**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

по дисциплине «Программирование сетевых приложений»

на тему:

**«Разработка HTTP­сервисов»**

Выполнил: студент гр. ИТИ-41

Федоренко А.В.

Принял: преподаватель

Карабчикова Е.А.

Гомель 2020

**Цель работы:** научиться разрабатывать приложения с использованием web-сокетов.

**Задание**: Разработать HTTP­сервис по выбранной теме ­ реализовать CRUD (Create-Read-Update-Delete) операции.

Обязательно оолжны использоваться запросы в стиле REST:

– GET для получения данных;

– POST для создания ресурса (сохранения данных);

– PUT для обновления ресурса (обновление данных);

– DELETE для удаления ресурса (удаления строк).

На сервере использовать любую БД для хранения данных. Клиентское приложение: это html­страница, где запросы (вызовы сервиса) осуществляются с использованием AJAX.

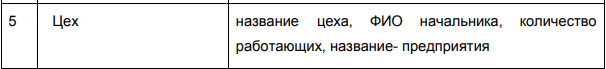


Рисунок 1 – Вариант задания



Рисунок 2 – html-страница

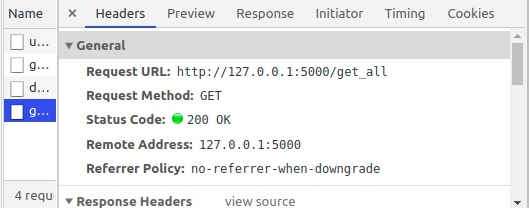


Рисунок 3 – GET-запрос

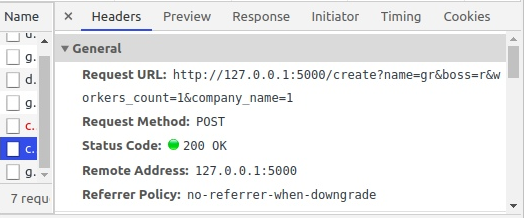


Рисунок 4 – POST-запрос

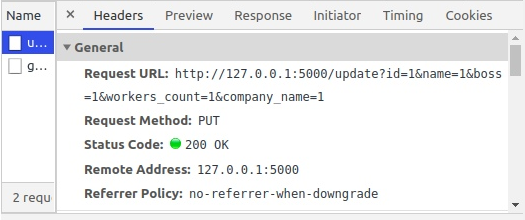


Рисунок 5 – PUT-запрос



Рисунок 5 – DELETE-запрос

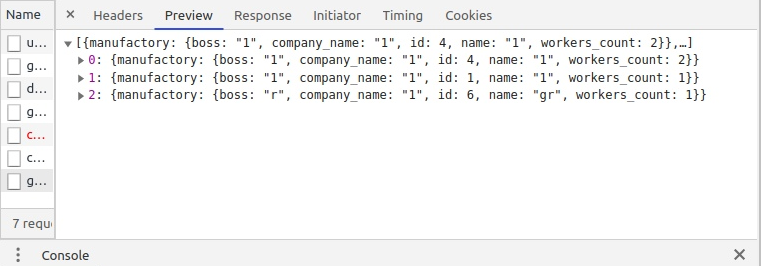


Рисунок 6 – Ответ в JSON-формате

**Вывод:** в ходе выполнения лабораторной работы был разработан http­сервис с реализацией CRUD.

**Приложение А**

**Листинг программы**

Класс Manufacture:

from app.db import db

class Manufactory(db.Model):

\_\_tablename\_\_ = 'manufactory'

id=db.Column('id', db.Integer, primary\_key=True)

name=db.Column('name', db.String(32))

boss=db.Column('boss', db.String(32))

workers\_count=db.Column('workers\_count', db.Integer)

company\_name=db.Column('company\_name', db.String(32))

@property

def serialize(self):

return {'manufactory': {'id': self.id, 'name': self.name, 'boss': self.boss, 'workers\_count': self.workers\_count,

'company\_name': self.company\_name}}

@property

def serialize\_many2many(self):

return [item.serialize for item in self.many2many]

create:

from flask import request, jsonify

from app import app, app\_logger

from app.db import db

from app.models.manufactory import Manufactory

logger = app\_logger.get\_logger(\_\_name\_\_)

@app.route('/create', methods=["POST"])

def create():

response = None

try:

logger.info('Handling request: Adding a record to a table')

manufactory\_name = request.args.get('name')

manufactory\_boss = request.args.get('boss')

manufactory\_workers\_count = request.args.get('workers\_count')

manufactory\_company\_name = request.args.get('company\_name')

manufactory = Manufactory(name=manufactory\_name, boss=manufactory\_boss,

workers\_count=manufactory\_workers\_count,

company\_name=manufactory\_company\_name)

db.session.add(manufactory)

db.session.commit()

response = jsonify({'success': 'Record added successfully.'})

response.status\_code = 200

except Exception as e:

db.session.rollback()

response = jsonify({'error': 'The request had invalid values.'})

response.status\_code = 400

logger.error(str(e))

finally:

return response

delete:

from flask import request, jsonify

from app import app, app\_logger

from app.db import db

from app.models.manufactory import Manufactory

logger = app\_logger.get\_logger(\_\_name\_\_)

@app.route('/delete', methods=["DELETE"])

def delete():

response = None

try:

logger.info('Handling request: Delete record by id')

mid = request.args.get('id')

db.session.query(Manufactory).filter\_by(id=mid).delete()

db.session.commit()

response = jsonify({'success': 'Record successfully deleted.'})

response.status\_code = 200

except Exception as e:

db.session.rollback()

response = jsonify({'error': 'The request had invalid values.'})

response.status\_code = 400

logger.error(str(e))

finally:

return response

get\_all:

from flask import jsonify

from app import app, app\_logger

from app.db import db

from app.models.manufactory import Manufactory

logger = app\_logger.get\_logger(\_\_name\_\_)

@app.route('/get\_all', methods=["GET"])

def get\_all():

try:

logger.info('Handling request: Getting a list of records')

query\_list = db.session.query(Manufactory)

response = jsonify([i.serialize for i in query\_list.all()])

return response

except Exception as e:

logger.error(str(e))

update:

from flask import request, jsonify

from app import app, app\_logger

from app.db import db

from app.models.manufactory import Manufactory

logger = app\_logger.get\_logger(\_\_name\_\_)

@app.route('/update', methods=["PUT"])

def update():

response = None

try:

logger.info('Handling request: Updating a record in a table')

manufactory\_id = request.args.get('id')

manufactory\_name = request.args.get('name')

manufactory\_boss = request.args.get('boss')

manufactory\_workers\_count = request.args.get('workers\_count')

manufactory\_company\_name = request.args.get('company\_name')

manufactory = db.session.query(Manufactory).get(manufactory\_id)

manufactory.name = manufactory\_name

manufactory.boss = manufactory\_boss

manufactory.workers\_count = manufactory\_workers\_count

manufactory.company\_name = manufactory\_company\_name

db.session.commit()

response = jsonify({'success': 'Record successfully updated.'})

response.status\_code = 200

except Exception as e:

db.session.rollback()

response = jsonify({'error': 'The request had invalid values.'})

response.status\_code = 400

logger.error(str(e))

finally:

return response

index.js:

document.addEventListener('DOMContentLoaded', () => {

get\_all();

});

function get\_all(){

// Инициализировать новый запрос

var request = new XMLHttpRequest();

request.open('GET', '/get\_all');

// Функция обратного вызова, когда запрос завершен

request.onload = () => {

// Извлечение данных JSON из запроса

const data = JSON.parse(request.responseText);

fillTable(data);

}

// Послать запрос

request.send(null);

}

function fillTable(data){

var table = document.getElementById("table");

var text = "<tr><td>Id</td><td>Name</td><td>Boss</td><td>Workers Count</td><td>Company</td><td>Update</td><td>Delete</td></tr>";

for(let i = 0; i < data.length; i++){

let id = data[i].manufactory.id;

let name = data[i].manufactory.name;

let boss = data[i].manufactory.boss;

let workers\_count = data[i].manufactory.workers\_count;

let company\_name = data[i].manufactory.company\_name;

text += "<tr>" +

`<td>${id}</td>` +

"<td>" +

`<input type=\"text\" class=\"\input\_text\" id=\"name${id}\_input\" value=\"${name}\">` +

"</td>" +

"<td>" +

`<input type=\"text\" class=\"\input\_text\" id=\"boss${id}\_input\" value=\"${boss}\">` +

"</td>" +

"<td>" +

`<input type=\"text\" class=\"\input\_text\" id=\"workers\_count${id}\_input\" value=\"${workers\_count}\">` +

"</td>" +

"<td>" +

`<input type=\"text\" class=\"\input\_text\" id=\"company\_name${id}\_input\" value=\"${company\_name}\">` +

"</td>" +

"<td>" +

`<a class="green" onclick=\"update\_manufactory(${id})\">Update</a>` +

"</td><td>" +

`<a class="green" onclick=\"remove\_manufactory(${id})\">Remove</a>` +

"</td></tr>";

}

text += "<tr><td></td><td><input type=\"text\" class=\"\input\_text\" id=\"name\_input\">" +

"</td><td><input type=\"text\" class=\"\input\_text\" id=\"boss\_input\">" +

"</td><td><input type=\"text\" class=\"\input\_text\" id=\"workers\_count\_input\">" +

"</td><td><input type=\"text\" class=\"\input\_text\" id=\"company\_name\_input\">" +

"</td><td><a class=\"green\" onclick=\"create\_manufactory()\">Add</a>" +

"</td><td></td></tr>"

table.innerHTML = text;

}

function remove\_manufactory(id){

var request = new XMLHttpRequest();

request.open("DELETE", '/delete?id=' + id);

// Функция обратного вызова, когда запрос завершен

request.onload = () => {

response = JSON.parse(request.responseText);

if(request.status == '200'){

get\_all();

WriteMessage(response.success, true);

}

else{

WriteMessage(response.error, false);

}

}

request.send();

}

function create\_manufactory(){

var request = new XMLHttpRequest();

var name = document.getElementById("name\_input").value;

var boss = document.getElementById("boss\_input").value;

var workers\_count = document.getElementById("workers\_count\_input").value;

var company\_name = document.getElementById("company\_name\_input").value;

request.open("POST", `/create?name=${name}&boss=${boss}&workers\_count=${workers\_count}&company\_name=${company\_name}`);

// Функция обратного вызова, когда запрос завершен

request.onload = () => {

response = JSON.parse(request.responseText);

if(request.status == '200'){

get\_all();

WriteMessage(response.success, true);

}

else{

WriteMessage(response.error, false);

}

}

request.send();

}

function update\_manufactory(id){

var request = new XMLHttpRequest();

var name = document.getElementById(`name${id}\_input`).value;

var boss = document.getElementById(`boss${id}\_input`).value;

var workers\_count = document.getElementById(`workers\_count${id}\_input`).value;

var company\_name = document.getElementById(`company\_name${id}\_input`).value;

request.open("PUT", `/update?id=${id}&name=${name}&boss=${boss}&workers\_count=${workers\_count}&company\_name=${company\_name}`);

// Функция обратного вызова, когда запрос завершен

request.onload = () => {

response = JSON.parse(request.responseText);

if(request.status == '200'){

get\_all();

WriteMessage(response.success, true);

}

else{

WriteMessage(response.error, false);

}

}

request.send();

}

function WriteMessage(message, success){

var now = new Date();

status\_text = document.getElementById("answertext");

status\_text.style.color = success ? "green" : "red";

Year = now.getFullYear();

Month = now.getMonth();

Day = now.getDate();

Hour = now.getHours();

Minutes = now.getMinutes();

Seconds = now.getSeconds();

status\_text.innerHTML = `${Year}.${Month}.${Day} ${Hour}:${Minutes}:${Seconds} - ${message}`;

}