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

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗИРОВАННОГО УЧЁТА РАСХОДОВ С ИСПОЛЬЗОВАНИЕМ РАСПОЗНАВАНИЯ И ОБРАБОТКИ ДАННЫХ ФИСКАЛЬНЫХ ЧЕКОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета компьютерных наук и информационных технологий
Кудашевой Миланы Микаиловны

Научный руководитель
доцент, к. ф.-м. н. _____ Ю. Н. Кондратова

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

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. По мере развития цифровых технологий и повсеместного распространения электронных сервисов автоматизация обработки финансовых документов становится востребованной как для частных лиц, так и для организаций. Кассовые чеки представляют собой важный источник информации о расходах, однако их ручная обработка требует значительных временных и трудовых затрат, а также повышает вероятность ошибок.

Использование QR-кодов с реквизитами фискального документа позволяет получать официальные сведения о покупке через API Федеральной налоговой службы (ФНС), что повышает надёжность учёта расходов. Такой подход позволяет автоматически формировать структурированные записи о транзакциях и использовать их для последующего анализа личного бюджета.

Цель бакалаврской работы — разработка и реализация программного комплекса для автоматизированного извлечения и обработки данных из кассовых чеков на основе QR-кодов с последующим импортом в систему учёта расходов.

Поставленная цель определила **следующие задачи**:

1. изучить существующие аналоги приложений для автоматизированного учёта расходов и их функциональные возможности;
2. рассмотреть инструменты и технологии разработки, включая сканирование QR-кодов и работу с API ФНС;
3. спроектировать и реализовать мобильное приложение для автоматизированного учёта расходов на основе данных кассовых чеков.

Методологические основы разработки мобильного приложения для автоматизированного учёта расходов с использованием распознавания и обработки данных фискальных чеков представлены в работах А. Остроуха, Н. Сурковой, А. Кузьмина, А. Бровко, А. Ермакова, А. Груздевой, И. Бессмертного, M. Li, T. Lv, Y. Huang, В. Ткаченко.

Практическая значимость бакалаврской работы заключается в создании инструмента, который объединяет сканирование QR-кодов, обработку полученных данных и их сохранение в системе учёта расходов. Реализация такого решения позволяет сократить объём ручного ввода, повысить достоверность финансовой информации и сделать повседневный контроль личных расходов более удобным.

Краткая характеристика материалов исследования. В работе использованы официальные материалы ФНС по проверке чеков, документация средств разработки iOS (Swift, SwiftUI, Charts, Keychain), серверного стека (FastAPI, Uvicorn, Pydantic, SQLite), научные публикации по обработке текстов и распознаванию данных, открытый датасет чековых позиций dremovd и внутренние материалы проекта для обучения модели DistilBERT, а также результаты реализации и функционального тестирования программного комплекса на iOS-клиенте и сервере FastAPI.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов («Обзор технологий для создания мобильного приложения» и «Разработка и реализация мобильного приложения»), заключения, списка использованных источников и двух приложений. Общий объём работы — 57 страниц, из них 49 страниц — основное содержание (введение, разделы, заключение), 26 рисунков, 1 таблица результатов функционального тестирования; приложение А — исходный код клиента NalogRuPython, приложение Б — flash-носитель с исходным кодом проекта; список использованных источников — 25 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Обзор технологий для создания мобильного приложения» посвящён анализу предметной области и обоснованию выбора средств разработки.

Рассмотрены приложения Monefy, Drebedengi, CoinKeeper, Money Lover и Wallet. Monefy отличается простым интерфейсом: добавление траты выполняется в один-два нажатия, расходы визуализируются круговыми диаграммами, поддерживаются несколько валют и пользовательские категории, синхронизация между устройствами; большинство продвинутых функций доступно в платной версии. Drebedengi является мультиплатформенным решением с автоматическим вводом трат через СМС от банков, сканированием чеков по QR-коду, планируемыми операциями и списком покупок; отмечаются устаревший интерфейс и ограниченная визуализация. CoinKeeper использует геймифицированный ввод расходов, напоминания о регулярных платежах и семейный доступ, однако пользователи отмечают редкие обновления и проблемы синхронизации. Money Lover поддерживает несколько кошельков, лимиты, учёт кредитов, календарь операций и графики, локализацию и экспорт данных; в бесплатной версии присутствует реклама. Wallet позволяет синхронизироваться с банковскими счётами, планировать бюджет, вести долги и цели накоплений, формировать детализированные отчёты; интерфейс сложнее для новичков, а банковская автоматизация доступна по подписке.

После анализа аналогов сформулирован набор ключевых функций разрабатываемого приложения. Основной акцент сделан на автоматизацию ввода данных, удобство пользовательского сценария и наглядность представления расходов:

1. ведение расходов вручную с выбором категории, суммы, даты и валюты;
2. сканирование QR-кодов кассовых чеков с камеры и из изображений;
3. получение официальной информации по чеку через API ФНС;
4. преобразование данных чека в модель расхода с возможностью подтверждения и редактирования;
5. автоматическая категоризация позиций чека и визуальная маркировка категорий;
6. отображение расходов и доходов в виде диаграмм для анализа бюджета;
7. локальное хранение данных о расходах пользователя.

Сравнение показало, что у популярных решений в целом хорошо реализованы ручной ввод и визуализация, однако автоматизация на основе официальных данных фискальных чеков с последующей интеллектуальной обработкой позиций представлена не в полном объёме, а расширенные функции часто доступны только по подписке. Это обосновало необходимость разработки собственного приложения, объединяющего официальную проверку чеков через ФНС, нормализацию наименований товаров и автоматическую категоризацию позиций в едином пользовательском сценарии.

В научной литературе для задач обработки чековых позиций рассматриваются подходы на основе трансформерных архитектур, методы извлечения значимой информации и классификации товарных наименований, а также вопросы устойчивости моделей к зашумлённым и неоднородным данным.

Для реализации мобильного приложения выбран набор средств, обеспечивающий разработку клиентской и серверной частей в единой прикладной логике: сканирование QR-кода чека, получение данных через API ФНС, обработка и сохранение расходов. Клиентская часть разработана на языке Swift в среде Xcode с применением SwiftUI; для сетевого взаимодействия организован REST-обмен с сервером, учётные данные ФНС сохраняются в Keychain. Серверная часть реализована на Python и FastAPI с запуском через Uvicorn, схемы запросов и ответов описаны на Pydantic; выделены модули авторизации, проверки чеков через API ФНС, обработки чековых данных и работы с базой. Для хранения данных применяется SQLite, для построения графиков — фреймворк Charts. Подсистема сканирования поддерживает съёмку с камеры и выбор изображения из галереи: QR-код извлекается локально средствами iOS, при неудаче изображение передаётся на сервер для резервной обработки. Для категоризации чековых позиций применяются нормализация текстов и дообученная модель DistilBERT, интегрированная в серверный контур верификации чеков. Выбранный стек позволяет решить ключевые практические задачи проекта: разработку мобильного интерфейса, безопасное хранение учётных данных, обмен с сервером, интеграцию с API ФНС, обработку QR-кодов и хранение информации о расходах.

Выводы по первому разделу: проведённый обзор подтвердил актуальность темы, позволил сформулировать функциональные требования и определить технологический стек программного комплекса.

Второй раздел «Разработка и реализация мобильного приложения» посвящён реализации программного продукта.

Основной упор сделан на самостоятельную разработку программного комплекса: мобильного клиента, серверной части, интеграции с API ФНС, нормализации и категоризации чековых позиций, а также проведение тестирования.

Пользовательский интерфейс реализован на SwiftUI и включает экраны авторизации и регистрации, списка финансовых операций с фильтрацией и сортировкой, сканера чеков с выбором камеры или галереи, подтверждения и редактирования позиций чека, построения диаграмм статистики по периодам и категориям, а также профиля пользователя с управлением учётной записью и авторизацией в ФНС. Навигация между экранами организована в рамках единого сценария учёта расходов и позволяет последовательно пройти путь от входа в приложение до сохранения операции, полученной из чека.

Разработан программный комплекс: iOS-клиент на SwiftUI и сервер на FastAPI. Практическая задача заключалась в создании мобильного инструмента, позволяющего минимизировать ручной ввод данных о покупках и повысить достоверность пользовательской финансовой статистики. Система построена по клиент-серверной архитектуре; взаимодействие реализовано по принципу REST с использованием формата JSON: клиент направляет запрос к endpoint, сервер выполняет бизнес-логику и возвращает унифицированный ответ. Клиентская часть отвечает за интерфейс (регистрация, список операций, сканирование чеков, подтверждение данных, статистика), а серверная — за прикладной API-слой, валидацию, обработку запросов и взаимодействие с внешними сервисами. Предусмотрен двухэтапный механизм извлечения QR-кода: сначала распознавание выполняется на устройстве средствами CoreImage, а в случае неудачи подключается резервная обработка изображения на сервере. Такая архитектура разделяет ответственность между слоями системы, упрощает развитие серверной логики и делает доработку функционала клиента предсказуемой.

Модель данных включает пользователя и финансовую операцию. На сервере пользователь хранится с уникальным идентификатором, email, именем и хэшем пароля; операция содержит наименование, категорию, сумму, дату, валюту и тип (доход или расход). На клиенте применяется модель ExpenseLog, согласованная с серверной структурой. Связь организована по схеме «один пользователь — множество операций»: каждая операция привязана к владель-

цу через идентификатор пользователя, что обеспечивает фильтрацию данных и контроль доступа.

Локальное хранилище на iOS реализовано в виде JSON-файла в директории Documents; при добавлении и изменении операций данные сначала сохраняются на устройстве, а при наличии сети синхронизируются с серверным API. На сервере используется SQLite; при активной сессии клиент направляет запросы на создание, обновление и удаление записей, после чего интерфейс обновляется без задержки. Такой подход обеспечивает быстрый доступ к операциям, устойчивость к сетевым сбоям и согласованность данных между клиентской и серверной частями.

Подсистема аутентификации включает регистрацию, вход и создание сессии для работы с операциями и серверными API. Клиент отправляет JSON-запросы на маршруты регистрации и входа; сервер проверяет уникальность email, валидирует пароль (длина, наличие строчных и заглавных латинских букв и цифр, проверка на слабые комбинации) и сохраняет хэш пароля. После успешной регистрации или входа клиент получает идентификатор пользователя, сохраняет данные сессии и передаёт идентификатор в менеджер данных для синхронизации операций. ИНН и пароль для ФНС хранятся в Keychain. На экране профиля предусмотрены редактирование имени, выход из системы, удаление аккаунта и авторизация в ФНС. При выходе очищаются локальные данные профиля, сессия и учётные данные ФНС; при удалении аккаунта сервер удаляет пользователя и связанные операции (каскадное удаление). Реализованные механизмы регистрации, входа и завершения сессии обеспечивают контролируемый доступ к данным пользователя и безопасное хранение чувствительной информации.

Подсистема учёта расходов и доходов поддерживает добавление, редактирование и удаление записей вручную и на основе чека. Каждая запись хранится в виде сущности ExpenseLog с наименованием, категорией, суммой, датой, валютой и типом транзакции. Добавление вручную выполняется через форму ввода на клиенте с последующим сохранением в локальное хранилище и синхронизацией с серверным API при активной сессии. Используется единый справочник категорий (в том числе «Продукты питания», «Электроника», «Аптека», «Досуг и развлечения» для расходов и «Зарплата», «Стипендия» для доходов); детализация категорий повышает точность фильтрации и наглядность анали-

за бюджета. Реализованы фильтрация по категориям и периоду, сортировка по дате, сумме и наименованию, проверка дубликатов по сумме, времени и схожести названия. Для аналитики реализован отдельный экран статистики на базе фреймворка Charts: круговая и столбчатая диаграммы распределения расходов по категориям, сравнительный график доходов и расходов. Предусмотрена фильтрация по периодам (неделя, месяц, год, весь период, произвольный интервал). При сетевых ошибках данные сохраняются локально.

Типовой сценарий работы с чеком включает сканирование QR-строки, запрос к серверу, обращение к API ФНС, нормализацию и категоризацию позиций на сервере и переход к экрану подтверждения на клиенте.

Реализовано сканирование QR-кода с камеры и из галереи. Подсистема сканирования поддерживает два режима: съёмку с камеры и выбор изображения из галереи. Локальное распознавание выполняется с помощью CoreImage и CIDetector; при неудаче изображение отправляется на сервер, где применяются предобработка (контраст, бинаризация, масштабирование, повороты) и декодирование QR-кода. Двухэтапная схема извлечения QR-строки повышает устойчивость подсистемы при работе с размытыми и неравномерно освещёнными снимками.

Для получения данных чека через API ФНС требуется авторизация в личном кабинете: пользователь вводит ИНН и пароль на клиенте, сервер выполняет проверку и возвращает идентификатор сессии для последующих запросов. Верификация чека выполняется в два шага с получением структуры чека и списка товарных позиций. На экране подтверждения пользователь проверяет и при необходимости редактирует позиции (наименование, категорию, сумму, дату), удаляет ненужные записи; преобразование данных чека в модель расхода сочетает автоматическое извлечение с ручной проверкой. Для каждой позиции выполняется проверка на дубликаты, в журнал добавляются только уникальные операции. Взаимодействие с API ФНС сосредоточено на сервере, что изолирует клиент от прямой работы с внешним сервисом и упрощает централизованную обработку ошибок.

Текст позиций в чеках содержит сокращения и шумовые элементы. На сервере выполняется приведение к единому регистру, унификация типовых сокращений по словарю (например, «мол.» заменяется на «молоко») и удаление служебных фрагментов и артикулов; за счёт этого варианты написания одного

товара приводятся к близкой форме. Нормализация и категоризация повышают однородность чековых данных, улучшают качество аналитики расходов и сокращают объём ручных правок со стороны пользователя.

Архитектура подсистемы учёта расходов и обработки чеков включает четыре уровня: интерфейсный слой iOS, слой прикладной логики клиента, серверный API-слой и слой хранения с внешними интеграциями (SQLite и API ФНС). Между уровнями реализован обмен по REST с сериализацией в JSON; ключевые сущности клиентской части связаны принадлежностью операций пользователю, управлением состоянием экранов и обращением к серверным сервисам через сетевой слой. Для категоризации применяется дообученная модель DistilBERT: по каждому наименованию определяется категория и при необходимости подкатегория; при недоступности модели или ошибке обработки назначается категория «Прочее», чтобы не прерывать пользовательский сценарий. Пользователь может изменить автоматически назначенную категорию на экране подтверждения.

Для обучения модели категоризации сформирован набор из 10 711 обезличенных наименований товарных позиций на основе открытого датасета dremovd и внутренних материалов проекта; каждая запись включает исходное наименование, метку категории и подготовленный текст. Модель обучалась различать 25 укрупнённых товарных категорий; после предобработки текстов выборка разделена на обучающую (8 568 примеров) и валидационную (2 143 примера) части в соотношении 80/20 со стратификацией по классам. Предобработка набора данных включала приведение текста к нижнему регистру, нормализацию пробелов, удаление спецсимволов и шумовых токенов, а также контроль дисбаланса классов и ручную проверку неоднозначных записей. Дообучение проводилось на базе DistilBERT; на этапе инференса тексты токенизируются WordPiece-токенизатором. Качество оценивалось на валидации по доле верных ответов и взвешенной F1-мере; достигнуто значение $F_1^{(w)} = 0{,}827$. Предсказанные метки сопоставляются с категориями учёта расходов в приложении; сохранённая модель категорий и токенизатор подключены к серверному контуру при верификации чека: для каждой позиции чека определяется категория, после чего клиент получает подготовленные данные для экрана подтверждения.

После реализации основных модулей проведено функциональное тестирование пользовательских сценариев и проверка устойчивости к типовым ошибкам ввода и внешним сбоям. Цель тестирования — подтвердить корректную работу

клиентской и серверной частей и оценить готовность программного комплекса к практическому использованию. Тестирование выполнялось на iOS-клиенте с подключением к серверу FastAPI и базе данных SQLite; для проверки сценариев использовались тестовые учётные записи приложения, тестовая авторизация в ФНС и набор изображений чеков различного качества (чёткие снимки, фото с шумом и изображения с частично нечитаемым QR-кодом). Сводные результаты зафиксированы в таблице функционального тестирования. Проверены сценарии:

1. регистрация, вход, выход и удаление учётной записи;
2. добавление, редактирование и удаление доходных и расходных операций;
3. фильтрация, сортировка и проверка дубликатов в списке транзакций;
4. сканирование QR-кода с камеры и из галереи;
5. авторизация в личном кабинете ФНС и получение данных чека;
6. подтверждение позиций чека и сохранение в журнал операций;
7. построение статистики по периодам и категориям.

По всем перечисленным сценариям зафиксирован успешный результат: создаётся учётная запись и формируется активная сессия; изменения операций сохраняются локально и синхронизируются с API; список транзакций корректно фильтруется и сортируется; повторные записи по близким параметрам не добавляются в журнал; QR-строка извлекается локально или через серверный резервный контур; по данным ФНС возвращаются структура чека и товарные позиции с нормализацией и категоризацией; диаграммы статистики по доходам, расходам и категориям строятся корректно. Дополнительно проверены негативные сценарии: неверные учётные данные ФНС, отсутствие сети, изображения без читаемого QR-кода, неполные данные формы; приложение выдаёт понятные сообщения об ошибках и не повреждает сохранённые данные пользователя. Полученные результаты подтверждают готовность программного комплекса к практическому использованию и согласованность данных между мобильным клиентом и серверной частью.

В результате самостоятельной разработки создан мобильный инструмент, охватывающий полный цикл работы с расходами: от ручного ввода и визуализации статистики до автоматического получения данных из фискальных чеков. Реализованы подсистемы аутентификации, учёта операций, сканирования и верификации чеков, нормализации и машинной категоризации позиций, а также

локального хранения с синхронизацией. Практическая часть работы подтверждена функциональным тестированием основных и негативных сценариев на связке iOS-клиента и серверной части FastAPI.

Выводы по второму разделу: разработан и протестирован программный комплекс, автоматизирующий учёт расходов по данным фискальных чеков с интеграцией ФНС, нормализацией наименований и машинной категоризацией позиций.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был проведён обзор существующих решений в области автоматизированного учёта расходов. Удалось решить задачу создания мобильного приложения для учёта расходов на основе информации с фискальных чеков. Дополнительно были рассмотрены современные подходы к обработке чековых данных, что позволило обосновать выбор клиент-серверной архитектуры и применяемых технологий.

В итоге разработан программный комплекс, состоящий из iOS-клиента на SwiftUI и серверной части на FastAPI. Реализованы ключевые функции: регистрация и аутентификация пользователя, управление расходами и доходами (добавление, редактирование, удаление), фильтрация и сортировка операций, визуализация статистики по периодам и категориям, сканирование QR-кодов с камеры и из галереи, авторизация в личном кабинете ФНС, получение и проверка данных чека, а также подтверждение и сохранение позиций чека в журнал операций.

Кроме того, реализованы этапы предобработки чековых данных: нормализация названий товаров и автоматическая категоризация позиций с помощью модели машинного обучения. Это делает данные более структурированными и уменьшает объём ручной корректировки пользователем.

Выполненное тестирование подтвердило корректную работу основных и негативных сценариев, устойчивость системы к типичным ошибкам ввода и сетевым сбоям, а также согласованность данных между мобильным клиентом и серверной частью.

Поставленная цель работы достигнута, все задачи выполнены. Разработанное решение может стать основой для дальнейшего развития сервисов персонального финансового учёта.