Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “Бази даних”

“Портал смішних картинок”

Керівник: доц. каф. ПІ Мазурова О.О.

Студент гр. ПІ - 15 – 2 Манаков К.А.

Комісія:

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дудар З.В.

Харків 2016

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Харківський національний університет радіоелектроніки\_\_\_\_\_\_\_\_\_\_

**Кафедра**\_\_\_Програмної інженерії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дисципліна**\_\_\_Бази даних \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Спеціальність** \_Програмна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Курс** \_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_ **Група** \_\_\_ПІ-15-2\_\_\_\_\_\_ **Семестр**\_\_\_\_ 3\_\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**на курсову роботу студента**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Манаков Костянтина Андрійовича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**1. Тема роботи:** «Портал смішних картинок»»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Строк здачі закінченої роботи** \_\_\_23.12.2016\_\_\_\_\_\_\_\_

**3. Вихідні дані для роботи:** методичні вказівки до виконання курсової роботи, вимоги до інформаційної системи, предметна область, що пов’язана *з порталом смішних картинок.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4. Зміст розрахунково - пояснювальної записки:** вступ, аналіз предметної області; постановка задачі; проектування бази даних; опис програми; висновки; перелік посилань.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5. Перелік графічного матеріалу:** загальна схема концептуальної моделі, ER-діаграма, структура 1НФ, 2НФ, 3НФ, схема БД в 3НФ, UML-діаграми, копії екранів (“скріншоти”) прикладної програми, приклади звітів прикладної програми\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6. Дата видачі завдання** \_\_\_09.09.16 р.\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Аналіз предметної області | 9.09.16 – 24.09.16 |  |
| 2 | Постановка задачі | 20.09.16 – 30.09.16 |  |
| 3 | Побудова ER-діаграми бази даних | 27.09.16 – 15.10.16 |  |
| 4 | Оформлення розділів 1, 2 та 3.1, 3.2 пояснювальної записки | 15.10.16 - 27.10.16 |  |
| 5 | Перша контрольна точка з курсового проекту | 24.10.16 – 28.10.16 |  |
| 6 | Нормалізація бази даних | 20.10.16 - 10.11.16 |  |
| 7 | Створення демо-версії програми | 20.10.16 – 20.11.16 |  |
| 8 | Тестування програми, наповнення бази даних | 15.11.16 - 25.11.16 |  |
| 9 | Друга контрольна точка з курсового проекту | 21.12.16– 02.12.16 |  |
| 10 | Реалізація остаточної версії програми | 1.12.16-15.12.16 |  |
| 11 | Оформлення інших розділів пояснювальної записки | 1.11.16 – 15.12.16 |  |
| 12 | Захист курсового проекту (третя контрольна точка) | 12.12.16- 23.12.16 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_ *доц. Мазурова О.О.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 р.

РЕФЕРАТ

Пояснительная записка к курсовой работе: с., рис., приложение, источников.

Целью курсовой работы является создание информационной системы «Портал смешных картинок».

Методы разработки основываются на использовании среды разработки JetBrains WebStorm, языка программирования TypeScript 2.0, компилятора активов Webpack, среды исполнения Node.js, backend-фреймворка Koa.js 2.0, frontend MVVM фреймворка VueJS. В качестве СУБД используется MySQL и ORM Sequelize.

Профиль на github: <https://github.com/monyk015>

В результате получена программе под названием “Портал смешных картинок“, которая позволяет просматривать посты, фильтровать по произвольному набору тэгов, оценивать, комментировать, оценивать комментарии, а так же выкладывать свои собственные посты с произвольным набором тэгов.

ПРОГРАММА, ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ, СУБД, MYSQL, VUEJS, KOAJS, RELATION, TYPESCRIPT, ORM, QUERY, NODEJS, SQL, INTELLIJ, МЕМ, ПОСТ, КОММЕНТАРИЙ

1 АНАЛИЗ И КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Мы живем в эпоху, когда человек научился удовлетворять абсолютно все свои потребности, кроме одной. Потребности в свежих смешных картинках в больших количествах прямо сейчас.

Очевидно, что вкусы на контент у всех людей разные и нужна удобная система для того, чтобы каждый мог получать именно то, что интересно ему и при этом ничего не упустить.

Кроме того, важно чтобы лента каждого пользователя формировалась на основе популярности тех или иных постов и оценок, которые им дают другие пользователи.

Отсюда необходимость в удобной и наглядной системе оценивания и просмотра оценок. И, конечно, нельзя забывать о простоте добавления контента.

Как итог, получаем такой набор сущностей приложения:

1. Собственно, сам пост. Нужно учитывать, что контент поста состоит из текста и картинки. Так же у поста есть оценка, которая считается как разность лайков и дизлайков. Их в качестве отдельных сущностей не храним.
2. Для того, чтобы легко определить тематику поста, каждый пост может иметь произвольное число тегов, так же как и на одном теге может находится произвольное число постов. Как только какой-либо тег впервые указан при создании поста, для него сразу создается запись.
3. Комментарии. Все просто: один пост, много комментариев. У каждого комментария также есть оценка по тому же принципу.
4. Пользователи. У пользователя много связей. Первое, нужно хранить все посты, которые этот пользователь создал. Также считаем рейтинг на основе суммарной оценки всех его постов. Для того, чтобы создать уникальную ленту, в которой есть интересующие конкретно этого пользователя посты, создаем подписки, а чтобы исключить посты под определенным тегом из всех его лент, держим информацию о заблокированных тегах. Также любой пользователь может быть администратором и иметь дополнительные возможности.

Используя эту систему можно:

1. Создавать персональную выборку пользователя по интересам.
2. Просматривать разные ленты постов по популярности, среди которых: “Все”, “Хорошее”, ”Лучшее”, ”Подписки”.
3. Добавлять посты
4. Оценивать посты
5. Комментировать посты
6. Оценивать комментарии
7. Помечать посты тегами
8. Редактировать посты
9. Осуществлять поиск постов по множеству тегов
10. Оформлять подписку на тег
11. Блокировать тег(исключить появление соответствующих постов из всех лент)
12. Удалять свои посты(администратор может удалять любые)

Резюмируя, “Портал смешных картинок” обеспечивает полноценную систему для создания и просмотра интересного персонализированного контента и выражения мнения об этом контенте.

2 ПОСТАНОВКА ЗАДАЧИ

Система должна отображать данные:

1. Общая лента постов «Все», которая включает абсолютно все добавленные посты, исключая те, которые забанил пользователь;
2. Общая лента постов «Хорошее», которая включает все посты, у которых рейтинг выше 20 и которые не забанил пользователь;
3. Общая лента постов «Лучшее», которая включает все, у которых рейтинг выше 100 и которые не забанил пользователь;
4. Выборка постов по конкретному тегу;
5. Персональная лента постов, которая включает посты в тех тегах, на которые пользователь подписан и не включает тех тегов, которые он забанил;
6. Комментарии по любому посту и их рейтинг;
7. Рейтинг поста;
8. Публичный профиль пользователя с рейтингом и списком всех его постов;
9. Фильтр постов в ленте по подмножеству любых параметров;

Система должна выводить следующую статистику для администратора:

1. Суммарный и средний рейтинг каждого из тегов;
2. Количество комментариев и постов за все время по дням;

Система должна поддерживать следующие операции:

1. Создание поста;
2. Удаление поста администратором или создателем;
3. Создание комментария;
4. Удаления комментария администратором или создателем;
5. Регистрация пользователя;
6. Бан пользователя администратором;

Система должна поддерживать следующие операции автоматизации:

1. Автоматическое оповещение пользователя по почте в случае появления нового лучшего поста в одной из подписок
2. Автоматическое создание тега в случае, если его не существовало раньше, при создании поста

3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

3.1 Описание функциональной структуры системы

Описание функциональной структуры подаю в виде Use Case диаграммы, которая показывает основные функции системы со стороны пользователя и администратора.

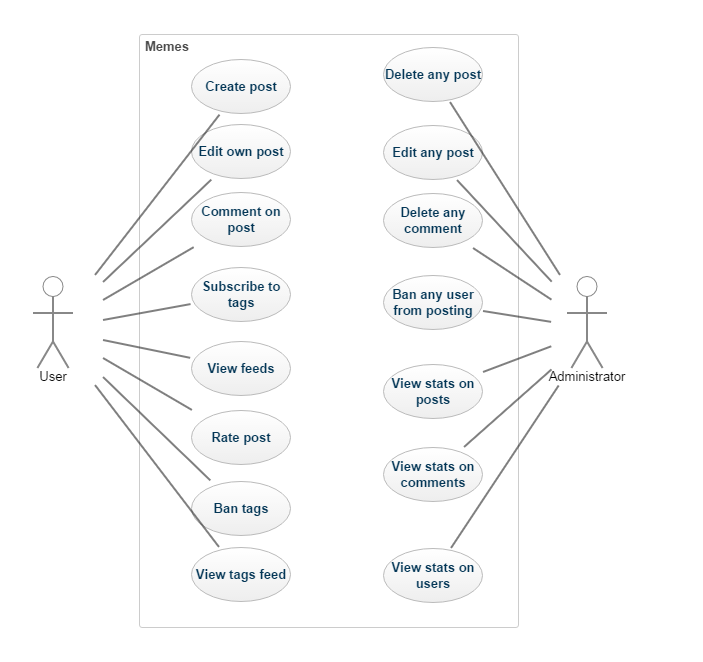


Рисунок 3.1 – Use Case диаграмма функционала Портала

3.2 Описание объектов и связей между ними

В системе существует 5 основных объектов: пользователь, пост, комментарий, тег, лента.

Пользователь просматривает ленту, где видит посты и их теги. Также может перейти к конкретному тегу или посту с комментариями к нему.

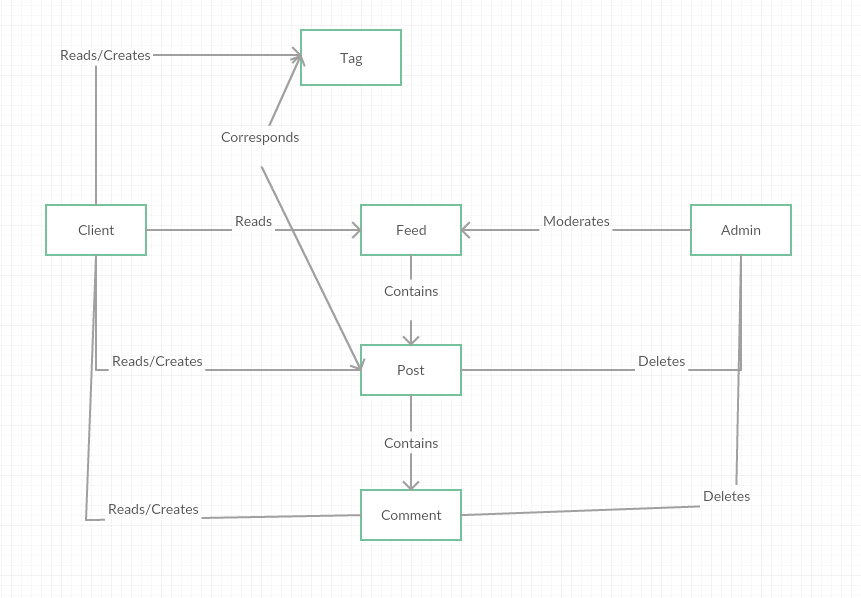


Рисунок 3.2 – Use Case действий объектов

3.3 Описание информационных требований пользователей

* 1. Клиент может формировать свою персональную ленту постов из подписок и банов, и просматривать ее, а также просматривать ленту каждого из конкретных тегов, создавать свои посты и оставлять под любыми постами комментарии;
  2. Админ может удалять любые посты, удалять комментарии, просматривать статистические данные, банить пользователей (разрешать только чтение);

3.4 Ограничения целостности

Ограничения по идентефикации объектов:

1. Комментарии идентифицируются уникальным номером;
2. Посты идентифицируются уникальным номером;
3. Пользователи идентифицируются уникальным номером, почтовый адрес совпадать не может, никнейм может;
4. Имена тегов совпадать не могут, но они все равно идентифицируются уникальным номером;

Ограничения отношений:

1. Каждый комментарий принадлежит одному посту;
2. Каждый пост принадлежит одному пользователю;
3. Каждый пользователь имеет много постов;
4. Каждый пост имеет много комментариев;
5. Каждый пост имеет много тегов;
6. Каждый тег принадлежит многим постам;
7. Каждый пользователь имеет много подписок на теги;
8. Каждая подписка принадлежит одному пользователю;
9. Каждый пользователь имеет много заблокированных тегов;
10. Каждый тег имеет много пользователей, заблокировавших его;

3.5 Описание алгоритмических зависимостей

1. Рейтинг поста = количество лайков – количество дизлайков;
2. Рейтинг комментария = количество лайков – количество дизлайков;
3. Рейтинг пользователя = сумма рейтингов всех его постов + сумма рейтинга всех его комментариев;

3.6 Требования к программной системе

Со стороны пользователя требований нет, так как это веб-приложение.

Со стороны сервера: Docker.

3.7 Построение ER-диаграммы

Сущности:

1. Сущность User. Атрибуты: id - primary key, is admin, is banned, nickname, email.
2. Сущность Post. Атрибуты: id – primary key, content – json состоящий из последовательности строк и картинок в произвольном порядке, rating.
3. Сущность Tag. Атрибуты: id – primary key, name.
4. Сущность Comment. Атрибуты: id – primary key, text, rating.

Отношения:

1. Subscriptions: User to Tag, Many to Many
2. Bans: User to Tag, Many to Many
3. Comments: User to Comment, One to Many
4. Posts: User to Post, One to Many
5. Tags: Tag to Post, Many to Many
6. Comments: Post to Comment, One to Many

|  |
| --- |
| Рисунок 3.3 – ER-диаграмма |

3.8 Построение схемы реляционной базы данных в третей нормальной форме

Нормализация – разделение таблицы на две или больше с целью исключения избытка информации и избегания аномалий баз данных.

1НФ представляет собой отношение. Для него характерны атомарность значений на пересечении столбцов и строк, кортежи и атрибуты должны быть неупорядочены и кортежи не дублируются.

Для 2НФ характерны особенности 1НФ и полная функциональная зависимость неключевых атрибутов от первичного ключа отношения.

3НФ – состояние отношения, которое находится во 2НФ и в котором между атрибутами нет транзитивных зависимостей.

Переход к следующей нормальной форме возможен только тогда, когда удовлетворяются все условия предыдущей формы.

Для предметной области построим универсальное отношение, которое включает описание ранее выделенных объектов и их свойств (см. рис. 3.4).

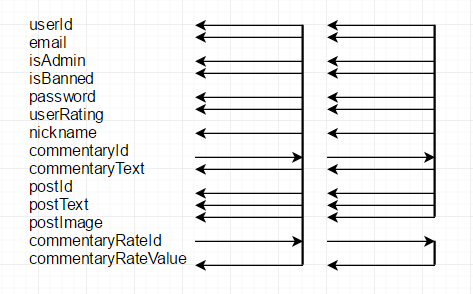


Рисунок 3.4 – Универсальное отношение

Данное универсальное отношение удовлетворяет требования 1НФ, то есть является отношением реляционных баз данных, а именно: кортежи не упорядочены, атрибуты не упорядочены, информация кортежей не дублируется, значения атрибутов атомарные. Это универсальное отношение не является 2НФ, так как атрибуты «email», «isAdmin», «password», «postId» и т.д. не полностью зависят от первичного ключа. Продолжим процедуру нормализации.

2НФ будет состоять из двух таблиц.

Выделим таблицу Т1, рисунок 3.5, и таблицу Т2, рисунок 3.6.

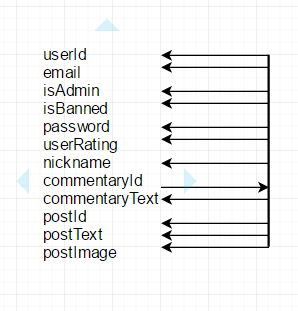


Рисунок 3.5 – Таблица Т1

Таблица Т1 находится в 2НФ, так как все атрибуты имеют полную функциональную зависимость от первичного ключа «commentaryId». Также присутствует транзитивная зависимость, что значит таблица не в 3 НФ.

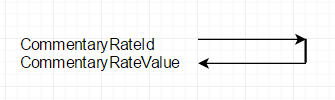


Рисунок 3.6 – Таблица Т2

Таблица Т2 находится в 2НФ и 3НФ, так как имеет всего 2 атрибута, один из которых первичный ключ. Таблица не имеет внешних ключей, что будет исправлено в процессе нормализации.

Для привидения Т1 к 3НФ разделим её на Т3, рисунок 3.7, Т4, рисунок 3.8 и Т5, рисунок 3.9.

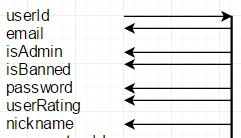


Рисунок 3.7 – Таблица Т3

Таблица Т3 находится в 3НФ, так как она находится в 2НФ и в ней отсутствуют транзитивные зависимости.



Рисунок 3.8 – Таблица Т4

Таблица Т4 находится в 3НФ, так как она находится в 2 НФ и в ней отсутствуют транзитивные зависимости.



Рисунок 3.9 – Таблица Т5

Таблица Т5 находится в 3НФ, так как она находится в 2 НФ и в ней отсутствуют транзитивные зависимости.

Данные таблицы далеки от представленных в ER-диаграмме, так как хоть они и в 3НФ, но после построения связей набор атрибутов изменится, но это не отменит того, что они будут в 3НФ. Это можно увидеть на конечной схеме базы данных (в 3НФ), рисунок 3.10.

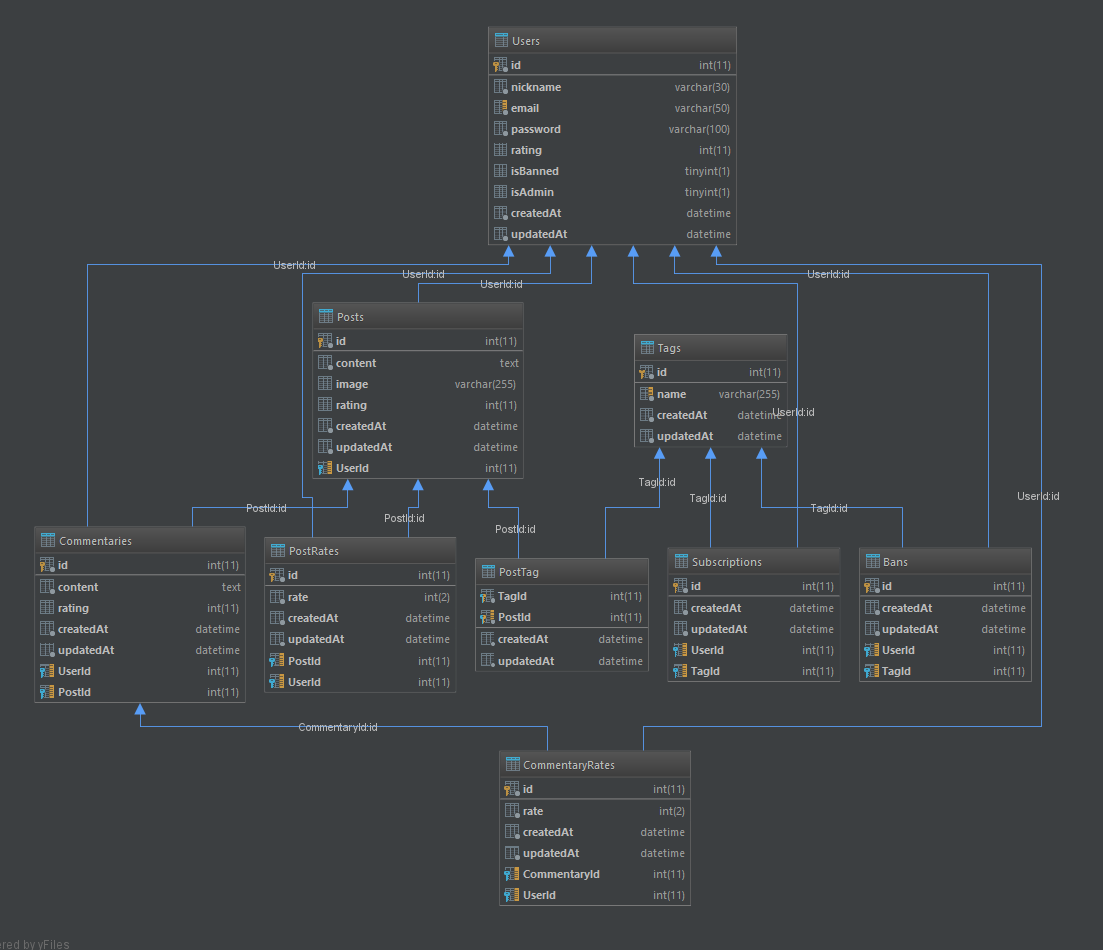


Рисунок 3.10 – Конечная схема БД

4 ОПИСАНИЕ ПРОГРАММЫ

4.1 Общие сведения

Разрабатываемый программный продукт создавался в IDE JetBrains WebStorm, с использованием JetBrains DataGrip для работы с БД и Docker для автоматизации развертывания и управления приложением. В качестве СУБД использовалась MySQL, но вместо нее могла использоваться любая другая реляционная СУБД, что не повлияло бы на процесс разработки и запуска.

4.2 Вызов и загрузка

Для запуска программы на сервере, нужно исполнить консольную команду docker-compose up в директории с программой, при этом кроме Docker на сервере не должно быть установлено абсолютно ничего.

4.3 Назначение и логическая структура

В постановке задачи, описанной выше, указаны все пункты, которые являются обязательными для выполнения. Данная программа соответствует этим требованиям целиком и полностью.

Выбор стиля интерфейса для пользователя обусловлен всеобщей тендецией к использованию Material Design в последнее время.

4.4 Описание физической модели БД

Конечная физическая модель базы данных полностью отображена на рисунке 3.10.

4.5 Описание программной реализации

На рисунке 4.1 приведена главная лента пользователя.

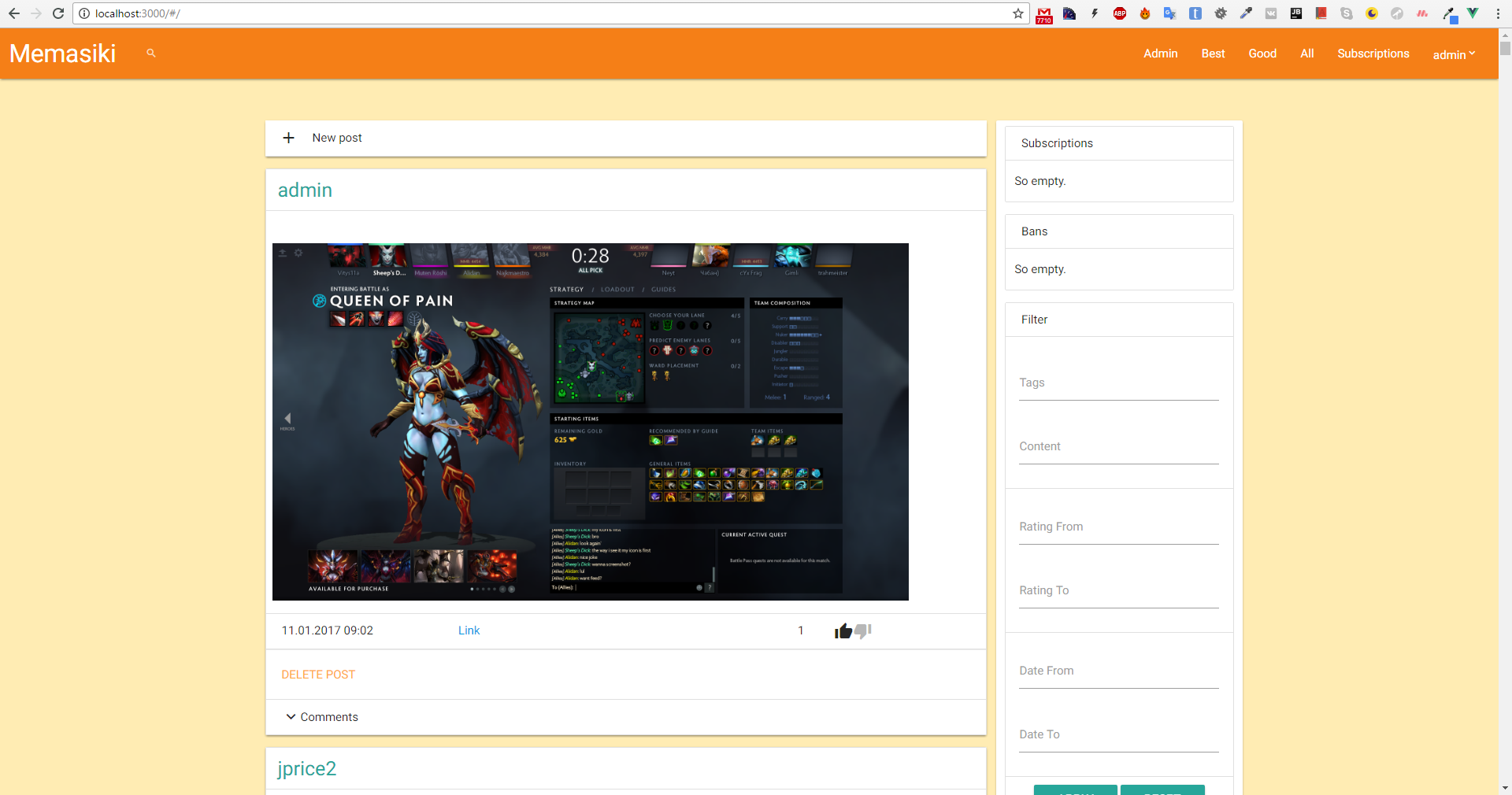


Рисунок 4.1 – Главная лента пользователя

Справа от ленты можно видеть фильтры, которые доступны только из главной ленты, на остальных они скрыты. Сверху главное меню сайта с поисковой строкой и навигацией по разным лентам, а также доступом к профилю и панелью со статистиками для администратора.

На рисунке 4.2 отображена страница профиля пользователя.

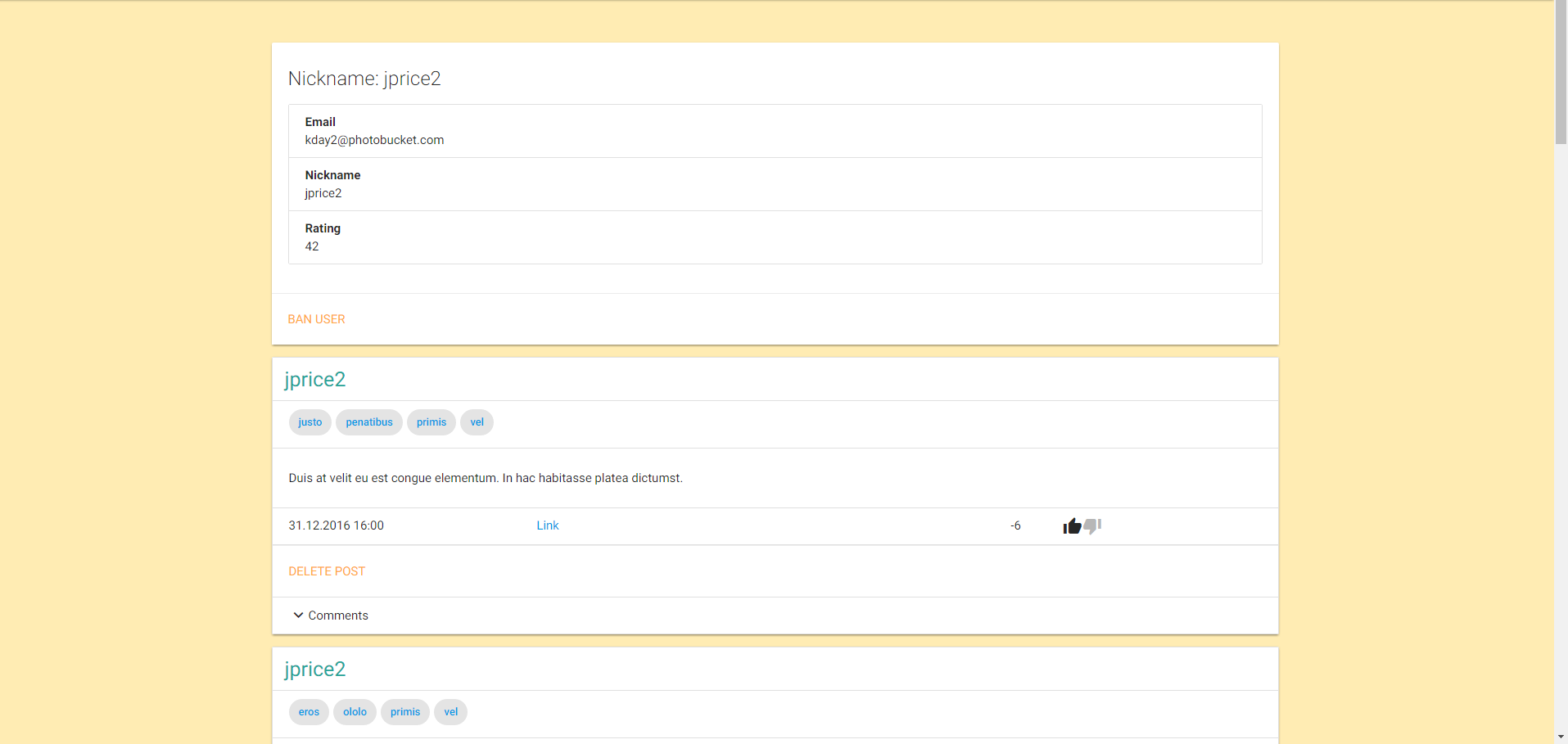


Рисунок 4.2 – Страница профиля пользователя

По клику на любой из тегов в шапке каждого из постов можно попасть на страницу тега, откуда доступны кнопки подписки или бана каждого тега. Страницу тега отображена на рисунке 4.3.

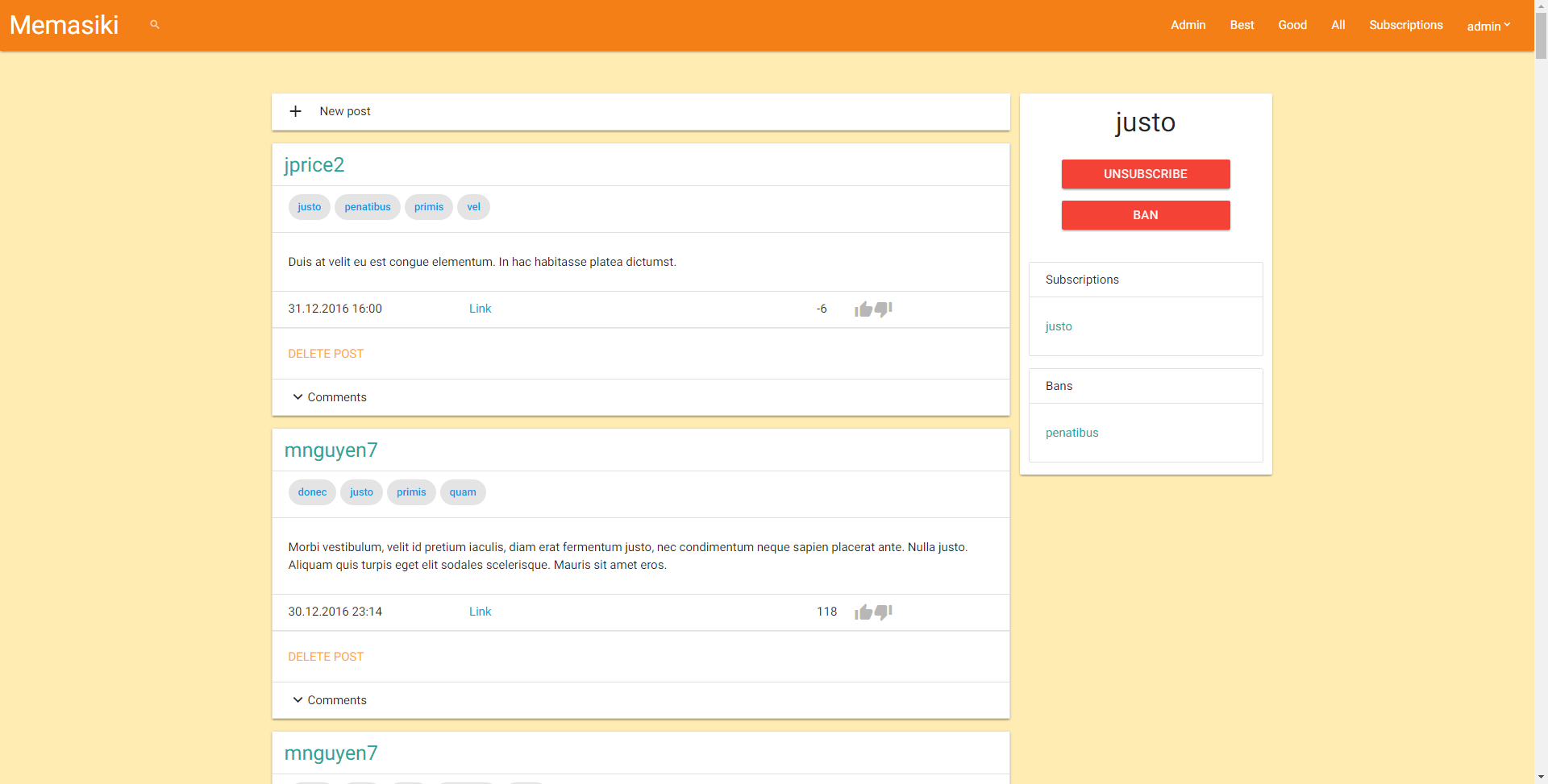


Рисунок 4.3 – Страница тега

Также на панели справа отображаются все активные подписки и баны пользователя.

При нажатии на кнопку «Admin», администратор может попасть на страницу статистик, которая частично отображена на рисунке 4.4.

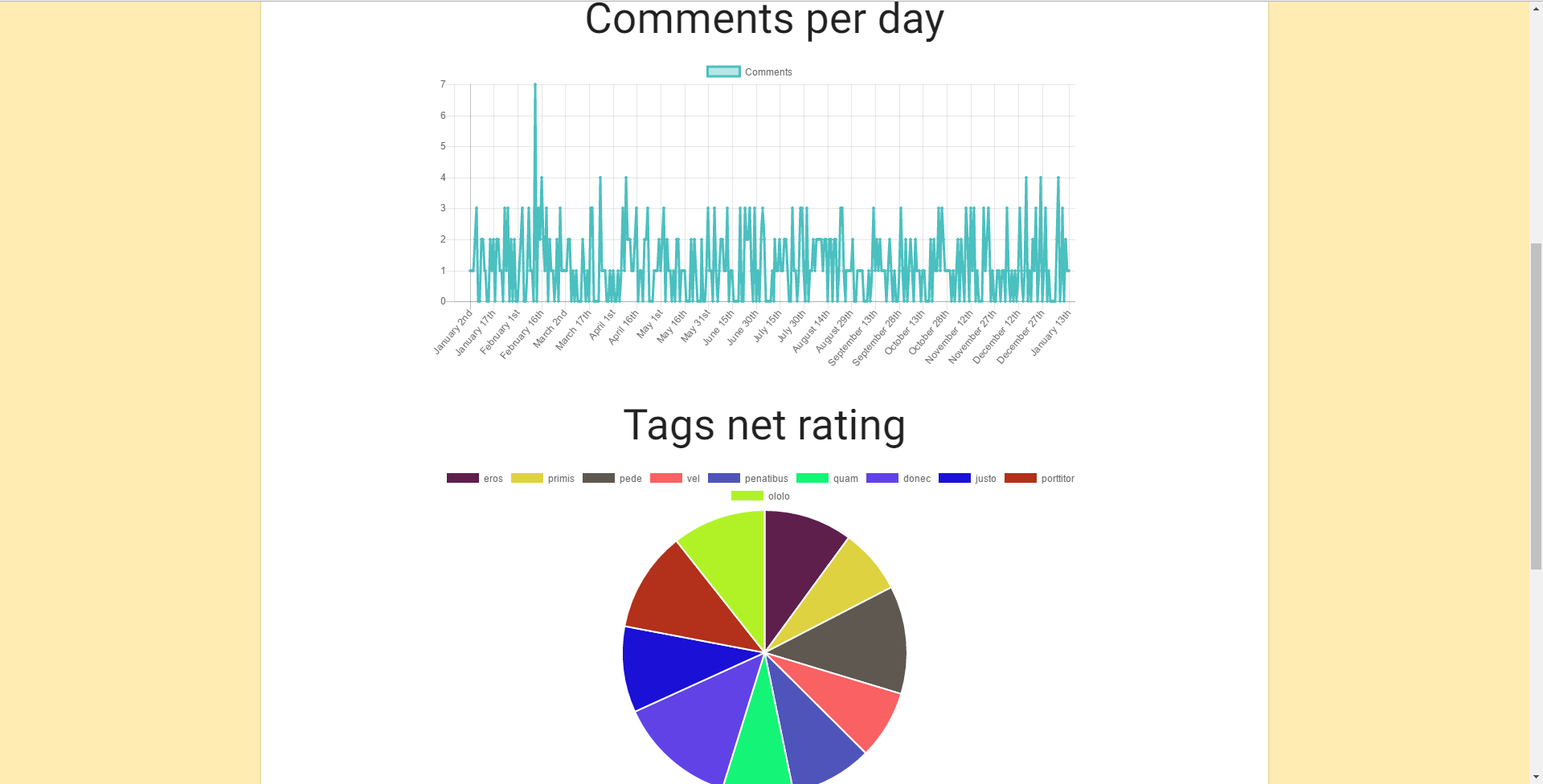


Рисунок 4.4 – Страница статистик

Далее приведены примеры запросов, использованных для получения информации. В большинстве своем запросы генерировались при помощи ORM Sequelize, но некоторые их фрагменты и даже небольшие запросы написаны вручную.

К примеру, так выглядит основной запрос получения постов с комментариями, тегами и лайками для конкретного пользователя.

async function getPosts(userId = null, where = null) {  
 return await Post.findAll({  
 where,  
 include: [  
 {  
 model: PostRate,  
 where: userId ? {  
 UserId: userId  
 } : null,  
 required: false  
 },  
 Tag,  
 User,  
 {  
 model: Commentary,  
 include: [User, {  
 model: CommentaryRate,  
 where: userId ? {  
 UserId: userId  
 } : null,  
 required: false  
 }]  
 }  
 ],  
 order: [  
 ['createdAt', 'DESC'],  
 [Tag, 'name'],  
 [Commentary, 'createdAt']  
 ]  
 })  
}

И вот один из вариантов преобразования этого запроса в SQL видно на рисунке 4.5.



Рисунок 4.4 – Сгенерированный запрос

Стоит отметить, что приведенный программный код ORM используется в программе несколько десятков раз для генераций различных лент для авторизованных и неавторизованных пользователей. Достигается это при помощи динамически задаваемого и передаваемого в качестве параметра функции условия where.

let whereClause: any = {}  
if (filter.tagsArray && filter.tagsArray.length)  
{  
 filter.tags = filter.tagsArray  
 .filter(tag => tag != '')  
 .map(tag => "'" + tag.trim() + "'")  
 whereClause.id = {  
 $in: db.literal(" (select `Posts`.`id` from `Posts` " +  
 "where (select count(`TagId`) from `PostTag` inner join `Tags` on `Tags`.`id` = `PostTag`.`TagId` " +  
 "where `PostTag`.`PostId` = `Posts`.`id` and `Tags`.`name` in (" + filter.tags.join(',') + ") ) = " + filter.tags.length + " ) ")  
 }  
}  
if (filter.content)  
 whereClause.content = {  
 $like: `%${filter.content}%`  
 }

К примеру, в этом коде, часть условия where была написана вручную, часть добавлена при помощи ORM.

А вот запросы, написанные целиков вручные, рядом со схожими запросами, сгенерированными при помощи ORM:

let postCountByDay = await Post.findAll({  
 attributes: [db.fn('date', db.col('createdAt')), db.fn('count', 'id')],  
 group: [db.fn('date', db.col('createdAt'))],  
 raw: true  
})  
let commentCountByDay = await Commentary.findAll({  
 attributes: [db.fn('date', db.col('createdAt')), db.fn('count', 'id')],  
 group: [db.fn('date', db.col('createdAt'))],  
 raw: true  
})  
  
let ratingCountByTag = await db.query("SELECT `Tag`.`name` as 'name', sum(`Posts`.`rating`) as 'rating'" +  
 " FROM `Tags` AS `Tag` LEFT OUTER JOIN (`PostTag` AS `Posts.PostTag` INNER JOIN `Posts` AS `Posts` ON `Posts`.`id` = `Posts.PostTag`.`PostId`)" +  
 " ON `Tag`.`id` = `Posts.PostTag`.`TagId` GROUP BY `Tag`.`id`;")  
  
let ratingAvgByTag = await db.query("SELECT `Tag`.`name` as 'name', avg(`Posts`.`rating`) as 'rating'" +  
 " FROM `Tags` AS `Tag` LEFT OUTER JOIN (`PostTag` AS `Posts.PostTag` INNER JOIN `Posts` AS `Posts` ON `Posts`.`id` = `Posts.PostTag`.`PostId`)" +  
 " ON `Tag`.`id` = `Posts.PostTag`.`TagId` GROUP BY `Tag`.`id`;")

Эти запросы используются для получения статистик администратора.

ВЫВОДЫ

В данной курсовой работе была разработана информационная система «Портал смешных картинок».

Работа над информационной системой требовала детального разбора будущей базы данных, в ходе которого были изучены различные этапы построения базы данных и проекта в целом. Среди них – ER-диаграмма, нормализация, концептуальный анализ, и т.д.

Впрочем, очевидно также, что на разработку программного продукта эти этапы не повлияли.

Все это в совокупности позволило получить работоспособную информационную систему, что удовлетворяет все требованиям, которые были поставлены. Наиболее ценным в этом проекте являются приобретенные навыки работы с реляционными СУБД в Node.js и полное осознавание их недостатков. Кроме того, использование SQL-запросов несомненно способствует закалению воли и терпения студента, как программиста.