

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS**

Разработка приложения для учета затрат для групповых покупок

Обучающийся / Student Сударушкин Ярослав Валентинович

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет программной инженерии и компьютерной техники

Группа/Group Р34101

Направление подготовки/ Subject area 09.03.04 Программная инженерия

Образовательная программа / Educational program Системное и прикладное программное обеспечение 2020

Язык реализации ОП / Language of the educational program Русский

Квалификация/ Degree level Бакалавр

Руководитель ВКР/ Thesis supervisor Логинов Иван Павлович, кандидат технических наук, Университет ИТМО, факультет программной инженерии и компьютерной техники, доцент (квалификационная категория "ординарный доцент")

(эл. подпись/ signature)

(Фамилия
И.О./ name
and
surname)

(эл. подпись/ signature)

(Фамилия
И.О./ name
and
surname)

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	6
СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ	8
ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ	10
ВВЕДЕНИЕ	12
1 Анализ предметной области и существующих программных средств	14
1.1 Особенности учета затрат в сфере общественного питания	14
1.2 Приложения для учета затрат	16
1.3 Методы ручного учета затрат.....	18
1.4 Анализ приложений с функцией сканирования чеков	19
1.5 Ограничения существующих решений в России	20
1.6 Заключение	20
2 Разработка программной архитектуры средств для разделения затрат.....	22
2.1 Выбор архитектурного подхода	22
2.2 Обоснование выбора телеграм-бота.....	24
2.3 Выбор облачного провайдера.....	26
2.4 Выбор языка программирования	26
2.5 Архитектура телеграм-бота	27
2.6 Взаимодействие с Yandex Cloud Functions и API Gateway	28
2.7 Заключение	28
3 Выполнение программной реализации.....	30
3.1 Изучение документации	30
3.2 Разработка фундамента для телеграм-бота.....	31
3.3 Анализ QR-кода	32
3.4 Интеграция с внешним сервисом	33
3.5 Проблемы с голосованием и переход на кнопки	34
3.6 Обеспечение доступности и эффективности	35
3.7 Заключение	36
4 Развертывание и тестирование приложения.....	37
4.1 Регистрация в Yandex Cloud	37
4.2 Настройка Yandex Cloud Functions	37

4.3 Настройка API Gateway	39
4.4 Тестирование.....	41
ЗАКЛЮЧЕНИЕ.....	44
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	46

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

API (Application Programming Interface) - Интерфейс прикладного программирования, набор инструментов и протоколов для создания программного обеспечения.

HTTP (Hypertext Transfer Protocol) - Протокол передачи гипертекста, основа для обмена данными в сети интернет.

JSON (JavaScript Object Notation) - Текстовый формат обмена данными, основанный на JavaScript.

OCR (Optical Character Recognition) - Оптическое распознавание символов, технология для преобразования различных типов документов, таких как отсканированные бумажные документы или PDF-файлы, в редактируемые и индексируемые данные.

QR (Quick Response) - Быстрый отклик, двумерный штрих-код, который может быть прочитан устройствами, оснащенными камерой.

Serverless - Архитектурный подход, при котором разработчики не управляют серверами, на которых выполняется их код.

URL (Uniform Resource Locator) - Единообразный указатель ресурса, адрес веб-ресурса в интернете.

Yandex Cloud - Облачная платформа, предоставляющая инфраструктуру и услуги для разработки и развертывания приложений.

БД (База данных) - Организованное хранилище данных, используемое для управления и хранения информации. ФНС (Федеральная налоговая служба) - Федеральный орган исполнительной власти, осуществляющий функции по контролю и надзору в области налогов и сборов в Российской Федерации.

Бот - Программное обеспечение, выполняющее автоматические задачи в интернете.

ВКР (Выпускная квалификационная работа) - это итоговый проект, выполняемый студентом в завершение обучения в высшем учебном заведении.

ФНС (Федеральная налоговая служба) - Федеральный орган исполнительной власти, осуществляющий функции по контролю и надзору в области налогов и сборов в Российской Федерации.

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

API Gateway - Сервис, который управляет приемом и обработкой API-запросов, обеспечивая взаимодействие между клиентами и backend-функциями.

Inline-кнопки - Элементы интерфейса в телеграм-ботах, позволяющие пользователю взаимодействовать с ботом, не покидая текущий чат.

Webhook - Метод, позволяющий одному приложению предоставлять данные в реальном времени другому приложению, обычно путем HTTP-запроса.

Буфер - Временное хранилище данных, используемое для временной передачи информации между процессами или устройствами.

Интерфейс пользователя - Часть программы, с которой взаимодействует пользователь, выполняя задачи в приложении.

Облако (Cloud) - Вычислительные ресурсы и услуги, предоставляемые через интернет на основе удаленных серверов.

Обработка изображений - Технологический процесс анализа и модификации изображений для улучшения их качества или извлечения информации.

Платежные документы - Документы, используемые для авторизации и управления финансами в платежных системах, необходимых для регистрации в облачных сервисах.

Распределение затрат - Процесс определения и распределения расходов между участниками групповой покупки.

Telegram-бот - Программное приложение, работающее в мессенджере Telegram и выполняющее автоматические задачи по взаимодействию с пользователями через чат.

Функция (Function) - В контексте serverless архитектуры, это самостоятельный блок кода, который выполняется в ответ на определенное событие.

Чек - Документ, подтверждающий покупку товаров или услуг, обычно содержащий перечень товаров, их стоимость и QR-код для автоматического распознавания.

ВВЕДЕНИЕ

В мире, где цифровизация проникает в каждый уголок нашей жизни, умение грамотно распоряжаться своими финансами становится критически важным навыком. В последнее время особую популярность набирают общие походы в рестораны, пабы и т.п. Это не только помогает расслабиться в конце загруженной недели, но и укрепить социальные связи между участниками. Однако, с ростом популярности такого вида покупок возникает необходимость в разработке инструментов, способных обеспечить эффективный учет затрат.

Традиционные методы учета часто не справляются с задачей обеспечения прозрачности и контроля над расходами при совместном финансировании. Бумажные записи легко теряются, а электронные таблицы могут быть слишком сложными для неподготовленного пользователя. В условиях, когда каждая копейка на счету, поиск решения становится актуальной задачей.

Среди современных технологических решений все большее распространение получают телеграм-боты. Они предоставляют удобный и доступный способ учета затрат, доступный прямо из мессенджера. Преимущества таких ботов включают в себя возможность мгновенного обмена информацией, автоматизацию расчетов и уведомления о совершенных операциях.

В итоге мы приходим к разработке приложения для учета затрат.

Целью разработки приложения является предоставить пользователям программное средство для распределения затрат при общих покупках. Задачами разработки являются:

1. Выполнить анализ предметной области и существующих программных средств.
2. Разработать программную архитектуру средств для разделения затрат.
3. Выполнить программную реализацию.
4. Выполнить развертывание и тестирование приложения.

Разработка такого приложения вносит значительный вклад в повышение финансовой грамотности пользователей и упрощение процесса учета затрат в групповых покупках. В дальнейшем планируется улучшение функционала и масштабирование приложения для охвата большего числа пользователей.

В итоге разработка и внедрение приложения для учета затрат в групповых покупках является важным шагом к упрощению управления финансами и повышению финансовой грамотности. Телеграм-боты предоставляют удобный и доступный инструмент, который может стать незаменимым помощником в современном мире.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ ПРОГРАММНЫХ СРЕДСТВ

В эпоху цифровых технологий и социальной активности, когда друзья и коллеги собираются вместе, чтобы насладиться ужином в ресторане или вечером в баре, вопрос справедливого разделения счета встает особенно остро. В наши дни, благодаря прогрессу в области мобильных приложений, мы имеем в распоряжении удобные инструменты, которые помогают не только контролировать личные финансы, но и с легкостью делить общие расходы.

Существует множество приложений, обещающих упростить процесс учета расходов. Однако не все они одинаково удобны или функциональны. Давайте рассмотрим, что уже есть на рынке, и выявим их сильные и слабые стороны, а также обозначим потенциал для дальнейшего развития.

1.1 Особенности учета затрат в сфере общественного питания

1. Общая сумма счета

В сфере общественного питания ключевым моментом является учет общей суммы счета. Она включает в себя не только стоимость заказанных блюд и напитков, но и чаевые, налоги. Это основа для последующего справедливого распределения расходов между участниками.

2. Индивидуальные затраты и разделение общих позиций

Каждый участник должен оплачивать именно то, что он заказал. Но как быть с общими позициями? Например, бутылка вина, которую делили несколько человек. Здесь важно найти баланс между справедливостью и удобством.

3. Частые изменения состава участников

Группы редко бывают статичными; люди могут присоединяться и уходить, что делает учет затрат еще более запутанным. Нужна система, которая сможет адаптироваться к таким изменениям на лету.

4. Автоматизация и удобство

Автоматизация процессов учета расходов значительно упрощает жизнь. Вместо того чтобы вручную делить счет, пользователи могут полагаться на инструменты, которые делают это за секунды.

5. Наличие QR-кода

QR-код на чеке – это не просто требование законодательства, это еще и возможность автоматизировать ввод данных о расходах, что идеально подходит для цифровых решений.

6. Точность и прозрачность;

Прозрачность процесса распределения затрат необходима для избегания конфликтов и недоразумений между участниками мероприятия.

1.1.2 Примеры ситуаций в общепите

Взглянем на типичные сценарии, с которыми сталкиваются группы людей в ресторанах и кафе, и обсудим, как телеграм-бот может упростить процесс расчетов.

1. Большая компания в ресторане

Представьте ситуацию: компания из 10 человек собирается в ресторане. В конце вечера один человек берет на себя ответственность и оплачивает весь счет. Теперь перед группой стоит задача – как справедливо разделить затраты? Ведь у каждого были свои индивидуальные заказы, плюс к этому были общие блюда и напитки.

2. Совместный ужин в кафе

Другой распространенный сценарий – группа из 5 человек ужинает в кафе. Каждый выбирает разные блюда, но при этом заказываются и общие салаты и десерт.

1.1.3 Необходимость автоматизации учета затрат в общепите

Учитывая все вышеизложенные особенности и примеры, становится очевидно, что автоматизация учета затрат в сфере общественного питания не просто желательна, она необходима. Ручное распределение затрат не только

утомительно и времязатратно, но и чревато ошибками, которые могут привести к неприятным моментам и конфликтам между участниками.

Автоматизация же позволяет:

- Сократить время на распределение затрат;
- Повысить точность и справедливость распределения;
- Снизить вероятность ошибок и конфликтов;
- Обеспечить прозрачность процесса для всех участников;
- Упростить учет дополнительных расходов и изменений в составе участников.

Один из способов автоматизации — использование данных с QR-кодов чеков от ФНС, которые теперь являются обязательными и содержат всю необходимую информацию о покупке. Это позволяет приложениям для учета затрат мгновенно получать данные о расходах, что значительно упрощает процесс ведения финансов.

1.2 Приложения для учета затрат

На сегодняшний день рынок предлагает широкий спектр приложений для управления личными и групповыми финансами. Каждое из них имеет свои уникальные функции и предназначено для решения определенных задач. Давайте рассмотрим несколько популярных приложений и выявим их основные характеристики и возможности, которые они предоставляют пользователям.

1.2.1 Банковские приложения

1. Сбербанк Онлайн

Описание: Приложение от крупнейшего российского банка предоставляет широкий спектр финансовых услуг, включая переводы средств между пользователями. Преимущества: Широкая пользовательская база, надежность и безопасность. Недостатки: Отсутствие автоматического

распределения общих затрат, что оставляет на плечах пользователей необходимость самостоятельно рассчитывать доли каждого участника.

2. Тинькофф

Описание: Приложение предлагает услуги переводов и функцию "Копилка" для совместного накопления средств.

Преимущества: Инновационные финансовые инструменты, удобство использования.

Недостатки: Как и в случае с Сбербанк Онлайн, отсутствует функция автоматического распределения затрат в ресторанах.

1.2.2 Splitwise

Описание: Это приложение позволяет создавать группы, добавлять расходы и делить их между участниками.

Преимущества: Интуитивно понятный интерфейс, поддержка различных валют, возможность учета долгов и затрат с разной пропорцией.

Недостатки: Некоторые функции доступны только в платной версии, отсутствует интеграция с чековыми системами для автоматического ввода данных.

1.2.3 Settle Up

Описание: Settle Up — это еще одно приложение для учета групповых затрат, которое синхронизирует данные между устройствами участников.

Преимущества: Доступность на различных платформах, поддержка офлайн-режима, учет разнообразных типов расходов.

Недостатки: Ограниченный функционал бесплатной версии, может быть неудобно для учета расходов в больших группах.

1.2.4 Tricount

Описание: Tricount разработан для того, чтобы сделать процесс учета расходов в группах максимально простым и непринужденным. Благодаря

удобному интерфейсу пользователи могут легко добавлять затраты и просматривать, как они распределяются между участниками.

Преимущества: Интуитивно понятный интерфейс, можно пользоваться без необходимости регистрации, поддерживает множества валют.

Недостатки: Очень ограниченный функционал в бесплатной версии и отсутствие автоматического сканирования чеков.

1.3 Методы ручного учета затрат

В эпоху цифровых технологий многие все еще придерживаются традиционных методов ручного учета затрат. Давайте рассмотрим наиболее популярные из них.

1.3.1 Google Sheets/Excel

Описание: Электронные таблицы, такие как Google Sheets и Excel, остаются одним из самых надежных и гибких инструментов для учета затрат. Они широко используются как в деловой среде, так и среди друзей для дележа расходов.

Преимущества: Возможность настройки формул, широкие возможности анализа данных, доступность и поддержка совместной работы.

Недостатки: Необходимость вручную вводить данные, что может быть трудоемким и повышать риск ошибок, особенно при работе с большими объемами информации.

1.3.2 Текстовые заметки и калькуляторы

Описание: Для более простых задач некоторые предпочитают использовать текстовые заметки на своих устройствах или бумаге, в сочетании с калькуляторами для подсчета затрат.

Преимущества: Простота и доступность, нет необходимости в специальных навыках или знаниях.

Недостатки: Сложность учета при большом количестве затрат, высокая вероятность ошибок, отсутствие функций автоматического распределения и сохранения истории расходов.

1.4 Анализ приложений с функцией сканирования чеков

Сканирование и автоматическое распознавание содержимого чеков — это инновационные функции, которые могут существенно упростить процесс учета затрат. Рассмотрим несколько приложений, предоставляющих такие возможности.

1.4.1 Expensify

Описание: Expensify — это приложение, созданное для облегчения процесса отчетности о расходах как для личного, так и для делового использования. Оно позволяет пользователям сканировать чеки и автоматически извлекать данные о покупках, что упрощает ведение финансовой отчетности.

Преимущества: Приложение отличается высокой точностью распознавания информации с чеков и возможностью интегрироваться с различными бухгалтерскими и финансовыми системами, что делает его удобным инструментом для бизнеса.

Недостатки: Подписка на Expensify может быть довольно дорогой, что делает его менее доступным для индивидуальных пользователей или небольших групп. Кроме того, настройка приложения для групповых затрат может потребовать определенных усилий и времени.

1.4.2 Receipt Bank

Описание: Receipt Bank предлагает бизнес-решения для управления квитанциями и счетами. Пользователи могут сканировать чеки и получать подробные отчеты о расходах.

Преимущества: Точность распознавания, поддержка различных форматов квитанций.

Недостатки: Высокая стоимость, ориентированность на корпоративных клиентов, а не на личное использование.

1.5 Ограничения существующих решений в России

Из-за политической и экономической ситуации многие международные приложения могут быть недоступны или ограничены в России. Это включает в себя:

Недоступность приложений в App Store и Google Play: Некоторые популярные международные приложения могут быть удалены из российских магазинов приложений, что затрудняет их установку и использование.

Ограниченная функциональность: Из-за санкций и ограничений, функциональность некоторых приложений может быть ограничена на территории России.

Вопросы конфиденциальности: Пользователи в России могут беспокоиться о безопасности своих данных при использовании международных сервисов.

1.6 Заключение

Анализ существующих решений показывает, что, несмотря на разнообразие доступных приложений и методов учета затрат, многие из них имеют свои недостатки и ограничения. Существующие приложения часто требуют ручного ввода данных, что увеличивает вероятность ошибок и усложняет процесс учета, особенно в больших группах. Кроме того, международные приложения могут быть недоступны или ограничены в России, что создает дополнительную потребность в локализованных решениях. Разработка телеграм-бота для учета затрат на групповые покупки, который использует возможность сканирования QR-кодов чеков и автоматического распределения затрат среди участников, может значительно облегчить этот процесс. Такой бот сможет не только упростить ввод данных,

но и предоставить удобный интерфейс для взаимодействия участников группы, делая процесс учета более прозрачным и эффективным.

2 РАЗРАБОТКА ПРОГРАММНОЙ АРХИТЕКТУРЫ СРЕДСТВ ДЛЯ РАЗДЕЛЕНИЯ ЗАТРАТ

Разработка архитектуры является важнейшим этапом создания приложения, определяющим успех проекта. В этой главе рассматриваются выбор архитектурного подхода, используемые технологии и инструменты, которые будут применены для создания телеграм-бота, предназначенного для учета затрат на групповые покупки. Особое внимание уделено использованию serverless-архитектуры, размещению в облачном сервисе Yandex Cloud Functions, а также выбору языка программирования Python и специализированных библиотек для распознавания QR-кодов и работы с API стороннего сервиса ФНС России proverkacheka.com.

2.1 Выбор архитектурного подхода

При разработке приложения для учета затрат при групповых покупках очень важно рассмотреть парочку возможных вариантов архитектурного подхода, таких как создание мобильного приложения, веб-приложения или телеграм-бота. Каждый из них имеет свои преимущества и недостатки, которые по своему влияют на принятие окончательного решения.

2.1.1 Мобильное приложение

Плюсы:

- Доступ к устройствам: Возможность использовать функции мобильных устройств, такие как камера, для сканирования QR-кодов.
- Офлайн – не помеха: Можно дополнять информацию даже при отсутствии интернета.
- Интерфейс: Возможность создать более сложный и интуитивно понятный интерфейс.

Минусы:

- Сложность разработки: Необходимо разрабатывать и поддерживать версии для различных операционных систем (iOS, Android).
- Установка: Нужно убедить пользователя установить приложение на свое устройство
- Обновления: Вместе с установкой пользователю необходимо ещё следить за обновлением и исправлением багов приложения.
- Затраты: Разработка и поддержка мобильного приложения требуют значительных ресурсов.

2.1.2 Веб-приложение

Плюсы:

- Кроссплатформенность: Доступно на любом устройстве с веб-браузером, независимо от операционной системы.
- Не требует установки: Пользователям не нужно устанавливать приложение, достаточно перейти по ссылке.
- Не требует обновлений: Аналогично пользователю не надо следить за обновлениями. Сайт уже будет новым.

Минусы:

- Только онлайн: Веб-приложению обязательно необходимо постоянное подключение к интернету.
- Ограничения: Ограниченный доступ к функциям устройства, таким как камера, для сканирования QR-кодов. Необходимо сразу присылать фотографию.
- Интерфейс: Меньшая гибкость в создании пользовательского интерфейса по сравнению с мобильными приложениями.

2.1.3 Телеграм-бот

Плюсы:

- Удобство доступа: Пользователи могут взаимодействовать с ботом напрямую из группового чата в Telegram, где обсуждаются мероприятия и расходы.
- Не требует установки: Пользователям не нужно устанавливать дополнительное приложение помимо Telegram, который большинство и так используют.
- Легкость в развертывании и поддержке: Быстрая интеграция с API Telegram и простота в обновлении функционала благодаря обширным готовым библиотекам.
- Serverless архитектура: Если последовать принципу отсутствия сервера, то на облачных сервисах, таких как Yandex Cloud Functions, можно снизить затраты на обслуживание.
- Интерактивные элементы: Возможность использовать голосование и другие интерактивные элементы для удобного взаимодействия с пользователями.

Минусы:

- Зависимость от платформы: Бот полностью зависит от стабильности и доступности платформы Telegram.
- Ограниченный интерфейс: Большая часть пользовательского интерфейса настраивается именно в приложении Telegram
- Безопасность: Возможны риски безопасности, связанные с передачей данных через мессенджер.

2.2 Обоснование выбора телеграм-бота

На основании анализа плюсов и минусов различных архитектурных подходов было принято решение о разработке телеграм-бота по следующим причинам:

1. Удобство доступа из группового чата:

Участники мероприятий часто общаются в групповых чатах в Telegram, и возможность взаимодействовать с ботом непосредственно из чата значительно упрощает процесс учета затрат.

2. Отсутствие необходимости установки:

Пользователи уже общаются через Телеграмм и добавление бота в чат не требует ничего докачивать

3. Доступность:

Бот развернутый на облачных сервисах доступен 24/7 из любой точки с интернетом, обеспечивая постоянный доступ к функционалу.

4. Serverless подход:

Можно развернуть бота в облачных сервисах, без дальнейшего администрирования сервера. А обращение при помощи Telegram API будет запускать код обработки запросов. Отсутствие необходимости поддерживать постоянно активным какой-то намного уменьшает стоимость обслуживания такого приложения.

5. Интерактивные элементы:

Библиотеки для телеграм-ботов имеют огромный функционал, позволяя создавать в чате подобие календаря, еженедельные опросы и т.п..

6. Легкость интеграции:

Простота интеграции с API Telegram и сторонними сервисами, такими как proverkacheka.com, позволяет быстро и эффективно реализовать необходимый функционал.

Таким образом, создание помощника пользователям для учета затрат при групповых покупках в виде телеграм-бота является оптимальным решением, учитывая удобство доступа, минимальные затраты на обслуживание и возможность использования интерактивных элементов для управления списком продуктов.

2.3 Выбор облачного провайдера

В нынешних условиях санкций и ограничений направленных на Россию, выбор облачного провайдера, предоставляющего стабильные и доступные услуги в России, является критически важным. В итоге выбран был Yandex Cloud по нескольким причинам:

1. Доступность и локализация:

Яндекс — это российская компания, что гарантирует стабильную работу сервиса без риска быть заблокированным или ограниченным из-за международных санкций.

2. Поддержка serverless-архитектуры:

Yandex Cloud Functions предоставляет все необходимые инструменты для разработки и управления бессерверными приложениями, что делает его идеальной платформой для реализации данного проекта.

3. Экономическая выгода:

Бесплатный лимит в 1 000 000 запросов в месяц для Functions и 100 000 запросов для API Gateway позволяют существенно сократить расходы на обслуживание, что особенно важно для этого небольшого проекта.

4. Доступность документации:

Вся документация для сервисов Yandex Cloud написана на русском языке [1]. Также есть примеры, помогающие разобраться в реализации телеграм-бота основываясь на Serverless и минимальной стоимости [2] и [3].

2.4 Выбор языка программирования

Для разработки телеграм-бота был выбран язык программирования Python. Основными причинами выбора Python были:

1. Широкое сообщество и множество библиотек:

Python является одним из самых популярных языков программирования с большим сообществом разработчиков, что обеспечивает доступ к обширной экосистеме библиотек и инструментов.

2. Поддержка библиотек для телеграм-ботов:

Библиотека `pyTelegramBotAPI` позволяет легко интегрировать бота с API Telegram, что существенно упрощает процесс разработки и ускоряет реализацию функционала.

3. Удобство работы с API:

Python имеет удобные и эффективные средства для работы с API, что важно для взаимодействия со сторонним сервисом `proverkacheka.com`.

4. Мощные инструменты для обработки изображений:

Есть несколько библиотек помогающих обрабатывать изображения, но впереди всех является библиотека от OpenCV. Она отлично справляется с поиском qr-кода на фотографии и его декодирования. Есть и другие возможности редактирования изображения.

2.5 Архитектура телеграм-бота

Архитектура разрабатываемого телеграм-бота должна выполнять следующие функции:

1. Интерфейс пользователя в Telegram:

Пользователи взаимодействуют с ботом через Telegram, запрашивает список команд, отправляет фотографии чеков и получает информацию о затратах.

2. Обработка изображений и распознавание qr-кодов:

Изображения чека с qr-кодом обрабатываются при помощи декодера от OpenCV, чтобы найти qr-код и декодировать его в информацию нужную стороннему сервису.

3. Запрос к стороннему сервису:

После распознавания QR-кода бот отправляет запрос из данными из чека к API сервиса `proverkacheka.com` для получения списка товаров и их стоимости.

4. Обработка и отображение данных:

Данные, полученные со стороннего сервиса обрабатываются в человекочитаемый список продуктов и передается пользователям. Также у

пользователей должен быть интерфейс чтобы взаимодействовать с этим списком продуктов.

2.6 Взаимодействие с Yandex Cloud Functions и API Gateway

Для реализации взаимодействия между Telegram и функциями в Yandex Cloud Functions используется следующая схема:

1. Webhook для Telegram:

Webhook перенаправляет сообщения из Telegram на адрес API Gateway, который заранее настроен и известен. Далее сообщение уже идет туда, как настроен API. Этот подход позволяет быстро и эффективно обрабатывать запросы пользователей, минимизируя задержки и улучшая производительность.

2. API Gateway:

В настройках API Gateway указывается идентификатор функции, которой будет передаваться сообщение. Далее в Yandex Cloud Functions уже идет вся работа с сообщением. Если сообщение не дошло до функции, то оно некоторое время будет пытаться по новой дойти, постепенно увеличивая время между попытками.

2.7 Заключение

В данной главе рассмотрены возможные варианты выбора архитектуры приложения и обоснован выбор в пользу телеграм-бота. Возможность реализовать serverless-архитектуру на облачных сервисах Yandex Cloud позволяет минимизировать стоимость обслуживания приложения, его масштабируемость и удобство использования. Язык программирования Python выбран за его популярность, доступность библиотек и удобство работы с API. Использование телеграм-бота вместо других вариантов позволяет снизить затраты и повысить удобство для пользователей, что делает его оптимальным решением для данного проекта.

В итоге схема архитектуры изображена на Рисунок 1

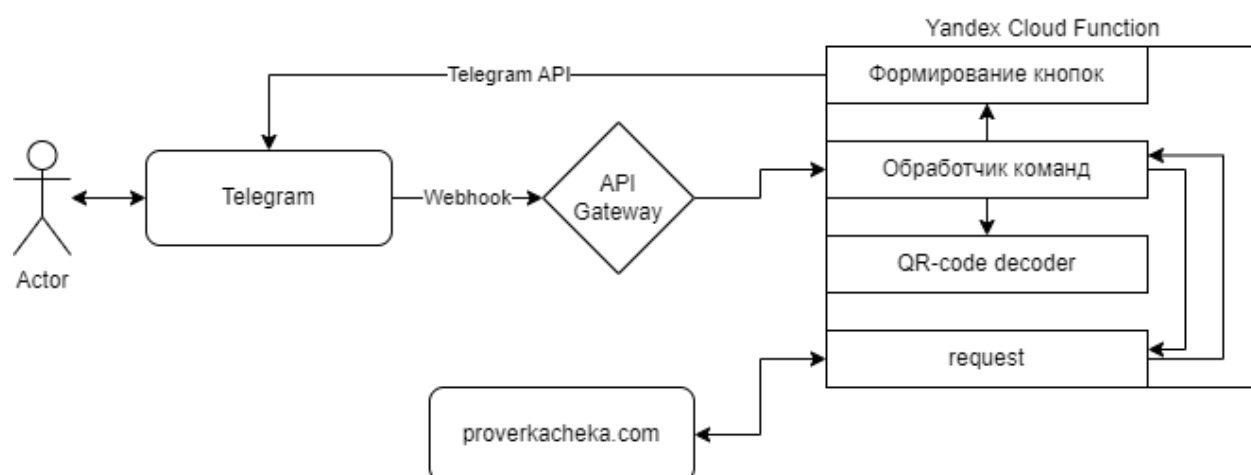


Рисунок 1 – Архитектура разрабатываемое приложения

3 ВЫПОЛНЕНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

Разработка приложения для учета затрат при групповых покупках в виде телеграм-бота включала несколько этапов, начиная с изучения необходимой документации и заканчивая интеграцией различных компонентов в единое целое. Основной целью было создать функционального телеграм-бота, который бы удовлетворял ранее описанным требованиям. Его функционалом должно быть распознавание данных с чеков, обработка полученного списка продуктов и предоставление удобного интерфейса для управления полученными данными. В данной главе описаны шаги, предпринятые для реализации проекта до разворота его на облачных сервисах, а также сложности и решения, с которыми пришлось столкнуться в процессе разработки.

3.1 Изучение документации

Любая разработка начинается с изучения документации используемых инструментов:

`pyTelegramBotAPI`:

- Просмотр официальной документации библиотеки `pyTelegramBotAPI` [3] для понимания основ создания и настройки телеграм-бота, а также для исправления ошибок по мере их появления.
- Также в случае проблем с Python стоит освежить знания с документацией по Python [4].
- Прочтение нескольких статей на Хабре по созданию телеграм-ботов с использованием данной библиотеки, чтобы понять лучшие практики и распространенные ошибки [5] [6].

`OpenCV`:

- Изучение примеров использования библиотеки `OpenCV` (в рамках Python называющиеся `opencv-python`) для распознавания и обработки изображений, особенно в контексте сканирования и распознавания QR-кодов.

- Объединение с телеграм-ботом: Исследование способов интеграции OpenCV с телеграм-ботом для автоматического извлечения данных с изображений чеков.

Proverkacheka.com:

- Подробное изучение документации стороннего сервиса от ФНС России proverkacheka.com, включая примеры запросов на Python для получения данных о чеках [7].
- Изучение в каком формате передается данные. А также эксперименты с ними.

3.2 Разработка фундамента для телеграм-бота

Разработка телеграм-бота началась с создания простого эхо-бота – бот который просто отвечает приветствием на сообщение пользователя.

Первым идет действие, с которого начинается разработка каждого телеграм-бота: Регистрация бота в Telegram. Создаем нового бота через BotFather, и получаем токен для доступа к API. Бот был назван @BillDivider_Bot.

Далее настройка pyTelegramBotAPI: Инициализация бота и настройка основных обработчиков сообщений.

Благодаря краткости Python и наличия готовой библиотеки, такой бот пишется в пару строк.

Ещё добавлен обработчик сообщений, принимающий фотографии. Это и будет будущий прием изображений с qr-кодом. Код на Python представлен в Листинг 1

```

@bot.message_handler(commands=['help', 'start'])
def say_hello(message):
    bot.send_message(message.chat.id, 'Привет, я запущен локально')

@bot.message_handler(func=lambda message: True, content_types=["photo"])
def scan_qr_code(message):
    file_path = bot.get_file(message.photo[-1].file_id).file_path
    file = bot.download_file(file_path)
    result_code, has_qr = scanner_cv2(file)

```

Листинг 1 – Телеграм-бот отвечающий на начало работы

3.3 Анализ QR-кода

В итоге во втором обработчике сообщений мы получаем фотографию с qr-кодом. Одной из ключевых задач в разработке приложения для учета затрат при групповых покупках было создание механизма автоматического считывания и анализа QR-кодов с фотографий чеков. Это даст пользователем возможность забыть про необходимость заполнять скучные гугл таблицы для таких расчетов.

Поскольку мы не хотим сохранять файл на сервере, данные изображения будут обрабатываться напрямую из буфера. Для преобразования байтового массива в массив данных используя библиотека Numpy.

Библиотека OpenCV включает класс QRCodeDetector, который предназначен для обнаружения и декодирования QR-кодов в изображениях.

Метод QRCodeDetector.detectAndDecode возвращает декодированные данные QR-кода и координаты его углов. В итоге мы получаем всю необходимую информацию для следующего этапа. Эта часть кода представлена в Листинг 2.

```
def scanner_qr_cv2(name_file: bytes):  
    nparr = np.frombuffer(name_file, np.uint8)  
    image = cv2.imdecode(nparr, cv2.IMREAD_UNCHANGED)  
    # Инициализируем детектор QRCode cv2  
    detector = cv2.QRCodeDetector()  
    # Стараемся обнаружить и декодировать  
    data, bbox, straight = detector.detectAndDecode(image)  
    f = (bbox is not None)  
    return data, f
```

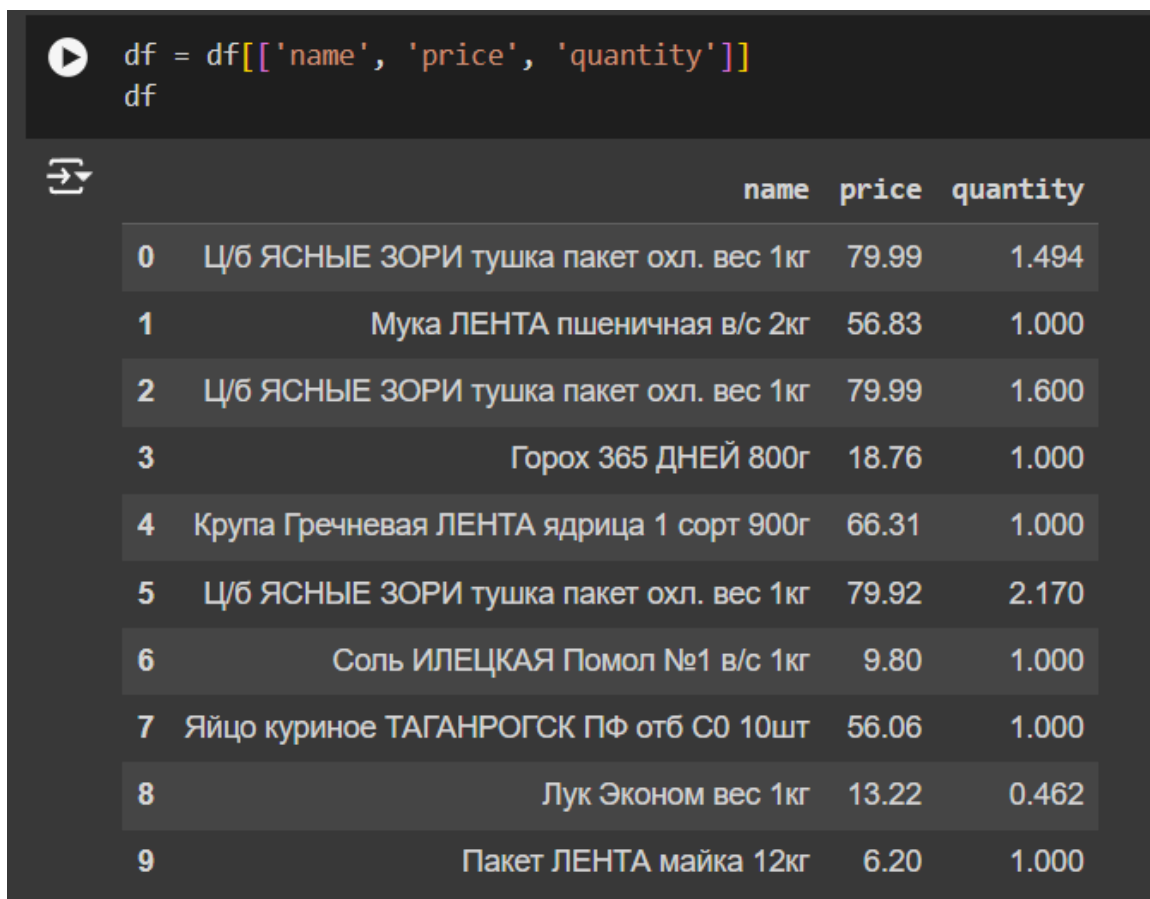
Листинг 2 –Получение данных из qr-кода

3.4 Интеграция с внешним сервисом

Регистрация на сайте proverkacheka.com: Нужно зарегистрироваться указав свою почту. После этого можно получить токен, чтобы использовать его при запросе.

Реализация запросов: Использование примеров из документации [7] для написания запросов к API и обработки полученных данных.

В ответ приходит огромное количество данных о магазине, продавце и т.д. Главное было выделить список товаров, отформатировать их при помощи средств библиотеки Numpy [8]. В итоге из огромного JSON получается миниатюрная табличка, которую можно преобразовать в список, который выглядит как Рисунок 2.



```
df = df[['name', 'price', 'quantity']]
df
```

	name	price	quantity
0	Ц/б ЯСНЫЕ ЗОРИ тушка пакет охл. вес 1кг	79.99	1.494
1	Мука ЛЕНТА пшеничная в/с 2кг	56.83	1.000
2	Ц/б ЯСНЫЕ ЗОРИ тушка пакет охл. вес 1кг	79.99	1.600
3	Горох 365 ДНЕЙ 800г	18.76	1.000
4	Крупа Гречневая ЛЕНТА ядрица 1 сорт 900г	66.31	1.000
5	Ц/б ЯСНЫЕ ЗОРИ тушка пакет охл. вес 1кг	79.92	2.170
6	Соль ИЛЕЦКАЯ Помол №1 в/с 1кг	9.80	1.000
7	Яйцо куриное ТАГАНРОГСК ПФ отб С0 10шт	56.06	1.000
8	Лук Эконом вес 1кг	13.22	0.462
9	Пакет ЛЕНТА майка 12кг	6.20	1.000

Рисунок 2 – Пример DataFrame с продуктами из чека

3.5 Проблемы с голосованием и переход на кнопки

Изначально планировалось реализовать работу с готовым списком продуктов через механизм голосования в Telegram. То есть полученный список отправлялся пользователям в виде голосования, которое люди голосовали, а когда все проголосовали, то завершали голосование и выдавалось сообщение о насчитанной сумме. Однако, возникла проблема: библиотека `pyTelegramBotAPI` не предоставляла удобного способа получения информации о том, кто именно за что проголосовал.

Наткнувшись на хорошую статью о реализации кнопок в телеграм-ботах [9], я решил изменить подход и реализовать интерфейс через `inline`-кнопки. Пример кнопок на Рисунок 3. В итоге было сделано:

- Созданы `inline`-кнопки для каждого продукта в списке.



Рисунок 3 – Пример inline-кнопок в Telegram

- Настроена логика, позволяющей пользователям нажимать кнопки для выбора продуктов, добавлять, убирать себя из списка.
- Сделана обработка нажатий кнопок и обновление списка продуктов в сообщении. А точнее, случай, когда человек пытается убрать себя из списка, в котором его нет. Ещё нельзя было выбрать количество продуктов больше, чем указано в чеке.

Кнопка «Просуммировать» считывает результаты:

- Суммирует затраты и выдает сообщение о сумме каждого человека.
- Также тут потребовалось несколько пользователей, чтобы проверить корректно ли работает в общем чате. Для корректной работы с несколькими пользователями я использовал дополнительный аккаунт Телеграм.

3.6 Обеспечение доступности и эффективности

Одной из основных достоинств данного приложения в решении задачи – это доступность его в групповом чате, в котором сразу обсуждается прошедшие события и можно попросить людей перекинуть потраченные деньги. Бота можно без проблем добавлять в чат. К тому же для людей беспокоящихся о приватности - данных бот считывает только сообщения, которые начинаются с /, то есть команды.

Тот факт, что бот считывает только команды также означает, что лишние сообщения не будут направлены телеграм-боту. В итоге не будет лишнего шума для обработчиков сообщений.

Если в чате несколько телеграм-ботов, то направить команду конкретно определенному боту можно указав после команды @<название_бота>

3.7 Заключение

Разработка телеграм-бота для учета затрат на групповые покупки прошла через несколько ключевых этапов, начиная с локальной разработки и заканчивая развертыванием в облаке. Основные сложности были успешно решены благодаря перечитыванию документации, а также использованию ещё больше возможностей библиотек. В итоге бот предоставляет удобный и экономически эффективный способ учета затрат, доступный для использования в группах и чатах.

4 РАЗВЕРТЫВАНИЕ И ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Развертывание и тестирование телеграм-бота для учета затрат при групповых покупках представляет собой важный этап в разработке, обеспечивающий его доступность и стабильность работы в реальных условиях. Этот процесс включает настройку облачной инфраструктуры, загрузку кода и зависимостей, а также проверку работоспособности бота в боевых условиях.

4.1 Регистрация в Yandex Cloud

Развертывание приложения начинается с регистрации в Yandex Cloud. Это облачный сервис, предоставляющий широкий спектр инструментов для разработки и развертывания приложений. В процессе регистрации необходимо завести платежный аккаунт. Стоит отметить, что, если у пользователя нет приветственного гранта, это может потребовать ввода небольшой суммы денег, чтобы был минимальный бюджет, однако, предполагаемая стоимость обслуживания для нашего бота составляет 0 рублей, благодаря бесплатным лимитам Yandex Cloud. Так что, ни один рубль не должен пропасть с первого взноса.

4.2 Настройка Yandex Cloud Functions

После успешной регистрации и настройки платежного аккаунта, следующим шагом является создание функции в Yandex Functions. Это serverless-решение позволяет запускать код в ответ на определенные события без необходимости управления серверами. Для создания функции сначала необходимо зайти в соответствующий раздел консоли Yandex Cloud и нажать "Создать функцию". В процессе создания функции загружаем код бота, а также файл с зависимостями, что позволит импортировать все необходимые

библиотеки, которые использовались при локальном развертывании. Их список, а также интерфейс, где можно загрузить код показаны на Рисунок 4.

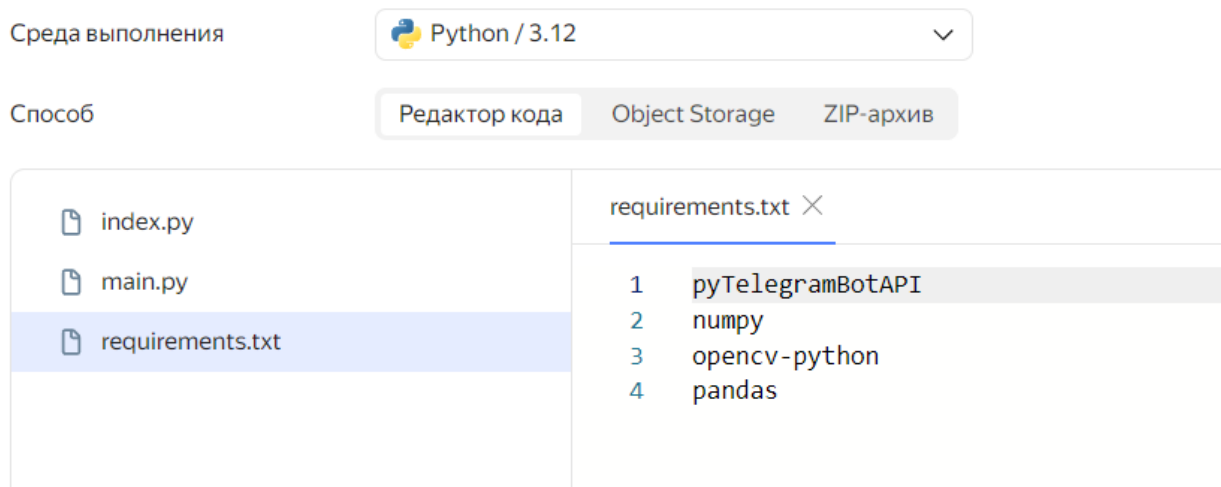


Рисунок 4 – Вкладка редактирования кода в Yandex Cloud Functions

Также не стоит забывать про необходимость прописать переменные окружения у функции. В моем случае получилось всего 3 переменные – токен Телеграма, токен сервиса proverkacheka.com, а также сам url до этого сервиса. Как это выглядит в настройках функции показано на Рисунок 5.

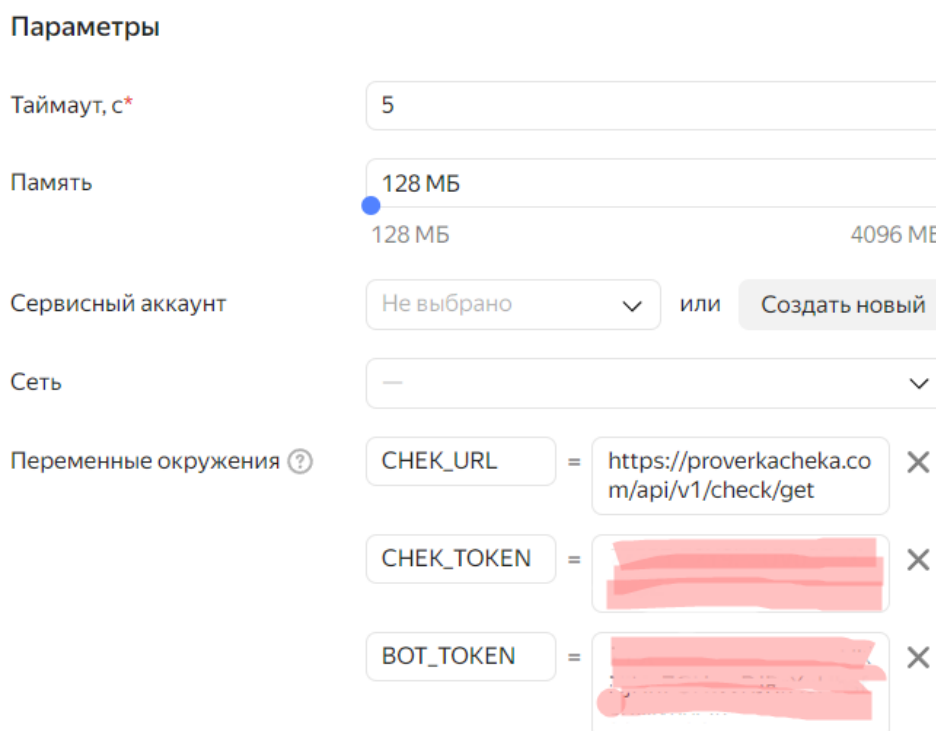


Рисунок 5 – Параметры Yandex Cloud Functions

После загрузки кода и зависимостей, функция надо сделать публичной, чтобы она могла принимать запросы извне. Для этого надо установить отвечающий за публичность ползунок в, чтобы загорелся зеленым и запомнить идентификатора функции (Function ID). Этот идентификатор позже используется при настройке API Gateway.

4.3 Настройка API Gateway

API Gateway служит точкой входа для всех HTTP-запросов, направляемых к функции. Для его создания нужно перейти в раздел API Gateway в консоли Yandex Cloud и создать новый API. В процессе создания необходимо указать идентификатор ранее созданной функции, чтобы запросы, приходящие на API Gateway, перенаправлялись на нужную функцию. В итоге его настройки на Рисунок 6

Общая информация

Идентификатор	d5dsmghsp3[REDACTED]
Статус	Active
Имя	for-python-tg-bot
Служебный домен	https://d5dsmghsp3[REDACTED].apigw.yandexcloud.net ¹
Служебный домен для WebSocket	wss://d5dsmghsp3[REDACTED].apigw.yandexcloud.net
Дата создания	15.05.2024, в 15:59

Спецификация

```
openapi: 3.0.0
info:
  title: for-python-tg-bot
  version: 1.0.0
servers:
- url: https://d5dsmghsp3[REDACTED].apigw.yandexcloud.net
paths:
  /:
    post:
      x-yc-apigateway-integration:
        type: cloud-functions
        function_id: d4em23t0dhrgc2[REDACTED]2
        operationId: tg-webhook-function
```

Рисунок 6 – Параметры API Gateway

Чтобы телеграм-бот мог получать сообщения от пользователей, необходимо настроить webhook. Webhook перенаправляет сообщения от Телеграм к API Gateway. Для этого был написан небольшой скрипт на Python, который изображен на . Он прокидывает webhook, указывая в нем URL API Gateway (URL находится на Рисунок 6, у цифры 1).

Это обеспечивает автоматическое перенаправление сообщений из Телеграмма на соответствующую функцию в Yandex Cloud.

```
import telebot

bot = telebot.TeleBot("<BOT_TOKEN>")

bot.remove_webhook()
bot.set_webhook("URL_API_GATEWAY")
```

Листинг 3 – Код webhook.py

4.4 Тестирование

После развертывания бота в Yandex Cloud начался этап тестирования, чтобы убедиться в его правильной работе и устойчивости. Ранее уже локально проводились небольшие проверки, где проверялись различные функции бота, включая сканирование QR-кодов, обработку данных и работу с кнопками. Эти тесты помогли идти в правильном направлении по запуску бота, вплоть до развертывания его в облаке. После развертывания проводился регресс, чтобы проверить, что ничего не изменилось по сравнению с локальным развертыванием.

Далее необходимо провести интеграционные тесты. Для этого я отправлял различные фотографии с QR-кодами и проверял корректность их обработки. Особое внимание уделялось функциональности inline-кнопок, позволяющих пользователям выбирать продукты из списка.

Чтобы убедиться в удобстве использования и стабильности работы бота, я поделился информацией о нем со своими коллегами, которые ранее использовали Google Таблицы для учета затрат. Они с воодушевлением помогли протестировать нового бота.

Во время тестирования выявились некоторые проблемы. Во-первых, мелким замечанием было отсутствие заголовка у сообщения с чеками. Без заголовка это сообщение выглядело неполноценно.

Во-вторых, возник вопрос, что делать если 1 товар был общим. Тогда нужно его делать между несколькими людьми. Но так как было ограничение по количеству выбравших этот предмет людей, то в итоге пришлось переделывать логику для кнопок и формулы расчета.

Первым делом нужно было не убирать кнопку «Добавить», даже если лимит превышен.

Далее формулы расчета. В итоге для каждого человека получается формула считалась как стоимость продукта, деленная на максимум между количеством продукта и всего выбравшим этот продукт людей. Точнее показано в Листинг 4

```
# Извлекаем пользователей и их количество

user_pattern = re.compile(r"(@?\w+)\s*\((\d+)\)")
users = user_pattern.findall(selected_users)

# Обновляем суммы для каждого пользователя
for user in users:
    username = user[0]
    user_count = int(user[1])

    user_totals[username] += (price * quantity) * (user_count /
max(quantity, total_selected))
```

Листинг 4 – Код расчета доли от стоимости продукта

К примеру, если продукт 1 и человек выбравший его будет один, то в итоге будет только стоимость продукта добавляться. Если продукт 1, один человек дважды отметился, а другой 1 раз, то стоимость продукта поделится между ними в отношении 2:1

Еще стоит обратить внимание на требования безопасности хранения данных. Так как данные о продуктах и пользователях не сохраняются ни в какой БД на стороне приложения, это минимизировало риски утечки информации и сократило расходы на поддержку базы данных. Проверка

производительности показала, что бот быстро обрабатывает запросы и корректно обновляет данные в сообщениях.

В итоге, после устранения всех выявленных проблем и успешного прохождения всех этапов тестирования, бот был признан готовым к использованию. Он работает на облачном сервисе без перебоев, предоставляя пользователям удобный инструмент для учета затрат при групповых покупках, легко интегрируется в групповые чаты и обеспечивает высокий уровень взаимодействия без необходимости установки дополнительного программного обеспечения.

ЗАКЛЮЧЕНИЕ

В рамках выполнения данной выпускной квалификационной работы была успешно разработана и внедрена система для учета затрат при групповых покупках в виде телеграм-бота. Основной целью проекта было создание удобного и эффективного инструмента, который позволял бы пользователям быстро и просто распределять расходы после совместных мероприятий, таких как походы в рестораны или бары.

В ходе работы были достигнуты следующие результаты по первоначальным задачам:

1. Анализ предметной области и существующих решений:

Были выделены основные особенности предметной области. Проверив существующие решения, получилось определить основные требования к разрабатываемому приложению и выбрать наиболее подходящие технологии.

2. Разработка программной архитектуры средств для разделения затрат:

Принято решение о создании телеграм-бота с использованием serverless-архитектуры, который будет размещен на облачных сервисах от Yandex Cloud. Этот подход обеспечил высокую доступность и масштабируемость решения при минимальных затратах на обслуживание.

Также были спроектированы основные компоненты системы, включая механизмы распознавания QR-кодов, интеграцию с внешними сервисами и интерфейс для взаимодействия с пользователями через телеграм.

3. Выполнение программной реализации:

С помощью библиотеки pyTelegramBotAPI был реализован телеграм-бот. С помощью OpenCV получилось декодировать qr-код получаем из бота. Также благодаря декодированию из qr-кода мы получаем данные о чеках через API сервиса proverkacheka.com. Была также разработана интерактивная система выбора продуктов с использованием inline-кнопок.

4. Развертывание и тестирование:

Бот был развернут на платформе Yandex Cloud, что обеспечило его круглосуточную доступность и стабильность работы. Проведено комплексное тестирование, включающее проверку всех функций и интеграций, что позволило выявить и устранить все критические ошибки.

5. Вывод

Разработанный телеграм-бот показал свою эффективность и удобство использования, что подтверждается положительными отзывами от первых пользователей. Бот значительно упрощает процесс распределения затрат и делает его более прозрачным и автоматизированным, что особенно ценно для больших групп людей.

6. Возможные перспективы улучшения:

- Добавление возможности ручного ввода списка продуктов на случай отсутствия QR-кода. Это сделает бота еще более универсальным и удобным.
- Проведение дополнительных тестов на многопоточность и улучшение алгоритмов обработки данных для повышения общей производительности системы.
- Реализация механизмов валидации данных при ручном вводе для предотвращения возможных атак и ошибок.
- Улучшение сканирования qr-кода воспользовавшись всеми возможностями OpenVC [10]

Таким образом, разработка телеграм-бота для учета затрат при групповых покупках можно считать успешным проектом, который достиг поставленных целей и имеет значительный потенциал для дальнейшего развития. Этот проект демонстрирует, как современные технологии могут помочь в создании практичных и эффективных решений, улучшающих повседневную жизнь пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация к Yandex Cloud – URL: <https://yandex.cloud/ru/docs> (дата обращения 21.05.2024)
2. Прыжок до небес: запускаем телеграм бота на Python в serverless облаке URL: <https://habr.com/ru/articles/550456/> (дата обращения 21.05.2024)
3. Документация PyTelegramBotAPI на русском языке – URL: <https://pytba.readthedocs.io/ru/latest/index.html> (дата обращения 21.05.2024)
4. Документация Python – URL: <https://www.python.org/doc/> (дата обращения 21.05.2024)
5. Самый полный стартовый гайд по ботам Telegram (python) – URL: <https://habr.com/ru/articles/697052/> (дата обращения 21.05.2024)
6. Создаём основу для диалогового Телеграм бота в облаке– URL: <https://habr.com/ru/articles/752658/> (дата обращения 21.05.2024)
7. Документация API proverkacheka.com – [Интернет ресурс] – URL: https://proverkacheka.com/files/help/documentation_api.docx (дата обращения 21.05.2024)
8. Документация Numpy – URL: <https://numpy.org/doc/> (дата обращения 21.05.2024)
9. Пагинация для telegram бота - URL: <https://habr.com/ru/articles/675404/> (дата обращения 21.05.2024)
10. QR-code. Обнаружить и расшифровать. Шаг 1 — Обнаружить – URL: <https://habr.com/ru/articles/708832/> (дата обращения 21.05.2024)