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

КУРСОВАЯ РАБОТА

по дисциплине «Введение в нереляционные базы данных»

Тема: Реализация сервиса по продаже остатков столярного производства

Студент гр. 2300 _____ Мироевский Е.Д.

Студент гр. 2300 _____ Омуралиев И.

Студент гр. 2300 _____ Бобров П.С.

Студент гр. 2300 _____ Шумов О.Д.

Студент гр. 2300 _____ Гаранин Р.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2025

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Мироевский Е.Д. 2300

Студент Омуралиев И. 2300

Студент Бобров П.С. 2300

Студент Шумов О.Д. 2300

Студент Гаранин Р.А. 2300

Тема работы: Реализация сервиса по продаже остатков столярного производства

Исходные данные:

Задача - создать сервис, где столяры смогут размещать информацию о том, какие у них есть остатки с производства (обрезки, некондиция, брак, опилки и прочее такое - с фото, габаритами и параметрами), примерные цены, адреса, где забрать. Нужно поддержать механизм отзывов, комментариев, страницы как покупателей, так и мастеров.

Используемая база данных – Neo4j.

Содержание пояснительной записки:

«Введение», «Сценарии 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: 21.05.2025

Дата защиты реферата: 21.05.2025

Студент гр. 2300	_____	Мироевский Е.Д.
Студент гр. 2300	_____	Омуралиев И.
Студент гр. 2300	_____	Бобров П.С.
Студент гр. 2300	_____	Шумов О.Д.
Студент гр. 2300	_____	Гаранин Р.А.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках курса «Введение в нереляционные базы данных» разработан сервис «База данных остатков столярного производства» в команде. Основные свойства проекта: единый и очевидный интерфейс для всех типов пользователей, производительность приложения. В приложении используется база данных Neo4j.

SUMMARY

As part of the course "Introduction to Non-relational Databases", the Carpentry Leftover Database service was developed in the team. The main features of the project are a single and obvious interface for all types of users, and application performance. The application uses the Neo4j database.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создание высокопроизводительного сервиса по продаже остатков столярного производства с единый и очевидный интерфейс для всех типов пользователей.

Принято рещение разработать сайт, на котором мастера смогут выкладывать объявления по продаже, а покупатели смогут просмотреть данные объявления, пользователи будут иметь возможность оставлять комментарии под различными объявлениями, а также оставлять отзывы о мастерах.

1. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

1.1. Макет UI

Перед началом создания самого приложения, требуется определиться, как будет выглядеть интерфейс приложения. Главное, чтобы он был интуитивно понятен и не различался для разных типов пользователей. В результате был разработан макет пользовательского интерфейса (см. рис. 1.1-1.11).

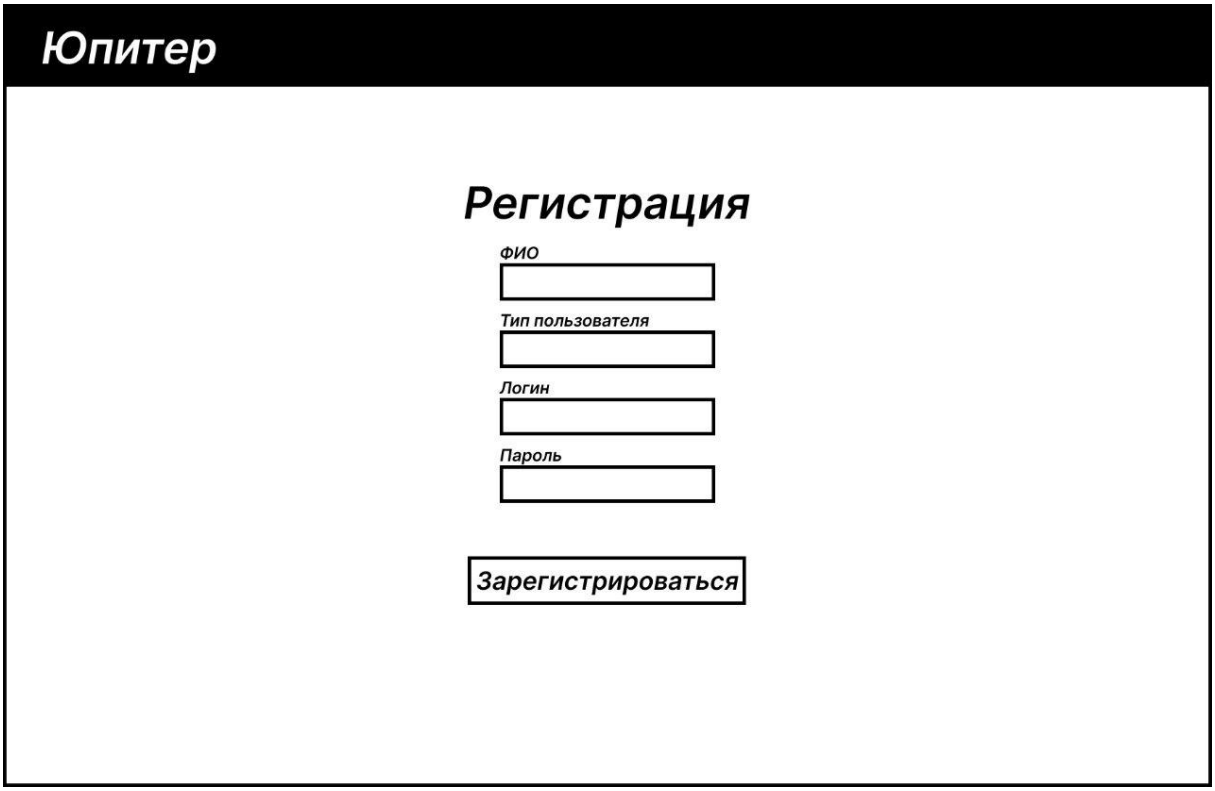

Рисунок 1.1 – Страница регистрации нового пользователя

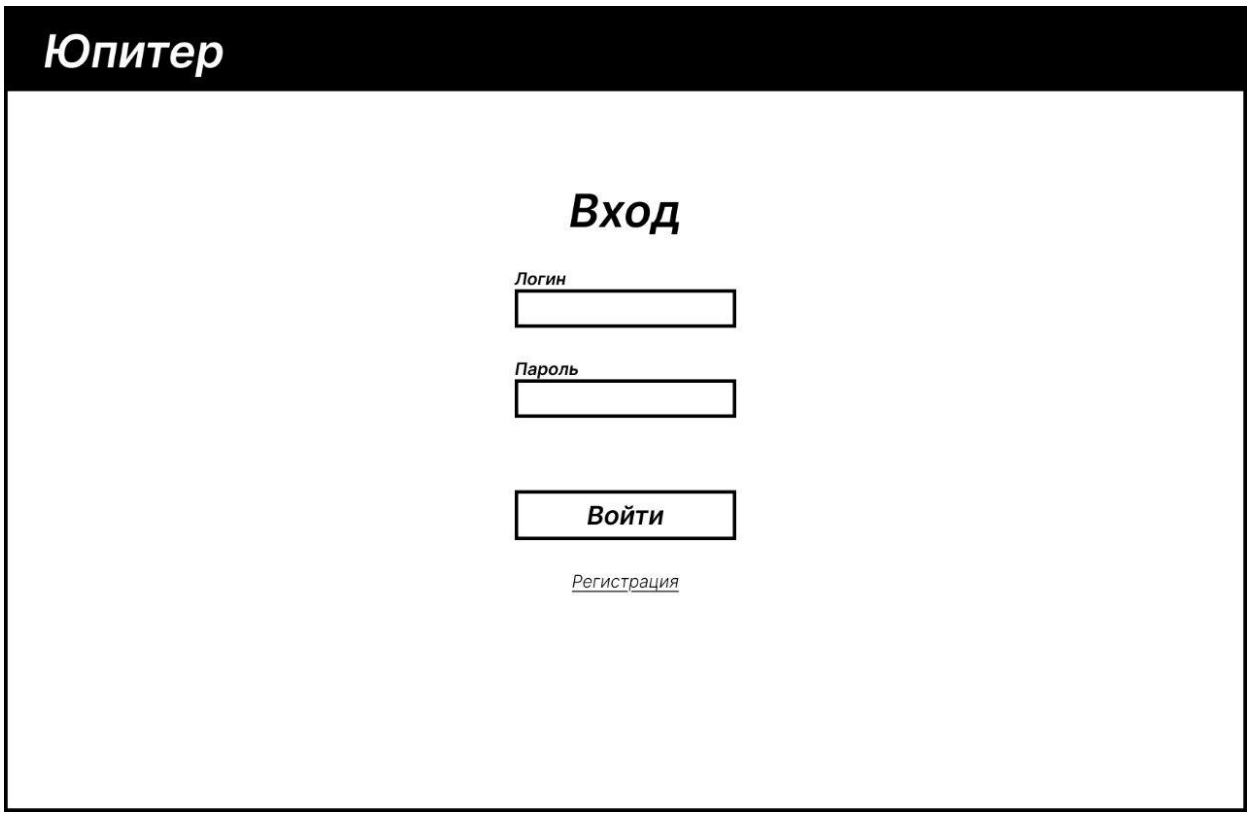

Рисунок 1.2 – Страница авторизации пользователя

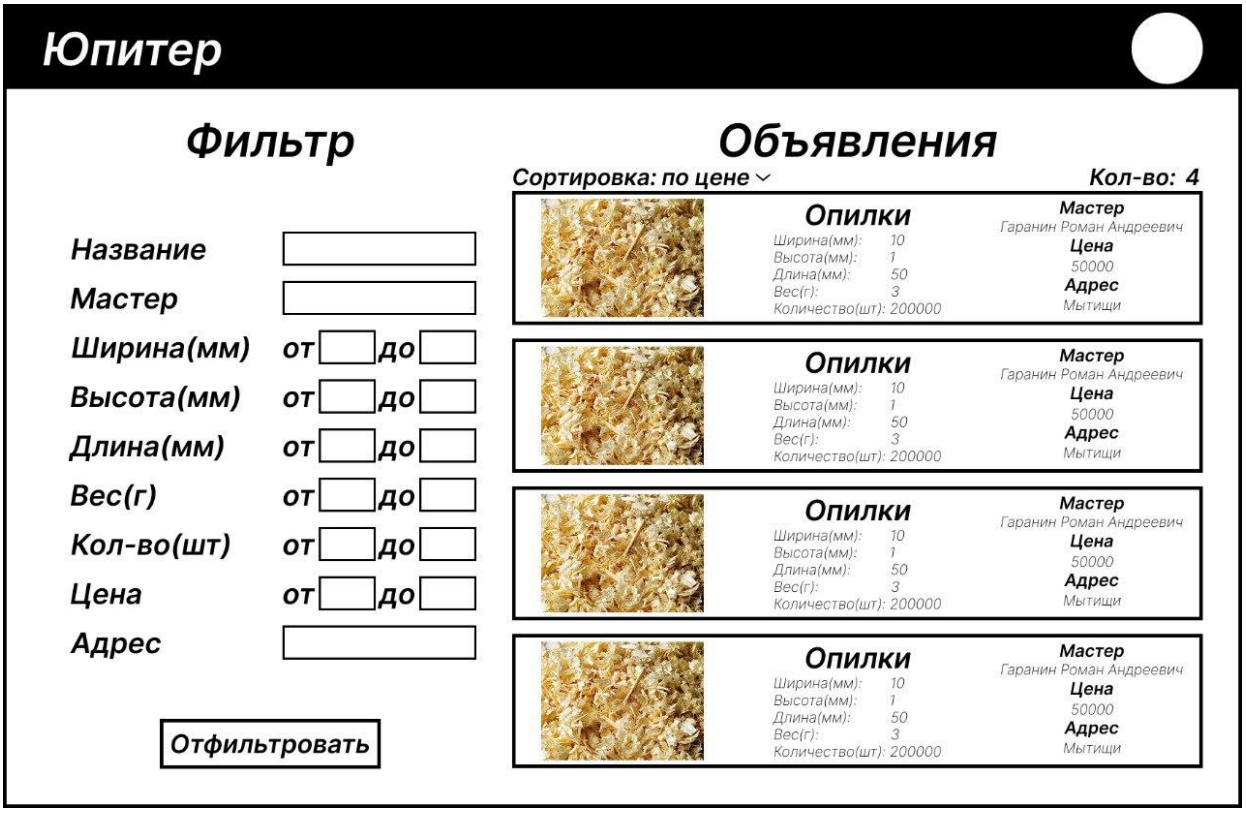

Рисунок 1.3 – Страница со всеми объявлениями и фильтрацией

Рисунок 1.4 – Страница определённого объявления с полной информацией о нём и комментариями от пользователей

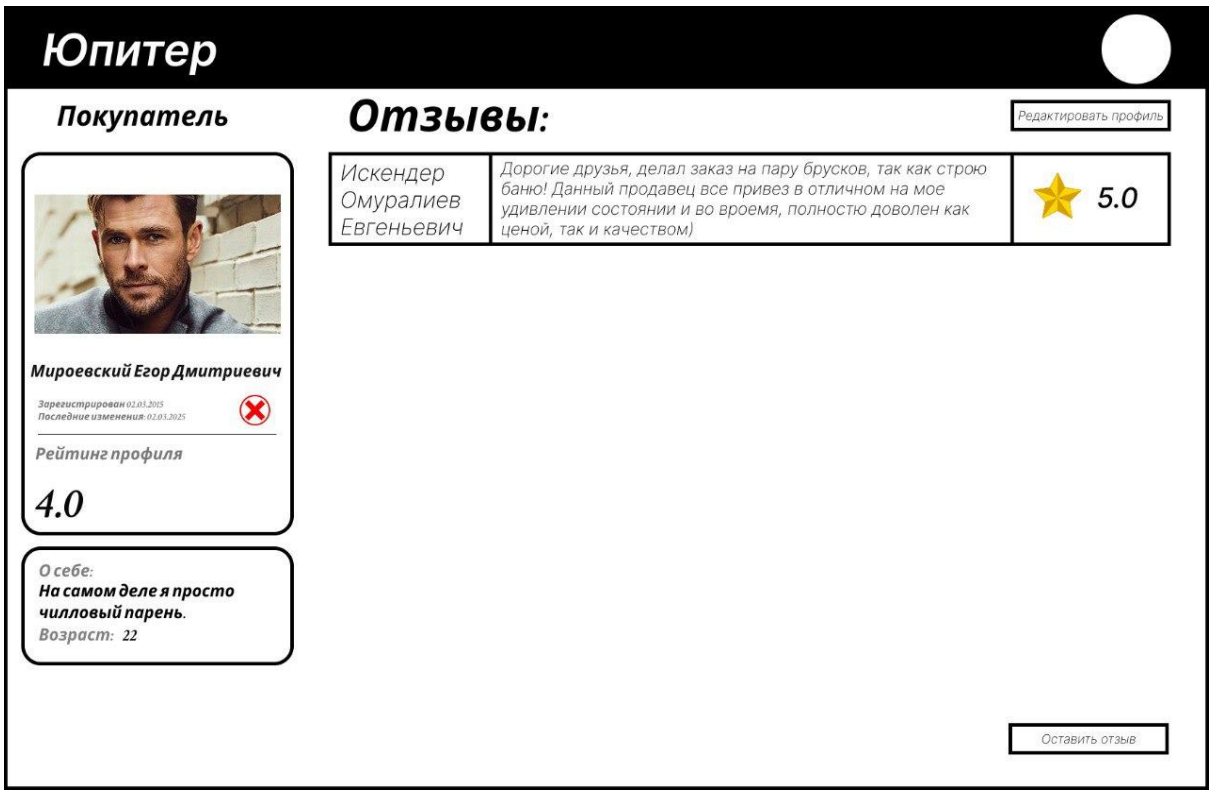

Рисунок 1.5 – Страница покупателя с полной информацией о нём и отзывами других пользователей

Рисунок 1.6 – Страница продавца с полной информацией о нём и отзывами других пользователей

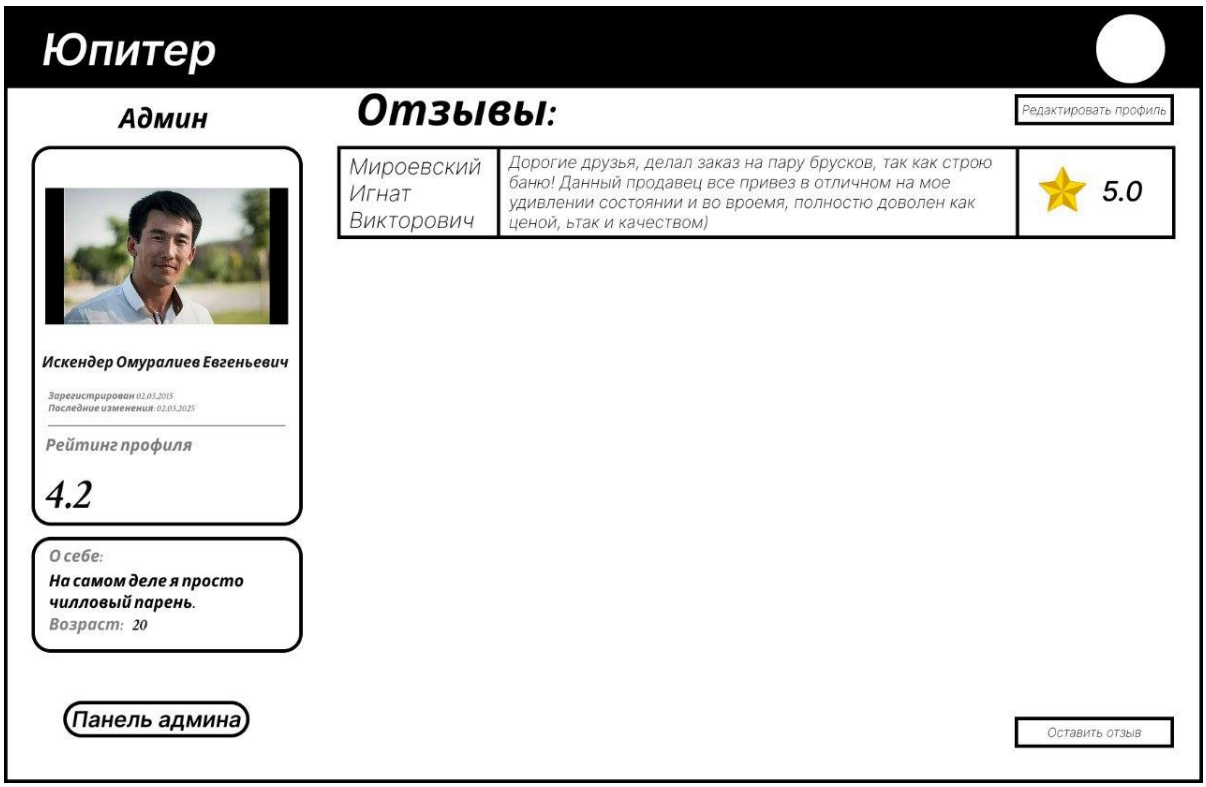

Рисунок 1.7 – Страница администратора с полной информацией о нём и отзывами других пользователей

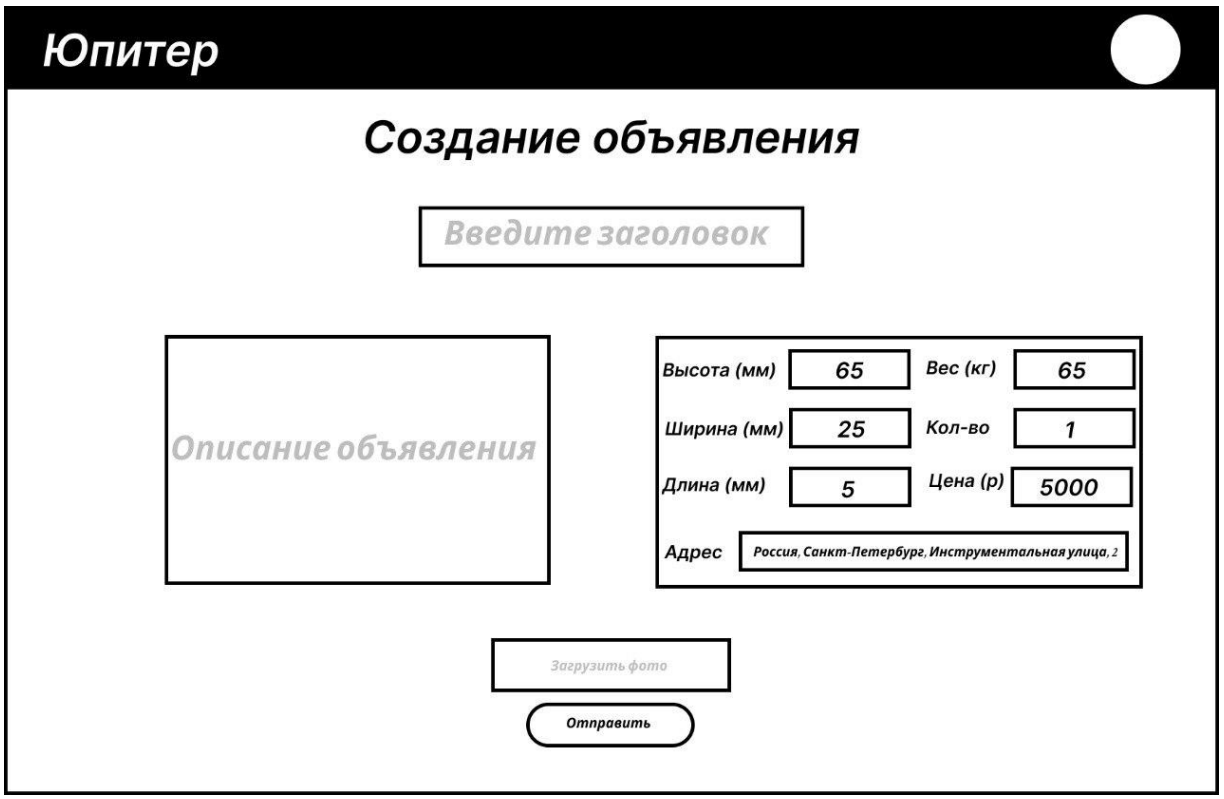

Рисунок 1.8 – Страница создания объявления

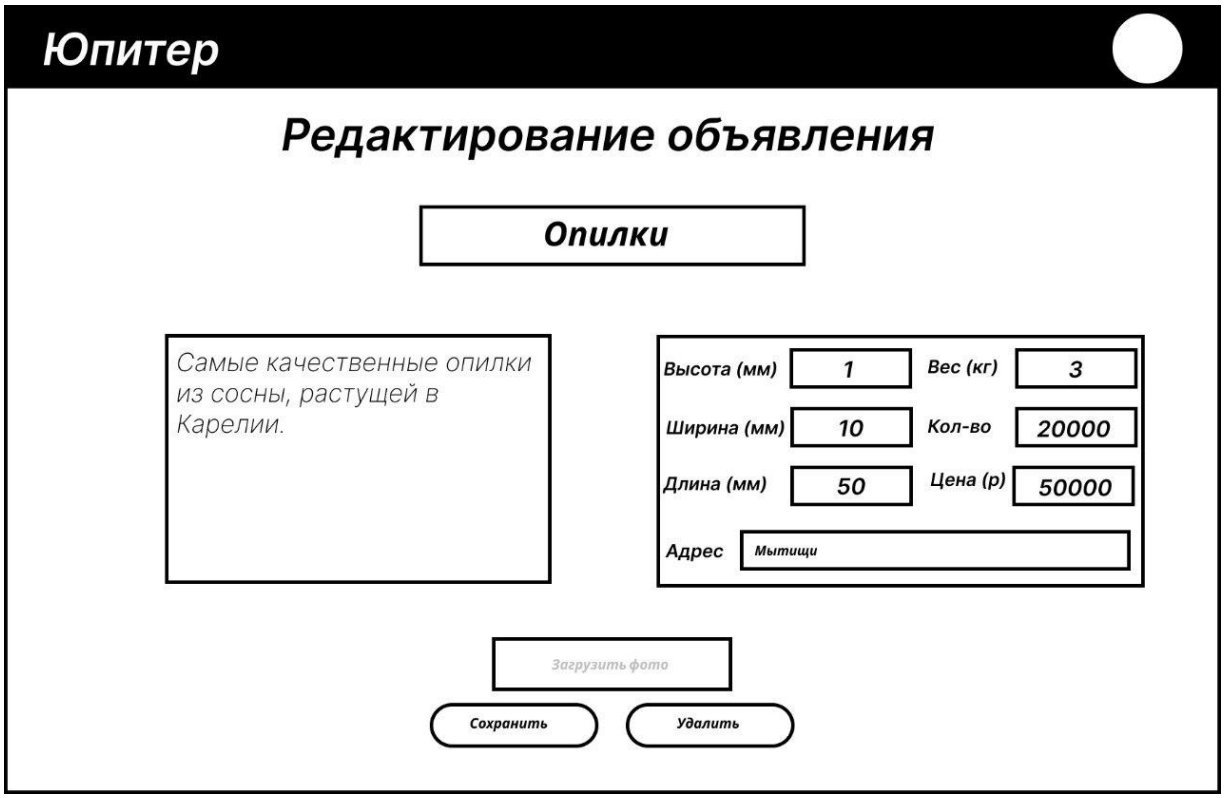

Рисунок 1.9 – Страница редактирования объявления

Рисунок 1.10 – Страница панели администратора сайта

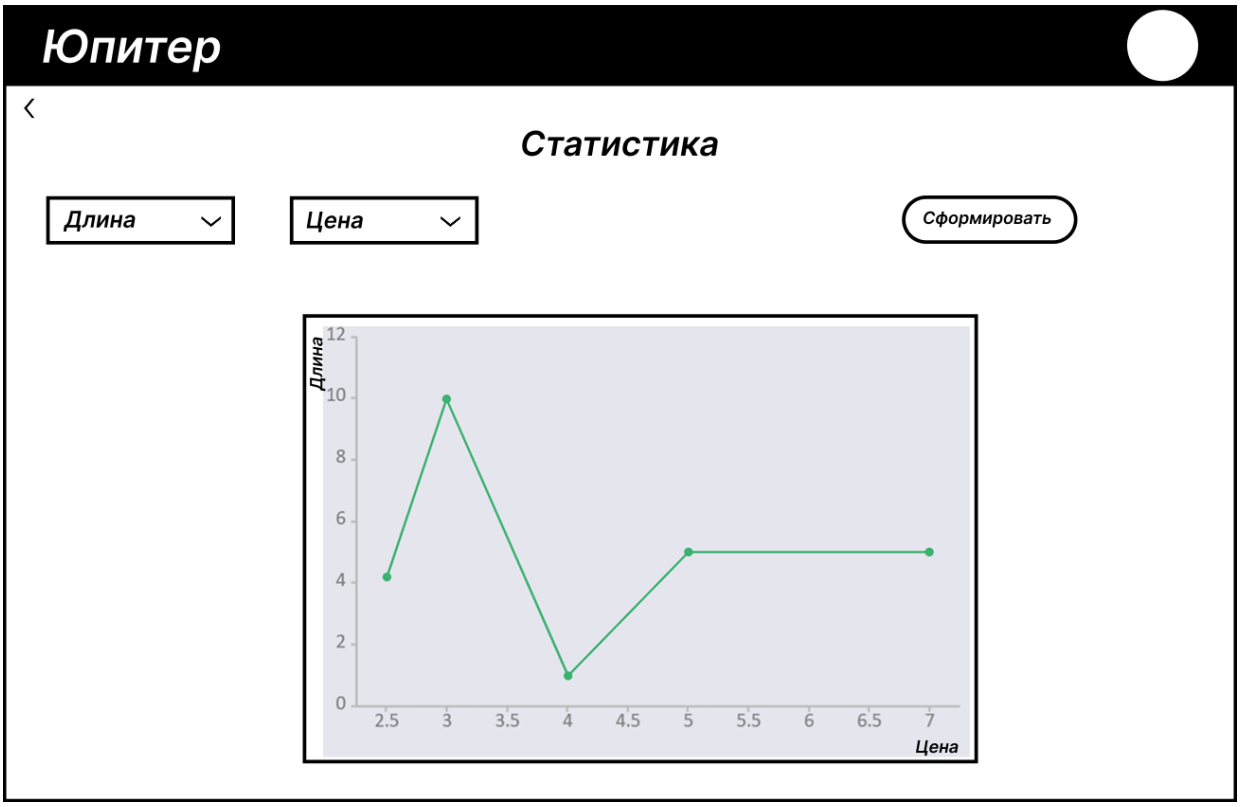

Рисунок 1.11 – Страница статистики для администратора сайта

1.2. Сценарии использования

В ходе подготовки к разработке приложения были также разработаны сценарии его использования, которые соответствуют интерфейсу приложения. В них учтён каждый тип пользователя.

Сценарий использования - "Покупатель"

Действующее лицо: Пользователь.

Предусловие: Пользователь авторизован на сайте как покупатель.

Основной сценарий:

- Пользователь заходит на основную страницу сайта со всеми объявлениями от столяров, а также видит количество объявлений.
- Пользователь просматривает предложения мастеров с фотографиями остатков, параметрами, ценами.
- Пользователь выбирает заинтересовавшее его объявление и, при нажатии на него, переходит на его страницу.
- На странице объявления пользователь читает описание объявления и комментарии, которые оставляют к данному объявлению.
- Пользователь переходит со страницы объявления на страницу мастера.
- На странице мастера пользователь видит информацию о столяре и отзывы о нём, а также статистику, которая представляет из себя среднюю оценку пользователя.
- Пользователь оставляет отзыв о столяре.
- Пользователь переходит ко всем объявлениям данного мастера.
- При нажатии на название сайта в левом верхнем углу пользователь переходит на основную страницу сайта.

Альтернативные сценарии:

- Если пользователь не авторизован, ему будет предложено авторизоваться или зарегистрироваться.

- Пользователь не может найти нужные товары и использует фильтрацию на основной странице или странице объявлений мастера для более удобного поиска.
- Пользователь заходит на главную страницу и сортирует объявления.
- Пользователь переходит на страницу объявления и оставляет комментарий.
- Пользователь заходит на страницу мастера или покупателя и оставляет отзыв.
- Пользователь при помощи иконки в правом верхнем углу заходит на свою страницу и видит информацию и отзывы от себе.
- Пользователь на своей странице редактирует информацию о себе.
- Пользователь удаляет свой оставленный комментарий или отзыв.

Сценарий использования - "Мастер"

Действующее лицо: Пользователь.

Предусловие: Пользователь авторизован на сайте как мастер.

Основной сценарий:

- Пользователь заходит в свой профиль при помощи иконки в правом верхнем углу.
- Пользователь на своей странице видит информацию и отзывы о себе.
- Пользователь нажимает кнопку создания объявления и переходит на соответствующую страницу.
- Пользователь заполняет информацию об остатках (название, параметры, цена, адрес, описание фото).
- Мастер публикует объявление и возвращается на страницу своего профиля.

Альтернативные сценарии:

- Если пользователь не авторизован, ему будет предложено авторизоваться или зарегистрироваться.

- Пользователь переходит на страницу объявления и оставляет комментарий.
- Пользователь переходит к своим объявлениям, заходит на своё объявление редактирует или удаляет его.
- Пользователь на своей странице редактирует информацию о себе.
- Пользователь заходит на страницу мастера или покупателя и оставляет отзыв.
- Пользователь удаляет оставленный комментарий или отзыв.

Сценарий использования - "Администратор"

Действующее лицо: Пользователь.

Предусловие: Пользователь авторизован на сайте как администратор.

Основной сценарий:

- Пользователь заходит в свой профиль при помощи иконки в правом верхнем углу.
- Пользователь на своей странице видит информацию и отзывы о себе.
- Пользователь переходит на страницу администрирования.
- Пользователь может редактировать любой профиль.
- Пользователь по нажатию на определённую кнопку выполняет экспорт хранимых в программе данных в машиночитаемом формате.
- Пользователь загружает файл данных в машиночитаемом формате и по нажатию на кнопку выполняет их импорт, заменяя все данные.
- Пользователь заходит в пункт управления для просмотра статистики.
- Пользователь выбирает два параметра и создает график зависимости одного параметра от другого.

Альтернативные сценарии:

- Если пользователь не авторизован, ему будет предложено авторизоваться или зарегистрироваться.
- Пользователь редактирует информацию своей страницы.

- Пользователь переходит на страницу мастера или покупателя и редактирует информацию о нём или оставляет отзыв.
- При переходе на страницу мастера или покупателя пользователь видит статистику о данном пользователе, которая представляет из себя среднюю оценку.
- При переходе на главную страницу со всеми объявлениями, а также при фильтрации, или на страницу со всеми объявлениями мастера пользователь видит статистику, выражающуюся в количестве объявлений.
- Пользователь переходит на страницу объявления и редактирует его информацию или оставляет комментарий.
- Пользователь заходит на страницу любого пользователя и блокирует его.
- Пользователь заходит в профиль любого пользователя или на страницу любого объявления и удаляет комментарий или отзыв.

Сценарий использования - "Регистрация и авторизация"

Действующее лицо: Пользователь.

Предусловие: Пользователь не авторизован.

Основной сценарий:

- Пользователь попадает на страницу авторизации.
- Пользователь, указывая логин и пароль авторизуется.
- Пользователь попадает на основную страницу сайта.
- Если пользователь не зарегистрирован, то он переходит на страницу регистрации.
- Пользователь заполняет краткую информацию о себе и указывает тип пользователя (мастер или покупатель).
- Пользователь регистрируется на сайте и попадает на страницу авторизации.

1.3. Вывод о преобладающем типе операций

В проекте будут преобладать операции чтения, так как почти для каждой страницы приходится делать запрос на чтение из БД, а также данные запросы будут отправляться при каждой фильтрации. Операций записи будет меньше по той причине, что комментарии/отзывы оставляются не на всех страницах, наряду с тем, что не все пользователи любят тратить время на то, чтобы эти отзывы или комментарии оставлять.

2. МОДЕЛЬ ДАННЫХ

2.1. Нереляционная модель

Графическое представление нереляционной модели см. на рис. 2.1.

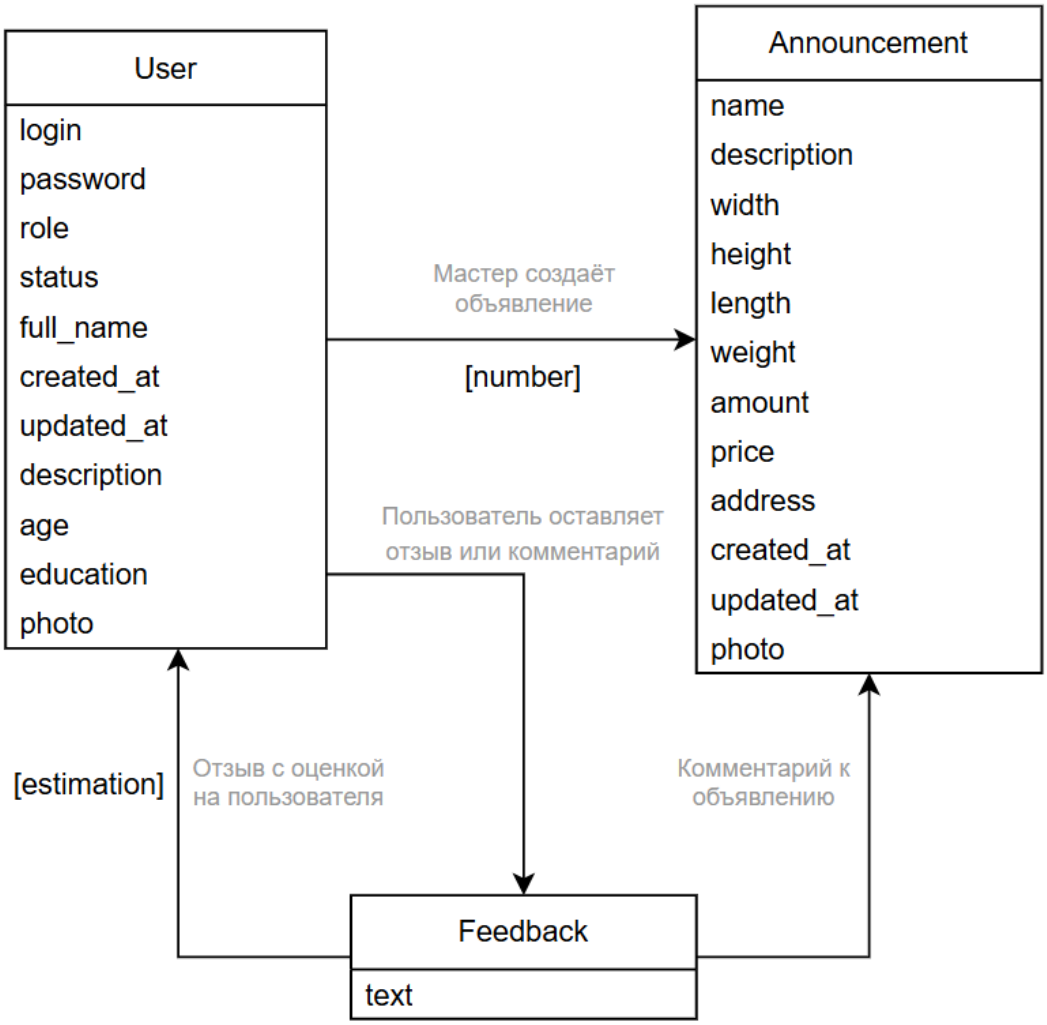

Рисунок 2.1 – Графическое представление нереляционной модели

В данной модели присутствуют следующие сущности с полями и связями:

User – пользователь (администратор, покупатель, столяр):

Название	Тип данных	Описание
login	STRING	Логин пользователя (уникален для каждого пользователя)
password	STRING	Пароль пользователя

role	STRING	Роль: администратор, покупатель или мастер
status	STRING	Статус: активен, заблокирован
full_name	STRING	ФИО пользователя
created_at	DATETIME	Дата и время регистрации аккаунта
updated_at	DATETIME	Дата и время обновления аккаунта
description	STRING	Описание пользователя
age	INTEGER	Возраст пользователя
education	STRING	Образование пользователя
photo	STRING	Ссылка на фотографию

Announcement – объявление, созданное мастером:

Название	Тип данных	Описание
name	STRING	Заголовок объявления
description	STRING	Описание объявления
width	FLOAT	Ширина товара
height	FLOAT	Высота товара
length	FLOAT	Длина товара
weight	FLOAT	Вес товара
amount	INTEGER	Количество товара
price	FLOAT	Цена за весь товар
address	STRING	Адрес для покупки товара
created_at	DATETIME	Дата и время регистрации объявления
updated_at	DATETIME	Дата и время обновления объявления
photo	STRING	Ссылка на фотографию

Feedback – отзыв или комментарий, оставленный пользователем:

Название	Тип данных	Описание
text	STRING	Текст отзыва/комментария

Связи модели со свойствами:

Родительский узел	Связь	Дочерний узел	Описание
User	-[number]->	Announcement	Объявление с определённым номером принадлежит конкретному пользователю
User	-->	Feedback	Комментарий оставлен конкретным пользователем
Feedback	-->	Announcement	Комментарий к конкретному объявлению
Feedback	-[estimation]->	User	Отзыв с оценкой о конкретном пользователе

Свойства number и estimation имеют тип данных INTEGER.

Оценка объёма информации, хранимой в модели, и избыточности:

Пусть оценка объёма информации будет зависеть от количества объявлений, так как основная задача сервиса - позволить мастерам создавать объявления с остатками и браком, чтобы покупатель смог их приобрести.

- N - количество объявлений.
- A - средний объём данных для одного объявления = 15 · 1b (name) + 300 · 1b (description) + 8b (width) + 8b (height) + 8b (length) + 8b (weight) + 8b (amount) + 8b (price) + 100 · 1b (address) + 16b (created_at) + 16b (updated_at) + 100 · 1b (photo) + 8b (number)
- N // 4 - среднее количество мастеров (пусть будет по 4 объявления на столяра)
- N // 2 - среднее количество покупателей (пусть будет по 2 объявления на покупателя)
- U - средний объём данных для одного пользователя = 10 · 1b (login) + 10 · 1b (password) + 15 · 1b (role) + 15 · 1b (status) + 50 · 1b (full_name) + 16b

(created_at) + 16b (updated_at) + 200 · 1b (description) + 8b (age) + 100 · 1b (education) + 100 · 1b (photo)

- M - среднее количество комментариев на объявлении и отзывов на пользователе. Примем M = 3
- F - средний объём данных для одного комментария = 300 · 1b (text)
- F_e - средний объём данных для одного отзыва = 300 · 1b (text) + 8b (estimation)

Следовательно, общий объём хранимой информации можно оценить по формуле:

B(N) = N · A + U · (N // 4 + N // 2) + M · N · F + M · (N // 4 + N // 2) · F_e = N · 603b + 540b · (3 · N // 4) + N · 900b + 924b · (3 · N // 4) = N · (603b + 405b + 900b + 693b) = 2601b · N

В нереляционной базе данных, помимо самих данных, сохраняется дополнительная служебная информация (метаданные), также память в графовой БД занимают связи. За чистый объём данных возьмём величину, рассчитанную в предыдущем пункте: B_clean(N) = 2601b · N. Фактический объём рассчитаем приближенно с учетом затрат памяти на узлы (32b) и связи (32b):

- Ann(N) - затраты на хранение узлов объявлений, связей комментариев к ним, связей от пользователя к объявлению = 32b · N + 32b · N · M + 32b · N
- Usr(N) - затраты на хранение узлов пользователей, связей комментариев к ним = 32b · (3 · N // 4) + 32b · M · (3 · N // 4)
- Fdb(N) - затраты на хранение узлов комментариев, связей пользователей к ним = 32b · (M · N + M · (3 · N // 4)) · 2
- B_actual(N) = B_clean(N) + Ann(N) + Usr(N) + Fdb(N) = 2601b · N + 160b · N + 96b + 336b · N = 3293b · N

  Таким образом, фактор избыточности выражается как:

Fct(N) = B_actual(N) / B_clean(N) = (3293b · N) / (2601b · N) = 1.27

Примеры запросов к модели:

Сценарий использования - "Покупатель"

Сценарий: Получить все объявления и их количество

Реализация:

```
MATCH (a:Announcement)
RETURN a.name, a.description, a.width, a.height, a.length, a.weight,
a.amount, a.price, a.address, a.created_at, a.updated_at, a.photo,
count(a) OVER () LIMIT 4
```

Итого:

Количество запросов: 1

Задействованные коллекции: Announcement

Сценарий: Выбор объявления

Реализация:

```
MATCH (u:User)-[r:CREATED_BY {number: 2}]->(a:Announcement)
RETURN a.name, a.description, a.width, a.height, a.length, a.weight,
a.amount, a.price, a.address, a.created_at, a.updated_at, a.photo
```

Итого:

Количество запросов: 1

Задействованные коллекции: User, Announcement

Сценарий: Просмотр описания и комментариев к объявлению

Реализация:

```
MATCH (a:Announcement {name: "Доски для мебели"})
OPTIONAL MATCH (f:Feedback)-[:COMMENT_ON]->(a)
RETURN a.description, collect(f.text) AS comments
```

Итого:

Количество запросов: 2

Задействованные коллекции: Announcement, Feedback

Сценарий: Переход на страницу мастера

Реализация:

```
MATCH (u:User)-[:CREATED_BY {number: 5}]->(a:Announcement)
```

```
OPTIONAL MATCH (f:Feedback)-[r:RATED_USER]->(u)
RETURN u.full_name, u.description, u.age, u.education, u.created_at,
u.updated_at, u.photo, avg(r.estimation)
```

Итого:

Количество запросов: 2

Задействованные коллекции: User, Announcement

Сценарий: Отзыв о мастере

Реализация:

```
MATCH (u1:User {login: "user123"}), (u2:User {full_name: "Петров Иван
Иванович"})
CREATE (f:Feedback {text: "Отличный мастер!"})
CREATE (f)-[:RATED_USER {estimation: 5}]->(u2)
RETURN f
```

Итого:

Количество запросов: 3

Задействованные коллекции: User, Feedback

Сценарий: Переход к объявлениям мастера

Реализация:

```
MATCH (u:User {login: "ivan_petrov"})-[r:CREATED_BY]->(a:Announcement)
RETURN r.number AS announcement_number, a.name, a.description, a.width,
a.height, a.length, a.weight, a.amount, a.price, a.photo
ORDER BY a.created_at DESC
```

Итого:

Количество запросов: 2

Задействованные коллекции: User, Announcement

Сценарий использования - "Мастер"

Сценарий: Переход на страницу своего профиля

Реализация:

```
MATCH (u:User {login: "ivan_petrov"})
OPTIONAL MATCH (f:Feedback)-[r:RATED_USER]->(u)
RETURN u.full_name, u.description, u.age, u.education, u.created_at,
u.updated_at, u.photo, avg(r.estimation)
```

Итого:

Количество запросов: 2

Задействованные коллекции: User, Feedback

Сценарий: Создание объявления

Реализация:

```
MATCH (u:User {login: "ivan_petrov"})
OPTIONAL MATCH (u)-[r:CREATED_BY]->(:Announcement)
WITH u, max(r.number) AS max_number
CREATE (a:Announcement {
    name: "Доски для мебели",
    description: "Качественные доски для мебели",
    width: 15,
    height: 3,
    length: 200,
    weight: 10,
    amount: 50,
    price: 700,
    address: "Москва, ул. Лесная, 10",
    created_at: datetime(),
    updated_at: datetime(),
    photo: "photo_url"
})
CREATE (u)-[:CREATED_BY {number: coalesce(max_number + 1, 1)}]->(a)
RETURN a
```

Итого:

Количество запросов: 4

Задействованные коллекции: User, Announcement

Сценарий использования - "Администратор"

Сценарий: Редакция любого профиля

Реализация:

```
MATCH (u:User {login: "ivan_petrov"})
SET u.full_name = "Иван Сергеевич Петров",
    u.description = "Опытный столяр с 10-летним стажем",
    u.age = 35,
    u.updated_at = datetime()
RETURN u
```

Итого:

Количество запросов: 2

Задействованные коллекции: User

Сценарий: Экспорт данных

Реализация:

```
CALL apoc.export.json.all("all_data.json", {useTypes: true})
```

Итого:

Количество запросов: 1

Сценарий: Импорт данных с заменой

Реализация:

```
MATCH (n) DETACH DELETE n
CALL apoc.import.json("all_data.json")
```

Итого:

Количество запросов: 3

Сценарий использования - "Регистрация и авторизация"

Сценарий: Авторизация

Реализация:

```
MATCH (u:User {login: "ivan_petrov", password: "password123"})
RETURN u
```

Итого:

Количество запросов: 1

Задействованные коллекции: User

Сценарий: Регистрация

Реализация:

```
CREATE (u:User {
    login: "pbs",
    password: "Newpassword123",
    role: "master",
    status: "active",
    full_name: "Бобров Павел Сергеевич",
    description: "Простой человек",
    created_at: datetime(),
    updated_at: datetime(),
    age: 30,
    education: "СПбГЭТУ «ЛЭТИ» ",
    photo: "some_path/photo.jpg"
})
RETURN u
```

Итого:

Количество запросов: 1

Задействованные коллекции: User

2.2. Реляционная модель

Графическое представление реляционной модели см. на рис. 2.2.

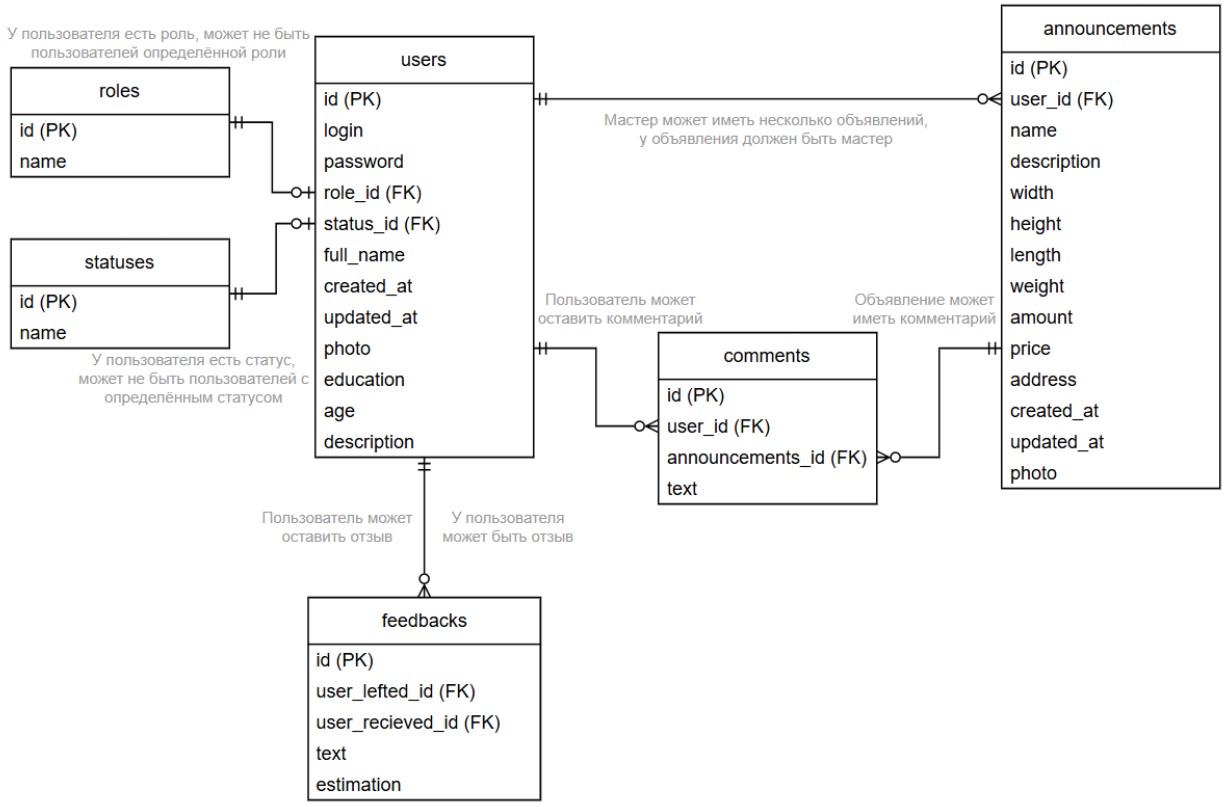

Рисунок 2.2 – Графическое представление реляционной модели

В данной модели присутствуют следующие сущности с полями и связями:

Таблица roles – Хранит все возможные роли пользователей.

Название	Тип данных	Описание
id	INTEGER PK	Уникальный идентификатор роли
name	VARCHAR(15)	Название роли (например, «мастер», «покупатель», «админ»)

Таблица statuses – Хранит все возможные статусы пользователей.

Название	Тип данных	Описание
id	INTEGER PK	Уникальный идентификатор статуса
name	VARCHAR(15)	Название статуса (например, «активен»,

		«заблокирован»)

Таблица users – Содержит информацию о всех пользователях.

Название	Тип данных	Описание
id	INTEGER PK	Уникальный идентификатор пользователя
login	VARCHAR(10)	Логин пользователя
password	VARCHAR(10)	Пароль пользователя
role_id	INTEGER FK → roles.id	Внешний ключ на таблицу ролей
status_id	INTEGER FK → statuses.id	Внешний ключ на таблицу статусов
full_name	VARCHAR(50)	ФИО пользователя
created_at	TIMESTAMP	Дата и время регистрации
updated_at	TIMESTAMP	Дата и время последнего обновления
description	TEXT	Описание пользователя
age	INTEGER	Возраст пользователя
education	VARCHAR(100)	Образование
photo	VARCHAR(100)	URL или путь к фотографии

Таблица announcements – Объявления, создаваемые мастерами.

Название	Тип данных	Описание
id	INTEGER PK	Уникальный идентификатор объявления
user_id	INTEGER FK → users.id	Внешний ключ на автора (мастера)
name	VARCHAR(50)	Заголовок объявления
description	TEXT	Подробное описание
width	NUMERIC(10,2)	Ширина товара
height	NUMERIC(10,2)	Высота товара
length	NUMERIC(10,2)	Длина товара

weight	NUMERIC(10,2)	Вес товара
amount	INTEGER	Количество единиц
price	NUMERIC(12,2)	Цена за весь товар
address	VARCHAR(100)	Адрес для встречи/покупки
created_at	TIMESTAMP	Дата и время создания
updated_at	TIMESTAMP	Дата и время последнего обновления
photo	VARCHAR(100)	URL или путь к фото

Таблица comments – Комментарии пользователей к объявлениям.

Название	Тип данных	Описание
id	INTEGER PK	Уникальный идентификатор комментария
user_id	INTEGER FK → users.id	Внешний ключ на автора комментария
announcement_id	INTEGER FK → announcements.id	Внешний ключ на объявление, к которому оставлен комментарий
text	TEXT	Текст комментария

Таблица feedbacks – Отзывы (рейтинг и текст) между пользователями.

Название	Тип данных	Описание
id	INTEGER PK	Уникальный идентификатор отзыва
user_left_id	INTEGER FK → users.id	Внешний ключ на пользователя, который оставил отзыв
user_received_id	INTEGER FK → users.id	Внешний ключ на пользователя, которому адресован отзыв
text	TEXT	Текст отзыва
estimation	SMALLINT	Оценка (например, от 1 до 5)

Оценка объёма информации, хранимой в модели, и избыточности:

Пусть оценка объёма информации будет зависеть от количества объявлений.

- N - количество объявлений.
- A - средний объём данных для одного объявления = 50 · 1b (name) + 300 · 1b (description) + 12b (width) + 12b (height) + 12b (length) + 12b (weight) + 4b (amount) + 14b (price) + 100 · 1b (address) + 4b (created_at) + 4b (updated_at) + 100 · 1b (photo)
- N // 4 - среднее количество мастеров (пусть будет по 4 объявления на столяра)
- N // 2 - среднее количество покупателей (пусть будет по 2 объявления на покупателя)
- U - средний объём данных для одного пользователя = 10 · 1b (login) + 10 · 1b (password) + 50 · 1b (full_name) + 4b (created_at) + 4b (updated_at) + 200 · 1b (description) + 4b (age) + 100 · 1b (education) + 100 · 1b (photo)
- ROLE - хранение ролей (пусть будет фиксированные 3 роли) = 3 · 15 · 1b (name) = 45b
- STATUS - хранение статусов (пусть будет фиксированные 2 статуса) = 2 · 15 · 1b (name) = 30b
- M - среднее количество комментариев на объявлении и отзывов на пользователе. Примем M = 3
- F - средний объём данных для одного комментария = 300 · 1b (text)
- F_e - средний объём данных для одного отзыва = 300 · 1b (text) + 2b (estimation)

B(N) = N · A + U · (N // 4 + N // 2) + M · N · F + M · (N // 4 + N // 2) · F_e + ROLE + STATUS = N · 624b + 482b · (3 · N // 4) + 3 · 300b · N + 3 · (3 · N // 4) · 302b + 45b + 30b = N · (624b + 361b + 900b + 678b) + 95b = 2563b · N + 75b

Можно опустить константы в виде ROLE и STATUS и получить в итоге: B(N) = 2563b · N

В реляционной базе данных, помимо самих данных, хранится дополнительная служебная информация (метаданные). За чистый объём данных возьмём величину, рассчитанную в предыдущем пункте: B_clean(N) = 2563b · N. За метаданные примем O' = 200b. Также для работы базы данных требуются id, рассчитаем новые оценки объёма данных для различных коллекций: A' = A + 4b (id) + 4b (user_id) U' = U + 4b (id) + 4b (role_id) + 4b (status_id) F' = F + 4b (id) + 4b (user_id) + 4b (announcement_id) F_e' = F_e + 4b (id) + 4b (user_left_id) + 4b (user_received_id)

Таким образом, B_actual можно рассчитать по следующей формуле:

B_actual(N) = N' · A + U' · (N // 4 + N // 2) + M · N · F' + M · (N // 4 + N // 2) · F_e' + O' · N = N · 632b + 494b · (3 · N // 4) + 3 · 312b · N + 3 · (3 · N // 4) · 314b + 200b · N = 2866b · N

В результате, фактор избыточности выражается как:

Fct(N) = B_actual(N) / B_clean(N) = (2866b · N) / (2563b · N) = 1.12

Примеры запросов к модели:

Сценарий использования — «Покупатель»

Получить все объявления и их количество

Реализация:

```
SELECT
    a.name, a.description, a.width, a.height, a.length,
    a.weight, a.amount, a.price, a.address,
    a.created_at, a.updated_at, a.photo,
    COUNT(*) OVER () AS total
FROM announcements a
LIMIT 4;
```

Итого:

Количество запросов: 1

Задействованные коллекции: announcements

Выбор объявления по номеру пользователя

Реализация:

```
SELECT
    a.name, a.description, a.width, a.height, a.length,
```

```
    a.weight, a.amount, a.price, a.address,
    a.created_at, a.updated_at, a.photo
FROM announcements a
JOIN users u ON a.user_id = u.id
WHERE u.id = 2;
```

Итого:

Количество запросов: 2

Задействованные коллекции: announcements, users

Просмотр описания и комментариев к объявлению

Реализация:

```
SELECT
    a.description,
    ARRAY_AGG(c.text) AS comments
FROM announcements a
LEFT JOIN comments c ON a.id = c.announcements_id
WHERE a.name = 'Доски для мебели'
GROUP BY a.id, a.description;
```

Итого:

Количество запросов: 3

Задействованные коллекции: announcements, comments

Переход на страницу мастера

Реализация:

```
SELECT
    u.full_name, u.description, u.age, u.education,
    u.created_at, u.updated_at, u.photo,
    AVG(f.estimation) AS avg_rating
FROM announcements a
JOIN users u ON a.user_id = u.id
LEFT JOIN feedbacks f ON f.user_recieved_id = u.id
WHERE a.id = 5
GROUP BY u.id;
```

Итого:

Количество запросов: 4

Задействованные коллекции: announcements, users, feedbacks

Отзыв о мастере

Реализация:

```
INSERT INTO feedbacks (user_lefted_id, user_recieved_id, text,
estimation)
```

```
VALUES (
    (SELECT id FROM users WHERE login = 'user123'),
    (SELECT id FROM users WHERE full_name = 'Петров Иван Иванович'),
    'Отличный мастер!',
    5
);
```

Итого:

Количество запросов: 3

Задействованные коллекции: users, feedbacks

Переход к объявлениям мастера

Реализация:

```
SELECT
    a.id AS announcement_number,
    a.name, a.description, a.width, a.height, a.length,
    a.weight, a.amount, a.price, a.photo
FROM users u
JOIN announcements a ON a.user_id = u.id
WHERE u.login = 'ivan_petrov'
ORDER BY a.created_at DESC;
```

Итого:

Количество запросов: 3

Задействованные коллекции: announcements, users

Сценарий использования — «Мастер»

Переход на страницу своего профиля

Реализация:

```
SELECT
    u.full_name, u.description, u.age, u.education,
    u.created_at, u.updated_at, u.photo,
    AVG(f.estimation) AS avg_rating
FROM users u
LEFT JOIN feedbacks f ON f.user_recieved_id = u.id
WHERE u.login = 'ivan_petrov'
GROUP BY u.id;
```

Итого:

Количество запросов: 3

Задействованные коллекции: users

Создание объявления

Реализация:

```
INSERT INTO announcements (
    user_id, name, description, width, height, length,
    weight, amount, price, address, created_at, updated_at, photo
)
VALUES (
    (SELECT id FROM users WHERE login = 'ivan_petrov'),
    'Доски для мебели',
    'Качественные доски для мебели',
    15, 3, 200, 10, 50, 700,
    'Москва, ул. Лесная, 10',
    NOW(), NOW(), 'photo_url'
);
```

Итого:

Количество запросов: 3

Задействованные коллекции: users, announcements

Сценарий использования — «Администратор»

Редакция любого профиля

Реализация:

```
UPDATE users
SET full_name = 'Иван Сергеевич Петров',
    description = 'Опытный столяр с 10-летним стажем',
    age = 35,
    updated_at = NOW()
WHERE login = 'ivan_petrov';
```

Итого:

Количество запросов: 2

Задействованные коллекции: users

Экспорт всех данных

Реализация:

```
COPY (SELECT row_to_json(t) FROM (SELECT * FROM users) t) TO
'users.json';
```

Итого:

Количество запросов: 1

Задействованные коллекции: users

Импорт данных (с заменой)

Реализация:

```
TRUNCATE comments, feedbacks, announcements, users RESTART IDENTITY
CASCADE;
```

Итого:

Количество запросов: 1

Задействованные коллекции: comments, feedbacks, announcements, users

Сценарий — «Регистрация и авторизация»

Авторизация

Реализация:

```
SELECT *
FROM users
WHERE login = 'ivan_petrov'
  AND password = 'password123';
```

Итого:

Количество запросов: 1

Задействованные коллекции: users

Регистрация

Реализация:

```
INSERT INTO users (
    login, password, role_id, status_id, full_name,
    description, created_at, updated_at, age, education, photo
)
VALUES (
    'pbs', 'Newpassword123',
    (SELECT id FROM roles WHERE name = 'master'),
    (SELECT id FROM statuses WHERE name = 'active'),
    'Бобров Павел Сергеевич',
    'Простой человек',
    NOW(), NOW(), 30,
    'СПбГЭТУ «ЛЭТИ»',
    'some_path/photo.jpg'
);
```

Итого:

Количество запросов: 1

Задействованные коллекции: users

2.3. Сравнение моделей

Удельный объём информации

NoSQL: объём данных, вычисленный ранее, оценивается по формуле B(N) = 2601b · N. Однако в документе хранятся избыточные данные, что увеличивает коэффициент избыточности Fct(N) = 1.27.

SQL: объём хранимых данных оценивается по формуле B(N) = 2563b · N. Однако в документе также могут храниться избыточные данные (индексы, метаданные), в результате чего коэффициент избыточности Fct(N) = 1.12.

Запросы по отдельным юзкейсам

NoSQL: количество запросов для реализации меньше за счет более лаконичного вида самой базы данных, в результате чего она состоит из меньшего количества различных коллекций, отсутствует как таковая операция JOIN.

SQL: для большей части юзкейсов требуется 2 и более запросов, что связано с объединение различных коллекций для получение полных данных об объектах. В результате реляционные базы данных проигрывают по части количества запросов.

Вывод

NoSQL оптимален для проектов, требующих гибкой схемы данных, быстрого прототипирования и горизонтального масштабирования, что особенно полезно для работы с разнородной информацией. Хотя SQL обеспечивает нормализацию данных, уменьшая избыточность и поддерживая сложные запросы с высокой целостностью, его жесткая схема может ограничивать гибкость. В нашем случае, где данные динамично меняются, а отзывы и объявления играют ключевую роль, NoSQL предлагает больше удобства и адаптивности для разработки.

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

3.1. Краткое описание

Для взаимодействаия с базой данных были реализованы следующие модули:

Модуль db_main.py:

- class DatabaseConnection – основной класс, в котором прописано подключение к базе данных

Модуль db_backup.py:

- class DatabaseExporter – класс, который делает бэкап базы данных. Он содержит два основных метода: один сохраняет бэкап в словарь, а другой записывает в json-файл.
- class DatabaseImporter – класс, который записывает в базу данных бэкап. У него есть два основных метода: один записывает данные из словаря, а другой данные из json-файла.

Модуль db_manager.py:

- class DatabaseManager – класс, реализующий выполнения всех сценариев использования, взаимодействуя с базой данных.

  Также существует файл *utils.py,* в котором прописаны различные вспомогательные функции.

Модуль app.py – представляет из себя REST API сервер, написанный на Flask, который реализует все сценарии использования приложения: получает с фронтенда запросы на получение, запись, сохранение и удаление, выполняет задачи с использование ранее описанных модулей и возвращает ответ в зависимости от результата выполнения или корректности переданных данных.

Сам интерфейс приложения реализован с использованием Node.js на языке программирования TypeScript с использованием фреймворка React.

В результате, в приложении используется три сервиса:

- db – сервис Neo4j

- backend – REST API сервис, реализованный на Flask
- frontend – сервис фронтенда, реализованный с использование React

Пароль для базы данных задаётся через переменные окружения, прописанные напрямую в docker-compose.

Для запуска приложения необходимо склонировать репозиторий, перейти в папку проекта и развернуть приложение при помощи docker.

3.2. Использованные технологии

Нереляционная база данных: Neo4j

Backend: Python 3.11, Flask

Frontend: Node.js 18, TypeScript, React, сборщик Vite

3.3. Схема экранов приложения

Схему экранов приложения см. на рис. 3.1.

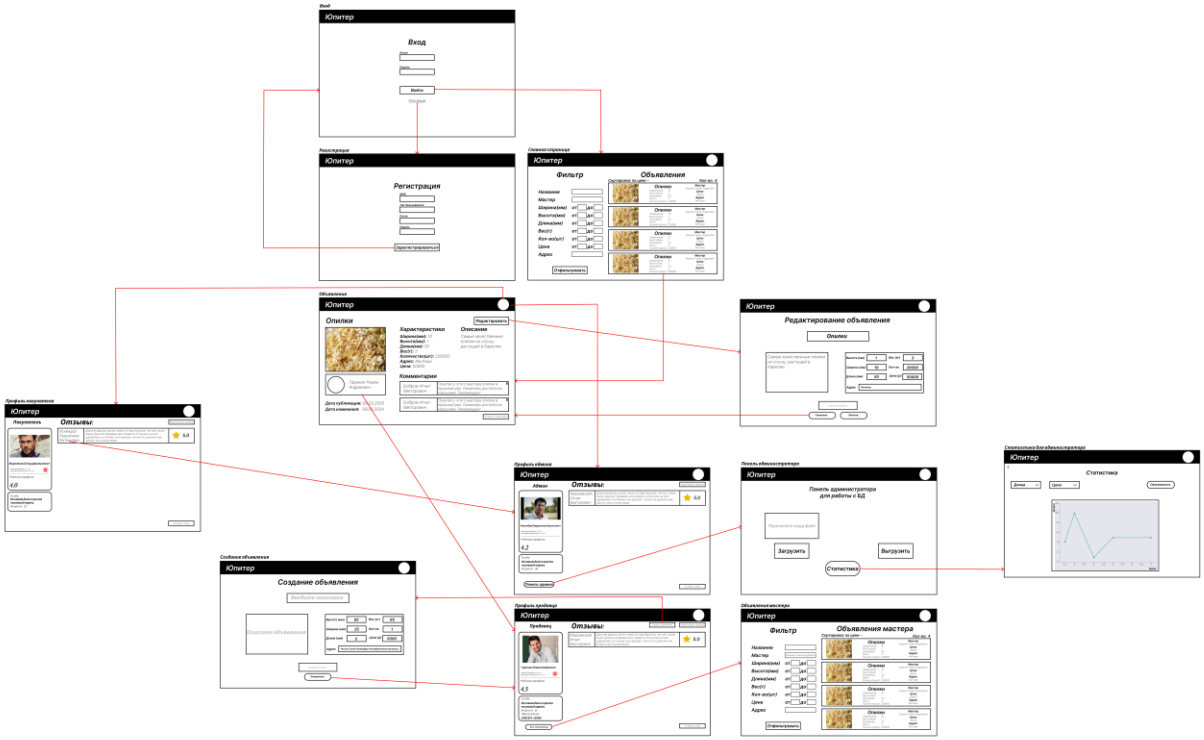

Рисунок 3.1 – Схема экранов приложения

4. ВЫВОДЫ

4.1. Достигнутые результаты

В ходе выполнения работы было разработано приложение-сервис по продаже остатков столярного производства, которое позволяет мастерам-столярам выкладывать объявления о продаже остатков производства, а покупателям просматривать объявления, все пользователи могут просматривать содержимое, оставлять комментарии на объявлении и отзывы о других пользователях, а администратор через свою панель способен скачивать и загружать бэкап базы данных, просматривать статистику, а также блокировать пользователей, редактировать профили и объявления.

4.2. Недостатки и пути для улучшения полученного решения

На данный момент не реализовано редактирование различных объектов и их удаление, а также просмотр всех пользователей администратором с фильтрацией.

4.3. Будущее развитие решения

Реализация не выполненных на данный момент функций, а также разработка приложений под различные операционные системы.

ЗАКЛЮЧЕНИЕ

Реализовано приложения с 3 видами пользователей (мастер, покупатель и администратор), у которых различные права. Приложение реализует множество сценариев использования, которые позволяют пользоваться им уже на данном этапе разработки. Все данные хранятся в графовой базе данных Neo4j. Также реализовано развёртывание приложения при помощи docker контейнера.

СПИСОК ЛИТЕРАТУРЫ

1. Ссылка на репозиторий с исходным кодом проекта: https://github.com/moevm/nosql1h25-joinery
2. Документация Neo4j // Neo4j.rb. URL: https://neo4jrb.readthedocs.io/en/stable/
3. Официальная документация Neo4j // Neo4j. URL: neo4j.com/docs

ПРИЛОЖЕНИЕ А
ИНСТРУКЦИЯ ПО РАЗВЁРТЫВАНИЮ

Запуск приложения:

```
git clone https://github.com/moevm/nosql1h25-joinery.git
cd nosql1h25-joinery
docker-compose up --build
```

Отладочные пользователи:

Администратор

- login - admin

- password - password

Мастер

- login - seller

- password - password

Покупатель

- login - buyer

- password - password