

Специалист по системам искусственного интеллекта

МГТУ им. Н.Э. Баумана

План занятий

Разработка чат-бота для поддержки учебного процесса

Часть 1 (Введение и создание бота)

- a. Знакомство с Telegram API и BotFather (регистрация бота, получение токена).
- b. Установка библиотек и настройка окружения.
- c. Первый бот: обработка команд /start и /help.
- d. Основы работы с telebot: отправка и получение сообщений.

Часть 2 (Кнопки и интерактивность)

- e. Типы кнопок: ReplyKeyboardMarkup и InlineKeyboardMarkup.
- f. Обработка нажатий на кнопки.
- g. Динамические меню (например, выбор темы урока).

План занятий

Разработка чат-бота для поддержки учебного процесса

Часть 3 (Работа с медиа (фото, документы))

- a. Прием изображений от пользователя.
- b. Отправка изображений и файлов (PDF, аудио).
- c. Обработка ошибок (например, если прислали не изображение).

Часть 4 (Интеграция с GigaChat API)

- d. Что такое GigaChat и зачем его подключать?
- e. Пример запроса к API (авторизация, отправка вопроса, получение ответа).
- f. Встраивание ИИ в бота (например, ответы на вопросы учеников).

План занятий



Разработка чат-бота для поддержки учебного процесса

Часть 5 (База знаний и SQLite)

- a. Создание базы данных для FAQ (вопрос → ответ).
- b. CRUD-операции: добавление, чтение, удаление данных.
- c. Поиск по ключевым словам (например, "как сдать домашку?").

Часть 6 (Личный кабинет и админ-панель)

- d. Авторизация пользователей (ученик/преподаватель).
- e. Хранение данных: прогресс ученика, настройки.



Часть 1 (Введение и создание бота)

Знакомство с BotFather

Чтобы создать бота, нам нужно обратиться к официальному инструменту Telegram - @BotFather. Это специальный бот, который помогает создавать и настраивать других ботов.

Пошаговая инструкция:

1. Найдите @BotFather в поиске Telegram
2. Начните диалог командой /newbot
3. Придумайте имя вашего бота (например, EnglishTeacherBot)
4. Получите уникальный токен доступа

Важно! Токен - это как пароль от вашего бота. Никому его не передавайте и не публикуйте в открытых источниках.

Установка библиотек

Для работы с Telegram API в Python мы будем использовать библиотеку **pyTelegramBotAPI**.

```
bash
```

```
pip install pyTelegramBotAPI
```

Структура проекта

Давайте создадим правильную структуру проекта с самого начала. Это поможет поддерживать порядок в коде.

Рекомендуемая структура папок:

```
my_telegram_bot/
```

```
└─ main.py      # Основной код бота
```

```
└─ config.py    # Храним токен и настройки
```

```
  └─ requirements.txt # Список зависимостей (опционально)
```


Первый код — бот-эхо

```
import telebot

bot = telebot.TeleBot(TOKEN)


@bot.message_handler(commands=['start'])
def send_welcome(message):
    bot.reply_to(message, "Привет! Я твой учебный бот.")

bot.polling()
```


Первый код — бот-эхо

Разберём ключевые моменты:

- `@bot.message_handler` - декоратор для обработки сообщений
- `commands=['start']` - фильтр для команды `/start`
- `bot.reply_to` - метод для ответа пользователю
- `bot.polling()` - запускает бесконечный цикл проверки сообщений



Часть 2 (Кнопки и интерактивность)



Типы кнопок

Telegram предлагает два основных типа кнопок:

1. ReplyKeyboardMarkup - обычная клавиатура под полем ввода
 - Проста в реализации
 - Всегда видна пользователю
2. InlineKeyboardMarkup - кнопки внутри сообщения
 - Могут обновлять сообщения
 - Позволяют создавать сложные интерфейсы

Примеры использования:

- Reply - для главного меню
- Inline - для опросов и действий

Создаем Reply-клавиатуру

```
from telebot import types

markup = types.ReplyKeyboardMarkup(resize_keyboard=True)

btn1 = types.KeyboardButton('Расписание')
btn2 = types.KeyboardButton('Домашнее задание')

markup.add(btn1, btn2)

@bot.message_handler(commands=['start'])
def start(message):
    bot.send_message(message.chat.id, "Выберите действие:", reply_markup=markup)
```

Создаем Reply-клавиатуру

Что важно:

- `resize_keyboard=True` - адаптирует размер кнопок
- `add()` - добавляет кнопки в ряд
- `reply_markup` - прикрепляет клавиатуру

Обработка нажатий Reply-кнопок

```
@bot.message_handler(func=lambda message: True)
def handle_buttons(message):
    if message.text == 'Расписание':
        bot.send_message(message.chat.id, "Понедельник: математика в
10:00")
    elif message.text == 'Домашнее задание':
        bot.send_message(message.chat.id, "Задача на странице 45")
```

Inline-кнопки

```
inline_markup = types.InlineKeyboardMarkup()

btn = types.InlineKeyboardButton(

    text="Подробнее",

    url="https://edu-site.com"

)

inline_markup.add(btn)
```

```
@bot.message_handler(commands=['info'])

def info(message):

    bot.send_message(

        message.chat.id,

        "Нажмите для дополнительной

информации:",

        reply_markup=inline_markup

    )
```


Inline-кнопки

Особенности:

- Могут открывать URL
- Могут вызывать callback-функции
- Не сохраняются после нажатия"

Обработка Inline-кнопок (callback)

```
@bot.callback_query_handler(func=lambda call: True)
```

```
def callback_handler(call):
```

```
    if call.data == 'show_more':
```

```
        bot.answer_callback_query(
```

```
            call.id,
```

```
            "Дополнительная информация загружается"
```


```
        )
```

```
        bot.send_message(call.message.chat.id, "Вот  
подробности...")
```


Важно:

- Используем `callback_query_handler`
- `call.data` - содержит идентификатор кнопки
- `answer_callback_query` - подтверждает нажатие"

callback query handler — это декоратор в библиотеке `pyTelegramBotAPI`, который позволяет обрабатывать **callback-запросы** (события, возникающие при нажатии на кнопку или действии внутри сообщения бота)



Часть 3 (Работа с медиа (фото, документы))



Отправка изображений (локальные файлы)

```
photo = open('lesson1.jpg', 'rb')  
bot.send_photo(  
    chat_id=message.chat.id,  
    photo=photo,  
    caption='Рисунок к уроку 1'  
)  
photo.close()
```

Отправка изображений (по URL)

```
url = 'https://example.com/lesson1.jpg'
bot.send_photo(
    chat_id=message.chat.id,
    photo=url,
    caption='Изображение из интернета'
)
```

Отправка документов

```
doc = open('materials.pdf', 'rb')  
bot.send_document(  
    chat_id=message.chat.id,  
    document=doc,  
    caption='Учебные материалы',  
    visible_file_name='lesson_materials.pdf'  
)  
doc.close()
```

Получение файлов от пользователей

```
file_id = message.photo[-1].file_id
file_info = bot.get_file(file_id)
downloaded_file = bot.download_file(file_info.file_path)
with open('user_photo.jpg', 'wb') as new_file:
    new_file.write(downloaded_file)
bot.reply_to(message, "Фото сохранено!")
```



Часть 4 (Интеграция с GigaChat API)

