Частное учреждение

профессиональная образовательная организация

ТЕХНИКУМ «БИЗНЕС И ПРАВО»

ЗАЩИЩЕНО С ОЦЕНКОЙ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и ФИО руководителя)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

**ОТЧЕТ О ВЫПОЛНЕНИИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

|  |  |
| --- | --- |
| по ПМ.05 | Проектирование и разработка информационных систем |
| Выполнил | Нагабедян А.Р. |
| Курс, группа | 3 курс, группа ИС-3 |
| Специальность | 09.02.07 Информационные системы и программирование |
| Руководитель практики | Гонгадзе А. С. |

г. Белореченск

2024

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| ВВЕДЕНИЕ | 3 |
| 1 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ | 5 |
| 1.1 Анализ предметной области | 5 |
| 1.2 Техническое задание | 7 |
| 1.3 Эскизный проект | 19 |
| 2 РАЗРАБОТКА И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ | 26 |
| 2.1 Тестирование клиентской части | 26 |
| 2.2 Руководство пользователя | 29 |
| ЗАКЛЮЧЕНИЕ | 33 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 33 |
| ПРИЛОЖЕНИЕ |  |

ВВЕДЕНИЕ

В современную эпоху цифровой трансформации автоматизация бизнес-процессов становится ключевым фактором повышения операционной эффективности и конкурентного преимущества. Компания "Granite", специализирующаяся на обработке натурального камня и производстве изделий из гранита, мрамора и других материалов, сталкивается с необходимостью оптимизации своих производственных и управленческих процессов.

Объект и предмет исследования

Объектом исследования является система управления заказами камнеобрабатывающего предприятия, включающая:

Процесс оформления заказов на обработку природного камня

Учет характеристик различных видов камня (твердость по Моосу плотность, цвет, тип)

Управление перечнем предоставляемых услуг по обработке камня

Контроль выполнения заказов и расчет их стоимости

Предметом исследования выступает разработка специализированной информационной системы для:

Автоматизации процессов оформления заказов на обработку камня

Учета физических характеристик различных видов природного камня

Управления перечнем услуг по обработке (резка, шлифовка, полировка и др.)

Контроля сроков выполнения заказов

Расчетов стоимости работ на основе веса камня и выбранных услуг

Цели проекта

Основной целью проекта является создание эффективной системы управления заказами для камнеобрабатывающего предприятия, которая обеспечит:

1 Автоматизацию ключевых процессов:

- Оформление новых заказов с указанием типа камня и требуемых услуг

- Расчет стоимости заказа на основе веса материала и выбранных услуг

- Контроль сроков выполнения заказов

2 Учет характеристик материалов:

- Ведение базы данных различных видов камня с их физическими свойствами

- Учет твердости по Моосу, плотности, цвета и типа камня

- Быстрый подбор камня по заданным характеристикам

3 Управление услугами:

- Ведение перечня доступных услуг по обработке камня

- Классификация услуг по типам обработки

- Возможность расширения списка услуг

4 Контроль выполнения заказов:

- Отслеживание статусов выполнения заказов

- Учет даты создания и выполнения заказов

- Формирование отчетности по выполненным работам

1. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

1.1 Анализ предметной области

«Granit» — это организация, предоставляющая услуги по обработке и продаже камней, а также консультации по выбору материалов и их применению.

Категории сотрудников «Granit» можно подразделить на несколько категорий:

1 Специалисты по обработке камней

- Занимаются непосредственной обработкой и подготовкой камней для продажи, выполняют замеры и подгоняют изделия под требования клиентов.

2 Закройщики

- Разрабатывают шаблоны и лекала для обработки камней, обеспечивают точность и качество работы.

3 Администраторы

- Занимаются приемом и обслуживанием клиентов, ведут записи о заказах, контролируют сроки выполнения работ и организуют деятельность компании.

4 Услуги компании «Granit» предоставляет широкий спектр услуг, включая:

- Обработку камней на заказ

- Продажу готовых изделий Консультации по выбору материалов

5 Каждый заказ имеет ряд характеристик, таких как

- Тип изделия (например, столешница, плитка, декоративный элемент и т.д.)

- Используемые материалы

- Сроки выполнения

- Стоимость

6 Учет клиентов и заказов

Каждый новый клиент пополняет перечень клиентов «Granit», ведется учет выполненных заказов. Для крупных организаций заключаются договора, позволяющие им поставлять материалы для компании. Каждая организация имеет свой уникальный номер в базе данных.

7 Характеристики заказов

В заказе указывается:

- ФИО клиента

- Тип изделия

- Используемые материалы

- Сроки выполнения

8 Автоматизация работы

Для упрощения работы администраторов будет разработана система, которая позволит автоматизировать учет заказов, клиентов и особенностей каждого изделия. Это значительно уменьшит трудоемкость работы и повысит эффективность обслуживания.

Виды запросов в информационной системе:

1 Получить все заказы

2 Добавить новый заказ

3 Удалить заказ по ID

4 Получить заказ по ID

5 Получить все услуги

6 Добавить новую услугу

7 Удалить услугу по ID

8 Получить услугу по ID

9 Получить все камни

10 Сохранить новый камень

11 Удалить камень по ID

12 Получить камень по ID

13 Получить всех пользователей

14 Получить пользователя по ФИО

15 Получить пользователя по email

16 Удалить пользователя по ID

17 Сохранить нового пользователя

18 Обновить информацию о заказе

19 Обновить информацию о камне

1. Обновить информацию об услуге

1.2 Техническое задание

1. Общие сведения

1.1 Полное наименование системы и ее условное обозначение

Полное наименование системы: Автоматизированная информационная система мастерская «Granit».

Краткое наименование системы: АИС «Granit».

1.2 Шифр темы или шифр (номер) договора

Шифр темы: АИС-А-228

Номер договора: №2/86-42-15-032 от 10.12.2024.

1.3. Наименование предприятий разработчика и заказчика системы, их реквизиты

Полное наименование заказчика Индивидуальный предприниматель Мойсиев Григорис Сергеевич.

Юридический и почтовый адрес: 352621, Краснодарский Край, г Белореченск, с. Школьное, ул. Мостовая д.3.

Контактный телефон: 8 918 171 62 67

ИНН: 236801242705

КПП: 231043001.

БИК: 040349602

ОГРН: 323237500388098

Расчетный счет:40802810030000039284.

Полное наименование разработчика: Нагабедян Аркадий Робертович.

Юридический и почтовый адрес: Краснодарский Край, г Белореченск, с. Школьное, ул. Кубанская д.18.

Контактный телефон +7 (918) 277 98 68.

2. Назначение и цели создания системы

Разрабатываемая система предназначена для управления мастерской «Granit». Она позволит автоматизировать процессы работы с заказами, клиентами. Система будет предоставлять информацию о клиентах, о заказах клиентов.

Основными целями создания информационной системы для мастерской «Granit» являются:

- Упрощение работы администрации и сотрудников мастерской по учету заказов и клиентов

- Повышение эффективности управления масерской

Для достижения поставленных целей информационная система должна решать следующие задачи:

- Учет текущих выполняемых заказов

- Ведение базы данных о клиентах и их заказах

- Автоматизация оформления заказов

3. Характеристика объектов автоматизации

Характеристика объектов автоматизации для мастерской:

В ходе разработки информационной системы для мастерской «Granit» автоматизируются процессы оформления заказов, просмотра списка заказов, консультации клиентов и отправка отчета в государственные органы.

Автоматизируемыми процессами будут такие операции, как оформление заказа, консультация клиента, отправка отчета в государственные органы. Система будет предоставлять возможность быстрого доступа к информации о заказах, клиентах и поставщиках.

При внедрении системы будет достигнуто:

- Упрощение процессов оформления заказов, учета камней и услуг

- Повышение эффективности работы сотрудников

Объекты автоматизации характеризуются:

- необходимостью доступа к базе данных клиентов.

- возможностью быстрого доступа к информации для принятия оперативных решений

- потребностью к постоянному доступу к сети Интернет

4. Требования к системе

4.1 Требования к структуре и функционированию системы

Система должная предполагать наличие следующих модулей, представленных на рисунке 1:

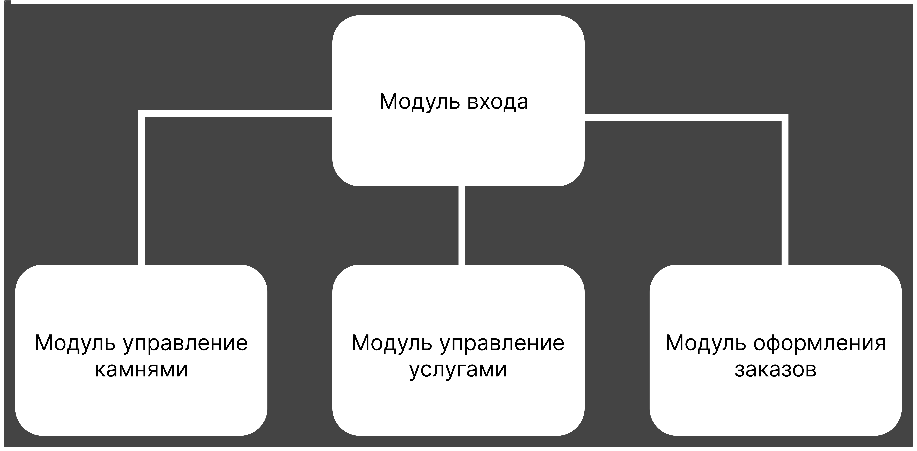


Рисунок 1 – Модули разрабатываемой системы

Серверная часть:

- Модуль авторизации

- Модуль управления камнями

- Модуль управления услугами

- Модуль оформления заказов

Клиентская часть:

- Интерфейс администратора (доступ ко всем модулям системы)

- Интерфейс оформления заказа

На данном этапе необходимо описать примерную архитектуру разрабатываемой системы. Для того, чтобы сделать информационную систему мастерской по обработки камня, мы будем использовать MVC архитектуру. Чтобы убедиться, что она нам точно подходит рассмотрим ее подробнее.

Модель MVC решает следующие задачи:

- изменение только пользовательского интерфейса, а не бизнес-логики приложения;

- использование в одном приложении разных интерфейсов с возможностью выбора;

- замена реакции приложения на действия пользователя за счет использования другого контроллера.

В более широком плане внедрение модели помогает:

- упростить код большого приложения, сделать его понятным и структурированным, облегчить поддержку, тестирование, повторное использование элементов;

- организовать независимую работу различных отделов, которые занимаются разработкой своей части программного продукта;

- упростить программную поддержку MVC-приложения за счет модификации отдельных компонентов, а не всей архитектуры.

На рисунке 2 представление MVC архитектуры.

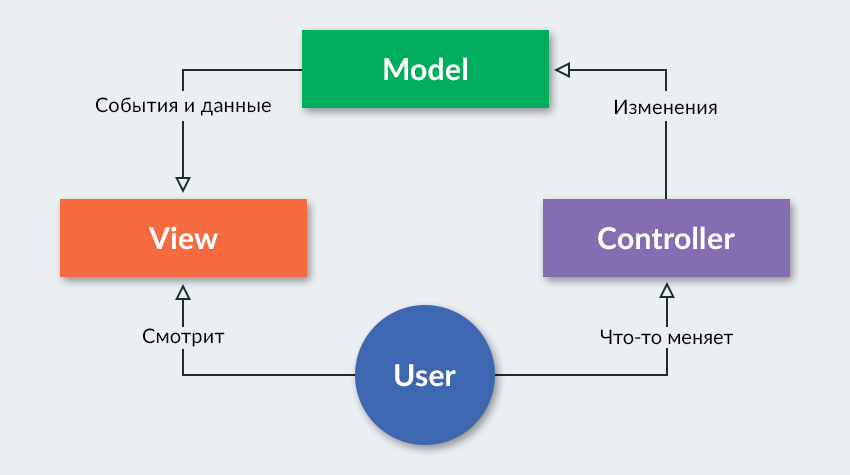


Рисунок 2 – Представление MVC архитектуры

4.2 Требования к видам обеспечения

Для функционирования информационной системы мастерской «Granit» работа клиента и сервера может быть организована следующим образом:

Клиентская часть:

- На компьютерах администраторов требуется наличие сервера и клиентских компьютеров, на которых будет производится работа с системой.

- В зависимости от роли и прав доступа, сотрудники смогут выполнять различные операции в приложении, такие как:

- Администраторы: управление заказами, управление списком клиентов, управление списком заказов.

Серверная часть:

- Сервер базы данных будет хранить всю информацию о клиентах, сотрудниках, заказах и поставщиках.

- Серверная часть должна быть реализована с использованием Java 17 и выше и Spring Framework.

- Система будет обеспечивать безопасность данных, авторизацию пользователей.

В данном случае клиентская часть системы будет представлять собой веб-приложение, запускаемое на компьютерах администраторов мастерской. Веб-приложение будет иметь графический интерфейс и предоставлять необходимый функционал в зависимости от роли пользователя.

Серверная часть будет состоять из сервера базы данных, хранящего всю необходимую информацию. Взаимодействие между клиентским веб-приложением и сервером базы данных будет осуществляться через специальный API, обеспечивающий обмен данными и выполнение запросов.

При анализе предметной области разрабатываемой системы были выделены следующие сущности, на основе которых будет разрабатываться база данных системы:

- Камень

- Заказ

- Услуга

База данных информационной системы должна состоять из следующих связанных таблиц:

-Stone – Таблица, содержащая информацию о камни

- Order – Таблица, содержащая информацию о заказе

-Service – Таблица, содержащая информацию о услуге

Данные о камни размещаются в таблице Stone:

Таблица 1 – Таблица Stone в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| id | integer | 11 | Идентификатор камня | Первичный ключ |
| name | varchar | 50 | ФИО клиента | NOT NULL |
| firmnessToMohs | varchar | 50 | Твёрдость по Моосу | NOT NULL |
| density | varchar | 50 | Плотность | NOT NULL |
| color | varchar | 50 | Цвет | NOT NULL |
| typeOfStone | varchar | 50 | Тип камня | NOT NULL |

Данные о заказах размещаются в таблице Order:

Таблица 2 – Таблица Order в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| Id | integer | 11 | Идентификатор заказа | Первичный ключ |
| stone\_id | integer | 11 | Идентификатор камня | Внешний ключ к таблице stone |
| service\_id | integer | 11 | Идентификатор клиента | Внешний ключ к таблице service |
| client | varchar | 50 | ФИО клиента | NOT NULL |
| weight | varchar | 50 | Вес камня | NOT NULL |
| totalPrice | varchar | 50 | Цена заказа | NOT NULL |
| OrderCreationDate | date |  | Дата создания заказа | NOT NULL |

Данные о услуги размещаются в таблице Service:

Таблица 3 – Таблица User в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| Id | integer | 11 | Идентификатор сотрудника | Первичный ключ |
| name | varchar | 50 | Название услуги | NOT NULL |

Данные о услуги размещаются в таблице User:

Таблица 4 – Таблица User в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| Id | integer | 11 | Идентификатор клиента | Первичный ключ |
| name | varchar | 50 | ФИО клиента | NOT NULL |
| email | varchar | 50 | Email пользователя | NOT NULL |
| password | varchar | 50 | Пароль | NOT NULL |

4.3 Требования к эргономике и технической эстетике

Общие требования:

- Фоновый цвет – белый (#FFF).

- Акцентный цвет – Синий(#005FDB).

- При наведении указателя мышки на элементы кнопок,

- Они должны выделяться, например, путем изменения цвета или появления подсветки экрана.

- Шрифты и размеры текста должны быть четкими и хорошо читаемыми.

- Адаптивный интерфейс

Элементы интерфейса:

- Таблицы имеют прозрачные границы ячеек и заголовков для более гладкого визуального восприятия.

- Кнопки имеют закругленные углы.

Страница авторизации:

- Все элементы страницы должны быть отцентрированы одновременно по вертикали и горизонтали.

- Заголовок страницы – «Авторизация».

- Под элементами ввода текста - Кнопка “Войти”

- Элементы ввода текста (логин и пароль) должны иметь одинаковую ширину.

- Кнопка «Войти» – Синий (#005FDB) цвет. Цвет шрифта – белый (#fff). При наведении на кнопку ее цвет становится темнее.

- Под кнопкой «Войти» располагается ссылка «Зарегистрироваться», ведущая на страницу регистрации

Страница регистрации:

- Все элементы страницы должны быть отцентрированы одновременно по вертикали и горизонтали.

- Заголовок страницы – «Регистрация».

- Под элементами ввода текста - Кнопка «Зарегистрироваться»

- Элементы ввода текста (логин и пароль) должны иметь одинаковую ширину.

- Кнопка «Зарегистрироваться» – Зеленый цвет. Цвет шрифта - белый (#fff). При наведении на кнопку ее цвет становится темнее.

- Под кнопкой «Зарегистрироваться» располагается текст «Уже есть аккаунт?» и ссылка «Войти», ведущая на страницу авторизации

* 1. Требования к содержанию и оформлению выводимых сообщений

- Отображение всплывающей подсказки в форме выноски снизу от поля ввода логина в случае ввода некорректного формата, уведомление автоматически исчезает, когда пользователь начнет заново вводить данные;

- Если пользователь вводит неверные данные и нажимает на кнопку войти, то сформируется окно с надписью: «Неверный email или пароль. Попробуйте заново»;

- Сообщения об ошибках при вводе некорректных данных (например, «Название должно начинаться с заглавной буквы и содержать только русские буквы»);

- Автоматическое исчезновение подсказки при начале нового ввода данных в соответствующее поле;

- Сообщения об ошибках при незаполненных обязательных полях (например, «Недопустимое заполнение» или «Пожалуйста, заполните все поля»);

4.5 Требования к видам обеспечения

Для функционирования системы требуется сервер и клиентских компьютеры, на которых будет производится запуск системы.

Требования для сервера БД:

- Windows 10 и выше;

- Процессор 2 ядра (8 логических потоков), частота – 2—2,53 - Оперативная память 8 Гб и выше

- Свободное дисковое пространство не менее 10 Гб

- SSD(твердотельный накопитель)

Требования к клиентским устройствам:

- Windows 10 и выше

- Процессор 2 ядра (8 логических потоков), частота – 2—2,53

- Оперативная память 8 ГБ и выше

- Монитор с разрешением не ниже 1920x1080

- Браузер Google Chrome Версия 131.0.6778.141, Microsoft Edge версия 131.0.2903.99, Яндекс Версия 24.12.0.1810.

4.6 Эскизы основных окон системы

На рисунках 4-9 представлены эскизы основных окон системы.

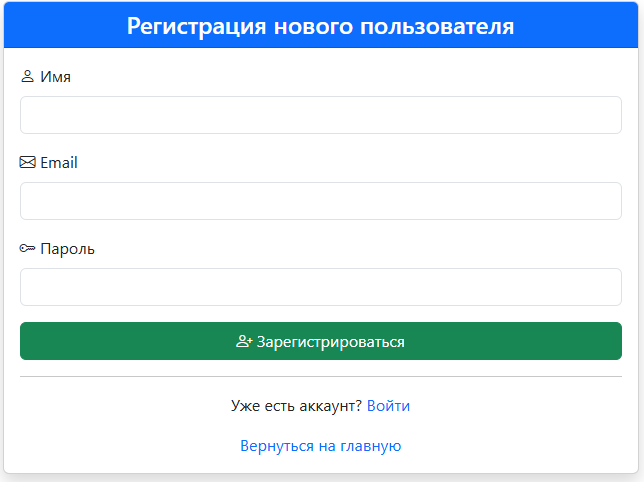


Рисунок 4 – Форма регистрации

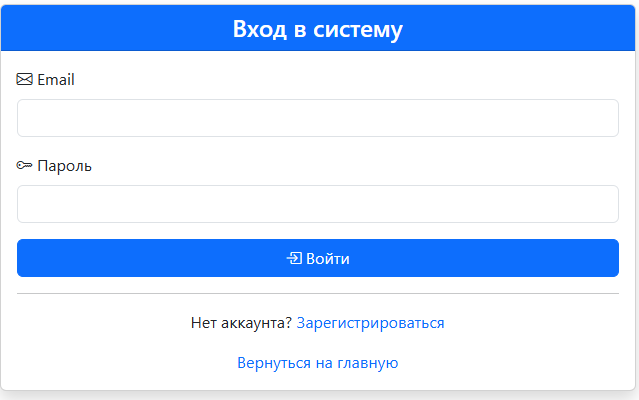


Рисунок 5 – Форма входа

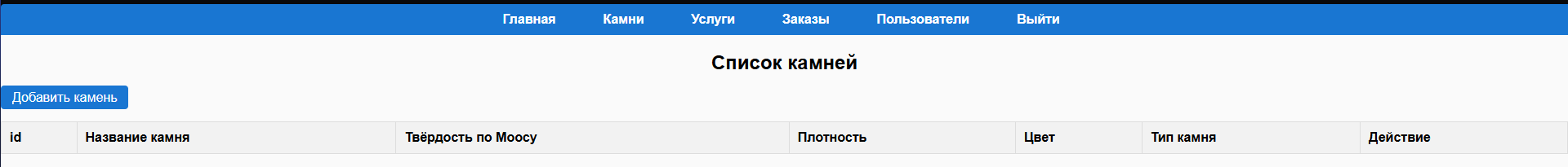


Рисунок 6 – Список камней

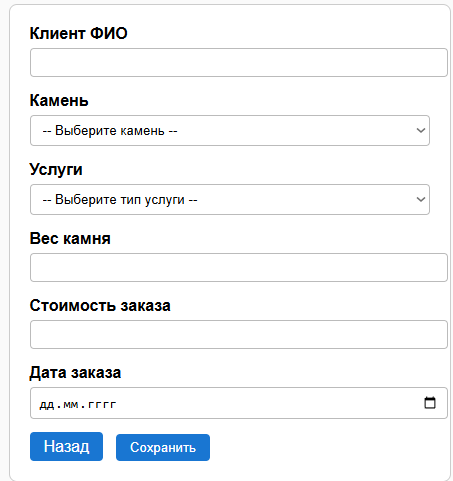


Рисунок 7 – Форма добавления заказа

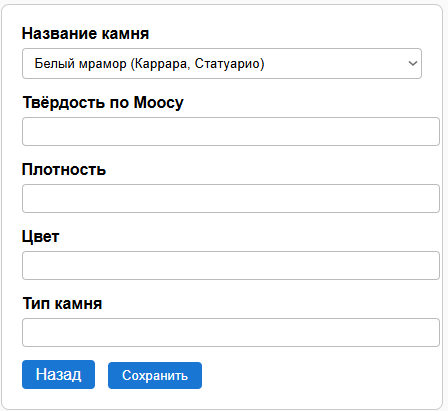


Рисунок 8 – Форма добавления камня

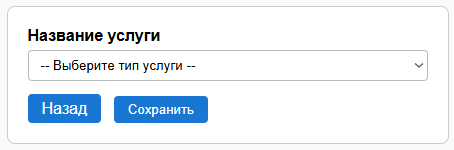


Рисунок 9 – Добавление услуги

5 Состав и содержание работ по созданию системы

Комплекс работ по созданию АИС мастерская «Granit» должен осуществляться в несколько этапов.

1. Анализ требований:

- Определение функциональных и нефункциональных требований к системе.

- Изучение бизнес-процессов мастерской.

2. Проектирование системы:

- Разработка концептуальной модели системы.

- Проектирование архитектуры системы (клиент-серверная, веб-приложение и т.д.).

- Проектирование базы данных для хранения информации о клиентах, заказах и т.д.

- Проектирование пользовательского интерфейса и навигации.

3. Разработка системы:

- Создание серверной части системы (бэкенд).

- Разработка клиентской части системы (фронтенд).

- Реализация модулей для управления клиентами, заказами и т.д.

4. Тестирование системы:

- Модульное тестирование отдельных компонентов системы.

- Интеграционное тестирование взаимодействия компонентов.

- Системное тестирование функциональности системы.

5. Развертывание и внедрение системы:

- Подготовка серверной и клиентской инфраструктуры.

- Установка и настройка системы на серверах и клиентских устройствах.

- Миграция и загрузка данных из существующих систем (если применимо).

- Обучение персонала работе с новой системой.

6. Опытная эксплуатация и доработка системы:

- Опытная эксплуатация системы в рабочей среде.

- Устранение выявленных недочетов и ошибок.

- Доработка системы по результатам опытной эксплуатации.

7. Ввод системы в промышленную эксплуатацию:

- Окончательный переход на использование новой системы.

- Вывод из эксплуатации старых систем (если применимо).

8. Сопровождение и развитие системы:

- Техническая поддержка и устранение возникающих ошибок.

- Обновление системы в соответствии с изменяющимися требованиями.

- Добавление новых функций и возможностей по мере необходимости.

9. Приемка работ

- Проведение предварительных испытаний на стенде исполнителя

6. Порядок контроля и приемки системы

Система передается в виде полностью функционирующего комплекса на базе средств вычислительной техники заказчика и исполнителя в сроки, установленные договором. Приемка системы осуществляется комиссией в составе уполномоченных представителей заказчика и исполнителя. Весь комплект документации на АИС «СЦМ», разработанный исполнителем, передается заказчику на бумажных (в двух экземплярах) и на USB-носитель(флешка), один экземпляр после подписания передается исполнителю.

7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие

Заказчик обязан предоставить сервер и персональные компьютеры, требования к которым указаны в разделе 4.2 «Требования к видам обеспечения».

Необходимое программное обеспечение для сервера:

- Операционная система: Windows 10

- СУБД: MySQL.

 Необходимое программное обеспечение для персонального компьютера:

- Операционная система: Windows 10;

- Google Chrome(x64)/ Opera(x64)

8.  Требования к документированию

Перечень подлежащих разработке комплектов и видов документов:

- Описание анализа предметной области

- Техническое задание

- Эскизный проект

- Веб-приложение

9. Источники разработки

Источниками разработки настоящего Технического Задания являются:

- перечень документации, представленный в разделе 1.2 «Основания для проведения работ»;

- ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы;

- ГОСТ 24.601-86 Автоматизированные системы. Стадии создания;

- ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению;

- ГОСТ 24.601 Исследование и обоснование создания АС.

1.3 Эскизный проект

1. Декомпозиция системы на комплексы и проработка архитектурных решений

Назначение системы АИС «Granit» предназначена для комплексного информационно управленческого обеспечения процессов в части исполнения процессов, описанных ниже:

- Управление заказами;

- Оформление заказа;

- Регистрация пользователей;

Информационная система управления «Granit» разрабатывается для повышения эффективности управления мастерской и качества обслуживания клиентов.

Система позволит централизованно хранить и обрабатывать всю информацию о заказах, сотрудниках, материалах. Для сотрудников мастерской система обеспечит удобный доступ ко всем необходимым данным и инструментам управления.

Основными целями создания АИС «Granit» являются:

- Повышение загрузки номерного фонда и выручки от продажи дополнительных услуг за счет эффективного управления предложением.

- Улучшение качества обслуживания клиентов благодаря оперативному реагированию сотрудников.

Реализация вышеперечисленных целей преследует достижение ключевой цели - повышения рентабельности и конкурентоспособности мастерской за счет оптимизации бизнес-процессов и улучшения качества обслуживания клиентов.

На данном этапе рассмотрим, как будет вести себя система при выполнении выделенных функций. Для этого будем использовать диаграммы состояний в нотации UML.

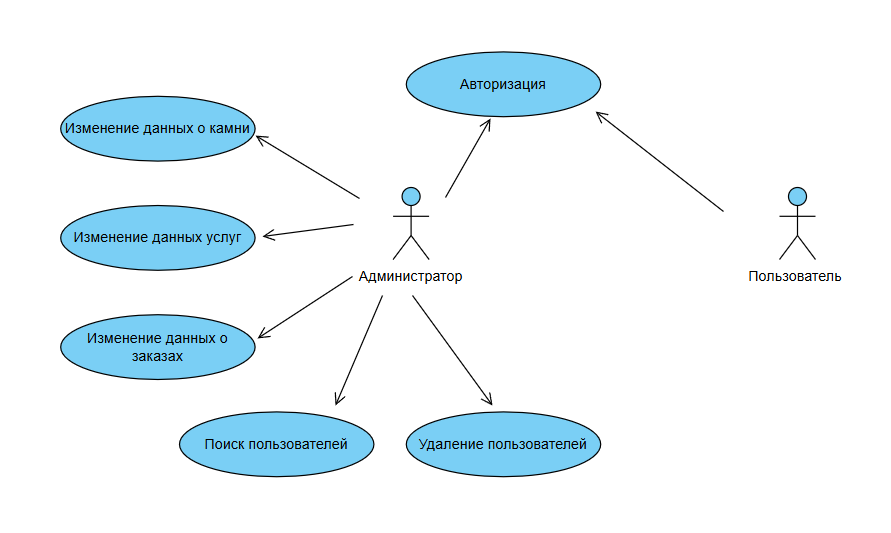


Рисунок 10 – Диаграмма вариантов использования

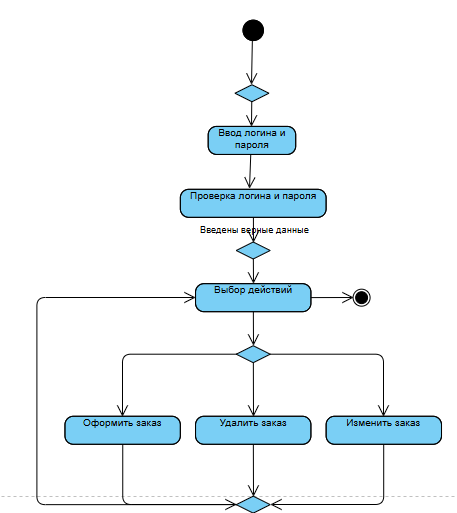


Рисунок 11 – Диаграмма состояния

Модель MVC решает следующие задачи:

- изменение только пользовательского интерфейса, а не бизнес-логики приложения;

- использование в одном приложении разных интерфейсов с возможностью выбора;

- замена реакции приложения на действия пользователя за счет использования другого контроллера.

В более широком плане внедрение модели помогает:

- упростить код большого приложения, сделать его понятным и структурированным, облегчить поддержку, тестирование, повторное использование элементов;

- организовать независимую работу различных отделов, которые занимаются разработкой своей части программного продукта;

- упростить программную поддержку MVC-приложения за счет модификации отдельных компонентов, а не всей архитектуры.

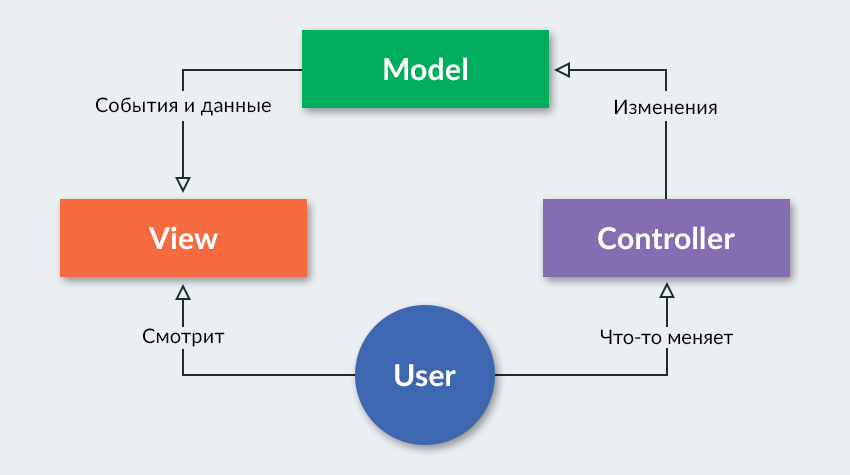


Рисунок 12 – Представление MVC архитектуры

Нашу систему можно разделить на отдельные модули, представленные на рисунке 13:

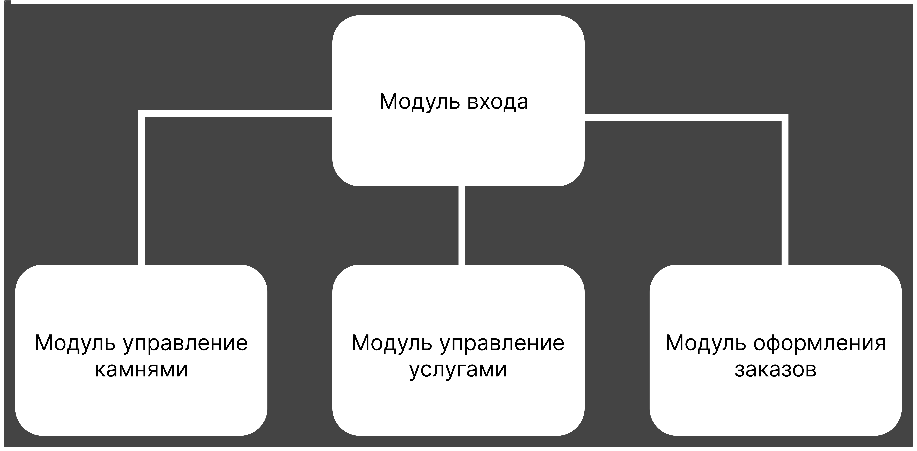


Рисунок 13 – Модули разрабатываемой системы

Серверная часть:

- Модуль авторизации

- Модуль управления камнями

- Модуль управления услугами

- Модуль оформления заказов

База данных информационной системы должна состоять из следующих связанных таблиц:

-Stone – Таблица, содержащая информацию о камни

- Order – Таблица, содержащая информацию о заказе

-Service – Таблица, содержащая информацию о услуге

Данные о камни размещаются в таблице Stone:

Таблица 1 – Таблица Stone в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| id | integer | 11 | Идентификатор камня | Первичный ключ |
| name | varchar | 50 | ФИО клиента | NOT NULL |
| firmnessToMohs | varchar | 50 | Твёрдость по Моосу | NOT NULL |
| density | varchar | 50 | Плотность | NOT NULL |
| color | varchar | 50 | Цвет | NOT NULL |
| typeOfStone | varchar | 50 | Тип камня | NOT NULL |

Данные о заказах размещаются в таблице Order:

Таблица 2 – Таблица Order в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| Id | integer | 11 | Идентификатор заказа | Первичный ключ |
| stone\_id | integer | 11 | Идентификатор камня | Внешний ключ к таблице stone |
| service\_id | integer | 11 | Идентификатор клиента | Внешний ключ к таблице service |
| client | varchar | 50 | ФИО клиента | NOT NULL |
| weight | varchar | 50 | Вес камня | NOT NULL |
| totalPrice | varchar | 50 | Цена заказа | NOT NULL |
| OrderCreationDate | date |  | Дата создания заказа | NOT NULL |

Данные о услуги размещаются в таблице Service:

Таблица 3 – Таблица User в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| Id | integer | 11 | Идентификатор сотрудника | Первичный ключ |
| name | varchar | 50 | Название услуги | NOT NULL |

Данные о услуги размещаются в таблице User:

Таблица 4 – Таблица User в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| Id | integer | 11 | Идентификатор клиента | Первичный ключ |
| name | varchar | 50 | ФИО клиента | NOT NULL |
| email | varchar | 50 | Email пользователя | NOT NULL |
| password | varchar | 50 | Пароль | NOT NULL |

2. Разработка операционной среды или требований к ней

Серверная часть: ОС Windows 10, СУБД MySQL

Клиентские рабочие места: ОС Windows 10

Использование сети Интернет для удаленного доступа.

3. Разработка методов анализа качества системы и метрик оценки

Метрики показателей качественной оценки:

- Надежность: Количество сбоев и их частота.

- Производительность: Время отклика системы на запросы.

- Масштабируемость: Возможность системы обрабатывать увеличивающиеся объемы данных и пользователей.

- Безопасность: Уровень защиты данных, количество инцидентов безопасности.

- Удобство использования: Оценка удобства интерфейса пользователями.

Методы анализа качества:

- Тестирование на надежность и отказоустойчивость.

- Производительность и нагрузочное тестирование.

- Безопасностное тестирование, включая тестирование на проникновение.

- Оценка удобства использования на основе обратной связи пользователей.

4. Планирование перспектив создания АСУ

 Основные концепции технологий разработки:

- Используемые технологии: Java Spring Boot MVC для серверной части, MySQL для базы данных.

Требования к программным средствам:

- Среда разработки: IntelliJ IDEA для разработки на Java.

База данных по автоматизации и контролю работ:

- Разработка базы данных для отслеживания задач и прогресса разработки.

- Использование систем мониторинга и отчетности для контроля качества кода и системы в целом.

Оформление пояснительной записки:

- Подробное описание архитектурных решений и технологий.

- Технические требования и спецификации.

- План внедрения и тестирования системы.

- Документация по эксплуатации и поддержке системы.

5. Эскиз пользовательского интерфейса АИС

Эскизы пользовательского интерфейса представлены в разделе 4.6 «Эскизы основных окон системы»

2 РАЗРАБОТКА И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Тестирование клиентской части

1. Тестирование интерфейса /stone/new

1.2 Проверка добавление нового камня:

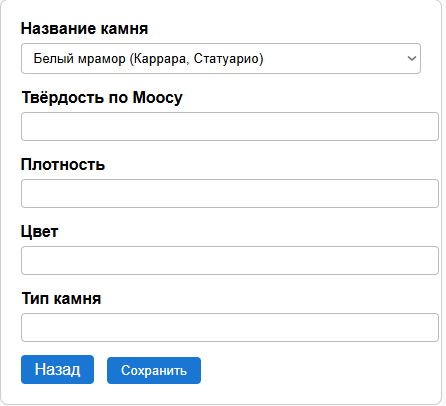


Рисунок 14 – Форма добавления камня

- Вводим в поля название камня, твёрдость по Моосу, плотность, цвет, тип камня.

- Нажимает кнопку «Сохранить».

- Проверяем, что камень был добавлен был успешно добавлен и отображается в списке.

1.2 Проверка валидации:

- Оставляем одно или несколько полей пустыми.

- Нажимаем «Добавить».

- Убеждаемся, что выводится уведомление об ошибке и камень не добавлен.

2. Тестирование интерфейса /stone/list

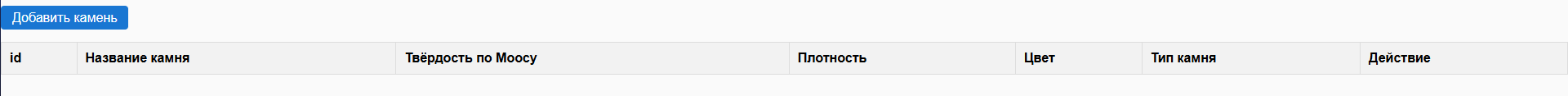


Рисунок 15 – Список камней

2.1 Проверка отображения списка камней:

- Открываем страницу «Камни».

- Убеждаемся, что таблица корректно отображает данные.

2.2 Проверка редактирования камня:

- Нажимаем «Редактировать» делаем изменения.

- Нажимаем «Сохранить».

- Проверяем, что изменения отображаются в списке.

2.3 Проверка удаления камня (роль ADMIN):

- Выбираем камень из списка.

- Нажимаем «Удалить».

- Убеждаемся, что камень удалён из таблицы.

3. Тестирование интерфейса /order/newOrder

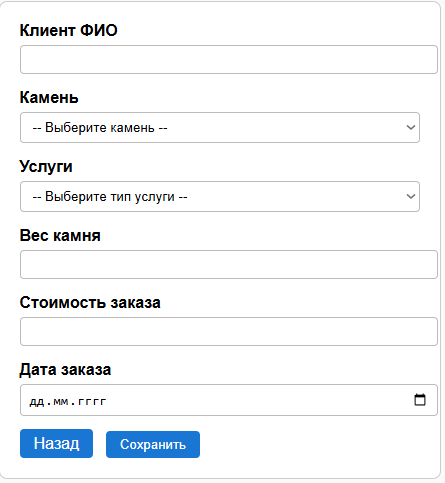


Рисунок 16 – Форма для добавления заказа

3.1 Проверка создания заказа:

- Вводим клиент ФИО камень, услуги, вес камня, стоимость заказа, дата заказа

- Нажимаем «Создать заказ».

- Убеждаемся, что заказ успешно создан.

3.2 Проверка валидации формы:

- Не заполняем одно из обязательных полей.

- Нажимаем «Создать заказ».

- Проверяем, что появляется сообщение об ошибке.

4. Тестирование интерфейса /stone/listOrder

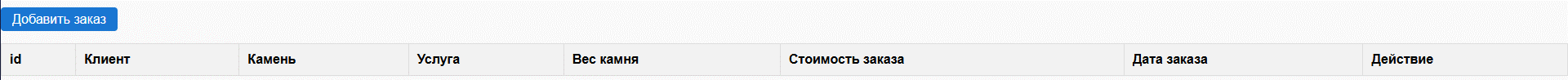


Рисунок 17 – Список заказов

4.1 Проверка отображения списка заказов:

- Открываем страницу «Заказы».

- Убеждаемся, что таблица корректно отображает данные.

4.2 Проверка редактирования заказа:

- Нажимаем «Редактировать» делаем изменения.

- Нажимаем «Сохранить».

- Проверяем, что изменения отображаются в списке.

4.3 Проверка удаления заказа (роль ADMIN):

- Выбираем заказ из списка.

- Нажимаем «Удалить».

- Убеждаемся, что заказ удалён из таблицы.

5. Тестирование интерфейса /order/newService

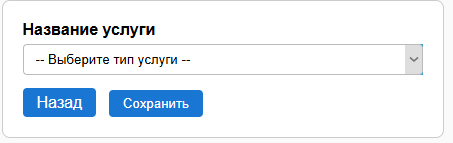


Рисунок 18 – Форма для создания услуг

5.1 Проверка создания услуги:

- Выбираем услугу из списка

- Нажимаем «Сохранить».

- Убеждаемся, что услуга сохранилась успешно создан.

6. Тестирование интерфейса /stone/listService

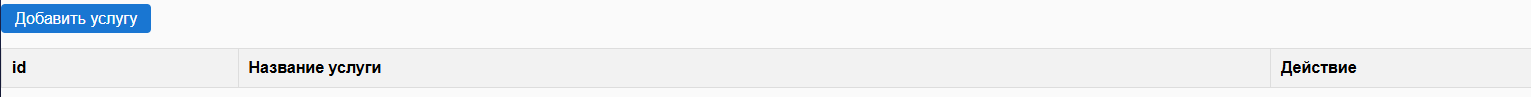


Рисунок 19 – Список услуг

6.1 Проверка отображения списка услуг:

- Открываем страницу «Услуги».

- Убеждаемся, что таблица корректно отображает данные.

6.2 Проверка редактирования услуг:

- Нажимаем «Редактировать» делаем изменения.

- Нажимаем «Сохранить».

- Проверяем, что изменения отображаются в списке.

6.3 Проверка удаления услуги (роль ADMIN):

- Выбираем услуги из списка.

- Нажимаем «Удалить».

- Убеждаемся, что услуги удалена из таблицы.

2.2 Руководство пользователя

1. Авторизация

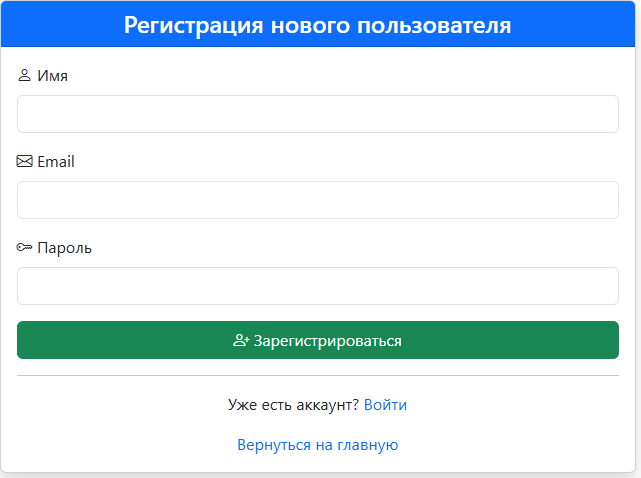


Рисунок 20 – Авторизация

Для начала работы с приложением необходимо авторизоваться. Введите email и пароль в соответствующие поля и нажмите кнопку «Войти».

Если вы являетесь администратором доступ к функциям будет ограничен в зависимости от вашей роли.

После успешной авторизации вы попадёте на главную страницу приложения.

2. Главное окно

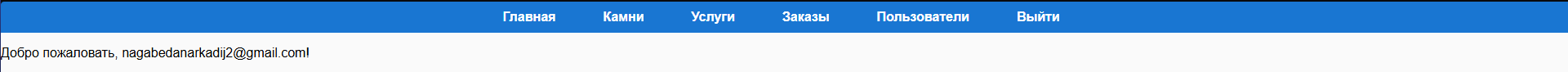


Рисунок 21 – Авторизация

Главное окно приложения состоит из меню, доступных в зависимости от вашей роли:

Для администратора:

- Управление услугами

- Управление заказами

- Управление камнями

- Управление Пользователями

Также в интерфейсе всегда доступна кнопка «О программе» и «Выход».

3. О программе

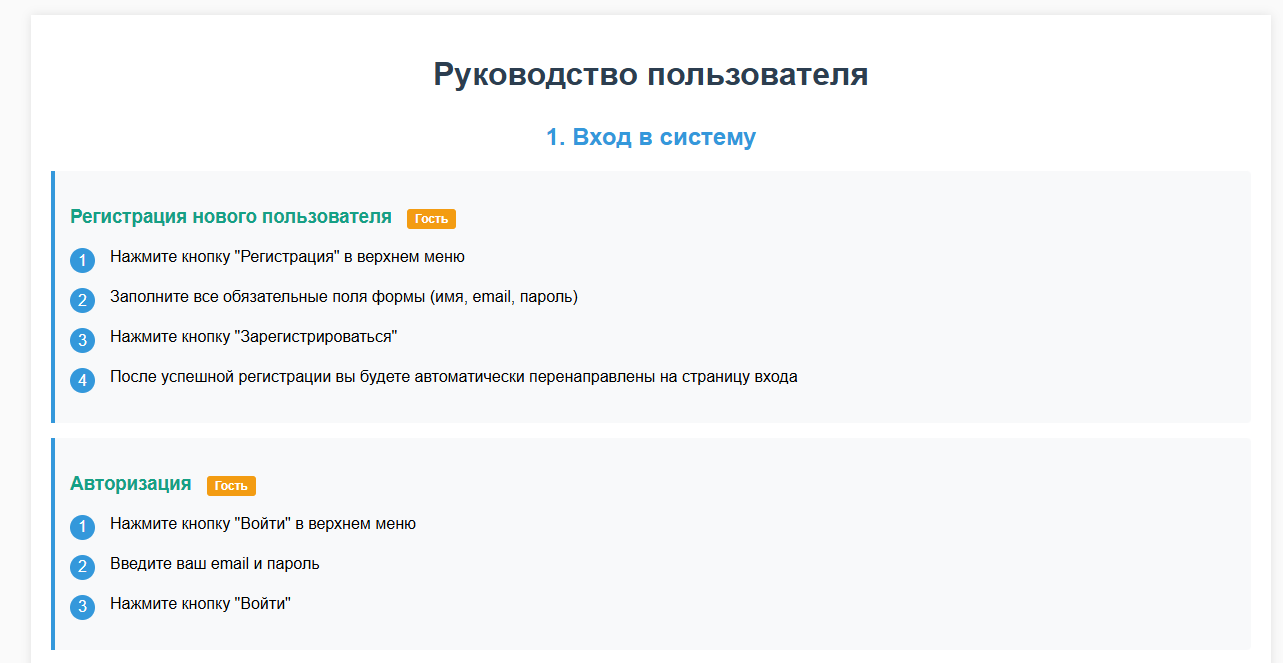
Раздел «О программе» содержит информацию о описании системы.  


Рисунок 22 – Руководство входа в систему

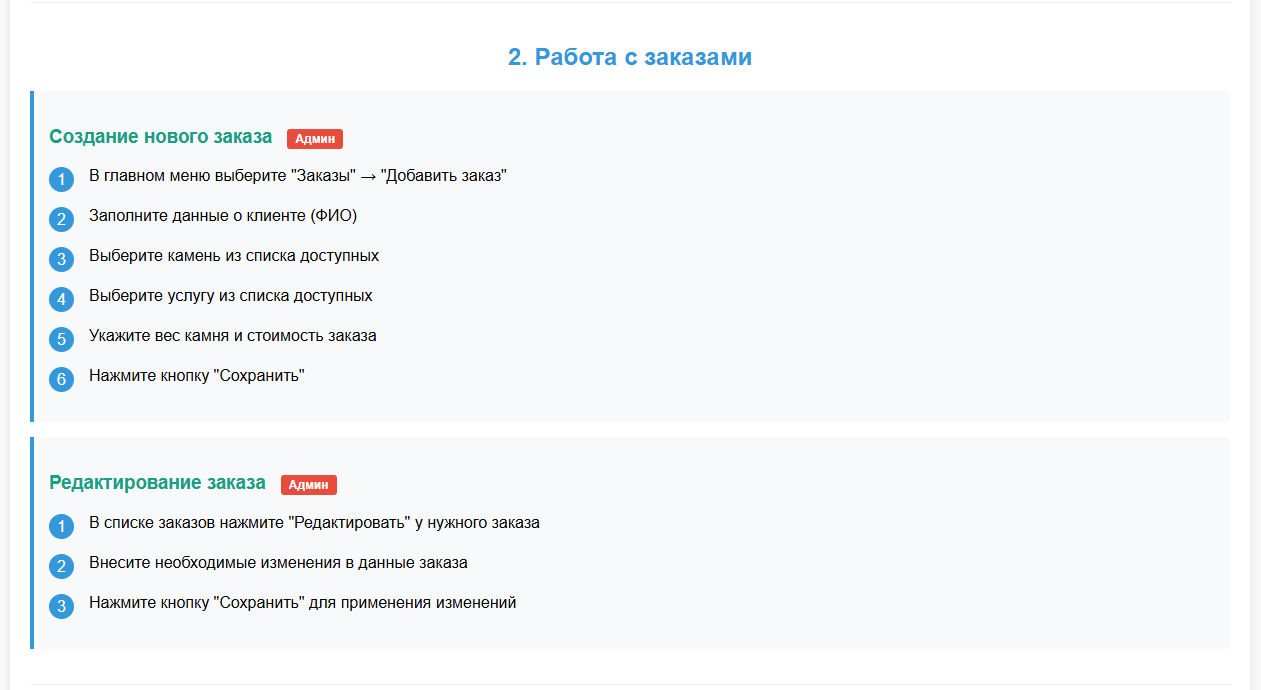


Рисунок 23 – Руководство работы с заказами

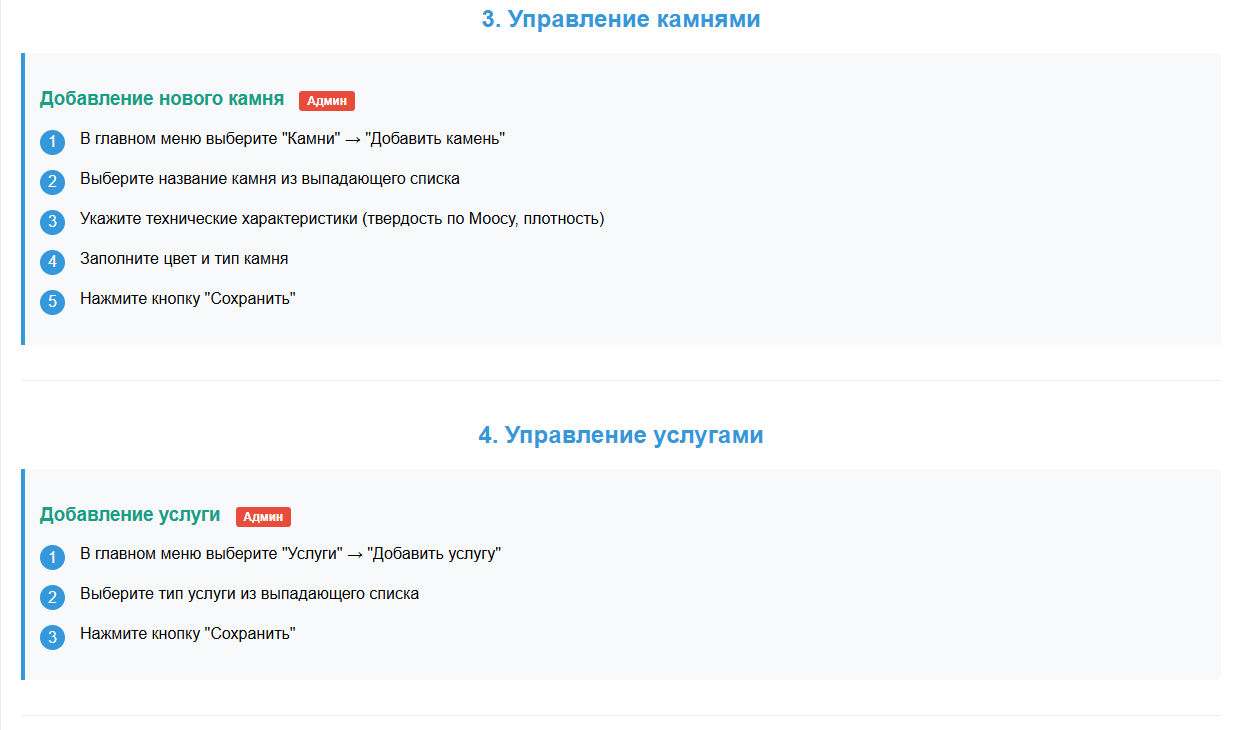


Рисунок 24 – Руководство работы с камнями и услугами

4. Выход

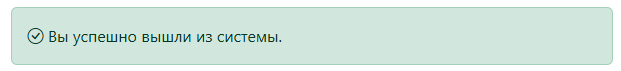


Рисунок 22 – Выход из системы

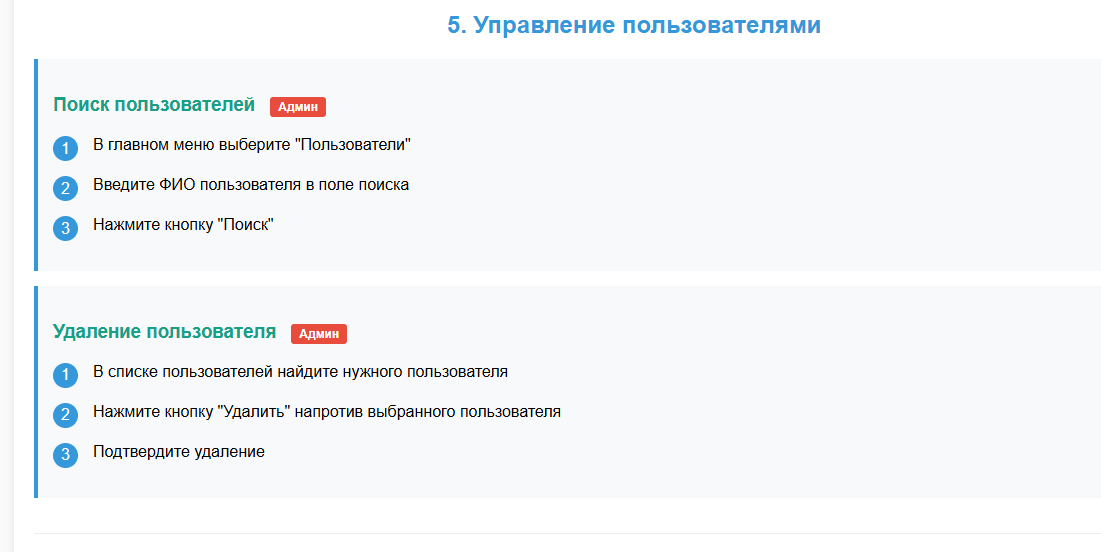


Рисунок 25 – Руководство управления пользователями

Для выхода из системы нажмите кнопку «Выход» в правом верхнем углу.

Для администратора

5 Камни

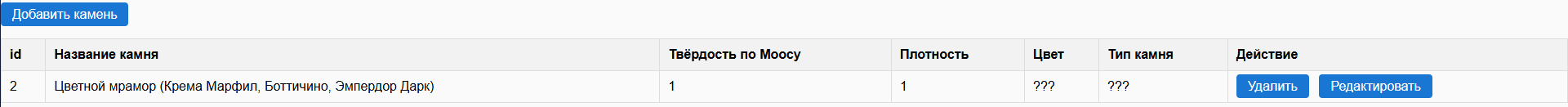


Рисунок 23 – Список камней

На этой странице распложен список всех материалов. Кнопки удалить, редактировать, добавить камень.

6 Услуги



Рисунок 24 – Список услуг

На этой странице расположен список всех услуг. Кнопки удалить, редактировать, добавить услугу.

7 Заказы

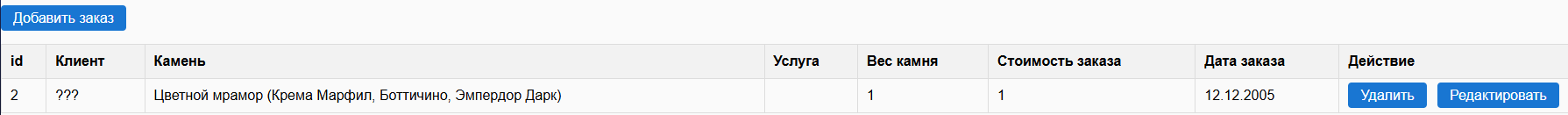


Рисунок 25 – Список заказов

На этой странице расположен список заказов. Кнопки удалить, редактировать, добавить заказ.

ЗАКЛЮЧЕНИЕ

В ходе прохождения учебной практики была спроектирована и реализована информационная система для мастерской «Ganit», предназначенная для автоматизации ключевых бизнес-процессов предприятия. В процессе разработки были подробно изучены структура и деятельность мастерской по обработки камня, определены требования к функциональности системы, выполнено проектирование архитектуры, разработаны модули учёта заказов, учёта камней, учёта услуг.

Система обеспечивает разграничение прав доступа пользователей в зависимости от их ролей (администратор, пользователь), что повышает безопасность и удобство эксплуатации. Внедрение авторизации, регистрационного механизма позволяет повысить эффективность работы персонала и улучшить взаимодействие с клиентами.

Особое внимание уделено удобству пользовательского интерфейса, а также корректной обработке ошибок, что обеспечивает положительный пользовательский опыт. Для клиентской части использован современный веб-интерфейс, построенный с использованием HTML, а серверная логика реализована на основе Spring Boot.

В результате внедрения информационной системы «Granit» достигаются следующие преимущества:

- Повышение точности и скорости обработки информации;

- Упрощение учёта заказов;

- Снижение количества рутинной работы;

- Улучшение качества обслуживания клиентов.

Разработанная система готова к эксплуатации и может быть адаптирована и расширена в будущем с учетом новых требований бизнеса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Java. Полное руководство, 12-е изд. : Пер. с англ. - СПб. "Диалектика•; 2023. - 1344 с.: ил. - Парал. тит. англ.

2. Spring в действии. 6-е изд./ пер. с англ.А. Н. Киселева. – М.: ДМК Пресс, 2022. – 544 с.: ил

3. Java Persistence API. Руководство разработчика / Пер. с англ. – СПб.: БХВ-Петербург, 2021. – 384 с.

4. ГОСТ 24.601 Исследование и обоснование создания АС

5. ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы;

6. Spring Security Reference [Электронный ресурс]. Режим доступа: <https://docs.spring.io/spring-security/reference/>

7. Hibernate для начинающих [Электронный ресурс]. Режим доступа: <https://habr.com/ru/articles/320542/>

8. Spring Guides [Электронный ресурс]. Режим доступа: <https://spring.io/guides>

9. MySQL Documentation [Электронный ресурс]. Режим доступа: <https://dev.mysql.com/doc/>

10. Thymeleaf [Электронный ресурс]. Режим доступа: <https://www.thymeleaf.org>

ПРИЛОЖЕНИЕ

SecurityConfig :

package org.example.prostorazvlecaus.config;  
  
import org.example.prostorazvlecaus.service.impl.CustomUserDetailsService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
  
@Configuration  
@EnableWebSecurity  
@EnableMethodSecurity(prePostEnabled = true)  
public class SecurityConfig {  
  
 private final CustomUserDetailsService userDetailsService;  
  
 @Autowired  
 public SecurityConfig(CustomUserDetailsService userDetailsService) {  
 this.userDetailsService = userDetailsService;  
 }  
  
 @Bean  
 public static PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
 .csrf(csrf -> csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/css/\*\*", "/js/\*\*", "/images/\*\*", "/", "/register/\*\*",  
 "/register-process", "/login/\*\*", "/guide" ,"/style.css").permitAll()  
 .requestMatchers("/order/\*\*", "/stone/\*\*", "/service/\*\*" ,"/user/\*\*").hasRole("ADMIN")  
 .anyRequest().authenticated()  
 )  
 .formLogin(form -> form  
 .loginPage("/login")  
 .loginProcessingUrl("/login")  
 .defaultSuccessUrl("/" , true )  
 .permitAll()  
 )  
 .logout(logout -> logout  
 .logoutUrl("/logout")  
 .logoutSuccessUrl("/login?logout") *// Перенаправление после выхода* .permitAll()  
 );  
  
 return http.build();  
 }  
  
 @Autowired  
 public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {  
 auth  
 .userDetailsService(userDetailsService)  
 .passwordEncoder(*passwordEncoder*());  
 }  
}

WebConfig:

package org.example.prostorazvlecaus.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
  
@Configuration  
public class WebConfig implements WebMvcConfigurer {  
 @Override  
 public void addViewControllers(ViewControllerRegistry registry) {  
 registry.addViewController("/stone").setViewName("stone/list");  
 registry.addViewController("/order").setViewName("order/listOrder");  
 registry.addViewController("/service").setViewName("service/listService");  
 registry.addViewController("/user").setViewName("user/listUser");  
 }  
  
 @Override  
 public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 *// Регистрация обработчиков статических ресурсов* registry.addResourceHandler("/css/\*\*")  
 .addResourceLocations("classpath:/static/css/");  
 registry.addResourceHandler("/js/\*\*")  
 .addResourceLocations("classpath:/static/js/");  
 registry.addResourceHandler("/images/\*\*")  
 .addResourceLocations("classpath:/static/images/");  
 }  
}

AuthController:

package org.example.prostorazvlecaus.controller;  
  
import org.example.prostorazvlecaus.model.User;  
import org.example.prostorazvlecaus.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.validation.BindingResult;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
  
@Controller  
public class AuthController {  
  
 @Autowired  
 private UserService userService;  
  
 @GetMapping("/login")  
 public String login(@RequestParam(value = "error", required = false) String error,  
 @RequestParam(value = "logout", required = false) String logout,  
 Model model) {  
 if (error != null) {  
 model.addAttribute("error", "Неверный логин или пароль!");  
 }  
 if (logout != null) {  
 model.addAttribute("logout", "Вы успешно вышли из системы!");  
 }  
 return "auth/login";  
 }  
  
 @GetMapping("/register")  
 public String showRegistrationForm(Model model) {  
 model.addAttribute("user", new User());  
 return "auth/register";  
 }  
  
 @PostMapping("/register-process")  
 public String processRegistration(@ModelAttribute("user") User user,  
 BindingResult bindingResult,  
 Model model) {  
 User existingUser = userService.getUserByEmail(user.getEmail());  
 if (existingUser != null) {  
 bindingResult.rejectValue("email", "error.user",  
 "Пользователь с таким email уже существует");  
 return "auth/register";  
 }  
 userService.saveUser(user);  
 return "redirect:/login?registered";  
 }  
}

HomeController:

package org.example.prostorazvlecaus.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
  
import java.security.Principal;  
  
@Controller  
public class HomeController {  
  
 @GetMapping("/")  
 public String home(Model model, Principal principal) {  
 if (principal != null) {  
 model.addAttribute("username", principal.getName());  
 }  
 return "index";  
 }  
  
 @GetMapping("/guide")  
 public String userGuide() {  
 return "guide";  
 }  
}

OrderControllers:

package org.example.prostorazvlecaus.controller;  
  
import org.example.prostorazvlecaus.model.Order;  
import org.example.prostorazvlecaus.model.Service;  
import org.example.prostorazvlecaus.model.Stone;  
import org.example.prostorazvlecaus.service.OrderService;  
import org.example.prostorazvlecaus.service.ServiceService;  
import org.example.prostorazvlecaus.service.StonesService;  
import org.springframework.security.access.prepost.PreAuthorize;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@Controller  
@PreAuthorize("hasRole('ADMIN')")  
@RequestMapping("/order")  
public class OrderControllers {  
  
 private final OrderService orderService;  
 private final StonesService stonesService;  
 private final ServiceService serviceService;  
  
 public OrderControllers(OrderService orderService , StonesService stonesService , ServiceService serviceService) {  
 this.orderService = orderService;  
 this.stonesService = stonesService;  
 this.serviceService = serviceService;  
 }  
  
 @GetMapping("/listOrder")  
 public String listOrder(Model model , Order order ,Stone stone , Service service) {  
 List<Order> orders = orderService.getOrder();  
 model.addAttribute("orders" , orders);  
 model.addAttribute("stone" , stone);  
 model.addAttribute("service" , service);  
 return "/order/listOrder";  
 }  
  
 @GetMapping("/newOrder")  
 public String newOrder(Model model, Order order) {  
 model.addAttribute("order", order);  
 List<Stone> stones = stonesService.getAllStones();  
 List<Service> services = serviceService.getAllServices();  
 model.addAttribute("allServices", services);  
 model.addAttribute("allStones", stones);  
 return "/order/newOrder";  
 }  
  
 @PostMapping("/saveOrder")  
 public String addOrder(@ModelAttribute Order order) {  
 orderService.addOrder(order);  
 return "redirect:/order/listOrder";  
 }  
  
 @GetMapping("/delete/{id}")  
 public String deleteOrder(@PathVariable Long id){  
 orderService.deleteOrder(id);  
 return "redirect:/order/listOrder";  
 }  
  
 @GetMapping("/create/{id}")  
 public String createOrder(@PathVariable Long id, Model model) {  
 Order order = orderService.getOrder(id);  
 model.addAttribute("order", order);  
 model.addAttribute("allServices", serviceService.getAllServices());  
 model.addAttribute("allStones", stonesService.getAllStones());  
 return "/order/createOrder";  
 }  
  
 @PostMapping("/create/{id}")  
 public String createOrder(@ModelAttribute Order order, @PathVariable Long id) {  
 order.setId(id);  
 orderService.addOrder(order);  
 return "redirect:/order/listOrder";  
 }  
}

ServiceControllers:

package org.example.prostorazvlecaus.controller;  
  
import org.example.prostorazvlecaus.model.Service;  
import org.example.prostorazvlecaus.model.ServiceType;  
import org.example.prostorazvlecaus.service.ServiceService;  
import org.springframework.security.access.prepost.PreAuthorize;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@Controller  
@PreAuthorize("hasRole('ADMIN')")  
@RequestMapping("/service")  
public class ServiceControllers {  
  
 private final ServiceService serviceService;  
  
 public ServiceControllers(ServiceService serviceService) {  
 this.serviceService = serviceService;  
 }  
  
 @GetMapping("/listService")  
 public String listService(Model model , Service service) {  
 List<Service> services = serviceService.getAllServices();  
 model.addAttribute("services" , services);  
 return "service/listService";  
 }  
  
 @GetMapping("/newService")  
 public String newService(Model model , Service service , ServiceType serviceType) {  
 model.addAttribute("service", new Service());  
 model.addAttribute("serviceTypes", ServiceType.*values*());  
 return "service/newService";  
 }  
  
 @PostMapping("/saveService")  
 public String addService(@ModelAttribute Service service) {  
 serviceService.addService(service);  
 return "redirect:/service/listService";  
 }  
  
 @GetMapping("/delete/{id}")  
 public String deleteService(@PathVariable Long id){  
 serviceService.deleteService(id);  
 return "redirect:/service/listService";  
 }  
  
 @GetMapping("/create/{id}")  
 public String createService(@PathVariable Long id, Model model) {  
 Service service = serviceService.getServiceById(id);  
 model.addAttribute("service", service);  
 model.addAttribute("serviceTypes", ServiceType.*values*());  
 return "service/createService";  
 }  
  
 @PostMapping("/create/{id}")  
 public String createService(@ModelAttribute Service service, @PathVariable Long id) {  
 service.setId(id);  
 serviceService.addService(service);  
 return "redirect:/service/listService";  
 }  
}

StonesControllers:

package org.example.prostorazvlecaus.controller;  
  
import org.example.prostorazvlecaus.model.Stone;  
import org.example.prostorazvlecaus.service.StonesService;  
import org.springframework.security.access.prepost.PreAuthorize;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@Controller  
@RequestMapping("/stone")  
@PreAuthorize("hasRole('ADMIN')")  
public class StonesControllers {  
  
 public final StonesService stonesService;  
  
 public StonesControllers(StonesService stonesService) {  
 this.stonesService = stonesService;  
 }  
  
 @GetMapping("/list")  
 public String listStones(Model model , Stone stone){  
 List<Stone> stones = stonesService.getAllStones();  
 model.addAttribute("stones", stones);  
 return "/stone/list";  
 }  
  
 @GetMapping("/new")  
 public String newStone(Model model , Stone stone){  
 model.addAttribute("stone", new Stone());  
 return "/stone/newStone";  
 }  
  
 @PostMapping("/save")  
 public String saveStone(@ModelAttribute Stone stone){  
 stonesService.saveStones(stone);  
 return "redirect:/stone/list";  
 }  
  
 @GetMapping("/delete/{id}")  
 public String deleteStone(@PathVariable Long id){  
 stonesService.deleteStones(id);  
 return "redirect:/stone/list";  
 }  
  
 @GetMapping("/create/{id}")  
 public String editStoneForm(@PathVariable Long id, Model model) {  
 model.addAttribute("stone", stonesService.getStones(id));  
 return "/stone/createStone";  
 }  
  
 @PostMapping("/create/{id}")  
 public String createStone(@ModelAttribute Stone stone, @PathVariable Long id) {  
 stone.setId(id);  
 stonesService.saveStones(stone);  
 return "redirect:/stone/list";  
 }  
}

UserControllers:

package org.example.prostorazvlecaus.controller;  
  
import org.example.prostorazvlecaus.model.User;  
import org.example.prostorazvlecaus.service.UserService;  
import org.springframework.security.access.prepost.PreAuthorize;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
  
import java.util.List;  
  
@Controller  
@RequestMapping("/user")  
@PreAuthorize("hasRole('ADMIN')")  
public class UserControllers {  
  
 public final UserService userService;  
  
 public UserControllers(UserService userService) {  
 this.userService = userService;  
 }  
  
 @GetMapping("/listUser")  
 public String listUser(Model model , User user){  
 List<User> users = userService.getAllUsers();  
 model.addAttribute("users", users);  
 return "/user/listUser";  
 }  
  
 @GetMapping("/delete/{id}")  
 public String deleteUser(@PathVariable Long id){  
 userService.deleteUser(id);  
 return "redirect:/user/listUser";  
 }  
  
 @GetMapping("/search")  
 public String searchUser(@RequestParam String searchTerm, Model model) {  
 List<User> users = userService.searchUser(searchTerm);  
 model.addAttribute("users", users);  
 return "user/listUser";  
 }  
}

Order:

package org.example.prostorazvlecaus.model;  
  
import jakarta.persistence.\*;  
  
import java.util.Date;  
  
@Entity  
@Table(name = "orders")  
public class Order {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String client;  
 private String weight;  
 private String totalPrice;  
 private Date OrderCreationDate;  
  
 @ManyToOne  
 @JoinColumn(name = "stone\_id")  
 private Stone stone;  
  
 @ManyToOne  
 @JoinColumn(name = "service\_id")  
 private Service service;  
  
  
 public Order() {  
 }  
  
 public Order(Long id, String client, String weight, String totalPrice, Date orderCreationDate) {  
 this.id = id;  
 this.client = client;  
 this.weight = weight;  
 this.totalPrice = totalPrice;  
 OrderCreationDate = orderCreationDate;}  
  
 public Stone getStone() {  
 return stone;  
 }  
  
 public void setStone(Stone stone) {  
 this.stone = stone;  
 }  
  
 public Service getService() {  
 return service;  
 }  
  
 public void setService(Service service) {  
 this.service = service;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getWeight() {  
 return weight;  
 }  
  
 public void setWeight(String weight) {  
 this.weight = weight;  
 }  
  
 public String getTotalPrice() {  
 return totalPrice;  
 }  
  
 public void setTotalPrice(String toralPrice) {  
 this.totalPrice = toralPrice;  
 }  
  
 public Date getOrderCreationDate() {  
 return OrderCreationDate;  
 }  
  
 public void setOrderCreationDate(Date orderCreationDate) {  
 OrderCreationDate = orderCreationDate;  
 }  
  
 public String getClient() {  
 return client;  
 }  
  
 public void setClient(String client) {  
 this.client = client;  
 }  
}

Role:

package org.example.prostorazvlecaus.model;  
  
import jakarta.persistence.\*;  
import java.util.ArrayList;  
import java.util.Collection;  
  
@Entity  
@Table(name = "roles")  
public class Role {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false, unique = true)  
 private String name;  
   
 @ManyToMany(mappedBy = "roles")  
 private Collection<User> users = new ArrayList<>();  
  
 public Role() {  
 }  
  
 public Role(String name) {  
 this.name = name;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Collection<User> getUsers() {  
 return users;  
 }  
  
 public void setUsers(Collection<User> users) {  
 this.users = users;  
 }  
}

Service:

package org.example.prostorazvlecaus.model;  
  
import jakarta.persistence.\*;  
  
import java.util.List;  
  
@Entity  
@Table(name = "service")  
public class Service {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String name;  
 private String serviceType;  
  
 @OneToMany(mappedBy = "service")  
 private List<Order> orders;  
  
 public Service() {  
 }  
  
 public Service(Long id, String name, String serviceType) {  
 this.id = id;  
 this.name = name;  
 this.serviceType = serviceType;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public List<Order> getOrders() {  
 return orders;  
 }  
  
 public void setOrders(List<Order> orders) {  
 this.orders = orders;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getServiceType() {  
 return serviceType;  
 }  
  
 public void setServiceType(String serviceType) {  
 this.serviceType = serviceType;  
 }  
  
 public String getServiceTypeDisplayName() {  
 if (serviceType == null) return "";  
 try {  
 return org.example.prostorazvlecaus.model.ServiceType.*valueOf*(serviceType).getDisplayName();  
 } catch (Exception e) {  
 return serviceType;  
 }  
 }  
}

ServiceType:

package org.example.prostorazvlecaus.model;  
  
public enum ServiceType {  
 *CUTTING*("Резка"),  
 *POLISHING*("Полировка"),  
 *GRINDING*("Шлифовка"),  
 *BURNING*("Обжиг камня"),  
 *THERMAL\_COLOR*("Термообработка цвета"),  
 *LASER\_ENGRAVING*("Лазерная гравировка"),  
 *HAND\_ENGRAVING*("Ручная гравировка"),  
 *BAS\_RELIEF*("Создание барельефа");  
  
 private final String displayName;  
  
 ServiceType(String displayName) {  
 this.displayName = displayName;  
 }  
  
 public String getDisplayName() {  
 return displayName;  
 }  
}

Stone:

package org.example.prostorazvlecaus.model;  
  
import jakarta.persistence.\*;  
  
import java.util.List;  
  
@Entity  
@Table(name = "stone")  
public class Stone {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String name;  
 private String firmnessToMohs; */\* твёрдость по Моосу \*/* private String density; */\* плотность \*/* private String color;  
 private String typeOfStone; */\* тип камня \*/* @OneToMany(mappedBy = "stone")  
 private List<Order> orders;  
  
 public Stone() {}  
  
 public Stone(Long id, String name, String firmnessToMohs, String density, String color, String typeOfStone) {  
 this.id = id;  
 this.name = name;  
 this.firmnessToMohs = firmnessToMohs;  
 this.density = density;  
 this.color = color;  
 this.typeOfStone = typeOfStone;  
 }  
  
 public Long getId() {  
 return id;  
 }  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public List<Order> getOrders() {  
 return orders;  
 }  
  
 public void setOrders(List<Order> orders) {  
 this.orders = orders;  
 }  
  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public String getFirmnessToMohs() {  
 return firmnessToMohs;  
 }  
 public void setFirmnessToMohs(String firmnessToMohs) {  
 this.firmnessToMohs = firmnessToMohs;  
 }  
 public String getDensity() {  
 return density;  
 }  
 public void setDensity(String density) {  
 this.density = density;  
 }  
 public String getColor() {  
 return color;  
 }  
 public void setColor(String color) {  
 this.color = color;  
 }  
 public String getTypeOfStone() {  
 return typeOfStone;  
 }  
 public void setTypeOfStone(String typeOfStone) {  
 this.typeOfStone = typeOfStone;  
 }  
}

User:

package org.example.prostorazvlecaus.model;  
  
import jakarta.persistence.\*;  
import java.util.ArrayList;  
import java.util.Collection;  
  
@Entity  
@Table(name = "users")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String name;  
  
 @Column(nullable = false, unique = true)  
 private String email;  
  
 @Column(nullable = false)  
 private String password;  
  
 @ManyToMany(fetch = FetchType.*EAGER*, cascade = CascadeType.*ALL*)  
 @JoinTable(  
 name = "users\_roles",  
 joinColumns = @JoinColumn(name = "user\_id", referencedColumnName = "id"),  
 inverseJoinColumns = @JoinColumn(name = "role\_id", referencedColumnName = "id")  
 )  
 private Collection<Role> roles = new ArrayList<>();  
  
 public User() {  
 }  
  
 public User(String name, String email, String password, Collection<Role> roles) {  
 this.name = name;  
 this.email = email;  
 this.password = password;  
 this.roles = roles;  
 }  
  
 public User(String name, String email, String password) {  
 this.name = name;  
 this.email = email;  
 this.password = password;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public Collection<Role> getRoles() {  
 return roles;  
 }  
  
 public void setRoles(Collection<Role> roles) {  
 this.roles = roles;  
 }  
  
 public void addRole(Role role) {  
 if (this.roles == null) {  
 this.roles = new ArrayList<>();  
 }  
 this.roles.add(role);  
 }  
}

OrderRepo:

package org.example.prostorazvlecaus.repo;  
  
import org.example.prostorazvlecaus.model.Order;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface OrderRepo extends JpaRepository<Order, Long> {}

RoleRepo:

package org.example.prostorazvlecaus.repo;  
  
import org.example.prostorazvlecaus.model.Role;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface RoleRepository extends JpaRepository<Role, Long> {  
 Role findByName(String name);  
}

ServiceRepo:

package org.example.prostorazvlecaus.repo;  
  
import org.example.prostorazvlecaus.model.Service;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface ServiceRepo extends JpaRepository<Service, Long> {}

StonesRepo:

package org.example.prostorazvlecaus.repo;  
  
import org.example.prostorazvlecaus.model.Stone;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface StonesRepo extends JpaRepository<Stone, Long> {}

UserRepo:

package org.example.prostorazvlecaus.repo;  
  
import org.example.prostorazvlecaus.model.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.List;  
  
public interface UserRepository extends JpaRepository<User, Long> {  
 User findByEmail(String email);  
 List<User> findByNameContainingIgnoreCaseOrEmailContainingIgnoreCase(  
 String name,  
 String email  
 );}

OrderService:

package org.example.prostorazvlecaus.service;  
  
import org.example.prostorazvlecaus.model.Order;  
  
import java.util.List;  
  
public interface OrderService {  
 List<Order> getOrder();  
 void addOrder(Order order);  
 void deleteOrder(Long id);  
 Order getOrder(Long id);  
}

ServiceService:

package org.example.prostorazvlecaus.service;  
  
import org.example.prostorazvlecaus.model.Service;  
  
import java.util.List;  
  
public interface ServiceService {  
 List<Service> getAllServices(); *// Переименуйте этот метод* void addService(Service service);  
 void deleteService(Long id);  
 Service getServiceById(Long id); *// И этот метод*}

StonesService:

package org.example.prostorazvlecaus.service;  
  
  
import org.example.prostorazvlecaus.model.Stone;  
  
import java.util.List;  
  
public interface StonesService {  
 List<Stone> getAllStones();  
 void saveStones(Stone stones);  
 void deleteStones(Long id);  
 Stone getStones(Long id);  
}

UserService:

package org.example.prostorazvlecaus.service;  
  
import org.example.prostorazvlecaus.model.User;  
import java.util.List;  
  
public interface UserService {  
 List<User> getAllUsers();  
 User getUserById(Long id);  
 User getUserByEmail(String email);  
 User saveUser(User user);  
 void deleteUser(Long id);  
 List<User> searchUser(String name);  
}

CustomUserDetailsService:

package org.example.prostorazvlecaus.service.impl;  
  
import org.example.prostorazvlecaus.model.Role;  
import org.example.prostorazvlecaus.model.User;  
import org.example.prostorazvlecaus.repo.UserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
  
import java.util.Collection;  
import java.util.Collections;  
import java.util.stream.Collectors;  
  
@Service  
public class CustomUserDetailsService implements UserDetailsService {  
  
 private final UserRepository userRepository;  
  
 @Autowired  
 public CustomUserDetailsService(UserRepository userRepository) {  
 this.userRepository = userRepository;  
 }  
  
 @Override  
 public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {  
 User user = userRepository.findByEmail(email);  
   
 if (user == null) {  
 throw new UsernameNotFoundException("Пользователь не найден с email: " + email);  
 }  
   
 return new org.springframework.security.core.userdetails.User(  
 user.getEmail(),  
 user.getPassword(),  
 mapRolesToAuthorities(user.getRoles())  
 );  
 }  
   
 private Collection<? extends GrantedAuthority> mapRolesToAuthorities(Collection<Role> roles) {  
 if (roles == null || roles.isEmpty()) {  
 *// Если у пользователя нет ролей, даем ему роль USER по умолчанию* return Collections.*singletonList*(new SimpleGrantedAuthority("ROLE\_USER"));  
 }  
   
 return roles.stream()  
 .map(role -> new SimpleGrantedAuthority("ROLE\_" + role.getName()))  
 .collect(Collectors.*toList*());  
 }  
}

implOrder:

package org.example.prostorazvlecaus.service.impl;  
  
import org.example.prostorazvlecaus.model.Order;  
import org.example.prostorazvlecaus.repo.OrderRepo;  
import org.example.prostorazvlecaus.service.OrderService;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class implOrder implements OrderService {  
  
 private final OrderRepo orderrepo;  
  
 public implOrder(OrderRepo orderrepo) {  
 this.orderrepo = orderrepo;  
 }  
  
 @Override  
 public List<Order> getOrder(){  
 return orderrepo.findAll();  
 }  
  
 @Override  
 public void addOrder(Order order) {  
 orderrepo.save(order);  
 }  
  
 @Override  
 public void deleteOrder(Long id){  
 orderrepo.deleteById(id);  
 }  
  
 @Override  
 public Order getOrder(Long id){  
 return orderrepo.findById(id).orElse(null);  
 }  
}

implService:

package org.example.prostorazvlecaus.service.impl;  
  
  
import org.example.prostorazvlecaus.model.Service;  
import org.example.prostorazvlecaus.repo.ServiceRepo;  
import org.example.prostorazvlecaus.service.ServiceService;  
  
import java.util.List;  
  
@org.springframework.stereotype.Service  
public class implServices implements ServiceService {  
  
 private final ServiceRepo servicerepo;  
  
 public implServices(ServiceRepo servicerepo) {  
 this.servicerepo = servicerepo;  
 }  
  
 @Override  
 public void addService(Service service) {  
 servicerepo.save(service);  
 }  
  
 @Override  
 public void deleteService(Long id){  
 servicerepo.deleteById(id);  
 }  
  
 @Override  
 public List<Service> getAllServices() {  
 return servicerepo.findAll();  
 }  
  
 @Override  
 public Service getServiceById(Long id) {  
 return servicerepo.findById(id).orElse(null);  
 }  
}

implStones:

package org.example.prostorazvlecaus.service.impl;  
  
  
import org.example.prostorazvlecaus.model.Service;  
import org.example.prostorazvlecaus.repo.ServiceRepo;  
import org.example.prostorazvlecaus.service.ServiceService;  
  
import java.util.List;  
  
@org.springframework.stereotype.Service  
public class implServices implements ServiceService {  
  
 private final ServiceRepo servicerepo;  
  
 public implServices(ServiceRepo servicerepo) {  
 this.servicerepo = servicerepo;  
 }  
  
 @Override  
 public void addService(Service service) {  
 servicerepo.save(service);  
 }  
  
 @Override  
 public void deleteService(Long id){  
 servicerepo.deleteById(id);  
 }  
  
 @Override  
 public List<Service> getAllServices() {  
 return servicerepo.findAll();  
 }  
  
 @Override  
 public Service getServiceById(Long id) {  
 return servicerepo.findById(id).orElse(null);  
 }  
}

UserServiceImpl:

package org.example.prostorazvlecaus.service.impl;  
  
import org.example.prostorazvlecaus.model.Role;  
import org.example.prostorazvlecaus.model.User;  
import org.example.prostorazvlecaus.repo.RoleRepository;  
import org.example.prostorazvlecaus.repo.UserRepository;  
import org.example.prostorazvlecaus.service.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
  
import java.util.Arrays;  
import java.util.List;  
  
@Service  
public class UserServiceImpl implements UserService {  
  
 private final UserRepository userRepository;  
 private final RoleRepository roleRepository;  
 private final PasswordEncoder passwordEncoder;  
  
 @Autowired  
 public UserServiceImpl(UserRepository userRepository,   
 RoleRepository roleRepository,   
 PasswordEncoder passwordEncoder) {  
 this.userRepository = userRepository;  
 this.roleRepository = roleRepository;  
 this.passwordEncoder = passwordEncoder;  
 }  
  
 @Override  
 public List<User> getAllUsers() {  
 return userRepository.findAll();  
 }  
  
 @Override  
 public User getUserById(Long id) {  
 return userRepository.findById(id).orElse(null);  
 }  
  
 @Override  
 public User getUserByEmail(String email) {  
 return userRepository.findByEmail(email);  
 }  
  
 @Override  
 public User saveUser(User user) {  
 *// Если это новый пользователь (регистрация), то назначаем роль и шифруем пароль* if(user.getId() == null) {  
 user.setPassword(passwordEncoder.encode(user.getPassword()));  
   
 *// Проверяем количество пользователей в системе* long userCount = userRepository.count();  
   
 *// Если первый пользователь, то даем ему роль ADMIN* if(userCount == 0) {  
 Role adminRole = roleRepository.findByName("ADMIN");  
   
 if(adminRole == null) {  
 adminRole = new Role("ADMIN");  
 roleRepository.save(adminRole);  
 }  
   
 user.setRoles(Arrays.*asList*(adminRole));  
 } else {  
 *// Для всех остальных пользователей роль USER* Role userRole = roleRepository.findByName("USER");  
   
 if(userRole == null) {  
 userRole = new Role("USER");  
 roleRepository.save(userRole);  
 }  
   
 user.setRoles(Arrays.*asList*(userRole));  
 }  
 }  
   
 return userRepository.save(user);  
 }  
  
 @Override  
 public void deleteUser(Long id) {  
 userRepository.deleteById(id);  
 }  
  
 @Override  
 public List<User> searchUser(String searchTerm) {  
 return userRepository.findByNameContainingIgnoreCaseOrEmailContainingIgnoreCase(  
 searchTerm,  
 searchTerm  
 );  
 }  
}

style.css:

*/\* Общие стили для всего приложения \*/*body {  
 font-family: Arial, sans-serif;  
 background: #fafafa;  
 margin: 0;  
 padding: 0;  
}  
  
header {  
 background: #1976d2;  
 color: white;  
 padding: 10px 0;  
 text-align: center;  
}  
  
h1, h2 {  
 text-align: center;  
}  
  
form {  
 max-width: 400px;  
 margin: 30px auto;  
 padding: 20px;  
 border: 1px solid #ccc;  
 background: #fff;  
 border-radius: 8px;  
}  
  
label {  
 display: block;  
 margin-bottom: 5px;  
 font-weight: bold;  
}  
  
input, select {  
 width: 100%;  
 padding: 6px 8px;  
 margin-bottom: 15px;  
 border: 1px solid #bbb;  
 border-radius: 4px;  
}  
  
button, a {  
 padding: 6px 14px;  
 border: none;  
 background: #1976d2;  
 color: #fff;  
 text-decoration: none;  
 border-radius: 4px;  
 cursor: pointer;  
 margin-right: 8px;  
}  
  
button:hover, a:hover {  
 background: #125ea2;  
}  
  
table {  
 width: 100%;  
 border-collapse: collapse;  
 margin: 20px 0;  
}  
  
th, td {  
 padding: 10px;  
 border: 1px solid #ddd;  
 text-align: left;  
}  
  
th {  
 background-color: #f2f2f2;  
}  
*/\* Стили для заголовка и навигационной панели \*/*header {  
 background-color: #1976d2; */\* Цвет фона заголовка \*/* padding: 10px 0; */\* Отступы сверху и снизу \*/*}  
  
.navbar-nav {  
 list-style: none; */\* Убираем маркеры списка \*/* padding: 0; */\* Убираем отступы \*/* margin: 0; */\* Убираем маргины \*/* display: flex; */\* Используем flexbox для выравнивания \*/* justify-content: center; */\* Центрируем элементы \*/*}  
  
.navbar-nav li {  
 margin: 0 15px; */\* Отступы между элементами \*/*}  
  
.navbar-nav a {  
 color: white; */\* Цвет текста ссылок \*/* text-decoration: none; */\* Убираем подчеркивание \*/* font-weight: bold; */\* Жирный шрифт для ссылок \*/* padding: 5px 10px; */\* Отступы внутри ссылок \*/* border-radius: 4px; */\* Скругленные углы \*/* transition: background-color 0.3s; */\* Плавный переход для фона \*/*}  
  
.navbar-nav a:hover {  
 background-color: #125ea2; */\* Цвет фона при наведении \*/*}

login.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Вход в систему</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.7.2/font/bootstrap-icons.css">  
</head>  
<body>  
<div class="container py-5">  
 <div class="row justify-content-center">  
 <div class="col-md-6">  
 <div class="card shadow">  
 <div class="card-header bg-primary text-white text-center">  
 <h1 class="h4 mb-0">Вход в систему</h1>  
 </div>  
 <div class="card-body">  
 <div th:if="${error}" class="alert alert-danger">  
 <i class="bi bi-exclamation-triangle"></i> Неверный email или пароль.  
 </div>  
 <div th:if="${param.registered}" class="alert alert-success">  
 <i class="bi bi-check-circle"></i> Регистрация успешна. Теперь вы можете войти.  
 </div>  
 <div th:if="${logout}" class="alert alert-success">  
 <i class="bi bi-check-circle"></i> Вы успешно вышли из системы.  
 </div>  
 <form th:action="@{/login}" method="post">  
 <div class="mb-3">  
 <label for="username" class="form-label"><i class="bi bi-envelope"></i> Email</label>  
 <input type="email" id="username" name="username" class="form-control" required>  
 </div>  
 <div class="mb-3">  
 <label for="password" class="form-label"><i class="bi bi-key"></i> Пароль</label>  
 <input type="password" id="password" name="password" class="form-control" required>  
 </div>  
 <div class="d-grid gap-2">  
 <button type="submit" class="btn btn-primary">  
 <i class="bi bi-box-arrow-in-right"></i> Войти  
 </button>  
 </div>  
 </form>  
 <hr>  
 <div class="text-center">  
 <p>Нет аккаунта? <a th:href="@{/register}" class="text-decoration-none">Зарегистрироваться</a></p>  
 <a th:href="@{/}" class="text-decoration-none">Вернуться на главную</a>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>  
</body>  
</html>

register.html:  
<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Регистрация</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.7.2/font/bootstrap-icons.css">  
 <link rel="stylesheet" th:href="@{/css/custom.css}">  
</head>  
<body>  
<div class="container py-5">  
 <div class="row justify-content-center">  
 <div class="col-md-6">  
 <div class="card shadow">  
 <div class="card-header bg-primary text-white text-center">  
 <h1 class="h4 mb-0">Регистрация нового пользователя</h1>  
 </div>  
 <div class="card-body">  
 <div th:if="${error}" class="alert alert-danger">  
 <i class="bi bi-exclamation-triangle"></i> <span th:text="${error}"></span>  
 </div>  
   
 <form th:action="@{/register-process}" th:object="${user}" method="post" class="needs-validation">  
 <div class="mb-3">  
 <label for="name" class="form-label"><i class="bi bi-person"></i> Имя</label>  
 <input type="text" id="name" th:field="\*{name}" class="form-control" required   
 minlength="2" maxlength="50">  
 <div class="invalid-feedback" th:if="${#fields.hasErrors('name')}" th:errors="\*{name}">  
 Пожалуйста, укажите имя.  
 </div>  
 </div>  
   
 <div class="mb-3">  
 <label for="email" class="form-label"><i class="bi bi-envelope"></i> Email</label>  
 <input type="email" id="email" th:field="\*{email}" class="form-control" required>  
 <div class="invalid-feedback" th:if="${#fields.hasErrors('email')}" th:errors="\*{email}">  
 Пожалуйста, укажите корректный email.  
 </div>  
 </div>  
   
 <div class="mb-3">  
 <label for="password" class="form-label"><i class="bi bi-key"></i> Пароль</label>  
 <input type="password" id="password" th:field="\*{password}" class="form-control"   
 required minlength="6">  
 <div class="invalid-feedback" th:if="${#fields.hasErrors('password')}" th:errors="\*{password}">  
 Пароль должен быть не менее 6 символов.  
 </div>  
 </div>  
   
 <div class="d-grid gap-2">  
 <button type="submit" class="btn btn-success">  
 <i class="bi bi-person-plus"></i> Зарегистрироваться  
 </button>  
 </div>  
 </form>  
   
 <hr>  
 <div class="text-center">  
 <p>Уже есть аккаунт? <a th:href="@{/login}" class="text-decoration-none">Войти</a></p>  
 <a th:href="@{/}" class="text-decoration-none">Вернуться на главную</a>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>  
</body>  
</html>

createOrder.html

<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Редактирование</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
  
<h1>Редактирование</h1>  
<form th:action="@{/order/create/{id}(id=${order.id})}" th:object="${order}" method="post">  
 <div>  
 <label>Клиент ФИО</label>  
 <input type="text" th:field="\*{client}" required>  
 </div>  
 <div>  
 <label>Камень</label>  
 <select th:field="\*{stone}" required>  
 <option value="">-- Выберите услугу --</option>  
 <option th:each="stone : ${allStones}"  
 th:value="${stone.id}"  
 th:text="${stone.name}"  
 th:selected="${stone.id == order.stone.id}"></option>  
 </select>  
 </div>  
 <div>  
 <label>Услуга</label>  
 <select th:field="\*{service}" required>  
 <option th:each="service : ${allServices}"  
 th:value="${service.id}"  
 th:text="${service.name}"  
 th:selected="${service.id == order.service.id}"></option>  
 </select>  
 </div>  
 <div>  
 <label>Вес камня</label>  
 <input type="number" th:field="\*{weight}" required>  
 </div>  
 <div>  
 <label>Стоимость заказа</label>  
 <input type="number" th:field="\*{totalPrice}" required>  
 </div>  
 <div>  
 <label>Дата заказа</label>  
 <input type="date" th:field="\*{orderCreationDate}" required>  
 </div>  
 <a th:href="@{listOrder}">Назад</a>  
 <button type="submit">Сохранить</button>  
</form>  
</body>  
</html>

listOrder.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Заказы</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
<h2>Список заказов</h2>  
<a th:href="@{newOrder}">Добавить заказ</a>  
<table>  
 <tr>  
 <th>id</th>  
 <th>Клиент</th>  
 <th>Камень</th>  
 <th>Услуга</th>  
 <th>Вес камня</th>  
 <th>Стоимость заказа</th>  
 <th>Дата заказа</th>  
 <th>Действие</th>  
 </tr>  
 <tr th:each="order : ${orders}">  
 <td th:text="${order.id}"></td>  
 <td th:text="${order.client}"></td>  
 <td th:text="${order.stone.name}"></td>  
 <td th:text="${order.service.name}"></td>  
 <td th:text="${order.weight}"></td>  
 <td th:text="${order.totalPrice}"></td>  
 <td th:text="${#dates.format(order.orderCreationDate, 'dd.MM.yyyy')}"></td>  
 <td>  
 <a th:href="@{/order/delete/{id}(id=${order.id})}">Удалить</a>  
 <a th:href="@{/order/create/{id}(id=${order.id})}">Редактировать</a>  
 </td>  
 </tr>  
</table>  
</body>  
</html>

newOrder.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Добавить заказ</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
</head>  
<body>  
  
<header th:replace="~{header :: header}"></header>  
  
<h1>Добавить заказ</h1>  
<form th:action="@{/order/saveOrder}" th:object="${order}" method="post">  
 <div>  
 <label>Клиент ФИО</label>  
 <input type="text" th:field="\*{client}" required>  
 </div>  
 <div>  
 <label>Камень</label>  
 <select th:field="\*{stone}" required>  
 <option value="">-- Выберите камень --</option>  
 <option th:each="stone : ${allStones}"  
 th:value="${stone?.id}"  
 th:text="${stone?.name}"  
 th:if="${stone != null}">  
 </option>  
 </select>  
 </div>  
  
 <div>  
 <label>Услуги</label>  
 <select th:field="\*{service}" required>  
 <option value="">-- Выберите тип услуги --</option>  
 <option th:each="service : ${allServices}"  
 th:value="${service.id}"  
 th:text="${service.name}"  
 th:if="${service != null}">  
 </option>  
 </select>  
 </div>  
  
 <div>  
 <label>Вес камня</label>  
 <input type="number" th:field="\*{weight}" required>  
 </div>  
 <div>  
 <label>Стоимость заказа</label>  
 <input type="number" th:field="\*{totalPrice}" required min="1">  
 </div>  
 <div>  
 <label>Дата заказа</label>  
 <input type="date" th:field="\*{orderCreationDate}" required>  
 </div>  
 <a th:href="@{listOrder}">Назад</a>  
 <button type="submit">Сохранить</button>  
</form>  
</body>  
</html>

createService.html:

<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Редактирование</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
  
<h1>Редактирование</h1>  
<form th:action="@{/service/create/{id}(id=${service.id})}" th:object="${service}" method="post">  
 <div>  
 <label>Название услуги</label>  
 <select th:field="\*{serviceType}" required>  
 <option value="">-- Выберите тип услуги --</option>  
 <option th:each="type : ${serviceTypes}"  
 th:value="${type}"  
 th:text="${type.displayName}">  
 </option>  
 </select>  
 </div>  
  
 <a th:href="@{listService}">Назад</a>  
 <button type="submit">Сохранить</button>  
</form>  
</body>  
</html>

listService:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Услуги</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
<h2>Список услуг</h2>  
<a th:href="@{newService}">Добавить услугу</a>  
<table>  
 <tr>  
 <th>id</th>  
 <th>Название услуги</th>  
 <th>Действие</th>  
 </tr>  
 <tr th:each="service : ${services}">  
 <td th:text="${service.id}"></td>  
 <td th:text="${service.serviceTypeDisplayName}"></td>  
 <td>  
 <a th:href="@{/service/delete/{id}(id=${service.id})}">Удалить</a>  
 <a th:href="@{/service/create/{id}(id=${service.id})}">Редактировать</a>  
 </td>  
 </tr>  
</table>  
</body>  
</html>

newService:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Добавить услугу</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
<h1>Добавить услугу</h1>  
<form th:action="@{/service/saveService}" th:object="${service}" method="post">  
 <div>  
 <label>Название услуги</label>  
 <select th:field="\*{serviceType}" required>  
 <option value="">-- Выберите тип услуги --</option>  
 <option th:each="type : ${serviceTypes}"  
 th:value="${type}"  
 th:text="${type.displayName}">  
 </option>  
 </select>  
 </div>  
 <a th:href="@{listService}">Назад</a>  
 <button type="submit">Сохранить</button>  
</form>  
</body>  
</html>

createStone.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Добавить услугу</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
<h1>Добавить услугу</h1>  
<form th:action="@{/service/saveService}" th:object="${service}" method="post">  
 <div>  
 <label>Название услуги</label>  
 <select th:field="\*{serviceType}" required>  
 <option value="">-- Выберите тип услуги --</option>  
 <option th:each="type : ${serviceTypes}"  
 th:value="${type}"  
 th:text="${type.displayName}">  
 </option>  
 </select>  
 </div>  
 <a th:href="@{listService}">Назад</a>  
 <button type="submit">Сохранить</button>  
</form>  
</body>  
</html>

listStone.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Список</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
  
 <h2>Список камней</h2>  
 <a th:href="@{new}">Добавить камень</a>  
 <table>  
 <tr>  
 <th>id</th>  
 <th>Название камня</th>  
 <th>Твёрдость по Моосу</th>  
 <th>Плотность</th>  
 <th>Цвет</th>  
 <th>Тип камня</th>  
 <th>Действие</th>  
 </tr>  
 <tr th:each="stone : ${stones}">  
 <td th:text="${stone.id}"></td>  
 <td th:text="${stone.name}"></td>  
 <td th:text="${stone.firmnessToMohs}"></td>  
 <td th:text="${stone.density}"></td>  
 <td th:text="${stone.color}"></td>  
 <td th:text="${stone.typeOfStone}"></td>  
 <td>  
 <a th:href="@{/stone/delete/{id}(id=${stone.id})}">Удалить</a>  
 <a th:href="@{/stone/create/{id}(id=${stone.id})}">Редактировать</a>  
 </td>  
 </tr>  
 </table>  
</body>  
</html>

newStone:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Добавить камень</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
 <h1>Добавить камень</h1>  
 <form th:action="@{/stone/save}" th:object="${stone}" method="post">  
 <div>  
 <label>Название камня</label>  
 <select th:field="\*{name}" required>  
 <optgroup label="Мрамор и мраморовидные породы">  
 <option th:value="'Белый мрамор (Каррара, Статуарио)'">Белый мрамор (Каррара, Статуарио)</option>  
 <option th:value="'Цветной мрамор (Крема Марфил, Боттичино, Эмпердор Дарк)'">Цветной мрамор (Крема Марфил, Боттичино, Эмпердор Дарк)</option>  
 <option th:value="'Оникс (полупрозрачный слоистый камень)'">Оникс (полупрозрачный слоистый камень)</option>  
 <option th:value="'Травертин (пористый известняк)'">Травертин (пористый известняк)</option>  
 </optgroup>  
 <optgroup label="Гранит">  
 <option th:value="'Абсолют Блэк (черный)'">Абсолют Блэк (черный)</option>  
 <option th:value="'Бьянко Антарктика (белый)'">Бьянко Антарктика (белый)</option>  
 <option th:value="'Габбро (темно-зеленый)'">Габбро (темно-зеленый)</option>  
 <option th:value="'Балморал Ред (красный)'">Балморал Ред (красный)</option>  
 </optgroup>  
 <optgroup label="Кварциты и кварцевые агломераты">  
 <option th:value="'Белый кварцит'">Белый кварцит</option>  
 <option th:value="'Таицкий кварцит (розовый)'">Таицкий кварцит (розовый)</option>  
 <option th:value="'Искусственный кварцевый камень (бренды типа Silestone, Caesarstone)'">Искусственный кварцевый камень (бренды типа Silestone, Caesarstone)</option>  
 </optgroup>  
 <optgroup label="Известняки">  
 <option th:value="'Иерусалимский камень'">Иерусалимский камень</option>  
 <option th:value="'Французский известняк'">Французский известняк</option>  
 </optgroup>  
 <optgroup label="Сланцы">  
 <option th:value="'Зеленый сланец'">Зеленый сланец</option>  
 <option th:value="'Фиолетовый сланец'">Фиолетовый сланец</option>  
 </optgroup>  
 <optgroup label="Песчаники">  
 <option th:value="'Радж Грин (Индия)'">Радж Грин (Индия)</option>  
 <option th:value="'Кантри Браун'">Кантри Браун</option>  
 </optgroup>  
 </select>  
 </div>  
 <div>  
 <label>Твёрдость по Моосу</label>  
 <input type="number" th:field="\*{firmnessToMohs}" required min="1" max="10">  
 </div>  
 <div>  
 <label>Плотность</label>  
 <input type="number" th:field="\*{density}" required min="1">  
 </div>  
 <div>  
 <label>Цвет</label>  
 <input type="text" th:field="\*{color}" required>  
 </div>  
 <div>  
 <label>Тип камня</label>  
 <input type="text" th:field="\*{typeOfStone}" required>  
 </div>  
  
 <a th:href="@{list}">Назад</a>  
 <button type="submit">Сохранить</button>  
 </form>  
</body>  
</html>

listUser:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Пользователи</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
  
<form class="form" th:action="@{/user/search}" method="get" >  
 <input type="text" name="searchTerm" placeholder="Поиск по ФИО">  
 <button type="submit">Поиск</button>  
 <a th:href="@{/user/listUser}">Сбросить</a>  
</form>  
  
<h2>Список пользователей</h2>  
<table>  
 <tr>  
 <th>id</th>  
 <th>ФИО</th>  
 <th>Email</th>  
 <th>Действие</th>  
 </tr>  
 <tr th:each="user : ${users}">  
 <td th:text="${user.id}"></td>  
 <td th:text="${user.name}"></td>  
 <td th:text="${user.email}"></td>  
 <td>  
 <a th:href="@{/user/delete/{id}(id=${user.id})}">Удалить</a>  
 </td>  
 </tr>  
</table>  
</body>  
</html>

Gaite.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity6">  
<head>  
 <meta charset="UTF-8">  
 <title>Руководство пользователя</title>  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
 <style>  
 .manual-container {  
 max-width: 1200px;  
 margin: 20px auto;  
 padding: 20px;  
 background-color: #fff;  
 box-shadow: 0 0 10px rgba(0,0,0,0.1);  
 }  
 .section {  
 margin-bottom: 40px;  
 border-bottom: 1px solid #eee;  
 padding-bottom: 20px;  
 }  
 h1 {  
 color: #2c3e50;  
 text-align: center;  
 margin-bottom: 30px;  
 }  
 h2 {  
 color: #3498db;  
 margin-top: 30px;  
 }  
 h3 {  
 color: #16a085;  
 margin-top: 20px;  
 }  
 .feature-card {  
 background: #f8f9fa;  
 border-left: 4px solid #3498db;  
 padding: 15px;  
 margin-bottom: 15px;  
 border-radius: 0 4px 4px 0;  
 }  
 .step {  
 display: flex;  
 margin-bottom: 15px;  
 align-items: flex-start;  
 }  
 .step-number {  
 background: #3498db;  
 color: white;  
 width: 25px;  
 height: 25px;  
 border-radius: 50%;  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 margin-right: 15px;  
 flex-shrink: 0;  
 }  
 .role-badge {  
 display: inline-block;  
 padding: 3px 8px;  
 background: #e74c3c;  
 color: white;  
 border-radius: 3px;  
 font-size: 12px;  
 margin-left: 10px;  
 }  
 .admin { background: #e74c3c; }  
 .user { background: #2ecc71; }  
 .guest { background: #f39c12; }  
 </style>  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
<div class="manual-container">  
 <h1>Руководство пользователя</h1>  
  
 <div class="section">  
 <h2>1. Вход в систему</h2>  
  
 <div class="feature-card">  
 <h3>Регистрация нового пользователя <span class="role-badge guest">Гость</span></h3>  
 <div class="step">  
 <div class="step-number">1</div>  
 <div>Нажмите кнопку "Регистрация" в верхнем меню</div>  
 </div>  
 <div class="step">  
 <div class="step-number">2</div>  
 <div>Заполните все обязательные поля формы (имя, email, пароль)</div>  
 </div>  
 <div class="step">  
 <div class="step-number">3</div>  
 <div>Нажмите кнопку "Зарегистрироваться"</div>  
 </div>  
 <div class="step">  
 <div class="step-number">4</div>  
 <div>После успешной регистрации вы будете автоматически перенаправлены на страницу входа</div>  
 </div>  
 </div>  
  
 <div class="feature-card">  
 <h3>Авторизация <span class="role-badge guest">Гость</span></h3>  
 <div class="step">  
 <div class="step-number">1</div>  
 <div>Нажмите кнопку "Войти" в верхнем меню</div>  
 </div>  
 <div class="step">  
 <div class="step-number">2</div>  
 <div>Введите ваш email и пароль</div>  
 </div>  
 <div class="step">  
 <div class="step-number">3</div>  
 <div>Нажмите кнопку "Войти"</div>  
 </div>  
 </div>  
 </div>  
  
 <div class="section">  
 <h2>2. Работа с заказами</h2>  
  
 <div class="feature-card">  
 <h3>Создание нового заказа <span class="role-badge admin">Админ</span></h3>  
 <div class="step">  
 <div class="step-number">1</div>  
 <div>В главном меню выберите "Заказы" → "Добавить заказ"</div>  
 </div>  
 <div class="step">  
 <div class="step-number">2</div>  
 <div>Заполните данные о клиенте (ФИО)</div>  
 </div>  
 <div class="step">  
 <div class="step-number">3</div>  
 <div>Выберите камень из списка доступных</div>  
 </div>  
 <div class="step">  
 <div class="step-number">4</div>  
 <div>Выберите услугу из списка доступных</div>  
 </div>  
 <div class="step">  
 <div class="step-number">5</div>  
 <div>Укажите вес камня и стоимость заказа</div>  
 </div>  
 <div class="step">  
 <div class="step-number">6</div>  
 <div>Нажмите кнопку "Сохранить"</div>  
 </div>  
 </div>  
  
 <div class="feature-card">  
 <h3>Редактирование заказа <span class="role-badge admin">Админ</span></h3>  
 <div class="step">  
 <div class="step-number">1</div>  
 <div>В списке заказов нажмите "Редактировать" у нужного заказа</div>  
 </div>  
 <div class="step">  
 <div class="step-number">2</div>  
 <div>Внесите необходимые изменения в данные заказа</div>  
 </div>  
 <div class="step">  
 <div class="step-number">3</div>  
 <div>Нажмите кнопку "Сохранить" для применения изменений</div>  
 </div>  
 </div>  
 </div>  
  
 <div class="section">  
 <h2>3. Управление камнями</h2>  
  
 <div class="feature-card">  
 <h3>Добавление нового камня <span class="role-badge admin">Админ</span></h3>  
 <div class="step">  
 <div class="step-number">1</div>  
 <div>В главном меню выберите "Камни" → "Добавить камень"</div>  
 </div>  
 <div class="step">  
 <div class="step-number">2</div>  
 <div>Выберите название камня из выпадающего списка</div>  
 </div>  
 <div class="step">  
 <div class="step-number">3</div>  
 <div>Укажите технические характеристики (твердость по Моосу, плотность)</div>  
 </div>  
 <div class="step">  
 <div class="step-number">4</div>  
 <div>Заполните цвет и тип камня</div>  
 </div>  
 <div class="step">  
 <div class="step-number">5</div>  
 <div>Нажмите кнопку "Сохранить"</div>  
 </div>  
 </div>  
 </div>  
  
 <div class="section">  
 <h2>4. Управление услугами</h2>  
  
 <div class="feature-card">  
 <h3>Добавление услуги <span class="role-badge admin">Админ</span></h3>  
 <div class="step">  
 <div class="step-number">1</div>  
 <div>В главном меню выберите "Услуги" → "Добавить услугу"</div>  
 </div>  
 <div class="step">  
 <div class="step-number">2</div>  
 <div>Выберите тип услуги из выпадающего списка</div>  
 </div>  
 <div class="step">  
 <div class="step-number">3</div>  
 <div>Нажмите кнопку "Сохранить"</div>  
 </div>  
 </div>  
 </div>  
  
 <div class="section">  
 <h2>5. Управление пользователями</h2>  
  
 <div class="feature-card">  
 <h3>Поиск пользователей <span class="role-badge admin">Админ</span></h3>  
 <div class="step">  
 <div class="step-number">1</div>  
 <div>В главном меню выберите "Пользователи"</div>  
 </div>  
 <div class="step">  
 <div class="step-number">2</div>  
 <div>Введите ФИО пользователя в поле поиска</div>  
 </div>  
 <div class="step">  
 <div class="step-number">3</div>  
 <div>Нажмите кнопку "Поиск"</div>  
 </div>  
 </div>  
  
 <div class="feature-card">  
 <h3>Удаление пользователя <span class="role-badge admin">Админ</span></h3>  
 <div class="step">  
 <div class="step-number">1</div>  
 <div>В списке пользователей найдите нужного пользователя</div>  
 </div>  
 <div class="step">  
 <div class="step-number">2</div>  
 <div>Нажмите кнопку "Удалить" напротив выбранного пользователя</div>  
 </div>  
 <div class="step">  
 <div class="step-number">3</div>  
 <div>Подтвердите удаление</div>  
 </div>  
 </div>  
 </div>  
</div>  
  
</body>  
</html>

Header.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity6">  
<link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
<head>  
 <meta charset="UTF-8">  
 <title>Header</title>  
</head>  
<body>  
<header th:fragment="header">  
 <div>  
 <ul class="navbar-nav me-auto mb-2 mb-lg-0">  
 <li><a th:href="@{/}">Главная</a></li>  
 <li sec:authorize="hasRole('ADMIN')">  
 <a th:href="@{/stone/list}">Камни</a>  
 </li>  
 <li sec:authorize="hasRole('ADMIN')">  
 <a th:href="@{/service/listService}">Услуги</a>  
 </li>  
 <li sec:authorize="hasRole('ADMIN')">  
 <a th:href="@{/order/listOrder}">Заказы</a>  
 </li>  
 <li sec:authorize="hasRole('ADMIN')">  
 <a th:href="@{/user/listUser}">Пользователи</a>  
 </li>  
 <li sec:authorize="!isAuthenticated()">  
 <a th:href="@{/login}">Войти</a>  
 </li>  
 <li sec:authorize="!isAuthenticated()">  
 <a th:href="@{/register}">Регистрация</a>  
 </li>  
 <li>  
 <a th:href="@{/guide}">О программе</a>  
 </li>  
 <li sec:authorize="isAuthenticated()">  
 <a th:href="@{/logout}">Выйти</a>  
 </li>  
 </ul>  
 </div>  
</header>  
</body>  
</html>

Index.html:

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity6">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" th:href="@{/style.css}" href="/style.css" />  
 <title>Главная страница</title>  
</head>  
<body>  
<header th:replace="~{header :: header}"></header>  
  
<div sec:authorize="isAuthenticated()">  
 <p>Добро пожаловать, <span sec:authentication="name"></span>!</p>  
</div>  
</body>  
</html>

Application.properties

spring.application.name=prostoRazvlecaus  
  
server.port=8084  
spring.datasource.url=jdbc:mysql://localhost:3306/stone  
spring.datasource.username=root  
spring.datasource.password=  
spring.jpa.hibernate.ddl-auto=update  
  
spring.mvc.format.date=yyyy-MM-dd  
  
*#spring.datasource.url=jdbc:postgresql://localhost:5432/practica1  
#spring.datasource.username=postgres  
#spring.datasource.password=admin  
#spring.jpa.hibernate.ddl-auto=update*