**РЕФЕРАТ**

Пояснительная записка содержит 74 листа, 1 таблицу, 11 источников,   
5 приложений.

РАЗРАБОТКА, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КАССА, ANDROID.

Объектом разработки является проект по автоматизации кассовой деятельности торговых предприятий.

Цель работы ⎯ разработка мобильного приложения для системы онлайн кассы.

В процессе работы был получен опыт мобильной разработки с использованием технологии Xamarin и разработки программной документации.

В результате выполнения дипломной работы был получен программный модуль готовый к внедрению и расширению, а также разработан удобный и понятный для пользователя интерфейс.

Степень внедрения — частичная.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| ВВЕДЕНИЕ | 7 |
| 1 Теоретическое обоснование проекта | 9 |
| 1.1 Анализ предметной области | 9 |
| 1.2 Выбор методов решения задач | 11 |
| 1.2.1 Выбор основной технологии разработки | 11 |
| 1.2.2 Выбор способа хранения данных | 15 |
| 1.3 Техническое задание | 16 |
| 2 Проектная часть | 17 |
| 2.1 Проектирование | 17 |
| 2.1.1 Проектирование структуры данных | 17 |
| 2.1.2 Проектирование вариантов использования | 18 |
| 2.2 Реализация функционала приложения | 20 |
| 2.2.1 Реализация сущности «Пользователь» | 20 |
| 2.2.2 Реализация сущности «Смена» | 21 |
| 2.2.3 Реализация сущности «Товар» | 22 |
| 2.2.4 Реализация сущности «Чек» | 23 |
| 2.2.5 Реализация сущности «Товар чека» | 24 |
| 2.2.6 Реализация сущности «Приёмка» | 25 |
| 2.2.7 Реализация сущности «Списание» | 26 |
| 2.3 Разработка пользовательского интерфейса | 26 |
| 2.4 База данных | 27 |
| 2.4.1 Реализация базы данных | 27 |
| 2.4.2 Нормализация базы данных | 28 |
| 2.5 Разработка руководства оператора | 29 |
| 3 Тестирование программного продукта | 31 |

|  |  |
| --- | --- |
| ЗАКЛЮЧЕНИЕ | 33 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 35 |
| ПРИЛОЖЕНИЕ А (обязательное) Текст технического задания | 36 |
| ПРИЛОЖЕНИЕ Б (обязательное) UML-диаграммы | 45 |
| ПРИЛОЖЕНИЕ В (обязательное) Скриншоты пользовательского интерфейса | 46 |
| ПРИЛОЖЕНИЕ Г (обязательное) Результаты тестирования | 52 |
| ПРИЛОЖЕНИЕ Д (обязательное) Текст руководства оператора | 58 |

**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 [8].

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 [8].

Целью данного проекта является разработка мобильного приложения для системы онлайн кассы.

Для выполнения проекта 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 программный модуль для мобильных, работающих на базе операционной системы Android;
* разработать программную документацию на данный программный модуль;
* 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 онлайн. 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.

Предметная область — онлайн касса. Онлайн-касса — это контрольно-кассовая техника, или ККТ, с модулем для выхода в интернет и фискальным накопителем. Она передает данные о каждой покупке ОФД — оператору фискальных данных. ОФД в свою очередь передает сведения в Федеральную налоговую службу. При необходимости онлайн-касса печатает обычные чеки, а электронные передает покупателям по номеру телефона или email.

Главными задачами онлайн 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 финансовых операций;
* автоматизация управления складом.

Базовые сущности предметной области:

* смена. Атрибуты: id смены, id пользователя, дата и время начала и завершения смены, текущий баланс, сумма продаж, сумма возвратов, сумма внесений в баланс, сумма изъятий из баланса;
* чек. Атрибуты: id чека, id смены, дата и время регистрации, итог, тип чека, товары;
* товар. Атрибуты: id товара, значение штрих-кода, наименование товара, количество, цена, скидка;
* пользователь; Атрибуты: — id пользователя, логин, пароль, фамилия, имя, отчество, роль, дата и время регистрации;
* приёмка товара. Атрибуты: id приёмки, id смены, товар, основание приёмки, количество принимаемого товара, цена закупки за единицу товара, дата и время приёмки;
* списание товара. Атрибуты: id приёмки, id смены, товар, основание списание, количество списываемого товара, дата и время списания.

1. **Выбор методов решения задач**

**1.2.1 Выбор основной технологии разработки**

Когда принимается решение о создании какого-либо программного продукта всегда остро встаёт вопрос об инструментах и методах разработки. Данный выбор по большей части зависит от платформы, на которой будущее приложение должно работать. На этапе выбора платформы можно выделить два подхода к разработке приложений: нативная (родная) и кроссплатформенная разработка.

При нативной разработке используются оригинальные языки программирования (далее ЯП) и инструменты операционной системы. Для Android — Java, а для iOS — Objective-C, Swift, C и С++. К преимуществам нативной разработки относятся наибольшая скорость работы приложения, интеграция с платформой и наиболее привычный для пользователей платформы вид пользовательского интерфейса. Однако, при необходимости разработки приложения для нескольких платформ затраты пропорционально увеличиваются, так как необходимо реализовывать одну и ту же логику по нескольку раз.

При нативной разработке используются специальные инструменты, позволяющие создавать приложения для нескольких платформ. Логика таких приложений сначала реализуется с использованием рабочего для фреймворка ЯП, после чего интерпретируется в нативный код. В отличие от нативной разработки, кроссплатформенная разработка не требует многократной реализации бизнес-логики приложения для нескольких платформ, что существенно экономит деньги и время. Однако за это приходится расплачиваться несколько более низкой скоростью работы приложения и, возможно, непривычным для пользователей конкретной платформы интерфейсом.

Однако в текущих реалиях по мере создания новых и развития уже существующих технологий кроссплатформенной разработки разрыв в скорости работы между нативными и кроссплатформенными приложениями стремительно сокращается. При этом готовность к несущественной доработке и немедленному развёртыванию приложений на иных платформах с каждым днём ценится всё сильнее, ибо это гораздо легче и менее затратно, нежели разрабатывать новую версию уже существующего приложения для отдельной платформы. Именно поэтому на данный момент разрабатывать данный проект куда выгоднее, используя кроссплатформенный подход.

В качестве инструментов разработки стоит рассмотреть 2 технологии: Xamarin, React Native.

React Native — это технология для разработки кроссплатформенных приложений для iOS и Android от компании Facebook. Данная технология использует ЯП JavaScript и базируется на библиотеке React. К достоинствам данной технологии можно отнести:

* использование популярного ЯП в качестве основы;
* ограниченная поддержка CSS, что позволяет создавать гибкий дизайн;
* возможность использования нативных элементов для оптимизации «узких мест» приложения;
* наиболее близкий к нативному пользовательский интерфейс.

К недостаткам же можно отнести использование того же JavaScript, который со временем продолжает улучшаться, но всё же в основе своей имеет недочёты. Помимо этого, JavaScript — интерпретируемый ЯП, из-за чего приложения, реализованные с его использованием, проигрывают в скорости работы приложениям, в основе которых используются компилируемые ЯП. Также к недостаткам можно отнести и смешанность в коде как JavaScript, так и CSS и язык разметки, что усложняет структуризацию и чтение программы программистами.

Xamarin — технология для разработки кроссплатформенных мобильных приложений с использованием ЯП C#, базирующаяся на платформе .NET от компании Microsoft.

К преимуществам данной технологии можно отнести:

* возможность параллельной разработки пользовательского интерфейса и логики приложения, благодаря наличию языка разметки XAML;
* близкий к нативному пользовательский интерфейс;
* возможность создания нативного интерфейса для каждой из поддерживаемых платформ;
* отличная интеграция с сервисами Microsoft;
* возможность одновременного создания как мобильного, так и десктопного приложения на базе UWP.

Недостатком можно назвать несколько больший размер приложений по сравнению с другими кроссплатформенными решениями.

Также упоминания доступна технология Flutter от компании Google, однако из-за использования ЯП Dart, как вариант не рассматривается.

В конечном итоге разработку приложения принято осуществлять с помощью технологии Xamarin.

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

ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET. 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 осуществляется работа с базой данных.

Entity Framework Core (EF Core) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping — отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища [2]. Если на физическом уровне осуществляются операции с таблицами, индексами, первичными и внешними ключами, то на концептуальном уровне, предоставляемом EF Core, осуществляются операции уже с объектами [2].

Центральной концепцией Entity Framework является понятие   
сущности (entity). Сущность представляет собой набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их коллекциями [2].

Отличительной чертой Entity Framework Core является использование запросов, составленных с помощью языка интегрированных запросов LINQ для выборки данных из базы данных.

**1.2.2 Выбор способа хранения данных**

Так как конечной платформой для работы приложения служит мобильное устройство, то можно выделить 3 подхода к хранению данных:

* хранения данных в текстовом файле формата json, либо xml;
* хранение данных в базе данных SQLite;
* хранение данных на удалённом сервере.

Хранение данных с использованием текстового файла является наиболее простым решением, так как для работы с ним не требуется использование дополнительных библиотек, и, следовательно, не требуется дополнительное место в памяти устройства для работы приложения. Для работы с данными достаточно настроить шаблон извлечения данных из файла. С другой стороны, для чтения файла необходимо выгрузить его в оперативную память, что при большом количестве данных может привести к её существенному перерасходу.

При использовании локальной базы данных SQLite появляется потребность в дополнительных компонентах, предназначенных для работы с базами данных, для которых необходим дополнительный объем памяти. Однако зачастую требуемым компонентам необходим несущественный объем памяти.   
В то же время для работы с базами данных не нужно полностью выгружать базу данных в оперативную память, что в свою очередь убирает проблему перерасхода оперативной памяти. При этом, скорость доступа к локальному хранилищу определяется лишь быстродействием устройства, на котором приложение осуществляет свою работу.

При хранении данных на удалённом сервере с приложения снимается излишняя нагрузка за обработку данных, так как основная работа с данными осуществляется на стороне сервера, а приложение лишь получает готовый ответ в заранее определенном формате, что в свою очередь увеличивает его быстродействие. Помимо этого, при данном подходе не требуется дополнительное место как для самой базы данных, так и для библиотек для работы с ней. Однако при данном подходе скорость получения данных значительно ниже, чем при использовании двух предыдущих подходов, и полностью зависит от скорости соединения между приложением и удалённым сервером.

Исходя из приведённых выше аргументов, в качестве способа хранения данных выбрана локальная база данных SQLite.

**1.3 Техническое задание**

Разработка программного продукта производится на основании технического задания А.В.00001-01 ТЗ 01-ЛУ. Текст технического задания представлен в приложении А.

**2 Проектная часть**

1. **Проектирование**

**2.1.1 Проектирование структуры данных**

Любое приложение так или иначе работает с данными, однако ни одно приложение не может обрабатывать хаотичные наборы данных. Данные, с которыми работает приложение, предварительно должны быть структурированы для их последующей обработки. Абстрагируясь от конкретных объектов и особенностей языка реализации, наиболее подробную структуру данных можно разработать с применением модели ER-модели.

ER-модель (модель «Сущность-связь») — модель данных, позволяющая описывать концептуальные схемы предметной области. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями [3].

Основные понятия:

* сущность — это объект, который может быть идентифицирован неким способом, отличающим его от других объектов. Сущность фактически представляет из себя множество атрибутов;
* связь — это ассоциация, установленная между несколькими   
  сущностями [3].

Существуют 3 типа связей между сущностями:

* один к одному — одиночный экземпляр сущности одного класса связан с одиночным экземпляром сущности другого класса;
* один ко многим — одиночный экземпляр сущности одного класса связан со многими экземплярами сущности другого класса;
* многие ко многим — многие экземпляры сущности одного класса связаны со многими экземплярами сущности другого класса [3].

Для разрабатываемого приложения была разработана ER-диаграмма, представленная в приложении Б, рисунок Б.1. Данная диаграмма показывает общую структуру организации данных и объектов приложения.

Конечные сущности данных:

* смена;
* пользователь;
* товар;
* чек;
* товар чека;
* списание;
* приёмка.

**2.1.2 Проектирование вариантов использования**

Любое приложение обладает набором функций, которые при начале разработки, как правило, оговорены между заказчиком и исполнителем и зафиксированы в техническом задании. При разработке программного продукта, важно предусмотреть различные варианты использования (прецеденты) данных функций и последовательность их выполнения. Также необходимо распределить данные функции между группами пользователей, если таковые имеются. Схему организации прецедентов наиболее наглядно представлять с помощью диаграммы прецедентов (UseCase-диаграммы).

Диаграмма прецедентов — диаграмма, отражающая отношения между участниками и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне [4].

Основные элементы диаграммы:

* участник — это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс) [4]. Участником может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности;
* прецедент — описание множества последовательных событий (включая варианты), выполняемых системой, которые приводят к наблюдаемому участником результату [4]. Прецедент представляет поведение сущности, описывая взаимодействие между участниками и системой. Прецедент не показывает процесс достижения некоторого результата, а только конкретное действие [4].

Для разрабатываемого приложения была разработана диаграмма прецедентов, представленная в приложении Б, рисунок Б.2.

Данная диаграмма показывает основной набор операция, осуществляемых действующими лицами с помощью разрабатываемого приложения:

* регистрация пользователей;
* изменение данных пользователей;
* удаление данных пользователей;
* регистрация товара;
* приёмка товара;
* списание товара;
* изменение данных товара;
* удаление данных товара;
* старт смены;
* завершение смены;
* авторизация;
* внесение средств;
* изъятие средств;
* продажа товара;
* возврат товара.

**2.2 Реализация функционала приложения**

При разработке приложения его компоненты принято структурировать в отдельные подсистемы для облегчения дальнейшей поддержки и модификации разработанного приложения. Выделенные подсистемы без учёта пользовательского интерфейса и механизмов из взаимодействия вместе представляют собой ядро приложения.

В данном случае ядро разрабатываемого приложения можно разделить на следующие компоненты:

* набор сущностей, реализованных в виде классов;
* контекста данных, с помощью которого производится взаимодействие с базой данных;
* методы, представляющие собой реализацию перечисленных ранее функций.

Для реализации связей между сущностями каждый класс предусматривает наличие навигационных свойств, представляющих собой атрибуты, содержащие экземпляры связанных сущностей.

**2.2.1 Реализация сущности «Пользователь»**

Данная сущность представляет собой компонент для работы с учётными записями пользователей приложения. Сущность содержит следующие атрибуты:

* идентификатор пользователя;
* логин;
* пароль;
* фамилия;
* имя;
* отчество;
* роль;
* дата и время регистрации.

Атрибуты «логин», «пароль», «фамилия», «имя» и «отчество» имеют строковый тип данных. Идентификатором пользователя является целочисленное значение, генерируемое автоматически при добавлении нового пользователя. В приложении предусмотрено разделение прав доступа между рядовыми операторами и администраторами, для чего и служит атрибут «роль», хранящий значение перечисления, содержащего заранее определённые роли пользователей.

Для данной сущности реализованы методы для работы с данными, предоставляющие функционал для регистрации, авторизации, изменения данных и удаления данных учётных записей пользователей. При регистрации, авторизации или изменении пользователей производится шифрование передаваемого значения пароля с использованием алгоритма SHA-512.

**2.2.2 Реализация сущности «Смена»**

Данная сущность представляет собой компонент для работы с данными смен приложения. Сущность содержит следующие атрибуты:

* идентификатор смены;
* идентификатор пользователя;
* дата и время начала;
* дата и время завершения;
* баланс;
* сумма продаж;
* сумма возвратов;
* сумма внесений;
* сумма изъятий;
* состояние.

Атрибуты «баланс», «сумма продаж», «сумма возвратов», «сумма внесений» и «сумма изъятий» имеют представляют собой действительные числа. Идентификатором смены является целочисленное значение, генерируемое автоматически при добавлении новой записи о новой смене. Идентификатор пользователя представляет собой целочисленное значение и ссылается на конкретный экземпляр учётной записи пользователя. В приложении предусмотрена возможность восстановления данных последней смены для конкретного пользователя в зависимости от её состояния. Атрибут «состояние» хранит значение перечисления, содержащего следующие значения:

* created — экземпляр смены создан;
* running —смена начата;
* finished — смена завершена.

Восстановлению подлежать экземпляры сущности со значением атрибута «состояние», равным «running».

Для данной сущности реализованы методы для работы с данными, предоставляющие функционал для добавления, старта, обновления данных продаж и возвратов, произведения внесения и изъятия.

**2.2.3 Реализация сущности «Товар»**

Данная сущность представляет собой компонент для работы с данными товаров. Сущность содержит следующие атрибуты:

* идентификатор товара;
* код;
* наименование;
* количество;
* цена;
* скидка.

Идентификатором товара является целочисленное значение, генерируемое автоматически при добавлении сущности данного типа. Атрибут «код» представляет собой уникальное строковое значение, содержащее   
буквенно-цифирную интерпретацию штрих-кода. Атрибут «наименование» содержит наименование товара в строковом формате. Атрибут «количество» содержит действительное число, отражает текущее количество конкретного товара на складе предприятия и не может принимать отрицательное значение. Атрибут «цена» отражает цену товара за единицу количества и представляется действительным числом. Атрибут «скидка» содержит целочисленное значение скидки в процентах в интервале от 0 до 100.

Для данной сущности реализованы методы для работы с данными, предоставляющие функционал для добавления, изменения и удаления данных продаж и возвратов, произведения внесения и списания.

**2.2.4 Реализация сущности «Чек»**

Данная сущность представляет собой компонент для работы с данными чеков. Сущность содержит следующие атрибуты:

* идентификатор чека;
* идентификатор смены;
* дата и время регистрации;
* итог;
* тип.

Идентификатором чека является целочисленное значение, генерируемое автоматически при добавлении сущности данного типа. Идентификатор смены представляет собой целочисленное значение и ссылается на идентификатор конкретного объекта смены. Атрибут «Дата и время регистрации» представляет собой значение типа даты и времени, содержащее дату и время момента, когда была произведена продажа или возврат товара. Атрибут «итог» отражает суммарную стоимость всех товаров, взводящих в чек с учётом их скидок и представляется действительным числом. Атрибут «тип» хранит значение перечисления, значения которого указывают на принадлежность чека к операции продажи, либо к операции возврата.

Для данной сущности реализованы методы для работы с данными, предоставляющие функционал для добавления и их поиска.

**2.2.5 Реализация сущности «Товар чека»**

Данная сущность является связующей между сущностью «Чек» и «Товар». Сущность содержит следующие атрибуты:

* идентификатор чека;
* идентификатор товара;
* количество;
* цена;
* скидка.

Идентификатором чека является целочисленное значение, ссылающееся на конкретный экземпляр чека. Идентификатор товара представляет собой целочисленное значение и ссылается на конкретный экземпляр товара. Атрибут «количество» содержит действительное число, отражает количество конкретного товара, проданного при регистрации чека. Атрибут «цена» отражает цену товара за единицу количества на момент продажи товара и представляется действительным числом. Атрибут «скидка» содержит целочисленное значение скидки в процентах в интервале от 0 до 100 на момент продажи товара.

**2.2.6 Реализация сущности «Приёмка»**

Данная сущность является вспомогательной сущностью для сущности «Товар» и отвечает за пополнение запасов товаров. Сущность содержит следующие атрибуты:

* идентификатор приёмки;
* идентификатор смены;
* идентификатор товара;
* основание;
* количество;
* цена закупки;
* дата и время приёмки.

Идентификатор приёмки является целочисленное значение, генерируемое автоматически при добавлении сущности данного типа. Идентификатором смены является целочисленное значение, ссылающееся на конкретный экземпляр смены. Идентификатор товара представляет собой целочисленное значение и ссылается на конкретный экземпляр товара. Атрибут «основание» представляет собой строковое значение, которое зачастую ссылается на документы о поставке. Атрибут «количество» содержит действительное число, отражающее количество конкретного товара, закупаемого у поставщика. Атрибут «цена» отражает цену товара за единицу количества на момент продажи товара и представляется действительным числом.

**2.2.7 Реализация сущности «Списание»**

Данная сущность является вспомогательной сущностью для сущности «Товар» и отвечает за списание некоторых запасов товаров. Сущность содержит следующие атрибуты:

* идентификатор списания;
* идентификатор смены;
* идентификатор товара;
* основание;
* количество;
* дата и время списание.

Идентификатор списания является целочисленное значение, генерируемое автоматически при добавлении сущности данного типа. Идентификатором смены является целочисленное значение, ссылающееся на конкретный экземпляр смены. Идентификатор товара представляет собой целочисленное значение и ссылается на конкретный экземпляр товара. Атрибут «основание» представляет собой строковое значение, которое зачастую ссылается на документы о списании, либо указывает на его причину. Атрибут «количество» содержит действительное число, отражающее количество конкретного товара, списываемого со склада.

**2.3 Разработка пользовательского интерфейса**

Первым делом необходимо определить структуру приложения. Xamarin предоставляет 4 шаблона приложения на выбор:

* Blank App — пустой шаблон, создающий проект с минимальной функциональностью;
* Tabbed App — проект приложения, которое использует вкладки для навигации между страницами;
* Shell — шаблон одностраничного приложения;
* Master Detail — проект для специально для тех случаев, когда должна быть функциональность для представления списка объектов и функциональность для вывода информации по каждому отдельному элементу списка.

Для данного проекта используется шаблон Master Detail. Пользовательский интерфейс создан при помощи языка разметки XAML Скриншоты пользовательского интерфейса представлены в приложении В.

**2.4 База данных**

**2.4.1 Реализация базы данных**

На основе анализа предметной области и созданной ER-диаграммы, представленной в приложении Б на рисунке Б.1, с помощью средств   
Entity Framework Core была разработана база данных, включающая в себя следующие объекты:

* таблица Items — таблица, хранящая данные о товарах;
* таблица Returns — таблица, хранящая данные о возвратах товара;
* таблица Sales — таблица, хранящая данные о продажах товара;
* таблица Shifts — таблица, хранящая данные о сменах;
* таблица Receipts — таблица, хранящая данные о чеках;
* таблица ReceiptItems — связующая таблица между таблицами Items и Receipts.;
* таблица Users — таблица, хранящая данные об операторах системы.

Так как связь базы данных и приложения осуществляется при помощи Entity Framework Core, то благодаря подходу Code First в приложении при помощи средств языка программирования можно создать контекст данных, при создании экземпляра которого, если база данных не существует, будет сгенерировано SQL-выражение для создания базы данных со структурой, соответствующей моделям, описанным в приложении. Выполнение запросов производится при помощи LINQ-запросов, которые, благодаря Entity Framework, транслируются в соответствующие SQL-выражения.

**2.4.2 Нормализация базы данных**

Обязательной составляющей разработки любой базы данных является нормализация таблиц базы данных. Нормализацией называется это процесс изменения информации и приведения общей структуры данных к нормальным формам. Благодаря нормализации, удаётся избежать структурных и логических проблем данных. Если в базе данных существуют одинаковые записи, то велика вероятность нарушения целостности данных при обновлении таблицы, но так структура процесса нормализации предполагает наличие связей между данными, нет необходимости в идентичных записях.

В общем случае для нормализации таблиц базы данных достаточно привести их к первой, второй и третьей нормальным формам. Для приведения к ним необходимо определить признаки каждой из трех нормальных форм:

1. признаки первой нормальной формы:

* отсутствие дублированных строк;
* хранение только атомарных значений в полях строки;
* хранение в поле данных лишь одного типа;
* отсутствие массивов и списков в любом виде;

1. признаки второй нормальной формы:

* наличие первичного ключа;
* наличие зависимости всех неключевых атрибутов от   
  первичного ключа;

1. признаки третьей нормальной формы:

* отсутствие зависимостей между неключевыми атрибутами.

Исходя из ER-модели, представленной в приложении Б на рисунке Б.1, можно сказать, что разработанная база данных полностью соответствует ранее приведённым критериям трёх нормальных форм и не требует нормализации.

**2.5** **Разработка руководства оператора**

Руководство оператора — документ, назначение которого заключается в предоставлении людям помощи в использовании некоторой системы [5]. Документ входит в состав технической документации на систему и, как правило, подготавливается техническим писателем [5].

Целевой аудиторией данного документа являются операторы, работающие с системой, частью которой является программа.

Задачей данного документа является обеспечение оператора возможностью выполнять свои обязанности в отношении системы (в состав которой входит программа) или каких-либо ее частей.

Так как объектом разработки данного проекта является мобильное приложение как компонент системы онлайн кассы, то разрабатывать руководство оператора целесообразно с использованием ГОСТ 19.505-79 ЕСПД «Руководство оператора. Требования к содержанию и оформлению».

Текст руководства пользователя 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 [6].

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 [6].

Ручное тестирование — это процесс поиска дефектов в работе программы, когда тестер проверяет работоспособность всех компонентов программы, как если бы он был пользователем [7]. Часто, для точности проверки, тестер использует заранее заготовленный план тестирования, в котором отмечены наиболее важные аспекты работы программы

Таблица1 — Тестовые данные

|  |  |  |
| --- | --- | --- |
| Наименование | Значение | Результат |
| Авторизация | Логин: root  Пароль: root | Переход на главную страницу |
| Начать смену | клик | Активация кнопок «продажа», «Возврат», «Пользователи», «Товар», «Внести», «Завершить. Деактивация кнопки «Начать» |
| Внести | 500 | Внесено: 500. Баланс: 500. Активация кнопки «Изъять» |
| Новый товар | Код: 12345678  Наименование: Хлеб  Цена: 30  Скидка: 0 | Добавлен новый товар |
| Изменение данных | Код: 12345678  Наименование: Хлеб  Цена: 30  Скидка: 10 | Скидка товара изменена на 10 |
| Приёмка | Код: 12345678  Основание: документ 1  Количество: 50  Цена закупки: 15 | Остаток увеличен с 0 до 50, итог — 750 |
| Продажа | Код: 12345678  Количество: 5  К оплате: 500 | Итог — 135, сдача — 365, идентификатор чека — 1, баланс смены — 635, продажи — 135, остаток — 45 |
| Возврат | Чек: 1  Количество: 2 | К возврату 54 руб. , баланс смены — 181, возвраты — 54, остаток — 47 |
| Изъять | 181 | Баланс — 400, изъято — 181 |
| Списать | Код: 12345678  Основание: просрочка  Количество: 10 | Остаток — 37 |
| Редактировать пользователя | Пароль: 1243 | Изменения сохранены |
| Завершить смену | клик | Обнуление показателей |
| Выход | клик | Переход на страницу авторизации |

Результаты тестирования представлены в приложении Г.

**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 дипломной 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 всего выше перечисленного, можно сказать, что цель дипломного проекта достигнута путем реализации поставленных задач.

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

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. METANIT.COM — Сайт о программировании [Электронный ресурс]. — Режим доступа: <https://metanit.com/sharp/xamarin/>. — Дата доступа: 05.05.20.
2. METANIT.COM — Сайт о программировании [Электронный ресурс]. —Режим доступа: <https://metanit.com/sharp/entityframeworkcore/1.1.php>. —   
   Дата доступа: 28.04.20.
3. Блог о Linux, Python, Vim и другом здоровом образе жизни [Электронный ресурс]. — Режим доступа: <https://proft.me/2013/05/27/uml-2-tipy-diagramm/#er>. — Дата доступа: 27.04.20.
4. НОУ «ИНТУИТ» [Электронный ресурс]. —   
   Режим доступа: <https://www.intuit.ru/studies/courses/1007/229/lecture/5954>. —   
   Дата доступа: 25.04.20.
5. Электронная библиотека студента «Библиофонд» [Электронный ресурс]. — Режим доступа: <https://www.bibliofond.ru/view.aspx?id=784939>. —   
   Дата доступа: 14.05.20.
6. Липаев, В. В. Программная инженерия. Методологические основы /   
   В. В. Липаев. – Москва : Теис, 2006. — 608 с.
7. Ручное и автоматизированное тестирование [Электронный ресурс]. — Режим доступа: <http://aprozorova.blogspot.com/2017/10/blog-post_11.html>. —   
   Дата доступа: 12.05.20.
8. Frontol 5. Руководство интегратора [Электронный ресурс]. —   
   Режим доступа: <https://www.atol.ru/>. — Дата доступа: 20.04.20.
9. Тюкачев, Н. А. С#. Основы программирования / Н. А. Тюкачев, В. Г. Хлебостроев, Джей Глинн. — Санкт-Петербург : Издательство Лань, 2018. — 272 с.
10. Вагнер, Б. Наиболее эффективное программирование на С# / Б. Вагнер. — Москва : Издательство Альфа-книга, 2018. — 240 с.
11. Троелсен, Э. Язык программирования С# 7 и платформы .NET и .NET Core / Э. Троелсен, Ф. Джепикс. — Киев : Издательство Диалектика, 2019. — 1328 с.