**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

Гончар А.С.

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.

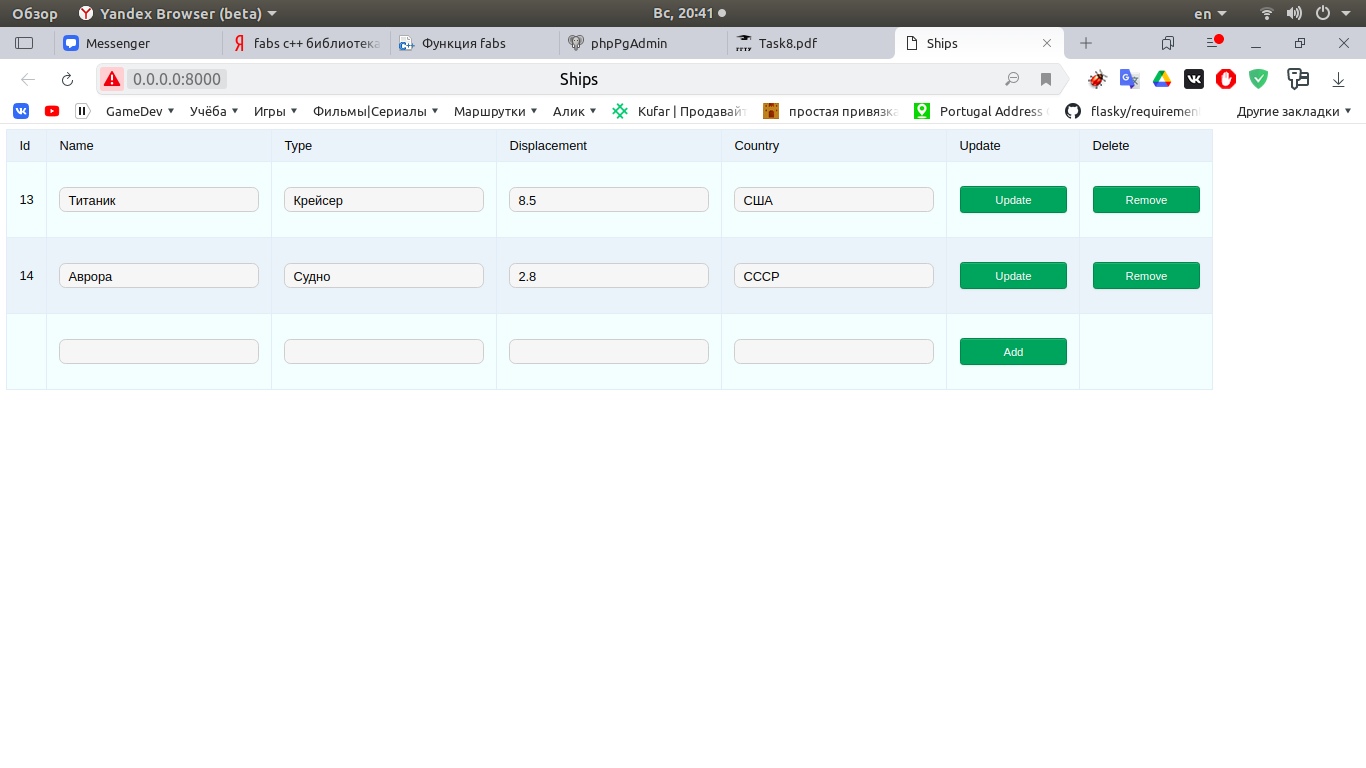


Рисунок 1 – html-страница

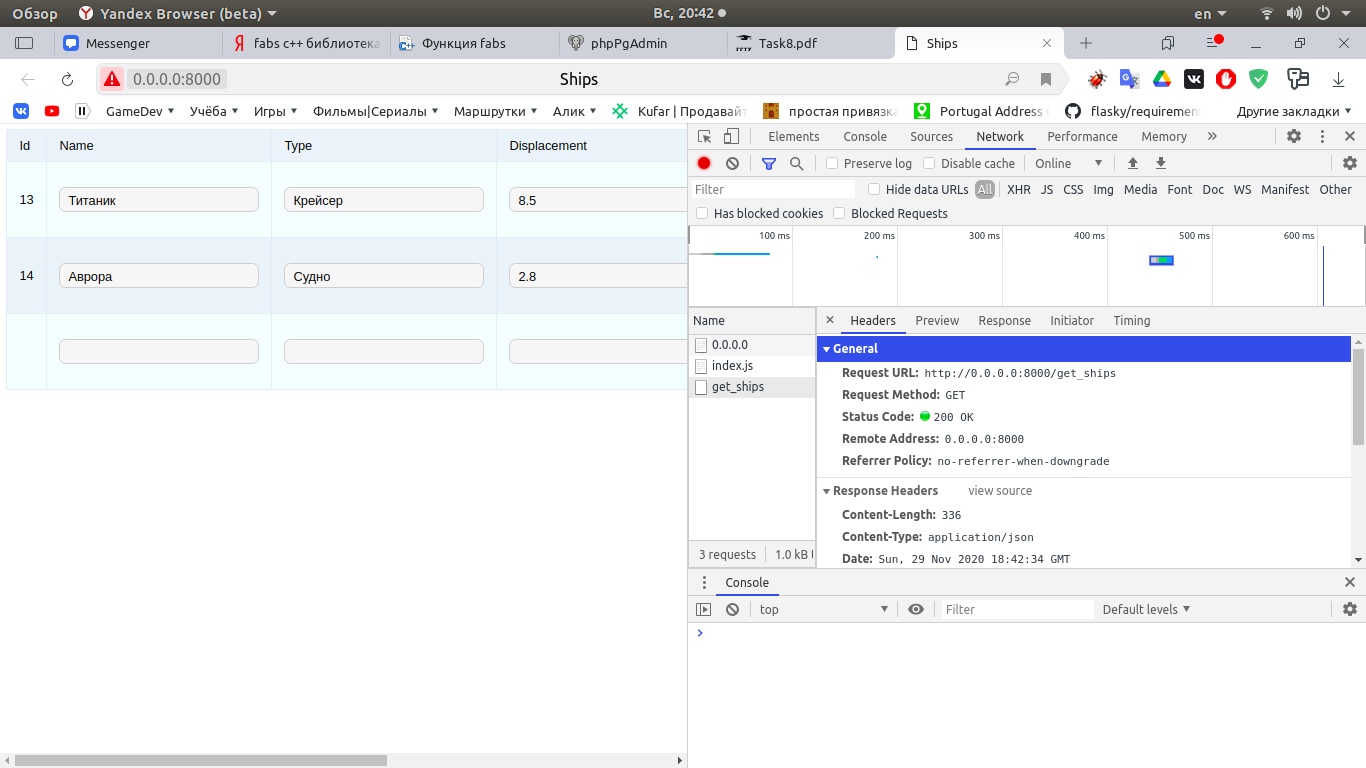


Рисунок 2 – GET-запрос

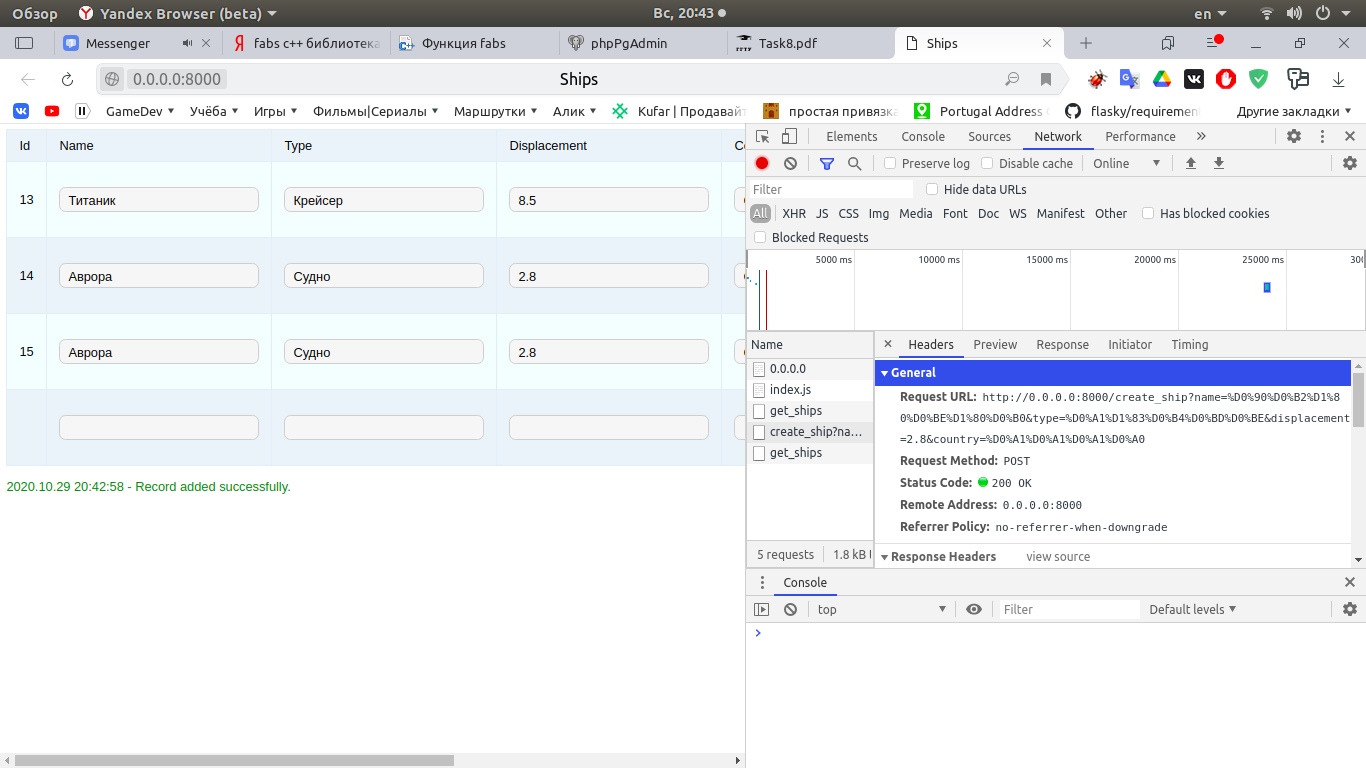


Рисунок 3 – POST-запрос

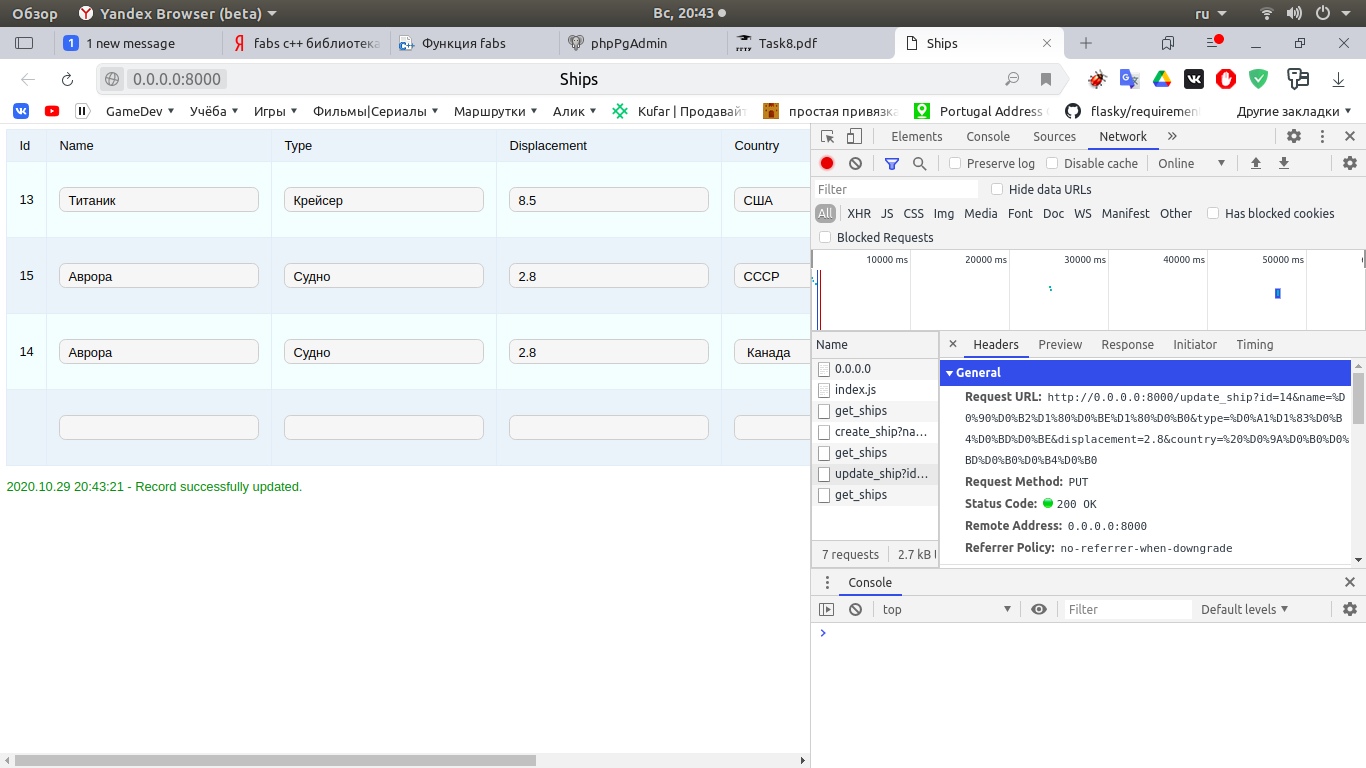


Рисунок 4 – PUT-запрос

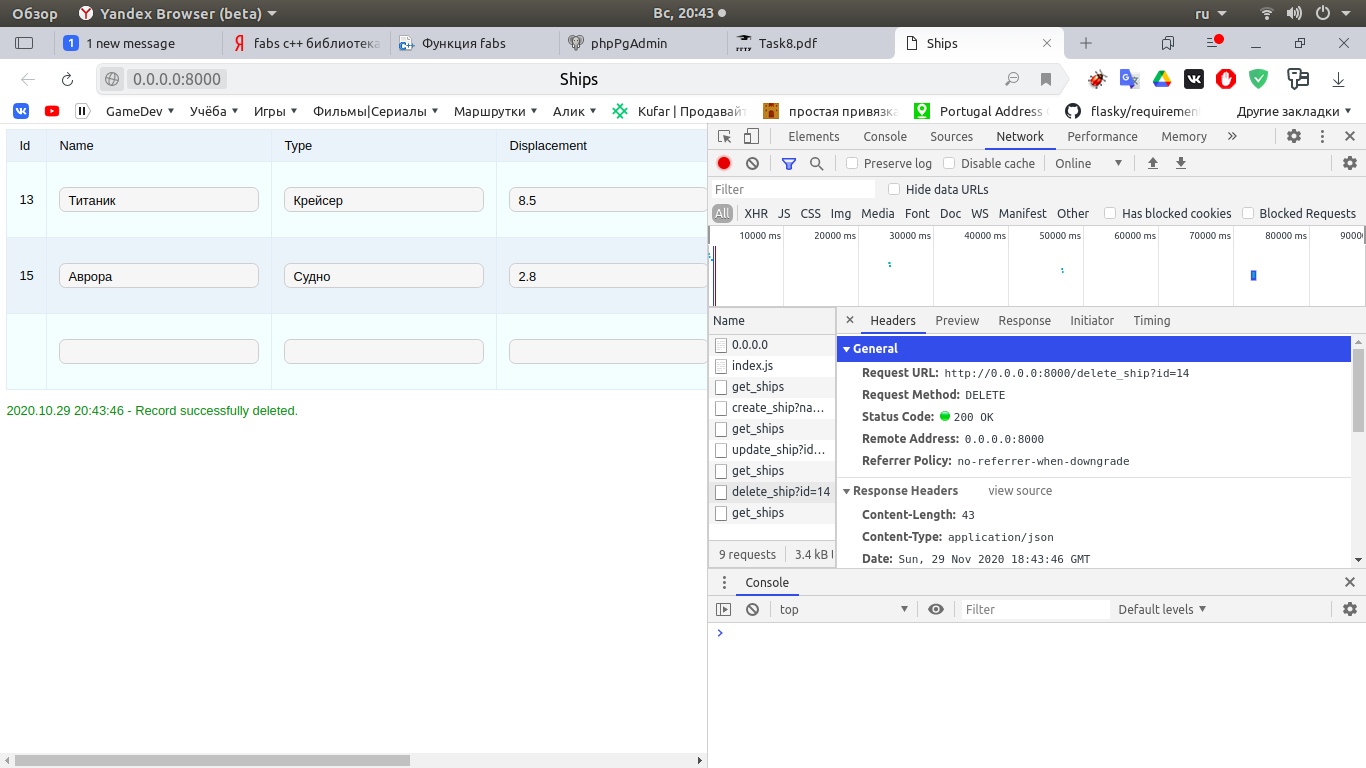


Рисунок 5 – DELETE-запрос

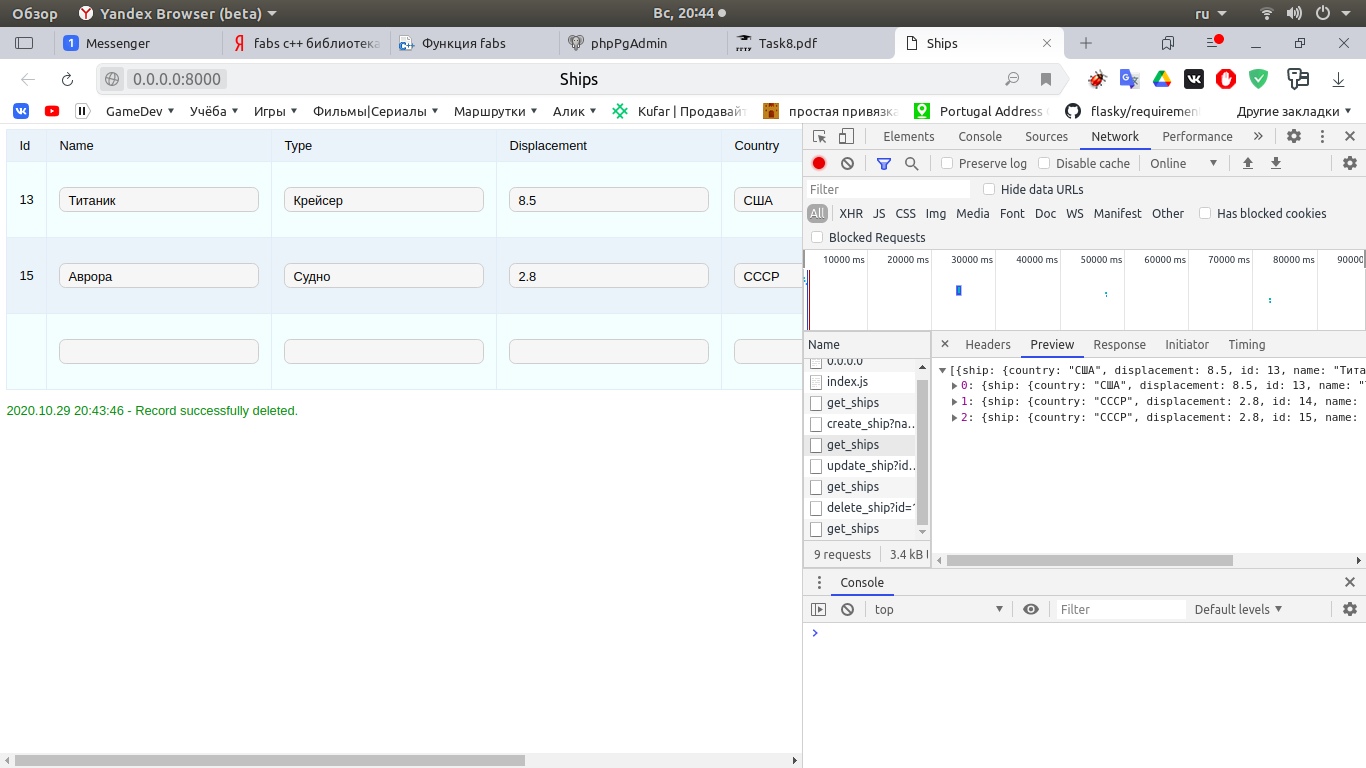


Рисунок 6 – Ответ в JSON-формате

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

**Приложение А**

**Листинг программы**

Класс Ship:

from app.db import db

class Ship(db.Model):

\_\_tablename\_\_ = 'ship'

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

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

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

displacement=db.Column('displacement', db.Float)

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

def validate(self):

try:

float(self.displacement)

except:

raise Exception('Invalid data.')

@property

def serialize(self):

return {'ship': {'id': self.id, 'name': self.name, 'type': self.type,

'displacement': self.displacement, 'country': self.country }}

@property

def serialize\_many2many(self):

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

create\_ship:

from flask import request, jsonify

from app import app, app\_logger

from app.db import db

from app.models.ship import Ship

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

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

def create\_ship():

response = None

try:

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

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

ship\_type = request.args.get('type')

ship\_displacement = request.args.get('displacement')

ship\_country = request.args.get('country')

ship = Ship(name=ship\_name, type=ship\_type,

displacement=ship\_displacement, country=ship\_country)

ship.validate()

db.session.add(ship)

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\_ship:

from flask import request, jsonify

from app import app, app\_logger

from app.db import db

from app.models.ship import Ship

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

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

def delete\_ship():

response = None

try:

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

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

db.session.query(Ship).filter\_by(id=ship\_id).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\_ships:

from flask import jsonify

from app import app, app\_logger

from app.db import db

from app.models.ship import Ship

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

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

def get\_ships():

try:

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

query\_ship\_list = db.session.query(Ship)

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

return response

except Exception as e:

logger.error(str(e))

update\_ship:

from flask import request, jsonify

from app import app, app\_logger

from app.db import db

from app.models.ship import Ship

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

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

def update\_ship():

response = None

try:

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

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

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

ship\_type = request.args.get('type')

ship\_displacement = request.args.get('displacement')

ship\_country = request.args.get('country')

ship = db.session.query(Ship).get(ship\_id)

ship.name = ship\_name

ship.type = ship\_type

ship.displacement = ship\_displacement

ship.country = ship\_country

ship.validate()

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\_ships();

});

function get\_ships(){

// Инициализировать новый запрос

var request = new XMLHttpRequest();

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

// Функция обратного вызова, когда запрос завершен

request.onload = () => {

// Извлечение данных JSON из запроса

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

fillShipsTable(data);

}

// Послать запрос

request.send(null);

}

function fillShipsTable(data){

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

var text = "<tr><td>Id</td><td>Name</td><td>Type</td><td>Displacement</td><td>Country</td><td>Update</td><td>Delete</td></tr>";

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

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

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

let type = data[i].ship.type;

let displacement = data[i].ship.displacement;

let country = data[i].ship.country;

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=\"type${id}\_input\" value=\"${type}\">` +

"</td>" +

"<td>" +

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

"</td>" +

"<td>" +

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

"</td>" +

"<td>" +

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

"</td><td>" +

`<a class="green" onclick=\"remove\_ship(${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=\"type\_input\">" +

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

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

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

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

table.innerHTML = text;

}

function remove\_ship(id){

var request = new XMLHttpRequest();

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

// Функция обратного вызова, когда запрос завершен

request.onload = () => {

response = JSON.parse(request.responseText);

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

get\_ships();

WriteMessage(response.success, true);

}

else{

WriteMessage(response.error, false);

}

}

request.send();

}

function create\_ship(){

var request = new XMLHttpRequest();

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

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

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

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

request.open("POST", `/create\_ship?name=${name}&type=${type}&displacement=${displacement}&country=${country}`);

// Функция обратного вызова, когда запрос завершен

request.onload = () => {

response = JSON.parse(request.responseText);

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

get\_ships();

WriteMessage(response.success, true);

}

else{

WriteMessage(response.error, false);

}

}

request.send();

}

function update\_ship(id){

var request = new XMLHttpRequest();

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

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

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

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

request.open("PUT", `/update\_ship?id=${id}&name=${name}&type=${type}&displacement=${displacement}&country=${country}`);

// Функция обратного вызова, когда запрос завершен

request.onload = () => {

response = JSON.parse(request.responseText);

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

get\_ships();

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}`;

}

index.html:

<html>

<head>

<title>Ships</title>

<link href="{{ url\_for('static', filename = 'css/index.css') }}" rel="stylesheet" type="text/css" media="all">

</head>

<body>

<div id="block\_custable">

<table id="table" class="table\_blur">

</table>

</div>

<p id="answertext"></p>

<script type="text/javascript" src="{{ url\_for('static', filename = 'js/index.js') }}" ></script>

</body>

</html>