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

РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С МАРКИРОВАННОЙ ПРОДУКЦИЕЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Кислициной Марии Олеговны

Научный руководитель
доцент, к. ф.-м. н. _____ М. И. Сафрончик

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2026

ВВЕДЕНИЕ

Внедрение системы обязательной цифровой маркировки товаров стало одним из ключевых изменений в российской розничной торговле последних лет. К 2026 году обязательной маркировке подлежит тридцать одна товарная группа, еще шестнадцать находятся на стадии пилотных проектов. Обеспечение прослеживаемости продукции от производителя до конечного потребителя требует от торговых сетей не только соблюдения нормативных требований, но и выстраивания внутренних процессов контроля: проверки корректности фискальных документов, сверки кодов маркировки при приемке и инвентаризации, оперативного обнаружения ошибок в данных.

С переходом на онлайн-кассы данные о каждой розничной продаже передаются в Федеральную налоговую службу в реальном времени через оператора фискальных данных, и любая ошибка в кассовом чеке немедленно становится доступна налоговым органам. При этом средства личного кабинета ОФД ограничены в возможностях поиска и фильтрации, а автоматическое информирование об ошибках отсутствует — на практике бухгалтер обнаруживает расхождения спустя дни или недели, когда ошибка уже успела размножиться.

Одновременно с контролем фискальных данных участники оборота решают задачи физической работы с товаром: приемка, инвентаризация, подготовка к отгрузке. Каждая операция предполагает собственные правила валидации и формат результата, а работа ведется одновременно с несколькими типами кодов — DataMatrix, линейными штриховыми кодами, агрегационными кодами и штрих-кодами маркетплейсов. Универсальные приложения для терминалов сбора данных, как правило, не обеспечивают необходимой гибкости.

Существующие решения в этой области являются либо частью крупных ERP-систем с высокой стоимостью внедрения и избыточной функциональностью, либо узкоспециализированными инструментами, не обеспечивающими сквозного контроля на всех этапах товарооборота.

Это определяет актуальность разработки специализированной клиент-серверной системы, объединяющей автоматизированный контроль фискальных данных и оперативный учет маркированной продукции.

Целью работы стала разработка клиент-серверной системы для автоматизации работы с маркированной продукцией в торговой сети, охватывающей две области: оперативный контроль фискальных данных и складской учет маркиро-

ванных товаров.

Для достижения поставленной цели необходимо было решить следующие задачи:

- провести анализ предметной области, включая изучение нормативной базы обязательной маркировки, роли оператора фискальных данных и существующих процессов складского учета;
- сформировать требования к разрабатываемой системе;
- обосновать выбор языков программирования, целевых платформ, средств хранения данных и сетевого взаимодействия;
- спроектировать архитектуру системы, включающую центральный сервер, клиентское приложение бухгалтера, локальный сервер магазина и мобильное приложение для терминалов сбора данных;
- реализовать компоненты системы и обеспечить их взаимодействие.

Для реализации системы используется следующее программное обеспечение:

- язык C# и платформа .NET Framework, с графическим фреймворком Windows Forms — для настольных компонентов;
- язык Kotlin и Android SDK — для мобильного приложения;
- система управления базами данных Microsoft SQL Server LocalDB — для центрального хранилища, и SQLite — для локального хранилища мобильного приложения.

Бакалаврская работа состоит из введения, 3 разделов, заключения, списка использованных источников, тринадцати приложений и цифрового носителя. Общий объем работы — 100 страниц, из них 80 страниц — основное содержание, включая 33 рисунков и 3 таблицы, список использованных источников информации — 21 наименование.

1 Анализ и описание предметной области

1.1 Система обязательной маркировки товаров: понятия и нормативная база

Национальная система «Честный знак» (оператор — ЦРПТ) выступает оператором ГИС МТ, отвечая за генерацию, выдачу и учет кодов маркировки. Носителем кода является DataMatrix — двумерный матричный код, содержащий идентификатор применения GTIN, серийный номер и криптографический компонент, что делает невозможной подделку.

Наряду с DataMatrix в складском учете применяются линейные коды EAN-13 и Code 128, поэтому участники оборота работают одновременно с двумя типами кодов. Жизненный цикл кода регламентирован Постановлением Правительства РФ № 515. Нормативную основу также образуют ФЗ № 487-ФЗ, № 488-ФЗ, Постановления № 174 и № 860, статья 15.12 КоАП и статья 171.1 УК РФ.

1.2 Роль оператора фискальных данных и необходимость оперативного контроля корректности чеков

Вывод маркированного товара из оборота при розничной продаже осуществляется через ККТ посредством ОФД. Любая ошибка в чеке — неверная ставка НДС, отсутствие реквизита, невалидный код — немедленно фиксируется на стороне ОФД. Возможности личного кабинета ограничены (период фильтрации не более 90 дней, узкий поиск по реквизитам), расхождения между данными 1С и ОФД не сигнализируются автоматически.

Для автоматизации сверки целесообразно использовать API ОФД «Ярус», позволяющий получать список касс, перечень чеков за период, содержимое документов, отчеты о сменах. Критичными видами ошибок, требующими информирования являются: неверная ставка НДС, отсутствие обязательных реквизитов, отсутствие смен за период, ошибки подписки на ОФД.

1.3 Автоматизация процессов складского и розничного учета маркированной продукции

Автоматизация складского учета. На каждом этапе участник обязан передавать сведения в ГИС МТ, и их корректность зависит от качества первичных данных при работе с товаром. Кроме DataMatrix и линейных кодов применяются агрегационные коды (17–22 символа, групповая упаковка) и штрих-коды

маркетплейсов.

Операции различаются правилами сканирования, допустимости дублирования и составом реквизитов (проверка маркировки, подтверждение наличия, подготовка к отгрузке на маркетплейсы). Оператор должен получать немедленную обратную связь об ошибке в момент сканирования, а ТСД должен быть полноценной частью клиент-серверной архитектуры, сохраняя работоспособность в офлайн-режиме.

Так же, существует проблема визуализации, длинные коды DataMatrix (129 символов) затрудняют визуальную оценку, что требует компактного представления данных.

1.4 Формирование требований к разрабатываемой системе

ЯРус-хост работает в фоновом режиме, выполняя по расписанию загрузку из ОФД, проверку целостности и резервное копирование. Функции: два вида загрузки (ручная и автоматическая), получение документов, их сохранение с классификацией, выявление ошибок с уведомлением через мессенджер. А так же, администрирование и интеграции (HTTP-интерфейс, управление токенами, интеграция с мессенджером MAX для уведомлений и удаленных команд).

ЯРус-клиент — инструмент бухгалтера для просмотра и анализа документов с авторизацией и настройкой подключения.

ШтрихКот-хост обслуживает мобильные устройства и управляет базой марок в оперативной памяти.

Мобильное приложение поддерживает 11 режимов работы (массовое сканирование с формированием файла и режимы с обращением к серверу), обеспечивает единообразную обработку: автоклассификацию, валидацию по правилам режима, статистику, формирование имени файла, сохранение; взаимодействие с сервером и настройку через QR-код.

1.4.1 Требования к информационному обеспечению

Подсистема ЯРус требует реляционной СУБД (справочные таблицы, документы, товарные позиции, привязки, служебные таблицы) с многоуровневым контролем целостности (физическая и логическая проверка), резервным копированием по расписанию с ротацией и защищенным восстановлением. Клиент не хранит данные локально.

Подсистема ШтрихКот использует облегченные способы хранения: локальный сервер обеспечивает быстрый поиск кода марки по владельцу, за счет хэш-таблицы, мобильное приложение использует локальную СУБД с отдельными таблицами для каждого режима, сохранением между сеансами и обновлениями.

1.4.2 Требования к техническому обеспечению

Настольные приложения должны работать на Windows 7, 8, 10, 11. Требования к ресурсам: ЯРус-хост — реляционная СУБД, 8 ГБ ОЗУ, 2 ГБ диска (база 400 МБ за полтора года при 200 тыс. записей). ЯРус-клиент — 4 ГБ ОЗУ. ШтрихКот-хост — 2 ГБ ОЗУ (база 800 МБ при миллионах записей). Мобильное приложение — терминалы АТОЛ Smart.Lite и Smart.Prime, Android 7.0. Десктопные компоненты распространяются как автономные exe-файлы, мобильное — как APK без магазина приложений.

1.4.3 Требования к надежности

Допустимое время восстановления: ЯРус-хост — не более 20 минут, ЯРус-клиент — не более 1 минуты, мобильное приложение восстанавливает данные из локальной СУБД с потерей не более одного кода. Все компоненты сохраняют работоспособность при потере связи со смежными. Сетевые операции выполняются асинхронно, в защищенных блоках, с предельными таймаутами для каждого типа запроса и логированием ключевых событий.

1.4.4 Требования к защите информации

Все компоненты функционируют в корпоративной сети без внешнего доступа, поэтому защита определяется разграничением прав и предотвращением случайных деструктивных действий.

1.4.5 Требования к эргономике

Хост и клиент предоставляют единообразный интерфейс просмотра документов с каскадной фильтрацией, экспортом в Excel и статистикой выделенных ячеек.

Мобильное приложение обеспечивает мгновенную обратную связь: расположение элементов по ходу рабочего процесса, звуковую индикацию, усечение длинных кодов, обратный порядок списка, вибрацию при ошибках, контрастную схему и поддержку аппаратных клавиш сканера.

2 Выбор инструментов реализации

Готовых решений для контроля фискальных данных, удовлетворяющих требованиям, нет: у «ОФД-Я» доступен только REST API без SDK. Для складского учета рынок предлагает DataMobile, «1С:Мобильная торговля», «Магазин 15»/«Склад 15», но они ориентированы на интеграцию с 1С, требуют бессрочной лицензии на каждое устройство и ограничены параметрическими настройками.

2.1 Выбор языков программирования и целевых платформ

Для настольных компонентов выбран единый стек: C#, .NET Framework 4.7.2, Windows Forms. .NET Framework предустановлен в Windows 10/11 и устанавливается через Windows Update в Windows 7 SP1/8.1, что позволяет распространять каждое приложение единым exe без среды выполнения. .NET (Core/5+) потребовала бы отдельной установки среды или увеличения размера файла, не давая функциональных преимуществ, а .NET 8 не поддерживает Windows 7. Windows Forms обеспечивает совместимость со всеми версиями Windows (в отличие от UWP, WinUI 3, MAUI), а встроенные компоненты реализуют все требования. WPF потребовал бы MVVM, XAML и привязок данных без преимуществ для табличных задач.

Мобильное приложение разработано на Kotlin (null-safety, корутины, доступ к Jetpack). Java отвергнут из-за отсутствия null-safety, кроссплатформенные фреймворки (Xamarin, MAUI, Flutter) — поскольку портирование не планируется, а SDK сканера АТОЛ поставляется как Java-библиотека для стандартной Android-разработки.

2.2 Хранение данных

ЯРус-хост — центральное хранилище, требующее хранимых процедур, представлений, транзакций, в сочетании с автономным exe. Для этого выбрана Microsoft SQL Server LocalDB, она работает в процессе, хранит базу в mdf-файле, не требует серверной службы, поддерживает полный T-SQL.

ШтрихКот-хост загружает базу марок в ОЗУ как хеш-таблицу, с поиском O(1). Встраиваемая СУБД дала бы накладные расходы при невостребованной персистентности.

Мобильное приложение использует SQLite через SQLiteOpenHelper: имена таблиц определяются в runtime по идентификатору режима, что несовместимо с ORM (Room, Realm), требующими описания структуры классами; миграция

через onUpgrade, транзакции при импорте.

2.3 Сетевое взаимодействие

Для подсистемы ЯРус выбран HTTP без шифрования: все компоненты работают в корпоративной сети за межсетевым экраном, HTTPS потребовал бы инфраструктуры сертификатов непропорционально уровню угрозы. HTTP-сервер реализован через System.Net.HttpListener, альтернативы ASP.NET Core и gRPC избыточны. Внешние сервисы (ОФД, MAX) доступны только по HTTPS.

Для подсистемы ШтрихКот выбраны TCP-сокеты: для фиксированного набора коротких текстовых команд обвязка HTTP не востребована; каждая команда отправляется по новому соединению.

2.4 Библиотеки

Система спроектирована с минимумом зависимостей. Newtonsoft.Json — для JSON (System.Text.Json недоступен в .NET Framework). NPOI — для Excel: свободна от лицензионных ограничений (в отличие от EPPlus), не требует доппакетов (в отличие от ClosedXML), формирует файл в памяти. BCrypt.Net — для хеширования паролей. TaskScheduler Managed Wrapper — для планировщика Windows.

В мобильном приложении: AndroidX (AppCompat, Fragment), Navigation Component с DrawerLayout, Material Design Components (единообразный вид на обеих версиях ATOL OS), kotlinx.coroutines (вместо устаревшего AsyncTask).

2.5 Интеграция с сервисами API

ЯРус-хост — единственный компонент, работающий с внешними сервисами. API ОФД использует POST с JSON-телом и токеном в заголовке Ofdapitoken. Из 15 методов используются 7: TP (торговые точки и ФН), KKTShift (смены), documentsShift (документы смены), getChequeLink, getFiscalDoc v2 (с резервным documents), closeShiftReport.

API MAX использует стандартные HTTP-методы и токен бота в заголовке Authorization. Поскольку хост работает за NAT без публичного HTTPS-адреса, вместо рекомендованного webhook используется long polling (GET /updates). Доступ к боту ограничен списком авторизованных пользователей в базе, администраторы получают уведомления о новых пользователях, что исключает несанкционированное взаимодействие.

3 Реализация программной системы

Система построена по архитектуре «клиент-сервер» из двух подсистем. ШтрихКот-хост обращается к ЯРус-хосту по HTTP для поиска чеков, выступая его клиентом. ЯРус-клиент и мобильное приложение реализованы как тонкие клиенты, ШтрихКот-хост занимает промежуточное положение (сервер для терминалов, толстый клиент относительно ЯРус-хоста). Все клиентские компоненты устойчивы к временной недоступности серверов.

3.1 Центральный сервер системы — приложение «ЯРус-хост»

Центральный сервер реализует непрерывное накопление фискальных данных. Архитектура многослойная:

- слой представления (девять форм, включая MainForm, DateRangeForm, формы администрирования и просмотра);
- сервисный слой (OfdDataProcessor, AutoLoadScheduler, BackupManager, IntegrityValidator, BotHandler);
- слой данных по паттерну «репозиторий» с интерфейсами и общей фабрикой IDbConnectionFactory.

HTTP-сервер собирается отдельным классом CompositionRoot, чтобы работать независимо от состояния главной формы. Загрузка реализует каскад запросов к API ОФД. JsonProcessor извлекает реквизиты, каждый документ обрабатывается немедленно и записывается в таблицу, оригинальный JSON сохраняется в файловый архив. NewCombinationDetector отслеживает новые привязки ФН к адресам, LoadReportBuilder формирует итоговый отчет.

База (SQL Server LocalDB) спроектирована в третьей нормальной форме, повторяющиеся значения вынесены в справочные таблицы со связью через целочисленные ключи. Группы таблиц: справочные, документов (четыре типа), товарных позиций, привязок (в пределах диапазона смен, т. к. кассы перемещаются), служебные.

Созданы покрывающие индексы и представление FilterDataView для каскадной фильтрации с динамическим вычислением границ через UNION ALL. Надежность хранения обеспечивают проверка целостности (физическая через DBCC CHECKDB, логическая через IntegrityValidator), резервное копирование (BACKUP DATABASE с ротацией), защищенное восстановление и плановая очистка устаревших данных единой транзакцией.

HTTP-сервер обрабатывает методы: служебные (/status, /auth/login), справочные (/users, /fiscaldrives, /filterdata, /actualtime), поиск документов и товарных позиций (базовый и расширенный), точечный поиск (/receipt, /receipt/search для подсистемы ШтрихКот). SQL-запросы строятся классами с интерфейсом IQueryBuilder, используемыми и формами, и сетевым интерфейсом.

Интеграция с MAX разделена между BotHandler (бизнес-логика, права) и MaxApiManager (протокол). Обработка входящих сообщений это цепочка проверок: тип события, авторизация отправителя, права доступа, сопоставление с командами. Длительные операции выполняются через async/await, при сбоях БД предусмотрены повторные попытки, события пишутся в логи. Хост регистрируется в планировщике Windows при первом запуске.

3.2 Клиентское приложение «ЯРус-клиент»: интерфейс бухгалтерии

Данный компонент представляет собой рабочее место бухгалтера без собственной базы, и взаимодействует с хостом только через HTTP API. Три слоя: представление (четыре формы, блокировка повторного запуска), сервисный (ApiClient, ConfigService, ExcelExporter, FilterContext), моделей. Форма LogInForm проверяет доступность хоста фоновым таймером (6 с) и получает список пользователей.

WorkForm содержит вкладки «Документы» и «Товары» с четырьмя каскадными фильтрами. Поскольку стандартный ComboBox не поддерживает множественный выбор, реализован компонент CustomMultiSelectComboBox из стандартных TextBox, Button, Form, CheckedListBox, CheckBox. Поиск собирает параметры, сериализует в JSON, отправляет POST, ответ десериализуется в DataTable и привязывается к DataGridView.

Реализованы статистика выделенных ячеек, операции с таблицей (выделение колонки, контекстное меню, открытие деталей), отображение времени последней загрузки.

Экспорт через ExcelExporter (NPOI) асинхронный, с корректной типизацией: длинные идентификаторы сохраняются как текст во избежание экспоненциальной формы.

Все обращения к серверу обернуты в try-catch-finally с таймаутами через CancellationTokenSource; при потере связи ранее загруженные результаты остаются доступны.

3.3 Локальный компонент — приложение «ШтрихКот-хост»

Даный компонент — посредник между терминалами и ЯРус-хостом. Обмен реализован поверх TcpListener на порту 8080, каждое соединение обрабатывается параллельно (HandleClientAsync), протокол текстовый построчный (UTF-8).

База марок реализована как Dictionary<string,string>, загрузка происходит с подменой ссылки одной операцией (обслуживание не прерывается). SearchCheck формирует GET-запрос с таймаутом 10 с, ответ преобразуется в текстовый список «Поле: Значение». HttpHostMonitor проверяет /status (таймаут 6 с). FileCheckerService выполняет пакетную проверку файла марок.

Длительные операции происходят асинхронно, реализован запрет повторного запуска и сворачивание в трей через NotifyIcon.

3.4 «ШтрихКот» — приложение для мобильных терминалов сбора данных

Приложение разработано на Kotlin по схеме Single Activity, где экраны — фрагменты, навигация через Navigation Component.

Типовой экран содержит поле ввода, прокручиваемый список кодов с обратной нумерацией и поле комментария; кнопки «Сохранить файл» и «Очистить», плавающие кнопки и аппаратные клавиши (F1 — очистка поля, F2 — удаление строки).

Логика разделена между фрагментами и сервисами. Общая логика обеспечена трехуровневой иерархией абстрактных классов и паттерном Template Method: BaseFragment (жизненный цикл, отмена задач), BaseScanFragment (шаблонные методы с переопределяемыми шагами), девять конкретных фрагментов задают свойства и переопределяют отличающиеся методы.

Хранение реализовано через SQLite, каждый код пишется в базу сразу после валидации, при запуске loadExistingData восстанавливает данные. Результаты сохраняются в txt, используя MediaStore с Android 10+ (Scoped Storage) или FileOutputStream на ранних версиях. ServerRepository реализует TCP-обмен на порту 8080 с командами check, search_check, send_file, ping. Отправка файла на сервер автоматическая, после сохранения. Настройка конфигурации спроектирована через сканирование QR-кода, защищенное паролем.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была разработана клиент-серверная система для автоматизации работы с маркированной продукцией в торговой сети.

Подсистема ЯРус автоматизирует получение и анализ фискальных данных, позволяя оперативно выявлять ошибки в кассовых документах до их обнаружения налоговыми органами. Подсистема ШтрихКот обеспечивает работу с маркированной продукцией на терминалах сбора данных, поддерживая контроль кодов маркировки в ходе складских операций.

Бухгалтерия получает инструмент, восполняющий ограничения личного кабинета ОФД в части поиска и фильтрации документов, а складской персонал — мобильное приложение, поддерживающее работу с различными типами кодов и настроенное под технологические процессы конкретного магазина.

Разработанное программное обеспечение встроено в операционную деятельность торговой сети. Организационные особенности предприятия и параметры взаимодействия с внешними сервисами задаются через конфигурацию, что делает систему пригодной для внедрения в различных торговых сетях, работающих с маркированной продукцией.

Модульная архитектура оставляет возможность дальнейшего развития системы: расширения номенклатуры обрабатываемых типов кодов маркировки, интеграции с дополнительными мессенджерами и внешними сервисами, подключения новых источников фискальных данных.

В ходе работы были поставлены и решены следующие задачи:

- проведен анализ предметной области, включая изучение нормативной базы и роли оператора фискальных данных в существующих процессах складского учета;
- сформированы требования к разрабатываемой системе;
- обоснован выбор инструментов реализации;
- спроектирована архитектура системы, включающая центральный сервер, клиентское приложение бухгалтера, локальный сервер магазина и мобильное приложение для терминалов сбора данных;
- реализованы компоненты системы и обеспечено их взаимодействие по выбранным сетевым протоколам.