

**КОЛЛЕДЖ КОСМИЧЕСКОГО МАШИНОСТРОЕНИЯ И ТЕХНОЛОГИЙ**

|  |
| --- |
|  |

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине «Прикладное программирование»**

**Тема: «Веб-интерфейс к автоматизированному рабочему**

**месту кассира»**

Выполнил студент

Андрейко Борис Александрович

Группа П2-17

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

**Королёв, 2020**

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ](#_Toc43564010) 3

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc43564011)

[1.1 Постановка задачи ВКР 6](#_Toc43564012)

[1.2 Анализ предметной области 6](#_Toc43564013)

[2. Проектная часть 16](#_Toc43564020)

[2.1 Алгоритмы решения 16](#_Toc43564021)

[2.2 Выбор инструментов, сред, языков 18](#_Toc43564022)

[2.3 Структура программного комплекса 21](#_Toc43564023)

[2.3.1 Логическое и физическое моделирование базы данных 22](#_Toc43564024)

[2.4 Описание процедур и функций 28](#_Toc43564025)

[2.5 Разработка автоматизированной системы 31](#_Toc43564026)

[2.6 Описание тестовых наборов 38](#_Toc43564027)

[3.Эксплутационная часть 40](#_Toc43564028)

[3.1. Эксплуатационный раздел 40](#_Toc43564029)

[3.1.1. Рекомендации по эксплуатации программного продукта 40](#_Toc43564030)

[3.1.2. Руководство пользователя 42](#_Toc43564031)

[3.1.3. Руководство системного администратора 44](#_Toc43564032)

[ЗАКЛЮЧЕНИЕ 45](#_Toc43564035)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 46](#_Toc43564036)

[ПРИЛОЖЕНИЯ 48](#_Toc43564037)

# ВВЕДЕНИЕ

Деятельность практически любого современного предприятия связана с обработкой и анализом больших объёмов информации. Так как ручной анализ не представляется возможным, на предприятиях используются различные автоматизированные информационные системы, облегчающие, а иногда и заменяющие процесс ручной обработки информации. Применение автоматизированных систем обладает целым рядом преимуществ, среди которых, прежде всего, следует отметить значительно более высокую скорость обработки информации, практически мгновенную реакцию на происходящие в системе события, возможность использования защищённых или зашифрованных каналы обмена информации, отсутствие человеческого фактора и многое другое. Особую важность и значимость использование информационных систем приобретает в сфере торговли, где важнейшими показателями качества функционирования системы является точность и скорость расчётов. С учётом происходящих изменений, когда большинство настольных приложений практически полностью заменяются их аналогами в веб-браузере, а так же необходимости экономии средств, выпускная квалификационная работа является актуальной и своевременной.

Целью работы является создание веб-интерфейса для автоматизированного рабочего места кассира.

Для достижения поставленной цели необходимо решить следующие задачи:

* Проанализировать организационно-штатную структуру фирмы «Нионка» и особенности работы кассира по оформлению отчётных документов.
* Рассмотреть преимущества удалённого доступа к рабочему месту кассира
* Выбрать средства и методы разработки веб-интерфейса для рабочего места кассира;

Объектом автоматизации является рабочее место кассира в фирме «Нионка».

В первой главе рассматриваются особенности функционирования фирмы «Нионка» – организационно-штатная структура, перечень оказываемых услуг, приводятся формулы расчёта полной стоимости товара.

Вторая глава посвящена выбору средств разработки веб-интерфейса для автоматизированного рабочего места кассира.

Третья глава посвящена технико-эксплуатационному обоснованию разработки программного обеспечения для автоматизации деятельности кассира. Проведённые расчёты позволяют сделать вывод о том, что разработка и внедрение веб-интерфейса для АРМ кассира является обосновании.

Для хранения информации об именах пользователей, товарах, транзакциях и прочих параметрах была использована СУБД MySQL, которая представляет собой функциональную СУБД, поддерживающую все необходимые в проекте функции – хранимые процедуры, масштабирование, возможность прямого общения с БД на языке SQL и др.

С целью защиты от несанкционированного доступа, а так же возможного перехвата информации заинтересованными лицами, в АРМ применено разграничение прав доступа – существуют профили администратора и продавца, так же используется механизм шифрования критически важных данных, таких как пароль.

Работа продавцов в программе реализована с использованием механизма сессий. Процедуры обмена информацией между модулями, которые связаны с передачей информации, реализуются с использованием транзакционного подхода, который предполагает, что операция либо выполняется полностью, либо не выполняется совсем.

Формирование отчётов за смену, а так же промежуточных отчётов можно выполнить в бумажном и электронном виде, так же как и выдачу чека для покупателя.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Постановка задачи ВКР

Задачей курсовой работы является разработка веб-приложения для автоматизации деятельности кассира. Существующие программные решения представляют собой приложения, работающие в операционной системе Windows или жестко привязанные к конкретному производителю POS-терминалов, что не позволяет выполнить гибкое конфигурирование системы, а так же получить доступ к операциям движения денежных средств вне программно-аппаратной платформы АРМ-кассира. Предлагаемая разработка основана на технологии клиент-сервер, которая позволяет клиенту не привязываться к конкретной Операционной системе – достаточно, чтобы на клиентском терминале был браузер – все операции выполняются через него. Внедрение разработанного программного обеспечения позволит проводить удалённый контроль операций с любого устройства, в котором установлен веб-браузер при условии, что устройство имеет доступ к сети Интернет.

## 1.2 Анализ предметной области

Фирма «Нионка» представляет собой коммерческую организацию в организационно-правовой форме общества с ограниченной ответственностью. Общество является юридическим лицом и действует в соответствии с нормативно-правовыми актами Российской Федерации и своим уставом. Общество в установленном законом порядке осуществляет следующие виды деятельности:

* изготовление и реализация тканей для мебели;
* разработка и реконструкция мебели по индивидуальным заказам;
* реализация мебельной фурнитуры.

Обобщённая организационная структура предприятия показана на рис. 1.1.



Рис. 1.1. Организационная структура предприятия ООО «Нионка»

Непосредственное руководство фирмой осуществляют Нишанов Виктор Троадиевич и Почоев Сайфиддинходжа Судуриддинович. Их должностные обязанности включают общее руководство фирмой, обоснование ценовой политики, издание приказов и распоряжений по предприятию.

Генеральный директор фирмы Петрушкин Виталий Петрович занимается приёмом на работу и увольнением с работы по согласованию с руководством.

Заместитель директора по производству – Петрушкина Анастасия Степановна выполняет обязанности директора в случае отсутствия его на рабочем месте.

Служба главного охранника (на схеме не показана) включает в себя Главного охранника Степанова Михаила Андреевича и охранников – Стенина Михаила Степановича и Любина Максима Антоновича. В их должностные обязанности входят: обеспечение безопасности фирмы, разбор инцидентов, взаимодействие с правоохранительными органами по вопросам обеспечения безопасности.

Бухгалтер имеет в своем подчинении 4 кассиров, которые обслуживают клиентов. Именно в интересах кассиров ведётся разработка веб-интерфейса для АРМ-кассира.

Так как деятельность фирмы связана с оказанием возмездных услуг населению, движение денежных средств как внутри фирмы, так и между внешними учреждениями должна отражаться в документах, которые именуются «кассовыми документами». Кассовые документы представляют собой документы, которые применяются для учета различных кассовых операций по получению либо выдаче денежных средств. Перечень кассовых документов фирмы представлен в таблице 1.1.

Таблица 1.1

Кассовые документы

|  |  |
| --- | --- |
| – | Заявление о регистрации контрольно-кассовой техники |
| КМ-1 | Акт о переводе показаний суммирующих денежных счетчиков на нули и регистрации контрольных счетчиков контрольно-кассовой машины |
| КМ-2 | Акт о снятии показаний контрольных и суммирующих денежных счетчиков при сдаче (отправке) контрольно-кассовой машины в ремонт и при возвращении ее в организацию |
| КМ-3 | Акт о возврате денежных сумм покупателям (клиентам) по неиспользованным кассовым чекам |
| КМ-4 | Журнал кассира-операциониста |
| КМ-5 | Журнал регистрации показаний суммирующих денежных и контрольных счетчиков контрольно-кассовых машин, работающих без кассира-операциониста |
| КМ-6 | Справка-отчет кассира-операциониста |
| КМ-7 | Сведения о показаниях счетчиков контрольно-кассовых машин |
| КМ-8 | Журнал учета вызовов технических специалистов и регистрации выполненных работ |
| КМ-9 | Акт о проверке наличных денежных средств кассы |
| КО-1 | Приходный кассовый ордер |
| КО-2 | Расходный кассовый ордер |
| КО-3 | Журнал регистрации приходных и расходных кассовых документов |
| КО-4 | Кассовая книга |
| цКО-5 | Книга учета принятых и выданных кассиром денежных средств |

Понятие кассовых операций включает совокупность материально-технических процедур по приему, хранению и выдаче наличных денег. В соответствии с законодательством юридические лица обязаны хранить свои средства в банках и осуществлять все платежи по безналичному расчету, но для осуществления выплат работникам и прочих текущих наличных расходов допускается хранение некоторой суммы в специально оборудованной кассе.

Осуществление наличного расчета и любых других операций с наличностью возможно только при наличии кассы, а касса на предприятии может существовать только при выполнении всех требований закона. Проведение кассовых операций регулируется Положением Центрального банка о «Порядке ведения кассовых операций с банкнотами и монетой банка России на территории Российской Федерации» от 12 октября 2011 года № 373-П.

Закон о кассовых операциях вступил в силу первого января 2012 года. С этого периода ведение учета кассовых операций стало обязательным для всех предпринимателей.

Документация кассовых операций

Кассовая операция – любая выплата и расход, а также получение средств, которые предприниматель осуществляет наличными деньгами. Основные кассовые операции отражают в следующих документах:

* кассовой книге;
* приходных ордерах;
* расходных ордерах;
* книге учета принятых и выданных кассиром денежных средств (журнал кассовых операций);
* расчетно-платежной ведомости;
* платежной ведомости.

Бланки документов должны быть установленной формы (перед использованием необходимо проверить, не является ли эта форма устаревшей). Каждый документ должен быть заполнен разборчиво и без помарок (вручную или на компьютере), должен быть подписан уполномоченным лицом (перечень таких лиц устанавливает руководитель предприятия по согласованию с главным бухгалтером). Положение о кассовых операциях предусматривает, что вести кассовые документы должен кассир, главный (старший) бухгалтер предприятия, руководитель или же любой другой работник, назначенный специальным приказом. Такой работник должен быть ознакомлен с положениями кассовой дисциплины и должен подписать договор о полной материальной ответственности.

Виды кассовых операций:

* прием наличных средств в кассу, выдача наличных из кассы и оформление вышеуказанных кассовых документов;
* оприходование наличных средств в кассу;
* хранение наличных средств и ценностей предприятия;
* порядок использования выручки и наличности, полученной из банка;
* соблюдение порядка и сроков сдачи наличных денег в банк.

В настоящее время, в соответствии с ФЗ-54, все денежные расчёты между розничным покупателем и продавцом должны выполняться с использованием контрольно-кассовой техники. Как правило, контрольно-кассовая машина не функционирует отдельно, а работает в составе программно-аппаратного комплекса автоматизированного рабочего места (АРМ) кассира. АРМ Кассира представляет (рис. 1.2) собой аппаратно-программный комплекс, состоящий из персонального компьютера, необходимых для работы Системы Периферийных Устройств (СПУ) и фискального регистратора ПРИМ-08К. На АРМ установлена программная часть клиентского программного обеспечения «АРМ Кассира».

Комплекс позволяет производить продажу платных и льготных, а также выдачу безденежных железнодорожных билетов в пригородном сообщении, вести первичную статистическую и бухгалтерскую отчётности кассиров билетных касс.

Автоматизированное рабочее место кассира состоит из:

* персонального компьютера начального уровня;
* установленного программного модуля в составе АСУ ППК «АРМ кассира» с функциями оформления всех видов проездных документов;
* системы периферийных устройств ввода-вывода информации (клавиатура, манипулятор типа «мышь», экран LCD);
* устройства подключения к сетям общего доступа (сетевая карта, GPRS со сторожевым таймером/CDMA);
* фискального регистратора ПРИМ-08К.
* Основные функции и сферы применения ПРИМ-08К:
* для торговли, в том числе при продаже товаров с оформлением товарного чека;
* для сферы услуг, в том числе для оформления авиаперевозок;
* для торговли нефтепродуктами и газовым топливом;
* для отелей и ресторанов.



Рис. 1.2. АРМ кассира

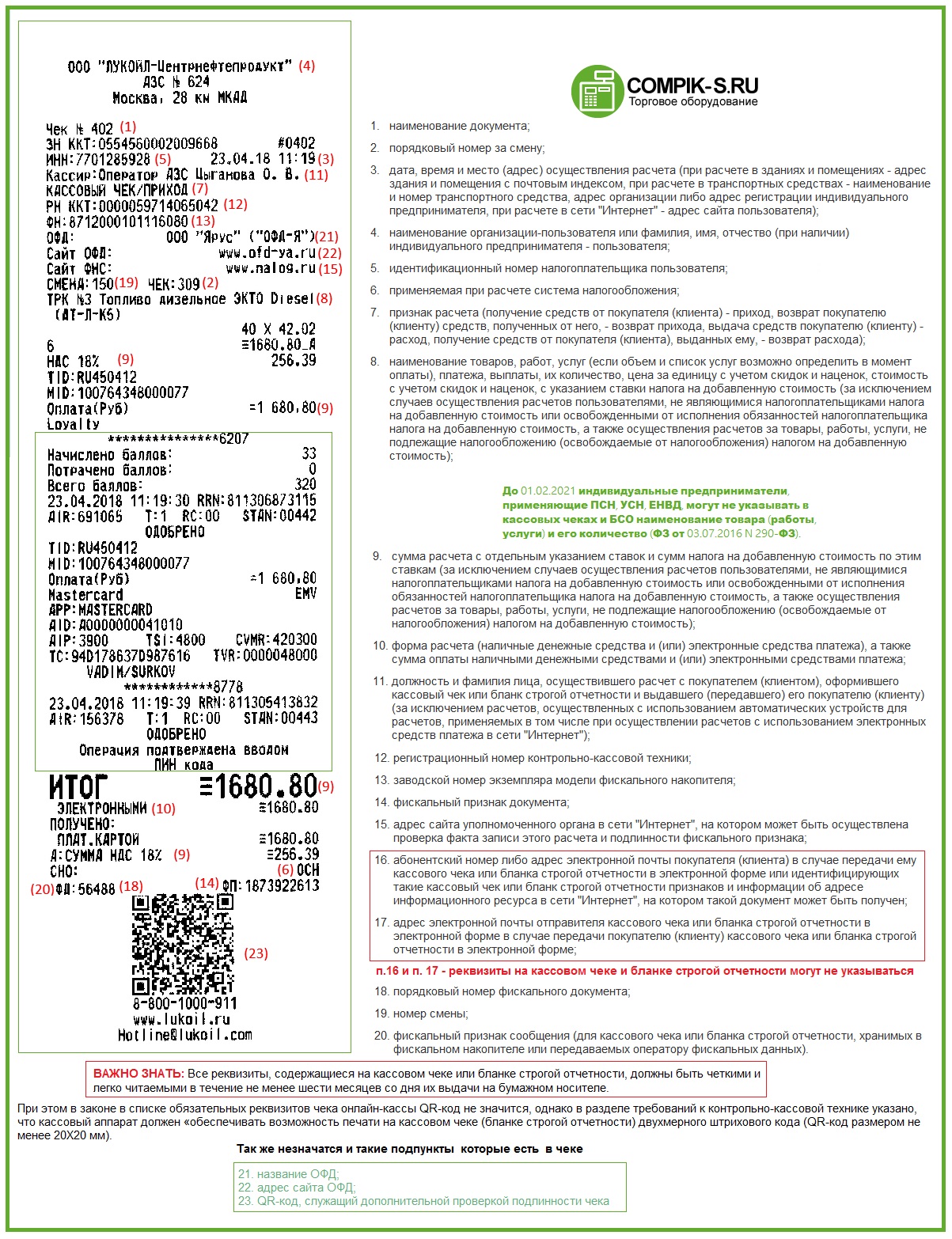


Рис. 1.3. Структура кассового чека.

Технические характеристики ПРИМ-08К:

Характеристики фискального блока

Фискальная память: 3424 сменных записей

Количество фискализаций/перерегистраций: 5

Количество активизаций: 20

Интерфейс обмена с накопителем и ЭКЛЗ: I2C

Программируемое клише: 6X40 символов

Программируемые пароли: 16 разрядов

Сохранность информации при выключении питания: не менее 10 лет

Разрядность денежных счетчиков: 16 разрядов (макс.)

Разрядность денежных счетчиков общего итога: 18 разрядов

Принцип печати Термопечать

Матрица символа Font A, Font B

Скорость печати: 100 мм / 31 строка в сек

Бумажная лента

ширина бумаги: 80 мм/58 мм

ширина печати: 72/50,8 мм

Макс. кол-во символов в строке 40 символов

Ресурс печати Печатающая головка: 60 млн. строк

Габаритные размеры: 142x132x199 мм

Масса 1,4 кг

Интерфейс сопряжения с ПК: RS-232C нуль-модемный; Centronics, USB (через дополнительный внешний адаптер)

Дополнительные возможности: Подключение денежного ящика (Разъем RJ12, питание 24 В)

Достоинства:

* Первый в России проверенный временем фискальный регистратор с ЭКЛЗ;
* Высокоскоростная (31 строка в секунду) бесшумная термопечать чека и штрих-кодов;
* Простейшая и быстрая смена рулона чековой ленты;
* Широкая лента, позволяющая формировать информативный чек;
* Самый компактный и легкий фискальный регистратор в России;
* Привлекательный эргономичный дизайн, минимальная занимаемая площадь;
* Простота и удобство в обслуживании;

Порядок выполнения операция при оформлении покупки показан на рис 1.4

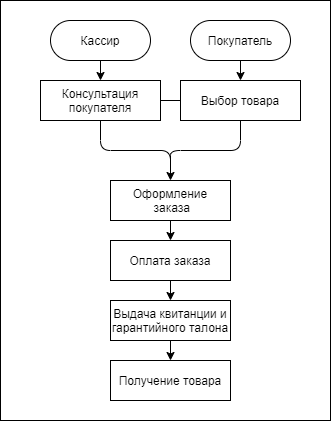


Рис. 1.4. Структура оплаты товаров.

В зависимости от вида расчётов, используются различные способы выдачи кассовых чеков, подтверждающие факт покупки товара.

# Проектная часть

## Алгоритмы решения

Для реализации АМБДК (Автоматизированная Мобильная База Данных Кассира) необходимы следующие алгоритмы:

1. Регистрация – добавление нового пользователя в систему;
2. Авторизация (предоставление лицу прав на выполнение определённых действий);
3. Создание новой транзакции – транзакции типа «Продажа» и транзакции типа «Возврат»;
4. Получение и отображение списка всех имеющихся в системе транзакций;
5. Просмотр полного содержимого выбранной из списка транзакции.
6. Выход – действие, обратное авторизации

На рисунке 2.1 приведена Use Case диаграмма, которая дает нам структуру взаимодействия кассира с программным продуктом АМБДК.

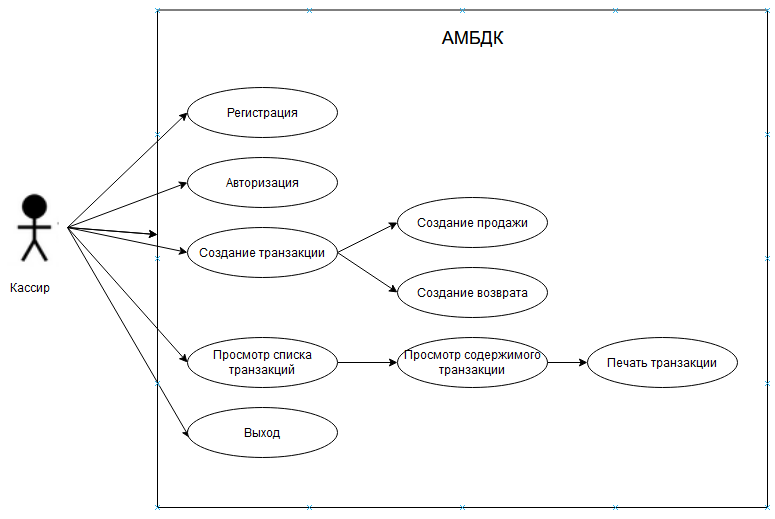


Рис 2.1. Use Case диаграмма АМБДК

Алгоритм 1. Регистрация

1. Начало.
2. Переходим на страницу регистрации
3. Заполняем данные и нажимаем «подтвердить»
4. Если данные корректны, то переходим к п.5. Иначе, к п. 3
5. Получаем уведомление об успешной регистрации, и происходит автоматическое перенаправление на страницу авторизации.
6. Конец.

Алгоритм 2. Авторизация

1. Начало.
2. Вводим данные для авторизации и нажимаем «войти».
3. Если данные корректны, то переходим к п. 4. Иначе, к п. 2
4. Происходит автоматическое перенаправление на страницу главного меню.
5. Конец

Алгоритм 3. Создание транзакции

1. Начало.
2. Авторизация в систему
3. Нажать на «Произвести операцию»
4. Заполнить данные, выбрать тип операции
5. Нажать «отправить»
6. Если данные корректны, то переход к п. 7. Иначе, к п. 4
7. Происходит автоматическое перенаправление на страницу главного меню.
8. Конец

Алгоритм 4. Печать транзакции

1. Начало
2. Авторизация
3. Получение списка всех транзакций на странице «Все отчёты»
4. Выбор требуемой транзакции из списка
5. Нажатие на кнопку «Печать»
6. Выполнение настройки печати через стандартный интерфейс печати браузера
7. Выполнение печати, закрытие стандартного интерфейса браузера
8. Конец

Алгоритм 5. Выход

1. Начало.
2. Кликаем по кнопке выход. Переходим к п.3.
3. Происходит перенаправление на начальную страницу.
4. Конец.

## Выбор инструментов, сред, языков

Выбор инструментов обуславливается требованиями проекта, поставленными перед проектом целями и предпочтениями самих разработчиков.

Требования к продукту и его разработке:

* Кроссплатформенность. Приложение должно запускаться на всех устройствах (компьютеры, планшеты, смартфоны);
* Время на разработку, отладку и тестирование: 3-4 месяца;
* Система контроля версий. Над проектом работает 3 программиста, поэтому система контроля версий необходима.

Наиболее оптимальным по времени и ресурсам способом реализовать кроссплатформенность было создание веб-приложения, так как в нём кроссплатформеность обеспечивается браузерами, а код самого приложения не зависит устройства. Это решение автоматически привело к выбору языков для написания front-end части – HTML, JavaScript, CSS.

Для серверной части был выбран язык программирования C# [7] - объектно-ориентированный язык семейства .NET. Целевой платформой был выбран .NET Core 3 – новейшая open-source реализация .NET, разрабатываемая Microsoft при поддержке всего .NET сообщества. Виртуальная машина .NET позволяет абстрагироваться от устройства, на котором запущен сервер, и писать платформонезависимый код, что значительно упрощает и ускоряет разработку.

На роль СУБД была выбрана MySQL[8] — популярная, быстрая и надёжная система управления реляционными базами данных, предоставляющая широкие возможности настройки, администрирования, разработки всех компонентов базы данных и управления ими.

Выбор её был обусловлен наличием достаточных знаний в функционале данной СУБД, простотой её использования, а также тем что существует готовый LINQ-to-SQL провайдер для взаимодействия с MySQL из .NET Core.

Среды разработки

Для работы с программным стеком front-end части приложения использовался Visual Studio Code[9] – кроссплатформенный open-source редактор от Microsoft. Его основные преимущества:

1. Скорость работы;
2. Хорошая поддержка множества форматов файлов и языков программирования, в том числе HTML, JS, CSS;
3. Огромный репозиторий плагинов, расширяющих возможности редактора: плагин для поддержки фреймворка Vue.js[10], плагин для интеграции с системами контроля версий и другие.

Для работы с C# использовался JetBrains Rider[11] - кроссплатформенная IDE для .NET, поддерживающая разработку консольных, серверных и графических приложений на .NET Framework и .NET Core, мобильных приложений на Xamarin и видеоигр на Unity. Для анализа C# кода в Rider используется ReSharper - широко известное расширение для IDE Visual Studio от Microsoft собственной разработки JetBrains. Он обеспечивает более умную, адаптивную и широкую помощь разработчику, чем IntelliSense в Visual Studio. Помимо этого, Rider значительно более производительный, чем другие IDE для .NET. Другие полезные особенности Rider:

1. Встроенный декомпилятор;

2. Встроенный NuGet клиент;

3. Интегрированный REST клиент, позволяющий прямо в IDE выполнять заданные HTTP запросы для тестирования веб-сервера;

4. Мощный отладчик кода;

5. Интеграция с системами контроля версий (VCS), позволяющая управлять коммитами, версиями, ветками и другими компонентами VCS через интерфейс;

6. Встроенный профайлер программ;

Для работы с MySQL была выбрана программа DataGrip[12] - кроссплатформенная IDE для работы с базами данных от JetBrains. Её преимущества:

1. Возможность использования в любой операционной системе;

2. Высокая производительность;

3. Широкий выбор поддерживаемых СУБД, включая MySQL;

4. Возможность подключения как к локальной БД, так и к удалённой по защищённому протоколу;

5. Инструмент разметки таблиц, включающий в себя визуальные редакторы для столбцов, индексов, первичных ключей, внешних ключей, проверки целостности, статистики, свойств хранилищ.

6. Инструмент создания диаграмм базы данных для визуализации ее структуры и построения логических отношений между таблицами;

7. Редактор SQL запросов с контекстными и интеллектуальными подсказками в процессе написания;

8. Инструменты автоматического рефакторинга таблиц;

9. Возможность экспорта всех данных в CSV, TXT и XML файлы, а также экспорт структуры базы данных и самих данных при помощи mysqldump через графический интерфейс.

## Структура программного комплекса

АМБДК включает в себя следующие элементы:

Веб интерфейс - включающего в себя страницы регистрации\авторизации, добавления и просмотра транзакций.

Серверная часть - включающая в себя контроллеры входящих запросов, базы данных и методы её изменения.

Структура проекта показана на рис 2.2.

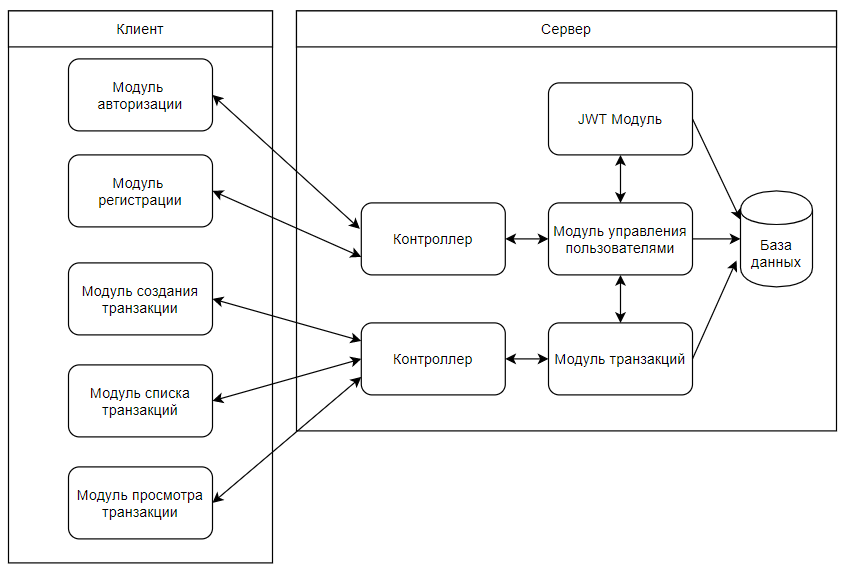


Рисунок 2.2. Схематичное отображение структуры проекта

### Логическое и физическое моделирование базы данных

Организация базы данных — это представление пользователя о той предметной области, информация о которой должна храниться в БД.

Результатом проектирования БД является набор из трёх ключевых составляющих схемы БД:

1. информация об объектах БД, относящихся к выбранной предметной области;
2. информация о свойствах этих объектов;
3. информация о связях и отношениях между этими объектами.

Такая модель не зависит от технической реализации базы данных и ЭВМ, на которой она функционирует.

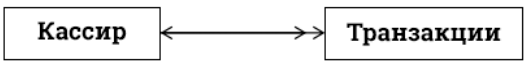
В предметной области «формирование транзакций» можно выделить следующие объекты:

объекты: Кассиры, Транзакции.

В предметной области можно определить следующие сущности:

* Сущность Кассиры атрибуты: Логин кассира, код кассира, id, пароль.
* Сущность Транзакции атрибуты: код кассира, массив мебели (номер, наименование, количество, цена за 1 ед), внесённая сумма, тип операции.

Рассмотрим типы связей между сущностями кассир и транзакции. Учитывая, что кассир видит не только свои транзакции - Связь один ко многим.



Структуру таблицы и их связей можно представить в виде ER-диаграммы, которая показана на рис 2.3.1:

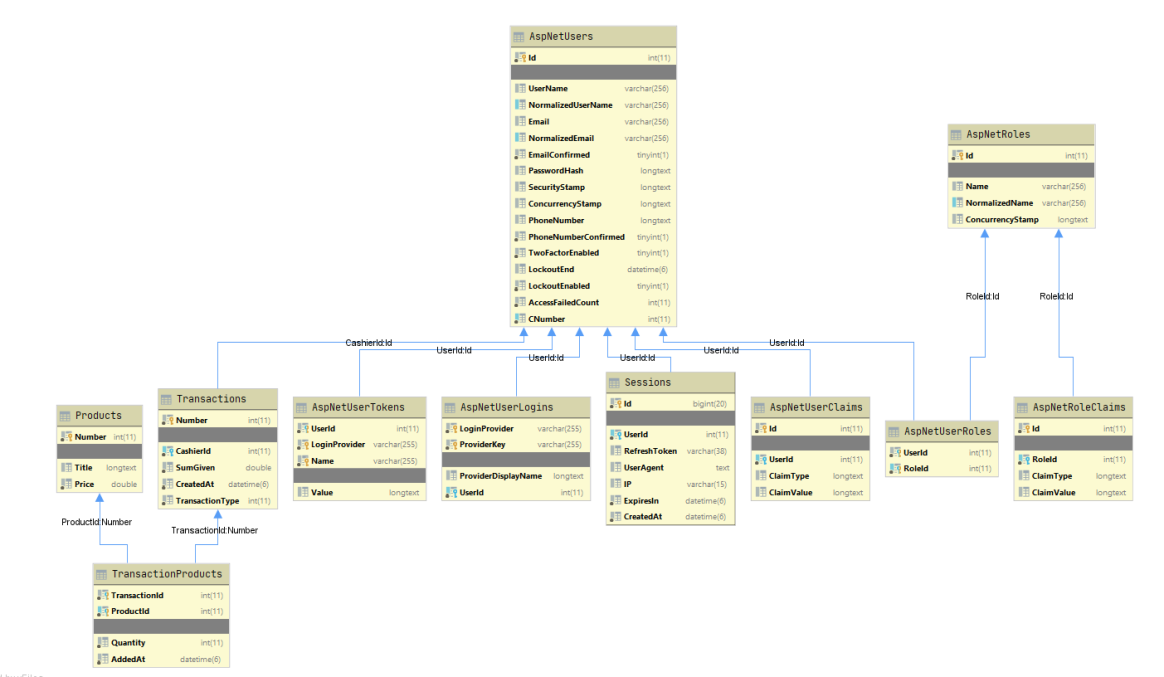


Рис. 2.3.1. *ER*-диаграмма

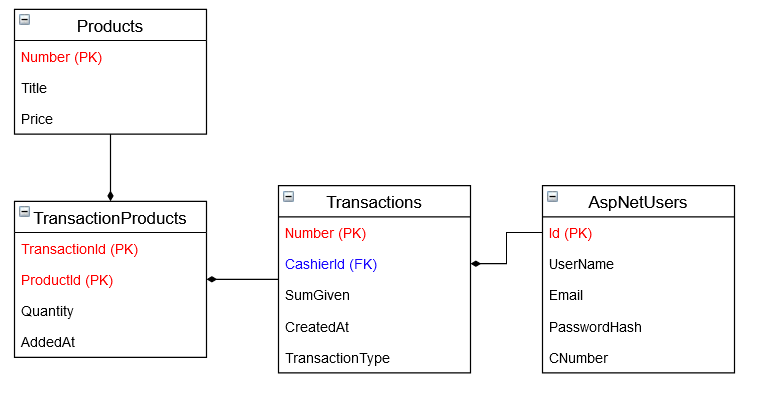
А также в виде Инфологической модели, которая показана на рис 2.3.2.:

Рис. 2.3.2. Инфологическая модель

Структура отношения Кассир можно представить в виде таблицы 2.3.1:

Таблица 2.3.1

Структура отношения Кассиры

|  |  |
| --- | --- |
| **Имя поля** | **Тип данных** |
| Код Кассира | Числовой |
| Логин | Текстовый |
| Пароль | Текстовый |

Структуру отношения Транзакции можно представить в виде таблицы 2.3.1.

Таблица 2.3.2.

Структура отношения Транзакции

|  |  |
| --- | --- |
| **Имя поля** | **Тип данных** |
| Номер транзакции | Числовой |
| Дата | Время |
| Массив мебели | Массив |
| Внесённая сумма | Числовой |
| Итоговая сумма | Числовой |
| Кассир | Объект |
| Сдача | Числовой |

Таблица «Пользователи» содержит общую информацию о пользователях/ кассирах. (Рисунок 2.3.3.)

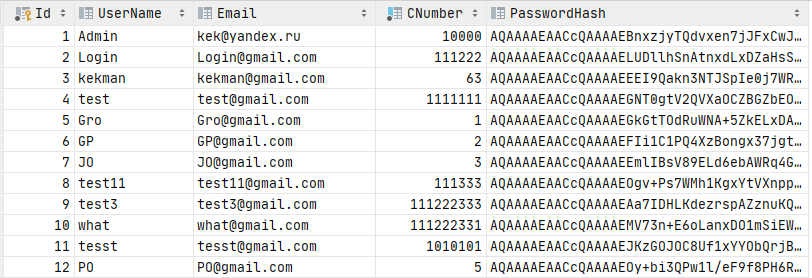


Рисунок 2.3.3. Основная часть структура таблицы «Пользователи» в обозревателе DataGrip

Таблица «Транзакции» содержит информацию о добавленных транзакциях. (Рисунок 2.3.4.)

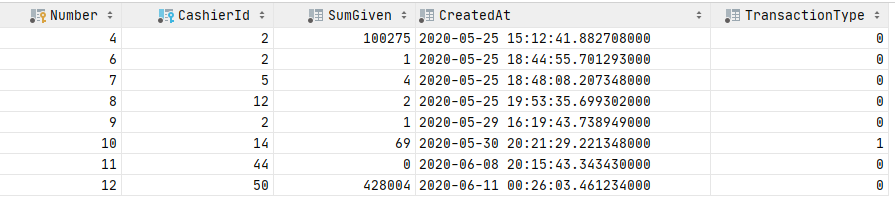


Рисунок 2.3.4. Структура таблицы «Транзакции» в обозревателе DataGrip

Таблица «Сессии» содержит информацию о текущих авторизованных сессиях. (Рисунок 2.3.5.)

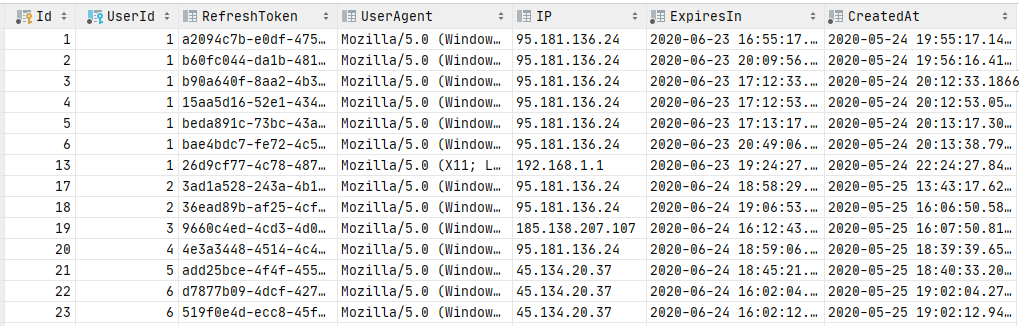


Рисунок 2.3.5 Структура таблицы «Сессии» в обозревателе DataGrip

## Описание процедур и функций

Данная программа состоит из 10 основных форм, класса для перемещения данных между формами и класса, который отвечает за запуск программного продукта. Каждая форма имеет функции и методы для полноценной работы программы. Листинги данных методов располагаются с [Приложения 1](#_AuthPage.vue_Листинг_-) Далее рассмотрим формы и их методы и функции:

* AuthPage – Имеет 2 режима - страница входа\ страница регистрации. На них расположены поля ввода логина и пароля в случае авторизации, и поля ввода логина, пароля и кода кассира в случае регистрации соответственно. Имеет методы отсылки данных на сервер.
* MainPage - это окно главного меню. Имеет только методы, вызываемые элементами формы;
* AddPage– страница для добавления транзакций. Содержит в себе таблицу для заполнения данных о мебели. А также следующие функции:

1. SetFurnutureMass - Формирование массива мебели необходимого размера
2. SetType – Выбор режима транзакции;
3. PostDate – Отправка данных на сервер;

ReportsPage - Страница просмотра транзакций с последующим выбором конкретной транзакции для более подробного изучения. Включает в себя следующие функции:

1. GetReports - GET запрос на сервер для получения списка транзакций.
2. ParseDate - метод формирования даты и времени конкретной транзакции.
3. SearchMehod - метод поиска по транзакциям (номер, цена, имя)
4. ShowModal - метод открывающий окно с дополнительной информацией по выбранной транзакции.
5. DeleteTrasiction - DELETE запрос на сервер для удаления транзакции.
6. Print - метод для вызова окна печати отчёта.

Серверная часть построена по архитектуре MVC (Model-View-Controller) и состоит из 3-х контроллеров:

1. AppController - отвечает за отдачу клиентского SPA приложения в ответ на на любой запрос, URL которого не удалось соотнести с каким-либо другим контроллером.
2. UsersController - предоставляет пути для всех действий, связанных с пользователями. В частности:
   1. POST /api/users - путь для добавления нового пользователя;
   2. DELETE /api/users/{number} - путь для удаления пользователя с указанным номером;
   3. POST /api/users/auth - путь для авторизации пользователя в системе;
   4. POST /api/users/refresh - путь для обновления пары JWT токенов.
3. TransactionsController - обеспечивает выполнение операций, связанных с транзакциями. В частности:
   1. POST /api/transactions - путь для добавления новой транзакции;
   2. GET /api/transactions - путь для получения списка всех транзакций;
   3. DELETE /api/transactions/{number} - путь для удаления транзакции с указанным номером.

При этом все пути TransactionsController-а требуют, чтобы пользователь был авторизован. В ином случае пользователь получит ошибку HTTP 401.

Описанные контроллеры не содержат в себе логики непосредственной манипуляции с базой данных. При помощи инъекции зависимостей (DI) они получают доступ к сервисам, которые уже выполняют операции над моделями. Есть три отдельных сервиса:

1. JwtAuthService - отвечает за формирование и обновление JWT токенов и хранение в СУБД списка активных сессий. Каждая сессия состоит из refresh token-а, IP пользователя, User-Agent-а пользователя и срока годности refresh token-а. Хранение такого набора данных позволяет по необходимости отзывать refresh token-ы, а также исключить вероятность получения доступа к защищённым частям приложения третьими лицами при помощи перехвата access token-а. JwtAuthService не зависит от сторонних элементов базы данных и структуры модели пользователя.
2. UsersService - предоставляет необходимый для UsersController функционал. Для авторизации пользователя и обновления пары токенов использует JwtAuthService, который получает через DI.
3. TransactionsService - сервис для управления транзакциями. Для связи транзакции с пользователем используется UsersService, также получаемый через DI.

### Разработка автоматизированной системы

Описание структуры модулей программы:

Модуль авторизации.

Предназначение – для защиты от посторонних лиц в личные данные кассира, требуется создать систему авторизации.

Главные задачи:

1. Авторизация зарегистрированного пользователя с помощью заполнения соответствующих полей в форме.

Реализация:

Для создания данной формы, был создана HTML разметка с интеграцией фреймворка VUE. При заполненных полях отправляется запрос на сервер с POST запросом на авторизацию пользователя, в теле ответа получается JWT Token предназначенный для идентификации пользователя в системе при отправке последующих запросов.

На рисунке 2.5.1 представлено окно авторизации:

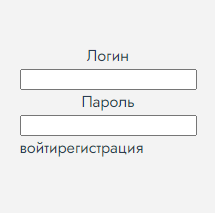


Рис. 2.5.1. Окно авторизации

Модуль регистрации

Предназначение – для добавления новых кассиров в систему.

Главные задачи:

Возможность регистрации нового пользователя в системе

Реализация:

Для создания данной формы, был создана HTML разметка с интеграцией фреймворка VUE. При заполненных полях отправляется запрос на сервер с POST запросом на регистрацию, данные нового пользователя – заносятся в базу данных для последующей авторизации.

На рисунке 2.5.2 представлено окно регистрации, чтобы попасть в нее из формы регистрации, требуется нажать на кнопку «Регистрация» в форме авторизации. Далее требуется заполнить все поля и нажать на кнопку «Регистрация», чтобы в будущем была возможность авторизации в программе. Либо нажать на кнопку «Назад», чтобы вернуться к форме авторизации.

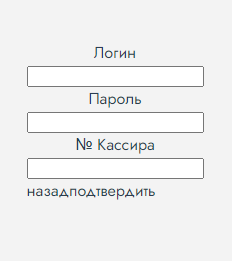


Рис.2.5.2. Окно регистрации

Модуль главного меню

Предназначение – для навигации по системе.

Главные задачи:

1. Отображение имени пользователя;
2. Выбор пунктов меню с последующим переходом в модуль.

Реализация:

Для создания данной формы, был создана HTML разметка с интеграцией фреймворка VUE. Для отображения имени берётся заголовок JWT Token’a из localStorage в котором содержится имя и код Кассира. Пункты меню заключают в себя HTML теги с прописными CSS классами.

На рисунке 2.5.3. представлено окно главного меню. На выбор представлены три кнопки для перехода между модулями:

* Произвести операцию;
* Все отчёты;
* Выход.

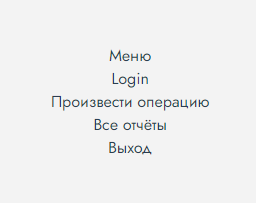


Рис. 2.5.3. Главное меню

Модуль Операций

Предназначение – добавление транзакций определённого типа, в базу данных.

Главные задачи:

1. Отправка отчёта;
2. Выбор необходимой операции (Продажа\Возврат);
3. Автоматический подсчёт сдачи, выдаваемой клиенту.

Реализация:

Для создания данной формы, был создана, HTML разметка с интеграцией фреймворка VUE а так же JS скрипты. Верхняя часть включает в себя выбор операции, созданный с помощью HTML тега select, кнопки назад – возвращающей к главному меню и отображением способа оплаты. Главная часть модуля включает в себя выбор количества мебели, реализованный с помощью input тега и Vue атрибута v-model, для отслеживания изменений – и последующего изменения количества мебели, так же массива строк, отвечающих за содержание транзакции и кнопки «Отправить» отправляющей занесённые данные на сервер. Нижняя часть включает в себя input поле внесено, отображение суммы, подсчитываемой с помощью Vue метода computed, который проходит по всему массиву строк, берёт цену за 1ед и умножает на количество. А также сдача, высчитываемая Внесено – Сумма.

На рисунках 2.5.4. и 2.5.5 представлены операции «Продажа» и «Возврат».

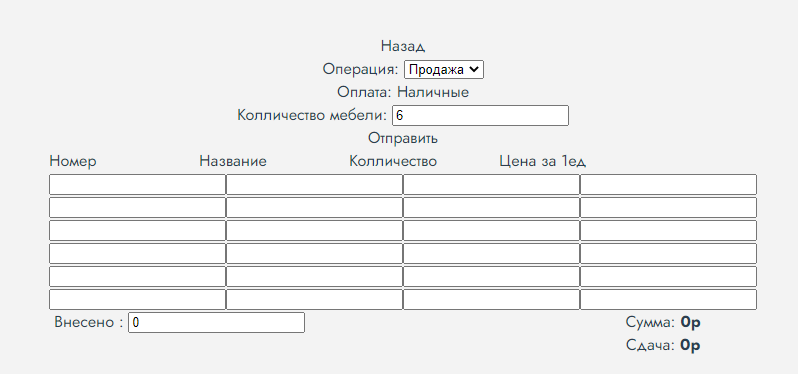


Рис. 2.5.4. Операция «Продажа»

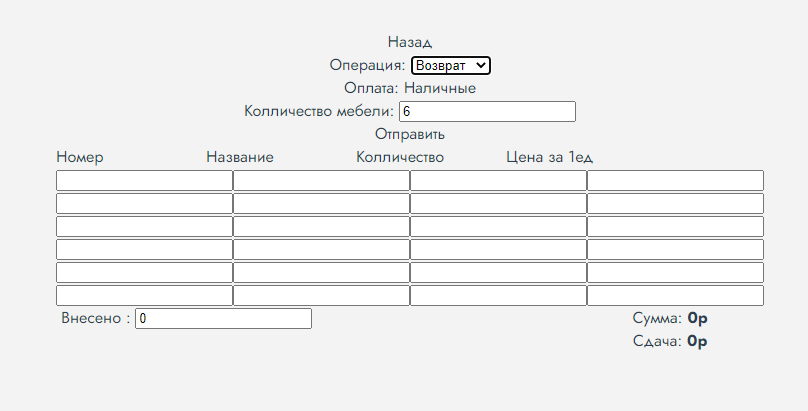


Рис. 2.5.5. Операция «Возврат»

Модуль Транзакций

Предназначение – Просмотр, сортировка и управление транзакций.

Главные задачи:

1. Отображение транзакций;
2. Сортировка и поиск транзакций;
3. Подробный просмотр отдельной транзакции;
4. Удаление транзакций;
5. Печать транзакции.

Реализация:

Для создания данной формы, был создана, HTML разметка с интеграцией фреймворка VUE, а также JS скрипты. Модуль транзакций включает в себя input поле для поиска транзакций, и selector для сортировки транзакций по виду операции, также список всех транзакций. По нажатию на транзакцию, она раскрывается в более подробный вид, где указаны дополнительные данные, а так же кнопка «Удалить» по нажатии на которую отправляется DELETE запрос на север, с id транзакции в заголовке. Так же кнопка печать которая вызывает функцию print() выполняя которую браузер открывает локальное модальное окно для печати страницы.

На Рис. 2.5.6. представлены отчеты по продажам и возвратам. Данные в таблице: № транзакции, имя, время, сумма.

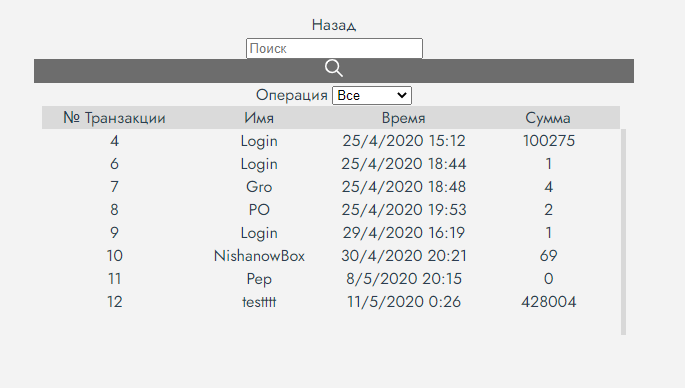


Рис. 2.5.6. Отчеты по продажам и возвратам.

На рисунке 2.5.7. Приведена одна из транзакций кассира NishanowBox. В данном окне представлены дополнительные сведения транзакции такие как: название товара, количество, цена за 1 единицу, сам Кассир, общая стоимость товара и время добавления. Также можно распечатать эту транзакцию, нажав на кнопку печать. Пример приведен на рисунке 2.5.8.



Рис.2.4.7. Транзакция №10 от NishanowBox

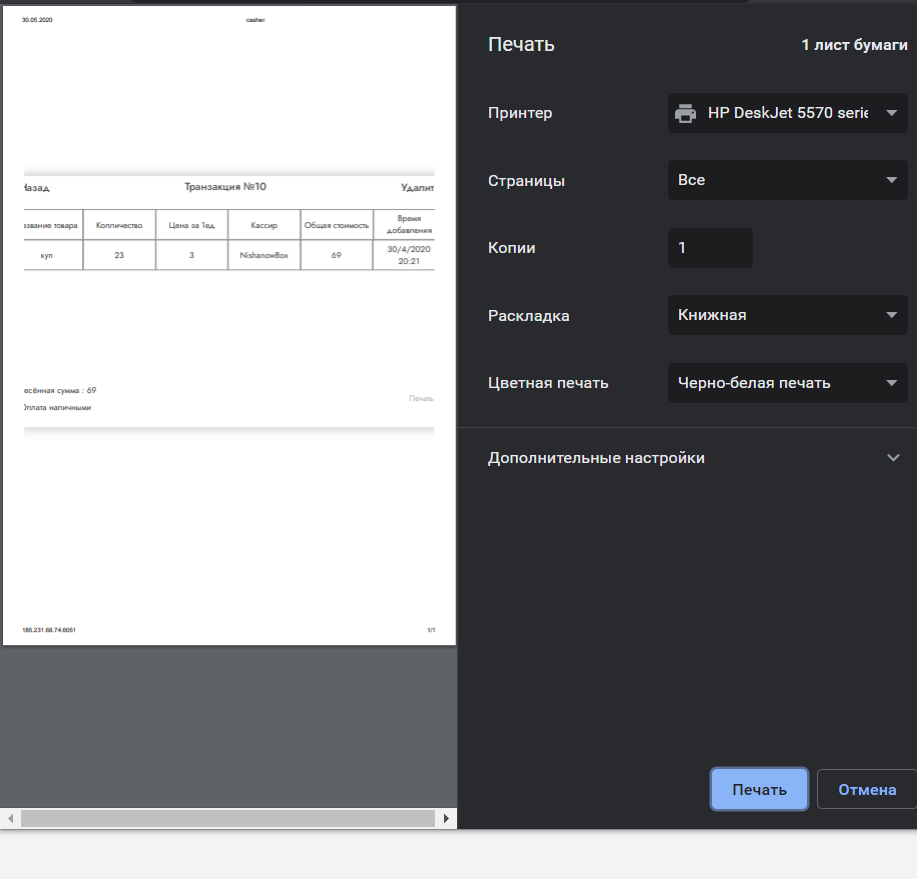


Рис.2.4.8. Печать транзакции №10 от NishanowBox.

### Описание тестовых наборов

1. **Авторизация**

**Константы:** Логин для кассира Нишанова = NishanowBox, Пароль = Nish1234

**Входные данные:** Логин, пароль.

**Тестовый сценарий работы:** Кассир авторизуется под своей учетной записью, вводя логин и пароль. Логин и пароль могут состоять только из латинских букв и цифр. В случае если кассир включит русскую раскладку или допустит ошибку в логине или пароле, то программа выдаст сообщение о неправильном вводе. А если кассир введет все правильно, то программа выдаст сообщение о успешном входе.

**Ожидаемый результат:** Сообщение об успешном входе.

**Тестовые наборы:**

1) ввожу логин: NishanowBox

Ввожу пароль: Nish1234

Результат: сообщение об успешном входе.

2) ввожу логин: 1234йцук

Ввожу пароль: qwer4321

Результат: неверный логин или пароль.

3) ввожу логин: 1234qwer

Ввожу пароль: йцук4321

Результат: неверный логин или пароль.

4) ввожу логин: 1234йцук

Ввожу пароль: йцук4321

Результат: неверный логин или пароль.

5) оставляю поле с логином или паролем пустыми

Результат: пожалуйста проверьте правильность введенных данных.

1. **Меню**

**Тестовый сценарий работы:** После того, как кассир успешно авторизовался, ему в поле выходит меню, в котором он может выбрать режим оплаты или возврата товара, а также выйти с аккаунта.

**Ожидаемый результат:** выбор между элементами меню «Оплата товара», «Возврат товара» и «Выход».

1. **Оплата товара.**

**Тестовый сценарий работы:** Открывается окно с интернет-корзиной. Кассир оформляет заказ от покупателя по штрихкоду от каждого товара и все данные о товаре (Наименование мебели, цена за 1 шт., кол-во и сумма) добавляются в интернет-корзину. Сумма высчитывается по формуле **цена \* кол-во = сумма.** Также под этим списком товаров есть еще и данные «Итого», в котором записывается суммарная цена за все товары. Кассир подсчитывает все это дело, покупатель платит и получает квитанцию.

**Ожидаемый результат:** Подсчет суммы и итого по формуле и вывод этих данных на экране.

**Тестовый набор:**

Наименование товара: стул

Цена: 20000

Кол-во: 4

Сумма = цена \* кол-во = 80000

1. **Возврат товара.**

**Тестовый сценарий работы:** Кассир оформляет возврат товара от покупателя. Если есть чек и штрих-код на мебели остался и не истек срок гарантии, то покупатель может вернуть товар, в противном случае – нет. Кассир пробивает чек и штрих-код на подлинность. В случае если чек и штрих-код действителен, то покупатель может вернуть товар и получить за него обратно деньги, в противном случае нет.

**Ожидаемый результат:** Возврат товара и цены за него.

1. **Выход.**

**Тестовый сценарий работы:** Кассир может просто выйти с аккаунта, нажав на кнопку выход и подтвердив свой выбор.

**Ожидаемый результат:** Выход из программы.

# 3. Эксплуатационная часть

## 3.1. Эксплуатационный раздел

## 3.1.1. Рекомендации по эксплуатации программного продукта

Разработанное программное обеспечение работает по технологии «Клиент–Сервер», что предполагает выполнение большинства операций на сервере. Клиенту необходим только доступ к Интернету или локальной сети и наличие на клиентском рабочем месте браузера. В таблице приведены характеристики серверной части, так как клиентская часть может быть любой – как настольным компьютером или ноутбуком, так и мобильным телефоном.

Таблица 3.1.

Характеристики серверной части

|  |  |
| --- | --- |
| Чипсет | Intel G45 Express |
| Процессоры | Intel Core 2 Duo / Quad |
| Память | 8GB DDR2-800 |
| Жёсткие диски | 1 Tb, RAID |
| Блок питания | 500W |
| Корпус | MINI TOWER |

Для обеспечения дополнительной защиты от сбоев электропитания, необходимо снабдить сервер бесперебойным источником питания мощностью не менее 700–800 Вт, что позволит защититься от кратковременных отключений электроэнергии и продолжить или штатно завершить работу при более продолжительных.

На сервере должен быть установлен Microsoft Internet Information Services, а так же СУБД MySQL.

## 3.1.2. Руководство пользователя

Для начала работы веб-интерфейсом APM-кассира необходимо набрать адрес веб-интерфейса в адресной строке браузера. В случае, если сервер находится в рабочем состоянии, то через 3–5 секунд в окне браузера должен отобразиться веб-интерфейс, в противном случае необходимо проверить наличие подключения к сети, а затем обратиться к системному администратору для выяснения причин недоступности веб-интерфейса. Интерфейс показан на рис 3.1.

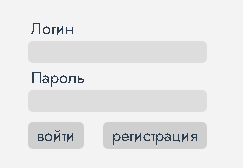


Рис. 3.1. Форма авторизации

После загрузки веб-интерфейса необходимо ввести учётные данные (логин и пароль) или зарегистрироваться, если у Вас нет учётных данных. В случае указания верных данных для входа. Произойдёт авторизация в системе и Вы попадёте в окно для просмотра операций, в случае, если указаны неверные учётные данные, необходимо проверить правильность ввода имени пользователя и пароля и повторить попытку.

В случае, если пользователь не помнит свои учётные данные, необходимо воспользоваться системой восстановления логина или пароля по ссылке, расположенной на странице входа.

Для начала работы необходимо выбрать, какую операцию необходимо совершить – выполнить продажу товара покупателю или выполнить возврат товара, либо просмотреть отчёт о продажах или возвратах. (Рис 3.2.)

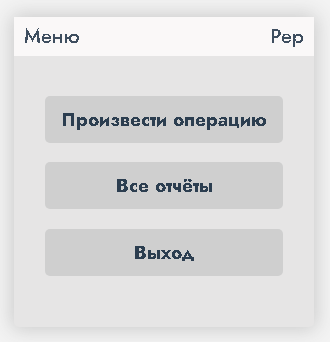


Рис. 3.2. Главное окно.

Для выполнения операции продажи необходимо выполнить щелчок по кнопке «Произвести операцию», после чего на экране отобразиться интерфейс, представленный на рис. 3.3.

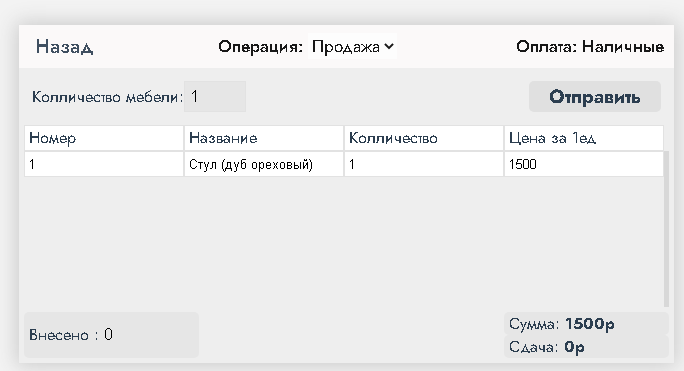


Рис. 3.3. Процесс оформления покупки

В данном интерфейсе необходимо выбрать количество покупок, для чего мышкой щелкнуть на счётчик «Количество мебели», щелчок по верхней стрелке увеличивает счётчик, по нижней – уменьшает.

Далее необходимо ввести порядковый номер, название мебели, а так же количество. При необходимости повторить операцию.

Далее необходимо ввести сумму, полученную от клиента, после чего произойдет автоматический расчёт сдачи, который отобразиться справа внизу. После возврата клиенту сдачи, необходимо нажать кнопку «Отправить» – при этом происходит отправка данных о покупке по защищённому каналу на сервер.

Выполнение операции возврата выполняется аналогичным образом – необходимо знать номер чека, который вводится в окне, далее производится возврат.

Поиск транзакций выполняется или по номеру транзакции, или по дате. Так же существует возможность вывести весь список.

## 3.1.3. Руководство системного администратора

Общие сведения о программном обеспечении:

Программное обеспечение представляет собой веб-интерфейс к автоматизированному рабочему месту кассира. Особых требований к клиентской части не предъявляется; параметры серверной части указаны в разделе 3.1.

Программное обеспечение должно включать в себя:

– операционную систему Windows 7/10 или Windows Server 2012;

– веб-сервер – Internet Information Services 10.0

– СУБД MySQL.

Для запуска веб-интерфейса необходимо скопировать папку с файлами приложения в заранее подготовленную папку IIS, выполнить перезапуск IIS.

# ЗАКЛЮЧЕНИЕ

В современном обществе работа фирмы зависит от скорости обработки информации, как внутренней, циркулирующие внутри неё, так и внешней – входящей, либо исходящей, так как предприятие являются частью глобальной телекоммуникационной инфраструктуры обработки и обмена информации.

Особое значение использование информационных технологий приобретает в сфере торговли, где необходим информационный обмен с органами налоговой инспекции. Так же необходимо оформлять покупателю фискальный чек, подтверждающий факт покупки товара или оплаты услуг. В выпускной квалификационной работе было выполнено проектирование и отладка программного обеспечения для рабочего места кассира. Программное обеспечение предполагается установить на рабочем месте в фирме «Нионка». В ходе выполнения выпускной квалификационной работы: проведен анализ особенностей процесса продажи товара покупателю с применением контрольно-кассовой техники и алгоритмы взаимодействия торговой точки с серверами налоговой инспекции; рассмотрены существующие аналоги АРМ кассира, определены их сильные и слабые стороны.

Разработанное в процессе выполнения ВКР программного обеспечение написано на языке программирования высокого уровня C# на основе принципов объектно-ориентированного программирования. Базовой технологией является ASP.NET. В IDE MS Visual Studio реализованы программные классы и бизнес-логика работы приложения. Хранения информации об именах пользователей, товарах, транзакциях и прочих параметрах выполняется в СУБД MySQL. Защита от несанкционированного доступа осуществляется с использованием ролевого принципа доступа, а так же шифрования некоторых полей базы данных. Работа продавцов в программе реализована с использованием механизма сессий. Процедуры обмена информацией между модулями, которые связаны с передачей информации реализуются с использованием транзакционного подхода, который предполагает, что операция либо выполняется полностью, либо не выполняется совсем.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мирошниченко Г.А. Реляционные базы данных: практические приемы оптимальных решений. – СПб.: БХВ-Петербург, 2005. – 400 с.: ил.
2. Полякова Л.Н. Основы SQL: учебное пособие /Л.Н. Полякова. – 2-е изд. испр. –М.: Интернет-Университет Информационных технологий; БИНОМ. Лаборатория знаний, 2007. – 223 с. : ил., табл. (Основы информационных технологий)
3. Смирнов С.Н., Киселёв А.В. Практикум по работе с базами данных. – М.: Гелиос АРВ, 2012. – 160 с.
4. Документация по C#: https://docs.microsoft.com/ru-ru/dotnet/csharp/
5. MySQL: <https://www.mysql.com/>
6. Visual Studio Code: [https://code.visualstudio.com/](https://vk.com/away.php?to=https%3A%2F%2Fcode.visualstudio.com%2F&cc_key=)
7. Прогрессивный JavaScript-фреймворк Vue.js: <https://ru.vuejs.org/>
8. Rider: The Cross-Platform .NET IDE from JetBeans: [https://www.jetbrains.com/rider/](https://vk.com/away.php?to=https%3A%2F%2Fwww.jetbrains.com%2Frider%2F&cc_key=)
9. DataGrip: The Cross-Platform IDE for Databases and SQL by JetBeans: [https://www.jetbrains.com/datagrip/](https://vk.com/away.php?to=https%3A%2F%2Fwww.jetbrains.com%2Fdatagrip%2F&cc_key=)

**Приложение 1.** **Страница Авторизации -**

<template>

    <div class="enter-page">

        <div class="enter-block" v-if="!add">

            <div class="login">

                <div>Логин</div>

                <input v-model="username" type="text">

            </div>

            <div class="password">

                <div>Пароль</div>

                <input v-model="password" type="password">

            </div>

            <div class="btns">

                <div class="btn" @click="authSend">

                    войти

                </div>

                <div class="btn" @click="add = true">

                    регистрация

                </div>

            </div>

        </div>

        <div class="register-block" v-if="add">

            <div class="login">

                <div>Логин</div>

                <input v-model="username" type="text">

            </div>

            <div class="password">

                <div>Пароль</div>

                <input v-model="password" type="password">

            </div>

            <div class="num-casher">

                <div>№ Кассира</div>

                <input v-model="cNumber" type="number">

            </div>

            <div class="btns">

                <div class="btn" @click="add = false">

                    назад

                </div>

                <div class="btn" @click="userAdd">

                    подтвердить

                </div>

            </div>

        </div>

    </div>

</template>

<script>

import { authService } from "../components/AuthService/authService";

import { http } from "../components/AuthService/httpService";

export default {

    props:{

        back:{

            type: Function,

            default: ()=>{}

        }

    },

    data(){

        return{

            add: false,

            username: "",

            password: "",

            cNumber: ""

        }

    },

    methods:{

        authSend(){

            authService.authorize(this.username,this.password)

            .then((response)=>{

                if(response < 400){

                    this.back();

                    localStorage.setItem('authorize', 1);

                }

            })

            .catch((error)=>{

                if(error.response.status >= 400){

                    alert(error.response.data.title);

                }

            })

        },

        userAdd(){

            http.post('api/users',{

                password: this.password,

                username: this.username,

                cNumber: Number(this.cNumber)

            })

            .then((response)=>{

                if(response.status < 400){

                    alert("Пользователь успешно зарегистрирован");

                    this.add = false;

                }

            })

            .catch((error)=>{

                if(error.response.status >= 400){

                    alert(error.response.data.title);

                }

            })

        },

    },

    mounted() {

        if(localStorage.getItem('authorize') == 1){

            this.back();

        }

    },

}

</script>

<style lang="scss">

    .login,.password,.num-casher{

        div{

            padding: 3px 3px 0;

            width: 100px;

            text-align: start;

        }

        input{

            border-radius: 5px;

            background: rgb(221, 221, 221);

            border: unset;

            padding: 3px 5px;

            outline: unset;

        }

    }

    .btns{

        display: flex;

        justify-content: space-between;

        margin: 10px 0;

    }

    .btn{

        padding: 2px 9px;

        border-radius: 5px;

        display: flex;

        align-items: center;

        cursor: pointer;

        transition: .2s;

        background: rgb(207, 207, 207);

        &:hover{

            color: white;

            background: rgb(165, 165, 165);

        }

    }

</style>

**Приложение 2. Страница Меню -**

<template>

    <div id="app">

        <div class="menu" v-if="page == 0">

            <div class="header">

                <div class="info">

                    Меню

                </div>

                <div class="name">

                    {{cashierName}}

                </div>

            </div>

            <div class="menu-elements">

                <div class="element" @click="page = 1">

                    Произвести операцию

                </div>

                <div class="element" @click="page = 2">

                    Все отчёты

                </div>

                <div class="element" @click="exit">

                    Выход

                </div>

            </div>

        </div>

        <addData

            v-if="page == 1"

            :back="toMenu"

        />

        <reports

            v-if="page == 2"

            :back="toMenu"

        />

        <authPage

            v-if="page == 3"

            :back="toMenu"

        />

    </div>

</template>

<script>

import addData from "./components/addData";

import reports from "./components/reports";

import authPage from "./components/authPage";

export default {

    name: 'App',

    components:{

        addData, reports, authPage

    },

    data(){

        return{

            page: 3

        }

    },

    computed:{

        cashierName(){

            return localStorage.getItem('name');

        }

    },

    methods:{

        toMenu(){

            this.page = 0;

        },

        exit(){

            this.page = 3;

            localStorage.clear();

        }

    }

}

</script>

<style lang="scss">

@import './assets/fonts.scss';

\*{

    margin: 0;

}

body{

    display: flex;

    align-items: center;

    justify-content: center;

    height: 100vh;

    background: rgb(243, 243, 243);

}

#app {

    font-family: 'Jost', sans-serif;

    -webkit-font-smoothing: antialiased;

    -moz-osx-font-smoothing: grayscale;

    text-align: center;

    color: #2c3e50;

}

::-webkit-scrollbar {

    width: 5px;

}

::-webkit-scrollbar-track {

    background: rgb(216, 216, 216);

}

::-webkit-scrollbar-thumb {

    background: #888;

    &:hover{

        background: #555;

    }

}

.menu{

    display: flex;

    flex-direction: column;

    align-items: center;

    background: rgb(230, 229, 229);

    padding: 0 0 10px;

    width: 300px;

    height: 300px;

    border-radius: 5px;

    box-shadow: 0px 0px 19px 3px #cccccc;

}

.header{

    width: 100%;

    display: flex;

    justify-content: space-between;

    padding: 5px 10px;

    box-sizing: border-box;

    font-size: 20px;

    font-weight: 500;

    opacity: .9;

    background: rgb(253, 251, 251);

}

.back{

    padding: 2px 6px;

    cursor: pointer;

    transition: .2s;

    border-radius: 5px;

    &:hover{

        background: rgb(199, 199, 199);

    }

}

.menu-elements{

    margin-top: 40px;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: space-between;

    height: 60%;

}

.element{

    padding: 10px 8px;

    background: rgb(207, 207, 207);

    border-radius: 5px;

    width: 100%;

    font-weight: 600;

    font-size: 19px;

    cursor: pointer;

    transition: .2s;

    &:hover{

        color: white;

        background: gray;

        transform: scale(1.05);

    }

}

</style>

**Приложение 3. Страница Операции -**

<template>

    <div class="operation-block">

        <div class="header">

            <div class="back" @click="back">

                Назад

            </div>

            <div class="what-opearation">

                Операция:

                <select v-model="operation" name="" id="">

                    <option :value="0">Продажа</option>

                    <option :value="1">Возврат</option>

                </select>

            </div>

            <div class="pay-method">

                Оплата: Наличные

            </div>

        </div>

        <div class="main-block">

            <div class="settings">

                <div class="chose-rows">

                    Колличество мебели:

                    <input type="number" v-model="rows">

                </div>

                <div class="send" @click="postData">

                    Отправить

                </div>

            </div>

            <div class="info-row">

                <div class="num">Номер</div>

                <div class="name">Название</div>

                <div class="quantity">Колличество</div>

                <div class="price">Цена за 1ед</div>

            </div>

            <div class="rows">

                <div v-for="(f,k) in mass" :key="k" class="row">

                    <input v-model="f.number" type="text">

                    <input v-model="f.name" type="text">

                    <input v-model="f.quantity" type="number">

                    <input v-model="f.price" type="number">

                </div>

            </div>

        </div>

        <div class="money-block">

            <div class="introduced" >

                Внесено : <input type="number" v-model="intrd"/>

            </div>

            <div class="about-mn">

                <div class="amout">Сумма: <span>{{sum}}р</span></div>

                <div class="change">Сдача: <span>{{intrd - sum > 0 ? intrd - sum : 0}}р</span></div>

            </div>

        </div>

    </div>

</template>

<script>

import { http } from './AuthService/httpService';

export default {

    props:{

        back:{

            type: Function,

            default: ()=>{}

        }

    },

    data(){

        return{

            rows: 6,

            intrd: 0,

            mass: [],

            operation: 0

        }

    },

    methods:{

        pushMass(mass){

            this.mass = mass;

        },

        postData(){

            for(let i of this.mass){

                if (i.number == "" || i.name == "" || i.quantity == "" || i.price == ""){

                    alert("Ошибка: Остались не заполненные поля \n Заполните поля, либо же выберете меньшее количество мебели");

                    return;

                }

            }

            http.post('/api/transactions',{

                type: this.operation,

                cashierNumber: 301231,

                products: this.mass.map(i => ({

                    number: Number(i.number),

                    title: i.name,

                    quantity: Number(i.quantity),

                    price: Number(i.price),

                })),

                sumGiven: this.sum

            })

            .then(()=>{

                this.back()

            })

        }

    },

    computed: {

        allFurnuture(){

            let mass = [];

            for(let i = 0; i< this.rows; i++){

                mass.push({

                    number: '',

                    name: '',

                    quantity: '',

                    price: ''

                });

            }

            this.pushMass(mass);

            return mass;

        },

        sum(){

            let sum = 0;

            console.log(this.allFurnuture.length);

            for(let i of this.mass){

                sum += (i.price ? Number(i.price) : 0) \* ( i.quantity ? Number(i.quantity) : 0);

            }

            return sum;

        }

    },

}

</script>

<style lang="scss" scoped>

    .operation-block{

        display: flex;

        flex-direction: column;

        align-items: flex-start;

        background: rgb(238, 238, 238);

        box-shadow: 0px 0px 19px 3px #cccccc;

    }

    .header{

        align-items: center;

    }

    .what-opearation,.pay-method{

        font-weight: 500;

        font-size: 17px;

        color: black;

        select{

            font-family: 'Jost';

            cursor: pointer;

            outline: unset;

            font-size: 17px;

            padding: 0px;

            margin: 0px;

            border: unset;

        }

    }

    .main-block{

        padding: 0px 5px 5px;

    }

    .settings{

        display: flex;

        justify-content: space-between;

        padding: 8px;

        margin: 5px 0;

    }

    .chose-rows{

        display: flex;

        align-items: center;

        input{

            background: rgb(230, 230, 230);

            border: unset;

            padding: 5px;

            border: 1px solid rgb(226, 226, 226);

            outline: unset;

            font-size: 16px;

            width: 50px;

        }

    }

    .send{

        padding: 2px 20px;

        border-radius: 5px;

        background: rgb(224, 224, 224);

        font-weight: 600;

        font-size: 19px;

        transition: .2s;

        cursor: pointer;

        &:hover{

            background: rgb(173, 173, 173);

            color: white;

        }

    }

    .rows{

        height: 156px;

        overflow-y: scroll;

        width: 100%;

    }

    .info-row{

        display: flex;

        width: 100%;

        padding-right: 4px;

        box-sizing: border-box;

        div{

            width: 25%;

            height: 20px;

            background: white;

            padding: 2px 4px;

            width: 150px;

            display: flex;

            align-items: center;

            border: 1px solid rgb(226, 226, 226);

        }

    }

    .row{

        input{

            height: 20px;

            width: 150px;

            padding: 2px 4px;

            background: white;

            border: 1px solid rgb(226, 226, 226);

            outline: unset;

        }

    }

    .money-block{

        display: flex;

        justify-content: space-between;

        width: 100%;

        padding: 0px 5px 5px;

        box-sizing: border-box;

    }

    .introduced{

        background: rgb(230, 230, 230);

        border-radius: 5px;

        display: flex;

        padding: 0 5px;

        box-sizing: border-box;

        align-items: center;

        input{

            background: rgb(230, 230, 230);

            border: unset;

            padding: 5px;

            outline: unset;

            font-size: 16px;

            width: 85px;

        }

    }

    .about-mn{

        width: 25.5%;

    }

    .amout,.change{

        padding: 0 5px;

        box-sizing: border-box;

        background: rgb(230, 230, 230);

        border-radius: 5px;

        text-align: start;

        width: 100%;

        span{

            font-weight: 600;

        }

    }

</style>

**Приложение 4. Модуль Транзакции -**

<template>

    <div class="modal-block">

        <div class="header">

            <div class="back" @click="backToReports">

                Назад

            </div>

            <div class="trans-num">

                Транзакция №{{modalObject.number}}

            </div>

            <div @click="deleteReport" class="back">

                Удалить

            </div>

        </div>

        <div class="main-info">

            <div class="info-row">

                <div class="name">Название товара</div>

                <div class="num">Колличество</div>

                <div class="price">Цена за 1ед</div>

                <div class="price">Кассир</div>

                <div class="price">Общая стоимость</div>

                <div class="price">Время добавления</div>

            </div>

            <div class="rows">

                <div v-for="(f,k) in modalObject.mass" :key="k" class="row">

                    <div>{{f.title}}</div>

                    <div>{{f.price}}</div>

                    <div>{{f.quantity}}</div>

                    <div>{{modalObject.name}}</div>

                    <div>{{f.price \* f.quantity}}</div>

                    <div>{{modalObject.time}}</div>

                </div>

            </div>

        </div>

        <div class="bottom-menu">

            <div class="info">

                <div class="total-price">

                    Внесённая сумма : {{modalObject.price}}

                </div>

                <div class="payment-method">

                    Оплата наличными

                </div>

            </div>

            <div class="print" onclick="print()">Печать</div>

        </div>

    </div>

</template>

<script>

import { http } from './AuthService/httpService'

export default {

    props:{

        backToReports:{

            type: Function,

            default: ()=>{}

        },

        modalObject:{

            type: Object,

            default: ()=>{}

        }

    },

    methods:{

        deleteReport(){

            let ans = confirm('Вы точно хотите удалить?');

            if (ans){

                http.delete('/api/transactions/'+this.modalObject.number)

                .then(()=>{

                    this.backToReports();

                })

            }

        }

    }

}

</script>

<style lang="scss" scoped>

    .modal-block{

        background: rgb(238, 238, 238);

        box-shadow: 0px 0px 19px 3px #cccccc;

        padding: 0 5px 20px;

    }

    .main-info{

        margin-top: 20px;

    }

    .rows{

        height: 260px;

        overflow-y: scroll;

        width: 830px;

    }

    .info-row{

        width: 99.5%;

        background: rgb(206, 206, 206);

    }

    .info-row,.row{

        display: flex;

        box-sizing: border-box;

        div{

            width: 25%;

            display: flex;

            align-items: center;

            justify-content: center;

            padding: 5px;

            border: 1px solid rgb(150, 150, 150);

            box-sizing: border-box;

        }

    }

    .bottom-menu{

        display: flex;

        justify-content: space-between;

        margin-top: 10px;

    }

    .print{

        width: 100px;

        height: 65px;

        display: flex;

        align-items: center;

        justify-content: center;

        border-radius: 5px;

        background: rgb(204, 204, 204);

        transition: .2s;

        cursor: pointer;

        &:hover{

            background: rgb(134, 134, 134);

            color: white;

        }

    }

    .total-price,.payment-method{

        padding: 3px 8px;

        border-radius: 5px;

        background: rgb(206, 206, 206);

        margin: 3px 0;

    }

</style>

**Приложение 5. Страница Транзакций -**

<template>

    <div class="page">

        <div class="report-block" v-if="!modal">

            <div class="header">

                <div class="back" @click="back">

                    Назад

                </div>

            </div>

            <div class="info-search">

                <div class="search">

                    <input v-model="buff" type="text" placeholder="Поиск">

                    <div @click="searchMeth" class="sr-btn">

                        <img src="static/img/search.svg" alt="">

                    </div>

                </div>

                <div class="operation-selector">

                    Операция

                    <select v-model="operation" name="" id="">

                        <option :value="null">Все</option>

                        <option :value="0">Продажа</option>

                        <option :value="1">Возврат</option>

                    </select>

                </div>

            </div>

            <div class="main-block">

                <div class="top-info-row">

                    <div class="trs">№ Транзакции</div>

                    <div class="name">Имя</div>

                    <div class="time">Время</div>

                    <div class="price">Сумма</div>

                </div>

                <div class="rows">

                    <div class="info-row" v-for="(i,k) in searchReq"

                        :key="k"

                        @click="openModal(i)"

                    >

                        <div class="trs">{{i.number}}</div>

                        <div class="name">{{i.cashier.username}}</div>

                        <div class="time">{{parseDate(i.createdAt)}}</div>

                        <div class="price">{{i.total}}</div>

                    </div>

                </div>

            </div>

        </div>

        <modal

            v-if="modal"

            :backToReports="closeModal"

            :modalObject="modalBuff"

        />

    </div>

</template>

<script>

import { http } from "./AuthService/httpService";

import modal from "./modal";

export default {

    props:{

        back:{

            type: Function,

            default: ()=>{}

        }

    },

    components:{

        modal

    },

    data(){

        return{

            transactions: [],

            search: "",

            buff: "",

            operation: null,

            modalBuff:{},

            modal: false

        }

    },

    methods:{

        parseDate(date){

            let d = new Date(date);

            return `${d.getDate()}/${d.getMonth()}/${d.getFullYear()} ${d.getHours()}:${d.getMinutes()}`

        },

        searchMeth(){

            this.search = this.buff;

        },

        openModal(e){

            this.modalBuff = {

                mass: e.products,

                time: this.parseDate(e.createdAt),

                name: e.cashier.username,

                number: e.number,

                price: e.total

            }

            this.modal = true;

        },

        closeModal(){

            this.modalBuff = {};

            this.modal = false;

        },

        getReports(){

            http.get('/api/transactions')

            .then((response)=>{

                this.transactions = response.data;

            })

        }

    },

    computed:{

        searchReq(){

            let s = this.search;

            let mass = [];

            if (this.search == ""){

                if(this.operation){

                    for(let t of this.transactions){

                        if(this.operation == t.change){

                            mass.push(t);

                        }

                    }

                }

                else

                    mass = this.transactions;

            }

            else{

                for(let t of this.transactions){

                    let n = t.number;

                    let un = t.cashier.username;

                    let p = t.total;

                    if((s == n || s== un || s == p) && (this.operation == null ? true : this.operation == t.change)){

                        mass.push(t);

                    }

                }

            }

            return mass

        }

    },

    mounted() {

        this.getReports();

    },

}

</script>

<style lang="scss" scoped>

.report-block{

    width: 600px;

    background: rgb(238, 238, 238);

    box-shadow: 0px 0px 19px 3px #cccccc;

    padding: 0 0 20px;

}

.info-row,.top-info-row{

    display: flex;

    box-sizing: border-box;

}

.info-row{

    cursor: pointer;

    transition: .2s;

    &:hover{

        transform: scale(1.05);

    }

}

.top-info-row{

    width: 99%;

    div{

        background: rgb(218, 218, 218);

    }

}

.info-search{

    display: flex;

    padding: 0 8px;

    margin: 20px 5px 20px 0;

    align-items: center;

    input{

        padding: 5px 8px;

        font-size: 17px;

        border-radius: 5px 0 0 5px;

        background: rgb(252, 252, 252);

        width: 150px;

        border: unset;

        outline: unset;

    }

}

.main-block{

    padding: 0 8px;

}

.rows{

    height: 206px;

    overflow-y: scroll;

    overflow-x: hidden;

}

.trs,.name,.time,.price{

    width: 25%;

    padding: 5px;

    border: 1px solid rgb(150, 150, 150);

    box-sizing: border-box;

}

.search{

    margin-right: 30px;

    display: flex;

    .sr-btn{

        width: 30px;

        border-radius: 0 5px 5px 0;

        height: 30px;

        background: rgb(153, 153, 153);

        display: flex;

        align-items: center;

        justify-content: center;

        cursor: pointer;

        transition: .2s;

        img{

            height: 18px;

        }

        &:hover{

            background: rgb(109, 109, 109);

        }

    }

}

.operation-selector{

    font-size: 17px;

    font-weight: 500;

    select{

        font-family: 'Jost';

        cursor: pointer;

        outline: unset;

        font-size: 17px;

        padding: 0px;

        margin: 0px;

        border: unset;

        background: rgb(252, 252, 252);

    }

}

</style>

**Приложение 6. Сервис Авторизации -**

import axios from "axios";

const ACCESS\_KEY = "@mas\_A6b7258334b5B0020D1078c24DcABE5d\_a";

const REFRESH\_KEY = "@mas\_145dC1da0ede8E020f33263dCE4Dfc33\_r";

const REFRESH\_LOCK\_KEY = "@mas\_3b0596BCFDd27F01b286FCbBF9988841\_lock";

const LAST\_REFRESH\_TIMESTAMP\_KEY = "@mas\_779fFe6CC87E48fDd39F47514660543A\_r\_timestamp";

const REFRESH\_TIMEOUT = 2000; // 2s

class AuthService {

    \_scheme = "Bearer";

    \_authPath = "/api/users/auth";

    \_refreshPath = "/api/users/refresh";

    \_logoutPath = "/api/users/logout";

    \_usernameField = "username";

    \_isRefreshLocked = false;

    \_parsedAccessToken = null;

    \_lastParsedAccessToken = null;

    \_authorizationPromise = null;

    \_autoRefresherTimer = null;

    \_lockPromise = null;

    \_subs = new Set();

    get accessToken() {

        return localStorage.getItem(ACCESS\_KEY) || null;

    }

    get refreshToken() {

        return localStorage.getItem(REFRESH\_KEY) || null;

    }

    get parsedAccessToken() {

        if (this.accessToken != null && this.accessToken != this.\_lastParsedAccessToken) {

            this.\_parsedAccessToken = this.parseJwtToken(this.accessToken);

            this.\_lastParsedAccessToken = this.accessToken;

        }

        return this.\_parsedAccessToken;

    }

    get isAuth() {

        return this.accessToken != null && this.parsedAccessToken.exp \* 1000 > Date.now();

    }

    get isAuthInProgress() {

        return this.isRefreshLocked || this.\_authorizationPromise != null;

    }

    get isRefreshLocked() {

        return localStorage.getItem(REFRESH\_LOCK\_KEY) === "1" || this.\_isRefreshLocked;

    }

    get lastRefreshTimestamp() {

        return parseInt(localStorage.getItem(LAST\_REFRESH\_TIMESTAMP\_KEY) || "0");

    }

    set lastRefreshTimestamp(value) {

        localStorage.setItem(LAST\_REFRESH\_TIMESTAMP\_KEY, value.toString());

    }

    constructor() {

        this.reqMiddleware = this.reqMiddleware.bind(this);

        this.resMiddleware = this.resMiddleware.bind(this);

    }

    init() {

        if (this.isAuth || this.refreshToken == null || this.refreshToken === "") {

            if (this.isAuth && this.\_autoRefresherTimer == null) {

                this.setupAutoRefresher();

            }

        }

        else {

            this.refreshTokens();

        }

        this.setupLockListener();

    }

    /\*\*\*

     \* Subscribes to authorization event

     \* @param event - event to subscribe ('auth' or 'logout')

     \* @param callback function called on auth

     \* @param initialOnly call function only once

     \* @returns {function(...[\*]=)} function to unsubscribe to event

     \*/

    subscribe(event, callback, initialOnly = false) {

        let fired = false;

        if (this.isAuth && event === "auth") {

            callback();

            fired = true;

        }

        const subInfo = { event, callback, initialOnly, fired };

        this.\_subs.add(subInfo);

        return () => {

            this.\_subs.delete(subInfo);

        };

    }

    /\*\*\*

     \* Authorize this client

     \* @param username - username, email, etc. (depends on server config)

     \* @param password

     \* @returns {Promise<void>}

     \*/

    authorize(username, password) {

        return axios.post(this.\_authPath, {

            [this.\_usernameField]: username,

            password

        }, { headers: { "Accept": "application/json"} })

        .then((response)=>{

            if(response.status < 400){

                this.storeTokens(response.data.accessToken, response.data.refreshToken);

                this.lastRefreshTimestamp = Date.now();

                this.setupAutoRefresher();

                this.dispatch("auth");

                localStorage.setItem('name',username);

                this.log("Authorized as " + username);

            }

            else{

                this.log(`Failed to authorize: ${response.status} ${response.statusText}`);

            }

            return response.status

        });

    }

    /\*\*\*

     \* Logout this client

     \* @returns {Promise<void>}

     \*/

    async logout() {

        try {

            this.lockRefresh();

            await axios.post(this.\_logoutPath, {

                refreshToken: this.refreshToken

                },

                {

                    headers: {

                        "Authorization": this.\_scheme + " " + this.accessToken

                    }

            });

            this.clearTokens();

            this.dispatch("logout");

            this.releaseRefresh();

        }

        catch (e) {

            this.log(e);

        }

    }

    /\*\*\*

     \* Refresh JWT tokens

     \* @returns {Promise<void>}

     \*/

    async refreshTokens() {

        try {

            if (this.isRefreshLocked) {

                this.log("[refreshTokens]: refresh is locked! Aborting refresh...");

                return;

            }

            this.lockRefresh();

            const response = await axios.post(this.\_refreshPath, {

                refreshToken: this.refreshToken

            }, { headers: { "Accept": "application/json" } });

            if (response.status < 400) {

                this.storeTokens(response.data.accessToken, response.data.refreshToken);

                this.lastRefreshTimestamp = Date.now();

                this.setupAutoRefresher();

                this.dispatch("auth");

                this.log("Tokens refreshed!");

            }

        }

        catch (e) {

            this.log(`[catch] Failed to refresh: ` + e.toString());

        }

        finally {

            this.releaseRefresh();

        }

    }

    async reqMiddleware(req) {

        await this.authorizationCheck();

        if (this.isAuth) {

            if (!req.headers)

                req.headers = {};

            req.headers["Authorization"] = this.\_scheme + " " + this.accessToken;

        }

        return req;

    }

    async resMiddleware(res) {

        return res;

    }

    dispatch(event) {

        let subsToRemove = [];

        this.\_subs.forEach(sub => {

            if ((!sub.initialOnly || !sub.fired) && sub.event === event) {

                sub.callback();

                sub.fired = true;

                if (sub.initialOnly) {

                    subsToRemove.push(sub);

                }

            }

        });

        subsToRemove.forEach((s) => this.\_subs.delete(s));

    }

    authorizationCheck() {

        if (this.refreshToken == null){

            return

        }

        if (this.\_authorizationPromise != null) {

            return this.\_authorizationPromise;

        }

        this.\_authorizationPromise = new Promise(async (resolve, reject) => {

            await this.checkRefreshLock();

            if (this.isAuth) {

                resolve();

                this.\_authorizationPromise = null;

                return;

            }

            if (this.refreshToken == null || this.refreshToken === "") {

                reject(new Error("Unauthorized: no refresh token"));

                this.\_authorizationPromise = null;

                return;

            }

            this.log("Unauthorized! Starting refresh...");

            try {

                await this.refreshTokens();

                resolve();

            }

            catch (e) {

                reject(e);

            }

            finally {

                this.\_authorizationPromise = null;

            }

        });

        return this.\_authorizationPromise;

    }

    parseJwtToken(token) {

        const payload = token.split(".")[1].replace(/-/g, "+").replace(/\_/g, "/");

        const decoded = decodeURIComponent(atob(payload).split('').map((c) => {

            return '%' + ('00' + c.charCodeAt(0).toString(16)).slice(-2);

        }).join(''));

        return JSON.parse(decoded);

    }

    storeTokens(accessToken, refreshToken) {

        localStorage.setItem(ACCESS\_KEY, accessToken);

        localStorage.setItem(REFRESH\_KEY, refreshToken);

    }

    clearTokens() {

        localStorage.removeItem(ACCESS\_KEY);

        localStorage.removeItem(REFRESH\_KEY);

    }

    setupAutoRefresher() {

        if (this.isRefreshLocked || this.\_autoRefresherTimer != null) {

            return;

        }

        // Random timer offset to reduce chances

        // of multiple tabs causing parallel tokens refresh

        const randomTimeOffset = Math.round(5000 \* Math.random());

        // 5s before token expiration

        const timeoutTime = this.parsedAccessToken.exp \* 1000 - Date.now() - 5000 - randomTimeOffset;

        this.\_autoRefresherTimer = setTimeout(async () => {

            this.log("AutoRefresher is fired");

            this.clearAutoRefresher();

            if (this.isRefreshLocked) {

                this.log("AutoRefresher dismissed (refresh is locked)");

                return;

            }

            else if (Date.now() - this.lastRefreshTimestamp < 60 \* 1000) {

                this.log("AutoRefresher dismissed (refresh is locked)");

                if (this.isAuth) {

                    this.setupAutoRefresher();

                }

                return;

            }

            await this.refreshTokens();

            this.log("AutoRefresher finished work");

        }, timeoutTime);

        this.log("AutoRefresher is set for " + timeoutTime + " ms");

    }

    setupLockListener() {

        window.addEventListener("storage", (e) => {

            this.log("Storage event");

            if (e.key !== REFRESH\_LOCK\_KEY) {

                return;

            }

            this.onRefreshLockToggle(e.newValue == null);

        });

        this.log("Lock listener is set");

    }

    clearAutoRefresher() {

        if (this.\_autoRefresherTimer != null) {

            clearTimeout(this.\_autoRefresherTimer);

            this.\_autoRefresherTimer = null;

        }

    }

    lockRefresh() {

        localStorage.setItem(REFRESH\_LOCK\_KEY, "1");

        this.onRefreshLockToggle();

    }

    releaseRefresh() {

        localStorage.removeItem(REFRESH\_LOCK\_KEY);

        this.onRefreshLockToggle(true);

    }

    onRefreshLockToggle(unlock = false) {

        if (unlock) {

            this.\_isRefreshLocked = false;

            this.log("[Lock event]: Released lock");

            if (this.isAuth) {

                this.setupAutoRefresher();

            }

        }

        else {

            this.log("[Lock event]: Locked");

            this.\_isRefreshLocked = true;

        }

    }

    checkRefreshLock() {

        if (!this.isRefreshLocked) {

            return new Promise(resolve => resolve());

        }

        this.log("Refresh is locked! Awaiting...");

        if (this.\_lockPromise != null) {

            return this.\_lockPromise;

        }

        this.\_lockPromise = new Promise((resolve, reject) => {

            const interval = 50;

            let timePassed = 0;

            let checkInterval = setInterval(() => {

                if (!this.isRefreshLocked) {

                    this.log("Refresh is released! Resolving...");

                    clearInterval(checkInterval);

                    resolve();

                    this.\_lockPromise = null;

                }

                else if (timePassed >= REFRESH\_TIMEOUT) {

                    clearInterval(checkInterval);

                    reject(new Error("Refresh timeout"));

                    this.\_lockPromise = null;

                    this.releaseRefresh();

                }

                timePassed += interval;

            }, interval);

        });

        return this.\_lockPromise;

    }

    log(message) {

        console.log(`[${Date.now()}]: ${message}`);

    }

}

const authService = new AuthService();

authService.init();

export { authService };