

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Московский государственный технический университет
имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

БАКАЛАВРСКАЯ ПРОГРАММА 09.03.01/03 Вычислительные машины, комплексы,
системы и сети

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Тип практики Эксплуатационная


Название
предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Студент ИУ6-63Б

Руководитель практики
от МГТУ им.
Н.Э.Баумана


(Подпись, дата)

03.03.2024 К.А. Логачев
(И.О. Фамилия)


(Подпись, дата)

03.03.24 С.С. Данилюк
(И.О. Фамилия)

Оценка

Отлично

2024 г.

ЗАДАНИЕ на производственную практику

по теме РАЗРАБОТКА ПРОТОТИПА ЧАТ-БОТА

Студент группы ИУ6-63Б

Логачев Кирилл Александрович

(Фамилия, имя, отчество)

Направление подготовки 09.03.01 Информатика и вычислительная техника

Бакалаврская программа 09.03.01/03 Вычислительные машины, комплексы, системы и сети

Тип практики Эксплуатационная практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Техническое задание _____

изучение особенностей конструирования и применения чат-ботов, реализация прототипа чат-бота _____

Оформление отчета по практике:

Отчет на 15-25 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

нет


Дата выдачи задания « 28 » июня 2024 г.

Руководитель практики
от МГТУ им. Н.Э.Баумана

Студент


28.06.2024
(Подпись, дата)

С.С. Данилюк
(И.О. Фамилия)


28.06.2024
(Подпись, дата)

К.А. Логачев
(И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Описание стека технологий	6
1.1 React.....	6
1.2 GitHub	7
1.3 GitHub Pages	7
1.4 Webpack	8
1.5 SCSS.....	8
2 Описание кода	10
3 Пример использования Чат-бота	17
3.1 Начало общения	17
3.2 Запрос имени пользователя	17
3.3. Запрос погоды	18
ЗАКЛЮЧЕНИЕ	20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	21

ВВЕДЕНИЕ

Целью данной практики являлась разработка прототипа чат-бота, способного обрабатывать команды пользователя и взаимодействовать с внешними сервисами. Основные задачи включали проектирование архитектуры системы, выбор и внедрение подходящих технологий, реализацию ключевого функционала, а также развертывание проекта на платформе GitHub Pages.

Проектирование чат-бота включало несколько ключевых этапов. Во-первых, требовалось определить функциональные и нефункциональные требования к системе, чтобы обеспечить её соответствие потребностям пользователей и оптимальное взаимодействие с внешними ресурсами. Для этого были проанализированы существующие решения, выявлены их сильные и слабые стороны, а также выбраны оптимальные подходы к реализации.

Во-вторых, значительное внимание было уделено выбору технологий. Оценка доступных инструментов и платформ, таких как библиотеки для обработки естественного языка, фреймворки для создания ботов и интеграции с API, позволила выбрать наиболее подходящие для выполнения поставленных задач. Особое внимание было уделено обеспечению безопасности данных и устойчивости к возможным сбоям.

В-третьих, реализация ключевого функционала включала разработку алгоритмов обработки команд, создание пользовательского интерфейса и настройку взаимодействия с внешними сервисами. Это потребовало не только технических знаний, но и умения эффективно решать проблемы и оптимизировать процессы.

Развертывание проекта на платформе GitHub Pages стало завершающим этапом практики, который обеспечил доступность чат-бота для пользователей и позволил продемонстрировать результаты работы. Этот этап также включал настройку систем контроля версий и автоматизацию процессов развертывания.

Создание такого прототипа потребовало глубокого изучения современных веб-технологий и их практического применения в условиях ограниченного времени и ресурсов. Этот процесс способствовал развитию навыков командной работы и проектного мышления, что особенно важно в условиях реальной разработки. Проектная деятельность также позволила укрепить навыки в области анализа требований, выбора инструментов и управления проектом, что является ценным опытом для будущей профессиональной деятельности в сфере информационных технологий.

1 Описание стека технологий

Для реализации проекта чат-бота был использован следующий стек технологий:

1.1 React

React — это библиотека JavaScript, предназначенная для создания динамических пользовательских интерфейсов. Основное преимущество React заключается в использовании виртуального DOM, который делает работу с динамическими изменениями в приложении более эффективной. Виртуальный DOM позволяет React обновлять только изменённые части интерфейса, что значительно повышает производительность приложения [1].

Листинг 1 – Пример использования React

```
import React, { useState } from 'react';

function ChatBot() {
  const [message, setMessage] = useState('');
  const [responses, setResponses] = useState([]);

  const handleSend = () => {
    setResponses([...responses, { text: message }]);
    setMessage('');
  };

  return (
    <div className="chat-bot">
      <div className="chat-window">
        {responses.map((response, index) => (
          <div key={index} className="chat-message">
            {response.text}
          </div>
        ))}
      </div>
      <input
        type="text"
        value={message}
        onChange={e => setMessage(e.target.value)}
      />
      <button onClick={handleSend}>Send</button>
    </div>
  );
}

export default ChatBot;
```

1.2 GitHub

GitHub — это платформа для управления версиями, основанная на системе Git. Она позволяет отслеживать изменения в коде, управлять версиями и сотрудничать над проектами. GitHub обеспечивает удобный интерфейс для работы с репозиториями, а также предоставляет возможности для код-ревью и управления задачами.

Листинг 2 – Пример команды Git для коммита изменений

```
git add .  
git commit -m "Добавил новые функции в чат-бота"  
git push origin main
```

1.3 GitHub Pages

GitHub Pages предоставляет возможность развертывания статических сайтов непосредственно из репозитория GitHub. Это идеальное решение для публикации документации, блогов или демо-версий проектов [3]. Для чат-бота это позволило сделать его доступным для пользователей без необходимости в отдельном хостинге.

GitHub Actions — это инструмент для автоматизации процессов CI/CD (непрерывной интеграции и доставки). С помощью GitHub Actions можно настроить автоматический запуск тестов, сборку проекта и развертывание приложения при каждом обновлении репозитория.

Листинг 3 – Пример файла конфигурации GitHub Actions (.github/workflows/deploy.yml)

```
name: Deploy ChatBot  
on:  
  push:  
    branches:  
      - main  
  
jobs:  
  build:  
    runs-on: ubuntu-latest  
    steps:  
      - name: Checkout code  
        uses: actions/checkout@v3  
  
      - name: Install dependencies  
        run: npm install
```

Продолжение листинга 3

```
- name: Build project
  run: npm run build

- name: Deploy to GitHub Pages
  uses: peaceiris/actions-gh-pages@v3
  with:
    github_token: ${ secrets.GITHUB_TOKEN }
    publish_dir: ./build
```

1.4 Webpack

Webpack — это инструмент для сборки модулей, который помогает оптимизировать и управлять зависимостями в проекте. Он позволяет объединять JavaScript, CSS и другие ресурсы в единые файлы, что улучшает производительность и упрощает процесс разработки [2].

Листинг 4 – Пример конфигурации WebPack (webpack.config.js)

```
js
Копировать код
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist')
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: 'babel-loader'
      },
      {
        test: /\.css$/,
        use: ['style-loader', 'css-loader']
      }
    ]
  }
};
```

1.5 SCSS

SCSS (Sassy CSS) — это расширение CSS, которое добавляет поддержку переменных, вложенных правил, миксинов и других удобных функций. Это упрощает создание и поддержку стилей, особенно в больших проектах.

Листинг 5 – Пример использования SCSS

```
scss
Копировать код
$primary-color: #3498db;

.chat-window {
  background-color: $primary-color;
  padding: 10px;
  border-radius: 5px;

  .chat-message {
    margin: 5px 0;
    color: white;
  }
}
```

Этот стек технологий обеспечил создание эффективного и масштабируемого чат-бота, предоставив мощные инструменты для разработки, управления версиями, автоматизации процессов и развертывания.

2 Описание кода

Код чат-бота написан на JavaScript с использованием библиотеки React и включает в себя набор функций, каждая из которых отвечает за выполнение определённых команд. Основные функции и их описание приведены ниже. Код использует различные вспомогательные функции и состояния для обеспечения эффективного взаимодействия с пользователем.

Функция `commandParser` предназначена для обработки вводимых пользователем команд. Она разбивает строку команды на основную команду и аргументы. Это позволяет ботам обрабатывать команды с различными параметрами и выполнять соответствующие действия.

Листинг 6 – Пример функции

```
// парсинг команды
export const commandParser = (fullCommand: string): CommandsType
=> {
  const words = fullCommand.split(' ');
  return {
    command: words[0],
    args: words.slice(1)
  };
};
```

Пример использования.

Если пользователь вводит команду `"/weather New York"`, функция `commandParser` разобьёт строку на команду `"/weather"` и аргумент `["New York"]`. Это позволит боту понять, что требуется выполнить команду погоды и передать аргумент для уточнения местоположения.

Функция `errorsMessageWrap` является обёрткой для обработки ошибок. Она используется для вызова командных функций с предварительной проверкой состояния и наличия необходимых данных. Если возникают ошибки, функция возвращает соответствующее сообщение об ошибке.

Листинг 7 – Пример функции

```
// оберточная функция ошибок
export const errorsMessageWrap = (
  isStart: boolean,
  numbers: NumberMathTypes,
  setBodyMessage: React.Dispatch<React.SetStateAction<string |
undefined>>,
```

Продолжение листинга 7

```
    setNumbers:
    React.Dispatch<React.SetStateAction<NumberMathTypes
    undefined>>
  ) => {
    return (commandFunction: () => void) => {
      if (isStart && !numbers) {
        commandFunction();
      } else if (numbers) {
        setBodyMessage(notMathCommand);
        setNumbers(undefined);
      } else {
        setBodyMessage(notStartChat);
      }
    };
  };
};
```

Пример использования.

Если бот находится в активном состоянии и пользователю требуется выполнить математическую операцию, функция проверяет, введены ли числа. Если числа введены, возвращается сообщение о недопустимой команде. Если нет, отображается сообщение о необходимости начать чат.

Функция `commandStart` инициирует общение с пользователем. При вызове команды `/start` бот проверяет текущее состояние и выводит приветственное сообщение, приглашая пользователя продолжить взаимодействие.

Листинг 8 – Пример функции

```
// команда /start
export const commandStart = (
  isStart: boolean,
  setIsStart:
    React.Dispatch<React.SetStateAction<boolean>>,
  setBodyMessage:
    React.Dispatch<React.SetStateAction<string | undefined>>
) => {
  if (!isStart) {
    setIsStart(true);
    setBodyMessage('Привет, меня зовут Чат-бот, а как зовут тебя?');
  } else {
    setBodyMessage('Вы уже начали чат');
  }
};
```

Пример использования.

Если бот ещё не начал диалог, функция установит состояние isStart в true и попросит пользователя ввести своё имя. Если чат уже начат, бот уведомит об этом.

Функция commandStop завершает сессию общения с ботом. При вызове команды /stop бот отправляет прощальное сообщение и завершает диалог.

Листинг 9 – Пример функции

```
// команда /stop
export const commandStop = (
  setIsStart:
React.Dispatch<React.SetStateAction<boolean>>,
  setBodyMessage:
React.Dispatch<React.SetStateAction<string | undefined>>
) => {
  setIsStart(false);
  setBodyMessage('Всего доброго, если хочешь поговорить
пиши /start');
};
```

Пример использования.

Команда /stop изменяет состояние isStart на false и выводит сообщение о завершении сессии.

Функция commandName используется для приветствия пользователя по имени. Бот запрашивает имя пользователя, сохраняет его и использует в дальнейших сообщениях для создания эффекта персонализированного общения.

Листинг 10 – Пример функции

```
// команда /name
export const commandName = (
  name: string,
  setBodyMessage:
React.Dispatch<React.SetStateAction<string | undefined>>
) => {
  const message =
    'Привет ' + name + ', приятно познакомиться. Я умею
считать, введи числа которые надо посчитать';
  setBodyMessage(message);
};
```

Пример использования.

Если пользователь ввёл своё имя, бот приветствует его по имени и приглашает ввести числа для дальнейших вычислений.

Функция `commandNumber` обрабатывает числовые данные, предоставленные пользователем. Она принимает две строки, преобразует их в числа и сохраняет их для дальнейших математических операций.

Листинг 11 – Пример функции

```
// команда /number
export const commandNumber = (
  firstNumberString: string,
  secondNumberString: string,
  setBodyMessage:
React.Dispatch<React.SetStateAction<string | undefined>>,
  setNumbers:
React.Dispatch<React.SetStateAction<NumberMathTypes |
undefined>>
) => {
  const first = parseInt(firstNumberString, 10);
  const second = parseInt(secondNumberString, 10);
  setNumbers({
    first,
    second
  });
  setBodyMessage('Введите одну из следующих команд: -, +,
*, /');
};
```

Пример использования.

Если пользователь вводит два числа, бот сохраняет их и предлагает ввести команду для выполнения математических операций.

Функция `commandMathAction` выполняет математические операции, такие как сложение, вычитание, умножение и деление. Она принимает введённые числа и команду, выполняет вычисления и выводит результат пользователю.

Листинг 12 – Пример функции

```
// команды для мат действий
export const commandMathAction = (
  numbers: NumberMathTypes,
  isStart: boolean,
  command: string,
  setBodyMessage:
React.Dispatch<React.SetStateAction<string | undefined>>,
```

Продолжение листинга 12

```
        setNumbers:
React.Dispatch<React.SetStateAction<NumberMathTypes |
undefined>>
    ) => {
        if (isStart) {
            if (numbers) {
                const actionMath = `${numbers.first} ${command}
${numbers.second}`;
                setBodyMessage(String(eval(actionMath)));

                setNumbers(undefined);
            } else {
                setBodyMessage(notKnowCommandBody);
            }
        } else if (!numbers) {
            setBodyMessage(notStartChat);
        }
    }
};
```

Пример использования.

Если пользователи ввели числа и команду, бот выполняет указанную математическую операцию и возвращает результат.

Функция `commandWeather` позволяет пользователю узнать текущую погоду на основе его геолокации. Бот запрашивает данные о местоположении и возвращает прогноз погоды, используя API.

Листинг 13 – Пример функции

```
javascript
Копировать код
// команда /weather
export const commandWeather = async (
    setBodyMessage:
React.Dispatch<React.SetStateAction<string | undefined>>
) => {
    const getPosition = async (): Promise<PositionType |
undefined> => {
        let positionRes: PositionType | undefined = undefined;

        const getPositionPromise = (options?: PositionOptions):
Promise<GeolocationPosition> => {
            return new Promise((resolve, reject) =>
                navigator.geolocation.getCurrentPosition(resolve,
reject, options)
            );
        };
    };
};
```

Продолжение листинга 13

```
        if (navigator.geolocation) {
            try {
                const position = await getPositionPromise();
                positionRes = {
                    latitude: position.coords.latitude,
                    longitude: position.coords.longitude
                };
            } catch (err) {
                setBodyMessage('При определении геопозиции
произошла ошибка');
                console.error(err);
            }
        } else {
            setBodyMessage('Включите пожалуйста геолокацию');
        }

        return positionRes;
    };

    const getWeather = async (latitude: number, longitude:
number) => {
        try {
            const req = await fetch(

`https://api.openweathermap.org/data/2.5/forecast?lat=${latitude}
&lon=${longitude}&cnt=10&lang=ru&appid=1168febae541d36c0213f5
b48ce1e406`

            );
            const body = await req.json();

            setBodyMessage(body['list'][9]['weather'][0]['description']);
        } catch (err) {
            setBodyMessage('Произошла ошибка с получением
данных');
            console.error(err);
        }
    };

    const position = await getPosition();
    if (position) {
        getWeather(position.latitude, position.longitude);
    }
};
```

Пример использования.

Когда пользователь запрашивает погоду, бот получает текущее местоположение пользователя и запрашивает данные о погоде с помощью API OpenWeatherMap, выводя описание погоды.

Функция notStart обрабатывает команды, которые не распознаны ботом. Она возвращает сообщение о том, что команда не распознана, и предлагает помощь.

Листинг 14 – Пример функции

```
// для не запланированного функционала
export const notStart = (
  setBodyMessage:
  React.Dispatch<React.SetStateAction<string | undefined>>
) => {
  setBodyMessage(notKnowCommandBody);
};
```

Пример использования.

Если пользователь вводит неизвестную команду, бот использует функцию notStart, чтобы уведомить о том, что команда не распознана, и предложить помощь.

3 Пример использования Чат-бота

В ходе работы над проектом были выполнены и задокументированы примеры взаимодействия пользователя с чат-ботом. Ниже приведены основные этапы работы, демонстрирующие, как бот обрабатывает различные команды и взаимодействует с пользователем.

3.1 Начало общения

Описание: при запуске чат-бота пользователь вводит команду /start. Это действие инициирует диалог с ботом и подготавливает его к дальнейшему взаимодействию. Данный процесс представлен на рисунке 1.

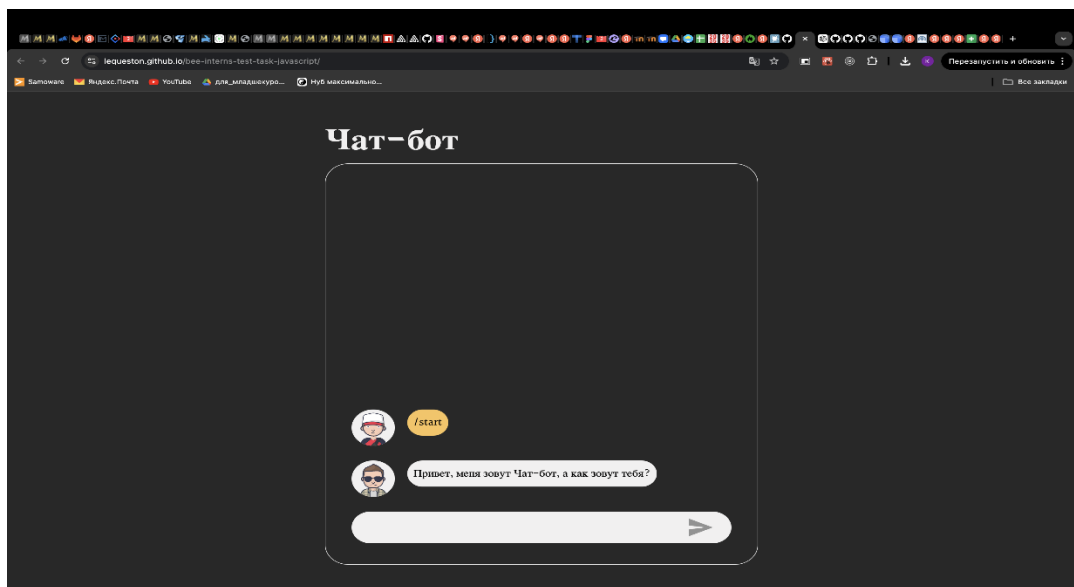


Рисунок 1 – Снимок экрана начала общения с ботом

Функция `commandStart` проверяет состояние `isStart`. Если сессия ещё не начата (`isStart` равно `false`), бот изменяет состояние на `true` и отправляет приветственное сообщение.

Если сессия уже начата, бот уведомляет пользователя, что чат уже начат.

3.2 Запрос имени пользователя

Описание: после инициализации общения, бот запрашивает имя пользователя с помощью команды `/name`. После того как пользователь вводит своё имя, бот сохраняет эту информацию и использует её в дальнейших сообщениях для создания эффекта персонализированного общения. Запрос имени пользователя можно увидеть на рисунке 2.

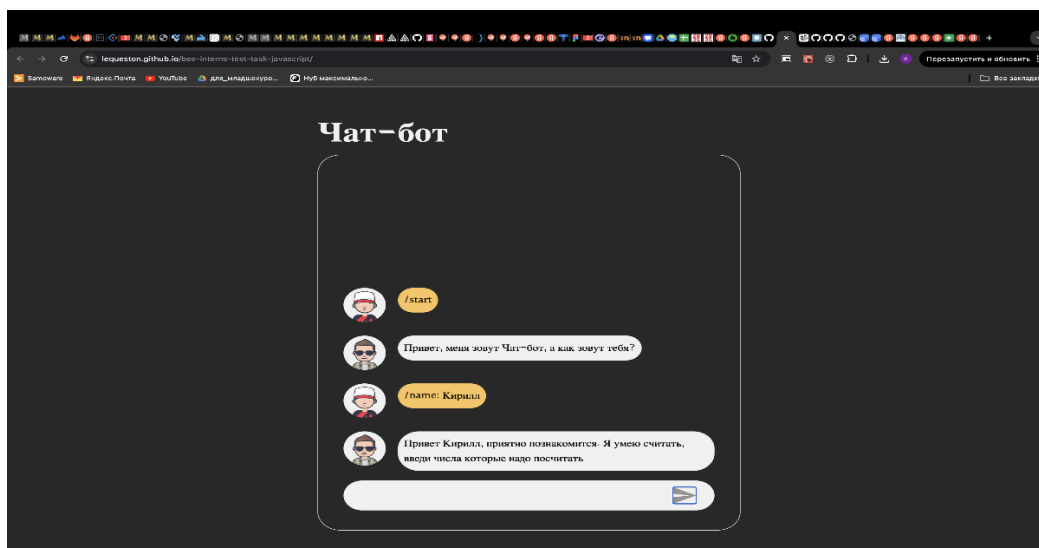


Рисунок 2 – Снимок экрана запроса имени пользователя

Функция `commandName` принимает имя пользователя и формирует персонализированное приветственное сообщение.

Бот приглашает пользователя ввести числа для последующих математических операций.

3.3. Запрос погоды

Описание: Пользователь может запросить текущий прогноз погоды с помощью команды `/weather`. Бот определяет геолокацию пользователя и возвращает прогноз погоды для данного региона. Пример данного взаимодействия представлен на рисунке 3.

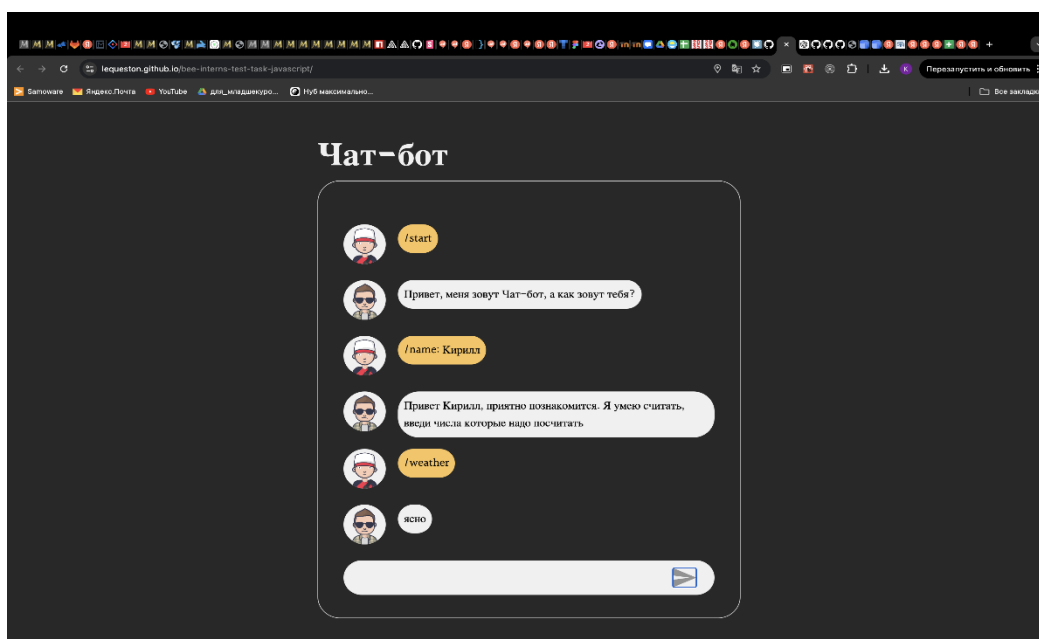


Рисунок 3 – Снимок экрана запроса погоды

Функция `commandWeather` использует API геолокации для определения текущего местоположения пользователя.

Затем бот отправляет запрос к API погоды (например, `OpenWeatherMap`) и возвращает описание текущих погодных условий.

ЗАКЛЮЧЕНИЕ

Разработка чат-бота, описанная в этом отчете, оформленном с учетом требований ГОСТ 7.32.2017, представляет собой успешный пример использования современных технологий для автоматизации рутинных процессов и улучшения взаимодействия с пользователем. В ходе работы был создан прототип, который выполняет базовые функции общения, такие как парсинг команд, обработка ошибок, выполнение математических операций и предоставление информации о погоде. Этот бот может стать основой для создания более сложных и многофункциональных решений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация React [Электронный ресурс]. – Режим доступа: <https://react.dev/learn> (дата обращения: 01.07.2024)
2. Документация JavaScript [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/JavaScript> (дата обращения: 01.07.2024)
3. Документация Github Pages [Электронный ресурс]. – Режим доступа: <https://docs.github.com/ru/pages> (дата обращения: 16.07.2024)