

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И  
МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Ордена Трудового Красного Знамени федеральное  
государственное бюджетное образовательное  
учреждение высшего образования  
«Московский технический университет связи и информатики»**

---

**Проектная работа  
To do telegram-бот  
по дисциплине  
«Проектный практикум»**

**Выполнил: студент гр. БВТ2201  
Шамсутдинов Р.Ф.**

**Проверил:**

**Москва, 2023 г**

**Цель работы:** сделать телеграм бота с возможностью отслеживания выполнения заданий

**Структура проекта:** код разделен на 3 файла + конфигурационный файл с паролями и токенами

## 1. Файл bot.py, в котором находятся хендлеры команд и запуск бота

Импортируем нужные модули

```
database.py M bot.py M X cfg.py logic.py M
bot.py > command_start_handler
1 import asyncio
2 import logging
3 import emoji
4
5 from datetime import datetime
6 from aiogram import Bot, Dispatcher, Router, types
7 from aiogram.utils.keyboard import InlineKeyboardBuilder, KeyboardBuilder, ReplyKeyboardBuilder
8 from aiogram.filters import Command
9 from aiogram.types import Message
10
11 from database import insert_into_tasks, get_tasks, get_tasks_boolean, insert_into_hide, complete_task, get_by_id
12 from logic import which_emoji, sort_by_date, chek_deadline, time_to_complete, time_to_deadline
13 from cfg import Bot_Token
14
15
```

**Создаем роутер, который будет собирать сообщения, хендлер команды приветствия и хендлер команды с выводом клавиатуры на 4 периода времени**

```
router = Router()

@router.message(Command(commands=["start"]))
async def command_start_handler(message):
    await message.answer(f"Здравствуйте, <b>{message.from_user.full_name}</b> Чтобы узнать больше 🤖 боте, воспользуйтесь командой /help")

@router.message(Command(commands=["time"]))
async def command_time_handler(message):
    try:
        kb = [
            [
                types.KeyboardButton(text="📅 сегодня"),
                types.KeyboardButton(text="📅 неделю")
            ],
            [
                types.KeyboardButton(text="📅 месяц"),
                types.KeyboardButton(text="📅 год")
            ]
        ]
        keyboard = types.ReplyKeyboardMarkup(
            keyboard=kb,
            resize_keyboard=True,
            input_field_placeholder='Выберите задания на какой этап вы хотите увидеть'
        )
        await message.answer('Какие какой этап?' , reply_markup=keyboard)
    except:
        await message.answer('Команда /time не сработала')
```

## Создаем хендлер команды с выводом клавиатуры с выбором типа заданий, которые нужно вывести

```
46
47 @router.message(Command(commands=["print"]))
48 async def command_print_handler(message):
49     try:
50         kb = [
51             [
52                 types.KeyboardButton(text="Все задания")
53             ],
54             [
55                 types.KeyboardButton(text="Только завершённые" + emoji.emojize("✅")),
56                 types.KeyboardButton(text="Только незавершённые" + emoji.emojize("❌"))
57             ],
58             [
59                 types.KeyboardButton(text="Пропущен дедлайн🔔")
60             ]
61         ]
62         keyboard = types.ReplyKeyboardMarkup(
63             keyboard=kb,
64             resize_keyboard=True,
65             input_field_placeholder='Выберите какие задания вы хотите увидеть'
66         )
67         await message.answer('Какие задания вывести?' , reply_markup=keyboard)
68     except:
69         await message.answer('Команда /print не сработала')
70
```

## Хенделер команды с выводом все заданий

```
71 @router.message(Command(commands=['all']))
72 async def command_all_tasks_handler(message):
73     try:
74         a = get_tasks()
75         if len(a) == 0:
76             await message.answer('Заданий нет')
77
78         else:
79             a = sort_by_date(a)
80             await message.answer('id, name, date, completed')
81
82             for task in a:
83
84                 if chek_deadline(task):
85                     task = list(map(str,task))
86                     task = which_emoji(task)
87                     task = " ".join(task)
88                     await message.answer(task)
89                 else:
90                     task = list(map(str,task))
91                     task = which_emoji(task)
92                     task = " ".join(task)
93                     await message.answer(f'<s>{task}</s>')
94     except:
95         await message.answer('Команда /all не сработала')
96
```

## Хендлеры команд с созданием задания и выводом незавершенных заданий

```
96
97 @router.message(Command(commands=['task']))
98 async def command_add_task_handler(message):
99     try:
100         msg = message.text.split()
101         date = msg[-2] + ' ' + msg[-1]
102         name = " ".join(msg[1:-2:])
103         dt = datetime.strptime(date, '%Y-%m-%d %H:%M')
104         insert_into_tasks([name, date])
105
106         await message.answer('Задание добавлено')
107     except:
108         await message.answer('Команда /task не сработала')
109
110 @router.message(Command(commands=['uncompleted']))
111 async def command_uncompleted_handler(message):
112     try:
113         d = get_tasks_boolean('false')
114         if len(d) == 0:
115             await message.answer('Таких заданий нет')
116         else:
117
118             d = sort_by_date(d)
119
120             await message.answer('id, name, date, completed')
121
122             for task in d:
123                 task = list(map(str, task))
124                 task = which_emoji(task)
125                 task = " ".join(task)
126                 await message.answer(task)
127     except:
128         await message.answer('Команда /uncompleted не сработала')
129
```

## Хендлер команды помощи, которая выводит список всех доступных команд и их описание

```
129
130 @router.message(Command(commands=['help']))
131 async def command_help_handler(message):
132     try:
133         a = ['/print Отображает клавиатуру 📄 командами для вывода заданий',
134             '/all Выводит все задания',
135             '/completed Выводит все завершенные задания ', '/uncompleted Выводит все незавершенные задания',
136             '/done Делает задание завершенным \n Пример использования: \n /done 2 3 \n Делает 2-06 и 3-06 задание завершенными',
137             '/del Удаляет задание \n Пример использования: \n /del 2 3 \n Удаляет 2-06 и 3-06',
138             '/task Добавляет задание \n Пример использования: \n /task Пробежать 10 метров 2023-05-21 19:30 \n Добавляет невыполн',
139             '/dead Отображает невыполненные задания 📄 пропущенным дедлайном',
140             '/to_dead Отображает сколько часов и минут осталось до дедлайна \n Пример использования: \n /to_dead 6 7 \n Показывае',
141             '/today Выводит задания на сегодня',
142             '/week Выводит задания на неделю',
143             '/month Выводит задания на месяц',
144             '/year Выводит задания на год',
145             '/time Выводит клавиатуру 📄 предыдущими 4 командами']
146         for item in a:
147             await message.answer(item)
148     except:
149         await message.answer('Команда /help не сработала')
150
151
```

## Хенделры команд с выводом завершенных заданий и удалением заданий

```
151
152 @router.message(Command(commands=['completed']))
153 async def command_completed_handler(message):
154     try:
155         d = get_tasks_boolean('true')
156         if len(d) == 0:
157             await message.answer('Таких заданий нет')
158         else:
159
160             d = sort_by_date(d)
161
162             await message.answer('id, name, date, completed')
163
164             for task in d:
165                 task = list(map(str, task))
166                 task = which_emoji(task)
167                 task = " ".join(task)
168                 await message.answer(task)
169     except:
170         await message.answer('Команда /completed не сработала')
171
172 @router.message(Command(commands=['del']))
173 async def command_del_task_handler(message):
174     try:
175         msg = message.text.split()[1::]
176         for id in msg:
177             insert_into_hide(int(id))
178             await message.answer(f'Задание номер {id} удалено')
179     except:
180         await message.answer('Команда /del не сработала')
181
```



## Хенделры команд завершения задания и выводом заданий с пропущенным дедлайном

```
181
182 @router.message(Command(commands=['done']))
183 async def command_done_tasks_handler(message):
184
185     try:
186         msg = message.text.split()[1::]
187         for id in msg:
188             complete_task(int(id))
189             await message.answer(f'Задание номер {id} выполнено')
190     except:
191         await message.answer('Команда /done не сработала')
192
193
194 @router.message(Command(commands=['dead']))
195 async def command_dead_handler(message):
196     try:
197         a = get_tasks_boolean('false')
198         if len(a) == 0:
199             await message.answer('Заданий нет')
200
201         else:
202             a = sort_by_date(a)
203             await message.answer('id, name, date, completed')
204
205             for task in a:
206
207                 if not chek_deadline(task):
208                     task[3] = emoji.emojize("💀")
209                     task = list(map(str,task))
210                     task = which_emoji(task)
211                     task = " ".join(task)
212                     await message.answer(task)
213
214     except:
215         await message.answer('Команда /dead не сработала')
216
217
```

## Хендлер команд с периодом времени, на который нужно вывести незавершенные задания

```
218 @router.message(Command(commands=['today', 'week', 'month', 'year']))
219 async def command_today_handler(message):
220     try:
221         a = get_tasks_boolean('false')
222
223         msg = message.text
224         print(msg)
225         if msg in ['/today', 'На сегодня']:
226             a = time_to_complete(0,a)
227         elif msg in ['/week', 'На неделю']:
228             a = time_to_complete(7,a)
229         elif msg in ['/month', 'На месяц']:
230             a = time_to_complete(30,a)
231         elif msg in ['/year', 'На год']:
232             a = time_to_complete(365,a)
233
234         if len(a) == 0:
235             await message.answer('Таких заданий нет')
236         else:
237             a = sort_by_date(a)
238
239             await message.answer('id, name, date, completed')
240
241             for task in a:
242                 task = list(map(str,task))
243                 task = which_emoji(task)
244                 task = " ".join(task)
245                 await message.answer(task)
246
247     except:
248         await message.answer(f'Команда {message.text} не сработала')
```

## Хендлер команды с выводом времени до завершения дедлайна задания

```
251 @router.message(Command(commands=['to_dead']))
252 async def command_to_dead_handler(message):
253     try:
254         msg = message.text.split()[1:]
255         for id in msg:
256
257             time = time_to_deadline(get_by_id(id)[0])
258             if time == '':
259                 await message.answer(f'Для задание номер {id} дедлайн закончился')
260             else:
261                 await message.answer(f'До задание номер {id} осталось {time}')
262     except:
263         await message.answer(f'Команда /to_dead не сработала')
264
```

Хендлер сообщений, создание мейн функции в которой происходит запуск бота и условие, при котором вызывается мейн функция

```
265
266 @router.message()
267 async def msg_handler(message):
268     try:
269         msg = message.text
270         if msg == "Все задания":
271             await command_all_tasks_handler(message)
272         elif msg == "Только завершённые✅":
273             await command_completed_handler(message)
274         elif msg == "Только незавершённые❌":
275             await command_uncompleted_handler(message)
276         elif msg == "Пропущен дедлайн":
277             await command_dead_handler(message)
278         elif msg == "На сегодня":
279             await command_today_handler(message)
280         elif msg == "На неделю":
281             await command_today_handler(message)
282         elif msg == "На месяц":
283             await command_today_handler(message)
284         elif msg == "На год":
285             await command_today_handler(message)
286         else:
287             await message.answer('Я вас не понимаю')
288     except TypeError:
289         await message.answer('Ошибка в типе сообщения')
290
291 async def main():
292     dp = Dispatcher()
293     dp.include_router(router)
294     bot = Bot(Bot_Token, parse_mode="HTML")
295     await dp.start_polling(bot)
296
297
298 if __name__ == "__main__":
299     logging.basicConfig(level=logging.INFO)
300     asyncio.run(main())
```



## 2. Файл database.py в котором прописана логика взаимодействия с базой данных

Импортируем нужные модули, подключаемся к базе данных и создаем курсор, с помощью которого будет обращаться к базе данных

```
database.py X bot.py M cfg.py logic.py M
database.py > get_by_id
1 import psycopg2
2
3
4 from cfg import bd_pass
5
6
7 conn = psycopg2.connect(database="to_do_bot", user="postgres",
8                          password=bd_pass, host="localhost", port="5432")
9
10 cursor = conn.cursor()
11
```

Функции создания главной таблицы с данными о заданиях и теневой таблицы, в которой будут записываться айди заданий, которые удалены, то есть те, которые не нужно показывать. А также функции заполнения этих таблиц

```
12
13 def create_table_tasks():
14     cursor.execute('CREATE TABLE tasks(id serial primary key,\
15                  name varchar(1024), date varchar(1024), completed boolean);')
16     conn.commit()
17
18 def create_table_hide():
19     cursor.execute('CREATE TABLE hide (id integer primary key);')
20     conn.commit()
21
22 def insert_into_tasks(msg):
23     name = msg[0]
24     date = msg[1]
25     complete = False
26     cursor.execute(f"INSERT INTO tasks (name, date, completed)\
27                  VALUES ('{name}', '{date}', {complete});")
28     conn.commit()
29
30 def insert_into_hide(id):
31     cursor.execute(f'INSERT INTO hide VALUES ({id});')
32     conn.commit()
33
```

## Функции сбора данных из таблиц и функция выполнения задания

```
33
34 def get_tasks():
35     cursor.execute('select * from tasks where\
36         id NOT IN (SELECT id FROM hide WHERE id IS NOT NULL );')
37     records = list(cursor.fetchall())
38     return records
39
40 def get_tasks_boolean(bol):
41     cursor.execute(f'select * from tasks where completed={bol} and\
42         id NOT IN (SELECT id FROM hide WHERE id IS NOT NULL );')
43
44     records = list(cursor.fetchall())
45     return records
46
47 def get_by_id(id):
48     cursor.execute(f'select * from tasks where id={id} and\
49         id NOT IN (SELECT id FROM hide WHERE id IS NOT NULL );')
50     records = list(cursor.fetchall())
51     return records
52
53
54 def complete_task(id):
55     cursor.execute(f'update tasks set completed=true where id={id};')
56     conn.commit()
57
58
59 if __name__ == '__main__':
60     # create_table_tasks()
61     # create_table_hide()
62     pass
```

### 3. Файл logic.py в котором написаны вспомогательные функции

Импортируем нужные модули, создаем функции добавления эмоджи и сортировки данных по дате дедлайна

```
1  import emoji
2
3
4  from datetime import datetime, timedelta
5
6
7  def which_emoji(a):
8
9      if a[3] == 'True':
10         a[3] = emoji.emojize("✅")
11     elif a[3] == 'False':
12         a[3] = emoji.emojize("❌")
13     return a
14
15
16 def sort_by_date(a):
17
18     a = list(map(list,a))
19     a.sort(key=lambda date: date[2])
20
21     return a
22
```

Создаем функции для получения оставшегося времени до конца дедлайна и проверки на пропущенный дедлайн

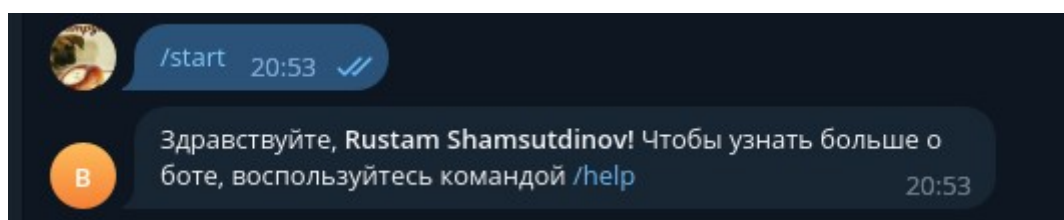
```
23
24 def time_to_deadline(task):
25     dl = datetime.strptime(task[2], '%Y-%m-%d %H:%M')
26     now = datetime.now()
27     rz = int((dl-now).total_seconds())
28     if rz <= 0:
29         res = ''
30     else:
31         if rz//3600 == 1:
32             res = f'{rz//3600} час {(rz-3600*(rz//3600))//60} минут'
33         elif rz//3600 == 0:
34             res = f'{rz//3600} часов {(rz-3600*(rz//3600))//60} минут'
35         else:
36             res = f'{rz//3600} часа {(rz-3600*(rz//3600))//60} минут'
37     return res
38
39
40 def chek_deadline(task):
41
42     return datetime.now() <= datetime.strptime(task[2], '%Y-%m-%d %H:%M')
43
44
```

Создаем функцию для определения того, в какой промежуток времени попадает дедлайн задания

```
45
46 def time_to_complete(days, task):
47     res = []
48     tod = datetime.now().date()
49     for ind, item in enumerate(task):
50         if days == 0:
51             dt = item[2].split()[0]
52             if (tod == datetime.strptime(dt, '%Y-%m-%d').date()):
53                 res.append(item)
54         elif days == 7:
55             dt = datetime.strptime(item[2], '%Y-%m-%d %H:%M').date()
56             dl = datetime.strptime((tod + timedelta(days=7)).strftime('%Y-%m-%d'), '%Y-%m-%d').date()
57             rz = ((dl - dt).days)
58
59             if 0 <= rz <= 7:
60                 res.append(item)
61         elif days == 30:
62             dt = datetime.strptime(item[2], '%Y-%m-%d %H:%M').date()
63             dl = datetime.strptime((tod + timedelta(days=30)).strftime('%Y-%m-%d'), '%Y-%m-%d').date()
64             rz = ((dl - dt).days)
65
66             if 0 <= rz <= 30:
67                 res.append(item)
68         elif days == 365:
69             dt = datetime.strptime(item[2], '%Y-%m-%d %H:%M').date()
70             dl = datetime.strptime((tod + timedelta(days=365)).strftime('%Y-%m-%d'), '%Y-%m-%d').date()
71             rz = ((dl - dt).days)
72
73             if 0 <= rz <= 365:
74                 res.append(item)
75     return res
76
```



#### 4. Демонстрация работы приложения





/help 20:54 ✓✓

/print Отображает клавиатуру с командами для вывода заданий 20:54

/all Выводит все задания 20:54

/completed Выводит все завершенные задания 20:54

/uncompleted Выводит все незавершенные задания 20:54

/done Делает задание завершенным

Пример использования:

/done 2 3

Делает 2-ое и 3-е задание завершенными 20:54

/del Удаляет задание

Пример использования:

/del 2 3

Удаляет 2-ое и 3-е 20:54

/task Добавляет задание

Пример использования:

/task Пробежать 10 метров 2023-05-21 19:30

Добавляет невыполненное задание "Пробежать 10 метров"

С дедлайном "2023-05-21 19:30"

20:54

/dead Отображает невыполненные задания с пропущенным дедлайном 🕒 20:54

/to\_dead Отображает сколько часов и минут осталось до дедлайна

Пример использования:

/to\_dead 6 7

Показывает сколько времени осталось до

дедлайнов 6-го и 7-го задания

20:54

/today Выводит задания на сегодня 20:54


/week Выводит задания на неделю 20:54


/month Выводит задания на месяц 20:54


/year Выводт задания на год 20:54

В

/time Выводит клавиатуру с предыдущими 4 командами 20:54

 /print 20:54 ✓

 Какие задания вывести? 20:54

 Все задания 20:54 ✓

id, name, date, completed 20:54

8 защитить проект 2023-05-19 13:10 ❌ 20:54


9 покушать ужин 2023-05-19 19:30 ✅ 20:54


10 лечь спать 2023-05-19 21:30 ✅ 20:54

6 доделать проект 2023-05-20 18:00 ❌ 20:54

7 защитить проект 2023-05-20 21:00 ❌ 20:54


12 написать ср по вышмату 2023-05-24 14:45 ❌ 20:54


 11 сделать кр по физике 2023-05-24 21:00 ❌ 20:54

 Только завершённые ✅ 20:54 ✓

id, name, date, completed 20:54

9 покушать ужин 2023-05-19 19:30 ✅ 20:54


 10 лечь спать 2023-05-19 21:30 ✅ 20:54




 Пропущен дедлайн 🐼 20:54 ✓

id, name, date, completed 20:54

8 защитить проект 2023-05-19 13:10 🐼 20:54

6 доделать проект 2023-05-20 18:00 🐼 20:54

 7 защитить проект 2023-05-20 21:00 🐼 20:54


 Выберите какие задания вы хотите увидеть  

Все задания

Только завершённые ✅

Только незавершённые ❌

Пропущен дедлайн 🐼

 Только незавершённые ❌ 20:55 ✓


id, name, date, completed 20:55


8 защитить проект 2023-05-19 13:10 ❌ 20:55

6 доделать проект 2023-05-20 18:00 ❌ 20:55

7 защитить проект 2023-05-20 21:00 ❌ 20:55

12 написать ср по вышмату 2023-05-24 14:45 ❌ 20:55

 11 сделать кр по физике 2023-05-24 21:00 ❌ 20:55

 Выберите какие задания вы хотите увидеть

/done 20:55 ✓✓

/done 7 20:55 ✓✓

В Задание номер 7 выполнено 20:55

/del 7 20:55 ✓✓

В Задание номер 7 удалено 20:55

/task написать отчет по проекту 2023-05-23 22:00 20:56 ✓✓

В Задание добавлено 20:56

/all 20:56 ✓✓

id, name, date, completed 20:56

8 защитить проект 2023-05-19 13:10 ❌ 20:56

9 покушать ужин 2023-05-19 19:30 ✓! 20:56

10 лечь спать 2023-05-19 21:30 ✓! 20:56

6 доделать проект 2023-05-20 18:00 ❌ 20:56

13 написать отчет по проекту 2023-05-23 22:00 ❌ 20:56

12 написать ср по вышмату 2023-05-24 14:45 ❌ 20:56

В 11 сделать кр по физике 2023-05-24 21:00 ❌ 20:56

/to\_dead 13 20:56 ✓✓

В До задание номер 13 осталось 1 час 3 минут 20:56

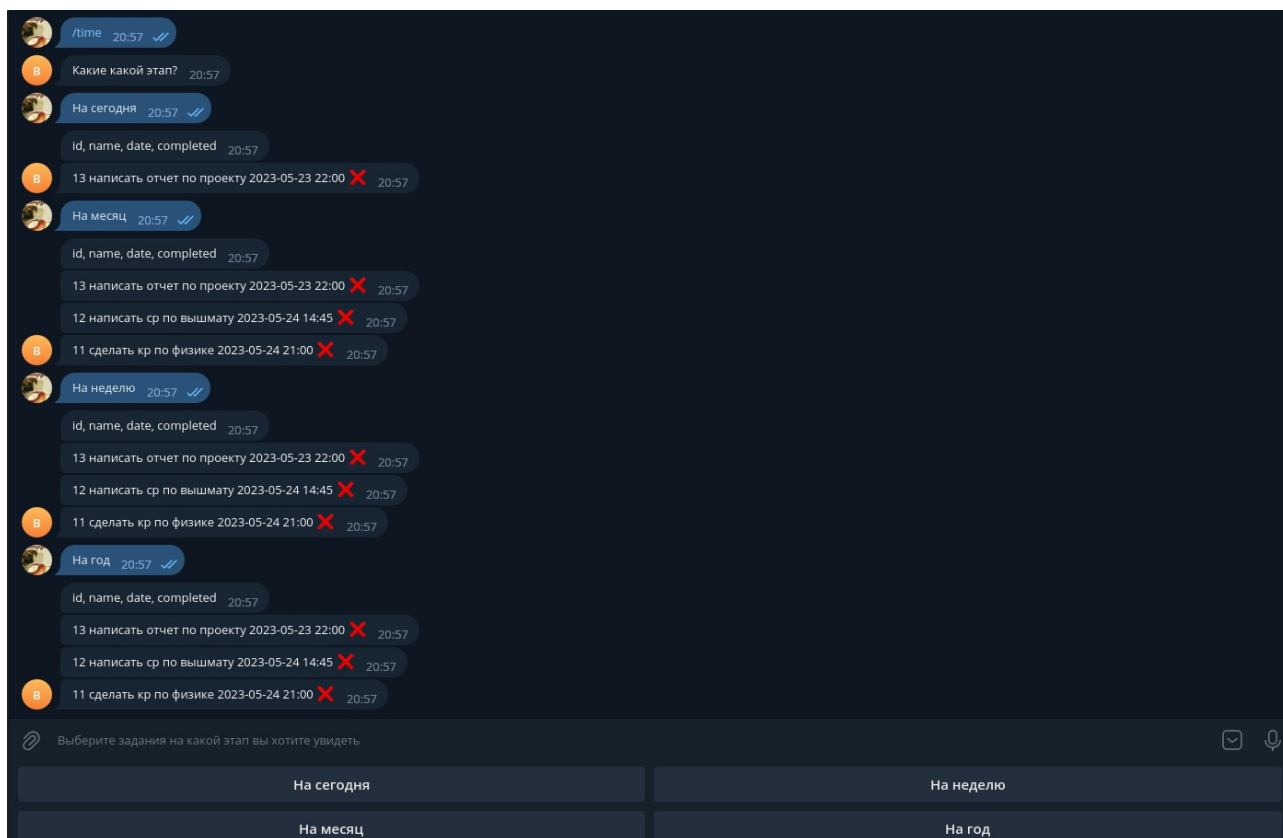
/dead 20:57 ✓✓

id, name, date, completed 20:57

8 защитить проект 2023-05-19 13:10 💀 20:57

В 6 доделать проект 2023-05-20 18:00 💀 20:57





**Вывод:** Прodelав данную работу, я создал телграм бота в котором могу отслеживать выполнение заданий