## Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

## (СибГУТИ)

09.03.01 Информатика и вычислительная техника

Профиль: Программное обеспечение средств вычислительной техники и авто- матизированных систем

## (очная форма обучения)

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

## на Кафедре прикладной информатики и кибернетики

Разработка Telegram-бота для перевода песен

## Выполнил:

студент института ИВТ

## гр. ИП-015 /Костыль В.Ю./

«27» мая 2023г. (подпись)

## Проверил:

Руководитель от СибГУТИ /Приставка П.А./

«27» мая 2023г. (подпись)

## Новосибирск 2023

**План-график проведения** производственной **практики**

Вид практики

## Костыль Валерии Юрьевны

Фамилия Имя Отчество студента

## института Информатика и вычислительная техника , 3 курса, гр. ИП- 01 5

Направление: 09.03.01 – Информатика и вычислительная техника

Код – Наименование направления (специальности)

## Профиль: Программное обеспечение средств вычислительной техники и автоматизированных систем

Место прохождения практики Кафедра прикладной информатики и кибернетики

Объем практики: **360/10** часов/ЗЕ Вид практики ***производственная***

Тип практики ***Технологическая (проектно-технологическая) практика***

## Срок практики с "30" января 2023 г.

по "27" мая 2023 г.

## Содержание практики\*:

|  |  |
| --- | --- |
| Наименование видов деятельности | Дата (начало – окончание) |
| 1. Общее ознакомление со структурным подразделением  предприятия, вводный инструктаж по технике безопасности | 30.01.2023–02.02.2023 |
| 2. Выдача задания на практику, деление студентов на группы  (если необходимо), определение конкретной индивидуальной темы, формирование плана работ | 03.02.2023–05.02.2023 |
| 3. Работа с библиотечными фондами структурного подразде-  ления или предприятия, сбор и анализ материалов по теме практики | 07.02.2023–12.02.2023 |
| 4. Выполнение работ в соответствии с составленным планом:   * Определение технологий, с которыми будет вестись работа * Изучение информации об API Telegram, в частности библиотеки aiogram и других * Создание бота * Формирование сообщения и его отправка | 14.02.2023 – 21.05.2023 |
| 5. Анализ полученных результатов и произведенной работы  Составление отчета по практике, защита отчета | 23.05.2023–27.05.2023 |

\*В соответствии с программой практики

## Руководитель от СибГУТИ / /

« » 2023г. (подпись)

# ЗАДАНИЕ НА ПРАКТИКУ

Разработка Telegram-бота для перевода песен по названию.

Работу можно разбить на несколько этапов:

1. Определение технологий, с которыми будет вестись работа.
2. Изучение информации об API Telegram, в частности библиотеки aiogram и других.
3. Создание бота перевода песен.
4. Формирование сообщения и его отправка.

# ВВЕДЕНИЕ

Бот — это небольшое приложение, которое самостоятельно выполняет заранее созданные задачи без участия пользователя. Telegram-бот умеет делать всё, что мог бы делать человек в чате: отвечать на вопросы, присылать ссылки на сайты или создавать мемы. Автоматически или по запросу он может отправлять:

● текстовые сообщения;

● картинки;  
● видео;  
● файлы.  
  
Боты умеют:

● Выполнять действия, которые нельзя настроить на канале. Например, продавать товары и принимать оплату, общаться с пользователями, скрывать личные данные. Боты для Телеграма могут собирать потенциальных клиентов, подключая их [к CRM](https://practicum.yandex.ru/blog/chto-takoe-crm-sistema-i-kak-ih-vybirat/), системе продажи билетов или платформе обмена сообщениями.

● Выполнять несколько разных команд одновременно.

Важная функция ботов — возможность запускать цепочку действий, постепенно запрашивая у пользователя новую информацию. Если отправить боту команду /start, на экране появятся кнопки.

● Интегрироваться с другими сервисами.

Например, можно настроить бота, чтобы пользователи могли выполнять быстрый поиск GIF-анимации во встроенном режиме. Если встроенные запросы будут включены, пользователи смогут ввести имя бота и свой запрос в поле ввода текста в любом чате и мгновенно получить нужную гифку.

# ПОСТАНОВКА ЗАДАЧИ

# Для реализации бота для перевода слов потребуется получить источник откуда будем брать перевод. Для этого использую сайт lyrsense.com. Для того, чтобы взять с сайта данные, нам потребуется подключить соответствующую библиотеку. Задача – разобраться с тем, как правильно использовать библиотеку, и как запрограммировать нашего бота, по требованию пользователя, эту информацию предоставить.

# ВЫБОР ПРОГРАММНЫХ СРЕДСТВ

В качестве языка программирования был выбран Python версии (3.11). Он располагает всем необходимым инструментарием для написания данной программы.

**Python** — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) или C++.

Python является мультипарадигменным языком программирования,поддерживающим императивное, процедурное, структурное, объектно-ориентированное программирование, метапрограммирование и функциональное программирование. Задачи обобщённого программирования решаются за счёт динамической типизации. Аспектно-ориентированное программирование частично поддерживается через декораторы, более полноценная поддержка обеспечивается дополнительными фреймворками. Такие методики как контрактное и логическое программирование можно реализовать с помощью библиотек или расширений. Основные архитектурные черты — [динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [полная интроспекция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_%D0%B2_Python), механизм [обработки исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка [многопоточных вычислений](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C) с глобальной блокировкой интерпретатора, высокоуровневые [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Поддерживается разбиение программ на [модули](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые, в свою очередь, могут объединяться в пакеты.

Стандартная библиотека включает большой набор полезных переносимых функций, начиная с возможностей для работы с текстом и заканчивая средствами для написания сетевых приложений. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений или разработка игр, могут реализовываться посредством обширного количества сторонних библиотек, а также интеграцией библиотек, написанных на Си или C++, при этом и сам интерпретатор Python может интегрироваться в проекты, написанные на этих языках. Существует и специализированный [репозиторий](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B9%22%20/o%20%22%D0%A0%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B9) программного обеспечения, написанного на Python, — [PyPI](https://ru.wikipedia.org/wiki/PyPI%22%20/o%20%22PyPI). Данный репозиторий предоставляет средства для простой установки пакетов в операционную систему и стал стандартом де-факто для Python. По состоянию на 2019 год в нём содержалось более 175 тысяч пакетов.

В качестве среды для программирования была выбрана IDE PyCharm от JetBrains, так как она предоставляет удобный инструментарий для работы с данным языком, такой как: удобная установка библиотек, подсветка синтаксиса, рекомендации по написанию кода и т.д.

**PyCharm** — это кроссплатформенная интегрированная среда разработки для языка программирования Python, разработанная компанией JetBrains на основе IntelliJ IDEA. Предоставляет пользователю комплекс средств для написания кода и визуальный отладчик.

Для создания бота использовалась библиотека aiogram

# ВЫПОЛНЕНИЕ РАБОТЫ

**Этап 1:** Определение технологий, с которыми будет вестись работа

Для написания Telegram ботов существует удобный инструментарий на языке Python. Из библиотек для создания бота был выбран aiogram, так как он представляет из себя простое и понятное решение, полностью покрывающее наши запросы.

**Этап 2:** Изучение информации об API Telegram, в частности библиотеки aiogram и других.

Были прочитаны документация и статьи на тему библиотек aiogram, dispatcher, executor, types.

**Этап 3:** Создание бота.

Прежде чем начинать разработку, бота необходимо зарегистрировать и получить его уникальный id, являющийся одновременно и токеном. Для этого в Telegram существует специальный бот — [@BotFather](https://telegram.me/botfather).  
Через него задаются внешние параметры бота такие как имя, главная иконка и т.д.

Токен который мы получили, в дальнейшем будем использовать для подключения к боту через Python.

telegram\_bot\_token = '61072881:AG0pPj-NbGM-cdA0A4RjQYY3SUGxPgrvtM'

bot = Bot(token=telegram\_bot\_token)

dp = Dispatcher(bot)

**Этап 4:** Формирование сообщения и его отправка.

Ниже описаны функции для отправки сообщений на определенные команды, введенные пользователем.

@dp.message\_handler(commands="Start")

async def start(message: types.Message):

await message.answer("Здравствуйте, введите песню, для поиска перевода!")

Следующий handler обрабатывает любое сообщение от пользователя (кроме приведенных выше), и запускает весь алгоритм перевода слов.

@dp.message\_handler()

async def search(message: types.Message):

query = message.text.strip()

url = f"https://lyrsense.com/search?s={query}"

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

find\_url = soup.find(class\_="myModerList").find('a')

find\_url = "https://lyrsense.com" + find\_url.get('href')

response = requests.get(find\_url)

soup = BeautifulSoup(response.text, 'html.parser')

name\_song = soup.find("h1", {"style": "padding-bottom:0;"}).text

song\_translate = soup.find("p", {"id": "ru\_text"}).find\_all('span')

song\_original = soup.find("p", {"id": "fr\_text"}).find\_all('span')

res =name\_song + '\n\n'

for i in song\_translate:

res+=i.text + '\n'

res += "\nОригинал песни:\n\n"

for i in song\_original:

res+=i.text + '\n'

await message.reply(res)

Подключение и создание базы данных.

conn = sqlite3.connect('search\_history.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS search\_history (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

query TEXT

)

''')

conn.commit()

Команда help отправляет пользователю сообщение о доступных командах.

@dp.message\_handler(commands='help')

async def clear\_history(message: types.Message):

await message.reply("Для поиска перевода введите название песни" + "\n Команда /history покажет историю запросов."+ "\n Команда /clearhistory очистит историю запросов.")

Команда history отправляет пользователю сообщение с историей его запросов.

@dp.message\_handler(commands="history")

async def history(message: types.Message):

user\_id = message.from\_user.id

cursor.execute('SELECT query FROM search\_history WHERE user\_id=?', (user\_id,))

search\_history = cursor.fetchall()

if search\_history:

history\_text = "История поиска:\n\n"

for query in search\_history:

history\_text += f"- {query[0]}\n"

else:

history\_text = "История поиска пуста."

await message.reply(history\_text)

Команда clearhistory удаляет историю запросов.

@dp.message\_handler(commands="clearhistory")

async def clear\_history(message: types.Message):

user\_id = message.from\_user.id

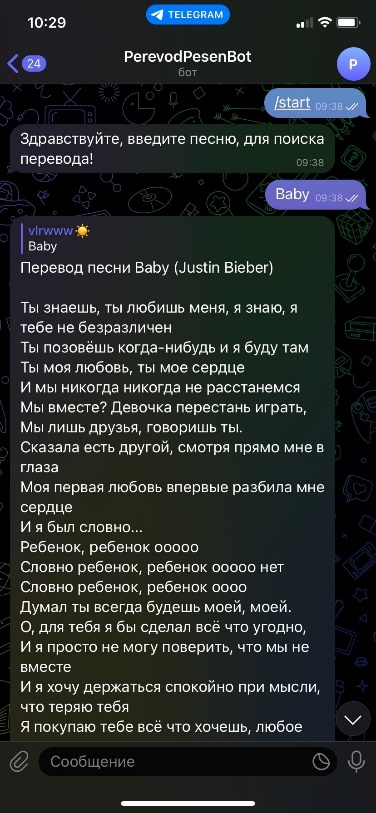
cursor.execute('DELETE FROM search\_history WHERE user\_id=?', (user\_id,))

conn.commit()

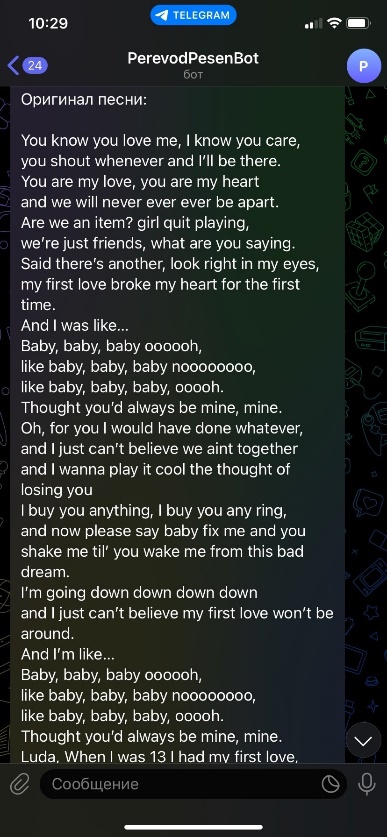
await message.reply("История поиска очищена.")

# ОПИСАНИЕ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

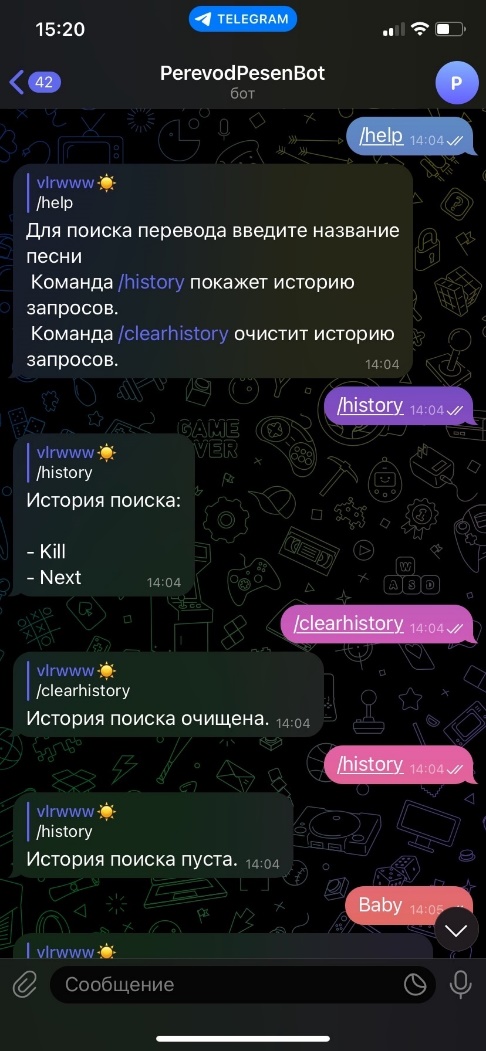
Для запуска работы телеграмм-бота нужно прописать команду /start, после чего ввести название нужной песни.



После чего бот выводит перевод песни на русский и оригинал на иностранном языке.



На скриншоте представлены команды и реакции бота на них.



# ЗАКЛЮЧЕНИЕ

Был создан бот для быстрого перевода песен по её названию с использованием библиотеки aiogram. Библиотека Aiogram Python – это мощный инструмент для создания ботов для Telegram на языке Python. Она предоставляет обширный набор функций и возможностей, которые позволяют создавать ботов любой сложности.

Получила опыт работы с различными библиотеками для создания ботов и работы с ними на языке Python.

#### СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Ю.М. Шыырап, Л.Ю. Забелин, М.Ю. Шыырап Основы программирования Python: Учебное пособие / Сибирский государственный университет телекоммуникаций и информатики; каф. систем автоматизированного проектирования. – Новосибирск, 2017. – 197 с. №164 от 20.12.18.
2. Документация по pyTelegramBotAPI [Электронный ресурс]: PyTelegramBot.URL: <https://pytba.readthedocs.io/ru/latest/index.html> (Дата обращения 10.05.2023)

#### ПРИЛОЖЕНИЯ

from aiogram import Bot, Dispatcher, executor, types

import requests

from bs4 import BeautifulSoup

import sqlite3

telegram\_bot\_token = '6107281:G0Pj-NbGM-cdA0A4RjQYY3SUGxPgrvtM'

bot = Bot(token=telegram\_bot\_token)

dp = Dispatcher(bot)

conn = sqlite3.connect('search\_history.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS search\_history (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

query TEXT

)

''')

conn.commit()

@dp.message\_handler(commands="start")

async def start(message: types.Message):

await message.answer("Здравствуйте, введите песню, для поиска перевода!")

@dp.message\_handler(commands="history")

async def history(message: types.Message):

user\_id = message.from\_user.id

cursor.execute('SELECT query FROM search\_history WHERE user\_id=?', (user\_id,))

search\_history = cursor.fetchall()

if search\_history:

history\_text = "История поиска:\n\n"

for query in search\_history:

history\_text += f"- {query[0]}\n"

else:

history\_text = "История поиска пуста."

await message.reply(history\_text)

@dp.message\_handler(commands="clearhistory")

async def clear\_history(message: types.Message):

user\_id = message.from\_user.id

cursor.execute('DELETE FROM search\_history WHERE user\_id=?', (user\_id,))

conn.commit()

await message.reply("История поиска очищена.")

@dp.message\_handler(commands='help')

async def clear\_history(message: types.Message):

await message.reply("Для поиска перевода введите название песни" + "\n Команда /history покажет историю запросов."+ "\n Команда /clearhistory очистит историю запросов.")

@dp.message\_handler()

async def search(message: types.Message):

query = message.text.strip()

url = f"https://lyrsense.com/search?s={query}"

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

find\_url = soup.find(class\_="myModerList").find('a')

find\_url = "https://lyrsense.com" + find\_url.get('href')

response = requests.get(find\_url)

soup = BeautifulSoup(response.text, 'html.parser')

name\_song = soup.find("h1", {"style": "padding-bottom:0;"}).text

song\_translate = soup.find("p", {"id": "ru\_text"}).find\_all('span')

song\_original = soup.find("p", {"id": "fr\_text"}).find\_all('span')

res = name\_song + '\n\n'

for i in song\_translate:

res += i.text + '\n'

res += "\nОригинал песни:\n\n"

for i in song\_original:

res += i.text + '\n'

await message.reply(res)

user\_id = message.from\_user.id

cursor.execute('INSERT INTO search\_history (user\_id, query) VALUES (?, ?)', (user\_id, query))

conn.commit()

if \_\_name\_\_ == "\_\_main\_\_":

executor.start\_polling(dp)

**Отзыв о работе студента**

|  |
| --- |
| Костыль Валерия Юрьевна |
| (ФИО студента) |

Уровень освоения компетенций

Костыль Валерия Юрьевна

(ФИО студента)

|  |  |
| --- | --- |
| Компетенции | Уровень сформированности компетенций |
| *ПК-1 - Способен разрабатывать требования и проектировать программное обеспечение* |  |

### отметка о зачете

Руководитель практики от СибГУТИ:

Должность руководителя подпись ФИО руководителя

### " " 20\_ г