МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет “Информатика и системы управления” Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования” Отчет по ДЗ

Выполнил:

Студент группы ИУ5-35Б

Костылев М.С.

Преподаватель:

Гапанюк Ю.Е.

Москва 2025

Задание: создание тг бота

Код программы:

```
#!/usr/bin/env python3
"""
Минимальный Telegram бот для заметок
"""

import logging
import sqlite3
from datetime import datetime
from telegram import Update, ReplyKeyboardMarkup, KeyboardButton
from telegram.ext import Application, CommandHandler, MessageHandler, filters, CallbackContext

#Настройка логирования
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    level=logging.INFO
)
logger = logging.getLogger(__name__)

TOKEN = "8467986685:AAHaFc0j-x25RULNKV8hmgzVkdJcrZFbwuo"

class SimpleNoteBot:
    def __init__(self):
        self.init_db()

    def init_db(self):
```

```
    """Инициализация базы данных"""
    try:
        self.conn = sqlite3.connect('simple_notes.db', check_same_thread=False)
        self.cursor = self.conn.cursor()

        #Создаем таблицу если ее нет
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS notes (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                user_id INTEGER NOT NULL,
                text TEXT NOT NULL,
                created_at TEXT DEFAULT CURRENT_TIMESTAMP
            )
        ''')
        self.conn.commit()
        logger.info("База данных инициализирована")

    except Exception as e:
        logger.error(f"Ошибка инициализации БД: {e}")
        raise

def add_note(self, user_id, text):
    """Добавление заметки"""
    try:
        self.cursor.execute(
            "INSERT INTO notes (user_id, text) VALUES (?, ?)",
            (user_id, text)
```

```
            )
            self.conn.commit()
            return self.cursor.lastrowid
        except Exception as e:
            logger.error(f"Ошибка добавления заметки: {e}")
            return None

    def get_notes(self, user_id, limit=10):
        """Получение заметок пользователя"""
        try:
            self.cursor.execute(
                "SELECT id, text, created_at FROM notes WHERE user_id = ? ORDER
BY id DESC LIMIT ?",
                (user_id, limit)
            )
            return self.cursor.fetchall()
        except Exception as e:
            logger.error(f"Ошибка получения заметок: {e}")
            return []

    def get_note_count(self, user_id):
        """Количество заметок пользователя"""
        try:
            self.cursor.execute(
                "SELECT COUNT(*) FROM notes WHERE user_id = ?",
                (user_id,)
            )
            return self.cursor.fetchone()[0]
```

```
        except Exception as e:
            logger.error(f"Ошибка подсчета заметок: {e}")
            return 0

    def delete_note(self, note_id, user_id):
        """Удаление заметки"""
        try:
            self.cursor.execute(
                "DELETE FROM notes WHERE id = ? AND user_id = ?",
                (note_id, user_id)
            )
            self.conn.commit()
            return self.cursor.rowcount > 0
        except Exception as e:
            logger.error(f"Ошибка удаления заметки: {e}")
            return False

bot_db = SimpleNoteBot()


async def start_command(update: Update, context: CallbackContext):
    """Обработчик /start"""
    user = update.effective_user
    note_count = bot_db.get_note_count(user.id)

    welcome_text = f"""
👋 Привет, {user.first_name}!
```

📝 *Простой бот для заметок*

Просто напиши мне текст, и я сохраню его как заметку.

Команды:
/start - начать
/notes - показать заметки
/delete [номер] - удалить заметку
/help - справка

У тебя сейчас *{note_count}* заметок.
"""

```
    keyboard = [
        [KeyboardButton("/notes"), KeyboardButton("/help")]
    ]
    reply_markup = ReplyKeyboardMarkup(keyboard, resize_keyboard=True)

    await update.message.reply_text(welcome_text, parse_mode='Markdown',
reply_markup=reply_markup)

async def notes_command(update: Update, context: CallbackContext):
    """Обработчик /notes"""
    user_id = update.effective_user.id
    notes = bot_db.get_notes(user_id, limit=10)
```

```
    if not notes:
        await update.message.reply_text("📭 У тебя пока нет заметок. Напиши что-нибудь!")
        return

    message = "📋 *Твои заметки:*\n\n"

    for note in notes:
        note_id, text, created_at = note

        #Обрезаем длинный текст
        if len(text) > 50:
            display_text = text[:50] + "..."
        else:
            display_text = text

        message += f"*#{note_id}*\n"
        message += f"{display_text}\n"
        message += f"_{created_at}_\n\n"

    total = bot_db.get_note_count(user_id)
    if total > 10:
        message += f" Всего заметок: {total}"

    await update.message.reply_text(message, parse_mode='Markdown')

async def delete_command(update: Update, context: CallbackContext):
```

```
    """Обработчик /delete"""
    if not context.args:
        await update.message.reply_text(
            "Используй: /delete [номер заметки]\n"
            "Пример: /delete 5\n\n"
            "Чтобы увидеть номера заметок, используй /notes"
        )
        return

    try:
        note_id = int(context.args[0])
        user_id = update.effective_user.id

        if bot_db.delete_note(note_id, user_id):
            await update.message.reply_text(f"✅ Заметка #{note_id} удалена!")
        else:
            await update.message.reply_text(f"❌ Заметка #{note_id} не найдена")

    except ValueError:
        await update.message.reply_text("❌ Номер заметки должен быть числом")

async def help_command(update: Update, context: CallbackContext):
    """Обработчик /help"""
    help_text = """
📚 *Справка по командам:*
```

Основные команды:
/start - начать работу
/notes - показать заметки
/delete [номер] - удалить заметку
/stats - статистика
/clear - удалить все заметки
/help - эта справка

Как использовать:
1. Просто напиши текст - я сохраню его
2. Используй /notes чтобы посмотреть все заметки
3. Используй /delete [номер] чтобы удалить

Примеры:
"Купить молоко" - сохранится как заметка
"Завтра встреча в 15:00" - тоже сохранится
"""

    await update.message.reply_text(help_text, parse_mode='Markdown')

async def handle_text_message(update: Update, context: CallbackContext):
    """Обработчик обычных текстовых сообщений"""

    if update.message.text.startswith('/'):
        return

    text = update.message.text.strip()

```
    user_id = update.effective_user.id

    if not text:
        await update.message.reply_text("Текст заметки не может быть пустым")
        return

    #Добавляем заметку
    note_id = bot_db.add_note(user_id, text)

    if note_id:
        #Форматируем ответ
        if len(text) > 100:
            display_text = text[:100] + "..."
        else:
            display_text = text

        reply = f"💾 *Заметка #{note_id} сохранена!*\n\n"
        reply += f"_{display_text}_\n\n"
        reply += f"Всего заметок: {bot_db.get_note_count(user_id)}"

        await update.message.reply_text(reply, parse_mode='Markdown')
    else:
        await update.message.reply_text("❌ Не удалось сохранить заметку")

async def error_handler(update: Update, context: CallbackContext):
    """Обработчик ошибок"""
    logger.error(f"Ошибка: {context.error}")
```

```
    if update and update.effective_message:
        try:
            await update.effective_message.reply_text("☹ Что-то пошло не так. Попробуй еще раз!")
        except:
            pass

async def stats_command(update: Update, context: CallbackContext):
    """Показать статистику"""
    user_id = update.effective_user.id

    try:
        #Получаем общее количество заметок
        bot_db.cursor.execute(
            "SELECT COUNT(*) as count FROM notes WHERE user_id = ?",
            (user_id,)
        )
        result = bot_db.cursor.fetchone()
        note_count = result[0] if result else 0

        #Получаем дату последней заметки
        bot_db.cursor.execute(
            "SELECT created_at FROM notes WHERE user_id = ? ORDER BY id DESC LIMIT 1",
            (user_id,)
        )
        result = bot_db.cursor.fetchone()
```

```
        if result and result[0]:
            last_date = result[0]
            try:
                #Если дата в формате SQLite
                if ' ' in last_date:
                    date_part, time_part = last_date.split(' ')
                    day, month, year = date_part.split('-')
                    last_date = f"{day}.{month}.{year} {time_part}"
            except:
                pass
        else:
            last_date = "еще нет"


        try:
            bot_db.cursor.execute(
                "SELECT category, COUNT(*) as count FROM notes WHERE user_id
= ? GROUP BY category ORDER BY count DESC LIMIT 5",
                (user_id,)
            )
            categories = bot_db.cursor.fetchall()
        except:
            categories = []


        message = f"📊 *Статистика заметок*\n\n"
        message += f"📝 Всего заметок: *{note_count}*\n"
        message += f" Последняя заметка: {last_date}\n"
```

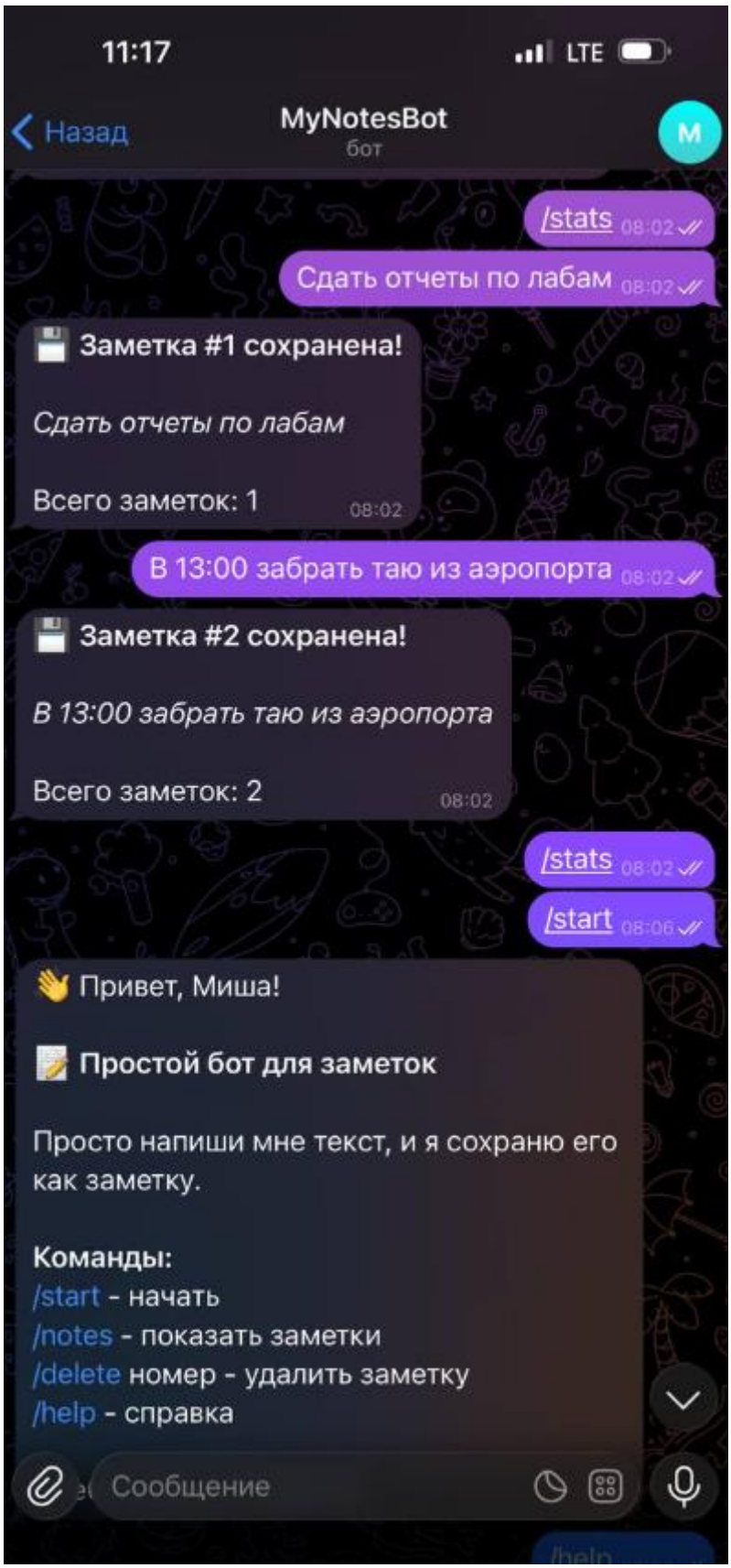
```
        if categories:
            message += f"\n🗂 *Категории:*\n"
            for cat in categories:
                category_name = cat[0] if cat[0] else 'без категории'
                count = cat[1]
                message += f"• #{category_name}: {count}\n"

        if note_count == 0:
            message += f"\n💡 *Совет:* Начните добавлять заметки! Просто напишите мне что-нибудь."

        await update.message.reply_text(message, parse_mode='Markdown')

    except Exception as e:
        logger.error(f"Ошибка в stats_command: {e}")
        await update.message.reply_text("❌ Произошла ошибка при получении статистики.")

async def clear_command(update: Update, context: CallbackContext):
    """Удалить все заметки"""
    user_id = update.effective_user.id
    note_count = bot_db.get_note_count(user_id)

    if note_count == 0:
        await update.message.reply_text("У вас нет заметок для удаления.")
        return

    #Простое удаление всех заметок
```

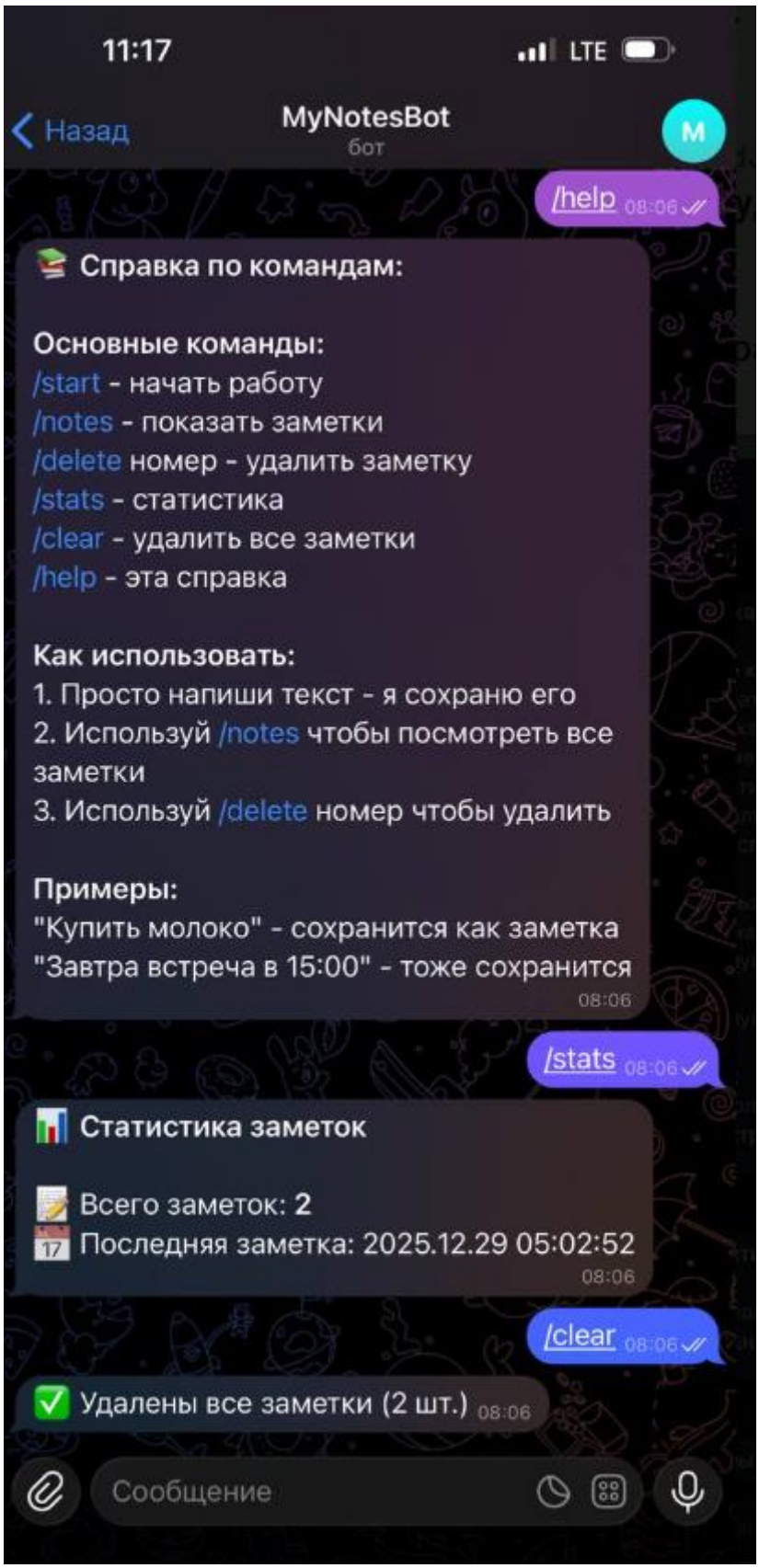
```
    try:
        bot_db.cursor.execute("DELETE FROM notes WHERE user_id = ?", (user_id,))
        bot_db.conn.commit()
        await update.message.reply_text(f"✅ Удалены все заметки ({note_count} шт.)")
    except Exception as e:
        await update.message.reply_text(f"❌ Ошибка при удалении: {e}")



def main():
    """Основная функция"""

    if TOKEN == "ВАШ_ТОКЕН_ЗДЕСЬ":
        print("❌ ОШИБКА: Не указан токен бота!")
        print("1. Откройте этот файл (simple_bot.py)")
        print("2. Замените 'ВАШ_ТОКЕН_ЗДЕСЬ' на токен от @BotFather")
        print("3. Сохраните и запустите: python simple_bot.py")
        return

    print("🤖 Запуск простого бота для заметок...")

    app = Application.builder().token(TOKEN).build()

    app.add_handler(CommandHandler("start", start_command))
    app.add_handler(CommandHandler("notes", notes_command))
    app.add_handler(CommandHandler("delete", delete_command))
```

```
    app.add_handler(CommandHandler("help", help_command))
    app.add_handler(CommandHandler("stats", stats_command))
    app.add_handler(CommandHandler("clear", clear_command))

    app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_text_message))

    app.add_error_handler(error_handler)

    print("✅ Бот запущен. Нажмите Ctrl+C для остановки")

    try:
        bot_info = app.bot.get_me()
        print(f" Бот: @{bot_info.username}")
        print(f"🔗 Ссылка: https://t.me/{bot_info.username}")
    except:
        print("⚠️  Не удалось получить информацию о боте")

    app.run_polling(drop_pending_updates=True)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print("\n👋 Остановка бота...")
        if hasattr(bot_db, 'conn'):
            bot_db.conn.close()
```

```
        print("✅ Бот остановлен")
    except Exception as e:
        print(f"❌ Ошибка: {e}")
        import traceback
        traceback.print_exc()
```

Работа программы:


11:17
LTE
Назад
MyNotesBot
бот
M
/stats 08:02
Сдать отчеты по лабам 08:02
Заметка #1 сохранена!
Сдать отчеты по лабам
Всего заметок: 1 08:02
В 13:00 забрать таю из аэропорта 08:02
Заметка #2 сохранена!
В 13:00 забрать таю из аэропорта
Всего заметок: 2 08:02
/stats 08:02
/start 08:06
Привет, Миша!
Простой бот для заметок
Просто напиши мне текст, и я сохраню его как заметку.
Команды:
/start - начать
/notes - показать заметки
/delete номер - удалить заметку
/help - справка
Сообщение


11:17
LTE
Назад
MyNotesBot
бот
M
/help 08:06
Справка по командам:
Основные команды:
/start - начать работу
/notes - показать заметки
/delete номер - удалить заметку
/stats - статистика
/clear - удалить все заметки
/help - эта справка
Как использовать:
1. Просто напиши текст - я сохраню его
2. Используй /notes чтобы посмотреть все заметки
3. Используй /delete номер чтобы удалить
Примеры:
"Купить молоко" - сохранится как заметка
"Завтра встреча в 15:00" - тоже сохранится
08:06
/stats 08:06
Статистика заметок
Всего заметок: 2
Последняя заметка: 2025.12.29 05:02:52
08:06
/clear 08:06
Удалены все заметки (2 шт.) 08:06
Сообщение