Индивидуальный проект для курса «Базы данных»

Кунин Илья, БПИ205

Кому нужна разрабатываемая программа, как она будет использоваться

Разрабатываемой программой является мессенджер. Программа предназначена для переписки пользователей в чатах. При регистрации каждый пользователь указывает базовые данные о себе: логин, пароль, имя, ник, фото и т.д. Пользователи могут отправлять текстовые сообщения с медиа вложениями, редактировать и удалять сообщения, оценивать сообщения других пользователей, добавлять друг друга в друзья. Также, пользователи имеют возможность настраивать приложение, при этом настройки сохраняются в том числе на серверной части приложения.

Функциональные требования

1. Хранить информацию о пользователях:
    a. логин
    b. пароль
    c. имя
    d. телефон
    e. адрес электронной почты
    f. дату рождения
    g. дату регистрации
    h. страна
    i. фотография
    j. друзья (контакты)
    k. запросы на добавление в друзья
    l. в каких чатах есть новые сообщения
    m. настройки аккаунта:
        i. закрытый / публичный аккаунт
        ii. включены ли уведомления
        iii. тип сетевого соединения
        iv. тема (светлая / темная)

2. Хранить информацию о чатах:
    a. создатель
    b. администраторы
    c. пользователи
    d. закрепленное сообщение
    e. дата создания
    f. название чата
    g. фотография (аватарка)
    h. список сообщений
        i. автор
        ii. является ли сообщение ответом на другое сообщение
        iii. текст
        iv. дата и время отправки
        v. ссылка на приложенный медиа файл(-ы)
        vi. кому понравилось сообщение (лайки)

3. Программа должна позволять пользователям:
    a. создавать аккаунт (логин, пароль, имя, телефон, email)
    b. редактировать данные аккаунта (имя, описание, телефон и т.д.)
    c. редактировать настройки приложения (уведомления, тема, тип соединения)
    d. удалять аккаунт

    e. создавать чаты
    f. изменять описание / фотографию чата
    g. отправлять / редактировать / удалять / закреплять сообщения в чаты(-ах)
    h. добавлять / удалять администраторов чата
    i. удалять чаты

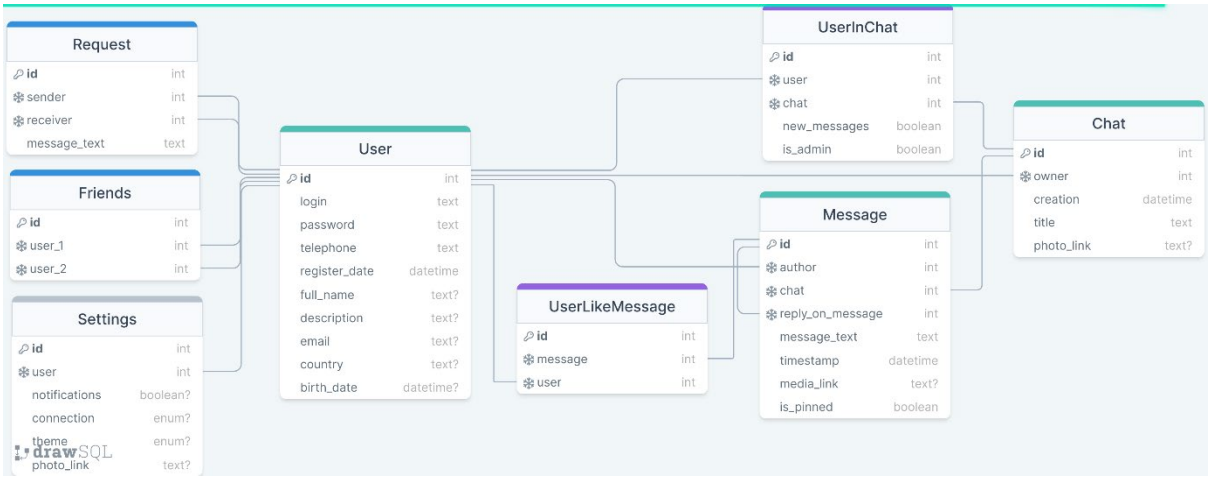
    j. искать других пользователей
    k. отправлять / принимать / отклонять запрос на добавление в друзья
    l. удалять пользователей из друзей

    m. оценивать сообщения других пользователей
    n. видеть в каких чатах есть новые сообщения

4. Ограничения на данные:
    a. большинство текстовых данных ограничены максимум 1000 символами, у некоторых полей меньше
    b. значения полей connection и theme настроек пользователя ограничены соответствующими enum
    c. удаленный (несуществующий) чат не может содержать пользователей и сообщений
    d. удаленный (несуществующий) пользователь не может быть автором сообщений, состоять в чатах и быть в контактах других пользователей

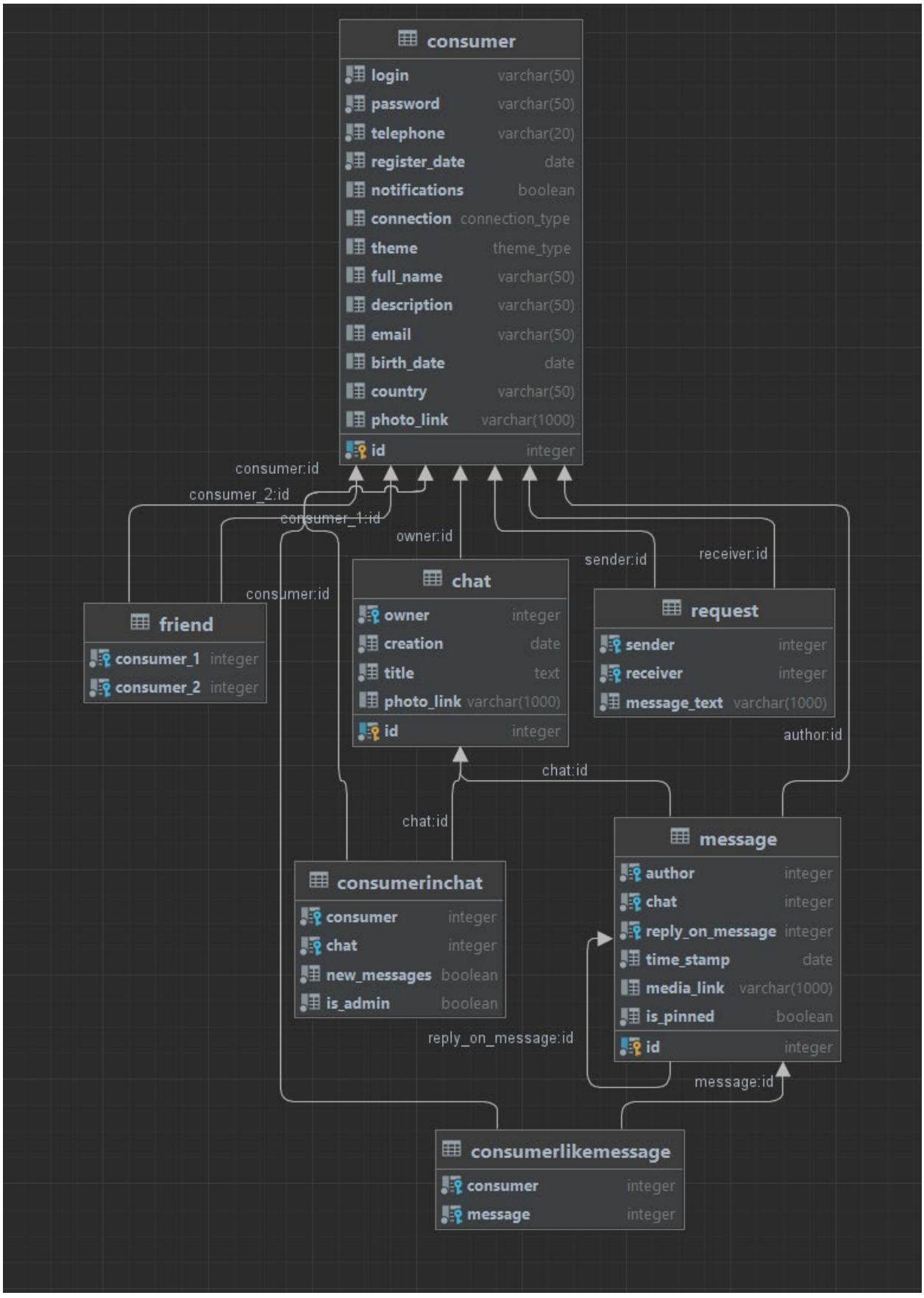
5. В проекте реализована концепция hard delete данных.

UML диаграмма базы данных



https://drawsql.app/teams/main-7/diagrams/messenger

DataGrip диаграмма базы данных



SQL инициализация таблиц

```sql
-- dialect Postgres SQL

DROP TABLE IF EXISTS Request;
DROP TABLE IF EXISTS Friend;
DROP TABLE IF EXISTS ConsumerInChat;
DROP TABLE IF EXISTS ConsumerLikeMessage;
DROP TABLE IF EXISTS Message;
DROP TABLE IF EXISTS Chat;
DROP TABLE IF EXISTS Consumer;

DROP TYPE IF EXISTS connection_type;
DROP TYPE IF EXISTS theme_type;

CREATE TYPE connection_type AS ENUM ('disabled', 'system', 'custom');

CREATE TYPE theme_type AS ENUM ('default', 'dark', 'special');

-- User is reserved keyword, so table called Consumer
CREATE TABLE Consumer (
    -- main fields
    id serial primary key NOT NULL,
    login varchar(50) NOT NULL,
    password varchar(50) NOT NULL,
    telephone varchar(20) NOT NULL,
    register_date date NOT NULL,

    -- settings
    notifications boolean,
    connection connection_type,
    theme theme_type,

    -- nullable fields
    full_name varchar(50),
    description varchar(50),
    email varchar(50),
    birth_date date,
    country varchar(50),
    photo_link varchar(1000)
);

CREATE TABLE Chat (
    id serial primary key NOT NULL,
    owner serial NOT NULL,
    creation date NOT NULL,
    title text NOT NULL,
    photo_link varchar(1000),

    foreign key (owner) references Consumer(id)
);
```

```sql
CREATE TABLE Message (
    id serial primary key NOT NULL,
    author serial NOT NULL,
    chat serial NOT NULL,
    reply_on_message int,
    message_text varchar(1000) NOT NULL,
    time_stamp date NOT NULL,
    media_link varchar(1000),
    is_pinned boolean NOT NULL,

    foreign key (author) references Consumer(id),
    foreign key (chat) references Chat(id),
    foreign key (reply_on_message) references Message(id)
);

CREATE TABLE Request (
    sender serial NOT NULL,
    receiver serial NOT NULL,
    message_text varchar(1000) NOT NULL,

    foreign key (sender) references Consumer(id),
    foreign key (receiver) references Consumer(id)
);

CREATE TABLE Friend (
    consumer_1 serial NOT NULL,
    consumer_2 serial NOT NULL,

    foreign key (consumer_1) references Consumer(id),
    foreign key (consumer_2) references Consumer(id)
);

CREATE TABLE ConsumerLikeMessage (
    consumer serial NOT NULL,
    message serial NOT NULL,

    foreign key (consumer) references Consumer(id),
    foreign key (message) references Message(id)
);

CREATE TABLE ConsumerInChat (
    consumer serial NOT NULL,
    chat serial NOT NULL,
    new_messages boolean NOT NULL,
    is_admin boolean NOT NULL,

    foreign key (consumer) references Consumer(id),
    foreign key (chat) references Chat(id)
);
```

SQL запросы

```sql
-- Сообщения чата с id = 1
SELECT login as message_sender, message_text, media_link, is_pinned,
reply_on_message, COUNT(likes) as likes
FROM Message
JOIN Consumer consumers on message.author = consumers.id
LEFT JOIN ConsumerLikeMessage likes on message.id = likes.message
WHERE Message.chat = 1
GROUP BY Message.id, login, message_text, media_link, is_pinned,
reply_on_message, time_stamp
ORDER BY time_stamp;

-- Добавление нового пользователя
BEGIN TRANSACTION;
INSERT INTO public.consumer (id, login, password, telephone, register_date,
notifications, connection, theme, full_name, description, email, birth_date,
country, photo_link)
VALUES (DEFAULT, 'login', 'password', '8 495 111-11-11', now(), null,
'system', null::theme_type, 'Full Name', null, 'email@website.com', null,
null, null);
COMMIT TRANSACTION;

-- Удаление пользователя с id 1
BEGIN TRANSACTION;
UPDATE Chat set owner = 0 WHERE owner = 1;  -- Set SuperUser as owner
DELETE FROM ConsumerLikeMessage WHERE consumer = 1;
DELETE FROM ConsumerInChat WHERE consumer = 1;
DELETE FROM Message WHERE author = 1;
DELETE FROM Request WHERE sender = 1;
DELETE FROM Request WHERE receiver = 1;
DELETE FROM Friend WHERE consumer_1 = 1;
DELETE FROM Friend WHERE consumer_2 = 1;
DELETE FROM Consumer WHERE id = 1;
COMMIT TRANSACTION;

-- Создать чат
BEGIN TRANSACTION;
INSERT INTO chat (id, owner, creation, title, photo_link)
VALUES (DEFAULT, 2, now(), 'New chat', null);
COMMIT TRANSACTION;

-- Добавить пользователя в чат и сделать его администратором
BEGIN TRANSACTION;
INSERT INTO ConsumerInChat (consumer, chat, new_messages, is_admin) VALUES
(3, 1, FALSE, FALSE);
UPDATE ConsumerInChat SET is_admin = TRUE WHERE consumer = 3;
COMMIT TRANSACTION;

-- Поиск пользователя по началу никнейма gamer_
SELECT * FROM consumer
WHERE position('gamer_' in login) > 0;
```

```sql
-- Отправить заявку на добавление в друзья от пользователя 1 пользователю 2,
после чего пользователь 2 принимает заявку
BEGIN TRANSACTION;
INSERT INTO Request (sender, receiver, message_text) VALUES (1, 2, 'Text');
DELETE FROM Request WHERE sender = 1 AND receiver = 2;
INSERT INTO Friend (consumer_1, consumer_2) VALUES (1, 2);
COMMIT TRANSACTION;

-- Показать чаты, где у пользователя 1 есть новые сообщения
SELECT * FROM Chat
JOIN ConsumerInChat consumerIn on chat.id = consumerIn.chat
WHERE consumerIn.consumer = 1 AND consumerIn.new_messages = TRUE;
```