Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление: 09.03.04 – Программная инженерия

Профиль: Разработка информационных систем

УДК 004.65

ОТЧЕТ

О КУРСОВОЙ РАБОТЕ

по дисциплине «Базы данных»

на тему

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ СТРОИТЕЛЬНОЙ ОРГАНИЗАЦИИ (65(3))

|  |  |
| --- | --- |
| Студент гр. РИС-23-1б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.Д. Мокрушин  подпись |
|  |  |
| Руководитель КР  Доцент кафедры ИТАС  канд. техн. наук, доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Петренко  подпись |
|  |  |

Пермь 2025

**РЕФЕРАТ**

Отчет 199 страниц, 11 рисунков, 2 таблицы, 7 источников, 9 приложений.

БАЗА ДАННЫХ, СТРОИТЕЛЬНАЯ ОРГАНИЗАЦИЯ, ВЕБ-ПРИЛОЖЕНИЕ, ASP.NET, C#, MY SQL, СУБД

Объектом исследования является исполнительные процессы строительной организации.

Предметом исследования является организация системы контроля за строительством объектов.

Целью курсовой работы является разработка информационной системы для автоматизации ключевых процессов строительной организации.

Задачи:

1. Анализ предметной области информационных систем автоматизации строительной организации.

2. Проектирование базы данных строительной организации.

3. Разработка база данных и веб-приложения.

В ходе курсового исследования применялись научные методы: анализ и моделирование.

Итогом работы стала разработанная база данных и веб-приложение для работы с ней.

Результат работы обладает практической значимостью, так как может повсеместно применяться в любых строительных организациях. Дальнейшее развитие может отразиться в совершенствовании данной системы исходя из нужд организации.

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc197622036)

[**1 Анализ предметной области** 5](#_Toc197622037)

[**1.1 Постановка задачи** 5](#_Toc197622038)

[**2. Проектирование базы данных** 11](#_Toc197622039)

[**2.1 Концептуальная модель** 11](#_Toc197622040)

[**2.2 Логическая модель** 14](#_Toc197622041)

[**2.3 Выбор системы управления базами данных** 16](#_Toc197622042)

[**2.4 Физическая модель** 18](#_Toc197622043)

[**2.6 Разграничение прав доступа** 25](#_Toc197622044)

[**3 Технология разработки базы данных и веб-приложения** 28](#_Toc197622045)

[**3.1 Реализация базы данных** 28](#_Toc197622046)

[**3.2 Разработка веб-приложения к базе данных** 32](#_Toc197622047)

[**3.3 Тестирование работы веб-приложения и базы данных** 37](#_Toc197622048)

[**ЗАКЛЮЧЕНИЕ** 42](#_Toc197622049)

[**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ** 44](#_Toc197622050)

[**ПРИЛОЖЕНИЕ А** 45](#_Toc197622051)

[**Концептуальная модель** 45](#_Toc197622052)

[**ПРИЛОЖЕНИЕ Б** 46](#_Toc197622053)

[**ПРИЛОЖЕНИЕ В** 47](#_Toc197622054)

[**ПРИЛОЖЕНИЕ Г** 48](#_Toc197622055)

[**ПРИЛОЖЕНИЕ Д Запросы на создание сущностей в базе данных** 49](#_Toc197622056)

[**ПРИЛОЖЕНИЕ Е** 60](#_Toc197622057)

[**ПРИЛОЖЕНИЕ Ж** 64](#_Toc197622058)

[**ПРИЛОЖЕНИЕ И** 71](#_Toc197622059)

**ВВЕДЕНИЕ**

Строительная отрасль требует точного планирования, координации большого числа ресурсов и строгого соблюдения сроков. Эффективное управление строительными процессами возможно только при наличии информационной системы, которая позволит в реальном времени контролировать расход материалов, использование техники и распределение рабочей силы.

Курсовая работа посвящена разработке системы управления строительной организацией. Она охватывает автоматизацию управления строительными управлениями и участками, учёт персонала и бригад, а также планирование и контроль графиков работ. Используемая база данных отражает реальную структуру строительного производства, включая объекты, клиентов, сметы и фактическое выполнение работ.

Объект исследования: производственные процессы строительной организации.

Предмет исследования: организация системы учета и контроля за строительством объектов.

Цель курсовой работы: разработать информационную систему для автоматизации ключевых процессов строительной организации.

Основные задачи исследования:

1. Анализ предметной области информационных систем автоматизации предприятий.

2. Проектирование базы данных авиастроительного предприятия.

3. Разработка базы данных и веб-приложения.

В ходе курсового исследования применялись научные методы: анализ и моделирование.

Актуальность работы обусловлена возрастающими требованиями к точности, надежности и скорости обработки информации в авиастроительной отрасли, где ошибки в управлении производством могут иметь серьезные последствия. Предлагаемое решение направлено на повышение эффективности управления производственными процессами через внедрение современной информационной системы.

# **1 Анализ предметной области**

## **1.1 Постановка задачи**

Для анализа предметной области рассмотрим постановку задачи 65 варианта.

Строительная организация занимается строительством различного рода объектов: жилых домов, больниц, школ, мостов, дорог и т.д. по договорам с заказчиками (городская администрация, ведомства, частные фирмы и т.д.). Каждая из перечисленных категорий объектов имеет характеристики, свойственные только этой или нескольким категориям: например, к характеристикам жилых домов относится этажность, тип строительного материала, число квартир, для мостов уникальными характеристиками являются тип пролетного строения, ширина, количество полос для движения.[1]

Структурно строительная организация состоит из строительных управлений, каждое строительное управление ведет работы на одном или нескольких участках, возглавляемых начальниками участков, которым подчиняется группа прорабов, мастеров и техников. Каждой категории инженерно-технического персонала (инженеры, технологи, техники) и рабочих (каменщики, бетонщики, отделочники, сварщики, электрики, шофера, слесари и пр.) также свойственны характерные только для этой группы атрибуты. Рабочие объединяются в бригады, которыми руководят бригадиры. Бригадиры выбираются из числа рабочих, мастера, прорабы, начальники участков и управлений назначаются из числа инженерно-технического персонала.[1]

На каждом участке возводится один или несколько объектов, на каждом объекте работу ведут одна или несколько бригад. Закончив работу, бригада переходит к другому объекту на этом или другом участке. Строительному управлению придается строительная техника (подъемные краны, экскаваторы, бульдозеры и т.д.), которая распределяется по объектам.

Технология строительства того или иного объекта предполагает выполнение определенного набора видов работ, необходимых для сооружения данного типа объекта. Например, для жилого дома — это возведение фундамента, кирпичные работы, прокладка водоснабжения и т.д. Каждый вид работ на объекте выполняется одной бригадой. Для организации работ на объекте составляются графики работ, указывающие, в каком порядке и в какие сроки выполняются те или иные работы, а также смета, определяющая, какие строительные материалы и в каких количествах необходимы для сооружения объекта. По результатам выполнения работ составляется отчет с указанием сроков выполнения работ и фактических расходов материалов.

**1.2 Требования, предъявляемые к базе данных**

В процессе анализа предметной области строительной организации были выделены следующие основные сущности:

* + - * Заказчик
* Строительное управление
* Участок
* Объект строительства
* Категория объекта
* Характеристика объекта
* Персонал
* Должность
* Категория персонала
* Атрибуты персонала
* Бригада
* Состав бригады
* Строительная техника
* Назначение техники
* Вид работ
* График работ
* Назначение работ бригадам
* Строительные материалы
* Смета / Расход материалов
* Отчет о выполнении работ

Основные компоненты системы и их взаимосвязи:

Строительная организация состоит из строительных управлений, каждое управление отвечает за определенный набор участков.

На каждом участке может возводиться один или несколько объектов.

Объекты характеризуются принадлежностью к определенной категории (жилой дом, мост и т.д.) и набором специфических характеристик (этажность, тип материала, ширина пролета и т.д.). Строительство каждого объекта ведется для определенного заказчика.

Персонал делится на инженерно-технический состав и рабочих. Каждый сотрудник имеет должность и может быть приписан к строительному управлению или конкретному участку.

Рабочие объединяются в бригады, возглавляемые бригадирами. Состав бригады определяет, какие рабочие входят в ту или иную бригаду.

Строительная техника приписывается строительному управлению и может быть назначена на конкретные объекты.

Для каждого объекта определяется график работ, включающий перечень видов работ, сроки их выполнения. Каждый вид работ на объекте выполняется определенной бригадой.

Для строительства объекта составляется смета, определяющая необходимые строительные материалы и их плановое количество. Фактический расход материалов также учитывается.

По результатам выполнения работ на объекте формируется отчет.

База данных должна обеспечивать хранение информации обо всех этих компонентах и их взаимосвязях, поддерживать целостность данных и предоставлять возможность эффективного извлечения информации.

**1.3 Требования к информационной системе**

Информационная система должна обеспечивать выполнение стандартных операций добавления, изменения, удаления и просмотра записей во всех таблицах базы данных. Кроме того, система должна реализовывать следующие специфические запросы для анализа и управления деятельностью строительной организации:

1. Получите перечень строительных управлений и/или участков и их руководителей.
2. Получите список специалистов инженерно-технического состава обозначенного участка или строительного управления с указанием их должностей.
3. Получите перечень объектов, возводимых указанным строительным управлением и/или участком, и графики их возведения.
4. Получите состав бригад, работавших (работающих) на строительстве указанного объекта.
5. Получите перечень строительной техники, приданной указанному строительному управлению.
6. Получите перечень строительной техники, выделенной на указанный объект либо работавшей на объекте в течение указанного периода времени.
7. Получите график и смету на строительство указанного объекта.
8. Получите отчет о сооружении указанного объекта.
9. Получите перечень объектов, возводимых в некотором строительном управлении или в целом по организации, на которых в обозначенный период времени выполнялся указанный вид строительных работ.
10. Получите перечень видов строительных работ, по которым имело место превышение сроков выполнения на указанном участке, строительном управлении или в целом по организации.
11. Получите перечень строительных материалов, по которым имело место превышение по смете на указанном участке, строительном управлении или в целом по организации.
12. Получите перечень видов строительных работ, выполненных указанной бригадой в течение обозначенного периода времени с указанием объектов, где эти работы выполнялись.
13. Получите перечень бригад, выполнявших указанный вид строительных работ в течение обозначенного периода времени с указанием объектов, где эти работы выполнялись.

**Выводы**

Анализ предметной области "Информационная система строительной организации" позволил выявить ключевые бизнес-процессы, организационную структуру и информационные потоки предприятия. Были определены основные сущности, такие как заказчики, строительные управления, участки, объекты, персонал, бригады, техника, виды работ, материалы, графики и сметы. Установлены взаимосвязи между этими сущностями, что является основой для проектирования реляционной базы данных.

Сформулированы детальные требования к базе данных, включая перечень необходимых для хранения атрибутов и связей. Также определены 13 ключевых запросов, которые информационная система должна обрабатывать для обеспечения эффективного управления строительной деятельностью, контроля ресурсов, сроков и затрат.

Полученные в ходе анализа данные и сформулированные требования создают прочную основу для последующих этапов разработки: проектирования концептуальной, логической и физической моделей базы данных, а также для разработки пользовательского интерфейса и функциональных модулей информационной системы.

# **2. Проектирование базы данных**

## **2.1 Концептуальная модель**

Концептуальное проектирование базы данных направлено на создание модели предметной области, которая не зависит от конкретной системы управления базами данных (СУБД) и особенностей программной реализации. На этом этапе определяются основные сущности, их атрибуты и взаимосвязи между ними.

Для информационной системы строительной организации были выделены следующие ключевые сущности:

* Заказчик (Clients): организации или частные лица, заказывающие строительство объектов;
* Объект строительства (Objects): строящиеся или построенные здания и сооружения (жилые дома, мосты, дороги и т.д.). Каждый объект связан с заказчиком и участком;
* Характеристика объекта (Characteristics): типовые характеристики, которые могут быть у объектов (например, «Этажность», «Тип материала», «Ширина пролета»);
* Значение характеристики объекта (Object\_Characteristics): конкретные значения характеристик для каждого объекта (например, для объекта «Жилой дом №1» характеристика «Этажность» имеет значение «16»);
* Строительное управление (Managements): структурное подразделение организации, отвечающее за ведение работ на нескольких участках;
* Участок (Sites): территориальная единица, на которой ведутся строительные работы по одному или нескольким объектам; принадлежит определенному строительному управлению;
* Персонал (Personnel): сотрудники строительной организации;
* Должность (Positions): классификатор должностей сотрудников (например, «Начальник участка», «Прораб», «Каменщик»);
* Бригада (Crews): группа рабочих, выполняющих определенные виды работ;
* Состав бригады (CrewMembers): связь между бригадами и персоналом, указывающая, какие сотрудники входят в состав конкретной бригады;
* Строительная техника (Equipment): машины и механизмы, используемые при строительстве (краны, экскаваторы и т.д.);
* Назначение техники (EquipmentAssignments): информация о закреплении техники за строительными управлениями или конкретными объектами на определенный период;
* Вид работ (WorkTypes): классификатор строительных работ (например, «Возведение фундамента», «Кирпичная кладка»);
* График работ (WorkSchedules): плановые и фактические сроки выполнения видов работ на объектах, а также плановая стоимость;
* Назначение работ бригадам (WorkAssignments): связь, указывающая, какая бригада выполняет определенный вид работ на объекте;
* Строительный материал (Materials): классификатор используемых строительных материалов;
* Смета/Расход материалов (MaterialConsumption): информация о плановом (смета) и фактическом расходе материалов на объекте;
* Пользователи (Users): учетные записи для доступа к информационной системе;
* Роли (Roles): определяют права доступа пользователей к функциям и данным системы;
* Журнал действий (Logs): запись действий пользователей в системе;
* Превышение сметы (BudgetExceeding): фиксация случаев превышения расхода материалов по сравнению со сметой (исходя из таблицы budgetexceeding в вашем SQL).

Эти сущности и их взаимосвязи (один-ко-многим, многие-ко-многим) формируют основу для понимания структуры данных. Например, одно «Строительное управление» может иметь много «Участков», а один «Участок» принадлежит только одному «Строительному управлению» (связь один-ко-многим). Один «Объект» может потребовать выполнения многих «Видов работ», и один «Вид работ» может выполняться на многих «Объектах» (связь многие-ко-многим, реализуемая через «График работ» или «Назначение работ бригадам»).

Для наглядного представления концептуальной модели часто используются ER-диаграммы (Entity-Relationship diagrams). В данном проекте для разработки концептуальной модели была использована нотация, позволяющая графически отобразить сущности, их атрибуты и связи. Разработанная концептуальная модель представлена в приложении А.

## **2.2 Логическая модель**

Логическая модель базы данных является следующим шагом после концептуального проектирования. Она преобразует концептуальную схему в структуру, понятную для реляционных СУБД, но все еще независимую от конкретной реализации. На этом этапе сущности преобразуются в таблицы, атрибуты – в столбцы таблиц, а связи реализуются с помощью первичных и внешних ключей.

При проектировании логической модели для информационной системы строительной организации была использована нотация Мартина (Crow's Foot), которая наглядно отображает таблицы, их поля и типы связей между ними (один-к-одному, один-ко-многим, многие-ко-многим) с указанием обязательности участия в связи.

Основные таблицы, разработанные на основе SQL-схемы и предметной области:

1. **Clients** (Заказчики): PK ClientID.
2. **Managements** (Строительные управления): PK ManagementID.
3. **Sites** (Участки): PK SiteID, FK ManagementID (связь с Managements).
4. **Objects** (Объекты строительства): PK ObjectID, FK ClientID, FK SiteID.
5. **Characteristics** (Характеристики объектов): PK CharacteristicID.
6. **Object\_Characteristics** (Значения характеристик объектов): PK ObjectCharacteristicID, FK ObjectID, FK CharacteristicID.
7. **Positions** (Должности): PK PositionID.
8. **Personnel** (Персонал): PK PersonnelID, FK PositionID, FK ManagementID (может быть NULL), FK SiteID (может быть NULL).
9. **Crews** (Бригады): PK CrewID.
10. **CrewMembers** (Состав бригады): PK (CrewID, PersonnelID), FK CrewID, FK PersonnelID.
11. **Equipment** (Строительная техника): PK EquipmentID.
12. **EquipmentAssignments** (Назначение техники): PK AssignmentID, FK EquipmentID, FK ObjectID (может быть NULL), FK ManagementID (может быть NULL).
13. **WorkTypes** (Виды работ): PK WorkTypeID.
14. **WorkSchedules** (Графики работ): PK ScheduleID, FK ObjectID, FK WorkTypeID.
15. **WorkAssignments** (Назначение работ бригадам): PK WorkAssignmentID, FK ObjectID, FK CrewID, FK WorkTypeID.
16. **Materials** (Строительные материалы): PK MaterialID.
17. **MaterialConsumption** (Смета/Расход материалов): PK MaterialConsumptionID, FK ObjectID, FK MaterialID.
18. **BudgetExceeding** (Превышения по смете): PK BudgetExceedingID, FK MaterialConsumptionID.

Первичные ключи (PK) уникально идентифицируют каждую запись в таблице. Внешние ключи (FK) обеспечивают ссылочную целостность, связывая таблицы между собой. Например, в таблице Sites поле ManagementID является внешним ключом, ссылающимся на первичный ключ ManagementID таблицы Managements.

Разработанная логическая модель, представленная в нотации Мартина, приведена в приложении Б.

Разработанная логическая модель базы данных на основе нотации Мартина (Crow's Foot) полностью отражает структуру данных авиастроительного предприятия, включая 22 взаимосвязанные сущности с четко определенными ключами и отношениями. Модель охватывает все ключевые аспекты: производство изделий с их категориями и характеристиками, организацию цехов и участков, управление персоналом и бригадами, а также процессы испытаний в лабораториях. Использование нотации Мартина с ее наглядными символами ("воронья лапка", прямая черта) позволило точно отобразить сложные связи между сущностями, включая обязательные и необязательные отношения. Четкая система первичных и внешних ключей обеспечивает целостность данных и правильную организацию отношений между таблицами.

## **2.3 Выбор системы управления базами данных**

Система управления базами данных (далее - СУБД) — это программный механизм, предназначенный для управления базой данных (то есть хранения, извлечения и обновления данных) эффективным, надежным и безопасным образом. СУБД обеспечивает физическую и логическую независимость данных, поддерживает транзакции, параллельный доступ и восстановление после сбоев, а также предоставляет язык запросов для взаимодействия пользователей и приложений с данными[3].

Современные СУБД обеспечивают надежное хранение информации, эффективный доступ к данным, поддержку целостности и безопасности информации. Основные функции СУБД включают управление данными во внешней памяти, управление буферами оперативной памяти, управление транзакциями, журнализацию и восстановление данных после сбоев, поддержку языков баз данных (SQL и другие). В промышленных системах наиболее распространены реляционные СУБД, основанные на табличной организации данных и использующие реляционную алгебру для обработки информации.

Популярные коммерческие СУБД включают Oracle Database, Microsoft SQL Server и IBM DB2, среди свободно распространяемых систем широко используются PostgreSQL и MySQL. При выборе СУБД для конкретного проекта учитываются такие факторы как масштаб системы, требования к производительности, необходимость обеспечения отказоустойчивости и распределенной обработки данных. Современные тенденции развития СУБД включают поддержку больших данных, облачных технологий и NoSQL-решений для работы с неструктурированной информацией. Для определения оптимальной для курсовой работы СУБД было проведено сравнение современных платформ. Сравнение СУБД представлено в таблице 1.

*Таблица 1 – Сравнение СУБД.*

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | MS SQL Server | MySQL | PostgreSQL |
| Тип лицензии | Проприетарная | Открытая (Oracle) | Открытая (BSD) |
| Разработчик | Microsoft | Oracle | Сообщество разработчиков |
| Поддержка ОС | Windows, Linux | Windows, Linux, macOS | Windows, Linux, macOS |
| Масштабируемость | Высокая | Средняя | Очень высокая |
| Производительность | Оптимизирована для больших нагрузок | Быстрая для веб-приложений | Высокая, особенно для сложных запросов |
| Поддержка JSON/NoSQL | Да (с SQL Server 2016) | Да (ограниченная) | Да (полноценная) |
| Транзакции | Полная поддержка ACID | Поддержка ACID (InnoDB) | Полная поддержка ACID |
| Репликация | Да (встроенная) | Да (основные методы) | Да (гибкие варианты) |
| Безопасность | Интеграция с Active Directory | Базовые механизмы | Расширенные механизмы (например, row-level security) |
| Стоимость | Платная (есть бесплатные версии) | Бесплатная (есть коммерческие редакции) | Полностью бесплатная |
| Использование | Корпоративные решения, аналитика | Веб-приложения, CMS | Сложные системы, аналитика, GIS |

Ключевые различия:

1.MS SQL — лучший выбор для корпоративных решений под Windows с глубокой интеграцией в экосистему Microsoft.

2.MySQL — оптимален для веб-проектов благодаря простоте и скорости, но уступает в функциональности для сложных задач.

3.PostgreSQL — наиболее продвинутая открытая СУБД с поддержкой сложных типов данных (GIS, JSON), но требует больше ресурсов для настройки.

Для реализации базы данных выбрана MySQL как надежная, производительная и экономически выгодная СУБД. Она идеально подходит для данного проекта благодаря простоте настройки, хорошей скорости работы с веб-приложениями и открытой лицензии. MySQL обеспечивает необходимую функциональность для работы с реляционными данными, поддержку транзакций и целостность информации, при этом не требуя значительных ресурсов для развертывания и сопровождения. Широкая распространенность СУБД гарантирует доступ к обширной документации и сообществу разработчиков. Выбор MySQL обусловлен оптимальным сочетанием производительности, надежности и стоимости для задач проекта.

## **2.4 Физическая модель**

Физическая модель базы данных была разработана на основе ранее созданной логической модели с использованием нотации Мартина (Crow's Foot) и реализована в СУБД MySQL. Эта модель представляет собой конкретное техническое воплощение структуры базы данных с учетом всех особенностей и ограничений выбранной системы управления. В физической модели каждая сущность была преобразована в таблицу с конкретными типами данных, соответствующими требованиям MySQL. Для всех первичных и внешних ключей были определены соответствующие ограничения целостности, обеспечивающие корректность хранимых данных.

Для улучшения производительности запросов были созданы индексы на часто используемых полях и полях, участвующих в условиях соединения таблиц. Физическая модель также включает определения представлений и хранимых процедур, реализующих бизнес-логику приложения. Все таблицы связаны между собой через механизм внешних ключей (FOREIGN KEY), что гарантирует соблюдение ссылочной целостности данных. Модель учитывает особенности хранения данных в MySQL, включая выбор подходящих движков хранения (InnoDB) и настройки кодировок (utf8mb4). Разработанная физическая модель обеспечивает эффективное хранение и обработку данных, соответствуя всем требованиям производительности и надежности для данного проекта. Реализация модели в MySQL подтвердила ее работоспособность и соответствие поставленным задачам, обеспечив стабильную работу приложения с базой данных.

Разработанная физическая модель представлена в Приложении В.

Описание основных таблиц физической модели:

1. **clients** (Заказчики): Хранит информацию о заказчиках. Первичный ключ: ClientID (INT, AUTO\_INCREMENT).
2. **managements** (Строительные управления): Хранит информацию о строительных управлениях. Первичный ключ: ManagementID (INT, AUTO\_INCREMENT).
3. **sites** (Участки): Хранит информацию об участках. Первичный ключ: SiteID (INT, AUTO\_INCREMENT). Внешний ключ: ManagementID (INT), ссылается на managements.ManagementID (ON DELETE CASCADE, ON UPDATE CASCADE).
4. **objects** (Объекты строительства): Хранит информацию об объектах строительства. Первичный ключ: ObjectID (INT, AUTO\_INCREMENT). Внешние ключи: ClientID (INT), ссылается на clients.ClientID (ON DELETE CASCADE); SiteID (INT), ссылается на sites.SiteID (ON DELETE CASCADE). Дополнительное поле: Budget (DECIMAL(15,2)) для бюджета объекта.
5. **characteristics** (Характеристики объектов): Справочник типовых характеристик объектов. Первичный ключ: CharacteristicID (INT, AUTO\_INCREMENT). Поле: CharacteristicName (VARCHAR(100)), DataType (ENUM).
6. **object\_characteristics** (Значения характеристик объектов): Связывает объекты с их характеристиками и хранит значения. Первичный ключ: ObjectCharacteristicID (INT, AUTO\_INCREMENT). Внешние ключи: ObjectID (INT), ссылается на objects.ObjectID (ON DELETE CASCADE); CharacteristicID (INT), ссылается на characteristics.CharacteristicID (ON DELETE CASCADE). Поле: Value (VARCHAR(255)). Создан индекс idx\_object\_characteristics\_objectid по полю ObjectID.
7. **positions** (Должности): Справочник должностей персонала. Первичный ключ: PositionID (INT, AUTO\_INCREMENT). Поле: PositionName (VARCHAR(100)).
8. **personnel** (Персонал): Хранит информацию о сотрудниках. Первичный ключ: PersonnelID (INT, AUTO\_INCREMENT). Внешние ключи: PositionID (INT), ссылается на positions.PositionID (ON DELETE RESTRICT); ManagementID (INT, NULLABLE), ссылается на managements.ManagementID (ON DELETE SET NULL); SiteID (INT, NULLABLE), ссылается на sites.SiteID (ON DELETE SET NULL). Поле: FullName (VARCHAR(255)).
9. **crews** (Бригады): Хранит информацию о рабочих бригадах. Первичный ключ: CrewID (INT, AUTO\_INCREMENT). Поле: CrewName (VARCHAR(100)).
10. **crewmembers** (Состав бригады): Связывает бригады и персонал. Составной первичный ключ: (CrewID, PersonnelID). Внешние ключи: CrewID (INT), ссылается на crews.CrewID (ON DELETE CASCADE); PersonnelID (INT), ссылается на personnel.PersonnelID (ON DELETE CASCADE).
11. **equipment** (Строительная техника): Справочник строительной техники. Первичный ключ: EquipmentID (INT, AUTO\_INCREMENT). Поле: EquipmentName (VARCHAR(100)).
12. **equipmentassignments** (Назначение техники): Информация о назначении техники. Первичный ключ: AssignmentID (INT, AUTO\_INCREMENT). Внешние ключи: EquipmentID (INT), ссылается на equipment.EquipmentID (ON DELETE CASCADE); ObjectID (INT, NULLABLE), ссылается на objects.ObjectID (ON DELETE CASCADE); ManagementID (INT, NULLABLE), ссылается на managements.ManagementID (ON DELETE CASCADE). Поля: StartDate (DATE), EndDate (DATE, NULLABLE).
13. **worktypes** (Виды работ): Справочник видов строительных работ. Первичный ключ: WorkTypeID (INT, AUTO\_INCREMENT). Поле: WorkTypeName (VARCHAR(100)).
14. **workschedules** (Графики работ): Плановые и фактические сроки выполнения работ на объектах. Первичный ключ: ScheduleID (INT, AUTO\_INCREMENT). Внешние ключи: ObjectID (INT), ссылается на objects.ObjectID (ON DELETE CASCADE); WorkTypeID (INT), ссылается на worktypes.WorkTypeID (ON DELETE RESTRICT). Поля: StartDate (DATE), EndDate (DATE), PlannedCost (DECIMAL(12,2)), ActualEndDate (DATE, NULLABLE).
15. **workassignments** (Назначение работ бригадам): Связывает объекты, бригады и виды работ. Первичный ключ: WorkAssignmentID (INT, AUTO\_INCREMENT). Внешние ключи: ObjectID (INT), ссылается на objects.ObjectID (ON DELETE CASCADE); CrewID (INT), ссылается на crews.CrewID (ON DELETE CASCADE); WorkTypeID (INT), ссылается на worktypes.WorkTypeID (ON DELETE RESTRICT). Поле: AssignmentDate (DATE).
16. **materials** (Строительные материалы): Справочник строительных материалов. Первичный ключ: MaterialID (INT, AUTO\_INCREMENT). Поле: MaterialName (VARCHAR(100)).
17. **materialconsumption** (Смета/Расход материалов): Информация о плановом и фактическом расходе материалов на объекте. Первичный ключ: MaterialConsumptionID (INT, AUTO\_INCREMENT). Внешние ключи: ObjectID (INT), ссылается на objects.ObjectID (ON DELETE CASCADE); MaterialID (INT), ссылается на materials.MaterialID (ON DELETE RESTRICT). Поля: Quantity (DECIMAL(10,2)), PlannedQuantity (DECIMAL(10,2)).
18. **budgetexceeding** (Превышения по смете): Фиксация превышений расхода материалов. Первичный ключ: BudgetExceedingID (INT, AUTO\_INCREMENT). Внешний ключ: MaterialConsumptionID (INT), ссылается на materialconsumption.MaterialConsumptionID (ON DELETE CASCADE). Поле: ExceededAmount (DECIMAL(10,2)).
19. **users** (Пользователи): Учетные записи пользователей системы. Первичный ключ: UserID (INT, AUTO\_INCREMENT). Внешний ключ: RoleID (INT), ссылается на roles.RoleID (ON DELETE RESTRICT). Поля: Username (VARCHAR(50), UNIQUE), PasswordHash (VARCHAR(255)), Balance (INT).
20. **roles** (Роли): Справочник ролей пользователей. Первичный ключ: RoleID (INT, AUTO\_INCREMENT). Поле: RoleName (VARCHAR(50), UNIQUE).
21. **rolepermissions** (Разрешения ролей): Связывает роли с разрешениями на таблицы. Составной первичный ключ: (RoleID, TableName, PermissionType). Внешний ключ: RoleID (INT), ссылается на roles.RoleID (ON DELETE CASCADE). Поля: TableName (VARCHAR(50)), PermissionType (ENUM).
22. **logs** (Журнал действий): Запись действий пользователей. Первичный ключ: LogID (INT, AUTO\_INCREMENT). Внешний ключ: UserID (INT), ссылается на users.UserID (ON DELETE CASCADE). Поля: Action (VARCHAR(255)), ActionTime (DATETIME).

Физическая модель базы данных представляет собой полную техническую спецификацию структуры хранения данных, включая таблицы, их атрибуты, типы данных, первичные и внешние ключи, индексы и ограничения целостности. Модель точно отражает все сущности предметной области и взаимосвязи между ними, обеспечивая корректное представление данных в реляционной форме. Она учитывает требования производительности, целостности данных и будущего масштабирования системы, являясь основой для физической реализации в выбранной СУБД. Разработанная модель полностью соответствует исходной логической схеме и готова к преобразованию в реальную базу данных.

**2.5 Нормализация базы данных**

Нормализация – это метод проектирования реляционных баз данных, направленный на устранение избыточности данных путем декомпозиции отношений (таблиц) на более простые структуры. Основные нормальные формы (НФ) включают:

* **Первая нормальная форма (1НФ):** Требует атомарности значений атрибутов (каждый столбец содержит неделимые значения) и отсутствия повторяющихся групп атрибутов в строках.
* **Вторая нормальная форма (2НФ):** Требует выполнения 1НФ и отсутствия частичных зависимостей неключевых атрибутов от составного первичного ключа. То есть каждый неключевой атрибут должен зависеть от *всего* первичного ключа, а не только от его части.
* **Третья нормальная форма (3НФ):** Требует выполнения 2НФ и отсутствия транзитивных зависимостей неключевых атрибутов. То есть неключевой атрибут не должен зависеть от другого неключевого атрибута.

Нормализация до 3НФ в большинстве случаев достаточна для практических приложений, обеспечивая баланс между целостностью данных и производительностью.

Процесс нормализации в разработанной базе данных для строительной организации обеспечивает оптимальную организацию информации, устраняя избыточность и поддерживая целостность данных. В нашей системе все таблицы приведены как минимум к третьей нормальной форме (3НФ), что подтверждается анализом ключевых сущностей и их взаимосвязей.

Рассмотрим, например, таблицу **personnel** (Персонал), которая хранит информацию о сотрудниках:

* PersonnelID (Первичный ключ)
* FullName
* PositionID (Внешний ключ к таблице **positions**)
* ManagementID (Внешний ключ к таблице **managements**, может быть NULL)
* SiteID (Внешний ключ к таблице **sites**, может быть NULL)

1. **1НФ:** Все атрибуты (PersonnelID, FullName, PositionID, ManagementID, SiteID) являются атомарными. Каждая строка уникально идентифицируется PersonnelID.
2. **2НФ:** Первичный ключ PersonnelID является простым (состоит из одного атрибута). Следовательно, частичные зависимости невозможны. Все неключевые атрибуты (FullName, PositionID, ManagementID, SiteID) полностью зависят от PersonnelID.
3. **3НФ:** Отсутствуют транзитивные зависимости. Например, информация о должности (PositionName), строительном управлении (ManagementName) или участке (SiteName) не хранится напрямую в таблице personnel, а вынесена в отдельные справочные таблицы (positions, managements, sites) и связана через внешние ключи (PositionID, ManagementID, SiteID). Это исключает избыточность (например, многократное хранение названия должности для разных сотрудников) и аномалии обновления.

Аналогичный анализ был проведен для всех таблиц базы данных, подтверждая их соответствие требованиям 3НФ. Например, таблица **materialconsumption** (Смета/Расход материалов) связывает объекты (ObjectID) и материалы (MaterialID) с количеством (Quantity, PlannedQuantity). Информация о названии объекта или материала хранится в их соответствующих справочных таблицах (objects, materials), избегая избыточности в materialconsumption.

Разработанная база данных соответствует требованиям третьей нормальной формы (3НФ), что обеспечивает оптимальную структуру хранения данных без избыточности и аномалий. Все таблицы удовлетворяют условиям нормализации, гарантируя целостность данных при операциях добавления, изменения и удаления. Нормализованная структура обеспечивает баланс между эффективностью хранения информации и производительностью запросов, полностью соответствуя потребностям предметной области.

## **2.6 Разграничение прав доступа**

MySQL предоставляет сложную систему привилегий, которая позволяет администраторам точно контролировать, кто может получить доступ к каким данным и какие операции может выполнять. Этот механизм реализуется через многоуровневую систему безопасности, охватывающую сервер, базы данных, таблицы и отдельные столбцы.

Основу системы составляют две ключевые команды: GRANT для предоставления прав и REVOKE для их отзыва.

Команда GRANT позволяет назначать конкретные привилегии пользователям для работы с определенными объектами базы данных. Например, можно предоставить права на чтение (SELECT) определенной таблицы или разрешить полное управление (ALL PRIVILEGES) всей базой данных. Важной особенностью является возможность указания хоста, с которого пользователь может подключаться, что дополнительно усиливает безопасность.

Обратная операция - REVOKE - используется для отзыва ранее выданных прав. Это особенно важно при изменении ролей сотрудников или обнаружении потенциальных угроз безопасности. MySQL поддерживает различные типы привилегий, включая данные (SELECT, INSERT, UPDATE, DELETE), структурные (CREATE, ALTER, DROP) и административные (GRANT OPTION, PROCESS).

Особенностью системы является иерархическая структура прав, где глобальные привилегии действуют на уровне всего сервера, а более специфические - на уровне баз данных, таблиц или столбцов. Дополнительную гибкость обеспечивают роли (с MySQL 8.0), позволяющие группировать наборы привилегий для разных категорий пользователей.

Запросы на создание пользователей представлены в Приложении Г. (Согласно ГОСТ 7.32, ссылки на приложения оформляются в тексте в круглых скобках, например: (Приложение Г)).

Для информационной системы были созданы следующие пользователи с соответствующими правами:

1. **admin**: Пользователь с полными административными правами (ALL PRIVILEGES) на все объекты базы данных. Используется для управления базой данных, создания/удаления таблиц, пользователей и выполнения любых операций.
2. **writer**: Пользователь с правами на чтение, добавление, изменение и удаление данных (SELECT, INSERT, UPDATE, DELETE), а также правами на работу с файлами (FILE) и выполнение хранимых процедур (EXECUTE). Предназначен для пользователей, осуществляющих ввод и корректировку данных.
3. **user**: Пользователь с минимальными правами, достаточными только для просмотра данных (SELECT) и выполнения хранимых процедур (EXECUTE) для получения отчетов и результатов запросов. Предназначен для конечных пользователей, которым нужен только доступ к информации.

Такое разграничение прав доступа обеспечивает необходимый уровень безопасности, гарантируя, что каждый пользователь имеет доступ только к тем данным и функциям, которые соответствуют его роли в системе.

**Выводы**

В ходе проектирования базы данных была последовательно разработана концептуальная, логическая и физическая модели, полностью отражающие структуру данных строительной организации. Концептуальная модель выявила ключевые сущности предметной области (заказчики, объекты, персонал, бригады, техника, работы, материалы и т.д.) и их взаимосвязи. Логическая модель детализировала их в виде таблиц с атрибутами, первичными и внешними ключами, определяя реляционную структуру. Физическая модель реализовала эту структуру с учетом особенностей выбранной СУБД MySQL, определив типы данных, индексы, представления и хранимые процедуры.

Все таблицы нормализованы до третьей нормальной формы, что исключает избыточность данных и обеспечивает их целостность, минимизируя аномалии при операциях с данными. Разработанная система разграничения прав доступа гарантирует безопасность информации, предоставляя пользователям различные уровни доступа в соответствии с их ролями (admin, writer, user).

В результате создана оптимальная структура базы данных, удовлетворяющая всем требованиям предметной области "Информационная система строительной организации", готовая к реализации и дальнейшей разработке клиентского приложения.

# **3 Технология разработки базы данных и веб-приложения**

## **3.1 Реализация базы данных**

Для программной реализации базы данных в рамках данного проекта было выбрано интегрированное решение OpenServer, предоставляющее готовую среду для развертывания и администрирования СУБД MySQL. Данная платформа позволяет локально эмулировать серверную инфраструктуру, обеспечивая стабильную работу с реляционными базами данных без необходимости сложной настройки окружения. В процессе разработки были реализованы SQL-запросы для создания и модификации таблиц, настроены ограничения целостности данных, включая первичные и внешние ключи, а также разработаны процедуры для автоматизации критически важных операций[5].

Использование OpenServer значительно ускорило процесс развертывания тестовой среды и отладки запросов, что особенно ценно на этапе проектирования структуры базы данных. Встроенные инструменты администрирования панели позволили оперативно мониторить выполнение запросов и оптимизировать производительность базы данных.

Кроме того, для удобного визуального управления базой данных в рамках OpenServer была задействована веб-интерфейсная система phpMyAdmin, которая предоставляет интуитивно понятный инструментарий для администрирования MySQL. Интеграция phpMyAdmin в OpenServer осуществляется через встроенную панель управления, позволяя одним кликом запускать веб-интерфейс без необходимости ручной настройки сервера или подключения к внешним ресурсам.

Использование phpMyAdmin в сочетании с OpenServer значительно упростило процесс отладки и тестирования базы данных. Это особенно полезно при работе со сложными запросами, представлениями и хранимыми процедурами, где визуализация структуры данных помогает избежать ошибок проектирования. Встроенные в phpMyAdmin средства оптимизации запросов дополняли функционал OpenServer, обеспечивая комплексный подход к управлению базой данных на всех этапах разработки.

Код реализации запросов для создания таблиц с учётом внешних ключей и ограничений представлен в *Приложении Д.*

Для реализации функциональности работы с данными в информационной системе были разработаны хранимые процедуры, обеспечивающие выполнение стандартных операций, таких как добавление, изменение, удаление и выборка данных. Процедуры инкапсулируют сложную бизнес-логику, что позволяет упростить взаимодействие с базой данных из клиентского приложения и минимизировать риски возникновения ошибок. Входные параметры процедур включают в себя идентификаторы записей, значения полей и другие переменные, необходимые для выполнения операций.

Полные тексты SQL-запросов на создание процедур приведены в *Приложении Е*.

1) Синтаксис создания таблиц:

CREATE TABLE TableName (

column1 datatype constraints,

column2 datatype constraints,

...

PRIMARY KEY (column1)

);

TableName – имя таблицы.

column1, column2, ... – названия столбцов.

datatype – тип данных столбца.

constraints – любые ограничения, такие как NOT NULL, UNIQUE, DEFAULT и т.д.

Чтобы интегрировать внешние ключи в таблицу с помощью SQL-кода,

нужно написать следующее;

ALTER TABLE TableName

ADD CONSTRAINT FK\_constraintName

FOREIGN KEY (columnName)

REFERENCES referencedTableName(referencedColumnName)

ON DELETE CASCADE ON UPDATE CASCADE;

TableName – имя таблицы, к которой добавляется внешний ключ.

FK\_constraintName – имя ограничения (может быть любым уникальным).

columnName – имя столбца, который является внешним ключом.

referencedTableName – имя таблицы, на которую ссылается внешний ключ.

referencedColumnName – имя столбца в таблице, на которую ссылается внешний ключ.

ON DELETE CASCADE ON OPDATE CASCADE – опциональная часть, определяющая действие при удалении связанной записи.

2)Синтаксис создания представлений:

CREATE VIEW ViewName AS

SELECT column1, column2, ...

FROM TableName

[WHERE condition]

[GROUP BY column1, column2, ...]

[HAVING condition]

[ORDER BY column1 [ASC|DESC], ...]

[WITH CHECK OPTION];

Описание параметров:

ViewName – имя создаваемого представления.

SELECT column1, column2, ... – список столбцов, включаемых в представление. Могут использоваться выражения, агрегатные функции (COUNT, SUM, AVG и др.) и соединения таблиц (JOIN).

FROM TableName – таблица или запрос, на основе которого строится представление.

WHERE condition (необязательно) – условие фильтрации данных.

GROUP BY column1, column2, ... (необязательно) – группировка данных по указанным столбцам.

HAVING condition (необязательно) – фильтрация результатов после группировки.

ORDER BY column1 [ASC|DESC], ... (необязательно) – сортировка данных.

WITH CHECK OPTION (необязательно) – гарантирует, что данные, вставляемые или обновляемые через представление, соответствуют условию WHERE.

3)Синтаксис создания процедур:

DELIMITER *//*

CREATE PROCEDURE ProcedureName(

[IN|OUT|INOUT] parameter1 datatype,

[IN|OUT|INOUT] parameter2 datatype,

...

)

BEGIN

*-- SQL-операторы*

SELECT ...;

INSERT ...;

UPDATE ...;

DELETE ...;

END *//*

DELIMITER ;

Описание параметров:

ProcedureName - имя создаваемой хранимой процедуры.

IN|OUT|INOUT - направление параметра:

IN - входной параметр (по умолчанию)

OUT - выходной параметр

INOUT - входной и выходной параметр

parameter1, parameter2, ... - имена параметров процедуры.

datatype - тип данных параметра (INT, VARCHAR, DATE и др.).

BEGIN ... END - блок, содержащий SQL-операторы процедуры.

Для администрирования и визуального управления базой данных использовался phpMyAdmin — веб-приложение, входящее в состав OpenServer и предоставляющее удобный графический интерфейс для работы с MySQL. Этот инструмент позволил упростить выполнение SQL-запросов, мониторинг структуры таблиц, а также настройку индексов и связей между данными.

## **3.2 Разработка веб-приложения к базе данных**

Для взаимодействия базы данных и пользователя было разработано веб-приложение с использованием платформы ASP.NET Core на языке C#. ASP.NET Core является кроссплатформенным, высокопроизводительным фреймворком с открытым исходным кодом для создания современных облачных интернет-приложений. Он позволяет разрабатывать веб-приложения, веб-API и микросервисы.

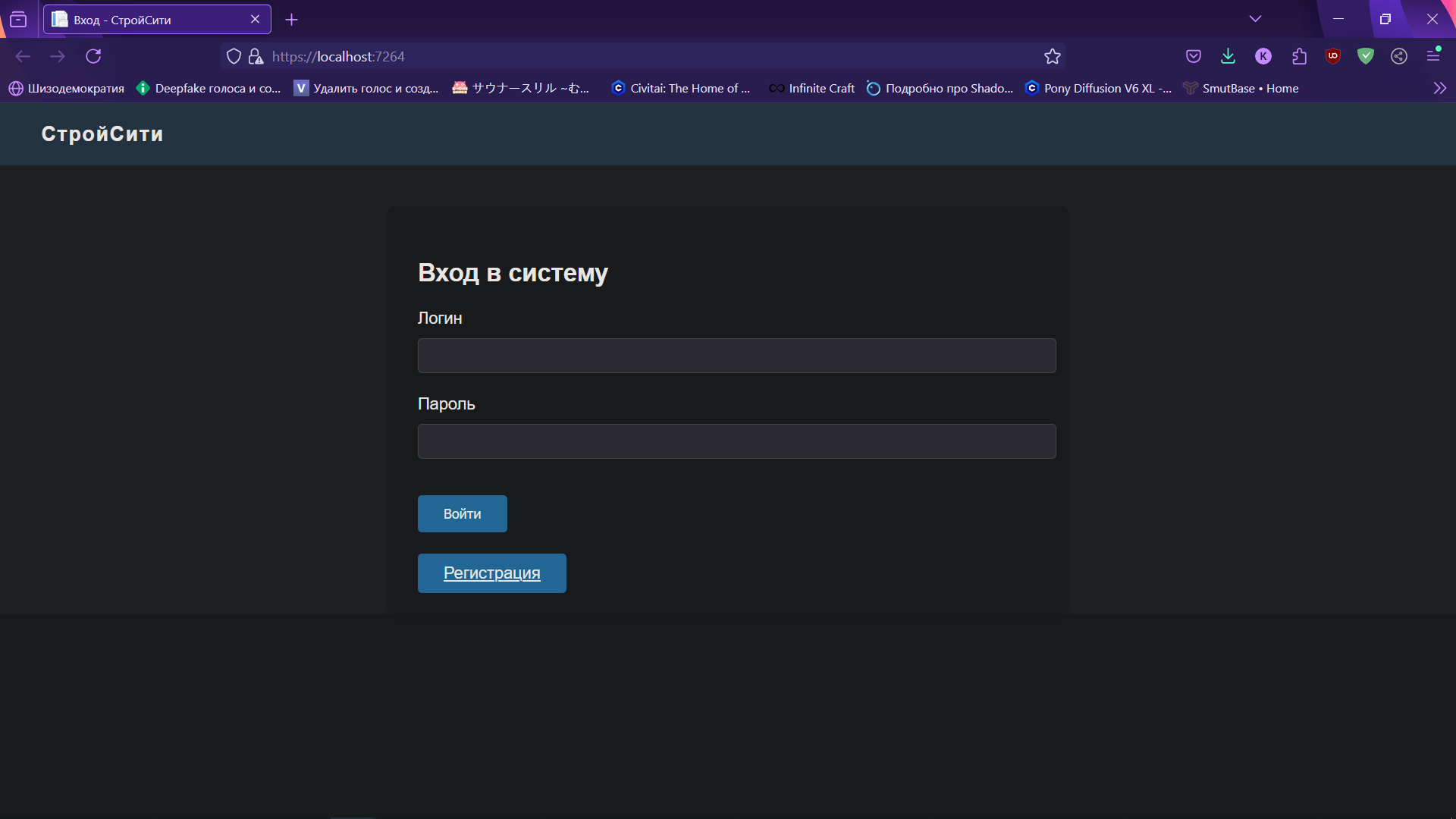
Выбор ASP.NET Core обусловлен его гибкостью, производительностью и широкими возможностями для создания масштабируемых веб-приложений. Использование C# в качестве основного языка программирования обеспечивает строгую типизацию, безопасность и доступ к обширной экосистеме .NET. ASP.NET Core поддерживает различные архитектурные шаблоны, такие как Model-View-Controller (MVC) и Razor Pages, что позволяет структурировать код и упростить разработку.

Разработанное веб-приложение предоставляет пользовательский интерфейс для взаимодействия с базой данных строительной организации. Приложение реализует логику авторизации, разграничения прав доступа в соответствии с ролями пользователей и выполнения операций с данными.

Код приложения представлен в Приложении И.

Структура веб-приложения:

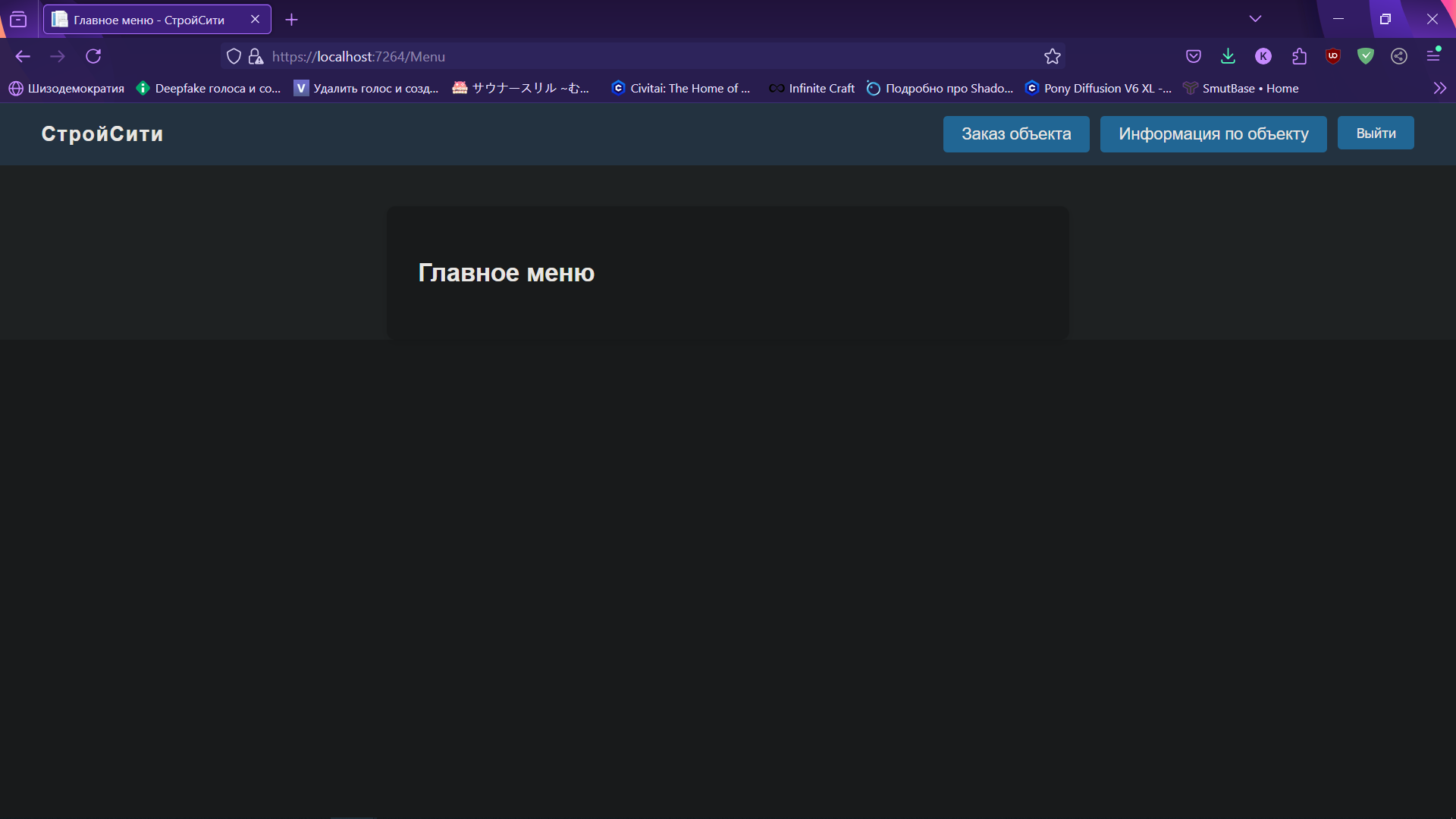
При доступе к веб-приложению пользователю отображается страница авторизации, изображенная на рисунке 1.



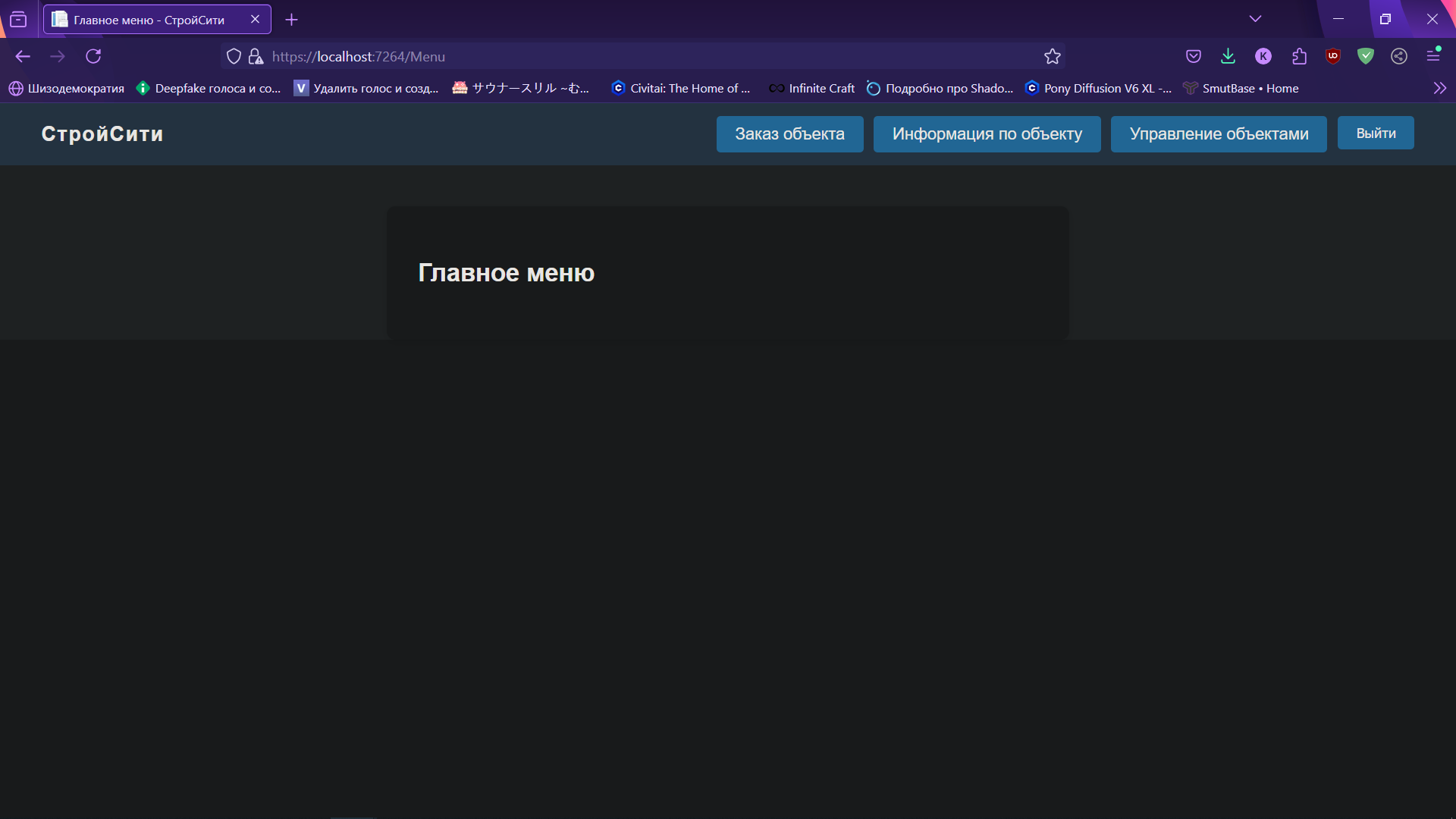
*Рис.1 – вход в систему*

Она содержит поля для ввода имени пользователя и пароля. Страница включает элементы управления для подтверждения ввода данных и регистрации пользователя(по умолчанию новый пользователь - клиент), администратор базы данных может назначить определенного пользователя админом или же менеджером(writer).

После успешной авторизации пользователь перенаправляется на веб-страницу, соответствующую его ролевым правам (admin, writer, user), что отображено на рисунках 2-3.



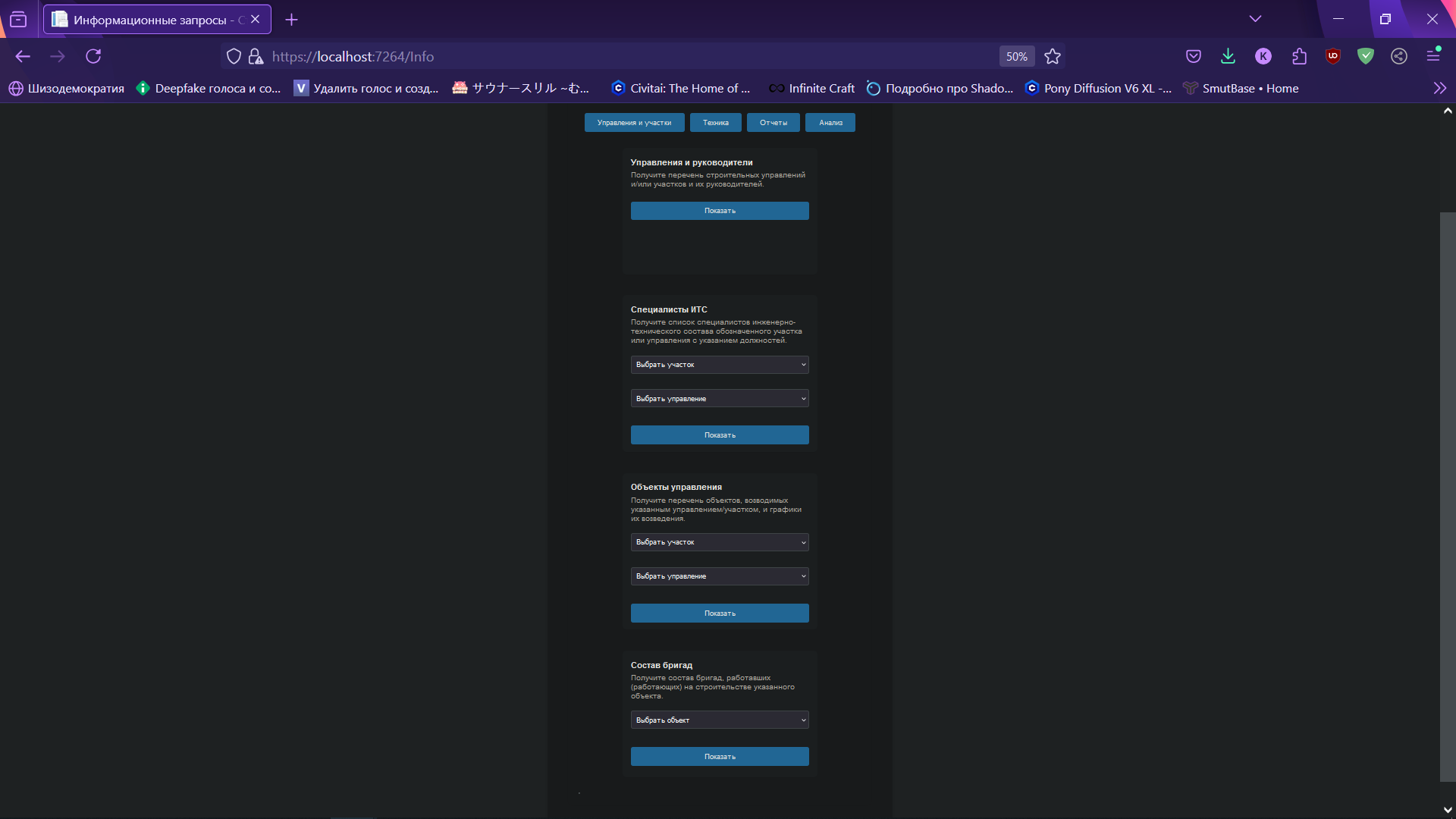
*Рисунок 2 – Панель пользователя – Строительная организация.*



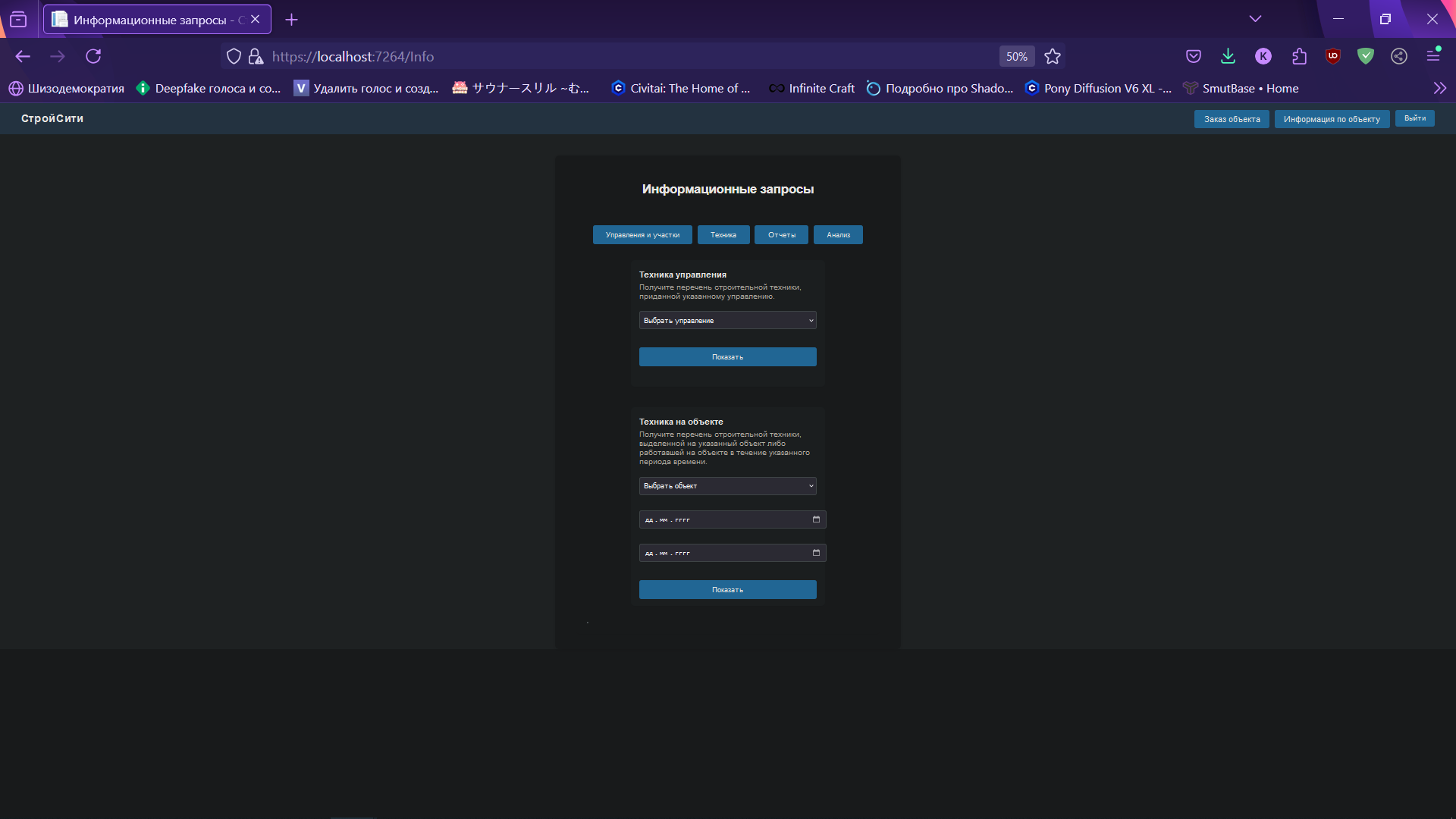
*Рисунок 3 – Панель администратора/менеджера – Строительная организация.*

Для роли client доступен минимальный функционал: просмотр информационных запросов и заказ объекта. Роль admin как и роль manager предоставляет полный доступ: все элементы для клиента, но и дополнительно появляется админ-панель. Общие для всех ролей возможности включают переключение между представлениями данных различных таблиц и выполнение предопределенных запросов. Интерфейсы адаптированы под уровень доступа каждой роли, что обеспечивает безопасность и удобство работы с данными.

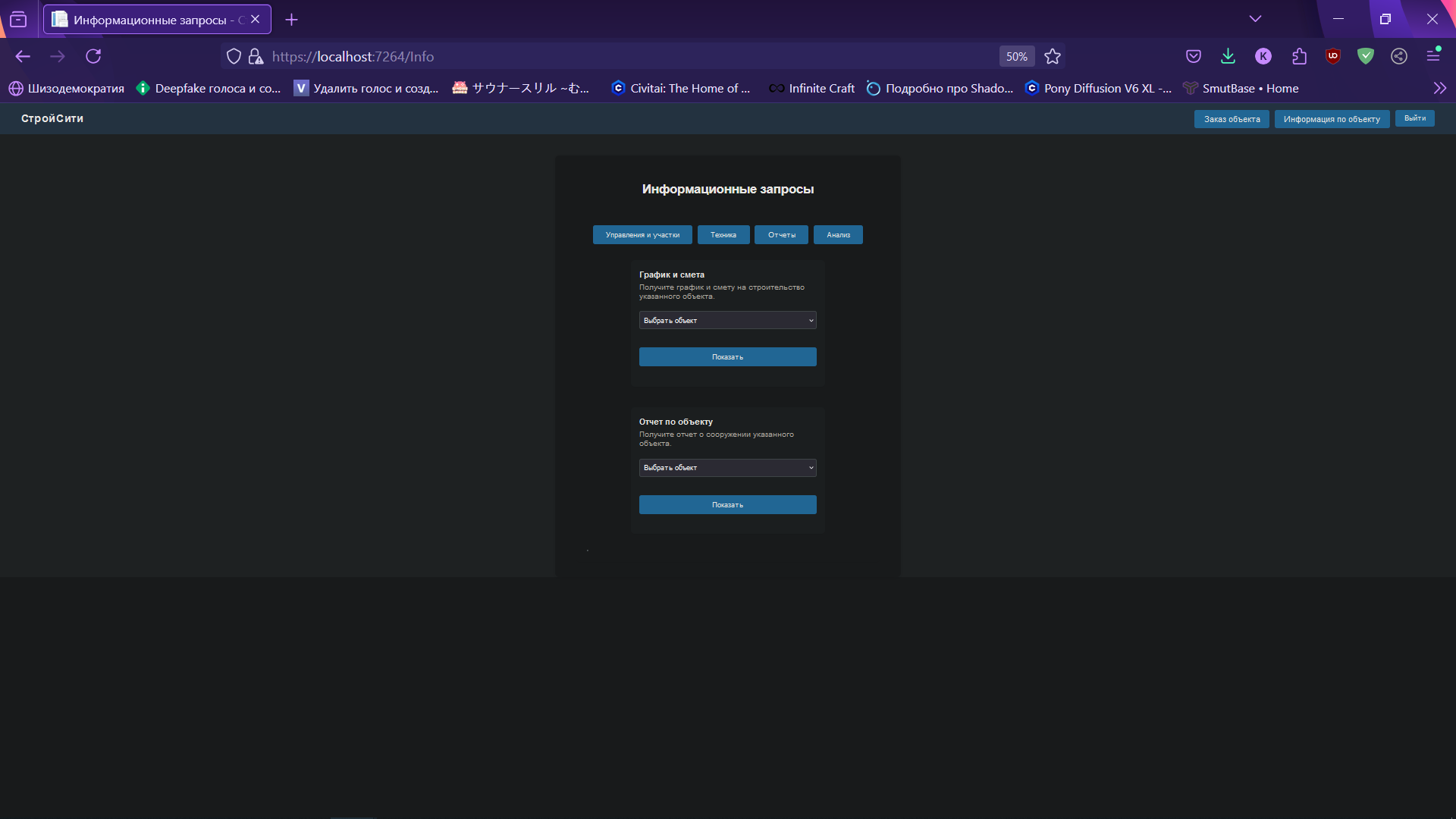
В приложении реализован интерфейс для выполнения различных запросов к базе данных. Форма запросов представлена на рисунке 5-8.



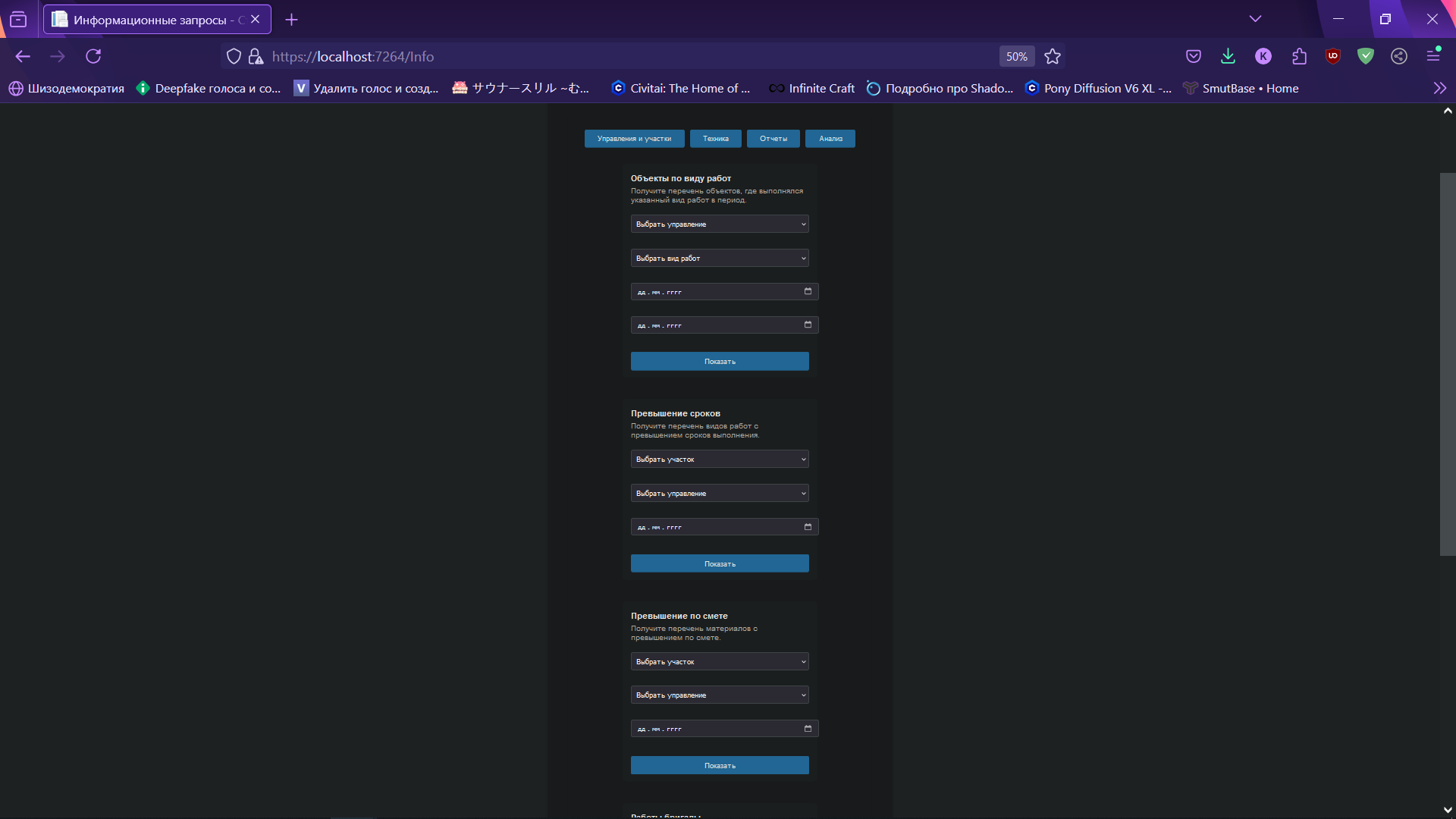
*Рисунок 5 – Форма выполнения запросов (управления и участки).*



*Рисунок 6 – Форма выполнения запросов (техника).*



*Рисунок 7 – Форма выполнения запросов (отчеты).*



*Рисунок 8 – Форма выполнения запросов (анализ).*

Данная форма предоставляет возможность выбора типа запроса из доступного списка, после чего пользователь может указать необходимые параметры для его выполнения. После заполнения параметров и подтверждения запроса система обрабатывает введенные данные и отображает результат выполнения на веб-странице. Такая реализация обеспечивает гибкость работы с базой данных, позволяя пользователям с разным уровнем доступа получать нужную информацию без необходимости написания SQL-запросов вручную.

## **3.3 Тестирование работы веб-приложения и базы данных**

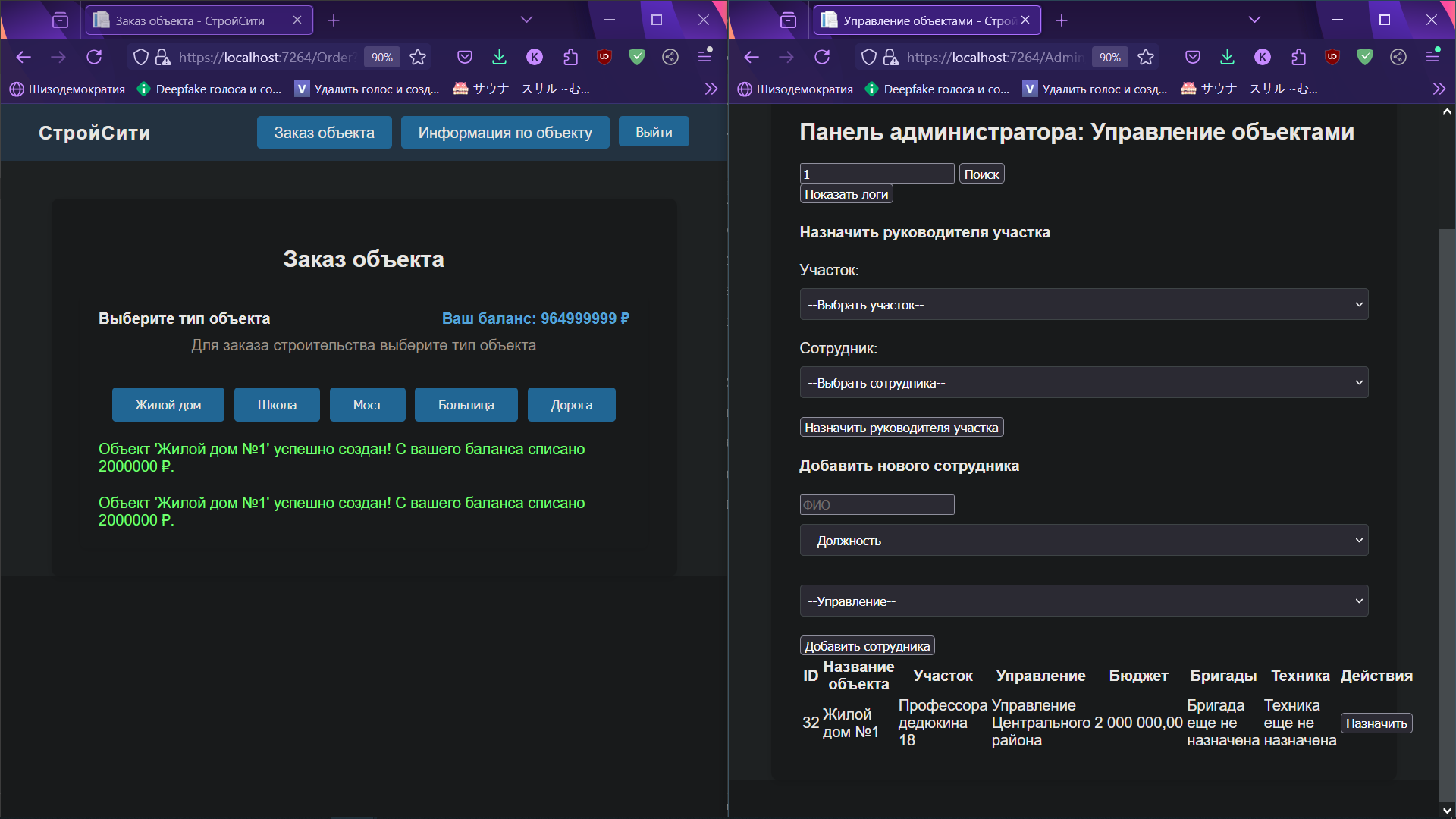
Тестирование приложения является важнейшим этапом разработки, позволяющим убедиться в корректности работы всех его компонентов и их взаимодействия. В процессе тестирования необходимо провести комплексную проверку функциональности приложения, начиная с базовых операций и заканчивая сложными сценариями работы.

Важно проверить взаимодействие между приложением и базой данных, убедившись, что все запросы выполняются правильно, данные сохраняются и извлекаются без ошибок, а транзакции обрабатываются в соответствии с ожиданиями. Тестирование должно охватывать различные роли пользователей, проверяя, что уровень доступа соответствует заданным правам, и каждая роль может выполнять только разрешенные операции. Необходимо проверить работу интерфейса, включая формы ввода данных, отображение результатов запросов и обработку ошибок.

Только после тщательной проверки всех аспектов работы приложения можно быть уверенным в его стабильности, надежности и готовности к использованию в реальных условиях.

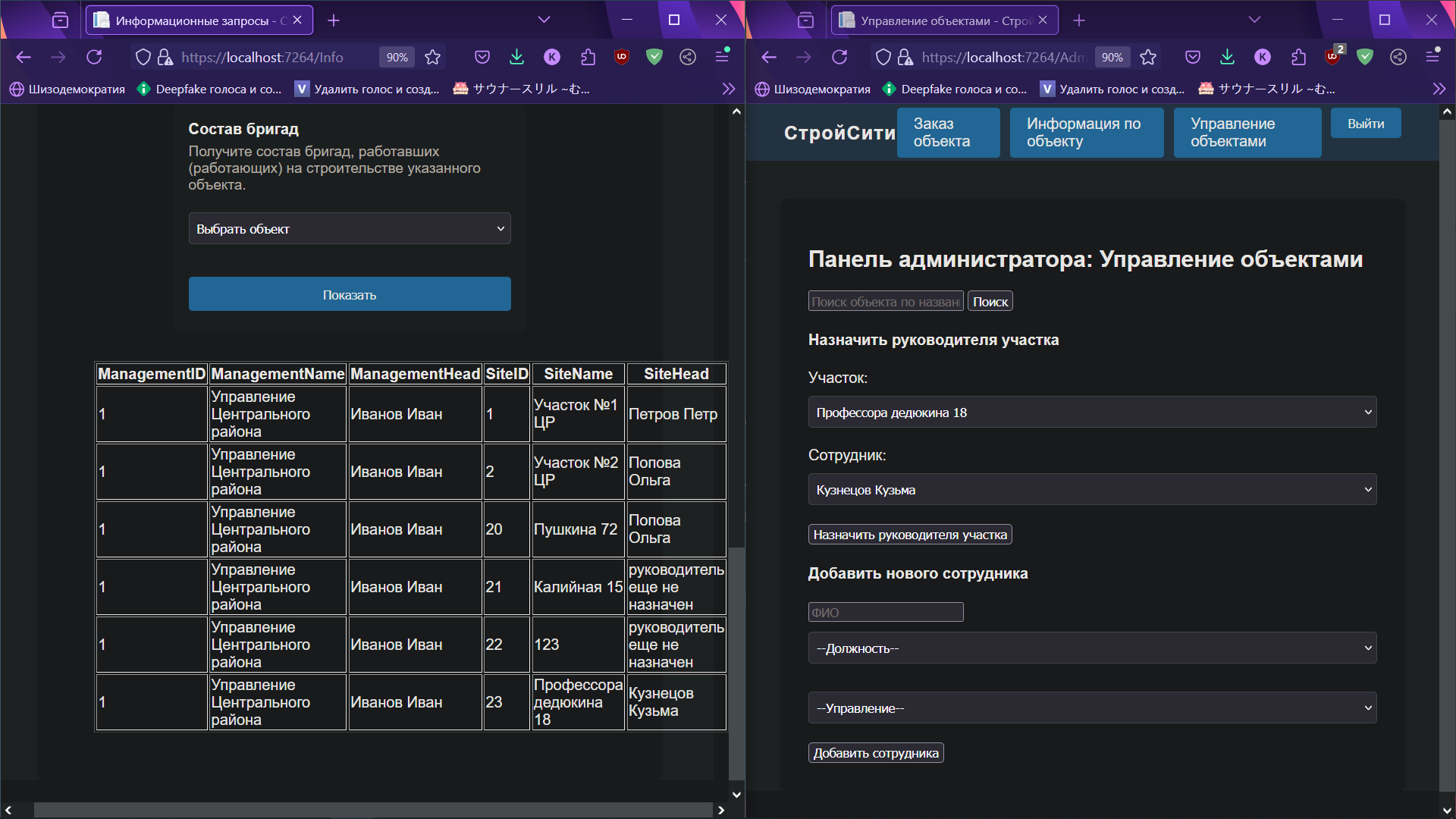
1)Добавление записи в приложении

Добавим один объект в таблицу, пусть это будет жилой дом, на рисунке 6. Показано что объект успешно создан и в админ панели он был добавлен для назначения техники, бригад, установления робот и их графиков

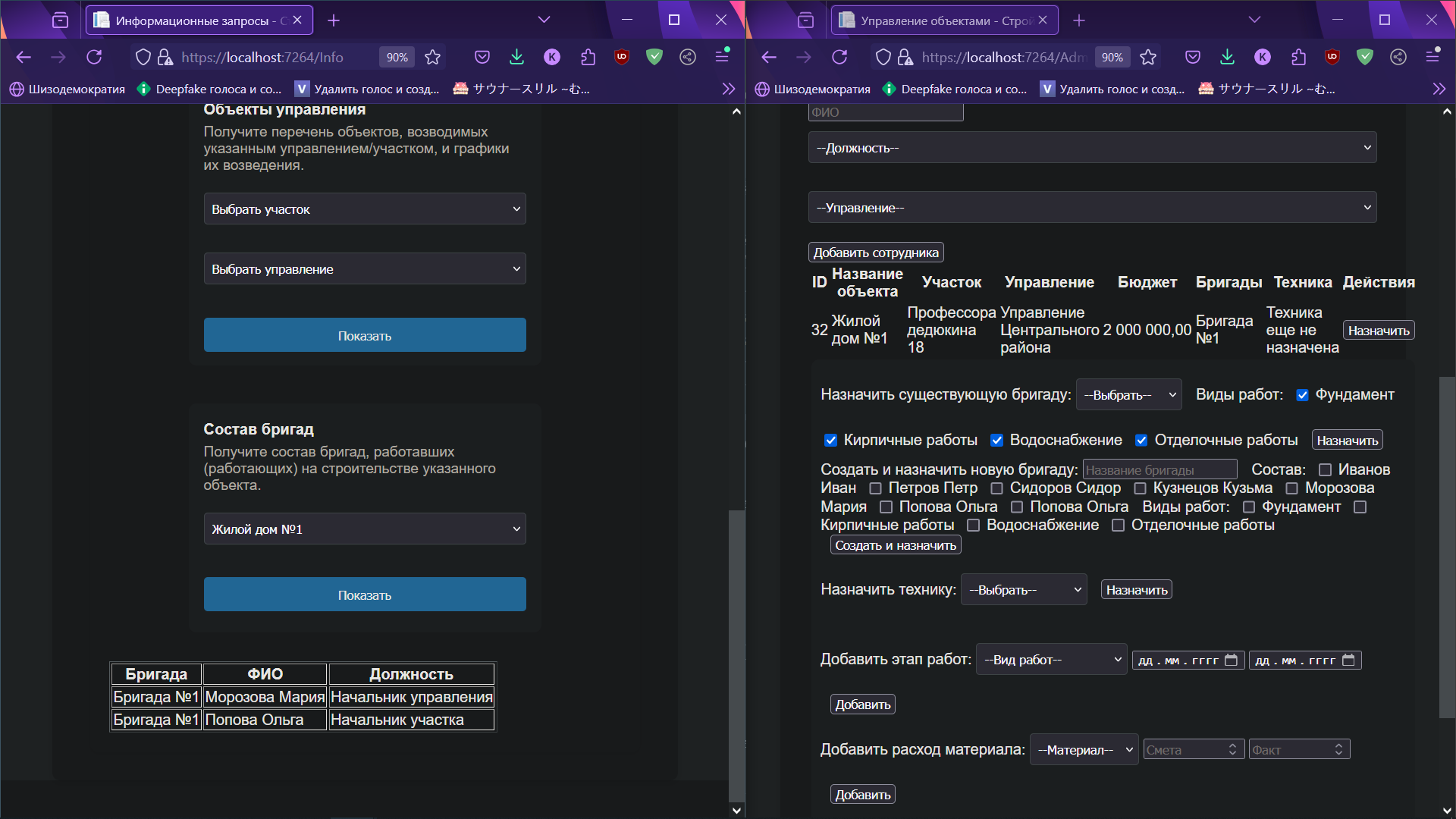


*Рисунок 6 – жилой дом успешно добавлен и готов к дальнейшей работе.*

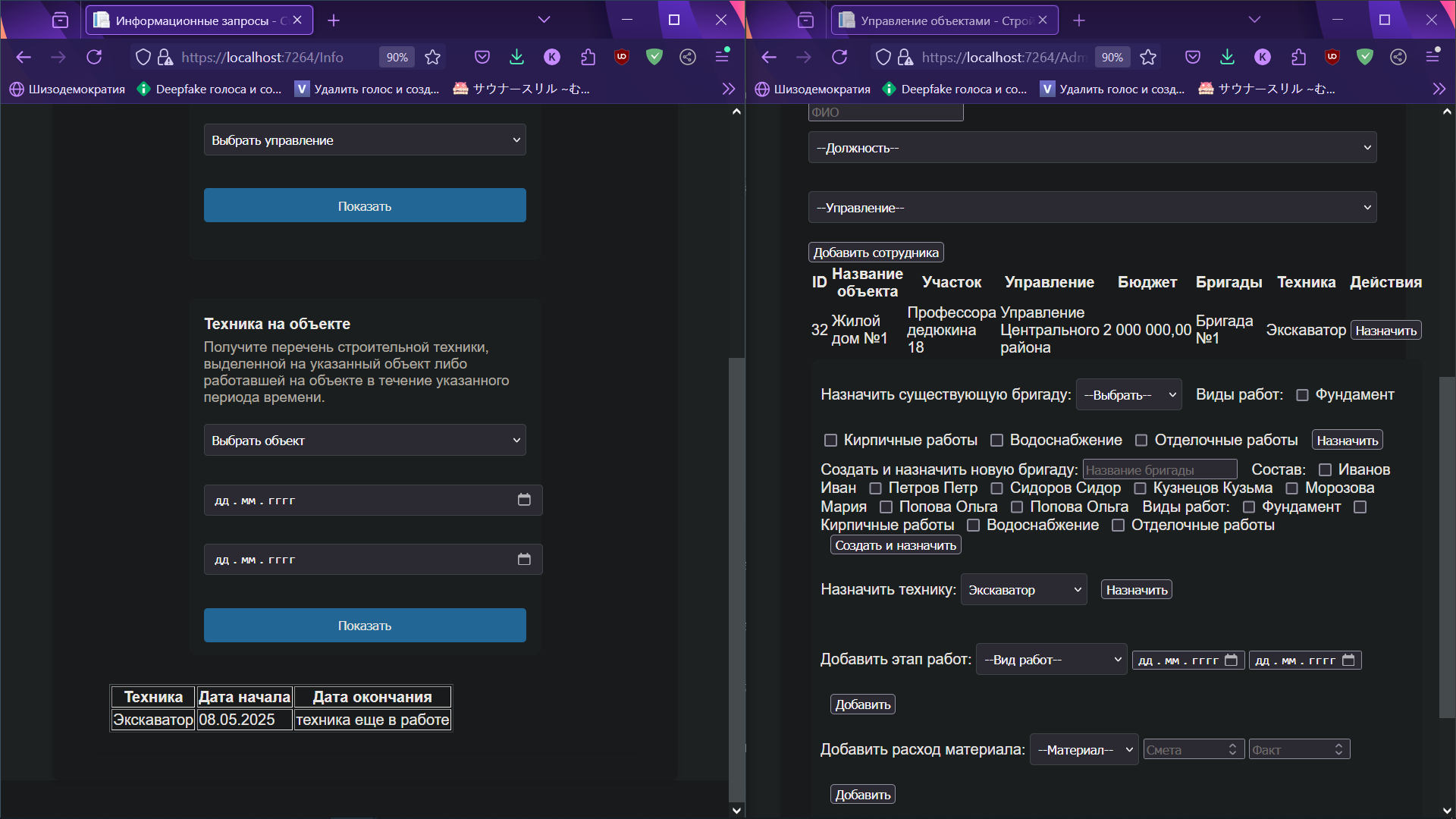
Теперь добавим начальника участка из существующего персонала, назначим технику на объект, назначим бригаду на объект, и определим работ, которые будут проводиться на объекте. Подтверждение перечисленных действий на рисунке 7-10.



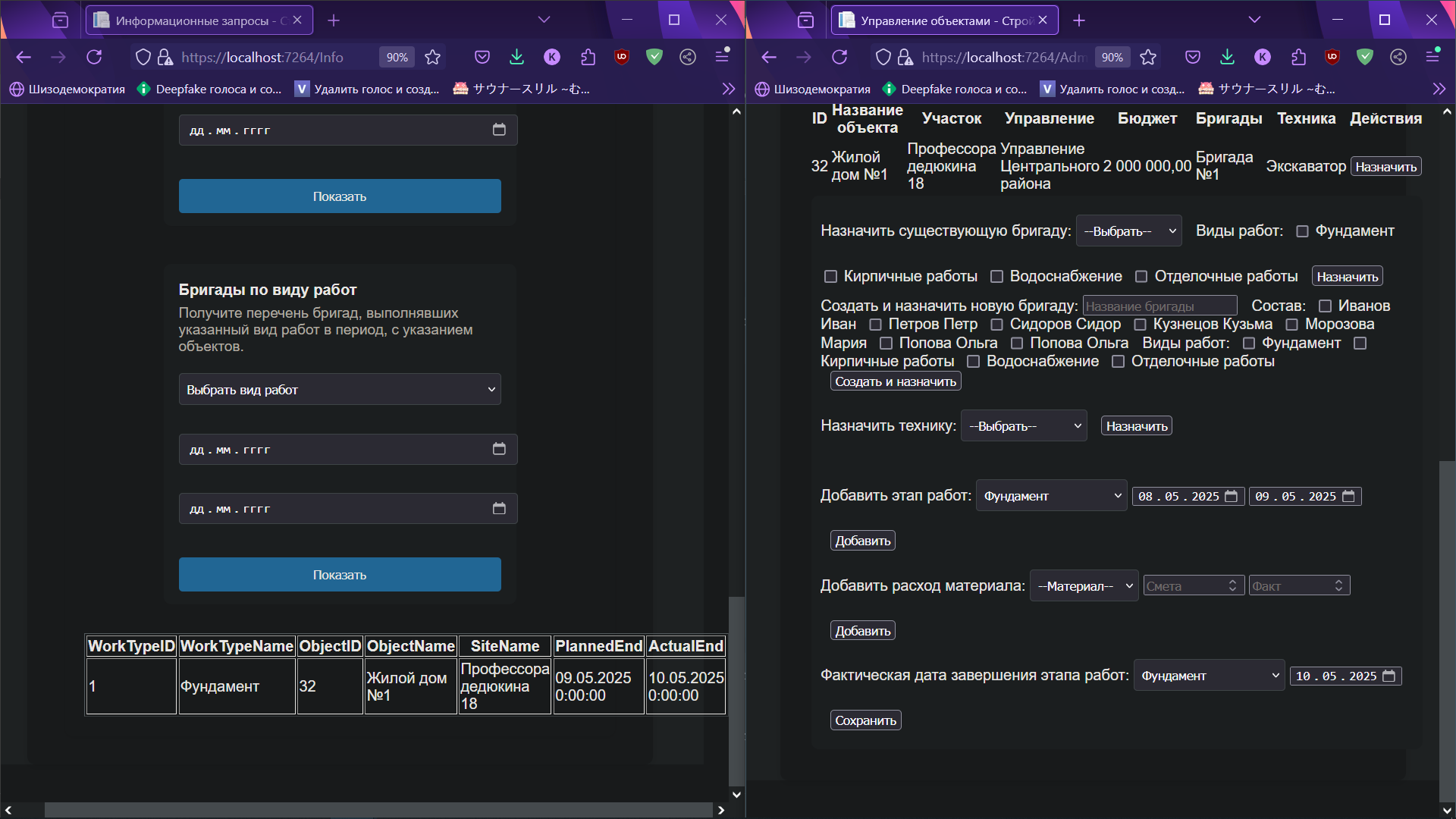
*Рисунок 7 –Назначение начальника участка.*



*Рисунок 8 –Назначение бригады.*

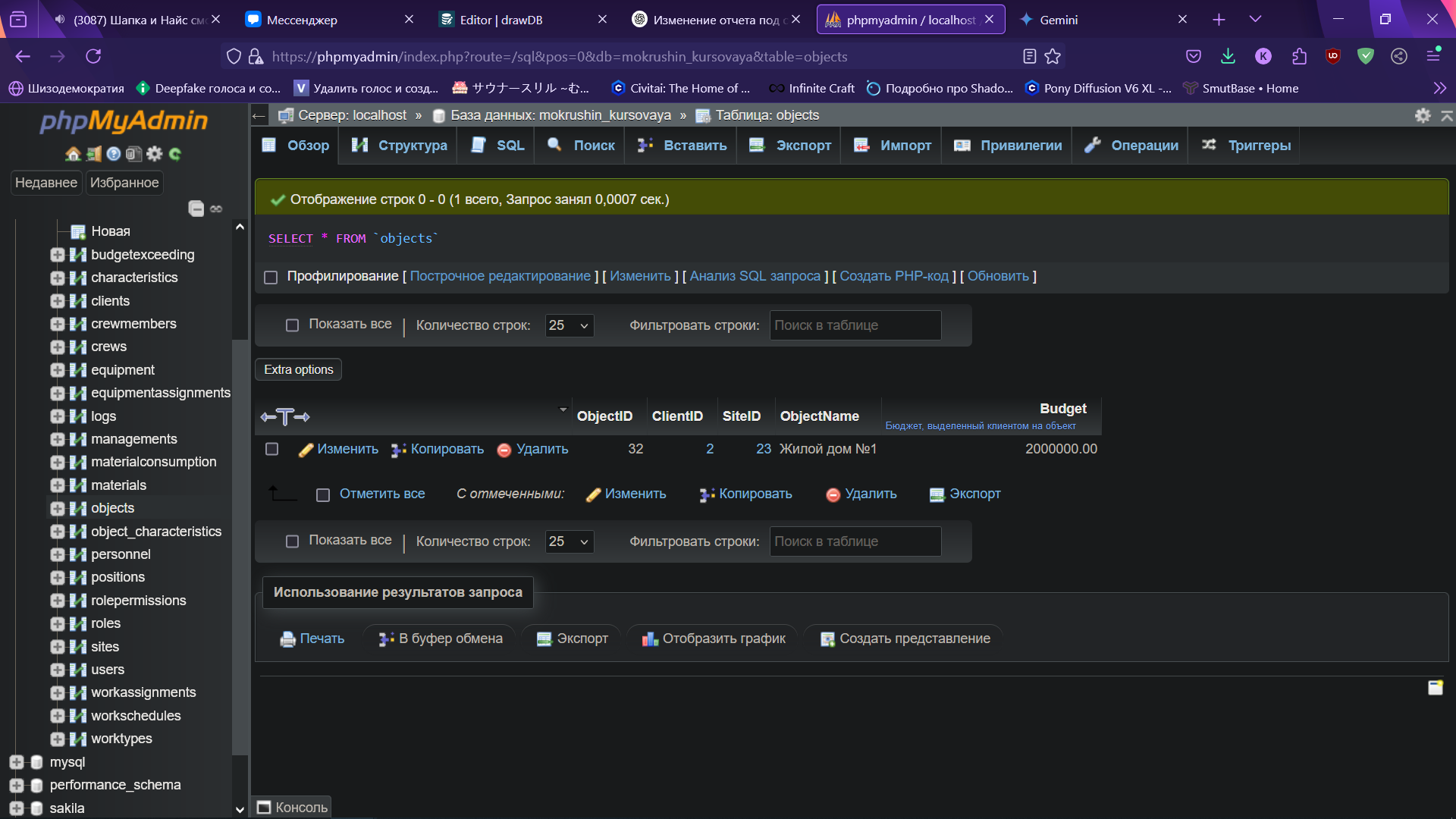


*Рисунок 9 –Назначение техники.*



*Рисунок 10 –Назначение работы.*

Новая записи в приложении должна добавиться в исходную базу данных. Таблицы с новой записью в phpMyAdmin представлена на рисунке 11.



*Рисунок 11 – Таблица объект на phpMyAdmin*

Теперь проведём тестирование запросов в информационной системе. Результат тестирование запросов представлен в таблице 2.

*Таблица 2 – Тестирование запросов в информационной системе.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № теста | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Результат | + | + | + | + | + | + | + | + | + | + | + | + | + | + |

**Выводы**

В ходе работы была разработана база данных на MySQL с продуманной структурой таблиц, связями и ограничениями. Реализованы хранимые процедуры для автоматизации операций и представления для удобного доступа к данным. Клиентское приложение получило интуитивный интерфейс с разграничением прав по ролям пользователей. Проведено комплексное тестирование, включая проверку запросов, целостности данных, безопасности и устойчивости системы. В результате создана стабильная и надежная информационная система, готовая к эксплуатации. Все компоненты работают согласованно, обеспечивая эффективное решение задач предметной области

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной курсовой работы была успешно разработана информационная система для строительной организации, включающая комплексное решение по автоматизации ключевых производственных процессов. Работа проводилась поэтапно, начиная с глубокого анализа предметной области и заканчивая реализацией полнофункционального решения.

На первом этапе были тщательно исследованы особенности производственных процессов, что позволило сформулировать четкие требования к информационной системе.

Цель работы была достигнута, и все поставленные задачи выполнены.

Проектирование базы данных осуществлялось с использованием современной методологии, включающей последовательную разработку концептуальной, логической и физической моделей. Особое внимание было уделено нормализации структуры данных, что позволило создать оптимальную схему хранения информации без избыточности. Выбор СУБД MySQL оказался полностью оправданным, так как данная система управления базами данных продемонстрировала все необходимые характеристики для работы с производственной информацией.

Реализованная система включает:

1. Полноценную базу данных, охватывающую все ключевые аспекты производства
2. Механизмы разграничения прав доступа для различных категорий пользователей
3. Эффективные инструменты для работы с данными

Тестирование системы подтвердило ее работоспособность и соответствие первоначальным требованиям. Разработанное решение успешно решает задачи учета производства, контроля качества, управления персоналом и ресурсами.

Перспективы развития системы включают несколько направлений:

* Расширение функционала планирования и контроля
* Интеграция с внешними системами
* Развитие аналитической подсистемы
* Улучшение пользовательского интерфейса и доступности
* Документооборот
* Система уведомлений

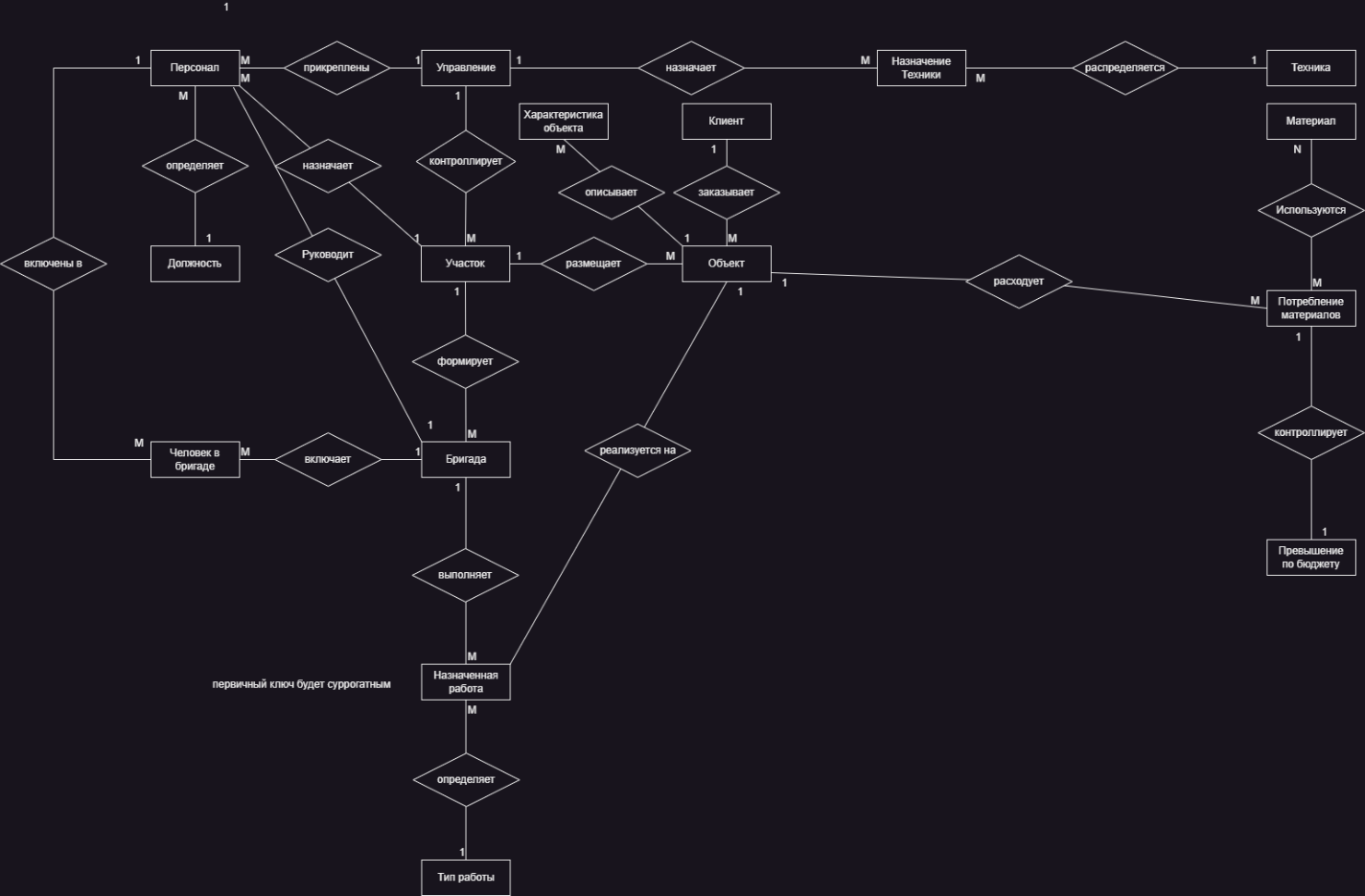
Практическая значимость работы заключается в том, что разработанная система может быть успешно внедрена в строительных организациях для повышения эффективности управления производственными процессами, сокращения временных затрат на обработку информации и минимизации количества ошибок, связанных с человеческим фактором.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Шаякбаров Н.Ф. Курсовая работа по базам данных: Методические указания. — Пермь: Изд-во ПНИПУ, 2017. — 57 с. (Примечание: Укажите точные данные методички, если они отличаются).
2. Чен П. П.–Ш. Модель "сущность–связь" — шаг к единому представлению данных // СУБД. — 1995. — № 3. — С. 137–158.
3. Мартин Дж. Проектирование баз данных. — М.: Вильямс, 2018. — 45 с.
4. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: Вильямс, 2010. — 1328 с.
5. Старкс Д. MySQL Security Handbook. — 2018. — 320 с.
6. Официальный сайт OpenServer [Электронный ресурс]. — URL: <https://ospanel.io/> (дата обращения: 04.05.2025).
7. Документация по ASP.NET Core | Microsoft Learn [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-8.0> (дата обращения: 07.05.2025).

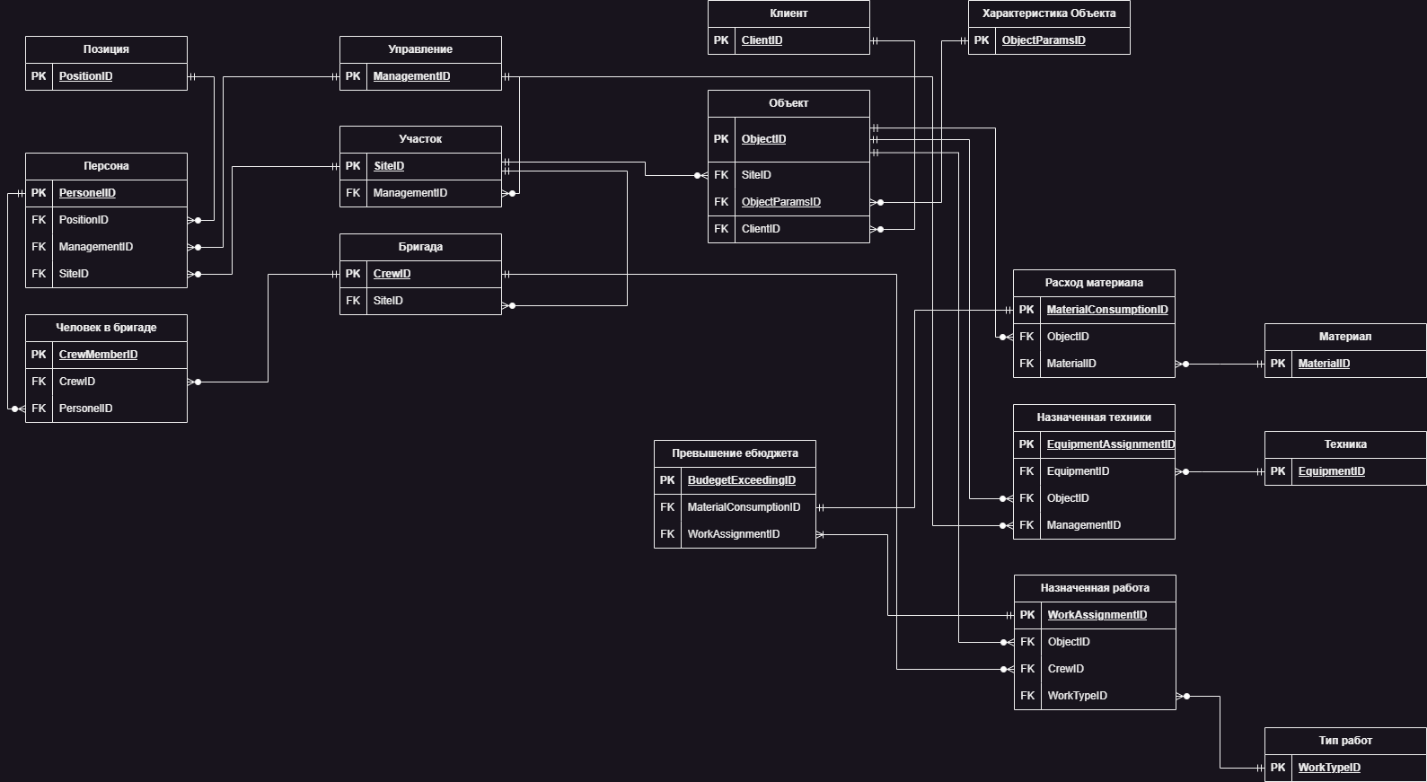
# **ПРИЛОЖЕНИЕ А**

# **Концептуальная модель**



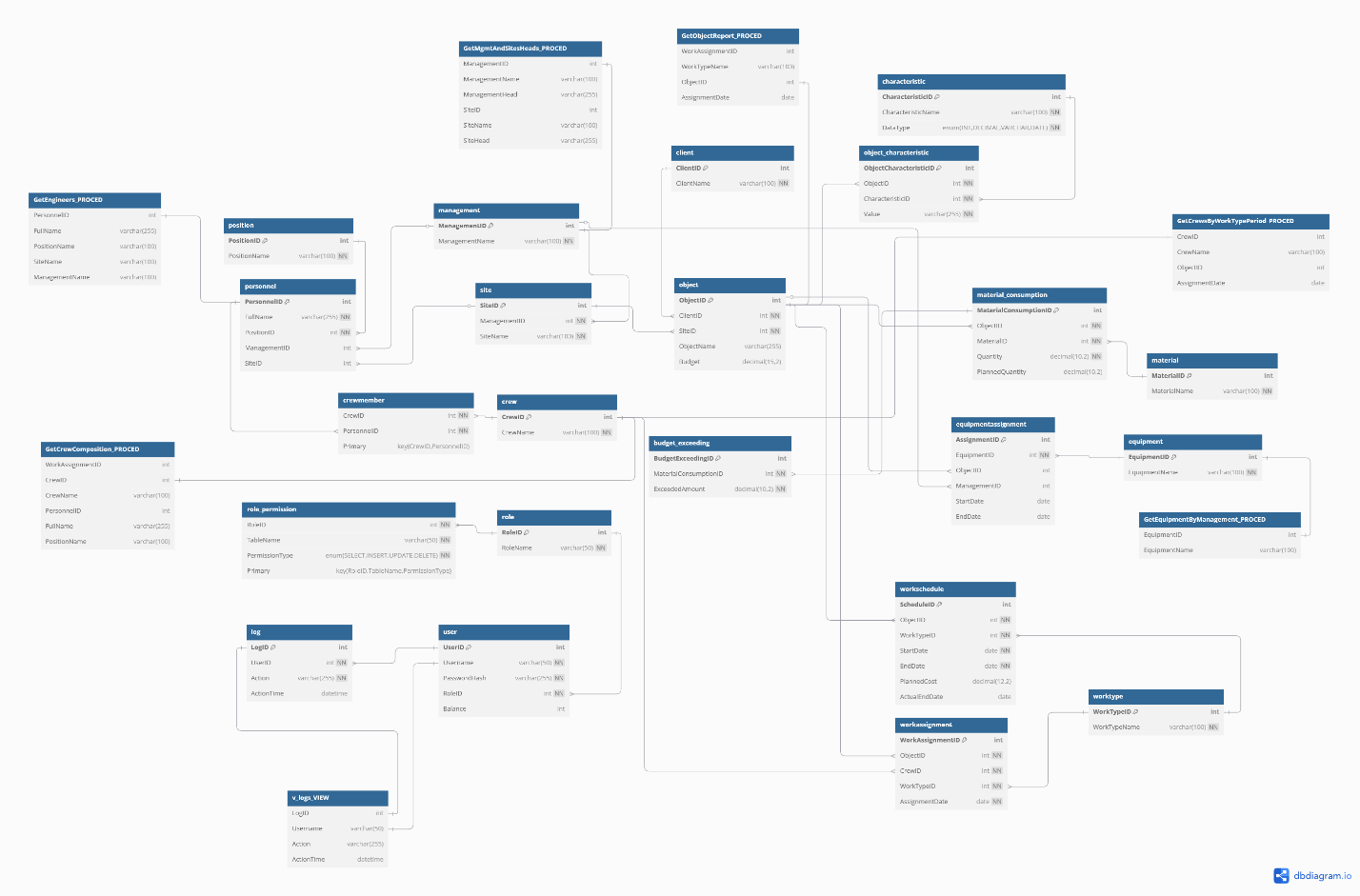
# **ПРИЛОЖЕНИЕ Б**

**Логическая модель**



# **ПРИЛОЖЕНИЕ В**

**Физическая модель**



# **ПРИЛОЖЕНИЕ Г**

**Запросы на создание пользователей в базе данных**

-- Запросы на создание пользователей и разграничение прав доступа -- Создание пользователя 'admin'@'localhost' с паролем 'admin\_password' CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin\_password'; -- Предоставление всех привилегий на базу данных 'mokrushin\_kursovaya' пользователю 'admin'@'localhost' GRANT ALL PRIVILEGES ON `mokrushin\_kursovaya`.\* TO 'admin'@'localhost'; -- Создание пользователя manager@'localhost' с паролем manager\_password' CREATE USER manager@'localhost' IDENTIFIED BY manager\_password'; -- Предоставление прав на выборку, вставку, обновление, удаление, работу с файлами и выполнение процедур GRANT SELECT, INSERT, UPDATE, DELETE, FILE, EXECUTE ON `mokrushin\_kursovaya`.\* TO manager@'localhost'; -- Создание пользователя client @'localhost' с паролем ‘client \_password' CREATE USER ‘client @'localhost' IDENTIFIED BY ‘client\_password'; -- Предоставление прав только на выборку и выполнение процедур GRANT SELECT, EXECUTE ON `mokrushin\_kursovaya`.\* TO client@'localhost'; -- Применение изменений привилегий FLUSH PRIVILEGES;

# **ПРИЛОЖЕНИЕ Д Запросы на создание сущностей в базе данных**

-

-- Структура таблицы `budgetexceeding`

--

CREATE TABLE `budgetexceeding` (

`BudgetExceedingID` int NOT NULL,

`MaterialConsumptionID` int NOT NULL,

`ExceededAmount` decimal(10,2) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Триггеры `budgetexceeding`

--

DELIMITER $$

CREATE TRIGGER `trg\_overbudget` AFTER INSERT ON `budgetexceeding` FOR EACH ROW BEGIN

INSERT INTO `Logs`(`UserID`,`Action`)

VALUES(1, CONCAT('Overbudget: consumption ID=',NEW.`MaterialConsumptionID`, ' excess=',NEW.`ExceededAmount`));

END

$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Структура таблицы `characteristics`

--

CREATE TABLE `characteristics` (

`CharacteristicID` int NOT NULL,

`CharacteristicName` varchar(100) NOT NULL,

`DataType` enum('INT','DECIMAL','VARCHAR','DATE') NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `characteristics`

--

INSERT INTO `characteristics` (`CharacteristicID`, `CharacteristicName`, `DataType`) VALUES

(1, 'Floors', 'INT'),

(2, 'MaterialType', 'VARCHAR'),

(3, 'ApartmentCount', 'INT'),

(4, 'SpanType', 'VARCHAR'),

(5, 'Width', 'DECIMAL'),

(6, 'LaneCount', 'INT'),

(7, 'RoadLength', 'DECIMAL'),

(8, 'BuildingArea', 'DECIMAL'),

(9, 'Capacity', 'INT'),

(10, 'FoundationType', 'VARCHAR');

-- --------------------------------------------------------

--

-- Структура таблицы `clients`

--

CREATE TABLE `clients` (

`ClientID` int NOT NULL,

`ClientName` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `clients`

--

INSERT INTO `clients` (`ClientID`, `ClientName`) VALUES

(1, 'Nikita'),

(2, 'Никитоска'),

(3, 'ООО Заказчик');

-- --------------------------------------------------------

--

--

CREATE TABLE `crewmembers` (

`CrewID` int NOT NULL,

`PersonnelID` int NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `crewmembers`

--

INSERT INTO `crewmembers` (`CrewID`, `PersonnelID`) VALUES

(2, 4),

(1, 5),

(1, 6);

-- --------------------------------------------------------

--

-- Структура таблицы `crews`

--

CREATE TABLE `crews` (

`CrewID` int NOT NULL,

`CrewName` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `crews`

--

INSERT INTO `crews` (`CrewID`, `CrewName`) VALUES

(1, 'Бригада №1'),

(2, 'Бригада №2');

-- --------------------------------------------------------

--

-- Структура таблицы `equipment`

--

CREATE TABLE `equipment` (

`EquipmentID` int NOT NULL,

`EquipmentName` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `equipment`

--

INSERT INTO `equipment` (`EquipmentID`, `EquipmentName`) VALUES

(1, 'Экскаватор'),

(2, 'Бульдозер'),

(3, 'Кран башенный');

-- --------------------------------------------------------

--

-- Структура таблицы `equipmentassignments`

--

CREATE TABLE `equipmentassignments` (

`AssignmentID` int NOT NULL,

`EquipmentID` int NOT NULL,

`ObjectID` int DEFAULT NULL,

`ManagementID` int DEFAULT NULL,

`StartDate` date DEFAULT NULL,

`EndDate` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- В таблице equipmentassignments поле ManagementID всегда NULL при назначении техники через интерфейс или INSERT,

-- потому что INSERT INTO equipmentassignments (ObjectID, EquipmentID, StartDate) ... не указывает ManagementID.

-- Чтобы техника была назначена на управление:

-- 1. При назначении техники на объект, ManagementID должен указываться явно, либо вычисляться по объекту.

-- Пример корректного INSERT:

-- (пусть @objectId — объект, @equipmentId — техника)

INSERT INTO equipmentassignments (EquipmentID, ObjectID, ManagementID, StartDate)

SELECT @equipmentId, @objectId, s.ManagementID, CURDATE()

FROM objects o

JOIN sites s ON o.SiteID = s.SiteID

WHERE o.ObjectID = @objectId;

-- Если техника назначается только на управление (без объекта):

-- INSERT INTO equipmentassignments (EquipmentID, ManagementID, StartDate) VALUES (@equipmentId, @managementId, CURDATE());

-- Исправьте код в C#:

-- Вместо

-- INSERT INTO equipmentassignments (ObjectID, EquipmentID, StartDate) VALUES (@obj, @eq, CURDATE())

-- используйте

-- INSERT INTO equipmentassignments (ObjectID, EquipmentID, ManagementID, StartDate)

-- SELECT @obj, @eq, s.ManagementID, CURDATE() FROM objects o JOIN sites s ON o.SiteID = s.SiteID WHERE o.ObjectID = @obj;

-- --------------------------------------------------------

--

-- Структура таблицы `logs`

--

CREATE TABLE `logs` (

`LogID` int NOT NULL,

`UserID` int NOT NULL,

`Action` varchar(255) NOT NULL,

`ActionTime` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `logs`

--

INSERT INTO `logs` (`LogID`, `UserID`, `Action`, `ActionTime`) VALUES

(1, 1, 'User created', '2025-05-02 15:15:10'),

(2, 2, 'User created', '2025-05-02 16:20:47');

-- --------------------------------------------------------

--

-- Структура таблицы `managements`

--

CREATE TABLE `managements` (

`ManagementID` int NOT NULL,

`ManagementName` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `managements`

--

INSERT INTO `managements` (`ManagementID`, `ManagementName`) VALUES

(1, 'Управление Центрального района'),

(2, 'Управление Северного района');

-- --------------------------------------------------------

--

-- Структура таблицы `materialconsumption`

--

CREATE TABLE `materialconsumption` (

`MaterialConsumptionID` int NOT NULL,

`ObjectID` int NOT NULL,

`MaterialID` int NOT NULL,

`Quantity` decimal(10,2) NOT NULL,

`PlannedQuantity` decimal(10,2) DEFAULT '0.00' COMMENT 'Плановое количество материала по смете'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- Пример корректного запроса для добавления/обновления сметы по материалу для объекта

-- ВАЖНО: @ObjectID и @MaterialID должны быть НЕ NULL и существовать в таблицах objects и materials!

-- Если запись уже есть, обновляем PlannedQuantity, иначе добавляем новую

INSERT INTO materialconsumption (ObjectID, MaterialID, PlannedQuantity, Quantity)

VALUES (1, 2, 100, 0) -- пример: объект 1, материал 2, смета 100, факт 0

ON DUPLICATE KEY UPDATE PlannedQuantity = VALUES(PlannedQuantity);

-- Если вы получаете ошибку "ObjectID не может быть NULL", значит вы передаете NULL вместо значения.

-- Проверьте, что переменные @ObjectID и @MaterialID заданы и существуют такие объекты и материалы.

-- Ошибка возникает, если ObjectID (или MaterialID) не существует в таблице objects (или materials).

-- Проверьте, что такие значения есть:

SELECT ObjectID FROM objects WHERE ObjectID = 1;

SELECT MaterialID FROM materials WHERE MaterialID = 2;

-- Если строки не возвращаются, используйте существующие значения из этих таблиц.

-- Например, если у вас есть ObjectID=26 и MaterialID=1, то запрос будет:

INSERT INTO materialconsumption (ObjectID, MaterialID, PlannedQuantity, Quantity)

VALUES (26, 1, 100, 0)

ON DUPLICATE KEY UPDATE PlannedQuantity = VALUES(PlannedQuantity);

-- Используйте только реальные ID, которые есть в таблицах objects и materials.

-- --------------------------------------------------------

--

-- Структура таблицы `materials`

--

CREATE TABLE `materials` (

`MaterialID` int NOT NULL,

`MaterialName` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `materials`

--

INSERT INTO `materials` (`MaterialID`, `MaterialName`) VALUES

(1, 'Цемент'),

(2, 'Кирпич'),

(3, 'Трубы ПВХ'),

(4, 'Штукатурка');

-- --------------------------------------------------------

--

-- Структура таблицы `objects`

--

CREATE TABLE `objects` (

`ObjectID` int NOT NULL,

`ClientID` int NOT NULL,

`SiteID` int NOT NULL,

`ObjectName` varchar(255) DEFAULT NULL,

`Budget` decimal(15,2) DEFAULT '0.00' COMMENT 'Бюджет, выделенный клиентом на объект'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- --------------------------------------------------------

--

-- Структура таблицы `object\_characteristics`

--

CREATE TABLE object\_characteristics (

ObjectCharacteristicID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

ObjectID INT NOT NULL,

CharacteristicID INT NOT NULL,

Value VARCHAR(255) NOT NULL,

FOREIGN KEY (ObjectID) REFERENCES objects(ObjectID) ON DELETE CASCADE,

FOREIGN KEY (CharacteristicID) REFERENCES characteristics(CharacteristicID) ON DELETE CASCADE

);

-- Индекс для быстрого поиска по объекту

CREATE INDEX idx\_object\_characteristics\_objectid ON object\_characteristics(ObjectID);

-- --------------------------------------------------------

--

-- Структура таблицы `personnel`

--

CREATE TABLE `personnel` (

`PersonnelID` int NOT NULL,

`FullName` varchar(255) NOT NULL,

`PositionID` int NOT NULL,

`ManagementID` int DEFAULT NULL,

`SiteID` int DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `personnel`

--

INSERT INTO `personnel` (`PersonnelID`, `FullName`, `PositionID`, `ManagementID`, `SiteID`) VALUES

(1, 'Иванов Иван', 1, 1, 1),

(2, 'Петров Петр', 2, 1, 1),

(3, 'Сидоров Сидор', 3, 1, 1),

(4, 'Кузнецов Кузьма', 2, 2, 3),

(5, 'Морозова Мария', 1, 2, 2),

(6, 'Попова Ольга', 2, 2, 2),

(7, 'Попова Ольга', 6, 2, 2);

-- --------------------------------------------------------

--

-- Структура таблицы `positions`

--

CREATE TABLE `positions` (

`PositionID` int NOT NULL,

`PositionName` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `positions`

--

INSERT INTO `positions` (`PositionID`, `PositionName`) VALUES

(1, 'Начальник управления'),

(2, 'Начальник участка'),

(3, 'Прораб'),

(4, 'Мастер'),

(5, 'Техник'),

(6, 'Инженер'),

(7, 'Технолог'),

(8, 'Каменщик'),

(9, 'Бетонщик'),

(10, 'Отделочник'),

(11, 'Сварщик'),

(12, 'Электрик'),

(13, 'Шофер'),

(14, 'Слесарь');

-- --------------------------------------------------------

--

-- Структура таблицы `rolepermissions`

--

CREATE TABLE `rolepermissions` (

`RoleID` int NOT NULL,

`TableName` varchar(50) NOT NULL,

`PermissionType` enum('SELECT','INSERT','UPDATE','DELETE') NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- --------------------------------------------------------

--

-- Структура таблицы `roles`

--

CREATE TABLE `roles` (

`RoleID` int NOT NULL,

`RoleName` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `roles`

--

INSERT INTO `roles` (`RoleID`, `RoleName`) VALUES

(1, 'admin'),

(2, 'manager'),

(5, 'user');

-- --------------------------------------------------------

--

-- Структура таблицы `sites`

--

CREATE TABLE `sites` (

`SiteID` int NOT NULL,

`ManagementID` int NOT NULL,

`SiteName` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `sites`

--

INSERT INTO `sites` (`SiteID`, `ManagementID`, `SiteName`) VALUES

(1, 1, 'Участок №1 ЦР'),

(2, 1, 'Участок №2 ЦР'),

(3, 2, 'Участок №1 СР');

-- --------------------------------------------------------

--

-- Структура таблицы `users`

--

CREATE TABLE `users` (

`UserID` int NOT NULL,

`Username` varchar(50) NOT NULL,

`PasswordHash` varchar(255) NOT NULL,

`RoleID` int NOT NULL,

`Balance` int DEFAULT '10000000'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `users`

--

INSERT INTO `users` (`UserID`, `Username`, `PasswordHash`, `RoleID`, `Balance`) VALUES

(1, 'Nikita', 'pmWkWSBCL51Bfkhn79xPuKBKHz//H6B+mY6G9/eieuM=', 1, 997999999),

(2, 'Никитоска', 'jSPPbIboNKeqbt7VTCbOK7LnSQNTjGG91dIZeZerL3I=', 5, 966999999);

--

-- Триггеры `users`

--

DELIMITER $$

CREATE TRIGGER `trg\_users\_insert` AFTER INSERT ON `users` FOR EACH ROW BEGIN

INSERT INTO `Logs`(`UserID`,`Action`)

VALUES(NEW.`UserID`,'User created');

END

$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Структура таблицы `workassignments`

--

CREATE TABLE `workassignments` (

`WorkAssignmentID` int NOT NULL,

`ObjectID` int NOT NULL,

`CrewID` int NOT NULL,

`WorkTypeID` int NOT NULL,

`AssignmentDate` date NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- --------------------------------------------------------

--

-- Структура таблицы `workschedules`

--

CREATE TABLE `workschedules` (

`ScheduleID` int NOT NULL,

`ObjectID` int NOT NULL,

`WorkTypeID` int NOT NULL,

`StartDate` date NOT NULL,

`EndDate` date NOT NULL,

`PlannedCost` decimal(12,2) DEFAULT '0.00' COMMENT 'Плановая стоимость работы',

`ActualEndDate` date DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

-- --------------------------------------------------------

--

-- Структура таблицы `worktypes`

--

CREATE TABLE `worktypes` (

`WorkTypeID` int NOT NULL,

`WorkTypeName` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

--

-- Дамп данных таблицы `worktypes`

--

INSERT INTO `worktypes` (`WorkTypeID`, `WorkTypeName`) VALUES

(1, 'Фундамент'),

(2, 'Кирпичные работы'),

(3, 'Водоснабжение'),

(4, 'Отделочные работы');

-- Представление для логов (v\_logs) с Username

CREATE OR REPLACE VIEW v\_logs AS

SELECT

l.LogID,

u.Username,

l.Action,

l.ActionTime

FROM logs l

LEFT JOIN users u ON l.UserID = u.UserID;

--

-- Индексы сохранённых таблиц

--

--

-- Индексы таблицы `budgetexceeding`

--

ALTER TABLE `budgetexceeding`

ADD PRIMARY KEY (`BudgetExceedingID`),

ADD KEY `MaterialConsumptionID` (`MaterialConsumptionID`);

--

-- Индексы таблицы `characteristics`

--

ALTER TABLE `characteristics`

ADD PRIMARY KEY (`CharacteristicID`);

--

-- Индексы таблицы `clients`

--

ALTER TABLE `clients`

ADD PRIMARY KEY (`ClientID`);

--

-- Индексы таблицы `crewmembers`

--

ALTER TABLE `crewmembers`

ADD PRIMARY KEY (`CrewID`,`PersonnelID`),

ADD KEY `PersonnelID` (`PersonnelID`);

--

-- Индексы таблицы `crews`

--

ALTER TABLE `crews`

ADD PRIMARY KEY (`CrewID`);

--

-- Индексы таблицы `equipment`

--

ALTER TABLE `equipment`

ADD PRIMARY KEY (`EquipmentID`);

--

-- Индексы таблицы `equipmentassignments`

--

ALTER TABLE `equipmentassignments`

ADD PRIMARY KEY (`AssignmentID`),

ADD KEY `EquipmentID` (`EquipmentID`),

ADD KEY `ObjectID` (`ObjectID`),

ADD KEY `ManagementID` (`ManagementID`);

--

-- Индексы таблицы `logs`

--

ALTER TABLE `logs`

ADD PRIMARY KEY (`LogID`),

ADD KEY `UserID` (`UserID`);

--

-- Индексы таблицы `managements`

--

ALTER TABLE `managements`

ADD PRIMARY KEY (`ManagementID`);

--

-- Индексы таблицы `materialconsumption`

--

ALTER TABLE `materialconsumption`

ADD PRIMARY KEY (`MaterialConsumptionID`),

ADD KEY `ObjectID` (`ObjectID`),

ADD KEY `MaterialID` (`MaterialID`);

--

-- Индексы таблицы `materials`

--

ALTER TABLE `materials`

ADD PRIMARY KEY (`MaterialID`);

--

-- Индексы таблицы `objects`

--

ALTER TABLE `objects`

ADD PRIMARY KEY (`ObjectID`),

ADD KEY `ClientID` (`ClientID`),

ADD KEY `SiteID` (`SiteID`);

--

-- Индексы таблицы `object\_characteristics`

--

ALTER TABLE `object\_characteristics`

ADD KEY `idx\_object\_characteristics\_objectid` (`ObjectID`);

--

-- Индексы таблицы `personnel`

--

ALTER TABLE `personnel`

ADD PRIMARY KEY (`PersonnelID`),

ADD KEY `PositionID` (`PositionID`),

ADD KEY `ManagementID` (`ManagementID`),

ADD KEY `SiteID` (`SiteID`);

--

-- Индексы таблицы `positions`

--

ALTER TABLE `positions`

ADD PRIMARY KEY (`PositionID`);

--

-- Индексы таблицы `rolepermissions`

--

ALTER TABLE `rolepermissions`

ADD PRIMARY KEY (`RoleID`,`TableName`,`PermissionType`);

--

-- Индексы таблицы `roles`

--

ALTER TABLE `roles`

ADD PRIMARY KEY (`RoleID`),

ADD UNIQUE KEY `RoleName` (`RoleName`);

--

-- Индексы таблицы `sites`

--

ALTER TABLE `sites`

ADD PRIMARY KEY (`SiteID`),

ADD KEY `ManagementID` (`ManagementID`);

--

-- Индексы таблицы `users`

--

ALTER TABLE `users`

ADD PRIMARY KEY (`UserID`),

ADD UNIQUE KEY `Username` (`Username`),

ADD KEY `RoleID` (`RoleID`);

--

-- Индексы таблицы `workassignments`

--

ALTER TABLE `workassignments`

ADD PRIMARY KEY (`WorkAssignmentID`),

ADD KEY `ObjectID` (`ObjectID`),

ADD KEY `CrewID` (`CrewID`),

ADD KEY `WorkTypeID` (`WorkTypeID`);

--

-- Индексы таблицы `workschedules`

--

ALTER TABLE `workschedules`

ADD PRIMARY KEY (`ScheduleID`),

ADD KEY `ObjectID` (`ObjectID`),

ADD KEY `WorkTypeID` (`WorkTypeID`);

--

-- Индексы таблицы `worktypes`

--

ALTER TABLE `worktypes`

ADD PRIMARY KEY (`WorkTypeID`);

--

-- AUTO\_INCREMENT для сохранённых таблиц

--

--

-- AUTO\_INCREMENT для таблицы `budgetexceeding`

--

ALTER TABLE `budgetexceeding`

MODIFY `BudgetExceedingID` int NOT NULL AUTO\_INCREMENT;

--

-- AUTO\_INCREMENT для таблицы `characteristics`

--

ALTER TABLE `characteristics`

MODIFY `CharacteristicID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=11;

--

-- AUTO\_INCREMENT для таблицы `clients`

--

ALTER TABLE `clients`

MODIFY `ClientID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `crews`

--

ALTER TABLE `crews`

MODIFY `CrewID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `equipment`

--

ALTER TABLE `equipment`

MODIFY `EquipmentID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT для таблицы `equipmentassignments`

--

ALTER TABLE `equipmentassignments`

MODIFY `AssignmentID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=7;

--

-- AUTO\_INCREMENT для таблицы `logs`

--

ALTER TABLE `logs`

MODIFY `LogID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `managements`

--

ALTER TABLE `managements`

MODIFY `ManagementID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `materialconsumption`

--

ALTER TABLE `materialconsumption`

MODIFY `MaterialConsumptionID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

--

-- AUTO\_INCREMENT для таблицы `materials`

--

ALTER TABLE `materials`

MODIFY `MaterialID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- AUTO\_INCREMENT для таблицы `objects`

--

ALTER TABLE `objects`

MODIFY `ObjectID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=27;

--

-- AUTO\_INCREMENT для таблицы `object\_characteristics`

--

ALTER TABLE `object\_characteristics`

MODIFY `ObjectCharacteristicID` int NOT NULL AUTO\_INCREMENT;

--

-- AUTO\_INCREMENT для таблицы `personnel`

--

ALTER TABLE `personnel`

MODIFY `PersonnelID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=8;

--

-- AUTO\_INCREMENT для таблицы `positions`

--

ALTER TABLE `positions`

MODIFY `PositionID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=15;

--

-- AUTO\_INCREMENT для таблицы `roles`

--

ALTER TABLE `roles`

MODIFY `RoleID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=15;

--

-- AUTO\_INCREMENT для таблицы `sites`

--

ALTER TABLE `sites`

MODIFY `SiteID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=18;

--

-- AUTO\_INCREMENT для таблицы `users`

--

ALTER TABLE `users`

MODIFY `UserID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `workassignments`

--

ALTER TABLE `workassignments`

MODIFY `WorkAssignmentID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=9;

--

-- AUTO\_INCREMENT для таблицы `workschedules`

--

ALTER TABLE `workschedules`

MODIFY `ScheduleID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

--

-- AUTO\_INCREMENT для таблицы `worktypes`

--

ALTER TABLE `worktypes`

MODIFY `WorkTypeID` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;

--

-- Ограничения внешнего ключа сохраненных таблиц

--

--

-- Ограничения внешнего ключа таблицы `budgetexceeding`

--

ALTER TABLE `budgetexceeding`

ADD CONSTRAINT `budgetexceeding\_ibfk\_1` FOREIGN KEY (`MaterialConsumptionID`) REFERENCES `materialconsumption` (`MaterialConsumptionID`) ON DELETE CASCADE;

--

-- Ограничения внешнего ключа таблицы `crewmembers`

--

ALTER TABLE `crewmembers`

ADD CONSTRAINT `crewmembers\_ibfk\_1` FOREIGN KEY (`CrewID`) REFERENCES `crews` (`CrewID`) ON DELETE CASCADE,

ADD CONSTRAINT `crewmembers\_ibfk\_2` FOREIGN KEY (`PersonnelID`) REFERENCES `personnel` (`PersonnelID`) ON DELETE CASCADE;

--

-- Ограничения внешнего ключа таблицы `equipmentassignments`

--

ALTER TABLE `equipmentassignments`

ADD CONSTRAINT `equipmentassignments\_ibfk\_1` FOREIGN KEY (`EquipmentID`) REFERENCES `equipment` (`EquipmentID`) ON DELETE CASCADE,

ADD CONSTRAINT `equipmentassignments\_ibfk\_2` FOREIGN KEY (`ObjectID`) REFERENCES `objects` (`ObjectID`) ON DELETE CASCADE,

ADD CONSTRAINT `equipmentassignments\_ibfk\_3` FOREIGN KEY (`ManagementID`) REFERENCES `managements` (`ManagementID`) ON DELETE CASCADE;

--

-- Ограничения внешнего ключа таблицы `logs`

--

ALTER TABLE `logs`

ADD CONSTRAINT `logs\_ibfk\_1` FOREIGN KEY (`UserID`) REFERENCES `users` (`UserID`) ON DELETE CASCADE;

--

-- Ограничения внешнего ключа таблицы `materialconsumption`

--

ALTER TABLE `materialconsumption`

ADD CONSTRAINT `materialconsumption\_ibfk\_1` FOREIGN KEY (`ObjectID`) REFERENCES `objects` (`ObjectID`) ON DELETE CASCADE,

ADD CONSTRAINT `materialconsumption\_ibfk\_2` FOREIGN KEY (`MaterialID`) REFERENCES `materials` (`MaterialID`) ON DELETE RESTRICT;

--

-- Ограничения внешнего ключа таблицы `objects`

--

ALTER TABLE `objects`

ADD CONSTRAINT `objects\_ibfk\_1` FOREIGN KEY (`ClientID`) REFERENCES `clients` (`ClientID`) ON DELETE CASCADE,

ADD CONSTRAINT `objects\_ibfk\_2` FOREIGN KEY (`SiteID`) REFERENCES `sites` (`SiteID`) ON DELETE CASCADE;

--

-- Ограничения внешнего ключа таблицы `object\_characteristics`

--

ALTER TABLE `object\_characteristics`

ADD CONSTRAINT `object\_characteristics\_ibfk\_1` FOREIGN KEY (`ObjectID`) REFERENCES `objects` (`ObjectID`) ON DELETE CASCADE,

ADD CONSTRAINT `object\_characteristics\_ibfk\_2` FOREIGN KEY (`CharacteristicID`) REFERENCES `characteristics` (`CharacteristicID`) ON DELETE CASCADE;

--

-- Ограничения внешнего ключа таблицы `personnel`

--

ALTER TABLE `personnel`

ADD CONSTRAINT `personnel\_ibfk\_1` FOREIGN KEY (`PositionID`) REFERENCES `positions` (`PositionID`) ON DELETE RESTRICT,

ADD CONSTRAINT `personnel\_ibfk\_2` FOREIGN KEY (`ManagementID`) REFERENCES `managements` (`ManagementID`) ON DELETE SET NULL,

ADD CONSTRAINT `personnel\_ibfk\_3` FOREIGN KEY (`SiteID`) REFERENCES `sites` (`SiteID`) ON DELETE SET NULL;

--

-- Ограничения внешнего ключа таблицы `rolepermissions`

--

ALTER TABLE `rolepermissions`

ADD CONSTRAINT `rolepermissions\_ibfk\_1` FOREIGN KEY (`RoleID`) REFERENCES `roles` (`RoleID`) ON DELETE CASCADE;

--

-- Ограничения внешнего ключа таблицы `sites`

--

ALTER TABLE `sites`

ADD CONSTRAINT `sites\_ibfk\_1` FOREIGN KEY (`ManagementID`) REFERENCES `managements` (`ManagementID`) ON DELETE CASCADE;

--

-- Ограничения внешнего ключа таблицы `users`

--

ALTER TABLE `users`

ADD CONSTRAINT `users\_ibfk\_1` FOREIGN KEY (`RoleID`) REFERENCES `roles` (`RoleID`) ON DELETE RESTRICT;

--

-- Ограничения внешнего ключа таблицы `workassignments`

--

ALTER TABLE `workassignments`

ADD CONSTRAINT `workassignments\_ibfk\_1` FOREIGN KEY (`ObjectID`) REFERENCES `objects` (`ObjectID`) ON DELETE CASCADE,

ADD CONSTRAINT `workassignments\_ibfk\_2` FOREIGN KEY (`CrewID`) REFERENCES `crews` (`CrewID`) ON DELETE CASCADE,

ADD CONSTRAINT `workassignments\_ibfk\_3` FOREIGN KEY (`WorkTypeID`) REFERENCES `worktypes` (`WorkTypeID`) ON DELETE RESTRICT;

--

-- Ограничения внешнего ключа таблицы `workschedules`

--

ALTER TABLE `workschedules`

ADD CONSTRAINT `workschedules\_ibfk\_1` FOREIGN KEY (`ObjectID`) REFERENCES `objects` (`ObjectID`) ON DELETE CASCADE,

ADD CONSTRAINT `workschedules\_ibfk\_2` FOREIGN KEY (`WorkTypeID`) REFERENCES `worktypes` (`WorkTypeID`) ON DELETE RESTRICT;

# **ПРИЛОЖЕНИЕ Е**

**Запросы на создание хранимых процедур в базе данных**

DELIMITER $$

--

-- Процедуры

--

DROP PROCEDURE IF EXISTS GetCrewComposition;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetCrewComposition` (IN `p\_ObjectID` INT)

BEGIN

SELECT w.WorkAssignmentID,c.CrewID,c.CrewName,cm.PersonnelID,p.FullName,pos.PositionName

FROM WorkAssignments w

JOIN Crews c ON c.CrewID=w.CrewID

JOIN CrewMembers cm ON cm.CrewID=c.CrewID

JOIN Personnel p ON p.PersonnelID=cm.PersonnelID

JOIN Positions pos ON pos.PositionID=p.PositionID

WHERE w.ObjectID=p\_ObjectID

ORDER BY c.CrewID,p.FullName;

END$$

DROP PROCEDURE IF EXISTS GetCrewsByWorkTypePeriod;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetCrewsByWorkTypePeriod` (IN `p\_WorkTypeID` INT, IN `p\_From` DATE, IN `p\_To` DATE)

BEGIN

SELECT DISTINCT wa.CrewID, c.CrewName, wa.ObjectID, wa.AssignmentDate

FROM WorkAssignments wa

JOIN Crews c ON c.CrewID=wa.CrewID

WHERE wa.WorkTypeID=p\_WorkTypeID

AND wa.AssignmentDate BETWEEN p\_From AND p\_To

ORDER BY c.CrewName, wa.AssignmentDate;

END$$

DROP PROCEDURE IF EXISTS GetCrewWorksPeriod;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetCrewWorksPeriod` (IN `p\_CrewID` INT, IN `p\_From` DATE, IN `p\_To` DATE)

BEGIN

SELECT wa.WorkAssignmentID, wt.WorkTypeName, wa.ObjectID, wa.AssignmentDate

FROM WorkAssignments wa

JOIN WorkTypes wt ON wt.WorkTypeID=wa.WorkTypeID

WHERE wa.CrewID=p\_CrewID

AND wa.AssignmentDate BETWEEN p\_From AND p\_To

ORDER BY wa.AssignmentDate;

END$$

DROP PROCEDURE IF EXISTS GetEngineers;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetEngineers` (IN `p\_SiteID` INT, IN `p\_ManagementID` INT)

BEGIN

SELECT p.PersonnelID,p.FullName,pos.PositionName,s.SiteName,m.ManagementName

FROM Personnel p

JOIN Positions pos ON pos.PositionID=p.PositionID

LEFT JOIN Sites s ON p.SiteID=s.SiteID

LEFT JOIN Managements m ON p.ManagementID=m.ManagementID

WHERE pos.PositionName IN ('Инженер','Технолог','Техник')

AND (p\_SiteID IS NULL OR p.SiteID=p\_SiteID)

AND (p\_ManagementID IS NULL OR p.ManagementID=p\_ManagementID)

ORDER BY p.FullName;

END$$

DROP PROCEDURE IF EXISTS GetEquipmentByManagement;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetEquipmentByManagement` (IN `p\_ManagementID` INT)

BEGIN

SELECT eq.EquipmentID, eq.EquipmentName

FROM EquipmentAssignments ea

JOIN Equipment eq ON eq.EquipmentID=ea.EquipmentID

WHERE ea.ManagementID=p\_ManagementID

ORDER BY eq.EquipmentName;

END$$

DROP PROCEDURE IF EXISTS GetEquipmentByObjectPeriod;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetEquipmentByObjectPeriod` (IN `p\_ObjectID` INT, IN `p\_From` DATE, IN `p\_To` DATE)

BEGIN

SELECT

eq.EquipmentID,

eq.EquipmentName,

ea.StartDate,

IFNULL(DATE\_FORMAT(ea.EndDate, '%Y-%m-%d'), 'техника еще в работе') AS EndDate

FROM EquipmentAssignments ea

JOIN Equipment eq ON eq.EquipmentID = ea.EquipmentID

WHERE ea.ObjectID = p\_ObjectID

OR (ea.StartDate <= p\_To AND (ea.EndDate >= p\_From OR ea.EndDate IS NULL))

ORDER BY ea.StartDate;

END$$

DROP PROCEDURE IF EXISTS GetMgmtAndSitesHeads;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetMgmtAndSitesHeads` ()

BEGIN

SELECT

m.ManagementID,

m.ManagementName,

COALESCE(mgr.FullName, 'руководитель еще не назначен') AS ManagementHead,

s.SiteID,

s.SiteName,

COALESCE(sht.FullName, 'руководитель еще не назначен') AS SiteHead

FROM Managements m

LEFT JOIN Personnel mgr

ON mgr.ManagementID = m.ManagementID AND mgr.PositionID = (SELECT PositionID FROM Positions WHERE PositionName='Начальник управления')

LEFT JOIN Sites s ON s.ManagementID = m.ManagementID

LEFT JOIN Personnel sht

ON sht.SiteID = s.SiteID AND sht.PositionID = (SELECT PositionID FROM Positions WHERE PositionName='Начальник участка')

ORDER BY m.ManagementID, s.SiteID;

END$$

DROP PROCEDURE IF EXISTS GetObjectReport;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetObjectReport` (IN `p\_ObjectID` INT)

BEGIN

SELECT wa.WorkAssignmentID, wt.WorkTypeName, wa.ObjectID, wa.AssignmentDate

FROM WorkAssignments wa

JOIN WorkTypes wt ON wt.WorkTypeID=wa.WorkTypeID

WHERE wa.ObjectID=p\_ObjectID

ORDER BY wa.AssignmentDate;

END$$

DROP PROCEDURE IF EXISTS GetObjectsByWorkTypePeriod;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetObjectsByWorkTypePeriod` (IN `p\_ManagementID` INT, IN `p\_WorkTypeID` INT, IN `p\_From` DATE, IN `p\_To` DATE)

BEGIN

SELECT DISTINCT

o.ObjectID,

o.ObjectName,

s.SiteName,

wt.WorkTypeName,

ws.StartDate,

ws.EndDate

FROM WorkSchedules ws

JOIN Objects o ON o.ObjectID = ws.ObjectID

JOIN Sites s ON s.SiteID = o.SiteID

JOIN Managements m ON m.ManagementID = s.ManagementID

JOIN WorkTypes wt ON wt.WorkTypeID = ws.WorkTypeID

WHERE ws.WorkTypeID = p\_WorkTypeID

AND ws.StartDate <= p\_To

AND ws.EndDate >= p\_From

AND (p\_ManagementID IS NULL OR m.ManagementID = p\_ManagementID)

ORDER BY o.ObjectID;

END$$

DROP PROCEDURE IF EXISTS GetObjectsSchedule;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetObjectsSchedule` (IN `p\_SiteID` INT, IN `p\_ManagementID` INT)

BEGIN

SELECT

o.ObjectID,

o.ObjectName,

s.SiteName,

ws.ScheduleID,

wt.WorkTypeName,

ws.StartDate,

ws.EndDate,

(SELECT MIN(ws2.StartDate) FROM WorkSchedules ws2 WHERE ws2.ObjectID = o.ObjectID AND ws2.WorkTypeID = 1) AS FoundationStartDate

FROM Objects o

JOIN Sites s ON o.SiteID = s.SiteID

JOIN Managements m ON s.ManagementID = m.ManagementID

JOIN WorkSchedules ws ON ws.ObjectID = o.ObjectID

JOIN WorkTypes wt ON wt.WorkTypeID = ws.WorkTypeID

WHERE (p\_SiteID IS NULL OR o.SiteID = p\_SiteID)

AND (p\_ManagementID IS NULL OR s.ManagementID = p\_ManagementID)

ORDER BY o.ObjectID, ws.StartDate;

END$$

DROP PROCEDURE IF EXISTS GetOverbudgetMaterials;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetOverbudgetMaterials` (IN `p\_SiteID` INT, IN `p\_ManagementID` INT)

BEGIN

SELECT

o.ObjectID,

o.ObjectName,

s.SiteName,

m.MaterialID,

m.MaterialName,

mc.PlannedQuantity AS EstimatedQty,

mc.Quantity AS ConsumedQty

FROM MaterialConsumption mc

JOIN Objects o ON o.ObjectID = mc.ObjectID

JOIN Sites s ON s.SiteID = o.SiteID

JOIN Materials m ON m.MaterialID = mc.MaterialID

WHERE mc.Quantity > mc.PlannedQuantity

AND (p\_SiteID IS NULL OR o.SiteID = p\_SiteID)

AND (p\_ManagementID IS NULL OR s.ManagementID = p\_ManagementID)

ORDER BY o.ObjectID, m.MaterialName;

END$$

DROP PROCEDURE IF EXISTS GetOverdueWorkTypes;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetOverdueWorkTypes` (IN `p\_SiteID` INT, IN `p\_ManagementID` INT)

BEGIN

SELECT

wt.WorkTypeID,

wt.WorkTypeName,

o.ObjectID,

o.ObjectName,

s.SiteName,

ws.EndDate AS PlannedEnd,

ws.ActualEndDate AS ActualEnd

FROM WorkSchedules ws

JOIN WorkTypes wt ON wt.WorkTypeID = ws.WorkTypeID

JOIN Objects o ON o.ObjectID = ws.ObjectID

JOIN Sites s ON s.SiteID = o.SiteID

WHERE ws.ActualEndDate IS NOT NULL

AND ws.EndDate < ws.ActualEndDate

AND (p\_SiteID IS NULL OR o.SiteID = p\_SiteID)

AND (p\_ManagementID IS NULL OR s.ManagementID = p\_ManagementID)

ORDER BY ws.ActualEndDate DESC;

END$$

DROP PROCEDURE IF EXISTS GetScheduleAndEstimate;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetScheduleAndEstimate` (IN `p\_ObjectID` INT)

BEGIN

-- График

SELECT ws.ScheduleID, wt.WorkTypeName, ws.StartDate, ws.EndDate

FROM WorkSchedules ws

JOIN WorkTypes wt ON wt.WorkTypeID = ws.WorkTypeID

WHERE ws.ObjectID = p\_ObjectID

ORDER BY ws.StartDate;

-- Смета (по materialconsumption)

SELECT mc.MaterialConsumptionID, m.MaterialName, mc.PlannedQuantity AS EstimatedQty, mc.Quantity AS ConsumedQty

FROM MaterialConsumption mc

JOIN Materials m ON m.MaterialID = mc.MaterialID

WHERE mc.ObjectID = p\_ObjectID

ORDER BY m.MaterialName;

END$$

DELIMITER ;

# **ПРИЛОЖЕНИЕ И**

**Код приложения**

**Структура сайта(обязательно для корректной работы):**

**└── КурсоваяМокрушин\**

**├── appsettings.json -- Конфигурация ASP.NET Core**

**└── Pages\**

**├── Admin\**

**│ ├── Objects.cshtml -- Razor-страница управления объектами (админка)**

**│ └── Objects.cshtml -- C#-кодбихайнд для Objects.cshtml**

**├── Info\**

**│ └── Index.cshtml -- Razor-страница информационных запросов**

**└── Order\**

**└── Index.cshtml -- Razor-страница оформления заказа объекта**

**Корневая папка:**

**Appsettings.json:**

**{**

**"Logging": {**

**"LogLevel": {**

**"Default": "Information",**

**"Microsoft.AspNetCore": "Warning"**

**}**

**},**

**"AllowedHosts": "\*",**

**"ConnectionStrings": {**

**"DefaultConnection": "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;"**

**}**

**}**

**Program.cs:**

**namespace КурсоваяМокрушин**

**{**

**public class Program**

**{**

**public static void Main(string[] args)**

**{**

**var builder = WebApplication.CreateBuilder(args);**

**// Add services to the container.**

**builder.Services.AddRazorPages();**

**builder.Services.AddSession();**

**var app = builder.Build();**

**// Configure the HTTP request pipeline.**

**if (!app.Environment.IsDevelopment())**

**{**

**app.UseExceptionHandler("/Error");**

**// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.**

**app.UseHsts();**

**}**

**app.UseHttpsRedirection();**

**app.UseSession();**

**app.UseStaticFiles();**

**app.UseRouting();**

**app.UseAuthorization();**

**app.MapRazorPages();**

**app.Run();**

**}**

**}**

**}**

**Menu.cshtml:**

**@page**

**@model MenuModel**

**@{**

**ViewData["Title"] = "Главное меню";**

**}**

**<h2>Главное меню</h2>**

**Menu.cshtml.cs:**

**using Microsoft.AspNetCore.Mvc.RazorPages;**

**public class MenuModel : PageModel**

**{**

**public void OnGet() { }**

**}**

**Index.cshtml(вход):**

**@\* filepath: c:\Users\kiraq\Desktop\C#\КурсоваяМокрушин\Pages\Index.cshtml \*@**

**@page**

**@model IndexModel**

**@{**

**ViewData["Title"] = "Вход";**

**}**

**<h2>Вход в систему</h2>**

**@if (!string.IsNullOrEmpty(Model.Message))**

**{**

**<div style="color:red;">@Model.Message</div>**

**}**

**<form method="post">**

**<label>Логин</label>**

**<input class="input" type="text" name="username" required>**

**<label>Пароль</label>**

**<input class="input" type="password" name="password" required>**

**<button class="button" type="submit">Войти</button>**

**</form>**

**<div style="margin-top:30px;">**

**<a class="button" href="/Register">Регистрация</a>**

**</div>**

**Index.cshtml.cs:**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.AspNetCore.Mvc.RazorPages;**

**using MySql.Data.MySqlClient;**

**using System.Security.Cryptography;**

**using System.Text;**

**namespace КурсоваяМокрушин.Pages**

**{**

**public class IndexModel : PageModel**

**{**

**[BindProperty] public string username { get; set; }**

**[BindProperty] public string password { get; set; }**

**public string Message { get; set; }**

**public IActionResult OnGet()**

**{**

**if (Request.Query.ContainsKey("logout"))**

**{**

**HttpContext.Session.Clear();**

**return RedirectToPage("/Index");**

**}**

**return Page();**

**}**

**public IActionResult OnPost()**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**string hash = Convert.ToBase64String(SHA256.Create().ComputeHash(Encoding.UTF8.GetBytes(password)));**

**using (var cmd = new MySqlCommand("SELECT UserID, RoleID FROM Users WHERE Username=@username AND PasswordHash=@hash", conn))**

**{**

**cmd.Parameters.AddWithValue("@username", username);**

**cmd.Parameters.AddWithValue("@hash", hash);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**{**

**HttpContext.Session.SetInt32("UserID", reader.GetInt32(0));**

**HttpContext.Session.SetInt32("RoleID", reader.GetInt32(1));**

**HttpContext.Session.SetString("Username", username);**

**return RedirectToPage("/Menu");**

**}**

**else**

**{**

**Message = "Неверный логин или пароль!";**

**return Page();**

**}**

**}**

**}**

**}**

**}**

**Pages:**

**Shared:**

**Layout.cshtml:**

**@\* filepath: c:\Users\kiraq\Desktop\C#\КурсоваяМокрушин\Pages\Shared\\_Layout.cshtml \*@**

**@{**

**var isLoggedIn = Context.Session.GetInt32("UserID") != null;**

**var roleId = Context.Session.GetInt32("RoleID");**

**var isAdminOrManager = roleId == 1 || roleId == 2;**

**}**

**<!DOCTYPE html>**

**<html lang="ru">**

**<head>**

**<meta charset="utf-8" />**

**<title>@ViewData["Title"] - СтройСити</title>**

**<link rel="stylesheet" href="~/css/styles.css" />**

**<style>**

**.topbar {**

**background: #2c3e50;**

**color: #fff;**

**display: flex;**

**align-items: center;**

**justify-content: space-between;**

**padding: 0 40px;**

**height: 60px;**

**}**

**.logo-title {**

**font-size: 1.25em;**

**font-weight: bold;**

**letter-spacing: 1px;**

**}**

**.logo-title a {**

**color: #fff;**

**text-decoration: none;**

**}**

**.nav-btns {**

**display: flex;**

**gap: 10px;**

**}**

**.nav-btns a, .nav-btns form button {**

**background: #2980b9;**

**color: #fff;**

**border: none;**

**padding: 8px 18px;**

**border-radius: 4px;**

**text-decoration: none;**

**font-weight: 500;**

**transition: background 0.2s;**

**cursor: pointer;**

**}**

**.nav-btns a:hover, .nav-btns form button:hover {**

**background: #1c5980;**

**}**

**.container {**

**background: #fff;**

**margin: 40px auto;**

**padding: 30px;**

**border-radius: 8px;**

**max-width: 600px;**

**box-shadow: 0 2px 8px rgba(0,0,0,0.1);**

**}**

**.nav-btns form {**

**display: inline;**

**margin: 0;**

**padding: 0;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="topbar">**

**<div class="logo-title">**

**<a asp-page="/Menu">СтройСити</a>**

**</div>**

**@if (isLoggedIn)**

**{**

**<div class="nav-btns">**

**<a asp-page="/Order/Index">Заказ объекта</a>**

**<a asp-page="/Info/Index">Информация по объекту</a>**

**@if (isAdminOrManager)**

**{**

**<a asp-page="/Admin/Objects">Управление объектами</a>**

**}**

**<form method="get" asp-page="/Index">**

**<input type="hidden" name="logout" value="1" />**

**<button type="submit">Выйти</button>**

**</form>**

**</div>**

**}**

**</div>**

**<div class="container">**

**@RenderBody()**

**</div>**

**</body>**

**</html>**

**Order:**

**Index.cshtml:**

**@\* filepath: c:\Users\kiraq\Desktop\C#\КурсоваяМокрушин\Pages\Order\Index.cshtml \*@**

**@page**

**@model OrderIndexModel**

**@{**

**ViewData["Title"] = "Заказ объекта";**

**var materials = Model.Materials ?? new List<(int Id, string Name)> {**

**(1, "Цемент"), (2, "Кирпич"), (3, "Трубы ПВХ"), (4, "Штукатурка")**

**};**

**var userBalance = Model.UserBalance ?? 0m;**

**}**

**<h2 style="text-align:center;">Заказ объекта</h2>**

**<div style="display:flex;justify-content:center;">**

**<div style="background:#fff;border-radius:8px;padding:20px;box-shadow:0 2px 8px rgba(0,0,0,0.05);min-width:350px;">**

**<div style="display:flex;justify-content:space-between;align-items:center;margin-bottom:10px;">**

**<span style="font-weight:bold;">Выберите тип объекта</span>**

**<span style="color:#2980b9;font-weight:bold;">Ваш баланс: @userBalance ₽</span>**

**</div>**

**<div style="color:#888;text-align:center;margin-bottom:20px;">Для заказа строительства выберите тип объекта</div>**

**<form method="post" id="orderForm" autocomplete="off">**

**<input type="hidden" id="activeForm" name="activeForm" value="">**

**<div style="display:flex;gap:10px;justify-content:center;flex-wrap:wrap;">**

**<button class="button" type="button" onclick="showForm('house')">Жилой дом</button>**

**<button class="button" type="button" onclick="showForm('school')">Школа</button>**

**<button class="button" type="button" onclick="showForm('bridge')">Мост</button>**

**<button class="button" type="button" onclick="showForm('hospital')">Больница</button>**

**<button class="button" type="button" onclick="showForm('road')">Дорога</button>**

**</div>**

**<div id="house-form" class="order-form" style="display:none;margin-top:30px;">**

**<h3>Жилой дом <span style="font-size:0.8em;color:#888;">(Стоимость: 2 000 000 ₽)</span></h3>**

**<label>Адрес</label>**

**<input class="input" type="text" name="address\_house" required autocomplete="off">**

**<label>Этажность</label>**

**<input class="input" type="number" name="Floors" autocomplete="off">**

**<label>Тип строительного материала</label>**

**<select class="input" name="MaterialType">**

**@foreach (var m in materials)**

**{**

**<option value="@m.Name">@m.Name</option>**

**}**

**</select>**

**<label>Число квартир</label>**

**<input class="input" type="number" name="ApartmentCount" autocomplete="off">**

**<label>Площадь здания (м²)</label>**

**<input class="input" type="number" name="BuildingArea" step="0.01" autocomplete="off">**

**<label>Тип фундамента</label>**

**<input class="input" type="text" name="FoundationType" autocomplete="off">**

**<button class="button" type="submit" onclick="setActiveForm('submit\_house')">Заказать</button>**

**</div>**

**<div id="school-form" class="order-form" style="display:none;margin-top:30px;">**

**<h3>Школа <span style="font-size:0.8em;color:#888;">(Стоимость: 5 000 000 ₽)</span></h3>**

**<label>Адрес</label>**

**<input class="input" type="text" name="address\_school" required autocomplete="off">**

**<label>Вместимость (чел.)</label>**

**<input class="input" type="number" name="Capacity" autocomplete="off">**

**<label>Площадь здания (м²)</label>**

**<input class="input" type="number" name="BuildingArea" step="0.01" autocomplete="off">**

**<label>Тип фундамента</label>**

**<input class="input" type="text" name="FoundationType" autocomplete="off">**

**<button class="button" type="submit" onclick="setActiveForm('submit\_school')">Заказать</button>**

**</div>**

**<div id="bridge-form" class="order-form" style="display:none;margin-top:30px;">**

**<h3>Мост <span style="font-size:0.8em;color:#888;">(Стоимость: 10 000 000 ₽)</span></h3>**

**<label>Адрес</label>**

**<input class="input" type="text" name="address\_bridge" required autocomplete="off">**

**<label>Тип пролетного строения</label>**

**<input class="input" type="text" name="SpanType" autocomplete="off">**

**<label>Ширина (м)</label>**

**<input class="input" type="number" name="Width" step="0.01" autocomplete="off">**

**<label>Количество полос</label>**

**<input class="input" type="number" name="LaneCount" autocomplete="off">**

**<button class="button" type="submit" onclick="setActiveForm('submit\_bridge')">Заказать</button>**

**</div>**

**<div id="hospital-form" class="order-form" style="display:none;margin-top:30px;">**

**<h3>Больница <span style="font-size:0.8em;color:#888;">(Стоимость: 7 000 000 ₽)</span></h3>**

**<label>Адрес</label>**

**<input class="input" type="text" name="address\_hospital" required autocomplete="off">**

**<label>Вместимость (чел.)</label>**

**<input class="input" type="number" name="Capacity" autocomplete="off">**

**<label>Площадь здания (м²)</label>**

**<input class="input" type="number" name="BuildingArea" step="0.01" autocomplete="off">**

**<label>Тип фундамента</label>**

**<input class="input" type="text" name="FoundationType" autocomplete="off">**

**<button class="button" type="submit" onclick="setActiveForm('submit\_hospital')">Заказать</button>**

**</div>**

**<div id="road-form" class="order-form" style="display:none;margin-top:30px;">**

**<h3>Дорога <span style="font-size:0.8em;color:#888;">(Стоимость: 3 000 000 ₽)</span></h3>**

**<label>Адрес</label>**

**<input class="input" type="text" name="address\_road" required autocomplete="off">**

**<label>Длина дороги (км)</label>**

**<input class="input" type="number" name="RoadLength" step="0.01" autocomplete="off">**

**<label>Ширина (м)</label>**

**<input class="input" type="number" name="Width" step="0.01" autocomplete="off">**

**<label>Тип фундамента</label>**

**<input class="input" type="text" name="FoundationType" autocomplete="off">**

**<button class="button" type="submit" onclick="setActiveForm('submit\_road')">Заказать</button>**

**</div>**

**</form>**

**@if (!string.IsNullOrEmpty(Model.Message))**

**{**

**<div style="color:green;margin-top:20px;">@Model.Message</div>**

**}**

**@if (!string.IsNullOrEmpty(Model.Message))**

**{**

**<div style="color:green;margin-top:20px;">@Model.Message</div>**

**<script>**

**// Очистка всех полей после успешного заказа**

**document.addEventListener('DOMContentLoaded', function() {**

**setTimeout(function() {**

**document.getElementById('orderForm').reset();**

**document.getElementById('activeForm').value = '';**

**document.querySelectorAll('.order-form').forEach(f => f.style.display = 'none');**

**}, 100);**

**});**

**</script>**

**}**

**</div>**

**</div>**

**<script>**

**function showForm(type) {**

**document.querySelectorAll('.order-form').forEach(f => {**

**f.style.display = 'none';**

**f.querySelectorAll('input,select,button').forEach(el => el.disabled = true);**

**});**

**var form = document.getElementById(type + '-form');**

**form.style.display = 'block';**

**form.querySelectorAll('input,select,button').forEach(el => el.disabled = false);**

**document.getElementById('activeForm').value = '';**

**}**

**function setActiveForm(name) {**

**document.getElementById('activeForm').value = name;**

**}**

**window.onload = function() { showForm('house'); }**

**</script>**

**Index.cshtml.cs:**

**using Microsoft.AspNetCore.Mvc; // Добавьте этот using**

**using Microsoft.AspNetCore.Mvc.RazorPages;**

**using MySql.Data.MySqlClient;**

**using System.Collections.Generic;**

**using Microsoft.AspNetCore.Http;**

**public class OrderIndexModel : PageModel**

**{**

**public List<(int Id, string Name)> Materials { get; set; } = new List<(int, string)>();**

**public decimal? UserBalance { get; set; }**

**public string Message { get; set; }**

**public void OnGet()**

**{**

**LoadMaterials();**

**LoadUserBalance();**

**// Не выполняем никаких действий по созданию объекта!**

**}**

**public IActionResult OnPost()**

**{**

**LoadMaterials();**

**LoadUserBalance();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**int? userId = HttpContext.Session.GetInt32("UserID");**

**string username = HttpContext.Session.GetString("Username");**

**if (userId == null || string.IsNullOrEmpty(username))**

**{**

**Message = "Ошибка: пользователь не авторизован.";**

**return Page();**

**}**

**var prices = new Dictionary<string, decimal>**

**{**

**{ "Жилой дом", 2000000m },**

**{ "Школа", 5000000m },**

**{ "Мост", 10000000m },**

**{ "Больница", 7000000m },**

**{ "Дорога", 3000000m }**

**};**

**string submitType = Request.Form["activeForm"];**

**if (string.IsNullOrEmpty(submitType))**

**return Page();**

**string objectType = null, address = null, name = null;**

**int? siteId = null;**

**try**

**{**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**decimal price = 0;**

**if (submitType == "submit\_house") price = prices["Жилой дом"];**

**else if (submitType == "submit\_school") price = prices["Школа"];**

**else if (submitType == "submit\_bridge") price = prices["Мост"];**

**else if (submitType == "submit\_hospital") price = prices["Больница"];**

**else if (submitType == "submit\_road") price = prices["Дорога"];**

**if (UserBalance < price)**

**{**

**Message = "Недостаточно средств для заказа объекта!";**

**return Page();**

**}**

**int clientId = GetOrCreateClient(conn, userId.Value);**

**int newObjectId = -1; // ID созданного объекта**

**var charValues = new Dictionary<string, string>();**

**if (submitType == "submit\_house")**

**{**

**objectType = "Жилой дом";**

**address = Request.Form["address\_house"];**

**int.TryParse(Request.Form["Floors"], out int floors);**

**string materialType = Request.Form["MaterialType"];**

**int.TryParse(Request.Form["ApartmentCount"], out int apartmentCount);**

**decimal.TryParse(Request.Form["BuildingArea"], out decimal buildingArea);**

**string foundationType = Request.Form["FoundationType"];**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["Floors"] = Request.Form["Floors"];**

**charValues["MaterialType"] = Request.Form["MaterialType"];**

**charValues["ApartmentCount"] = Request.Form["ApartmentCount"];**

**charValues["BuildingArea"] = Request.Form["BuildingArea"];**

**charValues["FoundationType"] = Request.Form["FoundationType"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else if (submitType == "submit\_school")**

**{**

**objectType = "Школа";**

**address = Request.Form["address\_school"];**

**int.TryParse(Request.Form["Capacity"], out int capacity);**

**decimal.TryParse(Request.Form["BuildingArea"], out decimal buildingArea);**

**string foundationType = Request.Form["FoundationType"];**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["Capacity"] = Request.Form["Capacity"];**

**charValues["BuildingArea"] = Request.Form["BuildingArea"];**

**charValues["FoundationType"] = Request.Form["FoundationType"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else if (submitType == "submit\_bridge")**

**{**

**objectType = "Мост";**

**address = Request.Form["address\_bridge"];**

**string spanType = Request.Form["SpanType"];**

**decimal.TryParse(Request.Form["Width"], out decimal width);**

**int.TryParse(Request.Form["LaneCount"], out int laneCount);**

**decimal.TryParse(Request.Form["RoadLength"], out decimal roadLength);**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["SpanType"] = Request.Form["SpanType"];**

**charValues["Width"] = Request.Form["Width"];**

**charValues["LaneCount"] = Request.Form["LaneCount"];**

**charValues["RoadLength"] = Request.Form["RoadLength"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else if (submitType == "submit\_hospital")**

**{**

**objectType = "Больница";**

**address = Request.Form["address\_hospital"];**

**int.TryParse(Request.Form["Capacity"], out int capacity);**

**decimal.TryParse(Request.Form["BuildingArea"], out decimal buildingArea);**

**string foundationType = Request.Form["FoundationType"];**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["Capacity"] = Request.Form["Capacity"];**

**charValues["BuildingArea"] = Request.Form["BuildingArea"];**

**charValues["FoundationType"] = Request.Form["FoundationType"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else if (submitType == "submit\_road")**

**{**

**objectType = "Дорога";**

**address = Request.Form["address\_road"];**

**decimal.TryParse(Request.Form["RoadLength"], out decimal roadLength);**

**decimal.TryParse(Request.Form["Width"], out decimal width);**

**string foundationType = Request.Form["FoundationType"];**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["RoadLength"] = Request.Form["RoadLength"];**

**charValues["Width"] = Request.Form["Width"];**

**charValues["FoundationType"] = Request.Form["FoundationType"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else**

**{**

**Message = "Ошибка: не выбран тип объекта или форма не отправлена корректно.";**

**return Page();**

**}**

**}**

**catch (System.Exception ex)**

**{**

**Message = ex.Message;**

**TempData["OrderMessage"] = Message;**

**return RedirectToPage();**

**}**

**}**

**public override void OnPageHandlerExecuted(Microsoft.AspNetCore.Mvc.Filters.PageHandlerExecutedContext context)**

**{**

**// Сообщение выводится только если это GET-запрос после RedirectToPage**

**if (TempData.ContainsKey("OrderMessage") && HttpContext.Request.Method == "GET")**

**Message = TempData["OrderMessage"] as string;**

**else**

**Message = null;**

**}**

**private int? CheckSiteExists(MySqlConnection conn, string address)**

**{**

**using (var cmd = new MySqlCommand("SELECT SiteID FROM sites WHERE SiteName = @name", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", address);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**return reader.GetInt32(0);**

**}**

**return null;**

**}**

**private int GetOrCreateSite(MySqlConnection conn, string address)**

**{**

**// Проверка: если участок с таким именем уже есть, выводим ошибку**

**int? existingSiteId = CheckSiteExists(conn, address);**

**if (existingSiteId != null)**

**{**

**// Проверяем, есть ли к этому участку уже привязанный объект**

**using (var cmd = new MySqlCommand("SELECT COUNT(\*) FROM objects WHERE SiteID = @siteId", conn))**

**{**

**cmd.Parameters.AddWithValue("@siteId", existingSiteId.Value);**

**int count = System.Convert.ToInt32(cmd.ExecuteScalar());**

**if (count > 0)**

**{**

**// Такой участок уже занят объектом**

**throw new System.Exception("Участок с таким адресом уже существует и занят объектом. Введите другой адрес.");**

**}**

**else**

**{**

**// Участок свободен, можно использовать**

**return existingSiteId.Value;**

**}**

**}**

**}**

**// Если нет, создать новый участок**

**using (var cmd = new MySqlCommand("INSERT INTO sites (ManagementID, SiteName) VALUES (1, @name); SELECT LAST\_INSERT\_ID();", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", address);**

**return System.Convert.ToInt32(cmd.ExecuteScalar());**

**}**

**}**

**private string GetNextObjectName(MySqlConnection conn, string baseName)**

**{**

**// Исправлено: используем ObjectName вместо ObjectType для подсчета**

**using var cmd = new MySqlCommand("SELECT COUNT(\*) FROM objects WHERE ObjectName LIKE @pattern", conn);**

**cmd.Parameters.AddWithValue("@pattern", baseName + " №%");**

**int count = System.Convert.ToInt32(cmd.ExecuteScalar());**

**return $"{baseName} №{count + 1}";**

**}**

**private int InsertObject(MySqlConnection conn, string name, int clientId, int siteId, decimal budget)**

**{**

**// Здесь siteId — это ID выбранного участка**

**using var cmd = new MySqlCommand("INSERT INTO objects (ObjectName, ClientID, SiteID, Budget) VALUES (@name, @client, @site, @budget); SELECT LAST\_INSERT\_ID();", conn);**

**cmd.Parameters.AddWithValue("@name", name);**

**cmd.Parameters.AddWithValue("@client", clientId);**

**cmd.Parameters.AddWithValue("@site", siteId);**

**cmd.Parameters.AddWithValue("@budget", budget);**

**return System.Convert.ToInt32(cmd.ExecuteScalar());**

**}**

**private int GetOrCreateClient(MySqlConnection conn, int userId)**

**{**

**// Получить имя пользователя**

**string username = null;**

**using (var cmd = new MySqlCommand("SELECT Username FROM users WHERE UserID=@uid", conn))**

**{**

**cmd.Parameters.AddWithValue("@uid", userId);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**username = reader.GetString(0);**

**}**

**if (string.IsNullOrEmpty(username))**

**throw new System.Exception("Не найден пользователь для создания клиента");**

**// Проверить, есть ли клиент с таким именем**

**int clientId = -1;**

**using (var cmd = new MySqlCommand("SELECT ClientID FROM clients WHERE ClientName=@name", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", username);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**clientId = reader.GetInt32(0);**

**}**

**if (clientId != -1)**

**return clientId;**

**// Если нет, создать клиента с именем пользователя**

**using (var cmd = new MySqlCommand("INSERT INTO clients (ClientName) VALUES (@name);", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", username);**

**cmd.ExecuteNonQuery();**

**}**

**// Получить только что созданный ClientID**

**using (var cmd = new MySqlCommand("SELECT ClientID FROM clients WHERE ClientName=@name", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", username);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**return reader.GetInt32(0);**

**}**

**throw new System.Exception("Не удалось создать клиента");**

**}**

**private void LoadMaterials()**

**{**

**Materials = new List<(int, string)>();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT MaterialID, MaterialName FROM materials", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**Materials.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**}**

**private void LoadUserBalance()**

**{**

**UserBalance = 0;**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**int? userId = HttpContext.Session.GetInt32("UserID");**

**if (userId == null)**

**return;**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT Balance FROM users WHERE UserID=@uid", conn);**

**cmd.Parameters.AddWithValue("@uid", userId.Value);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read() && !reader.IsDBNull(0))**

**UserBalance = reader.GetDecimal(0);**

**}**

**private void UpdateUserBalance(MySqlConnection conn, int userId, decimal amount)**

**{**

**using var cmd = new MySqlCommand("UPDATE users SET Balance = Balance - @amount WHERE UserID = @uid", conn);**

**cmd.Parameters.AddWithValue("@amount", amount);**

**cmd.Parameters.AddWithValue("@uid", userId);**

**cmd.ExecuteNonQuery();**

**}**

**private void GetManagementInfoForObject(MySqlConnection conn, int objectId)**

**{**

**string sql = @"**

**SELECT o.ObjectID, o.ObjectName, s.SiteID, s.SiteName, m.ManagementID, m.ManagementName**

**FROM objects o**

**JOIN sites s ON o.SiteID = s.SiteID**

**JOIN managements m ON s.ManagementID = m.ManagementID**

**WHERE o.ObjectID = @objectId**

**";**

**using var cmd = new MySqlCommand(sql, conn);**

**cmd.Parameters.AddWithValue("@objectId", objectId);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**// Process the data as needed**

**}**

**}**

**private (int siteId, int managementId, string siteName, string managementName) GetRandomSiteAndManagement()**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**// Получаем все участки с их управлениями**

**var sites = new List<(int siteId, int managementId, string siteName, string managementName)>();**

**using (var cmd = new MySqlCommand(@"**

**SELECT s.SiteID, s.SiteName, m.ManagementID, m.ManagementName**

**FROM sites s**

**JOIN managements m ON s.ManagementID = m.ManagementID**

**", conn))**

**using (var reader = cmd.ExecuteReader())**

**{**

**while (reader.Read())**

**sites.Add((reader.GetInt32(0), reader.GetInt32(2), reader.GetString(1), reader.GetString(3)));**

**}**

**// Случайно выбираем участок**

**var rnd = new Random();**

**var chosen = sites[rnd.Next(sites.Count)];**

**return chosen;**

**}**

**private void SaveObjectCharacteristics(int objectId, Dictionary<string, string> values)**

**{**

**// Карта: имя поля формы -> CharacteristicID**

**var charMap = new Dictionary<string, int>**

**{**

**{ "Floors", 1 },**

**{ "MaterialType", 2 },**

**{ "ApartmentCount", 3 },**

**{ "SpanType", 4 },**

**{ "Width", 5 },**

**{ "LaneCount", 6 },**

**{ "RoadLength", 7 },**

**{ "BuildingArea", 8 },**

**{ "Capacity", 9 },**

**{ "FoundationType", 10 }**

**};**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**foreach (var kv in charMap)**

**{**

**if (values.TryGetValue(kv.Key, out var val) && !string.IsNullOrWhiteSpace(val))**

**{**

**using var cmd = new MySqlCommand("INSERT INTO object\_characteristics (ObjectID, CharacteristicID, Value) VALUES (@obj, @cid, @val)", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@cid", kv.Value);**

**cmd.Parameters.AddWithValue("@val", val);**

**cmd.ExecuteNonQuery();**

**}**

**}**

**}**

**}**

**Info:**

**Index.cshtml:**

**using Microsoft.AspNetCore.Mvc; // Добавьте этот using**

**using Microsoft.AspNetCore.Mvc.RazorPages;**

**using MySql.Data.MySqlClient;**

**using System.Collections.Generic;**

**using Microsoft.AspNetCore.Http;**

**public class OrderIndexModel : PageModel**

**{**

**public List<(int Id, string Name)> Materials { get; set; } = new List<(int, string)>();**

**public decimal? UserBalance { get; set; }**

**public string Message { get; set; }**

**public void OnGet()**

**{**

**LoadMaterials();**

**LoadUserBalance();**

**// Не выполняем никаких действий по созданию объекта!**

**}**

**public IActionResult OnPost()**

**{**

**LoadMaterials();**

**LoadUserBalance();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**int? userId = HttpContext.Session.GetInt32("UserID");**

**string username = HttpContext.Session.GetString("Username");**

**if (userId == null || string.IsNullOrEmpty(username))**

**{**

**Message = "Ошибка: пользователь не авторизован.";**

**return Page();**

**}**

**var prices = new Dictionary<string, decimal>**

**{**

**{ "Жилой дом", 2000000m },**

**{ "Школа", 5000000m },**

**{ "Мост", 10000000m },**

**{ "Больница", 7000000m },**

**{ "Дорога", 3000000m }**

**};**

**string submitType = Request.Form["activeForm"];**

**if (string.IsNullOrEmpty(submitType))**

**return Page();**

**string objectType = null, address = null, name = null;**

**int? siteId = null;**

**try**

**{**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**decimal price = 0;**

**if (submitType == "submit\_house") price = prices["Жилой дом"];**

**else if (submitType == "submit\_school") price = prices["Школа"];**

**else if (submitType == "submit\_bridge") price = prices["Мост"];**

**else if (submitType == "submit\_hospital") price = prices["Больница"];**

**else if (submitType == "submit\_road") price = prices["Дорога"];**

**if (UserBalance < price)**

**{**

**Message = "Недостаточно средств для заказа объекта!";**

**return Page();**

**}**

**int clientId = GetOrCreateClient(conn, userId.Value);**

**int newObjectId = -1; // ID созданного объекта**

**var charValues = new Dictionary<string, string>();**

**if (submitType == "submit\_house")**

**{**

**objectType = "Жилой дом";**

**address = Request.Form["address\_house"];**

**int.TryParse(Request.Form["Floors"], out int floors);**

**string materialType = Request.Form["MaterialType"];**

**int.TryParse(Request.Form["ApartmentCount"], out int apartmentCount);**

**decimal.TryParse(Request.Form["BuildingArea"], out decimal buildingArea);**

**string foundationType = Request.Form["FoundationType"];**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["Floors"] = Request.Form["Floors"];**

**charValues["MaterialType"] = Request.Form["MaterialType"];**

**charValues["ApartmentCount"] = Request.Form["ApartmentCount"];**

**charValues["BuildingArea"] = Request.Form["BuildingArea"];**

**charValues["FoundationType"] = Request.Form["FoundationType"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else if (submitType == "submit\_school")**

**{**

**objectType = "Школа";**

**address = Request.Form["address\_school"];**

**int.TryParse(Request.Form["Capacity"], out int capacity);**

**decimal.TryParse(Request.Form["BuildingArea"], out decimal buildingArea);**

**string foundationType = Request.Form["FoundationType"];**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["Capacity"] = Request.Form["Capacity"];**

**charValues["BuildingArea"] = Request.Form["BuildingArea"];**

**charValues["FoundationType"] = Request.Form["FoundationType"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else if (submitType == "submit\_bridge")**

**{**

**objectType = "Мост";**

**address = Request.Form["address\_bridge"];**

**string spanType = Request.Form["SpanType"];**

**decimal.TryParse(Request.Form["Width"], out decimal width);**

**int.TryParse(Request.Form["LaneCount"], out int laneCount);**

**decimal.TryParse(Request.Form["RoadLength"], out decimal roadLength);**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["SpanType"] = Request.Form["SpanType"];**

**charValues["Width"] = Request.Form["Width"];**

**charValues["LaneCount"] = Request.Form["LaneCount"];**

**charValues["RoadLength"] = Request.Form["RoadLength"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else if (submitType == "submit\_hospital")**

**{**

**objectType = "Больница";**

**address = Request.Form["address\_hospital"];**

**int.TryParse(Request.Form["Capacity"], out int capacity);**

**decimal.TryParse(Request.Form["BuildingArea"], out decimal buildingArea);**

**string foundationType = Request.Form["FoundationType"];**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["Capacity"] = Request.Form["Capacity"];**

**charValues["BuildingArea"] = Request.Form["BuildingArea"];**

**charValues["FoundationType"] = Request.Form["FoundationType"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else if (submitType == "submit\_road")**

**{**

**objectType = "Дорога";**

**address = Request.Form["address\_road"];**

**decimal.TryParse(Request.Form["RoadLength"], out decimal roadLength);**

**decimal.TryParse(Request.Form["Width"], out decimal width);**

**string foundationType = Request.Form["FoundationType"];**

**name = GetNextObjectName(conn, objectType);**

**siteId = GetOrCreateSite(conn, address);**

**newObjectId = InsertObject(conn, name, clientId, siteId.Value, price);**

**charValues["RoadLength"] = Request.Form["RoadLength"];**

**charValues["Width"] = Request.Form["Width"];**

**charValues["FoundationType"] = Request.Form["FoundationType"];**

**UpdateUserBalance(conn, userId.Value, price);**

**Message = $"Объект '{name}' успешно создан! С вашего баланса списано {price} ₽.";**

**TempData["OrderMessage"] = Message;**

**SaveObjectCharacteristics(newObjectId, charValues);**

**return RedirectToPage(new { success = true });**

**}**

**else**

**{**

**Message = "Ошибка: не выбран тип объекта или форма не отправлена корректно.";**

**return Page();**

**}**

**}**

**catch (System.Exception ex)**

**{**

**Message = ex.Message;**

**TempData["OrderMessage"] = Message;**

**return RedirectToPage();**

**}**

**}**

**public override void OnPageHandlerExecuted(Microsoft.AspNetCore.Mvc.Filters.PageHandlerExecutedContext context)**

**{**

**// Сообщение выводится только если это GET-запрос после RedirectToPage**

**if (TempData.ContainsKey("OrderMessage") && HttpContext.Request.Method == "GET")**

**Message = TempData["OrderMessage"] as string;**

**else**

**Message = null;**

**}**

**private int? CheckSiteExists(MySqlConnection conn, string address)**

**{**

**using (var cmd = new MySqlCommand("SELECT SiteID FROM sites WHERE SiteName = @name", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", address);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**return reader.GetInt32(0);**

**}**

**return null;**

**}**

**private int GetOrCreateSite(MySqlConnection conn, string address)**

**{**

**// Проверка: если участок с таким именем уже есть, выводим ошибку**

**int? existingSiteId = CheckSiteExists(conn, address);**

**if (existingSiteId != null)**

**{**

**// Проверяем, есть ли к этому участку уже привязанный объект**

**using (var cmd = new MySqlCommand("SELECT COUNT(\*) FROM objects WHERE SiteID = @siteId", conn))**

**{**

**cmd.Parameters.AddWithValue("@siteId", existingSiteId.Value);**

**int count = System.Convert.ToInt32(cmd.ExecuteScalar());**

**if (count > 0)**

**{**

**// Такой участок уже занят объектом**

**throw new System.Exception("Участок с таким адресом уже существует и занят объектом. Введите другой адрес.");**

**}**

**else**

**{**

**// Участок свободен, можно использовать**

**return existingSiteId.Value;**

**}**

**}**

**}**

**// Если нет, создать новый участок**

**using (var cmd = new MySqlCommand("INSERT INTO sites (ManagementID, SiteName) VALUES (1, @name); SELECT LAST\_INSERT\_ID();", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", address);**

**return System.Convert.ToInt32(cmd.ExecuteScalar());**

**}**

**}**

**private string GetNextObjectName(MySqlConnection conn, string baseName)**

**{**

**// Исправлено: используем ObjectName вместо ObjectType для подсчета**

**using var cmd = new MySqlCommand("SELECT COUNT(\*) FROM objects WHERE ObjectName LIKE @pattern", conn);**

**cmd.Parameters.AddWithValue("@pattern", baseName + " №%");**

**int count = System.Convert.ToInt32(cmd.ExecuteScalar());**

**return $"{baseName} №{count + 1}";**

**}**

**private int InsertObject(MySqlConnection conn, string name, int clientId, int siteId, decimal budget)**

**{**

**// Здесь siteId — это ID выбранного участка**

**using var cmd = new MySqlCommand("INSERT INTO objects (ObjectName, ClientID, SiteID, Budget) VALUES (@name, @client, @site, @budget); SELECT LAST\_INSERT\_ID();", conn);**

**cmd.Parameters.AddWithValue("@name", name);**

**cmd.Parameters.AddWithValue("@client", clientId);**

**cmd.Parameters.AddWithValue("@site", siteId);**

**cmd.Parameters.AddWithValue("@budget", budget);**

**return System.Convert.ToInt32(cmd.ExecuteScalar());**

**}**

**private int GetOrCreateClient(MySqlConnection conn, int userId)**

**{**

**// Получить имя пользователя**

**string username = null;**

**using (var cmd = new MySqlCommand("SELECT Username FROM users WHERE UserID=@uid", conn))**

**{**

**cmd.Parameters.AddWithValue("@uid", userId);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**username = reader.GetString(0);**

**}**

**if (string.IsNullOrEmpty(username))**

**throw new System.Exception("Не найден пользователь для создания клиента");**

**// Проверить, есть ли клиент с таким именем**

**int clientId = -1;**

**using (var cmd = new MySqlCommand("SELECT ClientID FROM clients WHERE ClientName=@name", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", username);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**clientId = reader.GetInt32(0);**

**}**

**if (clientId != -1)**

**return clientId;**

**// Если нет, создать клиента с именем пользователя**

**using (var cmd = new MySqlCommand("INSERT INTO clients (ClientName) VALUES (@name);", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", username);**

**cmd.ExecuteNonQuery();**

**}**

**// Получить только что созданный ClientID**

**using (var cmd = new MySqlCommand("SELECT ClientID FROM clients WHERE ClientName=@name", conn))**

**{**

**cmd.Parameters.AddWithValue("@name", username);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**return reader.GetInt32(0);**

**}**

**throw new System.Exception("Не удалось создать клиента");**

**}**

**private void LoadMaterials()**

**{**

**Materials = new List<(int, string)>();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT MaterialID, MaterialName FROM materials", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**Materials.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**}**

**private void LoadUserBalance()**

**{**

**UserBalance = 0;**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**int? userId = HttpContext.Session.GetInt32("UserID");**

**if (userId == null)**

**return;**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT Balance FROM users WHERE UserID=@uid", conn);**

**cmd.Parameters.AddWithValue("@uid", userId.Value);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read() && !reader.IsDBNull(0))**

**UserBalance = reader.GetDecimal(0);**

**}**

**private void UpdateUserBalance(MySqlConnection conn, int userId, decimal amount)**

**{**

**using var cmd = new MySqlCommand("UPDATE users SET Balance = Balance - @amount WHERE UserID = @uid", conn);**

**cmd.Parameters.AddWithValue("@amount", amount);**

**cmd.Parameters.AddWithValue("@uid", userId);**

**cmd.ExecuteNonQuery();**

**}**

**private void GetManagementInfoForObject(MySqlConnection conn, int objectId)**

**{**

**string sql = @"**

**SELECT o.ObjectID, o.ObjectName, s.SiteID, s.SiteName, m.ManagementID, m.ManagementName**

**FROM objects o**

**JOIN sites s ON o.SiteID = s.SiteID**

**JOIN managements m ON s.ManagementID = m.ManagementID**

**WHERE o.ObjectID = @objectId**

**";**

**using var cmd = new MySqlCommand(sql, conn);**

**cmd.Parameters.AddWithValue("@objectId", objectId);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**// Process the data as needed**

**}**

**}**

**private (int siteId, int managementId, string siteName, string managementName) GetRandomSiteAndManagement()**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**// Получаем все участки с их управлениями**

**var sites = new List<(int siteId, int managementId, string siteName, string managementName)>();**

**using (var cmd = new MySqlCommand(@"**

**SELECT s.SiteID, s.SiteName, m.ManagementID, m.ManagementName**

**FROM sites s**

**JOIN managements m ON s.ManagementID = m.ManagementID**

**", conn))**

**using (var reader = cmd.ExecuteReader())**

**{**

**while (reader.Read())**

**sites.Add((reader.GetInt32(0), reader.GetInt32(2), reader.GetString(1), reader.GetString(3)));**

**}**

**// Случайно выбираем участок**

**var rnd = new Random();**

**var chosen = sites[rnd.Next(sites.Count)];**

**return chosen;**

**}**

**private void SaveObjectCharacteristics(int objectId, Dictionary<string, string> values)**

**{**

**// Карта: имя поля формы -> CharacteristicID**

**var charMap = new Dictionary<string, int>**

**{**

**{ "Floors", 1 },**

**{ "MaterialType", 2 },**

**{ "ApartmentCount", 3 },**

**{ "SpanType", 4 },**

**{ "Width", 5 },**

**{ "LaneCount", 6 },**

**{ "RoadLength", 7 },**

**{ "BuildingArea", 8 },**

**{ "Capacity", 9 },**

**{ "FoundationType", 10 }**

**};**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**foreach (var kv in charMap)**

**{**

**if (values.TryGetValue(kv.Key, out var val) && !string.IsNullOrWhiteSpace(val))**

**{**

**using var cmd = new MySqlCommand("INSERT INTO object\_characteristics (ObjectID, CharacteristicID, Value) VALUES (@obj, @cid, @val)", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@cid", kv.Value);**

**cmd.Parameters.AddWithValue("@val", val);**

**cmd.ExecuteNonQuery();**

**}**

**}**

**}**

**}**

**Index.cshtml.cs:**

**using Microsoft.AspNetCore.Mvc.RazorPages;**

**using MySql.Data.MySqlClient;**

**using System;**

**using System.Text;**

**using System.Collections.Generic;**

**public class InfoIndexModel : PageModel**

**{**

**public List<string> TableHeaders { get; set; } = new List<string>();**

**public List<List<string>> TableRows { get; set; } = new List<List<string>>();**

**public string QueryResult { get; set; }**

**private string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**public List<(int Id, string Name)> AllSites { get; set; } = new();**

**public List<(int Id, string Name)> AllManagements { get; set; } = new();**

**public List<(int Id, string Name)> AllObjects { get; set; } = new();**

**public List<(int Id, string Name)> AllWorkTypes { get; set; } = new();**

**public List<(int Id, string Name)> AllCrews { get; set; } = new();**

**public List<(int Id, string Name)> AllEquipment { get; set; } = new();**

**public List<(int Id, string Name)> AllPersonnel { get; set; } = new();**

**public List<(int Id, string Name)> AllMaterials { get; set; } = new();**

**public List<LogEntry> Logs { get; set; } = new();**

**public void OnGet()**

**{**

**LoadAllSites();**

**LoadAllManagements();**

**LoadObjects();**

**LoadAllWorkTypes();**

**LoadCrews();**

**LoadEquipment();**

**LoadPersonnel();**

**LoadMaterials();**

**LoadLogs();**

**}**

**public void OnPost()**

**{**

**LoadAllSites();**

**LoadAllManagements();**

**LoadObjects();**

**LoadAllWorkTypes();**

**LoadCrews();**

**LoadEquipment();**

**LoadPersonnel();**

**LoadMaterials();**

**LoadLogs();**

**if (Request.Form.ContainsKey("showMgmt"))**

**ShowTableFromProcedure("GetMgmtAndSitesHeads");**

**else if (Request.Form.ContainsKey("showEngineers"))**

**ShowTableFromProcedure("GetEngineers", new Dictionary<string, object> {**

**{ "p\_SiteID", TryParseInt(Request.Form["siteIdEngineers"]) },**

**{ "p\_ManagementID", TryParseInt(Request.Form["managementIdEngineers"]) }**

**});**

**else if (Request.Form.ContainsKey("showObjectsSchedule"))**

**ShowTableFromProcedure("GetObjectsSchedule", new Dictionary<string, object> {**

**{ "p\_SiteID", TryParseInt(Request.Form["siteIdObjects"]) },**

**{ "p\_ManagementID", TryParseInt(Request.Form["managementIdObjects"]) }**

**});**

**else if (Request.Form.ContainsKey("showCrewComposition"))**

**ShowCrewCompositionSimple(TryParseInt(Request.Form["objectId"]));**

**else if (Request.Form.ContainsKey("showEquipmentByManagement"))**

**ShowTableFromProcedure("GetEquipmentByManagement", new Dictionary<string, object> {**

**{ "p\_ManagementID", TryParseInt(Request.Form["managementId"]) }**

**});**

**else if (Request.Form.ContainsKey("showEquipmentByObjectPeriod"))**

**ShowEquipmentByObjectSimple(TryParseInt(Request.Form["objectIdEq"]));**

**else if (Request.Form.ContainsKey("showScheduleAndEstimate"))**

**ShowTableFromProcedure("GetScheduleAndEstimate", new Dictionary<string, object> {**

**{ "p\_ObjectID", TryParseInt(Request.Form["objectIdSch"]) }**

**});**

**else if (Request.Form.ContainsKey("showObjectReport"))**

**ShowTableFromProcedure("GetObjectReport", new Dictionary<string, object> {**

**{ "p\_ObjectID", TryParseInt(Request.Form["objectIdRep"]) }**

**});**

**else if (Request.Form.ContainsKey("showObjectsByWorkTypePeriod"))**

**ShowTableFromProcedure("GetObjectsByWorkTypePeriod", new Dictionary<string, object> {**

**{ "p\_ManagementID", TryParseInt(Request.Form["managementIdObj"]) },**

**{ "p\_WorkTypeID", TryParseInt(Request.Form["workTypeIdObj"]) },**

**{ "p\_From", TryParseDate(Request.Form["fromObj"]) ?? DateTime.Today },**

**{ "p\_To", TryParseDate(Request.Form["toObj"]) ?? DateTime.Today.AddDays(7) }**

**});**

**else if (Request.Form.ContainsKey("showOverdueWorkTypes"))**

**ShowTableFromProcedure("GetOverdueWorkTypes", new Dictionary<string, object> {**

**{ "p\_SiteID", TryParseInt(Request.Form["siteIdOver"]) },**

**{ "p\_ManagementID", TryParseInt(Request.Form["managementIdOver"]) }**

**});**

**else if (Request.Form.ContainsKey("showOverbudgetMaterials"))**

**ShowTableFromProcedure("GetOverbudgetMaterials", new Dictionary<string, object> {**

**{ "p\_SiteID", TryParseInt(Request.Form["siteIdMat"]) },**

**{ "p\_ManagementID", TryParseInt(Request.Form["managementIdMat"]) }**

**});**

**else if (Request.Form.ContainsKey("showCrewWorksPeriod"))**

**ShowTableFromProcedure("GetCrewWorksPeriod", new Dictionary<string, object> {**

**{ "p\_CrewID", TryParseInt(Request.Form["crewId"]) },**

**{ "p\_From", TryParseDate(Request.Form["fromCrew"]) ?? DateTime.Today },**

**{ "p\_To", TryParseDate(Request.Form["toCrew"]) ?? DateTime.Today.AddDays(7) }**

**});**

**else if (Request.Form.ContainsKey("showCrewsByWorkTypePeriod"))**

**ShowTableFromProcedure("GetCrewsByWorkTypePeriod", new Dictionary<string, object> {**

**{ "p\_WorkTypeID", TryParseInt(Request.Form["workTypeIdCrew"]) },**

**{ "p\_From", TryParseDate(Request.Form["fromCrewType"]) ?? DateTime.Today },**

**{ "p\_To", TryParseDate(Request.Form["toCrewType"]) ?? DateTime.Today.AddDays(7) }**

**});**

**}**

**private int? TryParseInt(string val)**

**{**

**if (int.TryParse(val, out int result))**

**return result;**

**return null;**

**}**

**private DateTime? TryParseDate(string val)**

**{**

**if (DateTime.TryParse(val, out DateTime dt))**

**return dt;**

**return null;**

**}**

**private void ShowTableFromProcedure(string procName, Dictionary<string, object> parameters = null)**

**{**

**TableHeaders = new List<string>();**

**TableRows = new List<List<string>>();**

**QueryResult = null;**

**try**

**{**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand(procName, conn);**

**cmd.CommandType = System.Data.CommandType.StoredProcedure;**

**if (parameters != null)**

**{**

**foreach (var p in parameters)**

**cmd.Parameters.AddWithValue(p.Key, p.Value ?? DBNull.Value);**

**}**

**using var reader = cmd.ExecuteReader();**

**// Для GetScheduleAndEstimate выводим оба результата (график и смета)**

**if (procName == "GetScheduleAndEstimate")**

**{**

**// График**

**TableHeaders = new List<string>();**

**for (int i = 0; i < reader.FieldCount; i++)**

**TableHeaders.Add(reader.GetName(i));**

**TableRows = new List<List<string>>();**

**bool hasRows = false;**

**while (reader.Read())**

**{**

**hasRows = true;**

**var row = new List<string>();**

**for (int i = 0; i < reader.FieldCount; i++)**

**row.Add(reader[i]?.ToString() ?? "");**

**TableRows.Add(row);**

**}**

**if (!hasRows)**

**{**

**TableHeaders = null;**

**TableRows = null;**

**QueryResult = "Нет данных для отображения.";**

**}**

**// Смета (вторая таблица)**

**if (reader.NextResult())**

**{**

**var estimateHeaders = new List<string>();**

**for (int i = 0; i < reader.FieldCount; i++)**

**estimateHeaders.Add(reader.GetName(i));**

**var estimateRows = new List<List<string>>();**

**bool hasEstimate = false;**

**while (reader.Read())**

**{**

**hasEstimate = true;**

**var row = new List<string>();**

**for (int i = 0; i < reader.FieldCount; i++)**

**row.Add(reader[i]?.ToString() ?? "");**

**estimateRows.Add(row);**

**}**

**if (hasEstimate)**

**{**

**// Сохраняем обе таблицы для вывода**

**QueryResult = "<b>График работ:</b>";**

**QueryResult += TableToHtml(TableHeaders, TableRows);**

**QueryResult += "<br/><b>Смета:</b>";**

**QueryResult += TableToHtml(estimateHeaders, estimateRows);**

**TableHeaders = null;**

**TableRows = null;**

**}**

**}**

**return;**

**}**

**// Для GetOverdueWorkTypes: всегда выводить таблицу, даже если пусто**

**if (procName == "GetOverdueWorkTypes")**

**{**

**for (int i = 0; i < reader.FieldCount; i++)**

**TableHeaders.Add(reader.GetName(i));**

**bool hasRows = false;**

**while (reader.Read())**

**{**

**hasRows = true;**

**var row = new List<string>();**

**for (int i = 0; i < reader.FieldCount; i++)**

**row.Add(reader[i]?.ToString() ?? "");**

**TableRows.Add(row);**

**}**

**if (!hasRows)**

**{**

**QueryResult = "Нет работ с превышением сроков.";**

**}**

**return;**

**}**

**// Обычный режим (одна таблица)**

**for (int i = 0; i < reader.FieldCount; i++)**

**TableHeaders.Add(reader.GetName(i));**

**bool hasRows2 = false;**

**while (reader.Read())**

**{**

**hasRows2 = true;**

**var row = new List<string>();**

**for (int i = 0; i < reader.FieldCount; i++)**

**row.Add(reader[i]?.ToString() ?? "");**

**TableRows.Add(row);**

**}**

**if (!hasRows2)**

**{**

**TableHeaders = null;**

**TableRows = null;**

**QueryResult = "Нет данных для отображения.";**

**}**

**}**

**catch (System.Exception ex)**

**{**

**TableHeaders = null;**

**TableRows = null;**

**QueryResult = "Ошибка: " + ex.Message;**

**}**

**}**

**// Вспомогательный метод для вывода двух таблиц в QueryResult**

**private string TableToHtml(List<string> headers, List<List<string>> rows)**

**{**

**if (headers == null || rows == null || rows.Count == 0)**

**return "<div>Нет данных</div>";**

**var html = "<table border='1' style='margin-top:10px;'>";**

**html += "<tr>";**

**foreach (var h in headers)**

**html += $"<th>{h}</th>";**

**html += "</tr>";**

**foreach (var row in rows)**

**{**

**html += "<tr>";**

**foreach (var cell in row)**

**html += $"<td>{cell}</td>";**

**html += "</tr>";**

**}**

**html += "</table>";**

**return html;**

**}**

**private void ShowCrewCompositionSimple(int? objectId)**

**{**

**TableHeaders = new List<string> { "Бригада", "ФИО", "Должность" };**

**TableRows = new List<List<string>>();**

**QueryResult = null;**

**if (objectId == null)**

**{**

**QueryResult = "Не выбран объект.";**

**return;**

**}**

**try**

**{**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand(@"**

**SELECT DISTINCT c.CrewName, p.FullName, pos.PositionName**

**FROM v\_workassignments wa**

**JOIN v\_crews c ON c.CrewID = wa.CrewID**

**JOIN v\_crewmembers cm ON cm.CrewID = c.CrewID**

**JOIN v\_personnel p ON p.PersonnelID = cm.PersonnelID**

**JOIN v\_positions pos ON pos.PositionID = p.PositionID**

**WHERE wa.ObjectID = @obj**

**ORDER BY c.CrewName, p.FullName**

**", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId.Value);**

**using var reader = cmd.ExecuteReader();**

**bool hasRows = false;**

**while (reader.Read())**

**{**

**hasRows = true;**

**TableRows.Add(new List<string>**

**{**

**reader.GetString(0),**

**reader.GetString(1),**

**reader.GetString(2)**

**});**

**}**

**if (!hasRows)**

**{**

**TableHeaders = null;**

**TableRows = null;**

**QueryResult = "Нет данных для отображения.";**

**}**

**}**

**catch (System.Exception ex)**

**{**

**TableHeaders = null;**

**TableRows = null;**

**QueryResult = "Ошибка: " + ex.Message;**

**}**

**}**

**private void ShowEquipmentByObjectSimple(int? objectId)**

**{**

**TableHeaders = new List<string> { "Техника", "Дата начала", "Дата окончания" };**

**TableRows = new List<List<string>>();**

**QueryResult = null;**

**if (objectId == null)**

**{**

**QueryResult = "Не выбран объект.";**

**return;**

**}**

**try**

**{**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand(@"**

**SELECT e.EquipmentName, ea.StartDate, ea.EndDate**

**FROM v\_equipmentassignments ea**

**JOIN v\_equipment e ON e.EquipmentID = ea.EquipmentID**

**WHERE ea.ObjectID = @obj**

**ORDER BY ea.StartDate**

**", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId.Value);**

**using var reader = cmd.ExecuteReader();**

**bool hasRows = false;**

**while (reader.Read())**

**{**

**hasRows = true;**

**TableRows.Add(new List<string>**

**{**

**reader.GetString(0),**

**reader.IsDBNull(1) ? "" : reader.GetDateTime(1).ToShortDateString(),**

**reader.IsDBNull(2) ? "техника еще в работе" : reader.GetDateTime(2).ToShortDateString()**

**});**

**}**

**if (!hasRows)**

**{**

**TableHeaders = null;**

**TableRows = null;**

**QueryResult = "Нет данных для отображения.";**

**}**

**}**

**catch (System.Exception ex)**

**{**

**TableHeaders = null;**

**TableRows = null;**

**QueryResult = "Ошибка: " + ex.Message;**

**}**

**}**

**private void LoadAllSites()**

**{**

**AllSites.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT SiteID, SiteName FROM v\_sites", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllSites.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllManagements()**

**{**

**AllManagements.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT ManagementID, ManagementName FROM v\_managements", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllManagements.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadObjects()**

**{**

**AllObjects.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT ObjectID, ObjectName FROM v\_objects", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllObjects.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllWorkTypes()**

**{**

**AllWorkTypes.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT WorkTypeID, WorkTypeName FROM v\_worktypes", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllWorkTypes.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadCrews()**

**{**

**AllCrews.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT CrewID, CrewName FROM v\_crews", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllCrews.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadEquipment()**

**{**

**AllEquipment.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT EquipmentID, EquipmentName FROM v\_equipment", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllEquipment.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadPersonnel()**

**{**

**AllPersonnel.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT PersonnelID, FullName FROM v\_personnel", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllPersonnel.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadMaterials()**

**{**

**AllMaterials.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT MaterialID, MaterialName FROM v\_materials\_consumption GROUP BY MaterialID, MaterialName", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllMaterials.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadLogs()**

**{**

**Logs.Clear();**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT LogID, Username, Action, ActionTime FROM v\_logs", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**Logs.Add(new LogEntry**

**{**

**LogID = reader.GetInt32(0),**

**Username = reader.GetString(1),**

**Action = reader.GetString(2),**

**ActionTime = reader.GetDateTime(3)**

**});**

**}**

**}**

**public class LogEntry**

**{**

**public int LogID { get; set; }**

**public string Username { get; set; }**

**public string Action { get; set; }**

**public DateTime ActionTime { get; set; }**

**}**

**Admin:**

**Object.cshtml:**

**@page**

**@model ObjectsModel**

**@{**

**ViewData["Title"] = "Управление объектами";**

**}**

**<h2>Панель администратора: Управление объектами</h2>**

**<form method="post" class="mb-3">**

**<div class="input-group mb-3">**

**<input type="text" name="search" value="@Model.SearchQuery" class="form-control" placeholder="Поиск объекта по названию или номеру..." />**

**<button type="submit" class="btn btn-primary">Поиск</button>**

**</div>**

**</form>**

**@if (!string.IsNullOrEmpty(Model.Message))**

**{**

**<div class="alert alert-warning">@Model.Message</div>**

**}**

**<!-- Кнопка для показа логов -->**

**<form method="post" class="mb-3">**

**<button type="submit" name="showLogs" class="btn btn-secondary">Показать логи</button>**

**</form>**

**<!-- Добавить руководителя участка -->**

**<div class="mb-4">**

**<h4>Назначить руководителя участка</h4>**

**<form method="post" class="row g-2 align-items-end">**

**<div class="col-auto">**

**<label>Участок:</label>**

**<select name="assignSiteId" class="form-select">**

**<option value="">--Выбрать участок--</option>**

**@foreach (var s in Model.AllSites)**

**{**

**<option value="@s.Id">@s.Name</option>**

**}**

**</select>**

**</div>**

**<div class="col-auto">**

**<label>Сотрудник:</label>**

**<select name="assignHeadPersonnelId" class="form-select">**

**<option value="">--Выбрать сотрудника--</option>**

**@foreach (var p in Model.AllPersonnel)**

**{**

**<option value="@p.Id">@p.Name</option>**

**}**

**</select>**

**</div>**

**<div class="col-auto">**

**<button type="submit" name="assignSiteHead" class="btn btn-success">Назначить руководителя участка</button>**

**</div>**

**</form>**

**@if (!string.IsNullOrEmpty(Model.AssignSiteHeadMessage))**

**{**

**<div class="alert alert-info mt-2">@Model.AssignSiteHeadMessage</div>**

**}**

**</div>**

**<!-- Добавить нового сотрудника -->**

**<div class="mb-4">**

**<h4>Добавить нового сотрудника</h4>**

**<form method="post" class="row g-2 align-items-end">**

**<div class="col-auto">**

**<input type="text" name="newPersonnelName" class="form-control" placeholder="ФИО" />**

**</div>**

**<div class="col-auto">**

**<select name="newPersonnelPositionId" class="form-select" id="newPersonnelPositionId" onchange="onPositionChange()">**

**<option value="">--Должность--</option>**

**@foreach (var pos in Model.AllPositions)**

**{**

**<option value="@pos.Id">@pos.Name</option>**

**}**

**</select>**

**</div>**

**<div class="col-auto">**

**<select name="newPersonnelManagementId" class="form-select" id="newPersonnelManagementId">**

**<option value="">--Управление--</option>**

**@foreach (var m in Model.AllManagements)**

**{**

**<option value="@m.Id">@m.Name</option>**

**}**

**</select>**

**</div>**

**<div class="col-auto" id="siteSelectContainer" style="display:none;">**

**<select name="newPersonnelSiteId" class="form-select" id="newPersonnelSiteId">**

**<option value="">--Участок--</option>**

**@foreach (var s in Model.AllSites)**

**{**

**<option value="@s.Id">@s.Name</option>**

**}**

**</select>**

**</div>**

**<button type="submit" name="addPersonnel" class="btn btn-primary">Добавить сотрудника</button>**

**</form>**

**@if (!string.IsNullOrEmpty(Model.AddPersonnelMessage))**

**{**

**<div class="alert alert-info mt-2">@Model.AddPersonnelMessage</div>**

**}**

**</div>**

**<script>**

**function onPositionChange() {**

**var posSelect = document.getElementById('newPersonnelPositionId');**

**var siteContainer = document.getElementById('siteSelectContainer');**

**var selectedText = posSelect.options[posSelect.selectedIndex].text;**

**if (selectedText === 'Начальник участка') {**

**siteContainer.style.display = '';**

**} else {**

**siteContainer.style.display = 'none';**

**document.getElementById('newPersonnelSiteId').value = '';**

**}**

**}**

**window.onload = function() {**

**onPositionChange();**

**};**

**</script>**

**@if (Model.Results != null && Model.Results.Any())**

**{**

**<table class="table table-bordered">**

**<thead>**

**<tr>**

**<th>ID</th>**

**<th>Название объекта</th>**

**<th>Участок</th>**

**<th>Управление</th>**

**<th>Бюджет</th>**

**<th>Бригады</th>**

**<th>Техника</th>**

**<th>Действия</th>**

**</tr>**

**</thead>**

**<tbody>**

**@foreach (var obj in Model.Results)**

**{**

**<tr>**

**<td>@obj.ObjectID</td>**

**<td>@obj.ObjectName</td>**

**<td>@obj.SiteName</td>**

**<td>@obj.ManagementName</td>**

**<td>@(obj.Budget.HasValue ? obj.Budget.Value.ToString("N2") : "-")</td>**

**<td>**

**@foreach (var crew in obj.Crews)**

**{**

**<div>@crew</div>**

**}**

**</td>**

**<td>**

**@foreach (var eq in obj.Equipment)**

**{**

**<div>@eq</div>**

**}**

**</td>**

**<td>**

**<form method="post" style="display:inline;">**

**<input type="hidden" name="selectObject" value="1" />**

**<input type="hidden" name="selectObjectId" value="@obj.ObjectID" />**

**<button type="submit" class="btn btn-sm btn-outline-primary">Назначить</button>**

**</form>**

**</td>**

**</tr>**

**@if (Model.SelectedObjectId == obj.ObjectID)**

**{**

**<tr>**

**<td colspan="8">**

**<div style="padding:10px;background:#f8f9fa;border-radius:8px;">**

**<form method="post" style="margin-bottom:10px;">**

**<input type="hidden" name="assignCrewObjectId" value="@obj.ObjectID" />**

**<div class="mb-2">**

**<label>Назначить существующую бригаду:</label>**

**<select name="crewId" class="form-select" style="width:auto;display:inline-block;">**

**<option value="">--Выбрать--</option>**

**@foreach (var c in Model.AllCrews)**

**{**

**<option value="@c.Id">@c.Name</option>**

**}**

**</select>**

**<span style="margin-left:10px;">Виды работ:</span>**

**@foreach (var w in Model.AllWorkTypes)**

**{**

**<label style="margin-left:5px;">**

**<input type="checkbox" name="worktype\_@w.Id" /> @w.Name**

**</label>**

**}**

**<button type="submit" name="assignCrew" class="btn btn-sm btn-success" style="margin-left:10px;">Назначить</button>**

**</div>**

**</form>**

**<form method="post" style="margin-bottom:10px;">**

**<input type="hidden" name="assignCrewObjectId" value="@obj.ObjectID" />**

**<div class="mb-2">**

**<label>Создать и назначить новую бригаду:</label>**

**<input type="text" name="newCrewName" placeholder="Название бригады" class="form-control" style="width:auto;display:inline-block;" />**

**<span style="margin-left:10px;">Состав:</span>**

**@foreach (var p in Model.AllPersonnel)**

**{**

**<label style="margin-left:5px;">**

**<input type="checkbox" name="person\_@p.Id" /> @p.Name**

**</label>**

**}**

**<span style="margin-left:10px;">Виды работ:</span>**

**@foreach (var w in Model.AllWorkTypes)**

**{**

**<label style="margin-left:5px;">**

**<input type="checkbox" name="worktype\_@w.Id" /> @w.Name**

**</label>**

**}**

**<button type="submit" name="assignNewCrew" class="btn btn-sm btn-success" style="margin-left:10px;">Создать и назначить</button>**

**</div>**

**</form>**

**<form method="post" style="margin-bottom:10px;">**

**<input type="hidden" name="assignEquipmentObjectId" value="@obj.ObjectID" />**

**<label>Назначить технику:</label>**

**<select name="equipmentId" class="form-select" style="width:auto;display:inline-block;">**

**<option value="">--Выбрать--</option>**

**@foreach (var eq in Model.AllEquipment)**

**{**

**<option value="@eq.Id">@eq.Name</option>**

**}**

**</select>**

**<button type="submit" name="assignEquipment" class="btn btn-sm btn-success" style="margin-left:10px;">Назначить</button>**

**</form>**

**<form method="post" style="margin-bottom:10px;">**

**<input type="hidden" name="addScheduleObjectId" value="@obj.ObjectID" />**

**<label>Добавить этап работ:</label>**

**<select name="scheduleWorkTypeId" class="form-select" style="width:auto;display:inline-block;">**

**<option value="">--Вид работ--</option>**

**@foreach (var w in Model.AllWorkTypes)**

**{**

**<option value="@w.Id">@w.Name</option>**

**}**

**</select>**

**<input type="date" name="scheduleStartDate" class="form-control" style="width:auto;display:inline-block;" />**

**<input type="date" name="scheduleEndDate" class="form-control" style="width:auto;display:inline-block;" />**

**<button type="submit" name="addSchedule" class="btn btn-sm btn-success" style="margin-left:10px;">Добавить</button>**

**</form>**

**<form method="post" style="margin-bottom:10px;">**

**<input type="hidden" name="addMaterialObjectId" value="@obj.ObjectID" />**

**<label>Добавить расход материала:</label>**

**<select name="materialId" class="form-select" style="width:auto;display:inline-block;">**

**<option value="">--Материал--</option>**

**@foreach (var m in Model.AllMaterials)**

**{**

**<option value="@m.Id">@m.Name</option>**

**}**

**</select>**

**<input type="number" step="0.01" name="estimatedQty" placeholder="Смета" class="form-control" style="width:100px;display:inline-block;" />**

**<input type="number" step="0.01" name="consumedQty" placeholder="Факт" class="form-control" style="width:100px;display:inline-block;" />**

**<button type="submit" name="addMaterialConsumption" class="btn btn-sm btn-success" style="margin-left:10px;">Добавить</button>**

**</form>**

**<form method="post" style="margin-bottom:10px;">**

**<input type="hidden" name="updateScheduleObjectId" value="@obj.ObjectID" />**

**<label>Фактическая дата завершения этапа работ:</label>**

**<select name="updateScheduleWorkTypeId" class="form-select" style="width:auto;display:inline-block;">**

**<option value="">--Вид работ--</option>**

**@foreach (var w in Model.AllWorkTypes)**

**{**

**<option value="@w.Id">@w.Name</option>**

**}**

**</select>**

**<input type="date" name="scheduleActualEndDate" class="form-control" style="width:auto;display:inline-block;" />**

**<button type="submit" name="updateScheduleEnd" class="btn btn-sm btn-success" style="margin-left:10px;">Сохранить</button>**

**</form>**

**@if (!string.IsNullOrEmpty(Model.AssignMessage))**

**{**

**<div class="alert alert-info" style="margin-top:10px;">@Model.AssignMessage</div>**

**}**

**</div>**

**</td>**

**</tr>**

**}**

**}**

**</tbody>**

**</table>**

**}**

**@if (Model.Logs != null && Model.Logs.Any())**

**{**

**<h4>Журнал действий (логи)</h4>**

**<table class="table table-bordered">**

**<thead>**

**<tr>**

**<th>LogID</th>**

**<th>Пользователь</th>**

**<th>Action</th>**

**<th>ActionTime</th>**

**</tr>**

**</thead>**

**<tbody>**

**@foreach (var log in Model.Logs)**

**{**

**<tr>**

**<td>@log.LogID</td>**

**<td>@log.Username</td>**

**<td>@log.Action</td>**

**<td>@log.ActionTime</td>**

**</tr>**

**}**

**</tbody>**

**</table>**

**}**

**Order.cshtml.cs:**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.AspNetCore.Mvc.RazorPages;**

**using MySql.Data.MySqlClient;**

**using System.Collections.Generic;**

**public class ObjectsModel : PageModel**

**{**

**public class ObjectInfo**

**{**

**public int ObjectID { get; set; }**

**public string ObjectName { get; set; }**

**public string SiteName { get; set; }**

**public string ManagementName { get; set; }**

**public decimal? Budget { get; set; }**

**public List<string> Crews { get; set; } = new();**

**public List<string> Equipment { get; set; } = new();**

**}**

**public class LogEntry**

**{**

**public int LogID { get; set; }**

**public string Username { get; set; }**

**public string Action { get; set; }**

**public DateTime ActionTime { get; set; }**

**}**

**public List<ObjectInfo> Results { get; set; } = new();**

**public string SearchQuery { get; set; }**

**public string Message { get; set; }**

**public List<(int Id, string Name)> AllCrews { get; set; } = new();**

**public List<(int Id, string Name)> AllEquipment { get; set; } = new();**

**public List<(int Id, string Name)> AllPersonnel { get; set; } = new();**

**public List<(int Id, string Name)> AllWorkTypes { get; set; } = new();**

**public List<(int Id, string Name)> AllMaterials { get; set; } = new();**

**public List<(int Id, string Name)> AllSites { get; set; } = new();**

**public List<(int Id, string Name)> AllPositions { get; set; } = new();**

**public List<(int Id, string Name)> AllManagements { get; set; } = new();**

**public string AssignSiteHeadMessage { get; set; }**

**public string AddPersonnelMessage { get; set; }**

**public int? SelectedObjectId { get; set; }**

**public string AssignMessage { get; set; }**

**public List<LogEntry> Logs { get; set; } = new();**

**public void OnGet()**

**{**

**LoadAllCrews();**

**LoadAllEquipment();**

**LoadAllPersonnel();**

**LoadAllWorkTypes();**

**LoadAllMaterials();**

**LoadAllSites();**

**LoadAllPositions();**

**LoadAllManagements();**

**Logs = new List<LogEntry>();**

**}**

**public void OnPost()**

**{**

**// Если отправлена форма поиска, только тогда выполнять поиск**

**if (Request.Form.ContainsKey("search"))**

**{**

**SearchQuery = Request.Form["search"];**

**Results = new List<ObjectInfo>();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**try**

**{**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using (var cmd = new MySqlCommand(@"**

**SELECT ObjectID, ObjectName, Budget, SiteName, ManagementName**

**FROM objects**

**WHERE ObjectName LIKE @q OR ObjectID LIKE @q**

**ORDER BY ObjectID**

**", conn))**

**{**

**cmd.Parameters.AddWithValue("@q", "%" + (SearchQuery ?? "") + "%");**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**Results.Add(new ObjectInfo**

**{**

**ObjectID = reader.GetInt32(0),**

**ObjectName = reader.GetString(1),**

**Budget = reader.IsDBNull(2) ? null : reader.GetDecimal(2),**

**SiteName = reader.IsDBNull(3) ? "" : reader.GetString(3),**

**ManagementName = reader.IsDBNull(4) ? "" : reader.GetString(4)**

**});**

**}**

**}**

**// Для каждого объекта получить бригады и технику**

**foreach (var obj in Results)**

**{**

**// Бригады**

**using (var cmd = new MySqlCommand(@"**

**SELECT CrewName**

**FROM crews**

**WHERE CrewID IN (SELECT CrewID FROM workassignments WHERE ObjectID = @oid)**

**", conn))**

**{**

**cmd.Parameters.AddWithValue("@oid", obj.ObjectID);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**var crew = reader.IsDBNull(0) ? "Бригада еще не назначена" : reader.GetString(0);**

**if (!obj.Crews.Contains(crew))**

**obj.Crews.Add(crew);**

**}**

**if (obj.Crews.Count == 0)**

**obj.Crews.Add("Бригада еще не назначена");**

**}**

**// Техника**

**using (var cmd = new MySqlCommand(@"**

**SELECT EquipmentName**

**FROM equipment**

**WHERE ObjectID = @oid**

**", conn))**

**{**

**cmd.Parameters.AddWithValue("@oid", obj.ObjectID);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**var eq = reader.IsDBNull(0) ? "Техника еще не назначена" : reader.GetString(0);**

**if (!obj.Equipment.Contains(eq))**

**obj.Equipment.Add(eq);**

**}**

**if (obj.Equipment.Count == 0)**

**obj.Equipment.Add("Техника еще не назначена");**

**}**

**}**

**}**

**catch (System.Exception ex)**

**{**

**Message = "Ошибка поиска: " + ex.Message;**

**}**

**if (Results.Count == 0)**

**Message = "Объекты не найдены.";**

**return; // <--- добавьте return, чтобы не сбрасывался SelectedObjectId после поиска**

**}**

**LoadAllCrews();**

**LoadAllEquipment();**

**LoadAllPersonnel();**

**LoadAllWorkTypes();**

**LoadAllMaterials();**

**LoadAllSites();**

**LoadAllPositions();**

**LoadAllManagements();**

**// Обработка выбора объекта для назначения**

**if (Request.Form.ContainsKey("selectObject"))**

**{**

**if (int.TryParse(Request.Form["selectObjectId"], out int selId))**

**SelectedObjectId = selId;**

**// После выбора объекта нужно снова заполнить Results, чтобы меню назначения появилось**

**// Показываем все объекты (или только выбранный, если хотите)**

**Results = new List<ObjectInfo>();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using (var cmd = new MySqlCommand(@"**

**SELECT ObjectID, ObjectName, Budget, SiteName, ManagementName**

**FROM objects**

**ORDER BY ObjectID**

**", conn))**

**{**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**Results.Add(new ObjectInfo**

**{**

**ObjectID = reader.GetInt32(0),**

**ObjectName = reader.GetString(1),**

**Budget = reader.IsDBNull(2) ? null : reader.GetDecimal(2),**

**SiteName = reader.IsDBNull(3) ? "" : reader.GetString(3),**

**ManagementName = reader.IsDBNull(4) ? "" : reader.GetString(4)**

**});**

**}**

**}**

**// Для каждого объекта получить бригады и технику**

**foreach (var obj in Results)**

**{**

**using (var cmd = new MySqlCommand(@"**

**SELECT CrewName**

**FROM crews**

**WHERE CrewID IN (SELECT CrewID FROM workassignments WHERE ObjectID = @oid)**

**", conn))**

**{**

**cmd.Parameters.AddWithValue("@oid", obj.ObjectID);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**var crew = reader.IsDBNull(0) ? "Бригада еще не назначена" : reader.GetString(0);**

**if (!obj.Crews.Contains(crew))**

**obj.Crews.Add(crew);**

**}**

**if (obj.Crews.Count == 0)**

**obj.Crews.Add("Бригада еще не назначена");**

**}**

**using (var cmd = new MySqlCommand(@"**

**SELECT EquipmentName**

**FROM equipment**

**WHERE ObjectID = @oid**

**", conn))**

**{**

**cmd.Parameters.AddWithValue("@oid", obj.ObjectID);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**var eq = reader.IsDBNull(0) ? "Техника еще не назначена" : reader.GetString(0);**

**if (!obj.Equipment.Contains(eq))**

**obj.Equipment.Add(eq);**

**}**

**if (obj.Equipment.Count == 0)**

**obj.Equipment.Add("Техника еще не назначена");**

**}**

**}**

**return;**

**}**

**// Назначить существующую бригаду**

**if (Request.Form.ContainsKey("assignCrew"))**

**{**

**if (int.TryParse(Request.Form["assignCrewObjectId"], out int objectId) && int.TryParse(Request.Form["crewId"], out int crewId))**

**{**

**var workTypeIds = new List<int>();**

**foreach (var key in Request.Form.Keys)**

**{**

**if (key.StartsWith("worktype\_") && Request.Form[key] == "on")**

**{**

**int workTypeId = int.Parse(key.Replace("worktype\_", ""));**

**workTypeIds.Add(workTypeId);**

**}**

**}**

**if (workTypeIds.Count == 0)**

**{**

**AssignMessage = "Выберите хотя бы один вид работ!";**

**SelectedObjectId = objectId;**

**return;**

**}**

**foreach (var workTypeId in workTypeIds)**

**{**

**AssignCrewToObject(objectId, crewId, workTypeId);**

**}**

**AssignMessage = "Бригада и виды работ назначены!";**

**SelectedObjectId = objectId;**

**}**

**}**

**// Создать новую бригаду и назначить**

**if (Request.Form.ContainsKey("assignNewCrew"))**

**{**

**if (int.TryParse(Request.Form["assignCrewObjectId"], out int objectId))**

**{**

**string newCrewName = Request.Form["newCrewName"];**

**if (string.IsNullOrWhiteSpace(newCrewName))**

**{**

**AssignMessage = "Введите название новой бригады!";**

**SelectedObjectId = objectId;**

**return;**

**}**

**int crewId = CreateCrew(newCrewName);**

**foreach (var key in Request.Form.Keys)**

**{**

**if (key.StartsWith("person\_") && Request.Form[key] == "on")**

**{**

**int personnelId = int.Parse(key.Replace("person\_", ""));**

**AddCrewMember(crewId, personnelId);**

**}**

**}**

**var workTypeIds = new List<int>();**

**foreach (var key in Request.Form.Keys)**

**{**

**if (key.StartsWith("worktype\_") && Request.Form[key] == "on")**

**{**

**int workTypeId = int.Parse(key.Replace("worktype\_", ""));**

**workTypeIds.Add(workTypeId);**

**}**

**}**

**if (workTypeIds.Count == 0)**

**{**

**AssignMessage = "Выберите хотя бы один вид работ!";**

**SelectedObjectId = objectId;**

**return;**

**}**

**foreach (var workTypeId in workTypeIds)**

**{**

**AssignCrewToObject(objectId, crewId, workTypeId);**

**}**

**AssignMessage = "Новая бригада создана и назначена!";**

**SelectedObjectId = objectId;**

**}**

**}**

**// Назначить технику**

**if (Request.Form.ContainsKey("assignEquipment"))**

**{**

**if (int.TryParse(Request.Form["assignEquipmentObjectId"], out int objectId) && int.TryParse(Request.Form["equipmentId"], out int equipmentId))**

**{**

**AssignEquipmentToObject(objectId, equipmentId);**

**AssignMessage = "Техника назначена!";**

**SelectedObjectId = objectId;**

**}**

**}**

**// Добавление этапа работ (workschedules)**

**if (Request.Form.ContainsKey("addSchedule"))**

**{**

**if (int.TryParse(Request.Form["addScheduleObjectId"], out int objectId) &&**

**int.TryParse(Request.Form["scheduleWorkTypeId"], out int workTypeId) &&**

**DateTime.TryParse(Request.Form["scheduleStartDate"], out DateTime startDate) &&**

**DateTime.TryParse(Request.Form["scheduleEndDate"], out DateTime endDate))**

**{**

**AddWorkSchedule(objectId, workTypeId, startDate, endDate);**

**AssignMessage = "Этап работ добавлен!";**

**SelectedObjectId = objectId;**

**}**

**else**

**{**

**AssignMessage = "Заполните все поля для этапа работ!";**

**}**

**return;**

**}**

**// Продлить/завершить этап работ**

**if (Request.Form.ContainsKey("updateScheduleEnd"))**

**{**

**if (int.TryParse(Request.Form["updateScheduleObjectId"], out int objectId) &&**

**int.TryParse(Request.Form["updateScheduleWorkTypeId"], out int workTypeId) &&**

**DateTime.TryParse(Request.Form["scheduleActualEndDate"], out DateTime actualEnd))**

**{**

**UpdateWorkScheduleEnd(objectId, workTypeId, actualEnd);**

**AssignMessage = "Фактическая дата окончания этапа работ сохранена!";**

**SelectedObjectId = objectId;**

**}**

**return;**

**}**

**// Продлить/завершить назначение техники**

**if (Request.Form.ContainsKey("updateEquipmentEnd"))**

**{**

**if (int.TryParse(Request.Form["updateEquipmentObjectId"], out int objectId) &&**

**int.TryParse(Request.Form["updateEquipmentId"], out int equipmentId) &&**

**DateTime.TryParse(Request.Form["equipmentActualEndDate"], out DateTime actualEnd))**

**{**

**UpdateEquipmentAssignmentEnd(objectId, equipmentId, actualEnd);**

**AssignMessage = "Фактическая дата окончания работы техники сохранена!";**

**SelectedObjectId = objectId;**

**}**

**return;**

**}**

**// Добавить расход материала**

**if (Request.Form.ContainsKey("addMaterialConsumption"))**

**{**

**if (int.TryParse(Request.Form["addMaterialObjectId"], out int objectId) &&**

**int.TryParse(Request.Form["materialId"], out int materialId) &&**

**decimal.TryParse(Request.Form["estimatedQty"], out decimal estimatedQty) &&**

**decimal.TryParse(Request.Form["consumedQty"], out decimal consumedQty))**

**{**

**AddOrUpdateMaterialConsumption(objectId, materialId, estimatedQty, consumedQty);**

**AssignMessage = "Расход материала добавлен!";**

**SelectedObjectId = objectId;**

**}**

**return;**

**}**

**// Добавить нового сотрудника**

**if (Request.Form.ContainsKey("addPersonnel"))**

**{**

**string name = Request.Form["newPersonnelName"];**

**int.TryParse(Request.Form["newPersonnelPositionId"], out int posId);**

**int.TryParse(Request.Form["newPersonnelManagementId"], out int mgmtId);**

**int.TryParse(Request.Form["newPersonnelSiteId"], out int siteId);**

**if (string.IsNullOrWhiteSpace(name) || posId == 0 || mgmtId == 0)**

**{**

**AddPersonnelMessage = "Заполните все поля!";**

**}**

**else**

**{**

**// Если выбран "Начальник участка", siteId обязателен**

**string posName = AllPositions.FirstOrDefault(p => p.Id == posId).Name;**

**if (posName == "Начальник участка" && siteId == 0)**

**{**

**AddPersonnelMessage = "Выберите участок для начальника участка!";**

**}**

**else**

**{**

**AddPersonnel(name, posId, mgmtId, siteId == 0 ? (int?)null : siteId);**

**AddPersonnelMessage = "Сотрудник добавлен!";**

**}**

**}**

**return;**

**}**

**// Назначить руководителя участка**

**if (Request.Form.ContainsKey("assignSiteHead"))**

**{**

**int.TryParse(Request.Form["assignSiteId"], out int siteId);**

**int.TryParse(Request.Form["assignHeadPersonnelId"], out int personnelId);**

**if (siteId == 0 || personnelId == 0)**

**{**

**AssignSiteHeadMessage = "Выберите участок и сотрудника!";**

**}**

**else**

**{**

**AssignSiteHead(siteId, personnelId);**

**AssignSiteHeadMessage = "Руководитель участка назначен!";**

**}**

**return;**

**}**

**// Показать логи**

**if (Request.Form.ContainsKey("showLogs"))**

**{**

**LoadLogs();**

**return;**

**}**

**}**

**private void LoadAllCrews()**

**{**

**AllCrews.Clear();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT CrewID, CrewName FROM crews", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllCrews.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllEquipment()**

**{**

**AllEquipment.Clear();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT EquipmentID, EquipmentName FROM equipment", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllEquipment.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllPersonnel()**

**{**

**AllPersonnel.Clear();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT PersonnelID, FullName FROM personnel", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllPersonnel.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllWorkTypes()**

**{**

**AllWorkTypes.Clear();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT WorkTypeID, WorkTypeName FROM worktypes", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllWorkTypes.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllMaterials()**

**{**

**AllMaterials = new List<(int, string)>();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT MaterialID, MaterialName FROM materials", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllMaterials.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllSites()**

**{**

**AllSites.Clear();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT SiteID, SiteName FROM sites", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllSites.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllPositions()**

**{**

**AllPositions.Clear();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT PositionID, PositionName FROM positions", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllPositions.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private void LoadAllManagements()**

**{**

**AllManagements.Clear();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT ManagementID, ManagementName FROM managements", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**AllManagements.Add((reader.GetInt32(0), reader.GetString(1)));**

**}**

**private int CreateCrew(string crewName)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("INSERT INTO crews (CrewName) VALUES (@name); SELECT LAST\_INSERT\_ID();", conn);**

**cmd.Parameters.AddWithValue("@name", crewName);**

**return System.Convert.ToInt32(cmd.ExecuteScalar());**

**}**

**private void AddCrewMember(int crewId, int personnelId)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("INSERT INTO crewmembers (CrewID, PersonnelID) VALUES (@crew, @person)", conn);**

**cmd.Parameters.AddWithValue("@crew", crewId);**

**cmd.Parameters.AddWithValue("@person", personnelId);**

**cmd.ExecuteNonQuery();**

**}**

**private void AssignCrewToObject(int objectId, int crewId)**

**{**

**// Требуется указать WorkTypeID, так как поле не имеет значения по умолчанию**

**// Для простоты можно выбрать первый WorkTypeID из таблицы или добавить выбор в интерфейс**

**int workTypeId = GetFirstWorkTypeId();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("INSERT INTO workassignments (ObjectID, CrewID, WorkTypeID) VALUES (@obj, @crew, @worktype)", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@crew", crewId);**

**cmd.Parameters.AddWithValue("@worktype", workTypeId);**

**cmd.ExecuteNonQuery();**

**}**

**private void AssignCrewToObject(int objectId, int crewId, int workTypeId)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**// Добавляем AssignmentDate = текущая дата**

**using var cmd = new MySqlCommand("INSERT INTO workassignments (ObjectID, CrewID, WorkTypeID, AssignmentDate) VALUES (@obj, @crew, @worktype, CURDATE())", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@crew", crewId);**

**cmd.Parameters.AddWithValue("@worktype", workTypeId);**

**cmd.ExecuteNonQuery();**

**}**

**private int GetFirstWorkTypeId()**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT WorkTypeID FROM worktypes ORDER BY WorkTypeID LIMIT 1", conn);**

**using var reader = cmd.ExecuteReader();**

**if (reader.Read())**

**return reader.GetInt32(0);**

**throw new System.Exception("Нет ни одного WorkTypeID в таблице worktypes");**

**}**

**private void AssignEquipmentToObject(int objectId, int equipmentId)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**// Получаем управление через объект и участок**

**using var cmd = new MySqlCommand(@"**

**INSERT INTO equipmentassignments (EquipmentID, ObjectID, ManagementID, StartDate)**

**SELECT @eq, @obj, s.ManagementID, CURDATE()**

**FROM objects o**

**JOIN sites s ON o.SiteID = s.SiteID**

**WHERE o.ObjectID = @obj**

**", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@eq", equipmentId);**

**cmd.ExecuteNonQuery();**

**}**

**private void AddWorkSchedule(int objectId, int workTypeId, DateTime start, DateTime end)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("INSERT INTO workschedules (ObjectID, WorkTypeID, StartDate, EndDate) VALUES (@obj, @wt, @start, @end)", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@wt", workTypeId);**

**cmd.Parameters.AddWithValue("@start", start);**

**cmd.Parameters.AddWithValue("@end", end);**

**cmd.ExecuteNonQuery();**

**}**

**private void UpdateWorkScheduleEnd(int objectId, int workTypeId, DateTime actualEnd)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("UPDATE workschedules SET ActualEndDate = @actualEnd WHERE ObjectID = @obj AND WorkTypeID = @wt", conn);**

**cmd.Parameters.AddWithValue("@actualEnd", actualEnd);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@wt", workTypeId);**

**cmd.ExecuteNonQuery();**

**}**

**private void UpdateEquipmentAssignmentEnd(int objectId, int equipmentId, DateTime actualEnd)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("UPDATE equipmentassignments SET EndDate = @end WHERE ObjectID = @obj AND EquipmentID = @eq", conn);**

**cmd.Parameters.AddWithValue("@end", actualEnd);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@eq", equipmentId);**

**cmd.ExecuteNonQuery();**

**}**

**private void AddMaterialConsumption(int objectId, int materialId, decimal estimatedQty, decimal consumedQty)**

**{**

**// В вашей таблице materialconsumption нет EstimatedQty и ConsumedQty, только ObjectID, MaterialID, Quantity**

**// Решение: добавлять две записи — одну для сметы (Quantity = estimatedQty, Type = 'estimate'), одну для факта (Quantity = consumedQty, Type = 'actual')**

**// Если поля Type нет, просто добавляйте только фактический расход (consumedQty)**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**// Добавляем только фактический расход (consumedQty)**

**using var cmd = new MySqlCommand("INSERT INTO materialconsumption (ObjectID, MaterialID, Quantity) VALUES (@obj, @mat, @qty)", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@mat", materialId);**

**cmd.Parameters.AddWithValue("@qty", consumedQty);**

**cmd.ExecuteNonQuery();**

**}**

**private void AddOrUpdateMaterialConsumption(int objectId, int materialId, decimal plannedQty, decimal consumedQty)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**// Проверяем, есть ли запись**

**using (var checkCmd = new MySqlCommand("SELECT COUNT(\*) FROM materialconsumption WHERE ObjectID=@obj AND MaterialID=@mat", conn))**

**{**

**checkCmd.Parameters.AddWithValue("@obj", objectId);**

**checkCmd.Parameters.AddWithValue("@mat", materialId);**

**var exists = (long)checkCmd.ExecuteScalar();**

**if (exists == 0)**

**{**

**using var cmd = new MySqlCommand("INSERT INTO materialconsumption (ObjectID, MaterialID, PlannedQuantity, Quantity) VALUES (@obj, @mat, @plan, @fact)", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@mat", materialId);**

**cmd.Parameters.AddWithValue("@plan", plannedQty);**

**cmd.Parameters.AddWithValue("@fact", consumedQty);**

**cmd.ExecuteNonQuery();**

**}**

**else**

**{**

**using var cmd = new MySqlCommand("UPDATE materialconsumption SET PlannedQuantity=@plan, Quantity=@fact WHERE ObjectID=@obj AND MaterialID=@mat", conn);**

**cmd.Parameters.AddWithValue("@obj", objectId);**

**cmd.Parameters.AddWithValue("@mat", materialId);**

**cmd.Parameters.AddWithValue("@plan", plannedQty);**

**cmd.Parameters.AddWithValue("@fact", consumedQty);**

**cmd.ExecuteNonQuery();**

**}**

**}**

**}**

**private void AddPersonnel(string name, int positionId, int managementId, int? siteId)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("INSERT INTO personnel (FullName, PositionID, ManagementID, SiteID) VALUES (@name, @pos, @mgmt, @site)", conn);**

**cmd.Parameters.AddWithValue("@name", name);**

**cmd.Parameters.AddWithValue("@pos", positionId);**

**cmd.Parameters.AddWithValue("@mgmt", managementId);**

**if (siteId.HasValue)**

**cmd.Parameters.AddWithValue("@site", siteId.Value);**

**else**

**cmd.Parameters.AddWithValue("@site", DBNull.Value);**

**cmd.ExecuteNonQuery();**

**}**

**private void AssignSiteHead(int siteId, int personnelId)**

**{**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**// Обновить PositionID на "Начальник участка" и SiteID**

**using var cmd = new MySqlCommand(@"**

**UPDATE personnel**

**SET PositionID = (SELECT PositionID FROM positions WHERE PositionName = 'Начальник участка'), SiteID = @site**

**WHERE PersonnelID = @pid**

**", conn);**

**cmd.Parameters.AddWithValue("@site", siteId);**

**cmd.Parameters.AddWithValue("@pid", personnelId);**

**cmd.ExecuteNonQuery();**

**}**

**private void LoadLogs()**

**{**

**Logs.Clear();**

**string connStr = "server=localhost;user=root;password=lYena00335;database=mokrushin\_kursovaya;";**

**using var conn = new MySqlConnection(connStr);**

**conn.Open();**

**using var cmd = new MySqlCommand("SELECT LogID, Username, Action, ActionTime FROM v\_logs ORDER BY ActionTime DESC", conn);**

**using var reader = cmd.ExecuteReader();**

**while (reader.Read())**

**{**

**Logs.Add(new LogEntry**

**{**

**LogID = reader.GetInt32(0),**

**Username = reader.GetString(1),**

**Action = reader.GetString(2),**

**ActionTime = reader.GetDateTime(3)**

**});**

**}**

**}**

**}**