Отчет по лабораторной работе

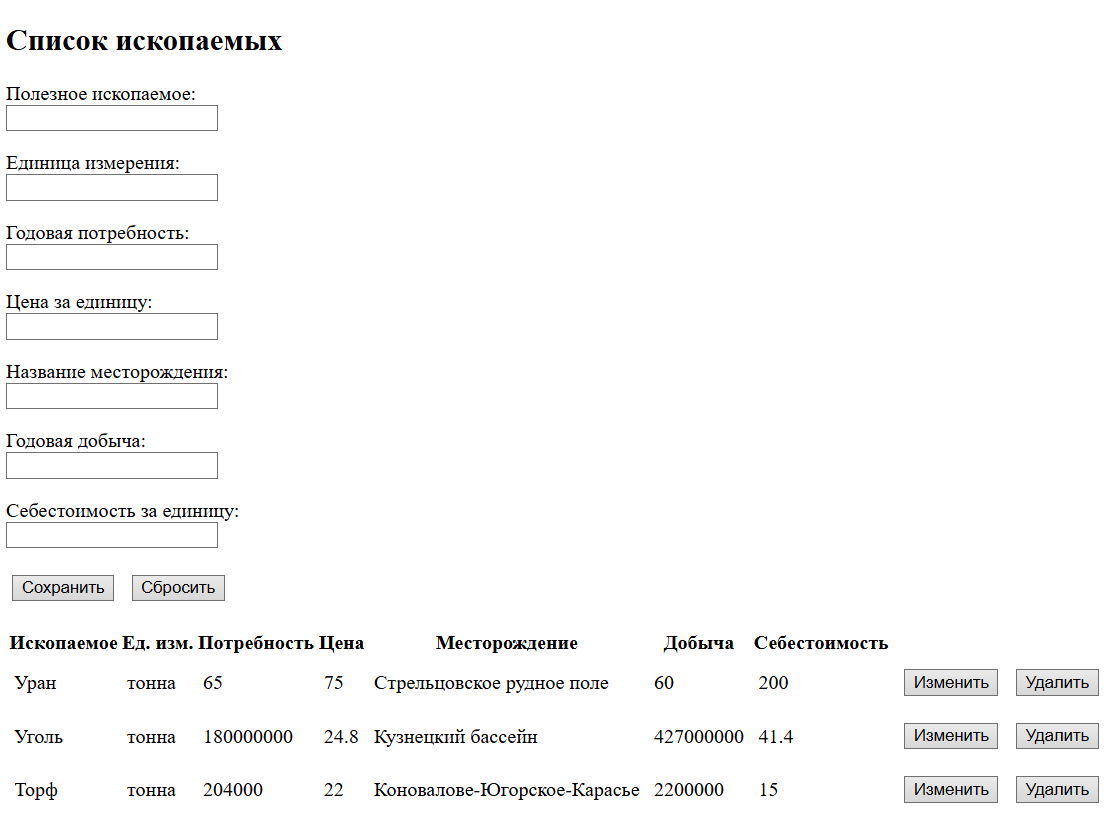
ИСП23.1А.

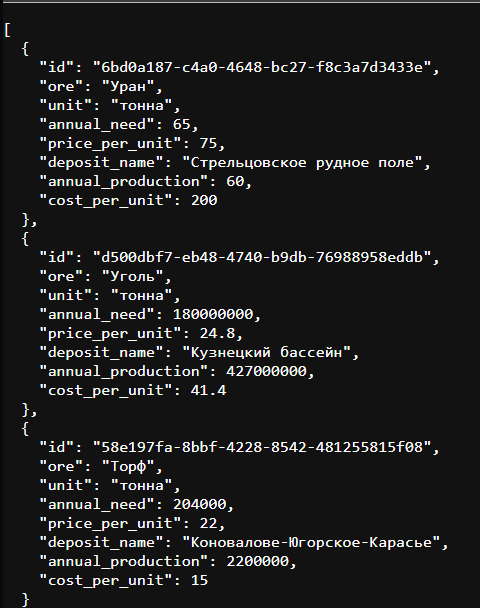
Сологуб Денис

**Предметная область:** *Добыча полезных ископаемых.* Минимальный список характеристик:

* полезное ископаемое,
* единица измерения,
* годовая потребность,
* цена за единицу,
* название месторождения,
* годовая добыча,
* себестоимость за единицу.

**Программа:**

****

****

**Полный код API.py**

import uuid  
from fastapi import FastAPI, Body, status  
from fastapi.responses import JSONResponse, FileResponse  
  
class Ore:  
 def \_\_init\_\_(self, ore, unit, annual\_need, price\_per\_unit, deposit\_name, annual\_production, cost\_per\_unit):  
 self.id = str(uuid.uuid4())  
 self.ore = ore  
 self.unit = unit  
 self.annual\_need = annual\_need  
 self.price\_per\_unit = price\_per\_unit  
 self.deposit\_name = deposit\_name  
 self.annual\_production = annual\_production  
 self.cost\_per\_unit = cost\_per\_unit  
  
ores = [  
 Ore("Уран", "тонна", 65, 75.0, "Стрельцовское рудное поле", 60, 200.0),  
 Ore("Уголь", "тонна", 180000000, 24.8, "Кузнецкий бассейн", 427000000, 41.4),  
 Ore("Торф", "тонна", 204000, 22.0, "Коновалове-Югорское-Карасье", 2200000, 15.0)  
]  
  
def find\_ore(id):  
 for ore in ores:  
 if ore.id == id:  
 return ore  
 return None  
  
app = FastAPI()  
  
@app.get("/")  
async def main():  
 return FileResponse("index.html")  
  
@app.get("/api/ores")  
def get\_ores():  
 return ores  
  
@app.get("/api/ores/{id}")  
def get\_ore(id):  
 ore = find\_ore(id)  
 if ore is None:  
 return JSONResponse(status\_code=status.HTTP\_404\_NOT\_FOUND, content={"message": "Ископаемое не найдено"})  
 return ore  
  
@app.post("/api/ores")  
def create\_ore(data=Body()):  
 ore = Ore(data["ore"], data["unit"], data["annual\_need"], data["price\_per\_unit"],  
 data["deposit\_name"], data["annual\_production"], data["cost\_per\_unit"])  
 ores.append(ore)  
 return ore  
  
@app.put("/api/ores")  
def edit\_ore(data=Body()):  
 ore = find\_ore(data["id"])  
 if ore is None:  
 return JSONResponse(status\_code=status.HTTP\_404\_NOT\_FOUND, content={"message": "Ископаемое не найдено"})  
 ore.ore = data["ore"]  
 ore.unit = data["unit"]  
 ore.annual\_need = data["annual\_need"]  
 ore.price\_per\_unit = data["price\_per\_unit"]  
 ore.deposit\_name = data["deposit\_name"]  
 ore.annual\_production = data["annual\_production"]  
 ore.cost\_per\_unit = data["cost\_per\_unit"]  
 return ore  
  
@app.delete("/api/ores/{id}")  
def delete\_ore(id):  
 ore = find\_ore(id)  
 if ore is None:  
 return JSONResponse(status\_code=status.HTTP\_404\_NOT\_FOUND, content={"message": "Ископаемое не найдено"})  
 ores.remove(ore)  
 return ore  
  
**Описание**

import uuid  
from fastapi import FastAPI, Body, status  
from fastapi.responses import JSONResponse, FileResponse

uuid — импортируем модуль для генерации уникальных идентификаторов.

FastAPI — нужен для создания приложения.

Body — для извлечения данных из тела запроса.

Status — содержит HTTP статус коды.

JSONResponse — возвращает JSON с кастомными настройками.

FileResponse — возвращает файл.

class Ore:  
 def \_\_init\_\_(self, ore, unit, annual\_need, price\_per\_unit, deposit\_name, annual\_production, cost\_per\_unit):  
 self.id = str(uuid.uuid4())  
 self.ore = ore  
 self.unit = unit  
 self.annual\_need = annual\_need  
 self.price\_per\_unit = price\_per\_unit  
 self.deposit\_name = deposit\_name  
 self.annual\_production = annual\_production  
 self.cost\_per\_unit = cost\_per\_unit

Класс Ore создает объект полезного ископаемого с уникальным ID. Содержит свойства: название, единица измерения, годовая потребность, цена за единицу, название месторождения, годовая добыча и себестоимость.

ores = [  
 Ore("Уран", "тонна", 65, 75.0, "Стрельцовское рудное поле", 60, 200.0),  
 Ore("Уголь", "тонна", 180000000, 24.8, "Кузнецкий бассейн", 427000000, 41.4),  
 Ore("Торф", "тонна", 204000, 22.0, "Коновалове-Югорское-Карасье", 2200000, 15.0)  
]

Список ores — предзаполненный массив с тремя примерами ископаемых. Каждый пример с конкретными параметрами.

def find\_ore(id):  
 for ore in ores:  
 if ore.id == id:  
 return ore  
 return None

Функция find\_ore — ищет ископаемые по ID в массиве ores. Возвращает объект или none если ничего не найдено.

app = FastAPI()

Инициализация основного объекта приложения.

@app.get("/")  
async def main():  
 return FileResponse("index.html")

Эндпоинт GET "/" — Возвращает HTML файл index.html как главную страницу.

@app.get("/api/ores")  
def get\_ores():  
 return ores

Эндпоинт GET "/api/ores" — Возвращает полный список всех ископаемых.

@app.get("/api/ores/{id}")  
def get\_ore(id):  
 ore = find\_ore(id)  
 if ore is None:  
 return JSONResponse(status\_code=status.HTTP\_404\_NOT\_FOUND, content={"message": "Ископаемое не найдено"})  
 return ore

Эндпоинт GET "/api/ores/{id}" — Ищет конкретное ископаемое по ID из URL. При отсутствии возвращает 404 ошибку.

@app.post("/api/ores")  
def create\_ore(data=Body()):  
 ore = Ore(data["ore"], data["unit"], data["annual\_need"], data["price\_per\_unit"],  
 data["deposit\_name"], data["annual\_production"], data["cost\_per\_unit"])  
 ores.append(ore)  
 return ore

Эндпоинт POST "/api/ores" — Создает новое ископаемое из данных тела запроса. Добавляет в массив и возвращает созданный объект.

@app.put("/api/ores")  
def edit\_ore(data=Body()):  
 ore = find\_ore(data["id"])  
 if ore is None:  
 return JSONResponse(status\_code=status.HTTP\_404\_NOT\_FOUND, content={"message": "Ископаемое не найдено"})  
 ore.ore = data["ore"]  
 ore.unit = data["unit"]  
 ore.annual\_need = data["annual\_need"]  
 ore.price\_per\_unit = data["price\_per\_unit"]  
 ore.deposit\_name = data["deposit\_name"]  
 ore.annual\_production = data["annual\_production"]  
 ore.cost\_per\_unit = data["cost\_per\_unit"]  
 return ore

Эндпоинт PUT "/api/ores" — Обновляет существующее ископаемое по ID из тела запроса. Изменяет все свойства объекта, а также при отсутствии возвращает 404 ошибку.

@app.delete("/api/ores/{id}")  
def delete\_ore(id):  
 ore = find\_ore(id)  
 if ore is None:  
 return JSONResponse(status\_code=status.HTTP\_404\_NOT\_FOUND, content={"message": "Ископаемое не найдено"})  
 ores.remove(ore)  
 return ore

Эндпоинт DELETE "/api/ores/{id}" — Удаляет ископаемое по ID из URL, убирает из массива и возвращает удаленный объект, При отсутствии возвращает 404 ошибку.

**Полный код index.html**

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8" />  
 <title>Добыча полезных ископаемых</title>  
 <style>  
 td {padding:5px;}  
 button {margin: 5px;}  
 </style>  
</head>  
<body>  
 <h2>Список ископаемых</h2>  
 <div>  
 <input type="hidden" id="oreId" />  
 <p>  
 Полезное ископаемое:<br/>  
 <input id="oreName" />  
 </p>  
 <p>  
 Единица измерения:<br />  
 <input id="oreUnit" />  
 </p>  
 <p>  
 Годовая потребность:<br />  
 <input id="annualNeed" type="number" />  
 </p>  
 <p>  
 Цена за единицу:<br />  
 <input id="pricePerUnit" type="number" />  
 </p>  
 <p>  
 Название месторождения:<br />  
 <input id="depositName" />  
 </p>  
 <p>  
 Годовая добыча:<br />  
 <input id="annualProduction" type="number" />  
 </p>  
 <p>  
 Себестоимость за единицу:<br />  
 <input id="costPerUnit" type="number" />  
 </p>  
 <p>  
 <button id="saveBtn">Сохранить</button>  
 <button id="resetBtn">Сбросить</button>  
 </p>  
 </div>  
 <table>  
 <thead>  
 <tr>  
 <th>Ископаемое</th>  
 <th>Ед. изм.</th>  
 <th>Потребность</th>  
 <th>Цена</th>  
 <th>Месторождение</th>  
 <th>Добыча</th>  
 <th>Себестоимость</th>  
 <th></th>  
 </tr>  
 </thead>  
 <tbody></tbody>  
 </table>  
  
 <script>  
 // Получение всех ископаемых  
 async function getOres() {  
 const response = await fetch("/api/ores", { method: "GET", headers: {"Accept":"application/json"} });  
 if (response.ok) {  
 const ores = await response.json();  
 const rows = document.querySelector("tbody");  
 rows.innerHTML = "";  
 ores.forEach(ore => rows.append(row(ore)));  
 }  
 }  
  
 // Получение одного  
 async function getOre(id) {  
 const response = await fetch(`/api/ores/${id}`, {method:"GET", headers:{"Accept":"application/json"}});  
 if (response.ok) {  
 const ore = await response.json();  
 document.getElementById("oreId").value = ore.id;  
 document.getElementById("oreName").value = ore.ore;  
 document.getElementById("oreUnit").value = ore.unit;  
 document.getElementById("annualNeed").value = ore.annual\_need;  
 document.getElementById("pricePerUnit").value = ore.price\_per\_unit;  
 document.getElementById("depositName").value = ore.deposit\_name;  
 document.getElementById("annualProduction").value = ore.annual\_production;  
 document.getElementById("costPerUnit").value = ore.cost\_per\_unit;  
 }  
 }  
  
 // Добавление  
 async function createOre(data) {  
 const response = await fetch("/api/ores", {  
 method: "POST",  
 headers: {"Accept":"application/json","Content-Type":"application/json"},  
 body: JSON.stringify(data)  
 });  
 if (response.ok) {  
 const ore = await response.json();  
 document.querySelector("tbody").append(row(ore));  
 }  
 }  
  
 // Изменение  
 async function editOre(data) {  
 const response = await fetch("/api/ores", {  
 method: "PUT",  
 headers: {"Accept":"application/json","Content-Type":"application/json"},  
 body: JSON.stringify(data)  
 });  
 if (response.ok) {  
 const ore = await response.json();  
 document.querySelector(`tr[data-rowid='${ore.id}']`).replaceWith(row(ore));  
 }  
 }  
  
 // Удаление  
 async function deleteOre(id) {  
 const response = await fetch(`/api/ores/${id}`, {method:"DELETE", headers:{"Accept":"application/json"}});  
 if (response.ok) {  
 document.querySelector(`tr[data-rowid='${id}']`).remove();  
 }  
 }  
  
 // Строка таблицы  
 function row(ore) {  
 const tr = document.createElement("tr");  
 tr.setAttribute("data-rowid", ore.id);  
 tr.innerHTML = `  
 <td>${ore.ore}</td>  
 <td>${ore.unit}</td>  
 <td>${ore.annual\_need}</td>  
 <td>${ore.price\_per\_unit}</td>  
 <td>${ore.deposit\_name}</td>  
 <td>${ore.annual\_production}</td>  
 <td>${ore.cost\_per\_unit}</td>  
 <td>  
 <button onclick=\"getOre('${ore.id}')\">Изменить</button>  
 <button onclick=\"deleteOre('${ore.id}')\">Удалить</button>  
 </td>  
 `;  
 return tr;  
 }  
  
 // Сброс формы  
 function resetForm() {  
 document.getElementById("oreId").value =  
 document.getElementById("oreName").value =  
 document.getElementById("oreUnit").value =  
 document.getElementById("annualNeed").value =  
 document.getElementById("pricePerUnit").value =  
 document.getElementById("depositName").value =  
 document.getElementById("annualProduction").value =  
 document.getElementById("costPerUnit").value = "";  
 }  
  
 document.getElementById("resetBtn").addEventListener("click", resetForm);  
  
 document.getElementById("saveBtn").addEventListener("click", async () => {  
 const id = document.getElementById("oreId").value;  
 const data = {  
 id: id,  
 ore: document.getElementById("oreName").value,  
 unit: document.getElementById("oreUnit").value,  
 annual\_need: parseFloat(document.getElementById("annualNeed").value),  
 price\_per\_unit: parseFloat(document.getElementById("pricePerUnit").value),  
 deposit\_name: document.getElementById("depositName").value,  
 annual\_production: parseFloat(document.getElementById("annualProduction").value),  
 cost\_per\_unit: parseFloat(document.getElementById("costPerUnit").value)  
 };  
 if (id === "") await createOre(data);  
 else await editOre(data);  
 resetForm();  
 });  
  
 getOres();  
 </script>  
</body>  
</html>

**Описание кода:**

<head>  
 <meta charset="utf-8" />  
 <title>Добыча полезных ископаемых</title>  
 <style>  
 td {padding:5px;}  
 button {margin: 5px;}  
 </style>  
</head>

Устанавливает название приложения

<input type="hidden" id="oreId" />

Скрытое поле для хранения ID. Нужно чтобы отличать создание нового от редактирования существующего.

<input id="oreName" />  
</p>  
<p>  
 Единица измерения:<br />  
 <input id="oreUnit" />  
</p>  
<p>  
 Годовая потребность:<br />  
 <input id="annualNeed" type="number" />  
</p>  
<p>  
 Цена за единицу:<br />  
 <input id="pricePerUnit" type="number" />  
</p>  
<p>  
 Название месторождения:<br />  
 <input id="depositName" />  
</p>  
<p>  
 Годовая добыча:<br />  
 <input id="annualProduction" type="number" />  
</p>  
<p>  
 Себестоимость за единицу:<br />  
 <input id="costPerUnit" type="number" />  
</p>

Поля для ввода информации о полезном ископаемом. Текстовое поле для ввода названия полезного ископаемого. Текстовое поле для ввода единицы измерения. Поля только для чисел (годовая потребность, цена, добыча, себестоимость)

<button id="saveBtn">Сохранить</button>  
<button id="resetBtn">Сбросить</button>

Кнопки управления: Кнопка сохранения данных, кнопка очистки формы.

<th>Ископаемое</th>  
<th>Ед. изм.</th>  
<th>Потребность</th>  
<th>Цена</th>  
<th>Месторождение</th>  
<th>Добыча</th>  
<th>Себестоимость</th>  
<th></th>

Определяет название колонок таблицы.

async function getOres() {  
 const response = await fetch("/api/ores", { method: "GET", headers: {"Accept":"application/json"} });  
 if (response.ok) {  
 const ores = await response.json();  
 const rows = document.querySelector("tbody");  
 rows.innerHTML = "";  
 ores.forEach(ore => rows.append(row(ore)));

Загружает весь список ископаемых с сервера, а также Заполняет таблицу данными.

async function getOre(id) {  
 const response = await fetch(`/api/ores/${id}`, {method:"GET", headers:{"Accept":"application/json"}});  
 if (response.ok) {  
 const ore = await response.json();  
 document.getElementById("oreId").value = ore.id;  
 document.getElementById("oreName").value = ore.ore;  
 document.getElementById("oreUnit").value = ore.unit;  
 document.getElementById("annualNeed").value = ore.annual\_need;  
 document.getElementById("pricePerUnit").value = ore.price\_per\_unit;  
 document.getElementById("depositName").value = ore.deposit\_name;  
 document.getElementById("annualProduction").value = ore.annual\_production;  
 document.getElementById("costPerUnit").value = ore.cost\_per\_unit;

Загружает одно ископаемое по ID и заполняет форму его данными для редактирования.

async function createOre(data) {  
 const response = await fetch("/api/ores", {  
 method: "POST",  
 headers: {"Accept":"application/json","Content-Type":"application/json"},  
 body: JSON.stringify(data)

});  
if (response.ok) {  
 const ore = await response.json();  
 document.querySelector("tbody").append(row(ore));

Отправляет данные на сервер для создания нового полезного ископаемого.

async function editOre(data) {  
 const response = await fetch("/api/ores", {  
 method: "PUT",  
 headers: {"Accept":"application/json","Content-Type":"application/json"},  
 body: JSON.stringify(data)  
 });  
 if (response.ok) {  
 const ore = await response.json();  
 document.querySelector(`tr[data-rowid='${ore.id}']`).replaceWith(row(ore));

Отправляет данные на сервер для изменения существующего полезного ископаемого.

async function deleteOre(id) {  
 const response = await fetch(`/api/ores/${id}`, {method:"DELETE", headers:{"Accept":"application/json"}});  
 if (response.ok) {  
 document.querySelector(`tr[data-rowid='${id}']`).remove();

Отправляет запрос на удаление полезного ископаемого.

function row(ore) {  
 const tr = document.createElement("tr");  
 tr.setAttribute("data-rowid", ore.id);  
 tr.innerHTML = `  
 <td>${ore.ore}</td>  
 <td>${ore.unit}</td>  
 <td>${ore.annual\_need}</td>  
 <td>${ore.price\_per\_unit}</td>  
 <td>${ore.deposit\_name}</td>  
 <td>${ore.annual\_production}</td>  
 <td>${ore.cost\_per\_unit}</td>  
 <td>  
 <button onclick=\"getOre('${ore.id}')\">Изменить</button>  
 <button onclick=\"deleteOre('${ore.id}')\">Удалить</button>  
 </td>  
 `;  
 return tr;

Создает одну строку таблицы с данными, добавляет кнопки Изменить/Удалить.

function resetForm() {  
 document.getElementById("oreId").value =  
 document.getElementById("oreName").value =  
 document.getElementById("oreUnit").value =  
 document.getElementById("annualNeed").value =  
 document.getElementById("pricePerUnit").value =  
 document.getElementById("depositName").value =  
 document.getElementById("annualProduction").value =  
 document.getElementById("costPerUnit").value = "";  
}

Очищает все поля формы.

document.getElementById("resetBtn").addEventListener("click", resetForm);

Обработчик кнопки "Сбросить", при нажатии на кнопку вызывает функцию очистки формы.

document.getElementById("saveBtn").addEventListener("click", async () => {  
 const id = document.getElementById("oreId").value;  
 const data = {  
 id: id,  
 ore: document.getElementById("oreName").value,  
 unit: document.getElementById("oreUnit").value,  
 annual\_need: parseFloat(document.getElementById("annualNeed").value),  
 price\_per\_unit: parseFloat(document.getElementById("pricePerUnit").value),  
 deposit\_name: document.getElementById("depositName").value,  
 annual\_production: parseFloat(document.getElementById("annualProduction").value),  
 cost\_per\_unit: parseFloat(document.getElementById("costPerUnit").value)  
 };  
 if (id === "") await createOre(data);  
 else await editOre(data);  
 resetForm();

Собирает данные из формы и решает: создавать новые или изменять существующий.

getOres();

При загрузке страницы сразу заполняет таблицу данными.