ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИМ. Н. П. ОГАРЁВА»**

Факультет довузовской подготовки и среднего профессионального образования

Выпускающая предметная цикловая комиссия (кафедра) общепрофессиональных и специальных (информационно-коммуникационных) дисциплин

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ**

на тему Разработка программы он-лайн касса

по ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем

Автор курсового проекта Р. А. Куроедов

(подпись) (дата)

Обозначение курсового проекта КП–02069964–09.02.03–11–19

Направление подготовки (специальность) 09.02.03 Программирование в компьютерных системах

Руководители проекта:

Преподаватель А. А. Прокин

(подпись) (дата)

Проект защищён \_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата

Саранск

2019

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИМ. Н. П. ОГАРЁВА»**

Факультет довузовской подготовки и среднего профессионального образования

Выпускающая предметная цикловая комиссия (кафедра) общепрофессиональных и специальных (информационно-коммуникационных) дисциплин

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

Студент Куроедов Роман Александрович

1. Тема: Разработка программы он-лайн касса
2. Срок представления проекта к защите 22.11.2019 г.
3. Исходные данные для научного проектирования: техническая документация с описанием программного модуля он-лайн касса, документация по использованию среды разработки Visual Studio 2017 Community
4. Содержание курсового проекта

ВВЕДЕНИЕ

4.1 Теоретическое обоснование проекта

4.2 Проектная часть описания внешнего вида разрабатываемой системы

4.3 Тестирование программного продукта

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А (обязательное) Скриншоты реализованного модуля

ПРИЛОЖЕНИЕ Б (обязательное) Функция перехода по страницам

ПРИЛОЖЕНИЕ В (обязательное) Код авторизации

ПРИЛОЖЕНИЕ Г (обязательное) Код внесения-изъятия средств

ПРИЛОЖЕНИЕ Д (обязательное) Код продажи-возврата товаров

ПРИЛОЖЕНИЕ Е (обязательное) Результаты тестирования

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Прокин

подпись, дата

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_ Р. А. Куроедов

подпись, дата

**РЕФЕРАТ**

Пояснительная записка содержит 43 листа, 5 рисунков, 1 таблицу,   
7 источников, 6 приложений.

ПРОГРАММА, КАССА, СКЛАД, ПРОДАЖА, ВОЗВРАТ, СМЕНА, АВТОРИЗАЦИЯ, ОН-ЛАЙН.

Объектом разработки является программный модуль он-лайн касса, включающий следующие функции:

* авторизация сотрудников предприятия;
* ведение сессионной статистики торгового предприятия;
* осуществление продажи товаров сотрудниками торгового предприятия;
* осуществления возврата товаров клиентами торгового предприятия;
* обновление статистики склада торгового предприятия.

Цель работы ⎯ автоматизация и снижение уровня трудоёмкости следующих производственных процессов:

* ведение сессионной статистики торгового предприятия;
* осуществление продажи товаров сотрудниками торгового предприятия;
* осуществления возврата товаров клиентами торгового предприятия;
* обновление статистики склада торгового предприятия.

В процессе работы использовались знания и опыт, полученные при изучении учебных курсов «Основы программирования» и «Прикладное программирование».

В результате выполнения курсовой работы был получен программный модуль готовый к внедрению и расширению, который включает в себя модули продажи и возврата товаров, ведения сессионной статистики и авторизации пользователей, а также разработан удобный и понятный для пользователя интерфейс.

Эффективность проекта:

* уменьшение количества трудовых и временных ресурсов, необходимых для обслуживания клиентов торгового предприятия;
* уменьшение количества трудовых и временных ресурсов, необходимых, для ведения ресурсной и финансовой отчетности;
* компактное и мало затратное, с точки зрения пространственных и финансовых ресурсов, хранение необходимых предприятию данных;
* быстрый доступ к данным.

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
| ВВЕДЕНИЕ | | 6 |
| 1 Теоретическое обоснование проекта | | 8 |
|  | 1.1 Анализ предметной области | 8 |
|  | 1.2 Выбор методов решения задач | 9 |
| 2 Проектная часть | | 15 |
|  | 2.1 Общая структура приложения | 15 |
|  | 2.2 Функция перехода по страницам | 18 |
|  | 2.3 Функция начала смены | 19 |
|  | 2.4 Функция внесения и изъятия денежных средств | 19 |
|  | 2.5 Функция продажи-возврата товаров | 20 |
| 3 Тестирование программного продукта | | 21 |

|  |  |
| --- | --- |
| ЗАКЛЮЧЕНИЕ | 23 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 25 |
| ПРИЛОЖЕНИЕ А (обязательное) Скриншоты реализованного модуля | 26 |
| ПРИЛОЖЕНИЕ Б (обязательное) Функция перехода по страницам | 28 |
| ПРИЛОЖЕНИЕ В (обязательное) Код авторизации | 29 |
| ПРИЛОЖЕНИЕ Г (обязательное) Код внесения-изъятия средств | 31 |
| ПРИЛОЖЕНИЕ Д (обязательное) Код продажи-возврата товаров | 33 |
| ПРИЛОЖЕНИЕ Е (обязательное) Результаты тестирования | 39 |

**ВВЕДЕНИЕ**

Становление и развитие информационных технологий началось во второй половине 20 века. Тогда мало кто мог представить, что всего через пол века от этих самых информационных технологий общество будет зависеть настолько, что, лишившись их, впадет в хаос.

21 век по праву называется веком информационных технологий, так как они присутствуют на каждом шагу. Ими пользуются все от мала до велика: обычные люди — в повседневной жизни, частные предприятия — в своей деятельности и так далее. Так же информационные технологии широко применяются и на государственном уровне в сферах здравоохранения, политики, обороны, финансов и так далее. В следствие чего применение некоторых видов информационных технологий фиксируется на законодательном уровне. Ярким примером этого является применение контрольно-кассовой техники при осуществлении наличных денежных расчетов и (или) расчётов с использованием платёжных карт.

С апреля 2016 года вступил в силу ряд поправок к 54-ФЗ «О применении контрольно-кассовой техники при осуществлении наличных денежных расчетов и (или) расчётов с использованием платёжных карт». В обновлении закона понятие ЭКЛЗ исчезает. Вместо него появляется фискальный накопитель.

Фискальный накопитель — шифровальные (криптографические) средства защиты фискальных данных в опломбированном корпусе, содержащие ключ фискального признака, обеспечивающие запись перечня фискальных данных, в некорректируемом виде, их энергонезависимое долговременное хранение, формирование фискального признака, аутентификацию электронных документов, направляемых в контрольно-кассовую технику оператором фискальных данных, а также при необходимости пользователя обеспечивающие шифрование фискальных данных, в целях обеспечения конфиденциальности информации, передаваемой оператору фискальных данных (ОФД).

Кассовый программно-технический комплекс передает данные каждого оформленного чека в виде электронного документа оператору фискальных данных, который в свою очередь будет накапливать, хранить и передавать эти данные в ФНС. Оператор фискальных данных осуществляет ежедневную передачу в адрес налоговых органов фискальных данных, а также обеспечивает возможность проверки применения ККТ через интернет и достоверность оформленного кассового чека в виде электронного документа. Пользователь, приобретая новое оборудование, может самостоятельно поставить его на учет в налоговые органы, а также должен заключить договор с одним из ОФД на передачу данных для взаимодействия с налоговой.

Целью курсовой работы является разработка программного модуля   
он-лайн касса.

Для выполнения курсовой работы поставлены следующие задачи:

* изучить и проанализировать предметную область на основе материалов, предоставленных в 54-ФЗ «О применении контрольно-кассовой техники при осуществлении наличных денежных расчетов и (или) расчётов с использованием платёжных карт»;
* разработать модуль ведения сессионной статистики торгового предприятия;
* разработать модуль, осуществляющий возможность продажи товаров и услуг;
* разработать модуль, осуществляющий возможность возврата средств за товары и услуги, приобретённые клиентом, но не удовлетворившие его потребности;
* осуществить тестирование разработанного программного модуля.

**1 Теоретическое обоснование проекта**

* 1. **Анализ предметной области**

В настоящее время торговля осуществляется повсеместно. Ей подвержены любые предприятия: как малые, в которых может состоять лишь 1 человек, так и гиганты, насчитывающие в своем штате миллионы сотрудников. Их объединяет то, что любая торговая деятельность облагается налогами со стороны государств. Налоговые организации для точного учета требуют от предприятий отчетность по своей финансовой деятельности. Бумажная отчетность может быть актуальна для малых предприятий, однако крупным компаниям неудобно отправлять тонны бумаги в налоговые инстанции, ровно, как и этим самым инстанциям неудобно обрабатывать информацию, хранящуюся во множественных кипах бумаг; куда удобнее предоставлять отчетность в режиме он-лайн. Помимо этого, торговым предприятиям удобно автоматизировать их торговую деятельность не только из-за налоговой составляющей, но и со стороны уменьшения трудоемкости самих процессов торговли и управления запасами товаров и услуг. Эти задачи и призваны решить он-лайн кассы.

Предметная область — он-лайн касса.

Главными задачами он-лайн кассы являются:

* ведение сессионной статистики;
* ведение статистики продаж;
* ведение статистики возвратов;
* финансовая и налоговая отчетность.

Работа с данным программным модулем будет производиться операторами торговых предприятий, и благодаря взаимодействию модуля с базой данных регистрация изменений количества товаров и ведение сессионной статистики будут осуществляться автоматически.

К достоинствам данного проекта можно отнести:

* уменьшение трудоёмкости финансовых процессов;
* автоматизация составления налоговой и финансовой отчетностей;
* автоматизация управления складом.

Базовые сущности этой предметной области:

* смены. Атрибуты смен — id смены, id пользователя, дата начала и завершения смены, продажи, возвраты, изъятия, внесения и текущий баланс;
* продажи. Атрибуты продаж — id товара, количество, id смены;
* товары. Атрибуты товара — id, наименование товара, количество, цена, скидка;
* возвраты; Атрибуты возврата — id товара, количество, id смены;
* пользователи; Атрибуты пользователя — id, пароль, фамилия, имя, отчество;
* товары, нерегистрируемые в базе данных — id смены, сумма.

**1.2 Выбор методов решения задач**

Когда появляется необходимость разработки программного модуля, остро встает вопрос выбора целевой платформы разработки, среды разработки и языка программирования, с помощью которого разрабатываемый модуль будет реализован.

В качестве целевой платформы разработки была выбрана десктоп-конфигурация с операционной системой (далее ОС) от компании Microsoft Windows, так как данная конфигурация является модульной и её при необходимости можно усовершенствовать, а не покупать новую; а ОС Windows является достаточно распространённой, имеет большой набор дополнительного инструментария и не требует специальных навыков от оператора контрольно-кассовой техники.

В качестве языка программирования (далее ЯП) для реализации поставленной задачи выбран объектно-ориентированный язык программирования от компании Microsoft C# и технология Windows Presentation Foundation(WPF) на платформе .NET Framework.

C# — простой, современный объектно-ориентированный и типобезопасный язык программирования. C# относится к широко известному семейству языков C.

C# является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Важная особенность таких компонентов — это модель программирования на основе свойств, методов и событий. Каждый компонент имеет атрибуты, предоставляющие декларативные сведения о компоненте, а также встроенные элементы документации. C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# отлично подходит для создания и применения программных компонентов.

Конкуренцию данному языку в десктоп-разработке могут составить такие ЯП, как C++, Java, JavaScript.

Для разработки десктоп-приложения на ЯП C++ необходим либо знания фреймворка Qt, либо знания WIN API. Разработка с использованием WIN API является долгим и сложным процессом, в отличие от разработки   
с применением Qt, так как он по сути является конструктором и является аналогом Windows Forms для C#. В отличие от приложения, разработанного с помощью C# Windows Forms, приложение C++ Qt является кроссплатформенным решением и не привязано к платформе .NET Framework, а, следовательно, и к операционной системе Windows. Однако сам ЯП C++ является более сложным и более низкоуровневым в сравнении с C#. Примером этого является механизм указателей и ссылок для работы с памятью, а также необходимость «вручную» выделять и освобождать память от объектов, что в неумелых руках может привести к утечке памяти и снижению производительности.

Следующий конкурент, Java, является кроссплатформенным высокоуровневым ЯП от компании Oracle. В отличие от C++ работа с памятью в данном ЯП происходит автоматически — выделение памяти под объекты и её освобождение автоматизированы — что частично решает проблему перерасхода памяти при разработке приложения новичком. Кроссплатформенность достигается при помощи использования виртуальной машины Java, преобразующей код Java в специальный байт-код на этапе компиляции. Для разработки графического пользовательского интерфейса с использованием Java существуют несколько фреймворков: Swing, AWT, JavaFX и так далее. Однако ни один из приведенных фреймворков не даёт нужного функционала для раздельной разработки пользовательского интерфейса и логики приложения, так как обе части пишутся на самом языке Java, что увеличивает время и сложность разработки.

JavaScript является интерпретируемым ЯП, первоначально созданным для web-разработки. Со временем язык сильно развился и в настоящее время используется в большинстве сфер программирования, начиная web-программирование и заканчивая мобильной разработкой. Одной из проблем в разработке может послужить динамическая типизация языка, что при недостатке знаний может приводить к фатальным ошибкам. Для разработки десктоп приложения на JavaScript требуется система Electron JS, позволяющая использовать в качестве пользовательского интерфейса разметку, созданную с помощью HTML и CSS. Однако, для работы приложений, созданных   
с помощью Electron JS необходим движок Chromium, используемый в таких браузерах как Google Chrome и Opera, и, входящий в саму систему Electron JS, что увеличивает объем памяти, занимаемый приложением. Так приложение, разработанное с помощью C# WPF .NET Framework, имеющее объем от единиц до двух десятков мегабайт, при разработке с помощью Electron JS будет занимать порядка 100 мегабайт. Так же так как JavaScript является интерпретируемым ЯП, то и уровень производительности данного приложения будет сравнительно ниже.

При заработке десктоп приложения на C# есть несколько технологий, позволяющих разрабатывать пользовательский интерфейс приложения: Windows Forms (WF), Windows Presentation Foundation (WPF), Universal Windows Platform (UWP).

WF является самой простой и самой старой из выше перечисленных технологий. Разработка с помощью данной технологии ведется быстро, так как WF является конструктором окон. С помощью WF очень трудно разрабатывать сложный интерфейс, а при масштабировании окон приложение будет выглядеть не самым лучшим образом. На сегодняшней день данная технология является устаревшей.

WPF является технологией разработки более гибкого пользовательского интерфейса на платформе .NET. Данная технология использует декларативный язык разметки XAML, созданный на основе XML. Это позволяет отделить процесс разработки интерфейса от разработки логики приложения, однако из-за этого скорость разработки, в сравнении с WF, значительно ниже. WPF предназначена для разработки сугубо десктоп приложений, однако код таких приложений в дальнейшем можно перенести в проекты UWP и Xamarin, так как они так же используют XAML и C#. Данная технология не поддерживается операционными системами, более ранними, чем Windows 7.

UWP является самой молодой из выше перечисленных технологий. Она так же, как и WPF использует XAML для создания пользовательского интерфейса. Отличием данной технологии от своего предшественника является то, что приложения пусть и разрабатываются лишь для Windows, однако теперь могут быть запущены и на мобильных устройствах с Windows Mobile в качестве ОС. Однако данная технология поддерживается лишь Windows 10 и   
Windows Mobile.

Для работы с базой данных используется технология Entity Framework 6,   
в силу отсутствия необходимости для программиста знания языка структурированных запросов SQL. Её аналогом является традиционная технология для работы с базами данных на платформе .NET Framework — ADO.NET.

ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология предоставляет набор классов, через которые осуществляется механизм отправки запросов к базам данных, установления подключения, получения ответов от базы данных и осуществления ряда других операций.

Функционал ADO.NET построен таким образом, чтобы предоставить разработчикам унифицированный интерфейс для работы с самыми различными СУБД.

Основу интерфейса взаимодействия с базами данных в ADO.NET представляет ограниченный круг объектов:

* Connection — установление подключения к источнику данных;
* Command — выполнение операций с данными в базе данных;
* DataReader — считывание полученных в результате запроса данных;
* DataSet — предназначен для хранения данных из базы данных и позволяет работать с ними независимо от самой базы данных;
* DataAdapter — посредник между DataSet и источником данных.

Главным образом, через эти объекты и будет идти работа с базой данных.

Entity Framework представляет специальную объектно-ориентированную технологию на базе.NET Framework для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне осуществляются операции с таблицами, индексами, первичными и внешними ключами, то на концептуальном уровне, предоставляемом Entity Framework, осуществляется работа с объектами.

Центральной концепцией Entity Framework является понятие   
сущности (entity). Сущность представляет собой набор данных, ассоциированных с определенным объектом (класс). Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Отличительной чертой Entity Framework является использование запросов, составленных с помощью языка интегрированных запросов LINQ для выборки данных из базы данных. С помощью LINQ возможно не только извлекать определенные строки, хранящие объекты из базы данных, но и получать объекты, связанные различными ассоциативными связями.

**2 Проектная часть**

**2.1 Общая структура приложения**

Процесс разработки любого приложения начинается с планирования. На данном этапе необходимо четко определиться со структурой проекта, проанализировать информацию относительно предметной области. Структуру приложения можно условно разделить на внешнюю и внутреннюю.

Внутренняя структура ⎯ это граф разделов и подразделов приложения. Этот вид структуры определяется информацией, доступной пользователю.

Внешняя структура представляется как расположение ключевых элементов пользовательского интерфейса.

Структура приложения должна определять интуитивно понятный пользовательский интерфейс.

На рисунке 1 представлена общая(постраничная) структура организации сайта.

Он-лайн касса

Продажа

Возврат

Смена

Рисунок 1 ⎯ Общая структура организации программного модуля

На рисунке 2 представлена структура страницы «Смена».

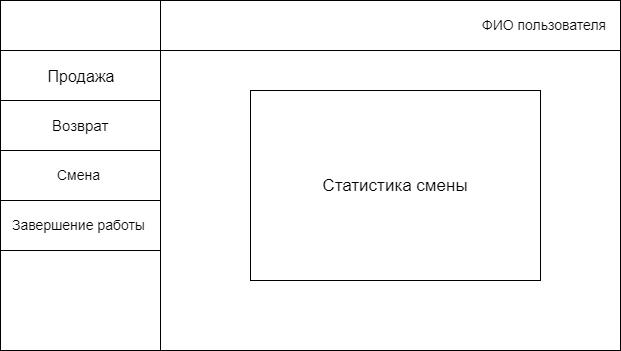


Рисунок 2 ⎯ Страница «Смена»

Структуры страниц «Продажа» и «Возврат» идентичны. На рисунке 3 представлена структура страниц «Продажа» и «Возврат».



Рисунок 3 ⎯ Страницы «Продажа» и «Возврат»

Помимо структуры пользовательского интерфейса на этапе проектирования так же важно определить структуру   
бизнес-процессов (далее БП) и информационных потоков (далее ИП).

На рисунке 4 представлена общая структура БП и ИП разрабатываемого программного модуля.

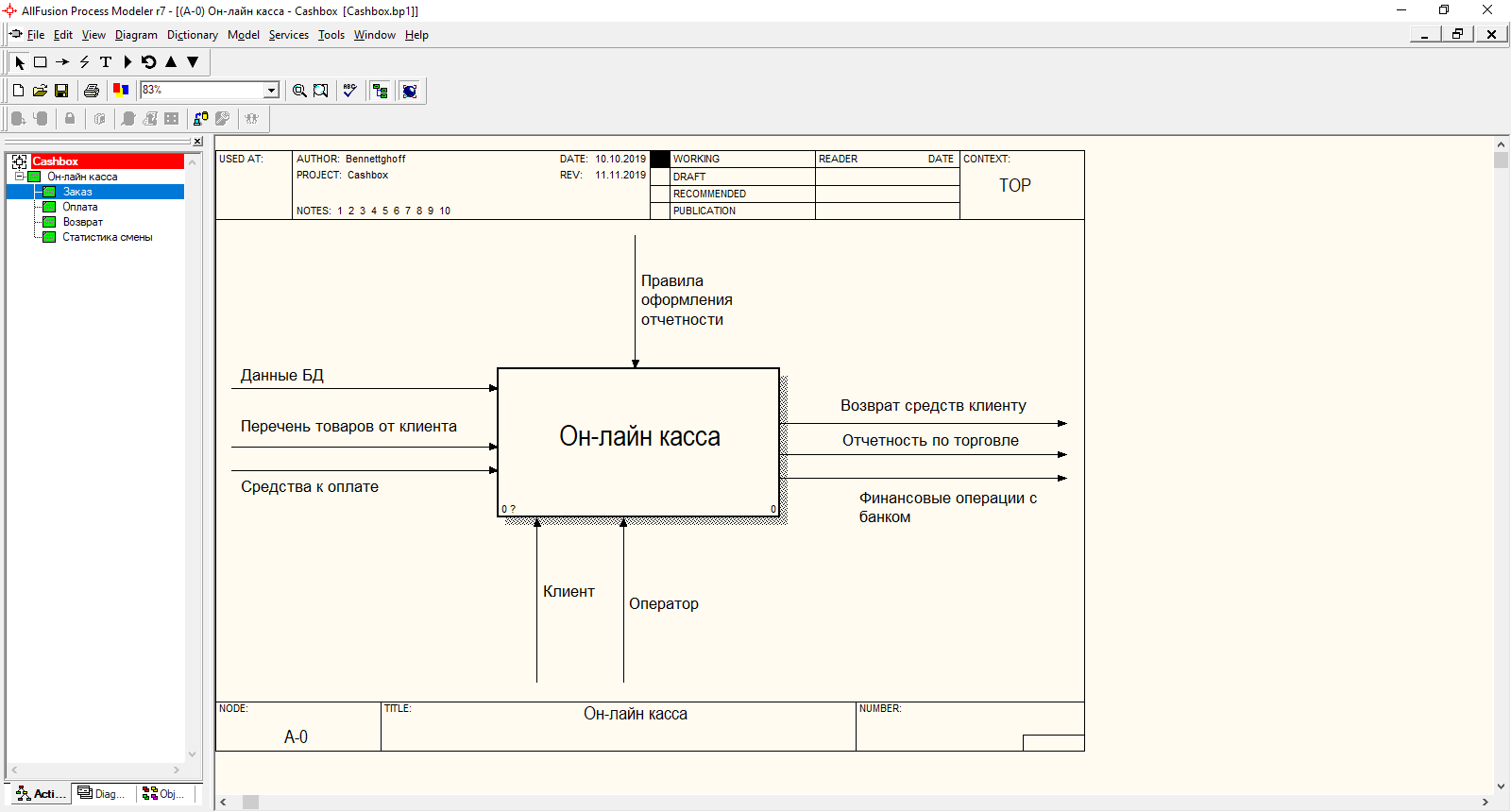


Рисунок 4 — Общая структура БП и ИП

На рисунке 5 представлена внутренняя структура БП и ИП разрабатываемого программного модуля.

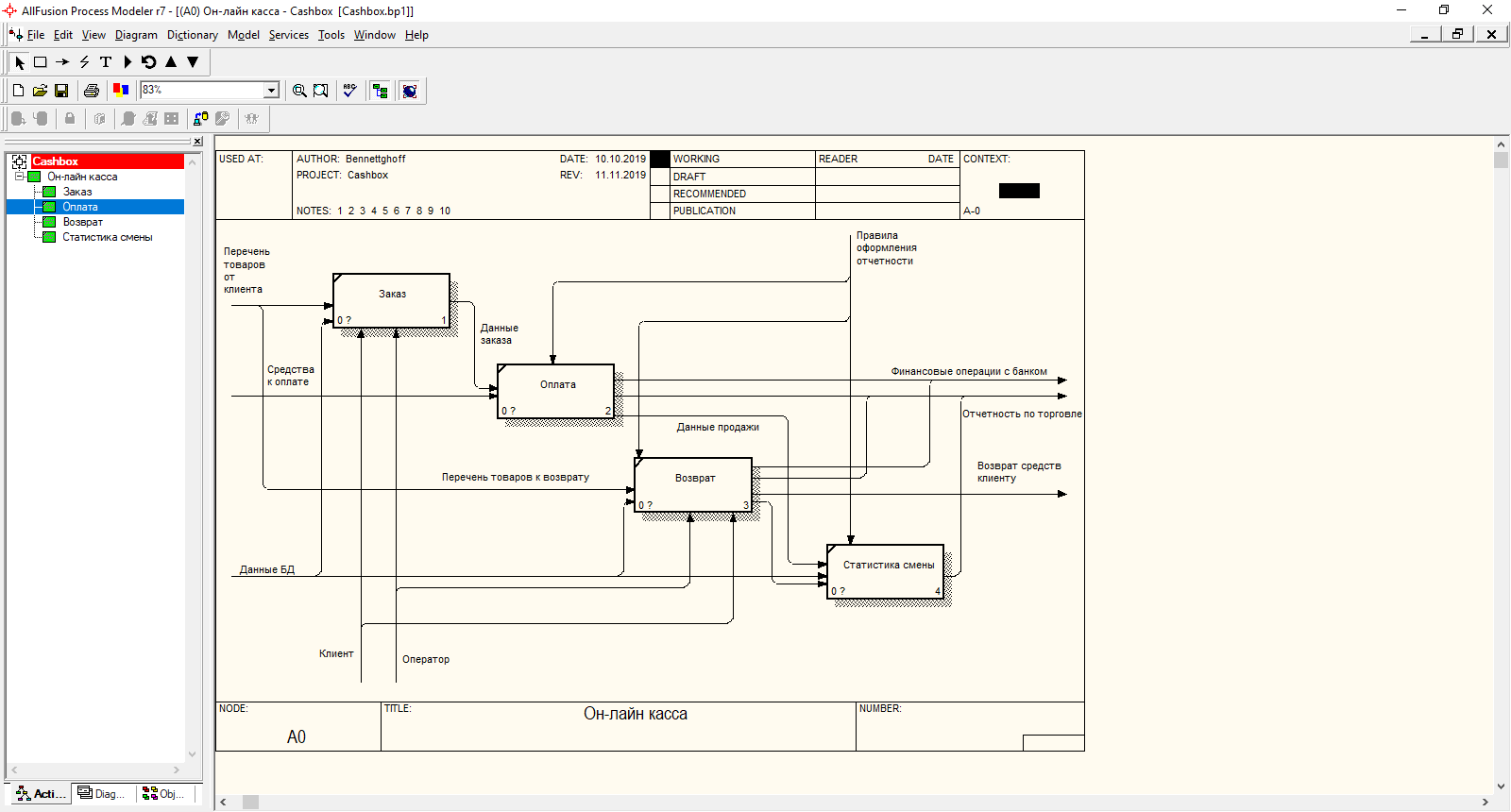


Рисунок 5 — Внутренняя структура БП и ИП

Скриншоты реализованного проекта представлены в приложении А.

**2.2 Функция перехода по страницам**

Функция перехода по страницам представляет собой обработчик   
события Click кнопок навигации на главной форме. При нажатии на одну из кнопок генерируется событие Click, которое вызывает свой обработчик PageSwitch(), и в зависимости от имени кнопки, сгенерировавшей событие URI фрейма, отвечающего за отображение страниц меняется на необходимый, а индикатор страницы перемещается к соответствующей кнопке. Код функции PageSwitch() представлен в приложении Б.

**2.3 Функция начала смены**

При нажатии на кнопку «Начать смену» вызывается команда AuthWndOpenCommand, вызывающая диалоговое окно авторизации. При заполнении полей и нажатии на кнопку «Вход» диалогового окна вызывается команда AuthorizationCommand. Данная команда вызывает метод класса User SignIn(), возвращающий объект класса User при успешных идентификации и аутентификации, в противном случае клиент получает соответствующее сообщение. После получения объекта класса User вызывается метод ShiftStart() класса Shift, возвращающий экземпляр класса Shift. Код, реализующий логику авторизации представлен в приложении В.

**2.4 Функция внесения и изъятия денежных средств**

При нажатии на кнопку «Добавить» или «Изъять» вызывается команда AddMoney или WithdrawMoney соответственно, вызывающая соответствующее диалоговое окно. При заполнении поля необходимой суммы и нажатии на кнопку операции диалогового окна вызывается команда метод AddMoneyAsync() или WithdrawMoneyAsync() класа Shift соответственно. Код, реализующий логику внесения-изъятия средств представлен в приложении Г.

**2.5 Функция продажи-возврата товаров**

Перед тем как осуществить продажу или возврат необходимо добавить товары в чек. Это производится посредством поиска товаров в баз данных, либо добавления товара со свободной ценой с помощью специальной клавиатуры страницы «Продажа» или «Возврат». После добавления товаров в чек можно отредактировать их количество, либо удалить их из чека с помощью поля ввода количества и кнопки удаления товара в области чека. При нажатии кнопки «Оплатить» вызывается диалоговое окно ввода передаваемой клиентом сумы. После чего, если необходимо вызвать сдачу, то выводится соответствующее сообщение. При нажатии кнопки «Возврат» выводится сообщение, о требовании передать клиенту нужную сумму. Код, реализующий логику продажи-возврата товаров представлен в приложении Д.

**3 Тестирование программного продукта**

Процесс тестирования программных модулей состоит в проверке корректности обработки модулями поступающей информации и получающихся на выходе данных в соответствии с функциями, представленными в спецификациях требований. Должна быть проверена корректность структуры модулей и примененных конструктивных элементов: циклов, блоков, переключателей и так далее.

Проверке подлежат маршруты обработки информации в каждом модуле и правильность их реализации в зависимости от исходных данных. Полнота теста определяется критериями выделения маршрутов для тестирования и степенью покрытия тестами требований спецификаций и возможных маршрутов исполнения программы. На каждом выделенном маршруте должна проверяться корректность выполняемых вычислений при некоторых фиксированных исходных данных. При этом выявляются ошибки неполного состава или некорректности условий при реализации частных маршрутов обработки данных, а также некоторые ошибки преобразования переменных. Для каждого выделенного маршрута по тексту программы формируется набор условий, определяющих его реализацию и используемый при создании соответствующего теста. Такое представление маршрутов позволяет упорядоченно контролировать достигнутый уровень проверки маршрутов и в некоторой степени предохраняет от случайного пропуска отдельных не протестированных маршрутов.

Ручное тестирование — это процесс поиска дефектов в работе программы, когда тестер проверяет работоспособность всех компонентов программы, как если бы он был пользователем. Часто, для точности проверки, тестер использует заранее заготовленный план тестирования, в котором отмечены наиболее важные аспекты работы программы

Таблица1 — Тестовые данные

|  |  |  |
| --- | --- | --- |
| Наименование | Значение | Результат |
| Начать смену | клик | Окно авторизации |
| Логин | root | Успешная авторизация  Активация кнопок «Добавить», «Изъять», «Продажа», «Возврат», «Завершить смену»  Деактивация кнопки  «Начать смену». |
| Пароль | 1243 |
| Добавить | 1000 | Внесения=1000  Денег в кассе=1000 |
| Код товара | 666666 | Товар «Клей «Момент»» |
| Код товара | 123456 | Товар «Молоко «Норов»» |
| Количество «Клей «Момент»» | 5 | Количество=5 |
| Свободная цена | 44 | Товар с ценой 44 |
| К оплате | 150 | Выдайте сдачу: 0,10 руб.  Денег в кассе=1149,90  Продажи=149,90 |
| Код товара | 5 | Товар «Клей «Момент»» |
| Количество «Клей «Момент»» | 2 | Количество=2 |
| Оплатить | клик | Передайте клиенту 28,00 руб.  Денег в кассе=1121,90  Возвраты=28.00 |
| Изъять | 121.90 | Денег в кассе=1000.00  Изъятия=121.90 |
| Завершить смену | клик | Денег в кассе=0  Продажи=0  Возвраты=0  Внесения=0  Изъятия=0  Деактивация кнопок «Добавить», «Изъять»  «Продажа», «Возврат», «Завершить смену»  Активация кнопки «Начать смену» |

Результаты тестирования представлены в приложении E.

**ЗАКЛЮЧЕНИЕ**

С внедрением информационных технологий производственные процессы сильно упростились за счёт их автоматизации. Автоматизация наиболее востребована в финансовой сфере, ведь именно на обороте ресурсов и финансов всегда держалась и будет держаться экономика, являющаяся неотъемлемой сферой жизни человеческого общества. В условиях динамически расширяющегося производства обрабатывать огромное количество финансовой информации вручную неудобно, а также требует слишком большого количества времени и ресурсов.

На сегодняшний день эти действия автоматизированы, что позволяет сократить время на обслуживание клиентов и составление финансовой и налоговой отчетностей, не занимая много времени у персонала, как было раньше.

В ходе выполнения курсовой работы был получен программный модуль, полностью готовый к внедрению и расширению, который включает себя модули продажи и возврата товаров, ведения сессионной статистики и авторизации пользователей, а также разработан удобный и понятный для пользователя интерфейс.

Основными задачами для разработки программного модуля являлись:

* ведение сессионной статистики торгового предприятия;
* осуществление продажи товаров сотрудниками торгового предприятия;
* осуществления возврата товаров клиентами торгового предприятия;
* обновление статистики склада торгового предприятия.

Данный проект направлен на облегчение работы сотрудников в сфере торговли путем автоматизации ведения налоговой и финансовой отчетности и управления складом. Разработанный программный модуль содержит в себе модули продажи и возврата товаров, ведения сессионной статистики и авторизации пользователей.

К достоинствам разработанного программного модуля можно отнести интуитивно понятный пользовательский интерфейс, что значительно упрощает освоение программы пользователем.

Исходя из этого, можно сказать, что цель курсового проекта достигнута путем реализации поставленных задач.

Разработанный программный модуль полностью удовлетворяет всем требованиям, поставленным на этапе постановки задач курсового проекта.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. METANIT.COM [Электронный ресурс]. —   
   Режим доступа: https://metanit.com/sharp/wpf/. — Дата доступа: 20.10.2019.
2. Stack Overflow [Электронный ресурс]. —  
    Режим доступа: https://ru.stackoverflow.com. — Дата доступа: 27.10.2019.
3. METANIT.COM [Электронный ресурс]. —   
   Режим доступа: https://metanit.com/sharp/entityframework/. — Дата доступа: 22.10.2019.
4. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер, Е. Матвеев. — Санкт-Петербург : Издательство Питер СПб, 2019. — 896 с.
5. Тюкачев, Н. А. С#. Основы программирования / Н. А. Тюкачев, В. Г. Хлебостроев, Джей Глинн. — Санкт-Петербург : Издательство Лань, 2018. — 272 с.
6. Вагнер, Б. Наиболее эффективное программирование на С# / Б. Вагнер. — Москва : Издательство Альфа-книга, 2018. — 240 с.
7. Троелсен, Э. Язык программирования С# 7 и платформы .NET и .NET Core / Э. Троелсен, Ф. Джепикс. — Киев : Издательство Диалектика, 2019. — 1328 с.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Скриншоты реализованного модуля**

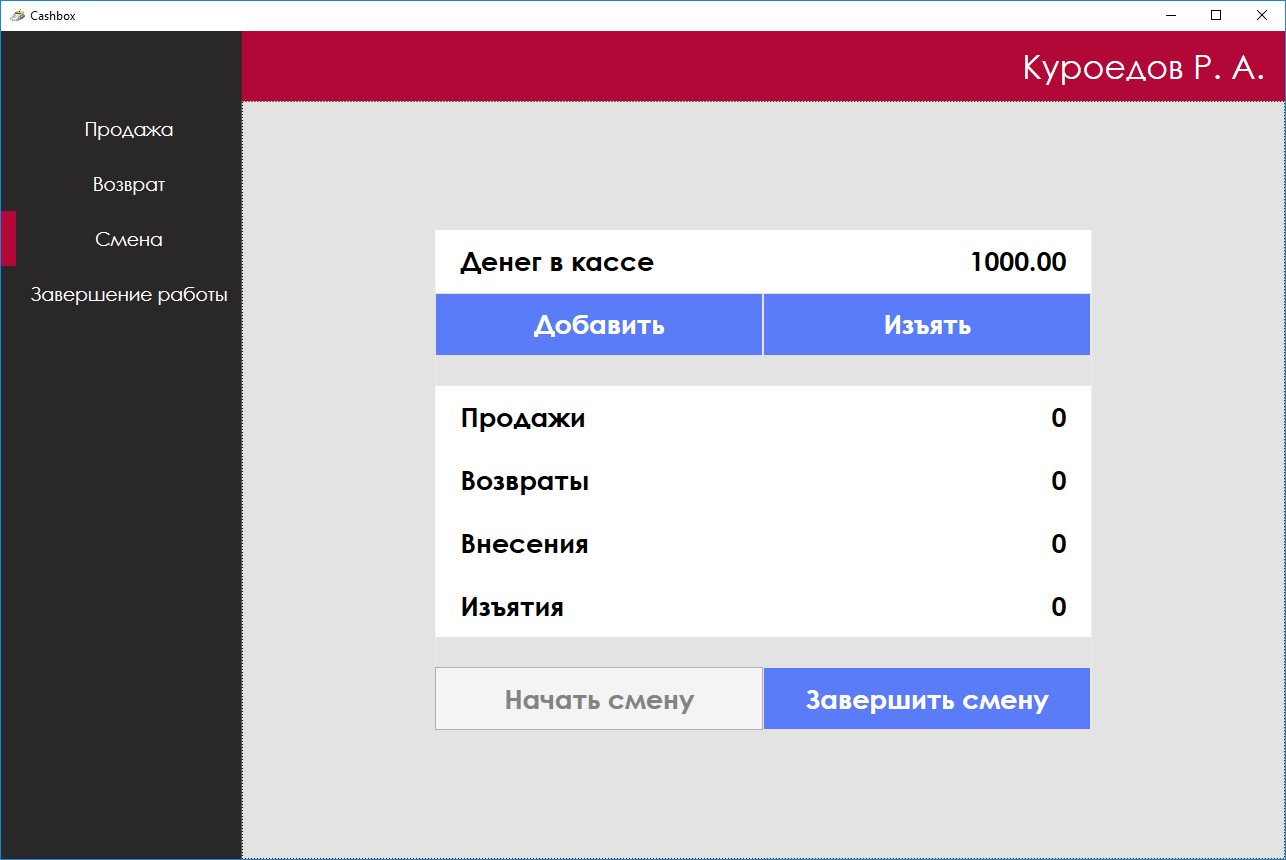


Рисунок А.1 ⎯ Скриншот страницы «Смена»

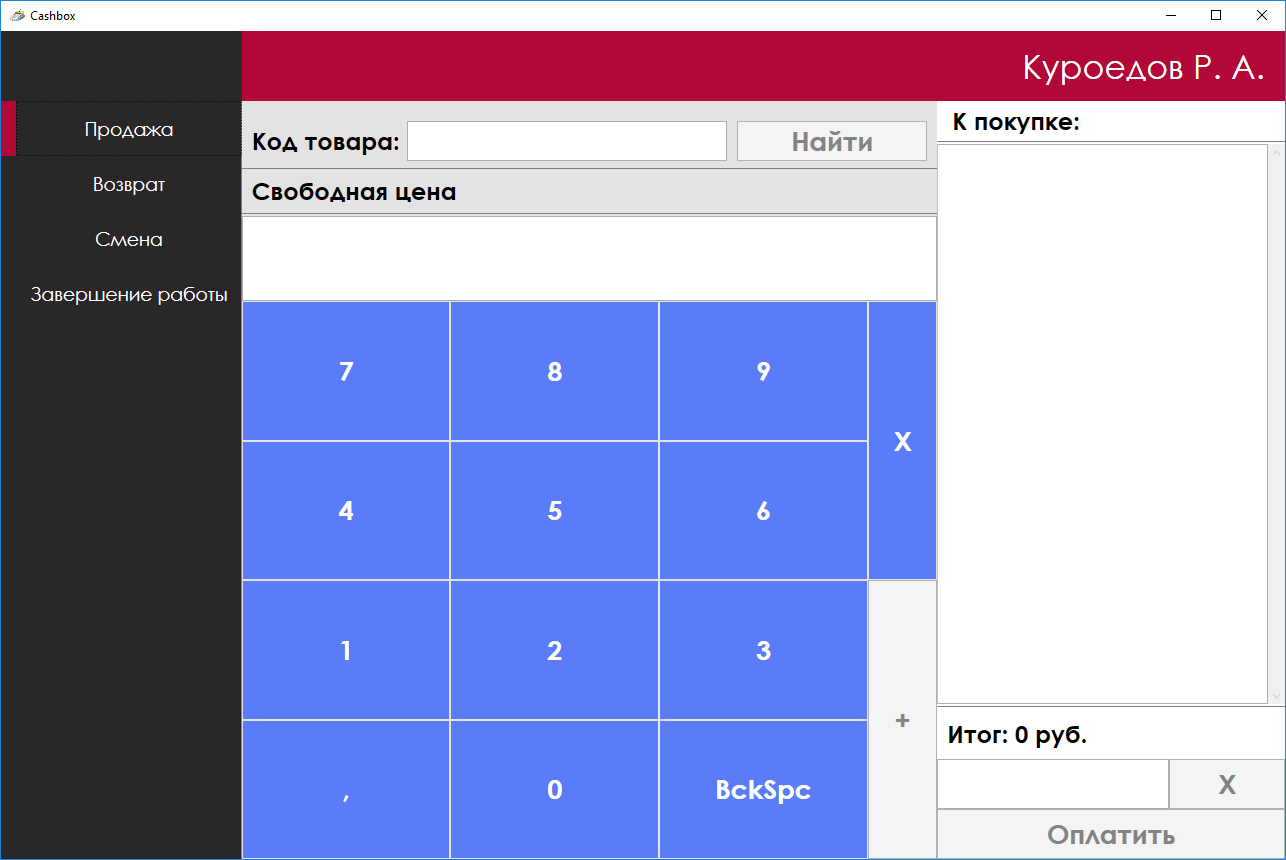


Рисунок А.2 ⎯ Скриншот страницы «Продажа»

Окончание ПРИЛОЖЕНИЯ А

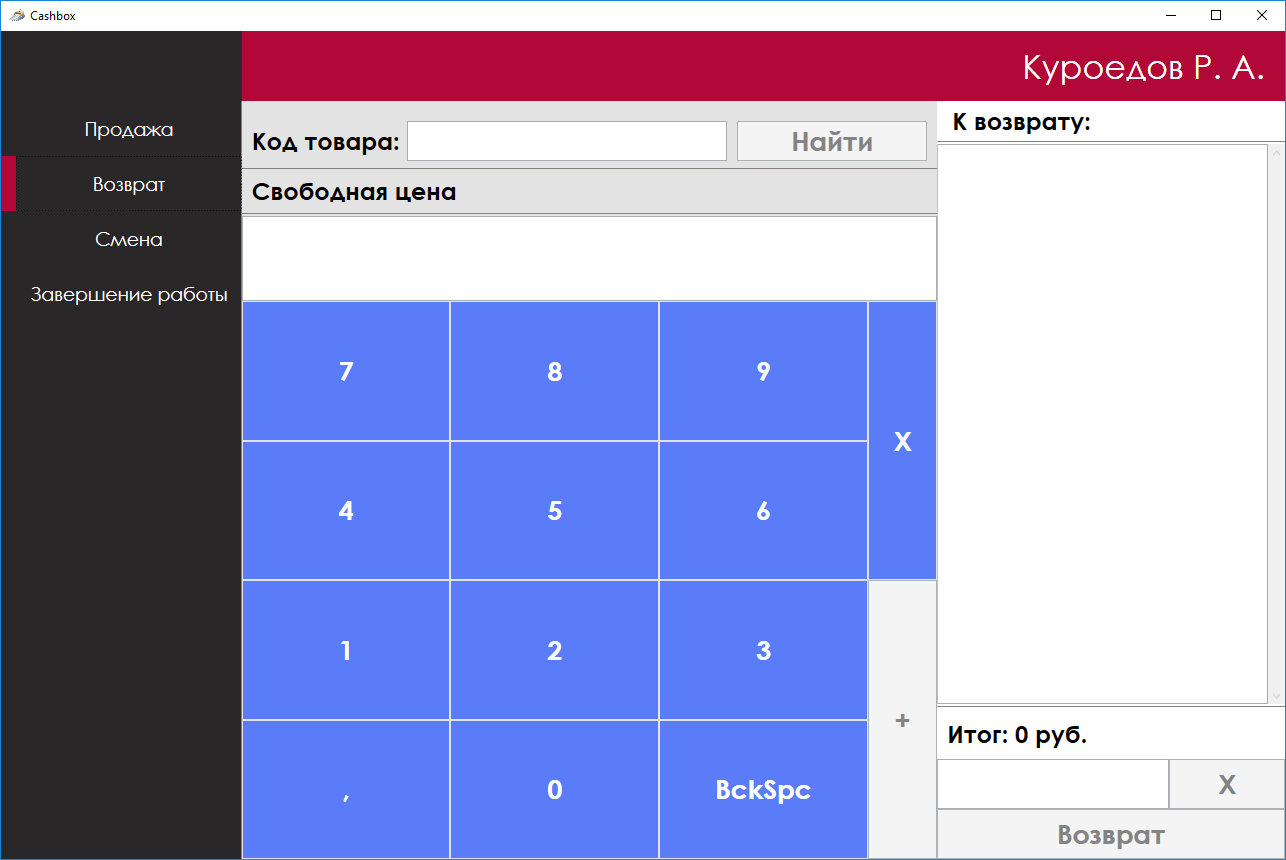


Рисунок А.3 ⎯ Скриншот страницы «Возврат»

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Функция перехода по страницам**

private void PageSwitch(object sender, EventArgs e)

{

Button btn = (Button)sender;

switch (btn.Name)

{

case "SellBtn":

MainAction.Source = new Uri("SellView.xaml", UriKind.Relative);

Grid.SetRow(PageIndicator, 0);

break;

case "ReturnBtn":

MainAction.Source = new Uri("ReturnView.xaml", UriKind.Relative);

Grid.SetRow(PageIndicator, 1);

break;

case "ShiftBtn":

MainAction.Source = new Uri("ShiftView.xaml", UriKind.Relative);

Grid.SetRow(PageIndicator, 2);

break;

case "ExitBtn":

Grid.SetRow(PageIndicator, 3);

Application.Current.Shutdown();

break;

}

**ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Код авторизации**

public RelayCommand AuthWndOpenCommand

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

\_wnd = null;

\_wnd = new AuthorizationWindow();

\_wnd.ShowDialog();

}));

}

}

public RelayCommand AuthorizationCommand

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

if (\_wnd.DialogResult == true)

{

CurrentUser = null;

CurrentUser = User.SignIn(EnteredLogin, EnteredPassword);

if (CurrentUser != null)

{

CurrentShift = null;

CurrentShift = Shift.ShiftStart(CurrentUser);

UserName = string.Format("{0} {1}. {2}.",

\_currentUser.GetInstance().SurName,

\_currentUser.GetInstance().Name[0],

\_currentUser.GetInstance().FatherName[0]);

ButtonIsEnabled = true;

using(CashboxDataContext db = new CashboxDataContext())

{

db.FreeItems.Add(new FreeItem(CurrentShift.SId));

db.SaveChanges();

}

}

}

}));

}

}

public static User SignIn(string uid, string pwd)

{

using (CashboxDataContext db = new CashboxDataContext())

{

User user = null;

try

{

db.DBConnectionCheck();

Окончание ПРИЛОЖЕНИЯ В

user = db.Users.FirstOrDefault(u => uid == u.UId && pwd == u.Password);

if (user != null)

{

if (SigningIn != null)

{

SigningIn(user, new SignInEventArgs(true));

return user;

}

}

else

{

if (SigningIn != null)

{

SigningIn(user, new SignInEventArgs(false));

return user;

}

}

}

catch (Exception e)

{

MessageBox.Show(e.Message);

}

return user;

}

}

public static Shift ShiftStart(IUser<User> user) //старт смены

{

using (CashboxDataContext db = new CashboxDataContext())

{

try

{

db.DBConnectionCheck();

Shift prevShift = db.Shifts.OrderByDescending(sh => sh.SId).FirstOrDefault();

decimal currCash = 0;

currCash = prevShift != null ? prevShift.CurrentCash : 0;

Shift currShift = new Shift(user, currCash);

db.Shifts.Add(currShift);

db.SaveChanges();

return currShift;

}

catch (Exception e)

{

MessageBox.Show(e.Message);

return null;

}

}

}

**ПРИЛОЖЕНИЕ Г**

**(обязательное)**

**Код внесения-изъятия средств**

public RelayCommand AddMoney

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

\_wnd = null;

\_wnd = new AddShiftWindow();

if(\_wnd.ShowDialog() == true)

{

Shift.AddMoneyAsync(MoneyToAddOrWithdraw, CurrentShift);

MoneyToAddOrWithdraw = (decimal)0.00;

}

}));

}

}

public RelayCommand WithdrawMoney

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

\_wnd = null;

\_wnd = new WithdrawShiftWindow();

if (\_wnd.ShowDialog() == true)

{

Shift.WithdrawMoneyAsync(MoneyToAddOrWithdraw, CurrentShift);

MoneyToAddOrWithdraw = (decimal)0.00;

}

}));

}

}

public static async void AddMoneyAsync(decimal money, Shift shift)

{

using (CashboxDataContext db = new CashboxDataContext())

{

try

{

db.DBConnectionCheck();

db.Shifts.Attach(shift);

shift.CashAdded += money;

shift.CurrentCash += money;

await db.SaveChangesAsync();

TransactionCompleted?.Invoke(shift, new ShiftTransactionEventArgs("Средства успешно добавлены!", true));

}

catch (Exception e)

{

TransactionCompleted?.Invoke(shift, new ShiftTransactionEventArgs(e.Message + "\nСредства добавлены не были!", false));

Окончание ПРИЛОЖЕНИЯ Г

}

}

}

public static async void WithdrawMoneyAsync(decimal money, Shift shift)

{

using (CashboxDataContext db = new CashboxDataContext())

{

try

{

db.DBConnectionCheck();

db.Shifts.Attach(shift);

if (shift.CurrentCash >= money)

{

shift.CurrentCash -= money;

shift.CashWithdrawn += money;

await db.SaveChangesAsync();

TransactionCompleted?.Invoke(shift, new ShiftTransactionEventArgs("Средства успешно изъяты!", true));

}

else

{

TransactionCompleted?.Invoke(shift, new ShiftTransactionEventArgs("Невозможно изъять введенную сумму, так как в кассе находится меньше средств, чем требуется", false));

}

}

catch (Exception e)

{

TransactionCompleted?.Invoke(shift, new ShiftTransactionEventArgs(e.Message + "\nСредства изъяты не были!", false));

}

}

}

**ПРИЛОЖЕНИЕ Д**

**(обязательное)**

**Код продажи-возврата товаров**

public RelayCommand AddCommand

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

Item item = Item.GetItem(this, IId);

if (item != null)

{

ItemParent prevItem = ItemsToSell.FirstOrDefault(i => i.IId == item.IId);

if(prevItem == null)

{

ItemsToSell.Add(item);

}

else

{

ItemsToSell.FirstOrDefault(i => i.IId == item.IId).Number++;

}

Conclusion = ItemsToSell.Conclude();

}

IId = "";

},

(obj) => IId.Length > 0));

}

}

public RelayCommand AddFreeCommand

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

ItemsToSell.Add(new FreeItem(App.ShiftVM.CurrentShift.SId, Convert.ToDecimal(FreePrice)));

Conclusion = ItemsToSell.Conclude();

FreePrice = "";

},

(obj) => FreePrice.Length > 0));

}

}

public RelayCommand RemoveCommand

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

ItemParent item = obj as ItemParent;

if (item != null)

Продолжение ПРИЛОЖЕНИЯ Д

{

ItemsToSell.Remove(item);

if (ItemsToSell.Count() > 0)

{

ItemsToSell.Conclude();

}

else

{

Conclusion = 0;

}

}

},

(obj) => ItemsToSell.Count > 0));

}

}

public RelayCommand SellCommand

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

bool isEnough = true;

foreach (var item in ItemsToSell)

{

if (item.IId != "Free")

{

isEnough = ((Item)item).NumberCheck();

if (!isEnough) { break; }

}

}

if (isEnough)

{

\_wnd = new PayBackWindow();

\_itemsPriceSum = ItemsToSell.Sum(i => i.Price);

\_wnd.ShowDialog();

}

},

(obj) => ItemsToSell.Count > 0));

}

}

public RelayCommand PayCashCommand

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

var currShift = App.ShiftVM.CurrentShift;

if (MoneyToPay > currShift.CurrentCash)

{

MessageBox.Show($"В кассе недостаточно денег для сдачи: {MoneyToPay - currShift.CurrentCash} руб.");

}

else

{

Продолжение ПРИЛОЖЕНИЯ Д

foreach(var item in ItemsToSell)

{

item.SellItemAsync(currShift);

}

\_wnd.Close();

\_wnd = null;

ItemsToSell.Clear();

if ((Conclusion - MoneyToPay) != 0)

{

MessageBox.Show(string.Format("Выдайте сдачу: {0} руб.", MoneyToPay - Conclusion));

}

Conclusion = 0;

}

},

(obj) => MoneyToPay >= \_itemsPriceSum));

}

}

public RelayCommand KeyboardCommand

{

get

{

\_relayCommand = null;

return \_relayCommand ??

(\_relayCommand = new RelayCommand(obj =>

{

Button btn = obj as Button;

switch (btn.Content)

{

case "X":

FreePrice = "";

break;

case "BckSpc":

if(FreePrice.Length>0)

{

FreePrice = FreePrice.Remove(FreePrice.Length - 1);

}

break;

case ",":

if (FreePrice.Contains(',')) { break; }

FreePrice += btn.Content;

break;

default:

if (FreePrice.IndexOf(',') == 0)

{

break;

}

if (FreePrice.IndexOf(',') != -1 && FreePrice.Substring(FreePrice.IndexOf(',')).Length == 3)

{

break;

}

FreePrice += btn.Content;

break;

}

}));

}

}

Продолжение ПРИЛОЖЕНИЯ Д

public static Item GetItem(object sender, string iId)

{

using (CashboxDataContext db = new CashboxDataContext())

{

Item item = null;

try

{

db.DBConnectionCheck();

item = db.Items.FirstOrDefault(i => i.IId == iId);

if (item != null && item.Number >= 1)

{

item.Number = 1;

ItemSearching?.Invoke(sender, new ItemSearchEventArgs($"Товар с идентификатором {iId} добавлен в чек!", true));

return item;

}

else

{

ItemSearching?.Invoke(sender, new ItemSearchEventArgs($"Товар с идентификатором {iId} отсутствует на складе, либо не зарегистрирован в базе данных !", false));

return null;

}

}

catch (Exception e)

{

ItemSearching?.Invoke(sender, new ItemSearchEventArgs($"Не удалось найти товар в базе данных!\n{e.Message}", false));

return null;

}

}

}

public bool NumberCheck()

{

using(CashboxDataContext db = new CashboxDataContext())

{

long \_checkedNum;

try

{

db.DBConnectionCheck();

\_checkedNum = db.Items.First(i => i.IId == this.IId).Number;

if (\_checkedNum < this.Number)

{

MessageBox.Show($"На складе не хватает {this.Number - \_checkedNum} единиц товара \"{this.Name}\"!");

return false;

}

else

{

return true;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return false;

}

Продолжение ПРИЛОЖЕНИЯ Д

}

}

public override async void SellItemAsync(Shift currShift)

{

using(CashboxDataContext db = new CashboxDataContext())

{

try

{

db.DBConnectionCheck();

Sale sale = db.Sales.FirstOrDefault(s => s.SId == currShift.SId && s.IId == this.IId);

Item item = db.Items.First(i => i.IId == this.IId);

if (sale == null)

{

db.Sales.Add(new Sale(currShift, this));

}

else

{

sale.Number += this.Number;

}

item.Number -= this.Number;

db.Shifts.Attach(currShift);

currShift.CashReceived += this.Price\*this.Number;

currShift.CurrentCash += this.Price\*this.Number;

await db.SaveChangesAsync();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

public override async void ReturnItemAsync(Shift currShift)

{

using (CashboxDataContext db = new CashboxDataContext())

{

try

{

db.DBConnectionCheck();

Return \_return = db.Returns.FirstOrDefault(r => r.SId == currShift.SId && r.IId == this.IId);

Item item = db.Items.First(i => i.IId == this.IId);

if (\_return == null)

{

db.Returns.Add(new Return(currShift, this));

}

else

{

\_return.Number += this.Number;

}

item.Number += this.Number;

db.Shifts.Attach(currShift);

currShift.CashReturned += this.Price \* this.Number;

currShift.CurrentCash -= this.Price \* this.Number;

await db.SaveChangesAsync();

}

catch (Exception ex)

{

Окончание ПРИЛОЖЕНИЯ Д

MessageBox.Show(ex.Message);

}

}

}

public override async void SellItemAsync(Shift currShift) //асинхронная продажа товара

{

using (CashboxDataContext db = new CashboxDataContext())

{

try

{

db.DBConnectionCheck();

db.Shifts.Attach(currShift);

FreeItem fItem = db.FreeItems.First(f => f.SId == currShift.SId);

fItem.CashSum += this.Price \* this.Number;

currShift.CashReceived += this.Price \* this.Number;

currShift.CurrentCash += this.Price \* this.Number;

await db.SaveChangesAsync();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

public override async void ReturnItemAsync(Shift currShift) //асинхронный возврат товара

{

using (CashboxDataContext db = new CashboxDataContext())

{

try

{

db.DBConnectionCheck();

db.Shifts.Attach(currShift);

FreeItem fItem = db.FreeItems.First(f => f.SId == currShift.SId);

fItem.CashSum -= this.Price \* this.Number;

currShift.CashReturned += this.Price \* this.Number;

currShift.CurrentCash -= this.Price \* this.Number;

await db.SaveChangesAsync();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}}

**ПРИЛОЖЕНИЕ Е**

**(обязательное)**

**Результаты тестирования**

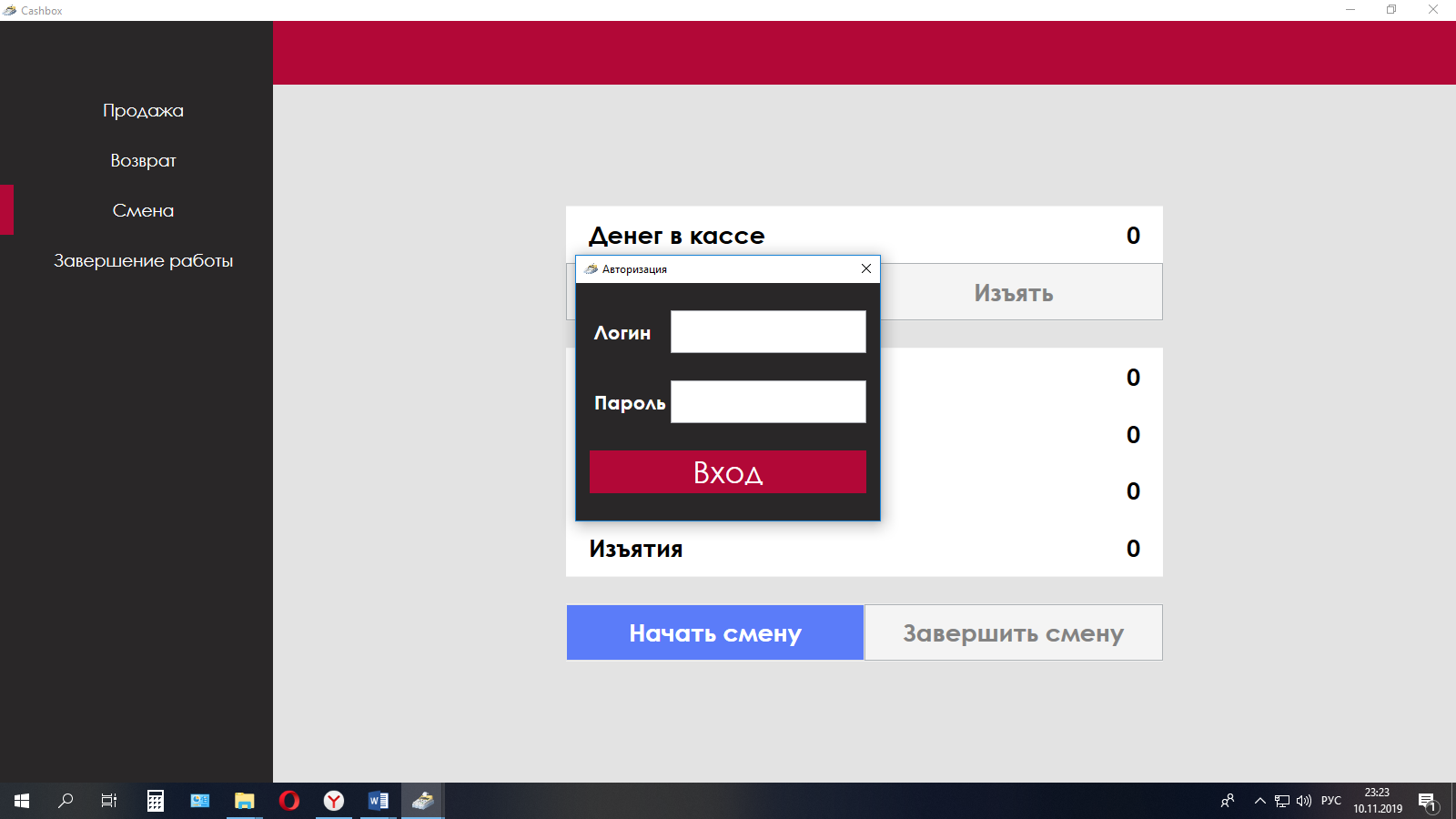


Рисунок Е.1 — Окно авторизации

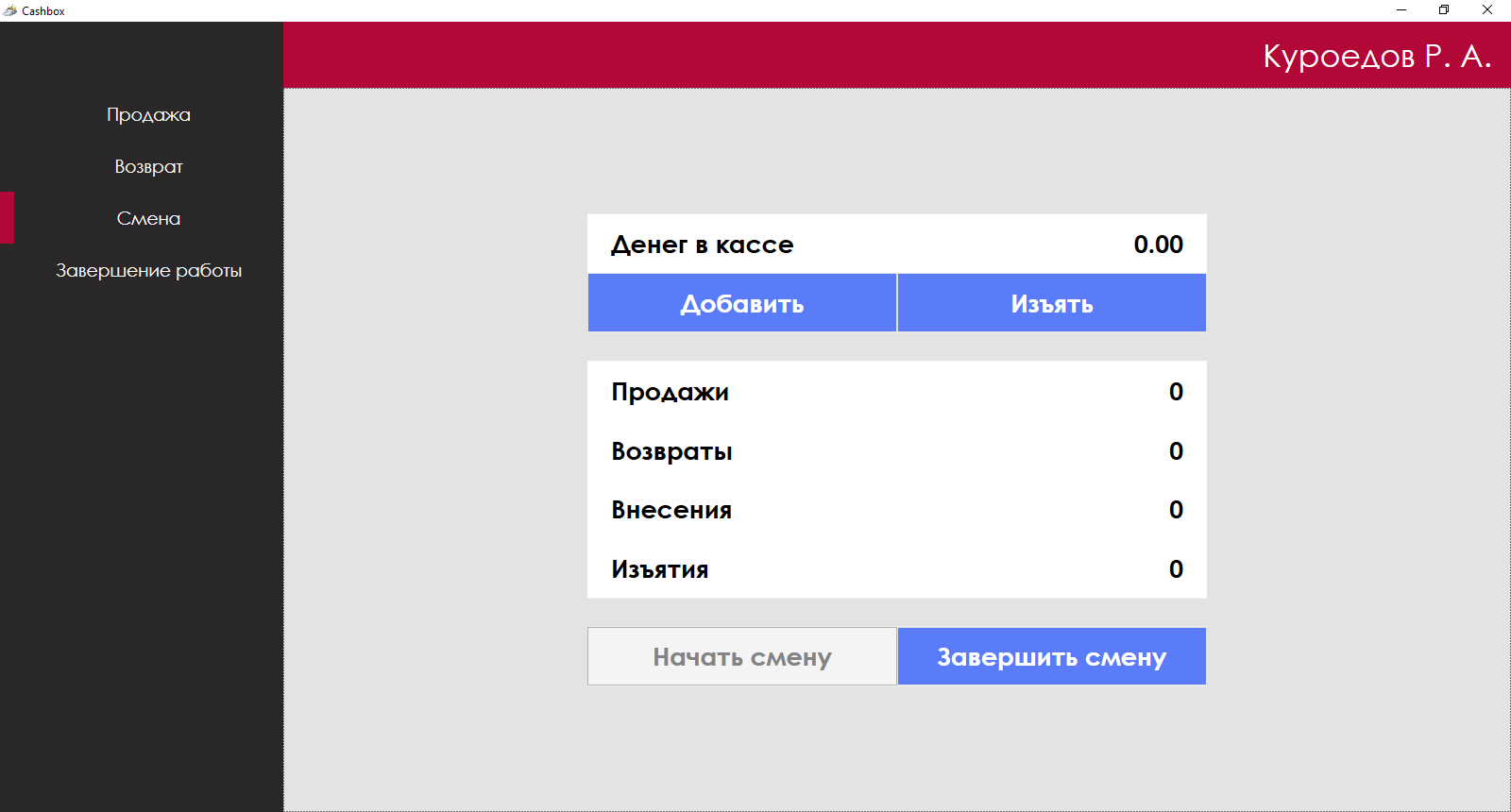


Рисунок Е.2 — Успешная авторизация

Продолжение ПРИЛОЖЕНИЯ Е

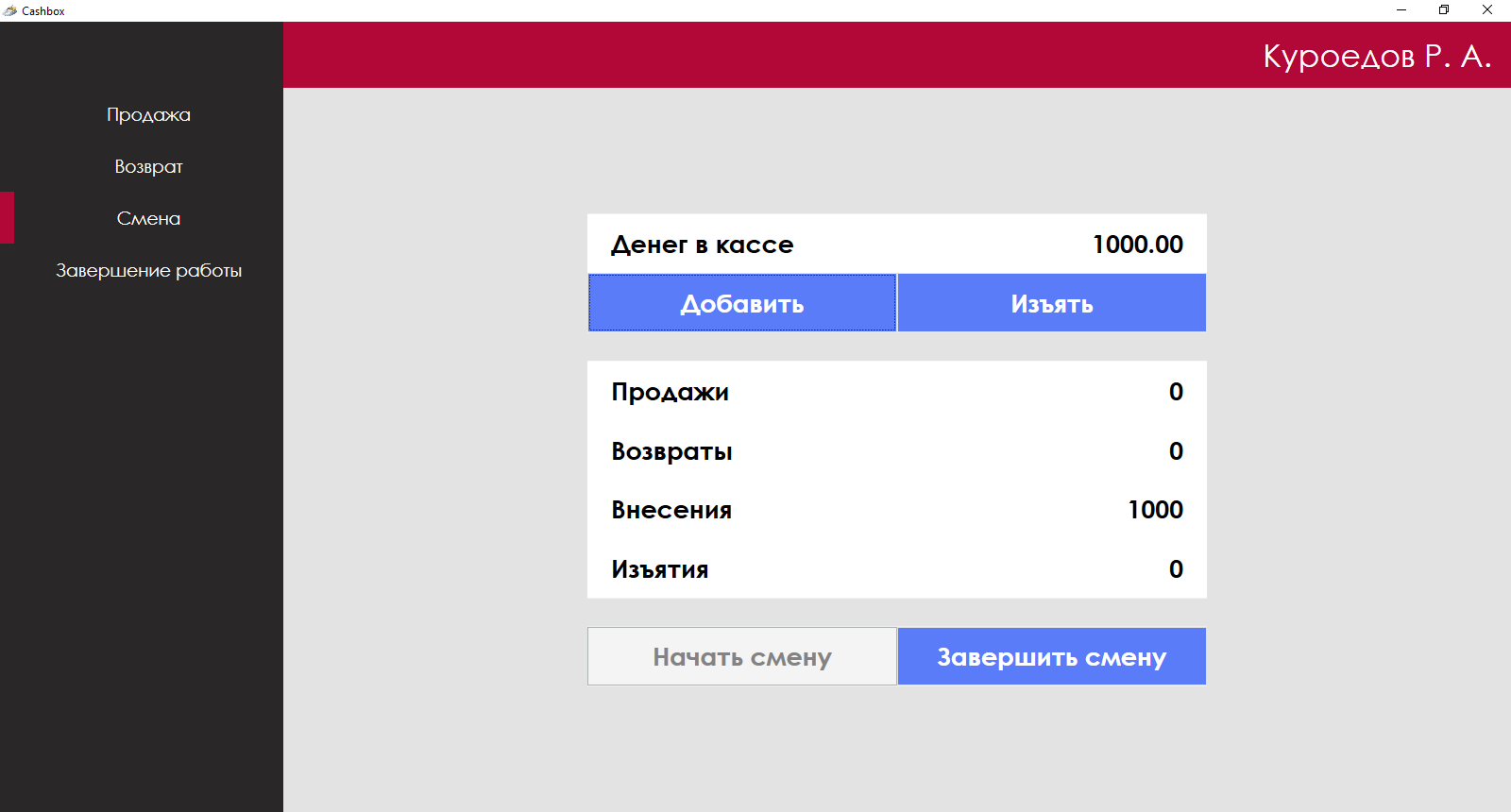


Рисунок Е.3 — Внесение средств

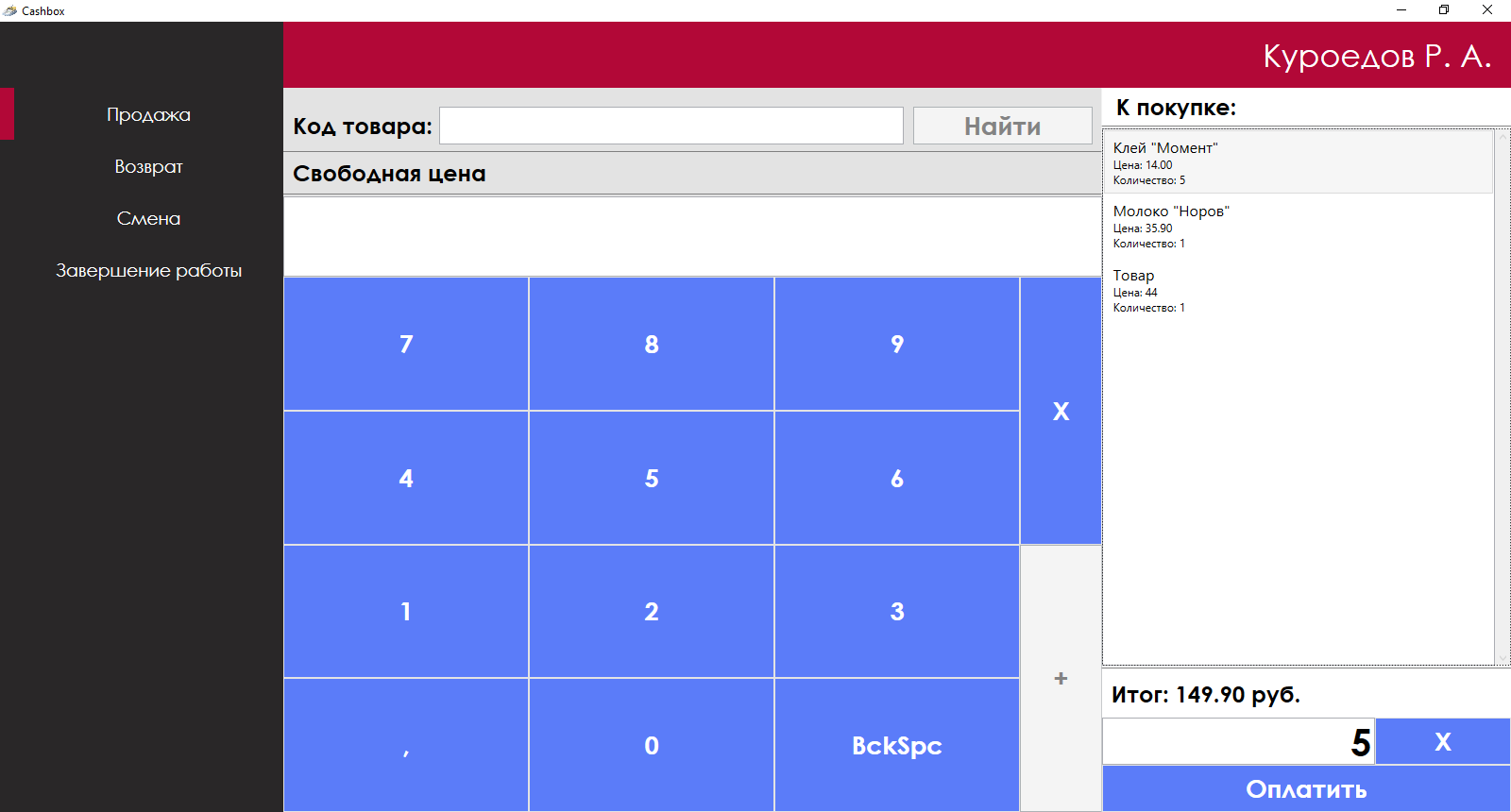


Рисунок Е.4 — Внесение в чек

Продолжение ПРИЛОЖЕНИЯ Е

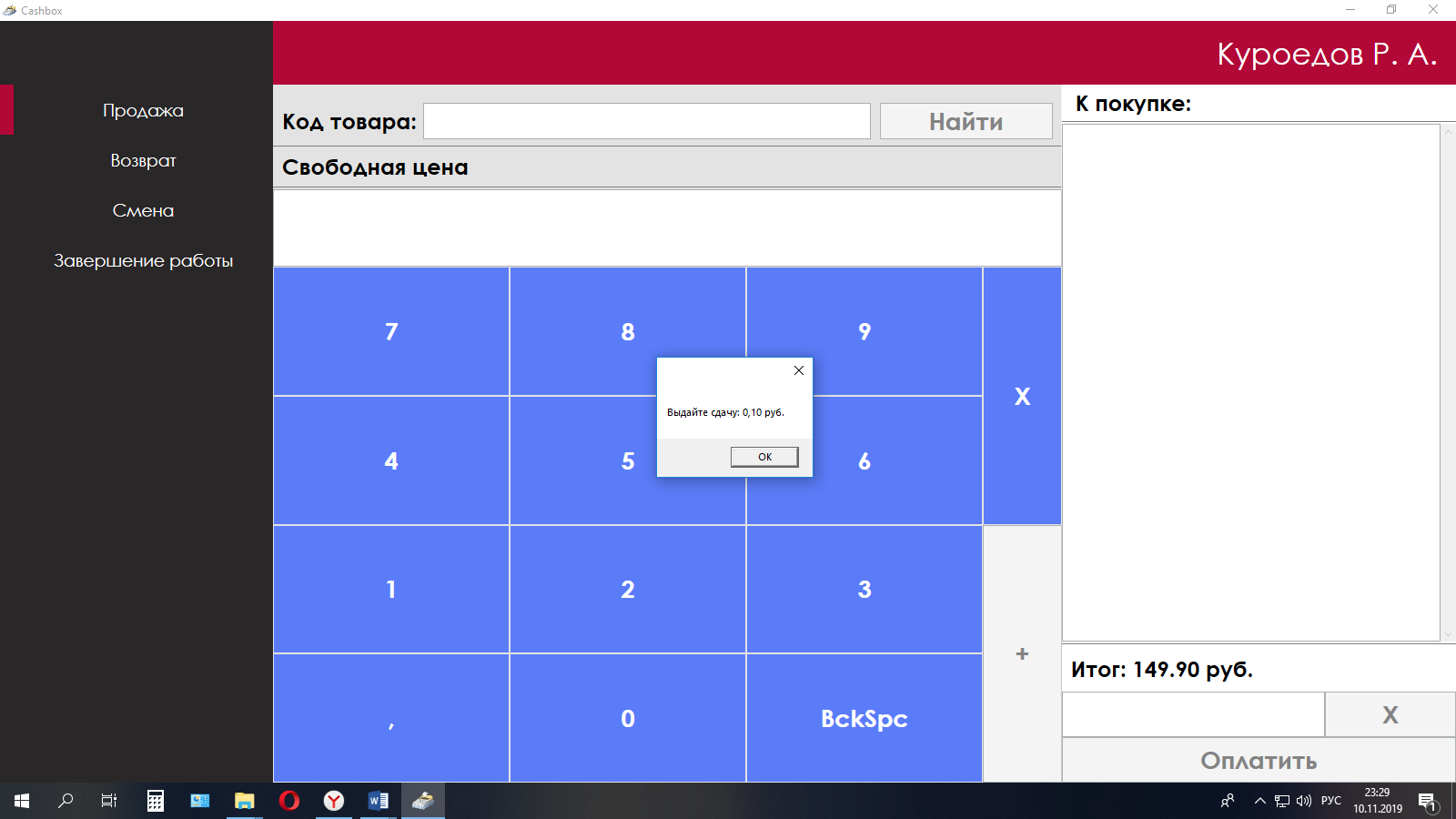


Рисунок Е.5 — Сдача

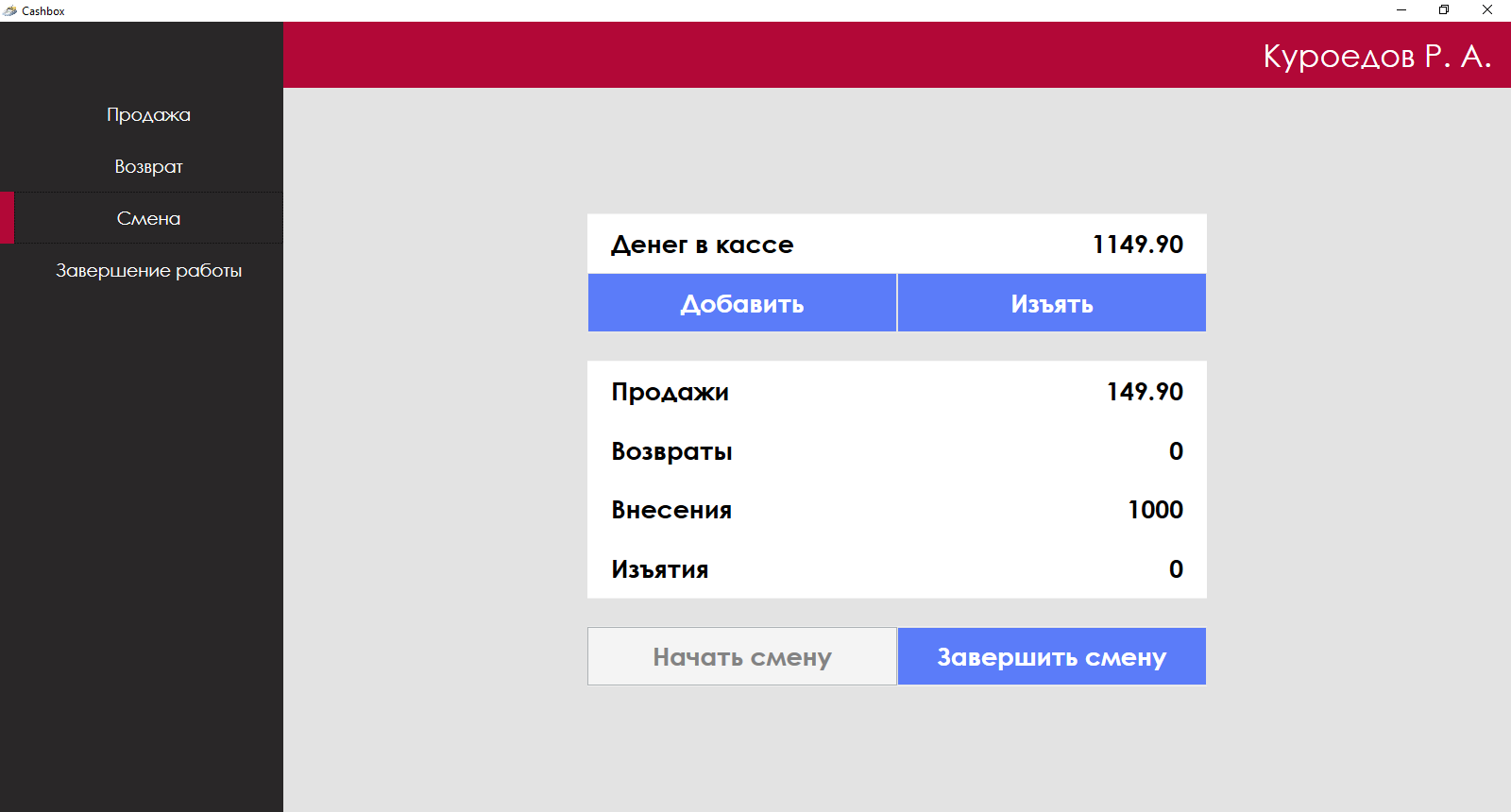


Рисунок Е.6 — Статистика после продажи

Продолжение ПРИЛОЖЕНИЯ Е

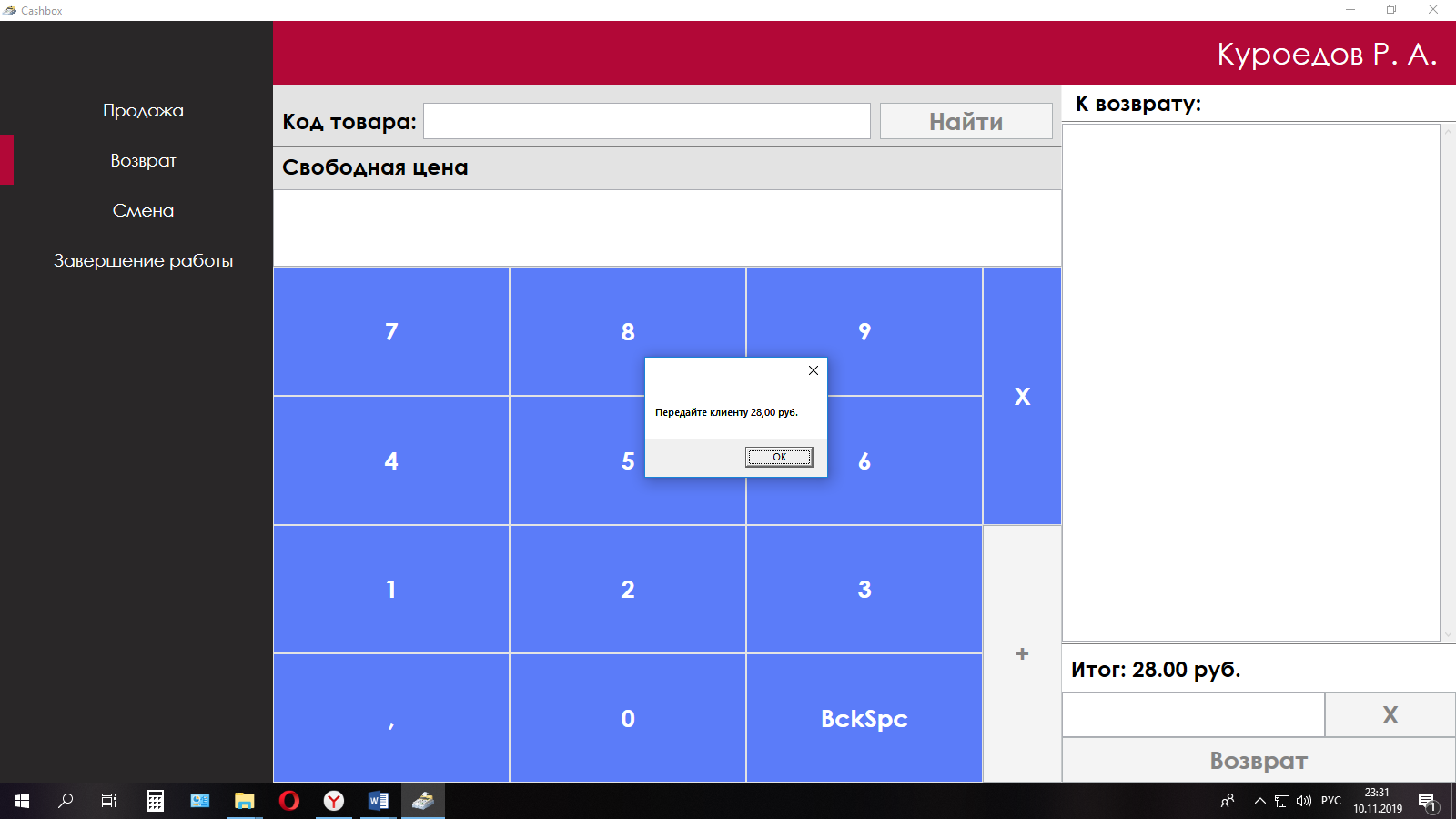


Рисунок Е.7 — Возврат

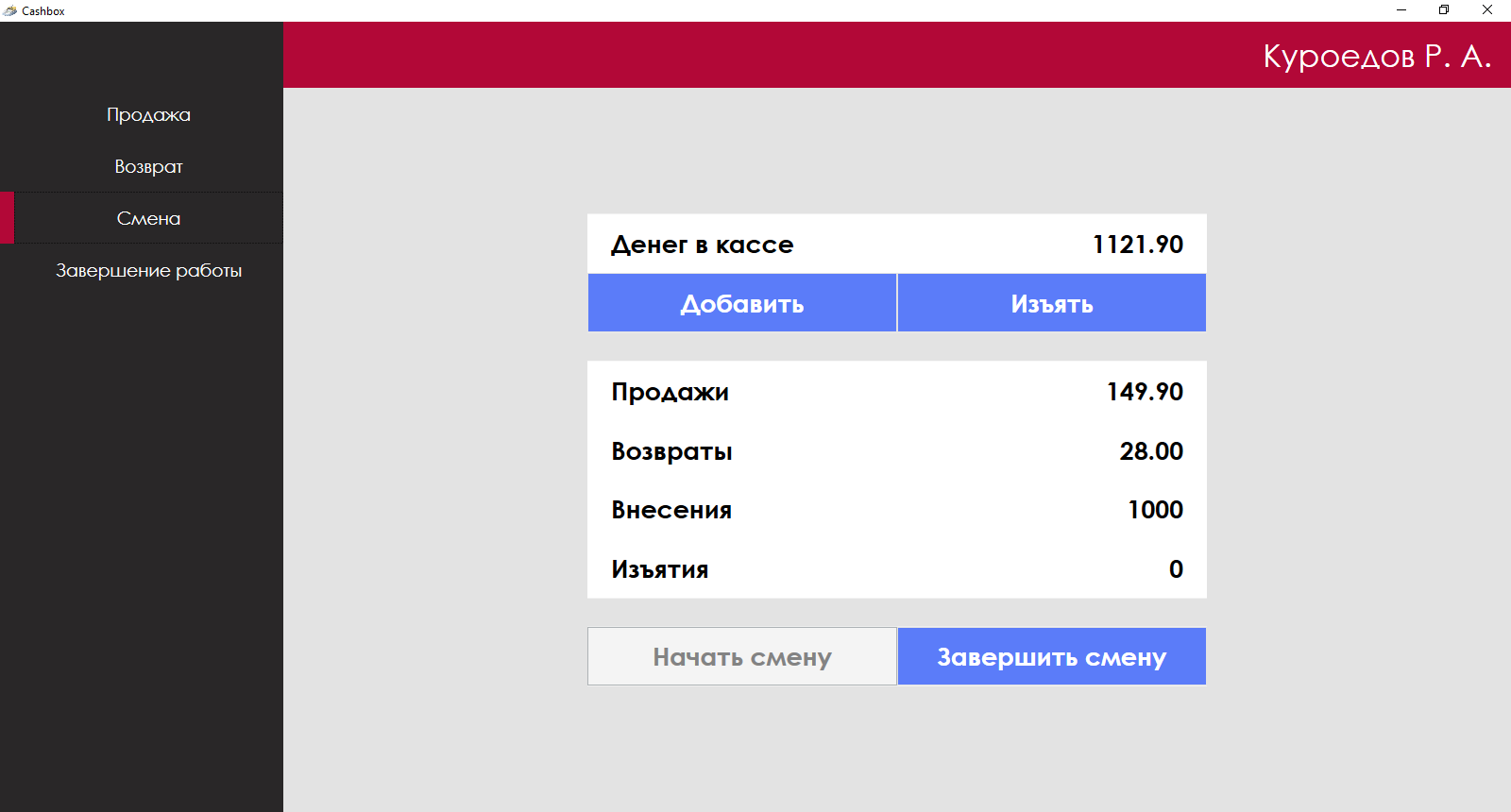


Рисунок Е.8 — Статистика после возврата

Окончание ПРИЛОЖЕНИЯ Е

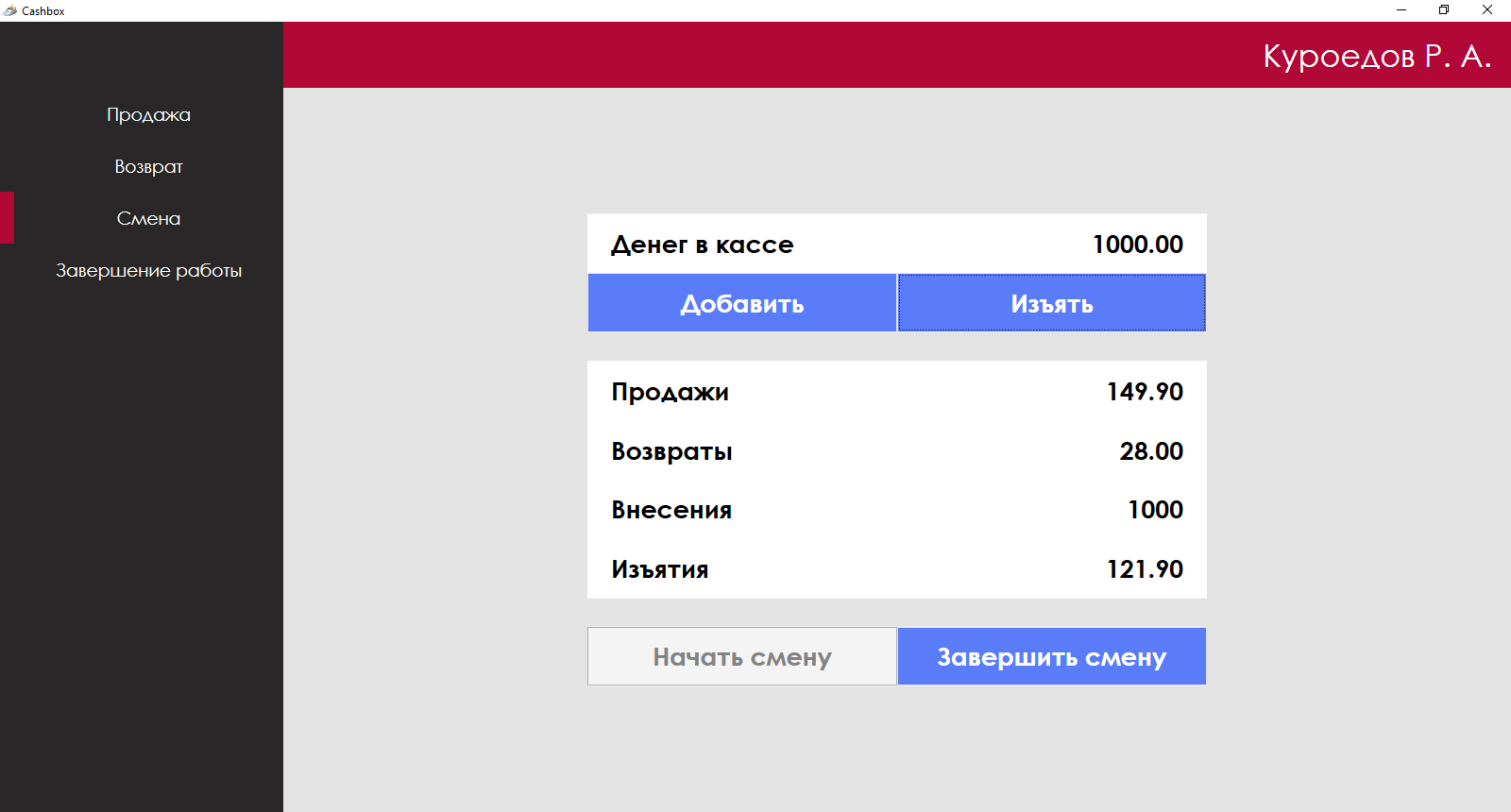


Рисунок Е.9 — Изъятие

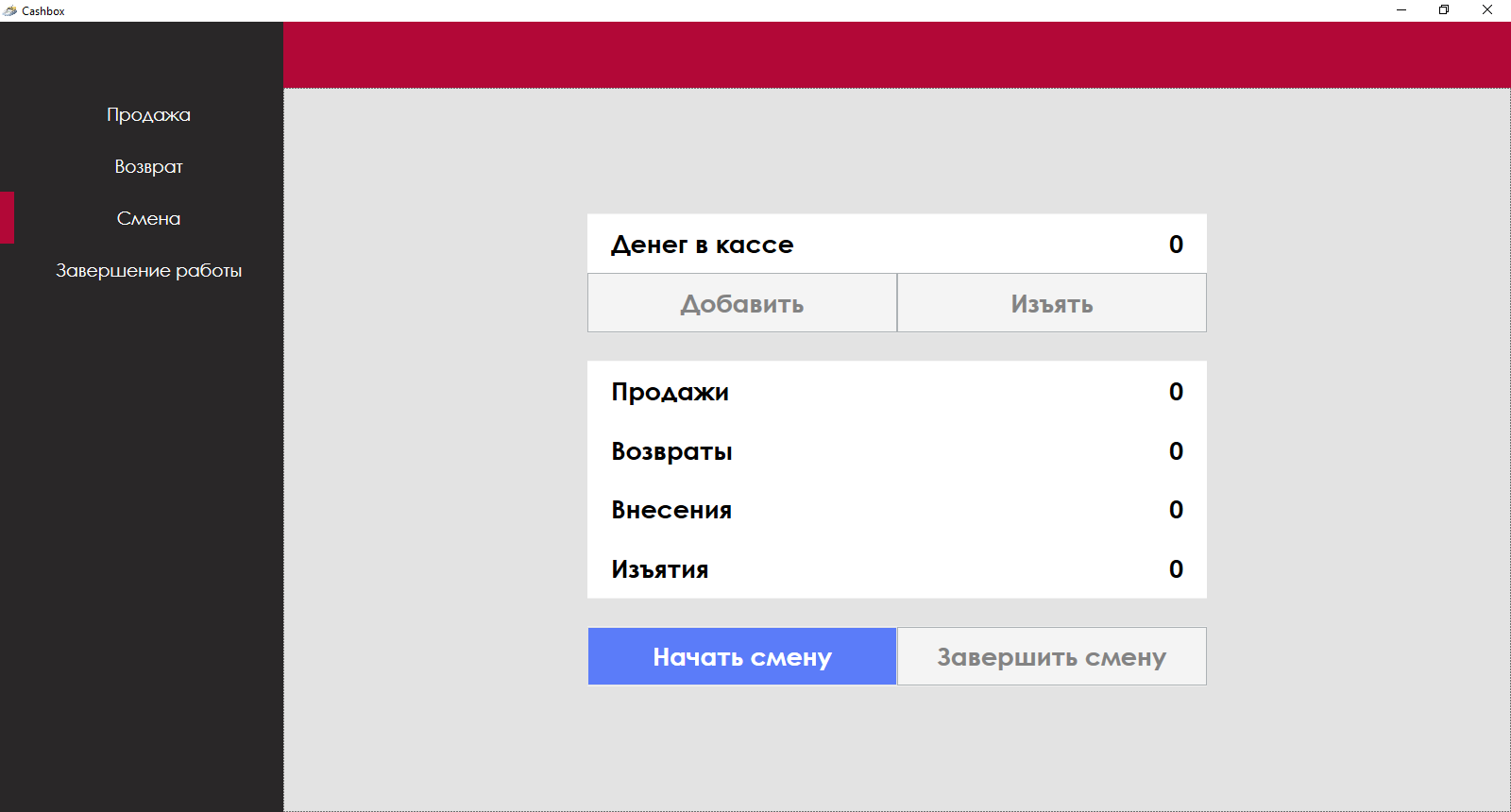


Рисунок Е.10 — Завершение смены