

|  |
| --- |
| **Автономное учреждение  профессионального образования**  **Ханты-Мансийского автономного округа – Югры**  **«СУРГУТСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**  **(АУ «Сургутский политехнический колледж»)** |
|  |
| СТРУКТУРНОЕ ПОДРАЗДЕЛЕНИЕ - 4  (Энергетическое отделение) |
|  |

**Отчет**

**по практической работе**

**«Технология разработки ПО»**

Установка операционной системы на базе ядра Linux на ПК

Выполнил:

студент очной формы обучения

Браславцев Трофим Сергеевич

Сургут 2024

**Введение**

GET и POST запросы – это важные элементы взаимодействия между клиентом и сервером в веб-приложениях. В данной работе будут рассмотрены принципы работы с клиент-серверными запросами.

**Цель работы**

Цель работы – изучить принципы работы с клиент-серверными запросами, научиться отправлять GET и POST запросы и анализировать их ответы.

**Практическая часть**

**Задание 1: Отправка GET-запросов**

**1. Реализуйте отправку GET-запроса к публичному API с использованием библиотеки requests.**

import requests

# URL публичного API

url = "https://jsonplaceholder.typicode.com/posts/1"

# Отправка GET-запроса

response = requests.get(url)

# Проверка статуса ответа

if response.status\_code == 200:

print("GET-запрос успешно выполнен")

# Вывод полученных данных

print(response.json())

else:

print(f"Ошибка: {response.status\_code}")

**2. Проверьте и проанализируйте полученные данные.**

После выполнения кода, я получил ответ от сервера в формате JSON, который можно проанализировать и использовать в дальнейшем.

**Задание 2: Отправка POST-запросов**

**1. Реализуйте отправку POST-запроса для передачи данных на сервер.**

import requests

# URL для отправки POST-запроса

url = "https://jsonplaceholder.typicode.com/posts"

# Данные для отправки

data = {

"title": "Заголовок",

"body": "Тело поста",

"userId": 1

}

# Отправка POST-запроса

response = requests.post(url, json=data)

# Проверка статуса ответа

if response.status\_code == 201:

print("POST-запрос успешно выполнен")

# Вывод полученных данных

print(response.json())

else:

print(f"Ошибка: {response.status\_code}")

**2. Проверьте корректность отправки данных и ответа сервера.**

После выполнения кода, я получил ответ от сервера, подтверждающий успешное создание ресурса.

**Задание 3: Анализ HTTP-заголовков**

**1. Реализуйте программу для анализа заголовков HTTP-запросов и ответов.**

import requests

# URL для GET-запроса

url = "https://jsonplaceholder.typicode.com/posts/1"

# Отправка GET-запроса

response = requests.get(url)

# Вывод заголовков ответа

print("Заголовки ответа:")

for header, value in response.headers.items():

print(f"{header}: {value}")

# Вывод информации о типе содержимого и коде ответа

print(f"Тип содержимого: {response.headers['Content-Type']}")

print(f"Код ответа: {response.status\_code}")

**2. Выведите информацию о типе содержимого и коде ответа.**

После выполнения кода, вы получите информацию о заголовках ответа, типе содержимого и коде ответа.

**Задание 4: Обработка ошибок**

**1. Реализуйте обработку ошибок при выполнении GET и POST запросов (например, 404 и 500 ошибки).**

import requests

# URL для GET-запроса

url = "https://jsonplaceholder.typicode.com/nonexistent"

try:

# Отправка GET-запроса

response = requests.get(url)

response.raise\_for\_status() # Выбрасывает исключение для ошибочных статусов

print("GET-запрос успешно выполнен")

print(response.json())

except requests.exceptions.HTTPError as err:

print(f"Ошибка HTTP: {err}")

except requests.exceptions.RequestException as err:

print(f"Ошибка запроса: {err}")

**2. Напишите программу для повторной отправки запроса в случае ошибки.**

import requests

# URL для GET-запроса

url = "https://jsonplaceholder.typicode.com/nonexistent"

def send\_request(url, retries=3):

for attempt in range(retries):

try:

response = requests.get(url)

response.raise\_for\_status()

print("GET-запрос успешно выполнен")

print(response.json())

return response

except requests.exceptions.HTTPError as err:

print(f"Ошибка HTTP: {err}")

if attempt < retries - 1:

print("Повторная попытка...")

else:

print("Превышено количество попыток")

except requests.exceptions.RequestException as err:

print(f"Ошибка запроса: {err}")

break

send\_request(url)

**Задание 5: Параметры запросов**

**1. Отправьте GET-запрос с параметрами (например, фильтры или пагинация).**

import requests

# URL для GET-запроса с параметрами

url = "https://jsonplaceholder.typicode.com/posts"

params = {

"userId": 1,

"\_start": 0,

"\_limit": 5

}

# Отправка GET-запроса с параметрами

response = requests.get(url, params=params)

# Проверка статуса ответа

if response.status\_code == 200:

print("GET-запрос с параметрами успешно выполнен")

# Вывод полученных данных

print(response.json())

else:

print(f"Ошибка: {response.status\_code}")

**2. Реализуйте POST-запрос с передачей данных в теле запроса.**

import requests

# URL для POST-запроса

url = "https://jsonplaceholder.typicode.com/posts"

# Данные для отправки в теле запроса

data = {

"title": "Заголовок",

"body": "Тело поста",

"userId": 1

}

# Отправка POST-запроса с данными в теле

response = requests.post(url, json=data)

# Проверка статуса ответа

if response.status\_code == 201:

print("POST-запрос с данными в теле успешно выполнен")

# Вывод полученных данных

print(response.json())

else:

print(f"Ошибка: {response.status\_code}")

**Вывод**

В данной работе были изучены принципы работы с клиент-серверными запросами, я научился отправлять GET и POST запросы, анализировать их ответы, обрабатывать ошибки и работать с параметрами запросов.