



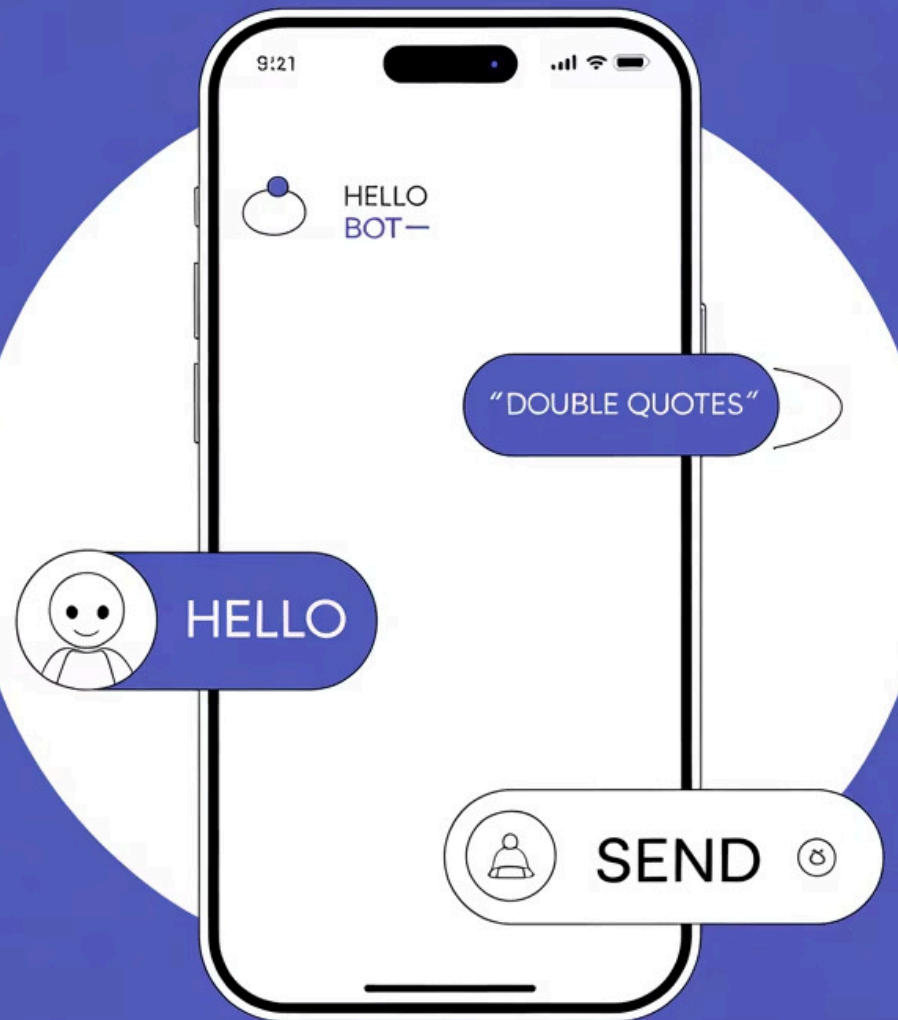
L1 — Введение: Telegram как мобильная платформа

Простой бот на Python за 3 месяца

Преподаватели: Мукасеев Евгений / Шестаков Антон

Добро пожаловать на курс по разработке Telegram-ботов! В ходе этого курса вы научитесь создавать функциональных ботов с нуля, даже если у вас минимальный опыт программирования.

Почему Telegram = «мобильная платформа»



Чат-интерфейс

Краткость и понятные кнопки делают взаимодействие интуитивным и удобным



Малый экран

Фокус на тексте - главная информация всегда на виду



Быстрая доставка

Мгновенные уведомления и стабильная работа API



Низкий порог входа

Видимый результат за 10 минут даже для новичков!

Что делаем за курс

Цель: Личный бот

- Работает на long polling
- Команды + интерактивное меню
- Локальная база данных SQLite
- Код размещён на GitHub
- Документация README + лицензия



Финальная демонстрация: 5-7 минут живого показа работы вашего бота

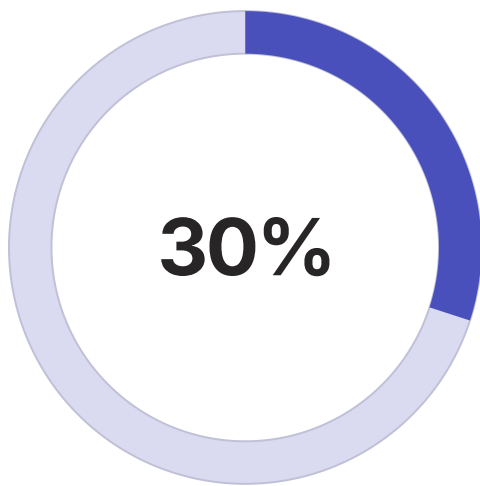
📋 **Ограничения:** Без Docker, webhooks, ORM и внешних API

Дорожная карта



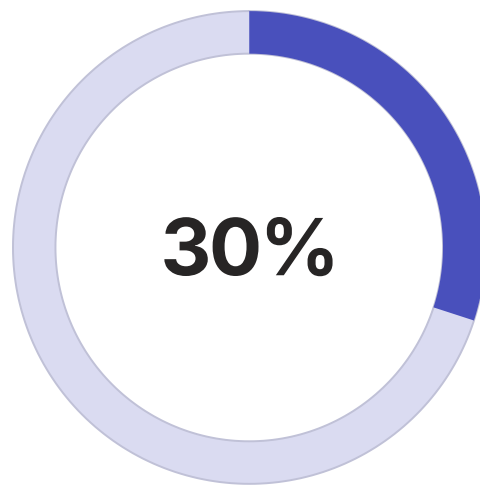
Структура курса: 1 лекция + 2 семинара в неделю через неделю

Оценивание



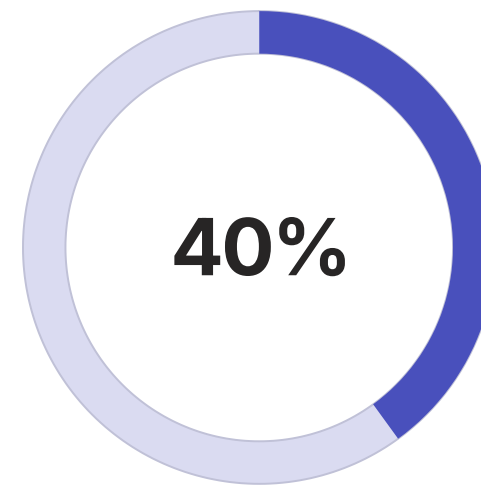
Посещение и активность

Участие в обсуждениях, работа на семинарах, своевременное выполнение заданий



Домашние задания

Регулярные задания после каждой темы



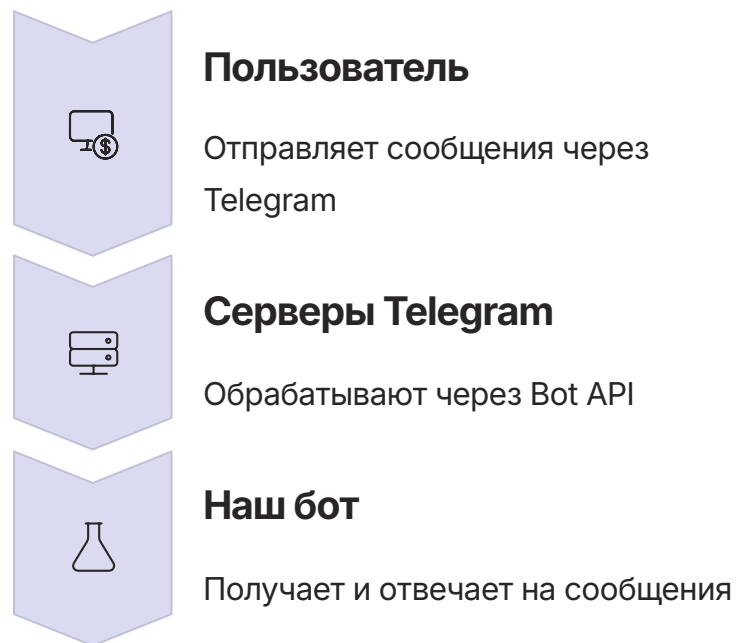
Итоговый проект

Функциональный бот с документацией и презентацией результатов

Критерии финального проекта:

- Бот запускается локально без ошибок
- Все команды работают корректно
- База данных сохраняет и извлекает информацию
- Наличие README с описанием функционала

Как работает бот (схема)

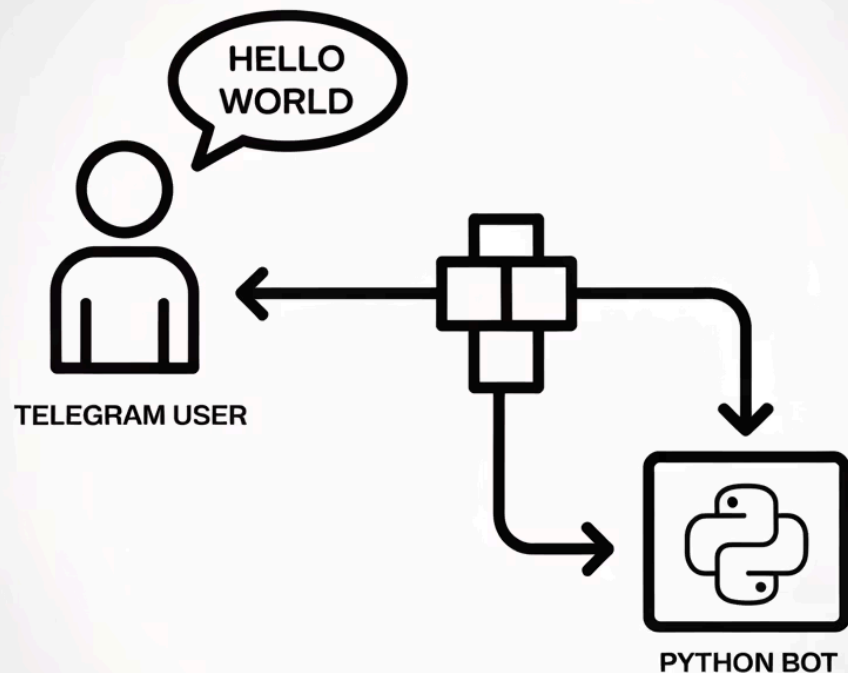


Long polling:

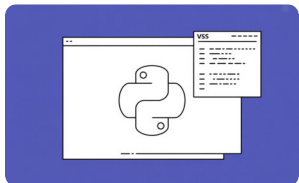
Бот регулярно опрашивает серверы Telegram на наличие новых апдейтов

Используем:

- pyTelegramBotAPI
- Application, Handlers
- Локальный запуск: `run_polling()`



Инструменты курса



Python 3.11 + VS Code

Современная версия языка и удобная среда разработки с подсветкой синтаксиса



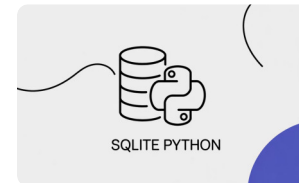
Git и GitHub

Система контроля версий и платформа для совместной работы, Pull Request и код-ревью



pyTelegramBotAPI 4.29.x

Современная библиотека для создания ботов



sqlite3, dotenv, logging

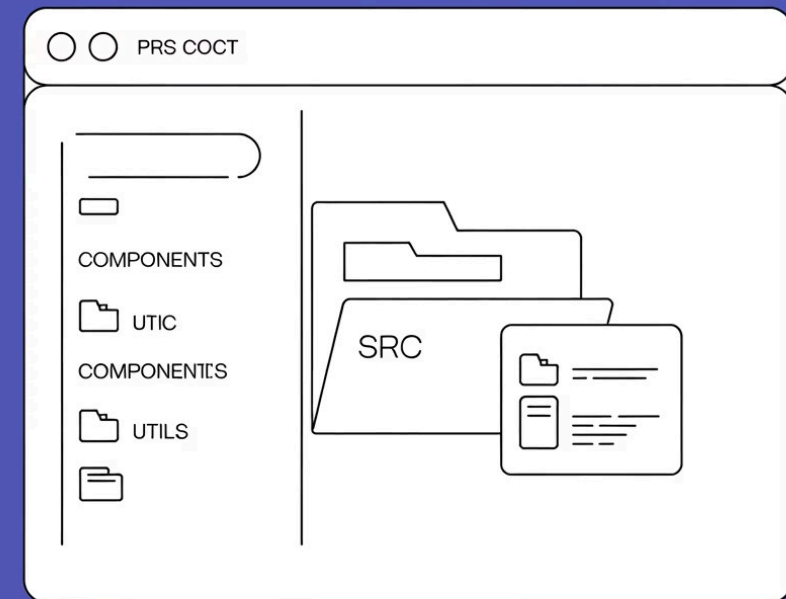
Встроенная БД, управление конфигурацией и логирование без внешних зависимостей

Структура проекта (минимум)

```
tg-bot-simple/  
├── .gitignore  
├── .env  
├── requirements.txt  
├── config.py  # .env + logging  
├── db.py      # БД и CRUD (позже)  
└── main.py    # бот и команды
```

❌ **Важно:** .env не коммитится в репозиторий!

Токен бота хранится только локально



Преимущества структуры:

- Четкое разделение ответственности
- Простота навигации в коде
- Легкость масштабирования проекта
- Соответствие принципам модульности

Установка окружения (обзор)

Установка базовых инструментов

Python 3.11, Git, Visual Studio Code

Проверка установки: `python --version`, `git --version`

Установка зависимостей

`pip install --upgrade pip`

`pip install python-telegram-bot python-dotenv`

или `pip install -r requirements.txt`

Создание и активация виртуального окружения

`python -m venv .venv`

Windows: `.venv\Scripts\activate`

Linux/Mac: `source .venv/bin/activate`

Проверка окружения

Запуск тестового скрипта

Проверка версий установленных пакетов

Детальные инструкции будут представлены на семинаре S1 и в раздаточных материалах

Мини-демо «Hello, Bot» (код)

```
import os
from dotenv import load_dotenv # читает переменные из файла .env
import telebot                # синхронная библиотека для Telegram Bot API

# 1) Загружаем .env и читаем токен
load_dotenv()
TOKEN = os.getenv("TOKEN")    # в .env должен быть TOKEN=... (получить у @BotFather)

if not TOKEN:
    raise RuntimeError("Не найден TOKEN в .env. Создай .env и добавь TOKEN=...")

# 2) Создаём объект бота
bot = telebot.TeleBot(TOKEN)  # здесь настраивается подключение к Bot API

# 3) Обработчик команды /start
@bot.message_handler(commands=['start'])
def start(message):
    # message — входящее сообщение от пользователя
    bot.reply_to(
        message,
        "Привет! Я живой"
        "Команды: /start, /help"
    )

# 4) Обработчик команды /help
@bot.message_handler(commands=['help'])
def help_cmd(message):
    bot.reply_to(
        message,
        "/start — начать диалог"
        "/help — показать подсказку"
    )

# 5) Точка входа: запускаем long polling
if __name__ == "__main__":
    # infinity_polling — бот бесконечно опрашивает серверы Telegram на новые сообщения
    # skip_pending=True — пропустить «старые» сообщения, накопленные пока бот был выключен
    bot.infinity_polling(skip_pending=True)
```

Безопасность и практики



Защита токенов и секретов

Храните только в .env файле

Никогда не коммитьте секреты в репозиторий

Используйте .env.example как шаблон без реальных данных



Грамотный .gitignore

Исключайте .env, .venv/, __pycache__/

Не коммитьте временные файлы и кеша

Используйте готовые шаблоны для Python-проектов



Безопасное логирование

Не включайте токены в логи

Логируйте ошибки для отладки

Используйте разные уровни логирования



Осмысленные коммиты

✓ "Добавлена обработка команды /help"

✗ "Фикс", "Обновление", "Изменения"

Частые вопросы (FAQ)

Можно обойтись без GitHub?

По требованиям курса необходим PR-процесс и код-ревью, поэтому GitHub обязателен. Это также даст вам ценный опыт работы с инструментами, используемыми в реальных проектах.

Нужен ли VDS/домен для бота?

Нет, в рамках курса все запускается локально на вашем компьютере. Это упрощает разработку и отладку, позволяя сосредоточиться на логике бота.

Почему без webhooks/Docker?

Мы фокусируемся на основах: коде бота и работе с SQLite. Это позволяет быстрее погрузиться в разработку без лишних сложностей с инфраструктурой.

Можно использовать aiogram?

По умолчанию используем pyTelegramBotAPI; aiogram опционально. Важно сначала освоить основной инструмент курса.



Мини-квиз

Вопросы:

1. Что такое long polling?
2. Где следует хранить токен бота?
3. Какую библиотеку мы используем для разработки ботов?
4. Зачем нужно виртуальное окружение (venv)?
5. Что такое PR (Pull Request)?



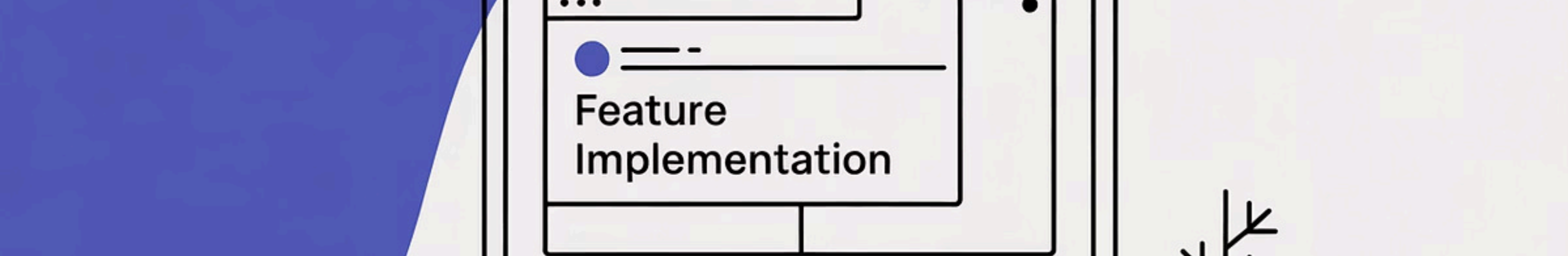
Мини-квиз

Вопросы:

1. Что такое long polling?
2. Где следует хранить токен бота?
3. Какую библиотеку мы используем для разработки ботов?
4. Зачем нужно виртуальное окружение (venv)?
5. Что такое PR (Pull Request)?

Ответы:

1. Метод, при котором бот регулярно опрашивает серверы Telegram на наличие новых сообщений
2. В .env файле, который не должен попадать в репозиторий
3. pyTelegramBotAPI версии 4.29.x
4. Для изоляции зависимостей проекта от системных пакетов Python
5. Предложение изменений кода для ревью перед слиянием в основную ветку



Feature Implementation

Ближайшие шаги (ноутбук и зарядка)

1

IDE и venv

Настроить окружение:

- Установить IDE, Python, Git
- Создать venv, установить зависимости
- Настроить .gitignore

2

Создать репозиторий

Разместить код на GitHub:

- Инициализировать Git-репозиторий
- Создать репозиторий на GitHub
- Выполнить push кода
- Ссылку прислать в группу для единого списка

3

Python

Повторим:

- списки/словари/функции
- list vs dict; if/elif; for; range(3); def
- «Hello, Bot» — первый запуск (если успеем)

Контакты и вопросы

Каналы связи:

- Канал курса: <https://t.me/+g1OIVUyJP6czZWQy>
- Вопросы соседу справа и соседу слева
- Оперативная связь: чат группы

Как задавать вопросы эффективно:

- Четко формулируйте проблему
- Добавляйте примеры кода с ошибкой
- Описывайте, что пробовали сделать для решения
- Прикладывайте логи ошибок (без токенов!)



Спасибо за внимание! Не забывайте о важности сохранения приватности токенов и данных.