**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

**Индивидуальное домашнее задание**

**по дисциплине «Нереляционные базы данных»**

**Тема: Сервис поиска фрилансеров**

|  |  |  |
| --- | --- | --- |
| Студент гр. 7303 |  | Батурин И. |
| Студент гр. 7303 |  | Мининг М.С. |
| Студент гр. 7303 |  | Шаталов Э.В. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2020

**ЗАДАНИЕ**

наИндивидуальное домашнее задание

|  |  |  |
| --- | --- | --- |
| Студенты Батурин И.  Мининг М.С.  Шаталов Э.В.  группа 7303 | | |
|  | | |
| Тема работы: сервис поиска фрилансеров. | | |
| Исходные данные:  Создание приложения, в функциональность которого входят авторизация и регистрация исполнителей и заказчиков, добавление новых заданий, выбор исполнителей заказчиком среди откликнувшихся, возможность добавления отзывов по завершению выполнения задания, совершение транзакций. | | |
| 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. 7303 |  | Батурин И. |
| Студент гр. 7303 |  | Мининг М.С. |
| Студент гр. 7303 |  | Шаталов Э.В. |
| Преподаватель |  | Заславский М.М. |

**Аннотация**

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

Найти исходный код и дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h20-freelancers/tree/dev.

**Summary**

Within the framework of this course, it was supposed to write an application in a team was one of the set topics. The topic of creating a service for freelancers was chosen.

You can find the source code and additional information here: https://github.com/moevm/nosql2h20-freelancers/tree/dev.

**содержание**

|  |  |  |
| --- | --- | --- |
| [1. ВВЕДЕНИЕ 6](#_Toc52402883)  [2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ 7](#_Toc52402884)  [3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ 8](#_Toc52402885)  [4. МОДЕЛЬ ДАННЫХ 20](#_Toc52402886)  [5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ 35](#_Toc52402887)  [6. ВЫВОДЫ 38](#_Toc52402888)  [7. ПРИЛОЖЕНИЯ 39](#_Toc52402889)  [8. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 40](#_Toc52402891) |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. введение

Целью работы является создание SPA-приложения в браузере, в функциональность которого входят размещение объявлений о работе с указанием сферы работы для работодателей и возможность просматривать задания и отвечать работодателям для исполнителей. Также пользователи будут иметь свой рейтинг, который формируется при оценке работодателя фрилансеру за проделанную работу и наоборот. Выбранный стек технологий: база данных Mongo db, бекенд реализован на языке JavaScript с фреймворком express, фронтенд написан на языке программирования JavaScript с использованием фреймворка Vue.js.

1. Качественные требования к решению

Требуется разработать user-friendly приложение, в функциональность которого будут входить: страницы для авторизации и регистрации, просмотра профиля с возможностью оставить отзыв, получения списка всех заданий с возможностью фильтрации, создания задания, совершения транзакций по завершению задания для работодателей и исполнителей; получение статистики, импорт и экспорт данных БД для администратора. В качестве системы управления базами данных использовать MongoDB.

1. Сценарии использования

**3.1 Макеты UI**

1. Авторизация существующих пользователей.

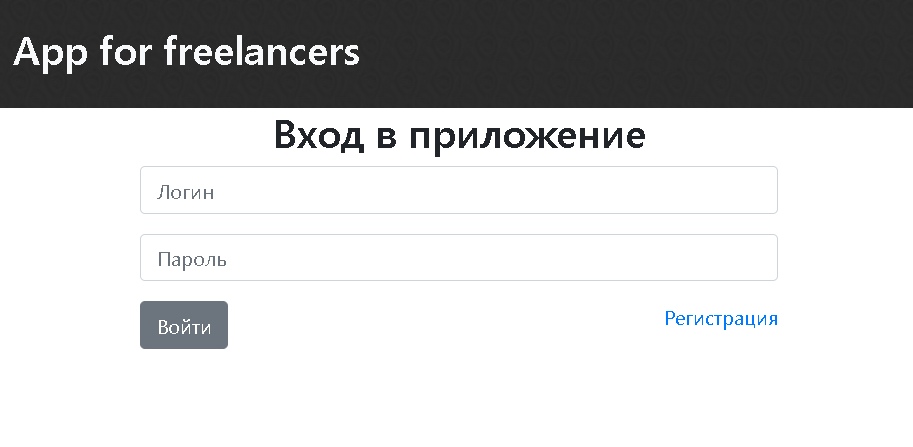


Рисунок 3.1. — Авторизация существующих пользователей.

1. Регистрация новых пользователей.

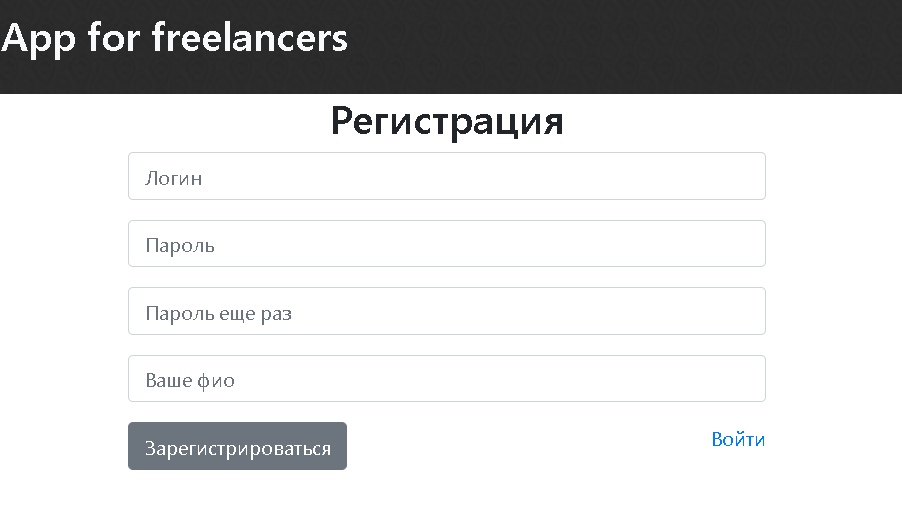


Рисунок 3.2. — Регистрация новых пользователей.

1. Список заданий.

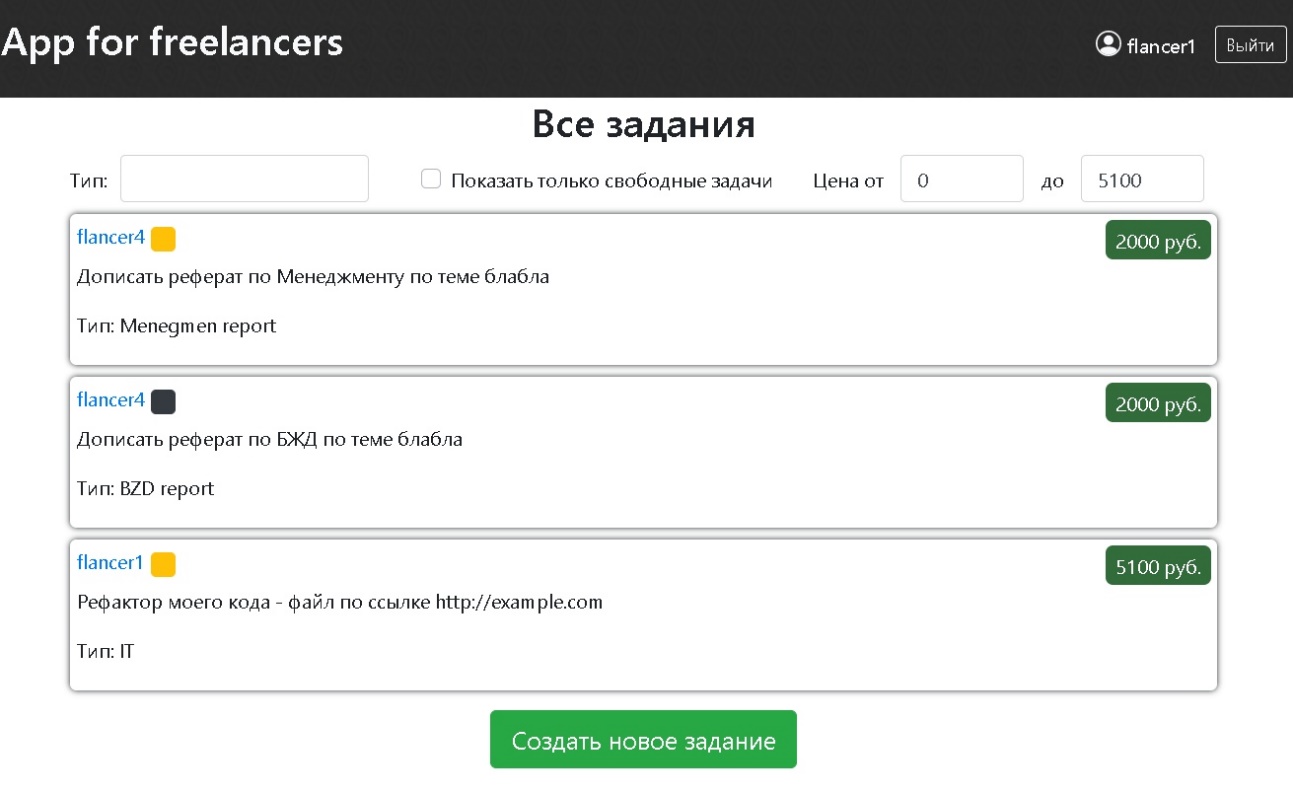


Рисунок 3.3. — Список заданий.

1. Создание нового задания.

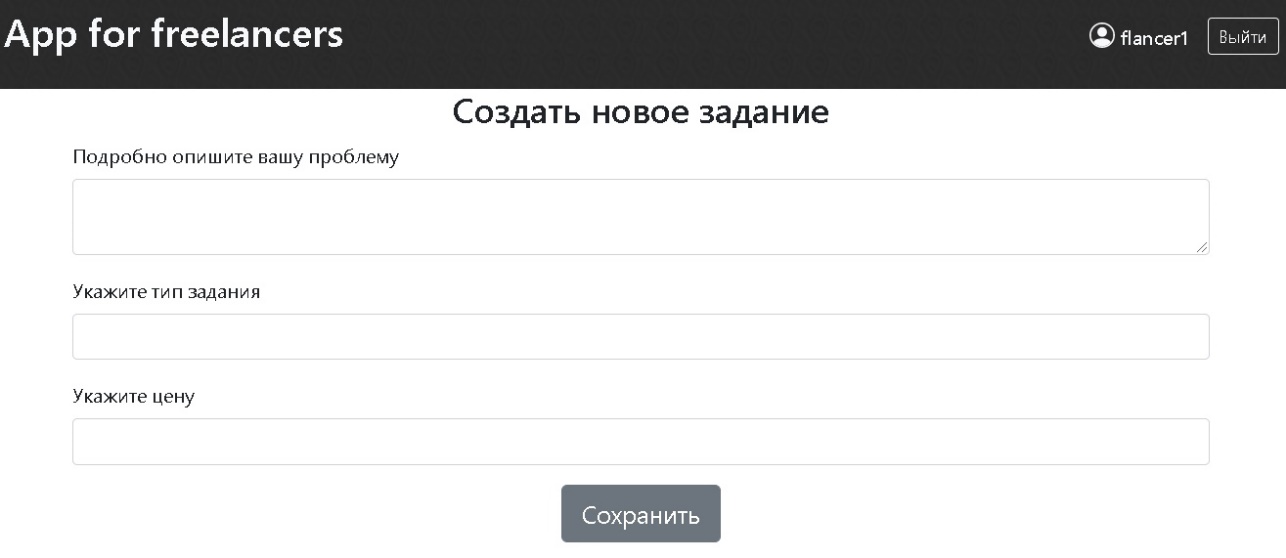


Рисунок 3.4. — Создание нового заданий.

1. Личный кабинет пользователя.

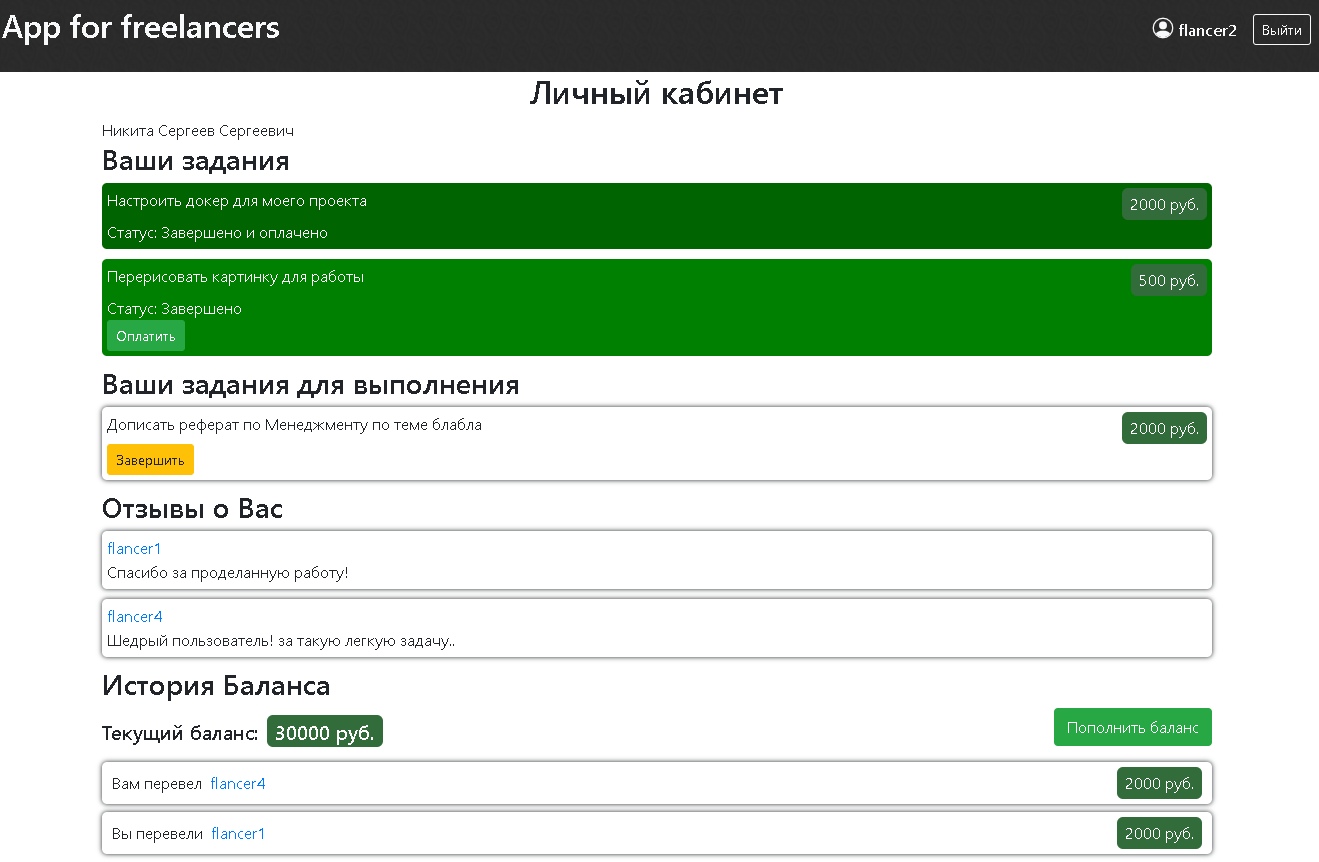


Рисунок 3.5 — Личный кабинет пользователя.

1. Просмотр выбранного задания, у которого уже есть исполнитель.

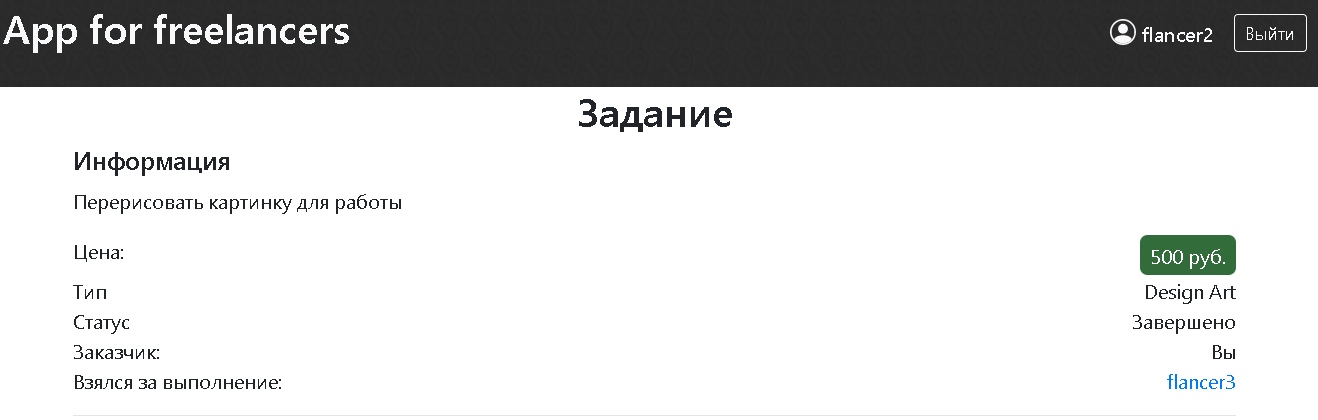


Рисунок 3.6 — Просмотр выбранного задания, у которого уже есть исполнитель.

1. Просмотр профиля пользователя.

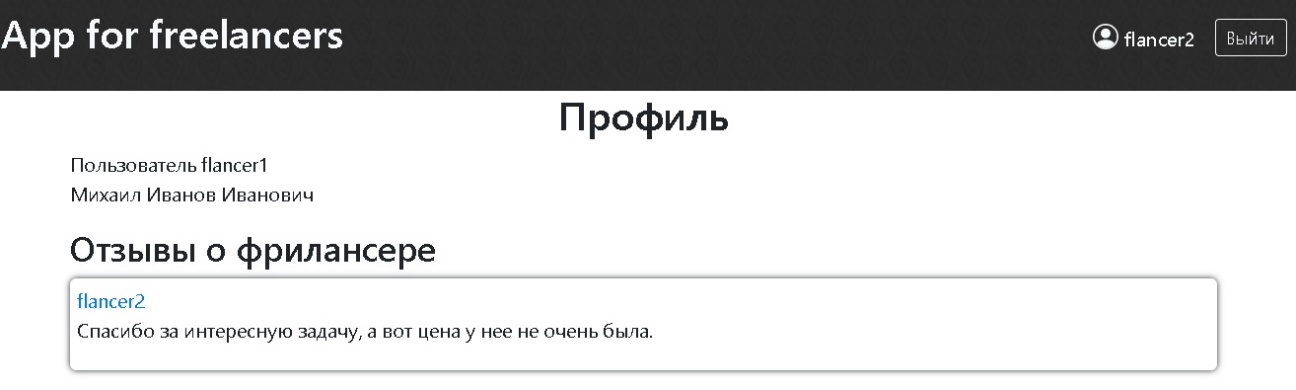


Рисунок 3.7 — Просмотр профиля пользователя.

1. Отправка задания на проверку с возможностью оставить отзыв о работодателе.

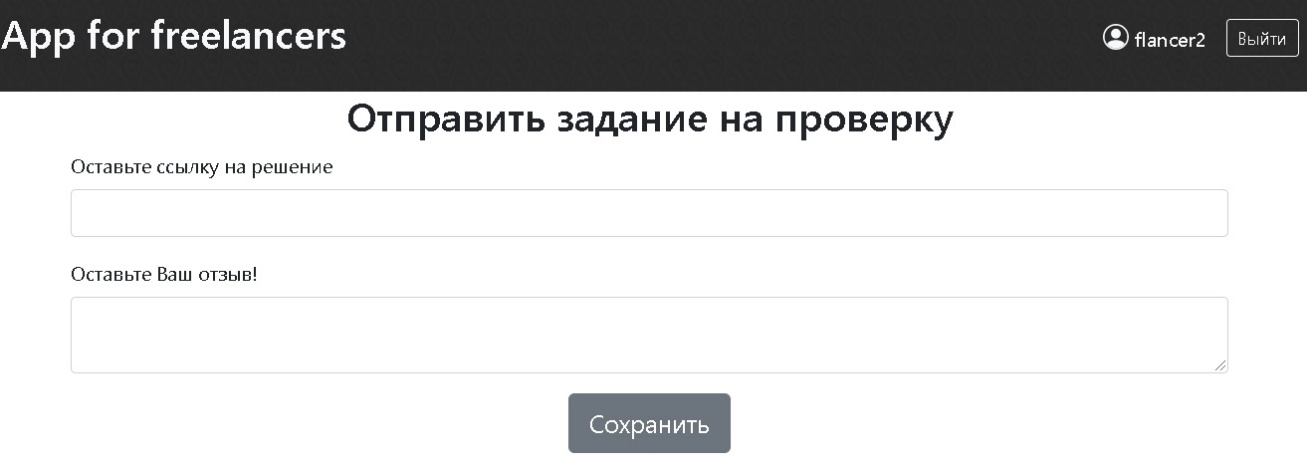


Рисунок 3.8 — Отправка задания на проверку с возможностью оставить отзыв о работодателе.

1. Оплата задания с возможностью оставить отзыв об исполнителе.

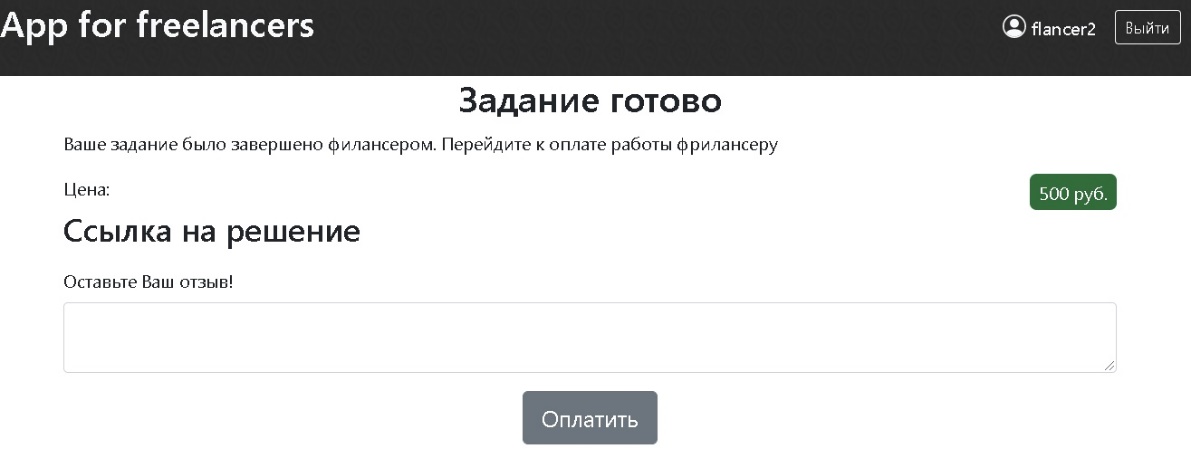
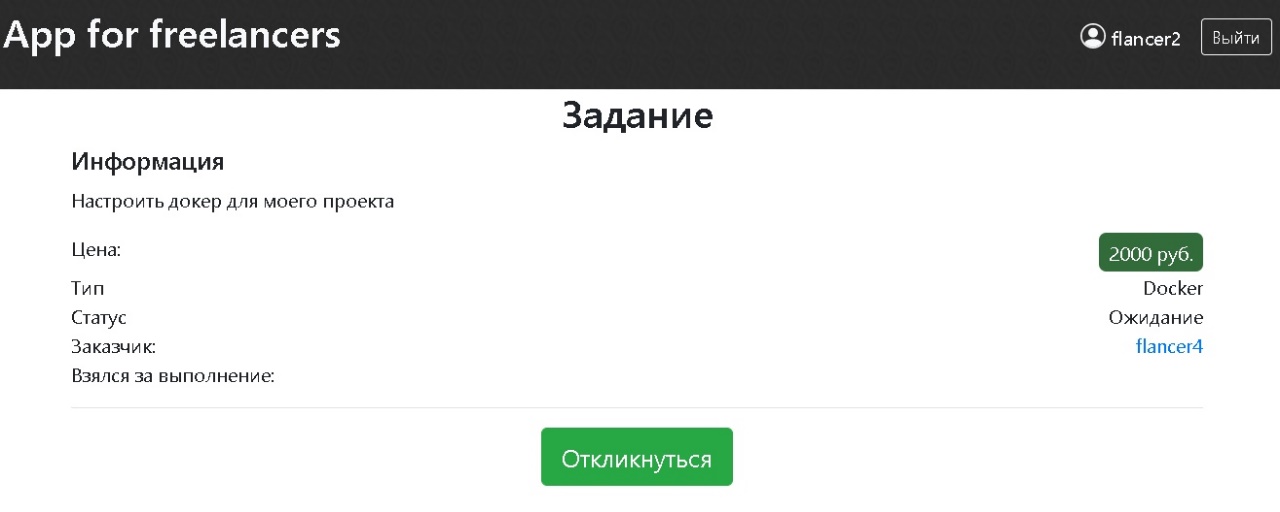


Рисунок 3.9 — Оплата задания с возможностью оставить отзыв об исполнителе.

1. Просмотр выбранного задания, у которого еще нет исполнителя.

Рисунок 3.10 — Просмотр выбранного задания, у которого еще нет исполнителя.

1. Импорт/экспорт БД.

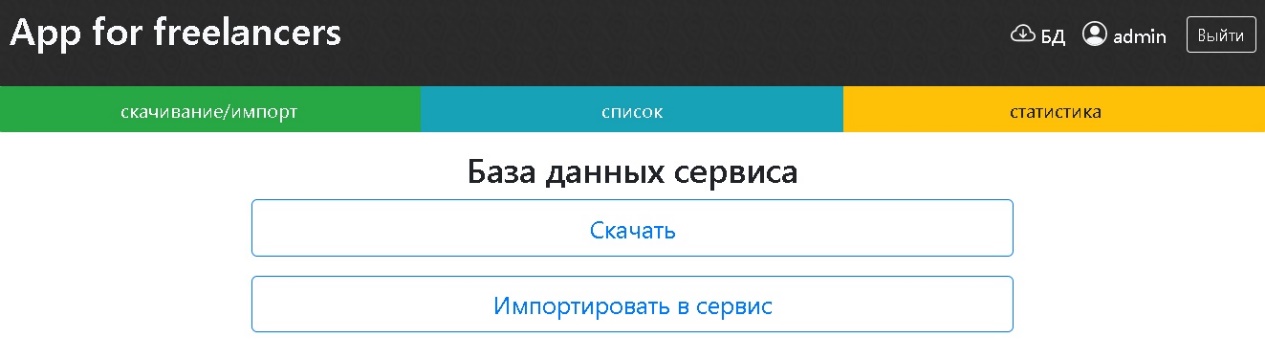


Рисунок 3.11 — Импорт/экспорт БД.

1. Список коллекций БД.

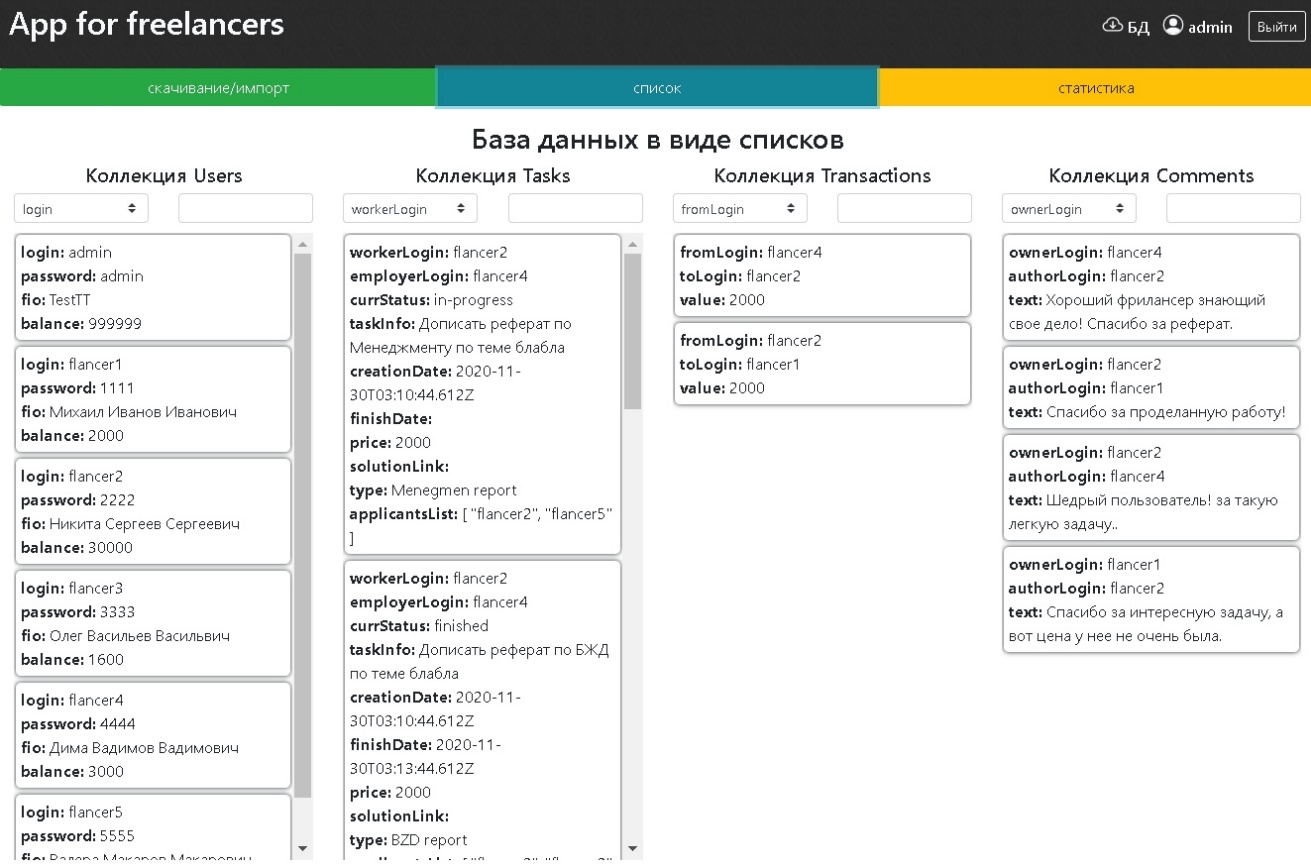


Рисунок 3.12 — Список коллекций БД.

1. Статистика всех пользователей.

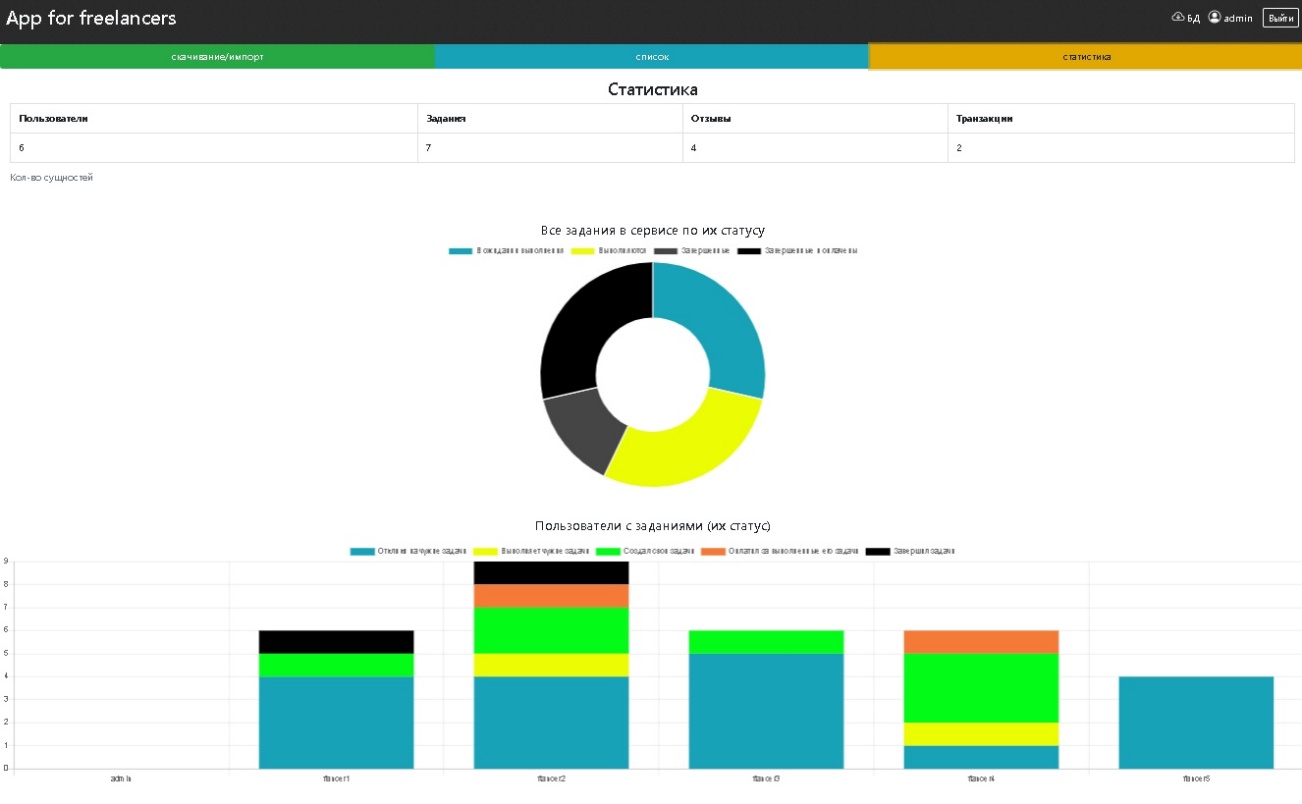


Рисунок 3.13 — Статистика всех пользователей.

1. Пополнение баланса.

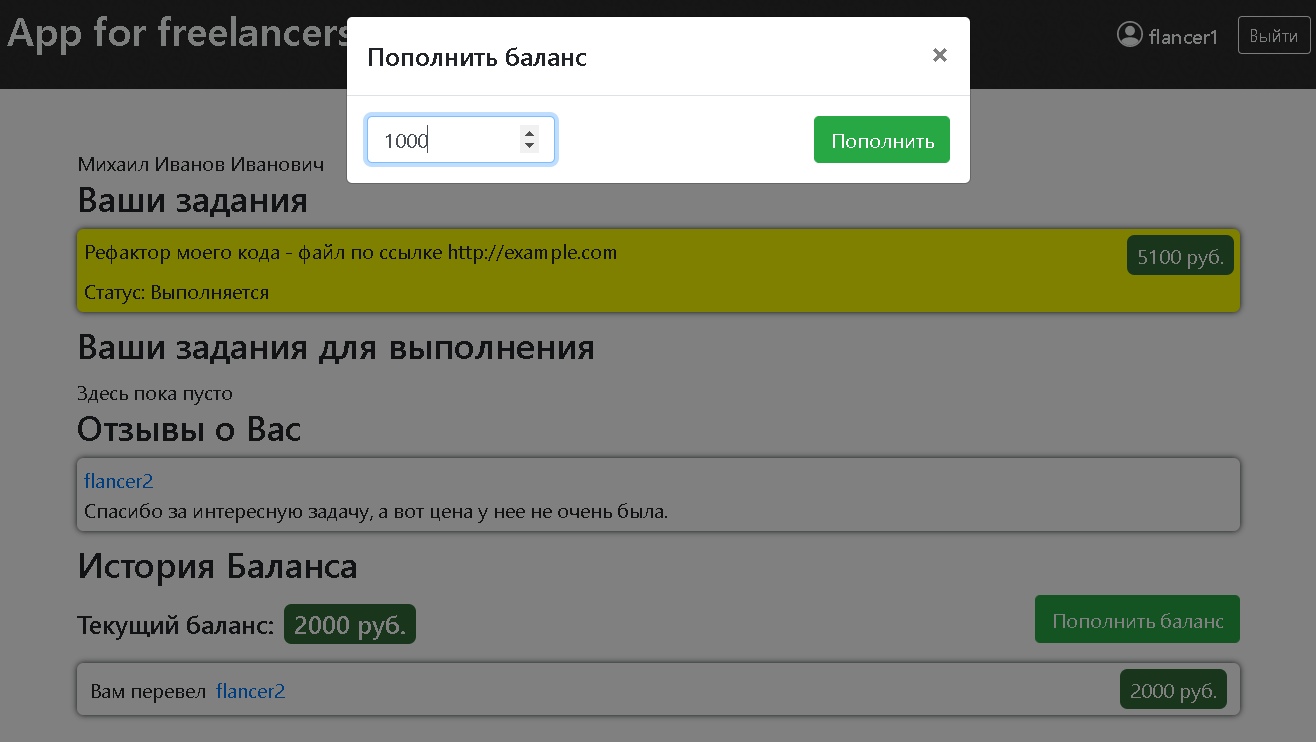


Рисунок 3.14 — Пополнение баланса.

1. Профиль пользователя с новым отзывом.

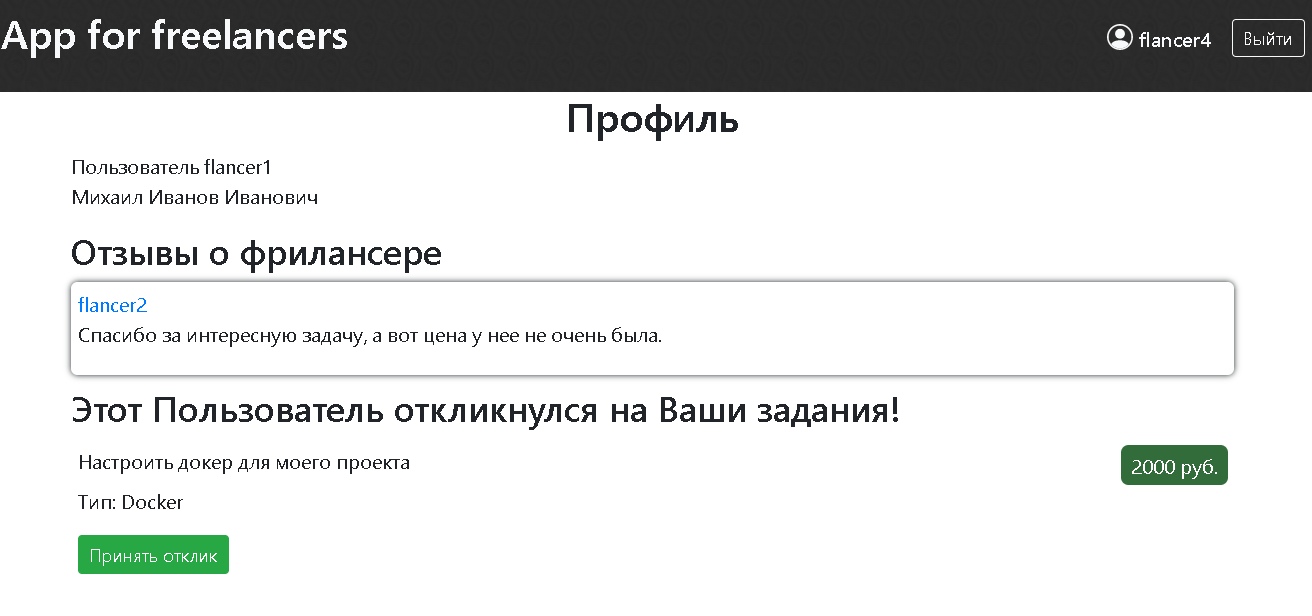


Рисунок 3.15 — Профиль пользователя с новым отзывом.

**3.2 Сценарии использования.**

**Действующее лицо: Пользователь**

1. Регистрация

* Пользователь входит в систему
* Пользователь выбирает регистрацию
* Пользователь заполняет форму и заканчивает регистрацию

1. Авторизация

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем

1. Просмотр всех заданий

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь попадает на главную, где видит все задания в системе
* Пользователь задает параметры задания (тип, цену) для фильтрации заданий

1. Добавление нового задания

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь попадает на главную, где видит все задания в системе
* Пользователь нажимает кнопку "Создать новое"
* Пользователь заполняет форму "Создать задание" и нажимает сохранить

1. Откликнутся на задание другого пользователя

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь попадает на главную, где видит все задания в системе
* Пользователь нажимает на одно из заданий
* Пользователь нажимает кнопку "Откликнутся"

1. Просмотр профиля других пользователей

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь нажимает на ФИО пользователя в задании или со своего личного кабинета

1. Просмотр в личном кабинете своих заданий и их статус

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь открывает личный кабинет

1. Просмотр в личном кабинете задания других пользователей, которые рассмотрели отклик пользователя и ожидают выполнения

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь открывает личный кабинет

1. Просмотр в личном кабинете отклики, которые оставили другие пользователи

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь открывает личный кабинет

1. Просмотр в личном кабинете все транзакции пользователя

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь открывает личный кабинет

1. Завершить задание другого пользователя, оставить комментарий в его профиле

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь открывает личный кабинет
* Пользователь нажимает "Завершить" в личном кабинете в "Ваши задания для выполнения"
* Пользователь отправляет форму для завершения задания, оставляя комментарий

1. Оплатить за завершенное задание другому пользователю, оставить комментарий в его профиле

* Пользователь входит в систему
* Пользователь авторизуется в системе со своим логином и паролем
* Пользователь открывает личный кабинет
* Пользователь нажимает "Оплатить" в личном кабинете в "Ваши задания" в задании, у которого статус *Завершенное*
* Пользователь оставляет комментарий пользователю и оплачивает за готовое задание

**Действующее лицо: Администратор (логин: admin, пароль: admin)**

1. Скачивание бд

* Пользователь входит в систему
* Пользователь открывает вкладку 'БД - скачивание/импорт'
* Пользователь нажимает скачать

1. Импорт другой бд в сервис

* Пользователь входит в систему
* Пользователь открывает вкладку 'БД - скачивание/импорт'
* Пользователь нажимает импортировать в сервис

1. Просмотр содержимого БД с помощью списков и поиска (фильтрации) данных.

* Пользователь входит в систему
* Пользователь открывает вкладку 'БД - список'
* Пользователь просматривает содержимое базы данных с возможностью фильтрации по свойствам сущностей.

1. Просмотр содержимого БД с помощью списков и поиска (фильтрации) данных.

* Пользователь входит в систему
* Пользователь открывает вкладку 'БД - статистика'
* Пользователь просматривает статистику базы данных.

1. модель данных
   1. **NoSQL MongoDB**
      1. **Список сущностей модели**

Разработанная модель включает следующие коллекции: User, Task, Comment, Transaction. Графическое представление этих коллекций приведено на рис. 4.1-4.4.

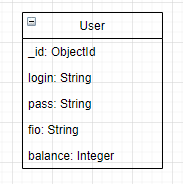


Рисунок 4.1 – Графическое представление модели данных «Пользователь»

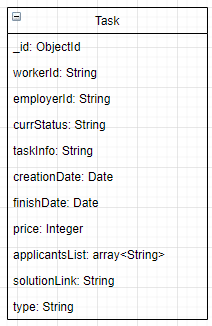


Рисунок 4.2 – Графическое представление модели данных «Задание»

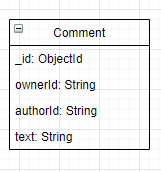


Рисунок 4.3 – Графическое представление модели данных «Комментарий»

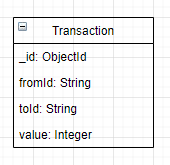


Рисунок 4.4 – Графическое представление модели данных «Транзакция»

* + 1. **Описание назначений коллекций, типов данных и сущностей.**

1. **Коллекция «User» хранит данные о существующих пользователях и их роли.**

* \_id - Уникальный идентификатор пользователя. Type: ObjectId. Size: 12B
* Login - Логин пользователя. Type: String. Size: 25B
* Password - Пароль пользователя. Type: String. Size: 25B
* Fio - ФИО зарегистрированного пользователя. Type: String. Size: 25B
* Balance - баланс пользователя. Type: Integer. Size: 8B

1. **Коллекция «Task» хранит созданные задания.**

* \_id - Уникальный идентификатор задания. Type: ObjectId. Size: 12B
* WorkerId - id фрилансера, выполняющего данное задание, пока работник не найден - поле пустое. Type: String. Size: 25B
* EmployerId - id работодателя, создавшего данное задание. Type: String. Size: 25B
* CurrStatus - текущий статус задания (Новое / В процессе / Выполнено). Type: String. Size: 25B
* TaskInfo - Текстовое описание задания. Type: String. Size: 25B
* Type - Категория задания. Type: String. Size: 25B
* CreationDate - Дата создания задания. Type: Date. Size: 41B
* FinishDate - Дата выполнения задания. Type: Date. Size: 41B
* Price – Цена, выставленная за выполнение задания. Type: Integer. Size: 8B
* ApplicantsList - лист претендентов на выполнение задачи. Type: Array. Size: m\*25B
* SolutionLink - ссылка на решение, которое отправляет фрилансер. Type: String. Size: 25B

1. **Коллекция «Comment» хранит отзывы пользователей системы.**

* \_id - Уникальный идентификатор комментария. Type: ObjectId. Size: 12B
* OwnerId - id адресанта. Type: String. Size: 25B
* AuthorId - id автора отзыва. Type: String. Size: 25B
* Text - Текст отзыва. Type: String. Size: 25B

1. **Коллекция «Transaction» хранит данные о произведенных и запланированных транзакциях.**

* \_id - Уникальный идентификатор пользователя. Type: ObjectId. Size: 12B
* FromId - id отправителя. Type: String. Size: 25B
* ToId - id получателя. Type: String. Size: 25B
* Value - Сумма операции. Type: Integer. Size: 8B
  + 1. **Оценка удельного объема информации, хранимой в модели.**
  1. **User.**

Чистый объем: (25 + 25 + 25 + 8)\*n = 83n

Фактический объем: (25 + 25 + 25 + 8 + 12(\_id))\*n= 95n,

где n – количество пользователей.

* 1. **Task.**

Чистый объем: (25\*5 + 8 + 82 *+* 25\*m)\*n = (149+ 25m)nФактический объем:(25\*5 + 8 + 82 + 25*m* + 12(\_id))\*n= (161+ 25m)\*n,

где n – количество задач, m – кол-во заявок на задачу.

* 1. **Comment**

Чистый объем: (25 \*3)\*n = 75n

Фактический объем: (25 \*3 + 12(\_id))\*n= 87n,

где n – количество комментариев.

* 1. **Transaction**

Чистый объем: (25 \*2 + 8)\*n = 58n

Фактический объем: (25 \*2 + 8 + 12(\_id))\*n= 70n,

где n – количество пользователей.

Тогда чистый объем информации будет равен:  
 N \* 83B (User) + N \* 3 \* 199B (Task) + N \* 5 \* 75B (Comment) + N \* 2 \* 58B (Transaction) = **N \* 1171B**  
Фактический объем информации будет равен:

N \* 95B (User) + N \* 3 \* 211B (Task) + N \* 5 \* 87B (Comment) + N \* 2 \* 70B (Transaction) = **N \* 1303B**

Отсюда, **избыточность**: 1303B \* N / 1171B \* N = **1.11**

* + 1. **Запросы к модели, с помощью которых реализуются сценарии использования.**

1. Добавление пользователя

db.users.insertOne(

'\_id': id,

'login': "policy",

'pass': "qwerty123",

'fio': "Policy Tribunal",

'balance': 0

)

1. Добавление таски

db.tasks.insertOne(

'\_id': id,

'WorkerId': "af1224df32dfd",

'EmployerId': "fgfytr454kk47",

'CurrStatus': "New",

'TaskInfo': "Нарисовать логотип для компании по продаже апельсинов",

'CreationDate': 02/09/2020,

'FinishDate': 01/01/2001,

'Price': 800,

'ApplicantsList': [""],

'SolutionLink': "",

)

1. Добавление транзакции

db.transactions.insertOne(

'\_id': id,

'FromId': "af1224df32dfd",

'ToId': "fgfytr454kk47",

'Value': 700

)

1. Добавление комментария

db.comms.insertOne(

'\_id': id,

'OwnerId': "af1224df32dfd",

'AuthorId': "fgfytr454kk47",

'Text': "Хороший пользователь. 10/10"

)

1. Пополнить баланс

db.users.update(

'\_id': id,

'balance': 500

)

1. Отображение всех активных тасок

db.tasks.find(

CurrStatus: "Active"

)

1. Отображение тасок 1 пользователя

db.tasks.find(

WorkerId: "sdf332dswr4e"

)

1. Отображение комментариев о пользователе

db.comms.find(

OwnerId: "sdf332dswr4e"

)

1. Изменение статуса таски

db.tasks.update(

'\_id': id,

'CurrStatus': "Active"

)

* 1. **NoSQL Neo4j**
     1. **Графическое представление.**

На рис. 4.5 приведено графического представление NoSQL Neo4j модели.

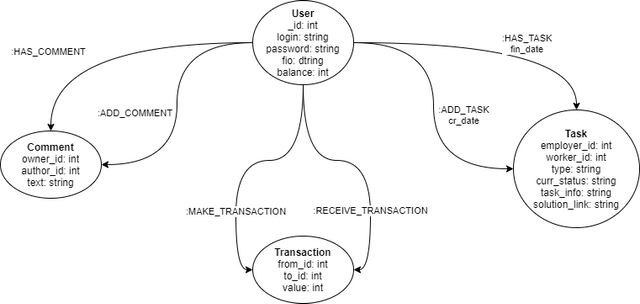


Рисунок 4.5 – Графическое представление модели

* + 1. **Описание назначений коллекций, типов данных и сущностей.**
  1. **«User» предназначен для хранения информации о пользователях**
* **\_id** - уникальный идентификатор пользователя. Type: int. Size: 4B
* **login** - логин пользователя. Type: string. Size: 25B
* **password** - пароль пользователя. Type: string. Size: 25B
* **fio** - ФИО зарегистрированного пользователя. Type: string. Size: 25B
* **balance** - текущий баланс пользователя: Type: int. Size: 4B

Total size: 4+25+25+25+4=**83B**

* 1. **«Task» предназначен для хранения информации о задании**
* **\_id** - идентификатор задания. Type: int. Size: 4B
* **employer\_id** - идентификатор создателя задания. Type: int. Size: 4B
* **worker\_id** - идентификатор исполнителя задания. Type: int. Size: 4B
* **curr\_status** - текущий статус задания. Type: string. Size: 25B
* **type** - категория задания. Type: String. Size 25B
* **task\_info** - текст задания. Type: string. Size: 25B
* **solution\_link** - ссылка на решение задания. Type: string. Size: 25B

Total size: 4+4+4+25+25+25+25=**112B**

* 1. **«Comment» предназначен для хранения комментария по завершению выполнения задания**
* **\_id** - идентификатор комментария. Type: int. Size: 4B
* **owner\_id** - идентификатор создателя задания. Type: int. Size: 4B
* **author\_id** - идентификатор исполнителя задания. Type: int. Size: 4B
* **text** - текст задания. Type: string. Size: 25B

Total size: 4+4+4+25=**37B**

* 1. **«Transaction» предназначен для хранения информации о денежных переводах**
* **\_id** - идентификатор транзакции. Type: int. Size: 4B
* **from\_id** - идентификатор отправителя. Type: int. Size: 4B
* **to\_id** - идентификатор получателя. Type: int. Size: 4B
* **value** - сумма оплаты. Type: int. Size: 4B

Total size: 4+4+4+4=**16B**

* 1. **Существует 6 связей между сущностями**
* **:ADD\_COMMENT** - данный пользователь добавил комментарий
* **:HAS\_COMMENT** - у данного пользователя есть комментарий
* **:ADD\_TASK** - данный пользователь добавил задание
* **cr\_date** - дата создания задания Type: date. Size: 8B
* **:HAS\_TASK** - у данного пользователя есть задание
* **fin\_date** - дата выполнения задания Type: date. Size: 8B
* **:MAKE\_TRANSACTION** - пользователь создал запрос на оплату
* **:RECEIVE\_TRANSACTION** - пользователь оплатил заказ

Total size: 8+8=**16B**

* + 1. **Оценка удельного объема информации, хранимой в модели.**
* N - количество пользователей
* Среднее количество комментарий каждого пользователя - 5
* Среднее количество заданий каждого пользователя - 3
* Среднее количество денежных переводов каждого пользователя - 2

1. Чистый объем:
   * User - N \* 83B
   * Task - N \* 3 \* 108B
   * Comment - N \* 5 \* 33B
   * Transaction - N \* 2 \* 12B
   * :ADD\_TASK - N \* 3 \* 8B
   * :HAS\_TASK - N \* 3 \* 8B

Занимаемый чистый объем - **N \* 620B**

1. Фактический объем:
   * User - N \* 83B
   * Task - N \* 3 \* 112B
   * Comment - N \* 5 \* 37B
   * Transaction - N \* 2 \* 16B
   * :ADD\_COMMENT - N \* 33B
   * :HAS\_COMMENT - N \* 5 \* 33B
   * :MAKE\_TRANSACTION - N \* 2 \* 12B
   * :RECEIVE\_TRANSACTION - N \* 2 \* 12B
   * :ADD\_TASK - N \* 8B
   * :HAS\_TASK - N \* 3 \* 8B

Занимаемый фактический объем - **N \* 914B**

**Избыточность:** 914B \*N / 620B \* N = **1.47**

* + 1. **Примеры запросов к модели.**

1. Добавления нового пользователя

CREATE (user1:User {...})

1. Найти комментарии, написанные пользователем с id 115

MATCH (k {\_id: 115}-[:HAS\_COMMENT]-(e) RETURN e.text, ...)

1. Подсчёт общего числа пользователей

MATCH (n:USER) RETURN count(n) as count

* 1. **SQL**
     1. **Графическое представление модели данных.**

На рис. 4.6 представлена модель данных для SQL.

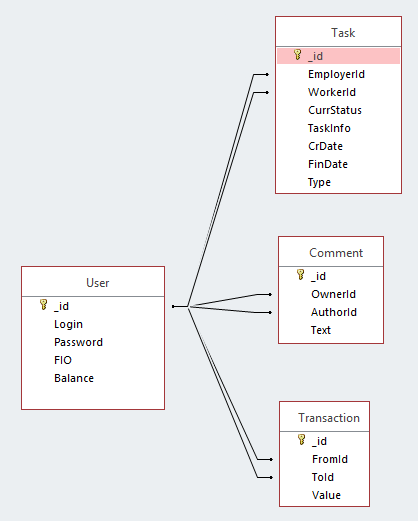


Рисунок 4.5 – Графическое представление SQL

* + 1. **Описание назначений коллекций, типов данных и сущностей.**
  1. **«User» предназначен для хранения информации о пользователях**
* **user\_id** - уникальный идентификатор пользователя. Type: string. Size: 25B
* **login** - логин пользователя. Type: string. Size: 25B
* **password** - пароль пользователя. Type: string. Size: 25B
* **fio** - ФИО зарегистрированного пользователя. Type: string. Size: 25B
* **balance** - текущий баланс пользователя: Type: int. Size: 8B

Total size: 25+25+25+25+8=**108B**

* 1. **«Task» предназначен для хранения информации о задании**
* **task\_id** - уникальный идентификатор задания. Type: string. Size: 25B
* **employer\_id** - идентификатор создателя задания. Type: string. Size: 25B
* **worker\_id** - идентификатор исполнителя задания. Type: string. Size: 25B
* **curr\_status** - текущий статус задания. Type: string. Size: 25B
* **type** - категория задания. Type: String. Size 25B
* **task\_info** - текст задания. Type: string. Size: 25B
* **solution\_link** - ссылка на решение задания. Type: string. Size: 25B

Total size: 25+25+25+25+25+25+25=**175B**

* 1. **«Comment» предназначен для хранения комментария по завершению выполненного задания**
* **comm\_id** - уникальный идентификатор комментария. Type: string. Size: 25B
* **owner\_id** - идентификатор создателя задания. Type: string. Size: 25B
* **author\_id** - идентификатор исполнителя задания. Type: string. Size: 25B
* **text** - текст задания. Type: string. Size: 25B

Total size: 25+25+25+25=**100B**

* 1. **«Transaction» предназначен для хранения информации о денежных переводах**
* **transaction\_id** - уникальный идентификатор транзакции. Type: string. Size: 25B
* **from\_id** - идентификатор отправителя. Type: string. Size: 25B
* **to\_id** - идентификатор получателя. Type: string. Size: 25B
* **value** - сумма оплаты. Type: int. Size: 8B

Total size: 25+25+25+8=**83B**

* + 1. **Оценка удельного объема информации, хранимой в модели.**
* N - количество пользователей
* Среднее количество комментарий каждого пользователя - 5
* Среднее количество заданий каждого пользователя - 3
* Среднее количество денежных переводов каждого пользователя - 2

1. Чистый объем:
   * User - N \* 108B
   * Task - N \* 3 \* 175B
   * Comment - N \* 5 \* 100B
   * Transaction - N \* 2 \* 83B

Занимаемый чистый объем - **N \* 1299B** 2. Фактический объем:

* User - N \* 108B
* Task - N \* 3 \* 150B
* Comment - N \* 5 \* 75B
* Transaction - N \* 2 \* 58B

Занимаемый фактический объем - **N \* 1049B**

**Избыточность:** 1299B \*N / 1049B \* N = **1.24**

* + 1. **Примеры запросов к модели**

1. Добавление пользователя

INSERT INTO User (\_id, Login, Password, FIO, Balance)

VALUES ("123", "policy", "qwerty123", "Policy Tribunal", 0);

1. Добавление комментария

INSERT INTO Comment (\_id, OwnerId, AuthorId, Text)

VALUES ("123", "af1224df32dfd", "fgfytr454kk47", "Хороший пользователь. 10/10");

1. Пополнить баланс

UPDATE Users SET balance = 500 WHERE \_id= af1224df32dfd;

1. Отображение всех активных тасок

SELECT \* FROM Task<br>

WHERE CurrStatus = 'Active';

* 1. **Итог**
     1. **Сравнение MongoDB с SQL**

Модель no-SQL вышла менее избыточной (1.11 против 1.24) а также менее объемной (N \* 1171B против N \* 1299B) Написание SQL-запросов занимает больше времени, и они получаются более громоздкими, чем написание аналогичных NoSQL-запросов на mongodb.

Mongodb в данном случае подходит в большей мере, чем SQL база данных.

* + 1. **Сравнение Neo4j с SQL**

Модель no-SQL вышла более избыточной (1.47 против 1.24), но менее объемной (914B \*N против N \* 1299B) Написание SQL-запросов занимает больше времени, чем написание аналогичных NoSQL-запросов на neo4j. neo4j в данном случае подходит в большей мере, чем SQL база данных, потому что она менее объёмная.

1. разработанное приложение
   1. **Схема экранов приложения**

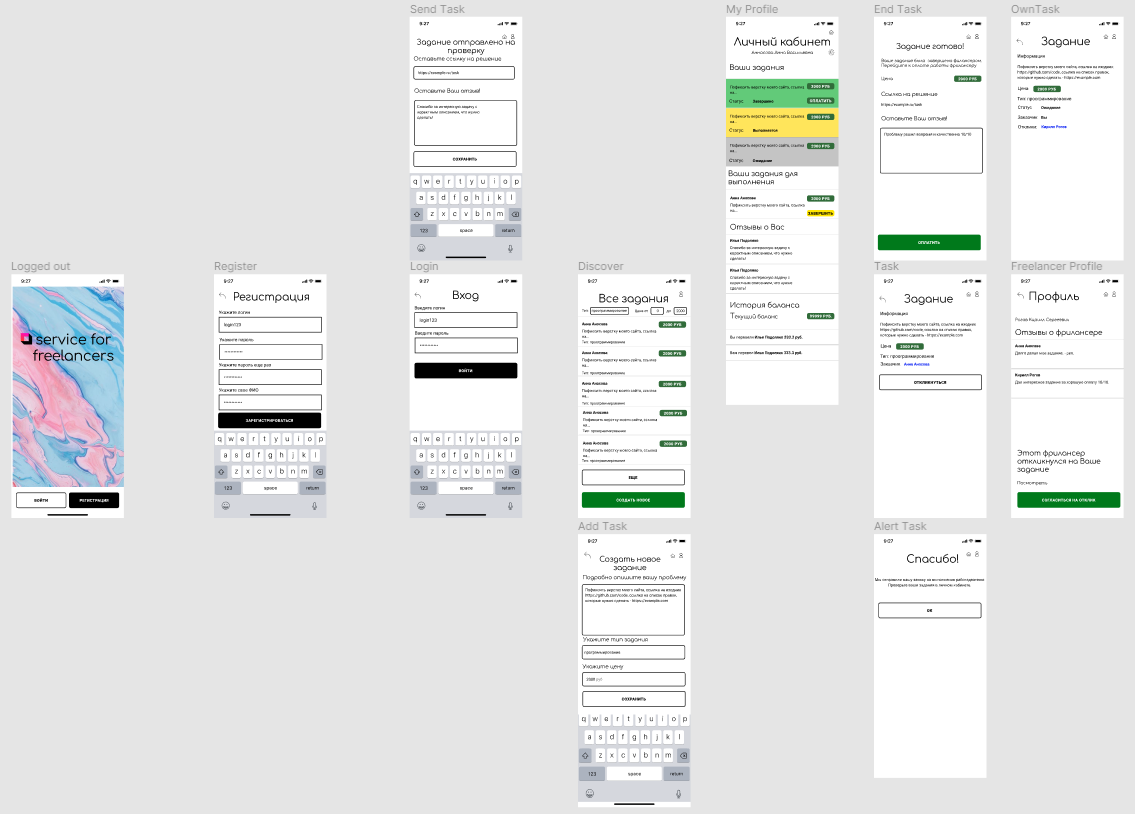


Рисунок 5.1 — Схема экранов приложения без переходов между экранами.

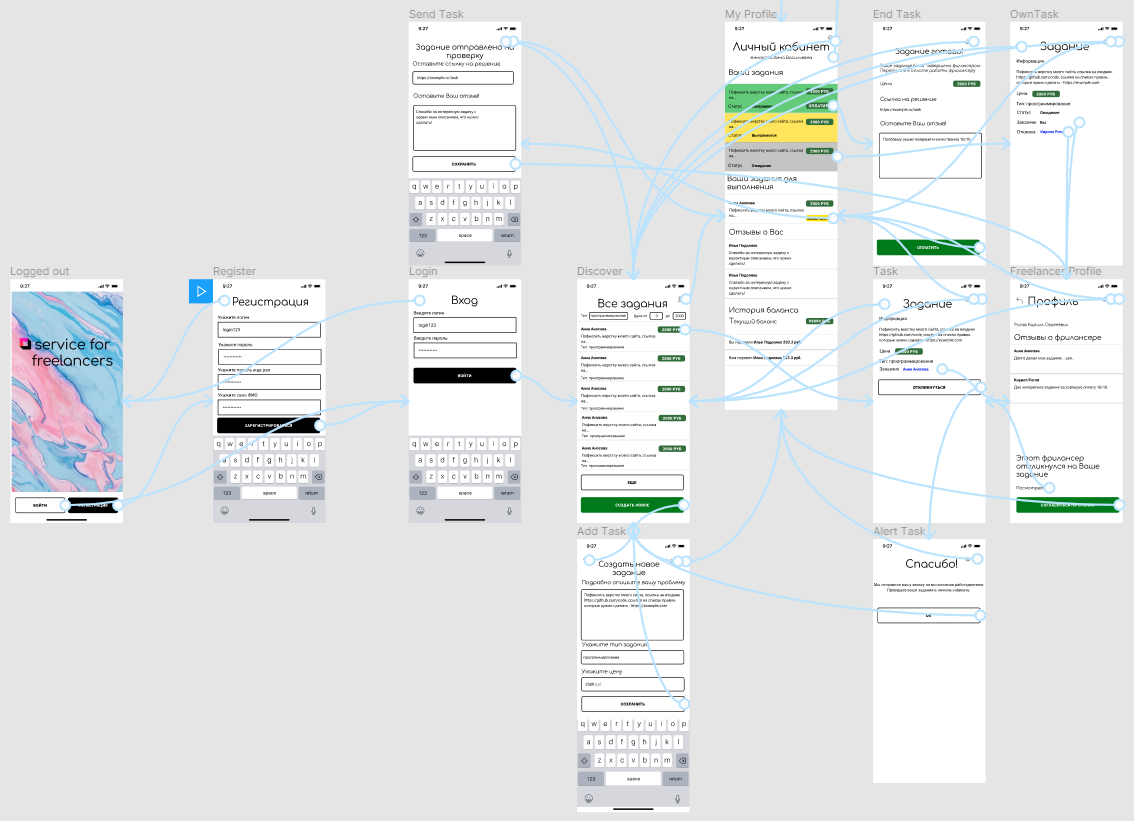


Рисунок 5.2 — Схема экранов приложения с переходов между экранами.

Logged out – Окно при первоначальном входе в приложение

Register – Окно регистрации

Login – Окно для авторизации

Discover – Список заданий сервиса

Send Task – Отправка решения на задачу

Add Task – Добавление задачи

My Profile – Профиль авторизованного пользователя

End Task – Окно после проверки корректности решения задания

Own Task – Задание с откликнувшимся пользователем

Task – Задание без откликнувшегося пользователя

Freelancer Profile – Профиль пользователя системы

After Task – Окно после отклика на заявку

* 1. **Использованные технологии.**

Node.js, Vue.js, CSS, Express.js, MongoDB.

* 1. **Ссылки на приложение.**

1. Github: https://github.com/moevm/nosql2h20-freelancers/tree/dev
2. выводы
   1. **Достигнутые результаты.**

В результате было разработано user-friendly приложение, в функциональность которого входит: страницы для авторизации и регистрации, просмотра профиля с возможностью оставить отзыв, получения списка всех заданий с возможностью фильтрации, создания задания, совершения транзакций по завершению задания для работодателей и исполнителей; получение статистики, импорт и экспорт данных БД для администратора. В качестве системы управления базами данных используется MongoDB.

* 1. **Недостатки и пути для улучшения полученного решения.**

К недостаткам текущей реализации можно отнести то, что не было подробно описано и реализовано пополнение счёта разными способами, редактирование и удаление заданий, использование только формата JSON для импорта- экспорта.

* 1. **Будущее развитие решения.**

Дальнейшее развитие приложения предполагает увеличение числа форматом для импорта-экспорта данных, авторизация и регистрация будет более безопасной с использованием токенов доступа, через телефон или почту. При дальнейшей реализации планируется улучшение UI, добавление возможности удаления и редактирования заданий, добавление чата и обмен документами между фрилансером и работодателем.

1. Приложения
   1. Документация по сборке и развертыванию приложения.

Инструкция для Docker.

1. Скачать репозиторий: git clone https://github.com/moevm/nosql2h20-freelancers.git.
2. Переключиться на ветку dev: git checkout dev
3. Перейти в папку с проектом: cd nosql2h20-freelancers
4. Выполнить команду: sudo docker-compose up –build.
5. список использованных источников
6. Документация Express.js — <https://expressjs.com/ru/starter/installing.html>
7. Документация Vue.js — <https://v3.vuejs.org/guide/introduction.html>.
8. Документация MongoDB — <https://docs.mongodb.com/>.
9. Github-репозиторий — https://github.com/moevm/nosql2h20-freelancers