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 | Григорьев Д. В. |
| 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 | 23 |
| 2 РАЗРАБОТКА И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ | 32 |
| 2.1 Тестирование клиентской части | 32 |
| 2.2 Руководство пользователя | 33 |
| ЗАКЛЮЧЕНИЕ | 36 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 37 |
| ПРИЛОЖЕНИЕ | 38 |

ВВЕДЕНИЕ

В настоящее время, в эпоху цифровых технологий, автоматизация процессов в различных сферах деятельности становится необходимостью для повышения эффективности и конкурентоспособности. Ателье «Грация» стремится предоставить своим клиентам качественную одежду и услуги по ремонту и подбору гардероба, а также упростить работу своего персонала. Для этого необходимо разработать и внедрить информационную систему, способную автоматизировать процессы оформления заказов, учета клиентов и сотрудников, а также возможность работы с поставщиками.

Объектом исследования является ателье «Грация», его структура, бизнес-процессы и информационные потоки.

Предметом исследования является разработка и внедрение информационной системы для ателье «Грация», способной автоматизировать процессы оформления заказов, учета клиентов и сотрудников, а также возможность работы с поставщиками.

Целью исследования является разработка и внедрение информационной системы для ателье «Грация», способной автоматизировать процессы оформления заказов, учета клиентов и сотрудников, а также возможность работы с поставщиками и особенностей объекта исследования.

Для достижения этой цели были сформулированы следующие задачи исследования:

- Анализ структуры, бизнес-процессов и информационных потоков ателье «Грация».

- Определение требований заказчика к информационной системе.

- Разработка концептуальной модели информационной системы.

- Разработка архитектуры информационной системы.

- Разработка базы данных информационной системы.

- Разработка модулей оформления заказов, учета клиентов и сотрудников, работы с поставщиками.

- Тестирование и отладка информационной системы.

- Внедрение информационной системы в ателье «Грация».

- Подготовка документации к информационной системе.

Для достижения поставленных задач были использованы следующие методы:

- Анализ документов и материалов, относящихся к ателье «Грация».

- Интервьюирование сотрудников ателье «Грация» для выявления бизнес-процессов и информационных потоков.

- Моделирование бизнес-процессов и информационных потоков с помощью специализированного программного обеспечения.

- Разработка концептуальной модели и архитектуры информационной системы с помощью UML-диаграмм.

- Разработка базы данных информационной системы с помощью SQL-языка.

- Разработка модулей оформления заказов, учета клиентов и сотрудников, работы с поставщиками с помощью Java-языка.

- Тестирование и отладка информационной системы с помощью специализированного программного обеспечения.

- Внедрение информационной системы в ателье «Грация» с учетом требований заказчика и особенностей объекта исследования.

- Подготовка документации к информационной системе с учетом требований заказчика и нормативных документов.

1 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

* 1. Анализ предметной области

Ателье «Грация» — это организация, предоставляющая услуги по пошиву, ремонту и изменению одежды, а также консультации по стилю и подбору гардероба.

Сотрудников ателье «Грация» можно подразделить на несколько категорий: мастера-портные, закройщики и администраторы. Каждая категория работников имеет свои уникальные атрибуты и характеристики, определяемые их профессиональными обязанностями.

Мастера-портные занимаются непосредственным пошивом и ремонтом одежды, выполняют замеры клиентов, подгоняют изделия по фигуре.

Закройщики разрабатывают выкройки, создают лекала и шаблоны для пошива одежды, обеспечивают точность и качество кроя.

Администраторы занимаются приемом и обслуживанием клиентов, ведут записи о заказах, контролируют сроки выполнения работ, организуют работу ателье.

Ателье «Грация» предоставляет широкий спектр услуг, включая пошив одежды на заказ, ремонт и изменение готовых изделий. Каждый заказ имеет ряд характеристик, таких как тип изделия (платье, костюм, пальто и т.д.), используемые материалы, сроки выполнения, стоимость и работы. Каждый новый клиент пополняет перечень клиентов ателье, ведется учет выполненных заказов.

С крупными организациями заключаются договора, позволяющие организациям поставлять материалы для ателье. Каждая организация имеет свой уникальный номер в базе данных.

В заказе указывается ФИО клиента, тип изделия, используемые материалы, сроки выполнения. Заказ может быть отменен не позднее чем за две недели до срока выполнения.

Для упрощения работы администратора будет разработана система, которая позволит автоматизировать работу по ведению учета о заказах, о клиентах, об особенностях каждого изделия, что позволит уменьшить трудоемкость работы.

Виды запросов в информационной системе:

1. Получить перечень поставщиков.

2. Получить данные о поставщиках данного материала.

3. Получить данные о поставках данного поставщика.

4. Получить сведения об организациях, с которыми заключены договора о поставках.

5. Реализовать добавление, редактирование, фильтрацию, удаление информации по поставщикам.

6. Получить сведения о клиентах.

7. Получить сведения о клиенте из заданного заказа.

8. Реализовать добавление, редактирование, фильтрацию, удаление информации по клиентам.

9. Получить количество выполненных заказов.

10. Получить сведения о количестве выполненных заказов с указанными характеристиками.

11. Получить сведения о конкретном выполненном заказе и о его характеристиках.

12. Получить список текущих заказов.

13. Реализовать добавление, редактирование, фильтрацию, удаление информации по заказам.

14. Реализовать возможность клиенту просматривать список своих заказов.

15. Получить список заказов у сотрудника.

16. Реализовать возможность завершить заказ сотруднику.

17. Реализовать возможность браться за заказ сотруднику.

18. Получить сведения о сотрудниках.

19. Реализовать добавление, редактирование, фильтрацию, удаление информации о сотрудниках.

20. Реализовать добавление, редактирование, фильтрацию, удаление информации по материалам.

* 1. Техническое задание

1. Общие сведения

1.1 Полное наименование системы и ее условное обозначение

Полное наименование системы: Автоматизированная информационная система ателье «Грация».

Краткое наименование системы: АИС «Грация».

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: Частное учреждение профессиональная образовательная организация ТЕХНИКУМ «БИЗНЕС И ПРАВО».

Сокращённое название: ЧУПОО ТЕХНИКУМ «БИЗНЕС И ПРАВО».

Юридический и почтовый адрес: 352632, Краснодарский Край, г Белореченск, ул Чапаева, д. 48.

Контактный телефон: +7 (861) 553 39 12.

ИНН: 2310029652

КПП: 236801001.

БИК: 040349722

ОГРН: 1022301620791

Расчетный счет: 407038106.

Полное наименование разработчика: Григорьев Денис Владиславович.

Юридический и почтовый адрес: Краснодарский Край, г Белореченск, пер. Зеленый, д. 27

Контактный телефон +7 (902) 758 00 03.

2. Назначение и цели создания системы

Разрабатываемая система предназначена для управления ателье «Грация». Она позволит автоматизировать процессы работы с заказами, клиентами и поставщиками. Система будет предоставлять информацию о клиентах, сотрудничающих поставщиках и о заказах клиентов.

Основными целями создания информационной системы для ателье «Грация» являются:

- Упрощение работы администрации и сотрудников ателье по учету номеров заказов и клиентов

- Повышение эффективности управления ателье

- Сохранение истории заказов для формирования отчета о проделанной работе

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 ателье:

В ходе разработки информационной системы для ателье «Грация» 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 и учета их номеров

- Повышение эффективности работы сотрудников

- Удобное ведение истории заказов

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 заказами (ведение базы данных заказов)

- Модуль управления клиентами (регистрация клиентов, ведение базы данных клиентов)

- Модуль управления поставщиками

- Модуль оформления заказов (оформление заказа с описанием его особенностей)

- Модуль управления отчетами (ведение отчетов, отправка отчетов в государственные органы)

- Модуль управления сотрудниками (ведение базы данных сотрудников)

Клиентская часть:

- Интерфейс администратора (доступ ко всем модулям системы)

- Интерфейс оформления заказа (указание всех подробностей, пожеланий и комментариев)

На данном этапе необходимо описать примерную архитектуру разрабатываемой системы. Для того, чтобы сделать информационную систему ателье, 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 ателье «Грация», 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 организаций, финансовый учет.

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 ателье. 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:

- Клиент

- Заказ

- Сотрудник

- Поставщик

- Материал

База данных информационной системы должна состоять из следующих связанных таблиц:

- Clients – Таблица, содержащая информацию о клиенте

- Orders – Таблица, содержащая информацию о заказе

- Employees – Таблица, содержащая информацию о сотруднике

- Organizations – Таблица, содержащая информацию о поставщике

- Materials – Таблица, содержащая информацию о материалах

Данные о клиентах размещаются в таблице Clients:

Таблица 1 – Таблица Clients в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| clientId | integer | 11 | Идентификатор клиента | Первичный ключ |
| name | varchar | 50 | ФИО клиента | NOT NULL |
| phone | varchar | 10 | Телефонный номер клиента | NOT NULL |

Данные о заказах размещаются в таблице Order:

Таблица 2 – Таблица Orders в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| orderId | integer | 11 | Идентификатор заказа | Первичный ключ |
| clientId | integer | 11 | Идентификатор клиента | Внешний ключ к таблице clients |
| order\_name | varchar | 50 | Название заказа | NOT NULL |
| type | varchar | 50 | Тип заказа | NOT NULL |
| order\_date | date |  | Дата создания заказа | NOT NULL |
| cost | float |  | Стоимость заказа | NOT NULL |
| status | varchar | 20 | Статус заказа (выполнен/выполняется) | NOT NULL |

Данные о сотрудниках размещаются в таблице Employee:

Таблица 3 – Таблица Employees в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| emplId | integer | 11 | Идентификатор сотрудника | Первичный ключ |
| post | varchar | 50 | Должность | NOT NULL |
| name | varchar | 50 | ФИО сотрудника | NOT NULL |

Данные о поставщиках размещаются в таблице Organization:

Таблица 4 – Таблица Organizations в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| organizationId | integer | 11 | Идентификатор поставщика | Первичный ключ |
| name | varchar | 50 | Название поставщика | NOT NULL |
| address | varchar | 40 | Адрес поставщика | NOT NULL |

Данные о материалах размещаются в таблице Material:

Таблица 5 – Таблица Materials в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| materialId | integer | 11 | Идентификатор материала | Первичный ключ |
| name | varchar | 50 | Название материала | NOT NULL |
| value | integer | 11 | Количество на складе | NOT NULL |

Таблица 6 – orders\_employees

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| emplId | integer | 11 | Идентификатор сотрудника | Внешний ключ к таблице employee |
| orderId | integer | 11 | Идентификатор заказа | Внешний ключ к таблице order |
| period | integer | 11 | Срок до сдачи заказа | NOT NULL |

Таблица 7 – orders\_materials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| materialId | integer | 11 | Идентификатор материала | Внешний ключ к таблице material |
| orderId | integer | 11 | Идентификатор заказа | Внешний ключ к таблице order |

Таблица 8 – organizations\_materials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| organizationId | integer | 11 | Идентификатор поставщика | Внешний ключ к таблице organization |
| materialId | integer | 11 | Идентификатор материала | Внешний ключ к таблице material |
| cost | float |  | Стоимость поставки | NOT NULL |
| value | integer | 11 | Количество товара | 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» – Синий (#005FDB) 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 логин 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»);

- Отображение ошибки при добавлении, редактировании, удалении данных;

- Сообщение об ошибке при проблемах с сервером (например, «Произошла ошибка при запуске сервера», «Пожалуйста, проверьте соединение к сети и попробуйте снова»).

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 4-12 представлены эскизы основных окон системы.



Рисунок 4 – Форма регистрации



Рисунок 5 – Форма входа

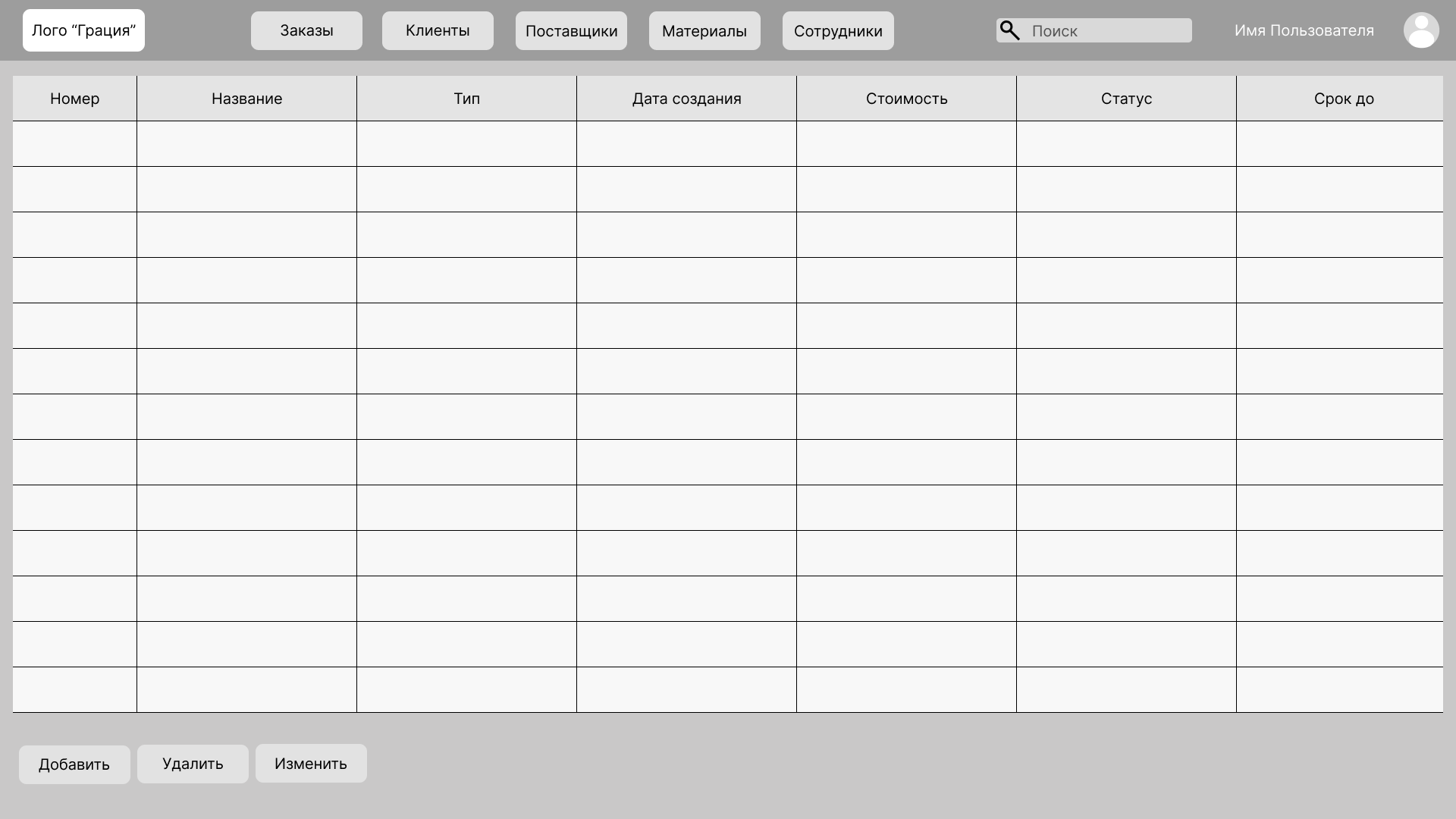


Рисунок 6 – Экран таблицы записей

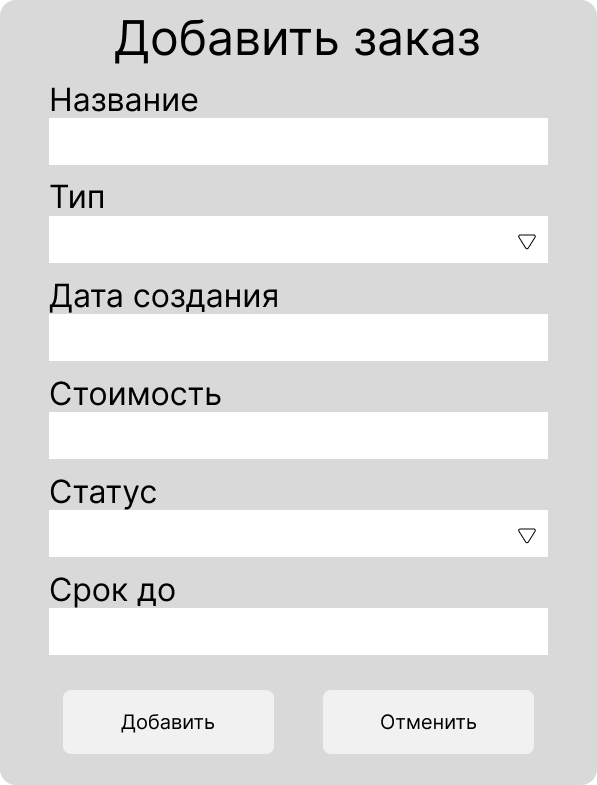


Рисунок 7 – Форма добавления заказа

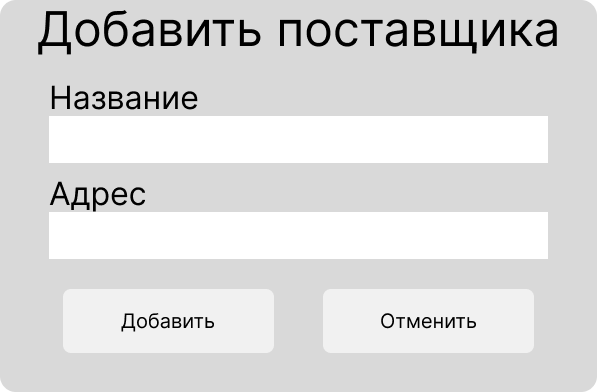


Рисунок 8 – Форма добавления поставщика

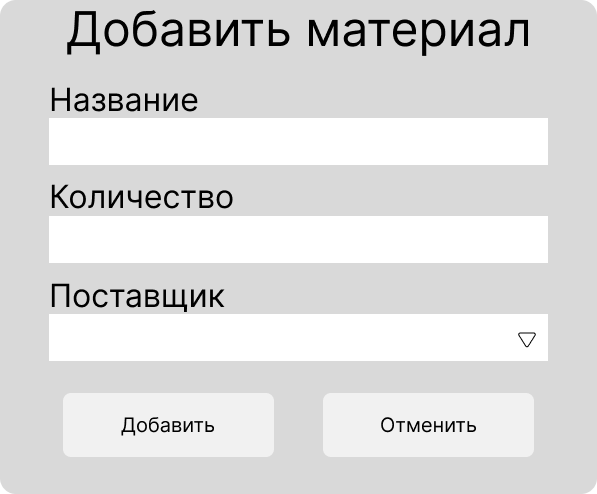


Рисунок 9 – Добавление материала

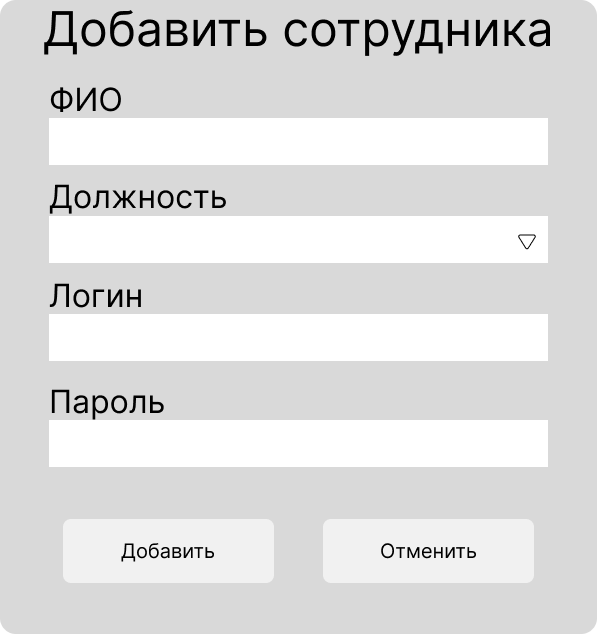


Рисунок 10 – Добавление сотрудника



Рисунок 11 – Поиск записей для удаления

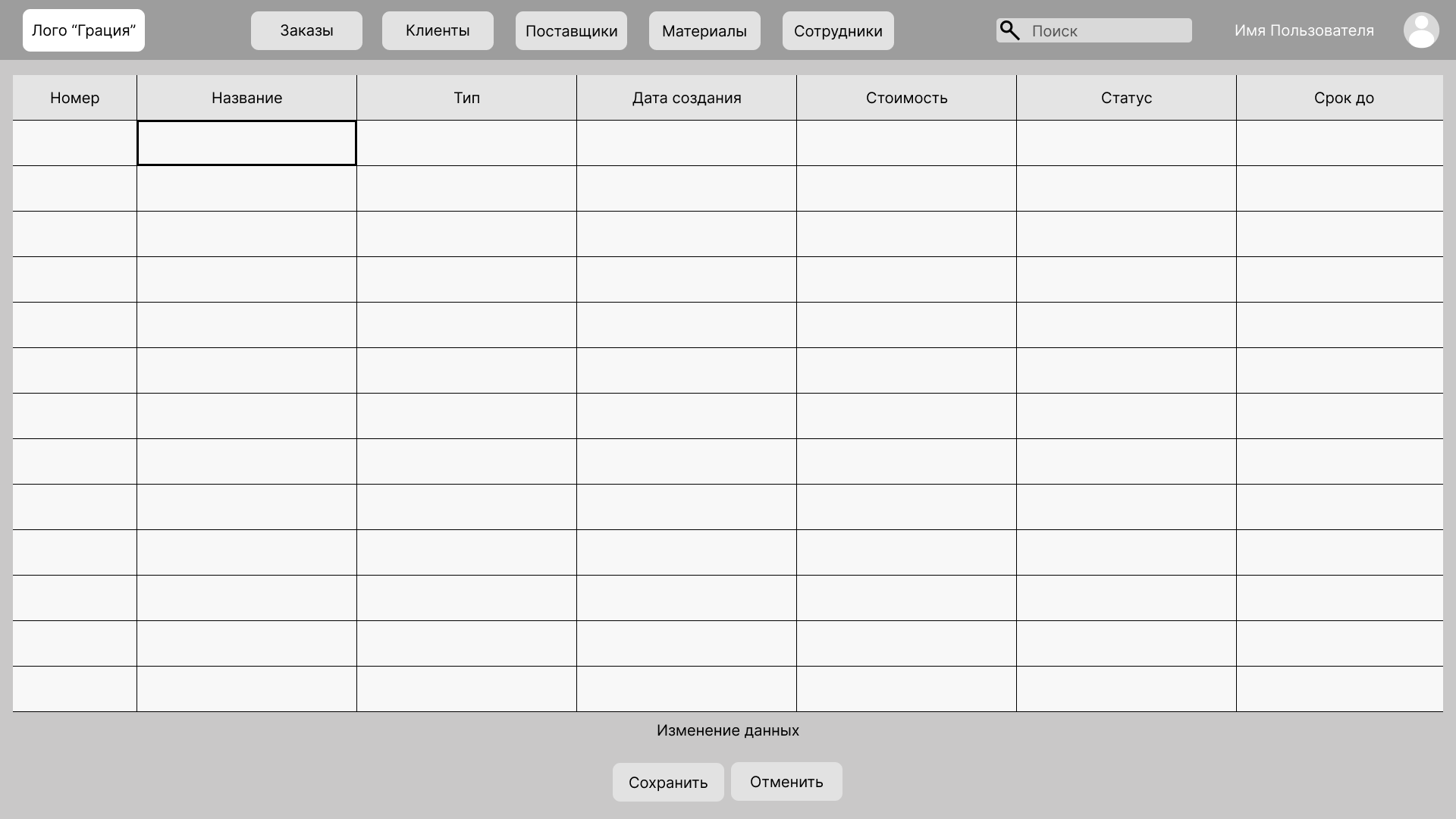


Рисунок 12 – Изменение записей

5. Состав и содержание работ по созданию системы

Комплекс работ по созданию АИС Ателье «Грация» должен осуществляться в несколько этапов.

1. Анализ требований:

- Сбор и анализ требований к системе со стороны заказчика и пользователей.

- Определение функциональных и нефункциональных требований к системе.

- Изучение бизнес-процессов гостиничного комплекса.

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 «Грация» предназначена для комплексного информационно управленческого обеспечения процессов в части исполнения процессов, описанных ниже:

- Управление заказами;

- Оформление заказа;

- Регистрация пользователей;

- Регистрация сотрудников:

- Регистрация поставщиков.

Информационная система управления «Грация» разрабатывается для повышения эффективности управления ателье и качества обслуживания клиентов.

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

Основными целями создания АИС «Грация» 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 ателье за счет оптимизации бизнес-процессов и улучшения качества обслуживания клиентов.

Для достижения поставленных целей информационная система управления ателье «Грация» должна решать следующие основные задачи:

- Ведение базы данных заказов, содержащей информацию о заказах, категориях заказах, их характеристиках и т.д.

- Учет заказов с возможностью поиска по заданным критериям (даты, категория, статус и т.д.).

- Регистрация данных о клиентах, включая личную информацию.

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

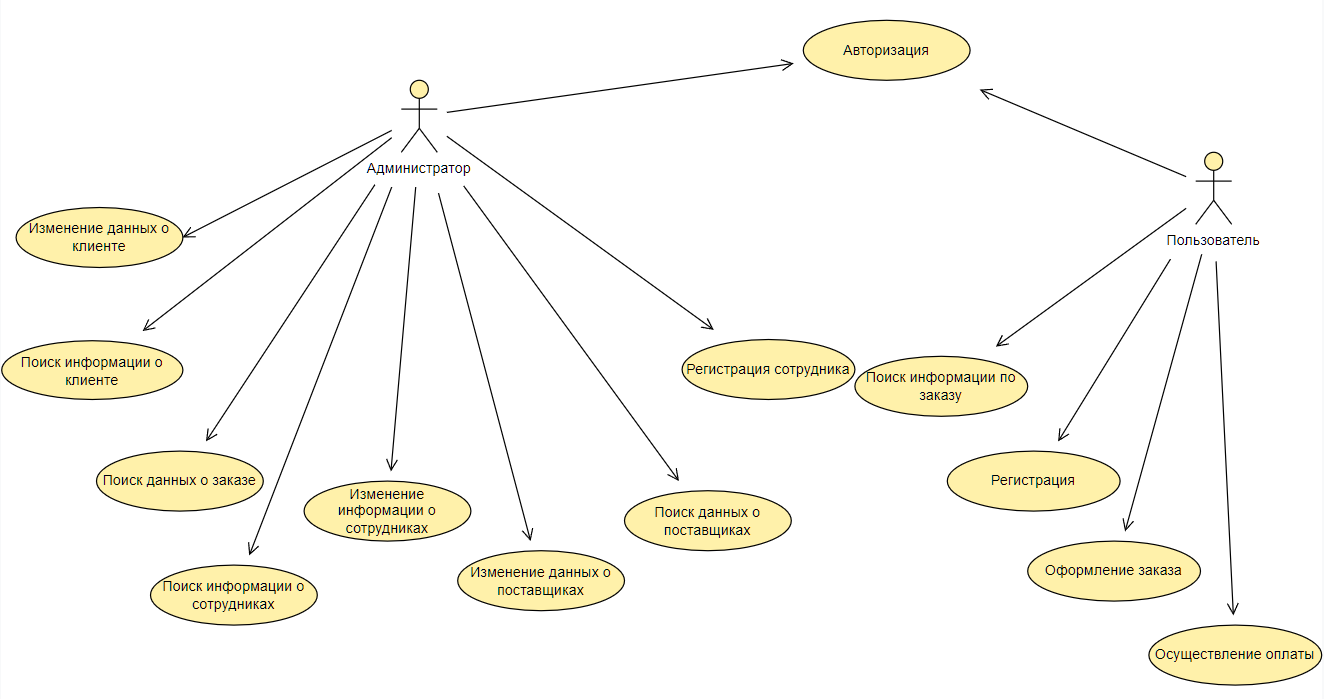


Рисунок 13 – Диаграмма вариантов использования

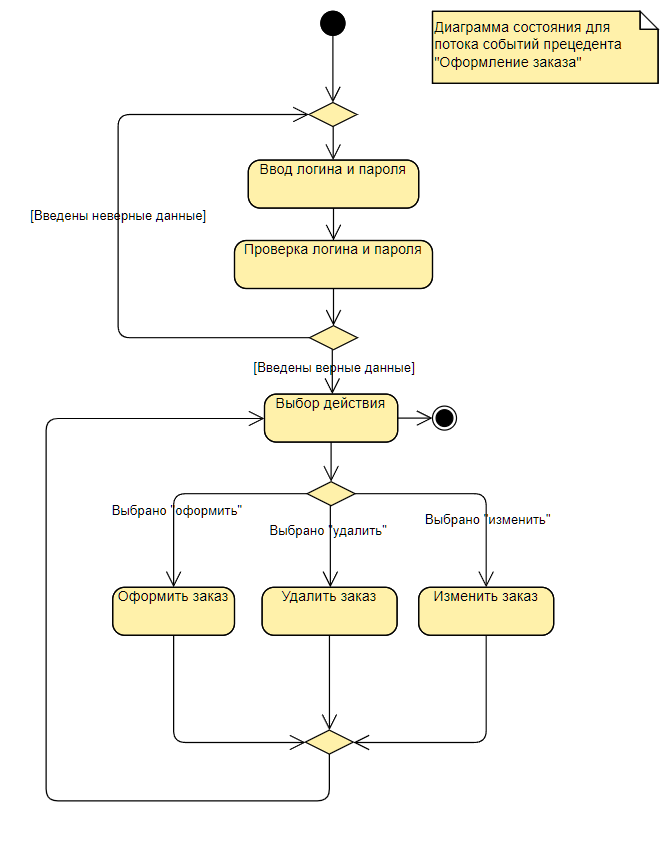


Рисунок 14 – Диаграмма состояния

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.

На рисунке 15 представление MVC архитектуры.



Рисунок 15 – Представление MVC архитектуры

Нашу систему можно разделить на отдельные модули, представленные на рисунке 16:



Рисунок 16 – Модули разрабатываемой системы

Серверная часть:

- Модуль авторизации

- Модуль управления заказами (ведение базы данных заказов)

- Модуль управления клиентами (регистрация клиентов, ведение базы данных клиентов)

- Модуль управления поставщиками

- Модуль оформления заказов (оформление заказа с описанием его особенностей)

- Модуль управления отчетами (ведение отчетов, отправка отчетов в государственные органы)

- Модуль управления сотрудниками (ведение базы данных сотрудников)

В состав персонала, необходимого для обеспечения эксплуатации АИС «Грация» в рамках соответствующих подразделений, необходимо выделение следующих ответственных лиц, представленных в таблице 9:

Таблица 9 – Состав персонала, необходимого для обеспечения эксплуатации АИС «Грация»

|  |  |  |
| --- | --- | --- |
| Должность | Количество | Подсистема |
| Администратор | 1 | Администрирование БД |
| Сотрудники | 1-5 | Управление статусами заказов |

Данные лица должны выполнять следующие функциональные обязанности:

- Сотрудник: управление статусами заказов

- Администратор: внесение правок в базу данных.

База данных информационной системы должна состоять из следующих связанных таблиц:

- Client – Таблица, содержащая информацию о клиенте

- Order – Таблица, содержащая информацию о заказе

- Employee – Таблица, содержащая информацию о сотруднике

- Organization – Таблица, содержащая информацию о поставщике

- Material – Таблица, содержащая информацию о материалах

Данные о клиентах размещаются в таблице Clients:

Таблица 10 – Таблица Clients в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| clientId | integer | 11 | Идентификатор клиента | Первичный ключ |
| name | varchar | 50 | ФИО клиента | NOT NULL |
| phone | varchar | 10 | Телефонный номер клиента | NOT NULL |

Данные о заказах размещаются в таблице Orders:

Таблица 11 – Таблица Orders в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| orderId | integer | 11 | Идентификатор заказа | Первичный ключ |
| clientId | integer | 11 | Идентификатор клиента | Внешний ключ к таблице clients |
| order\_name | varchar | 50 | Название заказа | NOT NULL |
| type | varchar | 50 | Тип заказа | NOT NULL |
| order\_date | date |  | Дата создания заказа | NOT NULL |
| cost | float |  | Стоимость заказа | NOT NULL |
| status | Status | 20 | Статус заказа (выполнен/выполняется) | NOT NULL |

Данные о сотрудниках размещаются в таблице Employees:

Таблица 12 – Таблица Employees в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| emplId | integer | 11 | Идентификатор сотрудника | Первичный ключ |
| post | varchar | 50 | Должность | NOT NULL |
| name | varchar | 50 | ФИО сотрудника | NOT NULL |

Данные о поставщиках размещаются в таблице Organization:

Таблица 13 – Таблица Organizations в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| organizationId | integer | 11 | Идентификатор поставщика | Первичный ключ |
| name | varchar | 50 | Название поставщика | NOT NULL |
| address | varchar | 40 | Адрес поставщика | NOT NULL |

Данные о материалах размещаются в таблице Material:

Таблица 14 – Таблица Materials в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| materialId | integer | 11 | Идентификатор материала | Первичный ключ |
| name | varchar | 50 | Название материала | NOT NULL |
| value | integer | 11 | Количество на складе | NOT NULL |

Таблица 15 – orders\_employees

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| emplId | integer | 11 | Идентификатор сотрудника | Внешний ключ к таблице employee |
| orderId | integer | 11 | Идентификатор заказа | Внешний ключ к таблице order |
| dataOfReady | date | 11 | Дата завершения заказа | NOT NULL |

Таблица 16 – orders\_materials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| materialId | integer | 11 | Идентификатор материала | Внешний ключ к таблице material |
| orderId | integer | 11 | Идентификатор заказа | Внешний ключ к таблице orders |

Таблица 17 – organizations\_materials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничения |
| organizationId | integer | 11 | Идентификатор поставщика | Внешний ключ к таблице organization |
| materialId | integer | 11 | Идентификатор материала | Внешний ключ к таблице material |
| cost | float |  | Стоимость поставки | NOT NULL |
| value | integer | 11 | Количество товара | 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: Spring Framework для серверной части, MySQL для базы данных, Vaadin для отображения.

- Подход к разработке: Agile с итеративным развитием.

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 AddClientView

1.1 Проверка добавления нового клиента:

- Вводим в поля логин, пароль, имя, фамилию, отчество и номер телефона клиента.

- Нажимает кнопку «Добавить».

- Проверяем, что клиент был успешно зарегистрирован и отображается в списке.

1.2 Проверка валидации:

- Оставляем одно или несколько полей пустыми.

- Нажимаем «Добавить».

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

2. Тестирование интерфейса ClientsView

2.1 Проверка отображения списка клиентов:

- Открываем страницу «Клиенты».

- Убеждаемся, что таблица корректно отображает данные.

2.2 Проверка редактирования клиента:

- Выбираем клиента в таблице.

- Нажимаем «Изменить», вносим изменения в ФИО.

- Нажимаем «Сохранить».

- Проверяем, что изменения отображаются в списке.

2.3 Проверка удаления клиента (роль ADMIN):

- Выбираем клиента из списка.

- Нажимаем «Удалить».

- Убеждаемся, что клиент удалён из таблицы.

3. Тестирование интерфейса AddOrderView

3.1 Проверка создания заказа:

- Выводим название, описание, цену.

- Выбираем клиента, материалы, сотрудников.

- Нажимаем «Создать заказ».

- Убеждаемся, что заказ успешно создан и появляется уведомление.

3.2 Проверка валидации формы:

- Не заполняем одно из обязательных полей.

- Нажимаем «Создать заказ».

- Проверяем, что появляется сообщение об ошибке.

4. Тестирование интерфейса ClientProfileView

4.1 Просмотр данных клиента:

- Авторизуемся под пользователем с ролью CLIENT.

- Открываем страницу «Мой профиль».

- Проверяем, что поля ФИО и телефон заполнены и недоступны для редактирования.

4.2 Изменение данных клиента:

- Нажимаем «Редактировать данные», меняем информацию.

- Нажимаем «Применять изменения».

- Убеждаемся, что данные успешные обновлены и отображается уведомление.

4.3 Отмена редактирования:

- Вносим изменения, нажимаем «Отменить изменения».

- Убеждаемся, что изменения не применились.

2.2 Руководство пользователя

1. Авторизация

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

Если вы являетесь администратором, сотрудником или клиентом, доступ к функциям будет ограничен в зависимости от вашей роли.

После успешной авторизации вы попадёте на главную страницу приложения.

2. Главное окно

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

Для администратора:

- Управление клиентами

- Управление сотрудниками

- Управление заказами

- Управление материалами

- Управление поставщиками

Для сотрудника:

- Создание и редактирование заказов

- Просмотр и редактирование клиентов

- Управление материалами

- Завершение заказов

Для клиента:

- Просмотр своих заказов

- Редактирование профиля

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

3. Добавление клиента

Для добавления клиента нажмите кнопку «Добавить клиента».  
Заполните форму (логин, пароль, имя, фамилию, отчество, номер телефона) и нажмите «Добавить».

Клиент будет сохранён в базе, а также будет создана учётная запись с ролью CLIENT.

4. Создание заказа

Чтобы создать заказ, перейдите в раздел «Создать заказ».

Заполните форму (название заказа, описание, материалы (выбираются из списка), клиент (выбирается из списка), ответственные сотрудники (выбираются из списка), стоимость).

Нажмите кнопку «Создать заказ», и он будет сохранён. Назначенные сотрудники будут автоматически привязаны к заказу.

5. Редактирование клиента

Перейдите в раздел «Клиенты», выберите нужного клиента в таблице и нажмите «Изменить».

В появившейся форме отредактируйте ФИО и нажмите «Сохранить».

6. Профиль клиента

Клиенты могут редактировать свой профиль, нажав на пункт «Мой профиль».

После редактирования необходимо нажать «Применить изменения» или «Отменить», чтобы сбросить изменения.

7. Управление материалами и поставщиками

Доступно сотрудникам и администраторам.

Позволяет добавлять, редактировать и удалять материалы, а также просматривать поставки и поставщиков.

8. Завершение заказа

Для завершения заказа сотрудник выбирает нужный заказ и нажимает «Завершить». Статус изменится на «Выполнен».

9. О программе

Раздел «О программе» содержит информацию об авторе и описании системы.

10. Выход

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

1.3 Описание страниц

1. Главная страница

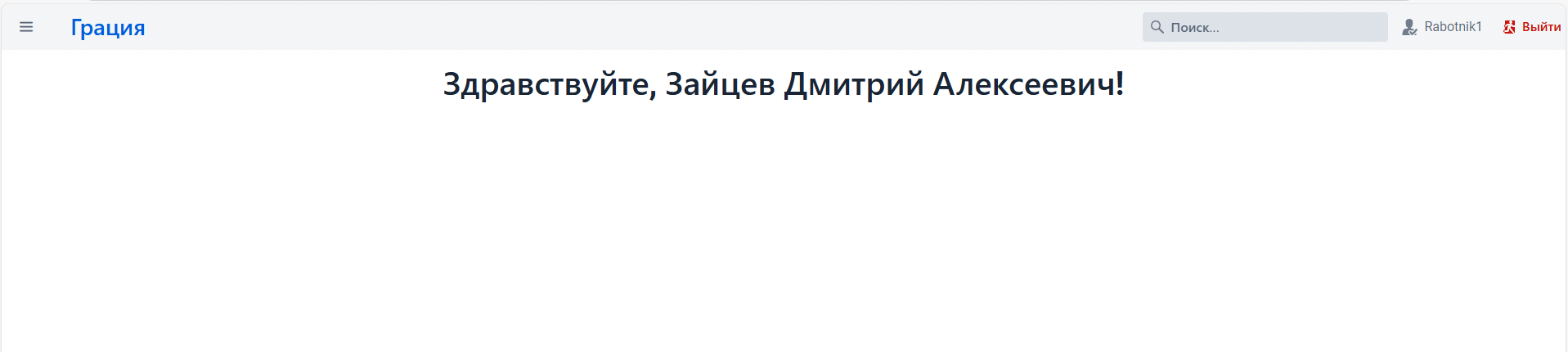


Рисунок 17 – Главная страница

На главной странице отображается приветственное сообщение для клиентов и сотрудников. Чтобы попасть на эту страницу, перейдите по любой из следующих ссылок /, /index, /main.

2. Заказы

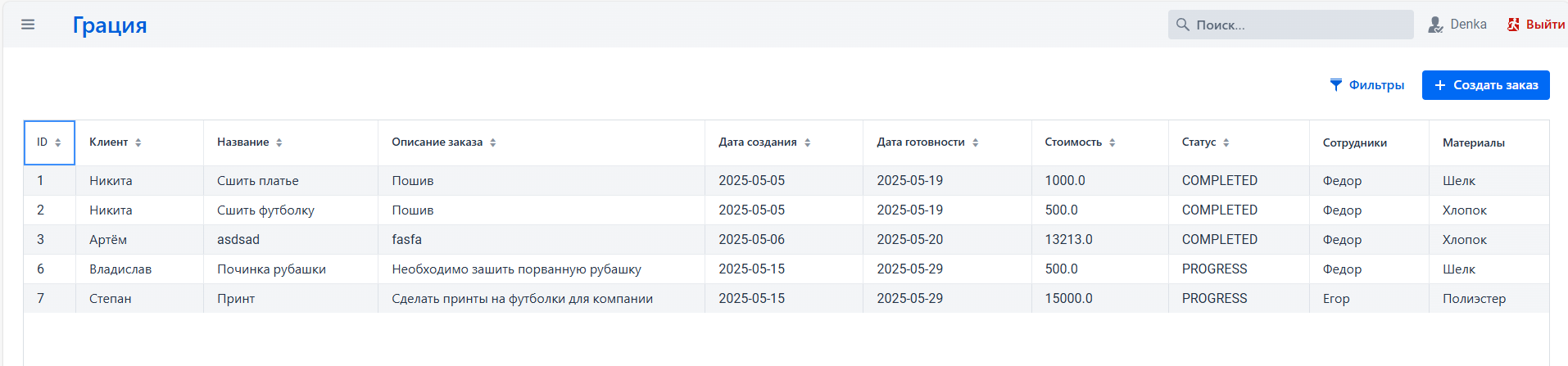


Рисунок 18 – Заказы

На странице отображается список заказов, фильтр и кнопка добавления заказа. При нажатии на кнопку «Фильтры» открывается окно с параметрами для фильтрации заказов. При нажатии на кнопку «Создать заказ» происходит переход на страницу «Создать заказ». При нажатии правой кнопкой мыши по любому объекту в таблице открывается контекстное меню. В контекстном меню присутствуют кнопки «Сведения о заказе», «Сведения о заказчике», «Изменить» и «Удалить». Кнопка «Удалить» видна только пользователям с ролью «Администратор». Чтобы попасть на эту страницу, перейдите по ссылке /orders-list.

3. Создать заказ

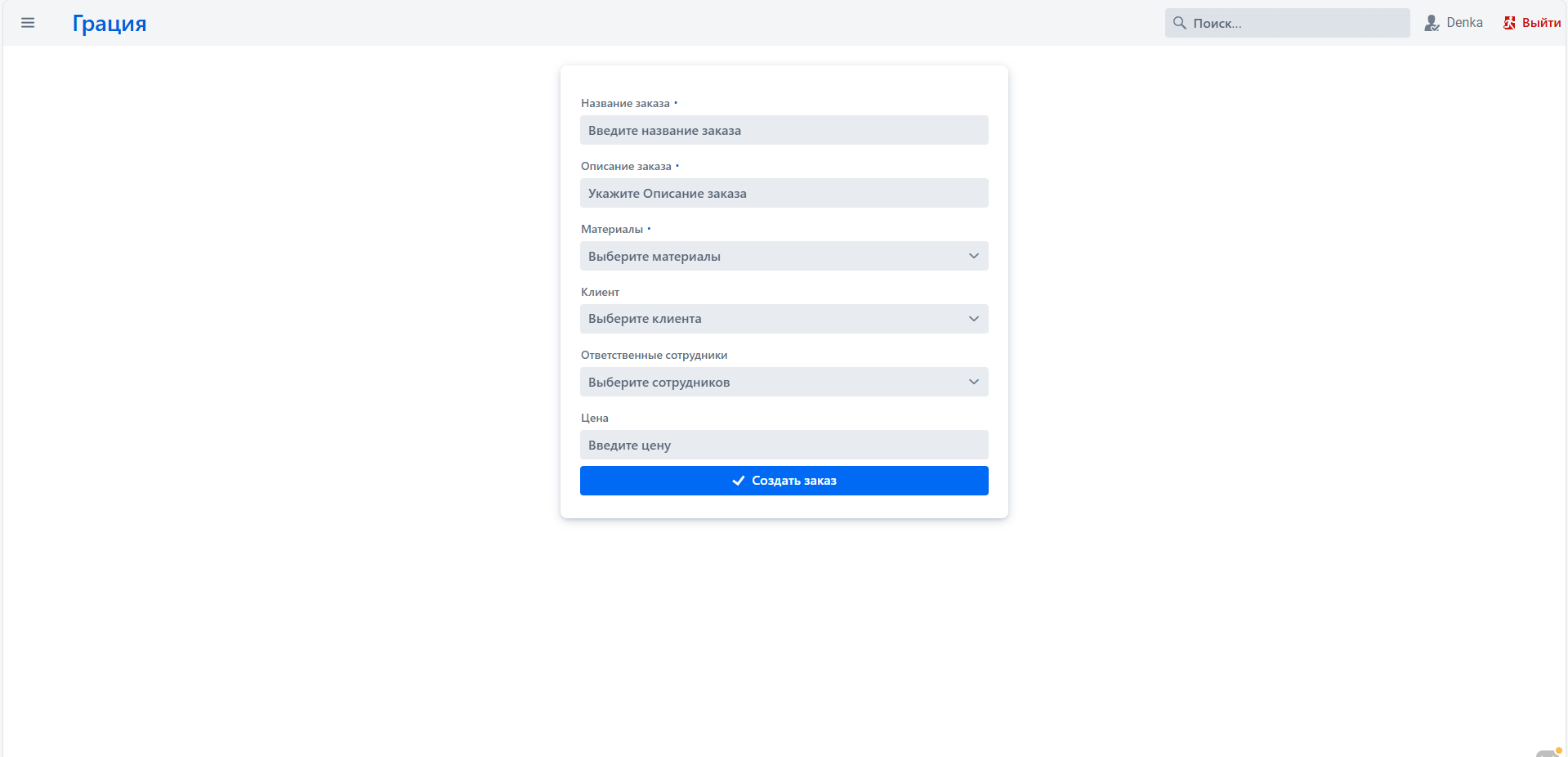


Рисунок 19 – Создать заказ

На странице отображается форма для создания нового заказа. После заполнения формы необходимо нажать на кнопку «Создать заказ». После нажатия на кнопку, создается новый заказ с указанными в форме характеристиками и происходит переадресация на страницу «Заказы».

4. Клиенты

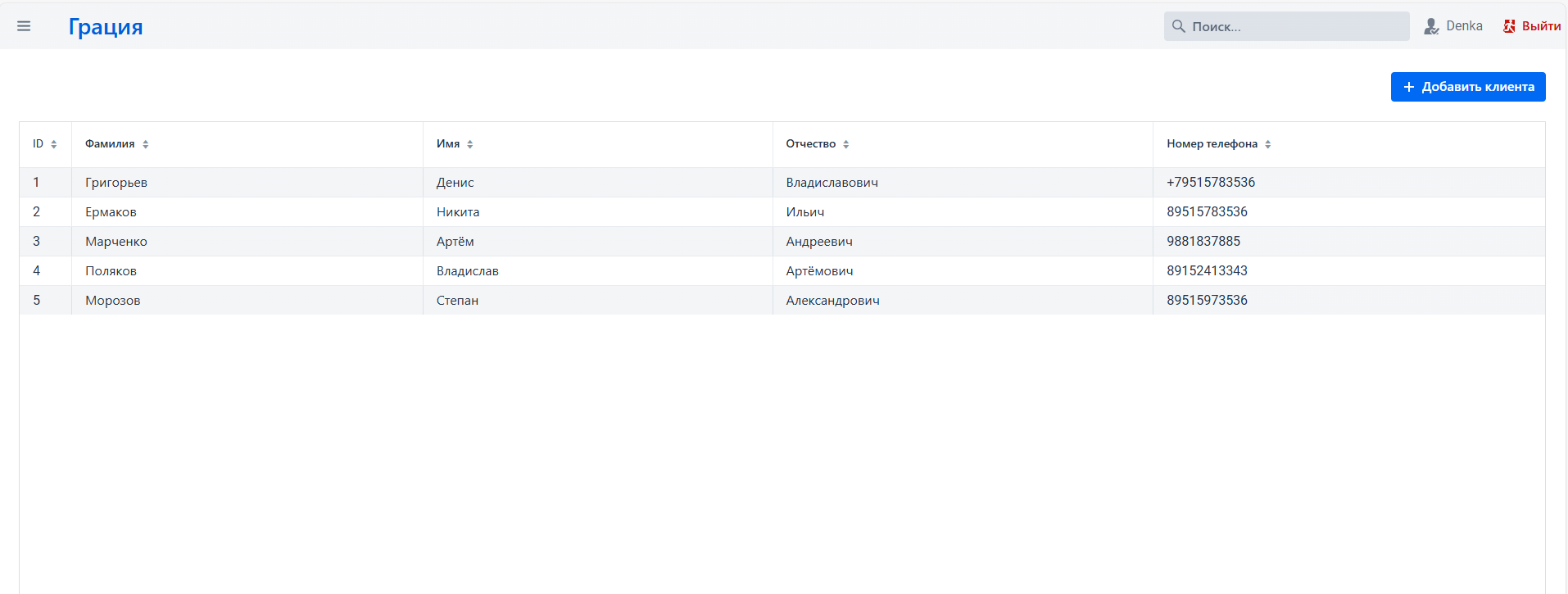


Рисунок 20 – Клиенты

На странице отображается список клиентов и кнопка добавления клиента. При нажатии на кнопку «Добавить клиента» происходит переход на страницу «Добавить клиента». При нажатии правой кнопкой мыши по любому объекту в таблице открывается контекстное меню. В контекстном меню присутствуют кнопки «Изменить» и «Удалить». Кнопка «Удалить» видна только пользователям с ролью «Администратор». Чтобы попасть на эту страницу перейдите по ссылке /clients-list.

5. Добавить клиента

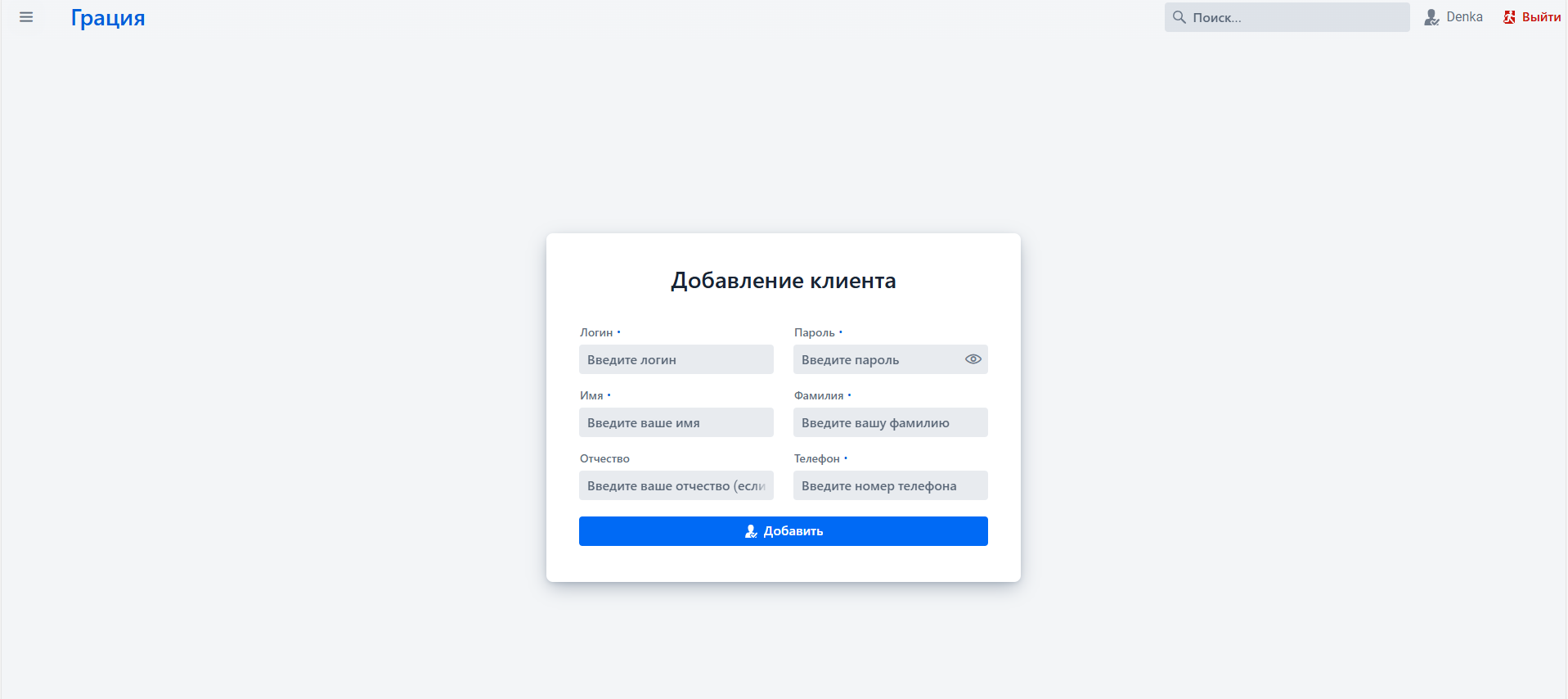


Рисунок 21 – Добавить клиента

На странице отображается форма для добавления клиента. После заполнения формы необходимо нажать на кнопку «Добавить». После нажатия на кнопку, создается аккаунт для клиента и происходит переадресация на страницу «Клиенты». Чтобы попасть на эту страницу перейдите по ссылке /clients-list.

6. Поставщики

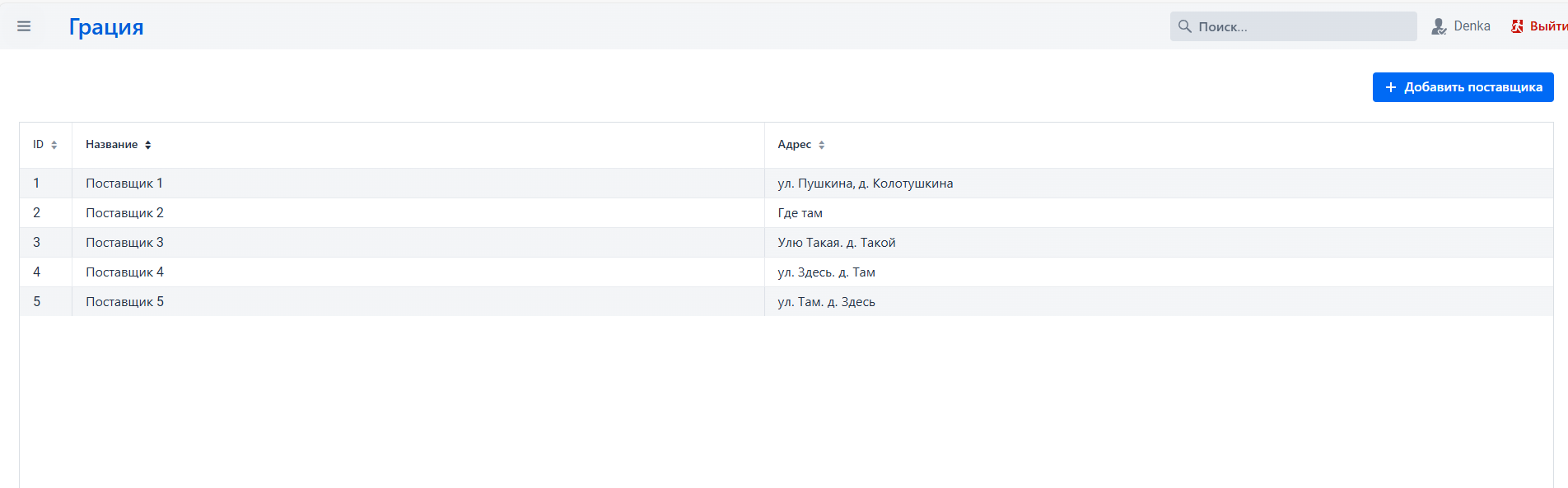


Рисунок 22 – Поставщики

На странице отображается список поставщиков и кнопка добавления поставщика. При нажатии на кнопку «Добавить поставщика» открывается окно с формой, в которою необходимо ввести данные о новом поставщике и нажать на кнопку «Добавить», для добавления нового поставщика. При нажатии правой кнопкой мыши по любому объекту в таблице открывается контекстное меню. В контекстном меню присутствуют кнопки «Изменить» и «Удалить». Кнопка «Удалить» видна только пользователям с ролью «Администратор». Чтобы попасть на эту страницу перейдите по ссылке /organizations-list.

7. Материалы



Рисунок 23 – Материалы

На странице отображается список материалов и кнопка «Добавить материал». При нажатии на кнопку «Добавить материал» открывается окно с формой, в которою необходимо ввести данные о новом материале и нажать на кнопку «Добавить». При нажатии правой кнопкой мыши по любому объекту в таблице открывается контекстное меню. В контекстном меню присутствуют кнопки «Изменить» и «Удалить». Кнопка «Удалить» видна только пользователям с ролью «Администратор». Чтобы попасть на эту страницу перейдите по ссылке /materials-list.

8. Сотрудники

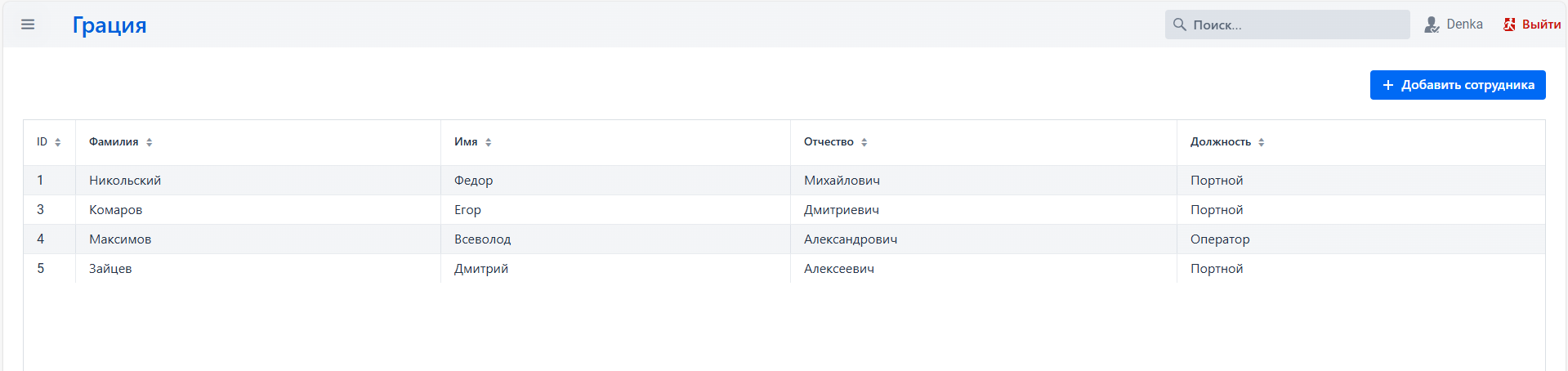


Рисунок 24 – Сотрудники

На странице отображается список сотрудников и кнопка «Добавить сотрудника». При нажатии на кнопку «Добавить сотрудника» происходит переход на страницу «Регистрация сотрудника». При нажатии правой кнопкой мыши по любому объекту в таблице открывается контекстное меню. В контекстном меню присутствуют кнопки «Изменить» и «Удалить». Кнопка «Удалить» видна только пользователям с ролью «Администратор». Чтобы попасть на эту страницу перейдите по ссылке /employees-list.

9. Регистрация сотрудника

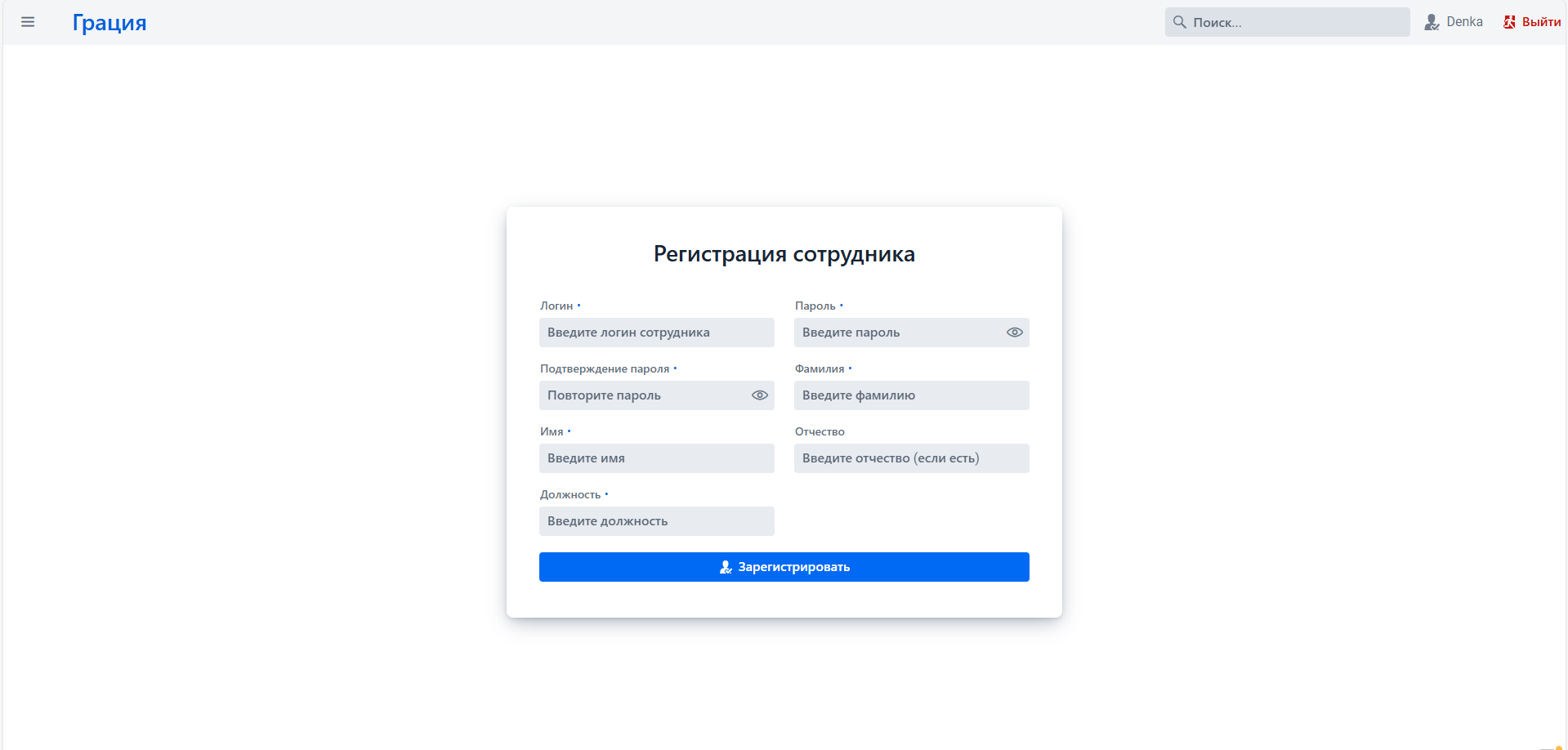


Рисунок 25 – Регистрация сотрудника

На странице отображается форма для регистрации нового сотрудника. После заполнения формы необходимо нажать на кнопку «Зарегистрировать». При нажатии на кнопку создается аккаунт для сотрудника и происходит переадресация на страницу «Сотрудники». Чтобы попасть на эту страницу перейдите по ссылке /employee-registration.

10. Поставки



Рисунок 26 – Поставки

На странице отображается список поставок материалов от поставщиков, фильтр и кнопка добавления поставки. При нажатии на кнопку «Фильтры» открывается окно с параметрами для фильтрации поставок. При нажатии на кнопку «Добавить поставку» открывается окно с формой, в которую нужно ввести данные о новой поставке. Чтобы попасть на эту страницу перейдите по ссылке /supplies.

11. О программе

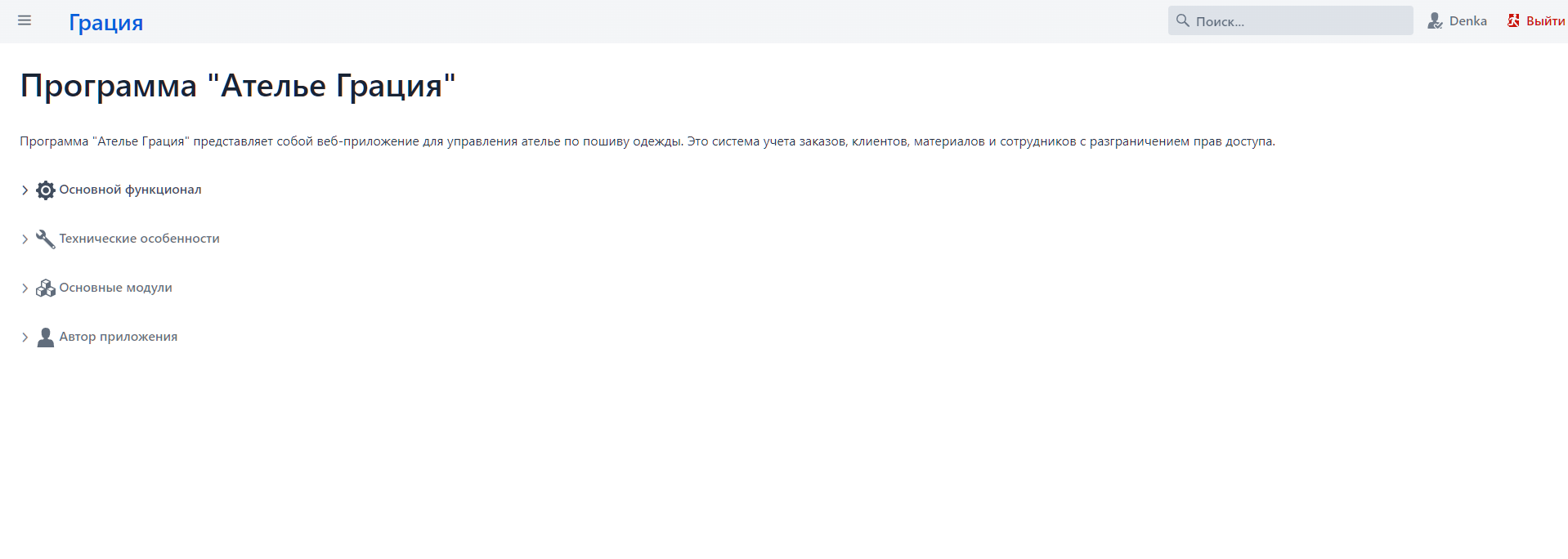


Рисунок 27 – О программе

На странице отображается информация о программе и её разработчике. Чтобы попасть на эту страницу перейдите по ссылке /about.

12. Мой профиль

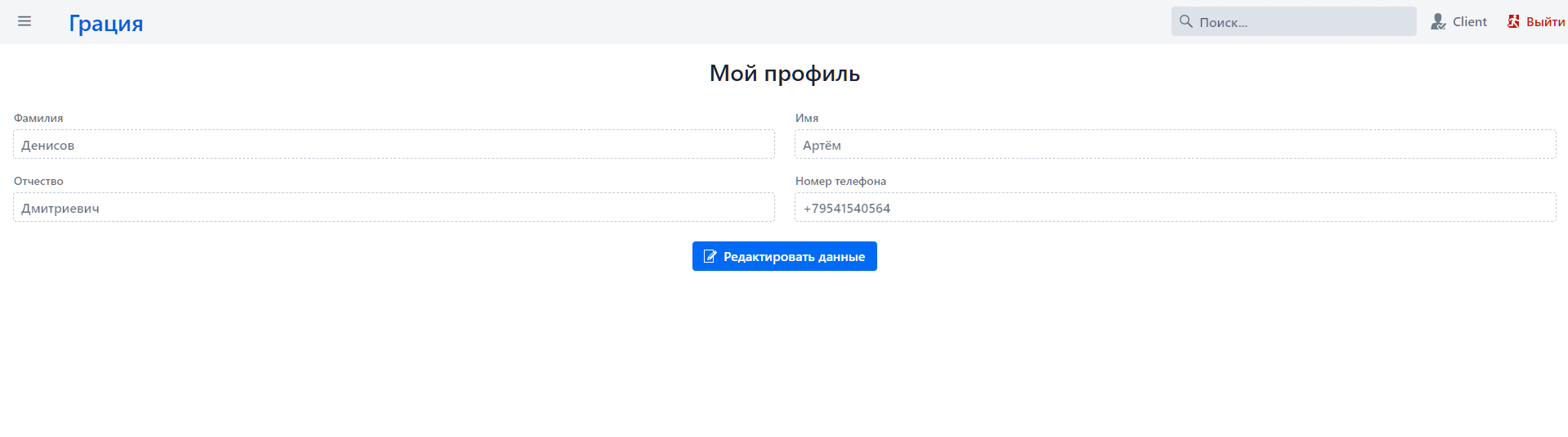


Рисунок 28 – Мой профиль

На странице отображается информация о клиенте и кнопка редактирования данных. При нажатии на кнопку «Редактировать данные» поля с данными о клиенте становятся доступными для редактирования и отображаются кнопки «Применить изменения» и «Отменить изменения». Чтобы попасть на эту страницу перейдите по ссылке /client-profile.

13. Мои заказы

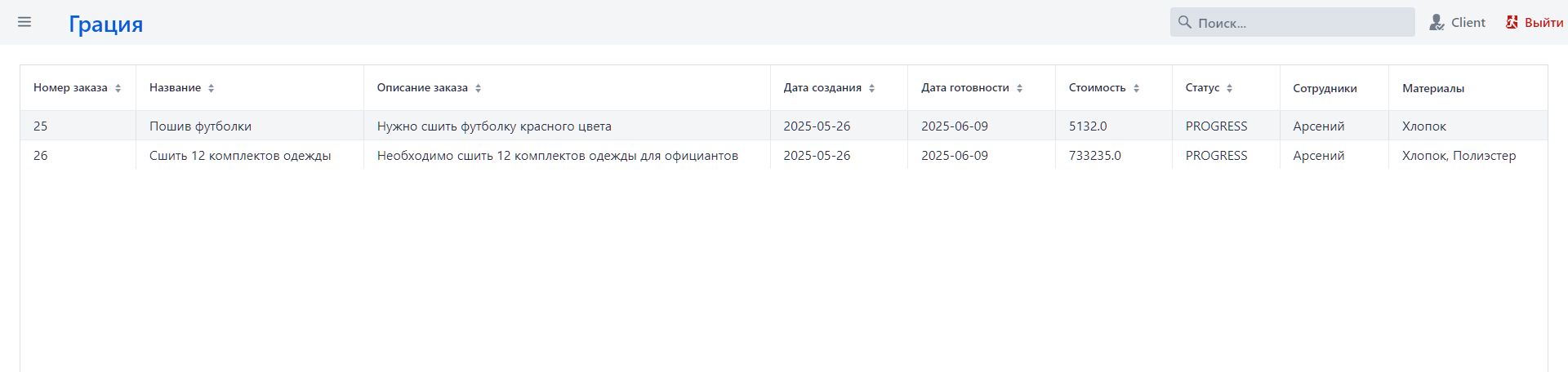


Рисунок 29 – Мои заказы

На странице отображается список заказов клиента. Чтобы попасть на эту страницу перейдите по ссылке /my-orders.

ЗАКЛЮЧЕНИЕ

В ходе прохождения учебной практики была спроектирована и реализована информационная система для ателье «Грация», предназначенная для автоматизации ключевых бизнес-процессов предприятия. В процессе разработки были подробно изучены структура и деятельность ателье, определены требования к функциональности системы, выполнено проектирование архитектуры, разработаны модули учёта заказов, клиентов, сотрудников, материалов и поставщиков.

Система обеспечивает разграничение прав доступа пользователей в зависимости от их ролей (администратор, сотрудник, клиент), что повышает безопасность и удобство эксплуатации. Внедрение авторизации, регистрационного механизма, а также функции по ведению истории заказов 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 Vaadin, а серверная логика реализована на основе Spring Boot.

В результате внедрения информационной системы «Ателье Грация» достигаются следующие преимущества:

- Повышение точности и скорости обработки информации;

- Упрощение учёта заказов, клиентов и сотрудников;

- 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. Vaadin Docs [Электронный ресурс]. Режим доступа: <https://vaadin.com/docs/>

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

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

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

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

ПРИЛОЖЕНИЕ

package com.denka88.ateliergrace.config;  
  
import com.denka88.ateliergrace.view.LoginView;  
import com.vaadin.flow.spring.security.VaadinWebSecurity;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
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;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig extends VaadinWebSecurity {  
   
 @Override  
 protected void configure(HttpSecurity *http*) throws Exception {  
   
 *http*.authorizeHttpRequests(*auth* -> *auth* .requestMatchers(antMatchers("/register")).permitAll()  
 .requestMatchers(antMatchers("/")).permitAll()  
 .requestMatchers(antMatchers("/main")).permitAll()  
 .requestMatchers(antMatchers("/test")).permitAll()  
 .requestMatchers(antMatchers("/login")).permitAll()  
 .requestMatchers(antMatchers("/employee-registration")).hasRole("ADMIN")  
 );  
   
 super.configure(*http*);  
   
 setLoginView(*http*, LoginView.class);  
  
 *http*.formLogin(*form* -> *form* .loginPage("/login")  
 .defaultSuccessUrl("/")  
 .permitAll()  
 );  
 }  
  
 @Bean  
 public BCryptPasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration *config*) throws Exception {  
 return *config*.getAuthenticationManager();  
 }  
   
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Auth;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.repo.AuthRepo;  
import com.denka88.ateliergrace.service.AuthService;  
import jakarta.transaction.Transactional;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
  
import java.util.Collections;  
import java.util.Optional;  
  
@Service  
public class AuthServiceImpl implements AuthService {  
  
 private final AuthRepo authRepo;  
 private final BCryptPasswordEncoder passwordEncoder;  
  
 public AuthServiceImpl(AuthRepo *authRepo*, BCryptPasswordEncoder *passwordEncoder*) {  
 this.authRepo = *authRepo*;  
 this.passwordEncoder = *passwordEncoder*;  
 }  
  
 @Override  
 public void register(String *login*, String *password*, UserType *userType*, Long *userId*) {  
 if (authRepo.existsByLogin(*login*)) {  
 throw new IllegalArgumentException("Login already exists");  
 }  
  
 Auth *auth* = new Auth();  
 *auth*.setLogin(*login*);  
 *auth*.setPasswordHash(passwordEncoder.encode(*password*)); // Кодируем пароль  
 *auth*.setUserType(*userType*);  
 *auth*.setUserId(*userId*);  
  
 authRepo.save(*auth*);  
 }  
  
 @Override  
 @Transactional  
 public void delete(Long *userId*) {  
 authRepo.deleteByUserId(*userId*);  
 }  
  
 @Override  
 public Optional<Auth> findByLogin(String *login*) {  
 return authRepo.findByLogin(*login*);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Client;  
import com.denka88.ateliergrace.repo.AuthRepo;  
import com.denka88.ateliergrace.repo.ClientRepo;  
import com.denka88.ateliergrace.repo.OrderRepo;  
import com.denka88.ateliergrace.service.AuthService;  
import com.denka88.ateliergrace.service.ClientService;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class ClientServiceImpl implements ClientService {  
  
 private final ClientRepo clientRepo;  
 private final AuthService authService;  
  
 public ClientServiceImpl(ClientRepo *clientRepo*, AuthService *authService*) {  
 this.clientRepo = *clientRepo*;  
 this.authService = *authService*;  
 }  
  
 @Override  
 public List<Client> findAll() {  
 return clientRepo.findAll();  
 }  
  
 @Override  
 public Optional<Client> findById(Long *id*) {  
 return clientRepo.findById(*id*);  
 }  
  
 @Override  
 public Client save(Client *client*) {  
 return clientRepo.save(*client*);  
 }  
  
 @Override  
 public void delete(Long *id*) {  
 Client *client* = clientRepo.findById(*id*).orElseThrow(() -> new IllegalArgumentException("Клиент не найден"));  
   
 authService.delete(*client*.getId());  
   
 clientRepo.delete(*client*);  
 }  
  
 @Override  
 public void update(Client *client*) {  
 clientRepo.save(*client*);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Auth;  
import com.denka88.ateliergrace.model.Client;  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.repo.AuthRepo;  
import com.denka88.ateliergrace.service.ClientService;  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.denka88.ateliergrace.service.EmployeeService;  
import com.vaadin.flow.spring.security.AuthenticationContext;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.stereotype.Service;  
  
import java.util.Optional;  
  
@Service  
public class CurrentUserServiceImpl implements CurrentUserService {  
  
 private final AuthRepo authRepo;  
 private final ClientService clientService;  
 private final AuthenticationContext authenticationContext;  
 private final EmployeeService employeeService;  
  
 public CurrentUserServiceImpl(AuthRepo *authRepo*, ClientService *clientService*, AuthenticationContext *authenticationContext*, EmployeeService *employeeService*) {  
 this.authRepo = *authRepo*;  
 this.clientService = *clientService*;  
 this.authenticationContext = *authenticationContext*;  
 this.employeeService = *employeeService*;  
 }  
  
 @Override  
 public Optional<Auth> getCurrentAuth() {  
 return authenticationContext.getAuthenticatedUser(UserDetails.class)  
 .flatMap(*user* -> authRepo.findByLogin(*user*.getUsername()));  
 }  
   
 @Override  
 public Optional<Employee> getCurrentEmployee() {  
 return getCurrentAuth()  
 .filter(*auth* -> *auth*.getUserType() == UserType.*EMPLOYEE*)  
 .flatMap(*auth* -> employeeService.findById(*auth*.getUserId()));  
 }  
  
 @Override  
 public Optional<Client> getCurrentClient() {  
 return getCurrentAuth()  
 .filter(*auth* -> *auth*.getUserType() == UserType.*CLIENT*)  
 .flatMap(*auth* -> clientService.findById(*auth*.getUserId()));  
 }  
  
 @Override  
 public Long getCurrentUserId() {  
 return getCurrentAuth()  
 .map(Auth::getUserId)  
 .orElse(null);  
 }  
  
 @Override  
 public UserType getCurrentUserType() {  
 return getCurrentAuth()  
 .map(Auth::getUserType)  
 .orElse(null);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.repo.EmployeeRepo;  
import com.denka88.ateliergrace.repo.OrderEmployeeRepo;  
import com.denka88.ateliergrace.service.AuthService;  
import com.denka88.ateliergrace.service.EmployeeService;  
import jakarta.transaction.Transactional;  
import org.springframework.stereotype.Service;  
  
import java.util.Comparator;  
import java.util.List;  
import java.util.Map;  
import java.util.Optional;  
import java.util.stream.Collectors;  
  
@Service  
public class EmployeeServiceImpl implements EmployeeService {  
   
 private final EmployeeRepo employeeRepo;  
 private final AuthService authService;  
 private final OrderEmployeeRepo orderEmployeeRepo;  
  
 public EmployeeServiceImpl(EmployeeRepo *employeeRepo*, AuthService *authService*, OrderEmployeeRepo *orderEmployeeRepo*) {  
 this.employeeRepo = *employeeRepo*;  
 this.authService = *authService*;  
 this.orderEmployeeRepo = *orderEmployeeRepo*;  
 }  
  
 @Override  
 public List<Employee> findAll() {  
 return employeeRepo.findAll();  
 }  
  
 @Override  
 public Optional<Employee> findById(Long *id*) {  
 return employeeRepo.findById(*id*);  
 }  
  
 @Override  
 public Employee save(Employee *employee*) {  
 return employeeRepo.save(*employee*);  
 }  
  
 @Override  
 @Transactional  
 public void delete(Long *id*) {  
 Employee *employee* = employeeRepo.findById(*id*)  
 .orElseThrow(() -> new IllegalArgumentException("Сотрудник не найден"));  
  
 authService.delete(*employee*.getId());  
   
 *employee*.getOrders().clear();  
 employeeRepo.deleteById(*id*);  
 }  
  
 @Override  
 public void update(Employee *employee*) {  
 employeeRepo.save(*employee*);  
 }  
   
 @Override  
 public List<Employee> findAllOrderByInProgressOrdersCount() {  
 Map<Long, Long> *employeeOrderCounts* = orderEmployeeRepo.countInProgressOrdersPerEmployee().stream().collect(Collectors.*toMap*(*arr*->(Long) *arr*[0], *arr*->(Long) *arr*[1]));  
   
 List<Employee> *allEmployees* = employeeRepo.findAll();  
   
 *allEmployees*.sort(Comparator.*comparingLong*(*employee*->*employeeOrderCounts*.getOrDefault(*employee*.getId(), 0L)));  
   
 return *allEmployees*;  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Material;  
import com.denka88.ateliergrace.model.Order;  
import com.denka88.ateliergrace.repo.MaterialRepo;  
import com.denka88.ateliergrace.repo.OrderRepo;  
import com.denka88.ateliergrace.service.MaterialService;  
import jakarta.transaction.Transactional;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class MaterialServiceImpl implements MaterialService {  
   
 private final MaterialRepo materialRepo;  
 private final OrderRepo orderRepo;  
  
 public MaterialServiceImpl(MaterialRepo *materialRepo*, OrderRepo *orderRepo*) {  
 this.materialRepo = *materialRepo*;  
 this.orderRepo = *orderRepo*;  
 }  
  
 @Override  
 public List<Material> findAll() {  
 return materialRepo.findAll();  
 }  
  
 @Override  
 public Optional<Material> findById(Long *id*) {  
 return materialRepo.findById(*id*);  
 }  
  
 @Override  
 public Material save(Material *material*) {  
 return materialRepo.save(*material*);  
 }  
  
 @Override  
 @Transactional  
 public void delete(Long *id*) {  
 if (*id* == null) {  
 throw new IllegalArgumentException("ID материала не может быть null");  
 }  
   
 Material *material* = materialRepo.findById(*id*)  
 .orElseThrow(() -> new IllegalArgumentException("Материал с ID " + *id* + " не найден"));  
   
 List<Order> *orders* = orderRepo.findByMaterialsId(*id*);  
  
 for (Order *order* : *orders*) {  
 *order*.getMaterials().remove(*material*);  
 orderRepo.save(*order*);  
 }  
  
 materialRepo.deleteById(*id*);  
 }  
  
 @Override  
 @Transactional  
 public void update(Material *material*) {  
 materialRepo.save(*material*);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.model.OrderEmployee;  
import com.denka88.ateliergrace.model.OrderEmployeeKey;  
import com.denka88.ateliergrace.model.Status;  
import com.denka88.ateliergrace.repo.EmployeeRepo;  
import com.denka88.ateliergrace.repo.OrderEmployeeRepo;  
import com.denka88.ateliergrace.repo.OrderRepo;  
import com.denka88.ateliergrace.service.OrderEmployeeService;  
import jakarta.transaction.Transactional;  
import org.springframework.stereotype.Service;  
  
import java.time.LocalDate;  
import java.util.List;  
  
@Service  
public class OrderEmployeeServiceImpl implements OrderEmployeeService {  
   
 private final OrderEmployeeRepo orderEmployeeRepo;  
 private final EmployeeRepo employeeRepo;  
 private final OrderRepo orderRepo;  
  
 public OrderEmployeeServiceImpl(OrderEmployeeRepo *orderEmployeeRepo*, EmployeeRepo *employeeRepo*, OrderRepo *orderRepo*) {  
 this.orderEmployeeRepo = *orderEmployeeRepo*;  
 this.employeeRepo = *employeeRepo*;  
 this.orderRepo = *orderRepo*;  
 }  
  
 @Override  
 public List<OrderEmployee> findAll() {  
 return orderEmployeeRepo.findAll();  
 }  
  
 @Override  
 public OrderEmployee save(OrderEmployee *orderEmployee*) {  
 return orderEmployeeRepo.save(*orderEmployee*);  
 }  
  
 @Override  
 public void update(OrderEmployee *orderEmployee*) {  
 orderEmployeeRepo.save(*orderEmployee*);  
 }  
  
 @Transactional  
 public void setReadyDate(Long *orderId*, Long *employeeId*, LocalDate *readyDate*) {  
 OrderEmployeeKey *key* = new OrderEmployeeKey(*orderId*, *employeeId*);  
 OrderEmployee *orderEmployee* = orderEmployeeRepo.findById(*key*)  
 .orElseThrow(() -> new IllegalArgumentException("Связь заказа и сотрудника не найдена"));  
 *orderEmployee*.setDateOfReady(*readyDate*);  
 orderEmployeeRepo.save(*orderEmployee*);  
 }  
  
 @Override  
 public void createAssignment(Long *orderId*, Long *employeeId*, LocalDate *readyDate*) {  
 OrderEmployee *assignment* = new OrderEmployee();  
 *assignment*.setId(new OrderEmployeeKey(*orderId*, *employeeId*));  
 *assignment*.setOrder(orderRepo.getReferenceById(*orderId*));  
 *assignment*.setEmployee(employeeRepo.getReferenceById(*employeeId*));  
 *assignment*.setDateOfReady(*readyDate*);  
  
 orderEmployeeRepo.save(*assignment*);  
 }  
  
 @Override  
 public Long getInProgressCount(Long *employeeId*) {  
 return orderEmployeeRepo.countByEmployeeIdAndOrderStatus(  
 *employeeId*,  
 Status.*PROGRESS* );  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.\*;  
import com.denka88.ateliergrace.repo.MaterialRepo;  
import com.denka88.ateliergrace.repo.OrderRepo;  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.denka88.ateliergrace.service.OrderEmployeeService;  
import com.denka88.ateliergrace.service.OrderService;  
import jakarta.persistence.EntityNotFoundException;  
import jakarta.transaction.Transactional;  
import org.springframework.stereotype.Service;  
  
import java.time.LocalDate;  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class OrderServiceImpl implements OrderService {  
  
 private final OrderRepo orderRepo;  
 private final MaterialRepo materialRepo;  
 private final CurrentUserService currentUserService;  
 private final OrderEmployeeService orderEmployeeService;  
  
 public OrderServiceImpl(OrderRepo *orderRepo*, MaterialRepo *materialRepo*, CurrentUserService *currentUserService*, OrderEmployeeService *orderEmployeeService*) {  
 this.orderRepo = *orderRepo*;  
 this.materialRepo = *materialRepo*;  
 this.currentUserService = *currentUserService*;  
 this.orderEmployeeService = *orderEmployeeService*;  
 }  
  
 @Override  
 public List<Order> findAll() {  
 return orderRepo.findAllWithEmployeesAndMaterials();  
 }  
  
 @Override  
 public Optional<Order> findById(Long *id*) {  
 return orderRepo.findById(*id*);  
 }  
  
 @Override  
 @Transactional  
 public Order save(Order *order*) {  
 *order*.setOrderDate(LocalDate.*now*());  
 *order*.setStatus(Status.*PROGRESS*);  
   
  
 return orderRepo.save(*order*);  
 }  
  
 @Override  
 @Transactional  
 public void delete(Long *id*) {  
 orderRepo.deleteById(*id*);  
 }  
  
 @Override  
 @Transactional  
 public void update(Order *order*) {  
 orderRepo.save(*order*);  
 }  
  
 @Override  
 public List<Order> findByClientId(Long *clientId*) {  
 return orderRepo.findByClientId(*clientId*);  
 }  
  
 @Override  
 @Transactional  
 public void completeOrder(Long *orderId*) {  
 Order *order* = orderRepo.findById(*orderId*)  
 .orElseThrow(() -> new IllegalArgumentException("Заказ не найден"));  
   
 *order*.setStatus(Status.*COMPLETED*);  
 orderRepo.save(*order*);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.OrganizationMaterial;  
import com.denka88.ateliergrace.model.OrganizationMaterialKey;  
import com.denka88.ateliergrace.repo.OrganizationMaterialRepo;  
import com.denka88.ateliergrace.service.OrganizationMaterialService;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class OrganizationMaterialServiceImpl implements OrganizationMaterialService {  
   
 private final OrganizationMaterialRepo organizationMaterialRepo;  
  
 public OrganizationMaterialServiceImpl(OrganizationMaterialRepo *organizationMaterialRepo*) {  
 this.organizationMaterialRepo = *organizationMaterialRepo*;  
 }  
  
 @Override  
 public List<OrganizationMaterial> findAll() {  
 return organizationMaterialRepo.findAll();  
 }  
  
 @Override  
 public OrganizationMaterial save(OrganizationMaterial *organizationMaterial*) {  
 return organizationMaterialRepo.save(*organizationMaterial*);  
 }  
  
 @Override  
 public void update(OrganizationMaterial *organizationMaterial*) {  
 organizationMaterialRepo.save(*organizationMaterial*);  
 }  
  
 @Override  
 public Optional<OrganizationMaterial> findById(OrganizationMaterialKey *id*) {  
 return organizationMaterialRepo.findById(*id*);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Organization;  
import com.denka88.ateliergrace.repo.OrganizationMaterialRepo;  
import com.denka88.ateliergrace.repo.OrganizationRepo;  
import com.denka88.ateliergrace.service.OrganizationService;  
import jakarta.transaction.Transactional;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class OrganizationServiceImpl implements OrganizationService {  
  
 private final OrganizationRepo organizationRepo;  
 private final OrganizationMaterialRepo organizationMaterialRepo;  
  
 public OrganizationServiceImpl(OrganizationRepo *organizationRepo*, OrganizationMaterialRepo *organizationMaterialRepo*) {  
 this.organizationRepo = *organizationRepo*;  
 this.organizationMaterialRepo = *organizationMaterialRepo*;  
 }  
  
 @Override  
 public List<Organization> findAll() {  
 return organizationRepo.findAll();  
 }  
  
 @Override  
 public Optional<Organization> findById(Long *id*) {  
 return organizationRepo.findById(*id*);  
 }  
  
 @Override  
 public Organization save(Organization *organization*) {  
 return organizationRepo.save(*organization*);  
 }  
  
 @Override  
 @Transactional  
 public void delete(Long *id*) {  
 Organization *organization* = organizationRepo.findById(*id*).orElseThrow(()-> new RuntimeException("Поставщик не найден"));  
 organizationMaterialRepo.deleteByOrganizationId(*id*);  
 organizationRepo.deleteById(*id*);  
 }  
  
 @Override  
 public void update(Organization *organization*) {  
 organizationRepo.save(*organization*);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Auth;  
import com.denka88.ateliergrace.repo.AuthRepo;  
import com.denka88.ateliergrace.service.AuthService;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
  
import java.util.Collections;  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class UserDetailsServiceImpl implements UserDetailsService {  
  
 private final AuthRepo authRepo;  
 private final BCryptPasswordEncoder passwordEncoder;  
  
 public UserDetailsServiceImpl(AuthRepo authRepo, BCryptPasswordEncoder passwordEncoder) {  
 this.authRepo = authRepo;  
 this.passwordEncoder = passwordEncoder;  
 }  
  
 @Override  
 public UserDetails loadUserByUsername(String login) throws UsernameNotFoundException {  
 Auth auth = authRepo.findByLogin(login)  
 .orElseThrow(() -> new UsernameNotFoundException("User not found: " + login));  
  
 return User.builder()  
 .username(auth.getLogin())  
 .password(auth.getPasswordHash())  
 .authorities("ROLE\_" + auth.getUserType())  
 .build();  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "Auth", uniqueConstraints = {  
 @UniqueConstraint(columnNames = "login")  
})  
public class Auth {  
   
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long authId;  
  
 @Column(nullable = false, length = 25)  
 private String login;  
  
 @Column(nullable = false)  
 private String passwordHash;  
  
 @Enumerated(EnumType.STRING)  
 @Column(nullable = false)  
 private UserType userType;  
  
 @Column(nullable = false)  
 private Long userId;  
  
 public Auth() {}  
  
 public Auth(String login, String passwordHash, UserType userType, Long userId) {  
 this.login = login;  
 this.passwordHash = passwordHash;  
 this.userType = userType;  
 this.userId = userId;  
 }  
  
 public Long getAuthId() {  
 return authId;  
 }  
  
 public void setAuthId(Long authId) {  
 this.authId = authId;  
 }  
  
 public String getLogin() {  
 return login;  
 }  
  
 public void setLogin(String login) {  
 this.login = login;  
 }  
  
 public String getPasswordHash() {  
 return passwordHash;  
 }  
  
 public void setPasswordHash(String passwordHash) {  
 this.passwordHash = passwordHash;  
 }  
  
 public UserType getUserType() {  
 return userType;  
 }  
  
 public void setUserType(UserType userType) {  
 this.userType = userType;  
 }  
  
 public Long getUserId() {  
 return userId;  
 }  
  
 public void setUserId(Long userId) {  
 this.userId = userId;  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.\*;  
  
import java.util.Set;  
  
@Entity  
@Table(name = "clients")  
public class Client {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 private String name;  
 private String surname;  
 private String patronymic;  
  
 private String phone;  
  
 @OneToMany(mappedBy = "client", cascade = CascadeType.ALL, orphanRemoval = true)  
 private Set<Order> orders;  
  
 public Client() {  
 }  
  
 public Client(Long id, String name, String surname, String patronymic, String phone, Set<Order> orders) {  
 this.id = id;  
 this.name = name;  
 this.surname = surname;  
 this.patronymic = patronymic;  
 this.phone = phone;  
 this.orders = orders;  
 }  
  
 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 getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getPatronymic() {  
 return patronymic;  
 }  
  
 public void setPatronymic(String patronymic) {  
 this.patronymic = patronymic;  
 }  
  
 public String getPhone() {  
 return phone;  
 }  
  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
  
 public Set<Order> getOrders() {  
 return orders;  
 }  
  
 public void setOrders(Set<Order> orders) {  
 this.orders = orders;  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.\*;  
  
import java.util.Set;  
  
@Entity  
@Table(name = "employees")  
public class Employee {  
   
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
   
 private String post;  
  
 private String name;  
 private String surname;  
 private String patronymic;  
  
 @OneToMany(mappedBy = "employee", cascade = CascadeType.ALL, orphanRemoval = true)  
 private Set<OrderEmployee> orders;  
  
 public Employee() {  
 }  
  
 public Employee(Long id, String post, String name, String surname, String patronymic, Set<OrderEmployee> orders) {  
 this.id = id;  
 this.post = post;  
 this.name = name;  
 this.surname = surname;  
 this.patronymic = patronymic;  
 this.orders = orders;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getPost() {  
 return post;  
 }  
  
 public void setPost(String post) {  
 this.post = post;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getPatronymic() {  
 return patronymic;  
 }  
  
 public void setPatronymic(String patronymic) {  
 this.patronymic = patronymic;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
  
 public Set<OrderEmployee> getOrders() {  
 return orders;  
 }  
  
 public void setOrders(Set<OrderEmployee> orders) {  
 this.orders = orders;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Employee employee = (Employee) o;  
 return id != null && id.equals(employee.id);  
 }  
  
 @Override  
 public int hashCode() {  
 return getClass().hashCode();  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.\*;  
  
import java.util.List;  
import java.util.Objects;  
import java.util.Set;  
  
@Entity  
@Table(name = "materials")  
public class Material {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 private String name;  
  
 private int value;  
  
 @ManyToMany(mappedBy = "materials", fetch = FetchType.LAZY)  
 private Set<Order> orders;  
  
 public Material() {  
 }  
  
 public Material(Long id, String name, int value, Set<Order> orders) {  
 this.id = id;  
 this.name = name;  
 this.value = value;  
 this.orders = orders;  
 }  
  
 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 int getValue() {  
 return value;  
 }  
  
 public void setValue(int value) {  
 this.value = value;  
 }  
  
 public Set<Order> getOrders() {  
 return orders;  
 }  
  
 public void setOrders(Set<Order> orders) {  
 this.orders = orders;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Material material = (Material) o;  
 return id != null && id.equals(material.id);  
 }  
  
 @Override  
 public int hashCode() {  
 return getClass().hashCode();  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.\*;  
  
import java.time.LocalDate;  
import java.util.Optional;  
import java.util.Set;  
  
@Entity  
@Table(name = "orders")  
public class Order {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 @ManyToOne  
 @JoinColumn(name = "client\_id")  
 private Client client;  
  
 private String orderName;  
 private String description;  
 private LocalDate orderDate;  
 private float cost;  
   
 @Enumerated(EnumType.STRING)  
 private Status status;  
  
 @ManyToMany(fetch = FetchType.LAZY)  
 @JoinTable(  
 name = "order\_materials",  
 joinColumns = @JoinColumn(name = "order\_id"),  
 inverseJoinColumns = @JoinColumn(name = "material\_id")  
 )  
 private Set<Material> materials;  
  
 @OneToMany(mappedBy = "order", cascade = CascadeType.ALL, orphanRemoval = true)  
 private Set<OrderEmployee> orderEmployees;  
  
 public Order(Long id, Client client, String orderName, String description, LocalDate orderDate, float cost, Status status, Set<Material> materials, Set<OrderEmployee> orderEmployees) {  
 this.id = id;  
 this.client = client;  
 this.orderName = orderName;  
 this.description = description;  
 this.orderDate = orderDate;  
 this.cost = cost;  
 this.status = status;  
 this.materials = materials;  
 this.orderEmployees = orderEmployees;  
 }  
  
 public Order() {  
  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public Client getClient() {  
 return client;  
 }  
  
 public void setClient(Client client) {  
 this.client = client;  
 }  
  
 public String getOrderName() {  
 return orderName;  
 }  
  
 public void setOrderName(String orderName) {  
 this.orderName = orderName;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public void setDescription(String type) {  
 this.description = type;  
 }  
  
 public LocalDate getOrderDate() {  
 return orderDate;  
 }  
  
 public void setOrderDate(LocalDate orderDate) {  
 this.orderDate = orderDate;  
 }  
  
 public float getCost() {  
 return cost;  
 }  
  
 public void setCost(float cost) {  
 this.cost = cost;  
 }  
  
 public Status getStatus() {  
 return status;  
 }  
  
 public void setStatus(Status status) {  
 this.status = status;  
 }  
  
 public Set<OrderEmployee> getOrderEmployees() {  
 return orderEmployees;  
 }  
  
 public void setOrderEmployees(Set<OrderEmployee> orders) {  
 this.orderEmployees = orders;  
 }  
  
 public Set<Material> getMaterials() {  
 return materials;  
 }  
  
 public void setMaterials(Set<Material> materials) {  
 this.materials = materials;  
 }  
  
 public void setClient(Optional<Client> client) {  
 this.client = client.orElse(null);  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.\*;  
import org.hibernate.annotations.OnDelete;  
import org.hibernate.annotations.OnDeleteAction;  
  
import java.time.LocalDate;  
import java.util.Date;  
  
@Entity  
public class OrderEmployee {  
  
 @EmbeddedId  
 private OrderEmployeeKey id;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @MapsId("employeeId")  
 @JoinColumn(name = "employee\_id")  
 @OnDelete(action = OnDeleteAction.CASCADE)  
 private Employee employee;  
  
 @ManyToOne(fetch = FetchType.LAZY)  
 @MapsId("orderId")  
 @JoinColumn(name = "order\_id")  
 @OnDelete(action = OnDeleteAction.CASCADE)  
 private Order order;  
   
 private LocalDate dateOfReady;  
  
 public OrderEmployee() {  
 }  
  
 public OrderEmployee(OrderEmployeeKey id, Employee employee, Order order, LocalDate dateOfReady) {  
 this.id = id;  
 this.employee = employee;  
 this.order = order;  
 this.dateOfReady = dateOfReady;  
 }  
  
 public OrderEmployeeKey getId() {  
 return id;  
 }  
  
 public void setId(OrderEmployeeKey id) {  
 this.id = id;  
 }  
  
 public Employee getEmployee() {  
 return employee;  
 }  
  
 public void setEmployee(Employee employee) {  
 this.employee = employee;  
 }  
  
 public Order getOrder() {  
 return order;  
 }  
  
 public void setOrder(Order order) {  
 this.order = order;  
 }  
  
 public LocalDate getDateOfReady() {  
 return dateOfReady;  
 }  
  
 public void setDateOfReady(LocalDate dateOfReady) {  
 this.dateOfReady = dateOfReady;  
 }  
  
 @Override  
 public String toString() {  
 return employee.getSurname() + " " + employee.getName() + " " + employee.getPatronymic();  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Embeddable;  
import jakarta.persistence.EmbeddedId;  
import jakarta.persistence.Entity;  
  
import java.io.Serializable;  
import java.util.Objects;  
  
  
@Embeddable  
public class OrderEmployeeKey implements Serializable {  
  
 @Column(name = "order\_id")  
 private Long orderId;  
  
 @Column(name = "employee\_id")  
 private Long employeeId;  
  
 public OrderEmployeeKey() {  
 }  
  
 public OrderEmployeeKey(Long orderId, Long employeeId) {  
 this.orderId = orderId;  
 this.employeeId = employeeId;  
 }  
  
 public Long getOrderId() {  
 return orderId;  
 }  
  
 public void setOrderId(Long orderId) {  
 this.orderId = orderId;  
 }  
  
 public Long getEmployeeId() {  
 return employeeId;  
 }  
  
 public void setEmployeeId(Long employeeId) {  
 this.employeeId = employeeId;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o == null || getClass() != o.getClass()) return false;  
 OrderEmployeeKey that = (OrderEmployeeKey) o;  
 return Objects.equals(orderId, that.orderId) && Objects.equals(employeeId, that.employeeId);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(orderId, employeeId);  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.\*;  
  
import java.util.HashSet;  
import java.util.Objects;  
import java.util.Set;  
  
@Entity  
@Table(name = "organizations")  
public class Organization {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
  
 private String name;  
 private String address;  
  
 @OneToMany(mappedBy = "organization", cascade = CascadeType.ALL, orphanRemoval = true)  
 private Set<OrganizationMaterial> materials;  
  
  
 public Organization() {  
 }  
  
 public Organization(Long id, String name, String address, Set<OrganizationMaterial> materials) {  
 this.id = id;  
 this.name = name;  
 this.address = address;  
 this.materials = materials;  
 }  
  
 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 getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 @Override  
 public String toString() {  
 return name;  
 }  
  
 public Set<OrganizationMaterial> getMaterials() {  
 return materials;  
 }  
  
 public void setMaterials(Set<OrganizationMaterial> materials) {  
 this.materials = materials;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Organization that = (Organization) o;  
 return id != null && id.equals(that.id);  
 }  
  
 @Override  
 public int hashCode() {  
 return getClass().hashCode();  
 }  
}

package com.denka88.ateliergrace.model;  
  
  
import jakarta.persistence.EmbeddedId;  
import jakarta.persistence.Entity;  
import jakarta.persistence.ManyToOne;  
import jakarta.persistence.MapsId;  
import org.hibernate.annotations.OnDelete;  
import org.hibernate.annotations.OnDeleteAction;  
  
@Entity  
public class OrganizationMaterial {  
   
 @EmbeddedId  
 private OrganizationMaterialKey id;  
   
 @ManyToOne  
 @MapsId("materialId")  
 @OnDelete(action = OnDeleteAction.CASCADE)  
 private Material material;  
   
 @ManyToOne  
 @MapsId("organizationId")  
 @OnDelete(action = OnDeleteAction.CASCADE)  
 private Organization organization;  
   
 private float cost;  
 private int value;  
  
 public OrganizationMaterial() {  
 }  
  
 public OrganizationMaterial(OrganizationMaterialKey id, Material material, Organization organization, float cost, int value) {  
 this.id = id;  
 this.material = material;  
 this.organization = organization;  
 this.cost = cost;  
 this.value = value;  
 }  
  
 public OrganizationMaterialKey getId() {  
 return id;  
 }  
  
 public void setId(OrganizationMaterialKey id) {  
 this.id = id;  
 }  
  
 public Material getMaterial() {  
 return material;  
 }  
  
 public void setMaterial(Material material) {  
 this.material = material;  
 }  
  
 public Organization getOrganization() {  
 return organization;  
 }  
  
 public void setOrganization(Organization organization) {  
 this.organization = organization;  
 }  
  
 public float getCost() {  
 return cost;  
 }  
  
 public void setCost(float cost) {  
 this.cost = cost;  
 }  
  
 public int getValue() {  
 return value;  
 }  
  
 public void setValue(int value) {  
 this.value = value;  
 }  
}

package com.denka88.ateliergrace.model;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Embeddable;  
  
import java.io.Serializable;  
import java.util.Objects;  
  
@Embeddable  
public class OrganizationMaterialKey implements Serializable {  
  
 @Column(name = "organization\_id")  
 private Long organizationId;  
  
 @Column(name = "material\_id")  
 private Long materialId;  
  
 public OrganizationMaterialKey() {  
 }  
  
 public OrganizationMaterialKey(Long organizationId, Long materialId) {  
 this.organizationId = organizationId;  
 this.materialId = materialId;  
 }  
  
 public Long getOrganizationId() {  
 return organizationId;  
 }  
  
 public void setOrganizationId(Long organizationId) {  
 this.organizationId = organizationId;  
 }  
  
 public Long getMaterialId() {  
 return materialId;  
 }  
  
 public void setMaterialId(Long materialId) {  
 this.materialId = materialId;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o == null || getClass() != o.getClass()) return false;  
 OrganizationMaterialKey that = (OrganizationMaterialKey) o;  
 return Objects.equals(organizationId, that.organizationId) && Objects.equals(materialId, that.materialId);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(organizationId, materialId);  
 }  
}

package com.denka88.ateliergrace.model;  
  
public enum Status {  
  
 PROGRESS("Выполняется"),  
 COMPLETED("Выполнен");  
  
 Status(String status) {  
  
 }  
  
  
}

package com.denka88.ateliergrace.model;  
  
public enum UserType {  
 CLIENT("Клиент"),   
 EMPLOYEE("Сотрудник"),   
 ADMIN("Администратор");  
  
 UserType(String type) {  
 }  
}

package com.denka88.ateliergrace.repo;  
  
  
import com.denka88.ateliergrace.model.Auth;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Modifying;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
import java.util.Optional;  
  
public interface AuthRepo extends JpaRepository<Auth, Long> {  
 Optional<Auth> findByLogin(String login);  
  
 boolean existsByLogin(String login);  
  
 @Modifying  
 @Query("DELETE FROM Auth a WHERE a.userId = :userId")  
 void deleteByUserId(@Param("userId") Long userId);  
}

package com.denka88.ateliergrace.repo;  
  
import com.denka88.ateliergrace.model.Client;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface ClientRepo extends JpaRepository<Client, Long> {  
  
}

package com.denka88.ateliergrace.repo;  
  
import com.denka88.ateliergrace.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface EmployeeRepo extends JpaRepository<Employee, Long> {  
   
}

package com.denka88.ateliergrace.repo;  
  
import com.denka88.ateliergrace.model.Material;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface MaterialRepo extends JpaRepository<Material, Long> {  
}

package com.denka88.ateliergrace.repo;  
  
import com.denka88.ateliergrace.model.OrderEmployee;  
import com.denka88.ateliergrace.model.OrderEmployeeKey;  
import com.denka88.ateliergrace.model.Status;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
import java.util.List;  
  
public interface OrderEmployeeRepo extends JpaRepository<OrderEmployee, OrderEmployeeKey> {  
  
 @Query("SELECT oe.employee.id, COUNT(oe) FROM OrderEmployee oe " +  
 "WHERE oe.order.status = com.denka88.ateliergrace.model.Status.PROGRESS " +  
 "GROUP BY oe.employee.id")  
 List<Object[]> countInProgressOrdersPerEmployee();  
   
 @Query("SELECT COUNT(oe) FROM OrderEmployee oe " +  
 "WHERE oe.employee.id = :employeeId " +  
 "AND oe.order.status = com.denka88.ateliergrace.model.Status.PROGRESS")  
 Long countByEmployeeIdAndOrderStatus(@Param("employeeId") Long employeeId, Status status);  
}

package com.denka88.ateliergrace.repo;  
  
import com.denka88.ateliergrace.model.Order;  
import org.springframework.data.jpa.repository.EntityGraph;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import java.util.List;  
  
public interface OrderRepo extends JpaRepository<Order, Long> {  
  
 @Query("SELECT DISTINCT o FROM Order o LEFT JOIN FETCH o.orderEmployees oe LEFT JOIN FETCH oe.employee LEFT JOIN FETCH o.materials")  
 List<Order> findAllWithEmployeesAndMaterials();  
  
 @EntityGraph(attributePaths = {"orderEmployees.employee", "materials"})  
 List<Order> findByClientId(Long *clientId*);  
  
 @Query("SELECT o FROM Order o JOIN o.materials m WHERE m.id = :materialId")  
 List<Order> findByMaterialsId(Long *materialId*);  
}

package com.denka88.ateliergrace.repo;  
  
import com.denka88.ateliergrace.model.OrganizationMaterial;  
import com.denka88.ateliergrace.model.OrganizationMaterialKey;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Modifying;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
public interface OrganizationMaterialRepo extends JpaRepository<OrganizationMaterial, OrganizationMaterialKey> {  
  
 @Modifying  
 @Query("DELETE FROM OrganizationMaterial om WHERE om.organization.id = :organizationId")  
 void deleteByOrganizationId(@Param("organizationId") Long organizationId);  
   
}

package com.denka88.ateliergrace.repo;  
  
import com.denka88.ateliergrace.model.Organization;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface OrganizationRepo extends JpaRepository<Organization, Long> {  
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.Auth;  
import com.denka88.ateliergrace.model.UserType;  
  
import java.util.Optional;  
  
public interface AuthService {  
   
 void register(String login, String password, UserType userType, Long userId);  
   
 void delete(Long userId);  
   
 Optional<Auth> findByLogin(String login);  
   
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.Client;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface ClientService {  
  
 List<Client> findAll();  
  
 Optional<Client> findById(Long id);  
  
 Client save(Client client);  
  
 void delete(Long id);  
  
 void update(Client client);  
  
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.Auth;  
import com.denka88.ateliergrace.model.Client;  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.model.UserType;  
  
import java.util.Optional;  
  
public interface CurrentUserService {  
   
 Optional<Auth> getCurrentAuth();  
   
 Optional<Client> getCurrentClient();  
   
 Optional<Employee> getCurrentEmployee();  
   
 Long getCurrentUserId();  
   
 UserType getCurrentUserType();  
   
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.Employee;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface EmployeeService {  
   
 List<Employee> findAll();  
   
 Optional<Employee> findById(Long id);  
   
 Employee save(Employee employee);  
   
 void delete(Long id);  
   
 void update(Employee employee);  
  
 List<Employee> findAllOrderByInProgressOrdersCount();  
   
   
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.Material;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface MaterialService {  
  
 List<Material> findAll();  
  
 Optional<Material> findById(Long id);  
  
 Material save(Material material);  
  
 void delete(Long id);  
  
 void update(Material material);  
  
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.model.OrderEmployee;  
import com.denka88.ateliergrace.model.OrderEmployeeKey;  
  
import java.time.LocalDate;  
import java.util.List;  
import java.util.Optional;  
  
public interface OrderEmployeeService {  
   
 List<OrderEmployee> findAll();  
   
 OrderEmployee save(OrderEmployee orderEmployee);  
   
 void update(OrderEmployee orderEmployee);  
   
 void setReadyDate(Long orderId, Long employeeId, LocalDate readyDate);  
  
 void createAssignment(Long orderId, Long employeeId, LocalDate readyDate);  
   
 Long getInProgressCount(Long employeeId);  
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.Order;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface OrderService {  
   
 List<Order> findAll();  
   
 Optional<Order> findById(Long id);  
   
 Order save(Order order);  
   
 void delete(Long id);  
   
 void update(Order order);  
   
 List<Order> findByClientId(Long clientId);  
  
 void completeOrder(Long orderId);  
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.OrganizationMaterial;  
import com.denka88.ateliergrace.model.OrganizationMaterialKey;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface OrganizationMaterialService {  
   
 List<OrganizationMaterial> findAll();  
   
 OrganizationMaterial save(OrganizationMaterial organizationMaterial);  
   
 void update(OrganizationMaterial organizationMaterial);  
   
 Optional<OrganizationMaterial> findById(OrganizationMaterialKey id);  
}

package com.denka88.ateliergrace.service;  
  
import com.denka88.ateliergrace.model.Organization;  
  
import java.util.List;  
import java.util.Optional;  
  
public interface OrganizationService {  
  
 List<Organization> findAll();  
  
 Optional<Organization> findById(Long id);  
  
 Organization save(Organization organization);  
  
 void delete(Long id);  
  
 void update(Organization organization);  
  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.Text;  
import com.vaadin.flow.component.details.Details;  
import com.vaadin.flow.component.html.\*;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.PermitAll;  
  
@Route(value = "about", layout = MainLayout.class)  
@PageTitle("О программе")  
@PermitAll  
public class AboutView extends VerticalLayout {  
  
 public AboutView() {  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
  
 H1 title = new H1("Программа \"Ателье Грация\"");  
 title.addClassName(LumoUtility.Margin.Bottom.LARGE);  
 title.addClassName(LumoUtility.TextAlignment.CENTER);  
  
 Paragraph description = new Paragraph(  
 "Программа \"Ателье Грация\" представляет собой веб-приложение для управления ателье по пошиву одежды. " +  
 "Это система учета заказов, клиентов, материалов и сотрудников с разграничением прав доступа."  
 );  
 description.addClassName(LumoUtility.Margin.Bottom.LARGE);  
  
 Details functionalitySection = createSection(  
 "Основной функционал",  
 VaadinIcon.COG,  
 createFunctionalContent()  
 );  
  
 Details technicalSection = createSection(  
 "Технические особенности",  
 VaadinIcon.WRENCH,  
 createTechnicalContent()  
 );  
  
 Details modulesSection = createSection(  
 "Основные модули",  
 VaadinIcon.CUBES,  
 createModulesContent()  
 );  
  
 Details authorSection = createSection(  
 "Автор приложения",  
 VaadinIcon.USER,  
 createAuthorContent()  
 );  
  
 add(  
 title,  
 description,  
 functionalitySection,  
 technicalSection,  
 modulesSection,  
 authorSection  
 );  
 }  
  
 private Component createFunctionalContent() {  
 VerticalLayout content = new VerticalLayout();  
 content.setSpacing(true);  
 content.setPadding(false);  
  
 Div adminDiv = new Div(new Text("Для администратора:"));  
 UnorderedList adminList = new UnorderedList(  
 new ListItem("Управление сотрудниками (добавление, редактирование, удаление)"),  
 new ListItem("Управление клиентами"),  
 new ListItem("Управление заказами"),  
 new ListItem("Управление материалами и поставщиками"),  
 new ListItem("Просмотр и управление поставками материалов")  
 );  
 adminDiv.add(adminList);  
  
 Div employeeDiv = new Div(new Text("Для сотрудников:"));  
 UnorderedList employeeList = new UnorderedList(  
 new ListItem("Создание и редактирование заказов"),  
 new ListItem("Просмотр списка клиентов"),  
 new ListItem("Управление материалами и поставщиками"),  
 new ListItem("Отслеживание поставок"),  
 new ListItem("Завершение заказов")  
 );  
 employeeDiv.add(employeeList);  
  
 Div clientDiv = new Div(new Text("Для клиентов:"));  
 UnorderedList clientList = new UnorderedList(  
 new ListItem("Просмотр своих заказов"),  
 new ListItem("Редактирование профиля")  
 );  
 clientDiv.add(clientList);  
  
 content.add(adminDiv, employeeDiv, clientDiv);  
 return content;  
 }  
  
 private Component createTechnicalContent() {  
 UnorderedList list = new UnorderedList(  
 new ListItem("Реализована система аутентификации и авторизации"),  
 new ListItem("Разграничение прав доступа по ролям (ADMIN, EMPLOYEE, CLIENT)"),  
 new ListItem("Валидация вводимых данных"),  
 new ListItem("Удобный интерфейс с фильтрацией и сортировкой"),  
 new ListItem("Уведомления об успешных операциях и ошибках")  
 );  
 return list;  
 }  
  
 private Component createModulesContent() {  
 OrderedList list = new OrderedList(  
 new ListItem("Управление заказами - создание, редактирование, просмотр и фильтрация заказов"),  
 new ListItem("Управление клиентами - регистрация, просмотр и редактирование данных клиентов"),  
 new ListItem("Управление сотрудниками - регистрация и управление учетными записями сотрудников"),  
 new ListItem("Управление материалами - учет материалов на складе"),  
 new ListItem("Управление поставщиками - ведение базы поставщиков материалов"),  
 new ListItem("Учет поставок - фиксация поступлений материалов от поставщиков"),  
 new ListItem("Личный кабинет клиента - просмотр своих заказов и профиля")  
 );  
 return list;  
 }  
  
 private Details createSection(String summary, VaadinIcon icon, Component content) {  
 Details section = new Details();  
 section.setSummaryText(summary);  
 section.setOpened(true);  
  
 Span summarySpan = new Span(icon.create(), new Text(" " + summary));  
 section.setSummary(summarySpan);  
  
 section.add(content);  
 section.addClassName(LumoUtility.Margin.Bottom.MEDIUM);  
 return section;  
 }  
  
 private Component createAuthorContent() {  
 VerticalLayout content = new VerticalLayout();  
 content.setSpacing(true);  
 content.setPadding(false);  
  
 Paragraph authorInfo = new Paragraph(  
 "Разработчик приложения: Григорьев Денис Владиславович. Псевдоним: Denka88. " +  
 "Для связи или получения дополнительной информации используйте следующие контакты:"  
 );  
  
 UnorderedList contactsList = new UnorderedList(  
 new ListItem("Телефон: +7 (902) 758-00-03")  
 );  
  
 content.add(authorInfo, contactsList);  
 return content;  
 }  
  
  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.Client;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.service.AuthService;  
import com.denka88.ateliergrace.service.ClientService;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H2;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.PasswordField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
@Route(value = "add-client", layout = MainLayout.class)  
@PageTitle("Добавить клиента")  
@RolesAllowed({"EMPLOYEE", "ADMIN"})  
public class AddClientView extends VerticalLayout {  
  
 private final ClientService clientService;  
 private final AuthService authService;  
  
 public AddClientView(ClientService clientService, AuthService authService) {  
 this.clientService = clientService;  
 this.authService = authService;  
  
 setSizeFull();  
 setSpacing(false);  
 setPadding(true);  
 addClassName(LumoUtility.Background.CONTRAST\_5);  
  
 Div card = new Div();  
 card.addClassNames(  
 LumoUtility.Padding.XLARGE,  
 LumoUtility.BorderRadius.LARGE,  
 LumoUtility.BoxShadow.MEDIUM,  
 LumoUtility.Background.BASE,  
 "registration-card"  
 );  
 card.setWidth("500px");  
  
 H2 title = new H2("Добавление клиента");  
 title.addClassNames(LumoUtility.Margin.Bottom.LARGE, LumoUtility.TextAlignment.CENTER);  
  
 TextField login = new TextField("Логин");  
 login.setWidthFull();  
 login.setRequired(true);  
 login.setPlaceholder("Введите логин");  
 login.setMinLength(5);  
  
 PasswordField password = new PasswordField("Пароль");  
 password.setWidthFull();  
 password.setRequired(true);  
 password.setPlaceholder("Введите пароль");  
 password.setMinLength(8);  
  
 TextField name = new TextField("Имя");  
 name.setWidthFull();  
 name.setRequired(true);  
 name.setPlaceholder("Введите ваше имя");  
  
 TextField surname = new TextField("Фамилия");  
 surname.setWidthFull();  
 surname.setRequired(true);  
 surname.setPlaceholder("Введите вашу фамилию");  
  
 TextField patronymic = new TextField("Отчество");  
 patronymic.setWidthFull();  
 patronymic.setPlaceholder("Введите ваше отчество (если есть)");  
  
 TextField phone = new TextField("Телефон");  
 phone.setWidthFull();  
 phone.setRequired(true);  
 phone.setPlaceholder("Введите номер телефона");  
  
 Button registerButton = new Button("Добавить", VaadinIcon.USER\_CHECK.create());  
 registerButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 registerButton.setWidthFull();  
 registerButton.addClassName(LumoUtility.Margin.Top.MEDIUM);  
  
 registerButton.addClickListener(e -> {  
 if (login.isEmpty() || password.isEmpty() || name.isEmpty() || surname.isEmpty() || phone.isEmpty()) {  
 showError("Заполните все обязательные поля");  
 return;  
 }  
  
 if(login.getValue().length() < 5){  
 showError("Неподходящая длина логина. Минимум 5 символов");  
 return;  
 }  
  
 if(password.getValue().length() < 8){  
 showError("Неподходящая длина пароля. Минимум 8 символов");  
 return;  
 }  
  
 try {  
 Client client = new Client();  
 client.setName(name.getValue());  
 client.setSurname(surname.getValue());  
 client.setPatronymic(patronymic.getValue());  
 client.setPhone(phone.getValue());  
  
 Client savedClient = clientService.save(client);  
 authService.register(  
 login.getValue(),  
 password.getValue(),  
 UserType.CLIENT,  
 savedClient.getId()  
 );  
  
 showSuccess("Регистрация успешна!");  
 UI.getCurrent().navigate("clients-list");  
 } catch (Exception ex) {  
 showError("Ошибка регистрации: " + ex.getMessage());  
 }  
 });  
  
 FormLayout form = new FormLayout();  
 form.addClassNames(LumoUtility.Padding.NONE);  
 form.add(  
 title, login, password,  
 name, surname, patronymic, phone,  
 registerButton  
 );  
 form.setResponsiveSteps(  
 new FormLayout.ResponsiveStep("0", 1),  
 new FormLayout.ResponsiveStep("500px", 2)  
 );  
 form.setColspan(title, 2);  
 form.setColspan(registerButton, 2);  
  
 card.add(form);  
  
 Div container = new Div(card);  
 container.setSizeFull();  
 container.addClassNames(  
 LumoUtility.Display.FLEX,  
 LumoUtility.JustifyContent.CENTER,  
 LumoUtility.AlignItems.CENTER  
 );  
  
 add(container);  
 }  
  
 private void showError(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
  
 private void showSuccess(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_SUCCESS);  
 }  
  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.\*;  
import com.denka88.ateliergrace.service.\*;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.combobox.MultiSelectComboBox;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.select.Select;  
import com.vaadin.flow.component.textfield.NumberField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.time.LocalDate;  
import java.util.HashSet;  
import java.util.Set;  
  
@PageTitle("Создать заказ")  
@Route(value = "add-order", layout = MainLayout.class)  
@RolesAllowed({"EMPLOYEE", "ADMIN"})  
public class AddOrderView extends VerticalLayout {  
  
 private final OrderService orderService;  
 private final MaterialService materialService;  
 private final CurrentUserService currentUserService;  
 private final ClientService clientService;  
 private final EmployeeService employeeService;  
 private final OrderEmployeeService orderEmployeeService;  
  
 public AddOrderView(OrderService orderService, MaterialService materialService,  
 CurrentUserService currentUserService, ClientService clientService, EmployeeService employeeService, OrderEmployeeService orderEmployeeService) {  
 this.orderService = orderService;  
 this.materialService = materialService;  
 this.currentUserService = currentUserService;  
 this.clientService = clientService;  
 this.employeeService = employeeService;  
 this.orderEmployeeService = orderEmployeeService;  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
  
 FormLayout form = new FormLayout();  
 form.addClassName(LumoUtility.Padding.LARGE);  
 form.addClassName(LumoUtility.BorderRadius.LARGE);  
 form.addClassName(LumoUtility.BoxShadow.SMALL);  
 form.addClassName(LumoUtility.Background.BASE);  
 form.setWidth("500px");  
 form.getStyle().set("margin", "auto");  
  
 TextField orderName = new TextField("Название заказа");  
 orderName.setWidthFull();  
 orderName.setRequired(true);  
 orderName.setPlaceholder("Введите название заказа");  
  
 TextField description = new TextField("Описание заказа");  
 description.setWidthFull();  
 description.setRequired(true);  
 description.setPlaceholder("Укажите Описание заказа");  
  
 MultiSelectComboBox<Material> materials = new MultiSelectComboBox<>("Материалы");  
 materials.setItems(materialService.findAll());  
 materials.setItemLabelGenerator(Material::getName);  
 materials.setWidthFull();  
 materials.setPlaceholder("Выберите материалы");  
 materials.setRequired(true);  
  
 Select<Client> clients = new Select<>();  
 clients.setLabel("Клиент");  
 clients.setItems(clientService.findAll());  
 clients.setItemLabelGenerator(Client::getName);  
 clients.setWidthFull();  
 clients.setPlaceholder("Выберите клиента");  
   
 MultiSelectComboBox<Employee> employees = new MultiSelectComboBox<>("Ответственные сотрудники");  
 employees.setItems(employeeService.findAllOrderByInProgressOrdersCount());  
 employees.setItemLabelGenerator(e ->  
 String.format("%s %s (активных: %d)",  
 e.getSurname(),  
 e.getName(),  
 orderEmployeeService.getInProgressCount(e.getId()))  
 );  
 employees.setWidthFull();  
 employees.setPlaceholder("Выберите сотрудников");  
  
 NumberField cost = new NumberField("Цена");  
 cost.setMin(0);  
 cost.setPlaceholder("Введите цену");  
   
 Button create = new Button("Создать заказ", VaadinIcon.CHECK.create());  
 create.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 create.setWidthFull();  
  
 create.addClickListener(e -> {  
 if (orderName.isEmpty() || description.isEmpty() || materials.isEmpty() || clients.isEmpty()) {  
 showError("Заполните все поля");  
 return;  
 }  
  
 Order order = new Order();  
 order.setOrderName(orderName.getValue());  
 order.setDescription(description.getValue());  
 order.setClient(clients.getValue());  
 order.setCost(cost.getValue().floatValue());  
   
  
 Set<Material> materialSet = new HashSet<>(materials.getValue());  
 order.setMaterials(materialSet);  
  
 try {  
 Order savedOrder = orderService.save(order);  
  
 if (!employees.getValue().isEmpty()) {  
 employees.getValue().forEach(employee -> {  
 orderEmployeeService.createAssignment(  
 savedOrder.getId(),  
 employee.getId(),  
 LocalDate.now().plusDays(14)  
 );  
 });  
 }  
   
 Notification.show(String.format(  
 "Заказ #%d '%s' успешно создан!",  
 savedOrder.getId(), savedOrder.getOrderName()),  
 5000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_SUCCESS);  
  
   
   
 UI.getCurrent().navigate("orders-list");  
 } catch (Exception ex) {  
 Notification.show("Ошибка при создании заказа: " + ex.getMessage(),  
 5000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
 });  
  
 form.add(orderName, description, materials, clients, employees, cost, create);  
 form.setResponsiveSteps(  
 new FormLayout.ResponsiveStep("0", 1)  
 );  
  
 Div container = new Div(form);  
 container.setWidthFull();  
 container.addClassName(LumoUtility.Display.FLEX);  
 container.addClassName(LumoUtility.JustifyContent.CENTER);  
  
 add(container);  
 setAlignItems(Alignment.CENTER);  
 }  
  
 private void showError(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.Client;  
import com.denka88.ateliergrace.service.ClientService;  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.vaadin.flow.component.Component;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.html.H2;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.util.Optional;  
  
@Route(value = "client-profile", layout = MainLayout.class)  
@PageTitle("Мой профиль")  
@RolesAllowed("CLIENT")  
public class ClientProfileView extends VerticalLayout {  
  
 private final CurrentUserService currentUserService;  
 private final ClientService clientService;  
 private final TextField id = new TextField("ID");  
 private final TextField surnameField = new TextField("Фамилия");  
 private final TextField nameField = new TextField("Имя");  
 private final TextField patronymicField = new TextField("Отчество");  
 private final TextField phoneField = new TextField("Номер телефона");  
  
 Button confirm = new Button("Применить изменения", VaadinIcon.*CHECK*.create());  
 Button reset = new Button("Отменить изменения", VaadinIcon.*CLOSE*.create());  
 Button editProfile = new Button("Редактировать данные", VaadinIcon.*EDIT*.create());  
   
   
  
 public ClientProfileView(CurrentUserService *currentUserService*, ClientService *clientService*) {  
 this.currentUserService = *currentUserService*;  
 this.clientService = *clientService*;  
  
 setSizeFull();  
 setAlignItems(Alignment.*CENTER*);  
   
 id.setVisible(false);  
  
   
 confirm.addThemeVariants(ButtonVariant.*LUMO\_PRIMARY*);  
 confirm.addClassName(LumoUtility.Margin.Bottom.*LARGE*);  
 confirm.setVisible(false);  
 confirm.addClickListener(*e*->{  
 surnameField.setReadOnly(true);  
 nameField.setReadOnly(true);  
 patronymicField.setReadOnly(true);  
 phoneField.setReadOnly(true);  
   
 editProfile.setVisible(true);  
 confirm.setVisible(false);  
 reset.setVisible(false);  
  
 Client updateClient = clientService.findById(Long.valueOf(id.getValue())).orElse(null);  
 if (updateClient != null) {  
 updateClient.setSurname(surnameField.getValue());  
 updateClient.setName(nameField.getValue());  
 updateClient.setPatronymic(patronymicField.getValue());  
 updateClient.setPhone(phoneField.getValue());  
 clientService.update(updateClient);  
 loadClientData();  
 Notification.show("Профиль обновлен")  
 .setPosition(Notification.Position.TOP\_CENTER);  
 }  
 });  
   
 reset.addThemeVariants(ButtonVariant.LUMO\_ERROR);  
 reset.addClassName(LumoUtility.Margin.Bottom.LARGE);  
 reset.setVisible(false);  
 reset.addClickListener(e->{  
 loadClientData();  
   
 surnameField.setReadOnly(true);  
 nameField.setReadOnly(true);  
 patronymicField.setReadOnly(true);  
 phoneField.setReadOnly(true);  
   
 editProfile.setVisible(true);  
 confirm.setVisible(false);  
 reset.setVisible(false);  
   
 });  
  
 editProfile.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 editProfile.addClassName(LumoUtility.Margin.Bottom.LARGE);  
 editProfile.addClickListener(e->{  
 surnameField.setReadOnly(false);  
 nameField.setReadOnly(false);  
 patronymicField.setReadOnly(false);  
 phoneField.setReadOnly(false);  
   
 editProfile.setVisible(false);  
 confirm.setVisible(true);  
 reset.setVisible(true);  
 });  
  
 HorizontalLayout buttons = new HorizontalLayout(editProfile, confirm, reset);  
 buttons.setWidthFull();  
 buttons.setJustifyContentMode(JustifyContentMode.CENTER);  
   
   
 add(  
 new H2("Мой профиль"),  
 createFormLayout(),  
 buttons  
 );  
  
 loadClientData();  
 }  
  
 private Component createFormLayout() {  
 FormLayout formLayout = new FormLayout();  
 formLayout.add(  
 surnameField,  
 nameField,  
 patronymicField,  
 phoneField  
 );  
  
 surnameField.setWidth("300px");  
 nameField.setWidth("300px");  
 patronymicField.setWidth("300px");  
 phoneField.setWidth("300px");  
  
 surnameField.setReadOnly(true);  
 nameField.setReadOnly(true);  
 patronymicField.setReadOnly(true);  
 phoneField.setReadOnly(true);  
  
 return formLayout;  
 }  
  
 private void loadClientData() {  
 Optional<Client> optionalClient = currentUserService.getCurrentClient();  
 optionalClient.ifPresent(client -> {  
 id.setValue(client.getId() != null ? String.valueOf(client.getId()) : "");  
 surnameField.setValue(client.getSurname() != null ? client.getSurname() : "");  
 nameField.setValue(client.getName() != null ? client.getName() : "");  
 patronymicField.setValue(client.getPatronymic() != null ? client.getPatronymic() : "");  
 phoneField.setValue(client.getPhone() != null ? client.getPhone() : "");  
 });  
 }  
   
   
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.Client;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.service.ClientService;  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.dialog.Dialog;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.contextmenu.GridContextMenu;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H3;  
import com.vaadin.flow.component.icon.Icon;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.util.List;  
  
@PageTitle("Клиенты")  
@Route(value = "clients-list", layout = MainLayout.class)  
@RolesAllowed({"ADMIN", "EMPLOYEE"})  
public class ClientsView extends VerticalLayout {  
  
 private final ClientService clientService;  
 private final Grid<Client> grid;  
 private final CurrentUserService currentUserService;  
  
 private FormLayout editForm = new FormLayout();  
  
 private TextField id = new TextField("ID");  
 private TextField editSurname = new TextField("Фамилия");  
 private TextField editName = new TextField("Имя");  
 private TextField editPatronymic = new TextField("Отчество");  
 private Button editButton = new Button("Сохранить", VaadinIcon.CHECK.create());  
  
 public ClientsView(ClientService clientService, CurrentUserService currentUserService) {  
 this.clientService = clientService;  
 this.currentUserService = currentUserService;  
 this.grid = new Grid<>(Client.class, false);  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
  
 setupGrid();  
 updateGrid();  
   
 Button addClient = new Button("Добавить клиента", VaadinIcon.PLUS.create());  
 addClient.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 addClient.addClassName(LumoUtility.Margin.Bottom.LARGE);  
 addClient.addClickListener(e->{  
 UI.getCurrent().navigate("/add-client");  
 });  
  
 editForm.setWidth("400px");  
 editForm.addClassNames(  
 LumoUtility.Padding.LARGE,  
 LumoUtility.BorderRadius.LARGE,  
 LumoUtility.BoxShadow.SMALL,  
 LumoUtility.Background.BASE  
 );  
  
 id.setVisible(false);  
 editName.setRequired(true);  
 editSurname.setRequired(true);  
 editPatronymic.setRequired(true);  
  
 editButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 editButton.addClassName(LumoUtility.Margin.Top.MEDIUM);  
 editButton.getStyle().set("margin-left", "auto");  
  
 editButton.addClickListener(e->{  
 if (editSurname.isEmpty() || editName.isEmpty() || editPatronymic.isEmpty()) {  
 showError("Заполните все поля");  
 return;  
 }  
   
 Client updateClient = clientService.findById(Long.valueOf(id.getValue())).orElse(null);  
 if (updateClient != null) {  
 updateClient.setSurname(editSurname.getValue());  
 updateClient.setName(editName.getValue());  
 updateClient.setPatronymic(editPatronymic.getValue());  
 clientService.update(updateClient);  
 updateGrid();  
 editForm.setVisible(false);  
 }  
 });  
  
 grid.addCellFocusListener(e->{  
 id.setValue(String.valueOf(e.getItem().map(Client::getId).orElse(null)));  
 editSurname.setValue(e.getItem().map(Client::getSurname).orElse("Не доступно"));  
 editName.setValue(e.getItem().map(Client::getName).orElse("Не доступно"));  
 editPatronymic.setValue(e.getItem().map(Client::getPatronymic).orElse("Не доступно"));  
 });  
  
 styleTextField(editSurname);  
 styleTextField(editName);  
 styleTextField(editPatronymic);  
  
 Button close = new Button(new Icon("lumo", "cross"), (e) -> editForm.setVisible(false));  
  
 HorizontalLayout formHeader = new HorizontalLayout();  
 formHeader.getStyle().set("justify-content", "space-between");  
  
 H3 title = new H3("Изменить клиента");  
  
 formHeader.add(title, close);  
  
 editForm.add(formHeader, id, editSurname, editName, editPatronymic, editButton);  
 editForm.setVisible(false);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(addClient);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(JustifyContentMode.END);  
   
 Div content = new Div(grid, editForm);  
 content.addClassName(LumoUtility.Display.FLEX);  
 content.addClassName(LumoUtility.FlexDirection.COLUMN);  
 content.addClassName(LumoUtility.Gap.LARGE);  
 content.setSizeFull();  
  
 add(buttonLayout, content);  
 }  
  
 private void setupGrid(){  
 grid.addClassName("force-focus-outline");  
 grid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 grid.setHeightFull();  
  
 grid.addColumn(Client::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 grid.addColumn(Client::getSurname)  
 .setHeader("Фамилия")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Client::getName)  
 .setHeader("Имя")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Client::getPatronymic)  
 .setHeader("Отчество")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Client::getPhone)  
 .setHeader("Номер телефона")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 GridContextMenu<Client> contextMenu = grid.addContextMenu();  
  
 Button editMenuItem = new Button("Изменить", VaadinIcon.EDIT.create());  
 editMenuItem.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(editMenuItem, e->{  
 if (editForm.isVisible()) {  
 editForm.setVisible(false);  
 } else if (!editForm.isVisible()) {  
 editForm.setVisible(true);  
 id.setValue(String.valueOf(e.getItem().map(Client::getId).orElse(null)));  
 editSurname.setValue(e.getItem().map(Client::getSurname).orElse("Не доступно"));  
 editName.setValue(e.getItem().map(Client::getName).orElse("Не доступно"));  
 editPatronymic.setValue(e.getItem().map(Client::getPatronymic).orElse("Не доступно"));  
 }  
 });  
  
 if(currentUserService.getCurrentUserType().equals(UserType.ADMIN)) {  
 Button deleteMenuItem = new Button("Удалить", VaadinIcon.TRASH.create());  
 deleteMenuItem.addThemeVariants(ButtonVariant.LUMO\_ERROR, ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(deleteMenuItem, e -> {  
 Dialog delete = new Dialog();  
  
 Button confirmDelete = new Button("Удалить");  
 confirmDelete.addThemeVariants(ButtonVariant.LUMO\_ERROR);  
 Button cancelDelete = new Button("Отмена");  
 cancelDelete.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(cancelDelete, confirmDelete);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(JustifyContentMode.BETWEEN);  
 delete.add(buttonLayout);  
 delete.setHeaderTitle(String.format("Удалить клиента \"%s\"", e.getItem().map(Client::getName).orElse(null)));  
 delete.open();  
   
 confirmDelete.addClickListener(event -> {  
 clientService.delete(e.getItem().map(Client::getId).orElse(null));  
 updateGrid();  
 Notification.show("Клиент удален")  
 .setPosition(Notification.Position.TOP\_END);  
 delete.close();  
 });  
   
 cancelDelete.addClickListener(event -> {  
 delete.close();  
 });  
 });  
 }  
 }  
  
 private void updateGrid(){  
 List<Client> clients = clientService.findAll();  
 grid.setItems(clients);  
 }  
  
 private void styleTextField(TextField textField) {  
 textField.setWidthFull();  
 textField.addClassName(LumoUtility.Margin.Bottom.SMALL);  
 textField.getElement().getThemeList().add("small");  
 }  
  
 private void showError(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.denka88.ateliergrace.service.EmployeeService;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.dialog.Dialog;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.contextmenu.GridContextMenu;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H3;  
import com.vaadin.flow.component.icon.Icon;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.util.List;  
  
@PageTitle("Сотрудники")  
@Route(value = "employees-list", layout = MainLayout.class)  
@RolesAllowed("ADMIN")  
public class EmployeesView extends VerticalLayout {  
  
 private final EmployeeService employeeService;  
 private final Grid<Employee> grid;  
 private final CurrentUserService currentUserService;  
  
 private FormLayout editForm = new FormLayout();  
 private TextField id = new TextField("ID");  
 private TextField editSurname = new TextField("Фамилия");  
 private TextField editName = new TextField("Имя");  
 private TextField editPatronymic = new TextField("Отчество");  
 private TextField editPost = new TextField("Должность");  
 private Button editButton = new Button("Сохранить", VaadinIcon.CHECK.create());  
  
 Dialog delete = new Dialog();  
  
 public EmployeesView(EmployeeService employeeService, CurrentUserService currentUserService) {  
 this.employeeService = employeeService;  
 this.currentUserService = currentUserService;  
 this.grid = new Grid<>(Employee.class, false);  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
  
 setupGrid();  
 updateGrid();  
  
 Button addEmployee = new Button("Добавить сотрудника", VaadinIcon.PLUS.create());  
 addEmployee.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 addEmployee.addClassName(LumoUtility.Margin.Bottom.LARGE);  
 addEmployee.addClickListener(e -> {  
 UI.getCurrent().navigate("/employee-registration");  
 });  
  
 editForm.setWidth("400px");  
 editForm.addClassNames(  
 LumoUtility.Padding.LARGE,  
 LumoUtility.BorderRadius.LARGE,  
 LumoUtility.BoxShadow.SMALL,  
 LumoUtility.Background.BASE  
 );  
  
 id.setVisible(false);  
 editSurname.setRequired(true);  
 editName.setRequired(true);  
 editPatronymic.setRequired(true);  
 editPost.setRequired(true);  
  
 editButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 editButton.addClassName(LumoUtility.Margin.Top.MEDIUM);  
 editButton.getStyle().set("margin-left", "auto");  
  
 editButton.addClickListener(e->{  
 if (editSurname.isEmpty() || editName.isEmpty() || editPatronymic.isEmpty() || editPost.isEmpty()){  
 showError("Заполните все поля");  
 return;  
 }  
   
 Employee updateEmployee = employeeService.findById(Long.valueOf(id.getValue())).orElse(null);  
 if (updateEmployee != null) {  
 updateEmployee.setSurname(editSurname.getValue());  
 updateEmployee.setName(editName.getValue());  
 updateEmployee.setPatronymic(editPatronymic.getValue());  
 updateEmployee.setPost(editPost.getValue());  
 employeeService.update(updateEmployee);  
 updateGrid();  
 editForm.setVisible(false);  
 }  
 });  
  
 grid.addCellFocusListener(e->{  
 id.setValue(String.valueOf(e.getItem().map(Employee::getId).orElse(null)));  
 editSurname.setValue(e.getItem().map(Employee::getSurname).orElse("Не доступно"));  
 editName.setValue(e.getItem().map(Employee::getName).orElse("Не доступно"));  
 editPatronymic.setValue(e.getItem().map(Employee::getPatronymic).orElse("Не доступно"));  
 editPost.setValue(e.getItem().map(Employee::getPost).orElse("Не доступно"));  
 });  
  
 styleTextField(editSurname);  
 styleTextField(editName);  
 styleTextField(editPatronymic);  
 styleTextField(editPost);  
  
 Button close = new Button(new Icon("lumo", "cross"), (e) -> editForm.setVisible(false));  
  
 HorizontalLayout formHeader = new HorizontalLayout();  
 formHeader.getStyle().set("justify-content", "space-between");  
  
 H3 title = new H3("Изменить сотрудника");  
  
 formHeader.add(title, close);  
  
 editForm.add(formHeader, id, editSurname, editName, editPatronymic, editPost, editButton);  
 editForm.setVisible(false);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(addEmployee);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(FlexComponent.JustifyContentMode.END);  
  
 Div content = new Div(grid, editForm);  
 content.addClassName(LumoUtility.Display.FLEX);  
 content.addClassName(LumoUtility.FlexDirection.COLUMN);  
 content.addClassName(LumoUtility.Gap.LARGE);  
 content.setSizeFull();  
  
 add(buttonLayout, content);  
 }  
  
 private void setupGrid(){  
 grid.addClassName("force-focus-outline");  
 grid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 grid.setHeightFull();  
  
 grid.addColumn(Employee::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 grid.addColumn(Employee::getSurname)  
 .setHeader("Фамилия")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Employee::getName)  
 .setHeader("Имя")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Employee::getPatronymic)  
 .setHeader("Отчество")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Employee::getPost)  
 .setHeader("Должность")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 GridContextMenu<Employee> contextMenu = grid.addContextMenu();  
  
 Button editMenuItem = new Button("Изменить", VaadinIcon.EDIT.create());  
 editMenuItem.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(editMenuItem, e->{  
 if(editForm.isVisible()){  
 editForm.setVisible(false);  
 }else if(!editForm.isVisible()){  
 editForm.setVisible(true);  
 id.setValue(String.valueOf(e.getItem().map(Employee::getId).orElse(null)));  
 editSurname.setValue(e.getItem().map(Employee::getSurname).orElse("Не доступно"));  
 editName.setValue(e.getItem().map(Employee::getName).orElse("Не доступно"));  
 editPatronymic.setValue(e.getItem().map(Employee::getPatronymic).orElse("Не доступно"));  
 editPost.setValue(e.getItem().map(Employee::getPost).orElse("Не доступно"));  
 }  
 });  
  
 if(currentUserService.getCurrentUserType().equals(UserType.ADMIN)) {  
 Button deleteMenuItem = new Button("Удалить", VaadinIcon.TRASH.create());  
 deleteMenuItem.addThemeVariants(ButtonVariant.LUMO\_ERROR, ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(deleteMenuItem, e -> {  
 Dialog delete = new Dialog();  
  
 Button confirmDelete = new Button("Удалить");  
 confirmDelete.addThemeVariants(ButtonVariant.LUMO\_ERROR);  
 Button cancelDelete = new Button("Отмена");  
 cancelDelete.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(cancelDelete, confirmDelete);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(JustifyContentMode.BETWEEN);  
 delete.add(buttonLayout);  
 delete.setHeaderTitle(String.format("Удалить сотрудника \"%s\"", e.getItem().map(Employee::getName).orElse(null)));  
 delete.open();  
   
 confirmDelete.addClickListener(event -> {  
 employeeService.delete(e.getItem().map(Employee::getId).orElse(null));  
 updateGrid();  
 Notification.show("Сотрудник удален")  
 .setPosition(Notification.Position.TOP\_END);  
 delete.close();  
 });  
   
 cancelDelete.addClickListener(event -> {  
 delete.close();  
 });  
 });  
 }  
 }  
  
 private void updateGrid(){  
 List<Employee> employees = employeeService.findAll();  
 grid.setItems(employees);  
 }  
  
 private void styleTextField(TextField textField) {  
 textField.setWidthFull();  
 textField.addClassName(LumoUtility.Margin.Bottom.SMALL);  
 textField.getElement().getThemeList().add("small");  
 }  
  
 private void showError(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.service.AuthService;  
import com.denka88.ateliergrace.service.EmployeeService;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H2;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.PasswordField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
@PageTitle("Регистрация сотрудника | Грация")  
@Route(value = "employee-registration", layout = MainLayout.class)  
@RolesAllowed("ADMIN")  
public class EmplRegistrationView extends VerticalLayout {  
  
 private final AuthService authService;  
 private final EmployeeService employeeService;  
  
 public EmplRegistrationView(AuthService *authService*, EmployeeService *employeeService*) {  
 this.authService = *authService*;  
 this.employeeService = *employeeService*;  
  
 setPadding(false);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.*LARGE*);  
  
 Div *card* = new Div();  
 *card*.addClassNames(  
 LumoUtility.Padding.*XLARGE*,  
 LumoUtility.BorderRadius.*LARGE*,  
 LumoUtility.BoxShadow.*MEDIUM*,  
 LumoUtility.Background.*BASE*,  
 "employee-registration-card"  
 );  
 *card*.setWidth("600px");  
  
 H2 *title* = new H2("Регистрация сотрудника");  
 *title*.addClassNames(LumoUtility.Margin.Bottom.*LARGE*, LumoUtility.TextAlignment.*CENTER*);  
  
 TextField *login* = new TextField("Логин");  
 *login*.setWidthFull();  
 *login*.setRequired(true);  
 *login*.setPlaceholder("Введите логин сотрудника");  
 *login*.setMinLength(5);  
  
 PasswordField *password* = new PasswordField("Пароль");  
 *password*.setWidthFull();  
 *password*.setRequired(true);  
 *password*.setPlaceholder("Введите пароль");  
 *password*.setMinLength(8);  
  
 PasswordField *confirmPassword* = new PasswordField("Подтверждение пароля");  
 *confirmPassword*.setWidthFull();  
 *confirmPassword*.setRequired(true);  
 *confirmPassword*.setPlaceholder("Повторите пароль");  
  
 TextField *surname* = new TextField("Фамилия");  
 *surname*.setWidthFull();  
 *surname*.setRequired(true);  
 *surname*.setPlaceholder("Введите фамилию");  
  
 TextField *name* = new TextField("Имя");  
 *name*.setWidthFull();  
 *name*.setRequired(true);  
 *name*.setPlaceholder("Введите имя");  
  
 TextField *patronymic* = new TextField("Отчество");  
 *patronymic*.setWidthFull();  
 *patronymic*.setPlaceholder("Введите отчество (если есть)");  
  
 TextField *post* = new TextField("Должность");  
 *post*.setWidthFull();  
 *post*.setRequired(true);  
 *post*.setPlaceholder("Введите должность");  
  
 Button *registerButton* = new Button("Зарегистрировать", VaadinIcon.*USER\_CHECK*.create());  
 *registerButton*.addThemeVariants(ButtonVariant.*LUMO\_PRIMARY*);  
 *registerButton*.setWidthFull();  
 *registerButton*.addClassName(LumoUtility.Margin.Top.*MEDIUM*);  
  
 *registerButton*.addClickListener(*e* -> {  
 if (*login*.isEmpty() || *password*.isEmpty() || *confirmPassword*.isEmpty() ||  
 *name*.isEmpty() || *surname*.isEmpty() || *post*.isEmpty()) {  
 showError("Заполните все обязательные поля");  
 return;  
 }  
  
 if (!*password*.getValue().equals(*confirmPassword*.getValue())) {  
 showError("Пароли не совпадают");  
 return;  
 }  
   
 if (*login*.getValue().length() < 5){  
 showError("Неподходящая длина логина. Минимум 5 символов");  
 }  
  
 if(*password*.getValue().length() < 8){  
 showError("Неподходящая длина пароля. Минимум 8 символов");  
 return;  
 }  
  
 try {  
 Employee *employee* = new Employee();  
 *employee*.setSurname(*surname*.getValue());  
 *employee*.setName(*name*.getValue());  
 *employee*.setPatronymic(*patronymic*.getValue());  
 *employee*.setPost(*post*.getValue());  
  
 Employee *savedEmployee* = *employeeService*.save(*employee*);  
 *authService*.register(  
 *login*.getValue(),  
 *password*.getValue(),  
 UserType.*EMPLOYEE*,  
 *savedEmployee*.getId()  
 );  
  
 showSuccess("Сотрудник успешно зарегистрирован!");  
 UI.*getCurrent*().navigate("employees-list");  
 } catch (Exception *ex*) {  
 showError("Ошибка регистрации: " + *ex*.getMessage());  
 }  
 });  
  
 FormLayout *form* = new FormLayout();  
 *form*.addClassNames(LumoUtility.Padding.*NONE*);  
 *form*.add(  
 *title*,  
 *login*, *password*, *confirmPassword*,  
 *surname*, *name*, *patronymic*,  
 *post*,  
 *registerButton* );  
 *form*.setResponsiveSteps(  
 new FormLayout.ResponsiveStep("0", 1),  
 new FormLayout.ResponsiveStep("500px", 2)  
 );  
 *form*.setColspan(*title*, 2);  
 *form*.setColspan(*registerButton*, 2);  
  
 *card*.add(*form*);  
  
 Div *container* = new Div(*card*);  
 *container*.setSizeFull();  
 *container*.addClassNames(  
 LumoUtility.Display.*FLEX*,  
 LumoUtility.JustifyContent.*CENTER*,  
 LumoUtility.AlignItems.*CENTER* );  
  
 add(*container*);  
 }  
  
 private void showError(String *message*) {  
 Notification.*show*(*message*, 3000, Notification.Position.*TOP\_CENTER*)  
 .addThemeVariants(NotificationVariant.*LUMO\_ERROR*);  
 }  
  
 private void showSuccess(String *message*) {  
 Notification.*show*(*message*, 3000, Notification.Position.*TOP\_CENTER*)  
 .addThemeVariants(NotificationVariant.*LUMO\_SUCCESS*);  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.Main;  
import com.vaadin.flow.component.login.LoginForm;  
import com.vaadin.flow.component.login.LoginI18n;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.\*;  
import com.vaadin.flow.server.auth.AnonymousAllowed;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
  
@Route(value = "login", autoLayout = false)  
@PageTitle("Вход | Грация")  
@AnonymousAllowed  
public class LoginView extends Main implements BeforeEnterObserver {  
  
 private final LoginForm login;  
  
 public LoginView() {  
 addClassNames(  
 LumoUtility.Display.FLEX,  
 LumoUtility.JustifyContent.CENTER,  
 LumoUtility.AlignItems.CENTER,  
 LumoUtility.FlexDirection.COLUMN,  
 "login-view"  
 );  
 setSizeFull();  
 getStyle().set("background", "var(--lumo-contrast-5pct)");  
  
 Div card = new Div();  
 card.addClassNames(  
 LumoUtility.Padding.XLARGE,  
 LumoUtility.BorderRadius.LARGE,  
 LumoUtility.BoxShadow.MEDIUM,  
 LumoUtility.Background.BASE,  
 "login-card"  
 );  
 card.setWidth("400px");  
  
 login = new LoginForm();  
 login.setAction("login");  
  
 LoginI18n loginForm = LoginI18n.createDefault();  
 loginForm.getForm().setTitle("Войти");  
 loginForm.getForm().setUsername("Логин");  
 loginForm.getForm().setPassword("Пароль");  
 loginForm.getForm().setSubmit("Войти");  
 loginForm.getForm().setForgotPassword("Забыли пароль?");  
 loginForm.setErrorMessage(new LoginI18n.ErrorMessage());  
 loginForm.getErrorMessage().setTitle("Неверные данные");  
 loginForm.getErrorMessage().setMessage("Проверьте логин и пароль и попробуйте снова");  
 login.setI18n(loginForm);  
  
 RouterLink registerLink = new RouterLink("Зарегистрироваться", RegistrationView.class);  
 registerLink.addClassNames(LumoUtility.Margin.Top.MEDIUM);  
  
 VerticalLayout layout = new VerticalLayout(login, registerLink);  
 layout.setPadding(false);  
 layout.setSpacing(false);  
 layout.setAlignItems(FlexComponent.Alignment.CENTER);  
  
 card.add(layout);  
 add(card);  
 }  
  
 @Override  
 public void beforeEnter(BeforeEnterEvent event) {  
 if (event.getLocation()  
 .getQueryParameters()  
 .getParameters()  
 .containsKey("error")) {  
 login.setError(true);  
 }  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.vaadin.flow.component.html.H1;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.router.RouteAlias;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.PermitAll;  
  
@Route(value = "main", layout = MainLayout.class)  
@RouteAlias(value = "index", layout = MainLayout.class)  
@RouteAlias(value = "", layout = MainLayout.class)  
@PageTitle("MAIN")  
@PermitAll  
public class MainView extends VerticalLayout {  
   
 private final CurrentUserService currentUserService;  
  
 public MainView(CurrentUserService currentUserService) {  
 this.currentUserService = currentUserService;  
  
 H1 title = new H1();  
  
 if (currentUserService.getCurrentUserType() == UserType.EMPLOYEE) {  
 currentUserService.getCurrentEmployee().ifPresent(employee -> {  
 String surname = employee.getSurname();  
 String name = employee.getName();  
 String patronymic = employee.getPatronymic();  
  
 String fio = surname + " " + name + " " + patronymic;  
  
 title.setText(String.format("Здравствуйте, %s!", fio));  
 });  
 }  
   
 if (currentUserService.getCurrentUserType() == UserType.CLIENT) {  
 currentUserService.getCurrentClient().ifPresent(client -> {  
 String surname = client.getSurname();  
 String name = client.getName();  
 String patronymic = client.getPatronymic();  
  
 String fio = surname + " " + name + " " + patronymic;  
  
 title.setText(String.format("Здравствуйте, %s!", fio));  
 });  
 }  
  
 title.getStyle().set("margin", "auto");  
 add(title);  
 }  
   
   
   
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.Material;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.denka88.ateliergrace.service.MaterialService;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.dialog.Dialog;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.contextmenu.GridContextMenu;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H3;  
import com.vaadin.flow.component.icon.Icon;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.popover.Popover;  
import com.vaadin.flow.component.textfield.IntegerField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.util.List;  
  
@PageTitle("Материалы")  
@Route(value = "materials-list", layout = MainLayout.class)  
@RolesAllowed({"ADMIN", "EMPLOYEE"})  
public class MaterialsView extends VerticalLayout {  
  
 private final MaterialService materialService;  
 private final Grid<Material> grid;  
 private final CurrentUserService currentUserService;  
  
 private FormLayout editForm = new FormLayout();  
 private TextField id = new TextField("ID");  
 private TextField editName = new TextField("Название");  
 private IntegerField editValue = new IntegerField("Кол-во");  
 private Button editButton = new Button("Сохранить", VaadinIcon.CHECK.create());  
  
 public MaterialsView(MaterialService materialService, CurrentUserService currentUserService) {  
 this.materialService = materialService;  
 this.currentUserService = currentUserService;  
 this.grid = new Grid<>(Material.class, false);  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
  
 setupGrid();  
 updateGrid();  
  
 Button addButton = new Button("Добавить материал", VaadinIcon.PLUS.create());  
 addButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 addButton.addClassName(LumoUtility.Margin.Bottom.LARGE);  
  
 Popover addMaterial = new Popover();  
 addMaterial.setModal(true);  
 addMaterial.setBackdropVisible(true);  
 addMaterial.setWidth("300px");  
 addMaterial.addClassName(LumoUtility.Padding.LARGE);  
  
 TextField name = new TextField("Название");  
 name.setWidthFull();  
  
 IntegerField value = new IntegerField("На складе");  
 value.setWidthFull();  
 value.setMin(1);  
 value.setMax(999999);  
 value.setStepButtonsVisible(true);  
  
 Button post = new Button("Добавить", VaadinIcon.CHECK.create());  
 post.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 post.setWidthFull();  
  
 post.addClickListener(e -> {  
 if (name.isEmpty() || value.isEmpty()) {  
 return;  
 }  
 Material material = new Material();  
 material.setName(name.getValue());  
 material.setValue(value.getValue());  
 materialService.save(material);  
 updateGrid();  
 name.clear();  
 value.clear();  
 });  
  
 FormLayout popoverForm = new FormLayout(name, value, post);  
 popoverForm.setWidth("400px");  
 addMaterial.add(popoverForm);  
 addMaterial.setTarget(addButton);  
  
 editForm.setWidth("400px");  
 editForm.addClassNames(  
 LumoUtility.Padding.LARGE,  
 LumoUtility.BorderRadius.LARGE,  
 LumoUtility.BoxShadow.SMALL,  
 LumoUtility.Background.BASE  
 );  
  
 id.setVisible(false);  
   
 editName.setMinLength(3);  
 editName.setRequired(true);  
  
 editValue.setStepButtonsVisible(true);  
 editValue.setMin(1);  
 editValue.setMax(999999);  
  
 editButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 editButton.addClassName(LumoUtility.Margin.Top.MEDIUM);  
 editButton.getStyle().set("margin-left", "auto");  
  
 editButton.addClickListener(e->{  
 if(editName.isEmpty()){  
 showError("Заполните все поля");  
 return;  
 }  
   
 if(editName.getValue().length() < 3){  
 showError("Неподходящая длина названия. Минимум 3 символа");  
 return;  
 }  
   
 Material updateMaterial = materialService.findById(Long.valueOf(id.getValue())).orElse(null);  
 if (updateMaterial != null) {  
 updateMaterial.setName(editName.getValue());  
 updateMaterial.setValue(editValue.getValue());  
 materialService.update(updateMaterial);  
 updateGrid();  
 editForm.setVisible(false);  
 Notification.show("Материал обновлен")  
 .setPosition(Notification.Position.TOP\_END);  
 }  
 });  
  
 grid.addCellFocusListener(e->{  
 id.setValue(String.valueOf(e.getItem().map(Material::getId).orElse(null)));  
 editName.setValue(e.getItem().map(Material::getName).orElse(null));  
 editValue.setValue(e.getItem().map(Material::getValue).orElse(null));  
 });  
  
 styleTextField(editName);  
 styleIntegerField(editValue);  
  
 Button close = new Button(new Icon("lumo", "cross"), (e) -> editForm.setVisible(false));  
  
 HorizontalLayout formHeader = new HorizontalLayout();  
 formHeader.getStyle().set("justify-content", "space-between");  
  
 H3 title = new H3("Изменить материал");  
  
 formHeader.add(title, close);  
  
 editForm.add(formHeader, id, editName, editValue, editButton);  
 editForm.setVisible(false);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(addButton);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(FlexComponent.JustifyContentMode.END);  
  
 Div content = new Div(grid, editForm);  
 content.addClassName(LumoUtility.Display.FLEX);  
 content.addClassName(LumoUtility.FlexDirection.COLUMN);  
 content.addClassName(LumoUtility.Gap.LARGE);  
 content.setSizeFull();  
  
 add(buttonLayout, content, addMaterial);  
 }  
  
 private void setupGrid(){  
 grid.addClassName("force-focus-outline");  
 grid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 grid.setHeightFull();  
  
 grid.addColumn(Material::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 grid.addColumn(Material::getName)  
 .setHeader("Название")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Material::getValue)  
 .setHeader("На складе")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 GridContextMenu<Material> contextMenu = grid.addContextMenu();  
  
 Button editMenuItem = new Button("Изменить", VaadinIcon.EDIT.create());  
 editMenuItem.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(editMenuItem, e->{  
 if(editForm.isVisible()){  
 editForm.setVisible(false);  
 }else if(!editForm.isVisible()){  
 editForm.setVisible(true);  
 id.setValue(String.valueOf(e.getItem().map(Material::getId).orElse(null)));  
 editName.setValue(e.getItem().map(Material::getName).orElse(null));  
 editValue.setValue(e.getItem().map(Material::getValue).orElse(null));  
 }  
 });  
  
   
   
 if(currentUserService.getCurrentUserType().equals(UserType.ADMIN)) {  
 Button deleteMenuItem = new Button("Удалить", VaadinIcon.TRASH.create());  
 deleteMenuItem.addThemeVariants(ButtonVariant.LUMO\_ERROR, ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(deleteMenuItem, e -> {  
 Dialog delete = new Dialog();  
   
 Button confirmDelete = new Button("Удалить");  
 confirmDelete.addThemeVariants(ButtonVariant.LUMO\_ERROR);  
 Button cancelDelete = new Button("Отмена");  
 cancelDelete.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(cancelDelete, confirmDelete);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(JustifyContentMode.BETWEEN);  
 delete.add(buttonLayout);  
 delete.setHeaderTitle(String.format("Удалить материал \"%s\"", e.getItem().map(Material::getName).orElse(null)));  
 delete.open();  
 confirmDelete.addClickListener(event -> {  
 materialService.delete(e.getItem().map(Material::getId).orElse(null));  
 updateGrid();  
 Notification.show("Материал удален")  
 .setPosition(Notification.Position.TOP\_END);  
 delete.close();  
 });  
   
 cancelDelete.addClickListener(event -> {  
 delete.close();  
 });  
 updateGrid();  
 });  
 }  
 }  
  
 private void updateGrid(){  
 List<Material> materials = materialService.findAll();  
 grid.setItems(materials);  
 }  
  
 private void styleTextField(TextField textField) {  
 textField.setWidthFull();  
 textField.addClassName(LumoUtility.Margin.Bottom.SMALL);  
 textField.getElement().getThemeList().add("small");  
 }  
  
 private void styleIntegerField(IntegerField integerField) {  
 integerField.setWidthFull();  
 integerField.addClassName(LumoUtility.Margin.Bottom.SMALL);  
 integerField.getElement().getThemeList().add("small");  
 }  
  
 private void showError(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.\*;  
import com.denka88.ateliergrace.service.\*;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.util.List;  
import java.util.stream.Collectors;  
  
@Route(value = "my-orders", layout = MainLayout.class)  
@PageTitle("Мои заказы")  
@RolesAllowed("CLIENT")  
public class MyOrdersView extends VerticalLayout {  
  
 private final OrderService orderService;  
 private final CurrentUserService currentUserService;  
 private final Grid<Order> grid;  
  
 public MyOrdersView(OrderService orderService, CurrentUserService currentUserService) {  
 this.orderService = orderService;  
 this.currentUserService = currentUserService;  
 this.grid = new Grid<>(Order.class, false);  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
  
 setupGrid();  
 updateGrid();  
  
 Div content = new Div(grid);  
 content.setSizeFull();  
 add(content);  
 }  
  
 private void setupGrid(){  
 grid.addClassName("force-focus-outline");  
 grid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 grid.setHeightFull();  
  
 grid.addColumn(Order::getId)  
 .setHeader("Номер заказа")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 grid.addColumn(Order::getOrderName)  
 .setHeader("Название")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Order::getDescription)  
 .setHeader("Описание заказа")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Order::getOrderDate)  
 .setHeader("Дата создания")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(order -> {  
 if (order.getOrderEmployees() == null || order.getOrderEmployees().isEmpty()) {  
 return "Не назначено";  
 }  
 return order.getOrderEmployees().stream()  
 .map(oe -> oe.getDateOfReady() != null ?  
 oe.getDateOfReady().toString() : "Не указана")  
 .collect(Collectors.joining(", "));  
 }).setHeader("Дата готовности")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Order::getCost)  
 .setHeader("Стоимость")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Order::getStatus)  
 .setHeader("Статус")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(order -> {  
 if(order.getOrderEmployees() == null || order.getOrderEmployees().isEmpty()){  
 return "Не назначены";  
 }  
 return order.getOrderEmployees().stream()  
 .map(OrderEmployee::getEmployee)  
 .map(Employee::toString)  
 .collect(Collectors.joining(", "));  
 }).setHeader("Сотрудники")  
 .setAutoWidth(true);  
  
 grid.addColumn(order -> {  
 if (order.getMaterials() == null || order.getMaterials().isEmpty()) {  
 return "Не назначены";  
 }  
 return order.getMaterials().stream()  
 .map(Material::getName)  
 .collect(Collectors.joining(", "));  
 }).setHeader("Материалы")  
 .setAutoWidth(true);  
 }  
  
 private void updateGrid(){  
 currentUserService.getCurrentClient().ifPresent(client -> {  
 List<Order> orders = orderService.findByClientId(client.getId());  
 grid.setItems(orders);  
 });  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.\*;  
import com.denka88.ateliergrace.service.\*;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.combobox.MultiSelectComboBox;  
import com.vaadin.flow.component.contextmenu.ContextMenu;  
import com.vaadin.flow.component.contextmenu.MenuItem;  
import com.vaadin.flow.component.dialog.Dialog;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.contextmenu.GridContextMenu;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H3;  
import com.vaadin.flow.component.icon.Icon;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.popover.Popover;  
import com.vaadin.flow.component.textfield.NumberField;  
import com.vaadin.flow.component.textfield.TextArea;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.data.renderer.ComponentRenderer;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
import com.vaadin.flow.component.combobox.ComboBox;  
import com.vaadin.flow.data.provider.ListDataProvider;  
  
import java.util.HashSet;  
import java.util.List;  
import java.util.Set;  
import java.util.stream.Collectors;  
  
@Route(value = "orders-list", layout = MainLayout.class)  
@PageTitle("Заказы")  
@RolesAllowed({"ADMIN", "EMPLOYEE"})  
public class OrdersView extends VerticalLayout {  
  
 private final OrderService orderService;  
 private final MaterialService materialService;  
 private final CurrentUserService currentUserService;  
 private final EmployeeService employeeService;  
 private final Grid<Order> grid;  
  
 private FormLayout editForm = new FormLayout();  
 private TextField id = new TextField("ID");  
 private TextField editOrderName = new TextField("Название заказа");  
 private TextField editType = new TextField("Описание");  
 private MultiSelectComboBox<Material> editMaterials = new MultiSelectComboBox<>("Материалы");  
 private Button editButton = new Button("Сохранить", VaadinIcon.CHECK.create());  
  
 private final TextArea detailsText = new TextArea();  
 private final TextArea clientDetails = new TextArea();  
  
 public OrdersView(OrderService orderService, MaterialService materialService,  
 CurrentUserService currentUserService, EmployeeService employeeService) {  
 this.orderService = orderService;  
 this.materialService = materialService;  
 this.currentUserService = currentUserService;  
 this.employeeService = employeeService;  
 this.grid = new Grid<>(Order.class, false);  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
  
 Button filterButton = new Button("Фильтры", VaadinIcon.FILTER.create());  
 filterButton.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
  
 Popover filterPopover = new Popover();  
 filterPopover.setWidth("300px");  
  
 FormLayout filterForm = new FormLayout();  
 filterForm.setWidthFull();  
  
 ComboBox<Status> statusFilter = new ComboBox<>("Статус заказа");  
 statusFilter.setItems(Status.values());  
 statusFilter.setItemLabelGenerator(status -> {  
 switch (status) {  
 case PROGRESS: return "Выполняется";  
 case COMPLETED: return "Выполнен";  
 default: return status.toString();  
 }  
 });  
 statusFilter.setClearButtonVisible(true);  
  
 MultiSelectComboBox<Material> materialsFilter = new MultiSelectComboBox<>("Материалы в заказе");  
 materialsFilter.setItems(materialService.findAll());  
 materialsFilter.setItemLabelGenerator(Material::getName);  
 materialsFilter.setClearButtonVisible(true);  
  
 ComboBox<Employee> employeeFilter = new ComboBox<>("Сотрудник");  
 employeeFilter.setItems(employeeService.findAll());  
 employeeFilter.setItemLabelGenerator(employee ->  
 String.format("%s %s", employee.getSurname(), employee.getName()));  
 employeeFilter.setClearButtonVisible(true);  
  
 Button applyFilters = new Button("Применить", VaadinIcon.CHECK.create());  
 applyFilters.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
  
 Button resetFilters = new Button("Сбросить", VaadinIcon.TRASH.create());  
 resetFilters.addThemeVariants(ButtonVariant.LUMO\_ERROR, ButtonVariant.LUMO\_TERTIARY);  
  
 HorizontalLayout buttonsLayout = new HorizontalLayout(applyFilters, resetFilters);  
 buttonsLayout.setWidthFull();  
 buttonsLayout.setJustifyContentMode(FlexComponent.JustifyContentMode.BETWEEN);  
  
 filterForm.add(statusFilter, materialsFilter, employeeFilter, buttonsLayout);  
 filterPopover.add(filterForm);  
 filterPopover.setTarget(filterButton);  
  
 applyFilters.addClickListener(e -> {  
 List<Order> allOrders = orderService.findAll();  
 ListDataProvider<Order> dataProvider = new ListDataProvider<>(allOrders);  
  
 dataProvider.addFilter(order -> {  
 boolean statusMatch = statusFilter.getValue() == null ||  
 order.getStatus() == statusFilter.getValue();  
  
 boolean materialsMatch = materialsFilter.getSelectedItems().isEmpty();  
  
 if (!materialsMatch) {  
 Set<Long> selectedMaterialIds = materialsFilter.getSelectedItems()  
 .stream()  
 .map(Material::getId)  
 .collect(Collectors.toSet());  
  
 materialsMatch = order.getMaterials().stream()  
 .map(Material::getId)  
 .anyMatch(selectedMaterialIds::contains);  
 }  
 boolean employeeMatch = employeeFilter.getValue() == null;  
 if (!employeeMatch) {  
 Long selectedEmployeeId = employeeFilter.getValue().getId();  
 employeeMatch = order.getOrderEmployees().stream()  
 .anyMatch(oe -> oe.getEmployee().getId().equals(selectedEmployeeId));  
 }  
  
 return statusMatch && materialsMatch && employeeMatch;  
 });  
  
 grid.setDataProvider(dataProvider);  
 filterPopover.close();  
 });  
  
 resetFilters.addClickListener(e -> {  
 statusFilter.clear();  
 materialsFilter.clear();  
 employeeFilter.clear();  
 grid.setItems(orderService.findAll());  
 filterPopover.close();  
 });  
  
 Button addOrder = new Button("Создать заказ", VaadinIcon.PLUS.create());  
 addOrder.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 addOrder.addClassName(LumoUtility.Margin.Bottom.LARGE);  
 addOrder.addClickListener(e->{  
 UI.getCurrent().navigate("/add-order");  
 });  
   
 HorizontalLayout toolbar = new HorizontalLayout(filterButton, addOrder);  
 toolbar.setWidthFull();  
 toolbar.setJustifyContentMode(FlexComponent.JustifyContentMode.END);  
  
 detailsText.setLabel("Данные о заказе");  
 detailsText.setWidthFull();  
 detailsText.setReadOnly(true);   
 detailsText.setVisible(false);  
   
 clientDetails.setLabel("Данные о заказчике");  
 clientDetails.setWidthFull();  
 clientDetails.setReadOnly(true);  
 clientDetails.setVisible(false);  
  
  
 ContextMenu closeDetailsOrder = new ContextMenu(detailsText);  
 MenuItem closeOrder = closeDetailsOrder.addItem("Закрыть");  
 closeOrder.setDisableOnClick(true);  
 closeOrder.addClickListener(e->{  
 detailsText.setVisible(false);  
 closeOrder.setEnabled(true);  
 });  
 ContextMenu closeDetailsClient = new ContextMenu(clientDetails);  
 MenuItem closeClient = closeDetailsClient.addItem("Закрыть");  
 closeClient.setDisableOnClick(true);  
 closeClient.addClickListener(e->{  
 clientDetails.setVisible(false);  
 closeClient.setEnabled(true);  
 });  
  
 setupGrid();  
 updateGrid();  
  
 editForm.setWidth("600px");  
 editForm.addClassNames(  
 LumoUtility.Padding.LARGE,  
 LumoUtility.BorderRadius.LARGE,  
 LumoUtility.BoxShadow.SMALL,  
 LumoUtility.Background.BASE  
 );  
 editForm.setResponsiveSteps(  
 new FormLayout.ResponsiveStep("0", 1)  
 );  
  
 id.setVisible(false);  
   
 editOrderName.setMinLength(3);  
 editOrderName.setRequired(true);  
   
 editType.setMinLength(3);  
 editType.setRequired(true);  
  
 editMaterials.setItems(materialService.findAll());  
 editMaterials.setItemLabelGenerator(Material::getName);  
 editMaterials.setWidthFull();  
  
 editButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 editButton.addClassName(LumoUtility.Margin.Top.MEDIUM);  
 editButton.getStyle().set("margin-left", "auto");  
  
 editButton.addClickListener(e -> {  
 if (editOrderName.isEmpty() || editMaterials.isEmpty() || editType.isEmpty()) {  
 showError("Заполните все поля");  
 return;  
 }  
   
 if (editOrderName.getValue().length() < 3 || editType.getValue().length() < 3){  
 showError("Неподходящая длина. Минимум 3 символа");  
 return;  
 }  
   
   
 Order updateOrder = orderService.findById(Long.valueOf(id.getValue())).orElse(null);  
 if (updateOrder != null) {  
 updateOrder.setOrderName(editOrderName.getValue());  
 updateOrder.setDescription(editType.getValue());  
 updateOrder.setMaterials(new HashSet<>(editMaterials.getSelectedItems()));  
 orderService.update(updateOrder);  
 updateGrid();  
 editForm.setVisible(false);  
 Notification.show("Заказ обновлен")  
 .setPosition(Notification.Position.TOP\_END);  
 }  
 });  
  
 grid.addCellFocusListener(e -> {  
 id.setValue(String.valueOf(e.getItem().map(Order::getId).orElse(null)));  
 editOrderName.setValue(e.getItem().map(Order::getOrderName).orElse("Не доступно"));  
 editType.setValue(e.getItem().map(Order::getDescription).orElse("Не доступно"));  
 editMaterials.clear();  
 if (e.getItem().isPresent()) {  
 editMaterials.select(e.getItem().get().getMaterials());  
 }  
  
 String detailsId = String.valueOf(e.getItem().map(Order::getId).orElse(null));  
 String detailsClient = String.valueOf(e.getItem().map(Order::getClient).orElse(null));  
 String detailsOrderName = String.valueOf(e.getItem().map(Order::getOrderName).orElse(null));  
 String detailsDescription = String.valueOf(e.getItem().map(Order::getDescription).orElse(null));  
 String detailsOrderDate = String.valueOf(e.getItem().map(Order::getOrderDate).orElse(null));  
 String detailsCost = String.valueOf(e.getItem().map(Order::getCost).orElse(null));  
 String detailsStatus = String.valueOf(e.getItem().map(Order::getStatus).orElse(null));  
 String detailsEmployees = String.valueOf(e.getItem().map(Order::getOrderEmployees).orElse(null));  
 String detailsMaterials = String.valueOf(e.getItem().map(Order::getMaterials).orElse(null));  
  
 String id = String.valueOf(e.getItem().map(Order::getClient).map(Client::getId).orElse(null));  
 String surname = e.getItem().map(Order::getClient).map(Client::getSurname).orElse(null);  
 String name = e.getItem().map(Order::getClient).map(Client::getName).orElse(null);  
 String patronymic = e.getItem().map(Order::getClient).map(Client::getPatronymic).orElse(null);  
 String phone = e.getItem().map(Order::getClient).map(Client::getPhone).orElse(null);  
   
 String clientDetailsContainer = String.format("ID: %s%nФамилия: %s%nИмя: %s%nОтчество: %s%nНомер телефона: %s%n",   
 id, surname, name, patronymic, phone);  
 String areaContainer = String.format("ID: %s%nЗаказчик: %s%nНазвание заказа: %s%nОписание: %s%nДата создания: %s%n" +  
 "Цена: %s%nСтатус заказа: %s%nСотрудники: %s%nМатериалы: %s%n", detailsId, detailsClient, detailsOrderName, detailsDescription, detailsOrderDate, detailsCost, detailsStatus, detailsEmployees, detailsMaterials);  
   
 clientDetails.setValue(clientDetailsContainer);  
 detailsText.setValue(areaContainer);  
 });  
  
 styleTextField(editOrderName);  
 styleTextField(editType);  
  
 Button close = new Button(new Icon("lumo", "cross"), (e) -> editForm.setVisible(false));  
  
 HorizontalLayout formHeader = new HorizontalLayout();  
 formHeader.getStyle().set("justify-content", "space-between");  
  
 H3 title = new H3("Изменить заказ");  
  
 formHeader.add(title, close);  
  
 editForm.add(formHeader, id, editOrderName, editType, editMaterials, editButton);  
 editForm.setVisible(false);  
   
   
 HorizontalLayout formAndDetails = new HorizontalLayout(editForm, detailsText, clientDetails);  
 formAndDetails.setWidthFull();  
 formAndDetails.setJustifyContentMode(FlexComponent.JustifyContentMode.END);  
  
 Div content = new Div(grid, formAndDetails);  
 content.addClassName(LumoUtility.Display.FLEX);  
 content.addClassName(LumoUtility.FlexDirection.COLUMN);  
 content.addClassName(LumoUtility.Gap.LARGE);  
 content.setSizeFull();  
  
 add(toolbar, content);  
 }  
  
 private void setupGrid(){  
 grid.addClassName("force-focus-outline");  
 grid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 grid.setHeightFull();  
  
 grid.addColumn(Order::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 grid.addColumn(Order::getClient)  
 .setHeader("Клиент")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Order::getOrderName)  
 .setHeader("Название")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Order::getDescription)  
 .setHeader("Описание заказа")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Order::getOrderDate)  
 .setHeader("Дата создания")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(order -> {  
 if (order.getOrderEmployees() == null || order.getOrderEmployees().isEmpty()) {  
 return "Не назначено";  
 }  
 return order.getOrderEmployees().stream()  
 .filter(oe -> oe.getDateOfReady() != null)  
 .findFirst()  
 .map(oe -> oe.getDateOfReady().toString())  
 .orElse("Не указана");  
 }).setHeader("Дата готовности").setSortable(true).setAutoWidth(true);  
  
 grid.addColumn(Order::getCost)  
 .setHeader("Стоимость")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Order::getStatus)  
 .setHeader("Статус")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(order -> {  
 if(order.getOrderEmployees() == null || order.getOrderEmployees().isEmpty()){  
 return "Не назначены";  
 }  
 return order.getOrderEmployees().stream()  
 .map(OrderEmployee::getEmployee)  
 .map(Employee::toString)  
 .collect(Collectors.joining(", "));  
 }).setHeader("Сотрудники");  
  
 grid.addColumn(order -> {  
 if (order.getMaterials() == null || order.getMaterials().isEmpty()) {  
 return "Не назначены";  
 }  
 return order.getMaterials().stream()  
 .map(Material::getName)  
 .collect(Collectors.joining(", "));  
 }).setHeader("Материалы");  
   
 GridContextMenu<Order> contextMenu = grid.addContextMenu();  
  
 Button openDetails = new Button("Сведения о заказе", VaadinIcon.CHECK.create());  
 openDetails.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
  
 contextMenu.addItem(openDetails, e->{  
 detailsText.setVisible(!detailsText.isVisible());  
 });  
  
 Button openClientDetails = new Button("Сведения о заказчике", VaadinIcon.CHECK.create());  
 openClientDetails.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
   
 contextMenu.addItem(openClientDetails, e->{  
 clientDetails.setVisible(!clientDetails.isVisible());   
 });  
  
 Button editMenuItem = new Button("Изменить", VaadinIcon.EDIT.create());  
 editMenuItem.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
   
 contextMenu.addItem(editMenuItem, e -> {  
 if (editForm.isVisible()) {  
 editForm.setVisible(false);  
 } else {  
 editForm.setVisible(true);  
 id.setValue(String.valueOf(e.getItem().map(Order::getId).orElse(null)));  
 editOrderName.setValue(e.getItem().map(Order::getOrderName).orElse(null));  
 editType.setValue(e.getItem().map(Order::getDescription).orElse(null));  
 editMaterials.clear();  
 if (e.getItem().isPresent()) {  
 editMaterials.select(e.getItem().get().getMaterials());  
 }  
 }  
 });  
   
 if(currentUserService.getCurrentUserType().equals(UserType.ADMIN)){  
 Button deleteMenuItem = new Button("Удалить", VaadinIcon.TRASH.create());  
 deleteMenuItem.addThemeVariants(ButtonVariant.LUMO\_ERROR, ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(deleteMenuItem, e->{  
 Dialog delete = new Dialog();  
  
 Button confirmDelete = new Button("Удалить");  
 confirmDelete.addThemeVariants(ButtonVariant.LUMO\_ERROR);  
 Button cancelDelete = new Button("Отмена");  
 cancelDelete.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(cancelDelete, confirmDelete);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(JustifyContentMode.BETWEEN);  
 delete.add(buttonLayout);  
 delete.setHeaderTitle(String.format("Удалить заказ \"%s\"", e.getItem().map(Order::getOrderName).orElse(null)));  
 delete.open();  
   
 confirmDelete.addClickListener(event -> {  
 orderService.delete(e.getItem().map(Order::getId).orElse(null));  
 updateGrid();  
 Notification.show("Заказ удален")  
 .setPosition(Notification.Position.TOP\_END);  
 delete.close();  
 });  
   
 cancelDelete.addClickListener(event -> {  
 delete.close();  
 });  
 });  
 }  
  
 if(currentUserService.getCurrentUserType().equals(UserType.EMPLOYEE)) {  
 grid.addColumn(new ComponentRenderer<>(order -> {  
 VerticalLayout actions = new VerticalLayout();  
 actions.setSpacing(false);  
 actions.setPadding(false);  
 actions.setWidthFull();  
  
 Popover setCost = new Popover();  
 setCost.setWidth("300px");  
 setCost.addClassName(LumoUtility.Padding.LARGE);  
 setCost.setModal(true);  
 setCost.setBackdropVisible(true);  
  
 NumberField cost = new NumberField("Стоимость заказа");  
 cost.setWidthFull();  
 cost.setMin(0);  
   
 Button completeBtn = new Button("Завершить", VaadinIcon.CHECK\_CIRCLE.create());  
 completeBtn.addThemeVariants(ButtonVariant.LUMO\_SUCCESS);  
 completeBtn.setWidthFull();  
 completeBtn.setEnabled(order.getStatus() != Status.COMPLETED);  
 completeBtn.addClickListener(e -> {  
 orderService.completeOrder(order.getId());  
 updateGrid();  
 Notification.show("Заказ #" + order.getId() + " завершен")  
 .setPosition(Notification.Position.TOP\_END);  
 });  
  
 actions.add(completeBtn);  
 return actions;  
 })).setHeader("Действия").setAutoWidth(true).setFlexGrow(0);  
 }  
 }  
  
 private void updateGrid(){  
 List<Order> orders = orderService.findAll();  
 grid.setItems(orders);  
 }  
  
 private void styleTextField(TextField textField) {  
 textField.setWidthFull();  
 textField.addClassName(LumoUtility.Margin.Bottom.SMALL);  
 textField.getElement().getThemeList().add("small");  
 }  
  
 private void showError(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
   
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.\*;  
import com.denka88.ateliergrace.service.\*;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.combobox.ComboBox;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.popover.Popover;  
import com.vaadin.flow.component.select.Select;  
import com.vaadin.flow.component.textfield.IntegerField;  
import com.vaadin.flow.component.textfield.NumberField;  
import com.vaadin.flow.data.provider.ListDataProvider;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.util.List;  
import java.util.Objects;  
  
@Route(value = "supplies", layout = MainLayout.class)  
@PageTitle("Поставки")  
@RolesAllowed({"ADMIN", "EMPLOYEE"})  
public class OrganizationMaterialView extends VerticalLayout {  
  
 private final OrganizationMaterialService organizationMaterialService;  
 private final OrganizationService organizationService;  
 private final MaterialService materialService;  
 private final Grid<OrganizationMaterial> grid;  
  
 public OrganizationMaterialView(OrganizationMaterialService organizationMaterialService,  
 OrganizationService organizationService,  
 MaterialService materialService) {  
 this.organizationMaterialService = organizationMaterialService;  
 this.organizationService = organizationService;  
 this.materialService = materialService;  
 this.grid = new Grid<>(OrganizationMaterial.class, false);  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
   
 Button filterButton = new Button("Фильтры", VaadinIcon.FILTER.create());  
 filterButton.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
   
 Popover filterPopover = new Popover();  
 filterPopover.setWidth("300px");  
   
 FormLayout filterForm = new FormLayout();  
 filterForm.setWidthFull();  
  
 ComboBox<Organization> organizationFilter = new ComboBox<>("Поставщик");  
 organizationFilter.setItems(organizationService.findAll());  
 organizationFilter.setItemLabelGenerator(organization -> String.format("%s", organization.getName()));  
 organizationFilter.setClearButtonVisible(true);  
   
 ComboBox<Material> materialFilter = new ComboBox<>("Material");  
 materialFilter.setItems(materialService.findAll());  
 materialFilter.setItemLabelGenerator(organization -> String.format("%s", organization.getName()));  
 materialFilter.setClearButtonVisible(true);  
   
 Button applyFilters = new Button("Применить", VaadinIcon.CHECK.create());  
 applyFilters.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
  
 Button resetFilters = new Button("Сбросить", VaadinIcon.TRASH.create());  
 resetFilters.addThemeVariants(ButtonVariant.LUMO\_ERROR, ButtonVariant.LUMO\_TERTIARY);  
  
 HorizontalLayout buttonsLayout = new HorizontalLayout(applyFilters, resetFilters);  
 buttonsLayout.setWidthFull();  
 buttonsLayout.setJustifyContentMode(FlexComponent.JustifyContentMode.BETWEEN);  
   
 filterForm.add(organizationFilter, materialFilter, buttonsLayout);  
 filterPopover.add(filterForm);  
 filterPopover.setTarget(filterButton);  
   
 applyFilters.addClickListener(e->{  
 List<OrganizationMaterial> allSupplies = organizationMaterialService.findAll();  
 ListDataProvider<OrganizationMaterial> dataProvider = new ListDataProvider<>(allSupplies);  
   
 dataProvider.addFilter(supplied -> {  
 boolean organizationMatch = organizationFilter.getValue() == null;  
 if (!organizationMatch) {  
 organizationMatch = Objects.equals(  
 supplied.getOrganization().getId(),  
 organizationFilter.getValue().getId()  
 );  
 }  
  
 boolean materialMatch = materialFilter.getValue() == null;  
 if(!materialMatch) {  
 materialMatch = Objects.equals(  
 supplied.getMaterial().getId(),  
 materialFilter.getValue().getId()  
 );  
 }  
   
 return organizationMatch && materialMatch;  
 });  
   
 grid.setDataProvider(dataProvider);  
 filterPopover.close();  
 });  
   
 resetFilters.addClickListener(e->{  
 organizationFilter.clear();  
 materialFilter.clear();  
 grid.setItems(organizationMaterialService.findAll());  
 filterPopover.close();  
 });  
  
 Button addButton = new Button("Добавить поставку", VaadinIcon.PLUS.create());  
 addButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 addButton.addClassName(LumoUtility.Margin.Bottom.LARGE);  
   
 HorizontalLayout toolbar = new HorizontalLayout(filterButton, addButton);  
 toolbar.setWidthFull();  
 toolbar.setJustifyContentMode(FlexComponent.JustifyContentMode.END);  
  
 setupGrid();  
 updateGrid();  
  
 Popover addSupplied = new Popover();  
 addSupplied.setModal(true);  
 addSupplied.setBackdropVisible(true);  
 addSupplied.setWidth("300px");  
 addSupplied.addClassName(LumoUtility.Padding.LARGE);  
  
 FormLayout formLayout = new FormLayout();  
 formLayout.setWidth("400px");  
  
 Select<Material> materialSelect = new Select<>();  
 materialSelect.setLabel("Материал");  
 materialSelect.setItems(materialService.findAll());  
 materialSelect.setItemLabelGenerator(Material::getName);  
 materialSelect.setWidthFull();  
  
 Select<Organization> organizationSelect = new Select<>();  
 organizationSelect.setLabel("Поставщик");  
 organizationSelect.setItems(organizationService.findAll());  
 organizationSelect.setItemLabelGenerator(Organization::getName);  
 organizationSelect.setWidthFull();  
  
 NumberField cost = new NumberField("Цена");  
 cost.setWidthFull();  
 cost.setMin(0);  
  
 IntegerField value = new IntegerField("Количество");  
 value.setWidthFull();  
 value.setMin(1);  
 value.setStepButtonsVisible(true);  
  
 Button post = new Button("Добавить", VaadinIcon.CHECK.create());  
 post.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 post.setWidth("400px");  
  
 post.addClickListener(e -> {  
 if (materialSelect.isEmpty() || organizationSelect.isEmpty() ||  
 cost.isEmpty() || value.isEmpty()) {  
 Notification.show("Заполните все поля")  
 .setPosition(Notification.Position.TOP\_END);  
 return;  
 }  
  
 try {  
 Material selectedMaterial = materialSelect.getValue();  
 Organization selectedOrganization = organizationSelect.getValue();  
  
 OrganizationMaterialKey key = new OrganizationMaterialKey(  
 selectedMaterial.getId(),  
 selectedOrganization.getId()  
 );  
  
 OrganizationMaterial supplied = new OrganizationMaterial();  
 supplied.setId(key);  
 supplied.setMaterial(selectedMaterial);  
 supplied.setOrganization(selectedOrganization);  
 supplied.setValue(value.getValue());  
 supplied.setCost(cost.getValue().floatValue());  
  
 organizationMaterialService.save(supplied);  
 updateGrid();  
  
 materialSelect.clear();  
 organizationSelect.clear();  
 cost.clear();  
 value.clear();  
  
 Notification.show("Поставка добавлена")  
 .setPosition(Notification.Position.TOP\_END);  
 } catch (Exception ex) {  
 Notification.show("Ошибка при добавлении поставки: " + ex.getMessage())  
 .setPosition(Notification.Position.TOP\_END);  
 }  
 });  
  
 formLayout.add(materialSelect, organizationSelect, cost, value, post);  
 addSupplied.add(formLayout);  
 addSupplied.setTarget(addButton);  
  
 Div content = new Div(grid);  
 content.setSizeFull();  
  
 add(toolbar, content, addSupplied);  
 }  
  
 private void setupGrid(){  
 grid.addClassName("force-focus-outline");  
 grid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 grid.setHeightFull();  
  
 // Стилизация колонок  
 grid.addColumn(om -> om.getOrganization().getName())  
 .setHeader("Поставщик")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(om -> om.getMaterial().getName())  
 .setHeader("Материал")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(OrganizationMaterial::getValue)  
 .setHeader("Количество")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(om -> String.format("%.2f", om.getCost()))  
 .setHeader("Цена")  
 .setSortable(true)  
 .setAutoWidth(true);  
 }  
  
 private List<OrganizationMaterial> updateGrid(){  
 List<OrganizationMaterial> organizationMaterials = organizationMaterialService.findAll();  
 grid.setItems(organizationMaterials);  
 return organizationMaterials;  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.Organization;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.denka88.ateliergrace.service.OrganizationService;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.dialog.Dialog;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.grid.contextmenu.GridContextMenu;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H3;  
import com.vaadin.flow.component.icon.Icon;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.popover.Popover;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.util.List;  
  
@PageTitle("Поставщики")  
@Route(value = "organizations-list", layout = MainLayout.class)  
@RolesAllowed({"ADMIN", "EMPLOYEE"})  
public class OrganizationsView extends VerticalLayout {  
  
 private final OrganizationService organizationService;  
 private final Grid<Organization> grid;  
 private final CurrentUserService currentUserService;  
  
 private FormLayout editForm = new FormLayout();  
  
 private TextField id = new TextField("ID");  
 private TextField editName = new TextField("Название");  
 private TextField editAddress = new TextField("Адрес");  
 private Button editButton = new Button("Сохранить", VaadinIcon.CHECK.create());  
  
 public OrganizationsView(OrganizationService organizationService, CurrentUserService currentUserService) {  
 this.organizationService = organizationService;  
 this.currentUserService = currentUserService;  
 this.grid = new Grid<>(Organization.class, false);  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
  
 setupGrid();  
 updateGrid();  
  
 Button addButton = new Button("Добавить поставщика", VaadinIcon.PLUS.create());  
 addButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 addButton.addClassName(LumoUtility.Margin.Bottom.LARGE);  
  
 Popover addOrganization = new Popover();  
 addOrganization.setModal(true);  
 addOrganization.setBackdropVisible(true);  
 addOrganization.setWidth("300px");  
 addOrganization.addClassName(LumoUtility.Padding.LARGE);  
  
 TextField name = new TextField("Название");  
 name.setWidthFull();  
  
 TextField address = new TextField("Адрес");  
 address.setWidthFull();  
  
 Button post = new Button("Добавить", VaadinIcon.CHECK.create());  
 post.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 post.setWidthFull();  
  
 post.addClickListener(e -> {  
 if (name.isEmpty() || address.isEmpty()) {  
 Notification.show("Заполните все поля")  
 .setPosition(Notification.Position.TOP\_END);  
 return;  
 }  
 Organization organization = new Organization();  
 organization.setName(name.getValue());  
 organization.setAddress(address.getValue());  
 organizationService.save(organization);  
 updateGrid();  
 name.clear();  
 address.clear();  
 Notification.show("Поставщик добавлен")  
 .setPosition(Notification.Position.TOP\_END);  
 });  
  
 FormLayout popoverForm = new FormLayout(name, address, post);  
 popoverForm.setWidth("400px");  
 addOrganization.add(popoverForm);  
 addOrganization.setTarget(addButton);  
  
 id.setVisible(false);  
   
 editName.setMinLength(3);  
 editName.setRequired(true);  
   
 editAddress.setMinLength(5);  
 editAddress.setRequired(true);  
  
 editForm.setWidth("400px");  
 editForm.addClassNames(  
 LumoUtility.Padding.LARGE,  
 LumoUtility.BorderRadius.LARGE,  
 LumoUtility.BoxShadow.SMALL,  
 LumoUtility.Background.BASE  
 );  
  
 editButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 editButton.addClassName(LumoUtility.Margin.Top.MEDIUM);  
 editButton.getStyle().set("margin-left", "auto");  
  
 editButton.addClickListener(e->{  
 if(editName.isEmpty() || editAddress.isEmpty()){  
 showError("Заполните все поля");  
 return;  
 }  
   
 if(editName.getValue().length() < 3){  
 showError("Неподходящая длина названия. Минимум 3 символа");  
 return;  
 }  
   
 if(editAddress.getValue().length() < 5){  
 showError("Неподходящая длина адреса. Минимум 5 символов");  
 return;  
 }  
   
 Organization updateOrganization = organizationService.findById(Long.valueOf(id.getValue())).orElse(null);  
 if (updateOrganization != null) {  
 updateOrganization.setName(editName.getValue());  
 updateOrganization.setAddress(editAddress.getValue());  
 organizationService.update(updateOrganization);  
 updateGrid();  
 editForm.setVisible(false);  
 Notification.show("Изменения сохранены")  
 .setPosition(Notification.Position.TOP\_END);  
 }  
 });  
  
 grid.addCellFocusListener(e->{  
 id.setValue(String.valueOf(e.getItem().map(Organization::getId).orElse(null)));  
 editName.setValue(e.getItem().map(Organization::getName).orElse("Не доступно"));  
 editAddress.setValue(e.getItem().map(Organization::getAddress).orElse("Не доступно"));  
 });  
  
 styleTextField(editName);  
 styleTextField(editAddress);  
  
 Button close = new Button(new Icon("lumo", "cross"), (e) -> editForm.setVisible(false));  
  
 HorizontalLayout formHeader = new HorizontalLayout();  
 formHeader.getStyle().set("justify-content", "space-between");  
  
 H3 title = new H3("Изменить поставщика");  
  
 formHeader.add(title, close);  
  
 editForm.add(formHeader, id, editName, editAddress, editButton);  
 editForm.setVisible(false);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(addButton);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(FlexComponent.JustifyContentMode.END);  
  
 Div content = new Div(grid, editForm);  
 content.addClassName(LumoUtility.Display.FLEX);  
 content.addClassName(LumoUtility.FlexDirection.COLUMN);  
 content.addClassName(LumoUtility.Gap.LARGE);  
 content.setSizeFull();  
  
 add(buttonLayout, content, addOrganization);  
 }  
  
 private void setupGrid(){  
 grid.addClassName("force-focus-outline");  
 grid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 grid.setHeightFull();  
  
 grid.addColumn(Organization::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 grid.addColumn(Organization::getName)  
 .setHeader("Название")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 grid.addColumn(Organization::getAddress)  
 .setHeader("Адрес")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 GridContextMenu<Organization> contextMenu = grid.addContextMenu();  
  
 Button editMenuItem = new Button("Изменить", VaadinIcon.EDIT.create());  
 editMenuItem.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(editMenuItem, e->{  
 if(editForm.isVisible()){  
 editForm.setVisible(false);  
 } else if (!editForm.isVisible()) {  
 editForm.setVisible(true);  
 id.setValue(String.valueOf(e.getItem().map(Organization::getId).orElse(null)));  
 editName.setValue(e.getItem().map(Organization::getName).orElse(null));  
 editAddress.setValue(e.getItem().map(Organization::getAddress).orElse(null));  
 }  
 });  
  
 if(currentUserService.getCurrentUserType().equals(UserType.ADMIN)) {  
 Button deleteMenuItem = new Button("Удалить", VaadinIcon.TRASH.create());  
 deleteMenuItem.addThemeVariants(ButtonVariant.LUMO\_ERROR, ButtonVariant.LUMO\_TERTIARY);  
 contextMenu.addItem(deleteMenuItem, e -> {  
 Dialog delete = new Dialog();  
  
 Button confirmDelete = new Button("Удалить");  
 confirmDelete.addThemeVariants(ButtonVariant.LUMO\_ERROR);  
 Button cancelDelete = new Button("Отмена");  
 cancelDelete.addThemeVariants(ButtonVariant.LUMO\_TERTIARY);  
  
 HorizontalLayout buttonLayout = new HorizontalLayout(cancelDelete, confirmDelete);  
 buttonLayout.setWidthFull();  
 buttonLayout.setJustifyContentMode(JustifyContentMode.BETWEEN);  
 delete.add(buttonLayout);  
 delete.setHeaderTitle(String.format("Удалить организацию \"%s\"", e.getItem().map(Organization::getName).orElse(null)));  
 delete.open();  
   
 confirmDelete.addClickListener(event -> {  
 organizationService.delete(e.getItem().map(Organization::getId).orElse(null));  
 updateGrid();  
 Notification.show("Поставщик удален")  
 .setPosition(Notification.Position.TOP\_END);  
 delete.close();  
 });  
   
 cancelDelete.addClickListener(event -> {  
 delete.close();  
 });  
 });  
 }  
 }  
  
 private void updateGrid(){  
 List<Organization> organizations = organizationService.findAll();  
 grid.setItems(organizations);  
 }  
  
 private void styleTextField(TextField textField) {  
 textField.setWidthFull();  
 textField.addClassName(LumoUtility.Margin.Bottom.SMALL);  
 textField.getElement().getThemeList().add("small");  
 }  
  
 private void showError(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.model.Client;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.service.AuthService;  
import com.denka88.ateliergrace.service.ClientService;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.formlayout.FormLayout;  
import com.vaadin.flow.component.html.Div;  
import com.vaadin.flow.component.html.H2;  
import com.vaadin.flow.component.html.Paragraph;  
import com.vaadin.flow.component.html.Span;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.notification.NotificationVariant;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.PasswordField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.router.RouterLink;  
import com.vaadin.flow.server.auth.AnonymousAllowed;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
  
@PageTitle("Регистрация | Грация")  
@Route(value = "register", autoLayout = false)  
@AnonymousAllowed  
public class RegistrationView extends VerticalLayout {  
  
 private final AuthService authService;  
 private final ClientService clientService;  
  
 public RegistrationView(AuthService authService, ClientService clientService) {  
 this.authService = authService;  
 this.clientService = clientService;  
  
 setPadding(false);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Background.CONTRAST\_5);  
  
 Div card = new Div();  
 card.addClassNames(  
 LumoUtility.Padding.XLARGE,  
 LumoUtility.BorderRadius.LARGE,  
 LumoUtility.BoxShadow.MEDIUM,  
 LumoUtility.Background.BASE,  
 "registration-card"  
 );  
 card.setWidth("500px");  
  
 H2 title = new H2("Регистрация");  
 title.addClassNames(LumoUtility.Margin.Bottom.LARGE, LumoUtility.TextAlignment.CENTER);  
  
 TextField login = new TextField("Логин");  
 login.setWidthFull();  
 login.setRequired(true);  
 login.setPlaceholder("Введите логин");  
 login.setMinLength(5);  
  
 PasswordField password = new PasswordField("Пароль");  
 password.setWidthFull();  
 password.setRequired(true);  
 password.setPlaceholder("Введите пароль");  
 password.setMinLength(8);  
  
 PasswordField confirmPassword = new PasswordField("Подтвердите пароль");  
 confirmPassword.setWidthFull();  
 confirmPassword.setRequired(true);  
 confirmPassword.setPlaceholder("Повторите пароль");  
  
 TextField name = new TextField("Имя");  
 name.setWidthFull();  
 name.setRequired(true);  
 name.setPlaceholder("Введите ваше имя");  
  
 TextField surname = new TextField("Фамилия");  
 surname.setWidthFull();  
 surname.setRequired(true);  
 surname.setPlaceholder("Введите вашу фамилию");  
  
 TextField patronymic = new TextField("Отчество");  
 patronymic.setWidthFull();  
 patronymic.setPlaceholder("Введите ваше отчество (если есть)");  
  
 TextField phone = new TextField("Телефон");  
 phone.setWidthFull();  
 phone.setRequired(true);  
 phone.setPlaceholder("Введите номер телефона");  
  
 Button registerButton = new Button("Зарегистрироваться", VaadinIcon.USER\_CHECK.create());  
 registerButton.addThemeVariants(ButtonVariant.LUMO\_PRIMARY);  
 registerButton.setWidthFull();  
 registerButton.addClassName(LumoUtility.Margin.Top.MEDIUM);  
  
 Span loginText = new Span("Уже есть аккаунт? ");  
 RouterLink loginLink = new RouterLink("Войти", LoginView.class);  
 loginLink.addClassName(LumoUtility.Margin.Left.SMALL);  
 HorizontalLayout loginLayout = new HorizontalLayout(loginText, loginLink);  
 loginLayout.setAlignItems(FlexComponent.Alignment.CENTER);  
 loginLayout.addClassName(LumoUtility.Margin.Top.MEDIUM);  
  
 registerButton.addClickListener(e -> {  
 if (login.isEmpty() || password.isEmpty() || confirmPassword.isEmpty() ||  
 name.isEmpty() || surname.isEmpty() || phone.isEmpty()) {  
 showError("Заполните все обязательные поля");  
 return;  
 }  
  
 if (!password.getValue().equals(confirmPassword.getValue())) {  
 showError("Пароли не совпадают");  
 return;  
 }  
   
 if(login.getValue().length() < 5){  
 showError("Неподходящая длина логина. Минимум 5 символов");  
 return;  
 }  
   
 if(password.getValue().length() < 8){  
 showError("Неподходящая длина пароля. Минимум 8 символов");  
 return;  
 }  
  
 try {  
 Client client = new Client();  
 client.setName(name.getValue());  
 client.setSurname(surname.getValue());  
 client.setPatronymic(patronymic.getValue());  
 client.setPhone(phone.getValue());  
  
 Client savedClient = clientService.save(client);  
 authService.register(  
 login.getValue(),  
 password.getValue(),  
 UserType.CLIENT,  
 savedClient.getId()  
 );  
  
 showSuccess("Регистрация успешна! Вы будете перенаправлены на страницу входа");  
 UI.getCurrent().navigate("login");  
 } catch (Exception ex) {  
 showError("Ошибка регистрации: " + ex.getMessage());  
 }  
 });  
  
 FormLayout form = new FormLayout();  
 form.addClassNames(LumoUtility.Padding.NONE);  
 form.add(  
 title, login, password, confirmPassword,  
 name, surname, patronymic, phone,  
 registerButton, loginLayout  
 );  
 form.setResponsiveSteps(  
 new FormLayout.ResponsiveStep("0", 1),  
 new FormLayout.ResponsiveStep("500px", 2)  
 );  
 form.setColspan(title, 2);  
 form.setColspan(registerButton, 2);  
 form.setColspan(loginLayout, 2);  
  
 card.add(form);  
  
 Div container = new Div(card);  
 container.setSizeFull();  
 container.addClassNames(  
 LumoUtility.Display.FLEX,  
 LumoUtility.JustifyContent.CENTER,  
 LumoUtility.AlignItems.CENTER  
 );  
  
 add(container);  
 }  
  
 private void showError(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_ERROR);  
 }  
  
 private void showSuccess(String message) {  
 Notification.show(message, 3000, Notification.Position.TOP\_CENTER)  
 .addThemeVariants(NotificationVariant.LUMO\_SUCCESS);  
 }  
}

package com.denka88.ateliergrace.view;  
  
import com.denka88.ateliergrace.MainLayout;  
import com.denka88.ateliergrace.model.\*;  
import com.denka88.ateliergrace.service.\*;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.html.H2;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.\*;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
import jakarta.annotation.security.PermitAll;  
  
import java.util.List;  
import java.util.stream.Collectors;  
  
@Route(value = "search", layout = MainLayout.class)  
@PageTitle("Результаты поиска")  
@PermitAll  
public class SearchResultsView extends VerticalLayout implements HasUrlParameter<String> {  
  
 private final OrderService orderService;  
 private final ClientService clientService;  
 private final EmployeeService employeeService;  
 private final MaterialService materialService;  
 private final OrganizationService organizationService;  
  
 public SearchResultsView(OrderService orderService,  
 ClientService clientService,  
 EmployeeService employeeService,  
 MaterialService materialService,  
 OrganizationService organizationService) {  
 this.orderService = orderService;  
 this.clientService = clientService;  
 this.employeeService = employeeService;  
 this.materialService = materialService;  
 this.organizationService = organizationService;  
  
 setPadding(true);  
 setSpacing(false);  
 setSizeFull();  
 addClassName(LumoUtility.Padding.LARGE);  
 }  
  
 @Override  
 public void setParameter(BeforeEvent event, String parameter) {  
 removeAll();  
  
 if (parameter == null || parameter.trim().isEmpty()) {  
 add(new H2("Введите поисковый запрос"));  
 return;  
 }  
  
 String searchQuery = parameter.toLowerCase();  
 add(new H2("Результаты поиска по запросу: \"" + searchQuery + "\""));  
  
 List<Order> orders = orderService.findAll().stream()  
 .filter(order -> order.getOrderName().toLowerCase().contains(searchQuery) ||  
 order.getDescription().toLowerCase().contains(searchQuery))  
 .toList();  
  
 if (!orders.isEmpty()) {  
 add(new H2("Заказы"));  
 Grid<Order> orderGrid = new Grid<>(Order.class, false);  
 orderGrid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 orderGrid.setHeightFull();  
  
 orderGrid.addColumn(Order::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 orderGrid.addColumn(Order::getClient)  
 .setHeader("Клиент")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 orderGrid.addColumn(Order::getOrderName)  
 .setHeader("Название")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 orderGrid.addColumn(Order::getDescription)  
 .setHeader("Описание заказа")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 orderGrid.addColumn(Order::getOrderDate)  
 .setHeader("Дата создания")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 orderGrid.addColumn(order -> {  
 if (order.getOrderEmployees() == null || order.getOrderEmployees().isEmpty()) {  
 return "Не назначено";  
 }  
 return order.getOrderEmployees().stream()  
 .filter(oe -> oe.getDateOfReady() != null)  
 .findFirst()  
 .map(oe -> oe.getDateOfReady().toString())  
 .orElse("Не указана");  
 }).setHeader("Дата готовности").setSortable(true).setAutoWidth(true);  
  
 orderGrid.addColumn(Order::getCost)  
 .setHeader("Стоимость")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 orderGrid.addColumn(Order::getStatus)  
 .setHeader("Статус")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 orderGrid.addColumn(order -> {  
 if(order.getOrderEmployees() == null || order.getOrderEmployees().isEmpty()){  
 return "Не назначены";  
 }  
 return order.getOrderEmployees().stream()  
 .map(OrderEmployee::getEmployee)  
 .map(Employee::toString)  
 .collect(Collectors.joining(", "));  
 }).setHeader("Сотрудники");  
  
 orderGrid.addColumn(order -> {  
 if (order.getMaterials() == null || order.getMaterials().isEmpty()) {  
 return "Не назначены";  
 }  
 return order.getMaterials().stream()  
 .map(Material::getName)  
 .collect(Collectors.joining(", "));  
 }).setHeader("Материалы");  
 orderGrid.setItems(orders);  
 add(orderGrid);  
 }  
  
 List<Client> clients = clientService.findAll().stream()  
 .filter(client -> client.getName().toLowerCase().contains(searchQuery) ||  
 client.getSurname().toLowerCase().contains(searchQuery) ||  
 client.getPatronymic().toLowerCase().contains(searchQuery) ||  
 client.getPhone().toLowerCase().contains(searchQuery))  
 .toList();  
  
 if (!clients.isEmpty()) {  
 add(new H2("Клиенты"));  
 Grid<Client> clientsGrid = new Grid<>(Client.class, false);  
 clientsGrid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 clientsGrid.setHeightFull();  
  
 clientsGrid.addColumn(Client::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 clientsGrid.addColumn(Client::getSurname)  
 .setHeader("Фамилия")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 clientsGrid.addColumn(Client::getName)  
 .setHeader("Имя")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 clientsGrid.addColumn(Client::getPatronymic)  
 .setHeader("Отчество")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 clientsGrid.addColumn(Client::getPhone)  
 .setHeader("Номер телефона")  
 .setSortable(true)  
 .setAutoWidth(true);  
 clientsGrid.setItems(clients);  
 add(clientsGrid);  
 }  
   
 List<Employee> employees = employeeService.findAll().stream()  
 .filter(employee -> employee.getName().toLowerCase().contains(searchQuery) ||  
 employee.getPatronymic().toLowerCase().contains(searchQuery) ||  
 employee.getSurname().toLowerCase().contains(searchQuery) ||  
 employee.getPost().toLowerCase().contains(searchQuery))  
 .toList();  
   
 if (!employees.isEmpty()) {  
 add(new H2("Сотрудники"));  
 Grid<Employee> employeesGrid = new Grid<>(Employee.class, false);  
 employeesGrid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 employeesGrid.setHeightFull();  
  
 employeesGrid.addColumn(Employee::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 employeesGrid.addColumn(Employee::getSurname)  
 .setHeader("Фамилия")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 employeesGrid.addColumn(Employee::getName)  
 .setHeader("Имя")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 employeesGrid.addColumn(Employee::getPatronymic)  
 .setHeader("Отчество")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 employeesGrid.addColumn(Employee::getPost)  
 .setHeader("Должность")  
 .setSortable(true)  
 .setAutoWidth(true);  
 employeesGrid.setItems(employees);  
 add(employeesGrid);  
 }  
   
 List<Material> materials = materialService.findAll().stream()  
 .filter(material -> material.getName().toLowerCase().contains(searchQuery)).toList();  
   
 if (!materials.isEmpty()) {  
 add(new H2("Материалы"));  
 Grid<Material> materialsGrid = new Grid<>(Material.class, false);  
 materialsGrid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 materialsGrid.setHeightFull();  
  
 materialsGrid.addColumn(Material::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 materialsGrid.addColumn(Material::getName)  
 .setHeader("Название")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 materialsGrid.addColumn(Material::getValue)  
 .setHeader("На складе")  
 .setSortable(true)  
 .setAutoWidth(true);  
 materialsGrid.setItems(materials);  
 add(materialsGrid);  
 }  
   
 List<Organization> organizations = organizationService.findAll().stream()  
 .filter(organization -> organization.getName().toLowerCase().contains(searchQuery) ||  
 organization.getAddress().toLowerCase().contains(searchQuery))  
 .toList();  
  
 if (!organizations.isEmpty()) {  
 add(new H2("Поставщики"));  
 Grid<Organization> organizationsGrid = new Grid<>(Organization.class, false);  
 organizationsGrid.addThemeVariants(  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_WRAP\_CELL\_CONTENT,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_COLUMN\_BORDERS,  
 com.vaadin.flow.component.grid.GridVariant.LUMO\_ROW\_STRIPES  
 );  
 organizationsGrid.setHeightFull();  
  
 organizationsGrid.addColumn(Organization::getId)  
 .setHeader("ID")  
 .setSortable(true)  
 .setAutoWidth(true)  
 .setFlexGrow(0);  
  
 organizationsGrid.addColumn(Organization::getName)  
 .setHeader("Название")  
 .setSortable(true)  
 .setAutoWidth(true);  
  
 organizationsGrid.addColumn(Organization::getAddress)  
 .setHeader("Адрес")  
 .setSortable(true)  
 .setAutoWidth(true);  
 organizationsGrid.setItems(organizations);  
 add(organizationsGrid);  
 }  
 }  
}

package com.denka88.ateliergrace;  
  
import com.denka88.ateliergrace.service.CurrentUserService;  
import com.vaadin.flow.component.UI;  
import com.vaadin.flow.component.applayout.AppLayout;  
import com.vaadin.flow.component.applayout.DrawerToggle;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.button.ButtonVariant;  
import com.vaadin.flow.component.html.H1;  
import com.vaadin.flow.component.html.Span;  
import com.vaadin.flow.component.icon.Icon;  
import com.vaadin.flow.component.icon.VaadinIcon;  
import com.vaadin.flow.component.orderedlayout.FlexComponent;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.Scroller;  
import com.vaadin.flow.component.sidenav.SideNav;  
import com.vaadin.flow.component.sidenav.SideNavItem;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Layout;  
import com.vaadin.flow.server.auth.AnonymousAllowed;  
import com.vaadin.flow.spring.security.AuthenticationContext;  
import com.vaadin.flow.theme.lumo.LumoUtility;  
  
@Layout  
@AnonymousAllowed  
public class MainLayout extends AppLayout {  
  
 private final CurrentUserService currentUserService;  
  
 public MainLayout(AuthenticationContext authenticationContext, CurrentUserService currentUserService) {  
 this.currentUserService = currentUserService;  
  
 DrawerToggle toggle = new DrawerToggle();  
  
 H1 title = new H1("Грация");  
 title.getStyle()  
 .set("font-size", "var(--lumo-font-size-xxl)")  
 .set("margin", "0")  
 .set("color", "var(--lumo-primary-text-color)")  
 .set("font-weight", "600")  
 .set("cursor", "pointer")  
 .set("transition", "color 0.2s");  
 title.addClassName(LumoUtility.Padding.Horizontal.LARGE);  
 title.addClickListener(e -> UI.getCurrent().navigate("/main"));  
 title.addClickListener(e -> title.getStyle().set("color", "var(--lumo-primary-color-50pct)"));  
 title.addClickListener(e -> title.getStyle().set("color", "var(--lumo-primary-text-color)"));  
  
 TextField searchField = new TextField();  
 searchField.setPlaceholder("Поиск...");  
 searchField.setWidth("300px");  
 searchField.setPrefixComponent(VaadinIcon.SEARCH.create());  
 searchField.addValueChangeListener(e -> {  
 if (e.getValue() != null && !e.getValue().trim().isEmpty()) {  
 UI.getCurrent().navigate("search/" + e.getValue().trim());  
 }  
 });  
  
 HorizontalLayout navbarRight = new HorizontalLayout();  
 navbarRight.setAlignItems(FlexComponent.Alignment.CENTER);  
 navbarRight.getStyle().set("margin-left", "auto").set("margin-right", "5px");  
  
 HorizontalLayout userControls = new HorizontalLayout();  
 userControls.setSpacing(false);  
 userControls.setAlignItems(FlexComponent.Alignment.CENTER);  
 userControls.addClassName(LumoUtility.Gap.SMALL);  
  
 Span username = new Span();  
 username.getStyle()  
 .set("font-weight", "500")  
 .set("color", "var(--lumo-secondary-text-color)")  
 .set("margin-right", "var(--lumo-space-s)");  
 currentUserService.getCurrentAuth().ifPresent(auth -> {  
 username.setText(auth.getLogin());  
 });  
  
 Icon userIcon = VaadinIcon.USER\_CHECK.create();  
 userIcon.getStyle()  
 .set("width", "var(--lumo-icon-size-s)")  
 .set("height", "var(--lumo-icon-size-s)")  
 .set("color", "var(--lumo-contrast-60pct)");  
  
 Button logout = new Button("Выйти", VaadinIcon.EXIT\_O.create(),  
 e -> authenticationContext.logout());  
 logout.addThemeVariants(ButtonVariant.LUMO\_TERTIARY\_INLINE);  
 logout.getStyle()  
 .set("margin-left", "var(--lumo-space-s)")  
 .set("color", "var(--lumo-error-text-color)")  
 .set("padding", "0");  
  
 userControls.add(userIcon, username, logout);  
  
 navbarRight.add(searchField, userControls);  
  
 SideNav nav = getSideNav(authenticationContext);  
 Scroller scroller = new Scroller(nav);  
 scroller.setClassName(LumoUtility.Padding.SMALL);  
  
 addToDrawer(scroller);  
 addToNavbar(toggle, title, navbarRight);  
 }  
 private SideNav getSideNav(AuthenticationContext authenticationContext) {  
 SideNav sideNav = new SideNav();  
 if (authenticationContext.hasRole("ADMIN")) {  
 sideNav.addItem(  
 new SideNavItem("Заказы", "/orders-list",VaadinIcon.PAPERCLIP.create()),  
 new SideNavItem("Клиенты", "/clients-list", VaadinIcon.USER.create()),  
 new SideNavItem("Поставщики", "/organizations-list", VaadinIcon.OFFICE.create()),  
 new SideNavItem("Материалы", "/materials-list", VaadinIcon.CUBE.create()),  
 new SideNavItem("Сотрудники", "/employees-list", VaadinIcon.USER\_STAR.create()),  
 new SideNavItem("Поставки", "/supplies", VaadinIcon.TRUCK.create()),  
 new SideNavItem("О приложении", "/about", VaadinIcon.INFO\_CIRCLE\_O.create())  
 );  
 }  
 else if (authenticationContext.hasRole("CLIENT")) {  
 sideNav.addItem(  
 new SideNavItem("Мои заказы", "/my-orders", VaadinIcon.PAPERCLIP.create()),  
 new SideNavItem("Мой профиль", "/client-profile", VaadinIcon.USER\_CARD.create()),  
 new SideNavItem("О приложении", "/about", VaadinIcon.INFO\_CIRCLE\_O.create())  
 );  
 }  
 else{  
 sideNav.addItem(  
 new SideNavItem("Заказы", "/orders-list",VaadinIcon.PAPERCLIP.create()),  
 new SideNavItem("Поставщики", "/organizations-list", VaadinIcon.OFFICE.create()),  
 new SideNavItem("Материалы", "/materials-list", VaadinIcon.CUBE.create()),  
 new SideNavItem("Поставки", "/supplies", VaadinIcon.TRUCK.create()),  
 new SideNavItem("Клиенты", "/clients-list", VaadinIcon.USER.create()),  
 new SideNavItem("О приложении", "/about", VaadinIcon.INFO\_CIRCLE\_O.create())  
   
 );  
 }  
 return sideNav;  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Auth;  
import com.denka88.ateliergrace.model.UserType;  
import com.denka88.ateliergrace.repo.AuthRepo;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.\*;  
import org.mockito.junit.jupiter.MockitoExtension;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
  
@ExtendWith(MockitoExtension.class)  
class AuthServiceImplTest {  
  
 @Mock  
 private AuthRepo authRepo;  
   
 @Mock  
 private BCryptPasswordEncoder passwordEncoder;  
  
 @InjectMocks  
 private AuthServiceImpl authService;  
  
 @BeforeEach  
 void setUp() {  
 MockitoAnnotations.*openMocks*(this);  
 }  
  
 @Test  
 void register() {  
 String *login* = "testUser";  
 String *password* = "password";  
 Long *userId* = 1L;  
 UserType *userType* = UserType.*CLIENT*;  
  
 *when*(authRepo.existsByLogin(*login*)).thenReturn(false);  
 *when*(passwordEncoder.encode(*password*)).thenReturn("hashedPassword");  
  
 authService.register(*login*, *password*, *userType*, *userId*);  
  
 ArgumentCaptor<Auth> *captor* = ArgumentCaptor.*forClass*(Auth.class);  
 *verify*(authRepo).save(*captor*.capture());  
  
 Auth *savedAuth* = *captor*.getValue();  
 *assertEquals*(*login*, *savedAuth*.getLogin());  
 *assertEquals*("hashedPassword", *savedAuth*.getPasswordHash());  
 *assertEquals*(*userType*, *savedAuth*.getUserType());  
 *assertEquals*(*userId*, *savedAuth*.getUserId());  
 }  
  
 @Test  
 void delete() {  
 Long *userId* = 123L;  
 authService.delete(*userId*);  
 *verify*(authRepo, *times*(1)).deleteByUserId(*userId*);  
 }  
  
 @Test  
 void findByLogin() {  
 String *login* = "user123";  
 Auth *mockAuth* = new Auth();  
 *mockAuth*.setLogin(*login*);  
 *when*(authRepo.findByLogin(*login*)).thenReturn(Optional.*of*(*mockAuth*));  
  
 Optional<Auth> *result* = authService.findByLogin(*login*);  
 *assertTrue*(*result*.isPresent());  
 *assertEquals*(*login*, *result*.get().getLogin());  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Client;  
import com.denka88.ateliergrace.repo.ClientRepo;  
import com.denka88.ateliergrace.service.AuthService;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
  
import java.util.List;  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.*verify*;  
import static org.mockito.Mockito.*when*;  
  
@ExtendWith(MockitoExtension.class)  
class ClientServiceImplTest {  
  
 @Mock  
 private ClientRepo clientRepo;  
  
 @Mock  
 private AuthService authService;  
  
 @InjectMocks  
 private ClientServiceImpl clientService;  
  
 @Test  
 void findAll() {  
 List<Client> clients = List.of(new Client());  
 when(clientRepo.findAll()).thenReturn(clients);  
  
 assertEquals(clients, clientService.findAll());  
 }  
  
 @Test  
 void findById() {  
 Client client = new Client();  
 client.setId(1L);  
 when(clientRepo.findById(1L)).thenReturn(Optional.of(client));  
  
 Optional<Client> result = clientService.findById(1L);  
 assertTrue(result.isPresent());  
 assertEquals(1L, result.get().getId());  
 }  
  
 @Test  
 void save() {  
 Client client = new Client();  
 clientService.save(client);  
 verify(clientRepo).save(client);  
 }  
  
 @Test  
 void delete() {  
 Client client = new Client();  
 client.setId(1L);  
 when(clientRepo.findById(1L)).thenReturn(Optional.of(client));  
  
 clientService.delete(1L);  
  
 verify(authService).delete(1L);  
 verify(clientRepo).delete(client);  
 }  
  
 @Test  
 void update() {  
 Client client = new Client();  
 clientService.update(client);  
 verify(clientRepo).save(client);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Employee;  
import com.denka88.ateliergrace.repo.EmployeeRepo;  
import com.denka88.ateliergrace.service.AuthService;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
  
import java.util.HashSet;  
import java.util.List;  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.verify;  
import static org.mockito.Mockito.when;  
  
@ExtendWith(MockitoExtension.class)  
class EmployeeServiceImplTest {  
  
 @Mock  
 private EmployeeRepo employeeRepo;  
  
 @Mock  
 private AuthService authService;  
  
 @InjectMocks  
 private EmployeeServiceImpl service;  
  
 @Test  
 void findAll() {  
 List<Employee> list = List.of(new Employee());  
 when(employeeRepo.findAll()).thenReturn(list);  
 assertEquals(list, service.findAll());  
 }  
  
 @Test  
 void findById() {  
 Employee employee = new Employee();  
 employee.setId(1L);  
 when(employeeRepo.findById(1L)).thenReturn(Optional.of(employee));  
 assertEquals(employee, service.findById(1L).orElse(null));  
 }  
  
 @Test  
 void save() {  
 Employee employee = new Employee();  
 service.save(employee);  
 verify(employeeRepo).save(employee);  
 }  
  
 @Test  
 void update() {  
 Employee employee = new Employee();  
 service.update(employee);  
 verify(employeeRepo).save(employee);  
 }  
  
 @Test  
 void delete() {  
 Employee employee = new Employee();  
 employee.setId(1L);  
 employee.setOrders(new HashSet<>());  
 when(employeeRepo.findById(1L)).thenReturn(Optional.of(employee));  
  
 service.delete(1L);  
  
 verify(authService).delete(1L);  
 verify(employeeRepo).deleteById(1L);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Material;  
import com.denka88.ateliergrace.model.Order;  
import com.denka88.ateliergrace.repo.MaterialRepo;  
import com.denka88.ateliergrace.repo.OrderRepo;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
  
import java.util.HashSet;  
import java.util.List;  
import java.util.Optional;  
import java.util.Set;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.verify;  
import static org.mockito.Mockito.when;  
  
@ExtendWith(MockitoExtension.class)  
class MaterialServiceImplTest {  
  
 @Mock  
 private MaterialRepo materialRepo;  
  
 @Mock  
 private OrderRepo orderRepo;  
  
 @InjectMocks  
 private MaterialServiceImpl materialService;  
  
 @Test  
 void findAll() {  
 List<Material> list = List.of(new Material());  
 when(materialRepo.findAll()).thenReturn(list);  
 assertEquals(list, materialService.findAll());  
 }  
  
 @Test  
 void findById() {  
 Material material = new Material();  
 material.setId(1L);  
 when(materialRepo.findById(1L)).thenReturn(Optional.of(material));  
  
 Optional<Material> result = materialService.findById(1L);  
 assertTrue(result.isPresent());  
 assertEquals(1L, result.get().getId());  
 }  
  
 @Test  
 void save() {  
 Material material = new Material();  
 materialService.save(material);  
 verify(materialRepo).save(material);  
 }  
  
 @Test  
 void delete() {  
 Long id = 1L;  
 Material material = new Material();  
 material.setId(id);  
 Order order = new Order();  
 order.setMaterials(new HashSet<>(Set.of(material)));  
  
 when(materialRepo.findById(id)).thenReturn(Optional.of(material));  
 when(orderRepo.findByMaterialsId(id)).thenReturn(List.of(order));  
  
 materialService.delete(id);  
  
 verify(materialRepo).deleteById(id);  
 }  
  
 @Test  
 void update() {  
 Material material = new Material();  
 materialService.update(material);  
 verify(materialRepo).save(material);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.\*;  
import com.denka88.ateliergrace.repo.EmployeeRepo;  
import com.denka88.ateliergrace.repo.OrderEmployeeRepo;  
import com.denka88.ateliergrace.repo.OrderRepo;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
  
import java.time.LocalDate;  
import java.util.List;  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.ArgumentMatchers.any;  
import static org.mockito.Mockito.verify;  
import static org.mockito.Mockito.when;  
  
@ExtendWith(MockitoExtension.class)  
class OrderEmployeeServiceImplTest {  
  
 @Mock  
 private OrderEmployeeRepo orderEmployeeRepo;  
  
 @Mock  
 private EmployeeRepo employeeRepo;  
  
 @Mock  
 private OrderRepo orderRepo;  
  
 @InjectMocks  
 private OrderEmployeeServiceImpl service;  
  
 @Test  
 void findAll() {  
 when(orderEmployeeRepo.findAll()).thenReturn(List.of(new OrderEmployee()));  
 assertEquals(1, service.findAll().size());  
 }  
  
 @Test  
 void save() {  
 OrderEmployee oe = new OrderEmployee();  
 service.save(oe);  
 verify(orderEmployeeRepo).save(oe);  
 }  
  
 @Test  
 void update() {  
 OrderEmployee oe = new OrderEmployee();  
 service.update(oe);  
 verify(orderEmployeeRepo).save(oe);  
 }  
  
 @Test  
 void setReadyDate() {  
 OrderEmployeeKey key = new OrderEmployeeKey(1L, 1L);  
 OrderEmployee oe = new OrderEmployee();  
 when(orderEmployeeRepo.findById(key)).thenReturn(Optional.of(oe));  
  
 service.setReadyDate(1L, 1L, LocalDate.now());  
  
 verify(orderEmployeeRepo).save(oe);  
 assertNotNull(oe.getDateOfReady());  
 }  
  
 @Test  
 void createAssignment() {  
 when(orderRepo.getReferenceById(1L)).thenReturn(new Order());  
 when(employeeRepo.getReferenceById(2L)).thenReturn(new Employee());  
  
 service.createAssignment(1L, 2L, LocalDate.now());  
  
 verify(orderEmployeeRepo).save(any(OrderEmployee.class));  
 }  
  
 @Test  
 void getInProgressCount() {  
 when(orderEmployeeRepo.countByEmployeeIdAndOrderStatus(1L, Status.PROGRESS)).thenReturn(3L);  
 assertEquals(3L, service.getInProgressCount(1L));  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Order;  
import com.denka88.ateliergrace.model.Status;  
import com.denka88.ateliergrace.repo.OrderRepo;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
  
import java.util.List;  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.ArgumentMatchers.*any*;  
import static org.mockito.Mockito.*verify*;  
import static org.mockito.Mockito.*when*;  
  
@ExtendWith(MockitoExtension.class)  
class OrderServiceImplTest {  
  
 @Mock  
 private OrderRepo orderRepo;  
  
 @InjectMocks  
 private OrderServiceImpl orderService;  
  
 @Test  
 void findAll() {  
 List<Order> *orders* = List.*of*(new Order());  
 *when*(orderRepo.findAllWithEmployeesAndMaterials()).thenReturn(*orders*);  
 *assertEquals*(*orders*, orderService.findAll());  
 }  
  
 @Test  
 void findById() {  
 Order *order* = new Order();  
 *order*.setId(1L);  
 *when*(orderRepo.findById(1L)).thenReturn(Optional.*of*(*order*));  
 *assertEquals*(*order*, orderService.findById(1L).orElse(null));  
 }  
  
 @Test  
 void save() {  
 Order *order* = new Order();  
 Order *saved* = new Order();  
 *saved*.setId(1L);  
  
 *when*(orderRepo.save(*any*())).thenReturn(*saved*);  
  
 Order *result* = orderService.save(*order*);  
  
 *assertNotNull*(*result*);  
 *assertEquals*(1L, *result*.getId());  
 *assertNotNull*(*order*.getOrderDate());  
 }  
  
 @Test  
 void delete() {  
 orderService.delete(1L);  
 *verify*(orderRepo).deleteById(1L);  
 }  
  
 @Test  
 void update() {  
 Order order = new Order();  
 orderService.update(order);  
 verify(orderRepo).save(order);  
 }  
  
 @Test  
 void findByClientId() {  
 List<Order> list = List.of(new Order());  
 when(orderRepo.findByClientId(1L)).thenReturn(list);  
 assertEquals(list, orderService.findByClientId(1L));  
 }  
  
 @Test  
 void completeOrder() {  
 Order order = new Order();  
 order.setId(1L);  
  
 when(orderRepo.findById(1L)).thenReturn(Optional.of(order));  
  
 orderService.completeOrder(1L);  
  
 assertEquals(Status.COMPLETED, order.getStatus());  
 verify(orderRepo).save(order);  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.OrganizationMaterial;  
import com.denka88.ateliergrace.model.OrganizationMaterialKey;  
import com.denka88.ateliergrace.repo.OrganizationMaterialRepo;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
  
import java.util.List;  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.*verify*;  
import static org.mockito.Mockito.*when*;  
  
@ExtendWith(MockitoExtension.class)  
class OrganizationMaterialServiceImplTest {  
  
 @Mock  
 private OrganizationMaterialRepo repo;  
  
 @InjectMocks  
 private OrganizationMaterialServiceImpl service;  
  
 @Test  
 void findAll() {  
 *when*(repo.findAll()).thenReturn(List.*of*(new OrganizationMaterial()));  
 *assertEquals*(1, service.findAll().size());  
 }  
  
 @Test  
 void save() {  
 OrganizationMaterial *om* = new OrganizationMaterial();  
 service.save(*om*);  
 *verify*(repo).save(*om*);  
 }  
  
 @Test  
 void update() {  
 OrganizationMaterial *om* = new OrganizationMaterial();  
 service.update(*om*);  
 *verify*(repo).save(*om*);  
 }  
  
 @Test  
 void findById() {  
 OrganizationMaterialKey *key* = new OrganizationMaterialKey(1L, 1L);  
 *when*(repo.findById(*key*)).thenReturn(Optional.*of*(new OrganizationMaterial()));  
 *assertTrue*(service.findById(*key*).isPresent());  
 }  
}

package com.denka88.ateliergrace.impl;  
  
import com.denka88.ateliergrace.model.Organization;  
import com.denka88.ateliergrace.repo.OrganizationMaterialRepo;  
import com.denka88.ateliergrace.repo.OrganizationRepo;  
import org.junit.jupiter.api.Test;  
import org.junit.jupiter.api.extension.ExtendWith;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.junit.jupiter.MockitoExtension;  
  
import java.util.List;  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.*verify*;  
import static org.mockito.Mockito.*when*;  
  
@ExtendWith(MockitoExtension.class)  
class OrganizationServiceImplTest {  
  
 @Mock  
 private OrganizationRepo organizationRepo;  
  
 @Mock  
 private OrganizationMaterialRepo organizationMaterialRepo;  
  
 @InjectMocks  
 private OrganizationServiceImpl service;  
  
 @Test  
 void findAll() {  
 *when*(organizationRepo.findAll()).thenReturn(List.*of*(new Organization()));  
 *assertEquals*(1, service.findAll().size());  
 }  
  
 @Test  
 void findById() {  
 Organization *organization* = new Organization();  
 *organization*.setId(1L);  
 *when*(organizationRepo.findById(1L)).thenReturn(Optional.*of*(*organization*));  
 *assertEquals*(*organization*, service.findById(1L).orElse(null));  
 }  
  
 @Test  
 void save() {  
 Organization *organization* = new Organization();  
 service.save(*organization*);  
 *verify*(organizationRepo).save(*organization*);  
 }  
  
 @Test  
 void update() {  
 Organization *organization* = new Organization();  
 service.update(*organization*);  
 *verify*(organizationRepo).save(*organization*);  
 }  
  
 @Test  
 void delete() {  
 Organization *organization* = new Organization();  
 *organization*.setId(1L);  
 *when*(organizationRepo.findById(1L)).thenReturn(Optional.*of*(*organization*));  
  
 service.delete(1L);  
  
 *verify*(organizationMaterialRepo).deleteByOrganizationId(1L);  
 *verify*(organizationRepo).deleteById(1L);  
 }  
}