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

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»
Отчет по лабораторной работе №6

Выполнил:
студент группы ИУ5-34Б
Барабанщиков Лев

Подпись и дата: 23.12.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2021 г.

## Постановка задачи

Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

## Текст программы (bot.py)

```
import asyncio
import logging

from aiogram import Bot, Dispatcher
from aiogram.types import BotCommand
from aiogram.contrib.fsm_storage.memory import MemoryStorage

from app.config import load_config
from app.handlers.exam import register_handlers_exam
from app.handlers.rk import register_handlers_rk
from app.handlers.common import register_handlers_common

logger = logging.getLogger(__name__)


async def set_commands(bot: Bot):
    commands = [
        BotCommand(command="/exam", description="Подготовка к экзаменам"),
        BotCommand(command="/rk", description="Подготовка к РК"),
        BotCommand(command="/cancel", description="Отменить текущее действие")
    ]
    await bot.set_my_commands(commands)


async def main():
    # Настройка логирования в stdout
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s - %(levelname)s - %(name)s - %(message)s",
    )
    logger.error("Starting bot")

    # Парсинг файла конфигурации
    config = load_config("config/bot.ini")

    # Объявление и инициализация объектов бота и диспетчера
    bot = Bot(token=config.tg_bot.token)
    dp = Dispatcher(bot, storage=MemoryStorage())

    register_handlers_exam(dp)
    register_handlers_rk(dp)
    register_handlers_common(dp, config.tg_bot.admin_id)

    # Установка команд бота
    await set_commands(bot)

    # Запуск поллинга
    # await dp.skip_updates()  # пропуск накопившихся апдейтов (необязательно)
```

```
    await dp.start_polling()


if __name__ == '__main__':
    asyncio.run(main())
```

**Текст exam.py**

```
import os

from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup

available_exam_subjects = ["Физика", "ТВиМС"]
available_exam_tickets = ["1", "2", "3", "4", "5"]

cur_path = os.path.dirname(os.path.abspath(__file__))


class ChooseExam(StatesGroup):
    waiting_for_exam_subject = State()
    waiting_for_exam_ticket = State()


async def exam_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for subject in available_exam_subjects:
        keyboard.add(subject)
    await message.answer("Выберите экзаменационный предмет:", reply_markup=keyboard)
    await ChooseExam.waiting_for_exam_subject.set()


async def exam_chosen(message: types.Message, state: FSMContext):
    if message.text not in available_exam_subjects:
        await message.answer("Выберите экзаменационный предмет, используя кнопки ниже.")
        return
    await state.update_data(chosen_exam=message.text)

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for ticket in available_exam_tickets:
        keyboard.add(ticket)
    # для простых шагов можно не указывать название состояния, обходясь next()
    await ChooseExam.next()
    await message.answer("Теперь выберите билет:", reply_markup=keyboard)


async def exam_ticket_chosen(message: types.Message, state: FSMContext):
    if message.text not in available_exam_tickets:
        await message.answer("Выберите билет, используя кнопки ниже.")
        return
    user_data = await state.get_data()
    img = open(os.path.join(cur_path, '..', '..', 'img', 'exam', user_data['chosen_exam'] + message.text + '.png'),
               'rb')
    await message.answer_photo(img, f"Ваш предмет - {user_data['chosen_exam']}. Билет
```

```
{message.text}.\n"
                                f"Учите, а потом можете подготовиться к РК: /rk",
                            reply_markup=types.ReplyKeyboardRemove())
    await state.finish()


def register_handlers_exam(dp: Dispatcher):
    dp.register_message_handler(exam_start, commands="exam", state="*")
    dp.register_message_handler(exam_chosen, state=ChooseExam.waiting_for_exam_subject)
    dp.register_message_handler(exam_ticket_chosen, state=ChooseExam.waiting_for_exam_ticket)
```

**Текст rk.py**

```
import os

from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters.state import State, StatesGroup

available_rk_subjects = ["МатСтат", "Правоведение", "Максвелл и ЭМ волны"]
available_rk_problem = ["1", "2", "3", "4", "5"]

cur_path = os.path.dirname(os.path.abspath(__file__))


class ChooseRK(StatesGroup):
    waiting_for_rk_subject = State()
    waiting_for_rk_problem = State()


async def rk_start(message: types.Message):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for subject in available_rk_subjects:
        keyboard.add(subject)
    await message.answer("Выберите тему РК:", reply_markup=keyboard)
    await ChooseRK.waiting_for_rk_subject.set()


# Обратите внимание: есть второй аргумент
async def rk_chosen(message: types.Message, state: FSMContext):
    if message.text not in available_rk_subjects:
        await message.answer("Выберите тему РК, используя кнопки ниже.")
        return
    await state.update_data(chosen_rk=message.text)

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for problem in available_rk_problem:
        keyboard.add(problem)
    # Для простых шагов можно не указывать название состояния, обходясь next()
    await ChooseRK.next()
    await message.answer("Выберите номер задачи:", reply_markup=keyboard)


async def rk_problem_chosen(message: types.Message, state: FSMContext):
```

```
    if message.text not in available_rk_problem:
        await message.answer("Выберите номер задачи, используя кнопки ниже.")
        return
    user_data = await state.get_data()
    img = open(os.path.join(cur_path, '..', '..', 'img', 'rk', user_data['chosen_rk'] + message.text + '.png'),
               'rb')
    await message.answer_photo(img, f"Вы выбрали {user_data['chosen_rk']}. Задача №{message.text}\n"
                                    f"Хотите подготовиться к экзамену? - /exam",
                               reply_markup=types.ReplyKeyboardRemove())
    await state.finish()


def register_handlers_rk(dp: Dispatcher):
    dp.register_message_handler(rk_start, commands="rk", state="*")
    dp.register_message_handler(rk_chosen, state=ChooseRK.waiting_for_rk_subject)
    dp.register_message_handler(rk_problem_chosen, state=ChooseRK.waiting_for_rk_problem)
```

**Текст common.py**

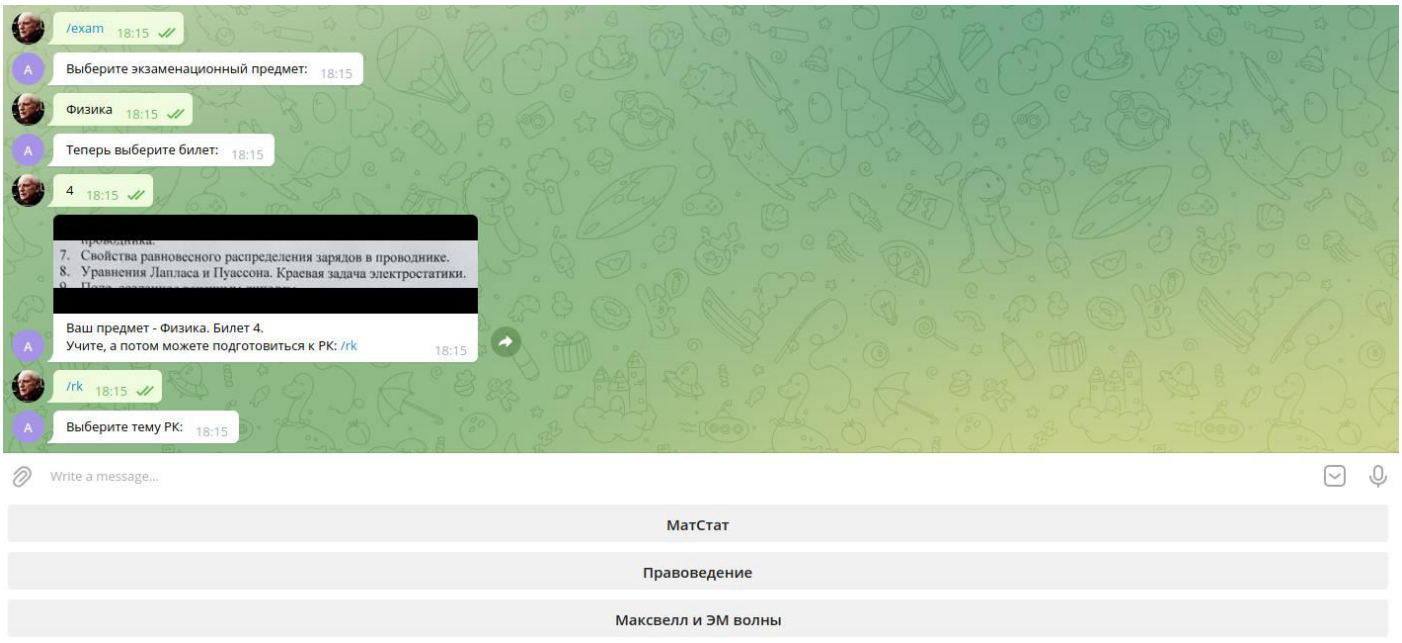
```
from aiogram import Dispatcher, types
from aiogram.dispatcher import FSMContext
from aiogram.dispatcher.filters import Text, IDFilter


async def cmd_start(message: types.Message, state: FSMContext):
    await state.finish()
    await message.answer(
        "Выберите, с чего хотите начать: Подготовка к РК (/rk) или Подготовка к экзамену (/exam).",
        reply_markup=types.ReplyKeyboardRemove()
    )


async def cmd_cancel(message: types.Message, state: FSMContext):
    await state.finish()
    await message.answer("Действие отменено", reply_markup=types.ReplyKeyboardRemove())


async def admin_command(message: types.Message):
    await message.answer("Поздравляю! Эта команда доступна только администратору бота.")


def register_handlers_common(dp: Dispatcher, admin_id: int):
    dp.register_message_handler(cmd_start, commands="start", state="*")
    dp.register_message_handler(cmd_cancel, commands="cancel", state="*")
    dp.register_message_handler(cmd_cancel, Text(equals="отмена", ignore_case=True), state="*")
    dp.register_message_handler(admin_command, IDFilter(user_id=admin_id),
                                commands="Эта команда доступна только админу")
```

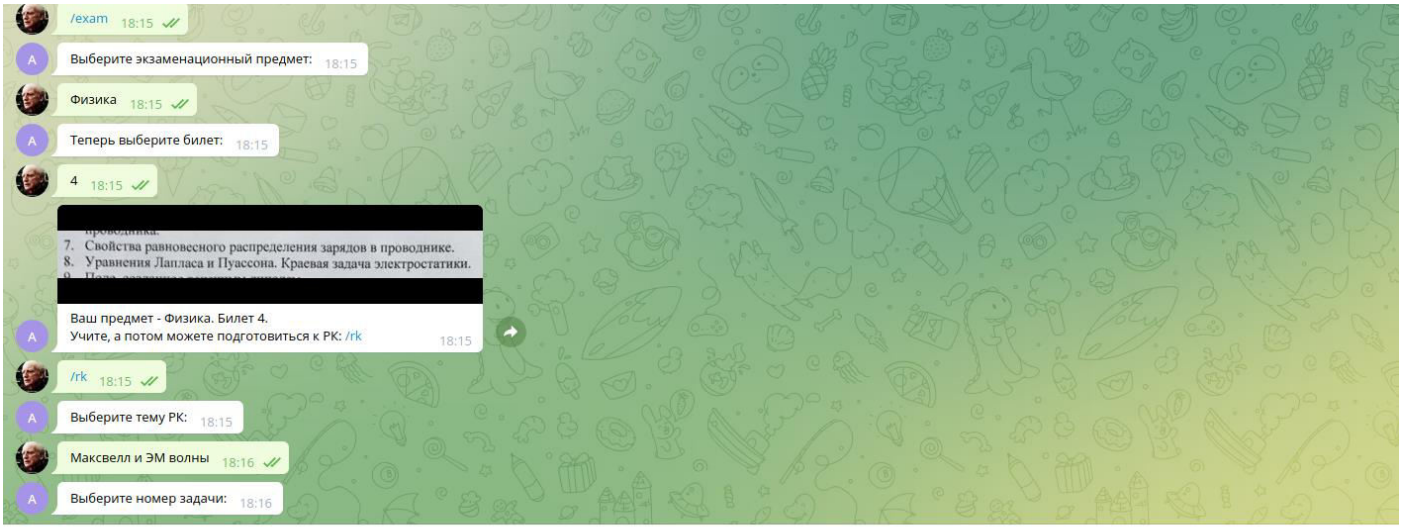
**Текст config.py**

```
import configparser
from dataclasses import dataclass


@dataclass
```

```
class TgBot:
    token: str
    admin_id: int


@dataclass
class Config:
    tg_bot: TgBot


def load_config(path: str):
    config = configparser.ConfigParser()
    config.read(path)

    tg_bot = config["tg_bot"]

    return Config(
        tg_bot=TgBot(
            token=tg_bot["token"],
            admin_id=int(tg_bot["admin_id"])
        )
    )
```

## Результат выполнения



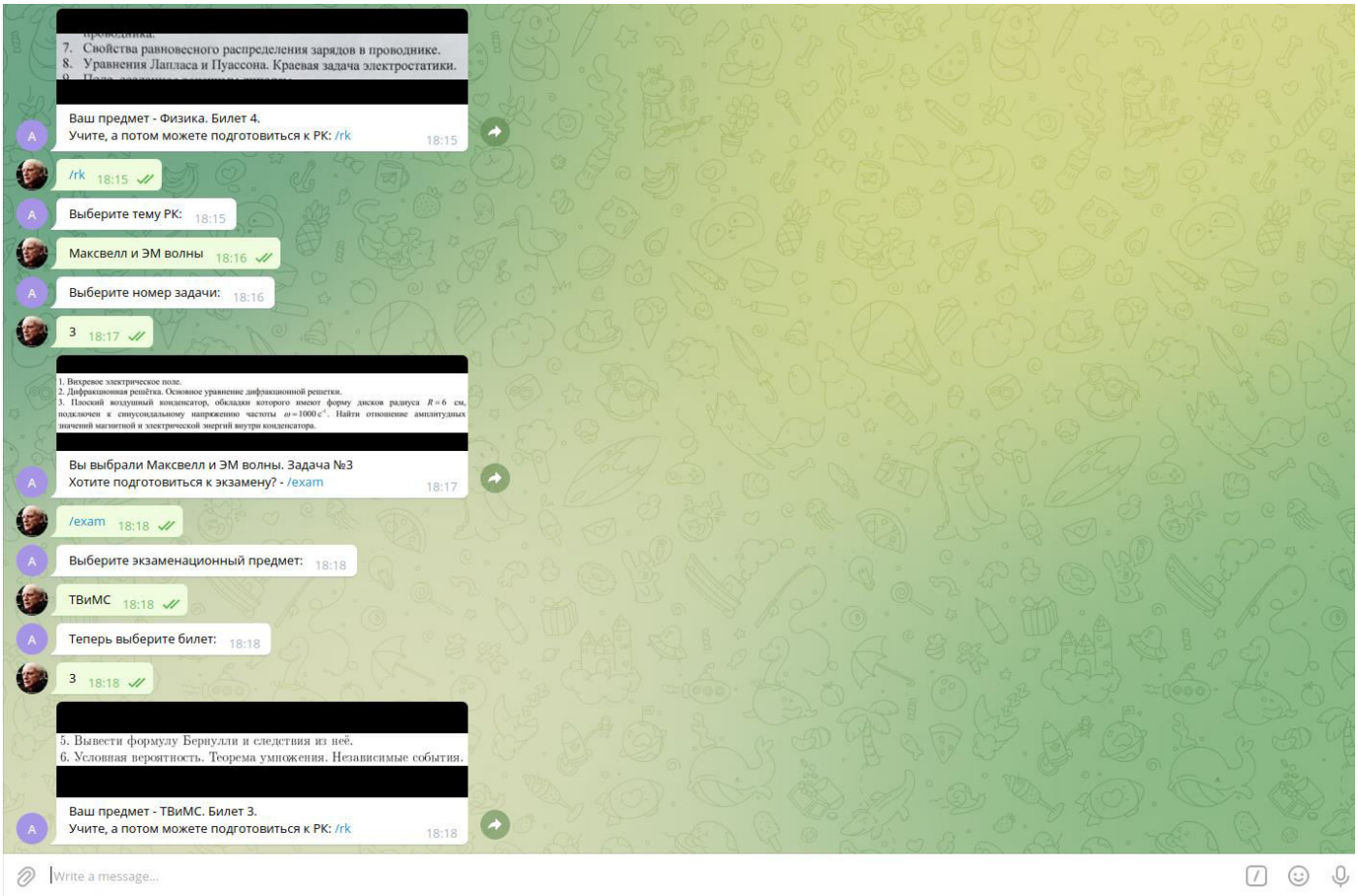

/exam 18:15
Выберите экзаменационный предмет: 18:15
Физика 18:15
Теперь выберите билет: 18:15
4 18:15
7. Свойства равновесного распределения зарядов в проводнике.
8. Уравнения Лапласа и Пуассона. Краевая задача электростатики.
Ваш предмет - Физика. Билет 4.
Учите, а потом можете подготовиться к РК: /rk 18:15
/rk 18:15
Выберите тему РК: 18:15
Максвелл и ЭМ волны 18:16
Выберите номер задачи: 18:16
Write a message...

1

2

3

4


7. Свойства равновесного распределения зарядов в проводнике.
8. Уравнения Лапласа и Пуассона. Краевая задача электростатики.
Ваш предмет - Физика. Билет 4.
Учите, а потом можете подготовиться к РК: /rk 18:15
/rk 18:15
Выберите тему РК: 18:15
Максвелл и ЭМ волны 18:16
Выберите номер задачи: 18:16
3 18:17
1. Вихревое электрическое поле.
2. Дифракционная решётка. Основное уравнение дифракционной решетки.
3. Плоский воздушный конденсатор, обкладки которого имеют форму дисков радиуса $R = 6$ см, подключен к синусоидальному напряжению частоты $\omega = 1000\,c^{-1}$. Найти отношение амплитудных значений магнитной и электрической энергий внутри конденсатора.
Вы выбрали Максвелл и ЭМ волны. Задача №3
Хотите подготовиться к экзамену? - /exam 18:17
/exam 18:18
Выберите экзаменационный предмет: 18:18
ТВиМС 18:18
Теперь выберите билет: 18:18
3 18:18
5. Вывести формулу Бернулли и следствия из неё.
6. Условная вероятность. Теорема умножения. Независимые события.
Ваш предмет - ТВиМС. Билет 3.
Учите, а потом можете подготовиться к РК: /rk 18:18
Write a message...