

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра математического обеспечения и применения ЭВМ**

**КУРСОВОЙ ПРОЕКТ**  
**по дисциплине «Введение в нереляционные базы данных»**  
**Тема: Разработка приложения «Каталог пород кошек»**

Студент гр. 8382	_____	Нечепуренко Н.А.
Студент гр. 8382	_____	Терехов А.Е.
Студент гр. 8382	_____	Мирончик П.Д.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург

2021

## ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студенты: Нечепуренко Н.А., Терехов А.Е., Мирончик П.Д.

Группа 8382

Тема работы: Разработка приложения «Каталог пород кошек»

Исходные данные:

Необходимо разработать веб-приложение со списком пород кошек и их некоторыми характеристиками. Реализованное приложение должно использовать Memcached в качестве СУБД.

Содержание пояснительной записки:

«Содержание», «Введение», «Качественные требования к решению», «Сценарии использования», «Модель данных», «Разработанное приложение», «Экспериментальная проверка теоретических оценок для Memcached», «Заключение», «Приложения».

Предполагаемый объём пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 06.09.2021

Дата сдачи работы: 26.12.2021

Студент гр. 8382

\_\_\_\_\_

Нечепуренко Н.А.

Студент гр. 8382

\_\_\_\_\_

Терехов А.Е.

Студент гр. 8382

\_\_\_\_\_

Мирончик П.Д.

Преподаватель

\_\_\_\_\_

Заславский М.М.

## АННОТАЦИЯ

В данном проекте было разработано веб-приложение для просмотра и фильтрации списка пород кошек. Реализация приложения предполагала одно ограничение – в качестве СУБД должен использоваться Memcached.

В процессе работы был разработан макет пользовательского интерфейса, анализ основных сценариев использования приложения конечным пользователем. Также была построена модель данных с учетом ограничений выбранного инструмента по работе с данными. Был проведен сравнительный анализ решения на основе Memcached и решения на основе реляционной БД Postgres по объему хранения данных и количеству запросов для CRUD операций.

Клиентская часть приложения была написана с использованием библиотеки React, серверная часть разработана на Java с использованием фреймворка Spring. Разработанное приложение может запущено с помощью средств контейнеризации и оркестрации, в частности docker и docker-compose.

## СОДЕРЖАНИЕ

<b>Введение</b>	<b>6</b>
<b>Качественные требования к решению</b>	<b>7</b>
<b>Сценарии использования</b>	<b>8</b>
Макет пользовательского интерфейса . . . . .	8
Основные сценарии использования . . . . .	8
Вывод . . . . .	10
<b>Модель данных</b>	<b>11</b>
Схема данных . . . . .	11
Нереляционная модель данных . . . . .	12
Оценка удельного объема информации, нереляционная модель . . .	14
Избыточность нереляционной модели . . . . .	15
Запросы к нереляционной СУБД . . . . .	16
Реляционная модель данных . . . . .	18
Запросы к реляционной СУБД . . . . .	19
Выводы . . . . .	20
<b>Разработанное приложение</b>	<b>22</b>
Краткое описание . . . . .	22
<b>Экспериментальная проверка теоретических оценок для Memcached</b>	<b>27</b>
<b>Заключение</b>	<b>29</b>
<b>Приложения</b>	<b>30</b>

Сборка и развертывание приложения . . . . .	30
---	----

## ВВЕДЕНИЕ

В настоящее время, в период пандемии, люди вынуждены проводить дома большую часть своего времени. Известно, что мелкие домашние животные, как, например, кошки, способны уменьшить эффект изоляции от социума, воспитать в людях доброту и другие положительные качества. К сожалению, некоторые люди подходят к выбору домашнего животного безответственно, из-за чего страдает как человек, так и животное.

Кошки дрессируются сложнее собак, они более независимые и их поведение может определяться не столько воспитанием, сколько некоторым присущим породе характеристикам. Среди задач веб-приложения можно выделить:

- Отображение списка пород кошек, взятого из публичных источников.
- Возможность прочитать описание отдельной породы, ее историю, особенности здоровья и некоторую другую справочную информацию.
- Возможность фильтровать породы кошек некоторым характеристикам, например, дружелюбность к детям, привязанность к семье и т.д.

Для решения данных задач было разработано веб-приложение на стеке React + Spring с использованием Memcached в качестве основного инструмента СУБД.

## 1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Разработанное решение должно обладать следующими свойствами/решать следующие задачи:

- В качестве СУБД используется Memcached
- Существует возможность импорта и экспорта БД
- В приложении можно осуществлять фильтрацию пород по численным характеристикам
- Приложение может быть развернуто средствами контейнеризации и оркестрации

## 2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

### 2.1. Макет пользовательского интерфейса

Общий вид макета пользовательского интерфейса представлен на рисунке 1.



Рисунок 1 – Макет пользовательского интерфейса

### 2.2. Основные сценарии использования

#### Фильтрация

1. Пользователь открывает главную страницу.
2. Пользователь выбирает нужные фильтры: строка поиска и числовые



фильтры.

3. Пользователь нажимает кнопку "Find".
4. В список ниже загружаются найденные результаты в виде карточек.

### **Открытие детальной информации о коте**

1. Возможны 3 варианта:
  - Пользователь открывает главную страницу и в списке котов нажимает на желаемую карточку.
  - Пользователь вводит в адресную строку ссылку на интересующего его кота.
  - Пользователь открывает просмотр всех данных в виде таблицы и выбирает интересующую его строку.
2. Открывается страница с детальной информацией о коте.

### **Просмотр графика зависимости одного параметра от другого**

1. Пользователь заходит на главную страницу.
2. Опционально. Пользователь фильтрует данные (см п. Фильтрация).
3. Пользователь нажимает кнопку "Compute graph".
4. Открывается страница с графиком для найденных результатов.
5. На графике пользователь выбирает из выпадающего списка поля по оси X и по оси Y.
6. Строится график зависимости поля Y от поля X. Для каждого значения X находятся коты с соответствующим значением и среди них считается среднее Y, которое берется как Y значение точки на графике.

### **Экспорт найденных результатов**

1. Пользователь заходит на главную страницу.
2. Опционально. Пользователь фильтрует данные.
3. Пользователь в разделе экспорта выбирает формат экспорта и нажимает на кнопку.

4. Выполняется экспорт данных путем переадресации на страницу, где на сервере формируется ответ в виде файла.

#### **Экспорт всех данных базы**

1. Пользователь заходит на главную страницу.
2. Пользователь в разделе экспорта всех данных выбирает тип экспорта.
3. Выполняется экспорт данных путем переадресации на страницу, где на сервере формируется ответ в виде файла.

#### **Просмотр содержимого базы в виде таблицы**

1. Пользователь заходит на главную страницу.
2. Пользователь в разделе экспорта всех данных выбирает "View as a table".
3. Открывается страница с данными базы, представленными в виде таблицы. В ней отображены только те поля, которые можно кратко записать в ячейки (длинные описания опущены).

### **2.3. Вывод**

На основании рассмотренных сценариев использования было выявлено, что основные операции при взаимодействии пользователей с приложением – операции чтения.

### 3. МОДЕЛЬ ДАННЫХ

#### 3.1. Схема данных

В приложении используется единственная сущность – Порода кошки.

Порода описывается следующими свойствами

*Таблица 1 – Описание свойств породы*

Название свойства	Описание
Название породы	Строка, может быть использована как первичный ключ
Привязанность к семье	Число от 0 до 5
Склонность к линьке	Число от 0 до 5
Простота ухода	Число от 0 до 5
Доброжелательность к незнакомцам	Число от 0 до 5
Общее здоровье	Число от 0 до 5
Общий интеллект	Число от 0 до 5
Доброжелательность к детям	Число от 0 до 5
Доброжелательность к животным	Число от 0 до 5
Игривость	Число от 0 до 5
Склонность к пению	Число от 0 до 5
Длина	Строка, пример "Medium"
Продолжительность жизни	Строка, пример "12 to 16 years"
Регион происхождения	Строка, пример "California, USA"
Вес	Строка, пример "5 to 10 pounds"
Информация об уходе	Текст
Дети и животные	Текст

Цвет и уход	Текст
Здоровье	Текст
История породы	Текст
Характер	Текст
Короткое описание	Текст
Изображение	Ссылка
Размер	Текст

Список пород кошек – почти постоянный, в том смысле, что может быть занесен в программу один раз и будет изменяться крайне редко. Основными запросами к приложению будут **получение** полной информации о каждой породе и **фильтрация** списка пород по заданным параметрам (например, по уровню интеллекта или продолжительности жизни).

Исходя из этого будет выбираться модель представления, в которой упомянутые операции будут выполняться быстрее других (изменение, добавление, удаление).

### 3.2. Нереляционная модель данных

В качестве нереляционной БД в приложении используется Memcached. Memcached позволяет хранить отображения вида строка-строка (грубо Map<String, String>), не имеет поддержки конкурентности, транзакционности и прочих прелестей, что накладывает серьезные ограничения на способ представления данных.

Также упомянем, что задание включает в себя требование о том, что некоторые операции над данными должны совершаться средствами используемой БД, что тяжело себе вообразить с таким инструментом, как Memcached.

Исходя из ограничений используемого инструмента и требований к быстротедействию операций, положим следующую модель.

Будем хранить 3 типа отображений: `all_cats, { breed_name }. { characteristic_name }` , `{ vital_stat / characteristic_name }. { vital_stat_value / characteristic_value }`.

Первое отображение специальное, имеет зарезервированный ключ и позволяет получить строку со списком всех названий пород.

Отображения вида `{ breed_name }. { characteristic_name }` позволяют хранить свойства породы. Например, `Abyssinian.health` содержит "Both pedigreed cats and mixed-breed cats have varying incidences...".

Отображения вида `{ vital_stat / characteristic_name }. { vital_stat_value / characteristic_value }` хранят список пород, удовлетворяющих заданным характеристикам. Например, `length.Medium` или `intelligence.4` содержит `Abyssinian` и др.

Отображение первого типа позволяет поддерживать список первичных ключей.

Отображения второго типа позволяют связывать первичный ключ записи с остальными полями.

Отображения третьего типа позволяют оптимизировать процесс фильтрации по характеристикам.

Графически модель приведена на рисунке 2.

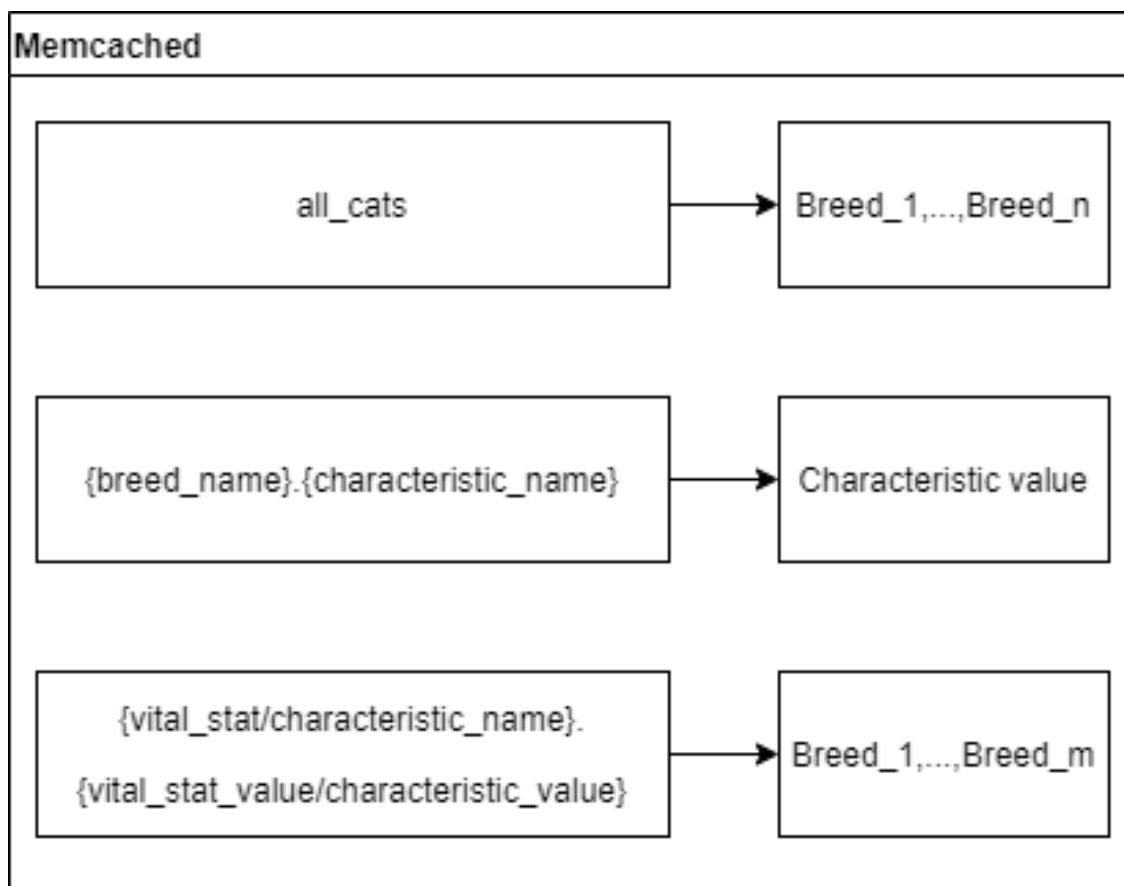


Рисунок 2 – Модель данных в Memcached

### 3.3. Оценка удельного объема информации, нереляционная модель

Размер символа определяется клиентской стороной. Примем его равным 2 байтам.

Пусть  $N$  – количество пород в базе.

Тогда отображение первого типа будет занимать  $8$  ("all\_cats") +  $(N-1)(,) + N * \text{BNAME\_SIZE}$ . Приняв верхнюю оценку для  $\text{BNAME\_SIZE}$  равную 20 символов получим примерно  $21N$  символов.

Отображения второго типа разделим на два вида: значения представленные числом или короткой строкой (до 30 символов) и значения, содержащие текст с длиной больше 30 символов.

Рассмотрим первый вид. Размер ключа  $BNAME\_SIZE + 1 (.) + CNAME\_SIZE = 46$ , при выборе  $CNAME\_SIZE$  в качестве 25 символов. 25 символов – длина "Potential for Playfulness". Значения первого вида – числа или короткие строки, например, 4 или "Quebec, Canada их размер оценим в 30 символов. Для каждой кошки имеется 14 характеристик подобного вида, получаем  $N * (46 + 30) * 14 = 1064N$  символов. Это верхняя оценка.

Рассмотрим второй вид. Размер ключа  $BNAME\_SIZE + 1 (.) + CNAME1\_SIZE = 36$ , при выборе  $CNAME1\_SIZE$  в качестве 15 символов. Для каждой кошки имеется 9 характеристик с текстом (url-картинки отнесем к этому виду), размер текста в среднем для каждой характеристики кошки примем равным 1000 символов. Получаем оценку  $N * (36 * 9 + 9 * 1000) = 9324N$  символов.

Итого для отображения второго типа  $1064N + 9324N = 10388N$  символов.

Рассмотрим отображения третьего типа. Размер ключа  $CNAME\_SIZE + 1 (.) + CHAR\_VAL\_LENGTH = 56$ ,  $CHAR\_VAL\_LENGTH = 30$  символов. Для численных значений характеристик имеем  $6 * 10 * KEY\_SIZE + N * BNAME\_SIZE = 3360 + 20N$ , так как у каждой кошки есть значение численное значение характеристики и только одно. Со строковыми характеристиками сложнее, они очень сильно варьируются, поэтому дадим грубую оценку в  $M * KEY\_SIZE + N * BNAME\_SIZE = N * 56 + 20N = 76N$ , где в качестве  $M$  было взято  $N$  – все значения различны. В реальных данных есть пересечения и размер модели будет меньше.

Итого получаем верхнюю оценку  $21N + 10388N + (76N + 20N + 3360) = 10505N + 3360$ . Или  $21010N + 6720$  байтов.

Рост происходит линейным образом.

### 3.4. Избыточность нереляционной модели

Для каждой характеристики имеем избыточность в виде повторного хранения названия породы для маппинга и хранения названия характеристики. Избыточность этого вида можно оценить как  $N * (14 * (CNAME\_SIZE + BNAME\_SIZE + 1) + 9 * (CNAME1\_SIZE + BNAME\_SIZE + 1)) = 968N$ .

Также все отображения третьего типа избыточны, они используются в поиске/фильтрации, их размер равен  $96N + 3360$ .

Общую избыточность можно оценить как  $1064N + 3360$ .

Для оценки избыточности получаем формулу  $(10505N + 3360) / (9441N)$ . В пределе для выражения для избыточности получаем значение 1.112.

### 3.5. Запросы к нереляционной СУБД

Запросы к Memcached имеют вид `get/add/remove key` и не представляют особого интереса.

Оценим количество запросов к БД для *CRUD* операций.

#### Добавление

Пусть нужно сохранить одну кошку породы Barsik. Тогда нужно прочитать кортеж по ключу `all_cats`, добавить в него строку Barsik и сохранить обратно. Получим 2 операции с базой. Количество ключей не изменится.

Затем для каждой строковой характеристики, которых может быть 9, нужно добавить запись `Barsik.characteristic_name = characteristic_value`. Еще 9 операций и столько же новых ключей.

Добавление составных характеристик.

- Во-первых, имеется 10 числовых характеристик, которые могут принимать 6 значений, но в случае с добавлением одной кошки нужно создать 10 новых пар ключ-значение или добавить в существующий кортеж 10



значений. Получим 20 операций с базой и 10 новых ключей.

- Во-вторых, есть 4 характеристики, значения которых пересекаются, но не очень часто, поэтому зачастую придется создавать новые пары, а не модифицировать существующие. Еще 6 операций и 4 новых ключа. Если кошек 50, то ключей в конечном итоге будет примерно 200.

Подводя итог, добавление в базу одной кошки требует 37 операций и 23 новых ключа. При 50 кошках получим 1850 операций и 1150 ключей. Выглядит максимально не оптимально, но таковы ограничения используемой базы.

### **Чтение**

Рассмотрим чтение кошек отфильтрованных по какому-либо параметру. Чтобы получить список пород удовлетворяющих параметру, нужно сделать всего один запрос. Но чтобы восстановить кошку, то есть собрать все поля из базы, нужно сделать:

- 9 запросов для каждой текстовой характеристики;
- 14 запросов для остальных характеристик;

Имеем 25 запросов в базу.

Чтение JSON сериализованной кошечки и десериализация могли бы выполняться за 1 запрос, но этого могло бы нарушить ограничения на использования БД (все основные операции должны выполняться средствами БД).

### **Обновление**

- Если пришлось изменить название породы, то нужно изменить его в кортеже `all_cats`, изменить 23 ключа и изменить его еще в 260 кортежах, где эта порода потенциально может быть. Итого 545 операций с базой.
- Чтобы изменить описательную характеристику, потребуется одна операция.
- Если изменяется какая-либо составная характеристика кошки, то нужно удалить кошку из кортежа с прошлой характеристикой, и добавить в

кортеж с новой характеристикой. Итого 2 операции с базой.

### **Удаление**

Удаляется кошка путем удаления названия породы из кортежа `all_cats` и еще 260 кортежей с характеристиками, и также удалится 23 пары ключ-значение. Всего  $(260 + 1) * 2 + 23 = 545$  операции с базой.

### **3.6. Реляционная модель данных**

Модель состоит из идентификатора, названия породы, 9 строковых характеристик, 10 числовых характеристик и 4 оставшихся строковых характеристик.

В качестве SQL решения будем опираться на Postgres.

Графическое представление модели приведено на рисунке 3.

Cat
id : bigserial not null primary key
breed_name : varchar not null
care : varchar
children_and_pets : varchar
color_and_grooming : varchar
health : varchar
history : varchar
personality : varchar
round_img_url : varchar
short_description : varchar
size : varchar
affectionate_with_family : int
amount_of_shedding : int
easy_to_groom : int
friendly_toward_strangers : int
general_health : int
intelligence : int
kid_friendly : int
pet_friendly : int
potential_for_playfulness : int
tendency_to_vocalize : int
length : varchar
life_span : varchar
origin : varchar
weight : varchar

Рисунок 3 – Модель данных в Postgres

### **Оценка удельного объема информации, хранимой в модели**

Пусть в базе лежит  $N$  пород, размер символа 2 байта, bigserial - 8, int - 4, тогда размер модели будет равен  $N * (8 + 20 + 9 * 1000 * 2 + 10 * 4 + 4 * 30 * 2) = 18308N$  байтов.

### **Избыточность модели**

В модели нет избыточности.

### 3.7. Запросы к реляционной СУБД

Для добавления кошки в таблицу нужен 1 запрос:

```
INSERT INTO cat (breed, care, ...)
VALUES (?, ?...);
```

При этом в таблицу добавится только 1 строка.

Для поиска кошки по любому параметру так же нужен всего 1 запрос:

```
SELECT *
FROM cat
WHERE param1 = ? AND param2 = ?
    OR param3 > ?;
```

Для обновления информации о кошке нужен вновь всего 1 запрос:

```
UPDATE cat
SET param1=?, ....
WHERE id=?;
```

Для удаления кошки так же 1 запрос:

```
DELETE
*
FROM cat
WHERE id = ?;
```

### 3.8. Выводы

Для нереляционной модели была получена оценка  $21010N + 6720$  байтов, где  $N$  – количество пород. Это оценка сверху, в ней принято допущение, что текстовые характеристики занимают 1000 символов, хотя они практически все меньше. Memcached позволяет хранить их без необходимости дополнения до 1000 символов. С другой стороны, модель имеет избыточные данные, используемые для маппинга ключа на свойства и вспомогательные отоб-

ражения для фильтрации.

Для реляционной модели была получена оценка 18308N байтов, данные хранятся без избыточности в одной таблице.

В обоих случаях объем данных растет линейно относительно количества пород.

При анализе количества запросов все встает на свои места.

Для выполнения необходимых функций в реляционной модели достаточно 1 запроса для каждого действия. В нереляционной модели для выполнения тех же операций необходимо поддерживать весьма специфические отображения, сложнее поддерживать согласованность данных, количество запросов неприлично большое даже для простейших CRUD операций.

В данной задаче совершенно неоправданно использование Memcached в качестве БД, как и во всех случаях его использования не по назначению. Memcached стоит использовать для кэширования примитивных значений, сериализованных объектов, но никак не для моделирования более сложных структур, требующих дополнительной логики.

## 4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

### 4.1. Краткое описание

Главная страница приложения выглядит следующим образом (см. рис. 4).

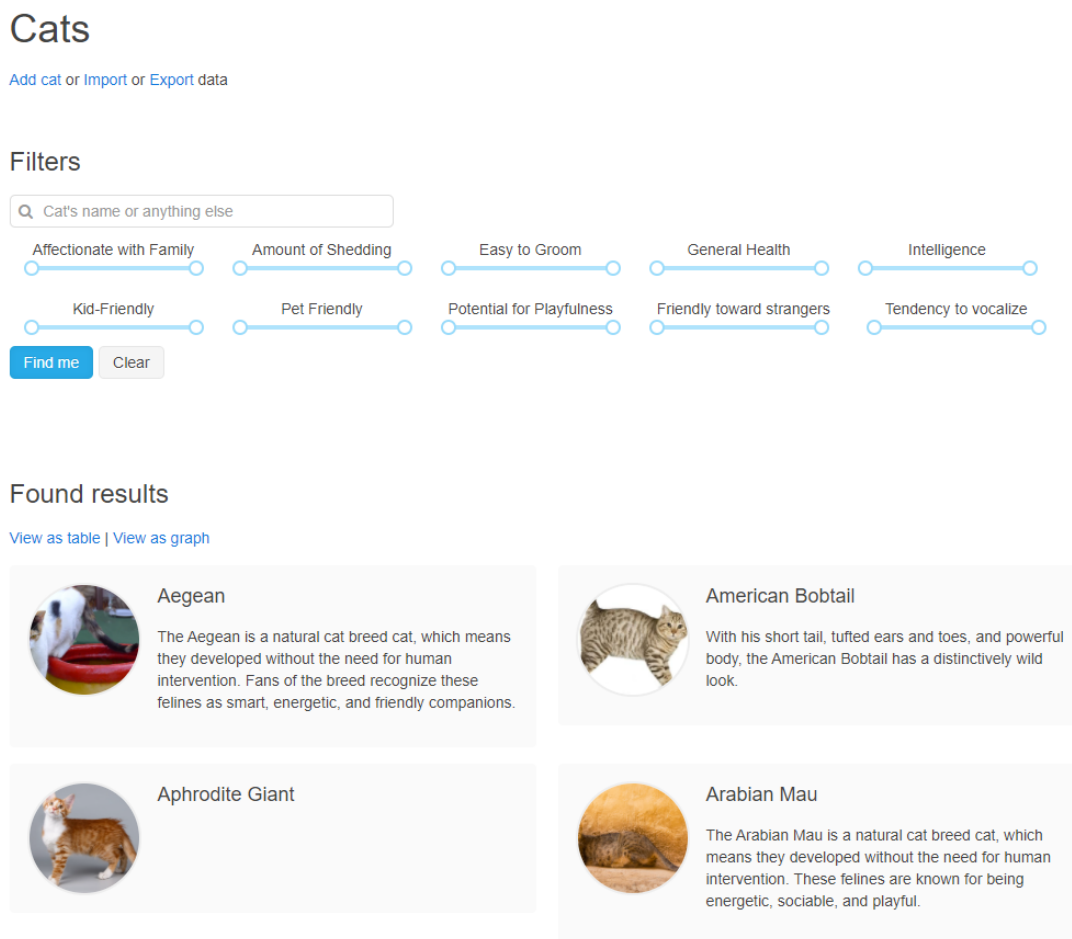


Рисунок 4 – Главная страница

Существует возможность поиска породы по названию (см. рис. 5).

# Cats

[Add cat](#) or [Import](#) or [Export data](#)

## Filters

Affectionate with Family

Amount of Shedding

Easy to Groom

General Health

Intelligence

Kid-Friendly

Pet Friendly

Potential for Playfulness

Friendly toward strangers


Tendency to vocalize

Find me

Clear

## Found results

[View as table](#) | [View as graph](#)



Aegean

The Aegean is a natural cat breed cat, which means they developed without the need for human intervention. Fans of the breed recognize these felines as smart, energetic, and friendly companions.

Рисунок 5 – Поиск породы по названию

Давайте выберем самых умных и ласковых с семьей кошек (см. рис. 6).

# Cats

[Add cat](#) or [Import](#) or [Export data](#)

## Filters

Affectionate with Family

Amount of Shedding

Easy to Groom

General Health

Intelligence

Kid-Friendly

Pet Friendly

Potential for Playfulness

Friendly toward strangers

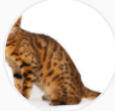
Tendency to vocalize

Find me

Clear


## Found results

[View as table](#) | [View as graph](#)



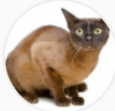
### Bengal Cats

The Bengal could never be called delicate. They're athletes: agile and graceful with a strong, muscular body, as befits a cat who looks as if they belong in the jungle.




### Chausie

The Chausie is a natural cat breed cat, which means they developed without the need for human intervention—in this case with the help of a natural mutation! These felines are known for being intelligent, energetic, and loving.



### European Burmese

The European Burmese has the charm and determination of his Siamese ancestors, and his voice is soft and sweet, belying his tendency to run the household with an iron paw sheathed in velvety fur.



### Lykoi

The Lykoi is a natural cat breed cat, which means they developed without the need for human intervention—in this case with the help of a natural mutation! These felines are known for being intelligent, energetic, and loving.

Рисунок 6 – Фильтрация пород по характеристикам

Список пород можно представить в виде таблицы (см. рис. 7).

Found results

[View as list](#) | [View as graph](#)

#	Name	Affectionate with Family	Amount of Shedding	Easy to Groom	General Health	Intelligence	Kid-Friendly	Pet Friendly	Potential for Playfulness	Friendly toward strangers	Tendency to vocalize	Length	Life span	Origin	Weight
1	Aegean	5	3	4	4	4	5	3	4	4	4	Medium	9 to 10 years	Greece	7 to 10 pounds
2	American Bobtail	4	4	3	4	4	4	4	4	4	3	Medium	11 to 15 years	United States and Canada	8 to 13 pounds
3	Aphrodite Giant	4	4	2	3	4	5	4	4	4	2	Large	12 to 15 years	Cyprus	11 to 24 pounds
4	Arabian Mau	4	2	5	4	4	4	3	4	4	3	Medium	12 to 14 years	Saudi Arabia	8 to 16 pounds
5	Asian	5	1	3	4	4	5	4	4	4	4	Medium	15 years	Great Britain	10 pounds
6	Australian Mist	5	2	4	4	4	4	3	4	3	2	Medium	15 to 18 years	Australia	8 to 15 pounds
7	Bambino	4	1	2	3	3	4	3	4	3	3	Short	9 to 15 years	United States	4 to 9 pounds
8	Bengal Cats	5	2	4	3	5	5	5	5	3	2	17 to 22 inches, not including tail	10 to 16 years	USA	8 to 17 pounds
9	Bombay	4	2	4	4	4	4	4	4	4	3	13 to 20 inches	12 to 20 years	Kentucky, USA	8 to 15 pounds
10	Brazilian Shorthair	4	1	4	3	4	5	4	4	4	3	Medium to large	14 to 20 years	Brazil	10 to 22 pounds
11	British	4	4	2	3	4	4	3	3	4	3	Medium to Large	15 to 17	Great Britain	9 to 18 pounds

Рисунок 7 – Представление пород в виде таблицы



Существует возможность просмотра графика для выборки для двух характеристик.

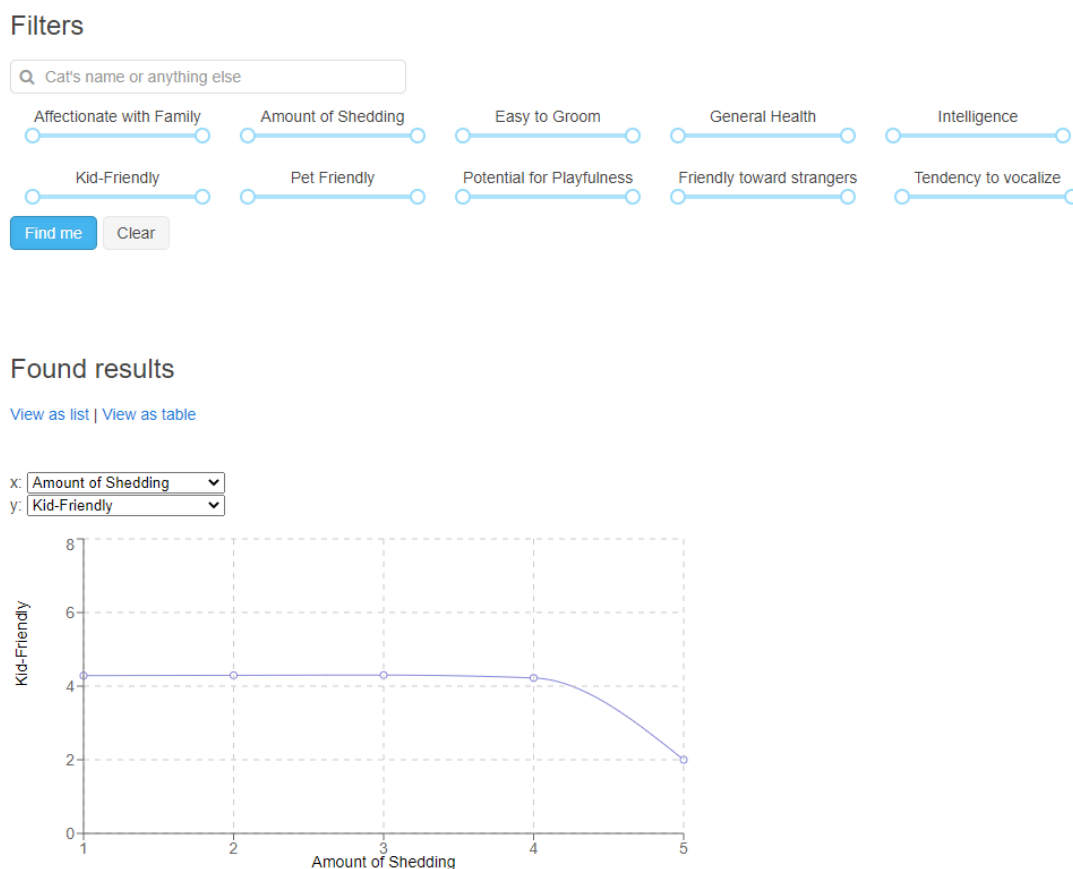


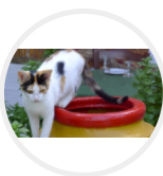
Рисунок 8 – График характеристик

Полученный график следует интерпретировать так: для кошек с характеристикой Amount of Shedding среднее Kid-Friendly равно 2.

Каждую породу можно рассмотреть отдельно в формате карточки (см. рис. 9).

[< Back](#)

## Aegean



The Aegean is a natural cat breed cat, which means they developed without the need for human intervention. Fans of the breed recognize these felines as smart, energetic, and friendly companions.

### Breed Characteristics:

Affectionate with Family: 5	Amount of Shedding: 3	Easy to Groom: 4	Friendly toward strangers: 4	General Health: 4
Intelligence: 4	Kid-Friendly: 5	Pet Friendly: 3	Potential for Playfulness: 4	Tendency to vocalize: 4

### Vital stats:

length: Medium	lifeSpan: 9 to 10 years	origin: Greece	weight: 7 to 10 pounds
----------------	-------------------------	----------------	------------------------

### History

The Aegean cat breed hails from the Greek islands of Cyclades, which are situated in the Aegean Sea. It's speculated that originating so close to the sea is why these felines are not afraid of water. The breed's friendliness towards humans is also said to come from their upbringing, being around sailors and fishermen all the time. In Greece, the Aegean is a super popular cat breed, although it is rarer to come across them in other countries. If you're considering adding an Aegean to your family, remember that many have ended up in shelters or in the care of rescue groups. Consider adoption if you decide this is the breed for you!

### Size

The Aegean is a medium-sized cat. As is often the case with cat breeds, exact size standards might vary. Most male Aegean cats weigh in at nine to ten pounds and most female Aegeans are between seven and nine pounds.

### Personality

The Aegean cat is one of the most friendly and sociable cat breeds around. They make great additions to any large family and will constantly look to seek out human interaction, whether that's to relax together during down times or for energetic play sessions. These athletic cats are also highly intelligent—in many cases

## Рисунок 9 – Карточка породы

Для импорта данных имеется простенькая форма (см. рис. 10).

[< Back](#)

## Import data

Выберите файл	Файл не выбран	Import
---------------	----------------	--------

## Рисунок 10 – Форма для импорта

## 5. ЭКСПЕРИМЕНТАЛЬНАЯ ПРОВЕРКА ТЕОРЕТИЧЕСКИХ ОЦЕНОК ДЛЯ MEMSCACHED

Зависимость фактического объема хранимых данных от количества пород приведена на рисунке 11. По оси X отложено количество пород, по оси Y количество занимаемого пространства в байтах.

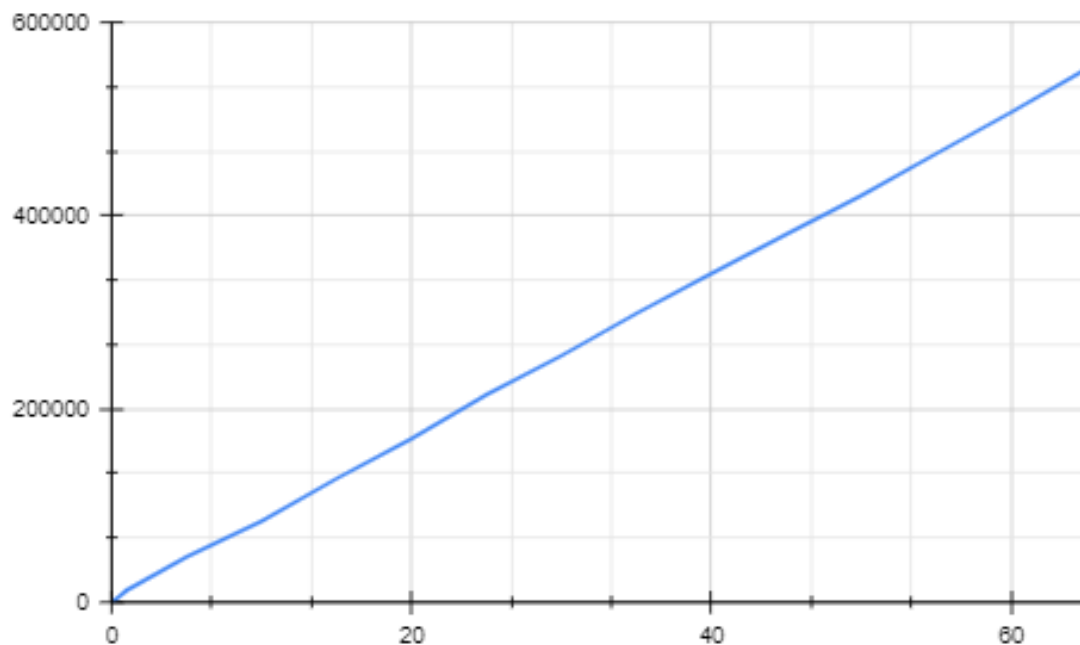


Рисунок 11 – Зависимость объема данных от числа пород

Теоретическая оценка асимптотика оказалась верной.

С временем выполнения операций немного сложнее. Существует некоторое постоянное время необходимое на установление соединения, время некоторой обработки данных, например, установка полей data transfer object. На рисунке 12 приведена зависимость времени выполнения операции от количества уже имеющихся пород в БД. По оси X отложено количество кошек в БД, по оси Y время в секундах

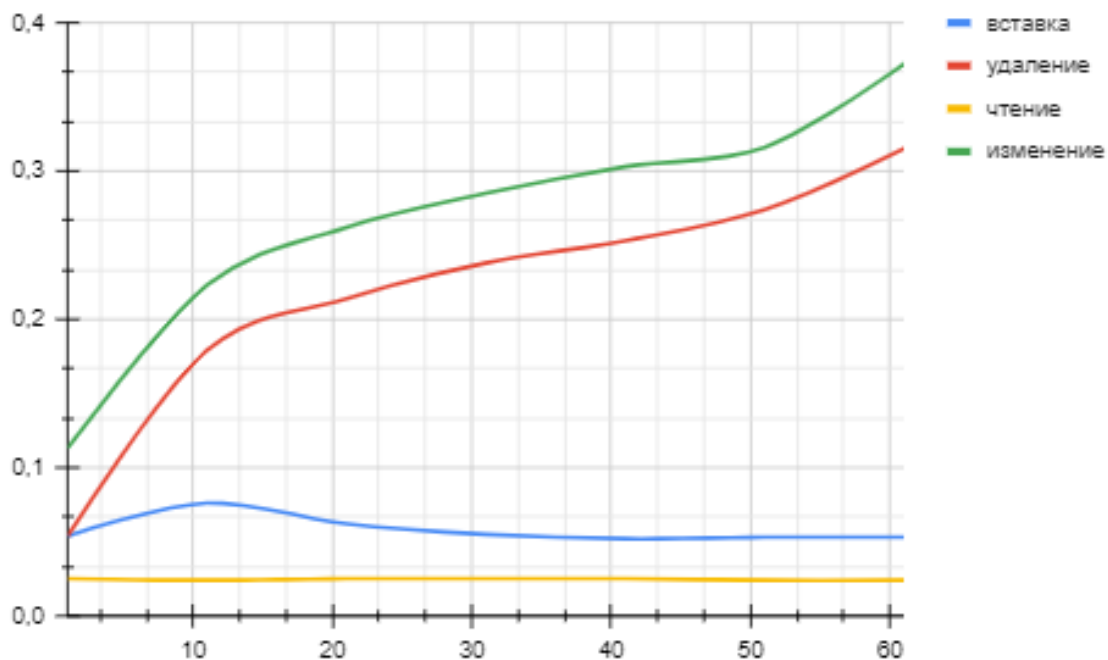


Рисунок 12 – Зависимость времени работы от числа пород

Стоит отметить, что пород кошек в мире существует не так много, поэтому решение не обязано масштабироваться на объем больший  $10^3$ . По графику можно определить, что чтение данных занимает константное время. Операции удаления и изменения наиболее затратные, как и ожидалось в теоретическом анализе.

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано приложение для поиска и фильтрации пород кошек по некоторым характеристикам.

Приложение позволяет:

- Просмотреть список пород кошек, взятый из известных источников
- Отфильтровать породы по некоторым желаемым характеристикам
- Импортировать и экспортировать данные
- Получить справочную информацию о породе

Дальнейшие шаги для улучшения решения:

- Использовать реляционную СУБД в качестве основной СУБД, а Memcached по его назначению как вспомогательный инструмент
- Для каждой породы добавить галерею с фотографиями
- Использовать больше публичных источников для сбора и корректировки данных

## **ПРИЛОЖЕНИЯ**

### **Сборка и развертывание приложения**

- Необходимо установить `docker` и `docker-compose`
- Запустить команду `docker-compose up` из корня проекта