

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

КУРСОВАЯ РАБОТА
по дисциплине «Введение в нереляционные базы данных»
Тема: Реализация информационной системы оконного завода

Студент гр. 2303	_____	Попов Д.А.
Студентка гр. 2303	_____	Абрамова Д.Б.
Студент гр. 23004	_____	Пашков Г.М.
Студент гр. 2304	_____	Емельянчик А.А.
Студентка гр. 2304	_____	Кроткина З.Э.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург
2025

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Попов Д.А. 2303

Студентка Абрамова Д.Б. 2303

Студент Пашков Г.М. 2304

Студент Емельянчик А.А. 2304

Студентка Кроткина З.Э. 2304

Тема работы: Реализация информационной системы оконного завода

Исходные данные:

Создать информационную систему для работы оконного завода. Роли пользователей - замерщики, администраторы, покупатели. Замерщики выезжают на адрес, измеряют параметры окон и создают заказы. Администраторы наблюдают за процессом изготовления окон и управляют доставкой до клиента / коммуникацией с ним.

Используемая база данных – ArangoDB.

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: 01.05.2025

Дата защиты реферата: 01.05.2025

Студент гр. 2303	_____	Попов Д.А.
Студентка гр. 2303	_____	Абрамова Д.Б.
Студент гр. 23004	_____	Пашков Г.М.
Студент гр. 2304	_____	Емельянчик А.А.
Студентка гр. 2304	_____	Кроткина З.Э.
Преподаватель	_____	Заславский М.М

АННОТАЦИЯ

В рамках дисциплины «Введение в нереляционные базы данных» разработано приложение в команде. Выбрана тема: «Реализация информационной системы оконного завода», используя базу данных ArangoDB. Во внимание принимаются такие аспекты как производительность приложения, удобство пользования.

SUMMARY

Within the framework of the discipline "Introduction to non-relational Databases", a team developed an application. The chosen topic is "Implementation of the window factory information system" using the ArangoDB database. Aspects such as application performance and user-friendliness are taken into account.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
1. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ	8
1.1. Макет UI	8
1.2. Сценарий использования	15
2. МОДЕЛЬ ДАННЫХ	23
2.1. Нереляционная модель.....	23
2.2. Реляционная модель	34
2.3. Сравнение моделей.....	44
3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ	50
3.1. Краткое описание	50
3.2. Схема экранов приложения	51
3.3. Используемые технологии.....	51
4. ВЫВОДЫ	52
4.1. Достигнутые результаты	52
4.2. Недостатки и пути для улучшения полученного решения	52
4.3. Будущее развитие решения	52
5. ЗАКЛЮЧЕНИЕ	53

6. СПИСОК ЛИТЕРАТУРЫ	54
7. ПРИЛОЖЕНИЯ.....	55
5.1. Документация по сборке и развертыванию приложения	55

ВВЕДЕНИЕ

Цель работы – создать высокопроизводительное и удобное решение для информационной системы оконного завода.

Решено разработать веб-приложение, которое позволит покупателям делать заказы, замерщикам просматривать имеющиеся заказы и выезжать на замер, а администраторам следить за процессом создания окон, назначать замерщиков на заказы.


1. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

Прежде чем приступить к разработке, мы детально проанализировали, как пользователи будут взаимодействовать с сайтом. В этом разделе описаны ключевые сценарии использования — типичные пути, по которым пользователи смогут решать свои задачи.

Каждый сценарий построен на основе реальных потребностей целевой аудитории и учитывает логику навигации, UI-элементы и возможные действия пользователей. Это помогает нам создать интуитивно понятный интерфейс, где все функции работают согласованно и предсказуемо.

1.1.Макет UI

registration



РЕГИСТРАЦИЯ

Имя

Фамилия

Номер телефона

Дата Рождения

Пароль

ПОДТВЕРДИТЬ

Рисунок 1.1 – Экран регистрации нового пользователя

АВТОРИЗАЦИЯ

Номер телефона

Пароль

ПОДТВЕРДИТЬ

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 1.2 – Экран авторизации пользователя в систему

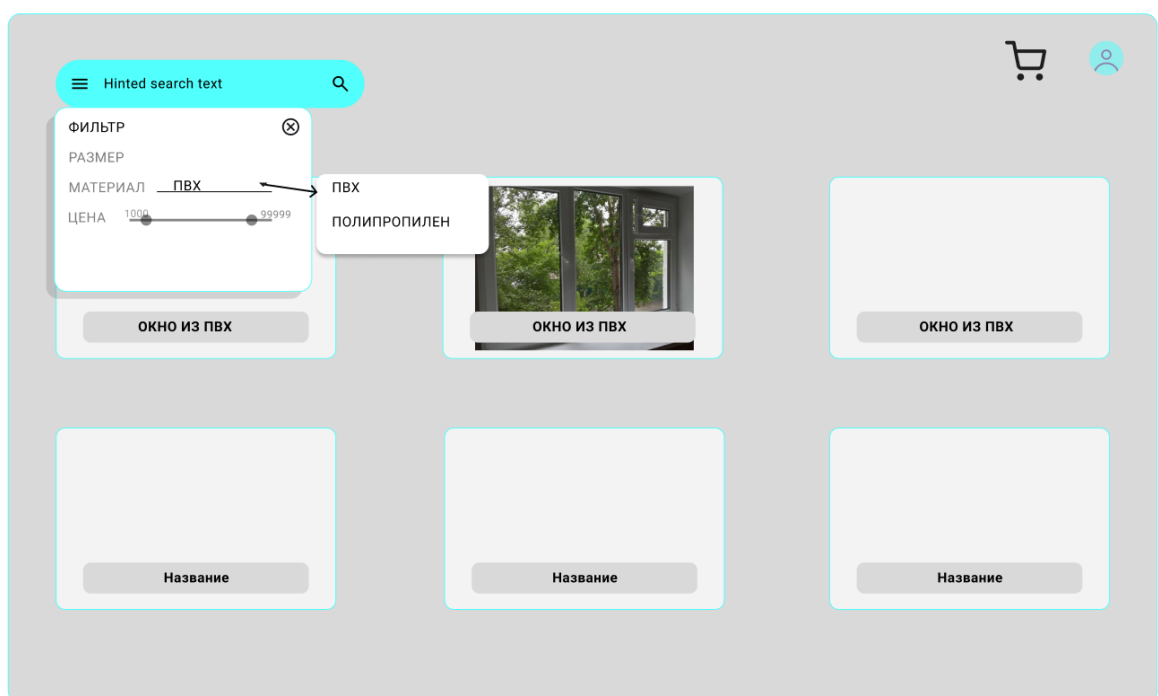


Рисунок 1.3 – Экран с каталогом и фильтрами

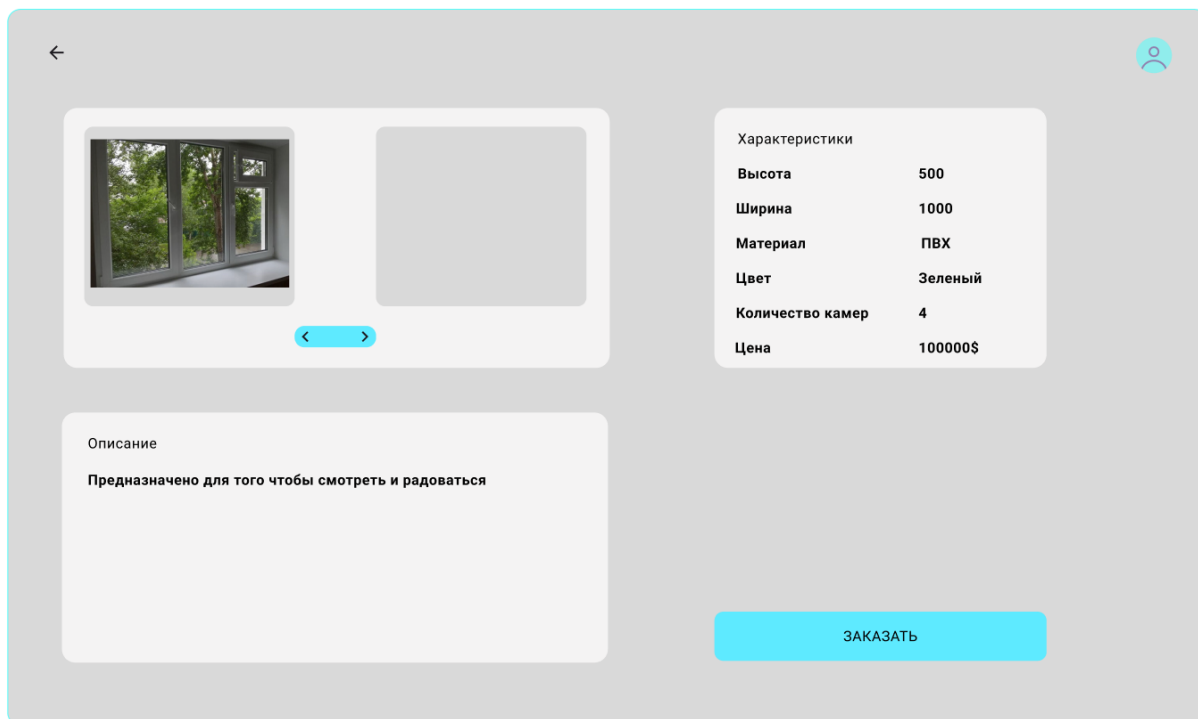


Рисунок 1.4 – Экран с карточкой продукта

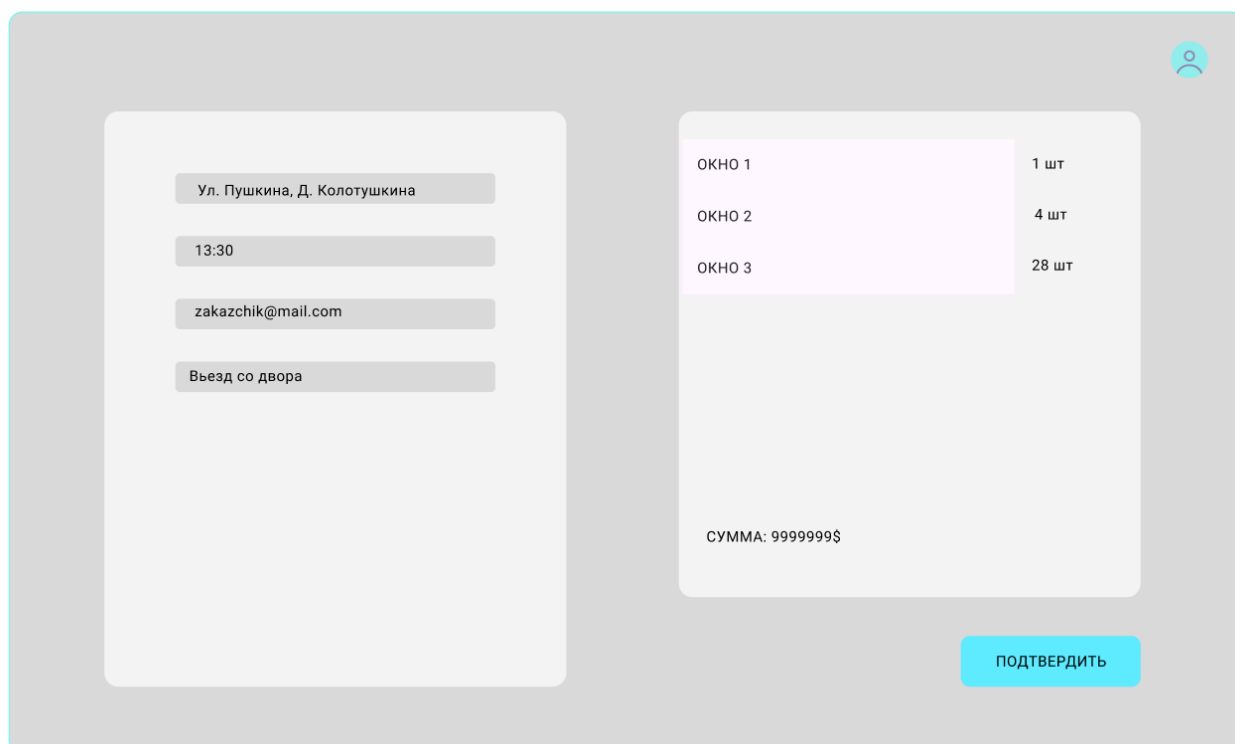


Рисунок 1.5 – Экран с заказом

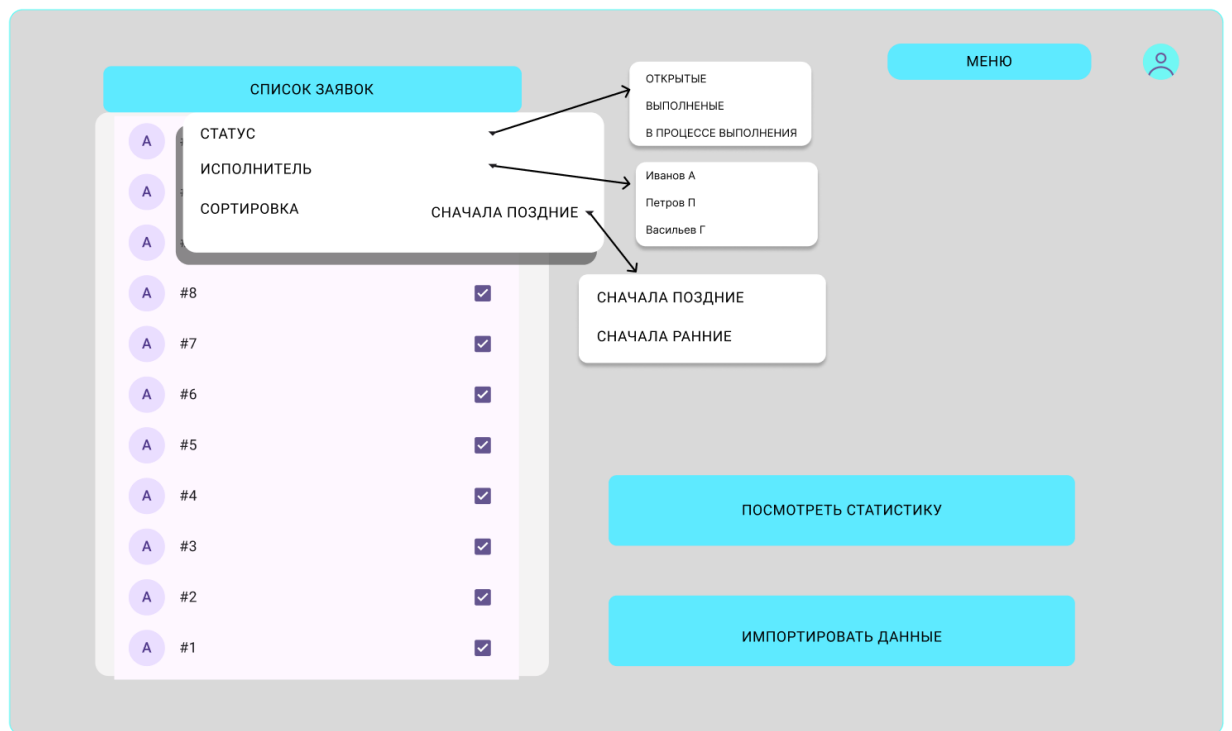


Рисунок 1.6 – Экран с Админ-панел



Рисунок 1.7 – Экран с просмотром имеющихся товаров, администратором

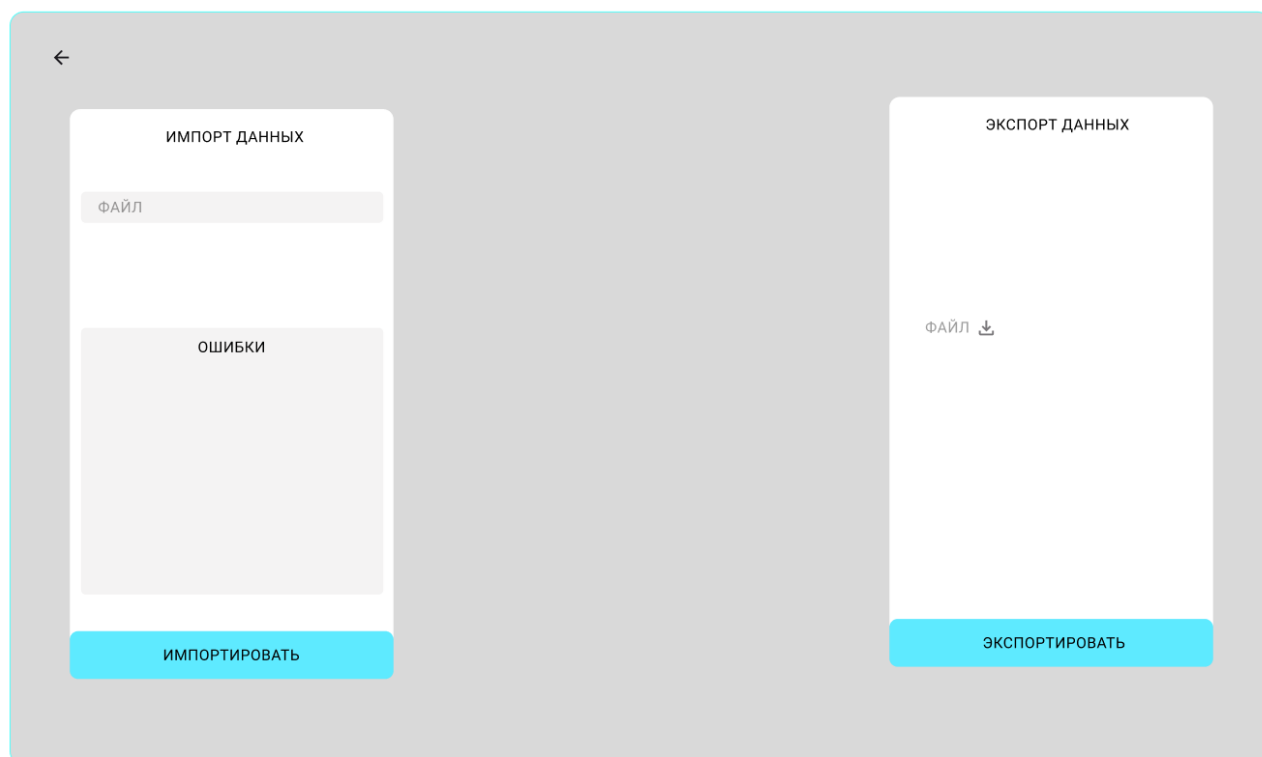


Рисунок 1.8 – Экран с Импортом/Экспортом данных в систему



Рисунок 1.9 – Экран с просмотром статистики

The screenshot shows a web interface for managing an order. On the left, there is a list of items under the heading 'НАЗНАЧИТЬ ЗАМЕРЩИКА' (Assign measurer). The first item is 'ЗАМЕРЩИК 1' (Measurer 1) with a checkbox checked. Below it are several 'List item' entries. On the right, there is a section titled 'ИНФОРМАЦИЯ О ЗАКАЗЕ' (Order Information) containing the following details: 'ФИО заказчика : Иван Иванов Иванович' (Client name: Ivan Ivanovich Ivanovich), 'Номер телефона заказчика: +79211110055' (Client phone number: +79211110055), 'email : test@mail.ru' (Email: test@mail.ru), and 'Комментарий: Хочу классный замер' (Comment: I want a great measurement). A blue button labeled 'подтвердить' (Confirm) is at the bottom right.

Рисунок 1.10 – Экран с просмотром информации о заказе/назначение
замерщика

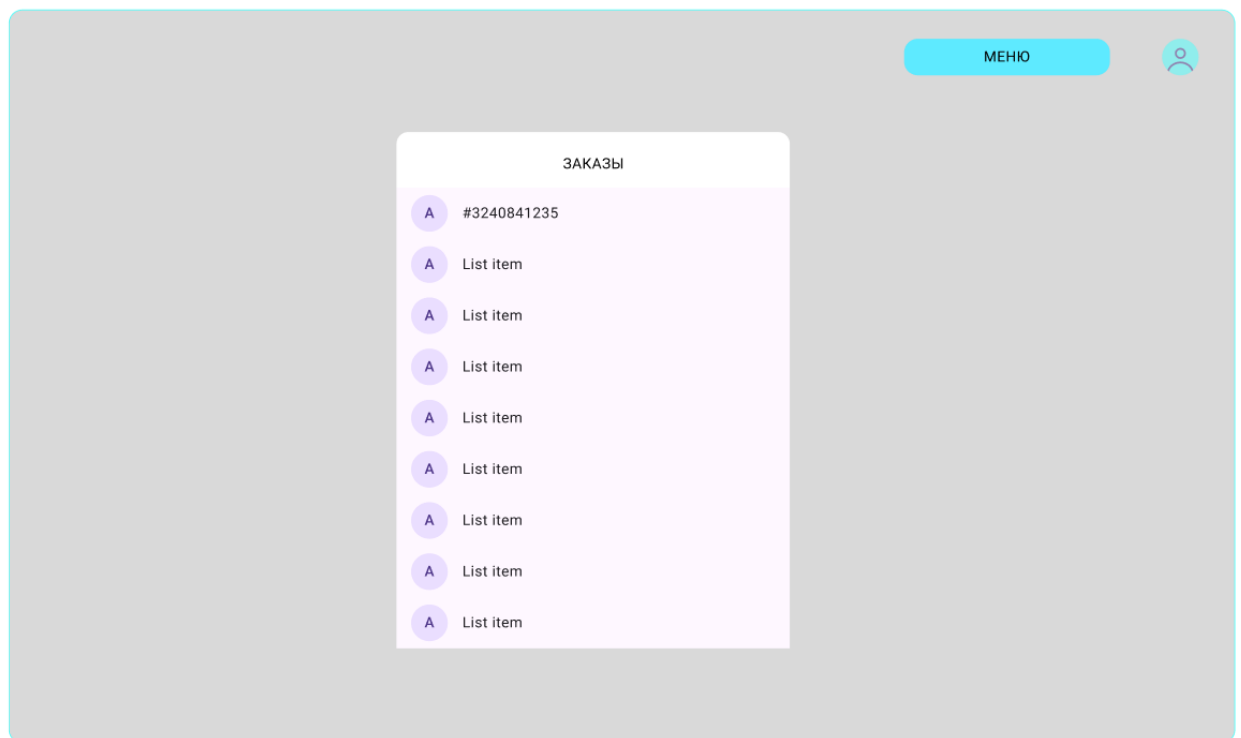


Рисунок 1.11 – Экран с просмотром имеющихся заказов, замерщиком

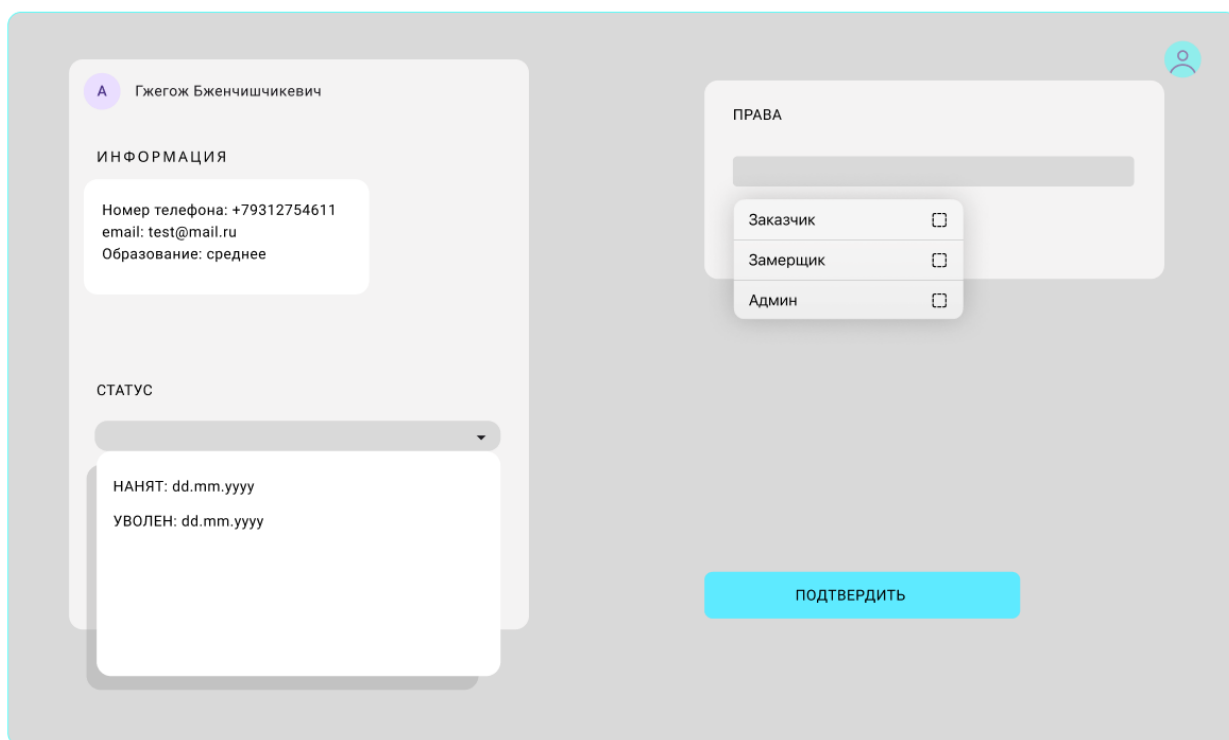


Рисунок 1.12 – Экран СуперАдмина, возможность менять роли пользователям

1.2. Сценарий использования

1. Регистрация и авторизация

Описание

Пользователь может зарегистрироваться в системе или войти в уже существующий аккаунт. После успешной авторизации система определяет его роль и предоставляет соответствующий интерфейс.

Шаги

Регистрация

1. Пользователь переходит на страницу регистрации.
2. Вводит следующие данные:

- Имя
- Пароль
- Телефон
- Дата рождения

3. Нажимает кнопку "Подтвердить".

Авторизация

1. Пользователь переходит на страницу входа.
2. Вводит телефон и пароль.
3. Нажимает кнопку "Войти".
4. Система проверяет данные и определяет роль пользователя.
5. После успешной авторизации пользователь попадает в личный кабинет.

Результат

- Пользователь зарегистрирован и подтверждён.
- В зависимости от роли открывается соответствующий интерфейс.

2. Просмотр каталога окон и заказ замера

Описание

Покупатель может изучать каталог окон, использовать фильтры для поиска нужных товаров и оформлять заявку на замер.

Шаги

1. Пользователь открывает каталог окон.
2. Использует фильтры для поиска по параметрам:
 - Размер
 - Материал
 - Цена

3. Выбирает интересующую модель и открывает страницу товара.
4. Нажимает кнопку "Заказать".
5. Заполняет форму заявки:
 - Адрес
 - Удобная дата и время
 - Почта
6. Подтверждает заявку.

Результат

- Заявка на замер создана.
- Покупатель получает уведомление о подтверждении заявки.

3. Обработка заявки на замер (Администратор)

Описание

Администратор обрабатывает заявки на замер, назначает замерщиков и контролирует процесс.

Шаги

1. Администратор входит в панель управления.
2. Открывает раздел "Заявки на замер".
3. Просматривает список новых заявок.
4. Выбирает заявку и назначает замерщика.
5. Подтверждает назначение.
6. Система уведомляет замерщика.

Результат

- Заявка обработана, замерщик назначен.
- Замерщик получает уведомление.

4. Выполнение замера (Замерщик)

Описание

Замерщик получает назначение, выполняет замер на объекте и фиксирует данные в системе.

Шаги

1. Входит в личный кабинет.
2. Просматривает список назначенных замеров.
3. Выбирает заявку, изучает детали (адрес, контакты).
4. Выезжает на объект и выполняет замер.
5. Подтверждает завершение замера.

Результат

- Данные замера сохранены в системе.
- Администратор получает уведомление.

5. Создание заказа (Администратор)

Описание

На основе данных замера администратор создаёт заказ и передаёт его в производство.

Шаги

1. Открывает панель управления.
2. Переходит в раздел "Замеры".
3. Выбирает завершённый замер.
4. Создаёт заказ:

- Устанавливает стоимость
 - Определяет сроки
 - Назначает ответственного за производство
5. Подтверждает создание заказа.

Результат

- Заказ передан в производство.
- Покупатель получает уведомление.

6. Оплата заказа (Покупатель)

Описание

Покупатель может оплатить заказ онлайн или при получении.

Шаги

1. Входит в личный кабинет.
2. Открывает раздел "Мои заказы".
3. Выбирает заказ и нажимает "Оплатить".
4. Выбирает способ оплаты:
 - Онлайн (картой)
 - По факту (наличными/картой при получении)
5. Оплачивает.

Результат

- Заказ переходит в статус "Оплачен" или "Ожидает оплаты".
- Администратор получает уведомление.

7. Управление системой (SuperAdmin)

Описание

SuperAdmin управляет аккаунтами администраторов, анализирует статистику и настраивает систему.

Шаги

1. Входит в панель управления.
2. Управляет аккаунтами администраторов:
 - Создает
 - Редактирует
 - Удаляет
3. Анализирует статистику заказов и оплат.

Результат

- Система настроена и управляется.
- Доступна полная аналитика.

8. Импорт данных в систему

Описание

Администратор или SuperAdmin может импортировать данные в систему из внешних источников. Это может быть полезно для загрузки каталога окон, списка замерщиков или других данных.

Шаги

1. Администратор/SuperAdmin входит в панель управления.
2. Переходит в раздел "Импорт данных".
3. Выбирает тип данных для импорта:
 - Каталог окон
 - Список замерщиков

- Заказы из внешней системы
- 4. Система проверяет данные на корректность.
- 5. Система сохраняет корректные данные или предоставляет отчёт с ошибками.

Результат

- Данные успешно импортированы.
- Предоставляется отчёт с проблемными записями при наличии ошибок.

9. Экспорт данных из системы

Описание

Администратор или SuperAdmin может экспортировать данные из системы для анализа, отчётности или интеграции с другими системами. Экспортируемые данные могут включать заказы, замеры, каталог окон и другую информацию.

Шаги

1. Администратор/SuperAdmin входит в панель управления.
2. Переходит в раздел "Экспорт данных".
3. Выбирает тип данных для экспорта:
 - Заказы
 - Замеры
 - Каталог окон
 - Пользователи
4. Указывает фильтры для выборки данных (если необходимо):
 - Дата начала и окончания
 - Статус заказа
 - Тип окна

5. Выбирает формат экспорта:
 - JSON
6. Нажимает кнопку "Экспорт".
7. Система формирует файл и предоставляет для скачивания.

Результат

- Данные успешно экспортированы в выбранный формат.

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

2. МОДЕЛЬ ДАННЫХ

2.1. Нереляционная модель

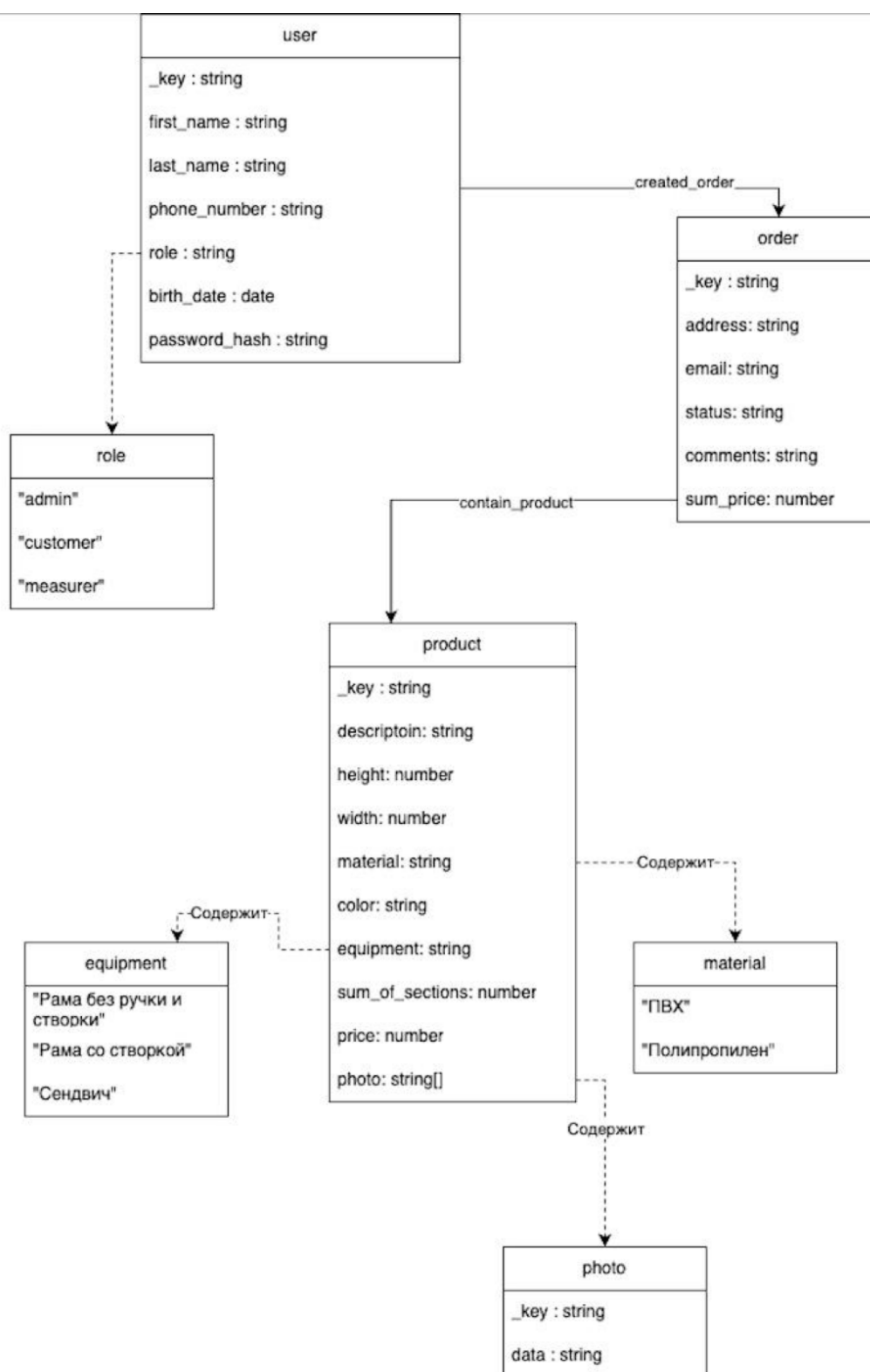


Рисунок 2.1 – Графическое представление

Сущности

user

Описание

Сущность "User" содержит данные о зарегистрированных пользователях в системе, используется для управления учетными записями и обеспечения безопасности доступа.

Поля

- **_key** : Уникальный идентификатор пользователя.
 - Тип данных : 'String'
- **first_name** : Имя пользователя.
 - Тип данных : 'String'
- **last_name** : Фамилия пользователя.
 - Тип данных : 'String'
- **phone_number** : Номер телефона пользователя.
 - Тип данных : 'String'
- **role** : Роль пользователя.
 - Тип данных : 'String'
- **birth_date** : Дата рождения пользователя.
 - Тип данных : 'Date'
- **password_hash**: Хэшированный пароль пользователя.
 - Тип данных: 'string'

order

Описание

Сущность "order" содержит в себе данные о созданном заказе, пользователем.

Поля

- **_key** : Уникальный идентификатор заказа.
 - Тип данных : 'string'
- **address** : Адрес доставки товара.
 - Тип данных : 'string'
- **email** : Email адрес заказчика.
 - Тип данных : 'string'
- **status** : Статус заказа : Выполнен/Открыт.
 - Тип данных : 'string'
- **comments** : Комментарий к заказу.
 - Тип данных : 'string'
- **sum_price** : Сумма заказа.
 - Тип данных : 'number'

product

Описание

Сущность "product" содержит в себе данные о конкретном товаре из заказа.

Поля

- **_key** : Уникальный идентификатор товара.
 - Тип данных : 'String'
- **description** : Описание товара.
 - Тип данных : 'String'
- **height** : Высота окна.
 - Тип данных : 'number'
- **width** : Ширина окна.
 - Тип данных : 'number'
- **material** : Материал окна (ПВХ / Полипропилен).

- Тип данных : 'string'
- **color** : Цвет окна.
 - Тип данных : 'string'
- **equipment** : оборудование (Рама без ручки и створки/Рама со створкой/Сендвич)
 - Тип данных: 'string'
- **sum_of_sections** : Количество секций окна.
 - Тип данных : 'number'
- **price** : Цена товара за одну штуку.
 - Тип данных : 'number'
- **photos** : Фото окна.
 - Тип данных : 'string[]'

photo

Описание

Сущность "photo" содержит в себе фото конкретного товара.

Поля

- **_key** : Уникальный идентификатор.
 - Тип данных : 'string'
- **data** : Данные фотографии.
 - Тип данных : 'string'

edge-коллекции

|name |Откуда(_from)|Куда(_to)|Описание |

|created_order |user |order |Пользователь создал заказ |

|contain_product|order |product |Какие товары входят в заказ|

Пример

created_order

```
{ "_from": "user/abc123", "_to": "order/ord001" }
```

Расчет размера записи

user:

- **_key** : 12 байт
- **first_name** : 41 байт (~20 символов на Кириллице)
- **last_name** : 41 байт (~20 символов на Кириллице)
- **phone_number** : 13 байт (11 цифр и символ "+")
- **role** : 16 байт (Заказчик/Замерщик/Админ - на Кириллице)
- **birth_date** : 11 байт (Формат "YYYY-MM-DD")
- **password_hash**: 65 байт (строка SHA-256 хэша)

Итого : **199 байт**

order:

- **_key** : 12 байт
- **address** : 121 байт (60 символов на Кириллице)
- **email** : 36 байт (35 символов на Латинице)
- **status** : 17 байт ("Выполнен" или "Открыт" на Кириллице)
- **comments** : 101 байт (50 символов на Кириллице)
- **order_list** : 120 байт (Массив из 10 ссылок на товары (пример: ["prod1", "prod2", ...])), каждая ссылка ~12 байт)
- **sum_price** : 8 байт (числовой тип)

Итого : **415 байт**

product:

- **_key** : 12 байт
- **description** : 201 байт (100 символов на Кириллице)
- **height** : 8 байт (Число)
- **weight** : 8 байт (Число)
- **material** : 25 байт ("ПВХ" или "Пропилен" на Кириллице)
- **color** : 25 байт (Строка с цветом на кириллице, например "белый")
- **equipment** : 50 байт
- **sum_of_sections** : 8 (числовой тип)
- **price** : 8 (числовой тип)
- **photos** : 48 (Массив из 4 ссылок на фотографии (12 * 4))

Итого : **393 байта**

photo:

- **_key** : 12 байт
- **data** : 512000 байт (фото (~500 КБ))

Итого : **512012 байт**

Сводная таблица размеров

--Сущность--	--Размер(байт)--
----user----	--199-----
---order---	--415-----
--product---	--393-----
--photo-----	--512012-----

Формула для расчета размера всей модели с учетом всех сущностей и хранением фото

$$TotalSize = UserSize \times N_u + OrderSize \times N_o + ProductSize \times N_p + PhotoSize \times N_{ph}, \text{ где}$$

N_u = количество пользователей $N_o = 2 \times N_u$ (2 заказа на пользователя) $N_p = 10 \times N_o = 20 \times N_u$ (10 товаров в заказе) $N_{ph} = 4 \times N_p = 80 \times N_u$ (4 фото на товар)

$$TotalSize = 199 \times N_u + 415 \times (2 \times N_u) + 393 \times (20 \times N_u) + 512012 \times (80 \times N_u), \text{ упростим формулу}$$

$$TotalSize \approx 199N_u + 830N_u + 6,860N_u + 40,960,960N_u + 21.5N_u + 470 + 512012 + 512012 + 744$$

Итоговая формула: $TotalSize \approx 40969849 \times N_u$ [байт]

Избыточность данных

- Для расчета чистых данных из предложенной модели были убраны:
- Все поля `_key` (уникальные идентификаторы по 12 байт)
- Технические поля (хэши, статусы)
- Размеры сущностей без технических полей
- Сущность Исключено Новый размер (байт) `user _key` (12), `password_hash` (65) $199 - 77 = 122$ `order _key` (12), `status` (17) $415 - 29 = 386$ `product _key` (12) $393 - 12 = 381$ `photo _key` (12) $512012 - 12 = 512000$
- $CleanSize = 122 \times N_u$ (user) + $386 \times N_o$ (order) + $381 \times N_p$ (product) + $512000 \times N_{ph}$ (photo) Подставим зависимости между сущностями: $N_o = 2 \times N_u$ (2 заказа на пользователя) $N_p = 10 \times N_o = 20 \times N_u$ (10 товаров в заказе) $N_{ph} = 4 \times N_p = 80 \times N_u$ (4 фото на товар)

- $\text{CleanSize} = 122 \times N_u + 386 \times (2 \times N_u) + 381 \times (20 \times N_u) + 512000 \times (80 \times N_u)$ Упростим и получим итоговую формулу: $\text{CleanSize} \approx 40968514 \times N_u$ [байт]
- $\text{Избыточность} = \text{TotalSize} / \text{CleanSize} \approx 1.00003$

Направление роста модели при увеличении количества объектов каждой сущности.

Размер модели растет линейно по каждому из параметров, что следует из формул выше.

Примеры данных

Пример 1

user: { "_key": "user123", "first_name": "Иван", "last_name": "Петров", "phone_number": "+79111234567", "role": "Заказчик", "birth_date": "1990-05-15", "password_hash": "a1b2c3d4e5f6...64" // SHA-256 хэш }

order: { "_key": "order456", "address": "г. Москва, ул. Ленина, д. 10, кв. 5", "email": "client@mail.ru", "status": "Открыт", "comments": "Подъезд со двора", "order_list": ["prod789", "prod012"], "sum_price": 25000 }

product: { "_key": "prod789", "description": "Окно ПВХ, белое, 2 створки", "height": 1.5, "width": 1.2, "material": "ПВХ", "color": "белый", "equipments": "Сендвич", "sum_of_sections": 2, "price": 12500, "photos": ["photo1", "photo2"] }

photo:

```
{ "_key": "photo123", "filepath": "/storage/products/photo123.jpg", "size_kb": 245, "uploaded_at": "2023-10-20T12:00:00Z" }
```

Пример 2

user:

```
{ "_key": "MASHKA", "first_name": "Мария", "last_name": "Машикова", "phone_number": "+79110987654", "role": "Заказчик", "birth_date": "2001-03-04", "password_hash": "vjs4hs7haj9v...23" // SHA-256 хэш }
```

order:

```
{ "_key": "order457", "address": "г. Москва, ул. Трубецкая, д. 7, кв. 38", "email": "mashka@mail.ru", "status": "Открыт", "comments": "Позвоните за 20 минут", "order_list": ["prod7"], "sum_price": 14000 }
```

product:

```
{ "_key": "prod7", "description": "Окно ПВХ, белое, 1 створка", "height": 1.5, "width": 1.0, "material": "ПВХ", "color": "белый", "equipments": "сендвич", "sum_of_sections": 1, "price": 14000, "photos": ["photo1"] }
```

photo:

```
{ "_key": "photo1", "filepath": "/storage/products/photo1.jpg", "size_kb": 245, "uploaded_at": "2023-10-20T12:00:00Z" }
```

Примеры запросов

Регистрация пользователя

```
LET new_user = { key: CONCAT("user", FIRST(RETURN NEW_UUID())),  
first_name: "Мария", last_name: "Семёнова", phone_number: "+79161234567",  
birth_date: "1995-07-22", password_hash:  
"5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8",  
created_at: DATE_NOW() } INSERT new_user INTO users RETURN new_user
```

Коллекции: 1 (users)

Авторизация пользователя

```
LET auth_data = FIRST( FOR u IN users FILTER u.phone_number ==  
"+79161234567" RETURN { valid: (u.password_hash ==  
"5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8"),  
user: u } ) RETURN auth_data.valid ? auth_data.user : null
```

Коллекции: 1 (users)

Создание заказа пользователем

```
LET order_id = CONCAT("order_", FIRST(RETURN NEW_UUID())) LET  
products = [ { key: "prod" + FIRST(RETURN NEW_UUID()), description: "Окно  
ПВХ 1500×1200", height: 1.5, width: 1.2, material: "ПВХ", color: "белый",  
equipment: "Сендвич", price: 12500 }, { key: "prod" + FIRST(RETURN  
NEW_UUID()), description: "Дверь балконная", height: 2.1, width: 0.7, material:  
"ПВХ", color: "коричневый", price: 8900 } ]
```

```
LET inserted_products = ( FOR p IN products INSERT p INTO products  
RETURN p._key )
```



```
LET new_order = { _key: order_id, user: "user_12345", // ID пользователя,
который создает заказ address: "г. Москва, ул. Тверская, 15, кв. 42", sum_price:
SUM(products[*].price), status: "new", created_at: DATE_NOW() }
```

```
INSERT new_order INTO orders
```

```
LET created_edges = ( FOR product_key IN inserted_products INSERT {
_from: "orders/" + order_id, _to: "products/" + product_key } INTO
contain_product RETURN 1 )
```

```
LET user_order_edge = { _from: "users/" + new_order.user, // ID
пользователя, который создал заказ _to: "orders/" + order_id }
```

```
INSERT user_order_edge INTO created_order
```

```
RETURN MERGE(new_order, { products: DOCUMENT('products',
inserted_products) })
```

Коллекции: 3 (orders, products, user)

Поиск товаров с фильтрами

```
FOR p IN products FILTER p.price >= 10000 AND p.price <= 15000 FILTER
p.material == "ПВХ" FILTER p.color IN ["белый", "серый"] SORT p.price ASC
LIMIT 10 RETURN MERGE(p, { photos: ( FOR photo IN photos FILTER
photo.product == p._key RETURN UNSET(photo, ['data']) ) })
```

Коллекции: 2 (products, photos)

2.2. Реляционная модель

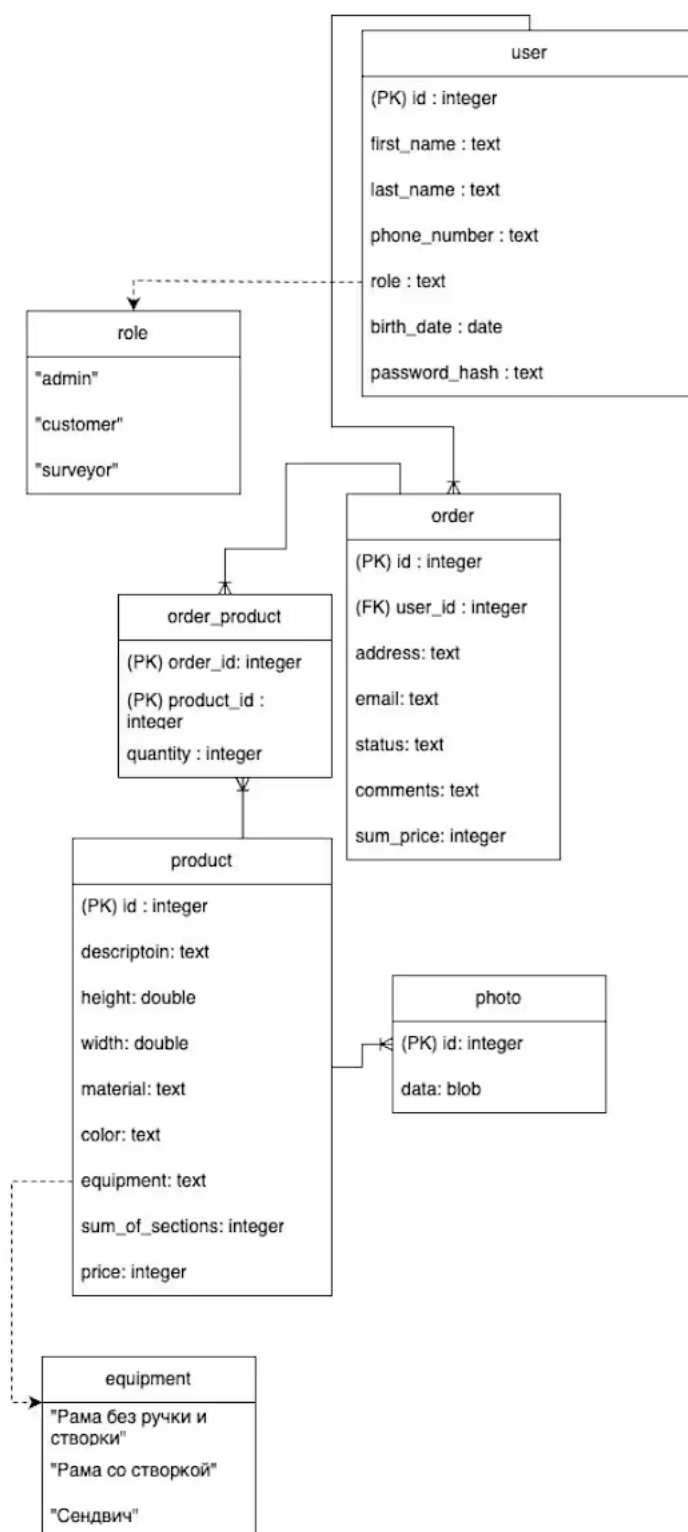


Рисунок 2.2 – Графическое представление

Сущности

user

Описание

Сущность "User" содержит данные о зарегистрированных пользователях в системе, используется для управления учетными записями и обеспечения безопасности доступа.

Поля

- **id** : Уникальный идентификатор пользователя.
 - Тип данных : 'integer'
- **first_name** : Имя пользователя.
 - Тип данных : 'text'
- **last_name** : Фамилия пользователя.
 - Тип данных : 'text'
- **phone_number** : Номер телефона пользователя.
 - Тип данных : 'text'
- **role** : Роль пользователя.
 - Тип данных : 'text'
- **birth_date** : Дата рождения пользователя.
 - Тип данных : 'date'
- **password_hash**: Хэшированный пароль пользователя.
 - Тип данных: 'text'

order

Описание

Сущность "order" содержит в себе данные о созданном заказе, пользователем.

Поля

- **id** : Уникальный идентификатор заказа.
 - Тип данных : 'integer'
- **user_id** : Уникальный идентификатор заказа.
 - Тип данных : 'integer'
- **address** : Адрес доставки товара.
 - Тип данных : 'text'
- **email** : Email адрес заказчика.
 - Тип данных : 'text'
- **status** : Статус заказа : Выполнен/Открыт.
 - Тип данных : 'text'
- **comments** : Комментарий к заказу.
 - Тип данных : 'text'
- **sum_price** : Сумма заказа.
 - Тип данных : 'integer'

product

Описание

Сущность "product" содержит в себе данные о конкретном товаре из заказа.

Поля

- **id** : Уникальный идентификатор товара.
 - Тип данных : 'integer'
- **description** : Описание товара.
 - Тип данных : 'text'
- **height** : Высота окна.
 - Тип данных : 'double'
- **width** : Ширина окна.

- Тип данных : 'double'
- **material** : Материал окна (ПВХ / Полипропилен).
 - Тип данных : 'text'
- **equipment** : Вариация окна (Рама без ручки и створки/ Рама со створкой/ Сэндвич).
 - Тип данных: 'text'
- **color** : Цвет окна.
 - Тип данных : 'text'
- **sum_of_sections** : Количество секций окна.
 - Тип данных : 'integer'
- **price** : Цена товара за одну штуку.
 - Тип данных : 'integer'

order_product

Описание

Связывает товары (product) с заказами (order), учитывая количество каждого товара.

Поля

- **order_id** : Ссылка на заказ. Тип : 'integer'
- **product_id** : Ссылка на товар. Тип : 'integer'
- **quantity** : Количество единиц товара. Тип : 'integer'

photo

Описание

Сущность "photo" содержит в себе фото конкретного товара.

Поля

- **id** : Уникальный идентификатор.
 - Тип данных : 'integer'

- **data** : Данные фотографии.
 - Тип данных : 'blob'

Расчет размера записи

user

- **id** : 4 байта (integer)
- **first_name** : 41 байт (~20 символов на Кириллице)
- **last_name** : 41 байт (~20 символов на Кириллице)
- **phone_number** : 13 байт (11 цифр + "+")
- **role** : 16 байт ("Заказчик"/"Замерщик"/"Админ")
- **birth_date** : 4 байта (date)
- **password_hash** : 65 байт (SHA-256 хэш) Итого: **184 байт**

order:

- **id** : 4 байта (integer)
- **user_id** : 4 байта (integer)
- **address** : 121 байт (~60 символов)
- **email** : 36 байт (~35 символов)
- **status** : 17 байт
- **comments** : 101 байт (~50 символов)
- **sum_price** : 4 байта (integer) Итого: **287 байт**

order_product:

- **order_id** : 4 байта (integer)
- **product_id** : 4 байта (integer)
- **quantity** : 4 байта (integer) Итого: **12 байт**

product:

- **id** : 4 байта (integer)
- **description** : 201 байт (~100 СИМВОЛОВ)
- **height** : 8 байт (double)
- **width** : 8 байт (double)
- **material** : 25 байт
- **color** : 25 байт
- **equipment** : 50 байт
- **sum_of_sections** : 4 байта
- **price** : 4 байта Итого: **329 байт**

photo:

- **id** : 4 байта (integer)
- **data** : 512000 байт (~500 КВ) Итого: **512004 байт**

Сводная таблица размеров

Сущность	Размер(байт)
user	184
order	287
order_product	12
product	329
photo	512004

Формула для расчета размера всей модели с учетом всех сущностей и хранением фото

$$TotalSize = (UserSize \times N_u) + (OrderSize \times N_o) + (ProductSize \times N_p) + (OrderProductSize \times N_{op}) + (PhotoSize \times N_{ph}), \text{ где}$$

N_u = количество пользователей $N_o = 2 \times N_u$ (2 заказа на пользователя) $N_p = 10 \times N_o = 20 \times N_u$ (10 товаров в заказе) $N_{ph} = 4 \times N_p = 80 \times N_u$ (4 фото на товар) $N_{op} = N_p = 20 \times N_u$ (каждый товар в заказе имеет запись в *order_product*)

$$TotalSize = 184 \times N_u + 287 \times 2N_u + 329 \times 20N_u + 12 \times 20N_u + 512004 \times 80N_u$$

Упростим и получим: $TotalSize \approx 40967898 \times N_u$ [байт]

Избыточность данных

Для расчета чистых данных были исключены:

Все ID поля (4 байта на каждое) Технические поля (статусы, связи между таблицами)

Размеры сущностей без технических полей: Сущность Исключено Новый размер (байт) user id (4) $184 - 4 = 180$ order id (4), user_id (4), status (17) $287 - 25 = 262$ product id (4) $329 - 4 = 325$ photo id (4) $512004 - 4 = 512000$ order_product order_id (4), product_id (4) $12 - 8 = 4$

$$CleanSize = 180 \times N_u + 262 \times 2N_u + 325 \times 20N_u + 512000 \times 80N_u + 4 \times 20N_u$$

Упростим: $CleanSize \approx 40967284 \times N_u$ [байт]

Избыточность = $TotalSize / CleanSize \approx 1.000015$

Направление роста модели при увеличении количества объектов каждой сущности

Размер модели растет линейно по каждому из параметров, что следует из формул выше.

Примеры данных

Таблица 2.1 - User

d	first_name	last_name	phone_number	role	birth_date	password_hash
	Иван	Петров	+79111234567	user	1990-05-15	a1b2c3d4e5f6...
	Мария	Сидорова	+79117654321	user	1985-11-23	x7y8z9a1b2c...
	Алексей	Смирнов	+79119876543	surveyor	1982-03-30	m3n4o5p6q7...
	Екатерина	Иванова	+79115556677	admin	1978-07-12	r8s9t0u1v2...

Таблица 2.2 - product

d	description	height	width	material	color	sum_of_section	price
	Окно ПВХ белое 2- створчатое	1.5	1.5	ПВХ	белый	2	125
	Окно ПВХ коричневое 3- створч.	1.8	1.5	ПВХ	коричневый	3	18000
	Окно	1.2	0.9	Полипропилен	серый	1	9500

	пропилен серое 1- створч.			Н			
--	---------------------------------	--	--	---	--	--	--

Таблица 2.3 - order

id	user_id	address	email	status	comments	sum_price
101	1	г. Москва, ул. Ленина, д.10, кв.5	ivan@mail.ru	Открыт	Позвонить за час	25000
102	2	г. СПб, Невский пр., д.15	maria@mail.ru	Выполнен	Код домофона 145	18000
103	3	г. Москва, ул. Тверская, д.20	ivan@mail.ru	Открыт	Оставить у консьержа	9500

Таблица 2.4 - order_product:

order_id	product_id	Quantity
101	1	2
102	2	1
103	3	1

Примеры запросов

Регистрация пользователя

```
INSERT INTO user (first_name, last_name, phone_number, birth_date, password_hash) VALUES ('Иван', 'Петров', '+79111234567', 'user', '1990-05-15', 'a1b2c3d4e5f6...');
```

Создается новая запись в таблице пользователей с указанными данными. В системе появляется новый пользователь. Коллекции: 1 (user) Количество запросов: 1

Получение отфильтрованного списка товаров

```
SELECT id, description, price FROM product WHERE material = 'ПВХ' AND height BETWEEN 1.0 AND 2.0 AND price <= 15000;
```

Выборка оконных конструкций по заданным параметрам (материал, размер, цена). Результат: Список товаров, соответствующих критериям поиска Коллекции: 1 (product) Количество запросов: 1

Получение детальной информации о товаре

```
SELECT p.*, ph.data as photo FROM product p LEFT JOIN photo ph ON p.id = ph.product_id WHERE p.id = 1;
```

Получение полной информации о конкретном товаре с его фотографиями. Результат: Все характеристики товара и связанные с ним изображения Коллекции: 2 (product, photo) Количество запросов: 1

Создание нового заказа

```
INSERT INTO order (user_id, address, email, status, comments, sum_price) VALUES (1, 'г. Москва, ул. Ленина, 10', 'ivan@mail.ru', 'Открыт', 'Позвонить за час', 0);
```

Формирование новой заявки на замер окон Результат: В системе создается запись о новом заказе со статусом "Открыт". Коллекции: 1 (order). Количество запросов: 1

Добавление товаров в заказ

```
INSERT INTO order_product (order_id, product_id, quantity) VALUES (101, 1, 2);
```

Что происходит: Привязка выбранных товаров к созданному заказу с указанием количества. Результат: Товары ассоциируются с конкретным заказом Коллекции: 1 (order_product) Количество запросов: 1 на каждый добавляемый товар

Назначение замерщика

```
INSERT INTO surveyor_order (surveyor_id, order_id) VALUES (301, 101);
```

Закрепление заявки за конкретным специалистом-замерщиком. Результат: Замерщик получает новый заказ Коллекции: 1 (surveyor_order) Количество запросов: 1

Обновление статуса заказа

```
UPDATE order SET status = 'Выполнен' WHERE id = 101;
```

Изменение статуса заказа после выполнения замера Результат: Заказ переходит в статус производства Коллекции: 1 (order) Количество запросов: 1

2.3. Сравнение моделей

Удельный объем информации

Сущность User:

- NoSQL: 199 байт
- SQL: 184 байт

В NoSQL объем данных может быть немного больше из-за поля `_key` и структуры JSON, в то время как SQL использует более компактные типы данных, что делает его более экономичным по памяти.

Сущность Order:

- NoSQL: 415 байт
- SQL: 287 байт

SQL занимает меньше, так как отсутствует вложенность и нет JSON-структуры. Однако не учитывается сущность `order_product` (см. ниже).

Связанные товары: Product и Order_Product

- NoSQL:
 - Product встроен в Order: 393 байт на 1 товар
- SQL:
 - Product: 329 байт
 - Order_Product: 12 байт (на 1 товар в заказе)
 - Итого: $329 + 12 = 341$ байт на 1 товар

SQL более экономичен, но требует двух таблиц и джойна.

Сущность Photo:

- NoSQL: 512012 байт
- SQL: 512004 байт

Разница почти незаметна — 8 байт. Может быть из-за структуры хранения метаданных в JSON (NoSQL)

Итог:

SQL, как правило, более эффективен по объему данных благодаря использованию компактных типов данных (целочисленные идентификаторы) и отсутствию дополнительных метаданных, в то время как NoSQL может требовать большего объема из-за структуры JSON и наличия дополнительных полей, таких как `_key`.

Запросы по отдельным юзкейсам

Юзкейс: Регистрация пользователя

NoSQL:

- Коллекции: 1 (users)
- Запросов: 1

Простая вставка в коллекцию с автогенерацией ключа

SQL:

- Таблицы: 1 (user)
- Запросов: 1

Тоже простая вставка

Вывод: Одинаково просто реализуется в обеих СУБД. Различия незначительны.

Юзкейс: Создание заказа пользователем

NoSQL:

- Коллекции: 3 (orders, products, users)
- Количество запросов: 4 (вставка продуктов, заказа, связей с товарами и пользователем)

Товары хранятся как объекты, без необходимости выделять отдельную таблицу связей. Можно сразу вернуть заказ с вложенными товарами (через DOCUMENT). Упрощается сериализация данных для клиента (например, в JSON).

SQL:

- Таблицы: 2 (order, order_product)
- Запросов: минимум 2 (1 — создание заказа, 1+ — на каждый товар в заказе)

Связь между заказом и товарами реализуется через промежуточную таблицу order_product. Количество товаров учитывается явно. Структура жёсткая, но обеспечивает надёжную целостность данных.

Вывод: NoSQL обеспечивает большую гибкость и быстрее реализует создание заказа с вложенными товарами в одной структуре. Это особенно удобно на этапе быстрого прототипирования, а также при возможных изменениях требований в будущем.

Юзкейс: Поиск и просмотр товаров

NoSQL:

- Коллекции: 2 (products, photos)
- Запросов: 1 (комбинированный — фильтрация + подзапрос для фотографий)

Фильтрация по нескольким параметрам (цена, материал, цвет), дополнительно загружаются фотографии через вложенный подзапрос. Гибкая работа с вложенными структурами.

SQL:

- Таблицы: 2 (product, photo)
- Запросов: 2 (или 1 — если комбинированный запрос со JOIN, как в примере)

Фильтрация по параметрам в одной таблице (материал, размеры, цена)
Получение полной информации с изображениями — через LEFT JOIN

Вывод: Обе СУБД позволяют эффективно выполнять фильтрацию и объединение данных. NoSQL удобен для гибкой работы с вложенными структурами и выборки частичных данных (например, без photo.data). SQL обеспечивает более прозрачные и оптимизированные объединения с помощью JOIN и чёткую типизацию.

Итог

- NoSQL — преимущественно подходит для гибкой, масштабируемой работы с данными, особенно если структура данных изменяется или если требуется высокая скорость разработки с минимальными требованиями к целостности данных.
- SQL — более предпочтителен для строгих логик, когда нужно гарантировать целостность данных, чёткие связи между сущностями и сложную аналитику.

Общий Вывод:

- SQL лучше подходит для случаев, где требуется строгая целостность данных, чёткие связи между сущностями и сложная аналитика, благодаря компактности, оптимизированным запросам и поддержке транзакций.
- NoSQL предпочтительнее для гибких и масштабируемых решений, особенно если структура данных изменяется или требуется высокая скорость разработки с минимальными требованиями к целостности

данных, при этом он позволяет легко работать с вложенными структурами и изменяющимися данными.

Для нашего интернет-магазина окон наиболее подходящей является NoSQL база данных, так как она обеспечивает гибкость в структуре данных, позволяя легко адаптировать систему под изменения без необходимости модификации схемы. NoSQL также лучше подходит для горизонтального масштабирования, что важно при росте бизнеса, и эффективен при работе с большими объемами данных, обеспечивая высокую производительность при запросах. В дополнение, NoSQL позволяет проще управлять вложенными данными, такими как товары в заказах, и ускоряет процесс разработки за счет меньшей сложности запросов.

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

3.1. Краткое описание

Модуль `db.py` для работы с базой данных ArangoDB:

- Устанавливает соединение с ArangoDB и создает БД при его первом обращении
- Инициализирует структуру БД: коллекции и связи между ними
- Заполняет БД тестовыми данными
- Реализует хеширование паролей с помощью `bcrypt`

Модуль `main.py` – Fast-API-приложение для правления заказами оконных-конструкций:

- Реализует аутентификацию (JWT) и авторизацию (роли: `customer`, `measurer`, `admin`, `superadmin`)
- Просмотр и фильтрация каталога оконных конструкций
- Создание и управление заказами
- Профиль пользователя
- Админ-панель (Управление пользователями, товарами, заказами)
- Поддерживает фильтры для каталога (По цене, размерам, материалу и др.)
- Включает API для проверки состояния системы

Использована двухсервисная архитектура приложения:

- `app` – сервис FastAPI – приложения
- `db` – сервис ArangoDB

Порт приложения пробрасывается только на `localhost`, пароль БД задается через переменные окружения, используется отдельный `.env` файл для конфигурации.

Используется легковесный официальный образ ArangoDB.

Для запуска необходимо:

1. Создать .env файл
2. Развернуть приложение через docker

3.2. Схема экранов приложения

Экраны приложения и переходы между ними см. на рисунке 3.1

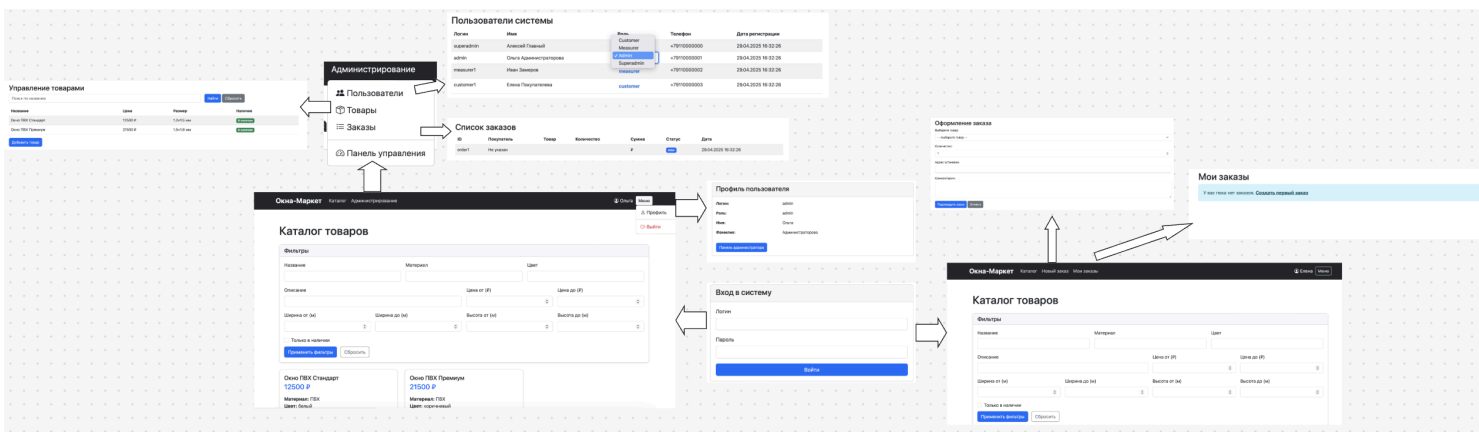


Рисунок 3.1 – Схема экранов приложения

3.3. Используемые технологии

БД: ArangoDB

Back-end: Python 3.9

Front-end: HTML, CSS, JS

4. ВЫВОДЫ

4.1. Достигнутые результаты

В ходе работы разработано приложение оконного завода, которое позволяет пользователям удобно и быстро добавлять товары в каталог, фильтровать товары в каталоге, менять роли пользователям, делать заказ пользователям, через панель администратора просматривать содержимое пользователей сайта, имеющихся товаров, заказов.

4.2. Недостатки и пути для улучшения полученного решения

На данный момент, не реализована регистрация новых пользователей в систему, импорт/экспорт, также необходимо добавить выброс ошибки, при добавлении некорректного товара в каталог, фильтры администратору во вкладку «Администрирование».

4.3. Будущее развитие решения

Реализация того, что не выполнено на данный момент, разработка нативных приложений для OS Windows и MacOS.

5. ЗАКЛЮЧЕНИЕ

Реализована ролевая модель (4 типа пользователей) с разными правами, работает каталог оконных конструкций с фильтрами (цена, размеры, материалы, описание, название, только в наличии). Есть личный кабинет для клиентов и панель управления для администраторов. Данные хранятся в ArangoDB с графовыми связями. Реализовано Docker-развертывание с автоинициализацией БД.

6. СПИСОК ЛИТЕРАТУРЫ

1. Ссылка на репозиторий: <https://github.com/moevm/nosql1h25-windowprod>
2. <https://docs.arangodb.com/stable/>

7. ПРИЛОЖЕНИЯ

5.1. Документация по сборке и развертыванию приложения

- Скачать проект из репозитория
- Развернуть приложение через docker командой *docker-compose build --no-cache && docker-compose up*
- Открыть приложение в браузере по адресу *localhost:8000*