**Отчёт по**

**Лабораторной работе**

**Шумов Владислав ИСП.23.1А**

Предметная область: *Судоходство.* Минимальный список характеристик:

* код корабля,
* название корабля,
* водоизмещение,
* порт приписки,
* капитан,
* номер причала,
* цель посещения.

1. **Создание проекта и файла** **main.py.** Создан Python-проект, в котором main.py реализует серверную часть. Используется библиотека FastAPI. Сервер запускается локально, обрабатывая API-запросы.

**Код main.py:**

from fastapi import FastAPI, HTTPException, Request  
from fastapi.responses import HTMLResponse, Response  
from docx import Document  
from io import BytesIO  
from typing import List, Dict  
from models import Ship, CreateShip  
  
app = FastAPI(title="Ship Management Web")  
  
ship\_list = []  
  
@app.get("/", response\_class=HTMLResponse)  
async def read\_root():  
 with open("templates/index.html", "r", encoding="utf-8") as f:  
 html\_content = f.read()  
 return HTMLResponse(content=html\_content)  
  
@app.get("/ships")  
async def get\_ships() -> List:  
 return ship\_list  
  
@app.get("/ships/download")  
async def export\_ships\_to\_word():  
 try:  
 if not ship\_list:  
 raise HTTPException(status\_code=404, detail="No ships available for export")  
   
 doc = Document()  
 doc.add\_heading('Ship Fleet Report', 0)  
 doc.add\_paragraph(f"Total ships in fleet: {len(ship\_list)}")  
 doc.add\_paragraph()  
  
 for i, ship in enumerate(ship\_list, 1):  
 if i > 1:  
 doc.add\_paragraph("-" \* 50)  
 doc.add\_paragraph()  
   
 doc.add\_heading(f'Ship #{ship.id}: {ship.name}', level=2)  
 doc.add\_paragraph(f"ID: {ship.id}")  
 doc.add\_paragraph(f"Ship Name: {ship.name}")  
 doc.add\_paragraph(f"Displacement: {ship.displacement} tons")  
 doc.add\_paragraph(f"Home Port: {ship.port}")  
 doc.add\_paragraph(f"Captain: {ship.captain}")  
 doc.add\_paragraph(f"Berth Number: {ship.berth\_num}")  
 doc.add\_paragraph(f"Destination: {ship.target}")  
  
 buffer = BytesIO()  
 doc.save(buffer)  
 buffer.seek(0)  
  
 return Response(  
 content=buffer.getvalue(),  
 media\_type='application/vnd.openxmlformats-officedocument.wordprocessingml.document',  
 headers={'Content-Disposition': 'attachment; filename="ship\_fleet\_report.docx"'}  
 )  
 except Exception as e:  
 print(f"Error in export\_ships\_to\_word: {e}")  
 raise HTTPException(status\_code=500, detail=f"Error generating report: {str(e)}")  
  
@app.get("/ships/{ship\_id}")  
async def get\_ship(ship\_id: int) -> Ship:  
 for ship in ship\_list:  
 if ship\_id == ship.id:  
 return ship  
 raise HTTPException(status\_code=404, detail="ship not found")  
  
@app.post("/ships")  
async def add\_ship(ship: CreateShip) -> Ship:  
 new\_id = max([s.id for s in ship\_list], default=0) + 1  
  
 ship\_response = Ship(id=new\_id,  
 name=ship.name,  
 displacement=ship.displacement,  
 port=ship.port,  
 captain=ship.captain,  
 berth\_num=ship.berth\_num,  
 target=ship.target)  
   
 ship\_list.append(ship\_response)  
 return ship\_response  
  
@app.put("/ships/{ship\_id}")  
async def edit\_ship(ship\_id: int, ship\_data: CreateShip) -> Ship:  
 for index, ship in enumerate(ship\_list):   
 if ship\_id == ship.id:  
 edit\_ship = Ship(  
 id=ship.id,   
 name=ship\_data.name,  
 displacement=ship\_data.displacement,  
 port=ship\_data.port,  
 captain=ship\_data.captain,  
 berth\_num=ship\_data.berth\_num,  
 target=ship\_data.target  
 )  
   
 ship\_list[index] = edit\_ship  
 return edit\_ship   
 raise HTTPException(status\_code=404, detail="ship not found")  
  
@app.delete("/ships/{ship\_id}")   
async def delete\_ship(ship\_id: int) -> Dict:  
 for index, ship in enumerate(ship\_list):   
 if ship\_id == ship.id:  
 del ship\_list[index]   
 return {"data": "ship deleted"}   
 raise HTTPException(status\_code=404, detail="ship not found")  
  
@app.get("/ships/download/{ship\_id}")  
async def export\_ship\_to\_word(ship\_id: int):  
 try:  
 for ship in ship\_list:  
 if ship\_id == ship.id:  
 doc = Document()  
 doc.add\_heading(f'Ship Report: {ship.name}', 0)  
   
 *# Данные корабля* ship\_data = [  
 ("ID", str(ship.id)),  
 ("Ship Name", ship.name),  
 ("Displacement", f"{ship.displacement} tons"),  
 ("Home Port", ship.port),  
 ("Captain", ship.captain),  
 ("Berth Number", str(ship.berth\_num)),  
 ("Destination", ship.target)  
 ]  
   
 for field, value in ship\_data:  
 p = doc.add\_paragraph()  
 p.add\_run(f"{field}: ").bold = True  
 p.add\_run(value)  
  
 buffer = BytesIO()  
 doc.save(buffer)  
 buffer.seek(0)  
  
 filename = f"ship\_{ship.id}\_report.docx"  
   
 return Response(  
 content=buffer.getvalue(),  
 media\_type='application/vnd.openxmlformats-officedocument.wordprocessingml.document',  
 headers={'Content-Disposition': f'attachment; filename="{filename}"'}  
 )  
   
 raise HTTPException(status\_code=404, detail="Ship not found")  
 except Exception as e:  
 print(f"Error in export\_ship\_to\_word: {e}")  
 raise HTTPException(status\_code=500, detail=f"Error generating ship report: {str(e)}")

**Models.py:**

from pydantic import BaseModel  
  
class CreateShip(BaseModel):  
 name: str  
 displacement: float  
 port: str  
 captain: str  
 berth\_num: int  
 target: str  
  
class Ship(CreateShip):  
 id: int

**Получение всех записей (GET):**

|  |
| --- |
| @app.get("/ships") async def get\_ships() -> List:  return ship\_list |

**Получение записи по id (GET):**

|  |
| --- |
| @app.get("/ships/{ship\_id}") async def get\_ship(ship\_id: int) -> Ship:  for ship in ship\_list:  if ship\_id == ship.id:  return ship  raise HTTPException(status\_code=404, detail="ship not found") |

**Добавление нового корабля (POST):**

|  |
| --- |
| @app.post("/ships") async def add\_ship(ship: CreateShip) -> Ship:  new\_id = max([s.id for s in ship\_list], default=0) + 1   ship\_response = Ship(id=new\_id,  name=ship.name,  displacement=ship.displacement,  port=ship.port,  captain=ship.captain,  berth\_num=ship.berth\_num,  target=ship.target)    ship\_list.append(ship\_response)  return ship\_response |

**Редактирование существующего корабля (PUT):**

|  |
| --- |
| @app.put("/ships/{ship\_id}") async def edit\_ship(ship\_id: int, ship\_data: CreateShip) -> Ship:  for index, ship in enumerate(ship\_list):   if ship\_id == ship.id:  edit\_ship = Ship(  id=ship.id,   name=ship\_data.name,  displacement=ship\_data.displacement,  port=ship\_data.port,  captain=ship\_data.captain,  berth\_num=ship\_data.berth\_num,  target=ship\_data.target  )    ship\_list[index] = edit\_ship  return edit\_ship   raise HTTPException(status\_code=404, detail="ship not found") |

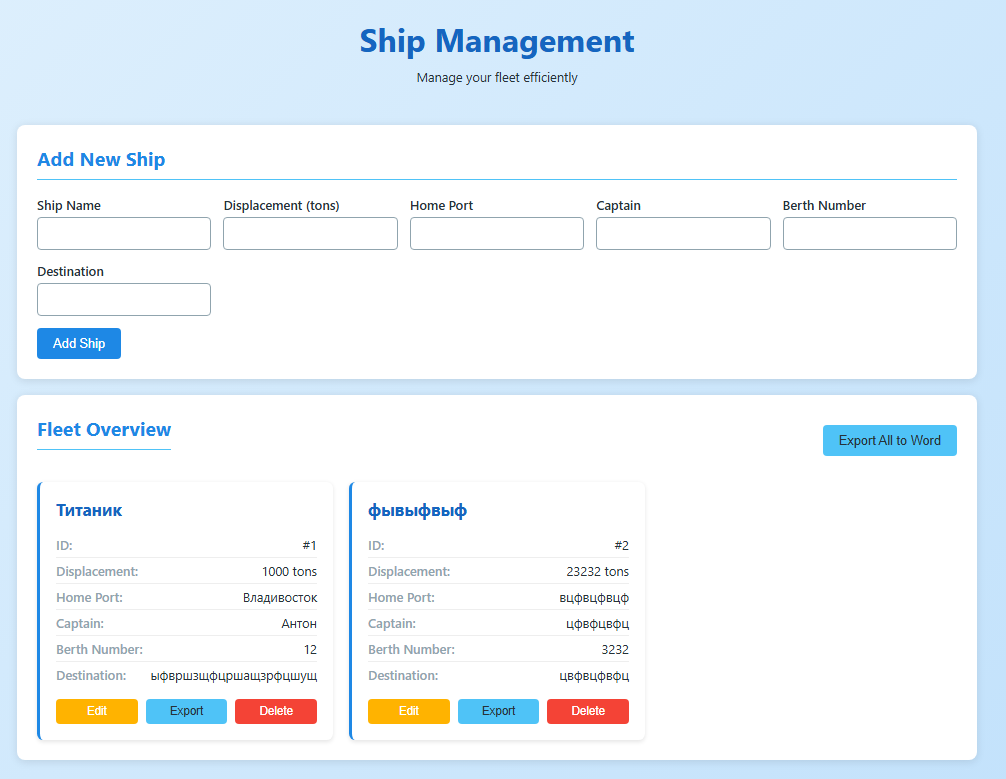
**Удаление корабля (DELETE):**

|  |
| --- |
| @app.delete("/ships/{ship\_id}")  async def delete\_ship(ship\_id: int) -> Dict:  for index, ship in enumerate(ship\_list):   if ship\_id == ship.id:  del ship\_list[index]   return {"data": "ship deleted"}   raise HTTPException(status\_code=404, detail="ship not found") |

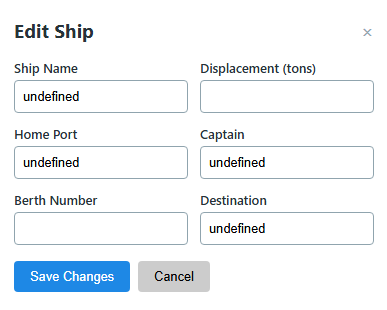
**Файл index.html:**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Ship Management</title>  
 <style>  
 :root {  
 --primary: #1e88e5;  
 --primary-dark: #1565c0;  
 --secondary: #4fc3f7;  
 --dark: #263238;  
 --light: #f5f5f5;  
 --gray: #90a4ae;  
 --success: #4caf50;  
 --danger: #f44336;  
 }  
  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
  
 body {  
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
 background: linear-gradient(135deg, #e3f2fd 0%, #bbdefb 100%);  
 color: var(--dark);  
 min-height: 100vh;  
 padding: 20px;  
 }  
  
 .container {  
 max-width: 1200px;  
 margin: 0 auto;  
 }  
  
 .header {  
 text-align: center;  
 margin-bottom: 30px;  
 padding: 20px;  
 }  
  
 .header h1 {  
 color: var(--primary-dark);  
 font-size: 2.5rem;  
 margin-bottom: 10px;  
 }  
  
 .card {  
 background: white;  
 border-radius: 10px;  
 padding: 25px;  
 margin-bottom: 20px;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 }  
  
 .card h2 {  
 color: var(--primary);  
 margin-bottom: 20px;  
 border-bottom: 2px solid var(--secondary);  
 padding-bottom: 10px;  
 }  
  
 .form-grid {  
 display: grid;  
 grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));  
 gap: 15px;  
 margin-bottom: 15px;  
 }  
  
 .form-group {  
 display: flex;  
 flex-direction: column;  
 }  
  
 label {  
 margin-bottom: 5px;  
 color: var(--dark);  
 font-weight: 500;  
 }  
  
 input {  
 padding: 10px;  
 border: 1px solid var(--gray);  
 border-radius: 5px;  
 font-size: 1rem;  
 }  
  
 input:focus {  
 outline: none;  
 border-color: var(--primary);  
 }  
  
 .btn {  
 padding: 10px 20px;  
 border: none;  
 border-radius: 5px;  
 font-size: 1rem;  
 cursor: pointer;  
 transition: all 0.3s;  
 }  
  
 .btn-primary {  
 background: var(--primary);  
 color: white;  
 }  
  
 .btn-primary:hover {  
 background: var(--primary-dark);  
 }  
  
 .btn-export {  
 background: var(--secondary);  
 color: var(--dark);  
 }  
  
 .btn-export:hover {  
 background: #29b6f6;  
 }  
  
 .btn-edit {  
 background: #ffb300;  
 color: white;  
 }  
  
 .btn-edit:hover {  
 background: #ffa000;  
 }  
  
 .btn-delete {  
 background: var(--danger);  
 color: white;  
 }  
  
 .btn-delete:hover {  
 background: #d32f2f;  
 }  
  
 .ships-grid {  
 display: grid;  
 grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));  
 gap: 20px;  
 }  
  
 .ship-card {  
 background: white;  
 border-radius: 8px;  
 padding: 20px;  
 border-left: 4px solid var(--primary);  
 box-shadow: 0 2px 5px rgba(0,0,0,0.1);  
 }  
  
 .ship-card h3 {  
 color: var(--primary-dark);  
 margin-bottom: 15px;  
 font-size: 1.3rem;  
 }  
  
 .ship-info {  
 margin-bottom: 15px;  
 }  
  
 .ship-field {  
 display: flex;  
 justify-content: space-between;  
 padding: 5px 0;  
 border-bottom: 1px solid #eee;  
 }  
  
 .ship-field:last-child {  
 border-bottom: none;  
 }  
  
 .field-label {  
 color: var(--gray);  
 font-weight: 500;  
 }  
  
 .ship-actions {  
 display: flex;  
 gap: 10px;  
 }  
  
 .ship-actions .btn {  
 flex: 1;  
 padding: 8px 12px;  
 font-size: 0.9rem;  
 }  
  
 .empty-state {  
 text-align: center;  
 padding: 40px;  
 color: var(--gray);  
 grid-column: 1 / -1;  
 }  
  
 .modal {  
 display: none;  
 position: fixed;  
 top: 0;  
 left: 0;  
 width: 100%;  
 height: 100%;  
 background: rgba(0,0,0,0.5);  
 z-index: 1000;  
 }  
  
 .modal-content {  
 background: white;  
 margin: 50px auto;  
 padding: 25px;  
 border-radius: 10px;  
 width: 90%;  
 max-width: 500px;  
 }  
  
 .modal-header {  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 margin-bottom: 20px;  
 }  
  
 .close {  
 font-size: 24px;  
 cursor: pointer;  
 color: var(--gray);  
 }  
  
 .form-actions {  
 display: flex;  
 gap: 10px;  
 margin-top: 20px;  
 }  
  
 @media (max-width: 768px) {  
 .form-grid {  
 grid-template-columns: 1fr;  
 }  
   
 .ships-grid {  
 grid-template-columns: 1fr;  
 }  
   
 .ship-actions {  
 flex-direction: column;  
 }  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <header class="header">  
 <h1>Ship Management</h1>  
 <p>Manage your fleet efficiently</p>  
 </header>  
  
 <div class="card">  
 <h2>Add New Ship</h2>  
 <form id="addShipForm">  
 <div class="form-grid">  
 <div class="form-group">  
 <label for="name">Ship Name</label>  
 <input type="text" id="name" name="name" required>  
 </div>  
 <div class="form-group">  
 <label for="displacement">Displacement (tons)</label>  
 <input type="number" id="displacement" name="displacement" step="0.1" required>  
 </div>  
 <div class="form-group">  
 <label for="port">Home Port</label>  
 <input type="text" id="port" name="port" required>  
 </div>  
 <div class="form-group">  
 <label for="captain">Captain</label>  
 <input type="text" id="captain" name="captain" required>  
 </div>  
 <div class="form-group">  
 <label for="berth\_num">Berth Number</label>  
 <input type="number" id="berth\_num" name="berth\_num" required>  
 </div>  
 <div class="form-group">  
 <label for="target">Destination</label>  
 <input type="text" id="target" name="target" required>  
 </div>  
 </div>  
 <button type="submit" class="btn btn-primary">Add Ship</button>  
 </form>  
 </div>  
  
 <div class="card">  
 <div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 20px;">  
 <h2>Fleet Overview</h2>  
 <button class="btn btn-export" onclick="shipManager.exportAllShips()">Export All to Word</button>  
 </div>  
 <div id="shipsList" class="ships-grid">  
 <div class="empty-state">Loading ships...</div>  
 </div>  
 </div>  
 </div>  
  
 <div id="editModal" class="modal">  
 <div class="modal-content">  
 <div class="modal-header">  
 <h2>Edit Ship</h2>  
 <span class="close">&times;</span>  
 </div>  
 <form id="editShipForm">  
 <input type="hidden" id="editId">  
 <div class="form-grid">  
 <div class="form-group">  
 <label for="editName">Ship Name</label>  
 <input type="text" id="editName" name="name" required>  
 </div>  
 <div class="form-group">  
 <label for="editDisplacement">Displacement (tons)</label>  
 <input type="number" id="editDisplacement" name="displacement" step="0.1" required>  
 </div>  
 <div class="form-group">  
 <label for="editPort">Home Port</label>  
 <input type="text" id="editPort" name="port" required>  
 </div>  
 <div class="form-group">  
 <label for="editCaptain">Captain</label>  
 <input type="text" id="editCaptain" name="captain" required>  
 </div>  
 <div class="form-group">  
 <label for="editBerthNum">Berth Number</label>  
 <input type="number" id="editBerthNum" name="berth\_num" required>  
 </div>  
 <div class="form-group">  
 <label for="editTarget">Destination</label>  
 <input type="text" id="editTarget" name="target" required>  
 </div>  
 </div>  
 <div class="form-actions">  
 <button type="submit" class="btn btn-primary">Save Changes</button>  
 <button type="button" class="btn" onclick="shipManager.closeModal()" style="background: #ccc;">Cancel</button>  
 </div>  
 </form>  
 </div>  
 </div>  
  
 <script>  
 class ShipManager {  
 constructor() {  
 this.apiBase = '/ships';  
 this.init();  
 }  
  
 init() {  
 this.bindEvents();  
 this.loadShips();  
 }  
  
 bindEvents() {  
 document.getElementById('addShipForm').addEventListener('submit', (e) => {  
 e.preventDefault();  
 this.addShip();  
 });  
  
 document.getElementById('editShipForm').addEventListener('submit', (e) => {  
 e.preventDefault();  
 this.updateShip();  
 });  
  
 document.querySelector('.close').addEventListener('click', () => this.closeModal());  
   
 document.getElementById('editModal').addEventListener('click', (e) => {  
 if (e.target.id === 'editModal') this.closeModal();  
 });  
 }  
  
 async loadShips() {  
 try {  
 const response = await fetch(this.apiBase);  
 const ships = await response.json();  
 this.displayShips(ships);  
 } catch (error) {  
 console.error('Error loading ships:', error);  
 alert('Failed to load ships');  
 }  
 }  
  
 displayShips(ships) {  
 const container = document.getElementById('shipsList');  
   
 if (ships.length === 0) {  
 container.innerHTML = '<div class="empty-state">No ships in fleet. Add your first ship above.</div>';  
 return;  
 }  
  
 container.innerHTML = ships.map(ship => `  
 <div class="ship-card">  
 <h3>${ship.name}</h3>  
 <div class="ship-info">  
 <div class="ship-field">  
 <span class="field-label">ID:</span>  
 <span>#${ship.id}</span>  
 </div>  
 <div class="ship-field">  
 <span class="field-label">Displacement:</span>  
 <span>${ship.displacement} tons</span>  
 </div>  
 <div class="ship-field">  
 <span class="field-label">Home Port:</span>  
 <span>${ship.port}</span>  
 </div>  
 <div class="ship-field">  
 <span class="field-label">Captain:</span>  
 <span>${ship.captain}</span>  
 </div>  
 <div class="ship-field">  
 <span class="field-label">Berth Number:</span>  
 <span>${ship.berth\_num}</span>  
 </div>  
 <div class="ship-field">  
 <span class="field-label">Destination:</span>  
 <span>${ship.target}</span>  
 </div>  
 </div>  
 <div class="ship-actions">  
 <button class="btn btn-edit" onclick="shipManager.editShip(${ship.id})">Edit</button>  
 <button class="btn btn-export" onclick="shipManager.exportShip(${ship.id})">Export</button>  
 <button class="btn btn-delete" onclick="shipManager.deleteShip(${ship.id})">Delete</button>  
 </div>  
 </div>  
 `).join('');  
 }  
  
 async addShip() {  
 const form = document.getElementById('addShipForm');  
 const formData = new FormData(form);  
   
 const shipData = {  
 name: formData.get('name'),  
 displacement: parseFloat(formData.get('displacement')),  
 port: formData.get('port'),  
 captain: formData.get('captain'),  
 berth\_num: parseInt(formData.get('berth\_num')),  
 target: formData.get('target')  
 };  
  
 try {  
 const response = await fetch(this.apiBase, {  
 method: 'POST',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(shipData)  
 });  
  
 if (response.ok) {  
 form.reset();  
 this.loadShips();  
 alert('Ship added successfully!');  
 } else {  
 throw new Error('Failed to add ship');  
 }  
 } catch (error) {  
 console.error('Error adding ship:', error);  
 alert('Failed to add ship');  
 }  
 }  
  
 async editShip(shipId) {  
 try {  
 const response = await fetch(`${this.apiBase}/${shipId}`);  
 const ship = await response.json();  
  
 document.getElementById('editId').value = ship.id;  
 document.getElementById('editName').value = ship.name;  
 document.getElementById('editDisplacement').value = ship.displacement;  
 document.getElementById('editPort').value = ship.port;  
 document.getElementById('editCaptain').value = ship.captain;  
 document.getElementById('editBerthNum').value = ship.berth\_num;  
 document.getElementById('editTarget').value = ship.target;  
  
 this.openModal();  
 } catch (error) {  
 console.error('Error loading ship for edit:', error);  
 alert('Failed to load ship data');  
 }  
 }  
  
 async updateShip() {  
 const shipId = document.getElementById('editId').value;  
 const formData = new FormData(document.getElementById('editShipForm'));  
   
 const shipData = {  
 name: formData.get('name'),  
 displacement: parseFloat(formData.get('displacement')),  
 port: formData.get('port'),  
 captain: formData.get('captain'),  
 berth\_num: parseInt(formData.get('berth\_num')),  
 target: formData.get('target')  
 };  
  
 try {  
 const response = await fetch(`${this.apiBase}/${shipId}`, {  
 method: 'PUT',  
 headers: {'Content-Type': 'application/json'},  
 body: JSON.stringify(shipData)  
 });  
  
 if (response.ok) {  
 this.closeModal();  
 this.loadShips();  
 alert('Ship updated successfully!');  
 } else {  
 throw new Error('Failed to update ship');  
 }  
 } catch (error) {  
 console.error('Error updating ship:', error);  
 alert('Failed to update ship');  
 }  
 }  
  
 async deleteShip(shipId) {  
 if (!confirm('Are you sure you want to delete this ship?')) return;  
  
 try {  
 const response = await fetch(`${this.apiBase}/${shipId}`, {  
 method: 'DELETE'  
 });  
  
 if (response.ok) {  
 this.loadShips();  
 alert('Ship deleted successfully!');  
 } else {  
 throw new Error('Failed to delete ship');  
 }  
 } catch (error) {  
 console.error('Error deleting ship:', error);  
 alert('Failed to delete ship');  
 }  
 }  
  
 async exportShip(shipId) {  
 try {  
 const response = await fetch(`/ships/download/${shipId}`);  
 if (response.ok) {  
 const blob = await response.blob();  
 const url = window.URL.createObjectURL(blob);  
 const a = document.createElement('a');  
 a.href = url;  
 a.download = `ship\_${shipId}\_report.docx`;  
 document.body.appendChild(a);  
 a.click();  
 window.URL.revokeObjectURL(url);  
 document.body.removeChild(a);  
 } else {  
 throw new Error('Failed to export ship');  
 }  
 } catch (error) {  
 console.error('Error exporting ship:', error);  
 alert('Failed to export ship');  
 }  
 }  
  
 async exportAllShips() {  
 try {  
 const response = await fetch('/ships/download');  
 if (response.ok) {  
 const blob = await response.blob();  
 const url = window.URL.createObjectURL(blob);  
 const a = document.createElement('a');  
 a.href = url;  
 a.download = 'ship\_fleet\_report.docx';  
 document.body.appendChild(a);  
 a.click();  
 window.URL.revokeObjectURL(url);  
 document.body.removeChild(a);  
 } else {  
 throw new Error('Failed to export fleet');  
 }  
 } catch (error) {  
 console.error('Error exporting fleet:', error);  
 alert('Failed to export fleet');  
 }  
 }  
  
 openModal() {  
 document.getElementById('editModal').style.display = 'block';  
 }  
  
 closeModal() {  
 document.getElementById('editModal').style.display = 'none';  
 }  
 }  
  
 document.addEventListener('DOMContentLoaded', () => {  
 window.shipManager = new ShipManager();  
 });  
 </script>  
</body>  
</html>

**Интерфейс:**

****

**Окно редактирования:**

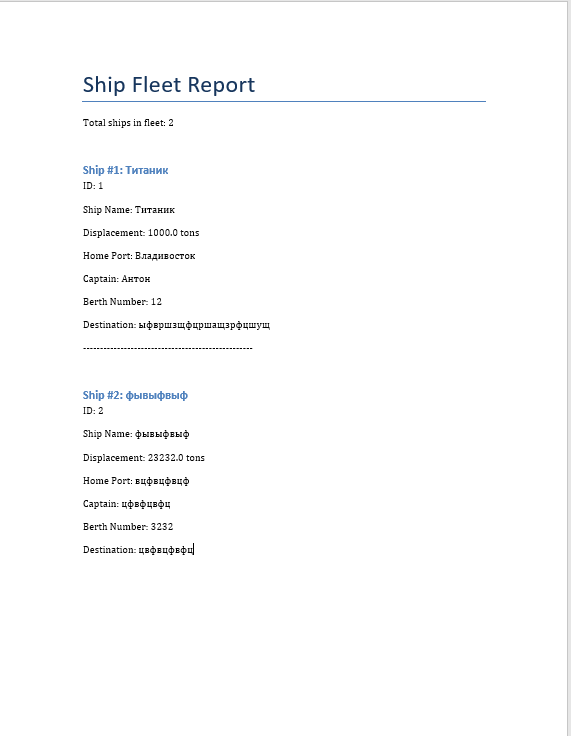
****

**2. Добавление новой функции экспорта данных в word**

**Экспорт всех кораблей:**

|  |
| --- |
| @app.get("/ships/download") async def export\_ships\_to\_word():  try:  if not ship\_list:  raise HTTPException(status\_code=404, detail="No ships available for export")    doc = Document()  doc.add\_heading('Ship Fleet Report', 0)  doc.add\_paragraph(f"Total ships in fleet: {len(ship\_list)}")  doc.add\_paragraph()   for i, ship in enumerate(ship\_list, 1):  if i > 1:  doc.add\_paragraph("-" \* 50)  doc.add\_paragraph()    doc.add\_heading(f'Ship #{ship.id}: {ship.name}', level=2)  doc.add\_paragraph(f"ID: {ship.id}")  doc.add\_paragraph(f"Ship Name: {ship.name}")  doc.add\_paragraph(f"Displacement: {ship.displacement} tons")  doc.add\_paragraph(f"Home Port: {ship.port}")  doc.add\_paragraph(f"Captain: {ship.captain}")  doc.add\_paragraph(f"Berth Number: {ship.berth\_num}")  doc.add\_paragraph(f"Destination: {ship.target}")   buffer = BytesIO()  doc.save(buffer)  buffer.seek(0)   return Response(  content=buffer.getvalue(),  media\_type='application/vnd.openxmlformats-officedocument.wordprocessingml.document',  headers={'Content-Disposition': 'attachment; filename="ship\_fleet\_report.docx"'}  )  except Exception as e:  print(f"Error in export\_ships\_to\_word: {e}")  raise HTTPException(status\_code=500, detail=f"Error generating report: {str(e)}") |

Пример документа:

****

**Экспорт корабля по id:**

|  |
| --- |
| @app.get("/ships/download/{ship\_id}") async def export\_ship\_to\_word(ship\_id: int):  try:  for ship in ship\_list:  if ship\_id == ship.id:  doc = Document()  doc.add\_heading(f'Ship Report: {ship.name}', 0)    *# Данные корабля* ship\_data = [  ("ID", str(ship.id)),  ("Ship Name", ship.name),  ("Displacement", f"{ship.displacement} tons"),  ("Home Port", ship.port),  ("Captain", ship.captain),  ("Berth Number", str(ship.berth\_num)),  ("Destination", ship.target)  ]    for field, value in ship\_data:  p = doc.add\_paragraph()  p.add\_run(f"{field}: ").bold = True  p.add\_run(value)   buffer = BytesIO()  doc.save(buffer)  buffer.seek(0)   filename = f"ship\_{ship.id}\_report.docx"    return Response(  content=buffer.getvalue(),  media\_type='application/vnd.openxmlformats-officedocument.wordprocessingml.document',  headers={'Content-Disposition': f'attachment; filename="{filename}"'}  )    raise HTTPException(status\_code=404, detail="Ship not found")  except Exception as e:  print(f"Error in export\_ship\_to\_word: {e}")  raise HTTPException(status\_code=500, detail=f"Error generating ship report: {str(e)}") |

Пример документа:

