje světlý a tmavý režim, mezi kterými si uživatel může přepínat. Implementace využívá knihovnu next-themes (25), která poskytuje React hook `useTheme` pro práci s tématem.

6.3.1 Přepínání témat

Přepínač témat je implementován v uživatelském menu, kde si uživatel může vybrat mezi systémovým, světlým nebo tmavým režimem. Při změně tématu se okamžitě aktualizuje vzhled celé aplikace díky CSS proměnným a Tailwind CSS třídám.

```
<DropdownMenuSubContent>
  <DropdownMenuItem onClick={() => setTheme("system")}>
    <Monitor className="mr-2 size-4" />
    System Default
    {theme === "system" && <Check />}
  </DropdownMenuItem>
  {/* Světlý a tmavý motiv */}
</DropdownMenuSubContent>
```

6.4 Animace a přechody

Pro lepší uživatelský zážitek aplikace využívá jemné animace a přechody, které poskytují vizuální zpětnou vazbu na akce uživatele. Tyto animace jsou implementovány pomocí CSS transitions a plugin tailwindcss-animate.

- Fade-in efekt při načítání nových příspěvků
- Přechody při otevírání a zavírání dialogů

- Plynulé přechody při změně témat
- Animace tlačítek při najetí myší a kliknutí

6.5 Uživatelská zkušenost (UX)

Kromě samotného UI byl kladen důraz i na celkovou uživatelskou zkušenost:

- **Okamžitá zpětná vazba** - Optimistické aktualizace UI při akcích jako like, follow nebo save
- **Toast notifikace** - Informují uživatele o výsledku jejich akcí (úspěch, chyba)
- **Skeleton loaders** - Zobrazují se během načítání obsahu, aby uživatel věděl, že se něco děje
- **Nekonečné scrollování** - Umožňuje plynulé procházení obsahu bez nutnosti manuálního přecházení mezi stránkami
- **Tooltips** - Poskytují dodatečné informace o uživatelích a prvcích UI

Tyto prvky společně vytvářejí plynulou a intuitivní uživatelskou zkušenost, která je klíčová pro sociální síť, kde uživatelé tráví významné množství času procházením obsahu a interakcí s ním.

7 Databázová vrstva

Databázová vrstva aplikace je postavena na PostgreSQL a ORM Prisma, což poskytuje typově bezpečný přístup k datům, snadnou definici schématu a efektivní dotazování. V této sekci popisuji návrh databázového schématu, implementaci relací mezi entitami a přístup k datům v aplikaci.

7.1 Schéma databáze

Databázové schéma je definováno v souboru `prisma/schema.prisma` pomocí Prisma Schema Language (11). Hlavní entity v databázi zahrnují:

- **User** - Uživatelské účty
- **Session** - Uživatelské relace pro autentizaci
- **Post** - Příspěvky uživatelů
- **Media** - Mediální soubory připojené k příspěvkům
- **Comment** - Komentáře k příspěvkům
- **Like** - Lajky na příspěvky
- **Saved** - Uložené příspěvky
- **Follow** - Vztahy sledování mezi uživateli
- **Notification** - Oznámení pro uživatele

7.2 Vztahy mezi entitami

Databázové schéma obsahuje několik typů vztahů mezi entitami:

7.2.1 One-to-Many vztahy

Příkladem One-to-Many vztahu je vztah mezi uživatelem a příspěvky. Jeden uživatel může mít mnoho příspěvků, ale každý příspěvek patří právě jednomu uživateli. Tento vztah je implementován pomocí cizího klíče v tabulce příspěvků.

7.2.2 Many-to-Many vztahy

Vztah sledování mezi uživateli je příkladem Many-to-Many vztahu. Uživatel může sledovat mnoho jiných uživatelů a zároveň být sledován mnoha uživateli. Tento vztah je implementován pomocí spojovací tabulky ‘Follow‘, která obsahuje dvojici cizích klíčů odkazujících na tabulku uživatelů.

7.3 Přístup k datům

Pro přístup k datům v databázi využívá aplikace Prisma Client, který je inicializován jako singleton pro zajištění efektivní správy připojení k databázi. Toto řešení zabraňuje vytváření mnoha instancí klienta, což by mohlo vést k problémům s výkonem.

7.3.1 Dotazování dat

Prisma poskytuje typově bezpečné API pro dotazování dat. Příklad dotazu pro získání příspěvků s vnořenými relacemi ukazuje, jak lze efektivně načítat související data v jednom požadavku:

```
const posts = await prisma.post.findMany({
  include: {
    user: {
      select: getUserDataSelect(user.id),
    },
    attachments: true,
    likes: {
      where: {
        userId: user.id,
      },
      select: {
        userId: true,
      },
    },
  },
});
```

7.3.2 Transakce

Pro zajištění atomicity operací, které ovlivňují více tabulek, využívá aplikace databázové transakce. To je důležité například při vytváření sledování uživatele, kdy je zároveň vytvořeno oznámení.

7.4 Vyhledávání v databázi

Aplikace implementuje fulltextové vyhledávání pomocí funkcí PostgreSQL. To umožňuje efektivní vyhledávání v obsahu příspěvků a údajích uživatelů.

7.5 Optimalizace výkonu

Pro zajištění dobrého výkonu při práci s databází jsou implementovány různé optimalizace včetně selektivního dotazování, efektivního stránkování a použití indexů.

8 Výzvy a řešení

Během vývoje aplikace jsem se setkal s různými technickými výzvami, které jsem musel vyřešit. V této sekci popisuji některé z nejzajímavějších problémů a jejich řešení.

8.1 Implementace real-time chatu

Jednou z největších výzev bylo implementovat real-time chat, který by umožňoval uživatelům komunikovat bez nutnosti obnovení stránky.

8.1.1 Problém

Implementace real-time funkcionality od základu by vyžadovala vytvoření a správu WebSocket připojení, řešení offline zpráv a synchronizace historie, implementaci presence (kdo je online) a správu chatovacích kanálů.

8.1.2 Řešení

Rozhodl jsem se využít službu Stream Chat (23), která poskytuje kompletní řešení pro real-time chat. Implementace zahrnovala integraci Stream Chat SDK, vytvoření serverového endpointu pro generování autentizačních tokenů, synchronizaci uživatelských profilů a přizpůsobení vzhledu.

Klíčovou součástí řešení byl hook ‘useInitializeChatClient‘, který zajišťuje inicializaci a správu Stream Chat klienta:

```
// Inicializace klienta s API klíčem
const client = StreamChat.getInstance(process.env.
   NEXT_PUBLIC_STREAM_KEY!);
// Připojení uživatele s jeho údaji a autentizačním tokenem
client.connectUser(
  { id: user.id, username: user.username, /* další údaje */ },
  async () => kyInstance.get('/api/get-token').json().then(data
   => data.token)
)
.then(() => setChatClient(client));
```

Tento přístup mi umožnil rychle implementovat komplexní funkci chatu bez nutnosti řešit nízkoúrovňové detaily WebSocket komunikace a správy stavu.

8.2 Optimalizace načítání dat

Další výzvou bylo efektivní načítání dat, zejména při implementaci nekonečného scrollování a cachování dat pro lepší výkon.

8.2.1 Problém

Při implementaci nekonečného scrollování jsem narazil na několik problémů, jako je efektivní načítání dalších stránek dat bez duplicit, aktualizace již načtených položek při změnách a zachování stavu scrollování při navigaci.

8.2.2 Řešení

Pro řešení těchto problémů jsem použil knihovnu TanStack Query (dříve React Query) (19), která poskytuje sofistikované nástroje pro správu asynchronních dat. Implementoval jsem ‘useInfiniteQuery‘ pro načítání dalších stránek dat při scrollování, využil jsem cachování dotazů a implementoval selektivní invalidaci cache při změnách dat.

Pro detekci scrollování k dolnímu okraji stránky jsem vytvořil komponentu ‘InfiniteScrollContainer‘, která využívá knihovnu ‘react-intersection-observer‘ pro sledování viditelnosti elementů:

```
const { ref } = useInView({
rootMargin: "200px",
onChange: (inView) => {
  if (inView) {
    onBottomReached();
  }
},
});
```

8.3 Nahrávání a zpracování médií

Implementace nahrávání médií představovala další výzvu, zejména s ohledem na uživatelský zážitek a efektivitu.

8.3.1 Řešení

Pro nahrávání souborů jsem využil službu UploadThing (27), která poskytuje efektivní řešení pro nahrávání souborů s přímým uploadem do cloud storage. Pro zpracování obrázků před nahráním jsem implementoval komponentu pro ořezávání s využitím knihovny ‘react-cropper‘.

Pro sledování průběhu nahrávání jsem vytvořil specializovaný hook ‘useMediaUpload‘, který poskytuje bohaté API pro práci s nahrávanými soubory včetně progresových indikátorů a možnosti odstranění souborů před odesláním.

8.4 Optimalizace výkonu na mobilních zařízeních

Vzhledem k tomu, že sociální síť je často používána na mobilních zařízeních, bylo důležité optimalizovat výkon i na méně výkonných zařízeních.

8.4.1 Řešení

Pro optimalizaci výkonu jsem implementoval lazy loading komponent a dat, code splitting, optimalizaci obrázků pomocí Next.js Image komponenty a skeleton loading místo tradičních loaderů.

Skeleton loading poskytuje lepší uživatelský zážitek tím, že místo klasického spinneru zobrazuje strukturu obsahu, který se načítá, což snižuje vnímanou dobu načítání:

```
<div className="w-full animate-pulse space-y-3 rounded-2xl bg-
    card p-5 shadow-sm">
  <div className="flex flex-wrap gap-3">
    <Skeleton className="size-12 rounded-full" />
    <div className="space-y-1.5">
      <Skeleton className="h-4 w-24 rounded" />
      <Skeleton className="h-4 w-20 rounded" />
    </div>
  </div>
  <Skeleton className="h-16 rounded" />
</div>
```

8.5 Implementace serverových komponent v Next.js

Součástí Next.js jsou React Server Components, které umožňují vytvářet komponenty, které běží pouze na serveru. Tato architektura přináší výhody, ale i výzvy.

8.5.1 Řešení

Pro efektivní práci se Server Components jsem implementoval "interleaving pattern", kde jsou serverové a klientské komponenty kombinovány tak, aby maximalizovaly výhody obou přístupů. Složité datové operace jsou prováděny na serveru pomocí Server Components, zatímco interaktivní části UI jsou implementovány jako Client Components. Data jsou předávána z Server Components do Client Components pomocí props.

Tento přístup umožňuje optimální rozdělení zodpovědností mezi server a klient a vede k lepšímu výkonu aplikace.

9 Bezpečnost

V rámci vývoje jsem implementoval několik bezpečnostních opatření na různých úrovních aplikace.

9.1 Autentizace

Systém autentizace je implementován pomocí knihovny Lucia (12), která poskytuje moderní a bezpečný přístup k autentizaci v Next.js aplikacích.

9.1.1 Bezpečné ukládání hesel

Pro hashování hesel je použit algoritmus Argon2 (16), který je odolný vůči útokům pomocí specializovaných hardwarových zařízení. Při hashování jsou použity bezpečné parametry pro zajištění odolnosti proti brute-force útokům.

9.1.2 Session management

Lucia poskytuje bezpečný způsob správy uživatelských relací pomocí HTTP cookies. Cookie obsahující token session je nastavena jako `HttpOnly`, což brání přístupu z JavaScriptu a snižuje riziko XSS útoků.

9.1.3 OAuth integrace

Pro autentizaci přes Google je použita knihovna Arctic, která implementuje bezpečnostní best practices pro OAuth 2.0, včetně použití PKCE (Proof Key for Code Exchange) pro zabránění CSRF útokům.

9.2 Autorizace

Autorizace zajišťuje, že uživatelé mají přístup pouze k funkcím a datům, ke kterým mají oprávnění. Implementace zahrnuje middleware pro ověření uživatele a kontrolu vlastnictví při modifikaci dat.

9.3 Ochrana proti běžným útokům

Aplikace implementuje různá opatření proti běžným typům útoků:

- **XSS (Cross-Site Scripting)** - Využití automatického escapování HTML v Reactu a bezpečných komponent pro renderování uživatelského obsahu

- **CSRF (Cross-Site Request Forgery)** - Automatická ochrana v Next.js a správné nastavení atributů cookies
- **Injection útoky** - Použití parametrizovaných dotazů v Prisma ORM pro ochranu proti SQL injection
- **Validace vstupu** - Důsledná validace všech uživatelských vstupů pomocí knihovny Zod na klientské i serverové straně

9.4 Bezpečnost externích služeb

Aplikace využívá několik externích služeb, které vyžadují správnou konfiguraci pro zajištění bezpečnosti:

9.4.1 UploadThing

Pro zabezpečení nahrávání souborů jsou implementována tato opatření:

- Omezení velikosti souborů
- Validace typů souborů (pouze obrázky a videa)
- Autentizace uživatelů před nahráváním
- Generování náhodných názvů souborů pro prevenci předvídatelných URL

9.4.2 Stream Chat

Pro zajištění bezpečnosti chat systému jsou implementována tato opatření:

- Tokeny pro autentizaci jsou generovány na serveru s omezenou platností
- Uživatelé mají přístup pouze k chatům, kde jsou členy
- API klíče jsou bezpečně uloženy v proměnných prostředí

9.5 Environmentální proměnné

Citlivé informace jako API klíče, přístupové údaje k databázi a tajné tokeny jsou uloženy v environmentálních proměnných, aby nebyly součástí verzovacího systému.

Proměnné začínající `NEXT_PUBLIC_` jsou dostupné i v klientském kódu, zatímco ostatní jsou dostupné pouze na serveru, což je důležité pro bezpečné nakládání s citlivými údaji.

10 Nasazení

Důležitou součástí vývoje webové aplikace je její nasazení do produkčního prostředí. V této sekci popisuji proces nasazení aplikace na veřejně dostupný server.

10.1 Infrastruktura a technologie

Pro nasazení aplikace jsem zvolil následující technologie:

- **PM2** - Process manager pro Node.js aplikace
- **Nginx** - Webový server a reverse proxy
- **Linux VPS** - Virtuální privátní server s operačním systémem Linux

10.2 Proces nasazení

10.2.1 Příprava aplikace pro produkci

Před nasazením bylo nutné provést několik úprav v kódu a konfiguraci aplikace:

- Nastavení environmentálních proměnných pro produkční prostředí (databázové přístupy, API klíče)
- Optimalizace build procesu pro produkci
- Konfigurace databáze pro produkční prostředí
- Aktualizace autorizovaných URI v Google Cloud Console pro OAuth přihlašování

Zvláště důležitá byla úprava nastavení v Google Cloud Console, kde bylo potřeba přidat novou autorizovanou doménu a callback URL pro Google OAuth:

- Přidání `https://social.prach.xyz` do seznamu autorizovaných JavaScript origins
- Přidání `https://social.prach.xyz/api/auth/callback/google` do seznamu autorizovaných redirect URI

Bez této změny by přihlašování pomocí Google účtu nefungovalo v produkčním prostředí. Build aplikace byl vytvořen pomocí příkazu:

```
npm run build
```

10.2.2 Konfigurace PM2

PM2 je process manager, který zajišťuje běh Node.js aplikací v produkčním prostředí. Umožňuje automatický restart aplikace v případě pádu, load balancing a monitorování výkonu (30). Aplikace byla spuštěna v PM2 pomocí příkazu:

```
pm2 start npm --name "social" -- run start --prefix /home/prach/social/
```

Tento příkaz spustí aplikaci pomocí skriptu `npm run start` v adresáři `/home/prach/social/` a pojmenuje proces v PM2 jako "social" pro snadnější správu. PM2 automaticky zajišťuje běh aplikace a její restart v případě pádu nebo restartu serveru.

10.2.3 Konfigurace Nginx

Pro vystavení aplikace na veřejnou doménu `social.prach.xyz` byl použit Nginx jako reverse proxy. Ten přeposílá požadavky z veřejné domény na lokální port, kde běží aplikace. Konfigurace Nginx:

```
server {
    server_name social.prach.xyz;
    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Tato jednoduchá konfigurace nastavuje Nginx jako reverse proxy, který přeposílá veškerý provoz z domény `social.prach.xyz` na lokálně běžící aplikaci na portu 3000.

10.3 SSL certifikát a Cloudflare

Pro zabezpečenou komunikaci přes HTTPS byly použity spravované SSL certifikáty poskytované službou Cloudflare (31). Na rozdíl od řešení jako Let's Encrypt a Certbot, Cloudflare nabízí jednodušší způsob správy SSL certifikátů jako součást své cloudové služby.

10.3.1 Integrace s Cloudflare

Cloudflare poskytuje několik klíčových služeb pro nasazení aplikace:

- **DNS správa** - Správa DNS záznamů pro subdoménu social.prach.xyz
- **SSL certifikáty** - Automatická správa a obnova SSL certifikátů
- **CDN** - Content Delivery Network pro rychlejší doručování statického obsahu
- **DDoS ochrana** - Základní ochrana proti DDoS útokům

Pro implementaci SSL s Cloudflare bylo nutné:

1. Přidat doménu prach.xyz do Cloudflare
2. Nastavit DNS záznam pro subdoménu social.prach.xyz
3. Aktivovat SSL certifikát v režimu "Full"
4. Nakonfigurovat Nginx pro správné zpracování HTTPS provozu

Díky Cloudflare je správa SSL certifikátu plně automatizovaná a nevyžaduje ruční obnovu.

11 Závěr

V rámci tohoto ročníkového projektu jsem úspěšně vytvořil plnohodnotnou sociální síť, která implementuje všechny požadované funkce a využívá moderní technologie pro vývoj webových aplikací. Podařilo se mi vybudovat komplexní systém, který zahrnuje registraci a přihlašování uživatelů, sdílení příspěvků s multimediálním obsahem, komentáře, hodnocení, ukládání oblíbených příspěvků, sledování uživatelů, real-time chat a systém oznámení.

Aplikace disponuje všemi funkcemi definovanými v zadání:

- Autentizace uživatelů pomocí hesla i Google účtu
- Sdílení příspěvků s textem, obrázky a videi
- Systém lajků, komentářů a ukládání příspěvků
- Sledování uživatelů a personalizovaný feed
- Real-time chat pro komunikaci mezi uživateli
- Systém oznámení pro různé typy interakcí
- Vyhledávání v příspěvcích a uživatelích
- Zobrazování trendujících hashtagů a zmínek uživatelů
- Přepínání mezi světlým a tmavým motivem
- Plně responzivní design pro všechny velikosti zařízení
- Efektivní cachování a optimalizace výkonu

Aplikace je postavena na moderních technologiích s využitím React, Next.js, TypeScript, Prisma, PostgreSQL a dalších knihoven a služeb. Díky serverovým komponentám a optimalizaci je aplikace rychlá a efektivní i při větším zatížení.

11.1 Výzvy a získané zkušenosti

11.1.1 Orientace v novém ekosystému

Jednou z největších výzev na začátku projektu bylo zorientovat se v ekosystému Next.js. Pochopení konceptů, jako jsou Server Components, Client Components, Server Actions a Route Handlers, vyžadovalo počáteční studium a experimentování. Postupně jsem však získal jasnou představu o architektuře a mohl efektivně implementovat jednotlivé funkce.

11.1.2 Správa stavu a mutace cache

Nejnáročnější částí implementace bylo správně navrhnout systém pro práci s asynchronními daty a jejich aktualizaci v reálném čase. Implementace TanStack Query mi poskytla nástroje pro řešení těchto problémů, ale správné použití funkcí `queryClient` a definice závislostí mezi dotazy vyžadovalo hluboké pochopení knihovny a pečlivé testování.

11.1.3 Integrace externích služeb

Integrace služeb jako Stream Chat pro real-time komunikaci a UploadThing pro nahrávání médií představovala další výzvu. Bylo nutné pochopit API těchto služeb, implementovat správnou autentizaci a zajistit bezproblémovou komunikaci mezi aplikací a externími službami.

11.2 Zhodnocení projektu

Projekt sociální sítě PrachSocial považuji za úspěšný - podařilo se mi implementovat všechny požadované funkce a aplikace je plně funkční a připravená pro reálné použití. Je nasazena na veřejně dostupné URL adrese `https://social.prach.xyz`, kde ji lze vyzkoušet.

Během vývoje jsem získal cenné zkušenosti s moderními technologiemi a přístupy ve vývoji webových aplikací. Naučil jsem se efektivně pracovat s Next.js, React Server Components, Prisma ORM a dalšími knihovnami, které představují současný standard v oboru. Osvojil jsem si pokročilé techniky jako optimistické aktualizace UI, správu cache, nekonečné scrollování a práci s real-time API.

Tyto zkušenosti považuji za velmi přínosné v oblasti vývoje webových aplikací. Projekt mi umožnil pracovat na komplexním systému od návrhu po nasazení a řešit reálné problémy, se kterými se setkávají vývojáři v praxi.

Celkově hodnotím práci na projektu jako velmi přínosnou a jsem přesvědčen, že vytvořená aplikace splňuje vysoké standardy moderních webových aplikací z hlediska funkčnosti, uživatelského zážitku i výkonu.

Seznam použitých zdrojů

[1] Vercel, Inc. (2023). *Next.js Documentation*. Získáno z https://nextjs.org/docs

[2] Vercel, Inc. (2023). *Next.js App Router*. Získáno z https://nextjs.org/docs/app

[3] Vercel, Inc. (2023). *Route Groups*. Získáno z https://nextjs.org/docs/app/building-your-application/routing/route-groups

[4] Vercel, Inc. (2023). *Dynamic Routes*. Získáno z https://nextjs.org/docs/app/building-your-application/routing/dynamic-routes

[5] Vercel, Inc. (2023). *Route Handlers*. Získáno z https://nextjs.org/docs/app/building-your-application/routing/route-handlers

[6] Vercel, Inc. (2023). *Project Organization and File Colocation*. Získáno z https://nextjs.org/docs/app/building-your-application/routing/colocation

[7] Vercel, Inc. (2023). *File Conventions*. Získáno z https://nextjs.org/docs/app/api-reference/file-conventions

[8] Meta Platforms, Inc. (2023). *React Documentation*. Získáno z https://react.dev/

[9] Meta Platforms, Inc. (2023). *React Hooks*. Získáno z https://react.dev/reference/react

[10] Prisma (2023). *Prisma Documentation*. Získáno z https://www.prisma.io/docs/

[11] Prisma (2023). *Prisma Schema*. Získáno z https://www.prisma.io/docs/orm/prisma-schema

[12] Lucia Auth (2023). *Lucia Documentation*. Získáno z https://lucia-auth.com/

[13] Stack Overflow (2021). *How to implement debounce in React?* Získáno z https://stackoverflow.com/questions/23123138/how-to-implement-debounce-in-react

[14] React Hook Form (2023). *React Hook Form Documentation*. Získáno z https://react-hook-form.com/

[15] Zod (2023). *Zod: TypeScript-first schema validation with static type inference*. Získáno z https://github.com/colinhacks/zod

[16] Node-RS (2023). *Node-RS Argon2: Rust Argon2 WASM binding for NodeJS*. Získáno z https://github.com/napi-rs/node-rs/tree/main/packages/argon2

[17] Pilcrow (2023). *Arctic: Minimal OAuth for JavaScript*. Získáno z https://github.com/pilcrowOnPaper/arctic

[18] Tiptap (2023). *Tiptap Documentation*. Získáno z https://tiptap.dev/

[19] TanStack (2023). *TanStack Query (React Query) Documentation*. Získáno z https://tanstack.com/query/latest/docs/react/overview

[20] TanStack (2023). *Infinite Queries - Showing pages in reversed order*. Získáno z https://tanstack.com/query/latest/docs/react/guides/infinite-queries#what-if-i-want-to-show-the-pages-in-reversed-order

[21] TanStack (2023). *Optimistic Updates*. Získáno z https://tanstack.com/query/latest/docs/react/guides/optimistic-updates

[22] React Intersection Observer (2023). *React Intersection Observer*. Získáno z https://github.com/thebuilder/react-intersection-observer

[23] Stream (2023). *Stream Chat API Documentation*. Získáno z https://getstream.io/chat/docs/

[24] PostgreSQL (2023). *Full Text Search*. Získáno z https://www.postgresql.org/docs/current/textsearch.html

[25] Next Themes (2023). *Next Themes Documentation*. Získáno z https://github.com/pacocoursey/next-themes

[26] Tailwind Labs (2023). *Tailwind CSS Documentation*. Získáno z https://tailwindcss.com/docs

[27] UploadThing (2023). *UploadThing Documentation*. Získáno z https://docs.uploadthing.com/

[28] Workos (2023). *Radix UI Documentation*. Získáno z https://www.radix-ui.com/docs/primitives/overview/introduction

[29] Shadcn (2023). *Shadcn/UI Components*. Získáno z https://ui.shadcn.com/docs

[30] Bergmann, A. (2023). *PM2 Documentation: Advanced, Production Process Manager for Node.js*. Získáno z https://pm2.keymetrics.io/docs/usage/quick-start/

[31] Cloudflare, Inc. (2023). *Cloudflare SSL/TLS Documentation*. Získáno z https://developers.cloudflare.com/ssl/