**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

**по дисциплине «Введение в нереляционные базы данных»**

Тема: **Сервис для текстовых фрилансеров**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2300 |  | Жохов К.С. |
| Студент гр. 2300 |  | Шамхалов Я.М. |
| Студент гр. 2382 |  | Ивашинников Л.Д. |
| Студентка гр. 2382 |  | Ульянова Е.А. |
| Студент гр. 2382 |  | Щедрин А.А. |
| Преподаватель |  | Заславский М.М |

Санкт-Петербург

2025

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент Жохов К.С. 2300  Студент Шамхалов Я.М. 2300  Студент Ивашинников Л.Д. 2382  Студентка Ульянова Е.А. 2382  Студент Щедрин А.А. 2382 | | | |
|  | | | |
| Тема работы: Реализация сервиса для текстовых фрилансеров | | | |
| Исходные данные:  Сделать сервис, где Заказчики могут найти Фринлансеров (редакторы, писатели). Заказчики создают заказы, оценивают их выполнение, расплачиваются. Фрилансеры берутся за заказы, сдают их по этапу, оценивают Заказчиков.  Используемая база данных – MongoDB. | | | |
| Содержание пояснительной записки:  «Введение», «Сценарий использования», «Модель данных», «Разработанное приложение», «Выводы», «Приложения», «Литература» | | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | | |
| Дата выдачи задания: 05.02.2025 | | | |
| Дата сдачи реферата: 22.05.2025 | | | |
| Дата защиты реферата: 22.05.2025 | | | |
| Студент гр. 2300 |  | Жохов К.С. |
| Студент гр. 2300 |  | Шамхалов Я.М. |
| Студент гр. 2382 |  | Ивашинников Л.Д. |
| Студентка гр. 2382 |  | Ульянова Е.А. |
| Студент гр. 2382 |  | Щедрин А.А. |
| 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 для текстовых фрилансеров с использованием базы данных MongoDB. Во внимание будут приниматься такие аспекты как производительность и удобство разработки.

**Summary**

As part of this course, it was supposed to develop an application in a team on one of the set topics. The theme of creating an application for text freelancers using the MongoDB database was chosen. Aspects such as performance and ease of development will be taken into account.**содержание**

Введение 6

1. СценариИ использования 7

1.1. Макеты UI 7

1.2. Описание сценариев использования 35

2. Модель данных 48

2.1. Нереляционная модель 48

2.2. Реляционная модель 76

2.3. Сравнение моделей 103

3. Разработанное приложение 106

3.1. Краткое описание 106

3.2. Используемые технологии 107

3.3. Снимки экранов приложения 108

4. ВЫВОДЫ 111

4.1. Достигнутые результаты 111

4.2. Недостатки и пути для улучшения полученного решения 111

4.3. Будущее развитие решения 111

5. список ЛИТЕРАТУРы 111

6. ПРИЛОЖЕНИЯ 111

6.1. Документация по сборке и развертыванию приложения 111

# Введение

Цель работы – создать высокопроизводительное и удобное решение для сервиса текстовых фрилансеров.

Принято решение разработать веб-приложение, которое позволит заказчикам создавать заказы, искать фрилансеров для выполнения этих заказов, оценивать их выполнение и расплачиваться. Фрилансеры, в свою очередь, смогут брать заказы, выполнять их поэтапно и оценивать заказчика.

# 1. СценариИ использования

Перед началом разработки мы проанализировали поведение будущих пользователей. Ниже представлены основные сценарии использования, отражающие реальные задачи и типовые пути их решения. Это позволяет создать удобный и понятный интерфейс.

## 1.1 Макеты UI

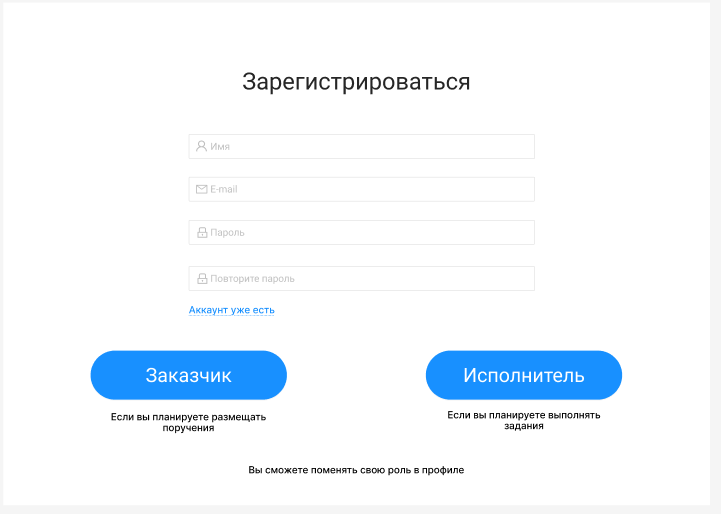


Рисунок 1.1 – Экран регистрации нового пользователя

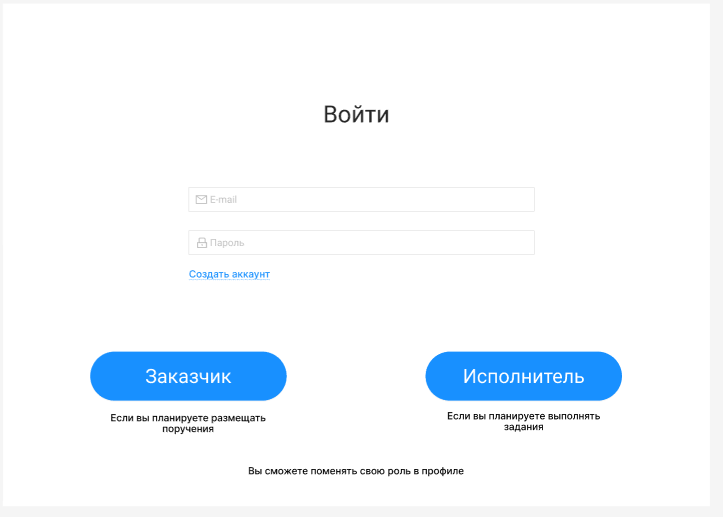


Рисунок 1.2 – Экран авторизации

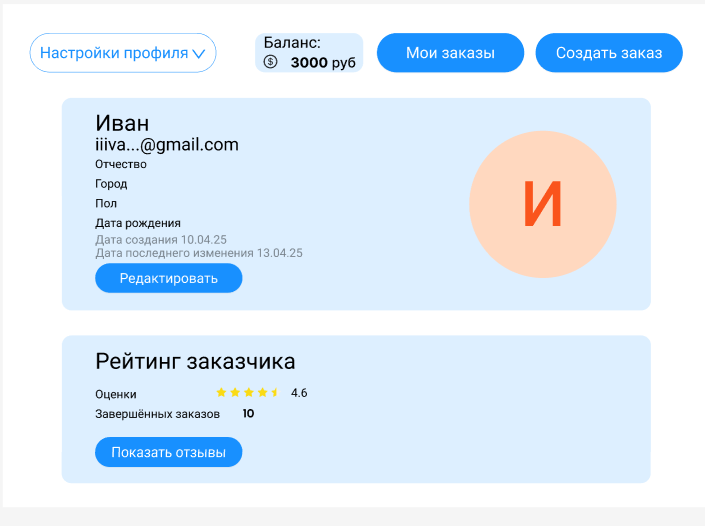


Рисунок 1.3 – Приватный профиль заказчика

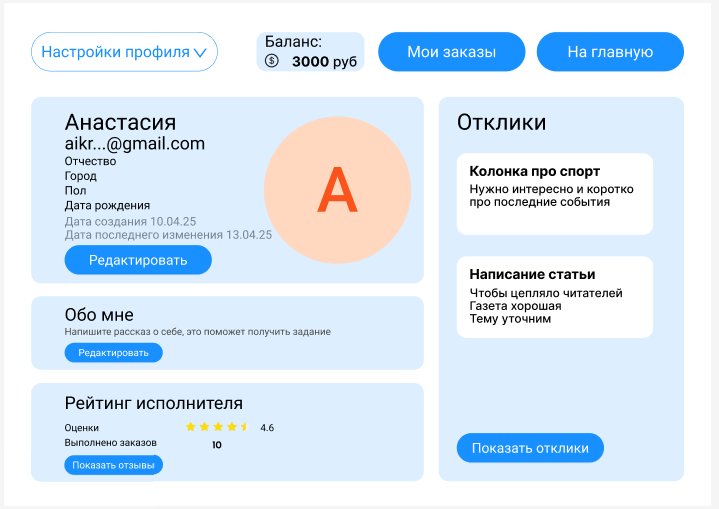


Рисунок 1.4 – Приватный профиль фрилансера

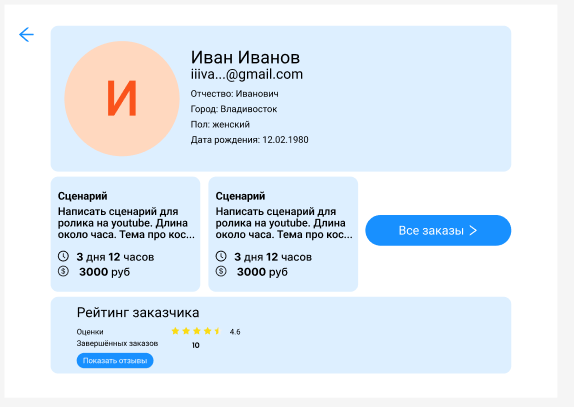


Рисунок 1.5 – Публичный профиль заказчика

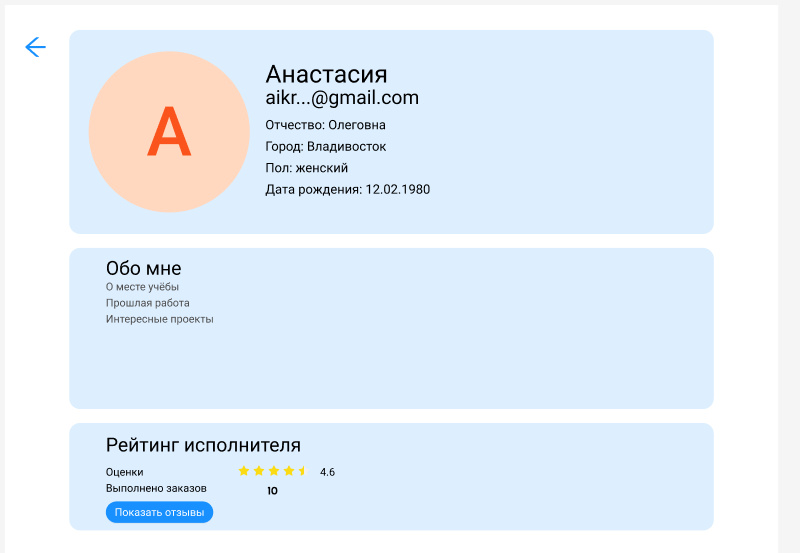


Рисунок 1.6 – Публичный профиль исполнителя

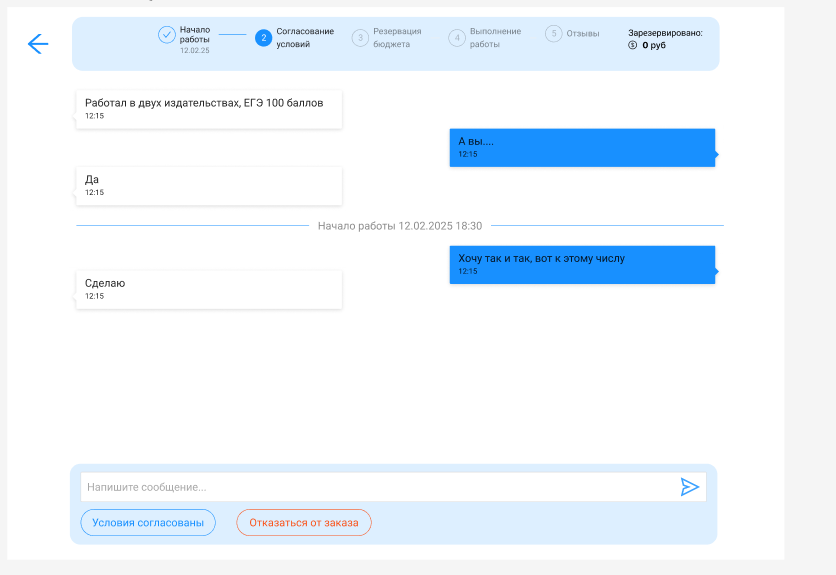


Рисунок 1.7 – Согласование условий от лица заказчика

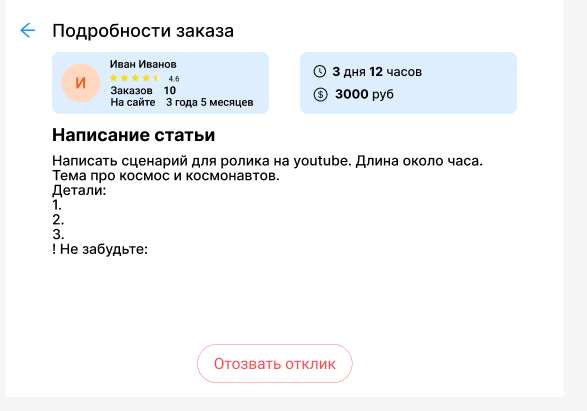


Рисунок 1.8 – Отклик на заказ от лица фрилансера

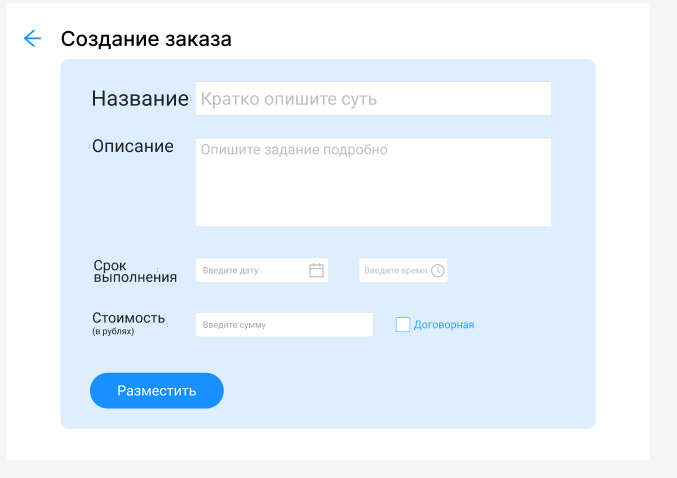


Рисунок 1.9 – Создание заказа заказчиком

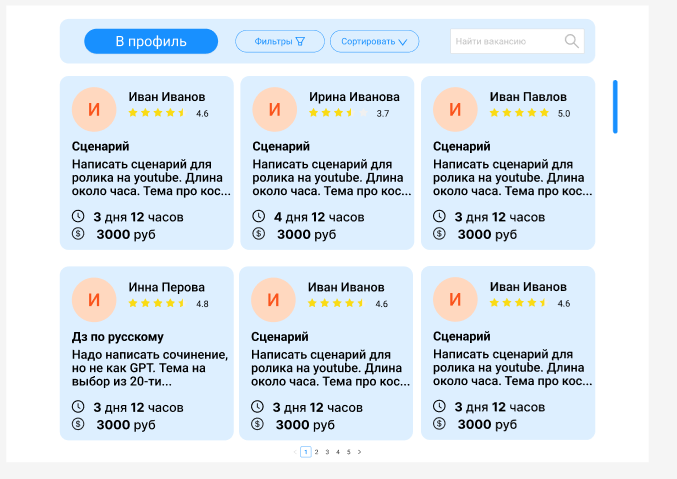


Рисунок 1.10 – Главная страница от лица фрилансера

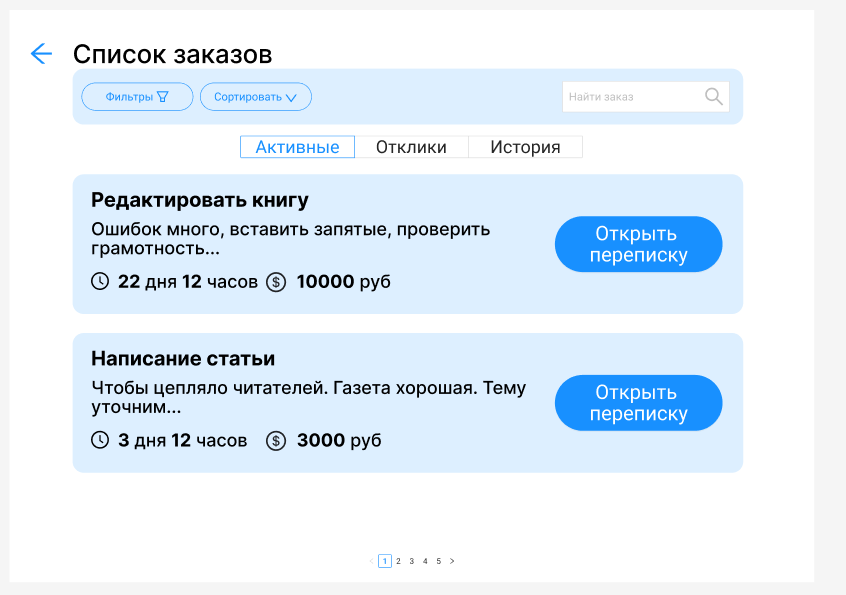


Рисунок 1.11 – Активные заказы исполнителя

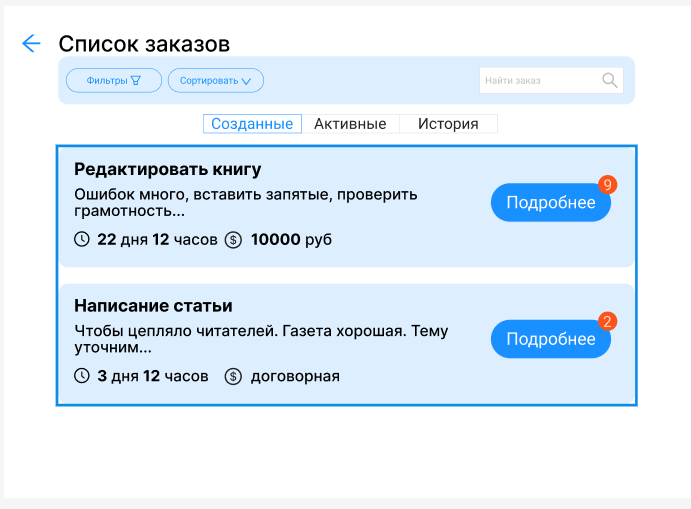


Рисунок 1.12 – Список созданных заказов заказчика

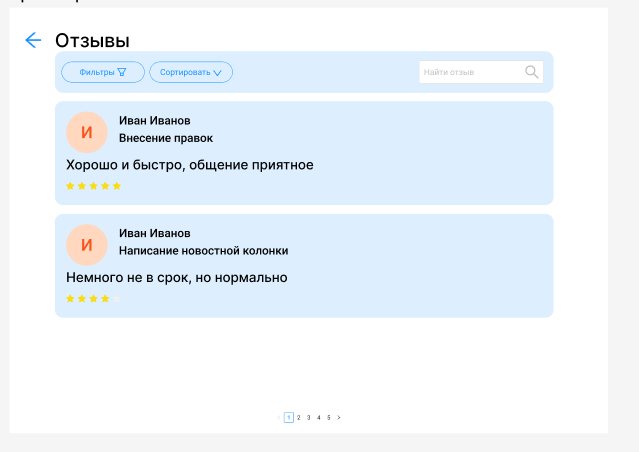


Рисунок 1.13 – Просмотр отзывов

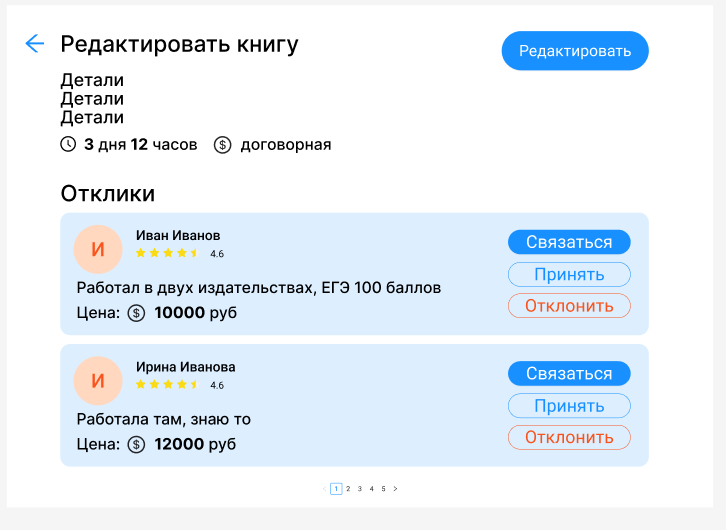


Рисунок 1.14 – Просмотр заказа от лица заказчика, этап выбора исполнителя

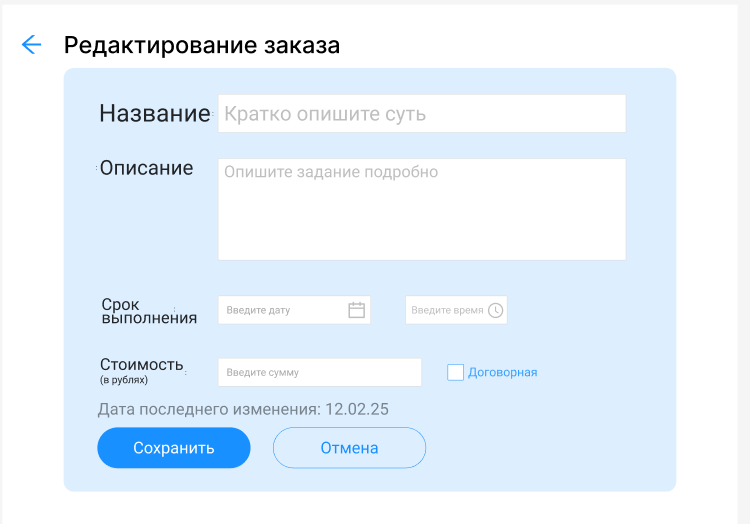


Рисунок 1.15 – Редактирование заказа заказчиком

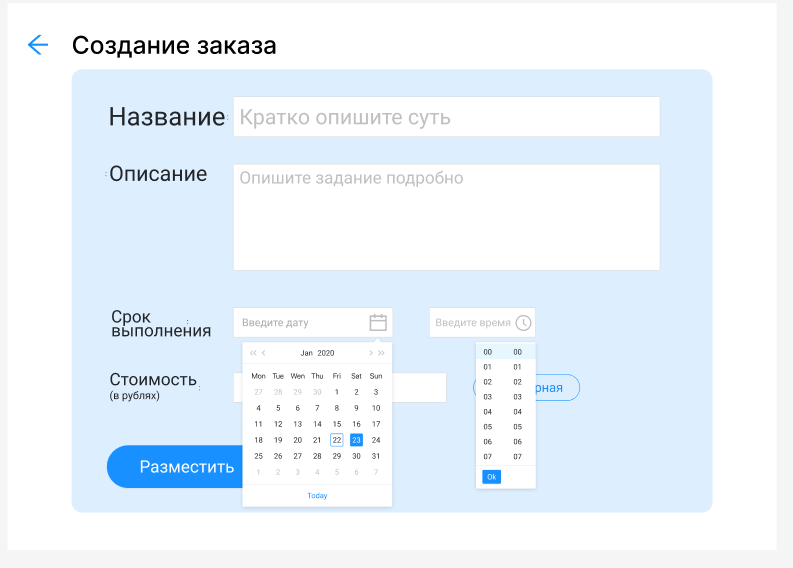


Рисунок 1.16 – Создание заказа заказчиком

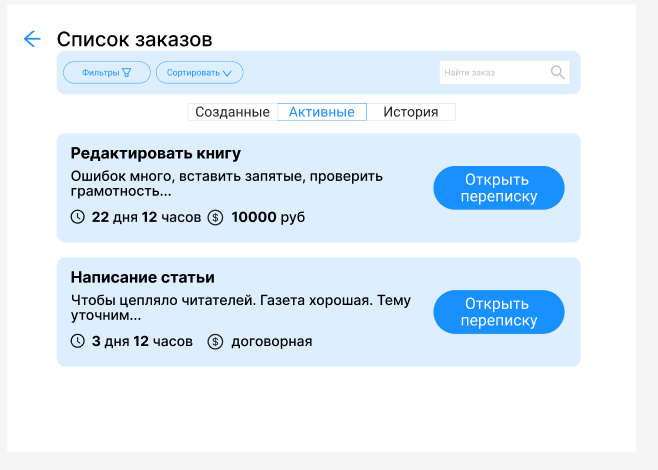


Рисунок 1.17 – Активные заказы заказчика

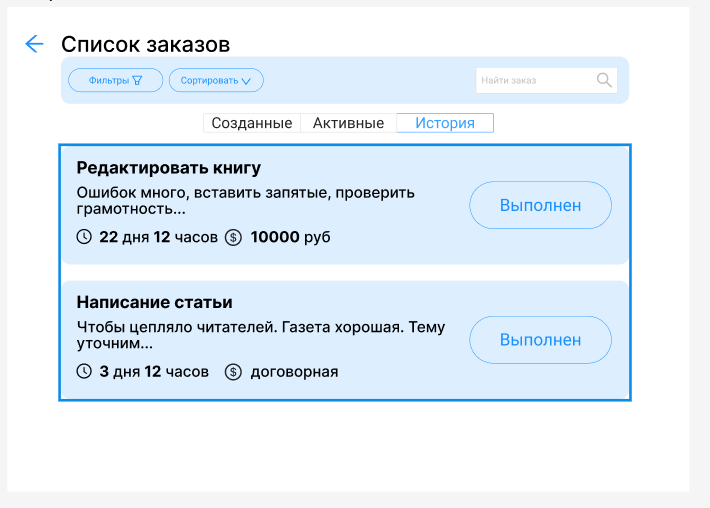


Рисунок 1.18 – История заказов заказчика

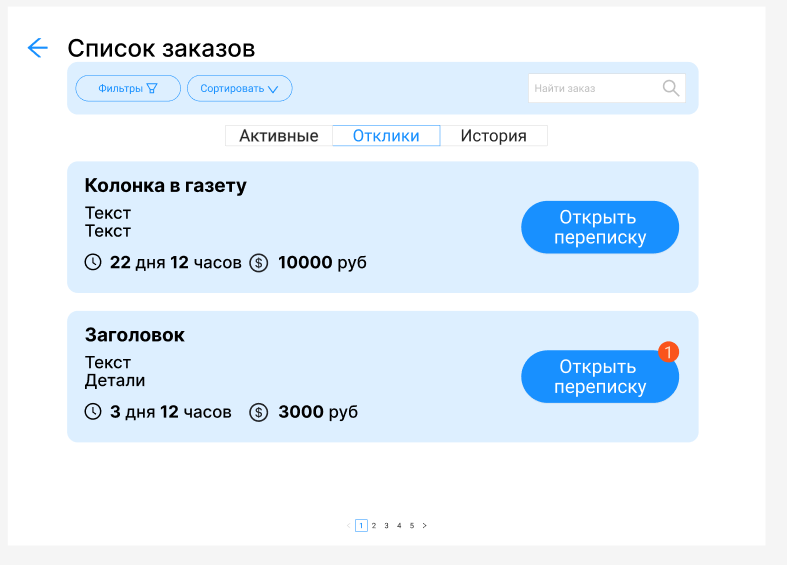


Рисунок 1.19 – Отклики исполнителя

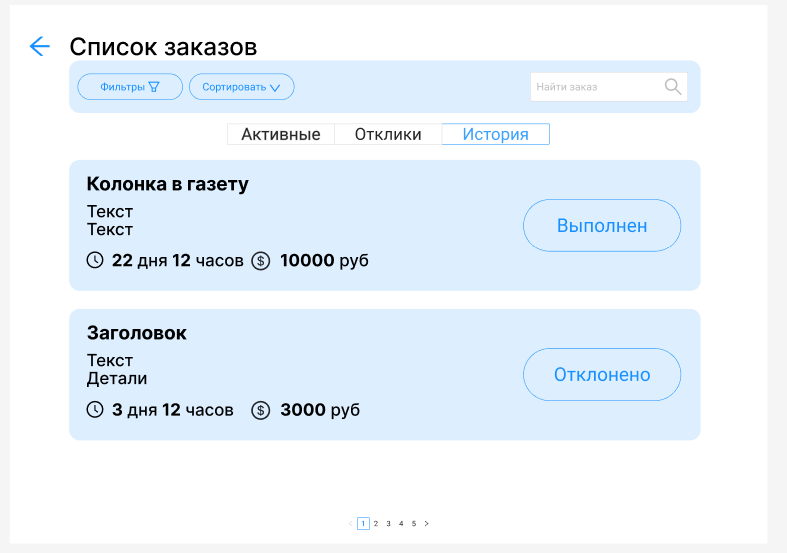


Рисунок 1.20 – История заказов исполнителя

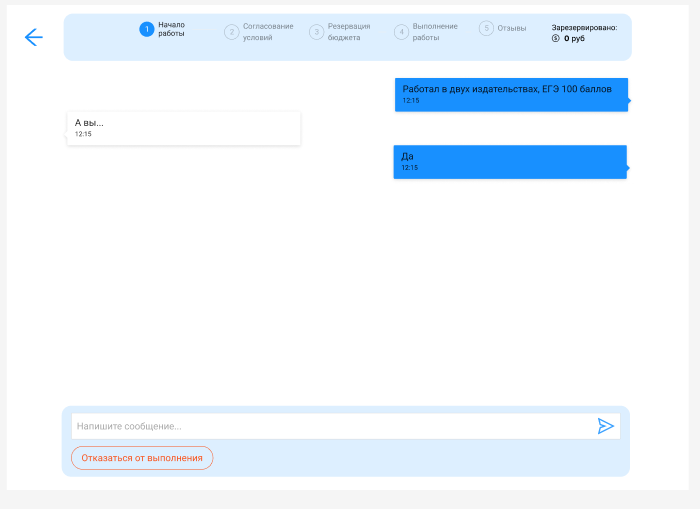


Рисунок 1.21 – Переписка от лица исполнителя, этап «Начало работы»

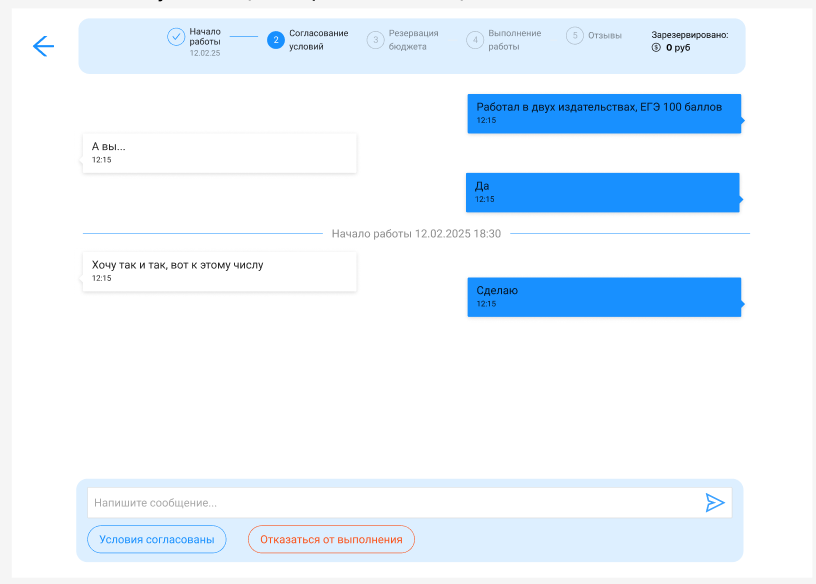


Рисунок 1.22 – Переписка от лица исполнителя, этап «Согласование условий»

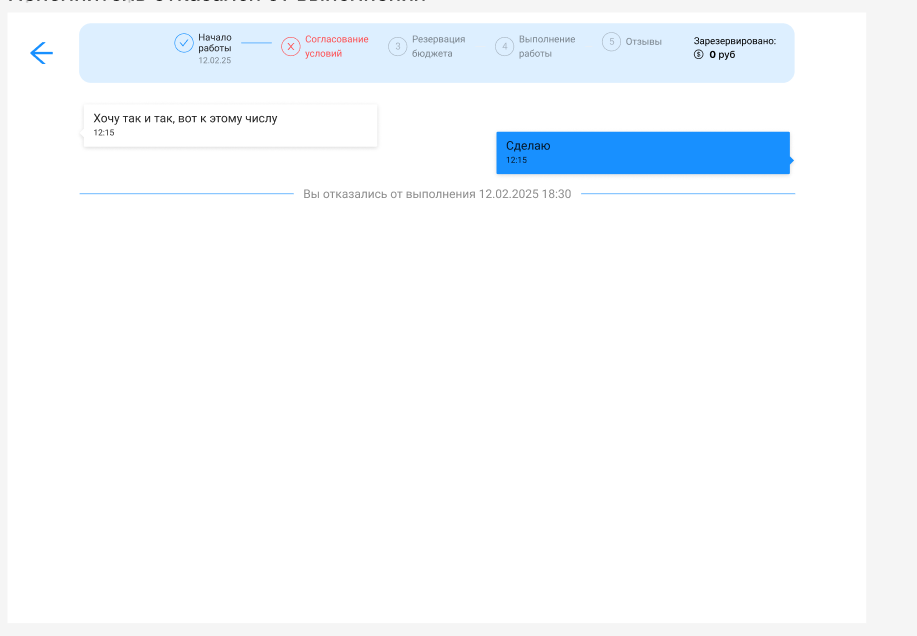


Рисунок 1.23 – Переписка от лица исполнителя, исполнитель отказался от выполнения

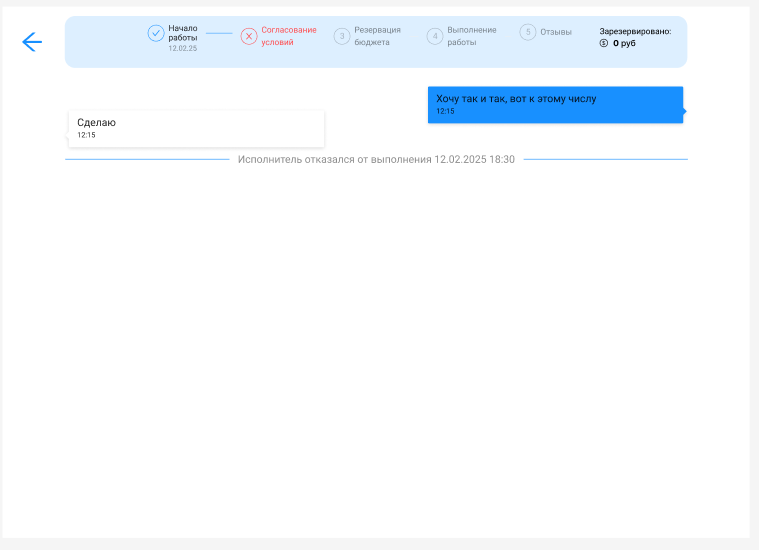


Рисунок 1.24 – Переписка от лица заказчика, исполнитель отказался от выполнения

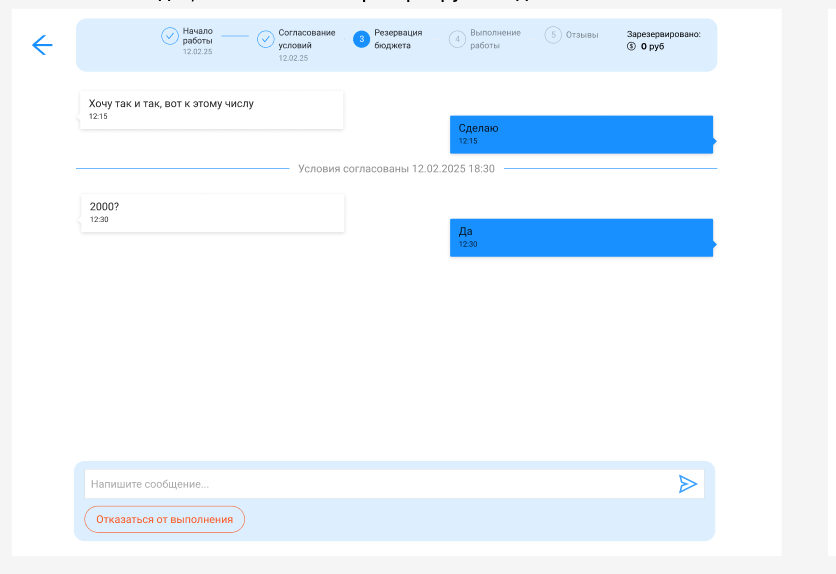


Рисунок 1.25 – Переписка от лица исполнителя, ожидание резервирования бюджета

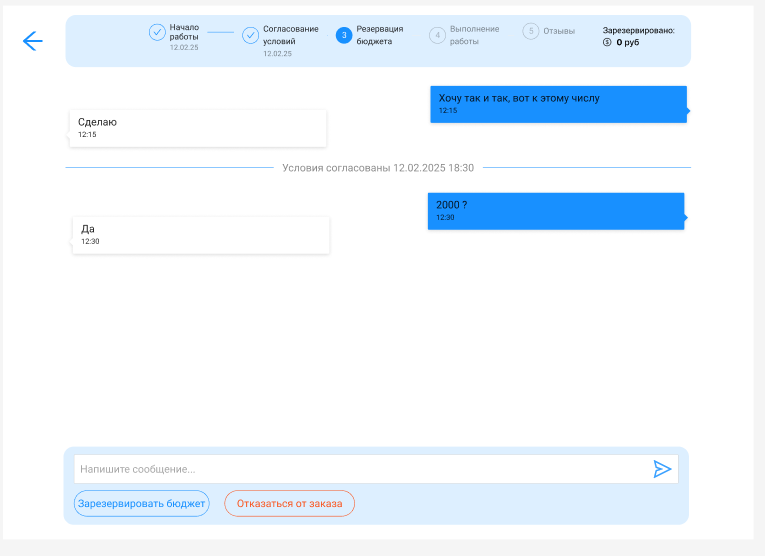


Рисунок 1.26 – Переписка от лица заказчика, резервирование бюджета

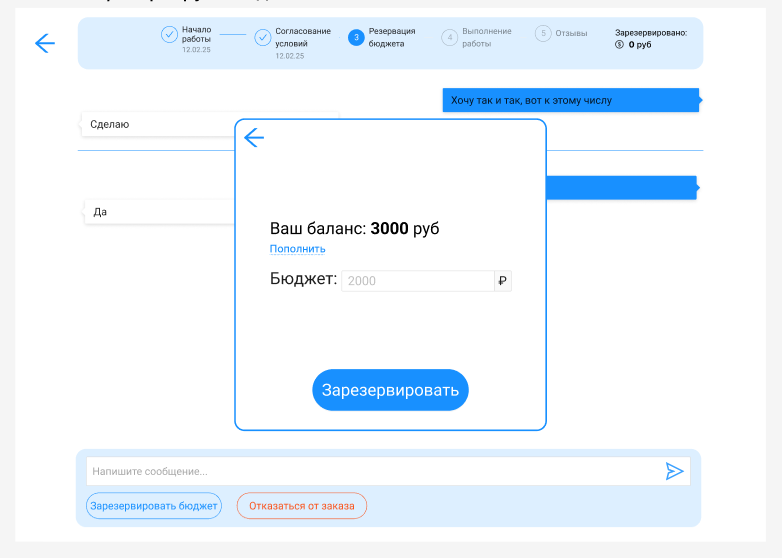


Рисунок 1.27 – Резервирование бюджета заказчиком

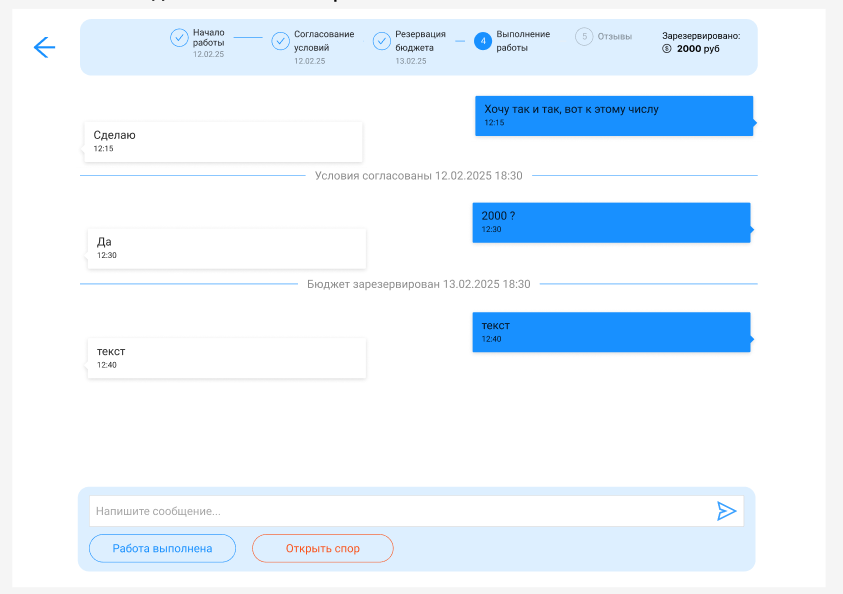


Рисунок 1.28 – Переписка от лица заказчика, ожидание выполненной работы исполнителем

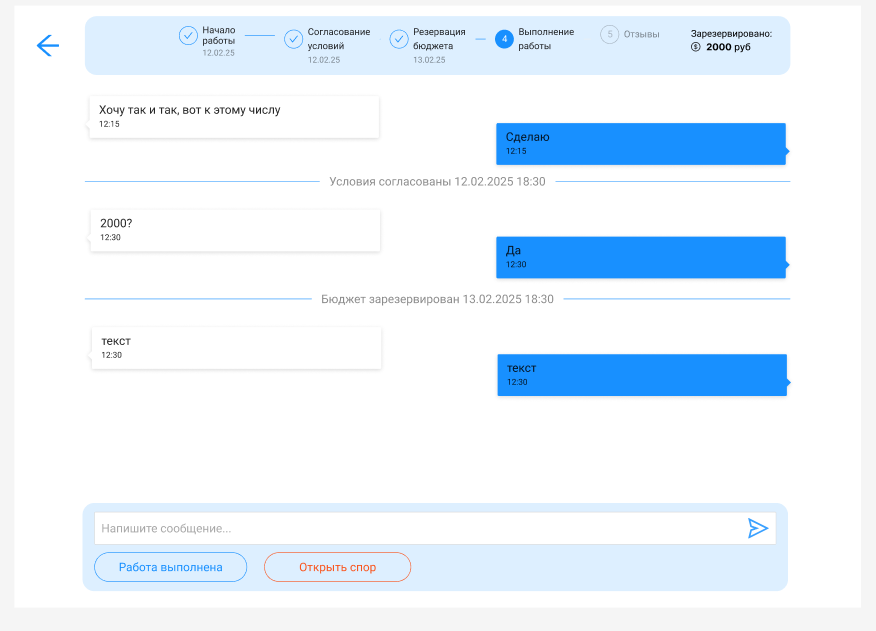


Рисунок 1.29 – Переписка от лица исполнителя, этап «Выполнение работы»

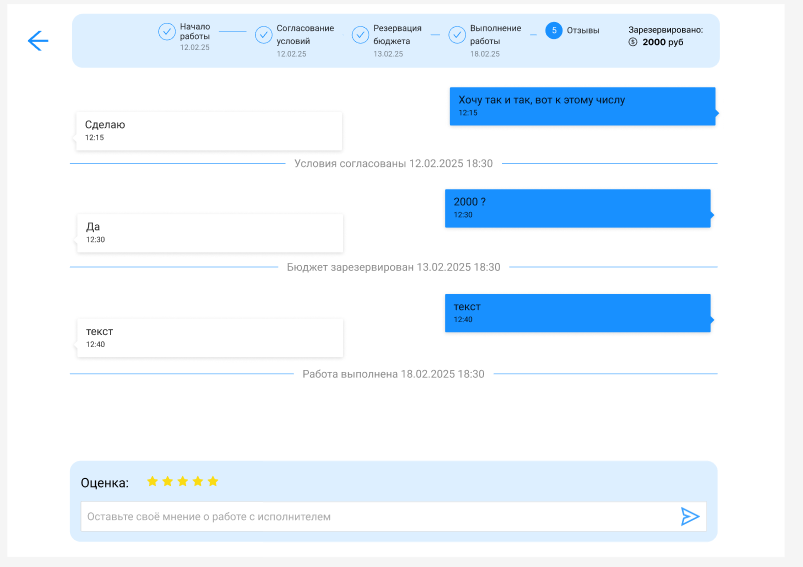


Рисунок 1.30 – Заказчик оставляет отзыв

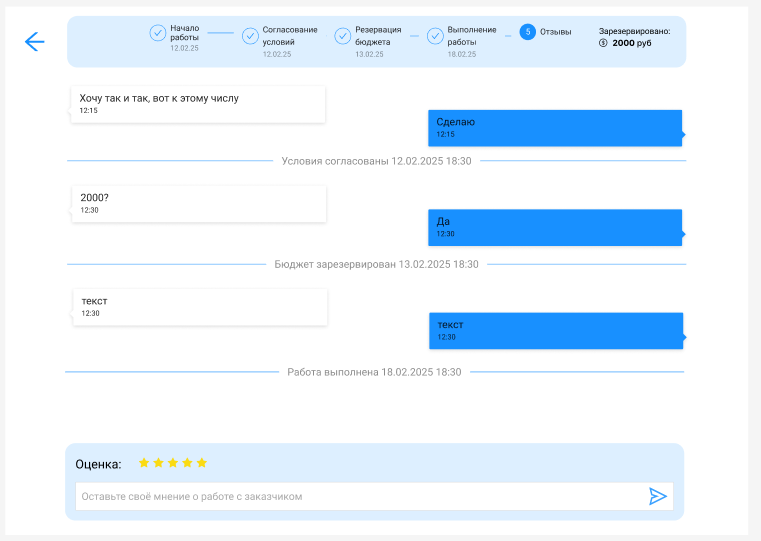


Рисунок 1.31 – Исполнитель оставляет отзыв

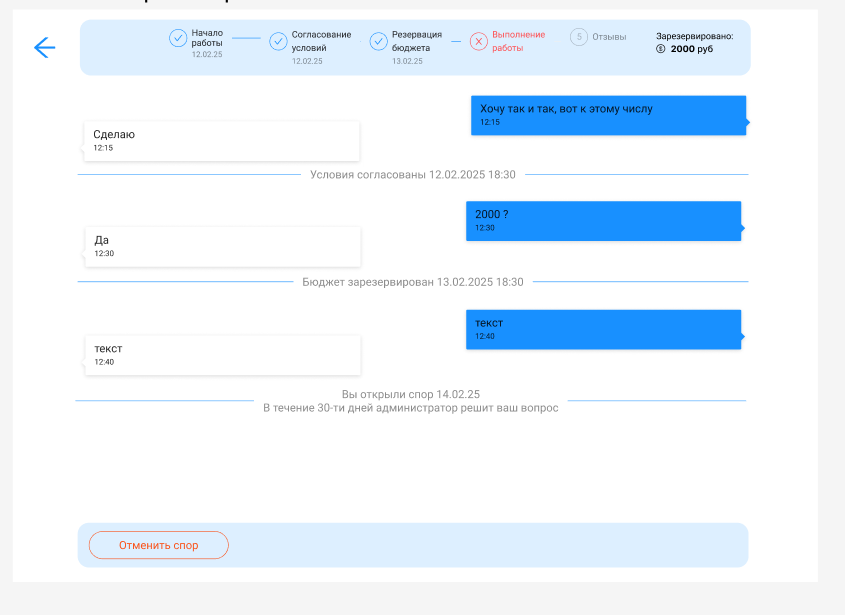


Рисунок 1.32 – Заказчик открыл спор (от лица заказчика)

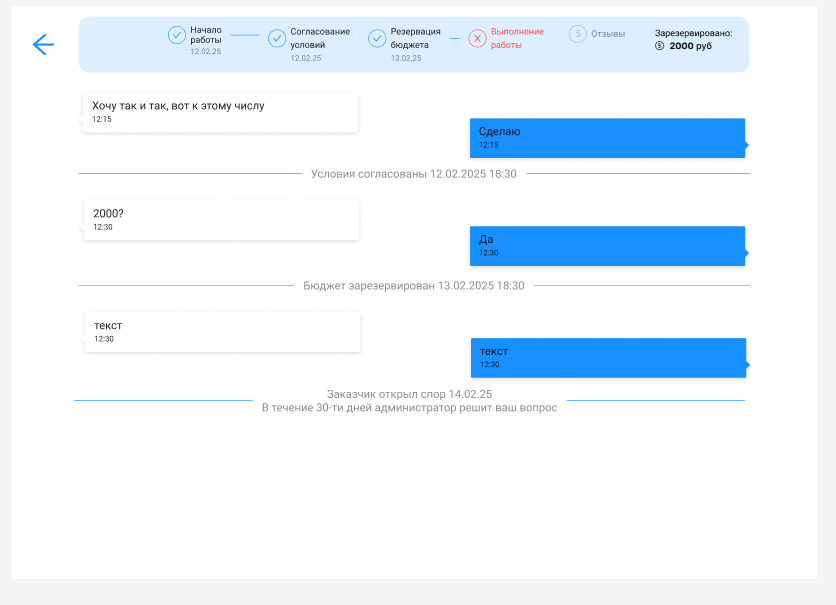


Рисунок 1.33 – Заказчик открыл спор (от лица исполнителя)

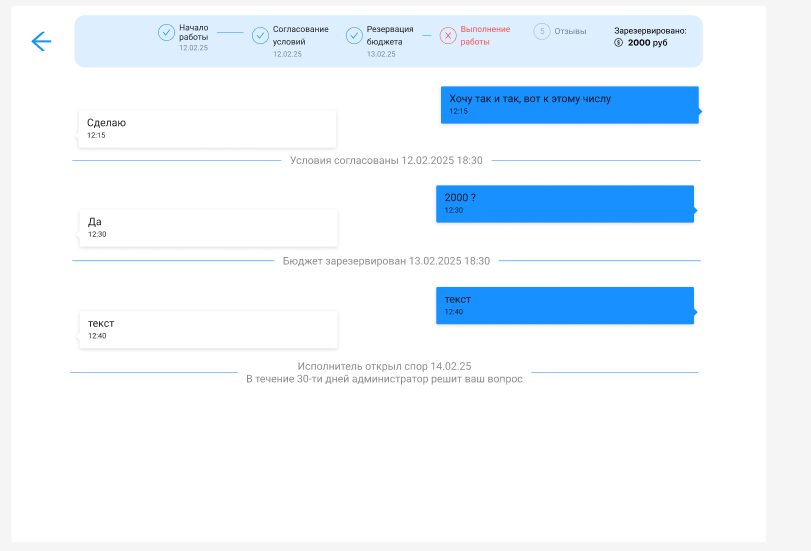


Рисунок 1.34 – Исполнитель открыл спор (от лица заказчика)

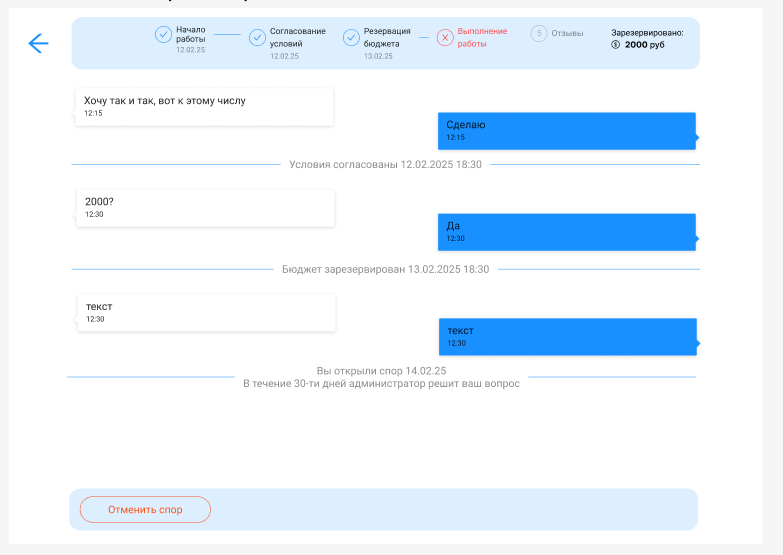


Рисунок 1.35 – Исполнитель открыл спор (от лица исполнителя)

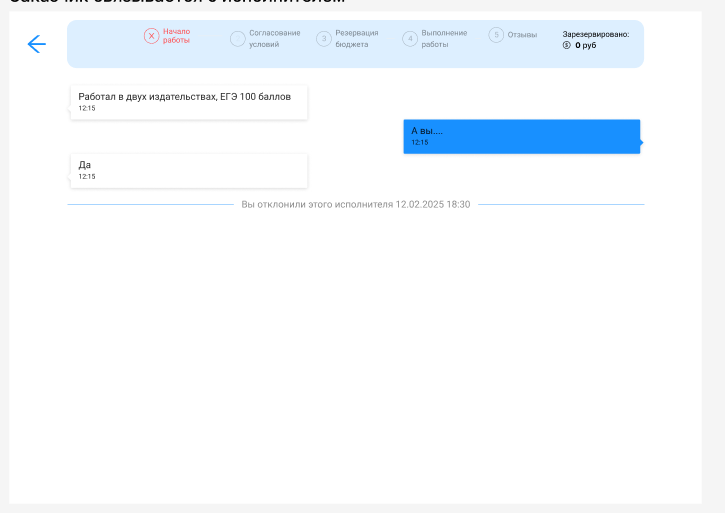


Рисунок 1.36 – Переписка со стороны заказчика, отказ от исполнителя

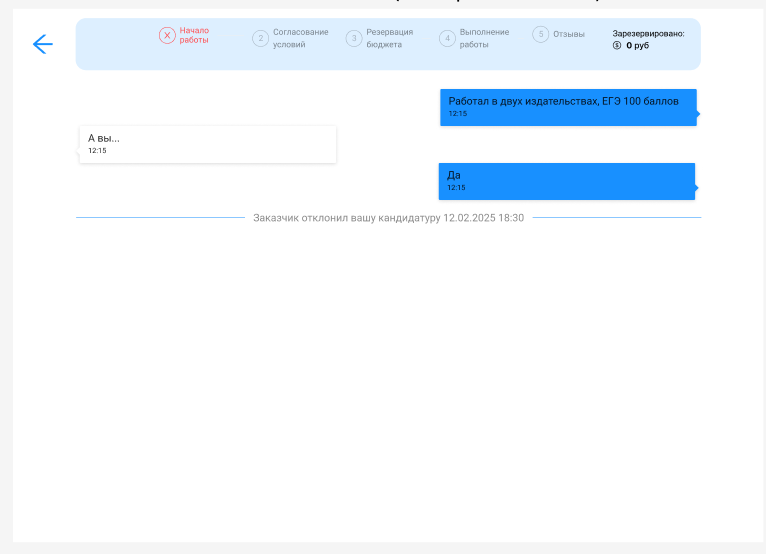


Рисунок 1.37 – Переписка от лица исполнителя, отказ со стороны заказчика

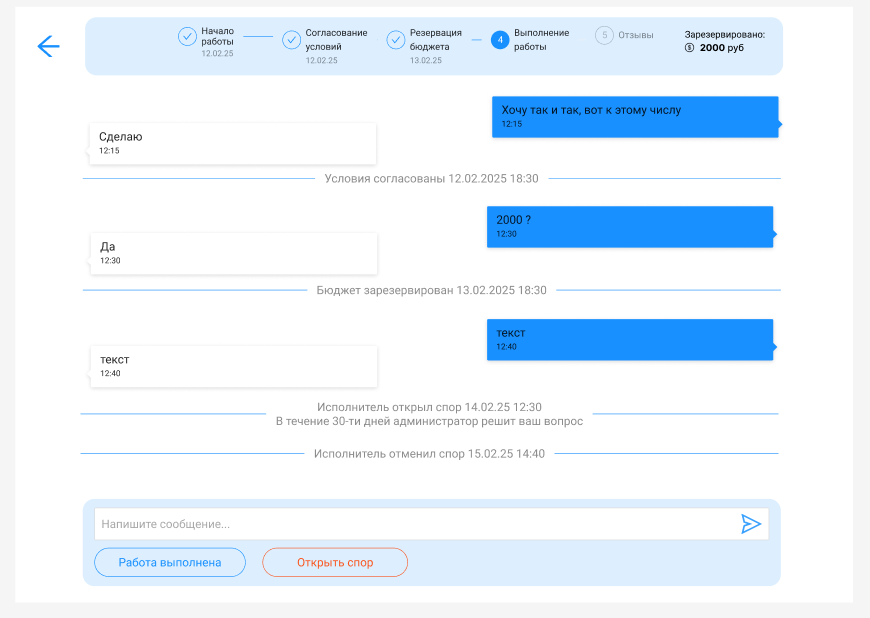


Рисунок 1.38 – Исполнитель отменил спор (переписка от лица заказчика)

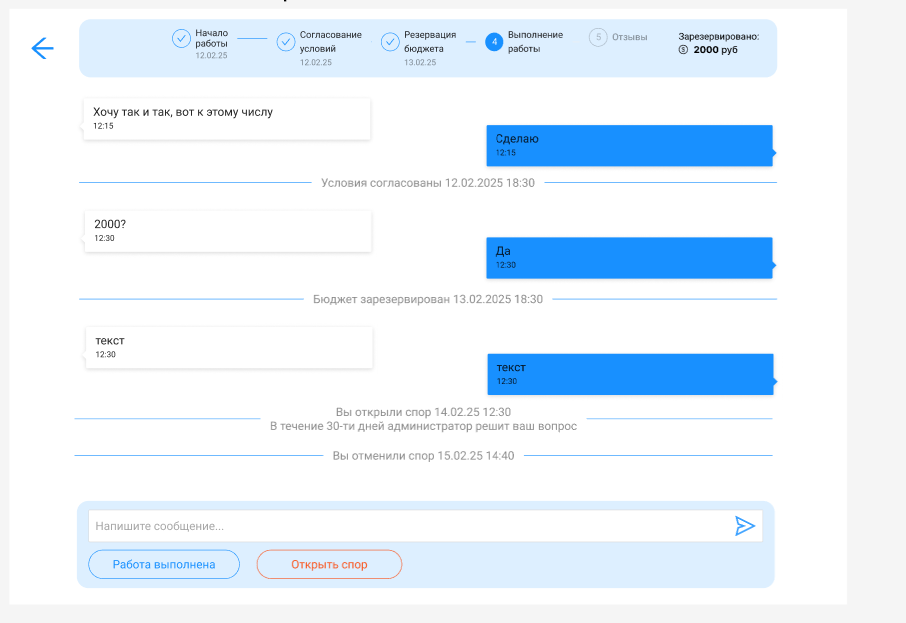


Рисунок 1.39 – Исполнитель отменил спор (переписка от лица исполнителя)

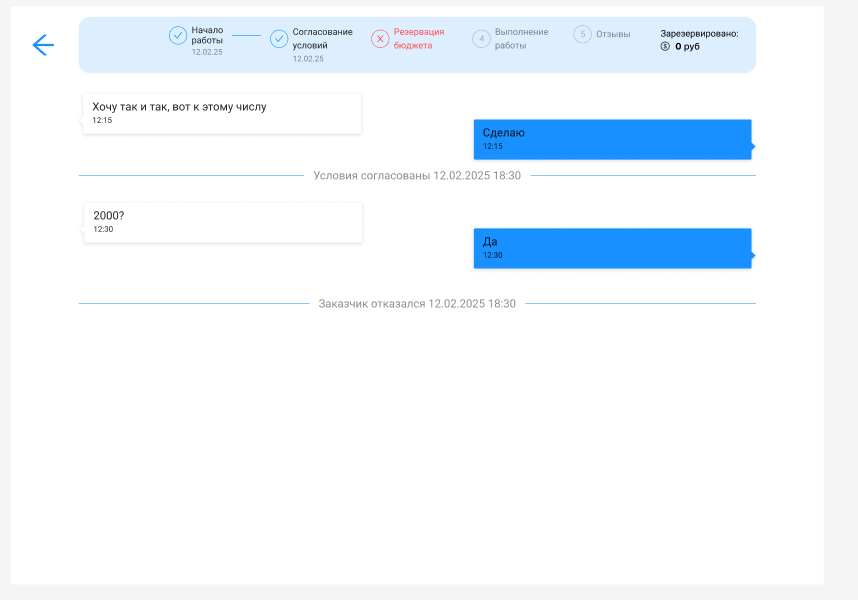


Рисунок 1.40 – Заказчика отказался от заказа (переписка от лица исполнителя)

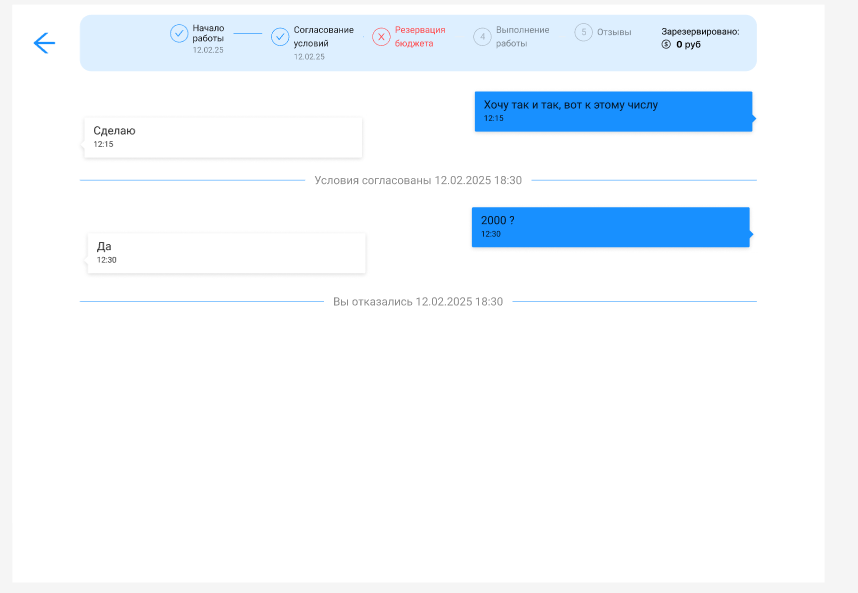


Рисунок 1.41 – Заказчика отказался от заказа (переписка от лица исполнителя)

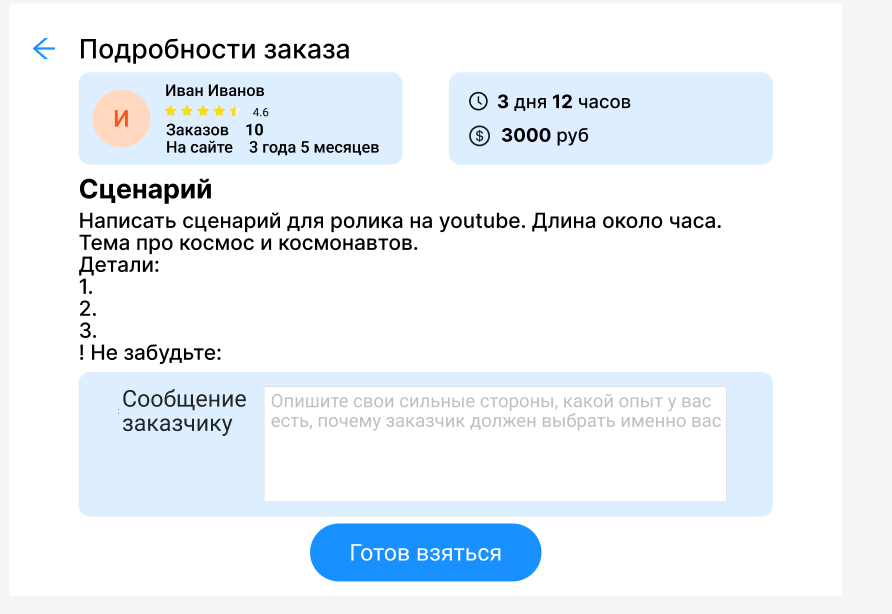


Рисунок 1.42 – Страница заказа от лица исполнителя до отклика

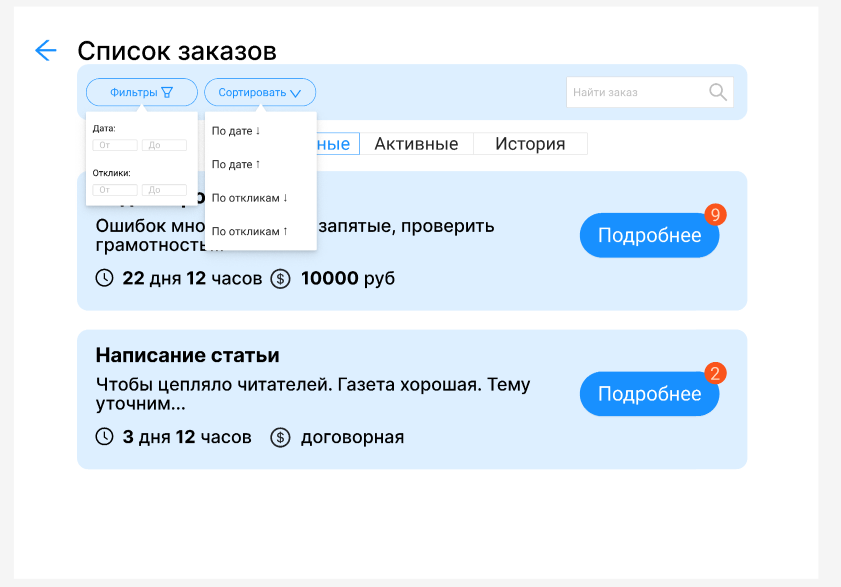


Рисунок 1.43 – Выпадающие списки на странице заказов заказчика

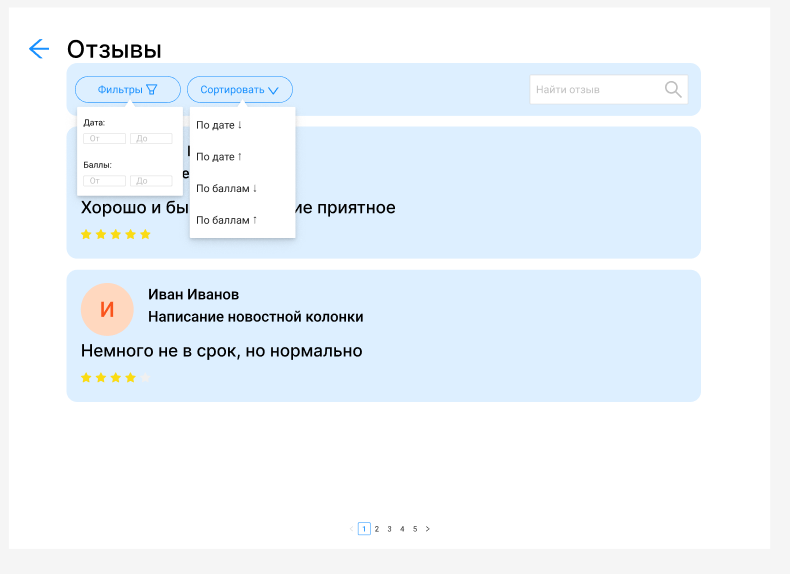


Рисунок 1.44 – Выпадающие списки на странице отзывов

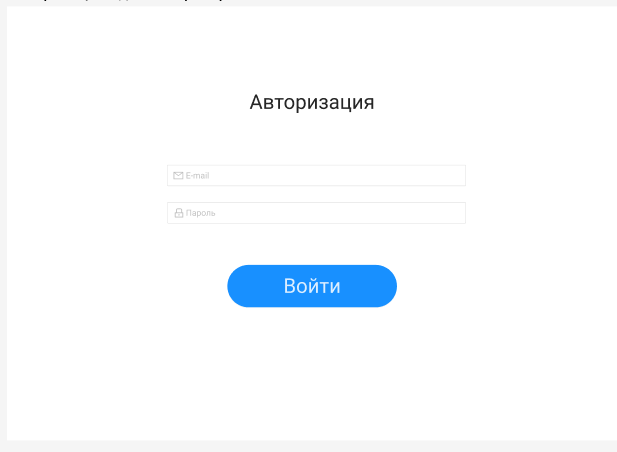


Рисунок 1.45 – Авторизация администратора

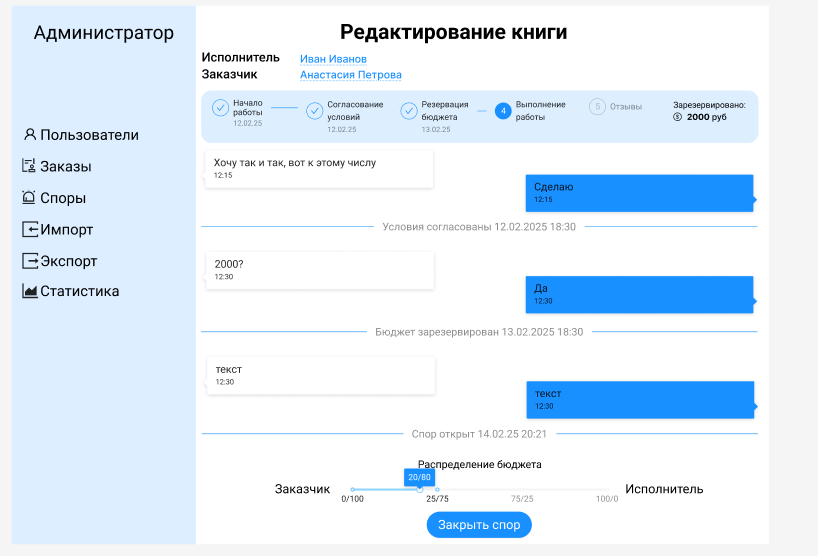


Рисунок 1.46 – Просмотр спора администратором

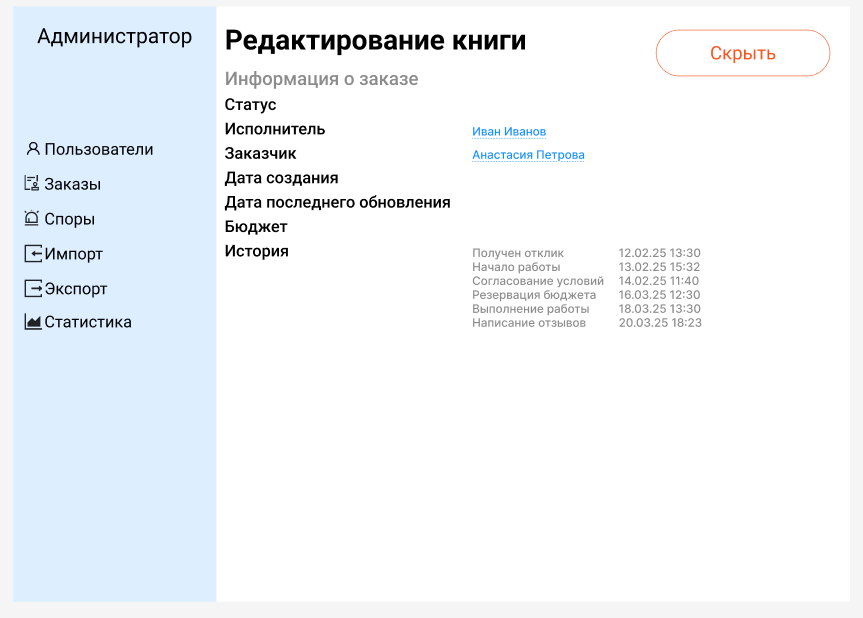


Рисунок 1.47 – Просмотр заказа администратором

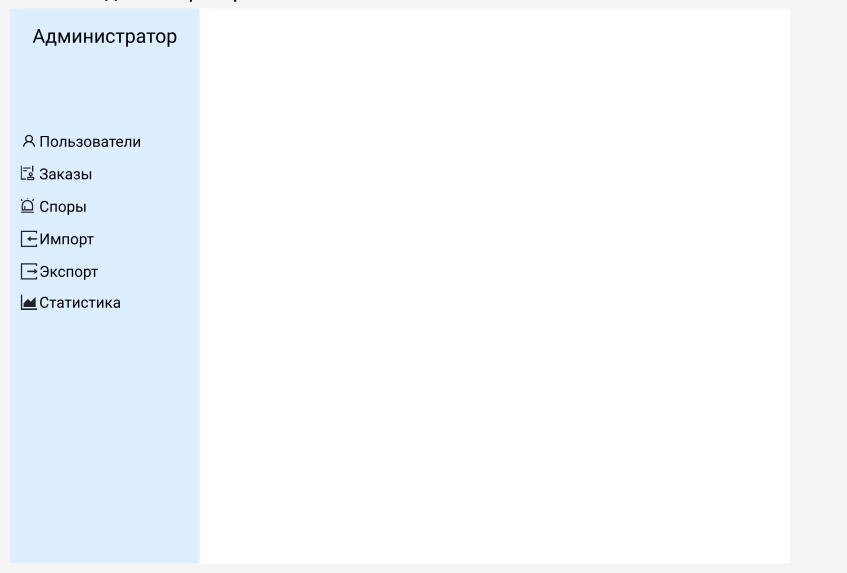


Рисунок 1.48 – Главная администратора

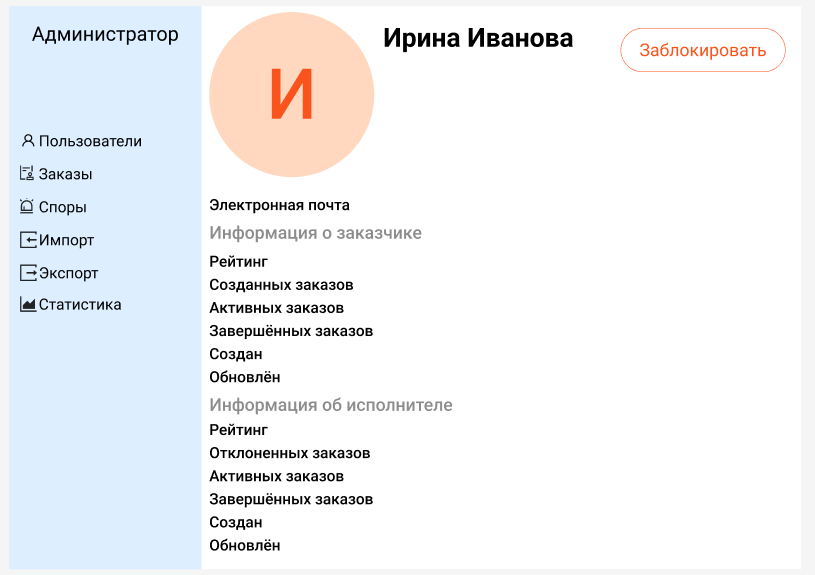


Рисунок 1.49 – Просмотр пользователя администратором

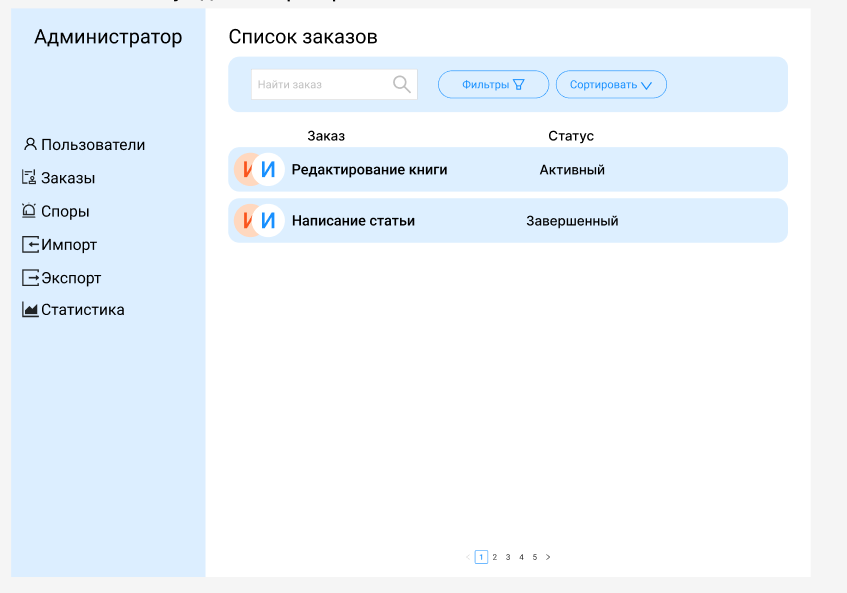


Рисунок 1.50 – Просмотр списка заказов администратором

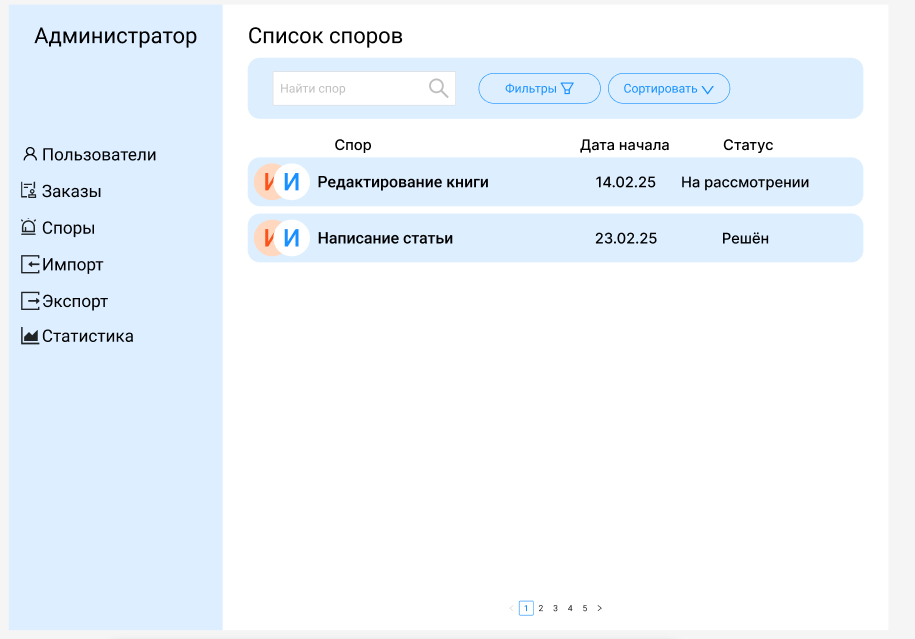


Рисунок 1.51 – Просмотр списка споров администратором

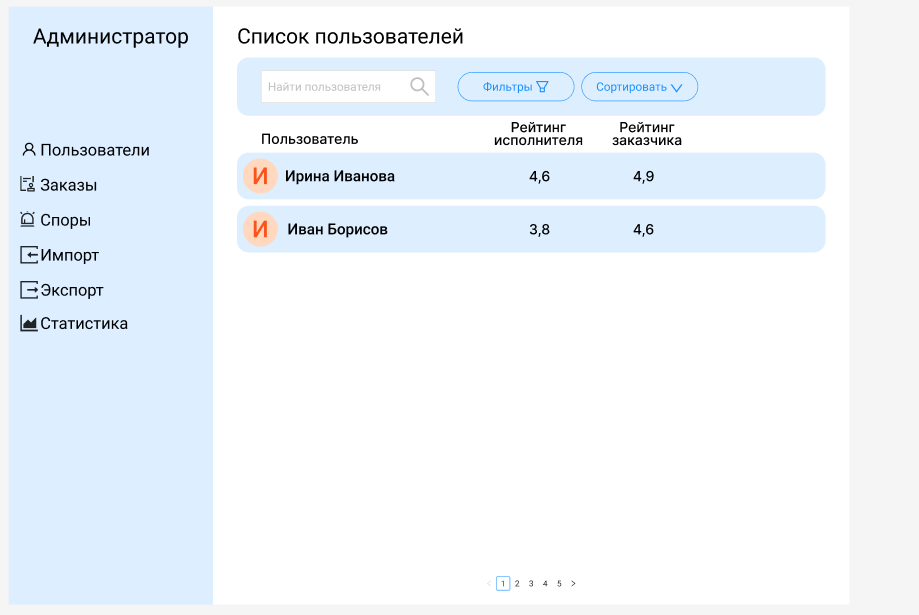


Рисунок 1.52 – Просмотр списка пользователей администратором

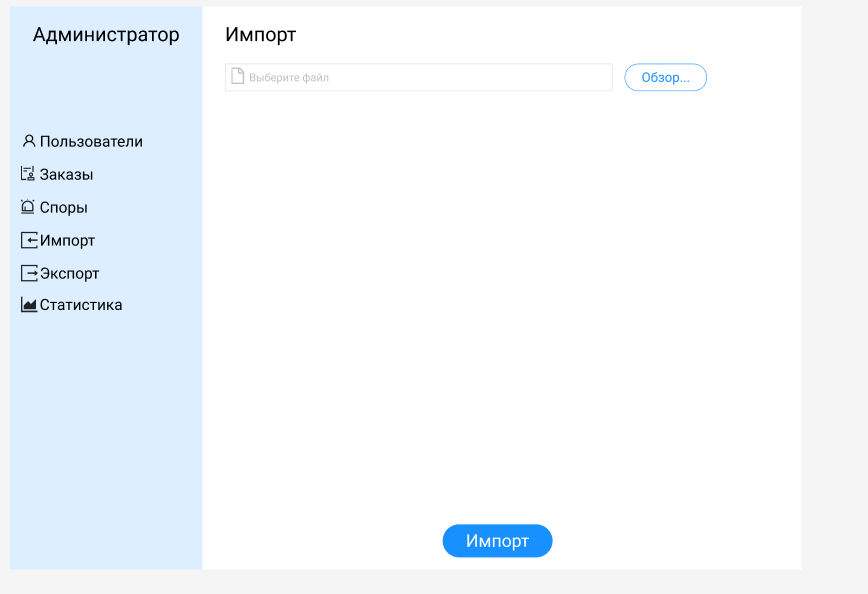


Рисунок 1.53 – Импорт данных, выполняемый администратором

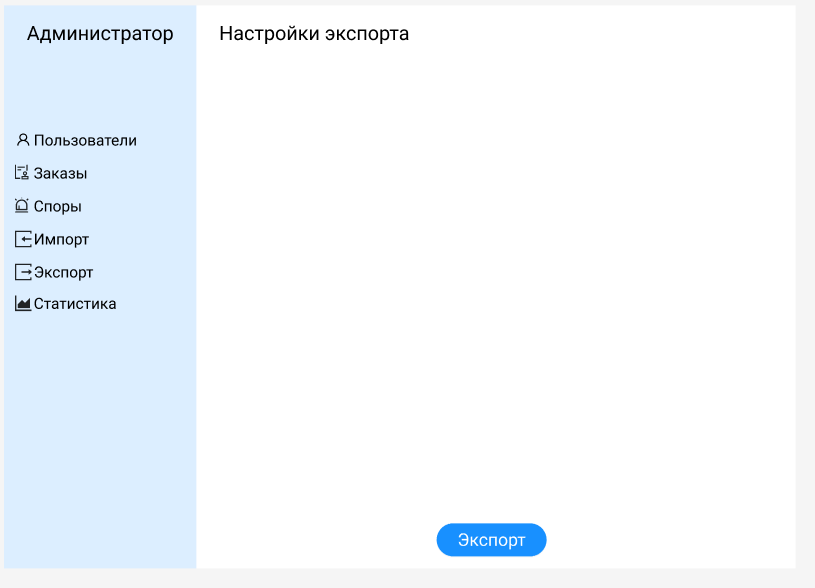


Рисунок 1.54 – Экспорт данных, выполняемый администратором

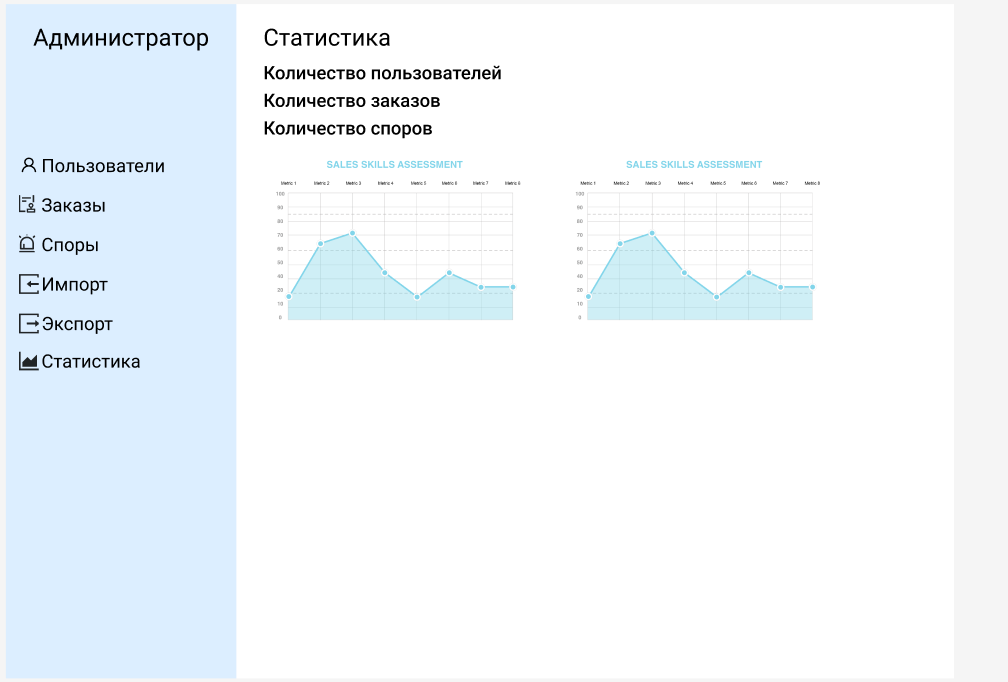


Рисунок 1.55 – Просмотр статистики администратором

## 1.2 Описание сценариев использования

1. **Регистрация пользователя**

Действующее лицо: пользователь (заказчик/исполнитель)

Основной сценарий:

1. Пользователь заполняет форму, указывая адрес электронной почты, публично отображаемое имя пользователя, пароль.
2. Пользователь выбирает роль “Заказчик” или “Исполнитель”.
3. Сайт проверяет данные пользователя.
4. Если всё заполнено верно, переход на страницу "Приватный профиль заказчика/исполнителя".

Альтернативный сценарий:

* Данные заполнены некорректно: пользователю показывается ошибка.
* Пользователь передумал регистрироваться и хочет авторизоваться: нажимает на “Аккаунт уже есть” и попадает на страницу авторизации.

## Авторизация пользователя

Действующее лицо: пользователь

Основной сценарий:

1. Пользователь заполняет форму, указывая адрес электронной почты и пароль.
2. Пользователь выбирает роль “Заказчик” или “Исполнитель”.
3. Сайт проверяет данные пользователя.
4. Если всё заполнено верно, переход на страницу "Приватный профиль заказчика/исполнителя".

Альтернативный сценарий:

* Данные заполнены некорректно: пользователю показывается ошибка.
* Пользователь решил зарегистрироваться: нажимает на “Создать аккаунт” и попадает на страницу регистрации.

## Редактирование профиля

Действующее лицо: пользователь

Основной сценарий:

1. Пользователь находится на странице "Приватный профиль заказчика/исполнителя".
2. Пользователь нажимает на кнопку “Редактировать” в нужном блоке.
3. Сайт делает поля в блоке интерактивными (превращает в поля ввода), кнопка “Редактировать” заменяется на “Сохранить изменения”.
4. Пользователь редактирует данные.
5. Сайт проверяет корректность новых данных и сохраняет их, если они корректны, либо показывает ошибку.
6. Сайт возвращается к исходному состоянию, но с изменённой информацией.

Альтернативный сценарий

* Новые данные некорректны: сайт показывает ошибку.
* Пользователь передумал: нажимает на кнопку отмена. Сайт возвращается к исходному состоянию, не применяя изменения.
* Пользователь перезагрузил страницу во время редактирования: изменения не применяются.

## Смена роли

Действующее лицо: пользователь

Основной сценарий:

1. Пользователь находится на странице "Приватный профиль заказчика/исполнителя".
2. Пользователь нажимает на кнопку “Настройки профиля”.
3. Появляется выпадающее меню с двумя кнопками “Заказчик” и “Исполнитель”. Активная роль выделена цветом.
4. Пользователь нажимает на роль, которая не выделена цветом.
5. Переход на страницу "Приватный профиль исполнителя" (если до этого был заказчиком) или "Приватный профиль заказчика" (если до этого был исполнителем).

Альтернативный сценарий

* Пользователь нажимает на текущую роль (которая выделена цветом): ничего не происходит.

## Заказчик публикует новый заказ

Действующее лицо: заказчик

Основной сценарий:

1. Заказчик находится на странице "Приватный профиль заказчика".
2. Заказчик нажимает на кнопку “Создать заказ”.
3. Переход на страницу "Создание заказа заказчиком".
4. Заказчик заполняет поля “Название” (текстовое поле), “Описание” (текстовое поле), “Срок выполнения” (дата и время) и “Стоимость” (число больше нуля, либо checkbox “по договорённости”).
5. Заказчик нажимает на кнопку “Разместить”.
6. Проводится валидация данных.
7. Если всё верно, создаётся заказ.
8. Переход на страницу "Просмотр заказа заказчиком (выбор исполнителя)".

Альтернативный сценарий

* Введены некорректные данные: заказчику показывается ошибка, заказ не создаётся.
* Заказчик может нажать на checkbox “По договорённости”, тогда цена не будет указана явно, поле ввода блокируется до повторного нажатия.
* Пользователь закрывает страницу до нажатия на кнопку “Разместить”: заказ не создаётся.
* Пользователь перезагружает страницу: информация обнуляется.

## Заказчик редактирует заказ

Действующее лицо: заказчик

Основной сценарий:

1. Заказчик находится на странице "Просмотр заказа заказчиком (выбор исполнителя)".
2. Заказчик нажимает на кнопку “Редактировать”.
3. Появляются поля для ввода (такие же как в форме создания заказа). Появляются кнопки “Сохранить”, “Отмена”.
4. В поля подставляется текущая информация.
5. Заказчик редактирует её и нажимает на одну из кнопок “Сохранить” или “Отмена”.
6. Введённые данные проверяются на корректность.
7. Если нажата кнопка “Сохранить”, изменения сохраняются, состояние страницы возвращается к исходному (но с новыми данными).
8. Если нажата кнопка “Отмена”, изменения не применяются, состояние страницы возвращается к исходному.

Альтернативный сценарий

* Новые данные некорректны: пользователю показывается ошибка.
* Пользователь перезагружает страницу или закрывает её: изменения не применяются.

## Стороны согласовали условия и работа началась

Действующее лицо: заказчик, исполнитель

Основной сценарий:

1. Заказчик и исполнитель согласовывают условия выполнения заказа в чате.
2. Заказчик и исполнитель нажимают кнопку на страницах "Согласование условий (от лица заказчика)" и "Согласование условий (от лица исполнителя)" “Условия согласованы”.
3. Статус заказа переводится в “Условия согласованы”.

Альтернативный сценарий

* Исполнитель не подтверждает условия: статус “Начало работы”.
* Заказчик не подтверждает условия: статус “Начало работы”.

## Стороны общаются в процессе выполнения заказа

Действующее лицо: заказчик, исполнитель

Основной сценарий:

1. Пользователи находятся на странице чата.
2. Пользователь (заказчик или исполнитель) пишет сообщение в текстовом поле.
3. Пользователь нажимает на кнопку отправки.
4. Сообщение отправляется второму участнику переписки.
5. Второй участник переписки получает сообщение без перезагрузки страницы.

Альтернативный сценарий

* Пользователь закрыл страницу во время написания сообщения: текст не сохраняется.
* Пользователь ничего не ввёл, но нажал на кнопку отправки: сообщение не отправляется.
* Пользователь ввёл слишком большое сообщение: автоматически обрезается до максимально допустимого размера и отправляется.
* Пользователь ввёл пробелы (или другие невидимые символы) в конце строки: пробелы обрезаются.

## Завершение заказа

Действующее лицо: заказчик, исполнитель

Основной сценарий:

1. Заказчик находится на странице "Заказчик ожидает выполнения работы исполнителем", исполнитель находится на странице "Исполнитель выполняет работу".
2. Исполнитель отправляет результаты, нажимает на кнопку “Работа выполнена”.
3. Заказчик видит изменение статуса в чате.
4. Заказчик проверяет работу, подтверждает выполнение, нажав на кнопку “Работа выполнена” или ждёт доработок.
5. Статус заказа меняется на “Завершён”.
6. Переход к сценарию “Стороны оставляют отзывы”.

Альтернативный сценарий

* У исполнителя или заказчика возникают непримиримые разногласия: переход к сценарию “Открытие спора”.

## Стороны оставляют отзывы

Действующее лицо: пользователь

Основной сценарий:

1. Заказ находится в статусе “Завершён”, заказчик и исполнитель находятся на страницах "Заказчик/исполнитель оставляет отзыв".
2. Заказчик/исполнитель нажимают на звёзды, выбирая их количество.
3. Заказчик/исполнитель вводит в текстовое поле развёрнутое мнение о работе со вторым участником заказа.
4. Заказчик/исполнитель нажимают на кнопку отправки.
5. Отзыв отправляется и появляется в публичном профиле.
6. Текстовое поле и количество звёзд блокируются для ввода.

Альтернативный сценарий

* Заказчик/исполнитель не пишут отзыв: ничего не происходит, возможность оставить отзыв сохраняется определённое время с момента завершения заказа.
* Отведённое время для оставления отзывов истекло: поля для ввода блокируются.

## Открытие спора

Действующее лицо: заказчик/исполнитель

Основной сценарий:

1. Исполнитель находится на странице "Исполнитель выполняет работу" или заказчик находится на странице "Заказчик ожидает выполнения работы исполнителем".
2. Исполнитель или заказчик нажимают на кнопку “Открыть спор”.
3. В чате появляется сообщение об открытии спора.
4. Стороны ожидают решения вопроса администратором в течение некоторого срока.

Альтернативный сценарий

* Инициатор спора решил отменить его: нажимает на кнопку “Отменить спор”, чат возвращается к состоянию до открытия спора, но в нём сохраняются уведомления об открытии и отмене спора.

## Заказчик смотрит отклики

Действующее лицо: заказчик

Предусловие: заказчик авторизован и находится на странице своего приватного профиля.

Основной сценарий:

1. Нажав на кнопку “Мои заказы”, заказчик попадает на страницу “Созданные заказы заказчика”.
2. Из предложенного списка заказчик выбирает интересующий его заказ и нажимает на кнопку “Подробнее”.
3. После шага 2 заказчик попадает на страницу "Просмотр заказа от лица заказчика (этап выбора исполнителя)". В разделе отклики представлены потенциальные исполнители, откликнувшиеся на предложение по данному заказу.
4. Заказчик просматривает кандидатов на выполнение заказа с использованием пагинации.

Альтернативный сценарий

* Заказчик хочет вернуться к списку своих заказов: нажимает на стрелочку влево (назад) в левой верхней части страницы.

## Заказчик общается в чате до согласования

Действующее лицо: пользователь

Предусловие: заказчик авторизован и находится на странице "Приватный профиль заказчика".

Основной сценарий:

1. Нажав на кнопку “Мои заказы”, заказчик попадает на страницу "Созданные заказы заказчика".
2. Из предложенного списка заказчик выбирает интересующий его заказ и нажимает на кнопку “Подробнее”.
3. После шага 2 заказчик попадает на страницу "Просмотр заказа от лица заказчика (этап выбора исполнителя)". В разделе отклики представлены потенциальные исполнители, откликнувшиеся на предложение по данному заказу.
4. Заказчик просматривает кандидатов на выполнение заказа с использованием пагинации.
5. Чтобы попасть в чат с интересующим исполнителем, заказчик нажимает на кнопку “Связаться” внутри блока соответствующего отклика.
6. После шага 5 заказчик попадает в чат с потенциальным исполнителем.
7. Чтобы написать сообщение потенциальному исполнителю, заказчик использует поле для ввода текста внизу страницы и, нажав на стрелочку справа от этого поля, отправляет сообщение исполнителю.

Альтернативный сценарий

* Заказчик нажимает на стрелочку влево (назад) в левом верхнем углу страницы, чтобы выбрать диалог с другим кандидатом.

## Исполнитель общается в чате до согласования

Действующее лицо: исполнитель

Предусловие: исполнитель авторизован и находится на странице "Приватный профиль исполнителя".

Основной сценарий:

1. Исполнитель нажимает на кнопку “Показать отклики”.
2. После шага 1 исполнитель попадает на страницу “Отклики исполнителя”.
3. Исполнитель ищет интересующий его заказ, используя пагинацию. В блоке интересующего заказа нажимает кнопку “Открыть переписку”.
4. После шага 3 исполнитель попадает в чат с заказчиком.
5. Чтобы написать сообщение заказчику, исполнитель использует поле для ввода текста внизу страницы и, нажав на стрелочку справа от этого поля, отправляет сообщение заказчику.

Альтернативный сценарий

* Исполнитель нажимает на стрелочку влево (назад) в левой верхней части страницы и возвращается на страницу просмотра своих откликов по заказам.

## Заказчик выбирает исполнителя

Действующее лицо: заказчик

Предусловие: заказчик авторизован и находится на странице "Приватный профиль заказчика".

Основной сценарий:

1. Нажав на кнопку “Мои заказы”, заказчик попадает на страницу “Список заказов заказчика”.
2. Из предложенного списка заказчик выбирает интересующий его заказ и нажимает на кнопку “Подробнее”.
3. После шага 2 заказчик попадает на страницу "Просмотр заказа от лица заказчика (этап выбора исполнителя)". В разделе отклики представлены потенциальные исполнители, откликнувшиеся на предложение по данному заказу.
4. Заказчик просматривает кандидатов на выполнение заказа с использованием пагинации.
5. В поле отклика подходящего кандидата заказчик нажимает кнопку “Принять”, чтобы выбрать кандидата в качестве исполнителя заказа.

Альтернативный сценарий

* На странице "Просмотр заказа от лица заказчика (этап выбора исполнителя)" в поле отклика подходящего кандидата заказчик нажимает кнопку “Связаться” (переходит в чат с потенциальным исполнителем). Чтобы выбрать кандидата в качестве исполнителя заказа, заказчик нажимает на кнопку “Принять” внизу страницы.
* На странице "Просмотр заказа от лица заказчика (этап выбора исполнителя)" в поле отклика неподходящего кандидата заказчик нажимает кнопку “Отклонить”.
* На странице "Просмотр заказа от лица заказчика (этап выбора исполнителя)" в поле отклика кандидата заказчик нажимает кнопку “Связаться” (переходит в чат с потенциальным исполнителем). Чтобы отказать кандидату, заказчик нажимает на кнопку “Отклонить” внизу страницы.

## Заказчик согласовывает условия в чате

Действующее лицо: заказчик

Предусловие: заказчик авторизован и находится на странице "Согласование условий (от лица заказчика)".

Основной сценарий:

1. Используя средства переписки, заказчик выдвигает условия выполнения заказа.

Альтернативный сценарий

* Заказчик не готов пойти на условия выбранного исполнителя и хочет выбрать другого кандидата: нажимает кнопку “Отказаться от заказа” внизу страницы.

## Исполнитель согласовывает условия в чате

Действующее лицо: исполнитель

Предусловие: исполнитель авторизован и находится на странице "Согласование условий (от лица исполнителя)".

Основной сценарий:

1. Используя средства переписки, исполнитель обсуждает с заказчиком условия выполнения заказа.

Альтернативный сценарий

* Исполнитель не готов пойти на условия заказчика: нажимает кнопку “Отказаться от выполнения” внизу страницы.

## Заказчик пополняет счёт

Действующее лицо: заказчик

Предусловие: заказчик авторизован и находится на странице "Приватный профиль заказчика".

Основной сценарий:

1. Заказчик нажимает на кнопку “Баланс” в верхней части страницы.
2. В выпадающем меню кнопки выбирает “Пополнение”.
3. В появившемся модальном окне необходимо ввести сумму пополнения.
4. Нажать кнопку “Пополнить”.
5. Сайт зачисляет выбранную сумму на баланс заказчика.

Альтернативный сценарий

* Заказчик передумал пополнять баланс: в появившемся модальном окне нажимает кнопку “Отменить”.
* Заказчик выводит деньги со своего баланса, используя последовательно кнопки “Баланс” в верхней части страницы, “Вывод” в выпадающем меню.

## Администратор смотрит статистику

Действующее лицо: администратор

Основной сценарий:

1. Администратор авторизован и находится на странице "Главная администратора".
2. Администратор выбирает раздел “Статистика” в боковом меню.
3. Переход на страницу "Статистика".
4. Администратор просматривает графики, метрики и прочее.

## Подсчёт статистики

Действующее лицо: все пользователи

Основной сценарий:

1. Сервер замеряет статистику запросов к API (успешные, ошибочные, время ответа).
2. Сервер считает количество заказов, количество пользователей, количество споров.

## Исполнитель ищет заказы

Действующее лицо: исполнитель

Предусловие: исполнитель авторизован и находится на странице "Приватный профиль исполнителя"

Основной сценарий:

1. Нажимает на кнопку “На главную” в верхней части страницы
2. В списке заказов на странице "Главная исполнителя" выбирает подходящий ему заказ
3. Попадает на страницу "До отклика на заказ от исполнителя" с подробностями заказа

Альтернативный сценарий

* Решает покинуть страницу заказа, нажимает на стрелку назад в верхней левой части экрана, возвращается на страницу "Главная исполнителя" со списком заказов, продолжает поиск
* Нажимает на иконку заказчика, попадает на страницу "Публичный профиль заказчика", нажимает на кнопку “Все заказы”, попадает на страницу "Публичные заказы заказчика", переходит на детали заказа, кликая на конкретный из списка, попадает на одну из двух страниц: "До отклика на заказ от исполнителя" или "После отклика на заказ от исполнителя"

## Исполнитель откликается на заказ

Действующее лицо: исполнитель

Предусловие: исполнитель авторизован и находится на странице "До отклика на заказ от исполнителя"

Основной сценарий:

1. Пишет сообщение заказчику в поле ввода и нажимает на кнопку “Готов взяться” в нижней части страницы
2. Отклик теперь отображается заказчику, а также на странице "Отклики исполнителя"
3. Страница переходит в состояние "После отклика на заказ от исполнителя"

Альтернативный сценарий

* Исполнитель уже откликался на заказ, поэтому находится на странице "После отклика на заказ от исполнителя", нажимает на кнопку “Отозвать отклик” внизу экрана, отклик перестает отображаться заказчику, а также на странице "Отклики исполнителя", страница переходит в состояние "До отклика на заказ от исполнителя"

## Исполнитель смотрит свои отклики

Действующее лицо: исполнитель

Предусловие: исполнитель авторизован и находится на странице "Приватный профиль исполнителя"

Основной сценарий:

1. На колонке с частью откликов в правой части экрана нажимает кнопку “Показать отклики”
2. В списке откликов по клику возможен переход либо на страницу "Заказчик связывается с исполнителем (от лица исполнителя)", либо на страницу "После отклика на заказ от исполнителя"

Альтернативный сценарий

* Для некоторой части откликов переход на страницу "После отклика на заказ от исполнителя" возможен из колонки на странице "Приватный профиль исполнителя"

## Исполнитель выводит средства

Действующее лицо: исполнитель

Предусловие: исполнитель авторизован и находится на странице "Приватный профиль исполнителя"

Основной сценарий:

1. Нажимает кнопку “Баланс” в верхней части страницы
2. В выпадающем меню выбирает “Вывод”
3. В модальном окне вводит корректную сумму вывода и нажимает кнопку “Вывести”
4. Ожидает поступление денег

Альтернативный сценарий

* Отменить вывод денег можно, нажав кнопку “Отменить” в модальном окне для ввода
* Пополнение баланса происходит в аналогичной последовательность, но с выбором кнопки “Пополнение” в выпадающем меню

## Экспорт

Действующее лицо: администратор

Предусловие: администратор авторизован и находится на странице "Главная администратора"

Основной сценарий:

1. В сайдбаре в левой части экрана нажимает раздел “Экспорт” и попадает на страницу "Экспорт"
2. Сформированный файл предлагается для скачки

Альтернативный сценарий

* При неудаче формирования файла или других технических проблемах файл не скачивается, выводится alert

## Импорт

Действующее лицо: администратор

Предусловие: администратор авторизован и находится на странице "Главная администратора"

Основной сценарий:

1. В сайдбаре в левой части экрана нажимает раздел “Импорт” и попадает на страницу "Импорт"
2. Выбирает файл импорта в форме для в верхней части раздела
3. Данные импортируются в систему из выбранного файла

Альтернативный сценарий

* Если файл не был выбран, то произойдет ошибка (страница "Ошибка импорта")
* При технических проблемах файл не загружается, выводится alert

# 2. Модель данных

## 2.1. Нереляционная модель

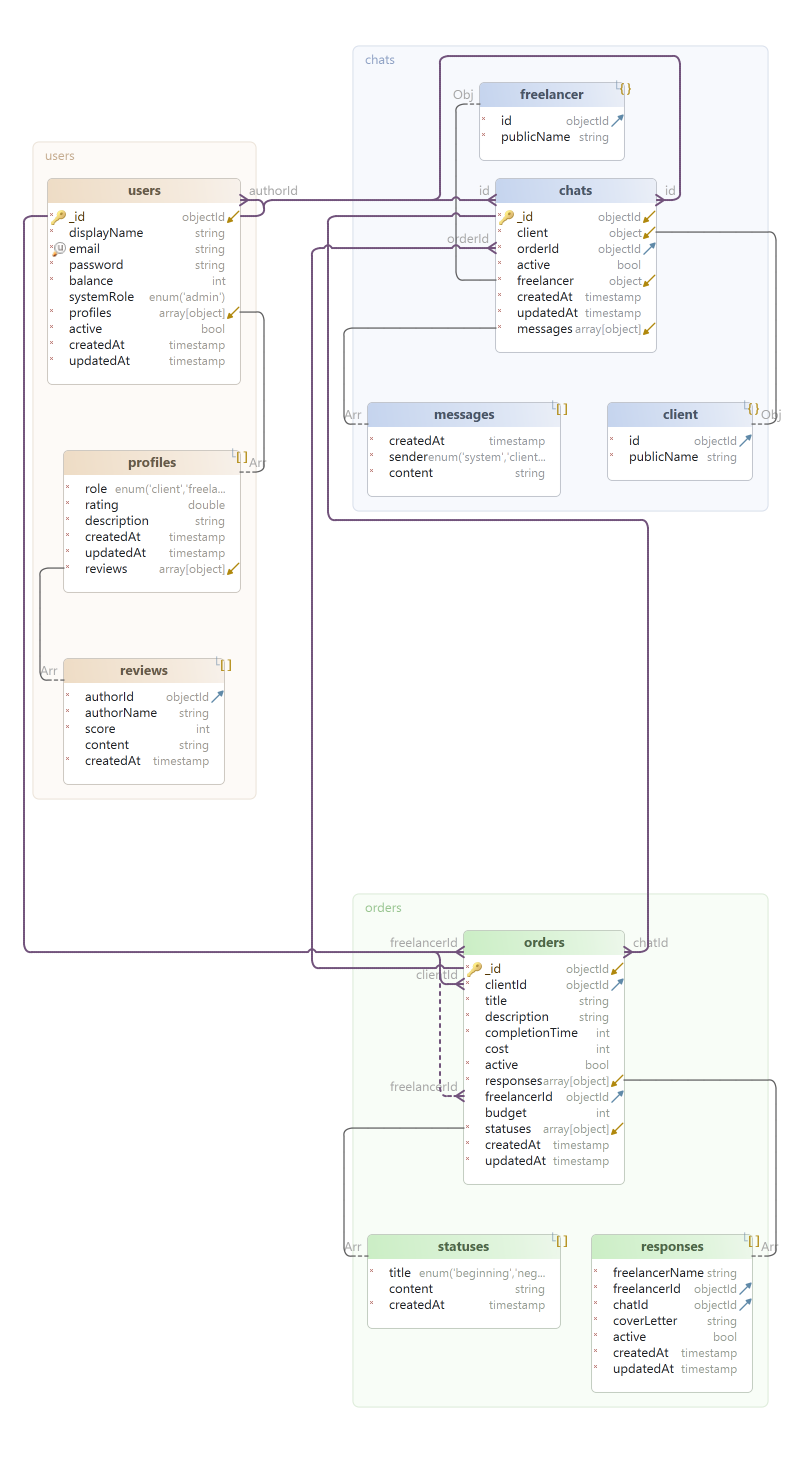
****

Рисунок 2.1 – Графическое представление нереляционной модели

### 1. chats

Коллекция переписок (чатов).

Назначение: хранит переписку после отклика и до последнего этапа заказа. Привязывается конкретно к одному заказу для заказчика и одного исполнителя.

Особенности:

* Флаг active позволяет запрещать новые сообщения в чат
* system тип отправителя предназначается для случаев наподобии споров, чтобы можно было оповестить участников о решении, а также для отображения перехода с этапа на этап

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬋ | \_id | objectId |  |
| \* | client | object | Информация о заказчике (одном из участников переписки). |
| \* ⬈ | client.id | objectId | ID заказчика. |
| \* | client.publicName | string | Публичное имя заказчика. |
| \* ⬈ | orderId | objectId | ID заказа, по которому ведётся переписка. |
| \* | active | bool DEFAULT true | Активная ли переписка, можно ли отправлять новые сообщения. |
| \* | freelancer | object | Информация об исполнителе (одном из участников переписки). |
| \* ⬈ | freelancer.id | objectId | ID исполнителя. |
| \* | freelancer.publicName | string | Публичное имя исполнителя. |
| \* | createdAt | timestamp DEFAULT ‘now()’ | Дата создания документа. |
| \* | updatedAt | timestamp DEFAULT ‘now()’ | Дата обновления документа. |
| \* | messages | array[object] | Список сообщений в чате. |
| \* | messages.createdAt | timestamp DEFAULT ‘now()’ | Дата создания документа. |
| \* | messages.sender | enum(‘system’,‘client’,‘freelancer’) | Отправитель сообщения (может быть система, исполнитель или заказчик). |
| \* | messages.content | string | Текст сообщения. |

##### 

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | \_id\_ | ON \_id |

##### Связи

| Type | Name | On |
| --- | --- | --- |
| Vir | fk\_chats\_users | ( freelancer.id ) ref [users](#users) (\_id) |
| Vir | fk\_chats\_users\_0 | ( client.id ) ref [users](#users) (\_id) |
| Vir | fk\_chats\_orders | ( orderId ) ref [orders](#orders) (\_id) |

### 2. orders

Коллекция заказов.

Назначение: хранит основные данные о заказе, размещенном заказчиком, на который откликаются и после различных согласований выполняют исполнители; также инкапсулирует его текущий статус

Особенности:

* Последний элемент в списке statuses есть текущий статус заказа
* Некоторые опциональные поля, такие как freelancerId и budget появляются по ходу выполнения заказа и его перехода в новые состояния и должны рассматриваться обязательно с учетом текущего статуса

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬋ | \_id | objectId |  |
| \* ⬈ | clientId | objectId | ID пользователя, разместившего заказ. |
| \* | title | string | Название статуса. |
| \* | description | string | Описание заказа, задаётся пользователем. |
| \* | completionTime | int | Срок выполнения заказа (int64), хранится в наносекундах. Максимальный срок при использовании int64 - 292 года. |
|  | cost | int | Стоимость заказа, может быть числом, может быть неопределённой. Если это поле не задано, считать стоимость договорной. |
| \* | active | bool DEFAULT true | Активен ли заказ (может быть неактивен, если заказ был скрыт администратором или удалён заказчиком). |
| \* | responses | array[object] | Список откликов от фрилансеров. |
| \* | responses.freelancerName | string | Публичное имя пользователя, откликнувшегося на заказ. |
| \* ⬈ | responses.freelancerId | objectId | ID пользователя, откликнувшегося на заказ. |
| \* ⬈ | responses.chatId | objectId | ID чата, созданного для переписки в рамках отклика. |
| \* | responses.coverLetter | string | Сопроводительное письмо фрилансера, по совместительству первое сообщение в переписке. |
| \* | responses.active | bool DEFAULT true | Активен ли отклик. |
| \* | responses.createdAt | timestamp DEFAULT ‘now()’ | Дата создания документа. |
| \* | responses.updatedAt | timestamp DEFAULT ‘now()’ | Дата обновления документа. |
| ⬈ | freelancerId | objectId | ID выбранного исполнителя, появляется в документе на определённом этапе заказа. |
|  | budget | int | Зарезервированный заказчиком бюджет сделки. Может быть не указан в коллекции до определённого этапа. В отличие от cost, используется для расчётов, а не для показа карточки заказа. |
| \* | statuses | array[object] | История изменений статусов заказа. Последний элемент массива всегда отражает актуальный статус. |
| \* | statuses.title | enum(‘beginning’,‘negotiation’,‘budgeting’,‘work’,‘reviews’,‘finished’,‘dispute’) | Название статуса. |
|  | statuses.content | string | Дополнительная информация, если нужно. |
| \* | statuses.createdAt | timestamp DEFAULT ‘now()’ | Дата создания документа. |
| \* | createdAt | timestamp DEFAULT ‘now()’ | Дата создания документа. |
| \* | updatedAt | timestamp DEFAULT ‘now()’ | Дата обновления документа. |

##### 

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | \_id\_ | ON \_id |

##### Связи

| Type | Name | On |
| --- | --- | --- |
| Vir | fk\_orders\_users | ( responses.freelancerId ) ref [users](#users) (\_id) |
| Vir | fk\_orders\_users\_0 | ( clientId ) ref [users](#users) (\_id) |
| Vir | fk\_orders\_users\_1 | ( freelancerId ) ref [users](#users) (\_id) |
| Vir | fk\_orders\_chats | ( responses.chatId ) ref [chats](#chats) (\_id) |

### users

Коллекция пользователей.

Назначение: хранит данные об отдельном пользователе и всех его профилях, зарегистрированных на один email, который должен быть уникальным в рамках сайта.

Особенности:

* Профили разделяются по роли, и все остальные коллекции ссылаются на на глобальный \_id коллекции users, но в зависимости от контекста приложение ищет нужный профиль

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬋ | \_id | objectId |  |
| \* | displayName | string | Отображаемое имя пользователя (ФИО или ФИ). |
| \* 🔍 | email | string | Адрес электронной почты. |
| \* | password | string | Хэш пароля. |
| \* | balance | int DEFAULT 0 | Количество денег на счету пользователя. |
|  | systemRole | enum(‘admin’) | Системная роль (администратор, модератор и т.п.). Определяет возможность пользователя авторизоваться и попадать на служебные страницы. |
| \* | profiles | array[object] | Профили пользователя (как заказчика или исполнителя). |
| \* | profiles.role | enum(‘client’,‘freelancer’) | Роль (заказчик или исполнитель). |
| \* | profiles.rating | double DEFAULT 0 | Рейтинг, основан на отзывах. |
| \* | profiles.description | string | Пользовательское поле “О себе”. |
| \* | profiles.createdAt | timestamp | Дата создания документа. |
| \* | profiles.updatedAt | timestamp | Дата обновления документа. |
| \* | profiles.reviews | array[object] | Отзывы на профиль пользователя. |
| \* ⬈ | profiles.reviews.authorId | objectId | ID пользователя, оставившего отзыв. |
| \* | profiles.reviews.authorName | string | Публичное имя пользователя, оставившего отзыв. |
| \* | profiles.reviews.score | int | Оценка (от 1 до 5). |
|  | profiles.reviews.content | string | Текст отзыва. |
| \* | profiles.reviews.createdAt | timestamp DEFAULT now() | Дата создания документа. |
| \* | active | bool DEFAULT true | Активен ли аккаунт пользователя в данный момент. Альтернатива удалению документа из коллекции. |
| \* | createdAt | timestamp DEFAULT ‘now()’ | Дата создания документа. |
| \* | updatedAt | timestamp DEFAULT ‘now()’ | Дата обновления документа. |

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | \_id\_ | ON \_id |
| 🔍 | unq\_users\_email | ON email |

##### Связи

| Type | Name | On |
| --- | --- | --- |
| Vir | fk\_users\_users | ( profiles.reviews.authorId ) ref [users](#users) (\_id) |

**Оценка объёма информации, хранимой в нереляционной модели**

Users:  
 reviews:  
 authorId: objectId, V = 12b  
 authorName: string, V = 30b  
 score: int (int64), V = 8b  
 content: string, V = 200b  
 createdAt: timestamp, 8b  
 Итого: 12 + 30 + 8 + 200 + 8 = 258b

 profiles:  
 role: enum, V = 15b  
 rating: double, V = 8b  
 description: string, V = 500b  
 createdAt: timestamp, 8b  
 updatedAt: timestamp, 8b  
 reviews: array[object], V = NR ⋅ 258b, где NR – количество отзывов  
 Итого: 15 + 8 + 500 + 8 + 8 + NR ⋅ 258 = (539 + NR ⋅ 258)b  
 В среднем на профиль приходится 10 отзывов, тогда V = 3119b

 users:  
 \_id: objectId, V = 12b  
 displayName: string, V = 30b  
 email: string, V = 30b  
 password: string, V = 60b  
 balance: int (int64), V = 8b  
 systemRole: enum, V = 10b  
 profiles: array[object], V = NP ⋅ 3119b, где NP – количество профилей. NP = 2 (фрилансер и заказчик), поэтому V = 6238b  
 active: bool, V = 1b  
 createdAt: timestamp, 8b  
 updatedAt: timestamp, 8b

Фактический объём коллекции users: 12 + 30 + 30 + 60 + 8 + 10 + 6238 + 1 + 8 + 8 = 6405b

Orders:  
 statuses:  
 title: enum, V = 15b  
 content: string, V = 100b  
 createdAt: timestamp, 8b  
 Итого: 15 + 100 + 8 = 123b

 responses:  
 freelancerName: string, V = 30b  
 freelancerId: objectId, V = 12b  
 chatId: objectId, V = 12b  
 coverLetter: string, V = 500b  
 active: bool, V = 1b  
 createdAt: timestamp, 8b  
 updatedAt: timestamp, 8b  
 Итого: 30 + 12 + 12 + 500 + 1 + 8 + 8 = 571b

 orders:  
 \_id: objectId, V = 12b  
 clientId: objectId, V = 12b  
 title: string, 50b  
 description: string, 1000b  
 completionTime: int (int64), V = 8b  
 cost: int (int64), V = 8b  
 active: bool, V = 1b  
 responses: array[object], V = NRes ⋅ 571b, где NRes – количество откликов. В среднем NRes = 5, поэтому V = 2855b  
 freelancerId: objectId, V = 12b  
 budget: int (int64), V = 8b  
 statuses: array[object], V = NS ⋅ 123b, где NS – количество статусов. В среднем NS = 5, поэтому V = 615b  
 createdAt: timestamp, 8b  
 updatedAt: timestamp, 8b

Фактический объём коллекции orders: 12 + 12 + 50 + 1000 + 8 + 8 + 1 + 2855 + 12 + 8 + 615 + 8 + 8 = 4597b

Chats:  
 client:  
 id: objectId, V = 12b  
 publicName: string, V = 30b  
 Итого: 12 + 30 = 42b

 messages:  
 createdAt: timestamp, 8b  
 sender: enum, V = 15b  
 content: string, V = 500b  
 Итого: 8 + 15 + 500 = 523b

 freelancer:  
 id: objectId, V = 12b  
 publicName: string, V = 30b  
 Итого: 12 + 30 = 42b

 chats:  
 \_id: objectId, V = 12b  
 client: object, V = 42b  
 orderId: objectId, V = 12b  
 active: bool, V = 1b  
 freelancer: object, V = 42b  
 createdAt: timestamp, 8b  
 updatedAt: timestamp, 8b  
 messages: array[object], V = NM ⋅ 523b, где M – количество сообщений. В среднем NM = 50, поэтому V = 26150b

Фактический объём коллекции chats: 12 + 42 + 12 + 1 + 42 + 8 + 8 + 26150 = 26275b

Фактический объём модели: NU ⋅ 6405 + NO ⋅ 4597 + NC ⋅ 26275, где NU – количество пользователей, NO – количество заказов, NC – количество чатов.  
Выразим всё через количество заказов: на 1 заказ в среднем приходится 0.5 пользователей (так как заказчики размещают и фрилансеры выполняют более одного заказа) и в среднем 3 чата (считая чаты откликов), поэтому:  
V(NO) = (86625 ⋅ NO)bytes

**Избыточность данных**

Для вычисления «чистого» объёма данных исключим из расчетов дублирующуюся и служебную информацию, тогда:  
Users:  
 reviews:  
 score: int (int64), V = 8b  
 content: string, V = 200b  
 Итого: 200 + 8 = 208b

 profiles:  
 role: enum, V = 15b  
 rating: double, V = 8b  
 description: string, V = 500b  
 reviews: array[object], V = NR ⋅ 208b, где NR – количество отзывов, в среднем NR = 10  
 Итого: 15 + 8 + 500 + 10 ⋅ 208 = 2603b

 users:  
 displayName: string, V = 30b  
 email: string, V = 30b  
 password: string, V = 60b  
 balance: int (int64), V = 8b  
 systemRole: enum, V = 10b  
 profiles: array[object], V = NP ⋅ 2603b, где NP – количество профилей. NP = 2 (фрилансер и заказчик), поэтому V = 5206b

«Чистый» объём коллекции users: 30 + 30 + 60 + 8 + 10 + 5206 = 5344b

Orders:  
 statuses:  
 title: enum, V = 15b  
 content: string, V = 100b  
 Итого: 115b

 responces:  
 coverLetter: string, V = 500b  
 Итого: 500b

 orders:  
 title: string, 50b  
 description: string, 1000b  
 completionTime: int (int64), V = 8b  
 cost: int (int64), V = 8b  
 responses: array[object], V = NRes ⋅ 500b, где NRes – количество откликов. В среднем NRes = 5, поэтому V = 2500b  
 budget: int (int64), V = 8b  
 statuses: array[object], V = NS ⋅ 115b, где NS – количество статусов. В среднем NS = 5, поэтому V = 575b

«Чистый» объём коллекции orders: 50 + 1000 + 8 + 8 + 2500 + 8 + 575 = 4149b

Chats:  
 messages:  
 sender: enum, V = 15b  
 content: string, V = 500b  
 Итого: 515b

 chats:  
 messages: array[object], V = NM ⋅ 515b, где NM – количество сообщений. В среднем NM = 50, поэтому V = 25750b

«Чистый» объём коллекции chats: 25750b

«Чистый» объём данных: NU ⋅ 5344 + NO ⋅ 4149 + NC ⋅ 25750, где NU – количество пользователей, NO – количество заказов, NC – количество чатов.  
Выразим всё через количество заказов:  
Vc(NO) = (84071 ⋅ NO)bytes

Избыточность:  
R(NO) = V(NO) / Vc(NO) = 86625 ⋅ NO / 84071 ⋅ NO = 1.03

**Направление роста модели при увеличении количества объектов каждой сущности**

При увеличении количества объектов любой из сущностей модель будет расти линейно.

**Примеры данных**

Коллекция *users:*

{  
 "\_id": ObjectId("64a1b2c3d4e5f6a7b8c9d0e1"),  
 "displayName": "Иван Иванов",  
 "email": "ivanov@example.com",  
 "password": "hash1",  
 "balance": 1000,  
 "systemRole": "user",  
 "profiles": [  
 {  
 "role": "client",  
 "rating": 4.5,  
 "description": "Заказчик Иван",  
 "createdAt": ISODate("2023-10-01T10:00:00Z"),  
 "updatedAt": ISODate("2023-10-01T10:00:00Z"),  
 "reviews": []  
 }  
 ],  
 "active": true,  
 "createdAt": ISODate("2023-10-01T10:00:00Z"),  
 "updatedAt": ISODate("2023-10-01T10:00:00Z")  
}

Коллекция *orders:*

{  
 "\_id": ObjectId("64a1b2c3d4e5f6a7b8c9d0e2"),  
 "clientId": ObjectId("64a1b2c3d4e5f6a7b8c9d0e1"),  
 "title": "Разработка сайта",  
 "description": "Создание лендинга",  
 "completionTime": 1000000000,  
 "cost": 1000,  
 "active": true,  
 "responses": [  
 {  
 "freelancerName": "Петр Петров",  
 "freelancerId": ObjectId("64a1b2c3d4e5f6a7b8c9d0e3"),  
 "chatId": ObjectId("64a1b2c3d4e5f6a7b8c9d0e4"),  
 "coverLetter": "Готов взяться на проект!",  
 "active": true,  
 "createdAt": ISODate("2023-10-01T10:00:00Z"),  
 "updatedAt": ISODate("2023-10-01T10:00:00Z")  
 }  
 ],  
 "freelancerId": ObjectId("64a1b2c3d4e5f6a7b8c9d0e3"),  
 "budget": 1000,  
 "statuses": [  
 {  
 "title": "beginning",  
 "content": "Заказ создан",  
 "createdAt": ISODate("2023-10-01T10:00:00Z")  
 }  
 ],  
 "createdAt": ISODate("2023-10-01T10:00:00Z"),  
 "updatedAt": ISODate("2023-10-01T10:00:00Z")  
}

Коллекция *chats*:

{  
 "\_id": ObjectId("64a1b2c3d4e5f6a7b8c9d0e4"),  
 "client": {  
 "id": ObjectId("64a1b2c3d4e5f6a7b8c9d0e1"),  
 "publicName": "Иван Иванов"  
 },  
 "orderId": ObjectId("64a1b2c3d4e5f6a7b8c9d0e2"),  
 "active": true,  
 "freelancer": {  
 "id": ObjectId("64a1b2c3d4e5f6a7b8c9d0e3"),  
 "publicName": "Петр Петров"  
 },  
 "createdAt": ISODate("2023-10-01T10:00:00Z"),  
 "updatedAt": ISODate("2023-10-01T10:00:00Z"),  
 "messages": [  
 {  
 "createdAt": ISODate("2023-10-01T10:00:00Z"),  
 "sender": "client",  
 "content": "Здравствуйте!"  
 },  
 {  
 "createdAt": ISODate("2023-10-01T10:05:00Z"),  
 "sender": "freelancer",  
 "content": "Добрый день!"  
 }  
 ]  
}

**Примеры запросов**

**Сценарий: Регистрация пользователя**

Основной сценарий:

Проверка уникальности email:

db.users.findOne({ email: "ivanov@example.com" });

* Количество запросов: 1
* Задействованные коллекции: users

Создание нового пользователя:

db.users.insertOne({  
 displayName: "Иван Иванов",  
 email: "ivanov@example.com",  
 password: "hash1",  
 balance: 1000,  
 systemRole: "user",  
 profiles: [  
 {  
 role: "client",  
 rating: 4.5,  
 description: "Заказчик Иван",  
 createdAt: new Date(),  
 updatedAt: new Date(),  
 reviews: []  
 }  
 ],  
 active: true,  
 createdAt: new Date(),  
 updatedAt: new Date()  
});

* Количество запросов: 1
* Задействованные коллекции: users

Итого:  
- Количество запросов: 2  
- Задействованные коллекции: users

**Сценарий: Авторизация пользователя**

Основной сценарий:

Проверка пользователя:

db.users.findOne({ email: "ivanov@example.com" });

* Количество запросов: 1
* Задействованные коллекции: users

Получение профиля пользователя:

db.users.findOne(  
 { email: "ivanov@example.com" },  
 { "profiles.role": 1 }  
);

* Количество запросов: 1
* Задействованные коллекции: users

Итого:  
- Количество запросов: 2  
- Задействованные коллекции: users

**Сценарий: Редактирование профиля**

Основной сценарий:

Обновление данных пользователя:

db.users.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e1") },  
 { $set: { displayName: "Иван Иванович" } }  
);

* Количество запросов: 1
* Задействованные коллекции: users

Обновление данных профиля:

db.users.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e1"), "profiles.role": "client" },  
 { $set: { "profiles.$.description": "Новое описание" } }  
);

* Количество запросов: 1
* Задействованные коллекции: users

Итого:  
- Количество запросов: 2  
- Задействованные коллекции: users

**Сценарий: Смена роли**

Основной сценарий:

Обновление роли в профиле:

db.users.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e1"), "profiles.role": "client" },  
 { $set: { "profiles.$.role": "freelancer" } }  
);

* Количество запросов: 1
* Задействованные коллекции: users

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: users

**Сценарий: Заказчик публикует новый заказ**

Основной сценарий:

Создание заказа:

db.orders.insertOne({  
 clientId: ObjectId("64a1b2c3d4e5f6a7b8c9d0e1"),  
 title: "Разработка сайта",  
 description: "Создание лендинга",  
 completionTime: 1000000000,  
 cost: 1000,  
 active: true,  
 responses: [],  
 statuses: [  
 {  
 title: "beginning",  
 content: "Заказ создан",  
 createdAt: new Date()  
 }  
 ],  
 createdAt: new Date(),  
 updatedAt: new Date()  
});

* Количество запросов: 1
* Задействованные коллекции: orders

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: orders

**Сценарий: Заказчик редактирует заказ**

Основной сценарий:

Обновление данных заказа:

db.orders.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e2") },  
 { $set: { title: "Обновленный проект", description: "Новое описание" } }  
);

* Количество запросов: 1
* Задействованные коллекции: orders

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: orders

**Сценарий: Стороны согласовали условия и работа началась**

Основной сценарий:

Обновление статуса заказа:

db.orders.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e2") },  
 { $push: { statuses: { title: "negotiation", content: "Условия согласованы", createdAt: new Date() } } }  
);

* Количество запросов: 1
* Задействованные коллекции: orders

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: orders

**Сценарий: Стороны общаются в процессе выполнения заказа**

Основной сценарий:

Добавление сообщения в чат:

db.chats.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e4") },  
 { $push: { messages: { sender: "client", content: "Здравствуйте!", createdAt: new Date() } } }  
);

* Количество запросов: 1
* Задействованные коллекции: chats

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: chats

**Сценарий: Завершение заказа**

Основной сценарий:

Обновление статуса заказа:

db.orders.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e2") },  
 { $push: { statuses: { title: "finished", content: "Заказ завершен", createdAt: new Date() } } }  
);

* Количество запросов: 1
* Задействованные коллекции: orders

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: orders

**Сценарий: Стороны оставляют отзывы**

Основной сценарий:

Добавление отзыва:

db.users.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e3"), "profiles.role": "freelancer" },  
 { $push: { "profiles.$.reviews": { authorId: ObjectId("64a1b2c3d4e5f6a7b8c9d0e1"), authorName: "Иван Иванов", score: 5, content: "Отлично!", createdAt: new Date() } } }  
);

* Количество запросов: 1
* Задействованные коллекции: users

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: users

**Сценарий: Исполнитель смотрит свои отклики**

Основной сценарий:

Получение откликов исполнителя:

db.orders.find({ "responses.freelancerId": ObjectId("64a1b2c3d4e5f6a7b8c9d0e3") });

* Количество запросов: 1
* Задействованные коллекции: orders

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: orders

**Сценарий: Исполнитель отзывает отклик**

Основной сценарий:

Обновление активности отклика:

db.orders.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e2"), "responses.freelancerId": ObjectId("64a1b2c3d4e5f6a7b8c9d0e3") },  
 { $set: { "responses.$.active": false } }  
);

* Количество запросов: 1
* Задействованные коллекции: orders

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: orders

**Сценарий: Исполнитель общается в чате до согласования**

Основной сценарий:

Добавление сообщения в чат:

db.chats.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e4") },  
 { $push: { messages: { sender: "freelancer", content: "Добрый день!", createdAt: new Date() } } }  
);

* Количество запросов: 1
* Задействованные коллекции: chats

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: chats

**Сценарий: Заказчик согласовывает условия в чате**

Основной сценарий:

Добавление системного сообщения о согласовании:

db.chats.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e4") },  
 { $push: { messages: { sender: "system", content: "Условия согласованы.", createdAt: new Date() } } }  
);

* Количество запросов: 1
* Задействованные коллекции: chats

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: chats

**Сценарий: Исполнитель согласовывает условия в чате**

Основной сценарий:

Добавление системного сообщения о согласовании:

db.chats.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e4") },  
 { $push: { messages: { sender: "system", content: "Условия согласованы.", createdAt: new Date() } } }  
);

* Количество запросов: 1
* Задействованные коллекции: chats

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: chats

**Сценарий: Подсчёт статистики**

Основной сценарий:

Получение количества заказов:

db.orders.countDocuments();

* Количество запросов: 1
* Задействованные коллекции: orders

Получение количества пользователей:

db.users.countDocuments();

* Количество запросов: 1
* Задействованные коллекции: users

Получение количества споров:

db.chats.countDocuments({ "messages.sender": "system", "messages.content": { $regex: /Спор/ } });

* Количество запросов: 1
* Задействованные коллекции: chats

Итого:  
- Количество запросов: 3  
- Задействованные коллекции: orders, users, chats

**Сценарий: Исполнитель ищет заказы**

Основной сценарий:

Получение списка активных заказов:

db.orders.find({ active: true });

* Количество запросов: 1
* Задействованные коллекции: orders

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: orders

**Сценарий: Исполнитель откликается на заказ**

Основной сценарий:

Создание отклика:

db.orders.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e2") },  
 {  
 $push: {  
 responses: {  
 freelancerName: "Петр Петров",  
 freelancerId: ObjectId("64a1b2c3d4e5f6a7b8c9d0e3"),  
 chatId: ObjectId("64a1b2c3d4e5f6a7b8c9d0e4"),  
 coverLetter: "Готов взяться на проект!",  
 active: true,  
 createdAt: new Date(),  
 updatedAt: new Date()  
 }  
 }  
 }  
);

* Количество запросов: 1
* Задействованные коллекции: orders

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: orders

**Сценарий: Исполнитель выводит средства**

Основной сценарий:

Обновление баланса пользователя:

db.users.updateOne(  
 { \_id: ObjectId("64a1b2c3d4e5f6a7b8c9d0e3") },  
 { $inc: { balance: -500 } }  
);

* Количество запросов: 1
* Задействованные коллекции: users

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: users

**Сценарий: Экспорт данных**

Основной сценарий:

Экспорт данных:

db.users.find();  
db.profiles.find();  
db.orders.find();  
db.chats.find();  
db.messages.find();  
db.responses.find();  
db.reviews.find();  
db.statuses.find();

* Количество запросов: 8
* Задействованные коллекции: Все коллекции

Итого:  
- Количество запросов: 8  
- Задействованные коллекции: Все коллекции

**Сценарий: Импорт данных**

Основной сценарий:

Импорт данных:

db.users.insertOne({  
 displayName: "Новый Пользователь",  
 email: "newuser@example.com",  
 password: "hash4",  
 balance: 0,  
 systemRole: "user",  
 profiles: [],  
 active: true,  
 createdAt: new Date(),  
 updatedAt: new Date()  
});

* Количество запросов: 1
* Задействованные коллекции: users

Итого:  
- Количество запросов: 1  
- Задействованные коллекции: users

## 2.2. Реляционная модель

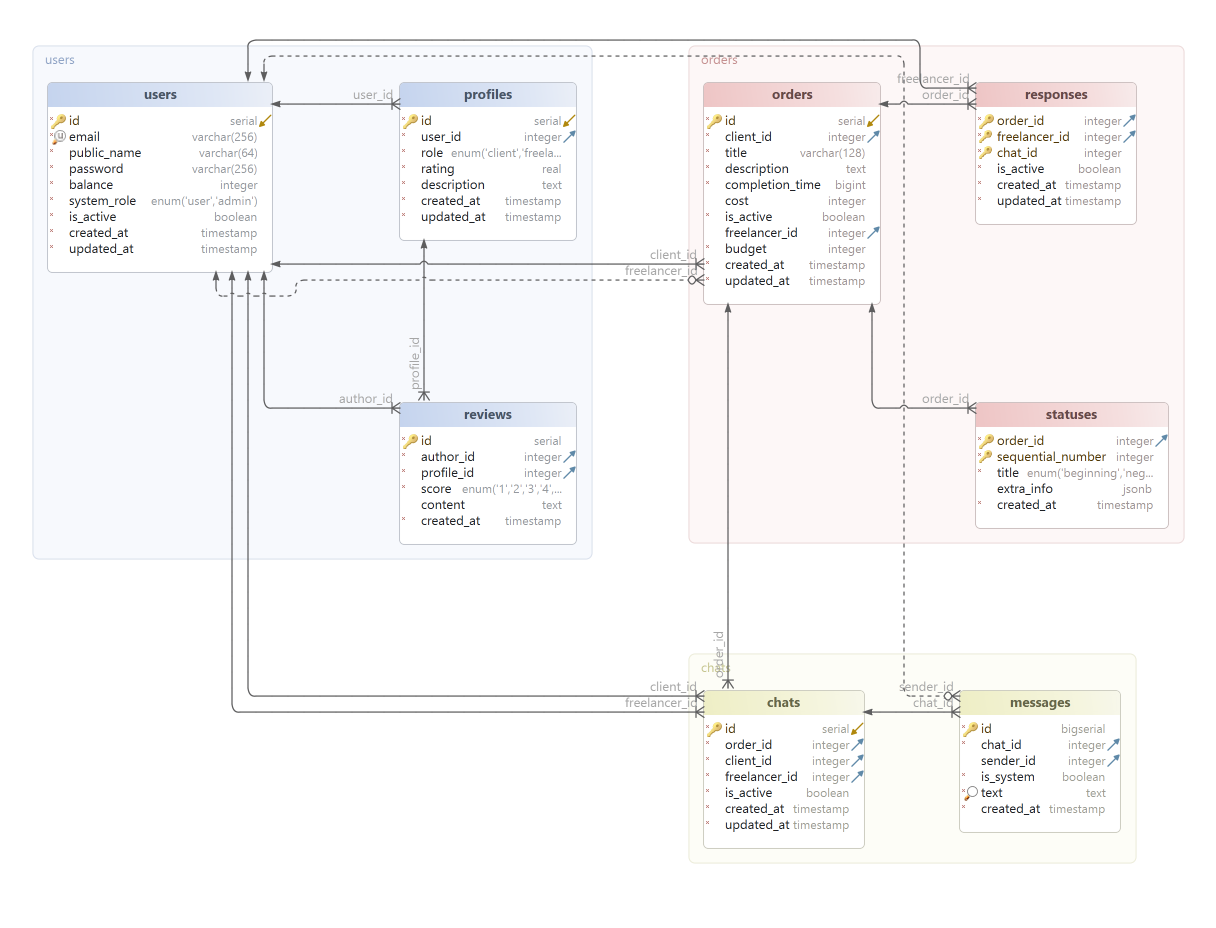
****

Рисунок 2.2 – Графическое представление реляционной модели

### 1. chats

Переписки.

Назначение: хранит переписку после отклика и до последнего этапа заказа. Привязывается конкретно к одному заказу для заказчика и одного исполнителя.

Особенности:

* Сами сообщения привязываются внешним ключом из отношения messages

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬋ | id | serial | ID переписки. |
| \* ⬈ | order\_id | integer | ID заказа, в рамках которого ведётся переписка. |
| \* ⬈ | client\_id | integer | ID пользователя, выступающего в роли заказчика. |
| \* ⬈ | freelancer\_id | integer | ID пользователя, выступающего в роли исполнителя. |
| \* | is\_active | boolean DEFAULT true | Является ли переписка активной (можно ли отправлять новые сообщения). |
| \* | created\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата создания записи. |
| \* | updated\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата обновления записи. |

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | pk\_chats | ON id |

##### Внешние ключи

| Type | Name | On |
| --- | --- | --- |
|  | fk\_chats\_users | ( client\_id ) ref [users](#users) (id) |
|  | fk\_chats\_users\_1 | ( freelancer\_id ) ref [users](#users) (id) |
|  | fk\_chats\_orders | ( order\_id ) ref [orders](#orders) (id) |

### 2. messages

Сообщения в переписке.

Назначение: сохраняет данные о сообщении в отдельной переписке

Особенности:

* Очередность сообщения в переписке определяется по возрастанию id для конкретного chat\_id
* Флаг is\_system требуется для системных сообщений, например, о смене этапов, спорах и так далее

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 | id | bigserial | ID сообщения (при потоке в 1 млн. сообщений в секунду должно хватить на 292 тысячи лет). |
| \* ⬈ | chat\_id | integer | ID переписки, к которой относится сообщение. |
| ⬈ | sender\_id | integer | ID отправителя сообщения (может быть NULL), если сообщение системное. |
| \* | is\_system | boolean DEFAULT false | Является ли сообщение системным (системные сообщения должны отрисовываться по другому). |
| \* 🔎 | text | text | Текст сообщения. |
| \* | created\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата создания записи. |

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | pk\_messages | ON id |
| 🔎 | idx\_messages | ON text |

##### Внешние ключи

| Type | Name | On |
| --- | --- | --- |
|  | fk\_messages\_chats | ( chat\_id ) ref [chats](#chats) (id) |
|  | fk\_messages\_users | ( sender\_id ) ref [users](#users) (id) |

### 3. orders

Заказы.

Назначение: хранит основные данные о заказе, размещенном заказчиком, на который откликаются и после различных согласований выполняют исполнители.

Особенности:

* История статусов привязывается внешним ключом из отношения statuses
* Отклики привязываются внешним ключом из отношения responses
* Выбранный исполнитель и согласованный бюджет должны устанавливаться по ходу флоу продвижения заказа по всем статусам

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬋ | id | serial | ID заказа. |
| \* ⬈ | client\_id | integer | ID пользователя, разместившего заказ. |
| \* | title | varchar(128) | Название заказа. |
| \* | description | text | Описание заказа, которое задаётся пользователем. |
| \* | completion\_time | bigint | Время, отведённое на выполнение заказа в наносекундах. Максимум при использовании int64 - 292 года. |
|  | cost | integer | Стоимость заказа. Если NULL - считать стоимость договорной. Используется только для показа в карточке заказа и никогда - для расчётов. |
| \* | is\_active | boolean DEFAULT true | Активен ли заказ (может быть неактивен, если заказ был скрыт администратором или удалён заказчиком). |
| ⬈ | freelancer\_id | integer DEFAULT NULL | ID выбранного исполнителя. Пока исполнитель не выбран, равен NULL. |
| \* | budget | integer DEFAULT 0 | Бюджет сделки, который резервируется в качестве гарантии оплаты услуг исполнителя. Может отличаться от cost. |
| \* | created\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата создания записи. |
| \* | updated\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата обновления записи. |

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | pk\_orders | ON id |

##### Внешние ключи

| Type | Name | On |
| --- | --- | --- |
|  | fk\_orders\_users | ( client\_id ) ref [users](#users) (id) |
|  | fk\_orders\_users\_1 | ( freelancer\_id ) ref [users](#users) (id) |

### profiles

Профили пользователей.

Назначение: профиль привязываемый к конкретному пользователю из отношения users.

Особенности:

* Отзывы на профиль привязываются внешним ключом из отношения reviews

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬋ | id | serial | ID профиля. |
| \* ⬈ | user\_id | integer | ID владельца профиля. |
| \* | role | enum(‘client’,‘freelancer’) | Роль (заказчик или исполнитель). |
| \* | rating | real DEFAULT 0 | Рейтинг профиля, основан на отзывах. |
| \* | description | text | Пользовательское поле “О себе”. |
| \* | created\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата создания записи. |
| \* | updated\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата обновления записи. |

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | pk\_profiles | ON id |

##### Внешние ключи

| Type | Name | On |
| --- | --- | --- |
|  | fk\_profiles\_users | ( user\_id ) ref [users](#users) (id) |

### responses

Отклики исполнителей на заказы.

Назначение: сохраняет данные об отклике исполнителя на заказ.

Особенности:

* На отклик обязательно создается один чат, chat\_id - его идентификатор

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬈ | order\_id | integer | ID заказа, на который оставлен отклик. |
| \* 🔑 ⬈ | freelancer\_id | integer | ID пользователя, оставившего отклик. |
| \* 🔑 | chat\_id | integer | ID переписки (чата), созданного в рамках отклика. |
| \* | is\_active | boolean DEFAULT true | Активен ли отклик. |
| \* | created\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата создания записи. |
| \* | updated\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата редактирования записи. |

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | pk\_responses | ON order\_id, freelancer\_id, chat\_id |

##### Внешние ключи

| Type | Name | On |
| --- | --- | --- |
|  | fk\_responses\_users | ( freelancer\_id ) ref [users](#users) (id) |
|  | fk\_responses\_orders | ( order\_id ) ref [orders](#orders) (id) |

### reviews

Отзывы пользователей.

Назначение: отзыв за авторством пользователя из отношения users на конкретный профиль из отношения profiles.

Особенности:

* Отзыв оставляется именно юзером, но на конкретный профиль, так как рейтинг заказчика и исполнителя у одного юзера может отличаться

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 | id | serial | ID отзыва. |
| \* ⬈ | author\_id | integer | ID пользователя, оставившего отзыв. |
| \* ⬈ | profile\_id | integer | ID профиля, на который оставлен отзыв. |
| \* | score | enum(‘1’,‘2’,‘3’,‘4’,‘5’) | Оценка (от 1 до 5). |
|  | content | text | Текст отзыва. |
| \* | created\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата создания записи. |

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | pk\_reviews | ON id |

##### Внешние ключи

| Type | Name | On |
| --- | --- | --- |
|  | fk\_reviews\_users | ( author\_id ) ref [users](#users) (id) |
|  | fk\_reviews\_profiles | ( profile\_id ) ref [profiles](#profiles) (id) |

### statuses

История изменения статусов заказов.

Назначение: статус для отдельного заказа с его порядковым номером.

Особенности:

* Статус заказа с большим sequential\_number есть его актуальный статус

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬈ | order\_id | integer | ID заказа, для которого присвоен статус. |
| \* 🔑 | sequential\_number | integer | Порядковый номер статуса по порядку. |
| \* | title | enum(‘beginning’,‘negotiation’,‘budgeting’,‘work’,‘reviews’,‘finished’,‘dispute’) DEFAULT ‘beginning’ | Название статуса. |
|  | extra\_info | jsonb | Дополнительная информация о статусе, если нужно. |
| \* | created\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата создания записи. |

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | pk\_statuses | ON order\_id, sequential\_number |

##### Внешние ключи

| Type | Name | On |
| --- | --- | --- |
|  | fk\_statuses\_orders | ( order\_id ) ref [orders](#orders) (id) |

### users

Таблица пользователей.

Назначение: сохраняет данные о пользователе, к которому привязываются отдельные профили.

Особенности:

* email выступает в роли уникального идентификатора
* system\_role позволяет выделять отдельные категории специфичных юзеров на сайте

| Idx | Name | Data Type | Description |
| --- | --- | --- | --- |
| \* 🔑 ⬋ | id | serial | ID пользователя. |
| \* 🔍 | email | varchar(256) | Электронная почта пользователя, должна быть уникальной. |
| \* | public\_name | varchar(64) | Публичное имя пользователя (ФИО или ФИ). |
| \* | password | varchar(256) | Пароль в захэшированном виде. |
| \* | balance | integer DEFAULT 0 | Количество средств на счету у пользователя. |
| \* | system\_role | enum(‘user’,‘admin’) DEFAULT ‘user’ | Системная роль (пользователь, администратор, модератор и т.п.). Определяет возможность пользователя авторизоваться и попасть на служебные страницы. |
| \* | is\_active | boolean DEFAULT true | Активен ли аккаунт пользователя в данный момент. |
| \* | created\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата создания записи. |
| \* | updated\_at | timestamp DEFAULT CURRENT\_TIMESTAMP | Дата обновления документа. |

##### 

##### Индексы

| Type | Name | On |
| --- | --- | --- |
| 🔑 | pk\_users | ON id |
| 🔍 | unq\_users | ON email |

**Оценка объёма информации, хранимой в реляционной модели**

Users:  
 reviews:  
 id: serial, V = 4b  
 author\_id: integer, V = 4b  
 profile\_id: integer, V = 4b  
 score: enum, V = 1b  
 content: text, V = 200b  
 created\_at: timestamp, V = 8b  
 Итого: 4 + 4 + 4 + 1 + 200 + 8 = 221b

 profiles:  
 id: serial, V = 4b  
 user\_id: integer, V = 4b  
 role: enum, V = 15b  
 rating: real, V = 4b  
 description: text, V = 500b  
 created\_at: timestamp, V = 8b  
 updated\_at: timestamp, V = 8b  
 Итого: 4 + 4 + 15 + 4 + 500 + 8 + 8 = 543b

 users:  
 id: serial, V = 4b  
 email: varchar(256), V = 30b  
 public\_name: varchar(64), V = 30b  
 password: varchar(256), V = 60b  
 balance: integer, V = 4b  
 system\_role: enum, V = 15b  
 is\_active: boolean, V = 1b  
 created\_at: timestamp, V = 8b  
 updated\_at: timestamp, V = 8b  
 Итого: 4 + 30 + 30 + 60 + 4 + 15 + 1 + 8 + 8 = 160b

Фактический объём users: NR ⋅ 221 + NP ⋅ 543 + NU ⋅ 160, где NR – количество отзывов, NP – количество профилей, NU – количество пользователей.  
Так как в среднем на профиль приходится 10 отзывов, а на пользователя – 2 профиля (фрилансер и заказчик), то объём: NU ⋅ (20 ⋅ 221 + 2 ⋅ 543 + 160) = (NU ⋅ 5666)b

Chats:  
 messages:  
 id: serial, V = 4b  
 chat\_id: integer, V = 4b  
 sender\_id: integer, V = 4b  
 is\_system: boolean, V = 1b  
 text: text, V = 500b  
 created\_at: timestamp, V = 8b  
 Итого: 4 + 4 + 4 + 1 + 500 + 8 = 521b

 chats:  
 id: serial, V = 4b  
 order\_id: integer, V = 4b  
 client\_id: integer, V = 4b  
 freelancer\_id: integer, V = 4b  
 is\_active: boolean, V = 1b  
 created\_at: timestamp, V = 8b  
 updated\_at: timestamp, V = 8b  
 Итого: 4 + 4 + 4 + 4 + 1 + 8 + 8 = 33b

Фактический объём chats: NM ⋅ 521 + NC ⋅ 33, где NM – количество сообщений, NC – количество чатов.  
Так как в среднем на чат приходится 50 сообщений, то объём: NC ⋅ (50 ⋅ 521 + 33) = (NC ⋅ 26083)b

Orders:  
 statuses:  
 order\_id: integer, V = 4b  
 sequential\_number: integer, V = 4b  
 title: enum, V = 15b  
 extra\_info: jsonb, V = 20b  
 created\_at: timestamp, V = 8b  
 Итого: 4 + 4 + 15 + 20 + 8 = 51b

 responses:  
 order\_id: integer, V = 4b  
 freelancer\_id: integer, V = 4b  
 chat\_id: integer, V = 4b  
 is\_active: boolean, V = 1b  
 created\_at: timestamp, V = 8b  
 updated\_at: timestamp, V = 8b  
 Итого: 4 + 4 + 4 + 1 + 8 + 8 = 29b

 orders:  
 id: serial, V = 4b  
 client\_id: integer, V = 4b  
 title: varchar(128), V = 50b  
 description: text, V = 1000b  
 completion\_time: bigint, V = 8b  
 cost: integer, V = 4b  
 is\_active: boolean, V = 1b  
 freelancer\_id: integer, V = 4b  
 budget: integer, V = 4b  
 created\_at: timestamp, V = 8b  
 updated\_at: timestamp, V = 8b  
 Итого: 4 + 4 + 50 + 1000 + 8 + 4 + 1 + 4 + 4 + 8 + 8 = 1095b

Фактический объём orders: NS ⋅ 51 + NRes ⋅ 29 + NO ⋅ 1095, где NS – количество статусов, NRes – количество откликов, NO – количество заказов.  
Так как в среднем на заказ приходится 5 откликов и 5 статусов, то объём: NO ⋅ (5 ⋅ 51 + 5 ⋅ 29 + 1095) = (NO ⋅ 1495)b

Фактический объём модели: NU ⋅ 5666 + NO ⋅ 1495 + NC ⋅ 26083, где NU – количество пользователей, NO – количество заказов, NC – количество чатов.  
Выразим всё через количество заказов аналогично нереляционной модели:  
V(NO) = (82577 ⋅ NO)bytes

**Избыточность данных**

Для вычисления «чистого» объёма данных исключим из расчетов дублирующуюся и служебную информацию, тогда:

Users:  
 reviews:  
 score: enum, V = 1b  
 content: text, V = 200b  
 Итого: 201b

 profiles:  
 role: enum, V = 15b  
 rating: real, V = 4b  
 description: text, V = 500b  
 Итого: 519b

 users:  
 email: varchar(256), V = 30b  
 public\_name: varchar(64), V = 30b  
 password: varchar(256), V = 60b  
 balance: integer, V = 4b  
 system\_role: enum, V = 15b  
 Итого: 30 + 30 + 60 + 4 + 15 = 139b

«Чистый» объём users: NR ⋅ 201 + NP ⋅ 519 + NU ⋅ 139, где NR – количество отзывов, NP – количество профилей, NU – количество пользователей.  
Так как в среднем на профиль приходится 10 отзывов, а на пользователя – 2 профиля (фрилансер и заказчик), то объём: NU ⋅ (20 ⋅ 201 + 2 ⋅ 519 + 139) = (NU ⋅ 5197)b

Chats:  
 messages:  
 is\_system: boolean, V = 1b  
 text: text, V = 500b  
 Итого: 501b

 chats:  
 is\_active: boolean, V = 1b  
 Итого: 1b

«Чистый» объём chats: NM ⋅ 501 + NC ⋅ 1, где NM – количество сообщений, NC – количество чатов.  
Так как в среднем на чат приходится 50 сообщений, то объём: NC ⋅ (50 ⋅ 501 + 1) = (NC ⋅ 25051)b

Orders:  
 statuses:  
 title: enum, V = 15b  
 extra\_info: jsonb, V = 20b  
 Итого: 35b

 responses:  
 is\_active: boolean, V = 1b  
 Итого: 1b

 orders:  
 title: varchar(128), V = 50b  
 description: text, V = 1000b  
 completion\_time: bigint, V = 8b  
 cost: integer, V = 4b  
 budget: integer, V = 4b  
 Итого: 50 + 1000 + 8 + 4 + 4 = 1066b

«Чистый» объём orders: NS ⋅ 35 + NRes ⋅ 1 + NO ⋅ 1066, где NS – количество статусов, NRes – количество откликов, NO – количество заказов.  
Так как в среднем на заказ приходится 5 откликов и 5 статусов, то объём: NO ⋅ (5 ⋅ 35 + 5 ⋅ 1 + 1066) = (NO ⋅ 1246)b

«Чистый» объём модели: NU ⋅ 5197 + NO ⋅ 1246 + NC ⋅ 25051, где NU – количество пользователей, NO – количество заказов, NC – количество чатов.  
Выразим всё через количество заказов аналогично нереляционной модели:  
Vc(NO) = (78998 ⋅ NO)bytes

Избыточность:  
R(NO) = V(NO) / Vc(NO) = 82577 / 78998 = 1.045

**Направление роста модели при увеличении количества объектов каждой сущности**

При увеличении количества объектов любой из сущностей модель будет расти линейно.

Примеры данных

*Таблица users*

| id | email | public\_name | password | balance | system\_role | is\_active | created\_at | updated\_at |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | [ivanov@example.com](https://mailto:ivanov@example.com/) | Иван Иванов | hash1 | 1000 | user | true | 2023-10-01 10:00:00 | 2023-10-01 10:00:00 |
| 2 | [petrov@example.com](https://mailto:petrov@example.com/) | Петр Петров | hash2 | 500 | user | true | 2023-10-01 10:05:00 | 2023-10-01 10:05:00 |
| 3 | [admin@example.com](https://mailto:admin@example.com/) | Админ Админов | hash3 | 0 | admin | true | 2023-10-01 10:10:00 | 2023-10-01 10:10:00 |

*Таблица profiles*

| id | user\_id | role | rating | description | created\_at | updated\_at |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | client | 4.5 | Заказчик Иван | 2023-10-01 10:00:00 | 2023-10-01 10:00:00 |
| 2 | 2 | freelancer | 4.7 | Фрилансер Петр | 2023-10-01 10:05:00 | 2023-10-01 10:05:00 |

*Таблица orders*

| id | client\_id | title | description | completion\_time | cost | is\_active | freelancer\_id | budget | created\_at | updated\_at |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | Разработка сайта | Создание лендинга | 1000000000 | 1000 | true | 2 | 1000 | 2023-10-01 10:00:00 | 2023-10-01 10:00:00 |

*Таблица chats*

| id | order\_id | client\_id | freelancer\_id | is\_active | created\_at | updated\_at |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 2 | true | 2023-10-01 10:00:00 | 2023-10-01 10:00:00 |

*Таблица messages*

| id | chat\_id | sender\_id | is\_system | text | created\_at |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | false | Здравствуйте! | 2023-10-01 10:00:00 |
| 2 | 1 | 2 | false | Добрый день! | 2023-10-01 10:05:00 |

*Таблица responses*

| order\_id | freelancer\_id | chat\_id | is\_active | created\_at | updated\_at |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 1 | true | 2023-10-01 10:00:00 | 2023-10-01 10:00:00 |

*Таблица reviews*

| id | author\_id | profile\_id | score | content | created\_at |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 2 | 5 | Отлично! | 2023-10-01 10:00:00 |

*Таблица statuses*

| order\_id | sequential\_number | title | extra\_info | created\_at |
| --- | --- | --- | --- | --- |
| 1 | 1 | beginning | {} | 2023-10-01 10:00:00 |

**Примеры запросов**

**Сценарий: Регистрация пользователя**

Основной сценарий:

1**.** Проверка уникальности email:

* SELECT id FROM users WHERE email = ‘ivanov@example.com’;
  + Количество запросов: 1
  + Задействованные коллекции: users

1. Создание нового пользователя:

* INSERT INTO users (email, public\_name, password, system\_role, is\_active) VALUES (‘ivanov@example.com’, ‘Иван Иванов’, ‘hash1’, ‘user’, true);
  + Количество запросов: 1
  + Задействованные коллекции: users

1. Создание профиля пользователя:

* INSERT INTO profiles (user\_id, role, description) VALUES (1, ‘client’, ‘Заказчик Иван’);
  + Количество запросов: 1
  + Задействованные коллекции: profiles

Итого:

* Количество запросов: 3
* Задействованные коллекции: users, profiles

**Сценарий: Авторизация пользователя**

Основной сценарий:

1. Проверка пользователя:

* SELECT id, password FROM users WHERE email = ‘ivanov@example.com’;
  + Количество запросов: 1
  + Задействованные коллекции: users

1. Получение профиля пользователя:

* SELECT role FROM profiles WHERE user\_id = 1;
  + Количество запросов: 1
  + Задействованные коллекции: profiles

Итого:

* Количество запросов: 2
* Задействованные коллекции: users, profiles

**Сценарий: Редактирование профиля**

Основной сценарий:

1. Обновление данных пользователя:

* UPDATE users SET public\_name = ‘Иван Иванович’ WHERE id = 1;
  + Количество запросов: 1
  + Задействованные коллекции: users

2. Обновление данных профиля:

* UPDATE profiles SET description = ‘Новое описание’ WHERE user\_id = 1;
  + Количество запросов: 1
  + Задействованные коллекции: profiles

Итого:

* Количество запросов: 2
* Задействованные коллекции**:** users, profiles

**Сценарий: Смена роли**

Основной сценарий:

1. Обновление роли в профиле:

* UPDATE profiles SET role = ‘freelancer’ WHERE user\_id = 1;
  + Количество запросов: 1
  + Задействованные коллекции: profiles

Итого:

* Количество запросов: 1
* Задействованные коллекции: profiles

**Сценарий: Заказчик публикует новый заказ**

Основной сценарий:

1. Создание заказа**:**

* INSERT INTO orders (client\_id, title, description, completion\_time, cost, is\_active) VALUES (1, ‘Разработка сайта’, ‘Создание лендинга’, 1000000000, 1000, true);
  + Количество запросов: 1
  + Задействованные коллекции: orders

2. Создание статуса заказа:

* INSERT INTO statuses (order\_id, sequential\_number, title) VALUES (1, 1, ‘beginning’);
  + Количество запросов: 1
  + Задействованные коллекции: statuses

Итого:

* Количество запросов: 2
* Задействованные коллекции: orders, statuses

**Сценарий: Заказчик редактирует заказ**

Основной сценарий:

1. Обновление данных заказа:

* UPDATE orders SET title = ‘Обновленный проект’, description = ‘Новое описание’ WHERE id = 1;
  + Количество запросов: 1
  + Задействованные коллекции: orders

Итого:

* Количество запросов: 1
* Задействованные коллекции: orders

**Сценарий: Стороны согласовали условия и работа началась**

Основной сценарий:

1. Обновление статуса заказа:

* INSERT INTO statuses (order\_id, sequential\_number, title) VALUES (1, 2, ‘negotiation’);
  + Количество запросов: 1
  + Задействованные коллекции: statuses

Итого:

* Количество запросов: 1
* Задействованные коллекции: statuses

**Сценарий: Стороны общаются в процессе выполнения заказа**

Основной сценарий:

1. Добавление сообщения в чат:

* INSERT INTO messages (chat\_id, sender\_id, text) VALUES (1, 1, ‘Здравствуйте!’);
  + Количество запросов: 1
  + Задействованные коллекции: messages

Итого:

* Количество запросов: 1
* Задействованные коллекции: messages

**Сценарий: Завершение заказа**

Основной сценарий:

1. Обновление статуса заказа:

* INSERT INTO statuses (order\_id, sequential\_number, title) VALUES (1, 3, ‘finished’);
  + Количество запросов: 1
  + Задействованные коллекции: statuses

Итого:

* Количество запросов: 1
* Задействованные коллекции: statuses

**Сценарий: Стороны оставляют отзывы**

Основной сценарий:

1. Добавление отзыва:

* INSERT INTO reviews (author\_id, profile\_id, score, content) VALUES (1, 2, 5, ‘Отлично!’);
  + Количество запросов: 1
  + Задействованные коллекции: reviews

Итого:

* Количество запросов: 1
* Задействованные коллекции: reviews

**Сценарий: Исполнитель смотрит свои отклики**

Основной сценарий:

1.Получение откликов исполнителя:

* SELECT \* FROM responses WHERE freelancer\_id = 2;
  + Количество запросов: 1
  + Задействованные коллекции: responses

Итого:

* Количество запросов: 1
* Задействованные коллекции: responses

**Сценарий: Исполнитель отзывает отклик**

Основной сценарий:

1. Обновление активности отклика:

* UPDATE responses SET is\_active = false WHERE order\_id = 1 AND freelancer\_id = 2;
  + Количество запросов: 1
  + Задействованные коллекции: responses

Итого:

* Количество запросов: 1
* Задействованные коллекции: responses

**Сценарий: Исполнитель общается в чате до согласования**

Основной сценарий:

1. Добавление сообщения в чат:

* INSERT INTO messages (chat\_id, sender\_id, text) VALUES (1, 2, ‘Добрый день!’);
  + Количество запросов: 1
  + Задействованные коллекции: messages

Итого:

* Количество запросов: 1
* Задействованные коллекции: messages

**Сценарий: Заказчик согласовывает условия в чате**

Основной сценарий:

1. Добавление системного сообщения о согласовании:

* INSERT INTO messages (chat\_id, is\_system, text) VALUES (1, true, ‘Условия согласованы.’);
  + Количество запросов: 1
  + Задействованные коллекции: messages

Итого:

* Количество запросов: 1
* Задействованные коллекции: messages

**Сценарий: Исполнитель согласовывает условия в чате**

Основной сценарий:

1. Добавление системного сообщения о согласовании:

* INSERT INTO messages (chat\_id, is\_system, text) VALUES (1, true, ‘Условия согласованы.’);
  + Количество запросов: 1
  + Задействованные коллекции: messages

Итого:

* Количество запросов: 1
* Задействованные коллекции: messages

**Сценарий: Подсчёт статистики**

Основной сценарий:

1. Получение количества заказов:

* SELECT COUNT(\*) FROM orders;
  + Количество запросов: 1
  + Задействованные коллекции: orders

2. Получение количества пользователей:

* SELECT COUNT(\*) FROM users;
  + Количество запросов: 1
  + Задействованные коллекции: users

3. Получение количества споров:

* SELECT COUNT(\*) FROM messages WHERE is\_system = true AND text LIKE ‘%Спор%’;
  + Количество запросов: 1
  + Задействованные коллекции: messages

Итого:

* Количество запросов: 3
* Задействованные коллекции: orders, users, messages

**Сценарий: Исполнитель ищет заказы**

Основной сценарий:

1. Получение списка активных заказов:

* SELECT \* FROM orders WHERE is\_active = true;
  + Количество запросов: 1
  + Задействованные коллекции: orders

Итого:

* Количество запросов: 1
* Задействованные коллекции**:** orders

**Сценарий: Исполнитель откликается на заказ**

Основной сценарий:

1. Создание отклика:

* INSERT INTO responses (order\_id, freelancer\_id, chat\_id, is\_active) VALUES (1, 2, 1, true);
  + Количество запросов: 1
  + Задействованные коллекции: responses

Итого:

* Количество запросов: 1
* Задействованные коллекции: responses

**Сценарий: Исполнитель выводит средства**

Основной сценарий:

1. Обновление баланса пользователя:

* UPDATE users SET balance = balance - 500 WHERE id = 2;
  + Количество запросов: 1
  + Задействованные коллекции: users

Итого:

* Количество запросов: 1
* Задействованные коллекции: users

**Сценарий: Экспорт данных**

Основной сценарий:

1. Экспорт данных:

* SELECT \* FROM users; SELECT \* FROM profiles; SELECT \* FROM orders; SELECT \* FROM chats; SELECT \* FROM messages; SELECT \* FROM responses; SELECT \* FROM reviews; SELECT \* FROM statuses;
  + Количество запросов: 8
  + Задействованные коллекции: Все таблицы

Итого:

* Количество запросов: 8

**2.3 Сравнение моделей**

*Удельный объём информации*

| Параметр | Нереляционная модель | Реляционная модель |
| --- | --- | --- |
| “Грязный” объём | 86625 ⋅ NO | 82577 ⋅ NO |
| “Чистый” объём | 84071 ⋅ NO | 78998 ⋅ NO |
| Избыточность | 1.03 | 1.045 |

*Запросы по отдельным юзкейсам*

| № | Название | Нереляционная, запросы | | Реляционная, запросы | Нереляционная, коллекции | Реляционная, коллекции |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Регистрация пользователя | | 2 | 3 | 1 (users) | 2 (users, profiles) |
| 2 | Авторизация пользователя | | 2 | 2 | 1 (users) | 2 (users, profiles) |
| 3 | Редактирование профиля | | 1 | 2 | 1 (users) | 2 (users, profiles) |
| 4 | Смена роли | | 1 | 1 | 1 (users) | 1 (profiles) |
| 5 | Заказчик публикует новый заказ | | 1 | 2 | 1 (orders) | 2 (orders, statuses) |
| 6 | Заказчик редактирует заказ | | 1 | 1 | 1 (orders) | 1 (orders) |
| 7 | Стороны согласовали условия и работа началась | | 1 | 1 | 1 (orders) | 1 (statuses) |
| 8 | Стороны общаются в процессе выполнения заказа | | 1 | 1 | 1 (orders) | 1 (messages) |
| 9 | Завершение заказа | | 1 | 1 | 1 (orders) | 1 (statuses) |
| 10 | Стороны оставляют отзывы | | 1 | 1 | 1 (users) | 1 (reviews) |
| 11 | Исполнитель смотрит свои отклики | | 1 | 1 | 1 (orders) | 1 (responses) |
| 12 | Исполнитель отзывает отклик | | 1 | 1 | 1 (orders) | 1 (responses) |
| 13 | Исполнитель общается в чате до согласования | | 1 | 1 | 1 (chats) | 1 (messages) |
| 14 | Заказчик согласовывает условия в чате | | 1 | 1 | 1 (chats) | 1 (messages) |
| 15 | Исполнитель согласовывает условия в чате | | 1 | 1 | 1 (chats) | 1 (messages) |
| 16 | Подсчёт статистики | | 3 | 3 | 3 (orders, users, chats) | 3 (orders, users, messages) |
| 17 | Исполнитель ищет заказы | | 1 | 1 | 1 (orders) | 1 (orders) |
| 18 | Исполнитель откликается на заказ | | 1 | 1 | 1 (orders) | 1 (responses) |
| 19 | Исполнитель выводит средства | | 1 | 1 | 1 (users) | 1 (users) |

**Вывод**

Нереляционная модель требует больше объёма, чем реляционная. Количество запросов к нереляционной немного меньше, чем к реляционной (в основном по всем юзкейсам требуется один запрос, кроме регистрации/авторизации). Избыточность у реляционной немногим меньше, чем у нереляционной. В целом, что SQL, что noSQL одинаково хорошо подходят к задаче. Здесь мало слабоструктурированных данных (только сообщения в переписке).

# 3. Разработанное приложение

**A. Краткое описание**

Проект реализован на языке Go, по принципам чистой архитектуры, которая предполагает строгую изоляцию бизнес-логики от деталей реализации — это база данных, веб-сервер и маршрутизация. Логика проекта делится на слои: API-контроллеры, слой сервисов, слой репозиториев. В каждом из этих слоёв реализована строгая зависимость только «вниз». Это позволяет модифицировать, тестировать, масштабировать отдельные компоненты.

Контроллеры, реализованные в директории internal/api, отвечают за приём HTTP-запросов, валидацию входных данных, передачу запросов в бизнес-логику. Маршрутизация организована с использованием net/http. Например, все маршруты, относящиеся к пользователям, заказам или чатам, сгруппированы отдельно, и каждому из них сопоставлены соответствующие обработчики. Контекст Go применяется для передачи таймаутов, пользовательских данных и других параметров по цепочке вызовов.

Слой бизнес-логики реализован в директории internal/service. Он инкапсулирует основную предметную область: создание пользователей, заказов, откликов, чатов, сообщений, пополнение баланса. Все сервисы работают через интерфейсы репозиториев, так как это обеспечивает полную независимость от способов хранения данных. Таким образом, при необходимости можно заменить MongoDB на другую нереляционную СУБД, не переписывая бизнес-логику. Это упрощает написание юнит-тестов, поскольку можно использовать мок-реализации.

Слой репозиториев, находится в internal/repository. Он инкапсулирует работу с базой данных MongoDB. Описаны методы для операций чтения, вставки, обновления и удаления документов в коллекциях users, orders, chats и messages. В коде используется единый экземпляр mongo.Client, созданный на этапе запуска приложения и передаваемый во все репозитории. Для обеспечения устойчивости к ошибкам и превышению времени отклика все операции выполняются с контекстом с таймаутом. Структуры данных, которыми оперирует приложение, определены в пакете internal/model. Каждая структура снабжена bson-тегами для корректной сериализации и десериализации в формате BSON при взаимодействии с MongoDB.

Проект использует централизованную конфигурацию через файл «.env», откуда при старте приложения загружаются параметры подключения к базе данных, порт сервера, JWT-секрет и другие переменные окружения. Конфигурация обрабатывается через отдельный модуль config. Также предусмотрены методы логирования, записи ошибок и мониторинга состояния приложения, которые могут быть дополнены средствами Prometheus и Grafana.

Проект контейнеризован с использованием Docker. В корне проекта лежит файл docker-compose.yml, который описывает связку сервисов: MongoDB, backend-приложение на Go, фронтенд-приложение на React/TypeScript и NGINX-прокси. Бэкенд собирается в отдельном Dockerfile, где в первом этапе происходит сборка бинарника, а на втором этапе — упаковка в минимальный образ на базе Alpine Linux. NGINX выступает в роли реверс-прокси и маршрутизирует HTTP-запросы: API-запросы по префиксу /api отправляются на backend, а все остальные запросы обслуживаются фронтендом как статические файлы.

Фронтенд и бэкенд работают независимо. Система легко масштабируется, поскольку backend реализует независимые компоненты с чётко определёнными интерфейсами. Такой подход упрощает отладку, модификацию бизнес-логики и расширение функциональности. Вся архитектура выстроена таким образом, чтобы обеспечить отказоустойчивость, масштабируемость и поддержку в долгосрочной перспективе.

**B. использованные технологии**

backend: Go 1.24

БД: MongoDB

Frontend: HTML, CSS, JavaScript

Контейнеризация: Docker

Прокси и маршрутизация подключений: NGNIX

C. Снимки экрана приложения

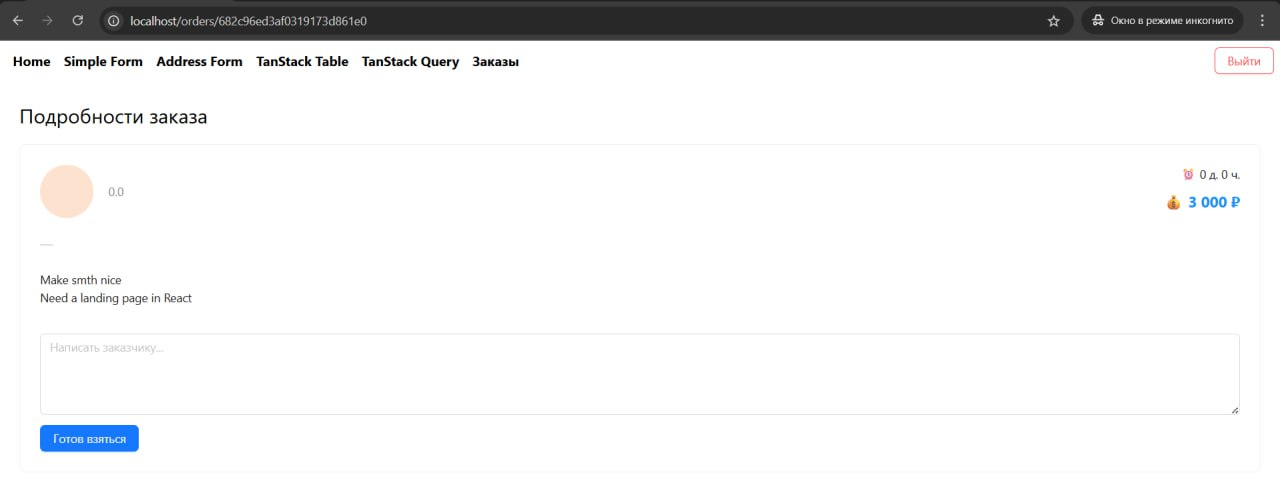
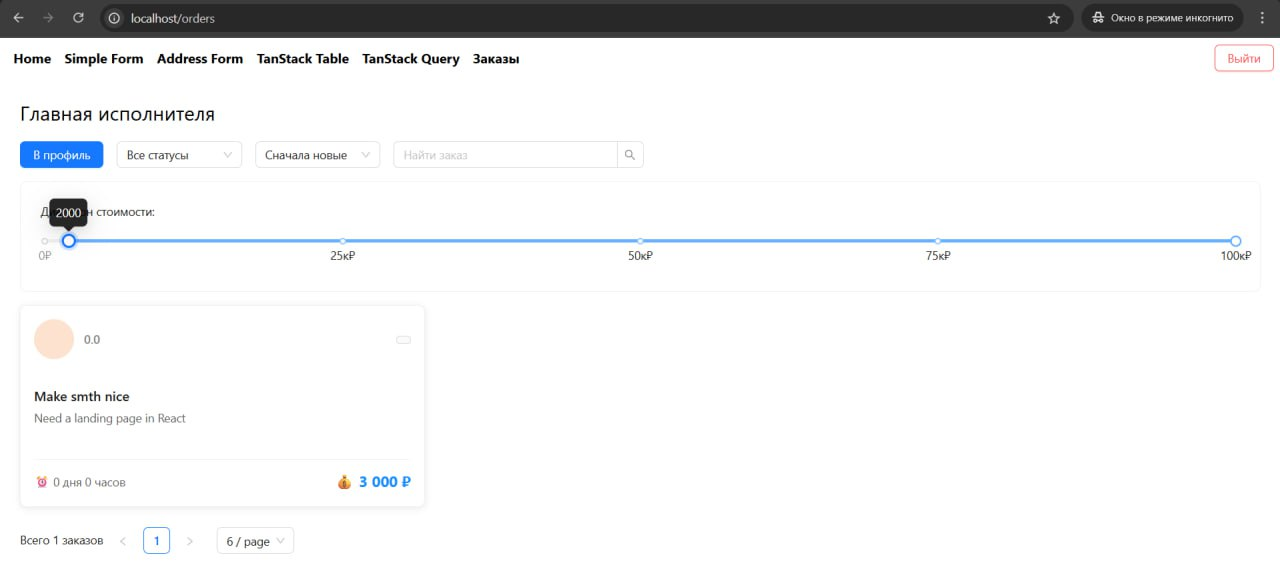
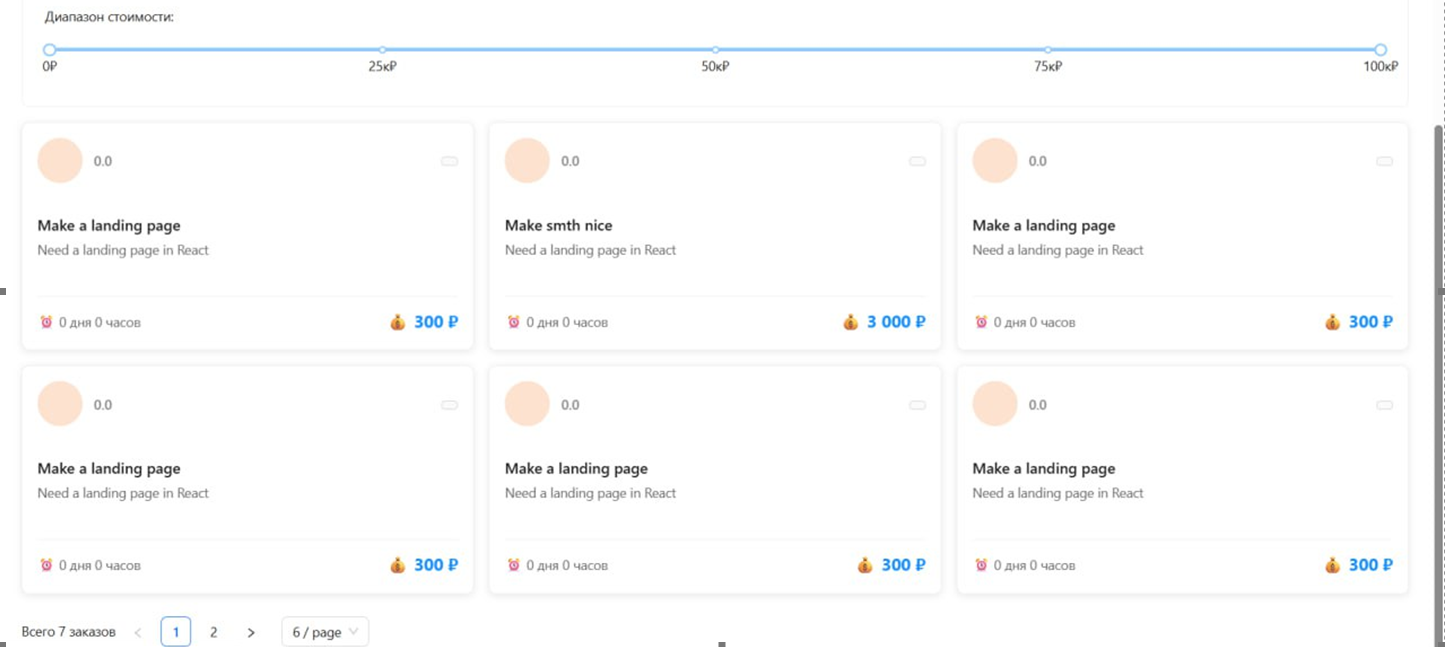
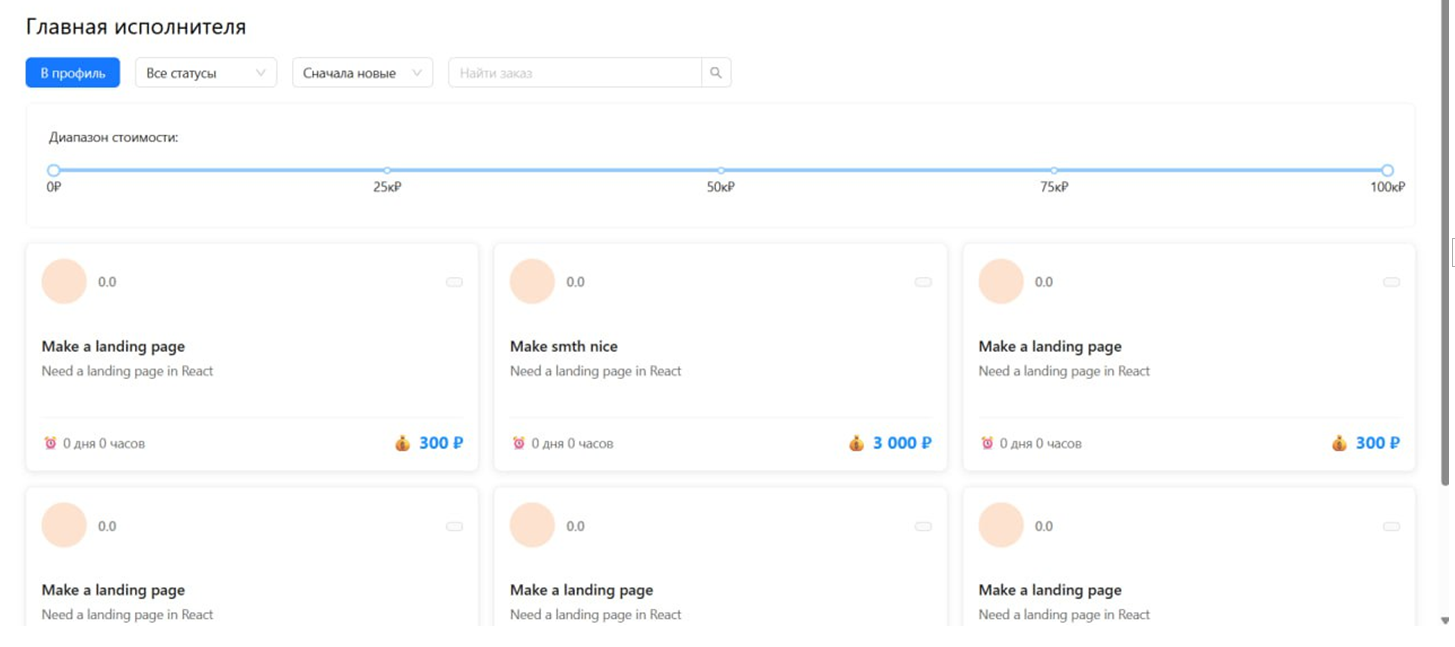


Рисунок 3.1 – подробности заказа

Рисунок 3.2 – главная страница исполнителя



Риснунок 3.3 – страница заказов



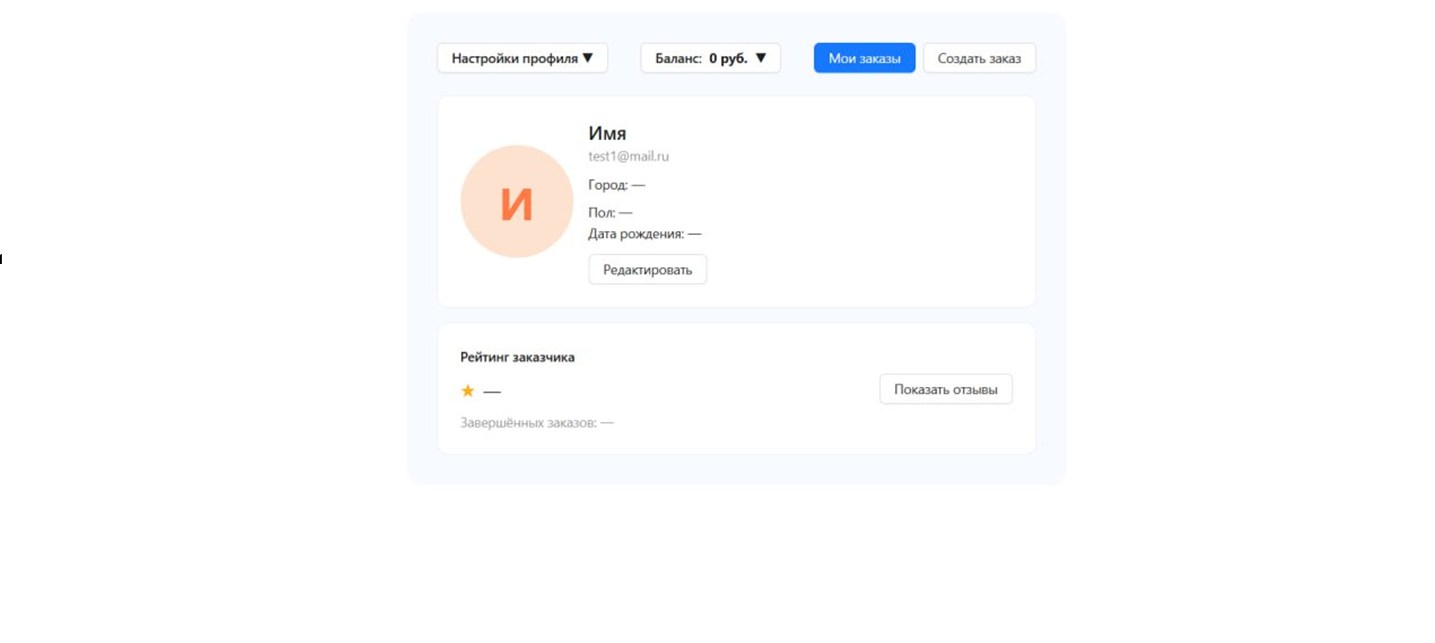


Рисунок 3.4 – страница с профилем

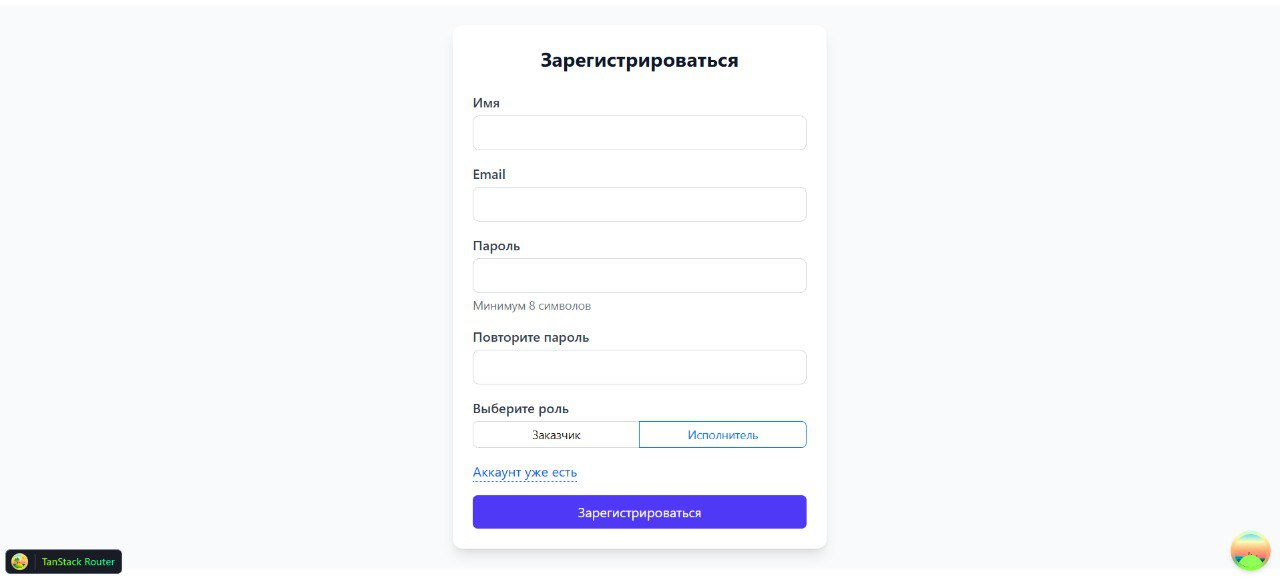


Рисунок 3.5 – страница регистрации

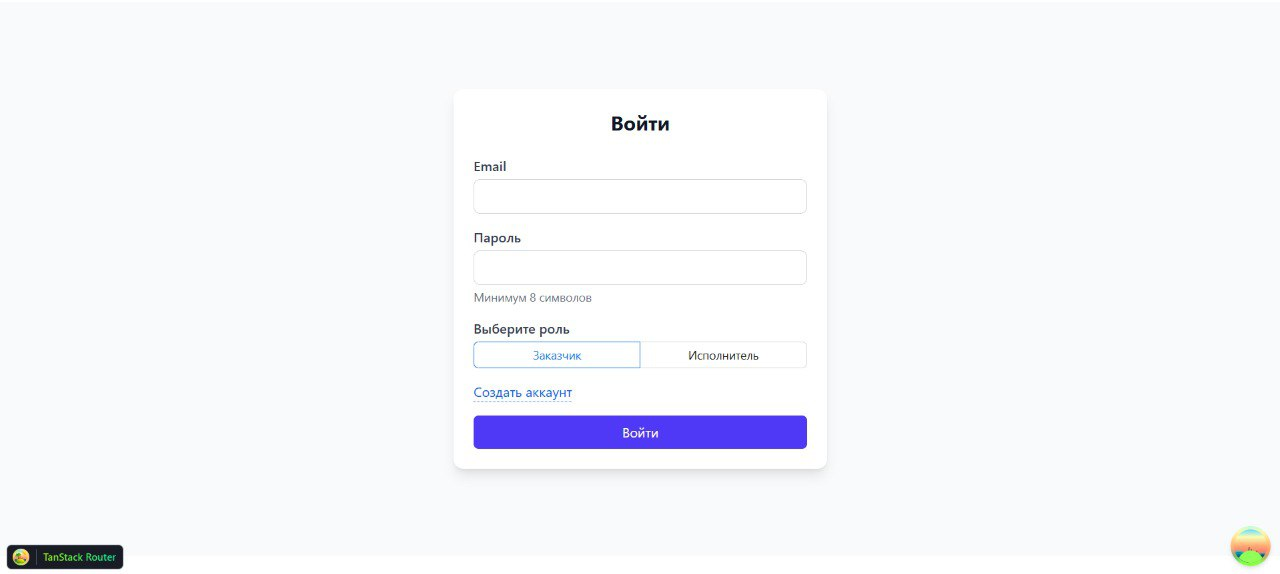


Рисунок 3.6 – страница входа в аккаунт

**5. ВЫВОДЫ**

**Достигнутые результаты**

В ходе работы было разработано приложение для фриланса, которое позволяет заказчикам находить исполнителей, исполнителям брать заказы, совершать сделки. Приложение использует нереляционную СУБД для хранения данных и построено по принципам чистой архитектуры.

Недостатки и пути улучшения полученного решения

Пока в приложении не готова возможность начать спор, чат, отображение этапов сделки и механизмы оплаты. Эти функции не удалось реализовать за отведенный срок из-за высоких времязатрат на их разработку.

**Будущее развития**

Разработка чата, этапов сделки, оплаты, спора. Разработка более глубокой системы рейтинга исполнителей и заказчиков. Создание нативного приложения.

**7. ПРИЛОЖЕНИЯ**

Документация по сборке и развёртыванию

* 1. Скачать проект из репозитория (ссылка ниже)
  2. Выполнить в корне команду docker compose up -d –build
  3. Зайти в браузере на <http://localhost:3000/>

**8. ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА**

1. Документация MongoDB https://github.com/mongodb/docs

2. Инструкция по Go <https://go.dev/tour/>

3. Репозиторий проекта <https://github.com/moevm/nosql1h25-writer>