МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ВНЕДРЕНИЕ НОВЫХ ТЕХНОЛОГИЙ В POS-СИСТЕМЫ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Шведова Евгения Валерьевича

Научный руководитель
доцент, к.ф.-м.н. _____ А. С. Иванова

Заведующий кафедрой
к.ф.-м.н., доцент _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Всего несколько десятилетий назад система торговых точек в своей основной форме представляла собой кассовый аппарат. Человек, управляющий кассовым аппаратом, вручную вводил цены на купленные товары, часто с помощью ценовых билетов.

Затем продавец брал деньги, клал их в кассу и вручали клиенту бумажный чек. В некоторых случаях единственной записью о сделке была бумажная копия квитанции.

По мере развития систем торговых точек они становились все более компьютеризированными, сохраняя базу данных продуктов на цифровом сервере.

Часто они включали в себя сканер штрих-кода, поэтому не было необходимости вводить цену вручную, а также хранить детали транзакции в электронном виде. [1]

В наше время все гораздо сложнее. Хотя все системы, что описаны выше, всё ещё используются, но на данный момент это является лишь небольшой частью огромного комплекса устройств под названием POS.

Тем не менее, развитие POS-систем не стоит на месте и по сей день ритейлеры стремятся развить технологии как можно лучше и удобней как для обычных покупателей, так и для продавцов.

Целью данной работы является разработка и внедрение клиентской части поддержки «умных» весов (весы определяют не только вес, но и распознают категорию товара, что позволяет увеличить пропускную способность кассы) и ассортиментного ценника (единый ценник для похожих категорий товаров) по заказу ритейлера.

Для выполнения данной цели необходимо:

1. создать класс для взаимодействия с «умными» весами;
2. реализовать функцию для определения товара и добавления в чек;
3. описать штрих-код ассортиментного ценника;
4. реализовать функцию сканирования ассортиментного ценника и товара из ассортиментной линейки.

Бакалаврская работа состоит из введения, двух глав, заключения, списка использованных источников и семи приложений. Объем работы 47 страниц. Список литературы включает 20 наименований.

В теоретической части были описаны используемые технологии и что из

себя представляет функция в POS-системе. В экспериментальной части работы описаны разработанные функции, продемонстрированы результаты работы функций.

1 Теоретическая часть

1.1 База данных

Согласно информации из книги К. Дж. Дейта «Введение в системы баз данных» [2], база данных — это некоторый набор перманентных (постоянно хранимых) данных, используемых прикладными программными системами какого-либо предприятия.

Преимуществами подхода, предусматривающего использование базы данных являются:

- Возможность совместного доступа к данным ;
- Сокращение избыточности данных;
- Устранение противоречивости данных;
- Возможность поддержки транзакций;
- Обеспечение целостности данных;
- Организация защиты данных;

1.2 JPOS

JPOS (сокращение от Java for Point of Sale Devices) [3] — это стандарт программного обеспечения для взаимодействия с точками продаж (POS), написанный на Java, со специализированным аппаратным периферийным оборудованием, обычно используемым для создания системы торговых точек. Преимуществами являются снижение стоимости POS-терминалов, независимость от платформы и снижение административных расходов. JPOS был основан на стандарте драйвера устройства POS для Windows, известном как OPOS. С тех пор JPOS и OPOS были объединены в общий стандарт UnifiedPOS. [4]

Стандарт JPOS включает определения «Объекты управления» и «Объекты обслуживания». Программное обеспечение POS связывается с объектами управления. Объекты управления загружаются и взаимодействуют с соответствующими объектами обслуживания. Служебные объекты иногда называют «JPOS драйверами».

1.3 Архитектура функции

В дальнейшем во время описании разработки можно будет часто услышать такое слово как **«Функция»**. Для лучшего понимания и ориентации среди классов следует сначала примерно описать как устроены эти самые функции и что они из себя представляют.

Вкратце, **Функция** – это набор действий, который производит POS. Вызывается она нажатием кнопки на кассе или же функция может быть вызвана из другой функции.

Архитектурно в первую очередь у функции есть **основной класс**, в котором описано в каком порядке должны будут запущены контроллеры. Если функции надо взаимодействовать с экранными формами, то создаются **модели**, в которых указано, какие элементы должны присутствовать на экране. Затем модели описываются в классе **конфигураторе**. Так же существует интерфейс **keys**, в котором записываются константы.

Контроллеры функции немного напоминают контроллеры из фреймворка spring: имеются, собственно, сами контроллеры, которые запускаются функцией. А для контроллеров делаются сервисы, в которых описан функционал, который можно вынести за пределы контроллера, чтобы сильно не засорять код и для соблюдения SOLID [5] принципов.

2 Внедрение разработок

2.1 Внедрение умных весов

Новое устройство поможет кассирам существенно сократить это время и значительно ускорить обслуживание покупателей. Система весов, оснащенная специальной камерой, которая самостоятельно распознаёт товар даже в пакете, таким образом, пропускная способность каждой кассы увеличивается.

2.2 Разработка функции определения товара

2.2.1 Классы устройства

2.2.2 JStoreSmartCamera

Для начала необходимо создать класс, через который будет происходит взаимодействие с устройством. Для этого был сначала создан интерфейс `JStoreSmartCamera`. Он наследуется от стандартного интерфейса `JStoreDevice`, в котором указаны минимально необходимые методы, для взаимодействия с любым устройством. Сам интерфейс содержит класс `SmartCameraPrediction` и обработчик исключений `SmartCameraException`. В `SmartCameraPrediction` указываются пареметр `guid` и список товаров `predictedList`.

2.2.3 JPosSmartCamera

Дальше был создан класс `JPosSmartCamera`, в котором имплементируются методы интерфесов `JStoreDevice` и `JStoreSmartCamera`. В классе также находится экземпляр класса `SmartCamera`, который находится в библиотеке, предоставленной подрядчиком и в которой происходит прямое взаимодействие с устройством.

Внимание стоит уделить только реализациям следующих методов:

В методе `identification` мы поддаём на вход вес, индификатор кассы и экземпляр класса `Properties`, в который будет записан результат после отправки запроса устройству. На выходе возвращается экземпляр класса `SmartCameraPrediction`, в котором хранятся идентификатор запроса и список предполагаемых продуктов.

В методе `selectChoice` на вход поддаются номер PLU товара и идентификатор запроса и эти данные отправляются устройству для самообучения сети.

Также, что бы POS смог увидеть новое устройство необходимо ввести данные устройства в следующих файлах:

— В файле jpos.xml, в котором указаны JPOS-дравера.
— И в posconfig.xml, в котором указываются само устройство и то, подключено ли оно к POS.

2.2.4 Функция

Так как данный функционал являются расширением уже существующей крупной функции, то в данной дипломной работе будут затрагиваться только те моменты, которые являются частью нового функционала и моменты, которые необходимы для понимания функции.

2.2.5 CheckExistsDevicesController

Первым делом необходимо проверить подключены ли устройства к POS, иначе отсутствие устройства или его временное отключение может привести к непредвиденным ошибкам. Для этого был создан контроллер `CheckExistsDevicesController`, в котором инициализируется обьект `searchType`.

После выполняется проверка. Если функция была вызвана нажатием указанной выше кнопкой, то идёт проверка на наличие камеры определяющей продукты. Если устройство отсутствует, то выводится ошибка об этом, так как в таком случае работа функции невозможна. Не подключенные весы не являются критической ошибкой, в таком случае программа попросит ввести вес товара вручную. Тем не менее необходимо вывести сообщение об отключении в логах.

2.2.6 SearchWeightItemCommandController

После проверки идёт непосредственно поиск товара. Для этого используется контроллер `SearchWeightItemCommandController`.

В зависимости от того как была вызвана функция полностью меняется работа контроллера. Для это было решено создать два класса `GroupProcessor` и `SmartCameraProcessor`, которые имплементируют интерфейс `SearchProcessor` с одним методом `process()`, в котором будет реализована работа контроллера. Определение класса происходит во время инициализации, показанной выше и зависит от параметра `searchType`.

В случая если была выбрана какая-то группа товаров, то будет срабатывать `process()` класса `GroupProcessor`, в котором сначала берутся из локальной базы данных все товары принадлежащие конкретной группе. После полученный список товаров сортируется по максимальному количеству, характеристике веса, и цене. Далее товары синхрнизируются с данными на сервере бэк офиса и после список выводится на экранной форме.

Если же была нажата новая кнопка, то будет срабатывать `process()` класса `SmartCameraProcessor`, в котором:

1. идёт подключение к устройству
2. устройству подаётся команда поиска подходящих весовых товаров.
3. по полученому списку PLU в локальной базе данных происходит поиск подходящих товаров
4. товары сортируются по алфавиту
5. товары синхрнизируются с данными на сервере бэк офиса
6. список выводится на экранной форме (рис. 1)

Если со стороны устройства срабатывает ошибка, то это значит, что устройство не смогло найти подходящие товары и в этом случая пробрасывается исключение и на экран выводится сообщение об этом.

2.2.7 SmartCameraLearningFilter

Последнее, что было добавлено в ходе разработки это дообучение. Для этого был создан специальный класс `SmartCameraLearningFilter`, который срабатывает только при определённых условиях. Данный класс является spring бином [6] и для начала необходимо указать его в отдельном xml-файле.

Далее можно приступать к написанию самого класса.

— Сначала проходит проверка на то, что устройство подключено и что был добавлен весовой товар.
— Если условие проходит, то происходит дообучение устройства.

Так как выбор товара и его продажа не были мной затронуты в ходе данной разработки, то данная часть функции показана не будет.

2.3 Внедренние ассортиментного ценника

Суть предлагаемых нововведений в том, что продавцам разрешат использовать единый ценник для одного наименования товара в ассортименте. Так, например, продукция одного производителя с одним наименованием и по

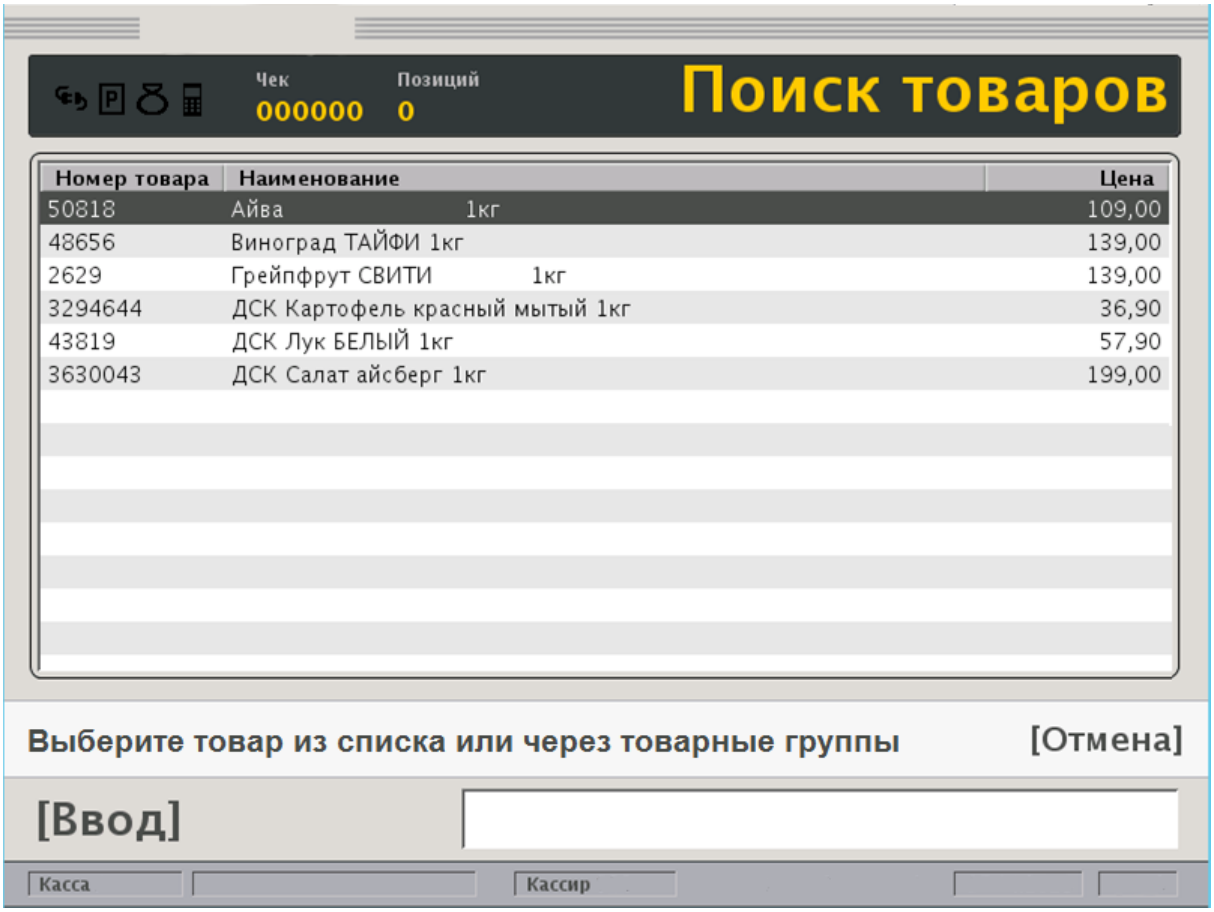

Рисунок 1 – Список предполагаемых товаров

одинаковой цене, но представленная в различных вариациях, сможет иметь один ценник.

2.4 Разработка функции сканирования ассортиментного ценника

Данная разработка является расширением функции добавления товара по ценнику, но так как она затригивает функцию целиком было решено рассказать полностью про всю функцию, но больше акцентируя внимание на тех моментах, которые не были показаны в предыдущей разработке.

2.4.1 Распознание ассортиментного ценника

Так как штрих-код, по сути, является стройкой текста, то нужно как-то понять, что в тексте написаны данные ассортиментного ценника. Для это в отдельном xml-файле, в котором хранятся характеристики штрих-кодов указываем новый ценник.

Стандарт ШК — `CODE128` (строка до 48 символов) [7], а количество символов 18. Главной отличительным символом ассортиментного ценника является символ «$» в начале строки. В качестве номера ассортиментной линейки

берутся первые 10 цифр после определяющего символа. Последние 7 цифр являются ценной.

Например, если сканер после сканирования ШК вернул «$0000000125003105О», то касса воспримет эту строку как ассортиментный ценник, у которого номер ассортиментной линейки 125, а цена товара данной линейки 310 рублей 50 копеек.

2.4.2 Функция

2.4.3 ScanTagBarcodeFormController

В контроллере `ScanTagBarcodeFormController` на экран выводится ЭФ с сообщением отсканировать ценник. (рис. 2)

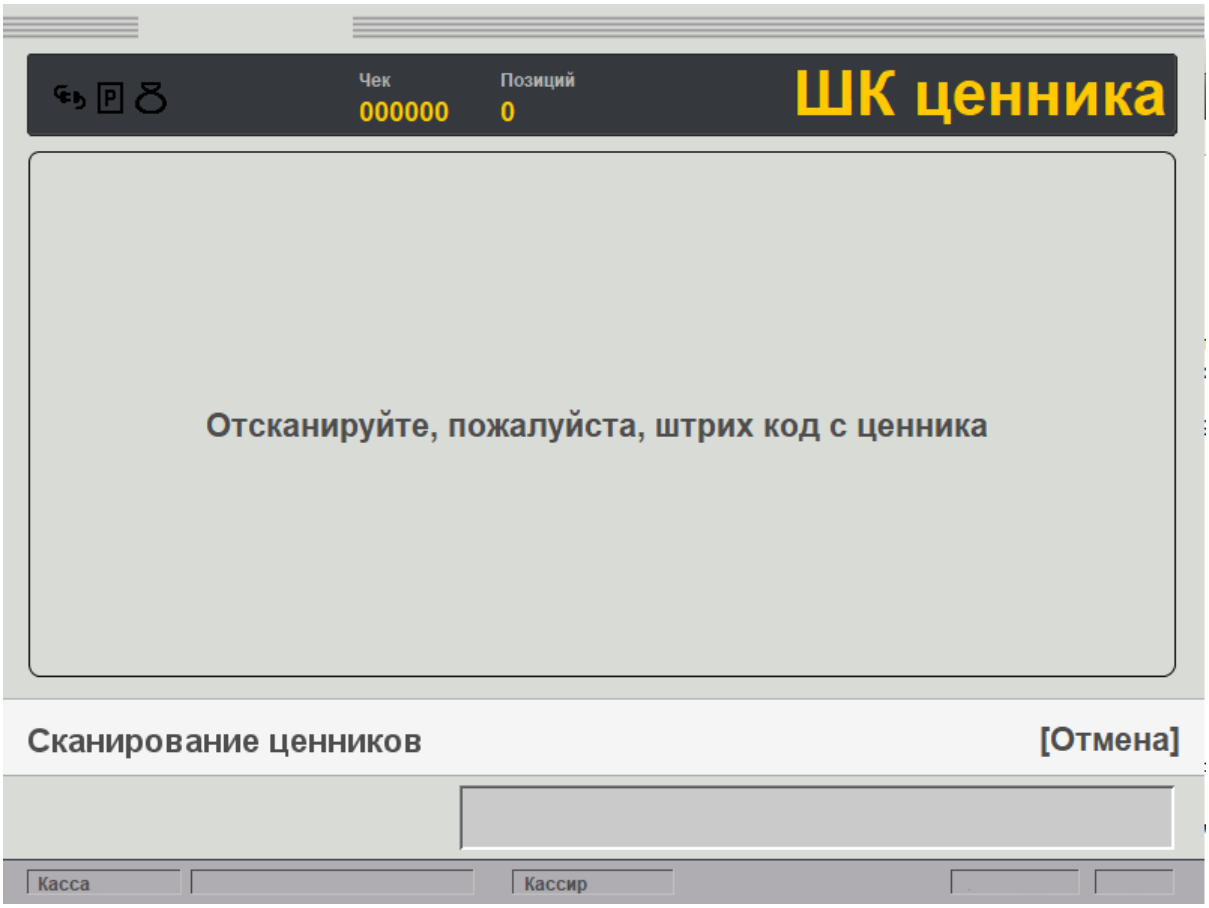

Рисунок 2 – ЭФ с сообщением отсканировать ценник

Сначала из отсканированных параметров берётся и сохраняется параметр `TAG_TYPE`, после чего идёт проверка на тип ценника.

— В случае, если это был обычный ценник, то сохраняются такие параметры как: EAN товара и его цена.
— Если был просканирован ассортиментный ценник, то сохраняются номер

ассортиментной линейки и её цена. После чего происходит проверка на существование данной линейки.

2.4.4 ScanItemBarcodeController

Следующим идёт контроллер `ScanItemBarcodeController`. Он срабатывает только если был просканирован ассортиментный ценник, в противном случае программа переходит сразу к следующему контроллеру. В нём на экране появляется сообщение о том, чтобы отсканировать ШК товара.(рис. 3)

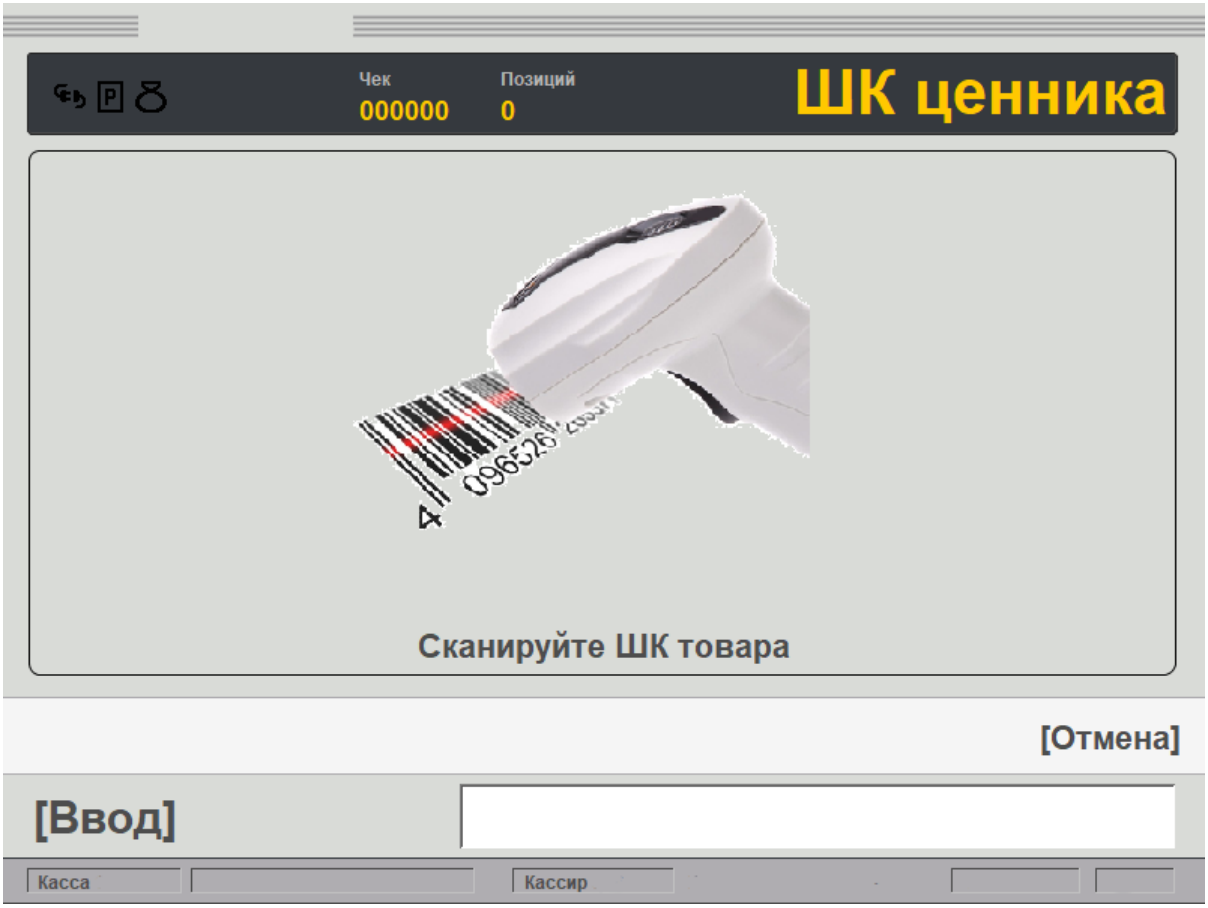

Рисунок 3 – ЭФ с сообщением отсканировать ШК товара

Далее сканируется ШК стандарта `EAN8` (восьмизначное число) [8], в котором хранится только EAN товара.

2.4.5 PriceTagSellingCommandController

После того, как ценник и товар были просканированы, осталось зарегистрировать товар в оформлении чека. Для этого используется контроллер `PriceTagSellingCommandController`.

Перед тем как добавить товар с ним происходит ряд проверок:

— Сначала проверяется находится ли товар в ассортиментной линейке, если был отсканирован соответствующий ценник.
— Далее идёт проверка для обычного ценника, где проверяется его срок годности.
— Третья проверка сверяет, что ценник не был использован больше одного раза в уже распечатанных чеках.
— И последним проверяется, что по одному ценнику можно продать до 10 товаров.

После прохождения всех проверок полученные параметры передаются функции регистрации товара, после чего товар добавляется в предварительный чек по цене указанной в ассортиментном ценнике, работа функции завершается.(рис. 4)

Рисунок 4 – Товар добавлен в предварительный чек

ЗАКЛЮЧЕНИЕ

В данной работе показана часть процесса разработки и внедрения «умных» весов и ассортиментного ценника по заказу ритейлера. Рассмотрена была разработка только со стороны кассы, разработка серверной части POS не являлась целью данной работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 How does a POS system work?. [Электронный ресурс]. — URL: https://www.mobiletransaction.org/how-pos-system-work/ (Дата обращения 10.05.2020). Загл. с экр. Яз. англ.

2 *Дейт, . .* Введение в системы баз данных / . . Дейт. — Вильямс, 2018. — P. 1328.

3 *Sun Microsystems, I.* Java for Retail POS / I. Sun Microsystems. — 1999. — P. 634.

4 UnifiedPOS. [Электронный ресурс]. — URL: https://www.omg.org/retail/unified-pos.htm (Дата обращения 10.05.2020). Загл. с экр. Яз. англ.

5 SOLID — принципы объектно-ориентированного программирования. [Электронный ресурс]. — URL: https://web-creator.ru/articles/solid (Дата обращения 10.05.2020). Загл. с экр. Яз. рус.

6 Руководство по Spring. Введение в Bean-ы.. [Электронный ресурс]. — URL: https://proselyte.net/tutorials/spring-tutorial-full-version/introduction-into-beans/ (Дата обращения 10.05.2020). Загл. с экр. Яз. рус.

7 Штрих-код Code 128. [Электронный ресурс]. — URL: https://www.labeljoy.com/ru/podderzhivat/tipy-shtrikh-kodov/code-128-ru/ (Дата обращения 10.05.2020). Загл. с экр. Яз. рус.

8 Штрих-код EAN 8. [Электронный ресурс]. — URL: https://www.labeljoy.com/ru/podderzhivat/tipy-shtrikh-kodov/ean-8-ru/ (Дата обращения 10.05.2020). Загл. с экр. Яз. рус.