МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ БИЗНЕСА

Кафедра Экономической информатики

**ОТЧЁТ О ВЫПОЛНЕНИИ РАСЧЕТНО-ГРАФИЧЕСКОГО ЗАДАНИЯ**

по дисциплине

«Разработка программных приложений»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Выполнил:** |  | **Проверил:** |  | |
| Студент | *Лапшин Е.В.* | Преподаватель | *Зайцев А.А.* | |
| Факультет | *Факультет бизнеса* |  |  | |
| Направление (специальность) подготовки | *Бизнес-информатика* | Балл: \_\_\_\_\_\_\_\_\_ | |  |
| Группа | *ФБИ-23* | Оценка *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | | |
| Шифр |  |  |  | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Подпись студента | | Подпись преподавателя | | |
| Дата сдачи: | «\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. | Дата защиты: | «\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. | |

Новосибирск 2025

Оглавление

[ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ 3](#_Toc198500693)

[Введение 4](#_Toc198500694)

[Глава 1. Теоретические основы разработки микросервисного приложения 5](#_Toc198500695)

[Глава 2. Разработка Telegram бота и Flask приложения. 7](#_Toc198500696)

[Задание 2. 7](#_Toc198500697)

[2.1 Общая часть 7](#_Toc198500698)

[2.3 Внешний сервис 15](#_Toc198500699)

[Заключение 17](#_Toc198500700)

[Список литературы 18](#_Toc198500701)

[Приложение 19](#_Toc198500702)

# **ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ**

Разработка Telegram-бота, реализующего учёт финансов пользователей. Бот должен позволять регистрироваться, добавлять доходы и расходы, выбирать способ оплаты, просматривать операции в разных валютах с использованием внешнего микросервиса.

# Введение

В данной работе реализован Telegram-бот для учета финансовых операций, который позволяет пользователю регистрироваться, добавлять операции с выбором типа, суммы, даты и способа оплаты, а также просматривать список операций с возможностью конвертации сумм в различные валюты. Для получения актуальных курсов валют используется внешний сервис, реализованный на Flask, который возвращает курс для валют USD и EUR по запросу. Бот взаимодействует с базой данных PostgreSQL для хранения информации о пользователях и операциях. Основная цель работы — продемонстрировать навыки разработки микросервисного приложения с использованием современных библиотек Python и взаимодействия с внешними API.

# Глава 1. Описание используемых технологий и инструментов

1.1 Язык программирования Python

Для разработки данного проекта использован язык программирования Python — один из самых популярных и универсальных языков программирования. Python характеризуется простым и понятным синтаксисом, обширной стандартной библиотекой и большим сообществом. Это позволяет быстро создавать как простые скрипты, так и сложные многокомпонентные системы.

1.2 Фреймворк aiogram для разработки Telegram-бота

Для создания Telegram-бота применён асинхронный фреймворк **aiogram**. Он обеспечивает удобный и современный API для работы с Telegram Bot API, поддерживает обработку сообщений, команд, клавиатур и управление состояниями пользователя через конечные автоматы (FSM — finite state machines).

Использование асинхронного подхода в aiogram позволяет эффективно обрабатывать множество входящих сообщений, не блокируя основной поток выполнения, что повышает производительность и отзывчивость бота.

1.3 Flask — легковесный веб-фреймворк

Внешний сервис, предоставляющий курсы валют, реализован с помощью микрофреймворка **Flask**. Flask позволяет быстро создавать REST API, которые принимают HTTP-запросы и возвращают ответы в формате JSON. Благодаря своей простоте и гибкости, Flask широко используется для создания небольших и средних веб-приложений и микросервисов.

1.4 База данных PostgreSQL и библиотека psycopg2

Для хранения данных проекта используется реляционная база данных **PostgreSQL**. Эта СУБД известна своей стабильностью, расширяемостью и поддержкой сложных запросов.

Взаимодействие Python-приложения с PostgreSQL организовано через библиотеку **psycopg2** — один из наиболее популярных адаптеров для работы с PostgreSQL в Python. Она позволяет выполнять SQL-запросы и получать результаты в удобном формате.

1.5 Дополнительные инструменты

**dotenv** — библиотека для управления переменными окружения, позволяющая хранить чувствительные данные (например, токен бота) вне исходного кода.

**logging** — стандартный модуль Python для организации логирования событий приложения.

# Глава 2. Разработка Telegram бота и Flask приложения.

# Задание 2.

Реализовать Telegram bot для учета финансов.

В качестве БД используется Postgres. В качестве библиотеки для интеграции с

telegram - aiogram 3.

Задание включает в себя 2 части:

1. Общая, которая включает в себя основу приложения, которая должна быть реализована вне зависимости от варианта.

2. Индивидуальная, которая включается в себя реализацию уникальной

функциональности для каждого варианта

# 2.1 Общая часть

2.1.1 База данных

Реализовал базу данных, с учетом индивидуального задания добавил в таблицу способ оплаты.

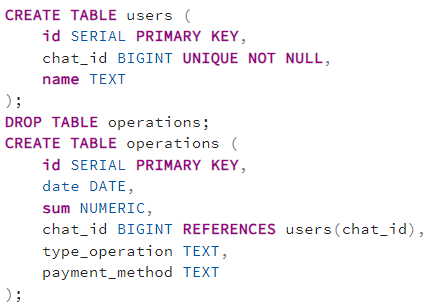


Рисунок 1 – таблицы users и operations.

2.1.2 Регистрация

Реализовал функциональность регистрации.

Регистрация осуществляется посредством отправки команды /reg. Сценарий обработки команды:

Для начала импортировал нужные библиотеки.

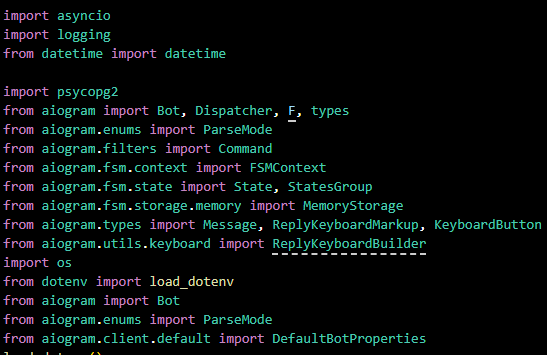


Рисунок 2 – импорт библиотек

После этого из виртуального окружения .env получил токен бота.



Рисунок 3 – получение токена

Подключился к базе данных из пункта 2.1.1



Рисунок 4 – подключение к бд

После этого создаю класс бота и создаю диспетчер.

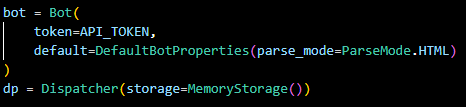


Рисунок 5 – создание бота и диспетчера.

Буду использовать машину состояний.

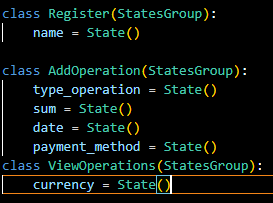


Рисунок 6 - состояния

1. Пользователь вводит команду /reg.

2. Бот проверяет, что пользователь не зарегистрирован.

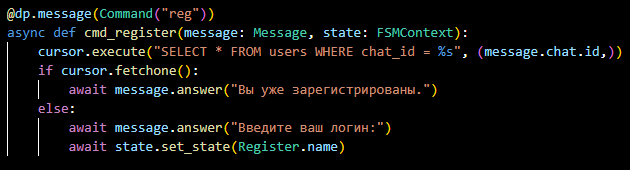


Рисунок 7 – обработчик команды /reg

3. Бот предлагает пользователю ввести логин.



Рисунок 8 – ответ бота

4. Пользователь вводит логин.

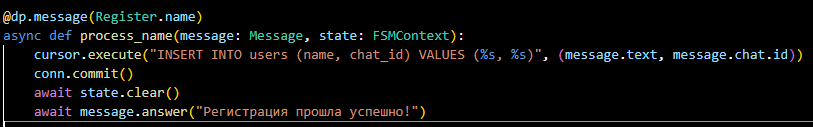
5. сохраняет логин и chat id пользователя в таблицу users. 

Рисунок 9 – добавление в таблицу users информации

6. выводит информацию пользователю о том, что он успешно зарегистрирован.



Рисунок 9 – ответ бота

2.1.3 Добавление новой операции

Реализовать функциональность добавления новой операции. Добавление операции осуществляется посредством отправки команды /add\_operation. Сценарий обработки команды:

1. Пользователь вводит команду /add\_operation.

2. Бот проверяет что пользователь зарегистрирован.

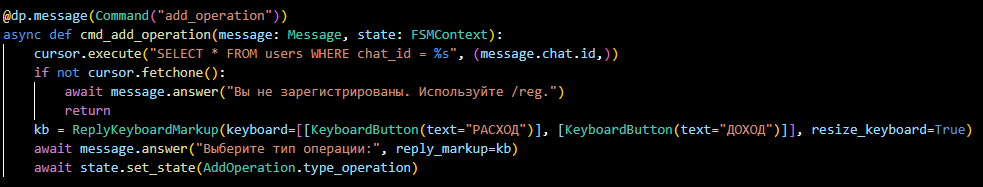


Рисунок 10 – обработчик команды /add\_operation

3. Бот выводит 2 кнопки с типом операции: РАСХОД, ДОХОД.



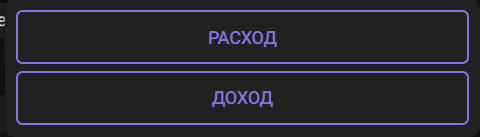


Рисунок 11-12 – выбор типа операции.

4. Пользователь выбирает тип добавляемой операции.

5. Бот предлагает ввести сумму операции в рублях.

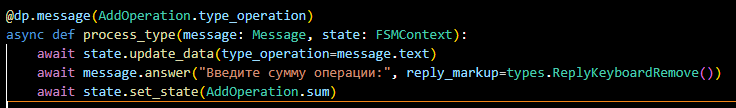


Рисунок 13 – сохранение типа операции и предложение ввести сумму

6. Пользователь вводит сумму в рублях.



Рисунок 14 – ответ бота

7. Бот предлагает указать дату операции.

Бот так же проверяет корректность введённой суммы.

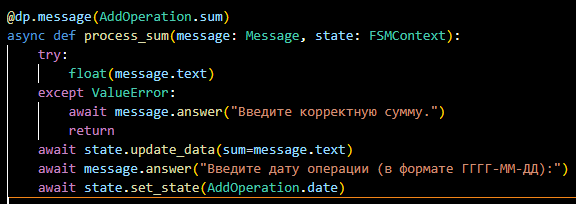


Рисунок 15 – код обработки состояния

8. Пользователь указывает дату операции.

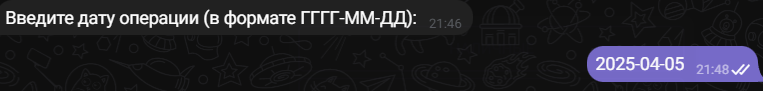


Рисунок 16 – ответ бота

9. По индивидуальному заданию номер 5 нужно узнать способ оплаты.

Реализовать функциональность способа оплаты.

Необходимо добавить в таблицу operations дополнительную колонку payment\_method, по следующему алгоритму.

2. Если пользователь нажал на кнопку ДОХОД, обработка завершается в колонку payment\_method устанавливается NULL. Если пользователь нажал на кнопку РАСХОД, то бот выводит кнопки: НАЛИЧНЫЕ, КАРТА.

3. Пользователь выбирает способ оплаты.

4. Бот сохраняет в БД (таблица operations): дату операции, сумму операции, chat id пользователя, тип операции, метод оплаты.

5. Бот сообщает пользователю о том, что операция добавлена.

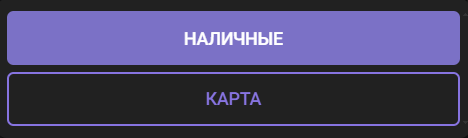




Рисунок 17-18 – тип оплаты

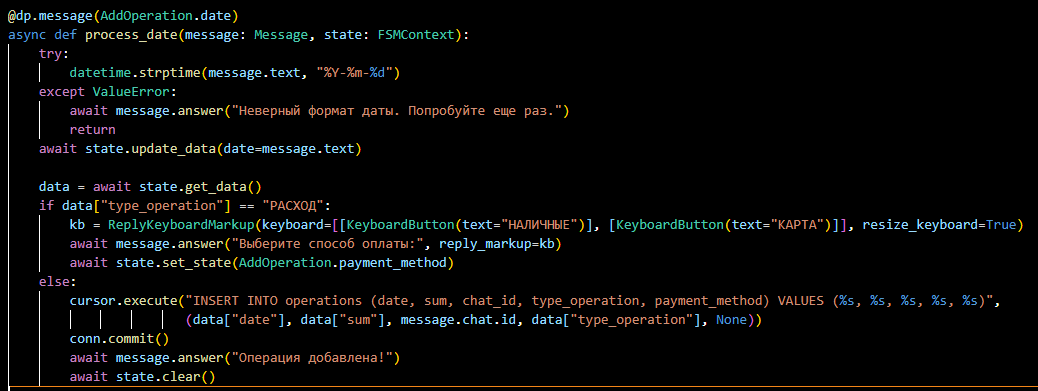


Рисунок 19 – состояние со способом оплаты.

9. Бот сохраняет в таблицу operations: дату операции, сумму операции, chat id пользователя, тип операции.

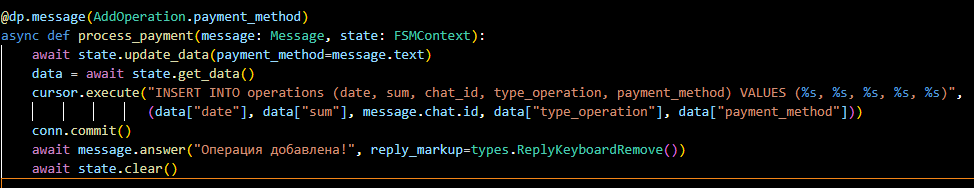


Рисунок 20 – добавление данных в таблицу

10. Бот сообщает пользователю о том, что операция добавлена.



Рисунок 21 – успешное добавление.

2.1.4 Просмотр операций пользователя

Реализовал функциональность просмотра операций пользователя. Просмотр операций пользователя осуществляется посредством отправки команды /operations.

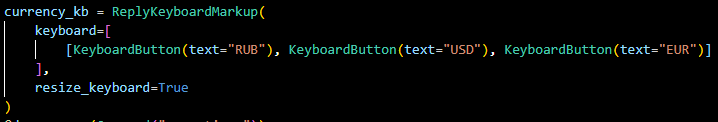


Рисунок 22 – кнопки для пользователя

Сценарий обработки команды:

1. Пользователь вводит команду /operations.

2. Бот проверяет, что пользователь зарегистрирован.

3. Бот выводит три кнопки: RUB, EUR, USD. Пользователь должен выбрать в какой валюте он желает получить информацию по операциям.

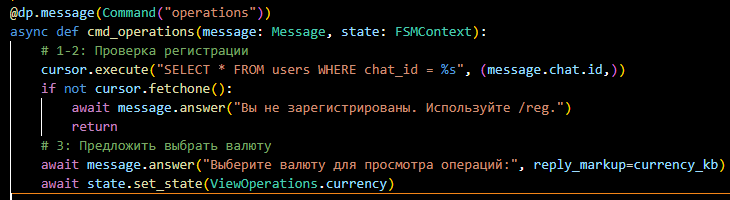


Рисунок 23 – обработка команды /operations

4. Пользователь выбирает одну из доступных валют.

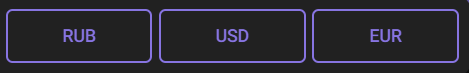


Рисунок 24 – кнопки для выбора валюты

5. Если пользователь выбрал EUR или USD, то бот отправляет запрос на

получение актуального курса во внешний сервис (см раздел 2.3).

6. получает по chat id все операции пользователя из operations.

7. выполняет конвертацию операций в выбранную пользователем в п. 4 валюту.

8. Бот выводит информацию пользователю по всем операциям.



Рисунок 25 – обработка выбранной валюты

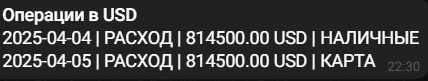


Рисунок 26 – ответ бота.

# 2.3 Внешний сервис

Реализовал Flask сервер, который принимает GET запрос на rate с query param

currency. Параметр currency может быть: EUR, USD.Курс для валют можно задать статически.



Рисунок 27 – фласк сервер

# Заключение

В ходе выполнения работы был разработан функциональный Telegram-бот с возможностью учета финансовых операций и поддержкой конвертации валют через внешний сервис курсов валют. Реализация позволила закрепить знания по работе с библиотекой aiogram, организации состояний пользователя с помощью FSM, взаимодействию с базой данных PostgreSQL, а также созданию и интеграции REST API на базе Flask. Полученное решение может быть расширено дополнительными функциями и служит хорошей основой для создания комплексных микросервисных приложений.

# Список литературы

1) aiogram — библиотека для создания Telegram-ботов [Электронный ресурс]. — Режим доступа: https://docs.aiogram.dev/en/latest/ (дата обращения: 17.05.2025).

2) Flask — микрофреймворк для Python [Электронный ресурс]. — Режим доступа: https://flask.palletsprojects.com/en/latest/ (дата обращения: 17.05.2025).

3) psycopg2 — PostgreSQL adapter for Python [Электронный ресурс]. — Режим доступа: https://www.psycopg.org/docs/ (дата обращения: 17.05.2025).

4) PostgreSQL: Documentation [Электронный ресурс]. — Режим доступа: https://www.postgresql.org/docs/ (дата обращения: 17.05.2025).

5) Fowler M. Microservices: a definition of this new architectural term // martinfowler.com. — 2014. — Режим доступа: https://martinfowler.com/articles/microservices.html (дата обращения: 17.05.2025).

6) Лутовинов А. А., Баранов В. В. Архитектура микросервисов: принципы, шаблоны и лучшие практики / А. А. Лутовинов, В. В. Баранов. — М.: Питер, 2020. — 256 с.

7) Документация Python [Электронный ресурс]. — Режим доступа: https://docs.python.org/3/ (дата обращения: 17.05.2025).

# Приложение

Ссылка на гитхаб - <https://github.com/lupusin/proga>