

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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные базы данных»
Тема: Соц. сеть для кроудсорсинга

Студенты гр. 7304

Юреть Е.А.

Дементьев М.Е.

Субботин А.С.

Преподаватель

Заславский М.М.

Санкт-Петербург

2020

ЗАДАНИЕ

Студенты

Юреть Е.А.

Дементьев М.Е.

Субботин А.С.

Группа 7304

Тема проекта: разработка соц. Сети для кроудсорсинга

Исходные данные:

Необходимо разработать веб-приложение, которое позволит регистрацию в соц. сети для кроудсорсинга в качестве исполнителя и заказчика для просмотра, выполнения, комментирования задач и для их публикации на основе нереляционной модели данных Neo4j.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложения»

«Литература»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты гр. 7304

Юреть Е.А.

Дементьев М.Е.

Субботин А.С.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания приложения соц. сети для кроудсорсинга.

Найти исходный код и всю дополнительную информацию можно по ссылке: <https://github.com/moevm/nosql2h20-crowd-neo4j>

SUMMARY

Within the framework of this course, it was assumed any application in the team on one of the set topics. The topic of creating a social application was chosen. networks for crowdsourcing.

You can find the source code and additional information here: <https://github.com/moevm/nosql2h20-crowd-neo4j>

ОГЛАВЛЕНИЕ

Введение	6
1. Качественные требования к решению	7
2. Сценарии использования	8
3. Модель данных	19
4. Разработанное приложение.....	28
5. Выводы.....	33
6. Приложения.....	34
Используемая литература	35

ВВЕДЕНИЕ

Цель работы – создать удобное приложение для передачи некоторых задач и функций неопределенному кругу исполнителей.

Было решено разработать веб-приложение, которое позволит регистрацию в соц. сети для краудсорсинга в качестве исполнителя и заказчика для просмотра, выполнения, комментирования задач и для их публикации.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение с использованием нереляционной базы данных –Neo4j, с возможностями создания, решения, комментирования задач, а также добавления статуса задачам и пользователям. Задачи можно просматривать, сортируя по фильтрам, импортировать и экспортировать. В приложении есть возможность просмотра статистики и рейтинга пользователей.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. МАКЕТ UI

1. Экран просмотра всех заданий (рис. 1).

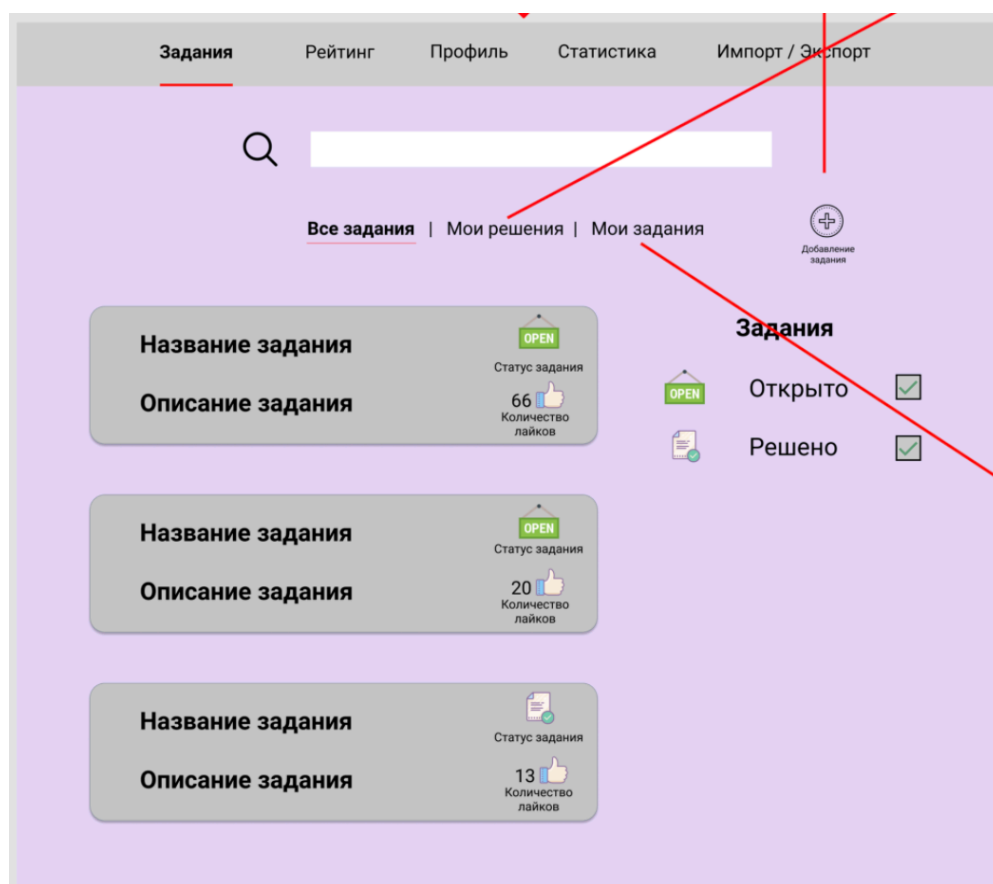


Рисунок 1. Экран просмотра всех заданий.

2. Экран добавления своего задания (рис. 2).

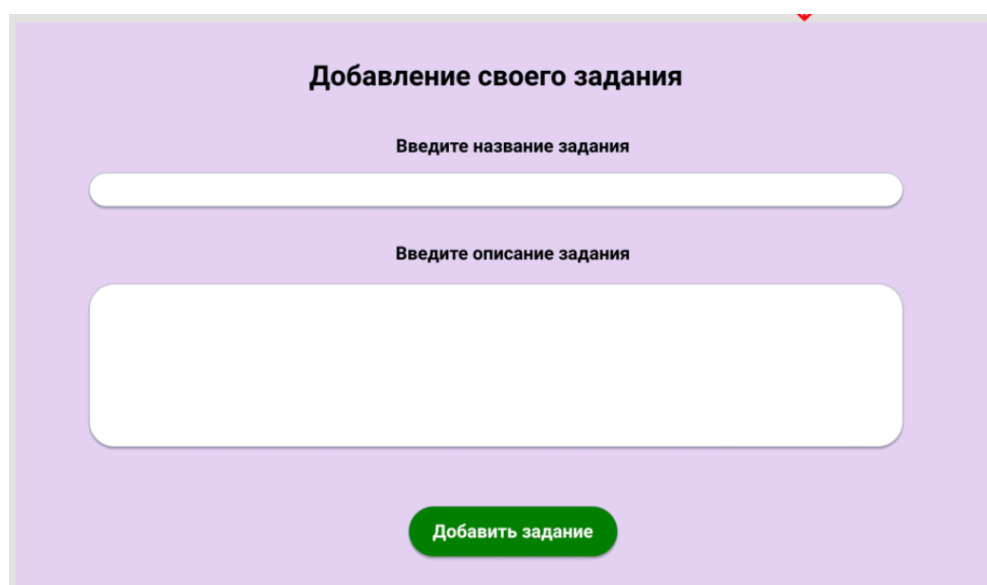


Рисунок 2. Экран добавления своего задания.

3. Экран заданий пользователя (рис. 3).

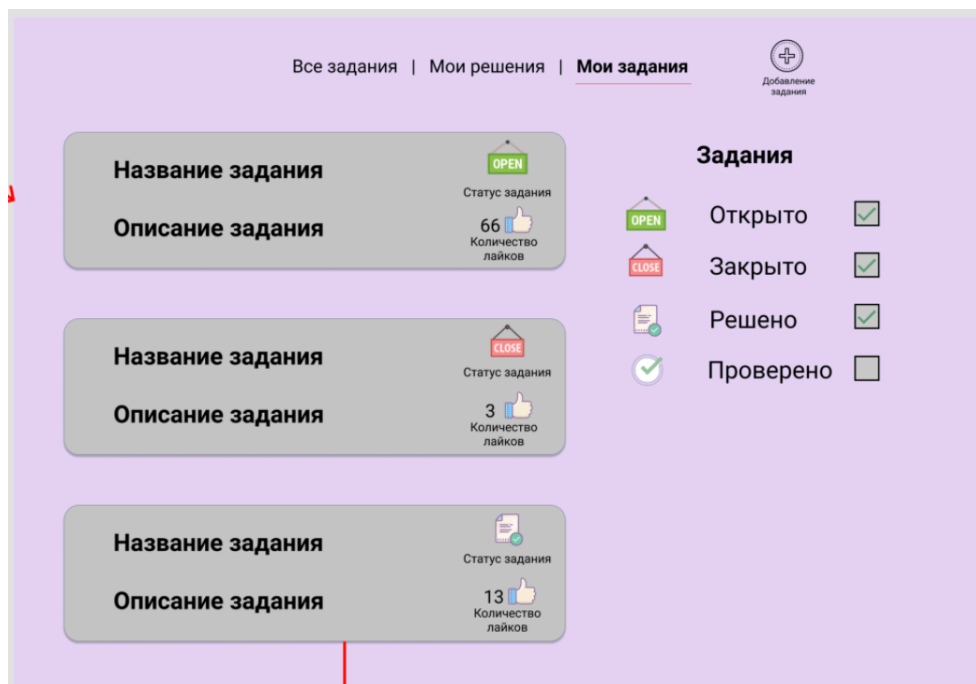


Рисунок 3. Экран заданий пользователя.

4. Экран решенных пользователем задач (рис. 4).

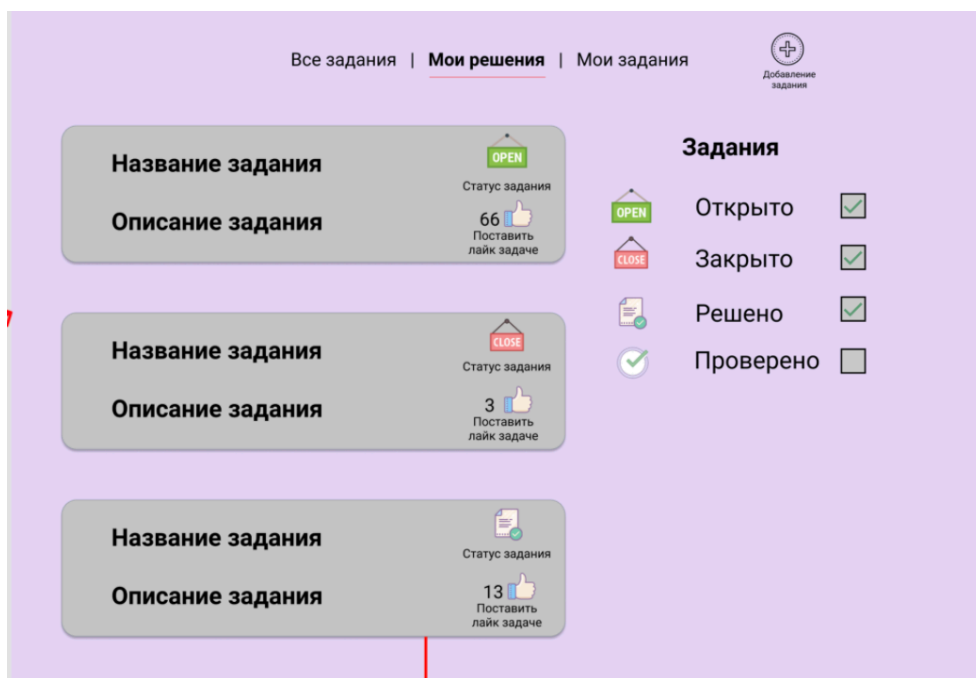


Рисунок 4. Экран решенных пользователем задач.

5. Экран конкретного задания пользователя (рис. 5).

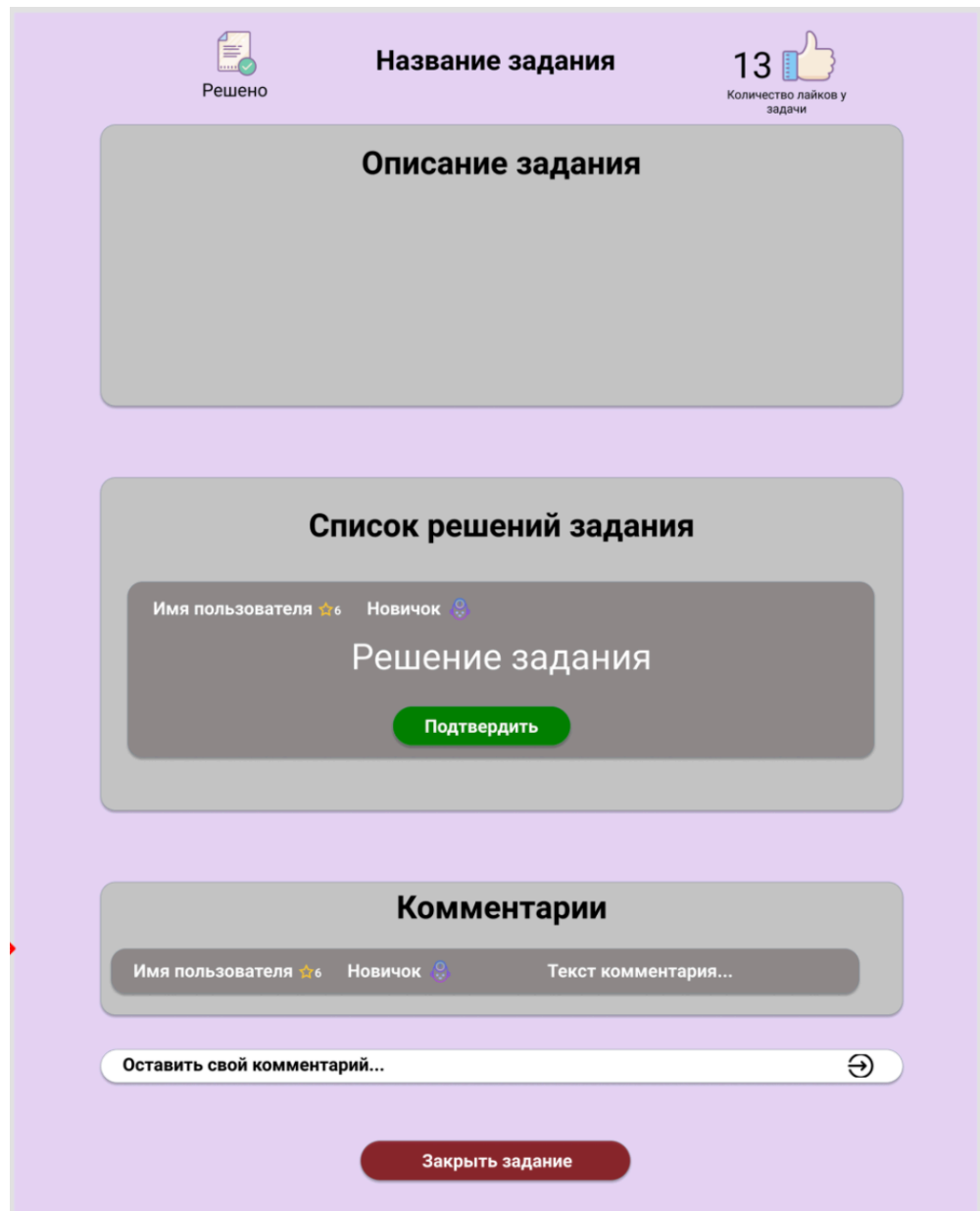


Рисунок 5. Экран конкретного задания пользователя.

6. Экран для решения и просмотра конкретного задания пользователем (рис. 6).

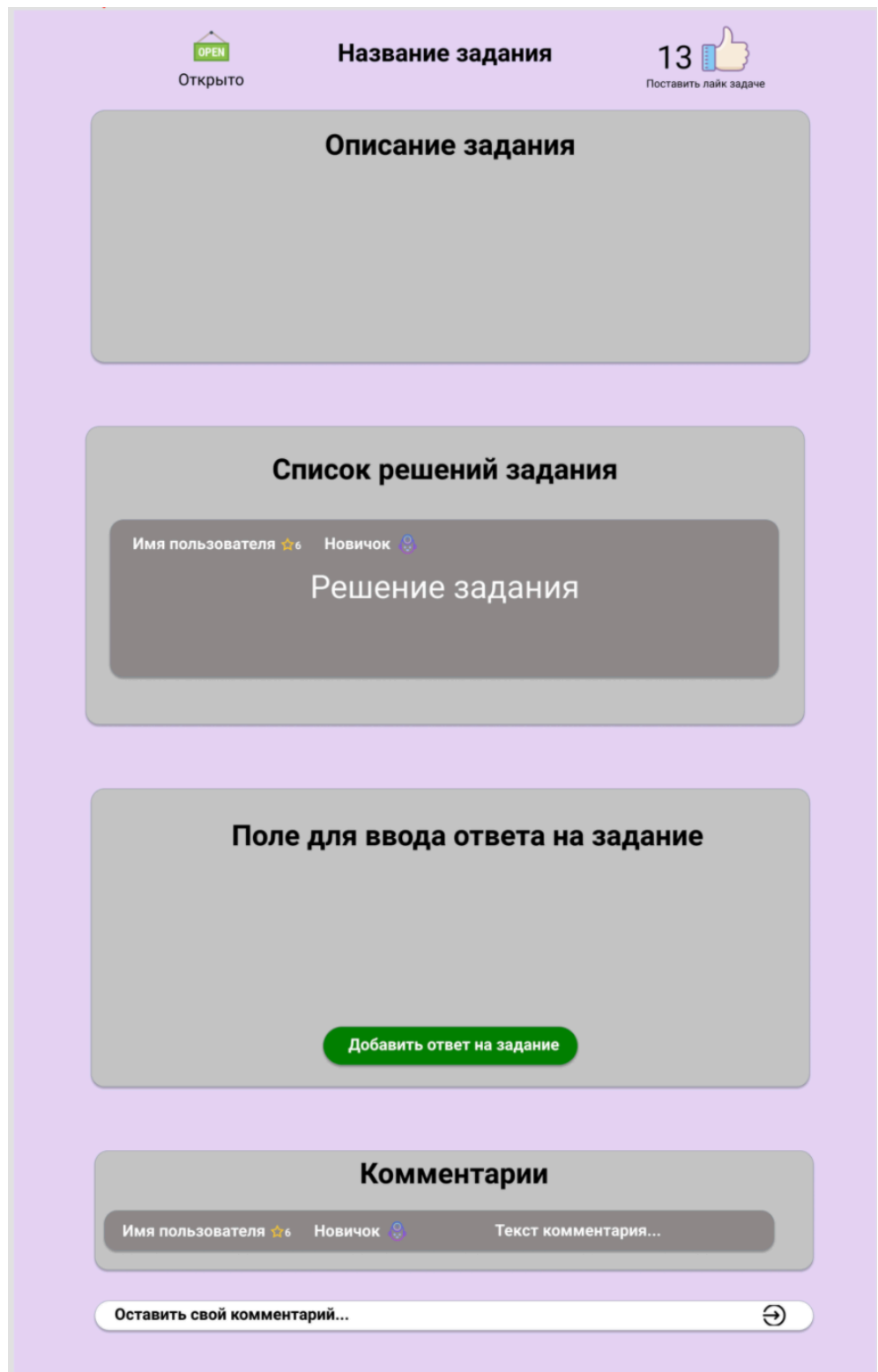


Рисунок 6. Экран для решения и просмотра конкретного задания пользователем.

7. Экран импорта и экспорта заданий (рис. 7).

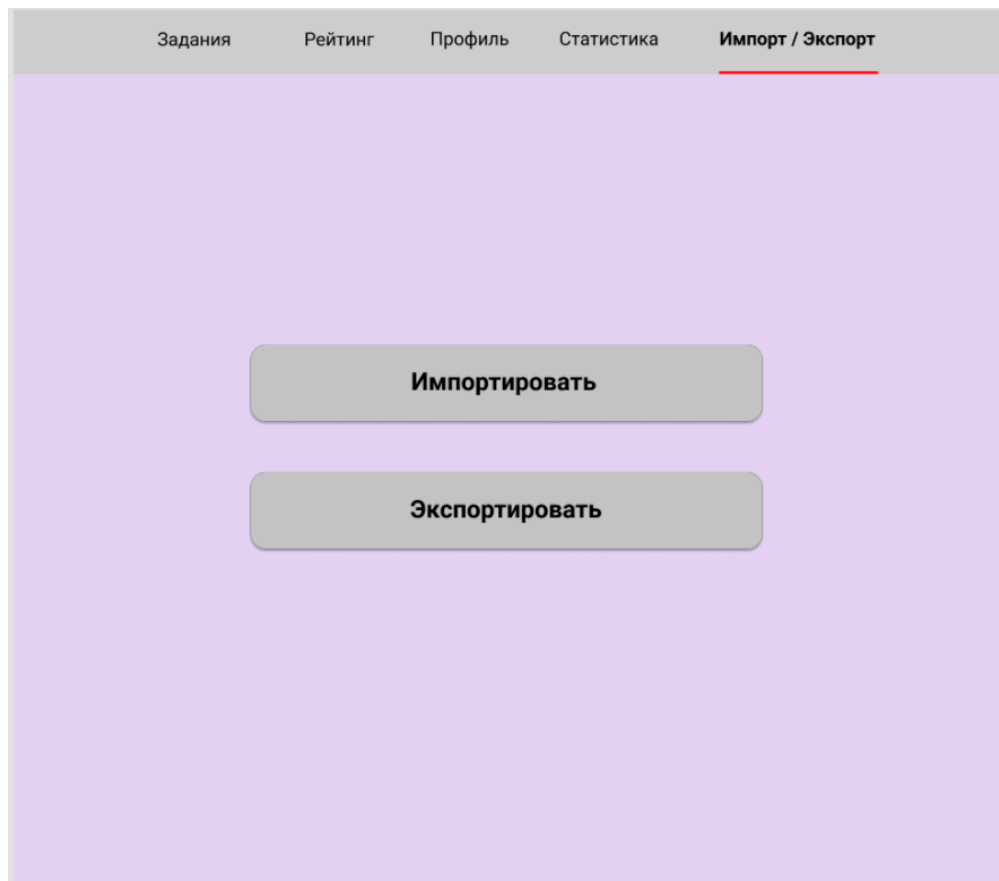


Рисунок 7. Экран импорта и экспорта заданий.

8. Экран для просмотра статистики системы (рис. 8).

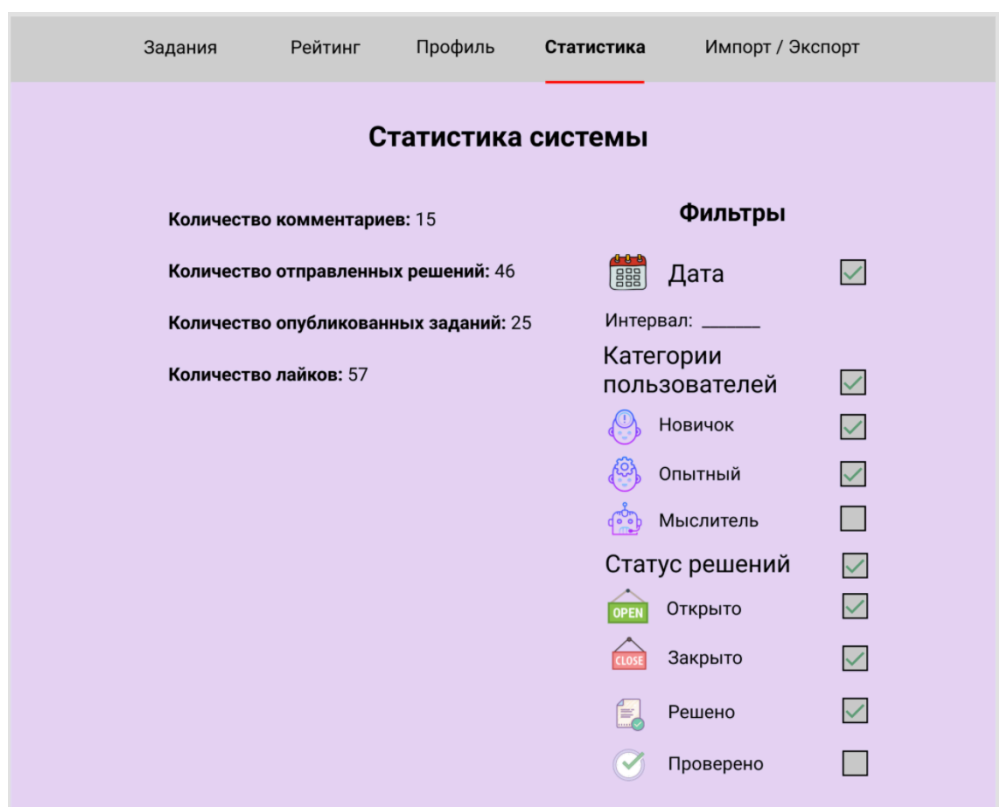


Рисунок 8. Экран для просмотра статистики системы.

9. Экран для просмотра профиля пользователя (рис. 9).

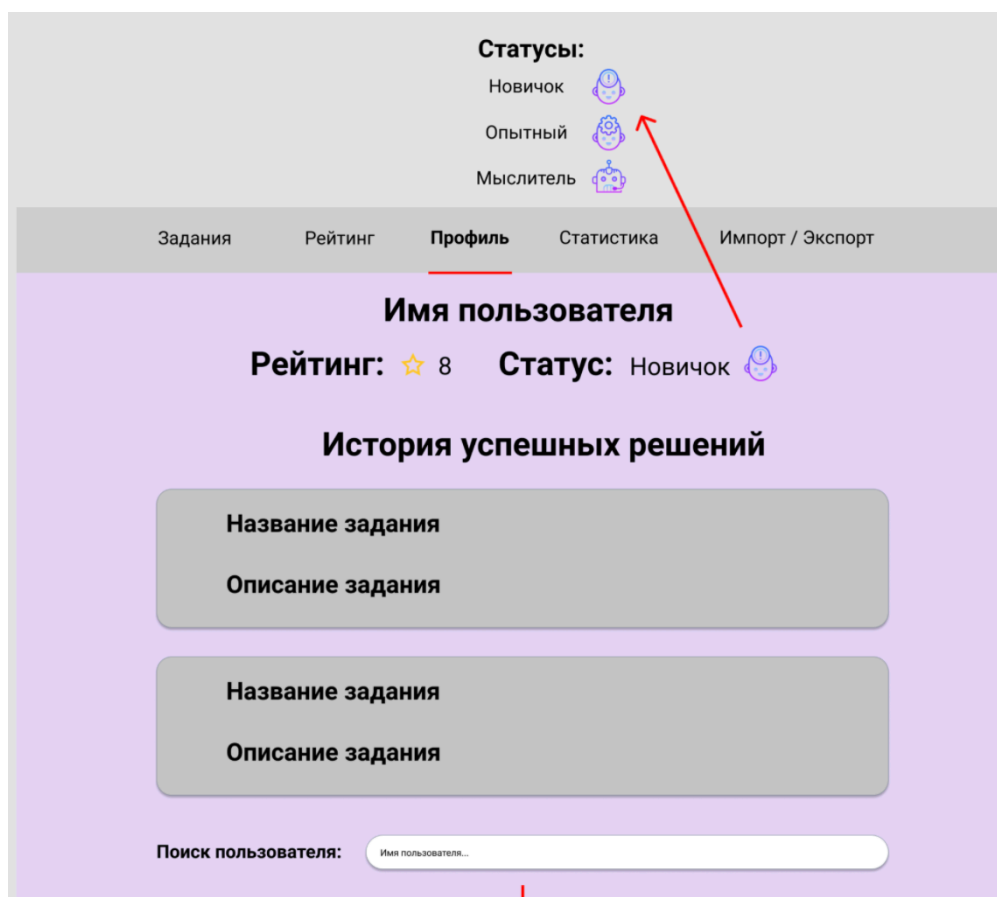


Рисунок 9. Экран для просмотра профиля пользователя.

10. Экран для просмотра рейтинга пользователей (рис. 10).

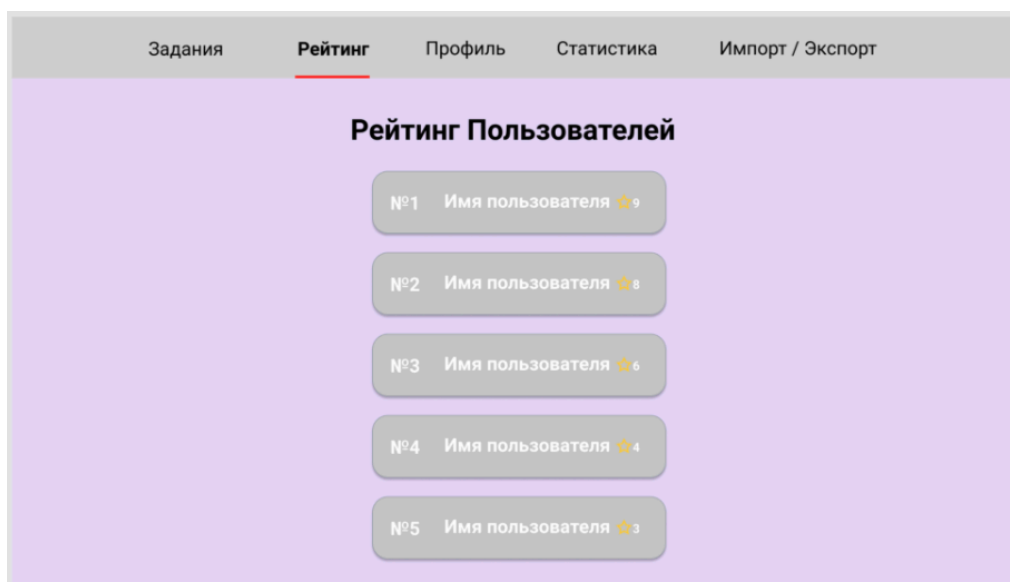
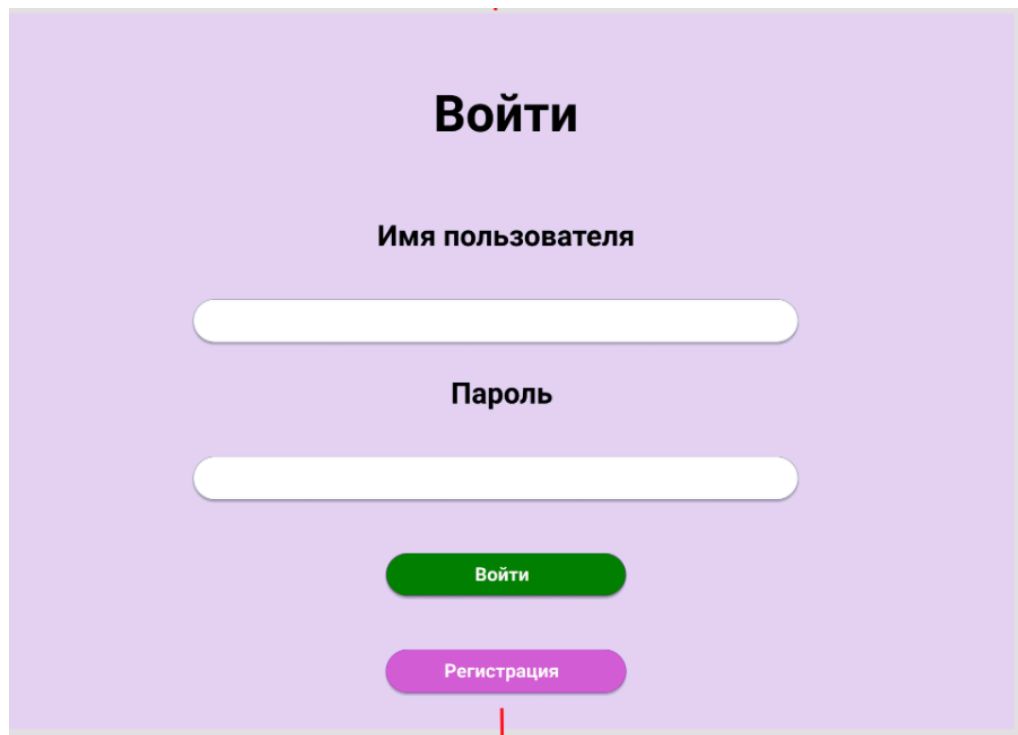


Рисунок 10. Экран для просмотра рейтинга пользователей.

11. Экран для авторизации пользователя (рис. 11).



The login screen features a light purple background. At the top center is the title "Войти" in bold black font. Below it is the label "Имя пользователя" in bold black font, followed by a white rounded rectangular input field. Underneath is the label "Пароль" in bold black font, followed by another white rounded rectangular input field. Below the password field are two buttons: a green rounded rectangle with the text "Войти" and a pink rounded rectangle with the text "Регистрация". A thin red vertical line is positioned at the bottom center of the screen.

Войти

Имя пользователя

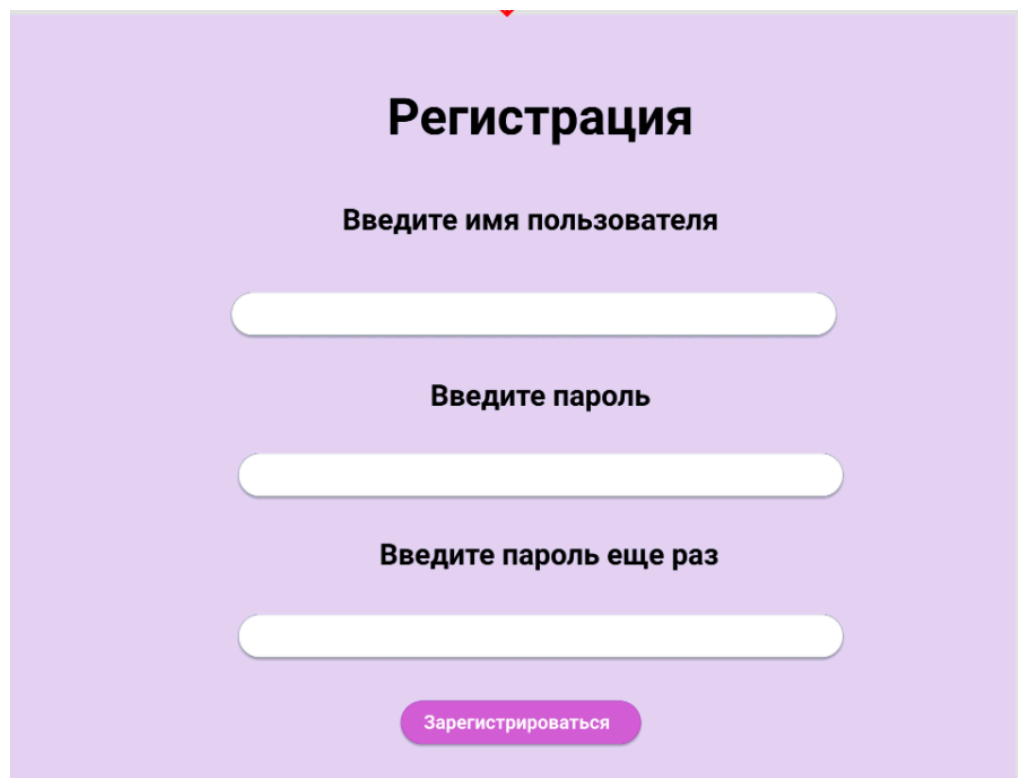
Пароль

Войти

Регистрация

Рисунок 11. Экран для авторизации пользователя.

12. Экран для регистрации пользователя (рис. 12).



The registration screen has a light purple background. At the top center is the title "Регистрация" in bold black font. Below it are three labels in bold black font: "Введите имя пользователя", "Введите пароль", and "Введите пароль еще раз". Each label is followed by a white rounded rectangular input field. At the bottom center is a pink rounded rectangle button with the text "Зарегистрироваться".

Регистрация

Введите имя пользователя

Введите пароль

Введите пароль еще раз

Зарегистрироваться

2.2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ.

Единственная роль в системе – Пользователь.

Сценарий использования “Авторизация”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.

Сценарий использования “Регистрация”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь нажимает кнопку “Регистрация”.
- 3) Пользователь задает поле “Введите имя пользователя”.
- 4) Пользователь задает поле “Введите пароль”.
- 5) Пользователь задает поле “Введите пароль еще раз”.
- 6) Пользователь нажимает кнопку “Зарегистрироваться”.

Сценарий использования “Просмотр заданий”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь просматривает задания вкладки “Все задания” / “Мои решения” / “Мои задания”.
- 6) Пользователь задает параметры фильтрации.
- 7) Пользователь просматривает статус заданий.
- 8) Пользователь просматривает количество “лайков” заданий.
- 9) Пользователь нажимает на конкретное задание.
- 10) Пользователь переходит на “Страница для решения и просмотра задания” / “Страница моего задания”.
- 11) Пользователь просматривает полное описание задания.
- 12) Пользователь просматривает список решений задания, имена пользователей авторов решения и их статус.
- 13) Пользователь просматривает список комментариев, имена пользователей комментаторов и их статус.

Сценарий использования “Поиск заданий”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь вводит в поле поиска название задания.
- 6) Пользователь выбирает искомое задание из выпадающего списка.

Сценарий использования “Добавление своего задания”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь нажимает на кнопку “Добавление задания”.
- 6) Пользователь переходит на страницу добавления своего задания.
- 7) Пользователь вводит название задания в первое поле.
- 8) Пользователь вводит описание задания во второе поле.
- 9) Пользователь нажимает кнопку “Добавить задание”

Сценарий использования “Комментирование и лайк заданий”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь переходит на вкладку “Все задания” / “Мои решения” / “Мои задания”.
- 6) Пользователь нажимает на кнопку “Лайк” конкретного задания.
- 7) Пользователь нажимает на конкретное задание.
- 8) Пользователь в поле “Оставить свой комментарий” вводит текст комментария.
- 9) Пользователь нажимает кнопку “Оставить комментарий”

Сценарий использования “Подтверждение решения моего задания”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь переходит на вкладку “Мои задания”.
- 6) Пользователь нажимает на конкретное задание.
- 7) Пользователь переходит на “Страница моего задания”.
- 8) Пользователь оценивает список предложенных решений и лучшее нажатием на кнопку “Подтвердить”.

Сценарий использования “Закрытие моего задания”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь переходит на вкладку “Мои задания”.
- 6) Пользователь нажимает на конкретное задание.
- 7) Пользователь переходит на “Страница моего задания”.

- 8) Пользователь нажимает на кнопку “Закрытие задания”.

Сценарий использования “Добавление решения к заданию”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь просматривает задания вкладки “Все задания” / “Мои решения”.
- 6) Пользователь нажимает на конкретное задание.
- 7) Пользователь переходит на “Страница для решения и просмотра задания”.
- 8) Пользователь вводит в поле для ввода ответ на задание.
- 9) Пользователь нажимает на кнопку “Добавить ответ на задание”.

Сценарий использования “Просмотр Рейтинга”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь переходит на вкладку “Рейтинг”.
- 6) Пользователь просматривает рейтинг лучших пользователей.

Сценарий использования “Просмотр профиля и поиск пользователей”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь переходит на вкладку “Профиль”.
- 6) Пользователь просматривает свой рейтинг, статус и историю успешных решений.
- 7) Пользователь вводит в поле поиска имя пользователя.
- 8) Пользователь выбирает искомый профиль из выпадающего списка.
- 9) Пользователь просматривает рейтинг, статус и историю успешных решений найденного.

Сценарий использования “Просмотр статистики системы”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь переходит на вкладку “Статистика”.
- 6) Пользователь задает параметры фильтрации.
- 7) Пользователь просматривает статистику системы.

Сценарий использования “Осуществление Импорта / Экспорта”

- 1) Вход на сайт соц. сети для кроудсорсинга.
- 2) Пользователь вводит имя пользователя в первое поле.
- 3) Пользователь вводит пароль во второе поле.
- 4) Пользователь нажимает кнопку “Войти”.
- 5) Пользователь переходит на вкладку “Импорт / Экспорт”.
- 6) Пользователь нажимает на кнопку Импорт для импорта данных.
- 7) Пользователь нажимает на кнопку Экспорта для экспорта данных.

Возможность пользователя добавлять, редактировать, просматривать данные, реализованные с помощью веб-интерфейса.

Для данного решения в большей степени будут использоваться операция чтения, так как приложение часто выводит задачи для просмотра (например, при поиске по фильтрам и тд), а также есть страницы для отображения данных системы – статистика, профиль и поиск по профилям, рейтинг.

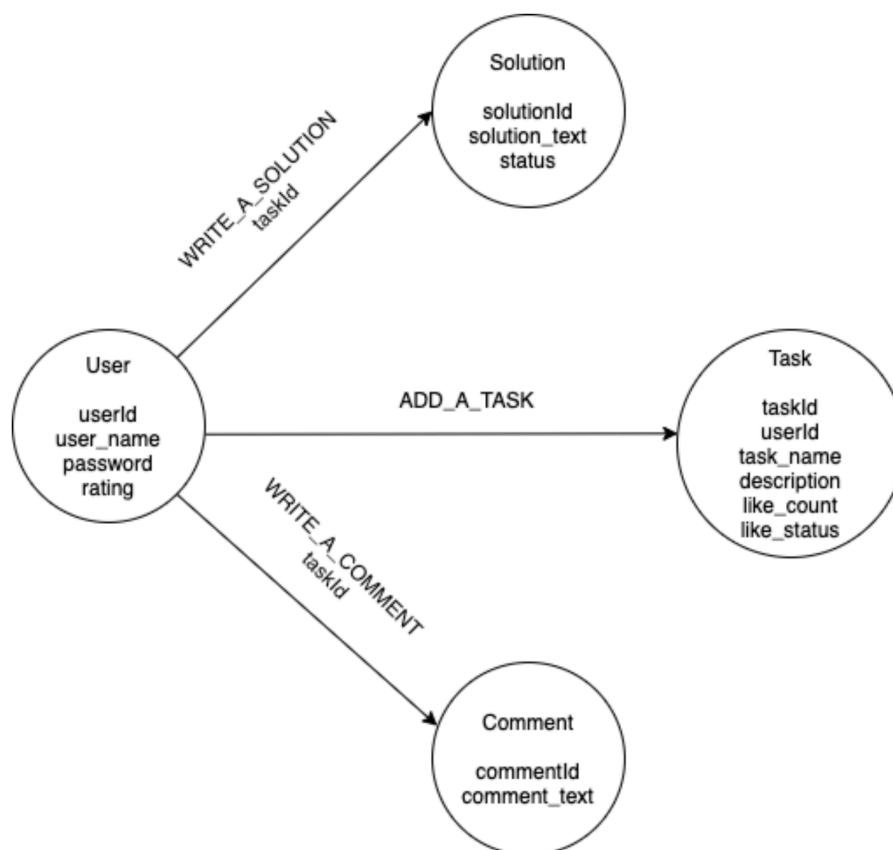
3. МОДЕЛЬ ДАННЫХ

Нереляционная модель данных в данной работе Neo4j.

3.1. NoSQL МОДЕЛЬ ДАННЫХ

3.1.1. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ

Графическое представление модели данных Neo4j:



3.1.2. ОПИСАНИЕ СУЩНОСТЕЙ, КОЛЛЕКЦИЙ, ТИПОВ ДАННЫХ

Модель состоит из 4 сущностей:

1. User – хранит информацию о пользователе. Содержит поля:

- “userId”, int – идентификатор. 4B
- “user_name”, string – имя пользователя. 50*2B
- “password”, string – пароль пользователя. 50*2B
- “rating”, int – рейтинг. 4B

Итого: 208B

2. Task

- “taskId”, int – идентификатор. 4B
- “userId”, int – id пользователя. 4B
- “task_name”, string – название задачи. 50*2B
- “description”, string – описание задачи. 100*2B
- “like_count”, int – количество лайков у задачи. 4B
- “task_status”, int – статус задачи. 4B

Итого: 316B

3. Comment

- “commentId”, int – идентификатор. 4B
- “comment_text” – string. 100 * 2B

Итого: 204B

4. Solution

- “solutionId”, int – идентификатор. 4B
- “solution_text”, string – текст решения. 100*2B
- “status”, int – флаг успешности решения. 4B

Итого: 208B

Существует 3 связи между сущностями:

1) WRITE_A_SOLUTION – User->Solution.

Данный пользователь пишет данные решения.

2) ADD_A_TASK – User->Task.

Пользователь создает данную задачу.

3) WRITE_A_COMMENT – User->Comment.

Пользователь пишет данный комментарий.

3.1.3. РАСЧЕТ ОБЪЕМА

Предположим, что имеется A пользователей, каждый из которых создал по 2 задачи, написал по 3 решения и оставил 5 комментариев.

Чистый объем:

- $A * 204B$ – пользователи
- $A * 2 * 308B$ – задачи
- $A * 5 * 200B$ – комментарии
- $A * 3 * 204B$ – решения

Чистый объем БД: $A * 3052$

Фактический объем:

- $A * 464B$ – пользователи
- $A * 2 * 576B$ – задачи
- $A * 5 * 336B$ – комментарии
- $A * 3 * 336B$ – решения
- $WRITE_A_SOLUTION - A * 3 * 34B$
- $ADD_A_TASK - A * 2 * 34B$
- $WRITE_A_COMMENT - A * 5 * 34B$

Фактический объем БД: $A * 4644$

Избыточность модели: $(A * 4644) / (A * 3052)$

3.1.4. ПРИМЕРЫ ЗАПРОСОВ

1) Добавить пользователя

CREATE (e:User {...})

2) Найти задачу, добавленную пользователем, с id 12

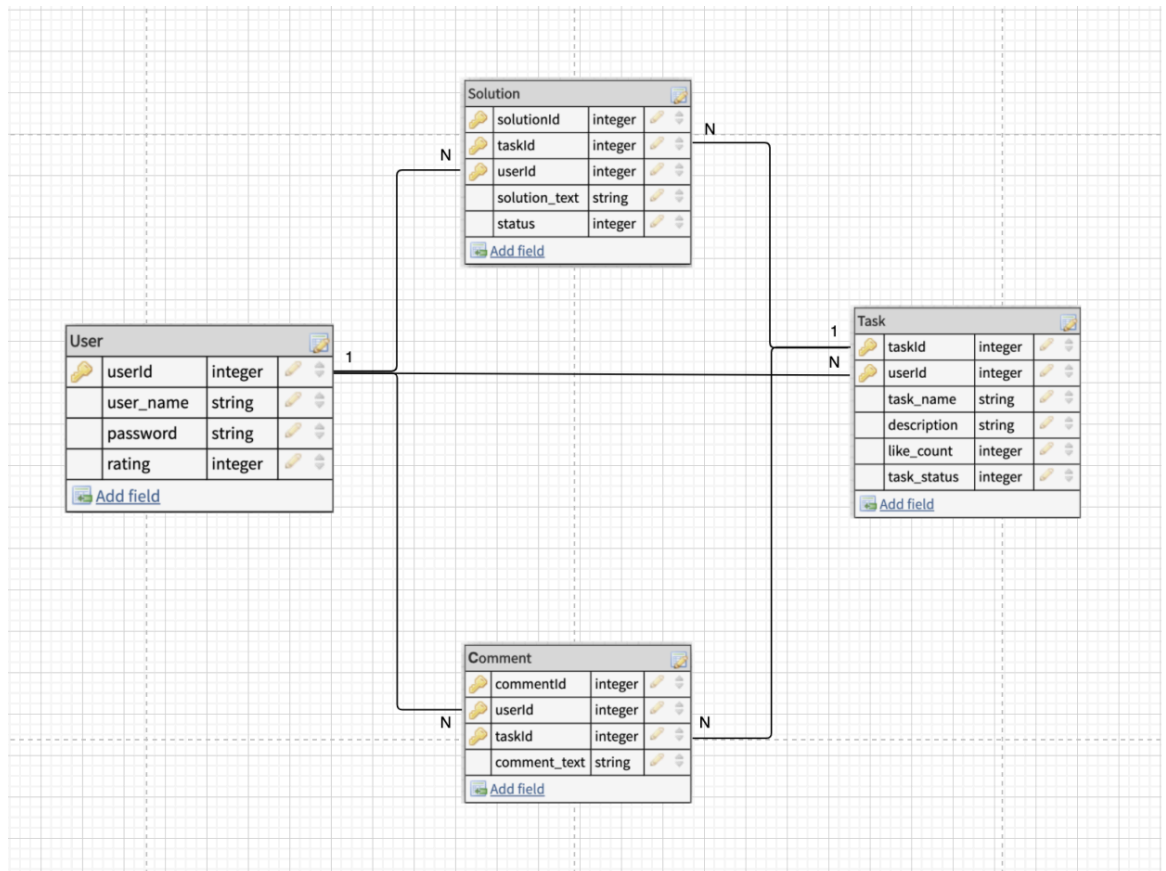
MATCH (k {_id: 12})-[:ADD_A_TASK]-(e)

RETURN e.task_name,

3.2. SQL МОДЕЛЬ ДАННЫХ

3.2.1. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ

Графическое представление модели данных MySQL:



3.2.2. СУЩНОСТИ МОДЕЛИ ДАННЫХ

В качестве реляционной СУБД использована MySQL, в которой созданы 4 таблицы:

1. User – хранит информацию о пользователе. Содержит поля:

- “userid”, int – идентификатор. 4B
- “user_name”, string – имя пользователя. 50*2B
- “password”, string – пароль пользователя. 50*2B
- “rating”, int – рейтинг. 4B

Итого: 208B

2. Task

- “taskId”, int – идентификатор. 4В
- “userId”, int – id пользователя. 4В
- “task_name”, string – название задачи. 50*2В
- “description”, string – описание задачи. 100*2В • “like_count”, int – количество лайков у задачи. 4В
- “task_status”, int – статус задачи. 4В

Итого: 316В

3. Comment

- “commentId”, int – идентификатор. 4В
- “userId”, int – id пользователя автора комментария. 4В
- “taskId”, int – id задачи. 4В
- “comment_text” – string. 100 * 2В

Итого: 212В

4. Solution

- “solutionId”, int – идентификатор. 4В
- “taskId”, int – id задачи. 4В
- “userId”, int – id пользователя автора решения. 4В
- “solution_text”, string – текст решения. 100*2В
- “status”, int – флаг успешности решения. 4В

Итого: 216В

3.2.3. Расчет объема

Предположим, что имеется А пользователей, каждый из которых создал по 2 задачи, написал по 3 решения и оставил 5 комментариев.

Чистый объем:

- $A * 204В$ – пользователи
- $A * 2 * 308В$ – задачи

- $A * 5 * 200B$ – комментарии
- $A * 3 * 204B$ – решения

Чистый объем БД: $A * 2432$

Фактический объем:

- $A * 208B$ – пользователи
- $A * 2 * 316B$ – задачи
- $A * 5 * 212B$ – комментарии
- $A * 3 * 216B$ – решения

Фактический объем БД: $A * 2548$

Избыточность модели: $(A * 2548) / (A * 2432)$

3.2.4. ПРИМЕРЫ ЗАПРОСОВ

1) Добавить пользователя

`INSERT INTO User VALUES(...)`

`INSERT INTO Task VALUES(...)`

2) Найти задачу, добавленную пользователем, с id 12

`SELECT *FROM Task INNER JOIN User ON User.userId = Task.userId AND
User.userId = 12.`

3.3. MONGODB МОДЕЛЬ ДАННЫХ

3.3.1. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ

Графическое представление модели данных mongodb:

```
User
{
  "_id": objectId,
  "user_name": String,
  "tasks": [
    {
      "_id": objectId
    }
  ]
  "solutions": [
    {
      "_id": objectId
    }
  ]
  "comments": [
    {
      "_id": objectId
    }
  ]
}

Task
{
  "_id": objectId,
  "task_name": String,
  "description": String,
  "like_count": Int,
  "task_status": Int,
  "solutions": [
    {
      "_id": objectId
    }
  ]
  "comments": [
    {
      "_id": objectId
    }
  ]
}

Solution
{
  "_id": objectId,
  "solution_text": String,
  "status": Int
}

Comment
{
  "_id": objectId,
  "comment_text": String,
}
```

3.3.2. СУЩНОСТИ МОДЕЛИ ДАННЫХ

Модель состоит из 4 сущностей

1) Пользователь (A – количество пользователей) $62B + 12B * B + 12B * C + 12B * D$

- “_id”: objectId – идентификатор, 12B
- “user_name”: String – имя пользователя, $50 * 1B$
- “tasks”: objectId[] – список задач, количество задач * 12B
- “solutions”: objectId[] – список решений, количество решений * 12B
- “comments”: objectId[] – список комментариев, количество комментариев * 12B

2) Задача (B – количество задач), $124B + 12B * C + 12B * D$

- “_id”: objectId – идентификатор, 12B
- “task_name”: String – название задачи, $50 * 1B$
- “description”: String – описание задачи, $50 * 1B$
- “like_count”: Int – количество лайков, $8 * 1B$
- “task_status”: Int – статус задачи, $4 * 1B$

- “solutions”: objectId[] – список решений, количество решений * 12В
- “comments”: objectId[] – список комментариев, количество комментариев * 12В

3) Решение (С – количество решений), 66В

- “_id”: objectId – идентификатор, 12В
- “solution_text ”: String – текст решения задачи, 50 * 1В
- “status”: Int – статус решения, 4 * 1В

4) Комментарий (D – количество комментариев), 62В

- “_id”: objectId – идентификатор, 12В
- “comment_text ”: String – текст решения задачи, 50 * 1В

3.3.3. РАСЧЕТ ОБЪЕМА

Предположим, что имеется А пользователей, каждый из которых создал по 2 задачи, написал по 3 решения и оставил 5 комментариев.

Чистый объем:

$$62В * А + 2 * 124В + 3 * 66В + 5 * 62В = 818В * А$$

Фактический объем:

$$62В * А + 12В * 2 * (124В + 12В * 66В + 12В * 62В) + 12В * 66В * 3 + 12В * 12В * 5 = А * 42998$$

$$\text{Избыточность модели: } (А * 42998) / (818 * А)$$

3.3.4. ПРИМЕРЫ ЗАПРОСОВ

1) Добавление задачи

```
db.tasks.insertOne({
  _id:1,
  "task_name": "TaskOne",
  "description": "This task about One",
  "like_count": 34,
  "task_status": 1,
  "solutions": [
    {
      "_id": 1
    },
    {
      "_id": 2
    }
  ]
  "comments": [
    {
      "_id": 3
    },
    {
      "_id": 6
    }
  ]
})
```

2) Поиск задачи

```
db.tasks.find({_id:1})
```

3.4 СРАВНЕНИЕ NEO4J, MONGODB И SQL МОДЕЛИ ДАННЫХ

SQL требует меньше памяти: $A * 2548$ против $A * 4644$ (Neo4j) и $A * 42998$ (mongodb), однако написание SQL запросов занимает больше времени. Запросы на SQL – языке являются более громоздкими, чем на языке Neo4j Cypher. Реализация модели данных в MySQL потребовало бы создать большее количество связей. Исходя из вышеперечисленного, можно сказать, что для рассматриваемой задачи Neo4j подходит в большей степени. Он побеждает mongodb, так как требует меньше памяти.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. КРАТКОЕ ОПИСАНИЕ

Разработано веб-приложение, в котором реализованы все заявленные варианты использования. Принцип работы: по запросу пользователя ему предоставляется соответствующая html-страница, отрендеренная из pug-шаблона, а наполнение самой страницы посылается сервером в виде json, в соответствии с которым на стороне клиента уже формируется наполнение определенного раздела.

4.2. СХЕМА ЭКРАНОВ ПРИЛОЖЕНИЯ И/ИЛИ СХЕМА ИНТЕРФЕЙСА КОМАНДНОЙ СТРОКИ

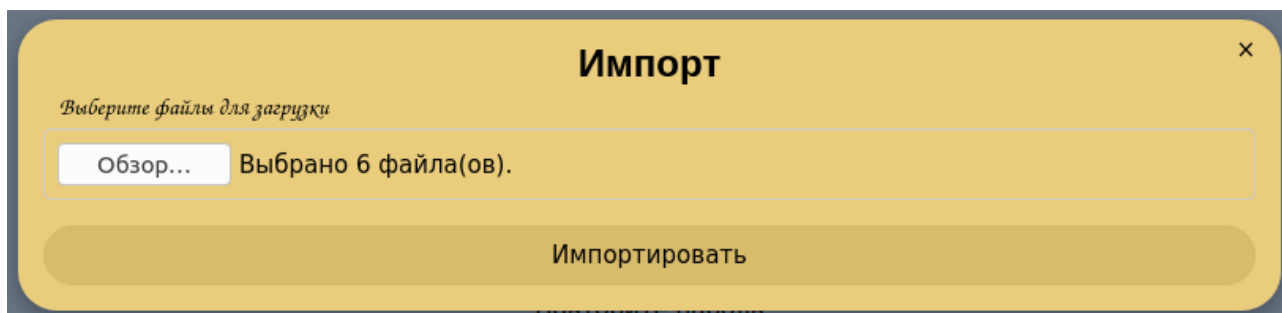
Страница авторизации пользователя:

The screenshot shows a login page with a light blue background. At the top, the title "Войти" is centered. Below it, the label "Имя пользователя" is followed by a text input field containing "Anton". Underneath, the label "Пароль" is followed by a password input field with masked characters ".....". Below the password field are three buttons: a green "Войти" button, an orange "Регистрация" button, and a yellow "Импорт БД" button.

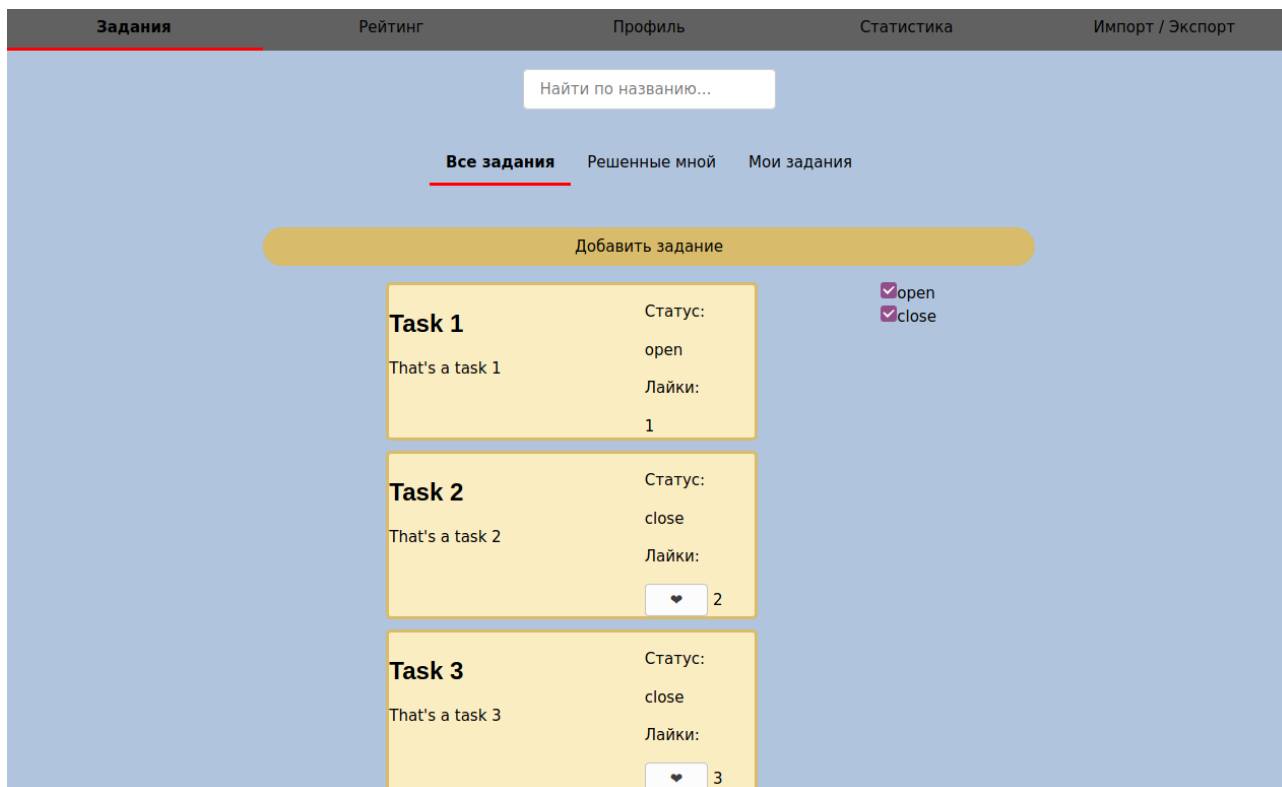
Страница регистрации пользователя:

The screenshot shows a registration page with a light blue background. At the top, the title "Регистрация" is centered. Below it, the label "Введите имя пользователя" is followed by a text input field. Underneath, the label "Пароль" is followed by a password input field. Below the password field, the label "Повторите пароль" is followed by another password input field. Below these fields are three buttons: a green "Зарегистрироваться" button, an orange "Авторизация" button, and a yellow "Импорт БД" button.

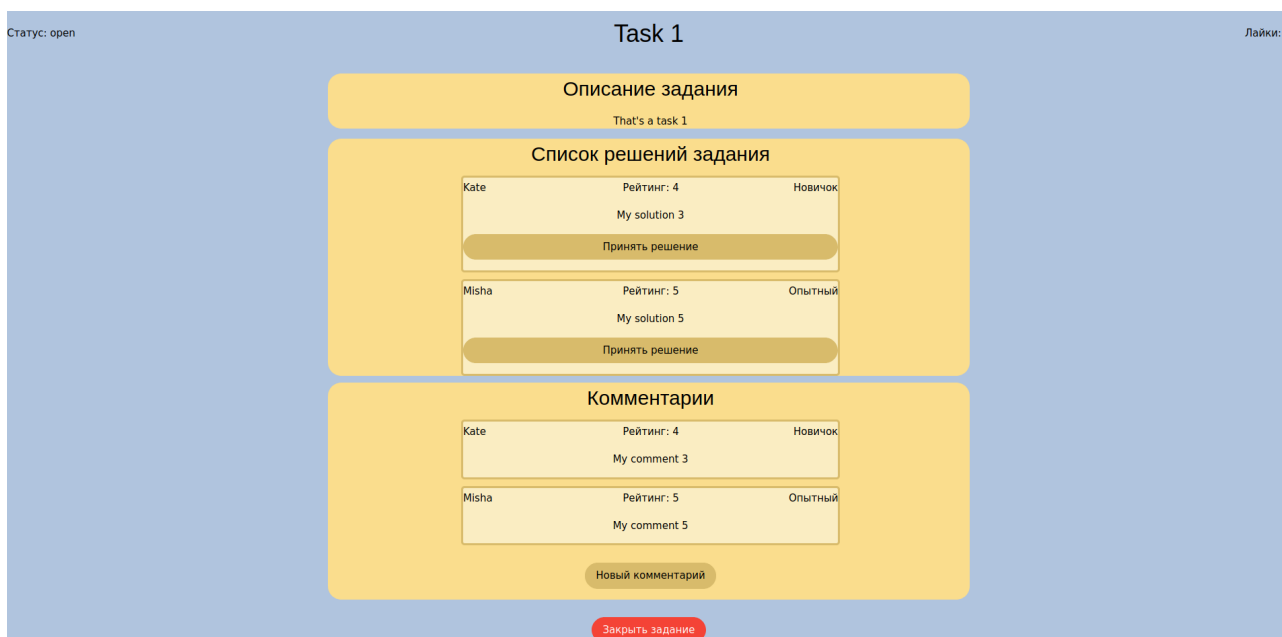
Окно импорта БД:



Страница с заданиями (возможна фильтрация и поиск):



Страница своего задания:



Страница чужого задания:

Статус: close

Task 2

Лайки: 2

Описание задания

That's a task 2

Список решений задания

Anton	Рейтинг: 3	Новичок
My solution 1		
Misha	Рейтинг: 5	Опытный
My solution 6		

Новое решение

Комментарии

Anton	Рейтинг: 3	Новичок
My comment 1		
Misha	Рейтинг: 5	Опытный
My comment 6		

Новый комментарий

Окно добавления задания:

Новое задание

Название

Описание

Добавить задание

Окно добавления нового решения:

Новое решение

Решение

Добавить решение

Окно добавления нового комментария:

Новый комментарий

Комментарий

Добавить комментарий

Страница рейтинга пользователей:

Задания	Рейтинг	Профиль	Статистика	Импорт / Экспорт
Рейтинг пользователей				
№0	Misha	5		
№1	Kate	4		
№2	Anton	3		

Страница просмотра своего профиля:

Задания	Рейтинг	Профиль	Статистика	Импорт / Экспорт
---------	---------	---------	------------	------------------

Anton

Рейтинг: 3 Статус: Новичок

Выйти

История успешных решений:

Task 2
That's a task 2

Статус:
close
Лайки:
2

Найти по имени...

Страница просмотра чужого профиля:

Задания	Рейтинг	Профиль	Статистика	Импорт / Экспорт
---------	---------	---------	------------	------------------

Misha

Рейтинг: 5 Статус: Опытный

Выйти

История успешных решений:

Task 1
That's a task 1

Статус:
open
Лайки:
1

Найти по имени...

Страница со статистикой, формируемой по отфильтрованным данным:

Задания	Рейтинг	Профиль	Статистика	Импорт / Экспорт
---------	---------	---------	------------	------------------

Статистика системы

Количество комментариев: 6
Количество отправленных решений: 6
Количество опубликованных заданий: 3
Количество лайков: 6

☒ Категории пользователей:
☒ Новичок
☒ Опытный
☒ Мыслитель

☒ Статус заданий:
☒ open
☒ close

Страница для выбора функции импорта или экспорта БД:

Задания	Рейтинг	Профиль	Статистика	Импорт / Экспорт
				Импортировать
				Экспортировать

Схему переходов предоставлять излишне, она уже есть для шага «Use Case» и модели данных (см. docs/ui/layout.png в репозитории проекта)

4.3. ИСПОЛЬЗОВАННЫЕ ТЕХНОЛОГИИ

Neo4j, express.js, html (from pug), CSS, W3CSS, JavaScript, jquery

4.4. ССЫЛКИ НА ПРИЛОЖЕНИЕ

Ссылка на github: <https://github.com/moevm/nosql2h20-crowd-neo4j>

5. ВЫВОДЫ

5.1. ДОСТИГНУТЫЕ РЕЗУЛЬТАТЫ

В ходе работы было разработано приложение с использованием нереляционной базы данных –Neo4j, с возможностями создания, решения, комментирования задач, а также добавления статуса задач и пользовательским решениям. Задачи можно просматривать, сортируя по фильтрам, а базу данных в виде csv-файлов импортировать и экспортировать. В приложении есть возможность просмотра статистики и рейтинга пользователей.

5.2. НЕДОСТАТКИ И ПУТИ ДЛЯ УЛУЧШЕНИЯ ПОЛУЧЕННОГО РЕШЕНИЯ

Модель данных, а с ней и приложение, требует доработки: лайк чужому заданию можно ставить любое число раз, потому что он не привязывается к конкретному человеку и невозможно отслеживать, кто уже ставил, а кто нет.

Также важно добавить больше обработчиков исключений: в текущем виде программа может упасть или выдать некорректный результат,

5.3. БУДУЩЕЕ РАЗВИТИЕ РЕШЕНИЯ

Возможно улучшение верстки интерфейса.

Возможно добавление других, кроме закрытого и открытого, статуса задач, это же актуально и для решения.

Возможно более подробное предоставление статистики, например и использованием графиков.

6. ПРИЛОЖЕНИЯ

6.1. ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

- 1) Клонировать себе на устройство репозиторий
- 2) Перейти в директорию проекта
- 3) Активировать скрипт создания папки для импорта командой `bash ./create_folder.sh`
- 4) Запустить `docker-compose` командой `docker-compose up --build`
- 3) Открыть приложение в браузере по адресу `localhost:3000`

ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА

1. Документация MongoDB: <https://docs.mongodb.com/manual/>
2. Документация к Neo4j: <https://neo4j.com/docs/>
3. Github-репозиторий: <https://github.com/moevm/nosql2h20-crowd-neo4j>