**Департамент образования и науки города Москвы**

**ГБОУ «Школа Глория»**

**БОТ ДЛЯ АНАЛИЗА ЭМОЦИОНАЛЬНОГО СОСТОЯНИЯ**

Участник:

ученик 10 «А» класса ГБОУ

«Школа Глория»

Бердников Артем Валерьевич

Руководители:

педагог ГБОУ «Школа Глория»

Зацепина Маргарита Викторовна

педагог технопарка «РГСУ»

Ковалёв Ягуар Игоревич

**Москва, 2025**

**Оглавление**

[**Методологический паспорт** 4](#_Toc189304754)

[**Теоретическая часть** 6](#_Toc189304755)

[**Что такое Telegram-боты и в чём принцип их работы?** 6](#_Toc189304756)

[**Для чего нужны Telegram-боты?** 6](#_Toc189304757)

[**Как работают Telegram-боты?** 7](#_Toc189304758)

[**В заключение:** 7](#_Toc189304759)

[**Обзор аналогов** 8](#_Toc189304760)

[**1.** **Мобильное приложение «Daylio»:** 8](#_Toc189304761)

[**2.** **Мобильное приложение «Moodfit»:** 8](#_Toc189304762)

[**Вывод:** 9](#_Toc189304763)

[**Выбор среды и технологий разработки** 10](#_Toc189304764)

[**Язык программирования – Python** 10](#_Toc189304765)

[Б**иблиотека для создания Telegram-бота – aiogram** 10](#_Toc189304766)

[**Система управления базой данных – PostgreSQL** 10](#_Toc189304767)

[**Практическая часть:** 12](#_Toc189304768)

[**Разработка концепта** 12](#_Toc189304769)

[**Статистика и метрика** 18](#_Toc189304770)

[**1.** **Взвешенное среднее настроение** 18](#_Toc189304771)

[**2.** **Тренд настроения** 18](#_Toc189304772)

[**3.** **Эмоциональная стабильность** 18](#_Toc189304773)

[**4.** **График** 19](#_Toc189304774)

[**Разработка Telegram-бота** 19](#_Toc189304775)

[**Тестирование** 27](#_Toc189304776)

[**Заключение:** 29](#_Toc189304777)

[**Перспектива развития проекта:** 29](#_Toc189304778)

# **Методологический паспорт**

Цель: Создать удобного и простого в использовании Telegram-бота, который будет проводить ежедневные опросы, собирать данные о настроении пользователей и помогать в поддержании эмоционального равновесия.

Предмет: Программирование.

Объект: Telegram-боты.

Актуальность: в современном мире человек живёт в постоянном напряжении и стрессе. Все большую популярность набирают боты, помогающие пользователям в заботе о себе. Отслеживание и запись состояния за день, а также анализ его изменения, наведёт человека на мысли: “что я сделал сегодня такого, из-за чего мое состояние хуже, чем вчера?” или “чем я занимался, что я чувствую себя лучше, чем на прошлой неделе?”. Таким образом, бот поможет выявлять закономерности своего хорошего самочувствия и пребывать в нем чаще.

Задачи:

1. Изучить техническую литературу по теме, выбрать технологии и среду разработки и ознакомиться с основными концепциями создания Telegram-бота, его возможным функционалом, программными решениями.

2. Спроектировать чат-бота для отслеживания эмоционального состояния, в котором пользователь тратил бы наименьшее время из возможного для прохождения опросов.

3.  Разработать и протестировать Telegram-бот.

Методы:

Анализ, синтез, программирование и проектирование баз данных.

Сроки выполнения:

1. Теоретическая часть: 20.11.2024 – 24.12.2024
2. Проектирование и разработка концепта: 25.12.2024 – 21.01.2025
3. Разработка и тестирование Telegram-бота: 22.01.2025 – 05.02.2025

# **Теоретическая часть**

## **Что такое Telegram-боты и в чём принцип их работы?**

Telegram-бот — Специальные аккаунты в Telegram, созданные для того, чтобы автоматически обрабатывать и отправлять сообщения. Пользователи могут взаимодействовать с ботами при помощи сообщений, отправляемых через обычные или групповые чаты. Telegram-бот умеет делать всё, что мог бы делать человек в чате: отвечать на вопросы, присылать файлы или ссылки на сайты.

## **Для чего нужны Telegram-боты?**

Боты могут быть очень полезны как для обычных пользователей, так и для бизнеса. Вот несколько примеров их применения:

* Развлечения: Боты могут отправлять шутки, загадывать загадки, проводить викторины и игры.
* Информация: Боты могут предоставлять новости, прогноз погоды, курсы валют, расписание транспорта и многое другое.
* Помощь: Боты могут помогать с поиском информации, переводом текста, конвертацией файлов, напоминать о событиях и т.д.
* Бизнес: Боты могут принимать заказы, отвечать на вопросы клиентов, проводить опросы, рассылать уведомления и выполнять другие задачи, автоматизируя работу компаний.
* Обучение: Боты могут помогать в изучении языков, предлагать обучающие материалы, проверять знания.
* Здоровье и самочувствие: Сюда относятся боты, помогающие отслеживать физическую активность, напоминать о приёме лекарств, предоставлять медитативный контент.

## **Как работают Telegram-боты?**

Принцип работы бота сравним с официантом в ресторане. Пользователь (посетитель) отправляет боту сообщение с запросом (заказом). Бот обрабатывает этот запрос и отправляет ответ (приносит блюдо). Технически это работает так:

1. Пользователь отправляет сообщение боту. Это сообщение может быть текстом, командой (например, /start), изображением или другим типом данных.
2. Telegram передает это сообщение на сервер, где "живет" бот.
3. Программа бота обрабатывает сообщение. Она анализирует запрос пользователя и определяет, что нужно сделать.
4. Бот формирует ответ и отправляет его обратно в Telegram.
5. Telegram доставляет ответ пользователю.

Важно понимать:

* Боты не имеют доступа к вашим личным сообщениям в Telegram. Они видят только те сообщения, которые вы отправляете непосредственно им.
* Боты не могут отправлять сообщения пользователям, пока пользователь не инициировал первый диалог.
* Все боты работают по правилам и алгоритмам, заданным их разработчиками.

## **В заключение:**

Telegram-боты — это мощный инструмент, который может значительно упростить жизнь и автоматизировать многие процессы. Благодаря своей простоте и гибкости, боты становятся все более популярными, в том числе и в сфере заботы о себе.

# **Обзор аналогов**

Очевидно, что уже существуют способы отслеживания своего эмоционального состояния. Разберем несколько популярных трекеров настроения Moodfit и Daylio, с целью выявить преимущества и недостатки, чтобы учитывать их при создании своего бота для отслеживания настроения

1. **Мобильное приложение «Daylio»:**

Это приложение в Google Play имеет больше 10 млн скачиваний, 400 тыс. отзывов и среднюю оценку 4,8 из 5. Оно действительно предоставляет обширный функционал: кроме записи своего настроения, можно отмечать, какими активностями вы сегодня занимались, давать личные комментарии по прожитому дню, оценивать свой сегодняшний сон, и все это путем прохождения ежедневных опросов. После этого можно в удобном формате просматривать статистику и динамику изменений, ставя для себя новые цели прямо в приложении.

Но за всем этим скрывается несколько минусов: в приложении есть реклама, а также для использования полного функционала Daylio и отключения рекламы нужно приобрести премиум подписку. Главный недостаток в установке отдельного приложения, что в разы неудобнее, чем просто запустить бота в уже установленном мессенджере. Гораздо лучше, когда все нужное собрано в одном приложении.

1. **Мобильное приложение «Moodfit»:**

Moodfit – одно из популярных приложений для отслеживания своего состояния. Оно разработано в помощь тем, кто хочет стать более осознанным, следить за своими эмоциями и управлять стрессом и тревожностью. В число функций приложения входят дневник настроений и благодарности с чартами для эмоций и рефлексии о том, за что вы можете быть благодарны. Также есть дополнительные функции, в том числе: отслеживание сна, питания и физической активности; инструменты для осознанности, дыхательные упражнения и упражнения для заземления; персонализированные отчеты, помогающие найти закономерности. Но все это лишь в премиум подписке. Без нее доступен лишь дневник настроений.

Как и с Daylio, минусом является необходимость установки отдельного приложения. Но кроме этого и уже упомянутой премиум подписки, в приложении не поддерживается русский язык.

## **Вывод:**

* + Стоит разработать и подобрать несколько метрик для статистики.
  + Для использования бота не требуется установка отдельного приложения, достаточно будет приложения Telegram.
  + Использование бота должно быть бесплатным.
  + Должен поддерживаться русский язык.

# **Выбор среды и технологий разработки**

## **Язык программирования – Python**

* **Преимущества:**
* Простота и читаемость синтаксиса, что ускоряет разработку.
* Широкое сообщество и множество библиотек для работы с API и базами данных.
* Хорошая поддержка асинхронного программирования, что важно для ботов.
* **Альтернативы:**
* JavaScript (Node.js): также популярен для разработки ботов, но может быть сложнее для новичков.
* Go: предлагает высокую производительность, но требует больше времени на изучение

## Б**иблиотека для создания Telegram-бота – aiogram**

* **Преимущества:**
* Асинхронная работа, что позволяет обрабатывать множество запросов одновременно.
* Простота использования и хорошая документация.
* Поддержка всех возможностей Telegram Bot API.
* **Альтернативы:**
* python-telegram-bot: более распространенная библиотека, но менее оптимизирована для асинхронного программирования.
* Telepot: менее популярная, но простая в использовании.

## **Система управления базой данных – PostgreSQL**

* **Преимущества:**
* Высокая производительность и надежность.
* Поддержка сложных запросов и транзакций.
* Хорошая интеграция с Python через библиотеки (например, psycopg2).
* **Альтернативы:**
* MySQL: популярная СУБД, но может быть менее удобна для работы с сложными данными.
* SQLite: легкая СУБД, но не подходит для масштабируемых решений.

Стоит упомянуть, что для развертывания будет использоваться Docker, что обеспечивает простоту настройки и удобство работы.

# **Практическая часть:**

## **Разработка концепта**

При запуске бот здоровается и предлагает пользователю зарегистрироваться и выбрать время, в которое он будет получать напоминание о ежедневных опросах. Для удобства и экономии времени пользователя, бот отправит сообщение с привязанной к нему клавиатурой, состоящей из кнопок, с помощью которых можно быстро выбрать время. После подтверждения в базу данных добавляется новый пользователь. Теперь каждый день в выбранное время пользователю будет предлагаться прохождение опроса.

Опросы максимально упрощены: в нем пользователь должен будет выбрать 1 из 5 смайликов, олицетворяющие 5 степеней эмоционального состояния от «очень плохо» до «очень хорошо». После прохождения, результат записывается в базу данных и привязывается к определенному пользователю.

В конце каждой недели пользователю будет направлена статистика прохождения опросов в течение этой недели. Так или иначе, пользователь может просмотреть свою актуальную недельную статистику в любой момент, воспользовавшись командой. Статистика основывается на нескольких метриках, которые будут описаны дальше.

**Описание диаграмм IDEF0**

Для более подробного понимания функционирования бота, необходимо определить бизнес-процесс, который будет его использовать. Был выбран бизнес-процесс «Проведение анализа эмоционального состояния».

IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Контекстная диаграмма IDEF0 для бизнес-процесса «Реализация услуг» представлена на рисунке 1.

|  |
| --- |
|  |

Рис. 1. Контекстная диаграмма «Проведение анализа эмоционального состояния»

Суть бизнес-процесса заключается в том, чтобы бот собирал данные о настроении и состоянии пользователя, после чего предоставлял ему статистику и мотивацию поддерживать свое эмоциональное благополучие.

Основными входными данными процесса являются:

* Запрос пользователя.
* График проведения опросов.
* Данные пользователя.

На выходе формируются:

* Пройденный опрос.
* Результат с подбадривающим сообщением.
* Недельные результаты.

Контроль над процессом осуществляется через алгоритм анализа, ФЗ о защите прав потребителей и пользовательское соглашение Telegram. Реализация услуг поддерживается программным обеспечением и необходимым оборудованием, то есть с помощью базы данных, Telegram и сервера. Для подробного понимания работы данного процесса необходимо создать декомпозицию вышеописанной диаграммы, которую можно посмотреть на рисунке 2.

|  |
| --- |
|  |

Рис. 2. Декомпозиции процесса «Проведение анализа эмоционального состояния»

Декомпозиция составлена из пяти основных процессов:

* Регистрация пользователя (A1) — сбор всех необходимых данных и добавление пользователя в базу данных.
* Составление графика опросов (A2) — пользователь выбирает время, в которое ему будут присылаться опросы.
* Проведение опроса (A3) — пользователь проходит опрос, выбирая 1 из 5 смайликов.
* Анализ опросов (A4) — получение данных об уже пройденных опросах пользователя и подведение статистики.
* Отправка результатов пользователю (A5) — отправка недельных результатов с подбадривающими фразами и статистики пользователю.

Первый этап, «Регистрация пользователя», предполагает обработку данных пользователя и запись пользователя в базу данных. На этом этапе используются контактные данные клиента.

Второй этап, «Составление графика опросов», включает сбор времени, введенное пользователем и добавление этого времени в базу данных к уже зарегистрированному пользователю.

Третий этап, «Проведение опросов», подразумевает сбор данных о настроении пользователя путём прохождения опроса. Выбранный пользователем результат записывается в базу данных.

Четвертый этап, «Анализ опросов», предполагает запрос ответов пользователя в пройденных опросах за неделю с базы данных и подведение статистики по заданным метрикам с помощью алгоритма анализа.

Заключительный этап, «Отправка результатов пользователю», подразумевает составление сообщения, включающее в себя уже подготовленный анализ и мотивирующие фразы, после чего составленное сообщение отправляется пользователю.

Диаграммы IDEF0 позволяют наглядно показать структуру процесса «Проведение анализа эмоционального состояния. Использование данной методологии способствует упрощению описания процессов, выделению их ключевых элементов и установлению логических связей между этапами. Это, в свою очередь, повышает эффективность управления проектами и позволяет минимизировать возможные ошибки на всех уровнях выполнения работ.

На основе полученных требований были построены 2 ERD-диаграммы (Entity-Relationship Diagram) структуры базы данных информационной системы, логическая (в нотации Чена) и физическая (в нотации Мартина), которые представлены на рисунках 3 и 4 соответственно.

|  |
| --- |
|  |

Рис. 3. Логическая ERD-диаграмма в нотации Чена

|  |
| --- |
|  |

Рис. 4. Физическая ERD-диаграмма в нотации Мартина

## **Статистика и метрика**

1. **Взвешенное среднее настроение**

Среднее взвешенное — это среднее арифметическое значение, в котором учтён вес каждого из слагаемых, для которых рассчитывается это среднее значение. Другими словами, каждому исходному показателю присваивается определённый вес.

В случае с настроением, ответы, которые даны менее чем 2 дня назад, имеют вес 1,5. Более давние ответы имеют вес 1.

Расчет происходит таким образом:

1. Присваивается вес каждому опросу за неделю.
2. Каждый результат опроса умножается на свой вес.
3. Суммируются взвешенные значения.
4. Полученная сумма делится на общий вес.

Далее среднее взвешенное округляется и в соответствие к нему ставится 1 из 5 результатов прохождения опроса. Это и является взвешенным средним настроением.

1. **Тренд настроения**

Данная метрика характеризует тенденцию изменения настроения за неделю: положительная, отрицательная или стабильная. Если среднее арифметическое результатов опроса за 2 половину недели больше, чем за 1 половину, значит динамика положительная. Если они равны, то стабильная. В ином случае динамика отрицательна.

1. **Эмоциональная стабильность**

Подсчет эмоциональной стабильности основана на подсчете среднеквадратического отклонения числового набора, состоящего из результатов опроса пользователя за неделю.

Среднеквадратичное отклонение — характеристика, использующаяся в статистике для измерения степени изменчивости или разброса данных. Обычно определяется как квадратный корень из дисперсии случайной величины. Чем меньше среднеквадратическое отклонение, тем ближе значения к среднему. Чем больше среднеквадратическое отклонение, тем больше разброс данных.

Таким образом, среднеквадратическое отклонение является идеальным вариантом для характеристики стабильности.

1. **График**

Важной частью статистического отчета является наглядное представление информации. Для этого по итогам результатов за неделю создается линейный график: вертикальная ось – результаты опроса, горизонтальная – время прохождения опроса.

Визуализация данных способствует лучшему пониманию информации и её запоминанию. С помощью линейного графика, пользователь может намного легче воспринять динамику изменений своего состояния и на основе этого сделать свои выводы.

## **Разработка Telegram-бота**

Разработка каждого Telegram-бота начинается с обращения к официальному боту Telegram, который называется BotFather. С помощью него создаются все новые пользовательские боты. После создания выдается уникальный токен, с помощью которого будет возможно написать код конкретно для этого новосозданного бота.

Далее будет представлено несколько основных программных решений. С полным кодом можно ознакомиться здесь (открыть репозиторий и добавить ссылку)

Создание базы данных

|  |
| --- |
| # Настройка логирования  logging.basicConfig(level=logging.INFO)  logger = logging.getLogger(\_\_name\_\_)  # SQL-запросы для создания таблиц  CREATE\_USERS\_TABLE = """  CREATE TABLE IF NOT EXISTS users (  id BIGINT PRIMARY KEY,  notification\_time TIME NOT NULL,  created\_at TIMESTAMP DEFAULT NOW(),  updated\_at TIMESTAMP DEFAULT NOW()  );  """  CREATE\_SURVEY\_RESULTS\_TABLE = """  CREATE TABLE IF NOT EXISTS survey\_results (  id SERIAL PRIMARY KEY,  user\_id BIGINT NOT NULL REFERENCES users (id),  answer INTEGER NOT NULL,  created\_at TIMESTAMP DEFAULT NOW()  );  """  async def setup\_database():  conn = await connect(config.DATABASE\_URL)  try:  await conn.execute(CREATE\_USERS\_TABLE)  await conn.execute(CREATE\_SURVEY\_RESULTS\_TABLE)  logger.info("Таблицы созданы.")  except Exception as e:  logger.error(f"Ошибка при создании таблиц: {e}")  finally:  await conn.close()  async def get\_database\_connection():  logger.info("Получение соединения с базой данных")  conn = await connect(config.DATABASE\_URL)  return conn |

project/app/database.py

Проведение опроса

|  |
| --- |
| from app.database import get\_database\_connection  from app.handlers.start import get\_main\_keyboard  # Настройка логирования  logger = logging.getLogger(\_\_name\_\_)  class SurveyState(StatesGroup):  in\_progress = State()  BUTTON\_TEXTS = {  1: "😭",  2: "🙁",  3: "😐",  4: "🙂",  5: "🤩",  }  keyboard = ReplyKeyboardMarkup(  keyboard=[  [KeyboardButton(text=BUTTON\_TEXTS[1]),  KeyboardButton(text=BUTTON\_TEXTS[2]),  KeyboardButton(text=BUTTON\_TEXTS[3]),  KeyboardButton(text=BUTTON\_TEXTS[4]),  KeyboardButton(text=BUTTON\_TEXTS[5])],  ],  resize\_keyboard=True,  )  # Функция для отправки опроса  async def survey\_handler(message: types.Message, state: FSMContext):  user\_id = message.from\_user.id  logger.info(f"Запуск опроса для пользователя {user\_id}")  await message.reply(  "Как вы себя чувствуете сегодня? Выберите смайлик:", reply\_markup=keyboard  )  await state.set\_state(SurveyState.in\_progress)  logger.info(f"Состояние установлено для пользователя {user\_id}")  # Обработчик нажатия кнопки  async def handle\_survey\_response(message: types.Message, state: FSMContext):  logger.info("Обработчик handle\_survey\_response вызван")  # Сохраняем результат в базе данных  mood = None  if message.text == BUTTON\_TEXTS[1]:  mood = 1  elif message.text == BUTTON\_TEXTS[2]:  mood = 2  elif message.text == BUTTON\_TEXTS[3]:  mood = 3  elif message.text == BUTTON\_TEXTS[4]:  mood = 4  elif message.text == BUTTON\_TEXTS[5]:  mood = 5  logger.info(f"Определённое настроение: {mood}")  if mood:  try:  db = await get\_database\_connection()  user\_id = message.from\_user.id  user\_exists = await db.fetchval(  """  SELECT id FROM users WHERE id = $1  """,  user\_id,  )  if not user\_exists:  await message.reply(  "Вы не зарегистрированы. Пожалуйста, зарегистрируйтесь с помощью команды /register."  )  return # Прерываем выполнение функции, если пользователь не зарегистрирован  # Сохраняем настроение в базу данных как новую запись  logger.info(f"Сохранение настроения пользователя {user\_id}: {mood}")  await db.execute(  """  INSERT INTO survey\_results (user\_id, answer)  VALUES ($1, $2)  """,  user\_id,  mood,  )  await message.reply(  f"Спасибо за ваш ответ! Ваше настроение: {BUTTON\_TEXTS[mood]}",  reply\_markup=get\_main\_keyboard(), # Отправляем клавиатуру с командами  )  # Завершаем опрос, удаляем состояние  await state.clear()  logger.info(f"Состояние очищено для пользователя {user\_id}")  except Exception as e:  logger.error(f"Ошибка при сохранении в БД: {e}")  await message.reply(  "Произошла ошибка при сохранении вашего ответа. Пожалуйста, попробуйте позже.",  reply\_markup=get\_main\_keyboard(), # Отправляем клавиатуру с командами  )  else:  logger.warning(  f"Пользователь {message.from\_user.id} выбрал неверный вариант: {message.text}"  )  await message.reply(  "Пожалуйста, выберите один из предложенных смайликов.",  reply\_markup=keyboard, # Отправляем клавиатуру со смайликами  )  # Регистрируем хэндлеры  def register\_handlers(dp: Dispatcher):  dp.message.register(survey\_handler, Command("survey"))  dp.message.register(handle\_survey\_response, SurveyState.in\_progress)  dp.message.register(send\_stats, Command("stats")) |

project/app/handlers/survey.py

Рассылка ежедневного опроса и недельных результатов

|  |
| --- |
| from app.database import get\_database\_connection  from app.handlers.survey import BUTTON\_TEXTS # Импортируем словарь с текстами кнопок  scheduler = AsyncIOScheduler()  async def send\_daily\_survey():  db = await get\_database\_connection()  try:  # Получаем всех пользователей и их время уведомлений  users = await db.fetch("SELECT id, notification\_time FROM users")  for user in users:  user\_id = user["id"]  notification\_time = user["notification\_time"]  # Проверяем, что текущее время совпадает с временем уведомления  now = datetime.now().time()  if (  now.hour == notification\_time.hour  and now.minute == notification\_time.minute  ):  await bot.send\_message(  user\_id, "Самое время пройти ежедневный опрос!\nИспользуйте команду /survey",  )  except Exception as e:  print(f"Ошибка при отправке ежедневного опроса: {e}")  finally:  await db.close()  async def send\_weekly\_statistics():  db = await get\_database\_connection()  try:  # Получаем всех пользователей  users = await db.fetch("SELECT id FROM users")  for user in users:  user\_id = user["id"]  # Получаем записи за последнюю неделю  results = await db.fetch(  """  SELECT answer, created\_at FROM survey\_results  WHERE user\_id = $1 AND created\_at >= NOW() - INTERVAL '7 days'  ORDER BY created\_at  """,  user\_id,  )  if results:  # Формируем статистику с текстами кнопок  stats = "\n".join(  [  f"{row['created\_at']}: {BUTTON\_TEXTS[row['answer']]}"  for row in results  ]  )  average\_mood = sum(row["answer"] for row in results) / len(results)  await bot.send\_message(  user\_id,  f"Ваша статистика за последнюю неделю:\n{stats}\n\nСреднее настроение: {BUTTON\_TEXTS[round(average\_mood)]}",  )  else:  await bot.send\_message(  user\_id, "У вас нет записей за последнюю неделю."  )  except Exception as e:  print(f"Ошибка при отправке статистики: {e}")  finally:  await db.close()  def setup\_scheduler():  # Ежедневный опрос (проверка каждую минуту)  scheduler.add\_job(send\_daily\_survey, "cron", minute="\*") # Проверка каждую минуту  # Еженедельная статистика  scheduler.add\_job(  send\_weekly\_statistics, "cron", day\_of\_week="sun", hour=10  ) # Отправка каждое воскресенье в 10 утра  scheduler.start() |

project/app/scheduler.py

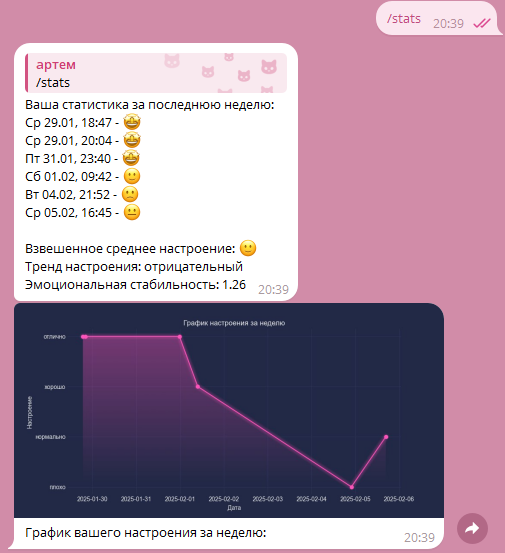
Создание линейного графика (при помощи библиотеки matplotlib)

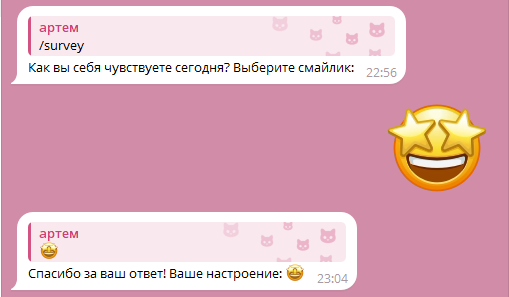
|  |
| --- |
| async def create\_mood\_chart(results):  # Создает график настроения за последнюю неделю.  if not results:  return None  # Подготовка данных  dates = [row["created\_at"] for row in results]  moods = [row["answer"] for row in results]  # Создание графика  plt.figure(figsize=(10, 5))  plt.plot(dates, moods, marker="o", linestyle="-", color="b")  plt.title("График настроения за неделю")  plt.xlabel("Дата")  plt.ylabel("Настроение")  plt.yticks(range(1, 6), [BUTTON\_TEXTS[i] for i in range(1, 6)]) # Подписи для оси Y  plt.grid(True)  # Сохранение графика в байты  buf = io.BytesIO()  plt.savefig(buf, format="png")  buf.seek(0)  plt.close()  return buf |

project/app/handlers/survey.py

## **Тестирование**

На рисунках 5, 6, 7 в качестве тестирования представлена демонстрация работы бота.





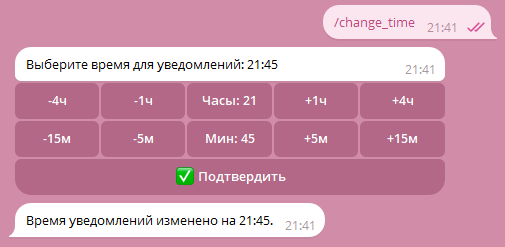


Рис. 5, 6, 7. Демонстрация работы бота.

Во время тестирования бота была обнаружена проблема, связанная с часовыми поясами. Изначально, в боте была установка на московский часовой пояс UTC+3. Это ограничивает зону работы бота, ведь пользователи с другим часовым поясом будут получать уведомления об опросах по московскому времени, а не по местному.

Проблема была решена при помощи 2 Python библиотек: tzwhere и geopy. Первая библиотека нужна для определения часового пояса по введенным координатам, а вторая библиотека нужна для получения координат по названию города.

Теперь при регистрации пользователь вводит название своего города. Если такой город не найден, то предлагается ввести это название на латинице, либо ввести другой, более крупный населенный пункт по близости.

(>> скрины)

# **Заключение:**

1. Изучена техническая литература по теме и основными концепции создания Telegram-ботов на Python.

2. Спроектирован концепт чат-бота для отслеживания эмоционального состояния, в котором пользователь может удобно и без лишней траты времени проходить ежедневные опросы.

3.  Разработан и протестирован сам Telegram-бот.

Цель проекта достигнута, все задачи выполнены

# **Перспектива развития проекта:**

1. Поддержка нескольких языков.
2. Комментарии и заметки как часть прохождения опроса. В недельных результатах нейросеть соберет все комментарии в один текст, «сожмет» его и направит пользователю.
3. Нейросеть, которая будет общаться с пользователем, поддерживать и давать мотивацию. Появится возможность рассказать о наболевшем или о своих успехах.
4. Дополнительный функционал: отслеживание сна, питания, физической активности, установка целей и так далее.

**Ссылки:**

1. Байкова, К. Д. АНАЛИЗ И РАЗРАБОТКА ФУНКЦИОНАЛЬНОГО TELEGRAM-БОТА / К. Д. Байкова, П. А. Медведева // Молодой исследователь Дона. – 2021. – № 6. – С. 15-20. – ISSN 2500-1779
2. Какие бывают Телеграм‑боты и для чего они нужны : сайт. – URL: <https://practicum.yandex.ru/blog/telegram-boty-kak-rabotayut-i-kak-nastroit/> (дата обращения: 12.01.2025)
3. Курс "Телеграм-боты на Python и AIOgram" : сайт. – URL: <https://stepik.org/120924> (дата обращения: 22.01.2025)
4. Любанович, Б. Простой Python. Современный стиль программирования. / Б. Любанович. – Санкт-Петербург : Питер, 2016. – 477 с. – ISBN 978-5-496-02088-6.
5. О персональных данных : Федеральный закон от 27.07.2006 N 152-ФЗ // Собрание законодательства РФ. – 2006. – № 31, ч. 1. – Ст. 3451.
6. Часовой пояс в ботах для Telegram : сайт. – URL: <https://4xpro.ru/profblog/telegram-timezones>/ (дата обращения: 02.02.2025)
7. Что такое бот в Telegram: виды и функции : сайт. – URL: <https://gb.ru/blog/chto-takoe-telegram/> (дата обращения: 21.11.2024)
8. Шумилина, М. А. РАЗРАБОТКА ЧАТ-БОТА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON В МЕССЕНДЖЕРЕ «TELEGRAM» / М. А. Шумилина, А. В. Коробко // Научные известия. – 2022. – № 28. – С. 47-54. – ISSN 2413–6492
9. aiogram : сайт. – URL: <https://aiogram.dev/> (дата обращения: 19.12.2024)
10. BotFather : сайт. – URL: <https://telegram.me/BotFather> (дата обращения: 22.01.2025)
11. docker : сайт. – URL: <https://docs.docker.com> (дата обращения: 20.12.2024)
12. matplotlib : сайт. – URL: <https://matplotlib.org/stable/index.html> (дата обращения: 29.01.2025)
13. Python : сайт. – URL: <https://www.python.org/> (дата обращения: 19.12.2024)