

**Московский государственный технический  
университет им. Н.Э. Баумана**

**Факультет ИУ  
Кафедра ИУ5**

**Курс «Основы информатики»  
Отчет лабораторной работе №6**

Выполнил студент группы ИУ5-33Б:  
Бакушев И.О.  
Подпись и дата:

Проверил преподаватель каф.:  
Гапанюк Ю. Е.  
Подпись и дата:

## Описание задания

1. Разработайте простого бота для Telegram. Бот должен использовать функциональность создания кнопок.

## Текст программы

```
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import Application, CommandHandler, CallbackQueryHandler, MessageHandler,
filters, ContextTypes
from datetime import datetime, timedelta
import asyncio

TOKEN = "0.0"

alarms = {}

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    keyboard = [
        [InlineKeyboardButton("Установить будильник",
callback_data="set_alarm")],
        [InlineKeyboardButton("Удалить будильник",
callback_data="remove_alarm")],
        [InlineKeyboardButton("Список будильников",
callback_data="list_alarms")],
        [InlineKeyboardButton("Меню", callback_data="menu")]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)
    await update.message.reply_text("Привет! Я бот-будильник.
Выберите действие:", reply_markup=reply_markup)

async def button_handler(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    query = update.callback_query
    await query.answer()

    if query.data == "set_alarm":
        await query.message.edit_text("Введите время будильника в
формате HH:MM.")
    elif query.data == "remove_alarm":
        await remove_alarm(update, context)
    elif query.data == "list_alarms":
        await list_alarms(update, context)
    elif query.data == "menu":
        await menu(update, context)

async def menu(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    keyboard = [
        [InlineKeyboardButton("Установить будильник",
callback_data="set_alarm")],
        [InlineKeyboardButton("Удалить будильник",
```

```

callback_data="remove_alarm"]],
    [InlineKeyboardButton("С п и с о к б у д и л ь н и к о в",
callback_data="list_alarms")]
]
reply_markup = InlineKeyboardMarkup(keyboard)
await update.message.reply_text("В ы б е р и т е д е й с т в и е:",
reply_markup=reply_markup)

async def remove_alarm(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    chat_id = update.effective_chat.id
    if chat_id in alarms and len(alarms[chat_id]) > 0:
        alarm = alarms[chat_id].pop(0)
        await update.callback_query.message.reply_text(f"Б у д и л ь н и к н а
{alarm['time']} у д а л ё н.")
    else:
        await update.callback_query.message.reply_text("У в а с н е т
у с т а н о в л е н н о г о б у д и л ь н и к а.")

async def list_alarms(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    chat_id = update.effective_chat.id

    if chat_id in alarms and len(alarms[chat_id]) > 0:
        alarm_times = "\n".join([f"Б у д и л ь н и к н а {alarm['time']}" for alarm in
alarms[chat_id]])
        await update.callback_query.message.reply_text(f"В а ш и
б у д и л ь н и к и :\n{alarm_times}")
    else:
        await update.callback_query.message.reply_text("У в а с н е т
у с т а н о в л е н н о г о б у д и л ь н и к а.")

async def text_handler(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    chat_id = update.effective_chat.id
    text = update.message.text

    try:
        alarm_time = datetime.strptime(text, '%H:%M').time()
        now = datetime.now()

        alarm_datetime = datetime.combine(now.date(), alarm_time)
        if alarm_datetime <= now:
            alarm_datetime += timedelta(days=1)

        time_to_wait = (alarm_datetime - now).total_seconds()
        alarm_str = alarm_datetime.strftime('%H:%M')

        if chat_id not in alarms:
            alarms[chat_id] = []

        alarm_task = asyncio.create_task(send_alarm(context, chat_id, time_to_wait,
alarm_str))

```

```

alarms[chat_id].append({"time": alarm_str, "task": alarm_task})

    await update.message.reply_text(f"Будильник установлен на {alarm_str}.")

    except ValueError:
        await update.message.reply_text("Ошибка: введите время в формате HH:MM.")

async def send_alarm(context: ContextTypes.DEFAULT_TYPE, chat_id: int, delay: float, alarm_time: str) -> None:
    await asyncio.sleep(delay)
    await context.bot.send_message(chat_id=chat_id, text=f"🕒 Время вставать! Это ваш будильник на {alarm_time}.")

def main() -> None:
    application = Application.builder().token(TOKEN).build()

    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("menu", menu))
    application.add_handler(CallbackQueryHandler(button_handler))
    application.add_handler(
        MessageHandler(filters.TEXT & ~filters.COMMAND, text_handler))


    print("Бот запущен.")
    application.run_polling()

if __name__ == "__main__":
    main()


```


## Выполнение программы

AB Введите время будильника в формате HH:MM. 17:58


 17:59 17:58 ✓✓

Будильник установлен на 17:59. 17:58


AB  Время вставать! Это ваш будильник на 17:59. 17:59

 /menu 17:59 ✓✓


AB Введите время будильника в формате HH:MM. 17:59

 18:10 17:59 ✓✓

AB Будильник установлен на 18:10. 17:59

 18:20 17:59 ✓✓

AB Будильник установлен на 18:20. 17:59

 /menu 17:59 ✓✓

Выберите действие: 17:59

Установить будильник

Удалить будильник

AB Список будильников

AB Ваши будильники:  
Будильник на 17:59  
Будильник на 18:10  
Будильник на 18:20 17:59