

ТЕЛЕГРАМ БОТ
СРСН

РАСПРЕДЕЛЕНИЕ ЗАДАЧ

Лукьянов Е. – верстка всего проекта.
Объединение его частей в единый проект.
Сопровождение кода

Серик Т. – работа с логикой команд бота:
отправка файлов, сообщений

Исламберді Ә. – реализация модуля анекдотов

Скрипченко В. – написание документации,
создание презентации, оформления бота



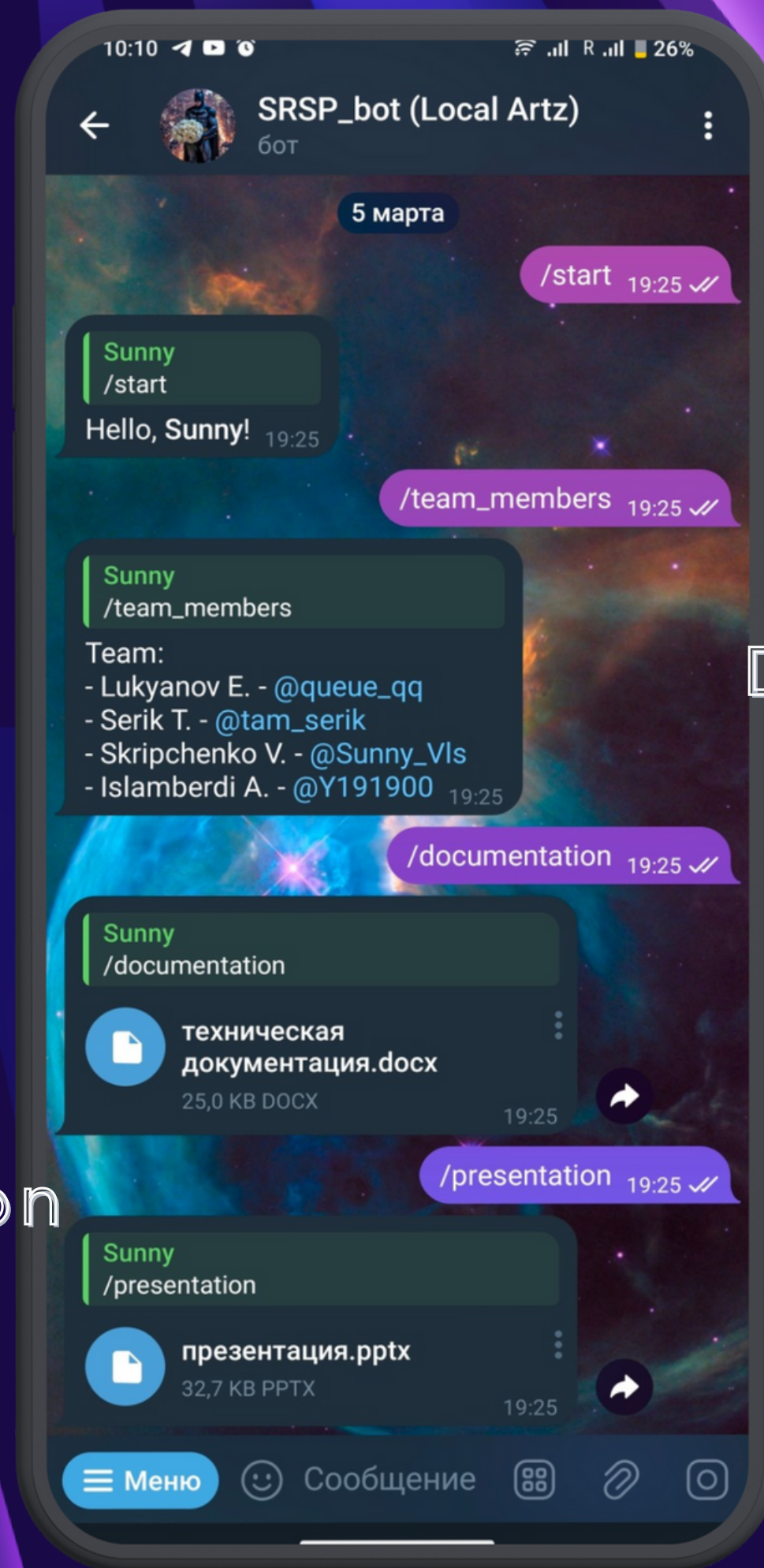
ТЕЛЕГРАМ БОТ SRSP_BOT

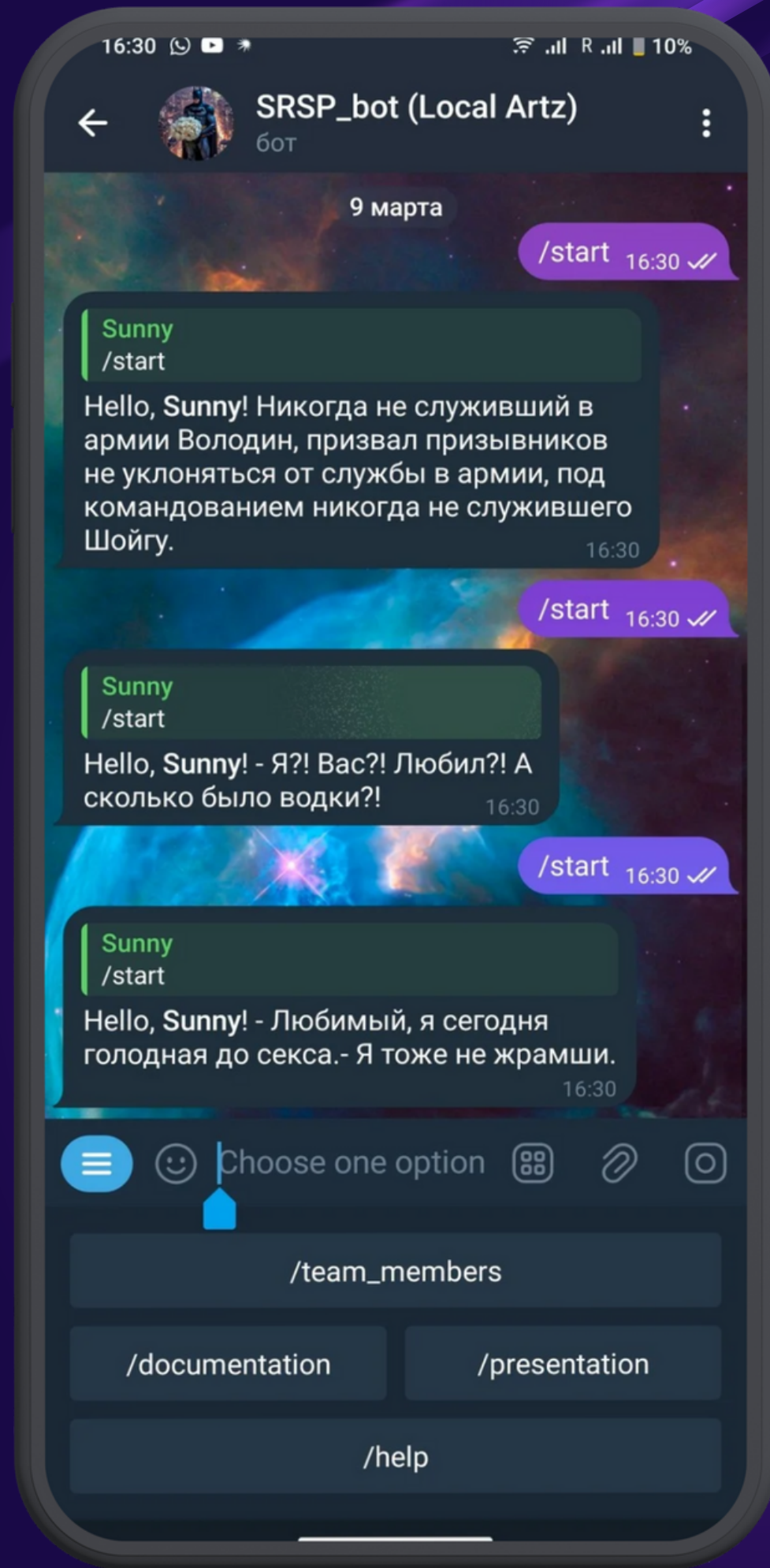
1 Team members

3 Presentation

КНОПКИ

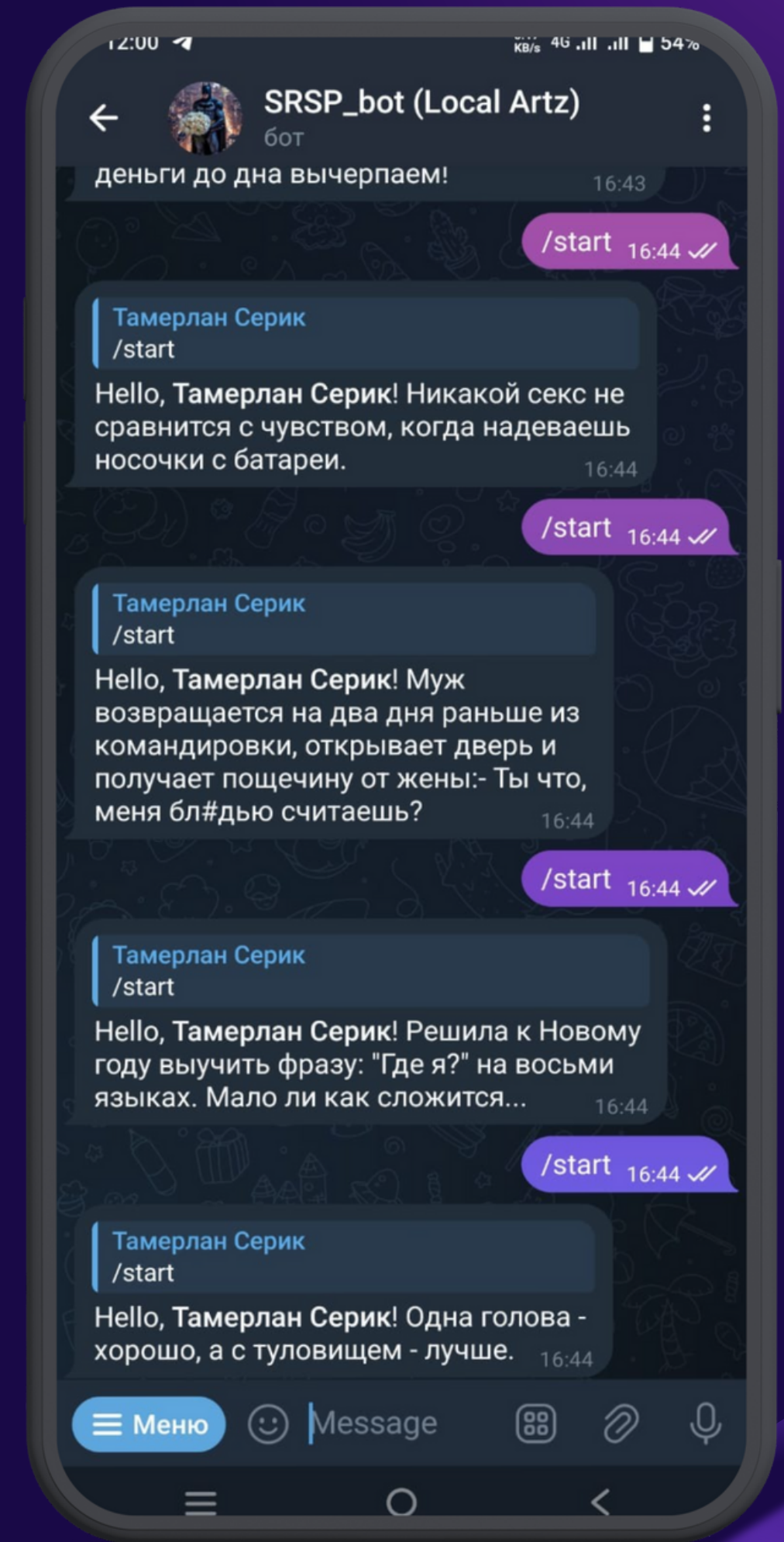
Documentation 2





Также в кнопках
подключен сайт
<https://nekdo.ru/>

При нажатии кнопки
Start, бот приветствует
пользователя и
отправляет ему
рандомный анекдот



ПРОЦЕСС РАЗРАБОТКИ

1

В файле main.py импортируются библиотеки и модули необходимые для дальнейшей работы бота:

```
main.py | Автор – Серик Т.
import asyncio # для асинхронного кода
import logging # для логирования

from aiogram.filters import Command # фильтр обработки команд
from aiogram.enums import ParseMode, ChatType # перечисления режима па
и типов чатов
from aiogram.client.default import DefaultBotProperties # базовая установка
глобальных настроек бота
from aiogram import (
    F, # "магические фильтры". удобны для фильтрации событий
    Bot, # сущность для взаимодействия с Telegram API
    html, # модуль форматирования сообщений в html
    Dispatcher, # компонент для обработки и перенаправления событий
)
from aiogram.types import (
    Message, # класс типа сообщений
    FSInputFile, # класс для отправки файлов из файловой системы
    BotCommand, # сущность команды бота из меню команд
)

import config # импортируется конфиг

from keyboard import main_kb # импортируется разметка клавиатуры

logger = logging.getLogger(__name__) # создается логгер и настраивается
logger.setLevel(logging.DEBUG if config.DEBUG else logging.INFO)

dp = Dispatcher() # создается диспатчер и экземпляр бота
bot = Bot(
    token=config.TOKEN,
    default=DefaultBotProperties(
        parse_mode=ParseMode.HTML
    )
)

async def main() -> None:
    """
    Main function to start the bot
    """

    logging.info("Starting bot")

    dp.message.filter(F.chat.type == ChatType.PRIVATE) # фильтруем диспатчер по
    приватным сообщениям
```

2

Реализуется файл-конфига config.py:

```
from dotenv import load_dotenv # необходимо для удобной выгрузки переменных
среды

if not load_dotenv(): # подгружаем переменные среды из .env-файла
    raise FileNotFoundError('.env file does not exist') # вызываем ошибку при его
отсутствии

TOKEN = os.environ.get('BOT_TOKEN') # задаем значение переменной TOKEN
if not TOKEN:
    raise ValueError("BOT_TOKEN is not set in the environment variables") # вызываем
ошибку при его отсутствии

DEBUG = os.environ.get('DEBUG', 'False').lower() in ('true', '1', 'yes', 'on') # создаем
кортеж синонимов для значения переменной debug.

TEAM_MEMBERS_STR = ""
Team:
- Lukyanov E. - @queue_qq
- Serik T. - @tam_serik
- Skripchenko V. - @Sunny_VIs
```

ПРОЦЕСС РАЗРАБОТКИ



Алижан реализует модуль с анекдотами

```
tgbotirsp > Modules > getAnecdote.py > getAnecdote
1 import asyncio
2 import aiohttp
3 from bs4 import BeautifulSoup
4
5 import random
6 import logging
7
8 logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
9
10 URL = 'https://nekdo.ru/ndom/'
11 HEADERS = {
12     "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.77 Safari/537.36"
13 }
14
15 > JOKES_list = [ ...
28
29 def findInSoup(soup: BeautifulSoup):
30     element = soup.find('div', id=True, class_=True)
31     return element
32
33 async def getAnecdote() -> str:
34     logging.info("Запрашиваю анекдот с сайта.")
35     async with aiohttp.ClientSession(headers=HEADERS) as session:
36         try:
37             async with session.get(URL) as response:
38                 if response.status == 200:
39                     logging.info("Успешно получен ответ от сервера.")
40                     html = await response.text()
41                     soup = BeautifulSoup(html, 'html.parser')
42
43                     element = findInSoup(soup)
44                     if element:
45                         logging.info("Анекдот успешно найден на странице.")
46                         return element.text.strip()
47
48                     logging.warning(f"Получен неудачный статус ответа: {response.status}. Использую локальный анекдот.")
49                     return random.choice(JOKES_list)
```



Совместными усилиями Серика Т. и Лукьянова Е. были реализованы функции обработки команд

```
.main.py | Авторы – Лукьянов Е., Серик Т.
@dp.message(Command("start"))
async def command_start_handler(message: Message) -> None:
    """
    Handler for the '/start' command
    """
    await message.reply(
        text=f"Hello, {html.bold(message.from_user.full_name)}!",
        reply_markup=main_kb
    )

@dp.message(Command("team_members"))
async def command_of_devers(message: Message):
    """
    Handler for the '/team_members' command
    """
    await message.reply(text=config.TEAM_MEMBERS_STR)

@dp.message(Command("help"))
async def cmd_help(message: Message):
    """
    Handler for the '/help' command
    """
    await message.reply(
        text="Вот список команд:\n/start - начать работу\n/help - помощь"
    )

@dp.message(Command("documentation"))
async def cmd_doc(message: Message):
    """
    Handler for the '/documentation' command
    """
    await message.reply_document(
        document=FSinputFile(path="assets/tech_doc.docx", filename="техническая документация.docx")
    )

@dp.message(Command("presentation"))
async def cmd_presentation(message: Message):
    """
    """
```