МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Балтийский федеральный университет имени Иммануила Канта»

ОНК «Институт высоких технологий»

ОТЧЁТ О ПРОХОЖДЕНИИ

УЧЕБНОЙ ТЕХНОЛОГИЧЕСКОЙ (ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ) ПРАКТИКИ

на базе Высшей школы компьютерных наук и искусственного интеллекта

Выполнил Римша Николай Сергеевич

студент очной формы обучения 1 курса

направления подготовки 01.03.02 Прикладная математика и информатика

профиль обучения «Информатика и программирование»

Руководитель практики

Ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тарачков М.В.

г. Калининград 2024 г.

Оглавление

[ВВЕДЕНИЕ 3](#_Toc169703047)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 4](#_Toc169703048)

[Телеграмм боты и как они работают 4](#_Toc169703049)

Асинхронное программирование [4](#_Toc169703049)

[Библиотека Aiogram 4](#_Toc169703049)

[Базы данных пользователей 4](#_Toc169703049)

[ГЛАВА 2. ЗАДАНИЕ НА ПРАКТИКУ 5](#_Toc169703050)

[Задача №1 «Регистрация бота» 5](#_Toc169703051)

[Задача №2 «Реализация команд» 6](#_Toc169703052)

[Задача №3 «Создание базы SQlite» 6](#_Toc169703053)

[Задача №4 «Регистрация пользователей» 8](#_Toc169703054)

[Глава 3. Выполнение заданий на практику 10](#_Toc169703055)

[Решение задачи №1 10](#_Toc169703056)

[Решение задачи №2 10](#_Toc169703057)

[Решение задачи №3 10](#_Toc169703058)

[Решение задачи №4 10](#_Toc169703059)

[ЗАКЛЮЧЕНИЕ 12](#_Toc169703060)

[СПИСОК ЛИТЕРАТУРЫ 13](#_Toc169703061)

[ПРИЛОЖЕНИЕ 1 15](#_Toc169703062)

[ПРИЛОЖЕНИЕ 2 16](#_Toc169703063)

**ВВЕДЕНИЕ**

Вид практики – Учебная технологическая (проектно-технологическая) практика (далее Учебная практика).

Цель учебной практики: получение первичных профессиональных умений навыков.

Задачи учебной практики:

1. Закрепление и углубление теоретических знаний в области информационных технологий;
2. Приобретение и развитие первичных профессиональных навыков и умений в области прикладной математики и информатики.

**ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Телеграмм боты**

Телеграмм боты — это специализированные аккаунты в мессенджере Telegram, которые управляются программным обеспечением, а не живым человеком. Эти боты могут принимать и отправлять сообщения так же, как обычные пользователи, но их возможности на этом не заканчиваются. Они могут выполнять разнообразные задачи, такие как обработка команд, предоставление информации, интеграция с внешними сервисами, управление подписками и многое другое. Телеграмм предоставляет мощный API, который позволяет разработчикам создавать и управлять ботами, а также настраивать их поведение в зависимости от потребностей пользователей. Это API включает в себя множество функций, от отправки простых текстовых сообщений до взаимодействия с более сложными мультимедийными данными и сервисами.

**Асинхронное программирование**

Асинхронное программирование — это метод программирования, который позволяет выполнять задачи, не блокируя основной поток исполнения программы. Это полезено для работы с операциями ввода-вывода, такими как сетевые запросы или взаимодействие с базой данных. В традиционном синхронном программировании выполнение таких операций могло бы замедлить всю программу. Асинхронное программирование решает эту проблему, позволяя другим частям программы продолжать работу, пока операция ввода-вывода выполняется. В Python для реализации асинхронного программирования используются ключевые слова async и await, а также библиотека asyncio, которая предоставляет инструменты для создания и управления асинхронными задачами.

**Библиотека Aiogram**

Aiogram — это мощная библиотека для разработки Telegram-ботов на языке программирования Python, которая поддерживает асинхронное программирование. Она предоставляет удобный интерфейс для работы с API Telegram, что позволяет разработчикам легко создавать и настраивать ботов. С помощью Aiogram можно обрабатывать различные команды, управлять состояниями бота, интегрировать его с внешними сервисами и базами данных, а также реализовывать сложные сценарии взаимодействия с пользователями. Библиотека активно развивается и поддерживается сообществом, что делает её отличным выбором для разработки современных и функциональных Telegram-ботов.

**Базы данных**

Для хранения данных пользователей в проектах часто используются базы данных. В данном проекте была выбрана база данных SQLite, которая является легковесной и не требует установки сервера. SQLite — это встраиваемая система управления базами данных (СУБД), которая хранит все данные в одном файле на диске. Этот подход упрощает развертывание и управление базой данных, особенно для небольших и средних проектов. SQLite обеспечивает высокую производительность и надежность, а также поддерживает основные функции реляционных баз данных, такие как транзакции, индексы и ограничение целостности данных. В нашем проекте SQLite использовалась для хранения информации о пользователях и их ежедневном потреблении воды, что позволило легко и эффективно управлять данными и выполнять необходимые операции.

**ГЛАВА 2. ЗАДАНИЕ НА ПРАКТИКУ**

**Задача №1 «Регистрация бота»**

Первая задача заключалась в создании и регистрации бота в Telegram. Для этого необходимо было создать нового бота с помощью @BotFather и получить уникальный токен доступа.

**Задача №2 «Реализация команд»**

Вторая задача включала реализацию основных команд бота, таких как /start и /help, с помощью асинхронных функций.

**Задача №3 «Создание базы SQlite»**

Третья задача состояла в создании базы данных для хранения информации о пользователях и их потреблении воды. Для этого была использована SQLite.

**Задача №4 «Регистрация пользователей»**

Четвертая задача заключалась в реализации механизма регистрации пользователей и записи их данных в базу данных SQLite. А также реализация дальнейшего использования данных из таблицы.

**Глава 3. Выполнение заданий на практику**

**Решение задачи №1**

Ваше решение, что использовали, какие были сложности

Например,

Для регистрации бота был использован @BotFather. После создания бота был получен токен доступа, который использовался для взаимодействия с API Telegram через библиотеку Aiogram.

Диалог с ботом BotFather предоставлен в Приложение №1.

**Решение задачи №2**

Для реализации команд были созданы соответствующие обработчики в Aiogram. Команда /start приветствует пользователя и говорит ему о дальнейших действиях. Команда /help предоставляет информацию о том как начать взаимодействие с ботом.

Определяем бота с помощью токена и создаём основную асинхронную функцию, запускаем пуллинг запросов от телеграмма для бота ,а также определяем роутер, который будет заменять работу Dispather в функциях работы с сообщениями в файле handlers.

**bot = Bot(token=API\_TOKEN)**

**dp = Dispatcher()**

users = get\_ids()

**async def main():**

dp.include\_router(router)

await dp.start\_polling(bot)

Теперь переносимся в файл *“handlers.py”* и реализуем там все нужные нам функции для работы с командами нашего бота

**@router.message(CommandStart())**

async def cmd\_start(message: Message):

await message.answer(

'Приветствую ,ученик, я твой водный сенсей. Введи команду /help для старта твоего пути водного шиноби!')

**@router.message(Command('help'))**

async def cmd\_help(message: Message):

await message.answer(f"Я бот который поможет тебе выпивать необходимое количество воды! \n "

f"Для того чтобы начать - выполни /register,а затем сообщай мне о выпитых тобою стаканах воды, "

f"выполняй дневную норму и копи {emoji.emojize(':fire:')}STREAK{emoji.emojize(':fire:')}.")

**@router.message(Command('register'))**

async def registration(message: Message):

(В дальнейшем, сделав базу данных, мы реализуем здесь регистрацию пользователей)

**@router.message(lambda message: message.text == 'Я выпил воды')**

async def process\_drink(message: Message):

(В дальнейшем, сделав базу данных, мы реализуем здесь обновления всех данных в таблице при нажатии пользователем кнопки “Я выпил воды”.

**@router.message(lambda message: message.text == 'Профиль')**

async def show\_profile(message: Message):

(В дальнейшем, сделав базу данных, мы реализуем здесь отображение данных о пользователе)

**@router.message(lambda message: message.text == 'Таблица Лидеров')**

async def show\_leaderboard(message: Message):

(В дальнейшем, сделав базу данных, мы реализуем здесь отображение таблицы лидеров)

**Решение задачи №3**

Для создания базы данных был использован sqlite3. Создание самой базы и функции добавления и получения из неё информации , для удобства произведём в другом файле под названием “database.py”.

**import sqlite3**

**database = sqlite3.connect("bot.sqlite")**

**def add\_user(message):**

(Реальизация регистрации пользователя)

**def upd\_date(message):**

(Реализация функции обновления даты питья воды у пользователя)

**def get\_last\_date(user\_id):**

(Функция для получения из базы данных по id пользователя даты последнего питья воды)

**def get\_streak(user\_id):**

(Функция для получения из базы данных по id пользователя значения его STREAK)

**def get\_ids():**

(Функция для получения из базы данных всех id пользователей)

**def get\_table(message):**

(Функция для получения базы данных в виде таблицы, для дальнейшего вывода)

**Решение задачи №4**

Теперь, реализовав базу данных, мы можем вернутся к функциям из handlers, и сделав

**from** database **import** *add\_user, plus\_streak, get\_streak, get\_last\_date, upd\_date, break\_streak, get\_table*

К **registration** добавляем:

add\_user(message)

await message.answer('Регистрация прошла успешно, твой путь начался!' + emoji.emojize(':flexed\_biceps:'),

reply\_markup=kb.main)

К **process\_drink** добавляем:

chat\_id = message.chat.id

curr\_streak = get\_streak(chat\_id)

last\_click = datetime.strptime(get\_last\_date(chat\_id), '%Y-%m-%d')

print(last\_click)

print(current\_date - last\_click)

print()

if (current\_date - last\_click) < timedelta(days=1):

await message.reply(f'Сегодня ты выпил уже достаточно!{emoji.emojize(':sweat\_droplets:')}')

return

if last\_click == 'date':

plus\_streak(message, curr\_streak)

upd\_date(message)

await message.reply(

f"Ты попил воды! Твой STREAK : {curr\_streak + 1} {emoji.emojize(':fire:')}.")

return

elif (current\_date - last\_click) > timedelta(days=2) and curr\_streak != 0:

curr\_streak = 0

await message.answer('Твой STREAK СБРОШЕН')

plus\_streak(message, curr\_streak)

upd\_date(message)

await message.reply(

f"Ты попил воды! Твой STREAK : {curr\_streak + 1} {emoji.emojize(':fire:')}.")

К **show\_profile** добавляем:

chat\_id = message.chat.id

curr\_streak = get\_streak(chat\_id)

await message.answer(

f"{emoji.emojize(':bust\_in\_silhouette:')} \n Имя: {message.chat.first\_name} \n {emoji.emojize(':fire:')} STREAK x{curr\_streak} {emoji.emojize(':fire:')}")

К **show\_leaderboard** добавляем:

table = get\_table(message)

await message.reply(table, parse\_mode='HTML')

Все нужные функции реализованы, работа бота продемонстрирована в Приложение №2

**ЗАКЛЮЧЕНИЕ**

В ходе практики были изучены основы языка программирования Python и создание ботов с использованием библиотеки Aiogram. Задачи были направлены на закрепление теоретического материала по асинхронному программированию, работе с API Telegram, а также взаимодействию с базами данных. Были изучены основные команды Telegram-ботов и их обработка. Особое внимание уделялось созданию базы данных SQLite и взаимодействию с ней.

Предлагалось решить задачи, связанные с регистрацией бота, реализацией команд, созданием базы данных и регистрацией пользователей. В результате практики были усовершенствованы мои компетенции и закреплены теоретические навыки.

В ходе учебной практики я научился: выполнять поставленные задачи, следуя условиям, изменять готовые решения по мере нахождения ошибок, искать альтернативные пути решения задач. Также я освоил новую для меня среду программирования и работу с GitHub (создание репозиториев для выгрузки решений задач). Я научился использовать асинхронные функции и циклы, обрабатывать команды, изучил библиотеки для работы с базами данных (sqlite3). Выполнил работу с различными API и приемами работы с ними.

По мере прохождения учебно-технологической практики я выполнил четыре задачи, в которых использовал полученные знания и закрепил навыки работы с Python и Aiogram. В течение практики задачи были выполнены, а цели достигнуты.

# СПИСОК ЛИТЕРАТУРЫ

**Перечень учебной литературы ресурсов сети «Интернет», необходимой для проведения практики**

1. Варфоломеева, Т. Н. Структуры данных и основные алгоритмы их обработки : учебное пособие / Т. Н. Варфоломеева. - Москва : ФЛИНТА, 2017. - 159 с. - ISBN 978-5-9765-3691-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1860018 (дата обращения: 19.01.2023). – Режим доступа: по подписке.
2. Гданский, Н. И. Основы теории и алгоритмы на графах : учебное пособие / Н. И. Гданский. — Москва : ИНФРА-М, 2020. — 206 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-014386-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/978686 (дата обращения: 19.01.2023). – Режим доступа: по подписке.
3. Затонский, А. В. Программирование и основы алгоритмизации. Теоретические основы и примеры реализации численных методов: учебное пособие / А.В. Затонский, Н.В. Бильфельд. — 2-е изд. — Москва: РИОР : ИНФРА-М, 2022. — 167 с. — (Высшее образование). — DOI: https: //www.dx.doi.org/10.12737/20468. - ISBN 978-5-369-01195-9. - Текст: электронный. - URL: https://znanium.com/catalog/product/1860435 (дата обращения: 16.02.2023). – Режим доступа: по подписке.
4. Гуриков, С. Р. Основы алгоритмизации и программирования на Python- Москва: НИЦ ИНФРА-М, 2023. - 343 с. ISBN 978-5-16-016906-4. - Текст: электронный. - URL: https://znanium.ru/catalog/document?id=424791 (дата обращения: 16.02.2023). – Режим доступа: по подписке.

**Перечень ресурсов информационно-телекоммуникационной сети «Интернет»**

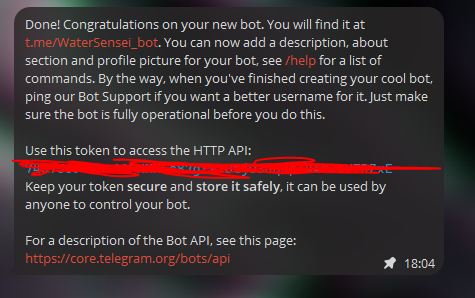
1. Aiogram Documentation: https://docs.aiogram.dev/
2. Telegram Bot API Documentation: https://core.telegram.org/bots/api
3. SQLite Documentation: https://www.sqlite.org/docs.html
4. Python Official Documentation:

https://docs.python.org/3/library/asyncio.html

**ПРИЛОЖЕНИЕ №1**

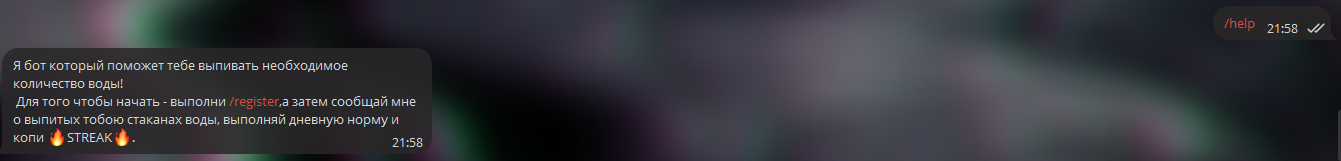
****

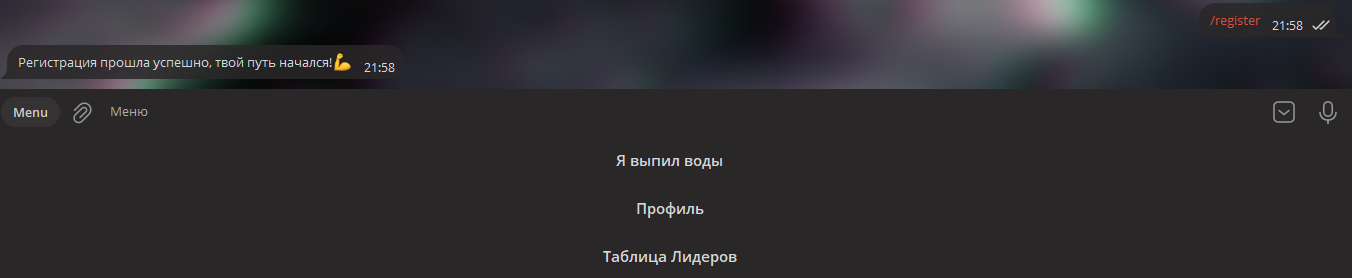
****



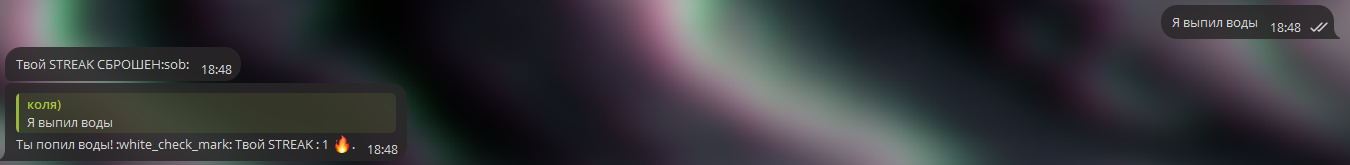
**ПРИЛОЖЕНИЕ №2**

****

****

****

****

**** ****