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 | Корнев В.В. |
| Курс, группа | 3 курс, группа ИС-3 |
| Специальность | 09.02.07 Информационные системы и программирование |
| Руководитель практики | Смольняков А.В. |

г. Белореченск

2023

**СОДЕРЖАНИЕ**

Введение 3

Проектирование информационной системы 4

Техническое задание 6

Эскизный проект 26

Разработка и документирование информационной системы 37

Руководство пользователя 47

Заключение

Список использованных источников

Приложение

**ВВЕДЕНИЕ**

В современном мире размеры обрабатываемой информации только увеличиваются, требуя от людей больше ресурсов и времени для ее обработки. Чтобы решить эту проблему, как можно больше областей подвергаются автоматизации, с целью экономить время людей и упростить работу. Поэтому мы облегчим работу менеджера в мебельном салоне.

Предметом исследования в данной работе является мебельный салон «МСВ», в котором потребовалось провести анализ предметной области, выявить сферу предприятия, нуждающуюся в модернизации и провести процесс ее автоматизации для улучшения работы мебельного салона.

После исследования структуры организации и проведения беседы с персоналом был выявлен объект исследования, а именно Менеджерский отдел предприятия, который в итоге проводимой работы будет автоматизирован при помощи информационной системы.

Целью данной работы является разработка информационной системы для автоматизации работы Менеджерского отдела в мебельном салон. Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ предметной области предприятия
2. Составить техническое задание (ТЗ)
3. Составить эскизный проект информационной системы
4. Разработать серверную часть приложения
5. Разработать клиентскую часть приложения
6. Составить руководство пользователя

Результатом проделанной работы будет являться информационная система, позволяющая менеджеру мебельного салона выполнять свои профессиональные обязанности с высокой скоростью.

**1.** **ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Анализ предметной области

Информационная система мебельного салона

Мебельная фабрика МСТ. Мебель образована в 1997 году и с тех пор успешно реализует свою продукцию на всей территории России. Компания не стоит на месте и постоянно развивается Мы производим корпусную мебель для жилых помещений: шкафы, тумбы, спальни.

Данная фирма была создана для производства и продажи мебели. Мебель производится в разных видах материала и размерах, в которых можно найти качественные предметы для обстановки домов.

Рабочих в мебельном салоне можно подразделить на несколько категорий: Комплектовщик, сборщик мебели, дизайнер, конструктор, столяр. Каждая из перечисленных категорий работников имеет уникальные атрибуты-характеристики, определяемые профессиональной направленностью.

В салоне мебели присутствуют много различных товаров. Мебель можно подразделить на множество различных видов. Каждый товар соответствует своей категории например (стул не должен быть размещён со шкафом) это никак не как не подходит для сочетания.

Для создания мебели необходимы различные типа материалов: Металл, нержавеющая сталь, алюминий, дерево, пластик. Для каждого вида мебели рассчитывается материал, в котором указывается количество требуемого материала. Деревянный стул (необходимо дерево, для создания металлического стола необходим металл). Существуют виды шкафов, тумб (однодверные шкафы, двухдверные шкафы, навесные шкафы, Угловые шкафы, Прикроватные тумбы, тумбы под обувь, тумбы под телевизор).

Был проведён анализ предметной области «**У**чет заказов в мебельном салоне», после которого было принято решение об автоматизации и улучшение некоторых функций, к ним были отнесены:

**Виды запросов в информационной системе:**

1. Получить список товара
2. Получить список товара по типу
3. Получить список товара по названию
4. Получить список товара по материалу
5. Получить список товара по категории
6. Получить список товара по количеству
7. Получить список клиентов
8. Получить список по имени
9. Получить список заказов
10. Получить список по названию
11. **ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

2.1 Общие сведения

2.1.1 Полное наименование системы и ее условное обозначение

Полное наименование системы: Мебельный салон Вованчика.

Краткое наименование системы: "МСВ".

2.1.2 Шифр темы или шифр (номер) договора

Шифр темы: Мебельный салон-10

Номер договора: №1/12-34-56-001 от 26.04.2023.

2.1.3. Наименование предприятий разработчика и заказчика системы, их реквизиты

Полное наименование заказчика: Смольняков А.В

Сокращённое название: МСТ.

Юридический и почтовый адрес: 404121, Краснодарский край, г. Белореченск, ул. Интернациональная, 34.

Контактный телефон: (+7 9676) 63-30-96.

ИНН: 3444049170

КПП: 343502001.

БИК: 041806001

ОГРН: 1023403440818

Расчетный счет: 40501810403492000005.

Полное наименование разработчика: Общество с ограниченной ответственностью "Корнев ПО".

Краткое наименование: ООО "Корнев ПО".

Юридический и почтовый адрес: 404130, Волгоградская область, город Волжский, улица Камская, дом 6.

Контактный телефон: 8-967-663-30-96.

ИНН: 3435653093.

КПП: 342341002.

Расчетный счет: 40702653800000000001.

2.1.4. Порядок оформления и предъявления заказчику результатов работ по созданию системы, ее частей и отдельных средств

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 “Мебельный салон”, разработанный исполнителем, передается заказчику на бумажных (в двух экземплярах) и на машинных носителях (DVD), один экземпляр после подписания передается исполнителю. Текстовые документы, передаваемые на машинных носителях, должны быть представлены в форматах PDF.

2.2. Назначение и цели создания (развития) системы

2.2.1. Назначение системы

Разрабатываемая система предназначена для упрощения работы мебельного салона, выполненных по заказу.

2.2.2 Цели создания системы

Основными целями создания “Мебельный салон” являются:

- Быстрый поиск нужных товаров;

- Доступность для любого клиента;

- Список типов товара. все будет представлено на одной странице.

Для достижения поставленных целей система должна решать следующие задачи:

- Наличие сформированной база данных различных товаров;

- Добавление нового товара в список;

- Редактирование товара в списке;

- Удаление товара в списке;

2.3. Характеристика объектов автоматизации

В ходе проведения работ по разработке Мебельный салон «МСВ» автоматизируются процессы для облегчения слежки за товаром в мебельном салоне.

Система будет собирать товары одной категории в один список. При выборе определенного товара его можно будет редактировать или удалить из списка товаров.

Также будет производится поиск товаров по названию.

При внедрении системы будет достигнуто:

- Простой поиск необходимых товаров;

- Удобное сравнение товаров по цене;

2.4. Требования к системе

2.4.1. Требования к системе в целом

2.4.1.1. Требования к структуре и функционированию системы

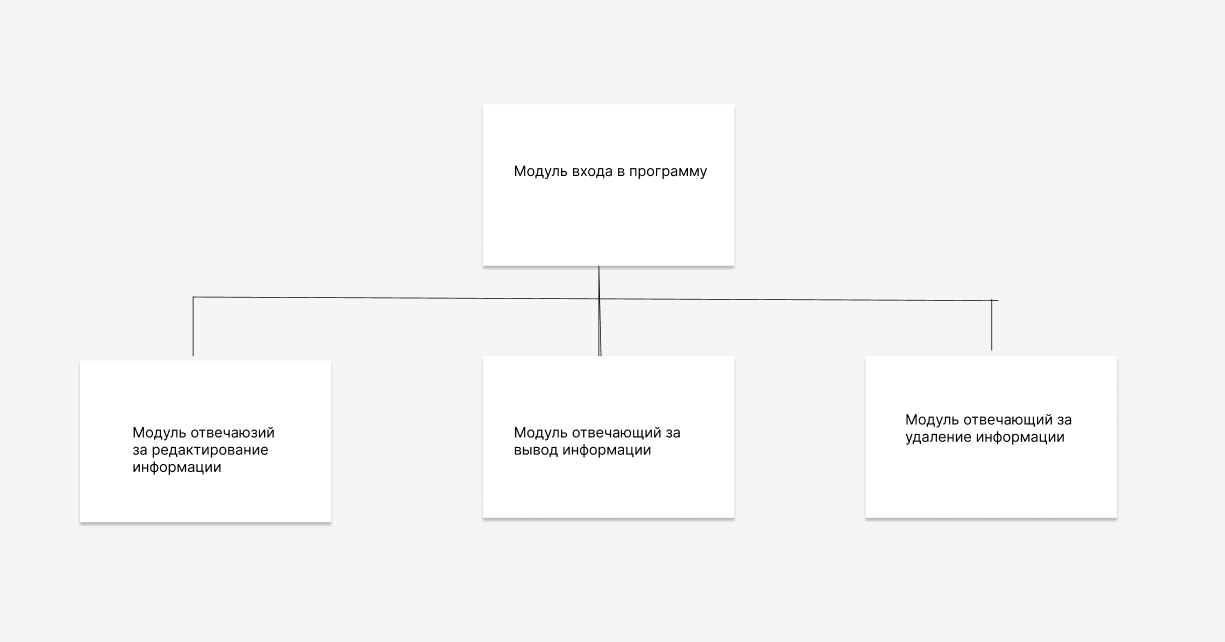


Рисунок 2.1. - Модули разрабатываемой системы

1. Модуль для входа программу. Благодаря нему, менеджер может зайти и пользоваться, введя свои данные.

2.Модуль получение и вывода информации. С помощью этого модуля пользователь может получать данные с БД, а так-же видеть информацию в таблице.

3.Модуль отвечающий за удаление информации. С помощью этого модуля пользователь может удалять нужную ему информацию из БД.

На данном этапе необходимо описать примерную архитектуру разрабатываемой системы.

Для того, чтобы сделать программу, мы будем использовать клиент-серверную архитектуру. Чтобы убедиться, что она нам точно подходит рассмотрим ее подробнее. В клиент серверной архитектуре имеется три звена:

- Представление данных — на стороне клиента.

- Прикладной компонент — на выделенном сервере приложений, здесь происходит вся бизнес-логика.

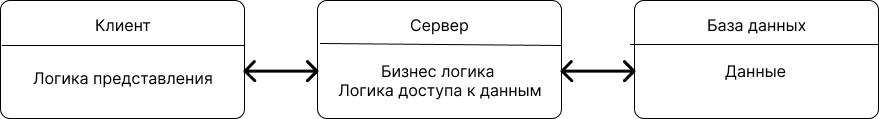
- Управление ресурсами — сервер БД, который и представляет запрашиваемые данные.

Рисунок 2.2. - Клиент-серверная трехзвенная архитектура Требования к системе разработаны в соответствии со стандартом качества программного обеспечения ISO 9126:2001, описывающим многоуровневую модель характеристик качества и соответствующий им набор атрибутов. Требования к Мебельный салон «МСВ» представлены в таблице 2.1.

Таблица 2.1. - Требования к Мебельному салону

|  |  |  |
| --- | --- | --- |
| Атрибуты | Метрики | Требования |
| 1 | 2 | 3 |
|  | Удобство |  |
| Простота использования | Среднее время, необходимое пользователю, чтобы найти элемент интерфейса | Поиск отдельного элемента интерфейса не должен занимать более 1 секунды |
| Привлекательность | Соответствие интерфейса требованиям заказчика | 97% элементов интерфейса должны быть адаптированы для пользователя и 99% для заказчика; |
| Обучаемость | Показатель, затрачиваемый пользователями на обучение работе с МСТ | Пользователь должен изучить работу ПО за первые 3-5 минут использования |
|  | Производительность |  |
| Временная эффективность | Время выполнения компонента программы | -Приложение должно обслуживать любую страницу не дольше 300 мсек (не включая задержки в сети); |

Продолжение таблицы 2.1.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Эффективность использования ресурсов | Объемы ресурсов требуемых для выполнения задач; | - Постоянная память, используемая программой – не более 100МБ;  -Приложение должно потреблять не более 4 Кбайт памяти на каждый неактивный сеанс с пользователем;  -Нагрузка на CPU и используемый объем жесткого диска на сервере баз данных не должны превышать 70%, а время обработки запросов не должно превышать 2 секунд;  -Нагрузка на CPU в режиме простоя приложения не должна превышать 1%;  -Время от возникновения ошибки до вывода предупреждения не должно превышать 25 мсек. |

Продолжение таблицы 2.1.

|  |  |  |
| --- | --- | --- |
|  | Переносимость |  |
| 1 | 2 | 3 |
| Удобство установки | Легкость установки | Необходима установка: Java, SQL Server, MSSQL |
| Адаптируемость | Способность ПО приспосабливаться к различным окружениям | -Программа должна корректно отображаться на всех доступных разрешениях экрана мониторов;  -Программа должна работать на любой версии Windows; |
| Способность к сосуществованию | Способность ПО сосуществовать с другими программами в общем окружении | Программа не должна занимать большие объемы памяти устройства (максимум 100МБ) |
| Удобство замены другого ПО данным | Совместимость структур | -Система должна быть разработана по общим принципам и не требовать навыков от пользователя; - Система должна быть применима вместо других программных систем для решения тех же задач в определенном окружении |

Продолжение таблицы 2.2.

|  |  |  |
| --- | --- | --- |
| Атрибуты | Метрики | Требования |
| Уровень зрелости | Среднее время работы без сбоев | -Система должна работать 24 часа в сутки;  -Система должна оповещать пользователя при некорректных действиях в системе;  -Данные, которые вводит пользователь должны сохраняться. |

Средняя доступность МСВ должна составлять не менее 99%

Среднее время между сбоями — это среднее время, за которое компонент или модуль может выполнять свои функции без перерыва. Измеряется от начала работы до момента следующего сбоя.

Среднее временя работы без сбоев должно составлять не менее 1500 часов.

В таблице 2.3. представлен перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

Таблица 2.3. - Перечень и критерии отказов для каждой функции

|  |  |
| --- | --- |
| Функция | Критерии отказа |
| 1 | 2 |
| Запуск программы | Отсутствие соединения с сервером и\или с БД. |
| Получить список товара. | Отсутствие соединения с сервером и\или с БД. |
| Получить список товара по названию | Отсутствие соединения с сервером и\или с БД. |
| Получить список товара по материалу | Отсутствие соединения с сервером и\или с БД. |
| Получить список товара по категории | Отсутствие соединения с сервером и\или с БД. |
| Получить список по количеству | Отсутствие соединения с сервером и\или с БД |
| Получить список клиентов | Отсутствие соединения с сервером и\или с БД |
| Получить список по имени | Отсутствие соединения с сервером и\или с БД |
| Получить список заказов | Отсутствие соединения с сервером и\или с БД |
| Получить список заказов по названию | Отсутствие соединения с сервером и\или с БД |

2.4.2. Требования к функциям выполняемым программным обеспечением

Имя: Запуск программы

Описание: Для того чтобы начать работать с системой пользователю нужно запустить программу

Действующий субъект: менеджер

Сценарий:

1. Система запускается и открывает главное окно приложения.

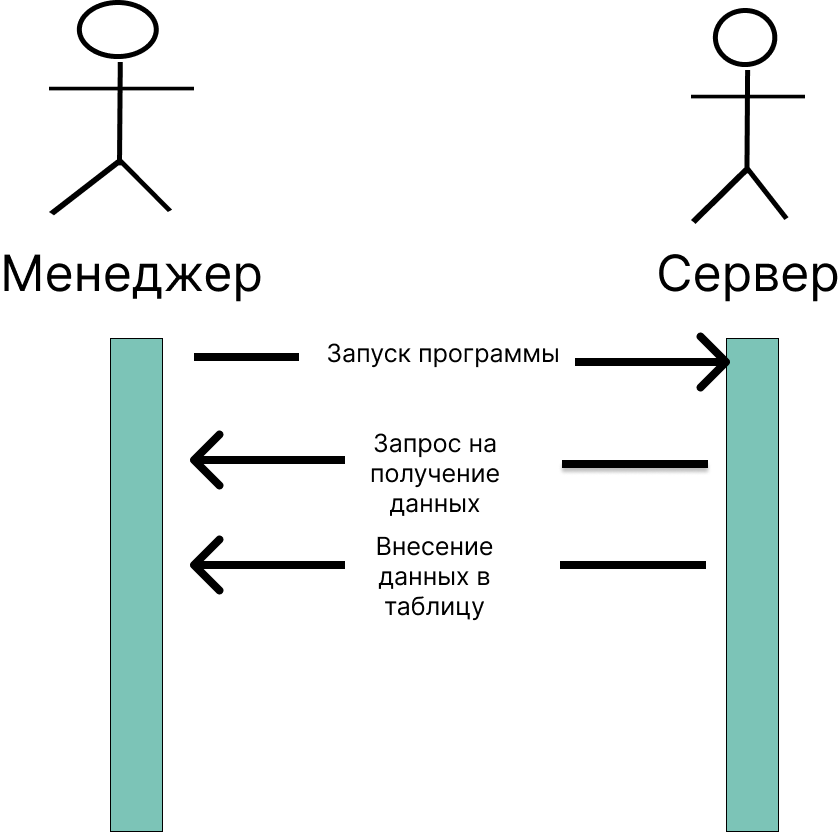


Рисунок 2.3. - Диаграмма последовательности сценария "Вход менеджера

Имя: Получение информации с сервера

Описание: менеджер имеет возможность получать различную информацию с

сервера.

Сценарий:

Система отображает запрошенную информацию в таблице

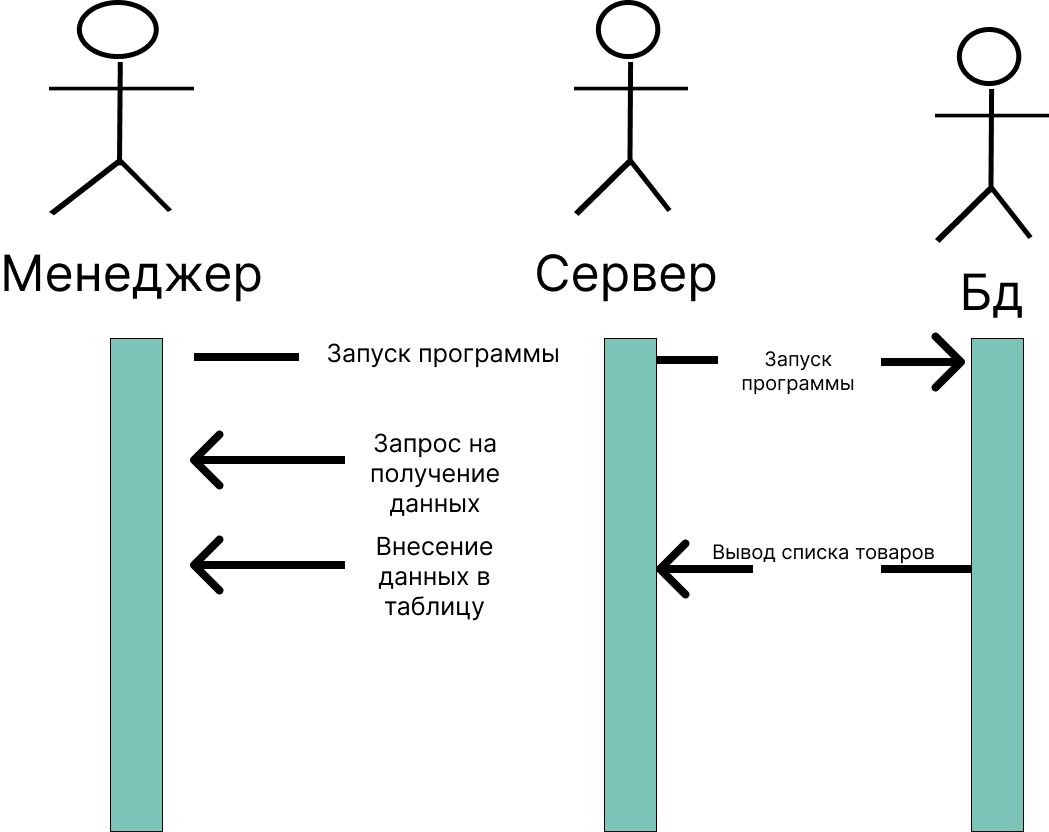


Рисунок 2.4. - Диаграмма последовательности сценария "Вход менеджера

1. Имя: Получить список товара в мебельном салоне по названию

Описание: менеджер имеет получить информацию о товаре по его имени

Сценарий:

Менеджер вводит в поисковик название нужного товара и система отображает его в таблице

2.4.3 Требования к видам обеспечения

- Обеспечить каждого специалиста персональным компьютером;

-Установить на каждый ПК программу;

Изучение персоналом функционал программы;

- Эксплуатация системы.

Для доступа к функционалу приложения необходим на конечном устройстве;

* Данное область для разработки программы были выделены следующие сущности, на основе которых будет разрабатываться база данных;

Разрабатываемая система требует в своей архитектуре наличие БД - сервера, или локального хоста запускаемого через xampp.

При анализе предметной области разрабатываемой системы были выделены следующие сущности:

* Products – таблица содержит информацию о продуктах
* Clients - таблица содержит информацию о клиентах
* Orders - таблица содержит информацию о заказах

Таблица 2.4. - Таблица products в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| Id\_product | Integer | 11 | Идентификатор продукта | Первичный ключ |
| category | Integer | 10 | категория | Not null |
| material | varchar | 40 | Материал | Not null |
| quanitity | varchar | 40 | Кол-во товара | Not null |

Таблица 2.5. - Таблица orders в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| Id\_order | Integer | 11 | Идентификатор заказа | Первичный ключ |
| Id\_client | Integer | 10 | Внешний ключ клиента | Not null |
| Id\_products | Integer | 10 | Внешний ключ продукта | Not null |
| Data | data |  | Дата заказа | Not null |

Таблица 2.6. - Таблица clients в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| Id\_clients | Integer | 11 | Идентификатор  пользователя | Первичный ключ |
| F\_name | Varchar | 10 | имя | Not null |
| L\_name | Varchar | 10 | Фамилия | Not null |
| S\_name | Integer | 11 | отчество | Not null |

2.4.4. Требования к эргономике и технической эстетике.

2.4.4.1. Требования к размещению элементов управления на

экранных формах

Общие требования:

- Фоновый цвет всех страниц –белый с серыми элементами.

- При наведении указателя мышки на элементы кнопок и ссылок,

Их цвет становится более темного оттенка.

- Любой текст программы можно выделить с помощью мыши.

Главная страница - тело

- Все элементы должны быть в таблице

-Цвет названия полей - черный

-Цвет границ таблицы- голубо - синий

-Кнопки имеют светло - синий цвет , черная окантовки границ кнопки

-Сверху в левом название программы

2.4.4.3. Требования к видам обеспечения

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

Требования для сервера БД:

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

- Оперативная память 8 Гб и выше

- Свободное дисковое пространство не менее 1 Гб

- HDD (жесткий диск), тип - SATA2, скорость вращения 7200

об/мин

Требования к клиентским устройствам:

Виртуальная машина JVM

Персональные компьютеры должны быть под управлением

Windows 10 или выше. Аппаратное обеспечение должно

соответствовать рекомендованными требованиями операционной

системы

Монитор на клиентских компьютерах должен быть с разрешением

не ниже 1680x1050 пикселей.

2.4.4.4 Эскизы основных окон системы



|  |  |  |  |
| --- | --- | --- | --- |
| Стадия работа | Выполняемые работы | Сроки | Итоги |
| 1 | 2 | 3 | 4 |
| Формирование требований | Обследование объектов автоматизации | Выполнено | Отчет о результатах обследования |
| Утверждение заказчиком ТЗ на создание системы |
| Проектирование | Разработка технического проекта на систему  Разработка прототипа системы | 28.04.2023  -  30.04.2023 | Спецификации программно- аппаратных средств системы |
| Разработка проектов организационно распорядительной,  программной и  эксплуатационной  документации на систему |
| Поставка программно-технических средств для эксплуатации | Поставка программно-технических средств для опытной эксплуатации на объектах автоматизации, входящих в состав опытной зоны | 01.05.2023  -  03.05.2023 | Акты; |
| Разработка программных средств | Разработка, отладка и тестирование программных средств | 04.05.2023  -  07.05.2023 | Комплект  Проектов  организационно-распорядительной, программной эксплуатационной документации на систему; |
| Приемка работ | Проведение предварительных испытаний на стенде исполнителя | 08.05.2023  -  12.05.2023 | Акт готовности подсистемы к развертыванию в опытной зоне. |

2.6. 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.

2.7. 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 2.4.3 «Требования к видам обеспечения».

Необходимое программное обеспечение для сервера:

- Операционная система Windows 10

- Сервер: MS SQL Server 2019

- СУБД: MSSQL для MS SQL Server.

2.8 Требования к документированию

Перечень подлежащих разработке комплектов и видов документов:

- Описание бизнес-процессов предметной области

- Техническое задание

- Эскизный проект

- Программа и методика испытаний

3.Эскизный проект

3.1. Общие положения

3.1.1 Полное наименование системы и ее условное обозначение

Полное наименование системы: Мебельный салон Вованчика.

Краткое наименование системы: МСВ.

3.1.2 Назначение системы

Разрабатываемая система предназначена для упрощения работы мебельного салона, выполненных по заказу.

-Быстрый поиск необходимых товаров;

-Удобства;

3.1.3 Цели создания системы

Разрабатываемая система предназначена для учета заказов в мебельном салоне

Система для учёта заказов, товаров, и клиентах позволит быстро просматривать необходимые данные.

Менеджер сможет с легкостью заносить информацию об товаре, клиентах и заказов, свободно редактировать и удалять, а также обеспечен удобный поиск по определенным параметрам.

Основными целями создания “Мебельный салон” являются:

- Список типов товара. все будет представлено на одной странице.

Для достижения поставленных целей система должна решать следующие задачи:

- наличие сформированной база данных различных товаров;

- добавление нового товара в список;

- редактирование товара в списке;

- удаление товара в списке;

На данном этапе разберем, то как будет вести себя программа при выполнение выделенных модулей. Для этого мы будем использовать диаграммы состояний в нотации UML.

На рисунке 3.1 представлена общая диаграмма состояний. При запуске программы, пользователь будет находиться в том, состоянии, когда ему необходимо авторизоваться, чтобы продолжить работу с системой и осуществлять деятельность согласно выделенным вариантам использования. 

Рисунок 3.1. - Общая диаграмма состояний программы

На рисунке 3.2 показано детальное описание работы пользователя с системой. Можно просматривать список всех товаров, редактировать, добавлять, удалять и переходить в другие вкладки.

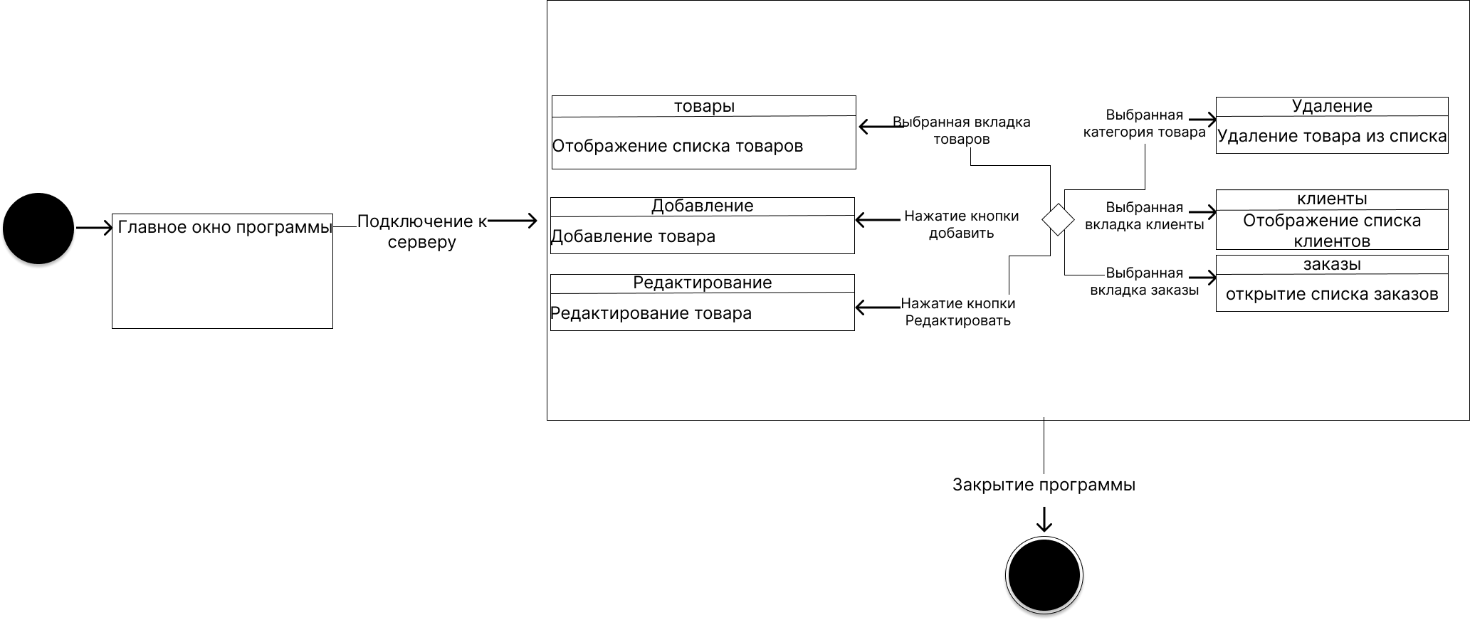


Рисунок 3.2 – Диаграмма состояния Просмотр списка заказов

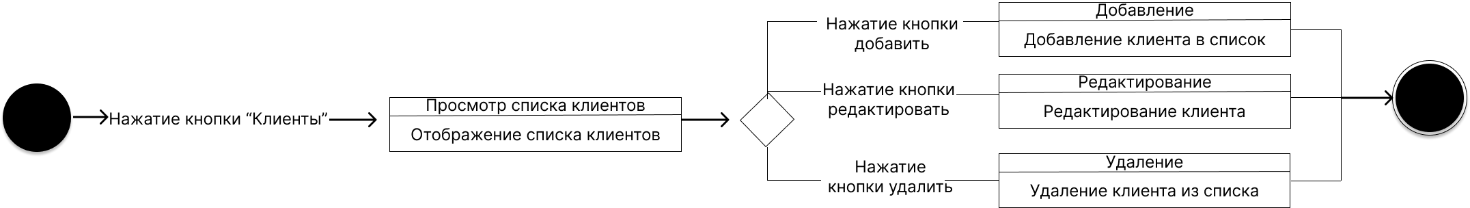
На рисунке 3.3, изображена декомпозиция состояния просмотра списка клиентов. После того как менеджер открыл программу, он может выбрать вкладку клиенты которая ему необходима. Далее он может добавлять, редактировать клиентов или же удалять.

Рисунок 3.3 – Диаграмма состояний "Просмотр списка заказов"

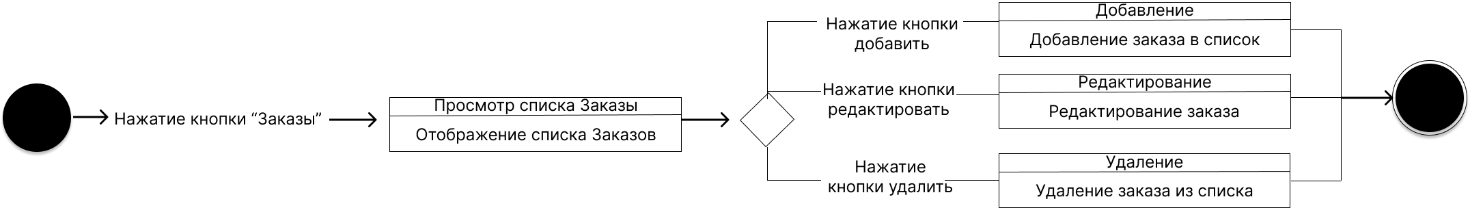
На рисунке 3.4, изображена декомпозиция состояния просмотра списка заказов. После того как менеджер открыл программу, он может выбрать вкладку клиенты которая ему необходима. Далее он может добавлять, редактировать заказы или же удалять.

Рисунок 3.4 – Диаграмма состояний "Просмотр списка клиентов"

На рисунке 3.5, изображена декомпозиция состояния просмотра списка товара. После того как менеджер открыл программу, он может выбрать вкладку клиенты которая ему необходима. Далее он может добавлять, редактировать товар или же удалять.

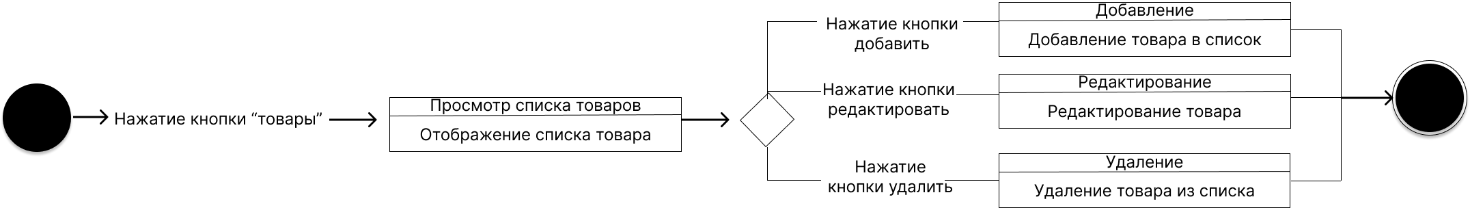


Рисунок 3.5 – Диаграмма состояний "Просмотр списка клиентов"

3.3 Основные технические решения

3.3.1 На данном этапе необходимо описать примерную архитектуру разрабатываемой системы.

Для того, чтобы сделать программу, мы будем использовать клиент-серверную архитектуру. Чтобы убедиться, что она нам точно подходит рассмотрим ее подробнее. В клиент серверной архитектуре имеется три звена:

- Представление данных — на стороне клиента.

- Прикладной компонент — на выделенном сервере приложений, здесь происходит вся бизнес-логика.

- Управление ресурсами — сервер БД, который и представляет запрашиваемые данные.

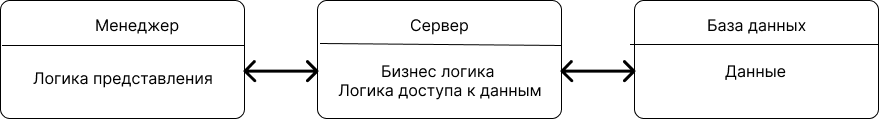
Схема клиент - серверной архитектуры представлена на рисунке 

Рисунок 3.6. - Клиент - серверная архитектура

Это одна из самых распространённых архитектур. Система делится на уровни, каждый из которых взаимодействует лишь с двумя соседними. Поэтому запросы к БД, которая обычно располагается в самом конце цепочки взаимодействия, проходят последовательно сквозь каждый «слой». Архитектура не подразумевает какое-то обязательное количество уровней — их может быть три, четыре, пять и больше.

Чаще всего используют трехзвенные системы: с уровнем представления (клиентом), уровнем логики и уровнем данных.

Такие архитектуры более разумно распределяют модули обработки данных, которые в этом случае выполняются на одном или нескольких отдельных серверах. Эти программные модули выполняют функции сервера для интерфейсов с пользователями и клиента - для серверов баз данных. Кроме того, различные серверы приложений могут взаимодействовать между собой для более точного разделения системы на функциональные блоки, выполняющие определенные роли. Схема модульной архитектуры представлена на рисунке 3.7.:



Рисунок 3.7. – Многоуровневая архитектура

Нашу систему можно разделить на отдельные модули, представленные на рисунке 3.8.:

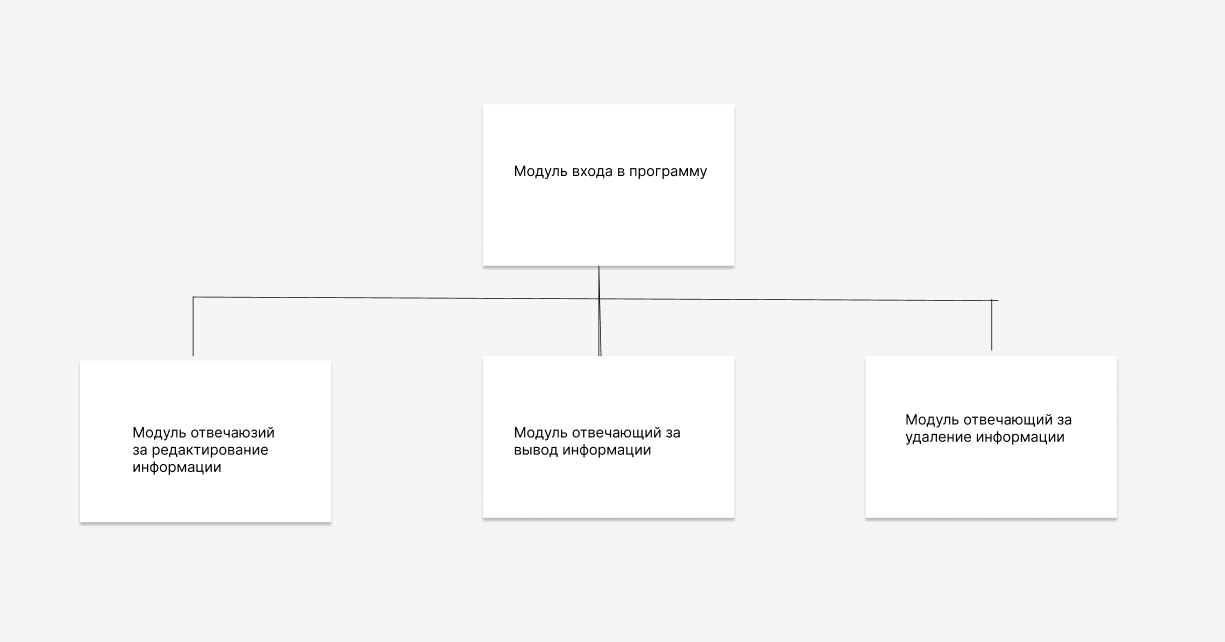


Рисунок 3.8. - Модули разрабатываемой системы

1.Модуль для входа программу. Благодаря нему, менеджер может зайти и пользоваться, введя свои данные.

2.Модуль получение и вывода информации. С помощью этого модуля пользователь может получать данные с БД, а так-же видеть информацию в таблице.

3.Модуль отвечающий за удаление информации. С помощью этого модуля пользователь может удалять нужную ему информацию из БД.

На рисунке 3.9. показана модель работы с системой, спроектированной по архитектуре клиент сервер.

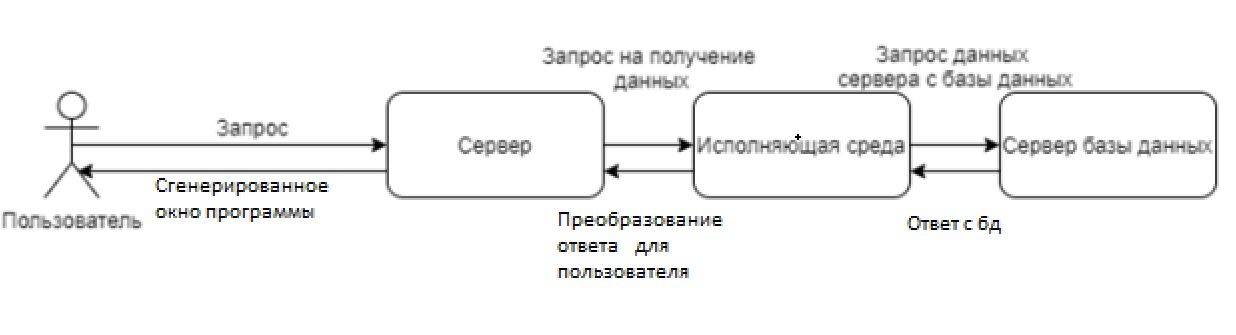


Рисунок 3.9. - Модель работы с системой, спроектированной по архитектуре клиент сервер

3.3.2. Решения по режимам функционирования, диагностированию работы системы

Система должна поддерживать работу в двух режимах:

- Основной режим, в котором все подсистемы выполняют свои основные функции;

- профилактический режим, все подсистемы не выполняют своих функций.

В основном режиме функционирования “МСВ” должна обеспечивать:

- Работу пользователей в режиме 24 часов 7 дней в неделю;

- Выполнение функций – сбор, обработка, загрузка и хранение данных.

В профилактическом режиме “МСВ” должна обеспечивать возможность проведения следующих работ:

- Модернизация серверной части ИС;

- Техническое обслуживание страниц-сайта и БД;

Средняя доступность МСВ должна составлять не менее 99%

Среднее время между сбоями — это среднее время, за которое компонент или модуль может выполнять свои функции без перерыва.

Измеряется от начала работы до момента следующего сбоя.

Среднее временя работы без сбоев должно составлять не менее 1500 часов.

3.3.3. Решения по численности, квалификации и функциям персонала АС, режимам его работы, порядку взаимодействия

В состав персонала, необходимого для обеспечения эксплуатации «МСВ» в рамках соответствующих подразделений Заказчика, необходимо выделение следующих ответственных лиц, представленных в таблице 3.1.:

Таблица 3.1. - Состав персонала, необходимого для обеспечения эксплуатации «МСВ»

|  |  |  |
| --- | --- | --- |
| Должность | Количество | Подсистема |
| Менеджер | 1 | Получение списка всех товаров, клиентов, заказов и получение информации |

Данные лицо должно выполнять следующие функциональные обязанности:

- Менеджер

- Работа со всеми функциями

3.3.4. Сведения об обеспечении заданных в техническом задании (ТЗ) потребительских характеристик системы (подсистем), определяющих ее качество

Обеспечение заявленных требований будет обеспечено следующим образом:

-Требования к надёжности обеспечиваются использованием отказоустойчивых решений, реализованных в составе Microsoft SQL Server.

-Требования к безопасности обеспечиваются использованием (возможностью использования) защищенных протоколов и средствами обеспечения безопасности, реализованными в Microsoft SQL Server.

-Требования к эргономике и технической эстетике обеспечиваются использованием JavaFx и CSS для создания графического интерфейса.

3.3.5. Состав функций, комплексов задач, реализуемых системой (подсистемой)

Выделенные варианты сценариев использования системы и соответствующие им диаграммы последовательностей приведены в разделе 2.4.2 «Требования к функциям, выполняемым ПО».

3.3.6. Решения по комплексу технических средств, его размещению на объекте

Для функционирования системы требуется наличие сервера и компьютера, на котором будет производится работа с программой. Перед вводом системы в действие необходимо:

- Интернет - соединение.

- ПК.

- JVM

- Эксплуатация системы.

3.3.7. Решения по составу информации, объему, способам ее организации, видам машинных носителей, входным и выходным документам и сообщениям, последовательности обработки информации и другим компонентам

Таблицы проектируемой базы данных, связи между ними, а также логическая схема база данных описана в разделе 2.4.3 «Требования к видам обеспечения».

Индекс это - структура данных, которая помогает MSSQL быстрее обнаружить отдельные записи в файле и сократить время выполнения запросов пользователей. Они повышают производительность СУБД таким образом, что если создать индекс по первичному ключу, а затем искать строку с данными, используя значения первичного ключа, то SQL-сервер сначала найдет значение индекса, а затем использует его для быстрого нахождения строки с данными. Без индекса было бы выполнено полное сканирование всех строк таблицы, что значительно бы потратило ресурсы. Задавая первичный ключ, автоматически устанавливается кластеризованный индекс.

Установим индексирование для полей в наших таблицах.

* Products – таблица содержит информацию о продуктах
* Clients - таблица содержит информацию о клиентах
* Orders - таблица содержит информацию о заказах

Таблица 3.2 – Таблица products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| Id\_product | Integer | 11 | Идентификатор продукта | Первичный ключ |
| category | Integer | 10 | категория | Not null |
| material | Varchar | 40 | Материал | Not null |
| quanitity | Varchar | 40 | Кол-во товара | Not null |

Таблица 3.3. - Таблица orders в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| Id\_order | Integer | 11 | Идентификатор заказа | Первичный ключ |
| Id\_client | Integer | 10 | Внешний ключ клиента | Not null |
| Id\_products | Integer | 10 | Внешний ключ продукта | Not null |
| Data | Data |  | Дата заказа | Not null |

Таблица 3.4. - Таблица clients в базе данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| Id\_clients | Integer | 11 | Идентификатор  пользователя | Первичный ключ |
| F\_name | Varchar | 10 | имя | Not null |
| L\_name | Varchar | 10 | Фамилия | Not null |
| S\_name | Integer | 11 | отчество | Not null |

3.3.8. Решения по составу программных средств, языкам деятельности, алгоритмам процедур и операций и методам их реализации

При разработке ИС использованы следующие программные компоненты:

- JavaFx интерфейс

- Хранилище данных создано на базе MSSQL

В качестве ОС клиентских устройств может использоваться:

- Microsoft Windows 10

Для реализации клиентской части могут использоваться:

- JavaFx;

- CSS;

Клиентское приложение должно быть разработано с применением объектно-ориентированной методологии, используя следующие паттерны проектирования:

- MVC;

- Одиночка;

- Итератор.

3.3.9. Эскиз пользовательского интерфейса МВС

Эскизы пользовательского интерфейса представлены в разделе 2.4.4.4 «Эскизы основных окон системы».

**4. РАЗРАБОТКА И ДОКУМЕНТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

4.1 Документирование API

Таблица1- Запросы для таблицы Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Адрес | Содержимое запроса | Вывод |
| post | orders/add | http://localhost:2825/api/v1/orders/add | {  "success": true,  "message":  "Заказ добавлен"  } |
| get | orders/all | http://localhost:2825/api/v1/orders/all | "success": true,  "message":  "Список заказов:",  "data": [  {  "id": 13,  "orders": "Дфвы",  "data":  "1221-12-12","clients": {  "id":20,  "name": "Фантом",  "lastname": "Ассасин",  "surname": "Асасинович  "products": {  "id":12,  "name": "Жфывыфвф  "products": "Дфыв",  "category": "Жыфв",  "material": "Двыфыв",  "qanitity": 1221 |
| delete | orders/delete/?id | http://localhost:2825/api/v1/orders/delete/?id=13 | {  "success": true,  "message":  "Заказ успешно удален" |
| get | orders/datetime?data= | http://localhost:2825/api/v1/orders/datetime?data=2004-12-12 | "success": true,  "message":  "Список заказов:",  "data": [  {  "id": 15,  "orders": "Жфы",  "data":  "2004-12-12",  "clients": {  "id": 26,  "name": "Нотдфывлдт","lastname": "Джагерфывлдо",  "surname": "Душныйфывлд"},  "products": {  "id": 20,  "products":  "Дфыв",  "category":  "Жыфв",  "material": "Двыфыв",  "qanitity": 1221 |
| post | orders/update | http://localhost:2825/api/v1/orders/update | {  "success": true,  "message":  "В заказ внесены изменения"  } |

Таблица 2 -Запросы для таблицы Products

|  |  |  |  |
| --- | --- | --- | --- |
| post | products/add | http://localhost:2825/api/v1/products/add | {  "success": true,  "message":  "Товар добавлен"  } |
| get | products/all | http://localhost:2825/api/v1/products/all | {  "success": true,  "message": "товар",  "data": [  "id": 20,  "products": "Дфыв",  "category": "Жыфв",  "material": "Двыфыв",  "qanitity": 1221 }]  } |
| delete | products/delete/?id= | http://localhost:2825/api/v1/products/delete/?id=21 | {  "success": true,  "message": "Удалено"  } |
| get | products/products?products= | http://localhost:2825/api/v1/products/products?products=Дфыв | {  "success": true,  "message": "товар",  "data": [  {  "id": 9,  "products": "Дфыв",  "category": "saddqw",  "material": "saddwq",  "qanitity": 123 }  ] |
| get | material?material= | http://localhost:2825/api/v1/products/material?material=saddwq | {  "success": true,  "message": "товар",  "data": [  {  "id":6,  "products": "saddqw",  "category": "saddqw",  "material": "saddwq",  "qanitity": 0  },  ]  } |
| get | products/category?category= | http://localhost:2825/api/v1/products/category?category=sadd | {  "success": true,  "message": "товар",  "data": [  {  "id": 1,  "name": "sadd",  "products": "sadd",  "category": "sadd",  "material": "sadd",  "qanitity": 0  }  ]  } |
| get | products/qanitity?qanitity= | http://localhost:2825/api/v1/products/qanitity?qanitity=123 | {  "success": true,  "message": "товар",  "data": [  {  "id": 8,  "name": "saddwq",  "products": "saddqw",  "category": "saddqw",  "material": "saddwq",  "qanitity": 123  },  }  ]  } |
| post | products/update | http://localhost:2825/api/v1/products/update | {  "success": true,  "message": "Товар изминен"  } |

Таблица3- Запросы для таблицы Clients

|  |  |  |  |
| --- | --- | --- | --- |
| post | clients/add | http://localhost:2825/api/v1/clients/add | {  "success": true,  "message":  "клиент добавлен"  } |
| get | clients/all | http://localhost:2825/api/v1/clients/all | {  "success": true,  "message": "клиенты",  "data": [  {  "id": 18,  "name":  "Луй",  "lastname": "Сыццуцв",  "surname": "Сывф" },  } |
| delete | clients/delete/?id= | http://localhost:2825/api/v1/clients/delete/?id=1 | {  "success": true,  "message": "клиент успешно удален"  } |

|  |  |  |  |
| --- | --- | --- | --- |
| get | clients/clients?name= | http://localhost:2825/api/v1/clients/clients?name=Фантом | {  "success": true,  "messag": "клиенты",  "data": [  {  "id": 19,  "name": "Фантом",  "lastname":   "Ассасин",  "surname": "Асасинович"  }  ]  } |
| post | clients/update | http://localhost:2825/api/v1/clients/update | {  "success": true,  "message":  "клиент изминен"  } |

**5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

В программе имеется функция поиска данных. Для поиска товара, клиентов, материала, категорий, количества товара и даты.

Чтобы начать поиск, нужной перейти в определенную таблицу(экран) которая есть в программе.У таблиц товаров, заказов и клиентов, есть свои поля поиска. То есть если вы перейдете на другую таблицу, то там будет поиск именно под активную таблицу (на которой находитесь в программе).



Рисунок 5.1 – руководство пользователя для поиска данных

В программе имеется функция добавления данных. Добавить новые данные в таблицу можно: товар, заказ, клиента. Чтобы вызвать окно добавления достаточно нажать на кнопку “Add” в определенном экране программы. При нажатии на кнопку, то откроется новое окно, где будут доступны поля для заполнения данных.

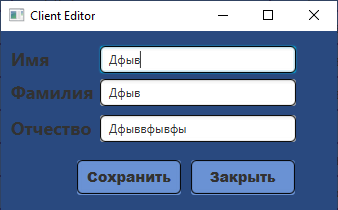


Рисунок 5.2 – руководство пользователя для добавления данных

В программе имеется функция редактирования данных. Редактировать информацию можно в таблице: товары, клиенты, заказы. Чтобы вызвать окно редактирования, нужно будет выбрать клиента, товар, или заказ, который будут редактировать. В случае не выбранного клиента , товара или же заказа, то программа выведет окно ошибки, что отсутствует выбранная клиент, заказ, или же товар.

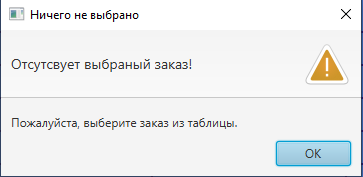


Рисунок 5.3 – руководство пользователя для редактирования данных

# **ЗАКЛЮЧЕНИЕ**

При работе с проектом были выполнены все поставленные цели и задачи. Удалось изучить область работы, составить техническое задание и эскизный проект. Так же удалось автоматизировать и оптимизировать работу данного предприятия. Решены все поставленные задачи. Результатом нашего проекта стала программа способная хранить и обрабатывать большое количество информации что существенно облегчит работу менеджеру предприятия. Программу можно обновлять и совершенствовать в зависимости от требований и задач.

Создание АИС способствует повышению эффективности производства экономического объекта и обеспечивает качество управления. Использование данной автоматизированной системы электронного документ оборота в реальных условиях приведёт к улучшению ряда экономических показателей:

­Улучшение временных показателей. Система позволяет сократить время на обработку и передачу данных, ускорить процессы принятия решений и реагирования на запросы. Это приводит к улучшению показателей времени отклика и сокращению времени выполнения задач;

Снижение затрат на обработку информации. Система позволяет сократить расходы на бумажные документы, их печать, хранение;

Повышение безопасности данных. Система позволяет обеспечить защиту информации от сторонних незащищённых источников.

**Список используемых источников**

1. Хантер Дэвид, XML. Базовый курс / Хантер Дэвид, Фаусетт Джо, Рафтер Джефф – пер.: Диалектика-Вильямс, 2018;
2. Руководство по SQLite [Электронный ресурс]. Режим доступа <https://www.sqlite.org/docs.html>;
3. Руководство по языку программирования Java [Электронный ресурс]. Режим доступа [https://metanit.com/java/tutorial](https://metanit.com/java/tutorial/);
4. Spring Boot – Spring по-русски! [Электронный ресурс]. Режим доступа http:spring-projects.ru/projects/spring-boot;
5. Кристиан Бауэр, Гэри Грегори, Гэвин Кинг, Java Persistence API и Hibernate – изд.: DMK Press, 2018;
6. Иан Ф. Дарвин, Java Cookbook: Problems and Solutions for Java Developers, изд.: O’Reilly, 2020
7. Кевлин Хенни, 97 Things every programmer should know: Collective wisdom from the experts – изд.: O’Reilly, 2020;
8. Бенджамин Дж. Еванс, Дэвид Фланаган, Java in a Nutshell: A Desktop Quick Reference, изд.: O’Reilly, 2020;
9. Javarush – онлайн-курс обучения программированию на Java [Электронный ресурс]. Режим доступа <https://javarush.com>;
10. Курс Java:обучение программированию на ява с нуля [Электронный ресурс]. Режим доступа https://code-basics.com;

**Приложение**

Приложение А-Листинг AddControllerController

public class AddClientsController {

@FXML

private TextField clientsLastname\_field;

@FXML

private TextField clientsName\_field;

@FXML

private TextField clientsSurname\_field;

private Stage editClientsStage;

private ClientsEntity clients;

private int clientsID;

private boolean okClicked = false;

public void setDialogStage (Stage dialogStage) {this.editClientsStage = dialogStage;}

@FXML

private void handleCancel() {editClientsStage.close();}

@FXML

private void handleOk() throws IOException {

if (isInputValid()) {

clients.setName(clientsName\_field.getText());

clients.setLastname(clientsLastname\_field.getText());

clients.setSurname(clientsSurname\_field.getText());

okClicked = true;

editClientsStage.close();

clientsData.set(clientsID, clients);

addClients(clients);

}

}

public void setLabels(ClientsEntity clientsIn, int clients\_id) {

this.clients = clientsIn;

this.clientsID = clients\_id;

clientsLastname\_field.setText(clients.getLastname());

clientsName\_field.setText(clients.getName());

clientsSurname\_field.setText(clients.getSurname());

}

private boolean isInputValid() {

String errorMessage = "";

try {

if (clientsName\_field.getText() == null || clientsName\_field.getText().length() == 0) errorMessage = "Не обнаружена Имя клиента!\n";

Приложение А-Листинг AddControllerController

if (!clientsName\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||clientsName\_field.getText() == null || clientsName\_field.getText().length() == 0) errorMessage += "Некорректное Имя!\n";

if (clientsLastname\_field.getText() == null || clientsLastname\_field.getText().length() == 0) errorMessage = "Не обнаружено Фамилия клиента!\n";

if (!clientsLastname\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||clientsLastname\_field.getText() == null || clientsLastname\_field.getText().length() == 0) errorMessage += "Некорректная Фамилия!\n";

if (clientsSurname\_field.getText() == null || clientsSurname\_field.getText().length() == 0) errorMessage = "Не обнаружено Отчество клиента !\n";

if (!clientsSurname\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||clientsSurname\_field.getText() == null || clientsSurname\_field.getText().length() == 0) errorMessage += "Некорректное Отчество!\n";

}catch (Exception e) {

System.out.println(e);

errorMessage += "Пустое поле!";

}

if (errorMessage.length() == 0) return true;

else {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.initOwner(editClientsStage);

alert.setTitle("Ошибка заполнения");

alert.setHeaderText("Пожалуйста, укажите корректные значения текстовых полей");

alert.setContentText(errorMessage);

alert.showAndWait();

return false;

}

}

public boolean isOkClicked(){return okClicked;}

public static void addClients(ClientsEntity clients) throws IOException {

System.out.println(clients.toString());

http.post(api+"clients/add", gson.toJson(clients).toString());

}

}

Приложение Б-Листинг AddProductsController

public class AddProductsController {

@FXML

private TextField productsProducts\_field;

@FXML

private TextField productsCategory\_field;

@FXML

private TextField productsMaterial\_field;

@FXML

private TextField productsQanitity\_field;

private Stage editProductsStage;

private ProductsEntity products;

private int productsID;

private boolean okClicked = false;

public void setDialogStage (Stage dialogStage) {this.editProductsStage = dialogStage;}

@FXML

private void handleCancel() {editProductsStage.close();}

@FXML

private void handleOk() throws IOException {

if (isInputValid()) {

products.setProducts(productsProducts\_field.getText());

products.setCategory(productsCategory\_field.getText());

products.setMaterial(productsMaterial\_field.getText());

products.setQanitity(productsQanitity\_field.getText());

okClicked = true;

editProductsStage.close();

productsData.set(productsID, products);

addProducts(products);

}

}

public void setLabels(ProductsEntity productsIn, int products\_id) {

this.products = productsIn;

this.productsID = products\_id;

productsProducts\_field.setText(products.getProducts());

productsCategory\_field.setText(products.getProducts());

productsMaterial\_field.setText(products.getProducts());

productsQanitity\_field.setText(products.getProducts());

}

private boolean isInputValid() {

String errorMessage = "";

try {

Приложение Б-Листинг AddProductsController

if (productsProducts\_field.getText() == null || productsProducts\_field.getText().length() == 0) errorMessage = "Не обнаружен !\n";

if (!productsProducts\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||productsProducts\_field.getText() == null || productsProducts\_field.getText().length() == 0) errorMessage += "Некорректное товар!\n";

if (productsCategory\_field.getText() == null || productsCategory\_field.getText().length() == 0) errorMessage = "Не обнаружен !\n";

if (!productsCategory\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||productsCategory\_field.getText() == null || productsCategory\_field.getText().length() == 0) errorMessage += "Некорректное категория!\n";

if (productsMaterial\_field.getText() == null || productsMaterial\_field.getText().length() == 0) errorMessage = "Не обнаружен !\n";

if (!productsMaterial\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||productsMaterial\_field.getText() == null || productsMaterial\_field.getText().length() == 0) errorMessage += "Некорректное материал!\n";

if (productsQanitity\_field.getText() == null || productsQanitity\_field.getText().length() == 0) errorMessage = "Не обнаружен !\n";

if (!productsQanitity\_field.getText().matches("[\\d0-9]{1,4}") ||productsQanitity\_field.getText() == null || productsQanitity\_field.getText().length() == 0) errorMessage += "Некорректное количество!\n";

}catch (Exception e){

System.out.println(e);

errorMessage += "Пустое поле!";

}

if (errorMessage.length() == 0) return true;

else {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.initOwner(editProductsStage);

alert.setTitle("Ошибка заполнения");

alert.setHeaderText("Пожалуйста, укажите корректные значения текстовых полей");

alert.setContentText(errorMessage);

alert.showAndWait();

return false;

}

}

public boolean isOkClicked(){return okClicked;}

public static void addProducts(ProductsEntity products) throws IOException {

System.out.println(products.toString());

http.post(api+"products/add", gson.toJson(products).toString());

}

}

Приложение В-Листинг EditClientsController

public class EditClientsController {

@FXML

private TextField clientsLastname\_field;

@FXML

private TextField clientsName\_field;

@FXML

private TextField clientsSurname\_field;

private Stage editClientsStage;

private ClientsEntity clients;

private int clientsID;

private boolean okClicked = false;

public void setDialogStage (Stage dialogStage) {this.editClientsStage = dialogStage;}

@FXML

private void handleCancel() {editClientsStage.close();}

@FXML

private void handleOk() throws IOException {

if (isInputValid()) {

clients.setLastname(clientsLastname\_field.getText());

clients.setName(clientsName\_field.getText());

clients.setSurname(clientsSurname\_field.getText());

okClicked = true;

editClientsStage.close();

clientsData.set(clientsID, clients);

updateClients(clients);

}

}

public void setLabels(ClientsEntity clientsIn, int clients\_id) {

this.clients = clientsIn;

this.clientsID = clients\_id;

clientsLastname\_field.setText(clients.getLastname());

clientsName\_field.setText(clients.getName());

clientsSurname\_field.setText(clients.getSurname());

}

private boolean isInputValid() {

String errorMessage = "";

try {

if (clientsName\_field.getText() == null || clientsName\_field.getText().length() == 0)

errorMessage = "Необнаружена Имя клиента!\n";

Приложение В-Листинг EditClientsController

if (!clientsName\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||clientsName\_field.getText() == null || clientsName\_field.getText().length() == 0) errorMessage += "Некорректное Имя!\n";

if (clientsLastname\_field.getText() == null || clientsLastname\_field.getText().length() == 0) errorMessage = "Не обнаружено Фамилия клиента!\n";

if (!clientsLastname\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||clientsLastname\_field.getText() == null || clientsLastname\_field.getText().length() == 0) errorMessage += "Некорректная Фамилия!\n";

if (clientsSurname\_field.getText() == null || clientsSurname\_field.getText().length() == 0) errorMessage = "Не обнаружено Отчество клиента !\n";

if (!clientsSurname\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||clientsSurname\_field.getText() == null || clientsSurname\_field.getText().length() == 0) errorMessage += "Некорректное Отчество!\n";

}catch (Exception e) {

System.out.println(e);

errorMessage += "Пустое поле!";

}

if (errorMessage.length() == 0) return true;

else {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.initOwner(editClientsStage);

alert.setTitle("Ошибка заполнения");

alert.setHeaderText("Пожалуйста, укажите корректные значения текстовых полей");

alert.setContentText(errorMessage);

alert.showAndWait();

return false;

}

}

public boolean isOkClicked(){return okClicked;}

public static void updateClients(ClientsEntity clients) throws IOException {

System.out.println(clients.toString());

http.post(api+"clients/update", gson.toJson(clients).toString());

}

}

Приложение Г-Листинг EditOrdersController

public class EditOrdersController {

@FXML

private TextField ordersName\_field;

@FXML

private ComboBox<ClientsEntity> ordersNameClient\_field;

@FXML

private ComboBox<ProductsEntity> ordersNameProducts\_field;

@FXML

private TextField orderData\_field;

private Stage editOrderStage;

private OrdersEntity orders;

private int ordersID;

private boolean okClicked = false;

private ObservableList<ClientsEntity> clientsOrderData = FXCollections.observableArrayList();

private ObservableList<ProductsEntity> productsOrderData = FXCollections.observableArrayList();

public void setDialogStage (Stage dialogStage) {this.editOrderStage = dialogStage;}

@FXML

private void handleCancel() {editOrderStage.close();}

@FXML

private void handleOk() throws IOException {

if (isInputValid()) {

orders.setOrders(ordersName\_field.getText());

orders.setClients(ordersNameClient\_field.getValue());

orders.setProducts(ordersNameProducts\_field.getValue());

orders.setData(orderData\_field.getText());

okClicked = true;

editOrderStage.close();

ordersData.set(ordersID, orders);

updateOrders(orders);

}

}

private void updateMap(){

clientsOrderData.addAll(clientsData);

productsOrderData.addAll(productsData);

}

@FXML

void initialize() throws Exception {

if (clientsOrderData.size() != clientsData.size() || productsOrderData.size() != productsData.size()){

updateMap();

Приложение Г-Листинг EditOrdersController

updateComboBox();

}

private void updateComboBox() throws Exception {

ordersNameClient\_field.getItems().addAll(clientsOrderData);

ordersNameProducts\_field.getItems().addAll(productsOrderData);

}

private boolean isInputValid() {

String errorMessage = "";

if (!ordersName\_field.getText().matches("^[A-ЯЁ][а-яё]+$") ||ordersName\_field.getText() == null || ordersName\_field.getText().length() == 0) errorMessage += "Некорректное значение названия заказа!\n";

if (ordersNameClient\_field.getValue() == null || ordersNameClient\_field.getValue().hashCode() == 0) errorMessage += "Не обнаружено Имя клиента!\n";

if (ordersNameProducts\_field.getValue() == null || ordersNameProducts\_field.getValue().hashCode() == 0) errorMessage += "Не обнаружен Товар!\n";

if (orderData\_field.getText() == null || orderData\_field.getText().length() == 0) errorMessage += "Не обнаружена дата заказа!\n";

if (!orderData\_field.getText().matches("[0-9]{4}-[0-9]{2}-[0-9]{2}") || orderData\_field.getText() == null) errorMessage += "Дата введенна некорректно! \n";

if (errorMessage.length() == 0) return true;

else {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.initOwner(editOrderStage);

alert.setTitle("Oops!");

alert.setHeaderText("Пожалуйста, укажите корректные значения текстовых полей");

alert.setContentText(errorMessage);

alert.showAndWait();

return false;

}

}

public void setLabels(OrdersEntity ordersIn, int orders\_id) {

this.orders = ordersIn;

this.ordersID = orders\_id;

ordersName\_field.setText(orders.getOrders());

ordersNameClient\_field.setValue(orders.getClients());

ordersNameProducts\_field.setValue(orders.getProducts());

orderData\_field.setText(String.valueOf(orders.getData()));

}

Приложение Д-Листинг EditProductsController

public class EditProductsController {

@FXML

private TextField productsProducts\_field;

@FXML

private TextField categoryProducts\_field;

@FXML

private TextField materialProducts\_field;

@FXML

private TextField qanitityProducts\_field;

private Stage editProductsStage;

private ProductsEntity products;

private int productsID;

private boolean okClicked = false;

public void setDialogStage (Stage dialogStage) {this.editProductsStage = dialogStage;}

@FXML

private void handleCancel() {editProductsStage.close();}

@FXML

private void handleOk() throws IOException {

if (isInputValid()) {

products.setProducts(productsProducts\_field.getText());

products.setCategory(categoryProducts\_field.getText());

products.setMaterial(materialProducts\_field.getText());

products.setQanitity(qanitityProducts\_field.getText());

okClicked = true;

editProductsStage.close();

productsData.set(productsID, products);

updateProducts(products);

}

}

public void setLabels(ProductsEntity productsIn, int products\_id) {

this.products = productsIn;

this.productsID = products\_id;

productsProducts\_field.setText(products.getProducts());

categoryProducts\_field.setText(products.getCategory());

materialProducts\_field.setText(products.getMaterial());

qanitityProducts\_field.setText(products.getQanitity());

}

private boolean isInputValid() {

String errorMessage = "";

Приложение Д-Листинг EditProductsController

try {

if (productsProducts\_field.getText() == null || productsProducts\_field.getText().length() == 0)

errorMessage = "Не обнаружен !\n";

if (!productsProducts\_field.getText().matches("^[A-ЯЁ][а-яё]+$") || productsProducts\_field.getText() == null || productsProducts\_field.getText().length() == 0)

errorMessage += "Некорректно введено Название товара начинается с заглавной буквы и на русском!\n";

if (categoryProducts\_field.getText() == null || categoryProducts\_field.getText().length() == 0)

errorMessage = "Не обнаружен !\n";

if (!categoryProducts\_field.getText().matches("^[A-ЯЁ][а-яё]+$") || categoryProducts\_field.getText() == null || categoryProducts\_field.getText().length() == 0)

errorMessage += "Некорректна введена категория, Она должна начинается с заглавной буквы и на русском!\n";

if (materialProducts\_field.getText() == null || materialProducts\_field.getText().length() == 0)

errorMessage = "Не обнаружен !\n";

if (!materialProducts\_field.getText().matches("^[A-ЯЁ][а-яё]+$") || materialProducts\_field.getText() == null || materialProducts\_field.getText().length() == 0)

errorMessage += "Некорректно введен материал, Он должен начинается с заглавной буквы и на русском !\n";

if (qanitityProducts\_field.getText() == null || qanitityProducts\_field.getText().length() == 0)

errorMessage = "Не обнаружен !\n";

if (!qanitityProducts\_field.getText().matches("[\\d1-9]{1,4}") ||qanitityProducts\_field.getText() == null || qanitityProducts\_field.getText().length() == 0)

errorMessage += "Некорректно введено количество, должно начинаться с заглавной буквы и на русском!\n";

}catch (Exception e){

System.out.println(e);

errorMessage += "Пустое поле!";

}

if (errorMessage.length() == 0) return true;

else {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.initOwner(editProductsStage);

alert.setTitle("Ошибка заполнения");

alert.setHeaderText("Пожалуйста, укажите корректные значения текстовых полей");

alert.setContentText(errorMessage);

alert.showAndWait();

return false;

}

}

Приложение Д-Листинг EditProductsController

public boolean isOkClicked(){return okClicked;}

public static void updateProducts(ProductsEntity products) throws IOException {

http.put(api+"products/update", gson.toJson(products).toString());

}

}

Приложение Е-Листинг HelloController

public class HelloController {

public static String api = "http://localhost:2825/api/v1/";

public static ObservableList<ClientsEntity> clientsData = FXCollections.observableArrayList();

public static ObservableList<ProductsEntity> productsData = FXCollections.observableArrayList();

public static ObservableList<OrdersEntity> ordersData = FXCollections.observableArrayList();

public static ObservableList<ProductsEntity> searchDataProducts = FXCollections.observableArrayList();

public static ObservableList<ProductsEntity> searchDataMaterial = FXCollections.observableArrayList();

public static ObservableList<ProductsEntity> searchDataCategory = FXCollections.observableArrayList();

public static ObservableList<ProductsEntity> searchDataQanitity = FXCollections.observableArrayList();

public static ObservableList<ClientsEntity> searchDataName = FXCollections.observableArrayList();

public static ObservableList<OrdersEntity> searchDataOrders = FXCollections.observableArrayList();

static HTTPUtils http = new HTTPUtils();

static Gson gson = new Gson();

@FXML

private TextField SearchFieldOrders;

@FXML

private TextField SearchFieldName;

@FXML

private TextField SearchFieldProducts;

@FXML

private TextField SearchFieldCategory;

@FXML

private TextField SearchFieldMaterial;

@FXML

private TextField SearchFielQanitity;

@FXML

public TableView<ClientsEntity> tableClients;

@FXML

private TableColumn<ClientsEntity, String> clientsName;

@FXML

private TableColumn<ClientsEntity, String> clientsLastname;

@FXML

private TableColumn<ClientsEntity, String> clientsSurname;

@FXML

public TableView<ProductsEntity> tableProducts;

@FXML

private TableColumn<ProductsEntity, String> productsName;

@FXML

private TableColumn<ProductsEntity, String> productsProducts;

@FXML

private TableColumn<ProductsEntity, String> productsCategory;

@FXML

private TableColumn<ProductsEntity, String> productsMaterial;

@FXML

private TableColumn<ProductsEntity, String> productsQanitity;

public TableView<OrdersEntity> tableOrders;

@FXML

private TableColumn<OrdersEntity, String> ordersName;

@FXML

private TableColumn<OrdersEntity, String> ordersNameClient;

@FXML

private TableColumn<OrdersEntity, String> ordersNameProducts;

@FXML

private TableColumn<OrdersEntity, String> orderData;

@FXML

private void initialize() throws Exception {

getDataClients();

getDataProducts();

getDataOrders();

updateTableOrders();

updateTableClients();

updateTableProducts();

}

private void updateTableClients() throws Exception {

clientsName.setCellValueFactory(new PropertyValueFactory<ClientsEntity, String>("name"));

clientsSurname.setCellValueFactory(new PropertyValueFactory<ClientsEntity, String>("surname"));

clientsLastname.setCellValueFactory(new PropertyValueFactory<ClientsEntity, String>("lastname"));

tableClients.setItems(clientsData);

}

private void updateTableProducts() throws Exception {

productsProducts.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("products"));

productsCategory.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("category"));

productsMaterial.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("material"));

productsQanitity.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("qanitity"));

tableProducts.setItems(productsData);

}

Приложение Е-Листинг HelloController

private void updateTableOrders() throws Exception {

ordersName.setCellValueFactory(new PropertyValueFactory<OrdersEntity, String>("orders"));

ordersNameClient.setCellValueFactory(new PropertyValueFactory<OrdersEntity, String>("clients"));

ordersNameProducts.setCellValueFactory(new PropertyValueFactory<OrdersEntity, String>("products"));

orderData.setCellValueFactory(new PropertyValueFactory<OrdersEntity, String>("data"));

tableOrders.setItems(ordersData);

}

@FXML

private void SearchDataProducts() throws IOException{

String tov = http.get("http://localhost:2825/api/v1/products/products?products=", SearchFieldProducts.getText());

System.out.println(tov);

JsonObject base = gson.fromJson(tov, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for(int i = 0; i < dataArr.size();i++){

ProductsEntity newpro = gson.fromJson(dataArr.get(i).toString(),ProductsEntity.class);

searchDataProducts.add(newpro);

}

//productsName.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("name"));

productsProducts.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("products"));

productsCategory.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("category"));

productsMaterial.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("material"));

productsQanitity.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("qanitity"));

tableProducts.setItems(searchDataProducts);

}

@FXML

private void SearchDataCategory() throws IOException{

String cat = http.get("http://localhost:2825/api/v1/products/category?category=", SearchFieldCategory.getText());

System.out.println(cat);

JsonObject base = gson.fromJson(cat, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for(int i = 0; i < dataArr.size();i++){

ProductsEntity newpro = gson.fromJson(dataArr.get(i).toString(),ProductsEntity.class);

searchDataCategory.add(newpro);

}

productsProducts.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("products"));

productsCategory.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("category"));

productsMaterial.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("material"));

Приложение Е-Листинг HelloController

productsQanitity.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("qanitity"));

tableProducts.setItems(searchDataCategory);

}

@FXML

private void SearchDataMaterial() throws IOException{

String rov = http.get("http://localhost:2825/api/v1/products/material?material=", SearchFieldMaterial.getText());

System.out.println(rov);

JsonObject base = gson.fromJson(rov, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for(int i = 0; i < dataArr.size();i++){

ProductsEntity nepro = gson.fromJson(dataArr.get(i).toString(),ProductsEntity.class);

searchDataMaterial.add(nepro);

}

productsProducts.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("products"));

productsCategory.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("category"));

productsMaterial.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("material"));

productsQanitity.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("qanitity"));

tableProducts.setItems(searchDataMaterial);

}

@FXML

private void SearchDataOrders() throws IOException{

String rov = http.get("http://localhost:2825/api/v1/orders/orders?orders=", SearchFieldOrders.getText());

System.out.println(rov);

JsonObject base = gson.fromJson(rov, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for(int i = 0; i < dataArr.size();i++){

OrdersEntity nepro = gson.fromJson(dataArr.get(i).toString(),OrdersEntity.class);

searchDataOrders.add(nepro);

}

ordersName.setCellValueFactory(new PropertyValueFactory<OrdersEntity, String>("orders"));

orderData.setCellValueFactory(new PropertyValueFactory<OrdersEntity, String>("data"));

ordersNameClient.setCellValueFactory(new PropertyValueFactory<OrdersEntity, String>("clients"));

ordersNameProducts.setCellValueFactory(new PropertyValueFactory<OrdersEntity, String>("products"));

tableOrders.setItems(searchDataOrders);

}

@FXML

private void SearchDataClients() throws IOException{

Приложение Е-Листинг HelloController

String rov = http.get("http://localhost:2825/api/v1/clients/clients?name=", SearchFieldName.getText());

System.out.println(rov);

JsonObject base = gson.fromJson(rov, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for(int i = 0; i < dataArr.size();i++){

ClientsEntity nepr = gson.fromJson(dataArr.get(i).toString(),ClientsEntity.class);

searchDataName.add(nepr);

}

clientsName.setCellValueFactory(new PropertyValueFactory<ClientsEntity, String>("name"));

clientsSurname.setCellValueFactory(new PropertyValueFactory<ClientsEntity, String>("surname"));

clientsLastname.setCellValueFactory(new PropertyValueFactory<ClientsEntity, String>("lastname"));

tableClients.setItems(searchDataName);

}

@FXML

private void SearchDataQanitity() throws IOException{

String rov = http.get("http://localhost:2825/api/v1/products/qanitity?qanitity=", SearchFielQanitity.getText());

System.out.println(rov);

JsonObject base = gson.fromJson(rov, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for(int i = 0; i < dataArr.size();i++){

ProductsEntity nepro = gson.fromJson(dataArr.get(i).toString(),ProductsEntity.class);

searchDataQanitity.add(nepro);

}

productsProducts.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("products"));

productsCategory.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("category"));

productsMaterial.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("material"));

productsQanitity.setCellValueFactory(new PropertyValueFactory<ProductsEntity, String>("qanitity"));

tableProducts.setItems(searchDataQanitity);

}

@FXML

void refrashtableProducts(ActionEvent event) throws Exception{

tableProducts.getItems().clear();

updateTableProducts();

tableProducts.setItems(productsData);

}

@FXML

void refrashtableOrders(ActionEvent event) throws Exception{

tableOrders.getItems().clear();

updateTableOrders();

Приложение Е-Листинг HelloController

tableOrders.setItems(ordersData);

}

@FXML

void refrashtableClients(ActionEvent event) throws Exception{

tableClients.getItems().clear();

updateTableClients();

tableClients.setItems(clientsData);

}

@FXML

private void click\_newClient() throws Exception {

ClientsEntity tempClients = new ClientsEntity();

clientsData.add(tempClients);

HelloApplication.showClientsAddDialog(tempClients, clientsData.size()-1);

tableClients.getItems().clear();

getDataClients();

}

@FXML

private void click\_newProducts() throws Exception {

ProductsEntity tempProducts = new ProductsEntity();

productsData.add(tempProducts);

HelloApplication.showProductsAddDialog(tempProducts, productsData.size()-1);

tableProducts.getItems().clear();

getDataProducts();

}

@FXML

private void click\_newOrders() throws IOException {

OrdersEntity tempOrder = new OrdersEntity();

ordersData.add(tempOrder);

HelloApplication.showPersonEditDialog(tempOrder, ordersData.size()-1);

}

@FXML

private void click\_editClients() {

ClientsEntity selectedClient = tableClients.getSelectionModel().getSelectedItem();

if (selectedClient != null)

HelloApplication.showClientsEditDialog(selectedClient, clientsData.indexOf(selectedClient));

else {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setTitle("Ничего не выбрано");

Приложение Е-Листинг HelloController

alert.setHeaderText("Отсутсвует выбраный клиент!");

alert.setContentText("Пожалуйста, выберите клиента из таблицы.");

alert.showAndWait();

}

}

@FXML

private void click\_editOrders() {

OrdersEntity selectedOrders = tableOrders.getSelectionModel().getSelectedItem();

if (selectedOrders != null)

HelloApplication.showPersonEditDialog(selectedOrders, ordersData.indexOf(selectedOrders));

else {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setTitle("Ничего не выбрано");

alert.setHeaderText("Отсутсвует выбраный заказ!");

alert.setContentText("Пожалуйста, выберите заказ из таблицы.");

alert.showAndWait();

}

}

@FXML

private void click\_removeClients() throws IOException {

ClientsEntity selectedClient = tableClients.getSelectionModel().getSelectedItem();

if (selectedClient != null) {

System.out.println(selectedClient.getId());

System.out.println(http.delete(api+"clients/delete/?id=", selectedClient.getId()));

clientsData.remove(selectedClient);

} else {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setTitle("Ничего не выбрано");

alert.setHeaderText("Отсутсвует выбраный клиент!");

alert.setContentText("Пожалуйста, выберите клиента из таблицы.");

alert.showAndWait();

}

}

@FXML

private void click\_removeProducts() throws IOException {

ProductsEntity selectedProducts = tableProducts.getSelectionModel().getSelectedItem();

if (selectedProducts != null) {

System.out.println(selectedProducts.getId());

System.out.println(http.delete(api+"products/delete/?id=", selectedProducts.getId()));

Приложение Е-Листинг HelloController

productsData.remove(selectedProducts);

} else {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setTitle("Ничего не выбрано");

alert.setHeaderText("Отсутсвует выбраный товар!");

alert.setContentText("Пожалуйста, выберите товар из таблицы.");

alert.showAndWait();

}

}

@FXML

private void click\_removeOrders() throws IOException {

OrdersEntity selectedOrder = tableOrders.getSelectionModel().getSelectedItem();

if (selectedOrder != null) {

System.out.println(selectedOrder.getId());

System.out.println(http.delete(api+"orders/delete/?id=", selectedOrder.getId()));

ordersData.remove(selectedOrder);

} else {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setTitle("Ничего не выбрано");

alert.setHeaderText("Отсутсвует выбраный заказ!");

alert.setContentText("Пожалуйста, выберите заказ из таблицы");

alert.showAndWait();

}

}

@FXML

private void click\_editProducts() {

ProductsEntity selectedProducts = tableProducts.getSelectionModel().getSelectedItem();

if (selectedProducts != null)

HelloApplication.showProductsEditDialog(selectedProducts, productsData.indexOf(selectedProducts));

else {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setTitle("Ничего не выбрано");

alert.setHeaderText("Отсутсвует выбраный издатель!");

alert.setContentText("Пожалуйста, выберите издателя из таблицы.");

alert.showAndWait();

}

}

public static void getDataClients() throws Exception {

String res = http.get(api,"clients/all");

System.out.println(res);

Приложение Е-Листинг HelloController

JsonObject base = gson.fromJson(res, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for (int i = 0; i < dataArr.size(); i++) {

ClientsEntity newClient = gson.fromJson(dataArr.get(i).toString(), ClientsEntity.class);

clientsData.add(newClient);

}

}

public static void getDataProducts() throws Exception {

String res = http.get(api,"products/all");

System.out.println(res);

JsonObject base = gson.fromJson(res, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for (int i = 0; i < dataArr.size(); i++) {

ProductsEntity newProducts = gson.fromJson(dataArr.get(i).toString(), ProductsEntity.class);

productsData.add(newProducts);

}

}

public static void getDataOrders() throws Exception {

String res = http.get(api,"orders/all");

System.out.println(res);

JsonObject base = gson.fromJson(res, JsonObject.class);

JsonArray dataArr = base.getAsJsonArray("data");

for (int i = 0; i < dataArr.size(); i++) {

OrdersEntity newOrders = gson.fromJson(dataArr.get(i).toString(), OrdersEntity.class);

ordersData.add(newOrders);

}

}

public static void updateClients(ClientsEntity clients) throws IOException {

http.post(api+"clients/update", gson.toJson(clients).toString());

}

public static void updateOrders(OrdersEntity orders) throws IOException {

http.post(api+"orders/update", gson.toJson(orders).toString());

}

}

Приложение Ё-Листинг ClientsEntity

@Data

public class ClientsEntity {

private long id;

private String name;

private String surname;

private String lastname;

@Override

public String toString(){

return surname +' '+ name.toUpperCase().charAt(0) + '.' + lastname.toUpperCase().charAt(0) +'.';

}

}

Приложение Ж-Листинг OrdersEntity

@Data

public class OrdersEntity {

private Long id;

private String orders;

private String data;

private ClientsEntity clients;

private ProductsEntity products;

}

Приложение З-Листинг ProductsEntity

@Data

public class ProductsEntity {

private Long id;

private String name;

private String products;

private String category;

private String material;

private String qanitity;

@Override

public String toString() {

return products;

}

}

Приложение И-Листинг BaseResponse

@Data

@AllArgsConstructor

@RequiredArgsConstructor

public class BaseResponse {

protected boolean success;

protected String message;

}

Приложение Й-Листинг ClientsListResponse

@Data

public class ClientsListResponse extends BaseResponse {

public ClientsListResponse(List<ClientsEntity> data) {

super(true, "Clients");

this.data = data;

}

private List<ClientsEntity> data;

}

Приложение К-Листинг ClientsResponse

@Data

public class ClientsResponse extends BaseResponse {

public ClientsResponse(boolean success, String message, ClientsEntity clients) {

super(success, message);

this.clients = clients;

}

public ClientsResponse(ClientsEntity clients) {

super(true,"Author Data");

}

private ClientsEntity clients;

}

Приложение Л-Листинг OrdersListResponse

@Data

public class OrdersListResponse extends BaseResponse {

public OrdersListResponse(List<OrdersEntity> data) {

super(true, "Orders");

this.data = data;

}

private List<OrdersEntity> data;

}

Приложение М-Листинг OrdersResponse

@Data

public class OrdersResponse extends BaseResponse {

public OrdersResponse(boolean success, String message, OrdersEntity orders) {

super(success, message);

this.orders = orders;

}

public OrdersResponse(OrdersEntity products){

super(true,"Orders Data");

}

private OrdersEntity orders;

}

Приложение Н-Листинг ProductsListResponse

@Data

public class ProductsListResponse extends BaseResponse {

public ProductsListResponse(List<ProductsEntity> data) {

super(true, "Products");

this.data = data;

}

private List<ProductsEntity> data;

}

Приложение О-Листинг ProductsResponse

@Data

public class ProductsResponse extends BaseResponse {

public ProductsResponse(boolean success, String message, ProductsEntity products) {

super(success, message);

this.products = products;

}

public ProductsResponse(ProductsEntity products){

super(true,"Products Data");

}

public ProductsResponse(Optional<ProductsEntity> products){

super(true,"Products Data");

}

private ProductsEntity products;

private Optional<ProductsEntity> products1;

}

Приложение П-Листинг HTTPUtils

public class HTTPUtils {

OkHttpClient client = new OkHttpClient();

public String post(String url, String json) throws IOException {

RequestBody body = RequestBody.create(json, MediaType.get("application/json; charset=utf-8"));

Request request = new Request.Builder().url(url).post(body).build();

try (Response response = client.newCall(request).execute()) {

return response.body().string();

}

}

public String get(String url, String args) throws IOException {

Request req = new Request.Builder().url(url+args).build();

try (Response response = client.newCall(req).execute()) {

return response.body().string();

}

}

public String put(String url, String json) throws IOException {

RequestBody body = RequestBody.create(json, MediaType.get("application/json; charset=utf-8"));

Request request = new Request.Builder().url(url).post(body).build();

try (Response response = client.newCall(request).execute()) {

return response.body().string();

}

}

public String delete(String url, Long id) throws IOException {

Request req = new Request.Builder().url(url+id).delete().build();

try (Response response = client.newCall(req).execute()) {

return response.body().string();

}

}

}

Приложение Р-Листинг HelloApplication

public class HelloApplication extends Application {

@Override

public void start(Stage stage) throws IOException {

FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("tabl.fxml"));

Scene scene = new Scene(fxmlLoader.load());

stage.setTitle("Мебельный салон");

stage.setScene(scene);

stage.show();

}

public static boolean showProductsAddDialog(ProductsEntity productsObj, int id) {

try {

FXMLLoader loader = new FXMLLoader();

loader.setLocation(HelloApplication.class.getResource("addProducts.fxml"));

AnchorPane page = (AnchorPane) loader.load();

Stage dialogStage = new Stage();

dialogStage.setTitle("Products Editor");

dialogStage.initModality(Modality.WINDOW\_MODAL);

Scene scene = new Scene(page);

dialogStage.setScene(scene);

AddProductsController controller = loader.getController();

controller.setDialogStage(dialogStage);

controller.setLabels(productsObj, id);

dialogStage.showAndWait();

return controller.isOkClicked();

} catch (IOException e) {

e.printStackTrace();

return false;

}

}

public static boolean showProductsEditDialog(ProductsEntity productsObj, int id) {

try {

FXMLLoader loader = new FXMLLoader();

loader.setLocation(HelloApplication.class.getResource("editProducts.fxml"));

AnchorPane page = (AnchorPane) loader.load();

Stage dialogStage = new Stage();

dialogStage.setTitle("Products Editor");

dialogStage.initModality(Modality.WINDOW\_MODAL);

Scene scene = new Scene(page);

dialogStage.setScene(scene);

Приложение Р-Листинг HelloApplication

EditProductsController controller = loader.getController();

controller.setDialogStage(dialogStage);

controller.setLabels(productsObj, id);

dialogStage.showAndWait();

return controller.isOkClicked();

} catch (IOException e) {

e.printStackTrace();

return false;

}

}

public static boolean showClientsAddDialog(ClientsEntity clientsObj, int id) {

try {

FXMLLoader loader = new FXMLLoader();

loader.setLocation(HelloApplication.class.getResource("addClients.fxml"));

AnchorPane page = (AnchorPane) loader.load();

Stage dialogStage = new Stage();

dialogStage.setTitle("Clients Editor");

dialogStage.initModality(Modality.WINDOW\_MODAL);

Scene scene = new Scene(page);

dialogStage.setScene(scene);

AddClientsController controller = loader.getController();

controller.setDialogStage(dialogStage);

controller.setLabels(clientsObj, id);

dialogStage.showAndWait();

return controller.isOkClicked();

} catch (IOException e) {

e.printStackTrace();

return false;

}

}

public static boolean showClientsEditDialog(ClientsEntity clientsobj, int id) {

try {

FXMLLoader loader = new FXMLLoader();

loader.setLocation(HelloApplication.class.getResource("editClients.fxml"));

AnchorPane page = (AnchorPane) loader.load();

Stage dialogStage = new Stage();

dialogStage.setTitle("Client Editor");

dialogStage.initModality(Modality.WINDOW\_MODAL);

Приложение Р-Листинг HelloApplication

Scene scene = new Scene(page);

dialogStage.setScene(scene);

EditClientsController controller = loader.getController();

controller.setDialogStage(dialogStage);

controller.setLabels(clientsobj, id);

dialogStage.showAndWait();

return controller.isOkClicked();

} catch (IOException e) {

e.printStackTrace();

return false;

}

}

public static boolean showPersonEditDialog(OrdersEntity ordersObj, int id) {

try {

FXMLLoader loader = new FXMLLoader();

loader.setLocation(HelloApplication.class.getResource("addOrders.fxml"));

AnchorPane page = (AnchorPane) loader.load();

Stage dialogStage = new Stage();

dialogStage.setTitle("Заказ");

dialogStage.initModality(Modality.WINDOW\_MODAL);

Scene scene = new Scene(page);

dialogStage.setScene(scene);

EditOrdersController controller = loader.getController();

controller.setDialogStage(dialogStage);

controller.setLabels(ordersObj, id);

dialogStage.showAndWait();

return controller.isOkClicked();

} catch (IOException e) {

e.printStackTrace();

return false;

}

}

}

Приложение С-Листинг module-info.java

module com.example.clients {

requires javafx.controls;

requires javafx.fxml;

requires com.dlsc.formsfx;

requires lombok;

requires com.google.gson;

requires okhttp3;

requires annotations;

opens com.example.clients to javafx.fxml;

exports com.example.clients;

exports com.example.clients.entity;

opens com.example.clients.entity to com.google.gson;

exports com.example.clients.controller;

opens com.example.clients.controller to javafx.fxml;

exports com.example.clients.response;

opens com.example.clients.response to com.google.gson;

}

Приложение Т-Листинг ClientsController

public class ClientsController {

private final ClientsService service;

public ClientsController(ClientsService service) {this.service = service;}

@PostMapping("/add")

public ResponseEntity<BaseResponse> add(@RequestBody ClientsEntity data){

try {

ClientsValidationUtils.validationClients(data);

service.save(data);

return ResponseEntity.ok(new BaseResponse(true,"клиент добавлен"));

} catch (Exception e){

return ResponseEntity. badRequest().body(new BaseResponse(false,e.getMessage()));

}

}

@PostMapping("/update")

public ResponseEntity<BaseResponse> update(@RequestBody ClientsEntity data){

try {

validationClients(data);

service.save(data);

return ResponseEntity.ok(new BaseResponse(true, "клиент изминен"));

} catch (Exception e){

return ResponseEntity.badRequest().body(new BaseResponse(false, e.getMessage()));

}

}

@DeleteMapping("/delete")

public ResponseEntity <BaseResponse> delete(@RequestParam Long id){

try {

service.delete(id);

return ResponseEntity.ok(new BaseResponse(true, "клиент успешно удален"));

} catch (Exception e) {

return ResponseEntity.badRequest().body(new BaseResponse(false, "не найдено"));

}

}

@GetMapping("/all")

public ResponseEntity<BaseResponse> getAll() {

try {

return ResponseEntity.ok(new ClientsListResponse(service.getAll()));

} catch (Exception e) {

return ResponseEntity.badRequest().body(new BaseResponse(false, e.getMessage()));

}

Приложение Т-Листинг ClientsController

}

@GetMapping("/clients")

public ResponseEntity<BaseResponse> findByClients(@RequestParam String name) {

try {

Iterable<ClientsEntity> productc = service.findByName(name);

if (productc.iterator().hasNext())

return ResponseEntity.ok(new ClientsListResponse(service.findByName(name)));

else

return ResponseEntity.ok(new BaseResponse(true,"Данные не найдены"));

}catch (Exception e){

return ResponseEntity.ok(new ClientsListResponse(service.findByName(name)));

}

}

}

Приложение У-Листинг OrdersController

public class OrdersController {

private OrdersService service;

private OrdersController(OrdersService service){ this.service = service; }

@GetMapping("/all")

public ResponseEntity<BaseResponse> getAll(){

return ResponseEntity.ok(new OrdersListResponse(service.getAll()));

}

@ValidateOnExecution

@PostMapping("/add")

public ResponseEntity<BaseResponse> add(@RequestBody OrdersEntity data){

try {

OrdersValidationUtils.ordersValidationUtils(data);

service.save(data);

return ResponseEntity.ok(new BaseResponse(true,"Заказ добавлен"));

} catch (Exception e){

return ResponseEntity. badRequest().body(new BaseResponse(false,e.getMessage()));

}

}

@PostMapping("/update")

public ResponseEntity <BaseResponse> update(@RequestBody OrdersEntity data){

try {

service.save(data);

return ResponseEntity.ok(new BaseResponse(true, "В заказ внесены изменения"));

} catch (Exception e) {

return ResponseEntity. badRequest().body(new BaseResponse(false,e.getMessage()));

}

}

@DeleteMapping("/delete")

public ResponseEntity <BaseResponse> delete(@RequestParam Long id){

try {

service.delete(id);

return ResponseEntity.ok(new BaseResponse(true, "Заказ успешно удален"));

} catch (Exception e) {

return ResponseEntity.badRequest().body(new BaseResponse(false, "не найдено"));

}

}

@GetMapping()

public ResponseEntity<BaseResponse> getProducts(@RequestParam String name){

return ResponseEntity.ok(new OrdersListResponse(service.getName(name)));

}

Приложение У-Листинг OrdersController

@GetMapping("/datetime")

public ResponseEntity<BaseResponse> findByData(@RequestParam String data) {

try {

Iterable<OrdersEntity> dat = service.findByData(data);

if (dat.iterator().hasNext())

return ResponseEntity.ok(new OrdersListResponse(service.findByData(data)));

else

return ResponseEntity.ok(new BaseResponse(false, "По такой дате, заказа не существует"));

} catch (Exception e) {

return ResponseEntity.ok(new OrdersListResponse(service.findByData(data)));

}

}

@GetMapping("/orders")

public ResponseEntity<BaseResponse> findByOrders(@RequestParam String orders) {

try {

Iterable<OrdersEntity> productc = service.findByOrders(orders);

if (productc.iterator().hasNext())

return ResponseEntity.ok(new OrdersListResponse(service.findByOrders(orders)));

else

return ResponseEntity.ok(new BaseResponse(true,"Данные не найдены"));

}catch (Exception e){

return ResponseEntity.ok(new OrdersListResponse(service.findByOrders(orders)));

}

}

}

Приложение Ф-Листинг ValidationExceptionClients

public class ValidationExceptionClients extends ValidationExceptionOrders

{

public ValidationExceptionClients(String message)

{

super(message);

}

}

Приложение Х-Листинг ValidationExceptionOrders

public class ValidationExceptionOrders extends Exception {

public ValidationExceptionOrders(String message)

{

super(message);

}

}

Приложение Ц-Листинг ValidationExceptionProducts

public class ValidationExceptionProducts extends Exception {

public ValidationExceptionProducts(String message)

{ super(message);

}

}

Приложение Ч-Листинг ClientsRepo

public interface ClientsRepo extends CrudRepository<ClientsEntity, Long> {

Iterable<ClientsEntity> findByName (String name);

}

Приложение Ш-Листинг OrdersRepo

public interface OrdersRepo extends CrudRepository<OrdersEntity, Long> {

Iterable<OrdersEntity> findByClients\_name(String name);

Iterable<OrdersEntity> findByData (String data);

Iterable<OrdersEntity> findByOrders (String orders);

}

Приложение Ъ-Листинг ProductsRepo

public interface ProductsRepo extends CrudRepository<ProductsEntity, Long> {

Optional<ProductsEntity> findById (Long id);

Iterable<ProductsEntity> findByCategory (String category);

Iterable<ProductsEntity> findByProducts (String products);

Iterable<ProductsEntity> findByMaterial (String material);

Iterable<ProductsEntity> findByQanitity (int qanitity);

}

Приложение Ы-Листинг BaseResponse

@Data

@AllArgsConstructor

@RequiredArgsConstructor

public class BaseResponse {

protected boolean success;

protected String message;

}

Приложение Ь-Листинг ClientsListResponse

@Data

@EqualsAndHashCode(callSuper = false)

public class ClientsListResponse extends BaseResponse

{

public ClientsListResponse(Iterable<ClientsEntity>data)

{

super(true,"клиенты");

this.data= data;

}

private Iterable<ClientsEntity>data;

}

Приложение Э-Листинг ClientsResponse

@Data

@EqualsAndHashCode(callSuper = false)

public class ClientsResponse extends BaseResponse

{

public ClientsResponse(boolean success, String message, ClientsEntity clients)

{

super(success,message);

this.clients = clients ;

}

public ClientsResponse(ClientsEntity clients)

{

super(true,"Clients data");

}

private ClientsEntity clients;

}

Приложение Ю-Листинг OrdersListResponse

@Data

@RequiredArgsConstructor

public class OrdersListResponse extends BaseResponse {

public OrdersListResponse(Iterable<OrdersEntity> data) {

super(true, "Список заказов:");

this.data=data;

}

private Iterable<OrdersEntity> data;

}

Приложение Я-Листинг OrdersResponse

@Data

@EqualsAndHashCode(callSuper = false)

public class OrdersResponse extends BaseResponse

{

public OrdersResponse(boolean success, String message, OrdersEntity products)

{

super(success,message);

this.products = products;

}

public OrdersResponse(OrdersEntity products)

{

super(true,"products data");

}

private OrdersEntity products;

}

Приложение А-А-Листинг ProductsListResponse

@Data

@EqualsAndHashCode(callSuper = false)

public class ProductsListResponse extends BaseResponse

{

public ProductsListResponse(Iterable<ProductsEntity>data)

{

super(true,"товар");

this.data = data;

}

private Iterable<ProductsEntity>data;

}

Приложение А-Б-Листинг ProductsResponse

@Data

@EqualsAndHashCode(callSuper=false)

public class ProductsResponse extends BaseResponse

{

public ProductsResponse(boolean success, String message, ProductsEntity products)

{

super(success,message);

this.products = products;

}

public ProductsResponse(ProductsEntity products)

{

super(true,"products");

}

private ProductsEntity products;

}

Приложение А-В-Листинг ClientsService

@Service

public class ClientsService {

private final ClientsRepo repo;

public ClientsService(ClientsRepo repo) {

this.repo = repo;

}

public void save(ClientsEntity clients) {repo.save(clients);}

public void delete(Long id){repo.deleteById(id);}

public Iterable<ClientsEntity> getAll(){return repo.findAll();}

public Iterable<ClientsEntity> findByName (String name) { return repo.findByName(name); }

}

Приложение А-Г-Листинг OrdersService

@Service

public class OrdersService {

private final OrdersRepo repo;

public OrdersService(OrdersRepo repo) {

this.repo = repo;

}

public Iterable<OrdersEntity> findByData (String data) { return repo.findByData(data); }

public void save(OrdersEntity orders) {repo.save(orders);}

public void delete(Long id){repo.deleteById(id);}

public Iterable<OrdersEntity> getAll(){return repo.findAll();}

public Iterable<OrdersEntity> getName(String name){

return repo.findByClients\_name(name); }

public Iterable<OrdersEntity> findByOrders (String orders) { return repo.findByOrders(orders); }

}

Приложение А-Д-Листинг ProductsService

@Service

public class ProductsService {

private final ProductsRepo repo;

public ProductsService(ProductsRepo repo) {

this.repo = repo;

}

public void save(ProductsEntity products) {repo.save(products);}

public void delete(Long id) { repo.deleteById(id); }

public Iterable<ProductsEntity> getAll(){return repo.findAll();}

public Iterable<ProductsEntity> findByProducts (String products) { return repo.findByProducts(products);}

public Iterable<ProductsEntity> findByCategory (String category) { return repo.findByCategory(category); }

public Iterable<ProductsEntity> findByMaterial (String material) { return repo.findByMaterial(material); }

public Iterable<ProductsEntity> findByQanitity (int qanitity) { return repo.findByQanitity(qanitity); }

}

Приложение А-Е-Листинг ClientsValidationUtils

public class ClientsValidationUtils

{

public static void validationClients(ClientsEntity data)throws ValidationExceptionClients {

String name = data.getName();

if (name == null || name.isBlank()) {

throw new ValidationExceptionClients("Имя клиента не может быть пустым");

} else if (name.length() < 3 || name.length() > 20) {

throw new ValidationExceptionClients("Имя должно быть в диапозоне от 3 до 20");

}

if (!name.matches("^[A-ЯЁ][а-яё]+$")) {

throw new ValidationExceptionClients("Имя должно начинаться с заглавной буквы и на русском языке");

}

String lastname = data.getLastname();

if (lastname == null || lastname.isBlank()) {

throw new ValidationExceptionClients("Фамилия не может быть пустым");

} else if (lastname.length() < 3 || lastname.length() > 20) {

throw new ValidationExceptionClients("Фамилия должна быть в диапозоне от 3 до 20");

}

if (!lastname.matches("^[A-ЯЁ][а-яё]+$")) {

throw new ValidationExceptionClients

("Фамилия должна начинаться с заглавной буквы и на русском языке");

}

String surname = data.getSurname();

if (surname == null || surname.isBlank()) {

throw new ValidationExceptionClients("Отчество не может быть пустым");

} else if (surname.length() < 3 || surname.length() > 20) {

throw new ValidationExceptionClients("Отчество должно быть в диапозоне от 3 до 20");

}

if (!surname.matches("^[A-ЯЁ][а-яё]+$")) {

throw new ValidationExceptionClients

("Отчество должно начинаться с заглавной буквы и на русском языке");

}

}

}

Приложение А-Ё-Листинг OrdersValidationUtils

public class OrdersValidationUtils {

public static void ordersValidationUtils(OrdersEntity orders) throws ValidationExceptionOrders {

String title = orders.getOrders();

if (title == null || title.isBlank()) {

throw new ValidationExceptionOrders("Поле название заказа не должно быть пустым");

} else if (title.length() < 3 || title.length() > 20){

throw new ValidationExceptionOrders("Имя должно быть в диапозоне от 3 до 20");

}

if (!title.matches("^[A-ЯЁ][а-яё]+$")){

throw new ValidationExceptionOrders("Название заказа должно начинаться с заглавной буквы и

на русском языке");

}

if (orders.getClients() == null) {

throw new ValidationExceptionOrders("нужно выбрать клиента");

}

if (orders.getProducts() == null) {

throw new ValidationExceptionOrders("нужно выбрать товар");

}

String date = orders.getData();

if (date == null || !(date.matches("[0-9]{4}-[0-9]{2}-[0-9]{2}"))) {

throw new ValidationExceptionOrders("Поле time не соответствует формату (2023-05-05)");

}

}

}

Приложение А-Ж-Листинг ProductsValidationUtils

public class ProductsValidationUtils {

public static void validationProducts(ProductsEntity data) throws ValidationExceptionProducts {

String products = data.getProducts();

if (products == null || products.isBlank()) {

throw new ValidationExceptionProducts

("Название товара пустое");

} else if (products.length() < 3 || products.length() > 20) {

throw new ValidationExceptionProducts

("Название товара должно быть в диапозоне от 3 до 20");

}

if (!products.matches("^[A-ЯЁ][а-яё]+$")) {

throw new ValidationExceptionProducts

("Название товара должно начинаться с заглавной буквы и на русском языке");

}

String category = data.getCategory();

if (category == null || category.isBlank()) {

throw new ValidationExceptionProducts("Название категории пустое");

} else if (category.length() < 3 || category.length() > 20) {

throw new ValidationExceptionProducts

("Название категории должно быть в диапозоне от 3 до 20");

}

if (!category.matches("^[A-ЯЁ][а-яё]+$")) {

throw new ValidationExceptionProducts

("Название категории должно начинаться с заглавной буквы и на русском языке");

}

String material = data.getMaterial();

if (material == null || material.isBlank()) {

throw new ValidationExceptionProducts("Название материала пустое");

} else if (material.length() < 3 || material.length() > 20) {

throw new ValidationExceptionProducts("Название материала должно быть в диапозоне от 3 до 20");

}

if (!material.matches("^[A-ЯЁ][а-яё]+$")) {

throw new ValidationExceptionProducts

("Название материала должно начинаться с заглавной буквы и на русском языке");

}

String productsIntQanitity = String.valueOf(data.getQanitity());

if (productsIntQanitity.isBlank()|| productsIntQanitity.equals("0")) {

throw new ValidationExceptionProducts("Не было введено никаких символов");

}else if (productsIntQanitity.isBlank() || !productsIntQanitity.matches("[\\d0-9]{1,4}"))

throw new ValidationExceptionProducts("Минимальное кол-во символов в количество 1 до 4символов");

Приложение А-З-Листинг ValidationUtils

public class ValidationUtils {

public static void ValidationClients(OrdersEntity orders) throws ValidationExceptionOrders {

}

}

Приложение А-И-Листинг ServerApplication

@SpringBootApplication

public class ServerApplication {

public static void main(String[] args) {

SpringApplication.run(ServerApplication.class, args);

}

}