Vysoké učení technické v Brně
Fakulta informačních technologií

# Projekt do predmetu PDB

## Návrh databázového systému

Firc Anton (xfirca00)
Lapšanský Tomáš (xlapsa00)

# 1 Špecifikácia

V rámci projektu navrhneme a implementujeme databázovú aplikáciu pre sociálnu sieť. Sieť poskytuje služby len registrovaným užívateľom. Každý užívateľ má svoju nástenku kam môže pridávať príspevky, môže chatovať s ostatnými užívateľmi alebo skupinou užívateľov, komentovať alebo označiť, že sa mu páčia príspevky iných užívateľov. Zároveň si každý užívateľ môže vytvoriť neobmedzený počet stránok. Stránky fungujú ako „virtuálny“ užívateľ ktorý môže pridávať príspevky. Administrácia je prístupná len cez profil autora stránky. Každý užívateľ má zároveň k dispozícii vlastný feed v ktorom vidí príspevky od sledovaných užívateľov alebo stránok.

## 1.1 Dostupné operácie

- ukladanie dát
  - vytvorenie nového užívateľa
  - pridanie / zmazanie profilovej fotky užívateľa
  - vytvorenie novej stránky
  - pridanie / zmazanie profilovej fotky stránky
  - sledovanie nového užívateľa/stránky
  - pridanie nového príspevku
  - označenie "páči sa mi"na príspevku/komentári
  - pridanie komentára k príspevku
  - vytvorenie chatu/chatovacej skupiny
  - napísanie správy užívateľovi/chatovacej skupine
- úprava dát
  - úprava vlastného profilu užívateľa / stránky
  - zmazanie užívateľa
  - zmazanie stránky
  - zrušenie sledovania užívateľa/stránky
  - zmazanie príspevku
  - odstránenie "páči sa mi"z príspevku/komentára
  - zmazanie komentára

## 1.2 Dostupné dotazy

- zobrazenie profilu stránky
- zobrazenie aktívnych stránok
- zobrazenie všetkých stránok
- zobrazenie profilu užívateľa
- prihlásenie / odhlásenie
- zobrazenie všetkých aktívnych užívateľov

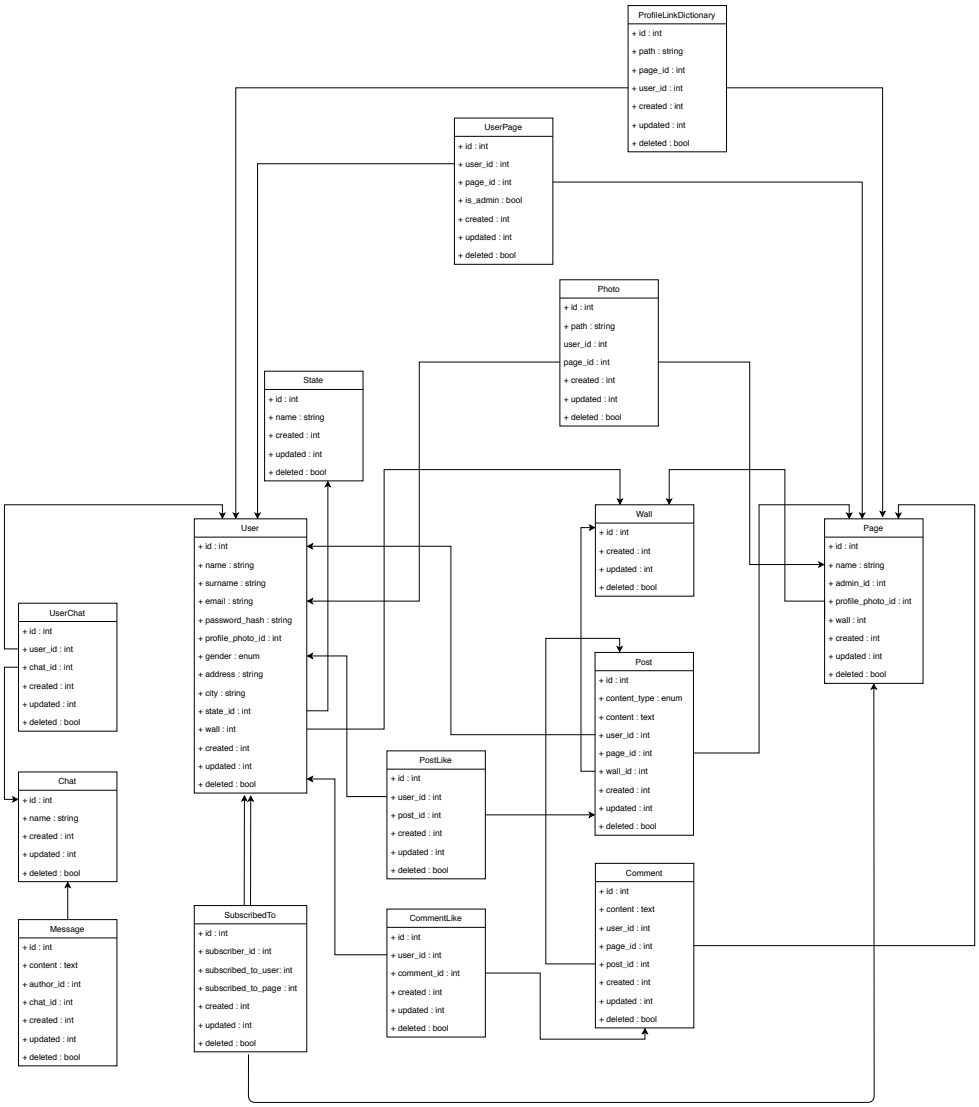
- zobrazenie koho užívateľ sleduje
- zobrazenie kto sleduje užívateľa / stránku
- zobrazenie všetkých štátov
- zobrazenie feed-u
- zobrazenie mojich chatov
- zobrazenie správ v chate

## 1.3 Testy

Testovanie funkčnosti systému bude prebiehať pomocou štandardných unit testov. Pre každú časť systému bude vytvorená samostatná sada testov, overujúca správne vytvorenie, zobrazenie, prípadnú úpravu a zmazanie dát obsluhovaných controllerom zodpovedným za danú časť. Poskytnutie stavu aplikácie, ako napríklad vytvorenie užívateľov pre testovanie chatov, zabezpečia inicializačné metódy jednotlivých testov.

# 2 Analýza

## 2.1 Schéma relačnej databázy




Obr. 1: Schéma relačnej databázy

## 2.2 NoSQL databáza

### Dáta

**Štruktúra** dát uložených v nerelačnej databáze je popísaná v časti 3. Dáta aplikácie s ktorými bude NoSQL databáza pracovať, ich rýchlosť vzniku a platnosť sú popísané v tabuľke 1.

| | Rýchlosť vzniku | Platnosť | Vierohodnosť |
|---|---|---|---|
| Užívatelia | - | 2w | - |
| Stránky | - | 2w | - |
| Slovník: URL na užívateľa / stránku | - | neobmedzená | - |
| Sledovanie užívateľov | 1/d/užívateľ | neobmedzená | - |
| Príspevky | 1/d/užívateľ | 1w | stredná |
| Chat | 1/w/užívateľ | neobmedzená | vysoká |
| Správy | 100/d/užívateľ | 2d | vysoká |
| Komentáre | 5/d/užívateľ | 1w | stredná |
| Páči sa mi | 10/d/užívateľ | 1w | stredná |

Tabuľka 1: Dáta NoSQL databázy, rýchlosť ich vzniku a platnosť

**Zdrojom** všetkých dát v rámci aplikácie je koncový užívateľ a jeho interakcia so systémom (pridávanie príspevkov, komentárov, písanie správ do chatu, ...). Koncový užívatelia budú rovnako ako jediný dáta z nerelačnej databázy **konzumovať**.

**Veľkosť** uložených dát závisí priamo-úmerne od počtu aktívnych užívateľov siete. Predpokladaný počet dát pripadajúcich na jedného užívateľa vyplýva z rýchlosti vzniku konkrétnych dát popísaných v tabuľke 1.

**Dotazy** nad nerelačnými dátami sú podmnožinou dostupných dotazov aplikácie popísaných v časti 1.1:

- zobrazenie profilu stránky
- zobrazenie aktívnych stránok
- zobrazenie profilu užívateľa
- prihlásenie / odhlásenie
- zobrazenie všetkých aktívnych užívateľov
- zobrazenie koho užívateľ sleduje
- zobrazenie kto sleduje užívateľa / stránku
- zobrazenie feed-u
- zobrazenie mojich chatov
- zobrazenie správ v chate

Najvyšší dôraz je kladený na **dostupnosť** dát, tak aby bola služba dostupná v maximálnej možnej miere každému jej užívateľovi. Dostupnosť môže byť zabezpečená aj na úkor vierohodnosti dát, a to hlavne v prípade príspevkov, komentárov a označení "páči sa mi". V ostatných prípadoch, teda pri chatoch a správach, je potreba klásť vyšší dôraz na vierohodnosť dát, tak aby tieto služby fungovali čo najviac v reálnom čase. V prípade príspevkov a komentárov nie je potreba aby boli tieto dáta okamžite dostupné všetkým užívateľom. Najmenší dôraz na vierohodnosť je kladený na označenia "páči sa mi"ktoré budú systémom agregované pre jednotlivé príspevky a zapisované naraz po dávkach v určitom časovom úseku od pridania prvého označenia "páči sa mi".

**Škálovateľnosť** hrá dôležitú rolu vzhľadom na vysoký počet používateľov sociálnych sietí. Je preto potreba zaistiť aby aj pri maximálnom využití zo strany užívateľov dochádzalo k poskytovaniu kvalitných služieb.

Údaje o užívateľoch a stránkach budú v rámci NoSQL databázy ukladané len pre nedávno aktívnych užívateľov (maximálne 2 týždne bez prihlásenia). Po prihlásení do systému bude užívateľovi zobrazený feed len od sledovaných užívateľov alebo stránok ktoré sú vedené ako aktívne. Príspevky vo feede budú radené podľa dátumu zverejnenia.

Podľa typu dát a vykonanej operácie bude prebiehať aktualizácia nerelačných dát. V prípade chatov a správ budu tieto dáta okamžite prenášané do NoSQL databázy a medzi všetkými jej clustrami. Príspevky a komentáre budú rovnako nahrávane okamžite, ale nemusia byť hneď dostupné. Označenia "páči sa mi"budú nahrávané v dávkach tak, ako bolo uvedené vyššie v tejto sekcii.

# 3 Návrh

Pre implementáciu relačnej databázy využijeme systém *MariaDB* nakoľko je dostupná ako open-source projekt, masovo využívaná a máme s týmto systémom predošlé kladné skúsenosti. Pre implementáciu nerelačnej databázy využijeme systém *Apache Cassandra* ktorý by mal poskytovať dobrú lokalitu uložených dát a teda poskytovať potrebnú rýchlosť pre našu aplikáciu.

Samotná databázová aplikácia bude implementovaná v jazyku *Java* a bude poskytovať dáta pomocou REST API.

### Schéma NoSQL databázy

```
type comment (
    ident bigint,
    author_name text,
    author_profile_url text,
    author_picture_url text,
    content text,
    comment_likes list<frozen<like>>,
    created_at timestamp
);


type like (
    ident bigint,
    author_name text,
    author_profile_url text,
    author_picture_url text,
    created_at timestamp
);

CREATE TABLE page {
    id bigint,
    admin_email text,
    last_active timestamp,
    name text,
    profile_path text,
    profile_photo_path text,
    PRIMARY KEY (id)
}
```

```
CREATE TABLE page_follows {
    follows_id bigint,
    user_email text,
    created_at timestamp,
    PRIMARY KEY (follows_id, user_email)
}
```

* *Who follows page with id : follows_ id*

```
CREATE TABLE page_post {
    page_id bigint,
    content_type text,
    content text,
    page_name text,
    comments list<comment>,
    likes list<like>,
    created_at timestamp,
    PRIMARY KEY ((page_id, content_type), created_at)
}

CREATE TABLE profile_link_dictionary {
    path text,
    page_id bigint,
    use_email text,
    PRIMARY KEY (path)
}

CREATE TABLE user (
    email text,
    name text,
    surname text,
    password_hash text,
    profile_path text,
    profile_photo_path text,
    owned_pages list<int>,
    last_active timestamp,
    status bool,
    created_at timestamp,
    PRIMARY KEY (email)
)

CREATE TABLE user_follower {
    user_email text,
    follower_email text,
    follower_id bigint,
    created_at timestamp,
    PRIMARY KEY (user_email, created_at)
}

CREATE TABLE user_follows {
    follows_email text,
    user_email text,
    created_at timestamp,
```

```
    PRIMARY KEY (follows_email, user_email)
}

CREATE TABLE chat {
    id uuid,
    name text,
    last_updated timestamp,
    PRIMARY KEY (id)
}

CREATE TABLE user_post {
    user_email text,
    user_profile_path text,
    content_type text,
    content text,
    comments list<comment>,
    likes list<like>,
    created_at timestamp,
    PRIMARY KEY ((user, content_type), created_at)
}

CREATE TABLE user.chat {
    user text,
    chat uuid,
    from timestamp,
    PRIMARY KEY (user, chat)
}

CREATE TABLE chat.message {
    from_user text,
    chat uuid,
    message text,
    time timestamp,
    PRIMARY KEY ((chat, from_user), time)
}
```

**Testovanie** bude pokryté unit testami v rámci Java prostredia. Testy budú spúšťané vždy po implementácii novej funkcionality alebo úprave už existujúcej.

**UserTest** : *Testovanie vytvorenia užívateľa, zobrazenie jeho profilu a zmazanie. Spojené s testovaním vytvorenia a vymazania štátov, ako predpoklad tvorby užívateľa.*
Kroky testu:

1. Vytvorenie nového štátu
2. Vytvorenie nového užívateľa
3. Zobrazenie profilu užívateľa
4. Prihlásenie
5. Odhlásenie

6. Zobrazenie všetkých užívateľov
7. Zmazanie užívateľa
8. Zmazanie štátu

**PageTest** : *Testovanie vytvorenia a zmazania stránky.*
Kroky testu:

1. Vytvorenie stránky
2. Zobrazenie profilu stránky
3. Zobrazenie profilu užívateľa a kontrola priradených stránok
4. Zmazanie stránky

**SubscriptionTest** : *Testovanie sledovania stránky a užívateľa.*
Kroky testu:

1. Sledovanie užívateľa
2. Sledovanie stránky
3. Zobrazenie sledovateľov užívateľa
4. Zobrazenie sledovateľov stránky
5. Zobrazenie koho užívateľ sleduje
6. Zrušenie sledovania užívateľa
7. Zrušenie sledovania stránky

**PostTests** : *Testovanie pridania príspevku užívateľom a stránkou a jeho následné zmazanie.*
Kroky testu:

1. Pridanie príspevku užívateľom
2. Pridanie príspevku stránkou
3. Zobrazenie profilu užívateľa a kontrola príspevku
4. Zobrazenie profilu stránky a kontrola príspevku
5. Zmazanie príspevku užívateľa
6. Zmazanie príspevku stránky
7. Zobrazenie profilu užívateľa a kontrola zmazania príspevku
8. Zobrazenie profilu stránky a kontrola zmazania príspevku

**ChatTest** : *Testovanie užívateľských správ.*
Kroky testu:

1. Zaslanie správy
2. Vytvorenie chatovacej skupiny
3. Zobrazenie chatov

# 4 Implementácia

## 4.1 Popis implementácie

Systém je implementovaný v jazyku Java 11, za použitia frameworku *SpringBoot*. Systém poskytuje REST API pre prácu s databázovým systémom poskytujúcim dáta pre sociálnu sieť.

Ako databázové systémy vyžíva aplikácia systémy *Maria DB* a *Apache Cassandra*. Pre zaistenie prenosu informácií medzi relačnou a NoSQL časťou systém používa dva postupy. *Okamžitý zápis do oboch systémov* a *riadenie udalosťami*. Okamžitý zápis do oboch systémov je použitý v prípade tvorby užívateľov a chatov. Takto je zabezpečená okamžitá konzistencia medzi doménovými modelmi. Udalosťami je riadený zvyšok systému. Riadenie udalosťami obsluhuje služba *Apache Kafka*. Implementácia v základe využíva tri brokery, ale tento počet je možné meniť. Týmto spôsobom je možné aplikáciu následne rozdeliť na dve „služby“, kde jedna obsluhuje operácie a druhá dotazy nad databázou. Za týmto účelom sú controllery rozdelené do dvoch častí, ktoré môžu byť samostatne použité pre vyššie uvedené riešenie. Ako už bolo uvedené, špeciálnou časťou systému sú *UserController* a *ChatController*, ktorí nepoužívajú riadenie udalosťami, takisto ako časti relačných dát ktoré nie sú prenášané do NoSQL časti aplikácie, a to konkrétne: API pre ukladanie fotiek a API pre správu štátov.

## 4.2 Spustenie aplikácie

Prerekvizity:

- Apache Kafka
- Apache Zookeeper
- Cassandra databáza
- Maria DB databáza

Všetky prerekvizity spolu s použitými verziami sú obsiahnuté v rámci docker prostredia, ktorého konfiguračný súbor sa nachádza v koreňovej zložke projektu.

Pred spustením aplikácie, je potreba konfigurovať cluster pre Kafku. Cluster, aj topic-y boli vytvorené za použitia východzích nastavení. Názvy použitých topicov ktoré je potreba vytvoriť sú nasledovné:

- comment-like-service-event
- comment-service-event
- like-service-event
- page-service-event
- post-service-event
- subscription-service-event
- user-service-event

Pre všetky topic-y bolo použité rozdelenie: *partitions: 3, replicas: 2*. Je dostupný aj Kafka-manager, bežiaci na porte 9000.

Samotná aplikácia využíva k prekladu *Maven*, pre spustenie je potreba nastaviť v konfiguračnom súbore *application.properties* IP adresy databáz a prihlasovacie údaje, spoločne s IP adresami dostupných Kafka brokerov a portu na ktorom bude aplikácia dostupná (východzie nastavenie je 8080). Následne volaním cieľa *mvn install* bude aplikácia preložená, a skompilovaný *war* súbor môže byť nasadený.

Inicalizačné skripty pre databázu sa nachádzajú rovnako v koreňovej zložke, pre každú databázu samostatne. Následne je pre obe databázy poskytnutý skript pre nahranie demo dát.

## 4.3 Demo rozhranie

Pre testovanie funkčnosti POST a DELETE dotazov na API, je k dispozícii sada stránok obsahujúcich formuláre pre tvorbu a úpravu dát na nasledujúcich adresách:

- /demo/user
- /demo/page
- /demo/post
- /demo/subscribe
- /demo/chats

## 4.4 REST API

Zoznam dostupných REST API endpointov:

- operácie
  - [POST] /user/create (String UserCreateDto)
  - [POST] /user/delete (String email)
  - [POST] /user/addProfilePic (String email, MultipartFile file)
  - [POST] /user/removeProfilePic (String email)
  - [POST] /user/update (String UserCreateDto)
  - [POST] /user/pass (String email, String password)
  - [POST] /page/create (String email, String name)
  - [POST] /page/profilePic (Long id, MultipartFile file)
  - [POST] /page/profilePic (Long id)
  - [POST] /user/post/create (String email, String contentType, String textContent)
  - [POST] /user/post/delete (String email, String contentType, String createdAt)
  - [POST] /page/post/create (Long pageId, String contentType, String textContent)
  - [POST] /page/post/delete (Long pageId, String contentType, String createdAt)
  - [POST] /post/like (String likeGiverMail, Long pageId, String userMail, String contentType, String createdAt)
  - [POST] /post/comment (String commentGiverMail, String content, Long pageId, String userMail, String contentType, String createdAt)

  - [POST] /comment/like (String liekGiverMail, Long postPageId, String postUsrMail, String contentType, String createdAt, Long commentId, String commentUserMail)
  - [POST] /subscribe/user (String email, String subscribes)
  - [POST] /unsubscribe/user (String userEmail, String unsubscribeFromEmail)
  - [POST] /subscribe/page (String email, Long subscribes)
  - [POST] /unsubscribe/page (Long pageId, String email)
  - [POST] /state/create (String name)
  - [POST] /state/delete (Long stateId)
  - [POST] /photo/upload/user (String ownerEmail, MultipartFile file)
  - [POST] /photo/upload/page (Long ownerId, MultipartFile file)
  - [POST] /chats/groups/create (String name)
  - [POST] /chats/groups/add (Long group, String email)
  - [POST] /chats/messages/send (String author, String receiver, Long chatId, String content)
- dotazy
  - [POST] /user/login (String email, String password)
  - [POST] /user/logout (String email)
  - [GET] /user/{profileSlug}
  - [GET] /users/active
  - [GET] /users/all
  - [GET] /page/{profileSlug}
  - [GET] /pages/active
  - [GET] /pages/all
  - [GET] /subscribers/user/{slug}
  - [GET] /subscribers/page/{slug}
  - [GET] /subscribed-to/{slug}
  - [GET] /states
  - [GET] /wall/{email}
  - [GET] /chats/profileSlug
  - [GET] /chats/messages/id