

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

**ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**  
**по дисциплине «Введение в нереляционные базы данных»**  
**Тема: Web-сайт, агрегирующий информацию о породах кошек.**

Студентка гр. 1304	_____	Хорошкова А.С.
Студентка гр. 1304	_____	Виноградова М.О.
Студент гр. 1303	_____	Макки К.Ю.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург  
2024

## ЗАДАНИЕ

Студентка Хорошкова А.С.

Студентка Виноградова М.О.

Студент Макки К.Ю.

Группа 1304

Тема: Web-сайт, агрегирующий информацию о породах кошек.

Исходные данные:

Web-сайт где есть возможность поиска с фильтрацией по многим параметрам.

Можно посмотреть породу, краткую информацию и описание. Ниже комментарии где есть возможность обсуждать.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложения»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 35 страниц.

Дата выдачи задания: 05.09.2024

Дата сдачи реферата: 12.12.2024

Дата защиты реферата: 13.12.2024

Студентка	_____	Хорошкова А.С.
Студентка	_____	Виноградова М.О.
Студент	_____	Макки К.Ю.
Преподаватель	_____	Заславский М.М.

## АННОТАЦИЯ

В рамках ИДЗ разработано веб-приложение, представляющее собой информацию о породах кошек. Приложение включает функционал для просмотра информации о породах кошек, их добавления, редактирования и удаления, а также возможность оставлять отзывы или комментариев. Реализована система фильтрации и поиска пород по различным критериям.

Для разработки использованы технологии React, Spring Framework, СУБД MongoDB.

Найти исходный код можно по ссылке: [nosql2h24-dogs](https://github.com/nosql2h24/dogs)

## SUMMARY

As part of the Individual Homework Assignment, a web application has been developed to provide information about cat breeds. The application includes functionality for viewing cat breed information, adding, editing, and deleting entries, as well as leaving reviews or comments. A system for filtering and searching breeds based on various criteria has been implemented.

The application was developed using React, Java, Spring Framework, and MongoDB.

The source code is available at the following link: [nosql2h24-dogs](https://github.com/nosql2h24/dogs)

## СОДЕРЖАНИЕ

1.	Введение	6
1.1.	Актуальность проблемы	6
1.2.	Постановка задачи	6
1.3.	Предлагаемое решение	6
1.4.	Качественные требования к решению	6
2.	Сценарии использования	7
2.1.	Макет UI	7
2.2.	Сценарий использования для импорта данных	12
2.3.	Сценарий использования для представления данных	14
2.4.	Сценарий использования для анализа данных	16
2.5.	Сценарий использования для экспорта данных	17
2.6.	Вывод	17
3.	Модель данных	19
3.1.	Нереляционная модель данных	19
3.2.	Аналог модели данных для SQL СУБД	24
3.3.	Сравнение моделей	27
4.	Разработанное приложение	29
5.	Выводы	31
6.	Приложения	33
7.	Литература	35

# **1. ВВЕДЕНИЕ**

## **1.1. Актуальность проблемы**

Рост популярности кошек как домашних питомцев вызывает необходимость в удобных инструментах для изучения, управления и поиска информации о породах кошек. Существующие решения часто не предлагают эффективных механизмов для поиска, фильтрации и управления информацией о породах, а также не позволяют пользователям оставлять отзывы или комментарии.

## **1.2. Постановка задачи**

Цель проекта — разработать веб-приложение, которое позволяет пользователям:

- Просматривать подробную информацию о разных породах кошек, включая их особенности и характеристики.
- Добавлять, редактировать и удалять информацию о породах.
- Фильтровать и искать породы по различным критериям.
- Оставлять отзывы или комментарии о породах. Решение должно обеспечивать надежное хранение данных и высокую производительность приложения.

## **1.3. Предлагаемое решение**

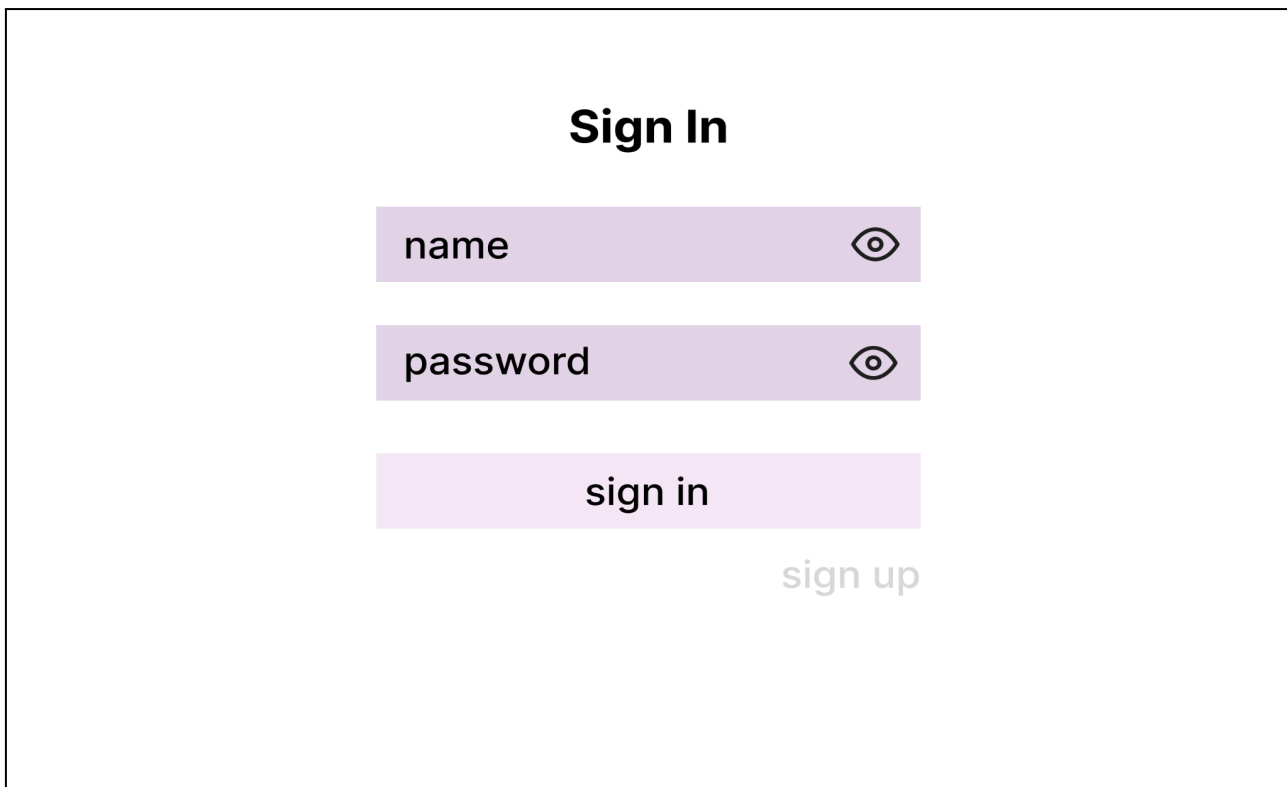
Веб-приложение разрабатывается с использованием React для фронтенда, Java и Spring Framework для серверной части, а также MongoDB для хранения данных. Приложение включает функционал поиска, фильтрации и публикации отзывов о породах кошек.

## **1.4. Качественные требования к решению**

Решение должно быть удобным, быстрым, надежным и легко расширяемым.

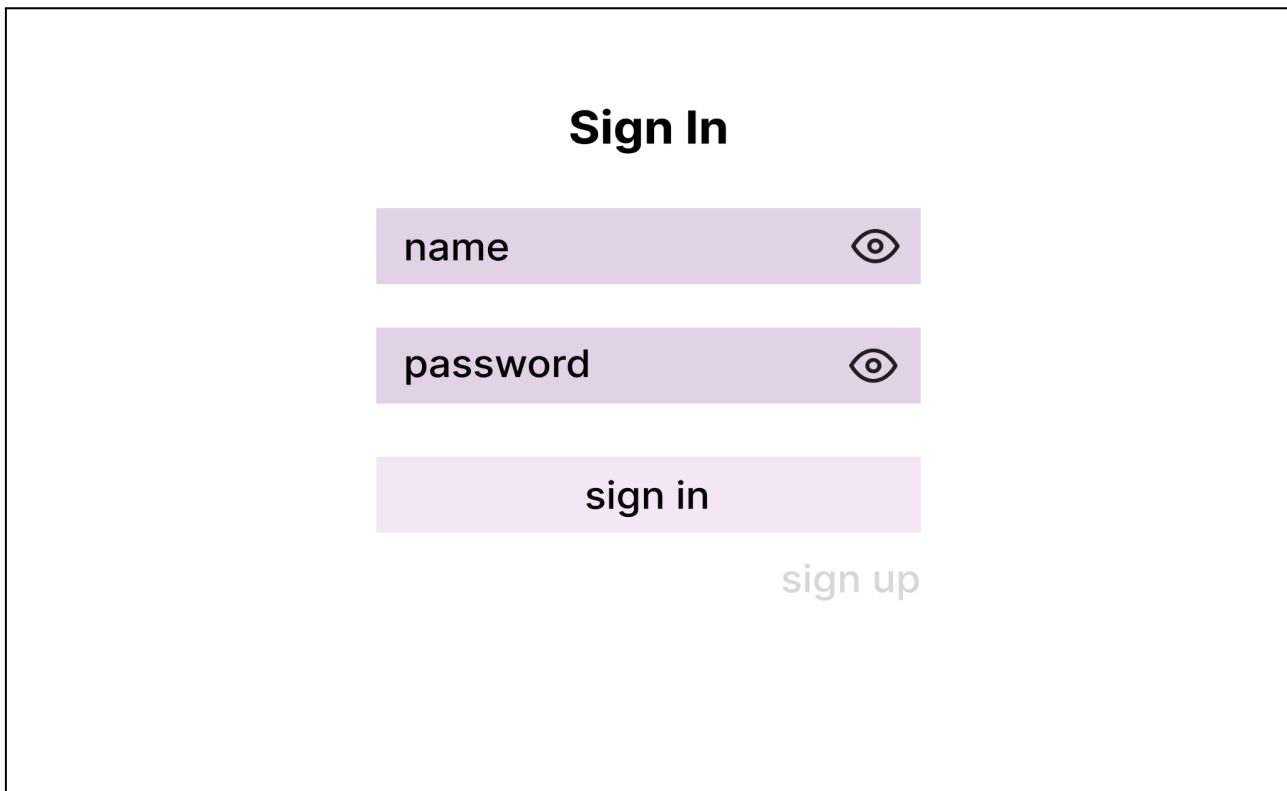
## 2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

### 2.1. Макет UI



The image shows a UI mockup for a 'Sign In' page. At the top center is the title 'Sign In' in a bold, black font. Below the title are two input fields, each with a light purple background and rounded corners. The first field is labeled 'name' and the second is labeled 'password'. To the right of each label is an eye icon, indicating a toggle for password visibility. Below the input fields is a primary button labeled 'sign in' with a light purple background and rounded corners. To the right of this button is a secondary link labeled 'sign up' in a lighter, grayish-purple font.

Рисунок 1. Страница Входа



The image shows a UI mockup for a registration page. It features the same layout as the 'Sign In' page, with a 'Sign In' title, 'name' and 'password' input fields with eye icons, a 'sign in' button, and a 'sign up' link. The overall design is clean and modern, using a light purple color scheme for the interactive elements.

Рисунок 2. Страница регистрации

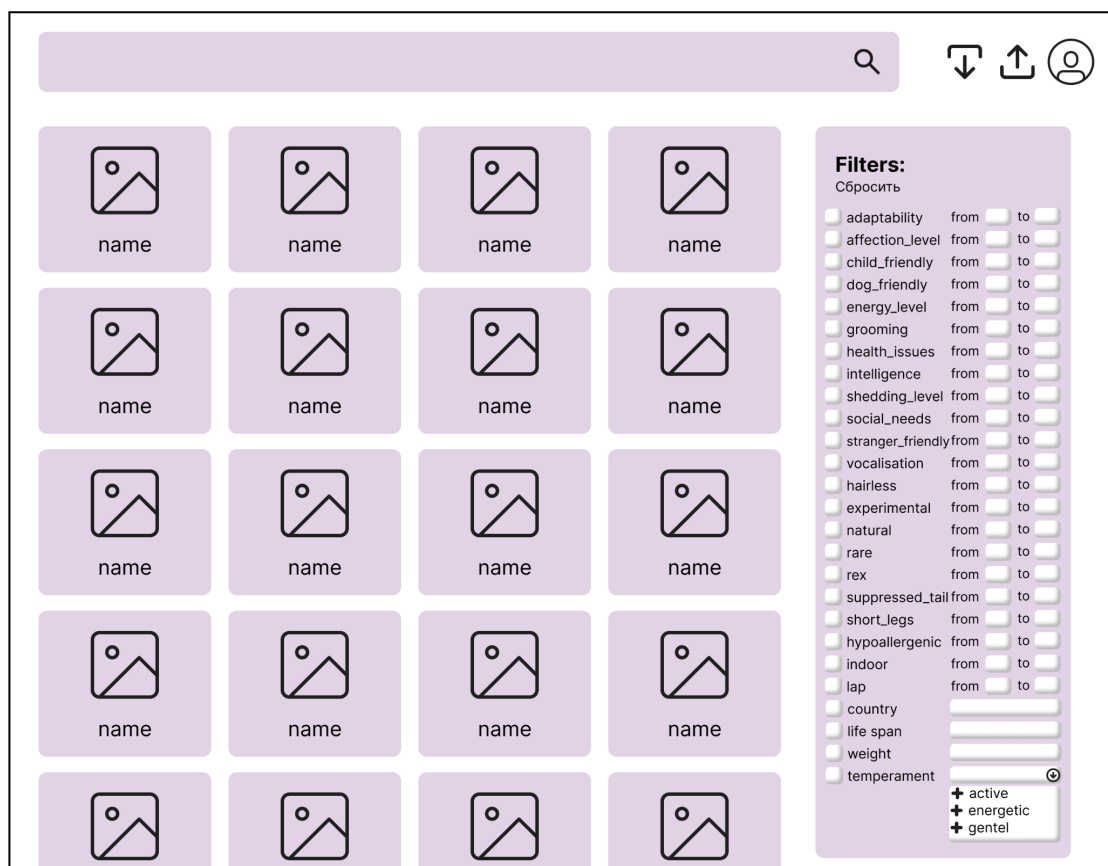


Рисунок 3. Главная страница администратора

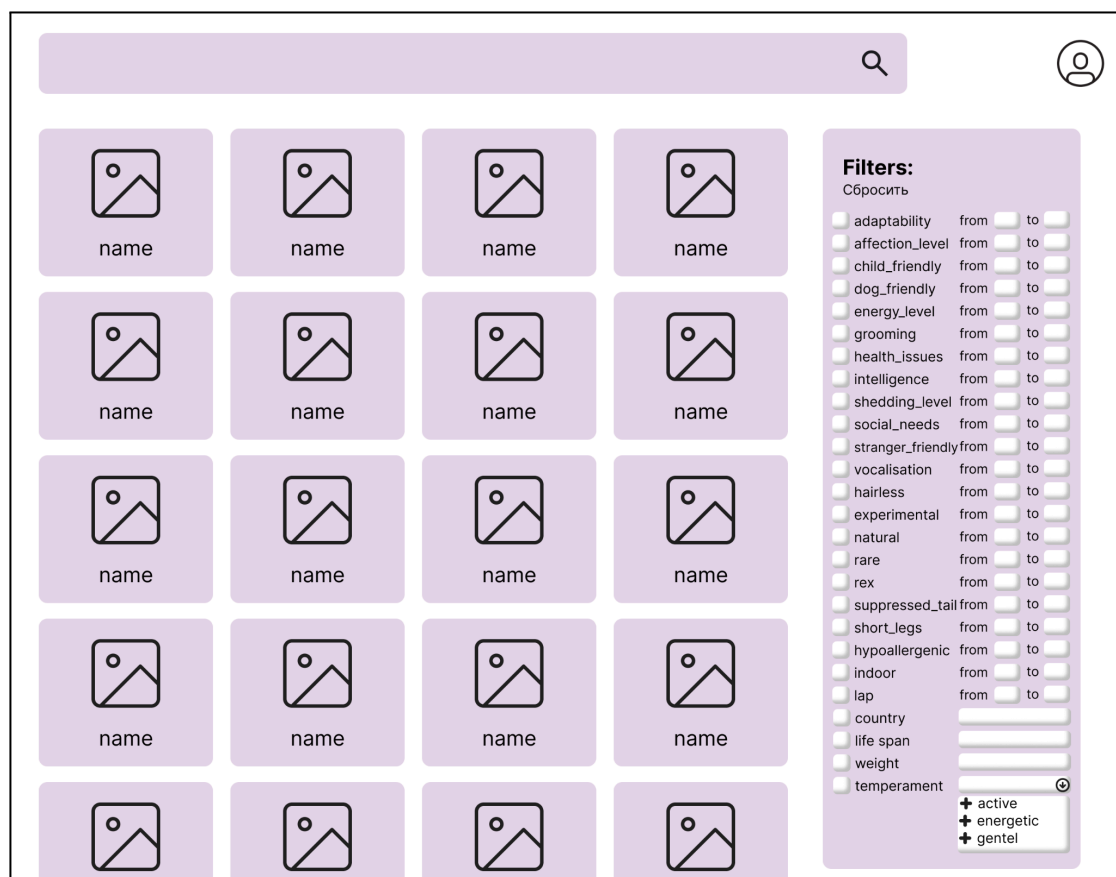


Рисунок 4. Главная страница пользователя

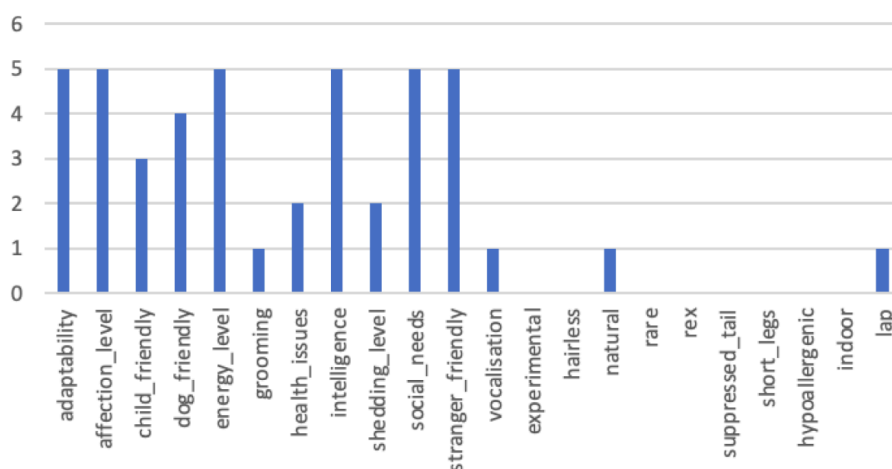




temperament: Active, Energetic, Independent, Intelligent, Gentle

life\_span: 14 - 15

origin: Egypt



## Links

<http://cfa.org/Breeds/BreedsAB/Abyssinian.aspx>

<http://www.vetstreet.com/cats/abyssinian>

<https://vcahospitals.com/know-your-pet/cat-breeds/abyssinian>

[https://en.wikipedia.org/wiki/Abyssinian\\_\(cat\)](https://en.wikipedia.org/wiki/Abyssinian_(cat))

name:

19.09.2001 17:00

♡ 10

name:

19.09.2001 17:00

♡ 0

name:

19.09.2001 17:00

♡ 0




Рисунок 5. Страница с информацией о кошках

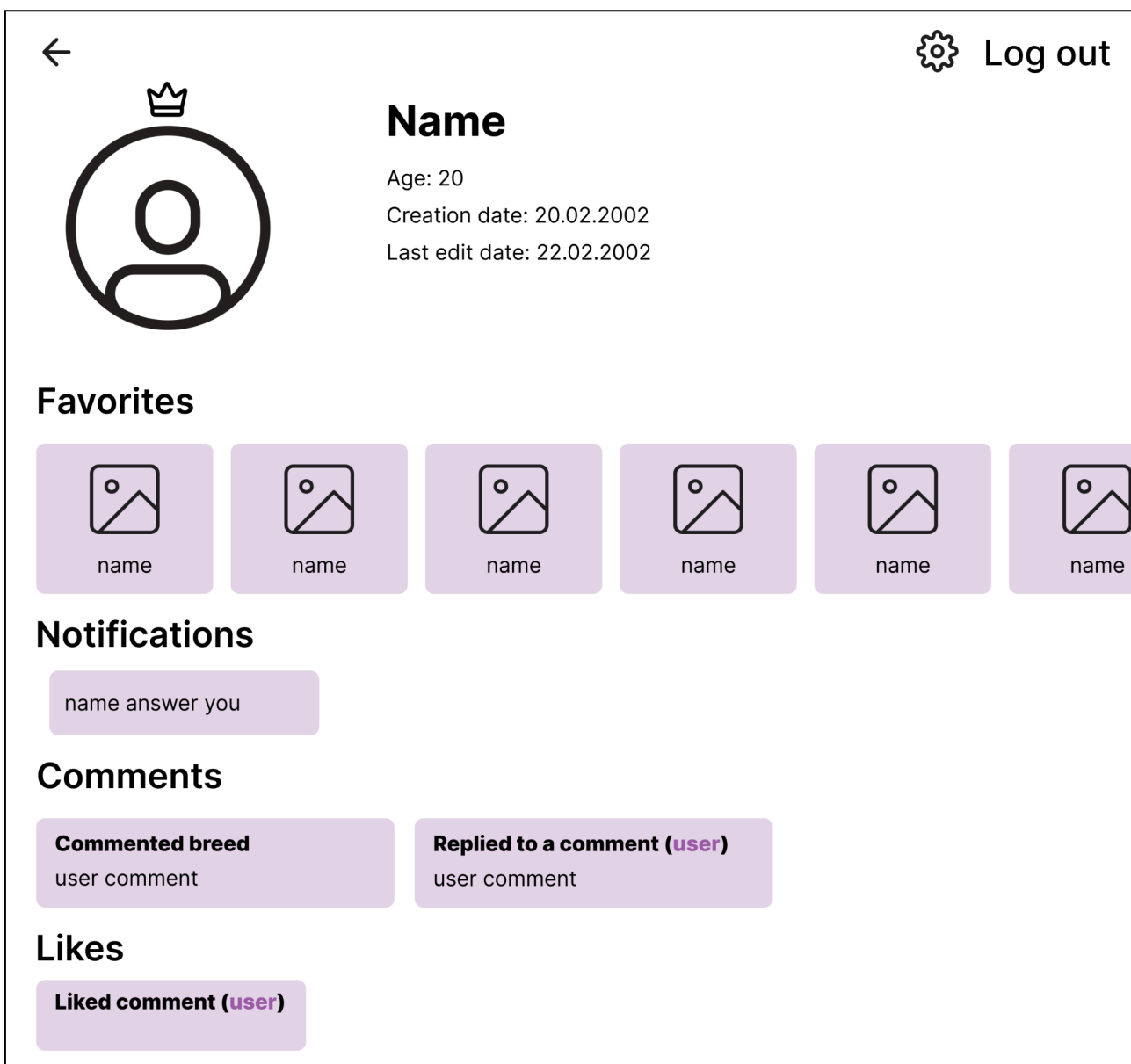


Рисунок 6. Страница профиля администратора

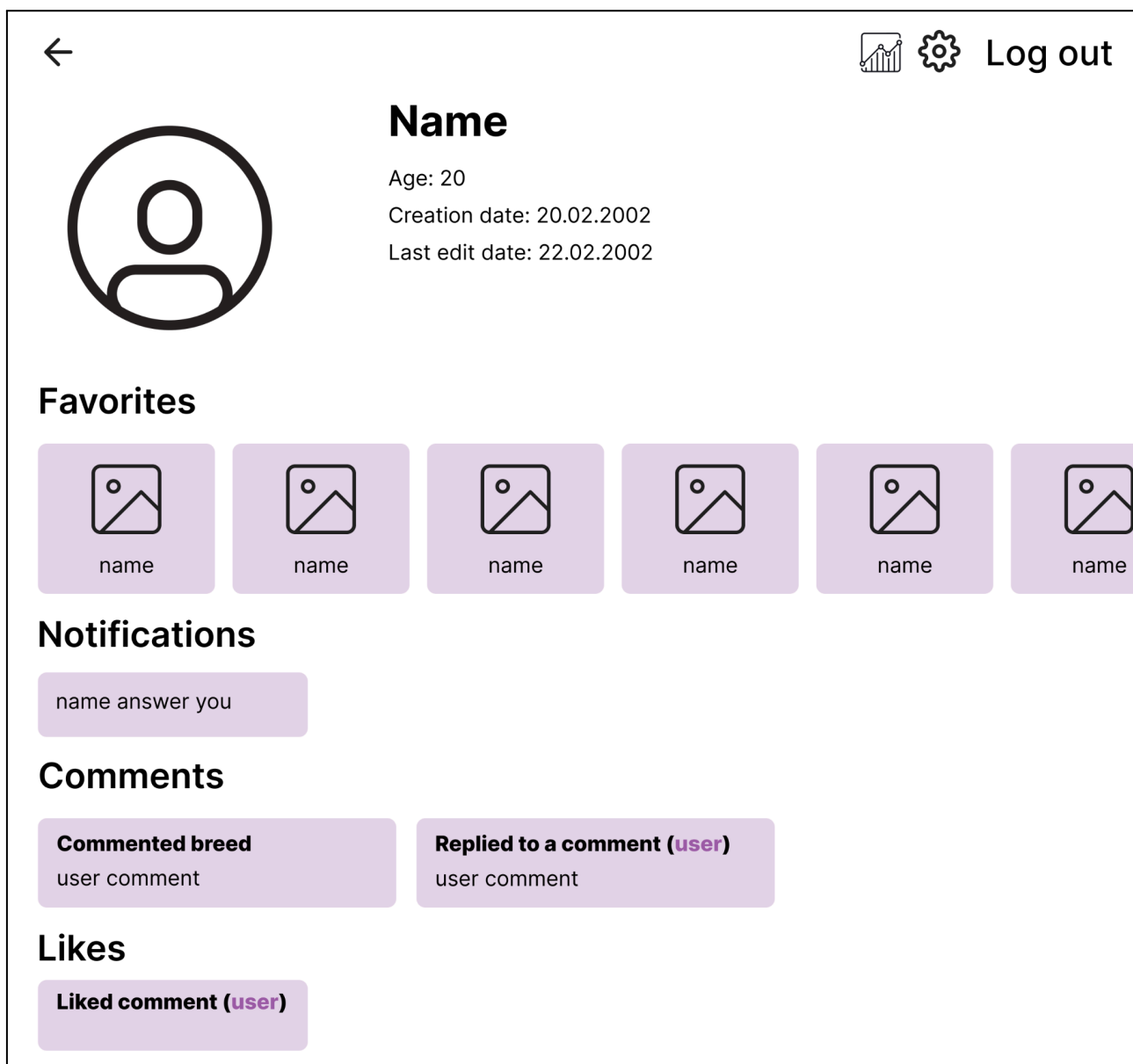




Рисунок 7. Страница профиля пользователя




Name

Age: 20


### Favorites




name




name




name



name



name



name

SAVE

Рисунок 8. Страница редактирование пользователя

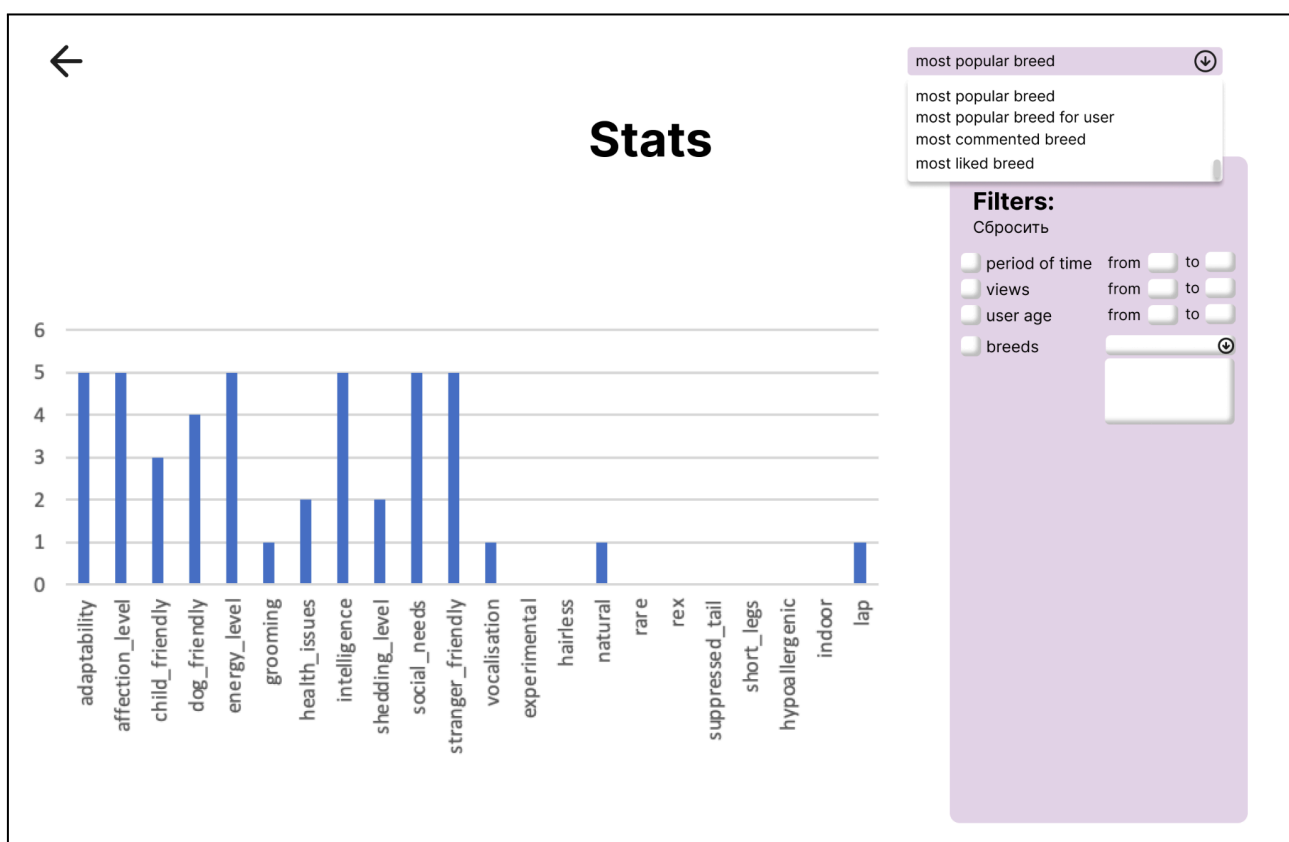


Рисунок 9. Страница статистика пользователя

## **2.2. Сценарий использования для импорта данных**

### **1. Импорт данных (Data Import)**

**Действующее лицо:** Администратор

**Основной сценарий:**

- 1) Администратор находится на Admin Main Screen.
- 2) Нажимает кнопку "Import Data".
- 3) Приложение предлагает выбрать файл на устройстве.
- 4) Администратор выбирает файл и подтверждает импорт.
- 5) Приложение проверяет данные на корректность и обновляет базу данных.

**Альтернативный сценарий:**

- 1) Если файл некорректен, приложение выводит сообщение об ошибке и предлагает выбрать правильный файл.

## **2.3. Сценарий использования для представления данных**

### **1. Поиск и фильтрация (Search and Filtering)**

**Действующее лицо:** Пользователь

**Основной сценарий:**

- 1) Пользователь находится на главном экране (Main Screen или Admin Main Screen).
- 2) Пользователь вводит запрос в строку поиска и/или применяет фильтры.
- 3) Пользователь нажимает на кнопку поиск или её аналог клавиатуре (enter).
- 4) Приложение отображает карточки животных, которые соответствуют запросу и фильтрам.

### **2. Просмотр карточки животного (View Breed Card)**

**Действующее лицо:** Пользователь

**Основной сценарий:**

- 1) Пользователь находится на главном экране (Main Screen или Admin Main Screen).
- 2) Нажимает на карточку конкретной породы.

- 3) Приложение отображает переходит на экран Breed Card с подробной информацией о породе, полезными ссылками, комментариями других пользователей.

### 3. Управление избранными (Favorites Management)

**Действующее лицо:** Пользователь

#### **Основной сценарий-1:**

- 1) Пользователь находится на Breed Card.
- 2) Нажимает кнопку "Add to Favorites", и порода добавляется в избранное.

#### **Основной сценарий-2:**

- 1) Пользователь находится на Profile Page или Admin Profile Page.
- 2) Пользователь нажимает на карточку одного из любимых животных.
- 3) Пользователь попадает на Breed Card

#### **Альтернативный сценарий:**

- 1) На Breed Card пользователь может повторно нажать на кнопку "Add to Favorites", и животное удаляется из любимых.
- 2) В режиме редактирования профиля пользователь может удалить породы из избранного на Edit Profile.

## **2.4. Сценарий использования для анализа данных**

### 1. Просмотр статистики (View Profile Stats)

**Действующее лицо:** Пользователь

#### **Основной сценарий:**

- 1) Пользователь находится на странице профиля (Profile Page или Admin Profile Page).
- 2) Нажимает на иконку статистики.
- 3) Приложение отображает страницу Profile Stats с графиками и статистикой из выпадающего списка (например, самые популярные породы по просмотрам для пользователя, самые залайканные породы) с примененными фильтрами и для выбранного подмножества данных.

## 2. Получение уведомления и переходит на страницу породы (Receive Notification and View Breed Card)

**Действующее лицо:** Пользователь

**Основной сценарий:**

- 1) Пользователь получает уведомление о событии.
- 2) Кликает на уведомление.
- 3) Приложение переходит на Breed Card, в котором произошло событие.

## 3. Просмотр своей истории активности на странице (User Activity)

**Действующее лицо:** Пользователь

**Основной сценарий-1:**

- 1) Пользователь находится на странице профиля (Profile Page или Admin Profile Page).
- 2) Нажимает на элемент секции последних комментариев пользователя.
- 3) Приложение переходит на Breed Card, в котором пользователь оставил комментарий.

**Основной сценарий-2:**

- 1) Пользователь находится на странице профиля (Profile Page или Admin Profile Page).
- 2) Нажимает на элемент секции последних лайков пользователя.
- 3) Приложение переходит на Breed Card, в котором пользователь поставил лайк.

## **2.5. Сценарий использования для экспорта данных**

### 1. Экспорт данных (Data Export)

**Действующее лицо:** Администратор

**Основной сценарий:**

- 1) Администратор находится на Admin Main Screen.
- 2) Нажимает кнопку "Export Data".

- 3) Приложение генерирует файл и предлагает сохранить его на устройство администратора.

#### **Альтернативный сценарий:**

- 1) Если возникла ошибка при генерации файла, приложение выводит сообщение об ошибке и предлагает повторить экспорт.

### **2.6. Операции, преобладающие в решении (чтение или запись)**

Преобладающими операциями будут как чтение, так и запись.

Чтение: Основной функционал включает в себя просмотр данных: поиск пород, фильтрация, просмотр карточек животных и статистики. Эти действия требуют чтения данных из базы данных и отображения их на экране.

Запись: Запись данных в систему происходит при регистрации пользователей, добавлении комментариев, лайков, добавлении пород в избранное, а также при экспорте и импорте данных для администратора

Чтение данных будет преобладать в рамках большинства взаимодействий пользователей, таких как поиск, фильтрация и просмотр информации о породах. Однако, запись данных будет важной операцией в контексте регистрации пользователей, редактирования профилей и взаимодействия с контентом (комментарии, лайки, избранное)



### 3. МОДЕЛЬ ДАННЫХ

#### 3.1. Нереляционная модель данных

##### 1. Графическое представление модели Модель состоит из трех коллекций BREED:

```
[
  {
    "_id": "BREED_ID_1",
    "name": "SOME_BREED_NAME",
    "image": "SOME_URL_TO_IMAGE",
    "parameters": {
      "PARAMETER_NAME_1": "PARAMERET_VALUE_1",
      "PARAMETER_NAME_2": "PARAMERET_VALUE_2",
      ...
      "PARAMETER_NAME_N": "PARAMERET_VALUE_N"
    },
    "comments": [
      {
        "id": "COMMENT_ID_1",
        "author": "SOME_USER_ID",
        "parent_comments_id": "SOME_PARENT_COMMENT_ID",
        "date": ISODate("SOME_DATE"),
        "text": "SOME_TEXT",
        "likes": [
          "SOME_USER_ID_1",
          "SOME_USER_ID_2",
          ...
          "SOME_USER_ID_N"
        ]
      },
      ....
      {
        "id": "COMMENT_ID_N",
        ....
      }
    ]
  },
  ...
  {
    "_id": "BREED_ID_N",
    ...
  }
]
```

Так как с ростом количества комментариев коллекция может превысить максимальный размер коллекции mongodb (16мб), коллекция BREED может быть использована совместно со спецификацией GridFS.

## USER:

```
[ {
  "_id": "USER_ID_1",
  "name": "SOME_NAME",
  "age": SOME_AGE,
  "isAdmin": TRUE_OR_FALSE,
  "passwordHash": "SOME_HASH",
  "image": "SOME_URL_TO_IMAGE",
  "creation_date": ISODate("SOME_DATE"),
  "last_date": ISODate("SOME_ANOTHER_DATE"),
  "favorites":
    [ "SOME_BREED_ID_1",
      "SOME_BREED_ID_2", ...,
      "SOME_BREED_ID_N", ]
},
...
{
  "_id": "USER_ID_N",
  "name": "SOME_NAME",
  "age": SOME_AGE,
  "isAdmin": TRUE_OR_FALSE,
  "passwordHash": "SOME_HASH",
  "image": "SOME_URL_TO_IMAGE",
  "creation_date": ISODate("SOME_DATE"),
  "last_date": ISODate("SOME_ANOTHER_DATE"),
  "favorites":
    [ "SOME_BREED_ID_1",
      "SOME_BREED_ID_2", ...,
      "SOME_BREED_ID_N", ]
}
]
```

## EVENT:

```
[ {
  "_id": "EVENT_ID_1",
  "user_id": "SOME_USER_ID",
  "breed_id": "SOME_BREED_ID",
  "type": "SOME_TYPE",
  "date": ISODate("SOME_DATE")
},
...
{
  "_id": "EVENT_ID_N",
  "user_id": "SOME_USER_ID",
  "breed_id": "SOME_BREED_ID",
  "type": "SOME_TYPE",
  "date": ISODate("SOME_DATE")
}
]
```

## 2. Описание назначений коллекций, типов данных и сущностей

**Breed:** хранит информацию о различных породах собак, включая их характеристики, изображения, комментарии от пользователей и лайки на комментарии.

### Содержимое:

\_id: string - Уникальный идентификатор породы.

name: string - Название породы.

image: string - URL изображения породы.

parameters\*: array - Массив с параметрами породы.

comments: array - Массив комментариев, где каждый комментарий является объектом, содержащим:

id: string - Уникальный идентификатор комментария.

author: string - Id автора комментария.

parent\_comments\_id: string|null - Идентификатор родительского комментария (null, если комментарий родительский в своей ветке).

date: ISODate - Дата и время написания комментария.

text: string - Текст комментария.

likes: array - Массив id пользователей, которые поставили лайк конкретному комментарию.

\*Элементы массива parameters:

weight\_imperial: string

weight\_metric: string

vetstreet\_url: string

temperament: string

origin: string

country\_codes: string

country\_code: string

description: string

life\_span: string

indoor: integer

alt\_names: string  
adaptability: integer  
affection\_level: integer  
child\_friendly: integer  
dog\_friendly: integer  
energy\_level: integer  
grooming: integer  
health\_issues: integer  
intelligence: integer  
shedding\_level: integer  
social\_needs: integer  
stranger\_friendly: integer  
vocalisation: integer  
experimental: integer  
hairless: integer  
natural: integer  
rare: integer  
rex: integer  
suppressed\_tail: integer  
short\_legs: integer  
wikipedia\_url: string  
hypoallergenic: integer

**User:** хранит информацию о пользователях системы, включая имя, возраст, время создания и последнего редактирования профиля, наличие статуса администратора и избранные породы.

Содержимое:

\_id: string - Уникальный идентификатор пользователя.

name: string - Имя пользователя.

age: integer - Возраст пользователя.

isAdmin: boolean - Наличие у пользователя статуса администратора.  
passwordHash: string - Хеш пароля пользователя.  
image: string|null - URL изображение пользователя (null, если изображения нет).  
creation\_date: ISODate - Дата и время создания учетной записи пользователя.  
last\_date: ISODate - Дата и время последнего входа пользователя.  
favorites: array - Массив id пород, добавленных пользователем в избранное.

**Event:** хранит сведения о действиях пользователей, связанных с породами собак.

Содержимое:

\_id: string - Уникальный идентификатор действия.  
user\_id: string - Id пользователя, совершившего действие.  
breed\_id: string - Id породы, с которой связано событие.  
type: string - Тип произошедшего события.  
date: ISODate - Дата и время возникновения события.

### 3. Оценка удельного объема информации, хранимой в модели

Один лайк занимает 36 байт (один Id пользователя) Один комментарий занимает 540 байтов + 36 x кол-во лайков Одна запись в коллекции Breed в среднем занимает 806 байтов + (540 байтов + 36 x среднее кол-во лайков) x среднее кол-во комментариев

Одна запись о любимой породе занимает 36 байтов (один Id породы) Предположим, что у одного пользователя в среднем 5 любимых пород. Тогда: Одна запись в коллекции User в среднем занимает 337 байтов

Одна запись в коллекции Event занимает 136 байтов.

В качестве переменной возьмем количество пользователей. Количество пород возьмем равное 200. Пусть пользователь в среднем оставляет 20 комментариев, ставит 100 лайков, имеет 100 любимых пород и генерирует 1000 событий. Тогда объём модели занимает:  $200 \times 806 + n \times 540 + n \times 36 + n \times 337 + n \times 100 \times 36 + 1000 \times 136 \times n = 161200 + 140513 \times n$  байтов

### Избыточность данных:

В модели используются избыточные данные об ссылающихся друг на друга id (события, комментарии и лайки ссылаются на пользователей, события ссылаются на породы и тп).

### Избыточность:

$$(161200 + 140513 \times n) / (36 \times 200 + 20 \times 36 \times n + 36 \times n) = (297200 + 4513 \times n) / (x \times n + 756 + 7200)$$

Направление роста модели при увеличении количества объектов каждой сущности. Рост линейно зависит от количества пользователей.

## **4. Примеры запросов**

### **Добавление пользователя:**

```
db.user.insertOne({ _id: "user_id", user_name: "SASHA", "isAdmin": false,
"passwordHash": "hashed_password_1", "image":
"https://example.com/images/user1.jpg", "creation_date": "2023-01-01T08:00:00Z",
"last_date": "2023-10-01T09:00:00Z", "favorites": [ "breed_id_1" ] });
```

### **Получение списка всех пород/фильтрация:**

```
db.breed.find({});
db.breed.find({ "parameters.activity": 5 });
```

### **Получение карточки породы (один запрос):**

```
db.breed.findOne({ _id: "breed_id" });
```

### **Добавления комментария:**

```
db.breed.updateOne( { _id: "breed_id" }, { $push: { comments: { id: "comment_id",
author: "author_id", parent_comments_id: "parent_comment_id", date: now(), text:
"Смотрю на эту кошечку уже сутки!", likes: [] } } }
db.event.insertOne({user_id: "author_id", breed_id: "breed_id", type:
"comment_added", date: now()});
```

### **Получение комментариев:**

```
db.breed.find( { _id: "breed_id" }, { comments: 1 })
```

**Лайк комментарию:**

```
db.breed.updateOne( { _id: "BREED_ID", "comments.id": "comment_id" }, {  
  $addToSet: { "comments.$.likes": "user_id" } } );  
db.event.insertOne({ user_id: "user_id", breed_id: "breed_id", type:  
  "comment_liked", date: now() });
```

**Удаление лайка:**

```
db.breed.updateOne( { _id: "BREED_ID", "comments.id": "comment_id" }, { $pull:  
  { "comments.$.likes": "user_id" } } )  
db.event.insertOne({ user_id: "user_id", breed_id: "breed_id", type:  
  "comment_unliked", date: now() });
```

**Профиль пользователя:**

```
db.user.findOne({ _id: "user_id" });
```

**Изменение профиля пользователя:**

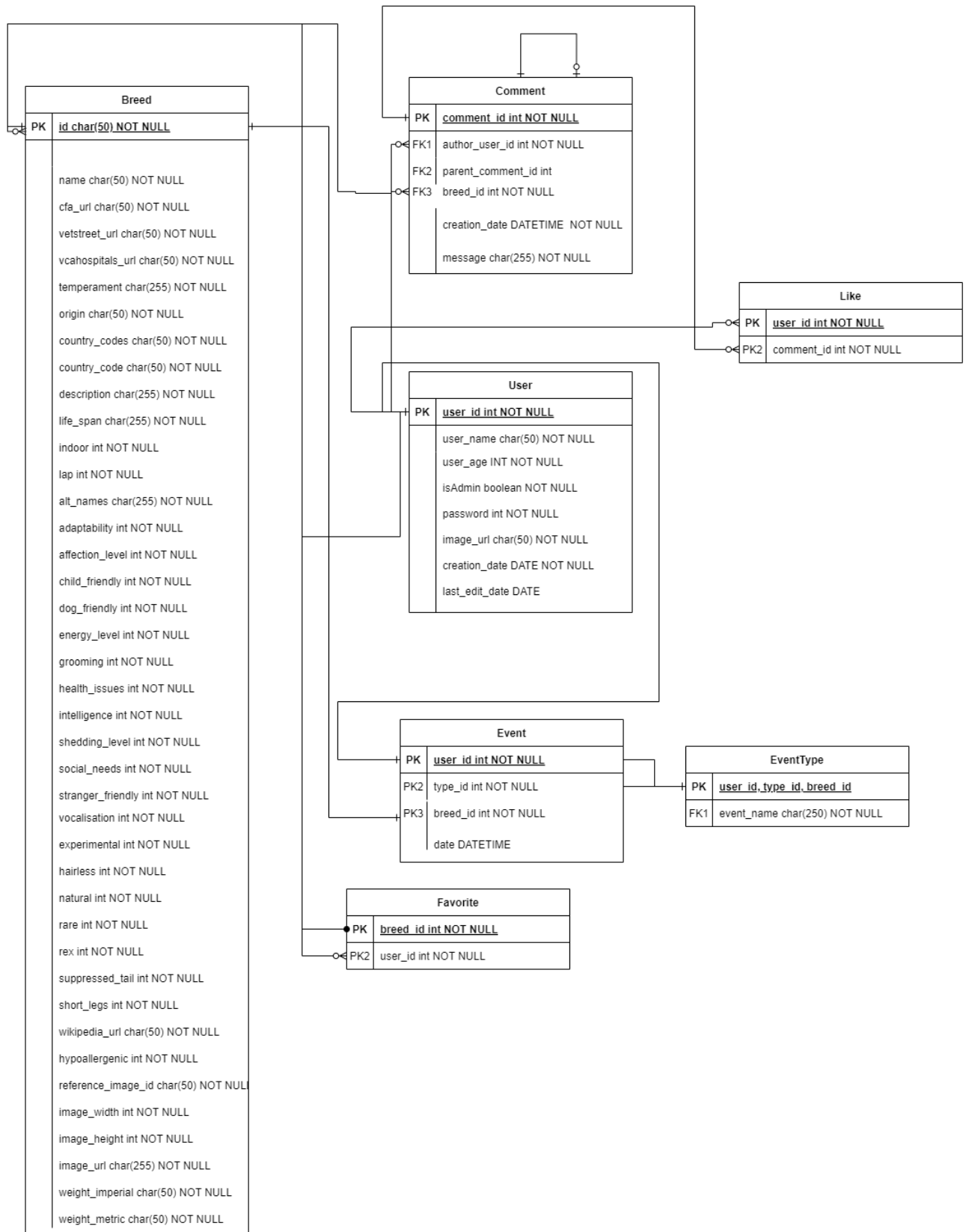
```
db.user.updateOne( { _id: "user_id" }, { $set: { name: "new_name", age: new_age,  
  image: "url_to_new_image", last_date: now() }
```

**Статистика:**

```
db.event.find({ user_id: "user_id", type: "comment_unliked" });
```

## 3.2. Аналог модели данных для SQL СУБД

### 1. Графическое представление модели





## 2. Описание назначений коллекций, типов данных и сущностей

Таблица **Breed** предназначена для хранения данных о конкретной породе. В таблице **Breed** хранятся идентификатор породы (внешний ключ) и различные характеристики породы, такие как вес (char(50)), описание темперамента(char(255)), общее описание породы (char(255)), страна происхождения (char(50)), энергичность(int) и тд.

Таблица **Comment** предназначена для хранения данных о комментариях. В таблице **Comment** хранятся идентификатор комментария (int), идентификатор автора комментария(int), идентификатор родительского комментария (int), идентификатор породы которой был написан комментарий (int), полная дата создания комментария(datetime), текст комментария(char(255)).

Таблица **User** предназначена для хранения данных о пользователях. В таблице **User** хранятся идентификатор пользователя (int), имя пользователя (char(50)), возраст пользователя (int), флаг админа (boolean), пароль (int), ссылка на картинку(char(50)), дата создания профиля(date), дата последнего редактирования профиля(date).

Таблица **Event** предназначена для хранения данных о событиях(для статистики). В таблице **Event** хранятся идентификатор пользователя(int), тип события(int), идентификатор породы(int), дата события(datetime).

Таблица **EventType** предназначена для хранения данных о типе события. В таблице **\*EventType** хранятся идентификатор пользователя(int), тип события(int), идентификатор породы(int), имя события(char(250)).

Таблица **Favorite** предназначена для хранения данных о любимых породах. В таблице **Favorite** хранятся идентификаторы пользователя(int) и породы(int).

Таблица **Like** предназначена для хранения данных о понравившихся комментариях. В таблице **Like** хранятся идентификаторы пользователя(int) и комментария(int).

### 3. Оценка объема информации, хранимой в модели

Таблица **Breed** статична, количество памяти для одной породы равно 1971 байт

Количество памяти для одного комментария равно 275 байт

Количество памяти для одного пользователя равно 122 байт

Количество памяти для одного события равно 20 байт

Количество памяти для одного типа события равно 254 байт

Количество памяти для одного лайка равно 8 байт

Количество памяти для одного фаворита равно 8 байт

Среднее число пород - 200

Среднее число комментариев пользователя - 20

Среднее число любимых пород пользователя - 100

Среднее число понравившихся комментариев - 100

Среднее число событий пользователя - 1000

Формула:  $1971 \times 200 + (122 + 20 \times 275 + 1000 \times (20 + 254) + 100 \times 8 + 100 \times 8) \times n$   
 $n = 394200 + 33822 \times n$

#### Избыточность данных:

Избыточность модели Избыточность в таблице **Breed**

breed\_id - 4 байта

Избыточность в таблице **Comment**

comment\_id, author\_user\_id, parent\_comment\_id, breed\_id - 16 байт

Избыточность в таблице **User**

user\_id, isAdmin - 5 байт

Избыточность в таблицы **Like, Favorite, Event, EventType**

избыточны - 298 байт

Формула:  $1967 \times 200 + (117 + 20 \times 259) \times n = 393400 + 5297 \times n$

#### 4. Примеры данных

**Breed {**

```
"id": "aege",  
"weight_imperial": "7 - 10",  
"weight_metric": "3 - 5",  
"name": "Aegean",  
"vetstreet_url": "http://www.vetstreet.com/cats/aegean-cat",  
"temperament": "Affectionate, Social, Intelligent, Playful, Active",  
"origin": "Greece",  
"country_codes": "GR",  
"country_code": "GR",  
"description": "Native to the Greek islands known as the Cyclades in the Aegean Sea,  
these are natural cats, meaning they developed without humans getting involved in  
their breeding. As a breed, Aegean Cats are rare, although they are numerous on their  
home islands. They are generally friendly toward people and can be excellent cats for  
families with children.",  
"life_span": "9 - 12",  
"indoor": 0,  
"alt_names": "",  
"adaptability": 5, "affection_level": 4,  
"child_friendly": 4,  
"dog_friendly": 4,  
"energy_level": 3,  
"grooming": 3,  
"health_issues": 1,  
"intelligence": 3,  
"shedding_level": 3,  
"social_needs": 4,  
"stranger_friendly": 4,  
"vocalisation": 3,
```

```
"experimental": 0,
"hairless": 0,
"natural": 0,
"rare": 0,
"rex": 0,
"suppressed_tail": 0,
"short_legs": 0,
"wikipedia_url": "https://en.wikipedia.org/wiki/Aegean_cat",
"hypoallergenic": 0,
"reference_image_id": "ozEvzdVM-",
"image_width": 1200,
"image_height": 800,
"image_url": "https://cdn2.thecatapi.com/images/ozEvzdVM-.jpg"
}
```

### **Comment**

```
{
  "comment_id": 0,
  "author_user_id": 0,
  "parent_comment_id": null,
  "breed_id_int": 0,
  "creation_date": "9999-12-31 23:59:59"
  "message": "cute"
}
```

### **User**

```
{
  "user_id": 0,
  "user_name": "Naruto",
  "user_age": 22,
```

```
“isAdmin”: false,  
“password”:123456,  
“image_url”:””,  
“creation_date”:2001-01-01,  
“last_edit_date”:2001-01-01  
}
```

### **Like**

```
{  
“user_id”:0,  
“comment_id”:0  
}
```

### **Favorite**

```
{  
“user_id”:0,  
“breed_id”:0  
}
```

### **Event**

```
{  
“user_id”:0,  
“type_id”:0,  
“breed_id”:0,  
“date”:2001-01-01  
}
```

### **EventType**

```
{  
“user_id”:0,
```

```
“type_id”:0,  
“breed_id”:0,  
“event_name”:”comment”  
}
```

## **5. Примеры запросов**

### **Добавление пользователя:**

```
INSERT INTO USER (user_id, ..., ) VALUES(...);
```

### **Получение списка всех пород/фильтрация:**

```
SELECT * FROM BREED; SELECT * FROM BREED WHERE activity=5;
```

### **Получение карточки породы (один запрос):**

```
SELECT * FROM BREED WHERE breed_id =”breed_id”;
```

### **Добавления комментария:**

```
INSERT INTO COMMENT (comment_id, author_user_id, parent_comment_id,  
breed_id, creation_date, message) VALUES (...);
```

### **Получение комментариев:**

```
SELECT * FROM COMMENT WHERE breed_id =”breed_id”;
```

### **Лайк комментарию:**

```
INSERT INTO LIKE (user_id, comment_id) VALUES(...);
```

### **Удаление лайка:**

```
DELETE FROM LIKE WHERE user_id =”user_id” && comment_id  
=”comment_id”;
```

### **Профиль пользователя:**

```
SELECT * FROM USER WHERE user_id =”user_id”;
```

### **Изменение профиля пользователя:**

```
UPDATE USER SET user_name =”user_name” ... WHERE user_id =”user_id”;
```

### **Статистика:**

```
SELECT * FROM event INNER JOIN eventtype ON type_id WHERE user_id  
=”user_id”;
```

### **3.3 Сравнение моделей**

#### **Удельный объем информации:**

При сравнении удельного объема информации получилось, что нереляционная модель занимает  $161200 + 140513 \times n$  байтов, а реляционная модель занимает  $394200 + 33822 \times n$  байтов. Таким образом, при количестве пользователей более 2-х реляционная модель будет занимать меньше места, чем нереляционная. Так как случай с двумя и менее пользователями не представляет интереса для нашей задачи, эти случаи можно не рассматривать.

#### **Запросы по отдельным юзкейсам:**

Для всех юзкейсов реляционная модель требует одного запроса, а нереляционная для части юзкейсов (добавление, удаление комментариев, добавление, удаление лайков) требует два запроса в две коллекции.

## 4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

### 4.1 Краткое описание

Весь код приложения разделен на две основные части: back-end и front-end, каждая из которых выполняет свою уникальную роль в архитектуре приложения.

**Back-end** реализован с использованием фреймворка Spring Boot, который обеспечивает создание RESTful API для взаимодействия с клиентской частью приложения. Бэкенд отвечает за обработку запросов от фронтенда, а также за взаимодействие с базой данных MongoDB, где хранится информация о породах кошек, пользователях и комментариях.

Основные функции бэкенда включают:

Обработка API-запросов: Бэкенд принимает запросы от фронтенда, обрабатывает их и возвращает соответствующие ответы. Например, при запросе информации о конкретной породе кошки, бэкенд извлекает данные из базы данных и отправляет их обратно на фронтенд.

Управление данными: Бэкенд реализует CRUD-операции (создание, чтение, обновление, удаление) для управления данными о породах кошек и комментариях пользователей. Это позволяет пользователям добавлять свои комментарии, просматривать информацию о породах и взаимодействовать с другими пользователями.

Аутентификация и авторизация: Бэкенд также отвечает за управление пользователями, включая регистрацию, вход в систему и проверку прав доступа, что обеспечивает безопасность приложения.



**Front-end** разработан с использованием библиотеки React, что позволяет создавать динамичные и отзывчивые пользовательские интерфейсы. Фронтенд взаимодействует с API бэкенда для получения и отображения данных, а также для отправки пользовательских действий, таких как добавление комментариев.

Ключевые аспекты фронтенда включают:

Структура проекта: В проекте используется компонентный подход, что позволяет разделить интерфейс на переиспользуемые компоненты. Основные директории включают:

- `src/pages`: Здесь находятся страницы приложения, которые обеспечивают навигацию и отображение различных представлений, таких как список пород кошек и страницы с деталями о конкретной породе.

- `src/components`: В этой директории хранятся компоненты, реализующие отдельные функциональные части интерфейса, такие как формы для добавления комментариев, списки пород и кнопки.

- `src/ui`: Данная директория содержит кастомные текстовые формы, селекторы, дэйтпикеры, чекбоксы и иконки, которые используются в различных компонентах приложения

Управление состоянием: Работа с данными на фронтенде реализуется через систему управления состоянием с использованием библиотеки Redux. Это позволяет эффективно управлять состоянием приложения, обеспечивая централизованное хранилище для всех данных, что упрощает взаимодействие между компонентами и позволяет легко отслеживать изменения состояния.

## **4.2 Используемые технологии**

БД: MongoDB.

Back-end: Java, Spring, JavaScript

Front-end: JavaScript, React, JSX, HTML, CSS.

API: Cat API

## 4.3 Схема экранов приложения

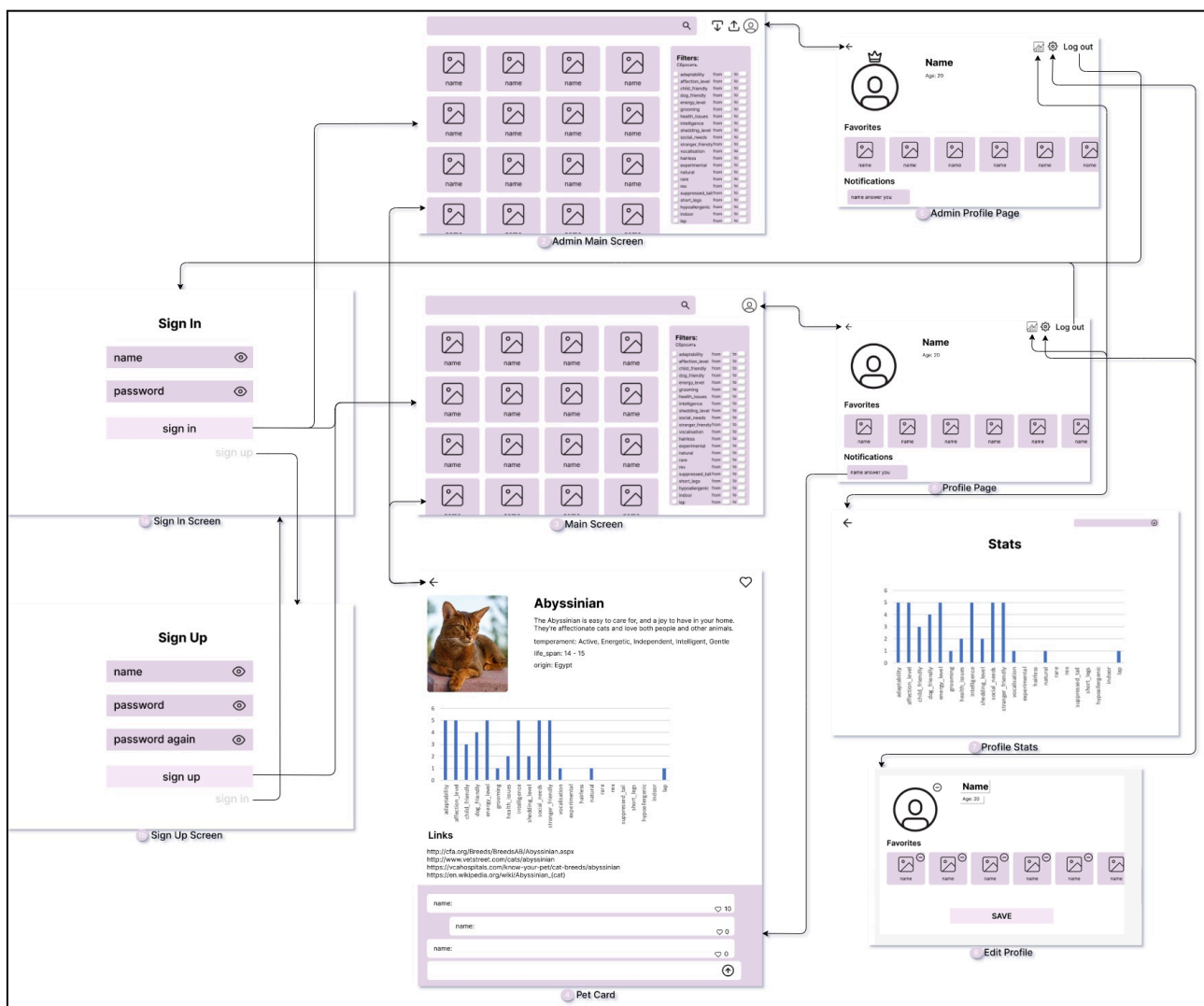


Рисунок 10. Схема экранов приложения

## 5. ВЫВОДЫ

### 5.1 Достигнутые результаты

В ходе разработки было создано веб-приложение, предназначенное для работы с породами кошек. Это приложение предоставляет пользователям множество возможностей, включая поиск информации о различных породах, добавление новых пород в базу данных, а также возможность оставлять комментарии и отзывы о каждой породе.

Ключевые функции приложения включают:

Поиск пород: Пользователи могут легко находить интересующие их породы кошек, используя удобный интерфейс и фильтры.

Добавление пород: Приложение позволяет пользователям вносить новые породы в базу данных, что способствует расширению информации и актуальности данных.

Комментарии и отзывы: Пользователи могут оставлять свои мнения и отзывы о породах, что создает сообщество и позволяет обмениваться опытом.

Регистрация и редактирование профиля: Реализована система регистрации, которая позволяет пользователям создавать свои профили и редактировать их по мере необходимости. Это обеспечивает персонализированный опыт использования приложения.

Статистика и фильтрация данных: Пользователи могут просматривать статистику по породам и использовать фильтры для более удобного поиска информации.

Интеграция с Cat API: Приложение использует Cat API для получения актуальной информации о породах кошек, что гарантирует, что данные всегда будут свежими и точными.

Импорт и экспорт данных: Реализованы функции импорта и экспорта данных, что позволяет пользователям удобно переносить информацию и создавать резервные копии.

## **5.2 Недостатки и пути для улучшения**

Несмотря на достигнутые результаты, приложение имеет несколько недостатков, которые могут повлиять на его дальнейшее развитие и использование:

Недостаток мультимедийных материалов: В приложении отсутствует возможность загрузки мультимедийных материалов, таких как видео о породах кошек. Это ограничивает возможности пользователей в получении более полной информации о породах. В будущем можно рассмотреть добавление функции загрузки видео, что позволит пользователям делиться своими впечатлениями и опытом, а также улучшит восприятие информации.

Ограниченная языковая поддержка: В настоящее время приложение доступно только на русском языке, что может ограничить его аудиторию. Для расширения пользовательской базы стоит добавить поддержку нескольких языков, что сделает приложение доступным для международных пользователей.

Отсутствие мобильной версии: В настоящее время приложение доступно только в веб-формате. Разработка мобильных версий для платформ iOS и Android значительно повысит доступность и удобство использования приложения, позволяя пользователям взаимодействовать с ним в любое время и в любом месте.

## **5.3 Будущее развитие решения**

Предполагается внедрение возможности добавление функции загрузки видео и мультимедийных материалов для более полного представления о кошках. Также планируется поддержка многоязычности и создание мобильных версий для улучшения доступности приложения.

## **6. ПРИЛОЖЕНИЯ**

### **6.1 Документация по сборке и развертыванию приложения.**

1. Клонировать репозиторий с проектом с помощью команды:

```
git clone git@github.com:moevm/nosql2h24-dogs.git
```

2. Перейти в директорию проекта

3. Собрать контейнеры приложения с помощью среды разработки или через терминал используя команду:

```
docker-compose build --no-cache
```

4. Запустить контейнеры с помощью среды разработки или через терминал используя команду:

```
docker-compose up
```

5. Открыть приложение в браузере по адресу: <http://127.0.0.1:3000> или <http://localhost:3000>

### **6.2 Инструкция для пользователя.**

#### **1. Оставление отзыва.**

Для оставления отзыва нужно прокрутить страницу до блока с отзывами, нажать кнопку "Оставить отзыв" и заполнить форму. В обязательном порядке нужно указать имя, возраст, выбрать породу и оценить её.

#### **2. Просмотр статистики.**

Для просмотра статистики по породам или пользователям нужно выбрать "Кастомизированную статистику" на главной странице. Далее пользователь может выбрать дату и тип статистики. Приложение автоматически сгенерирует отчет на основе введенных данных.

## 7. ЛИТЕРАТУРА

1. Ссылка на GitHub - [Электронный ресурс]. URL: <https://github.com/moevm/nosql2h24-dogs>.
2. React - [Электронный ресурс]. URL: <https://reactjs.org/> (дата обращения: 12.12.2024).
3. Spring Framework - [Электронный ресурс]. URL: <https://spring.io/> (дата обращения: 12.12.2024).
4. MongoDB - [Электронный ресурс]. URL: <https://www.mongodb.com/> (дата обращения: 12.12.2024).
5. Cat API - [Электронный ресурс]. URL: <https://thecatapi.com/> (дата обращения: 12.12.2024).
6. NPM - [Электронный ресурс]. URL: <https://www.npmjs.com/> (дата обращения: 12.12.2024).