

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

**ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**  
**по дисциплине «Нереляционные базы данных»**

Тема: приложение для краутфайдинга

Студент гр. 7304

Ажель И.

Студент гр. 7304

Комаров А.О.

Студент гр. 7304

Петруненко Д.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

2020

## **ЗАДАНИЕ**

### **НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**

Студенты

Ажель И.

Комаров А.О.

Петруненко Д.А.

группа 7304

Тема работы: приложение для краутфайдинга.

Исходные данные:

Создание приложения, в функциональность которого входят добавление заданий для разметки изображений и текста.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 25 страниц.

Дата выдачи задания: 18.09.2020

Дата сдачи реферата:

Дата защиты реферата:

Студент гр. 7304

Ажель И.

Студент гр. 7304

Комаров А.О.

Студент гр. 7304

Петруненко Д.А.

Преподаватель

Заславский М.М.

## **АННОТАЦИЯ**

В рамках данного курса предполагалось какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания для краутфайдинга.

## СОДЕРЖАНИЕ

|  |    |
|--|----|
| 1. ВВЕДЕНИЕ.....   | 6  |
| 2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ .....                               | 7  |
| 3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ.....   | 8  |
| 4. МОДЕЛЬ ДАННЫХ ..... <b>Ошибка! Закладка не определена.</b>            |    |
| 5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ <b>Ошибка!       Закладка не определена.</b> |    |
| 6. ВЫВОДЫ.....   | 31 |
| 7. ПРИЛОЖЕНИЯ.....   | 32 |
| 8. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....                                | 33 |

## **I. ВВЕДЕНИЕ**

Целью работы является создание приложения, в функциональность которого входят добавление заданий для разметки изображений и текста. Выбранный нами стек технологий включает в себя Vue.js[1], CSS, Flask, Mongo[2].

## **II. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ**

Требуется разработать user-friendly приложение, в функциональность которого будут входить: страница со списком заданий, содержащая описание задания, статистика, фильтрация и сортировка по работодателям, добавление заданий, импорт и экспорт данных БД. В качестве системы управления базами данных использовать MongoDB [2].

### III. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

#### 3.1 Макеты UI

##### 1. Страница заказов.

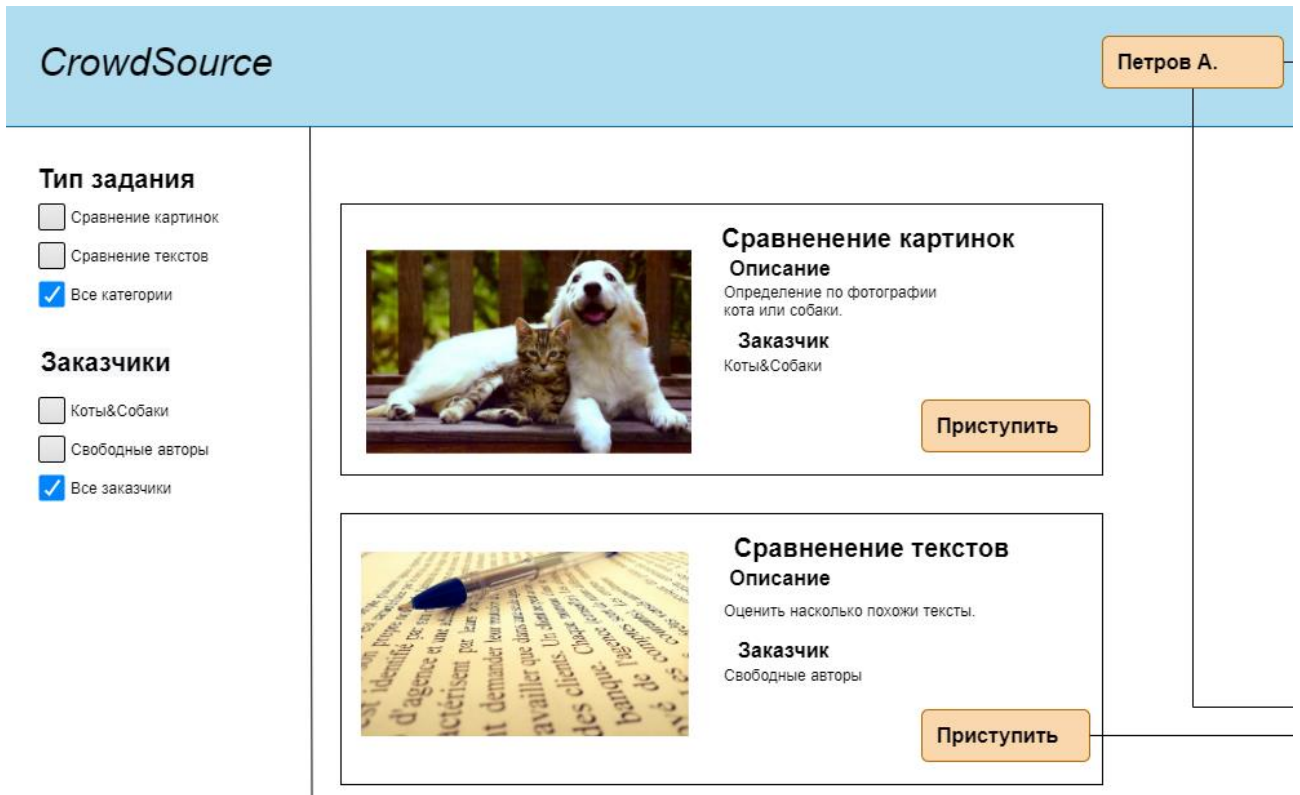


Рисунок 3.1. — Просмотр страницы заказов.

##### 2. Экспорт/импорт данных.

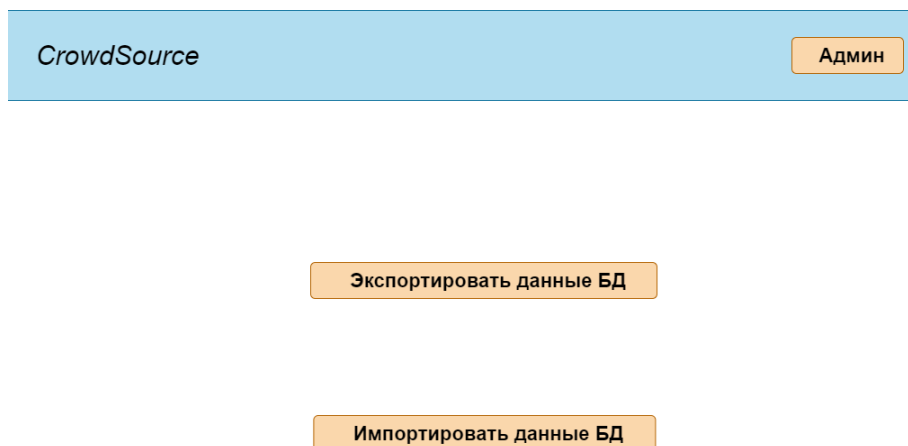
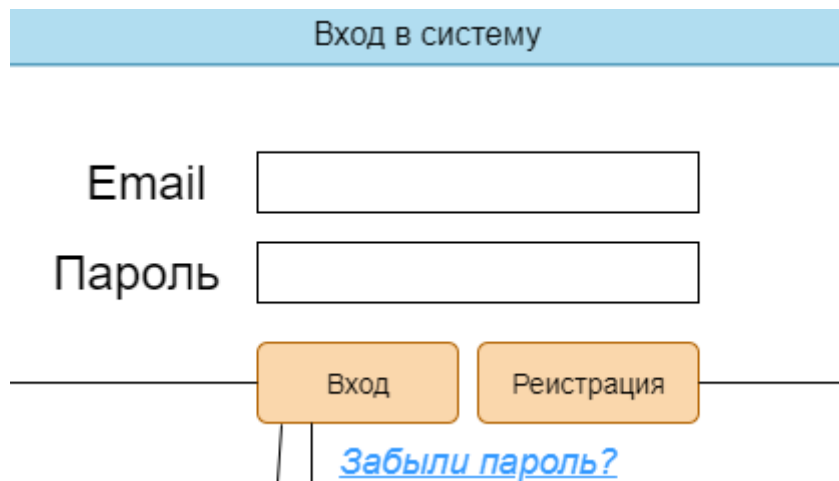


Рисунок 3.2. — Экспорт/импорт данных.

##### 3. Вход в систему.





The diagram shows a login form titled "Вход в систему" (Login to the system). It includes two input fields: "Email" and "Пароль" (Password). Below the password field are two buttons: "Вход" (Login) and "Регистрация" (Registration). A link labeled "Забыли пароль?" (Forgot password?) is positioned below the "Вход" button. A horizontal line is drawn below the "Вход" and "Регистрация" buttons, with a vertical line extending downwards from the "Вход" button.

Вход в систему

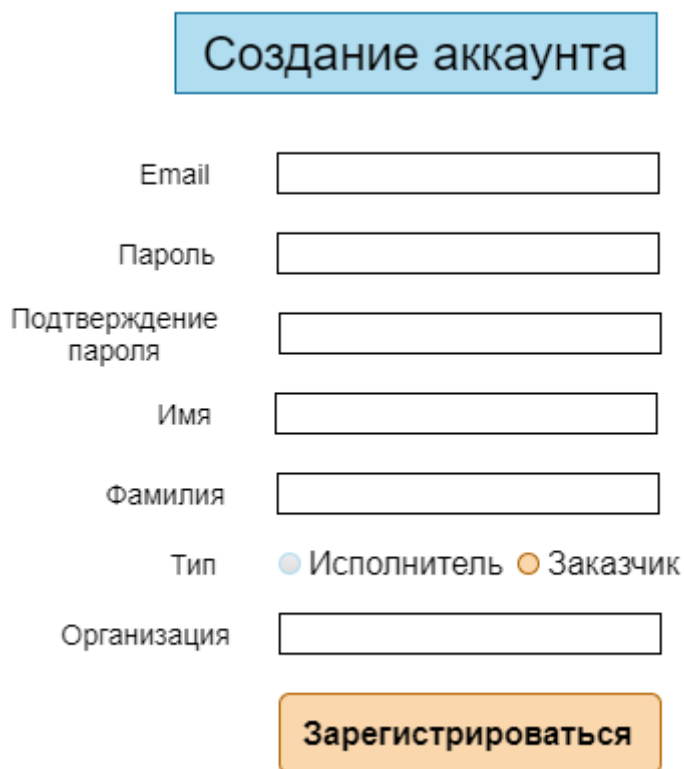
Email

Пароль

[Забыли пароль?](#)

Рисунок 3.3. — Форма входа в систему.

#### 4. Создание аккаунта.



The diagram shows an account creation form titled "Создание аккаунта" (Create account). It includes several input fields: "Email", "Пароль" (Password), "Подтверждение пароля" (Confirm password), "Имя" (Name), "Фамилия" (Surname), and "Организация" (Organization). There is also a "Тип" (Type) field with two radio button options: "Исполнитель" (Executor) and "Заказчик" (Client). A "Зарегистрироваться" (Register) button is located at the bottom of the form.

Создание аккаунта

Email

Пароль

Подтверждение пароля

Имя

Фамилия

Тип ☐ Исполнитель ☐ Заказчик

Организация

Рисунок 3.4. — Создание аккаунта.

5. Добавление заказа.

Добавление заказа

Название заказа

Собаки

Описание

Введите описание заказа

Данные для заказа

Введите варианты ответов:

Фотоданные

Текстовые данные

| Варианты ответов(через запятую) | Описание                              | Действие                            |
|---------------------------------|---------------------------------------|-------------------------------------|
| Да, Нет                         | Считаете ли вы собак привлекательными | <input checked="" type="checkbox"/> |

Создать

Закрыть

Рисунок 3.5 — Добавление заказа.

6. Ввод данных изображения.

Фотоданные

Текстовые данные

Ввести вручную

Добавить строку

| Варианты ответов(через запятую) | Фотография   |
|---------------------------------|--|
| Собака, кошка                   | <div><div></div><div><input checked="" type="checkbox"/></div></div> |

Рисунок 3.6 — Ввод данных изображения.

7. Ввод текстовых данных.

Данные для заказа

Фотоданные

Текстовые данные

Ввести вручную

| Варианты ответов(через запятую) | Описание                              | Действие                            |
|---------------------------------|---------------------------------------|-------------------------------------|
| Да, Нет                         | Считаете ли вы собак привлекательными | <input checked="" type="checkbox"/> |

Рисунок 3.7 — Текстовые данные.

8. Разметка фотоданных.

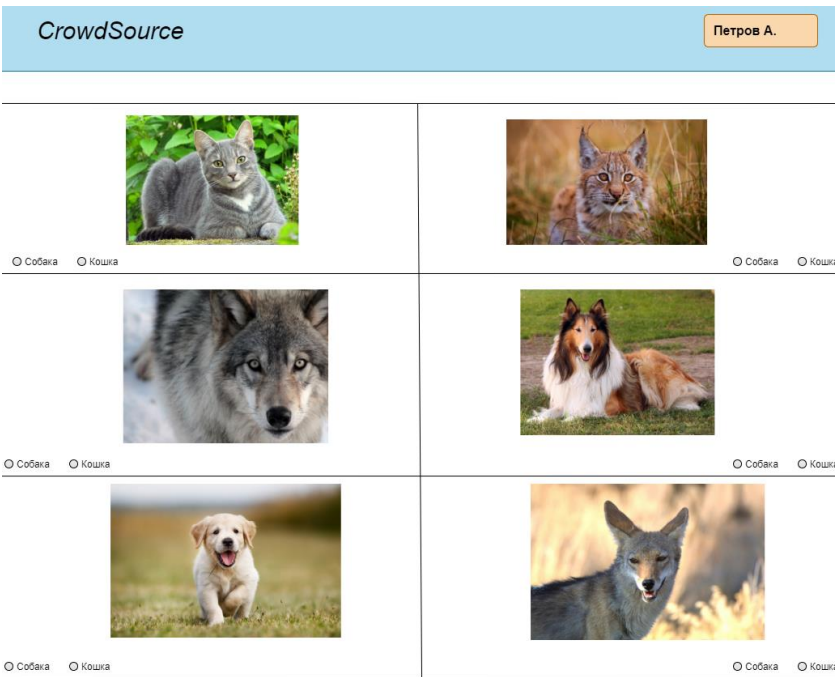


Рисунок 3.8 — Разметка фотоданных.

9. Личный кабинет.

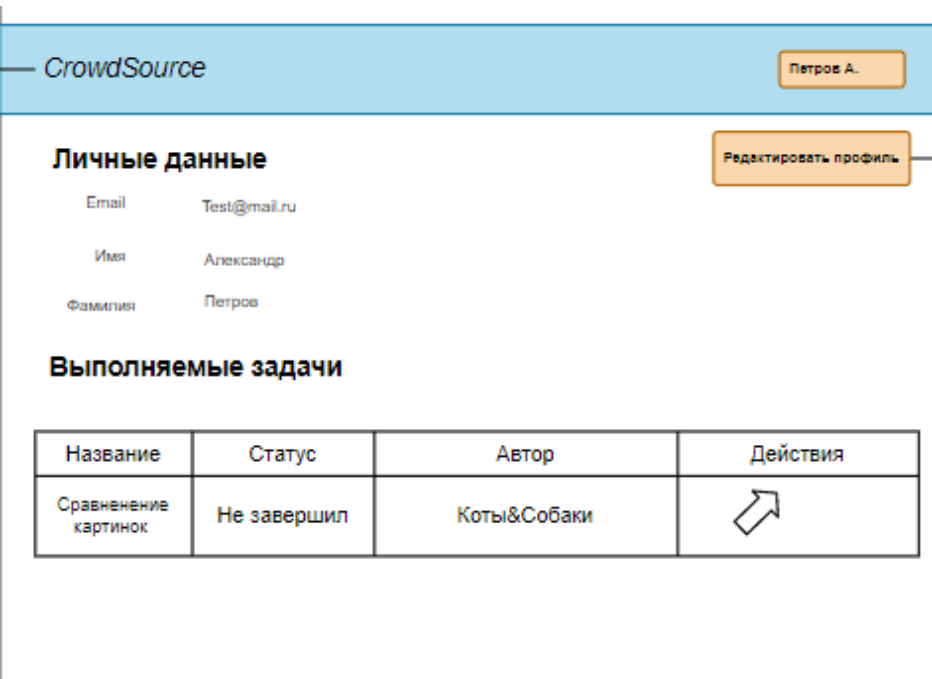


Рисунок 3.9 — Статус заказа.

### Настройка профиля.

## Настройка профиля

Имя

Фамилия

Изменить

Закрыть

Рисунок 3.9 — Настройка профиля.

### 10.Статистика заказа

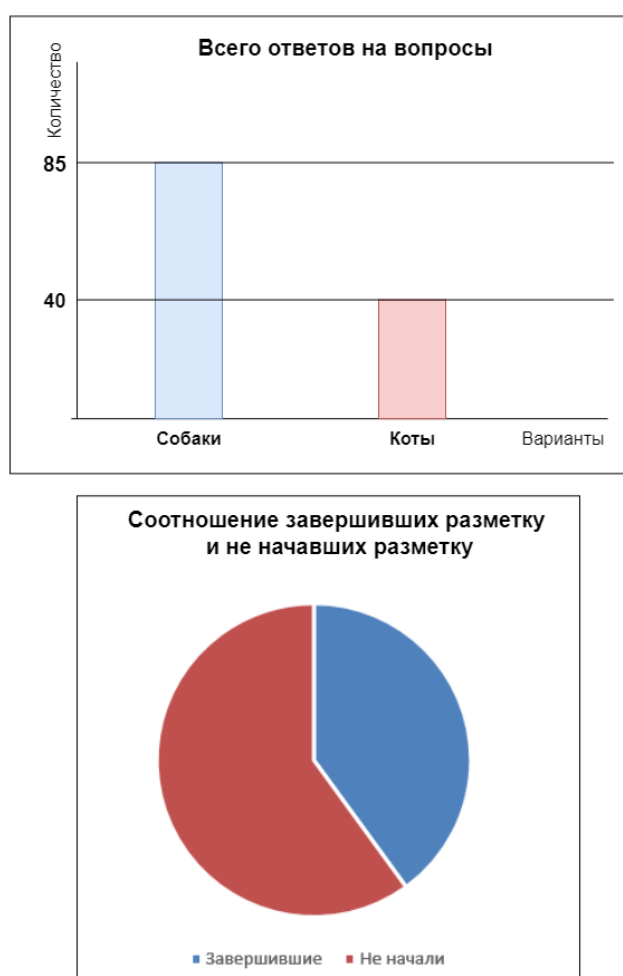


Рисунок 3.10 — Статистика.

### **3.2 Сценарии использования.**

#### **Сценарий – «Просмотр заказов».**

Действующее лицо: Исполнитель, заказчик

Основной сценарий:

1. Юзер открывает главную страницу приложения.
2. Юзер авторизируется.
3. Юзер открывает вкладку заказов

Альтернативный сценарий:

1. Пользователь получает заказы через API.

#### **Сценарий – «Просмотр конкретного заказа».**

Действующее лицо: Исполнитель, заказчик

Основной сценарий:

1. Пользователь открывает главную страницу приложения.
2. Пользователь просматривает заказы.
3. Пользователь нажимает на интересующий заказ.

Результат:

1. Осуществляется переход на страницу с информацией о заказе.

Альтернативный сценарий:

1. Пользователь получает информацию о заказе через API.

#### **Сценарий – «Удаление заказа».**

Действующее лицо: Исполнитель, заказчик

Предусловие:

1. Пользователь находится на главной странице приложения.

Основной сценарий:

1. Пользователь находит нужный ему заказ.
2. Пользователь нажимает на кнопку «Delete» в строке нужного ему заказа.
3. Открывается модальное окно с подтверждением удаления заказа

4. Пользователь нажимает «Yes».

Альтернативный сценарий:

1. Пользователь в модальном окне нажимает кнопку «No» и отказывается от удаления.

2. Пользователь удаляет запись через API.

Результат:

1. Из таблицы пропала запись, для которой была нажата кнопка «Delete».

### **Сценарий – «Добавление нового заказа».**

Действующее лицо: Исполнитель, заказчик

Предусловие:

1. Пользователь находится на главной странице приложения.

Основной сценарий:

1. В открывшемся модальном окне пользователь вводит данные:

- а. Название заказа
- б. Текстовые или данные изображений
- с. Данные разметки

2. Пользователь нажимает кнопку «SUBMIT».

Альтернативный сценарий:

1. Пользователь закрывает модальное окно без нажатия на кнопку «SUBMIT».

2. Пользователь вводит данные не во все поля модального окна.

3. Пользователь добавляет заказ через API.

Результат:

1. В таблице появился новый заказ

### **Сценарий – «Установка фильтра».**

Действующее лицо: Пользователь

Предусловие:

1. Пользователь находится на главной странице приложения.

Основной сценарий:

1. Пользователь нажимает на кнопку «FILTER»

2. В открывшемся модальном окне в разделе «FILTER BY» пользователь указывает данные в тех полях, по которым хочет произвести фильтрацию:

- а. По заказчикам
- б. По заказчикам

5. Пользователь нажимает кнопку «PERFORM».

Альтернативный сценарий:

- 1. Пользователь закрывает модальное окно без нажатия на кнопку «PERFORM».
- 2. Пользователь выполняет фильтрацию через API.

Результат:

- 1. В таблице произвелась фильтрация по указанным значениями.

### **Сценарий – «Загрузка бэкапа данных».**

Действующее лицо: Заказчик

Предусловие:

- 1. Пользователь находится на главной странице приложения.

Основной сценарий:

- 1. Пользователь нажимает на кнопку «Выгрузка данных».
- 2. В открывшемся модальном окне с помощью кнопки «выгрузить» пользователь выбирает в файловой системе нужный файл с данными.
- 3. Пользователь нажимает на кнопку «импорт»

Альтернативный сценарий:

- 1. Пользователь закрывает модальное окно без нажатия на кнопку «импорт».
- 2. Пользователь нажимает на кнопку «экспорт
- 3. Пользователь загружает пустой файл.
- 4. Пользователь загружает файл, данные в котором не содержат названия полей из коллекции в бд.
- 5. Пользователь выполняет загрузку бэкапа через API.

Результат:

- 1. Заказ теперь содержит данные из загруженного файла.

### **Сценарий – «Просмотр статистики».**

Действующее лицо: Заказчик

Предусловие:

1. Пользователь находится на главной странице приложения и переходит на страницу «statistics» с помощью кнопки «STATISTICS».
2. Пользователь находится на странице «statistics».

Основной сценарий:

1. Пользователь нажимает кнопку статистика.

Альтернативный сценарий:

1. Пользователь не нажимает на кнопку статистика.
2. Пользователь выполняет получение статистики по кораблям через API.

Результат:

1. На странице появляется статистика по заказу.



## **1. Сценарий использования – «Авторизация»**

### **Основной сценарий – «Авторизация»**

**Действующее лицо: Неавторизованный пользователь**

- Пользователь открывает наше приложение;
- Пользователь открывает окно авторизации;
  1. В данном окне можно авторизоваться, введя e-mail и пароль;
  2. Выйти из учетной записи, если это необходимо и пользователь уже авторизован;
  3. Зарегистрироваться в новом окне, заполнив данные о себе, придумав пароль и выбрав роль заказчик или исполнитель. После этого нажимает кнопку «Зарегистрироваться»;

### **Альтернативный сценарий.**

**Действующее лицо: Неавторизованный пользователь**

- Пользователь запускает приложение;
- Пользователь без авторизации нажимает кнопку “вход”;
- Всплывает окно: ”пожалуйста введите данные пользователя”;
- Автоматически открывается окно авторизации;
  - a. В данном окне можно авторизоваться, введя e-mail и пароль;
  - b. Выйти из учетной записи, если это необходимо и пользователь уже авторизован;
  - c. Зарегистрироваться в новом окне, заполнив данные о себе, придумав пароль и выбрав роль. После этого нажимает кнопку «Зарегистрироваться»;

## **2. Сценарий использования – «Добавление или удаление заказов»**

### **Основной сценарий – «Добавление заказа»**

**Действующее лицо: Заказчик**

- Заказчик открывает приложение;
- Заказчик входит в окно авторизации;
  - a. В данном окне можно авторизоваться, введя e-mail и пароль;
  - b. Выйти из учетной записи, если это необходимо и заказчик уже авторизован;

- После авторизации окно заказов в центральной части приложения меняется на форму заказчика, в которой можно управлять находящимися в БД своими заказами и запрашивать статистику;
- Нажимает на кнопку «Добавить заказ» и Заказчик заполняет все необходимые данные, также данные можно импортировать на кнопку «Загрузить данные» (экспорт – «Выгрузить данные»).

#### **Альтернативный сценарий.**

##### **Действующее лицо: Заказчик**

- Заказчик, войдя в приложение, входит в окно авторизации;
- Заказчик неправильно указывает e-mail и пароль;
- Заказчик остается в статусе «неавторизованного пользователя».

#### **Дополнительный сценарий – «Просмотр и удаление заказов»**

##### **Действующее лицо: Заказчик**

- Заказчик открывает приложение;
- Заказчик входит в окно авторизации;
  - а. В данном окне можно авторизоваться, введя e-mail и пароль;
  - б. Выйти из учетной записи, если это необходимо и заказчик уже авторизован;
  - в. Зарегистрироваться в новом окне, заполнив данные о себе и придумав;
- После авторизации окно заказов в центральной части приложения меняется на форму заказчика, в которой можно управлять находящимися в БД своими заказами и запрашивать статистику;
- Нажимает на кнопку «Удалить заказ» и Заказчик удаляет заказ.

### **3. Сценарий использования – «Исполнение или удаление заказов»**

#### **Основной сценарий – «Добавление заказа»**

##### **Действующее лицо: Исполнитель**

- Исполнитель открывает приложение;
- Исполнитель входит в окно авторизации;
  - а. В данном окне можно авторизоваться, введя e-mail и пароль;

- б. Выйти из учетной записи, если это необходимо и исполнитель уже авторизован;
- После авторизации окно заказов в центральной части приложения меняется на форму исполнителя, в которой можно добавляться на новые заказы и удалять уже существующие;
- Нажимает на кнопку «Приступить» и исполнитель приступает к исполнению заказа;

#### **Альтернативный сценарий.**

##### **Действующее лицо: Исполнитель**

- Исполнитель, войдя в приложение, входит в окно авторизации;
- Исполнитель неправильно указывает e-mail и пароль;
- Исполнитель остается в статусе «неавторизованного пользователя».

#### **Дополнительный сценарий – «Просмотр и удаление заказов»**

##### **Действующее лицо: Исполнитель**

- Исполнитель открывает приложение;
- Исполнитель входит в окно авторизации;
  - а. В данном окне можно авторизоваться, введя e-mail и пароль;
  - б. Выйти из учетной записи, если это необходимо и исполнитель уже авторизован;
  - с. Зарегистрироваться в новом окне, заполнив данные о себе и придумав;
- После авторизации окно заказов в центральной части приложения меняется на форму исполнителя.
- Нажимает на кнопку «Удалить заказ» и исполнитель удаляет из списка исполняемых заказов.

#### **4. Сценарий использования – «Просмотр статистики»**

##### **Основной сценарий – «Получение статистических метрик»**

##### **Действующее лицо: Заказчик**

1. Заказчик открывает приложение;
2. Заказчик входит в окно авторизации;
  - а. В данном окне можно авторизоваться, введя e-mail и пароль;

- b. Выйти из учетной записи, если это необходимо и заказчик уже авторизован;
  - c. Зарегистрироваться в новом окне, заполнив данные о себе и придумав пароль;
- 3. После авторизации окно заказов в центральной части приложения меняется на форму заказчика, в которой можно управлять находящимися в БД своими заказами и запрашивать статистику;
- 4. Заказчик нажимает на кнопку «Статистика» где заказчику выводится основные статистические метрики.

#### **Альтернативный сценарий.**

##### **Действующее лицо: Заказчик**

- 1. Заказчик, войдя в приложение, входит в окно авторизации;
- 2. Заказчик неправильно указывает e-mail и пароль;
- 3. Заказчик остается в статусе «неавторизованного пользователя».

#### **5. Сценарий использования – «Добавление данных для разметки»**

##### **Основной сценарий – «Добавление данных для разметки»**

##### **Действующее лицо: Заказчик**

- 1. Заказчик открывает приложение;
- 2. Заказчик входит в окно авторизации;
  - a. В данном окне можно авторизоваться, введя e-mail и пароль;
  - b. Выйти из учетной записи, если это необходимо и заказчик уже авторизован;
  - c. Зарегистрироваться в новом окне, заполнив данные о себе и придумав пароль;
- 3. После авторизации окно заказов в центральной части приложения меняется на форму заказчика, в которой можно управлять находящимися в БД своими заказами и запрашивать статистику;
- 4. Заказчик нажимает на кнопку «Добавить данные» где заказчик выбирает как загрузить файлы.
- 5. Заказчик загружает файлы в БД.

#### **Альтернативный сценарий.**

##### **Действующее лицо: Заказчик**

1. Заказчик, войдя в приложение, входит в окно авторизации;
2. Заказчик неправильно указывает e-mail и пароль;
3. Заказчик остается в статусе «неавторизованного пользователя».

## **6. Сценарий использования – «Выгрузка и загрузка структуры базы данных»**

### **Основной сценарий – «Выгрузка базы данных»**

#### **Действующее лицо: Админ**

1. Админ открывает приложение;
2. Админ входит в окно авторизации;
  - a. В данном окне можно авторизоваться, введя e-mail и пароль;
  - b. Выйти из учетной записи, если это необходимо и админ уже авторизован;
3. После авторизации открывается окно для импорта/экспорта структуры базы данных.
4. Админ нажимает на кнопку «Экспортировать данные БД».
5. Админ скачивает файлы БД.

### **Дополнительный сценарий – «Загрузка базы данных»**

#### **Действующее лицо: Админ**

1. Админ открывает приложение;
2. Админ входит в окно авторизации;
  - a. В данном окне можно авторизоваться, введя e-mail и пароль;
  - b. Выйти из учетной записи, если это необходимо и админ уже авторизован;
3. После авторизации открывается окно для импорта/экспорта структуры базы данных.
4. Админ нажимает на кнопку «Импортировать данные БД».
5. Админ загружает файлы БД.

### **Альтернативный сценарий.**

#### **Действующее лицо: Админ**

1. Админ, войдя в приложение, входит в окно авторизации;
2. Админ неправильно указывает e-mail и пароль;
3. Админ остается в статусе «неавторизованного пользователя».

## IV. МОДЕЛЬ ДАННЫХ

### NoSQL Модель данных

#### 1. Графическое представление

Графическое представление модели данных MongoDB показана на рис. 1

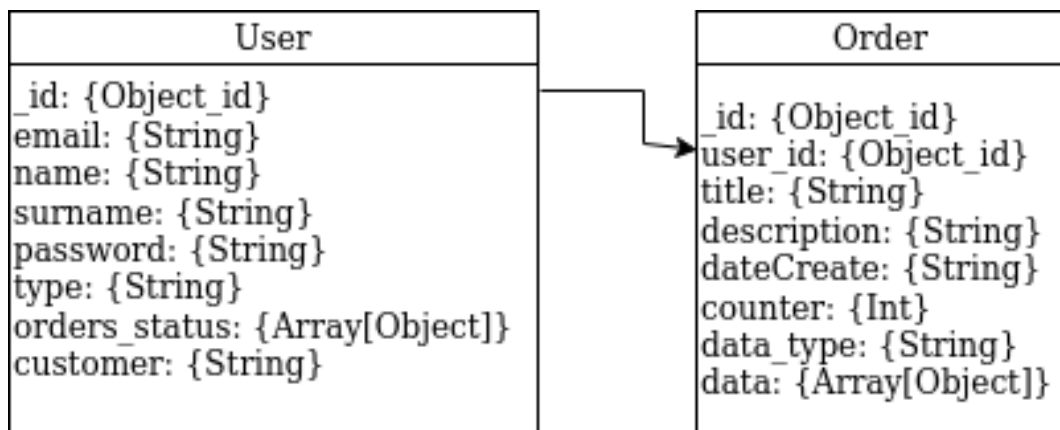


Рис. 1

#### 2. Сущности модели данных

Всего было выделено 2 сущности: User, Order.

User — сущность пользователя, содержащая следующие атрибуты:

- «`_id`», Object\_id — уникальный идентификатор пользователя. 12В
- «`email`», String — email пользователя. 50\*2В
- «`name`», String — имя пользователя. 50\*2В
- «`surname`», String — фамилия пользователя. 50\*2В
- «`password`», String — пароль пользователя. 50\*2В
- «`type`», String — тип пользователя(заказчик/исполнитель). 50\*2В
- «`orders_status`», Array[Object] - информация о текущих и выполненных заказах исполнителя.

Object {

    title: string, - имя заказа 50\*2В

    status: string — состояние выполнения. 50\*2В

}

- «`customer`», String - имя заказчика. 50\*2В

Order — сущность заказа, содержит следующие атрибуты:

- «`_id`», Object\_id — уникальный идентификатор заказа. 12В
- «`user_id`», Object\_id — уникальный идентификатор пользователя. 12В
- «`title`», String — имя заказа. 50\*2В
- «`description`», String — описание заказа. 50\*2В
- «`dateCreate`», String — дата создания заказа. 50\*2В
- «`counter`», int — количество пользователей выполнивших заказ. 4ВВ
- «`data_type`», String — тип данных для заказа. 50\*2В
- «`data`», Array[Object] — данные и результаты выполнения заказа.

```
Object {
  valueObj: string, - ссылка на фото либо текст. 50*2B
  { [key: string(50*2B)]: int(4BB) } - словарь для хранения результатов.
}
```

### 3. Оценка объема информации

Пусть А - количество пользователей, С — количество заказов, Е- количество ответов, D - количество объектов в наборе данных.

Тогда хранение чистых данных будет занимать:

$$A*(500 + C*200) + C*(4B04B + D*(100 + E*104B))$$

Фактический объем:

$$A*(512 + C*200) + C*(4B28 + D*(100 + E*104B))$$

$$\text{Избыточность модели: } \frac{A*(512 + C*200) + C*(4B28 + D*(100 + E*104B))}{A*(500 + C*200) + C*(4B04B + D*(100 + E*104B))}$$

Пусть количество пользователей А - 100 , количество заказов С — 1000, среднее количество отетов Е— 3, среднее количество данных в заказе D — 10.

Фактический объем:

$$100*(200512) + 100*(4B28 + 10*(100 + 312)) = 20506000(B) \approx 19.5 (MB)$$

### 4. Запросы

Создание нового пользователя:

```
db.users.insertOne(
  '_id': id,
  'email': email,
  'name': name,
  'surname': surname,
  'password': password,
  'type': type,
  'order_status': [],
  'customer': customer
)
```

Подсчет общего числа пользователей:

```
db.users.count()
```

Подсчет общего числа заказов:

```
db.orders.count()
```

# SQL Модель данных

## 1. Графическое представление

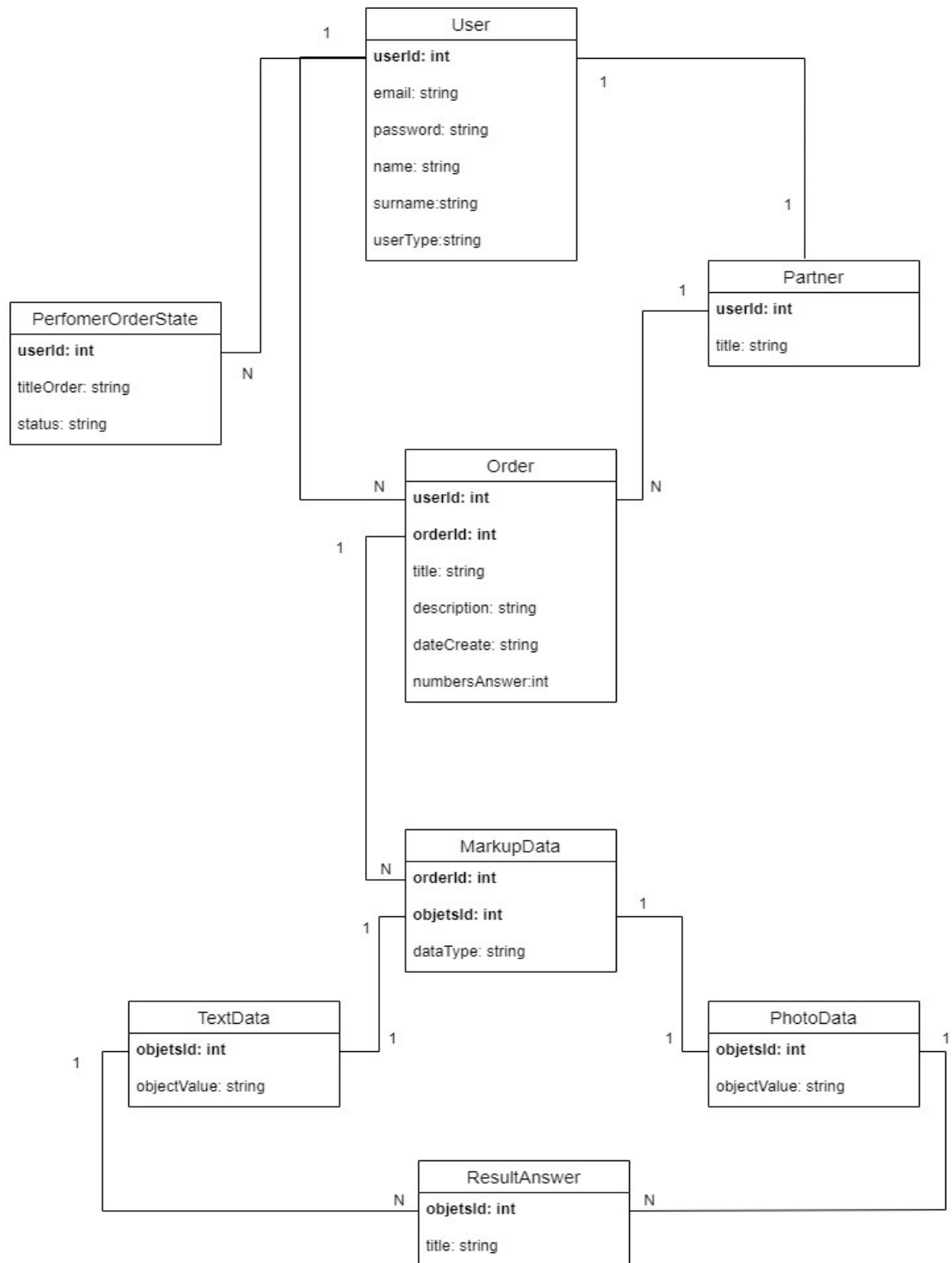


Рис. 2



## 2. Сущности модели данных

В качестве реляционной СУБД использована MySql, в которой создано 8 таблиц:

· User-хранит информацию о пользователе. Содержит поля:

- o Userid – уникальный идентификатор пользователя. Тип- int  $V = 4BB$
- o Email – @mail пользователя. Тип – string.  $V = 50 * 2B = 100B$
- o Password – пароль пользователя. Тип – string.  $V = 50 * 2B = 100B$
- o Name– имя пользователя. Тип – string.  $V = 50 * 2B = 100B$
- o Surname– фамилия пользователя. Тип – string.  $V = 50 * 2B = 100B$

Итого: 4B04BB

· PerfomerOrderState – хранит статус заказа Содержит поля:

- o Userid– уникальный идентификатор статуса заказа. Тип- int  $V = 4BB$
- o titleOrder– название заказа. Тип – string.  $V = 50 * 2B = 100B$
- o status– статус заказа. Тип – string.  $V = 50 * 2B = 100B$

Итого: 204BB

· Order – хранит информацию о заказе Содержит поля:

- o Userid – уникальный идентификатор пользователя. Тип- int  $V = 4BB$
- o Orderid– уникальный идентификатор заказа. Тип- int  $V = 4BB$
- o Title–название заказа. Тип – string.  $V = 50 * 2B = 100B$
- o Description–описание заказа. Тип – string.  $V = 50 * 2B = 100B$
- o dataCreate–дата создания заказа. Тип – string.  $V = 50 * 2B = 100B$
- o numberAnswer–количество ответов. Тип- int  $V = 4BB$

Итого:312B

· Partner – хранит информацию о заказчике Содержит поля:

- o Userid– уникальный идентификатор заказчика. Тип- int  $V = 4BB$
- o Title – имя заказчика. Тип – string.  $V = 50 * 2B = 100B$

Итого:104BB

· MarkupData – хранит информацию о данных для разметки Содержит поля:

- o Orderid– уникальный идентификатор заказа. Тип- int  $V = 4BB$
- o Objectsid– уникальный идентификатор объекта для разметки. Тип- int  $V = 4BB$
- o dataType– тип данных для разметки. Тип – string.  $V = 50 * 2B = 100B$

Итого:108B

· TextData – хранит текстовые данные для разметки Содержит поля:

- o Objectsid– уникальный идентификатор объекта для разметки. Тип- int  $V = 4BB$
- o objectValue– объект текста. Тип – string.  $V = 50 * 2B = 100B$

Итого:104BB

· PhotoData – хранит фото данные для разметки Содержит поля:

- o Objectsid– уникальный идентификатор объекта для разметки. Тип- int  $V = 4BB$
- o objectValue– объект изображения. Тип- string  $V = 100B$

Итого:10B

· ResultAnswer – хранит результаты разметки Содержит поля:

- o Objectsid– уникальный идентификатор объекта для разметки. Тип- int  $V = 4BB$
- o title– значение разметки. Тип – string.  $V = 50 * 2B = 100B$

Итого:104BB

### **3. Оценка объема информации**

Есть А пользователей, В из которых заказчики. У каждого заказчика имеется по С заказов. D из которых заказы текстовые. Так же имеется Е ответов пользователей

Чистый объем:

- $A * 4B04BB$  количество пользователей
- $C * 204B$  количество статусов заказа
- $C * 312$  количество информации о заказах
- $B * 104B$  количество заказчиков
- $C * 108$  количество информации о данных для разметки

- $D*104B$  количество текстовых данных
- $(C-D)*10B$  количество данных изображений
- $C*104BB$  количество ответов размеченных данных
- $E*104BB$  количество ответов пользователей

Чистый объем БД =  $A*4B04BB + C*204B + C*312 + B*104B + C*108 + D*104B + (C-D)*10B + C*104BB + E*104BB$

Фактический объем:

- $A*4B04BB$  количество пользователей
- $C*204B$  количество статусов заказа
- $C*312$  количество информации о заказах
- $B*104B$  количество заказчиков
- $C*108$  количество информации о данных для разметки
- $D*104B$  количество текстовых данных
- $(C-D)*10B$  количество данных изображений
- $C*104BB$  количество ответов размеченных данных
- $E*104BB$  количество ответов пользователей

Фактический объем БД =  $A*4B04BB + C*204B + C*312 + B*104B + C*108 + D*104B + (C-D)*10B + C*104BB + E*104BB$

Избыточность модели:  $(A*4B04BB + C*204B + C*312 + B*104B + C*108 + D*104B + (C-D)*10B + C*104BB + E*104BB) \setminus (A*4B04BB + C*204B + C*312 + B*104B + C*108 + D*104B + (C-D)*10B + C*104BB + E*104BB)$

Пусть 30% пользователей заказчики,  $A$  — количество пользователей, тогда  $B=0.3A$ , 1000 заказов, так же пусть имеется 10000 данных для заказов 7500 из которых картинки, тогда 2500 текстовых. Количество ответов пользователя 3.  $4B04BA + 204B*0.3A + 10000*204B + 1000*312 + 0.3A*104B + 1000*108 + 10000*108 + 2500*104B + 7500*100 + 1000*104B + 3000*104B = 22910000 + 207560A$

Если  $A=100$ , Тогда фактический объем занимает  $4B3666000B = 4B3,5MB$

#### **4. Запросы**

1.Добавить пользователя:

```
INSERT INTO User VALUES (...)
```

2.Найти автора заказа с индексом 2

```
SELECT * FROM Order INNER JOIN User ON User.id=2
```

3.Подсчет фото с котом в названии

```
SELECT COUNT(*) as count FROM db.PhotoData WHERE objectsValue='*cat'
```

#### **V. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ**

##### **5.1. Схема экранов приложения**



Рисунок 5.1 — Схема экранов приложения.

**5.2. Используемые технологии.**

Node.js, Vue.js, CSS, Flask, Mongo.

**5.3. Ссылки на приложение.**

1. Github: <https://github.com/moevm/nosql2h20-crowdsource-mongo>

## **VI. ВЫВОДЫ**

### **6.1. Достигнутые результаты.**

Было разработано user-friendly приложение, в функциональность которого входит: авторизация пользователей, создание и исполнение заказов по разметке текста и изображений, импорт и экспорт данных БД. В качестве системы управления базами данных используется MongoDB.

### **6.2. Недостатки и пути для улучшения полученного решения.**

К недостаткам текущей реализации можно отнести скудный набор разметок.

### **6.3. Будущее развитие решения.**

Дальнейшее развитие приложения предполагает увеличение числа форматов разметки, улучшение статистики и интерфейса разметки.

## **VII. ПРИЛОЖЕНИЯ**

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

Инструкция для Docker.

1. Скачать репозиторий [4].
2. Внутри папки директории проекта открыть терминал.
3. Выполнить команду `docker-compose up --build`.



## **VIII. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Документация Vue.js — <https://v3.vuejs.org/guide/introduction.html>.
2. Документация MongoDB — <https://docs.mongodb.com/>.
3. Github-репозиторий — <https://github.com/moevm/nosql2h20-crowdsourcing-mongo>