**Telegram-бот «Финансовый помощник FinHelper»**

**Чевычелов Максим**

**Оглавление**

[**1. Введение** 2](#_Toc198066517)

[**2. Главное меню** 3](#_Toc198066518)

[**3. Последовательность работы с транзакциями** 4](#_Toc198066519)

[**4. Баланс и статистика** 5](#_Toc198066520)

[**4.1 Показать баланс** 5](#_Toc198066521)

[**4.2 Детальная статистика** 5](#_Toc198066522)

[**5. Экспорт данных** 6](#_Toc198066523)

[**5.1 CSV** 6](#_Toc198066524)

[**5.2 XLSX** 6](#_Toc198066525)

[**6. Визуализация** 7](#_Toc198066526)

[**7. Курсы валют** 7](#_Toc198066527)

[**8. Меню настроек** 8](#_Toc198066528)

[**8.1 Выбор валюты** 8](#_Toc198066529)

[**8.2 Управление категориями** 8](#_Toc198066530)

[**8.3 Установка бюджета** 8](#_Toc198066531)

[**9. Фоновые задачи** 9](#_Toc198066532)

[**10. Серверная часть (Flask API)** 10](#_Toc198066533)

[**11. Технологический стек** 11](#_Toc198066534)

**1. Введение**

Телеграм-бот «Финансовый помощник» создан для тех, кто хочет держать под контролем свои личные финансы без выхода из мессенджера. Он объединяет в себе:

* Быстрый ввод операций (доходы и расходы).
* Моментальный просмотр баланса и детальной статистики.
* Экспорт данных для внешней обработки.
* Гибкую настройку категорий и бюджетов.
* Автоуведомления и отчёты.

**2. Главное меню**

Клавиатура отображается при старте и после каждой завершённой операции. Состоит из 9 кнопок:

|  |  |  |
| --- | --- | --- |
| **Раздел** | **Кнопки** | **Описание** |
| Транзакции | Добавить расход Добавить доход | Запуск диалога: выбор типа → ввод суммы → выбор категории |
| Баланс и отчёты | Показать баланс Статистика | Сводка по текущему месяцу → переход в подменю статистики |
| Валюты | Курс валют | Запрос внешнего API → вывод курсов базовой валюты пользователя |
| Экспорт | Экспорт в CSV Экспорт в XLSX | Генерация файлов → отправка через Telegram |
| Визуализация | Диаграммы | Построение графиков: столбчатая диаграмма + линейный тренд |
| Настройки | Настройки | Переход в меню управления параметрами бота |

**3. Последовательность работы с транзакциями**

1. **Выбор действия**  
   • Нажмите «Добавить расход» или «Добавить доход».  
   • Бот сохраняет тип операции во внутреннем контексте.
2. **Ввод суммы**  
   • Ожидается число (целое или с десятичной частью).  
   • При вводе некорректного формата бот присылает предупреждение и запрашивает повтор.
3. **Выбор категории**  
   • Формируется кнопочная клавиатура из категорий пользователя (расход/доход).  
   • Кнопка «Отмена» возвращает в главное меню без сохранения.
4. **Фиксация операции**  
   • Если категория найдена, сохраняется запись в БД.  
   • Бот отправляет сообщение: «✅ Расход/Доход на сумму X сохранён в категории Y».  
   • Возврат в главное меню.

**Внутренние состояния** (ConversationHandler): STATE\_AMOUNT → STATE\_CATEGORY → END.

**4. Баланс и статистика**

**4.1 Показать баланс**

* Рассчитывается сумма доходов и расходов с первого числа текущего месяца.
* Выводится:
  + **Баланс** = Доходы − Расходы.
  + **Доходы** и **Расходы** по отдельности.

**4.2 Детальная статистика**

При нажатии «Статистика» открывается подменю:

|  |  |
| --- | --- |
| Кнопка | Действие |
| За день | Сумма расходов за сегодня + диаграмма по категориям (если есть траты) |
| За неделю | Линейный график баланса за последние 7 дней |
| За месяц | Общая сумма расходов за месяц + диаграмма распределения по категориям |
| Назад | Возврат в главное меню |

* **Диаграммы** строятся в utils.viz:
  + plot\_monthly\_category\_bar(data) — столбцы категорий.
  + plot\_balance\_trend(trend) — линия изменения баланса.

**5. Экспорт данных**

**5.1 CSV**

* Разделитель — ;.
* BOM-префикс для корректного отображения в Excel.
* Столбцы: ID, Сумма, Тип, Категория, Дата/Время.

**5.2 XLSX**

* Используется pandas + openpyxl.
* Лист Transactions: отформатированные колонки «Сумма» (с двумя знаками после запятой) и «Дата/Время».

После генерации файлы автоматически отправляются пользователю.

**6. Визуализация**

1. **Расходы по категориям**
   * Столбчатая диаграмма (категория vs. сумма).
   * Поворот подписей по оси X для читаемости.
2. **Тренд баланса**
   * Линейный график даты vs. суммарный баланс.
   * Автоматический формат оси дат.

Кнопка «Диаграммы» выполняет оба построения последовательно.

**7. Курсы валют**

* Запрос к внутреннему Flask API: /api/rates?base=<валюта>&symbols=….
* Основной источник — exchangerate.host, резервный — open.er-api.com.
* Вывод списка валют с курсами и датой актуализации.
* При сбое выводится сообщение об ошибке.

**8. Меню настроек**

**8.1 Выбор валюты**

* Кнопки: RUB, USD, EUR, Отмена.
* Валюта сохраняется в профиле пользователя и влияет на все операции.

**8.2 Управление категориями**

* **Добавить категорию**
  1. Ввод названия (текст).
  2. Выбор типа (Расход/Доход).
  3. Подтверждение сохранения.
* **Удалить категорию**
  1. Выбор из списка.
  2. Подтверждение удаления («Да»/«Нет»).

**8.3 Установка бюджета**

1. Выбор категории расходов.
2. Ввод суммы месячного лимита.
3. Сохранение в таблице Budget.

**Уведомления:**

* При расходе ≥80% лимита — предупреждение.
* При превышении ≥100% — тревожное сообщение.

**9. Фоновые задачи**

Используется библиотека **APScheduler** для cron-заданий:

* **send\_daily\_summary**
  + Рассылка ежедневной сводки (общие расходы) в час, заданный в DAILY\_SUMMARY\_HOUR.
* **check\_budgets**
  + Проверка всех активных бюджетов ежедневно в 09:00.
  + Отправка предупреждений при достижении порогов.

Фоновые задачи инициализируются в init\_scheduler.

**10. Серверная часть (Flask API)**

* **Эндпоинт**: /api/rates (GET).
* Параметры запроса: base (базовая валюта), symbols (список через запятую).
* Логика работы:
  1. Пробуем exchangerate.host.
  2. Если не удалось — полный список и фильтрация.
  3. Если и это не сработало — резерв open.er-api.com.
* Ответ: JSON { base, date, rates }.

**11. Технологический стек**

* Python 3.11
* Flask + SQLAlchemy
* python-telegram-bot
* APScheduler
* Pandas, openpyxl, matplotlib