# Пояснительная записка

к проекту «Telegram-бот для учёта трат и совместного ведения бюджета.

**1. Введение**

Цель проекта:

Создать Telegram-бота, упрощающего фиксацию личных расходов и позволяющего вести общий бюджет нескольким пользователям.

Актуальность:

В современном ритме жизни люди часто не успевают записывать траты и теряют контроль над личным и семейным бюджетом. Мессенджеры доступны в любой момент, поэтому внедрение учёта расходов прямо в чат позволяет быстро и удобно фиксировать расходы

Задачи проекта

* Организовать хранение информации о тратах в реляционной базе данных.
* Реализовать возможность создания общего счёта с паролем и подключения к нему нескольких пользователей.
* Предоставить пользователю отчёты в виде графиков за заданный период.
* Обеспечить удобный пошаговый ввод данных с помощью FSM.

**2. Основная часть**

**2.1 Архитектура и основные функции**

Telegram-бот реализован на Python с использованием библиотеки aiogram 3.x.

FSM (Finite State Machine) отвечает за многошаговое взаимодействие: ввод суммы, категории, комментария.

ORM-слой на SQLAlchemy 2.x обеспечивает работу с базой: модели, сессии, транзакции.

Основные команды и сценарии:

* /start — регистрация или просто вывод списка команд.
* /add (добавить трату) — последовательный ввод данных о трате:
  + Сумма.
  + Категория (клавиатура с наиболее частыми или дефолтными).
  + Комментарий (опционально).
* /report (получить отчет) — генерация и отправка круговой диаграммы расходов за последние N дней.
* Создать новый счет— создание нового счёта (код + пароль), автоматическая привязка пользователя.
* Подключиться к существующему счету — подключение к существующему счёту по коду и паролю.
* Покинуть текущий счет — выход из текущего счёта без удаления исторических данных.
* Получить код и пароль счета – прислать еще раз код и пароль от счета в удобной форме
* /settings (Настройки) — меню, объединяющее пункты «Создать новый счет», «Подключиться к существующему счету», «Покинуть счёт», «Получить код и пароль счета».

**2.2 Структура данных и модели**

В качестве центральных сущностей проекта используются три модели:

Account

* id — уникальный идентификатор счёта.
* code — публичный код для подключения.
* password — пароль доступа.

Связь «один-ко-многим» с пользователями и тратами.

User

* id — внутренний идентификатор.
* tg\_id — Telegram-ID для привязки.
* account\_id — внешний ключ на Account, может быть NULL для личных трат.

Связь «многие-к-одному» с Account и «один-ко-многим» с Expense.

Expense

* id — уникальный идентификатор.
* user\_id, account\_id — ссылки на автора и счёт.
* amount — сумма траты.
* category — категория (строка).
* comment — дополнительный текст.
* created\_at — метка времени.

Связи между моделями описаны через ForeignKey и relationship для удобного доступа к связанным объектам в коде.

**2.3 Реализация FSM и сохранение данных**

При вызове /add бот переводит пользователя в состояние Form.amount.

В каждом последующем состоянии, через state.update\_data() накапливаются промежуточные значения.

После ввода комментария в process\_comment собираются все данные из контекста, вызывается функция add\_expense(), сохраняющая запись в базе, а затем FSM сбрасывается в главное меню.

**2.4 Генерация отчётов**

Для отчёта используется matplotlib: круговая диаграмма расходов по категориям.

Функция generate\_expense\_report(days, user\_id, session):

* Вычисляет дату начала периода.
* Фильтрует Expense по account\_id (если подключён) или по user\_id для личных трат.
* Группирует по категориям и суммирует.
* Рисует диаграмму и добавляет подпись «Всего потрачено: XXX».
* Сохраняет в BytesIO и возвращает готовый поток для отправки.

**3. Заключение**

Проект Telegram-бота для учёта трат и совместного бюджета решает задачу быстрого фиксирования расходов и визуализации статистики.

Достигнутые результаты:

* Надёжное хранение данных и ORM-модели на SQLAlchemy.
* Пошаговый ввод затрат с сохранением промежуточных данных.
* Работа с общим счётом: создание, подключение, выход без потери истории.
* Генерация наглядных диаграмм расходов.

Перспективы развития:

* Добавить учёт доходов и финансовые цели.
* Реализовать другие типы отчётов: динамика по дням, прогнозы.
* Подключить веб-панель для работы вне Telegram.
* Внедрить интеграцию с внешними сервисами (банковскими API).

Проект демонстрирует принципы асинхронного программирования в Telegram-ботах, использование FSM и ORM-моделей, а также основы визуализации данных.