Gymnázium Arabská, Praha 6, Arabská 14
Obor programování

Audiosociální síť: Ding

Rory Beneš, Kryštof Breburda, Adam Suchý

Květen, 2023

Prohlašujeme, že jsme jedinými autory 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 udělujeme 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 ..... dne .....

Rory Beneš .....

Kryštof Breburda .....

Adam Suchý .....

Abstrakt

V rámci tohoto ročníkového projektu jsme vytvořili sociální síť Ding, která je vyhrazena pro zvukové nahrávky bez mluveného slova, filtrovaného umělou inteligencí. Ding je webová stránka napsaná v programovacím jazyce JavaScript ve frameworku Vue.js. Serverová část sociální sítě je napsána v jazyce Rust a využívá databázi PostgreSQL.

Obsah

1 Úvod

Ding je sociální síť zaměřená na nahrávky zvuků. Mluvené slovo je na síti zakázáno a automaticky blokováno. Uživatelé se mohou navzájem poslouchat a na své příspěvky odpovídat.

1.1 Původní znění zadání

Cílem by bylo vytvořit sociální síť, na kterou by se nedalo přidat nic jiného než zvukové soubory, které by nesměly obsahovat mluvené slova (nebo by množství bylo omezené). Mluvená slova by se automaticky filtrovala. Aplikace by byla webová stránka.

UI by bylo ve stylu Twitteru, ale upravené pro audio formát. Uživatel by mohl nahrávat audio hned na webové stránce, nebo nahrát už vytvořený soubor. Měl by také feed audia od dalších uživatelů.

Později bychom mohli pomocí elektronu udělat android / IOS aplikaci. Primárně by ale aplikace žila na webu.

1.2 Aplikace

Sociální sítě slouží jako platformy pro sdílení informací, zábavu, komunikaci a sociální interakce online. V dnešní době jsme nimi obklopeni naprosto ze všech stran a každý dobře víme jak taková platforma vypadá, jak přibližně funguje a co od ni můžeme očekávat. Základní funkčnosti spolu většina těchto sítí sdílí: přidávání příspěvků, sledování uživatelů, komentáře atd. Naši sociální síť od těch ostatních rozlišuje primárně obsah jednotlivých příspěvků. Audio.

Ačkoli byly sociální sítě vynalezeny hlavně pro účel rychleji a efektivněji spojovat a propojovat jednotlivé uživatele, dnes už nesou zodpovědnost za mnohem více. Díky obrovskému počtu denodenních uživatelů a zájem obyčejných lidí se staly tyto prostory centrem reklam, kde každá firma a každá společnost soupeří o čas a pozornost svých uživatelů. To dělají nejrůznějšími způsoby.

Prostost audia a zvukových příspěvků je v dnešní době příjemné zjednodušení. Příspěvky jsou v naší aplikaci bez popisků a náhledových obrázků, které by vás chtěli vtáhnout k sobě. Uživatel nikdy neví co přímo očekávat. Každý příspěvek jinak obsahuje viditelné informace ohledně autorského uživatele (profilová fotka a jméno), možnost příspěvku přidat Ding (ekvivalent klasického Like, se kterými se setkáme kdekoliv jinde), či ho okomentovat. Dále lze zobrazit komentáře od ostatních uživatelů, případně jakéhokoliv z nich navštívit na jeho profilu a poslechnout ji všechny jejich příspěvky, nebo si příspěvek uložit do složky Saved Posts, kde ho v seznamu všech vašich ostatních uložených příspěvků kdykoliv zase naleznete.

2 Frontend

Celý frontend naší aplikace je postavený ve Vue.js, moderním JavaScriptovém frameworku určeného k tvorbě uživatelského rozhraní. Zaměřuje se na deklarativní renderování a složení komponent. Pro komunikaci frontendu s backendem pracujeme s REST API, který pro přenos dat využívá základních protokolů a technologií, jako je protokol HTTP, přičemž těla požadavků a odpovědí jsou ve formátu JSON.

2.1 Design UI

Vzhled uživatelského rozhraní je samozřejmě jednou z nejdůležitějších vlastností stránky sociální sítě. Lišit se může v závislosti například na cílové skupině uživatelů, či zaměření sítě. Nicméně, ty nejdůležitější obecné principy jsme museli dodržovat a brát v úvahu i při našem návrhu. Design musí být konsistentní, přehledný a jednoduchý. Když se se stránkou uživatel setká poprvé, nemusí ho nijak zvlášť ohromit, stačí, aby se v ní byl hned schopný zorientovat a bezmyšlenkovitě s ní interagovat. Hlavní funkce a navigace by měly být intuitivní, viditelné a snadno dostupné.

Je nesmírně důležité, aby byl v celé aplikaci zachován jednotný design daných prvků designu (jako jsou např. tlačítka, ikony, či formulářové pole) a aby si tyto prvky se sebou navzájem dobře vizuálně vycházeli. Proto jsme v průběhu projektu vybírali jen z úzkého výběru barev, kterými jsou hlavně odstíny tmavší šedi, které se navzájem doplňujou, nikoli dráždí. Tyto šedé tóny konstrastuje pestrá fialová, kterou využíváme k zbarvení a odlišení některých funkcí stránky. Křiklavý rozpor šedého pozadí a fialových detailů uživateli poukazuje na důležitost těchto funkcí. Touto fialovou se rozhodně šetří, zesiluje tím právě dojem její rarity a důsledně i hodnoty. Právě touto barvou se hlavně chlubí všudypřítomné tlačítko "New Post"(Nový příspěvek), které i svou velikostí je pro uživatelé naproto nepřehlédnutelné.

Vedle barev prkvů je ale samozřejmě důležitý i jejich tvar a viditelný obsah. Chtěli jsme, aby stránka působila přátelsky a přívětivě. Všechna tlačítka, příspěvky a inputové pole jsou zaoblovány, zdají se tak sympatičtější. Na ostrou hranu prakticky napříč celým programem nenarazíte, tento trend ctí i font hlavního loga, se kterým se nový uživatel setkává například hned u vytváření vlastního účtu. I ikony ze sady "heroicons", se kterými jsme se rozhodli pracovat, na oko působí příjemně.

Obrázek 1: Logo Ding

2.2 Router

Vue router je oficiální router pro Vue.js. Důvodem, proč jsme se rozhodli pracovat s Vue router je, že nám umožňuje přepínat mezi jednotlivými stránkami bez nutnosti načítat celou stránku

znovu, což je výhodné pro uživatele, protože se mu stránka bude načítat rychleji, což zlepší celkový dojem z aplikace.

2.3 Komponenty

Komponenty jsou základní stavební jednotkou Vue.js aplikace. Využíváme je k vytvoření uživatelského rozhraní. Komponenty jsou znovupoužitelné a díky tomu nemusíme po každé vytvářet stejné bloky kódu, tudíž šetříme čas a zjednodušujeme si práci. Importujeme je do souboru, kde je můžeme zavolat. Komponenty jsou velmi užitečné pro vytváření opakujících se elementů, jako jsou například příspěvky, komentáře, nebo nastavení uživatelského profilu.

2.4 Vytváření nového příspěvku

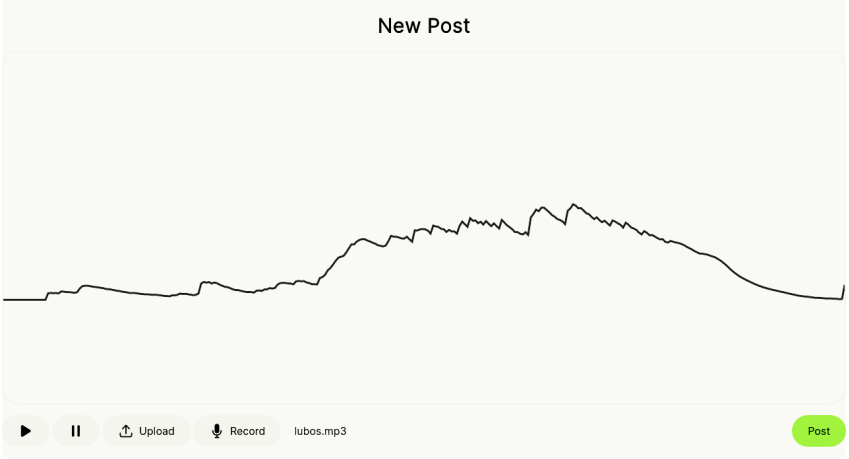
Uživatel může nový příspěvek vytvořit po kliknutí na tlačítko "New Post"na domovské stránce. Po kliknutí na tlkačítko je uživatel přesměrován na stránku, kde může vytvořit nový příspěvek. Uživatel může příspěvek vytvořit dvěma způsoby, buť nahráním audio souboru ze zařízení, nebo nahráním zvuku pomocí mikrofonu. Pokud uživatel začne nahrávat zvuk pomocí mikrofonu, tak se mu zobrazí vizualizér. vizualizér se mu zobrazí i v případě, že nahrává audio soubor ze zařízenní a začne ho přehrávat.

2.4.1 Vizualizér

Vizualizér se uživateli zobrazí ve chvíli, co začne pomocí mikrofonu nahrávat zvuk, či po zahájení přehrávaní nahraného audio souboru. Je vytvořený pomocí "web audio API"a HTML elementu `<canvas>`. Na canvas se vykresluje linka, která mění svou výšku podle hlasitosti zvuku. Ta je získána pomocí `AnalyserNode`. Délka linky je omezená, aby se nezobrazovala mimo canvas. Tohoto jsme docílili pomocí `Queue` datové struktury, která jak název napovídá je fronta. Jakmile se linka dostane na konec canvasu, tak se vykreslovat nepřestane, akorát se zleva postupně stará data odebírají.

AnalyserNode přijímá zvukový signál a generuje výstupní data, která obsahují informace o frekvenčním spektru a hlasitosti signálu. Tyto informace používáme k vizualizaci zvukového signálu skrze metodu `getByteFrequencyData()`, která umožňuje získávání informací o frekvenčním spektru signálu v podobě pole bytů, kde každý prvek pole reprezentuje hlasitost signálu v daném frekvenčním pásmu.[1]

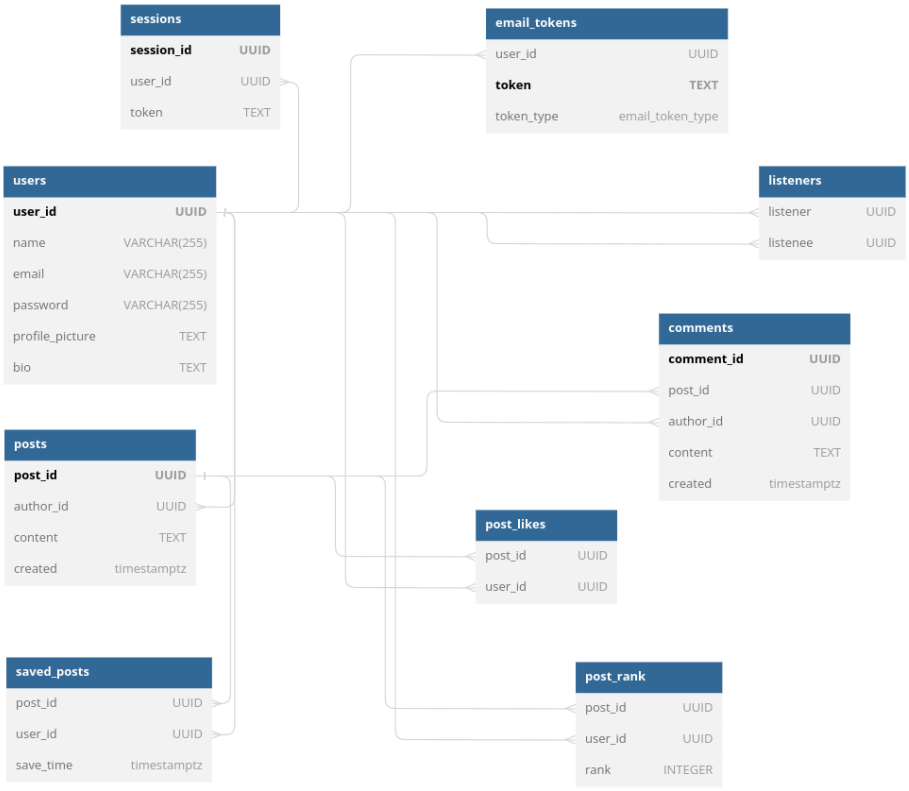
AnalyserNode má také několik vlastností, které ovlivňují jeho chování. Vlastnost `fftSize` určuje množství frekvencí, které FFT (Fast Fourier Transform) algoritmus rozeznává.



Obrázek 2: Vizualizér

3 Backend

Backend, tedy serverová část, je napsána v programovacím jazyce Rust, za využití knihovny Tide, a interaguje s PostgreSQL databází. Značná část logiky a integrita je řešena skrz omezení ve schématu databázových tabulek a skrz ručně napsané SQL příkazy. Frontend se serverem komunikuje pomocí paradigmatu REST API, přes protokol HTTP, přičemž těla požadavků a odpovědí jsou ve formátu JSON.



Obrázek 3: ER diagram

3.1 Databáze

Jako databázi jsme zvolili PostgreSQL kvůli rychlosti a množství funkcí, co dokáže. Celkově má databázové schéma devět tabulek, několik funkcí a náhledů, jak je možno vidět na obrázku 3.

Jedna z výhod PostgreSQL je zabudované full-text vyhledávání, které jsme využili ve funkci vyhledávání uživatelů. Ve vyhledávání pomocí samotného jména uživatele není full-text vyhledávání nejlepší metodou, jelikož pracuje s celými slovy, ale velmi se osvědčilo jako sekundární vyhledávání v celém textu popisku, který si uživatel nastaví.

3.2 AI rozpoznávání mluvy

Jeden z hlavních bodů v zadání bylo automatické filtrování mluveného slova. To je možné provést několika způsoby:

1. Analýza frekvencí ve zvukovém souboru.
    - Existuje mnoho algoritmu na rozpoznání hlasu, ale ne specificky na mluvené slovo.

- Rychlejší, než použití umělé inteligence.

2. Použití umělé inteligence

   (a) Využití jednoho z několika veřejných API na rozpoznávání hlasu.
      - Jestliže je nutné API posílat každý příspěvek, pak bude provoz sítě velmi drahý.

   (b) Využití již natrénovaného modelu na vlastních strojích.
      - Provozovatel sítě musí mít k dispozici stroje, které dokáží příspěvky rychle zpracovat.

   (c) Natrénování vlastního rozpoznávacího modelu a jeho užití na vlastních strojích.
      - Provozovatel musí mít hodně prostředků nejen ke zpracovávání příspěvků, ale také ke trénování modelu.

Pro Ding jsme zvolili variantu 2b. Pro rozpoznávání mluveného slova používáme AI model "Whisper AI"od americké společnosti OpenAI [2]. Tento model je veřejně přístupný a je možné si ho spustit na osobním počítači, nebo serveru. Původní model velmi využívá GPU na grafické kartě počítače. Většina serverů ale grafické karty nemá a tudíž je nutné výpočty provádět na samotném CPU. Tento problém řeší projekt whisper.cpp od Georgiho Gerganova, který model upravil tak, aby ho bylo možné spustit na mnohem slabších zařízeních, než jsou speciální počítače OpenAI.

Při přidání nového příspěvku se tedy nejdříve zvukový soubor zkontroluje, že je funkční a ve správném formátu. Dál se využije AI modelu pro přepsání jakéhokoliv mluveného slova. Pokud je příspěvek v nesprávném formátu, nebo v něm model najde slova, pak je zpětně smazán. Je tedy možné, že na pár vteřin (nebo minut, záleží na zatížení stroje), bude příspěvek přístupný, i když by měl být smazán.

3.3 E-mail a odložené úkoly

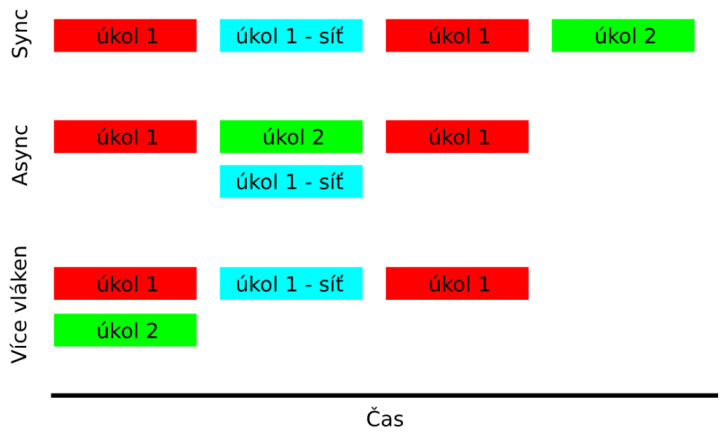
Ding je navržen tak, aby každý požadavek čekal co nejkratší dobu. Například na vyslání e-mailu, nebo kontrolu příspěvku (popisována v sekci 3.2) požadavek nemusí čekat. Knihovna Tide tuto vlastnost nemá zabudovanou a bylo tedy potřeba si vytvořit vlastní systém pro odbavování těchto "odložitelných"úkolů.

Celá aplikace běží v asynchronní smyčce, což umožňuje aplikaci dělání více věcí najednou. Narozdíl od čistého programování s vlákny není paralelizace "pravá". V asynchronní smyčce může být mnoho úkolů, které se mají konat, a procesor mezi nimi přepíná (obrázek 4). V tradičním programování s vlákny beží všechny úkoly najednou na více procesorových jádrech, nebo se o přepínání stará operační systém. Operační systém však neví, kdy se aplikaci hodí, či nehodí, aby procesor přepnul mezi úkoly, a programátor si musí hlídat, aby dvě vlákna neupravovala informace na jednom místě. V asynchronním programování je tento problém značně zredukovaný, jelikož procesor mezi úkoly přepíná jen při klíčovém slově `await`.

3.4 Autentizace uživatelů

Autentizace uživatelů je provedena přes relační tokeny. Aby uživatel dostal token pro svou relaci, musí vyslat požadavek `POST /api/sessions` se svými přihlašovacími údaji (email, heslo). Backend tyto údaje ověří vůči informacím v databázi a jestliže souhlasí, pak vytvoří pro uživatele nový relační token, který složí k autentizaci a identifikaci dalších požadavků. Tento token si prohlížeč ukládá v paměti `window.localStorage`.

Před uložením do databáze jsou hesla zahašována algoritmem `PBKDF2` se 100 000 iteracemi. V databázovém sloupci s heslem je také uložena sůl použita při hašování a informace o hašovacím algoritmu pro případ, že by bylo algoritmus potřeba v budoucnu změnit.



Obrázek 4: Synchronní, asynchronní a vícevláknové programování

3.5 Feed a hodnocení příspěvků

Jedna z nejdůležitějších častí sociálních sítí je jejich hlavní stránka / feed. Je to nekonečný proud příspěvků od uživatelů sítě, algoritmicky vybraných na míru pro každého uživatele zvlášť. Ding není další z mnoha sociálních sítí, které chtějí udržet pozornost uživatele co nejdelší dobu. Věříme, že uživatelé na Dingu spíše zůstanou kvůli skvělému obsahu.

Algoritmus pro hlavní feed tedy není složitý. Při každém požadavku `GET /api/posts/feed` se aktualizuje hodnocení pro nové příspěvky, které se pro požadavek vrací jako odpověď. Toto zaručí, že při dalších požadavcích se nevrací ty samé příspěvky. Další kritérium je čas, kdy byl příspěvek vytvořen. Novější příspěvky jsou algoritmem preferovány.

4 Závěr

Na samotný závěr ročníkového projektu bychom naši práci zhodnotili velmi kladně. Šli jsme do něj jako skupina svými schopnostmi a zkušenostmi velmi nevyrovnaná, s cílem se naučit něco nového. Zkušeností jsme nabrali ohromné množství, nebylo to rozhodně nic jednoduchého, ale na konec se nám podařilo vytvořit aplikaci, se kterou jsme velmi spokojeni.

Odkazy

[1] Mozilla Contributors. *AnalyserNode*. 2023. URL: `https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode` (cit. 01. 05. 2023).

[2] OpenAI. *Whisper: Supervising Graph Neural Networks for Flexible Predicate Prediction*. 2022. URL: `https://openai.com/research/whisper` (cit. 30. 04. 2023).

Seznam obrázků