|  |  |
| --- | --- |
| **­­** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ \_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ДОМАШНЕМУ ЗАДАНИЮ***

***НА ТЕМУ:***

***\_\_\_\_\_\_\_\_\_\_\_Сайт списка сетей второго уровня Ethereum\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_ИУ5-45Б\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_А.П. Талаев \_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_А.И. Канев\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2025 г.*

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 1](#_Toc185524408)

[ВВЕДЕНИЕ 3](#_Toc185524409)

[1 Лабораторная работа №1 Калькулятор. HTML/CSS 5](#_Toc185524410)

[2 Лабораторная работа №2 Калькулятор. JavaScript 11](#_Toc185524411)

[3 Лабораторная работа №3 Простое веб-приложение. Верстка 15](#_Toc185524412)

[4 Домашнее задание 17](#_Toc185524413)

[5 Лабораторная работа №4 API на NestJS 20](#_Toc185524414)

[ЗАКЛЮЧЕНИЕ 20](#_Toc185524414)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc185524415)

ВВЕДЕНИЕ

Цели лабораторных работ:

1. Знакомство с инструментами построения пользовательских

интерфейсов web-сайтов: HTML, CSS. Создание калькулятора. Верстка на HTML, CSS.

1. Знакомство с инструментами построения пользовательских интерфейсов web-сайтов: HTML, CSS, JavaScript. Создание калькулятора. Функции на JavaScript.
2. Знакомство с node, npm, написание простого приложения на JavaScript.
3. Реализация собственного API на Node.js. Тестирование через Postman/Insomnia 5 методов: список с фильтрацией, получение одной записи, добавление, редактирование, удаление.

ЛАБОРАТОРНАЯ РАБОТА №1 “CALCULATOR. HTML/CSS”

**Задание**: Создание калькулятора. Верстка на HTML, CSS.

Цель данной лабораторной работы - знакомство с инструментами построения пользовательских интерфейсов web-сайтов: HTML, CSS. В ходе выполнения работы предстояло ознакомиться с кодом реализации простого калькулятора, и затем выполнить задание по варианту – скопировать для калькулятора стилистику веб-ресурса [1]. (см Рисунок 1 – Исходный сайт.).

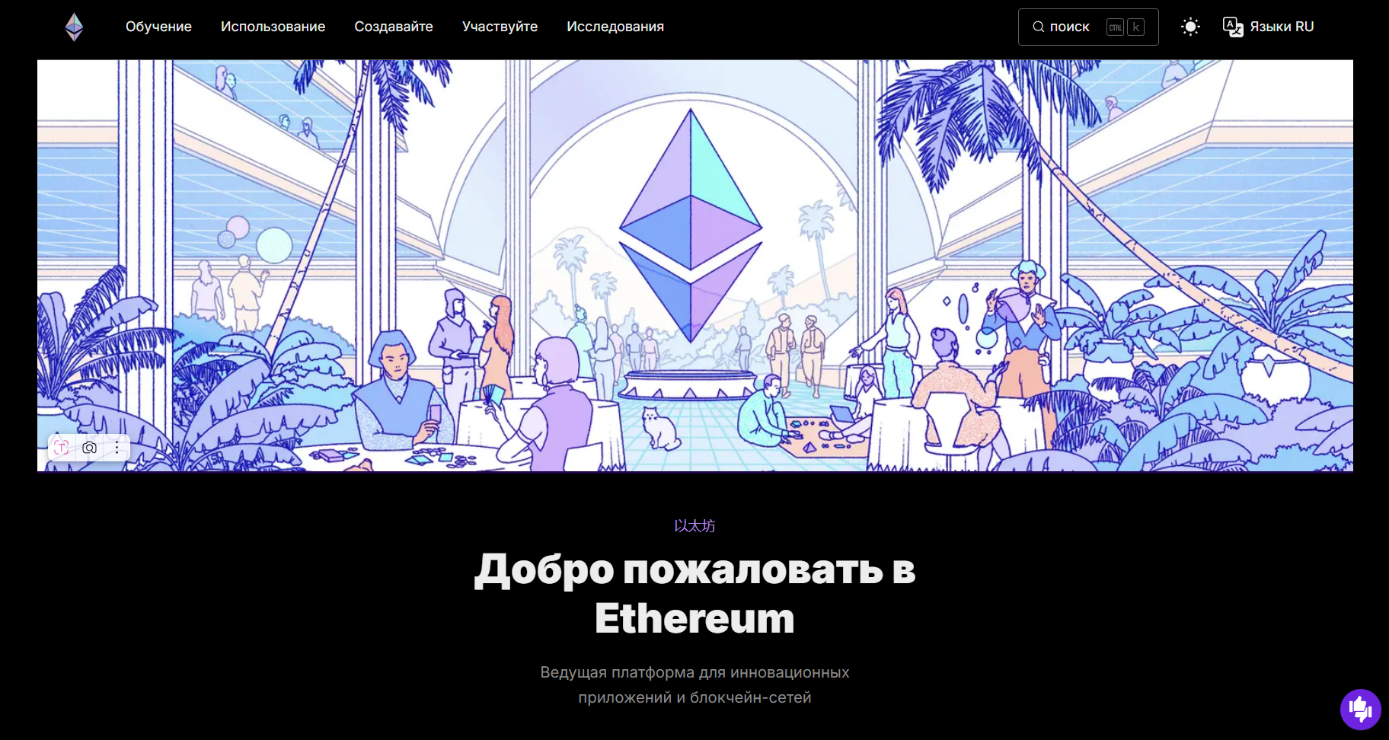


Рисунок 1 – Исходный сайт.

В результате верстки калькулятора была полностью скопирована цветовая палитра, включая цвета и тени кнопок при наведении, а также положение элементов на странице (см. рисунок 2).

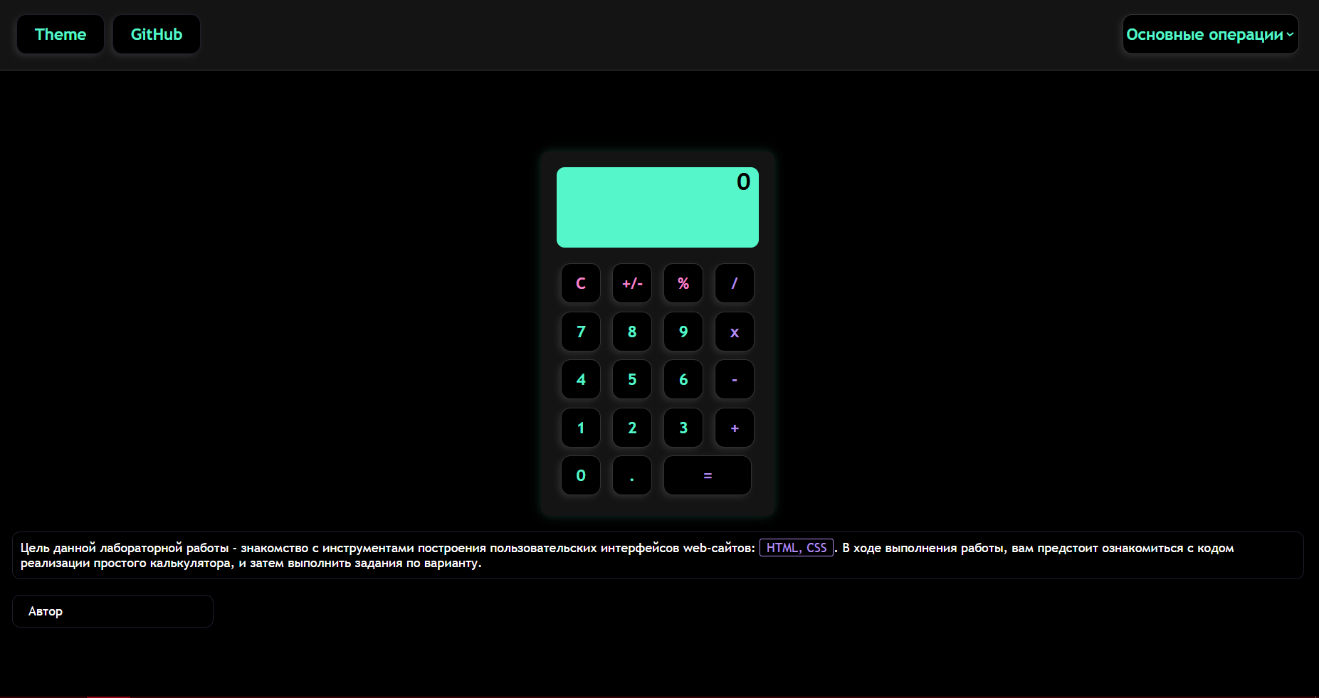


Рисунок 2 - Калькулятор со скопированной стилистикой

Рассмотрим основные скопированные элементы подробнее:

1. Стиль кнопок со стилистикой сайта (см. рисунок 3)

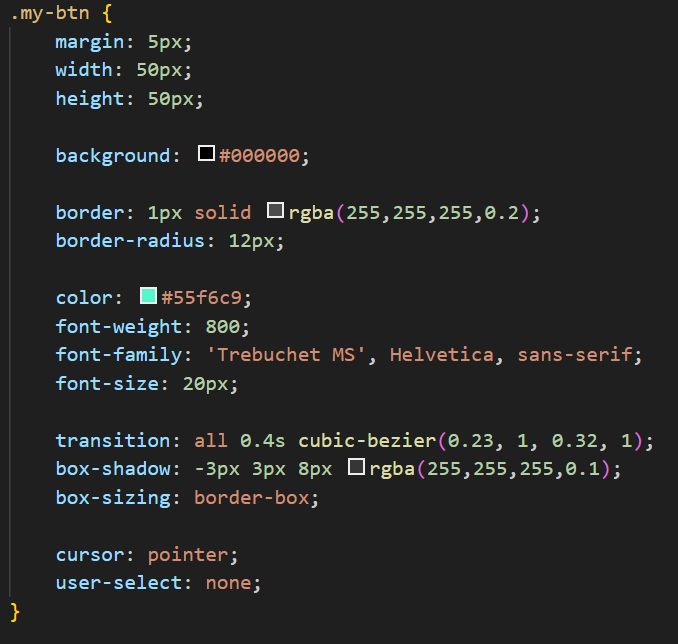


Рисунок 3 – Основной стиль кнопки

1. Скопированные эффекты теней (см. рисунок 4)

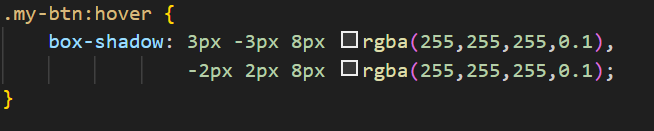


Рисунок 4 – Эффекты движения теней у кнопок при наведении

1. Стили кнопок разных типов (см. рисунок 5)

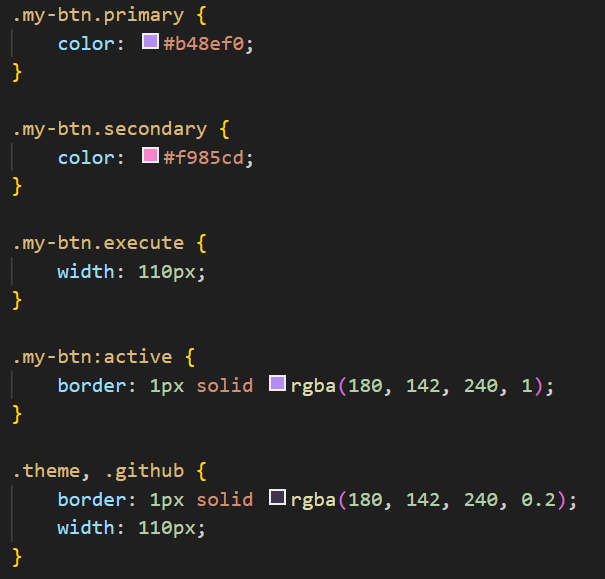


Рисунок 5 – Стили кнопок разных типов

ЛАБОРАТОРНАЯ РАБОТА №2 “CALCULATOR. JAVASCRIPT”

**Задание**: Создание калькулятора. Функции на JavaScript.

Цель данной лабораторной работы - знакомство с инструментами построения пользовательских интерфейсов web-сайтов: HTML, CSS, JavaScript. В ходе выполнения работы мы продолжили реализовывать простой калькулятор. При помощи JavaScript кнопкам калькулятора был добавлен функционал. Также были реализованы кнопки дополнительных операций и кнопка операции по индивидуальному заданию “Сложение последовательных резисторов при напряжении 220 - сила тока”.

Вычисление силы тока происходит по формуле (формула 1),

(1)

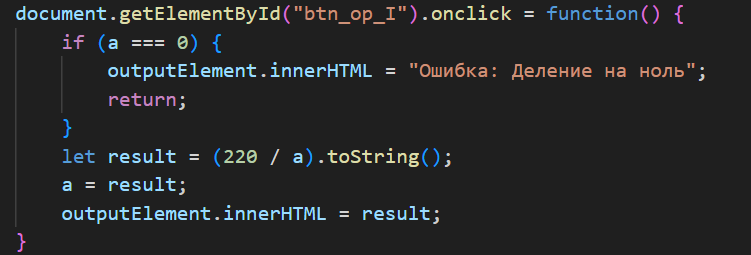


Рисунок 6 – Реализация сложения последовательных резисторов при напряжении 220 - силы тока в JavaScript

Этот код привязан к элементу btn\_op\_I на HTML странице калькулятора (см. рисунок 7).



Рисунок 7 – Реализация кнопки расчета силы тока в HTML

Итоговая страница калькулятора с дополнительным функционалом и индивидуальной операцией изображена на рисунке 8.

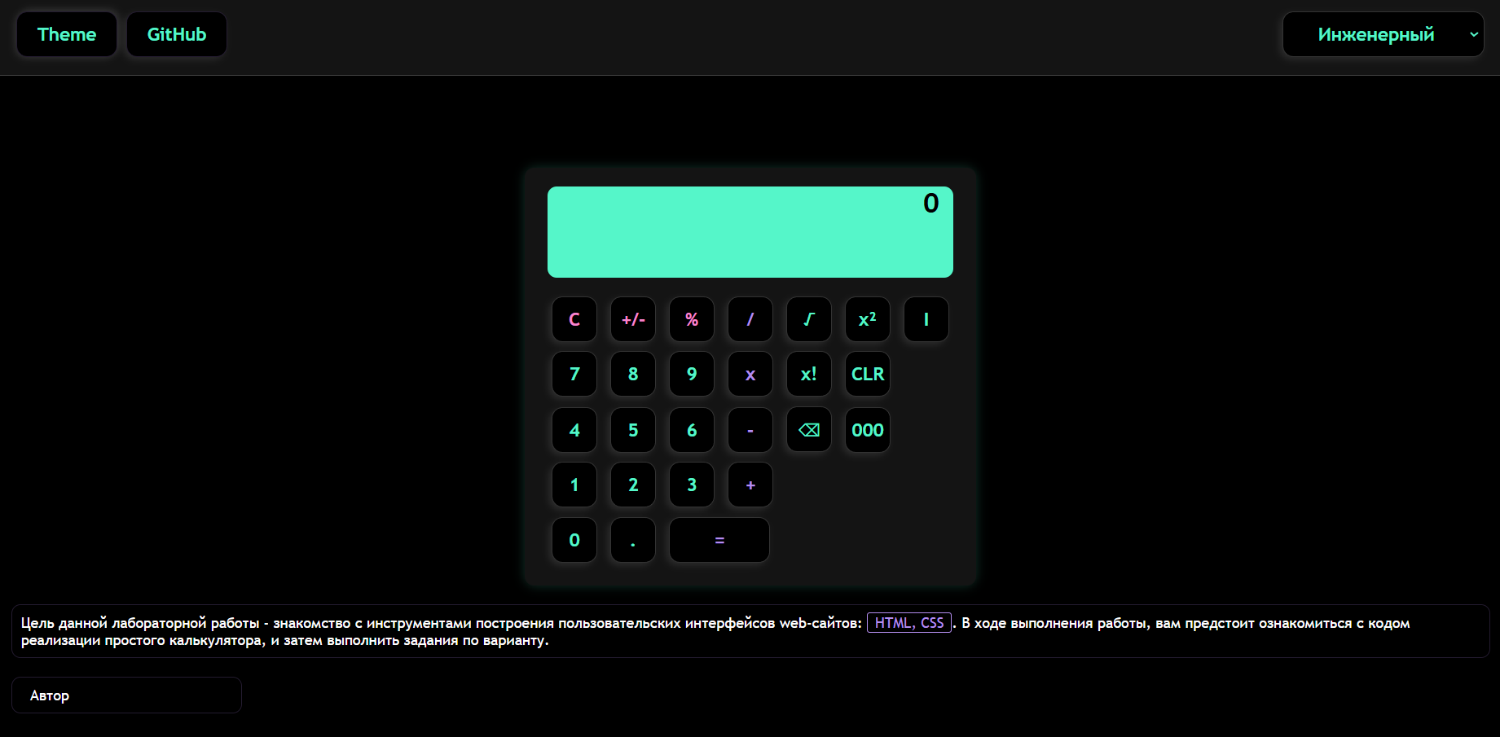


Рисунок 8 – Калькулятор с дополнительными функциями

ЛАБОРАТОРНАЯ РАБОТА №3 “ПРОСТОЕ ВЕБ-ПРИЛОЖЕНИЕ. ВЕРСТКА”

**Задание**: Знакомство с node, npm. Верстка интерфейса с карточками (страница списка с фильтрацией и страница подробнее), данные через mock объекты (коллекция). Добавить кнопку добавления (копировать первую карточку), кнопку удаления карточки.

Создадим собственный класс компонента для кнопки добавления (см. рисунок 9). Этот класс будет содержать методы для генерации HTML-кода элемента, установки обработчика событий и визуализации кнопки.



Рисунок 8 – Код класса кнопки добавления

Обработчик нажатия на кнопку удаления трека из списка будет реализован внутри класса самой карточки внутри функции render() (см. Рисунок 9).

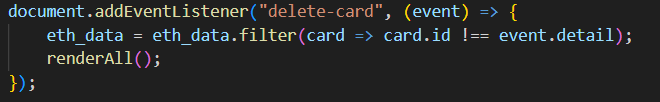


Рисунок 9 – Обработка удаления

Продемонстрируем работу кнопок добавления карточки Ethereum Mainnet и удаления (см. рисунок 10, 11)

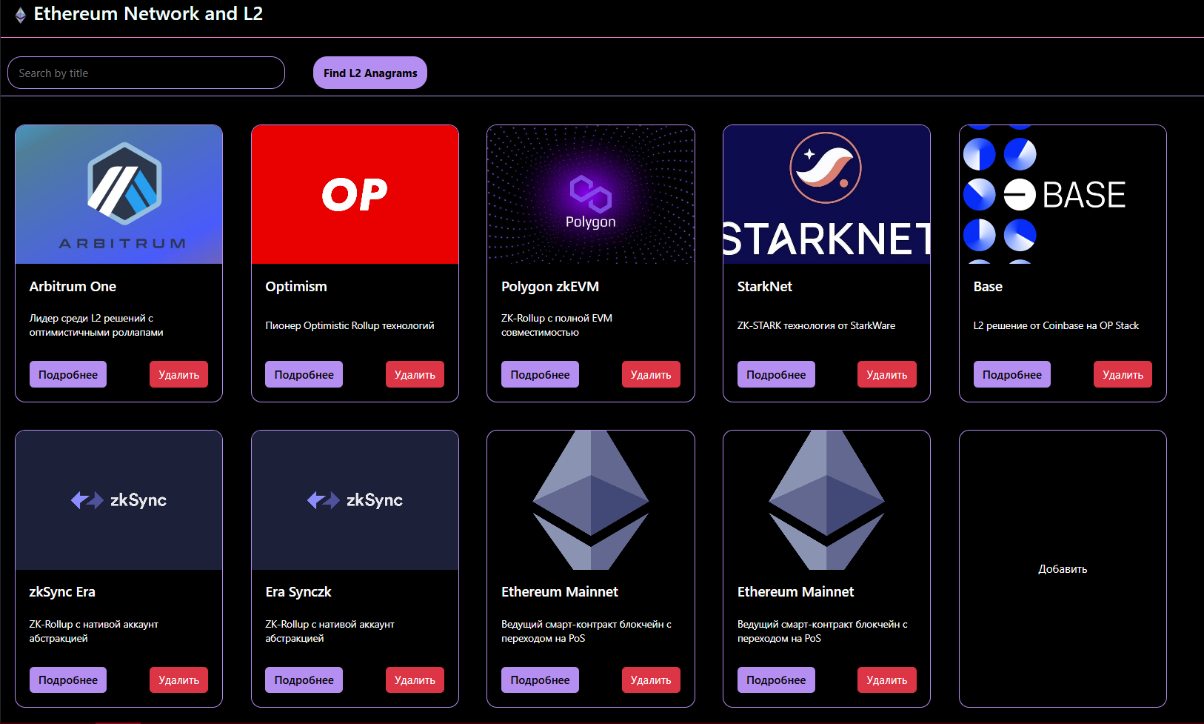


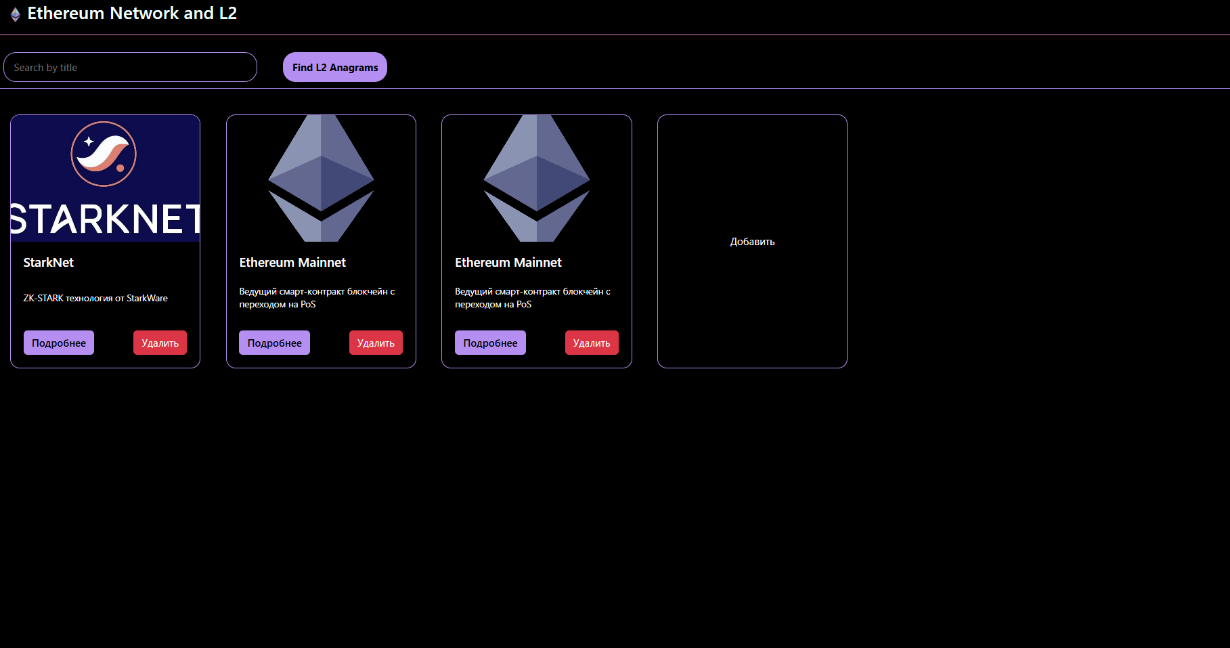
Рисунок 10 – Список после добавления 

Рисунок 11 – Список после удаления

Реализация сортировки карточек с помощью поиска подстроки внутри функции рендеринга (см. рисунок 12)

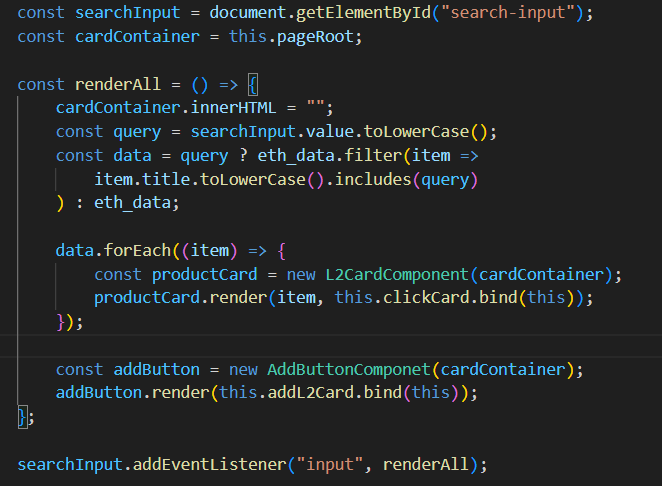


Рисунок 12 – Функция поиска подстроки

Продемонстрируем работу сортировки карточек с помощью поиска подстроки (см. рисунок 13)

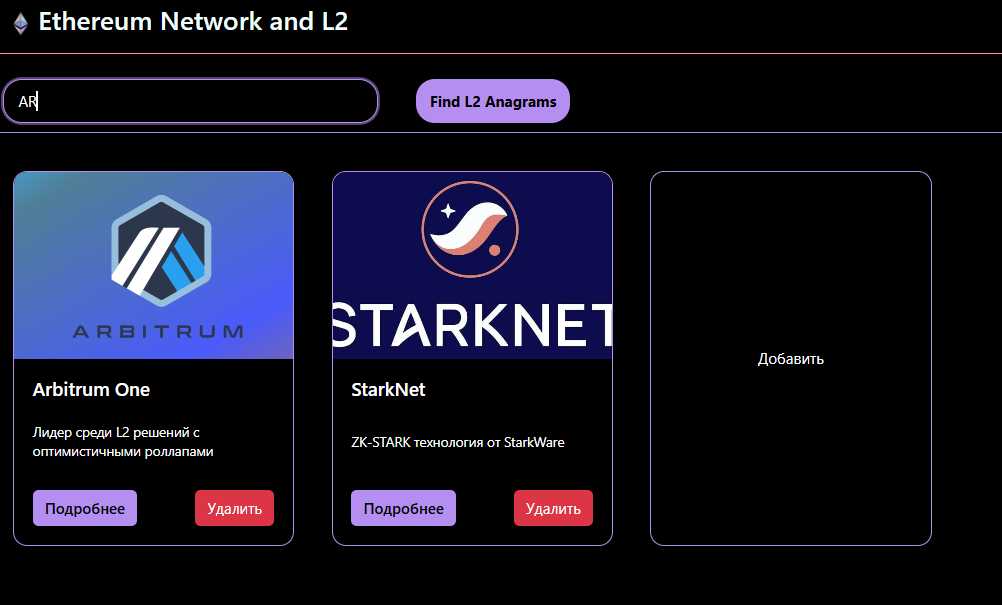


Рисунок 13 – Результат поиска

Итоговая страница основного списка карточек приведена на рисунке 13, страница сети L2– на рисунке 14.

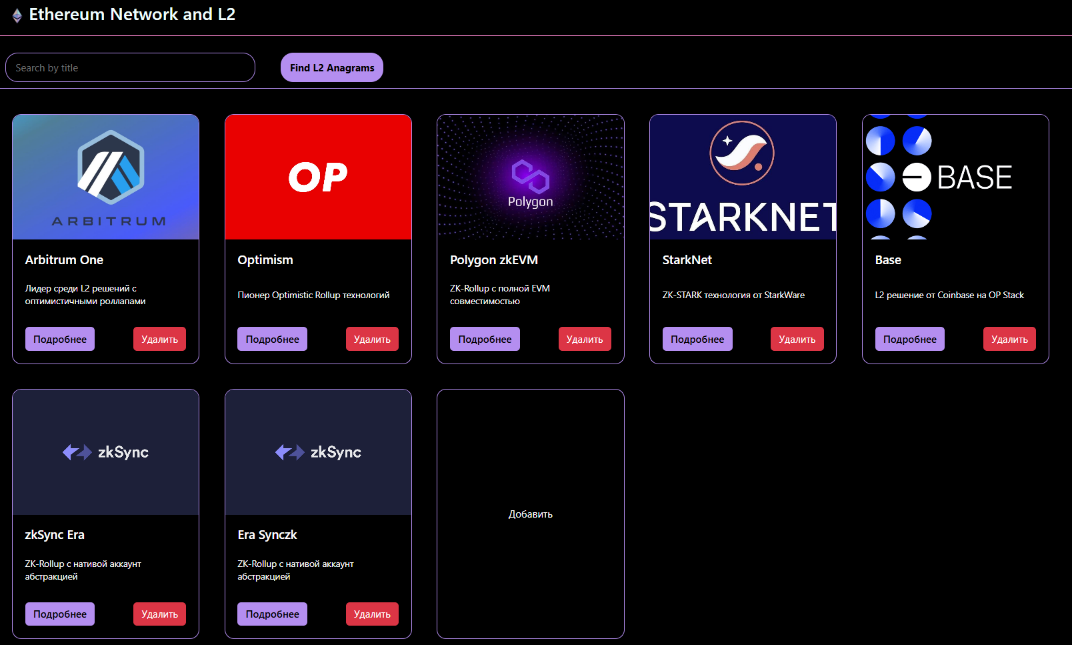


Рисунок 13 – Главная страница

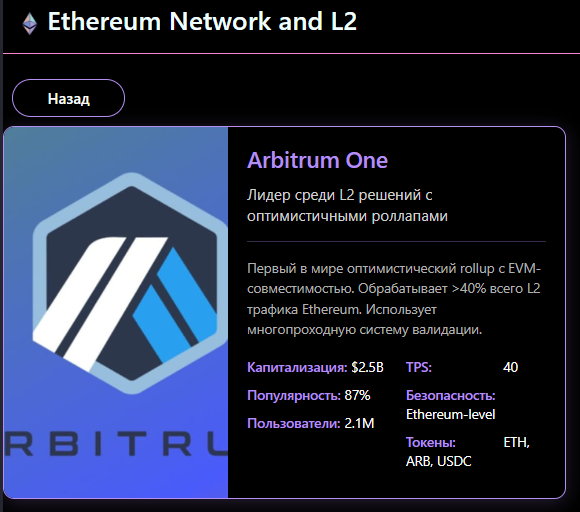


Рисунок 14 – Страница карточки

**ДОМАШНЕЕ ЗАДАНИЕ**

Задания в соответствии с вариантом:

1. Напишите функцию anagram, которая будет принимать на вход массив слов и группировать его на группы слов-анаграмм. Выводить только группы из двух и более слов. Слова в группах, как и сами группы, должны быть отсортированными.
2. Напишите функцию concatenate, которая принимает массив строк и символ-разделитель. Функция должна вернуть строчку склеенную по данному разделителю.
3. Напишите функцию erase, которая очищает массив от нежелательных значений, таких как false, undefined, пустые строки, ноль, null.
4. Даны массив строк words и строка str. Обе строки состоят только из строчных английских букв. Необходимо посчитать все слова, которые являются префиксами строки str и вывести их количество.

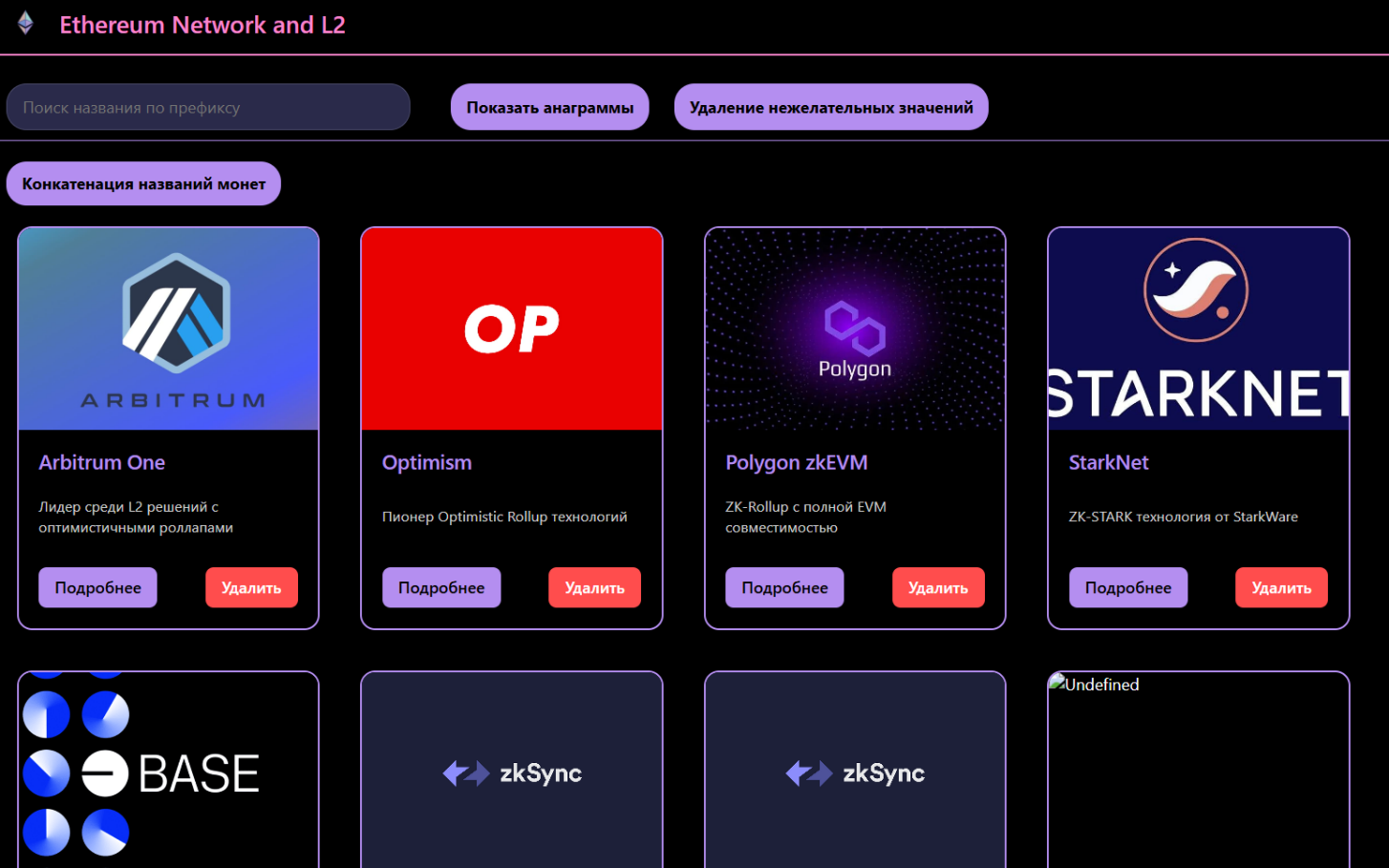


Рисунок 15 – Общий вид сайта с новыми функциями

Функция поиска анаграмм среди названий карточек была интегрирована в сайт, как сортировка, которая отображает только карточки, являющиеся анаграммами (см. рисунок 16, 17).



Рисунок 16 – Код функции поиска анаграмм

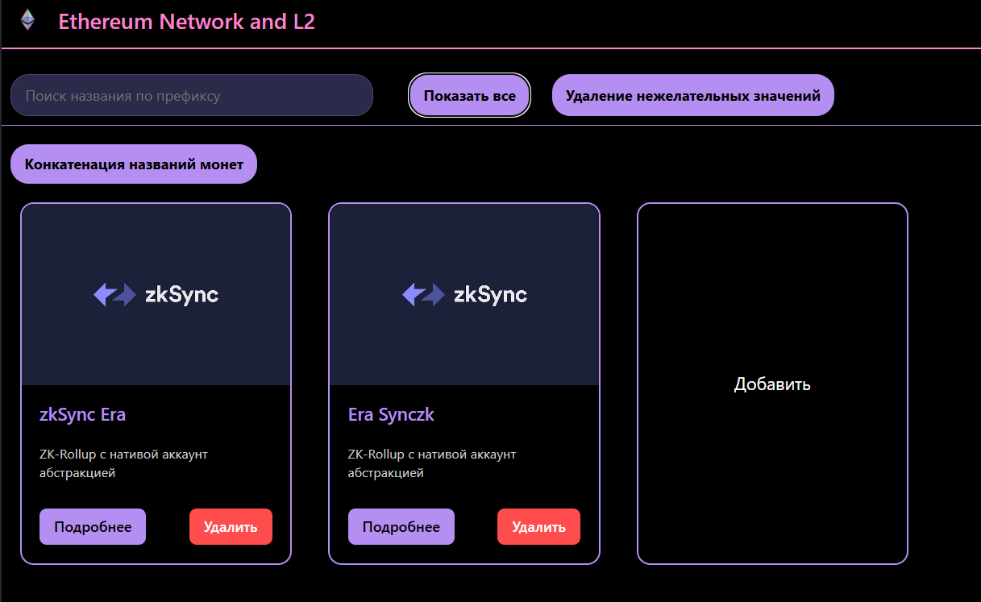


Рисунок 17 – Результат работы поиска анаграмм

Функция конкатенации строк складывает все названия монет для каждой сети второго уровня и выводит результат на главной странице (см. рисунок 18, 19).



Рисунок 18 –Код функции конкатенации строк

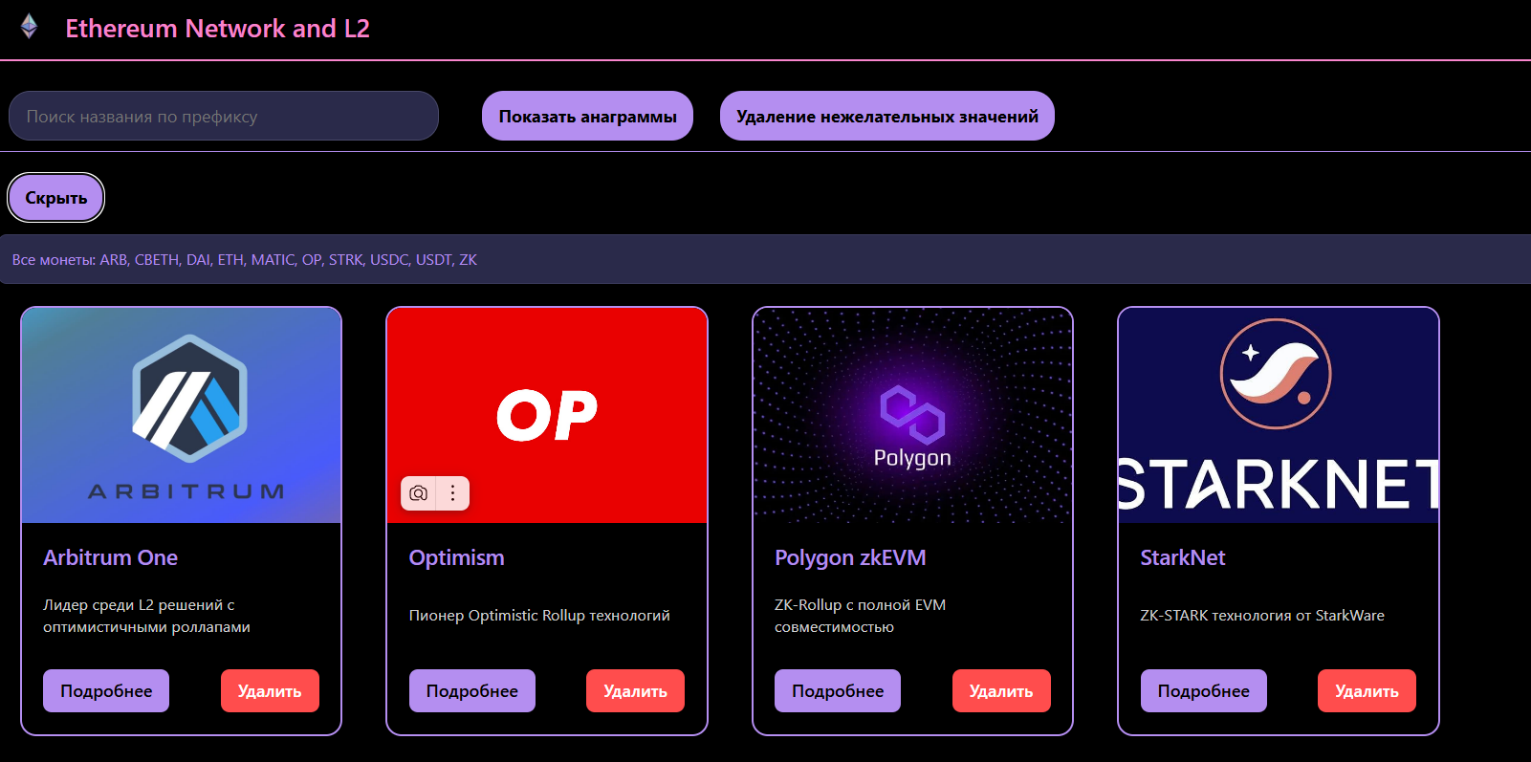


Рисунок 19 –Результат работы конкатенации строк

Функция удаления нежелательных данных удаляет данные карточки, название которой является нежелательным. Для примера в массив с данными была добавлена карточка с неизвестным названием (см. рисунок 20, 21, 22).

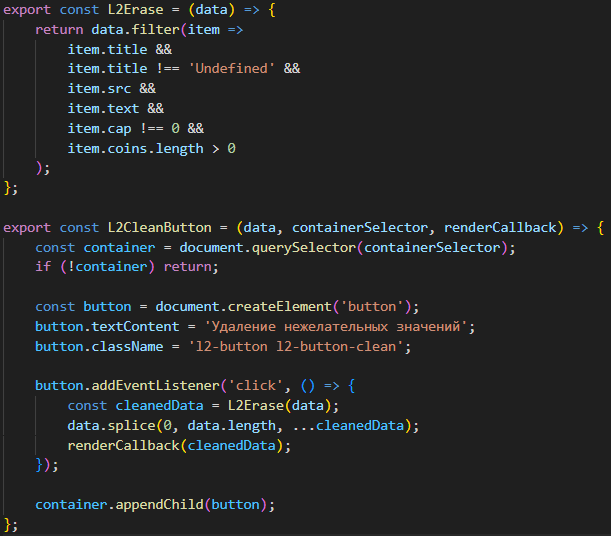


Рисунок 20 –Код функции удаления нежелательных данных

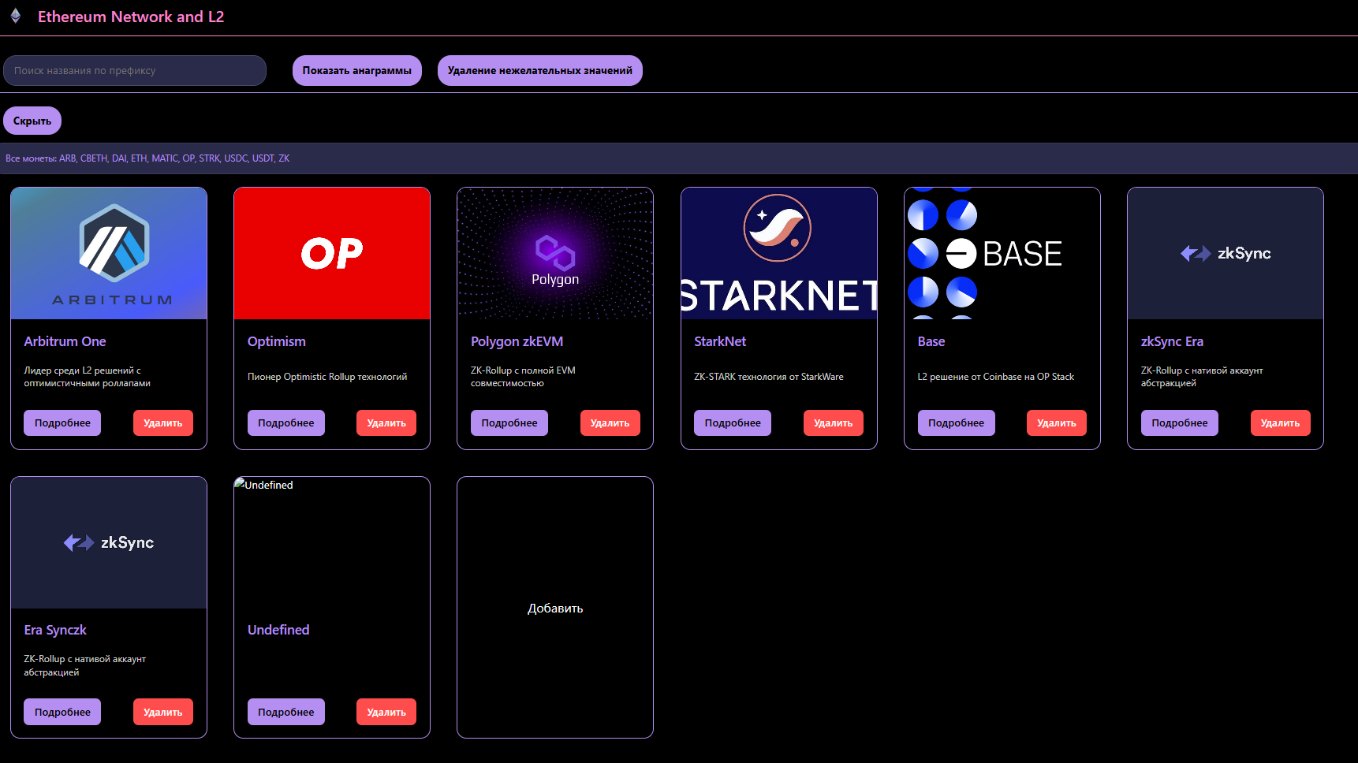


Рисунок 21 – Вид сайта с карточкой с неверными данными

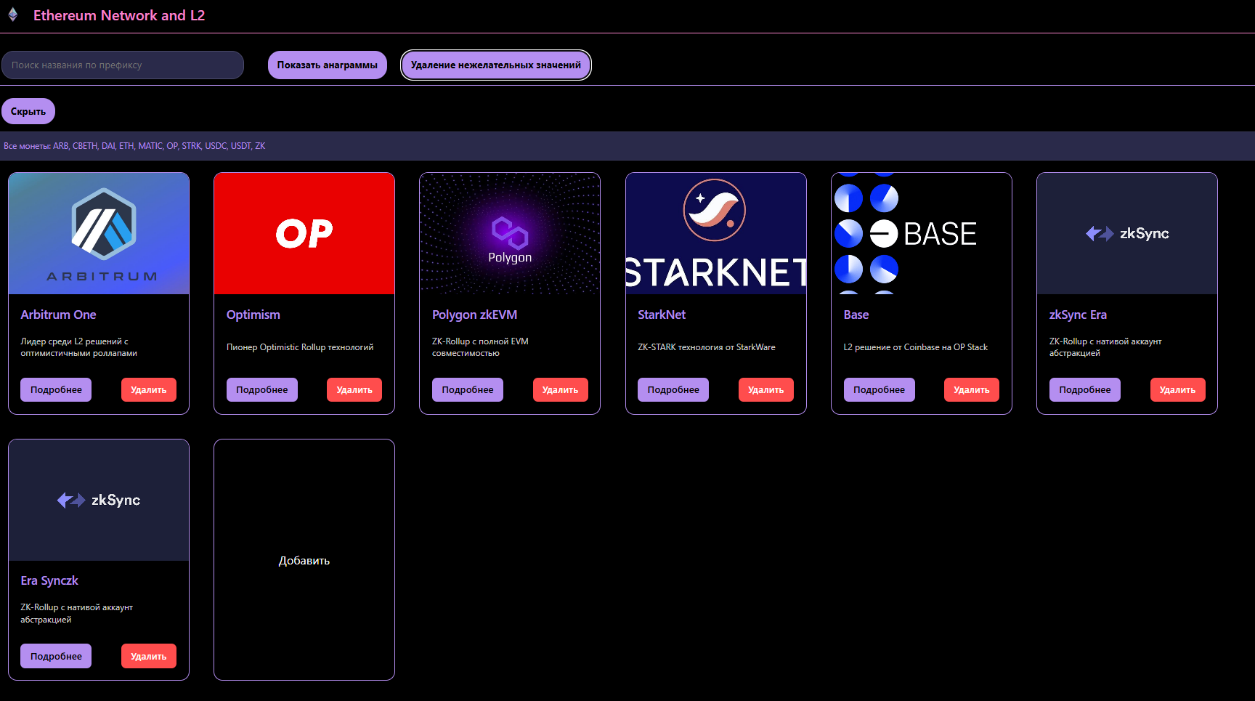


Рисунок 22 – Вид сайта после нажатия на кнопку “Удаление нежелательных значений”

Функция подсчета количества слов с искомым префиксом была интегрирована в сайт как поиск карточки по префиксу ее названия (см. рисунок 20, 21).

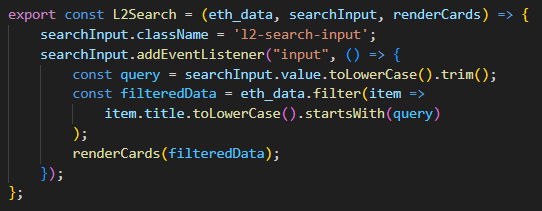


Рисунок 23 –Код функции поиска префикса

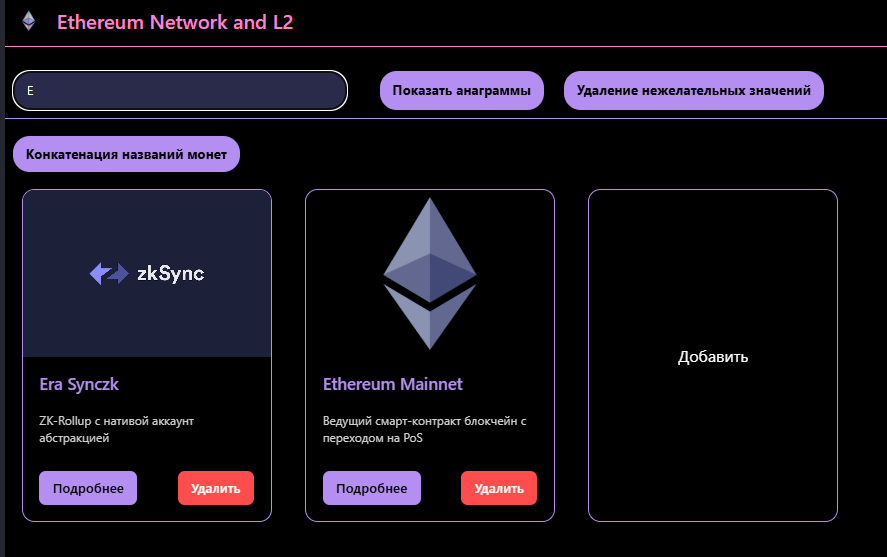


Рисунок 24 – Результат работы поиска

Все функции вызываются в методе render() главной страницы.

**ЛАБОРАТОРНАЯ РАБОТА №4 “СОЗДАНИЕ БЭКЕНДА НА JAVASCRIPT”**

**Задание:** Реализация собственного API на Node.js. Тестирование через Postman/Insomnia 5 методов: список с фильтрацией, получение одной записи, добавление, редактирование, удаление.

В соответствие с методическими указаниями было создано собственное API на NestJS, приведу основные моменты, которые были реализованы в коде.

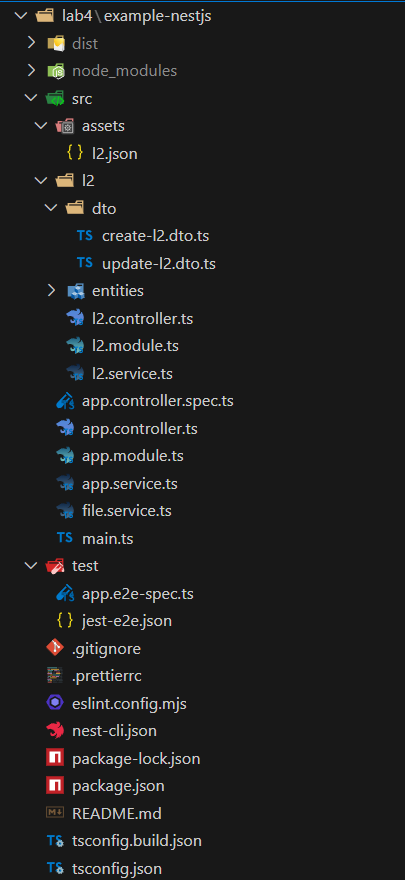


Рисунок 25 – итоговая структура проекта

Исходя из json-файла, карточка имеет следующий интерфейс (см. Рисунок 26).

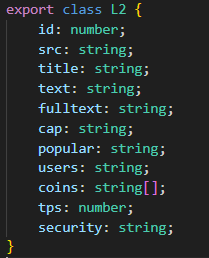


Рисунок 26 – Содержание файла l2.entity.ts

Реализация функций, используемых в запросах (см. Рисунок 27).

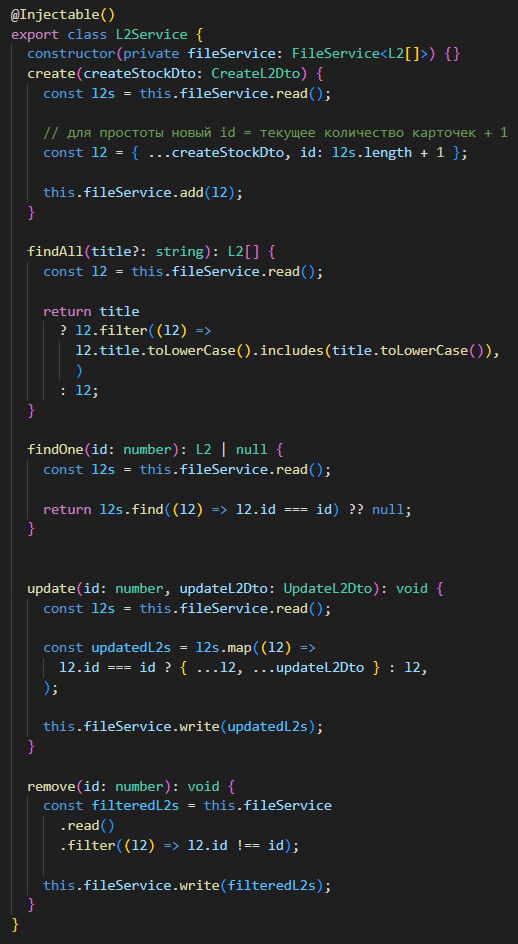


Рисунок 27 – Функции используемые в POST, GET, PATCH, DELETE запросах

Теперь покажем работоспособность всех запросов с помощью Postman. Начала всех эндпоинтов будет одинаковое – <http://localhost:3000>. Сначала проверим получение всех карточек и получение конкретной карточки по названию. (см. Рисунок 28, 29)

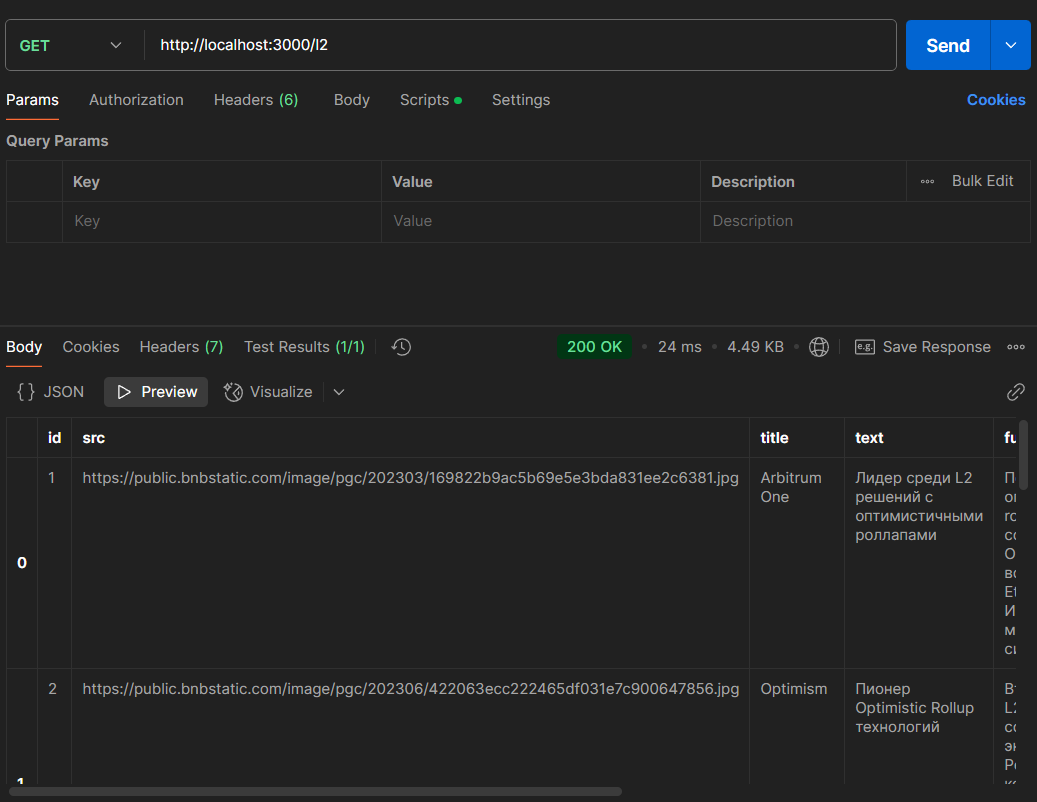


Рисунок 28 –GET-запрос для получения всех карточек

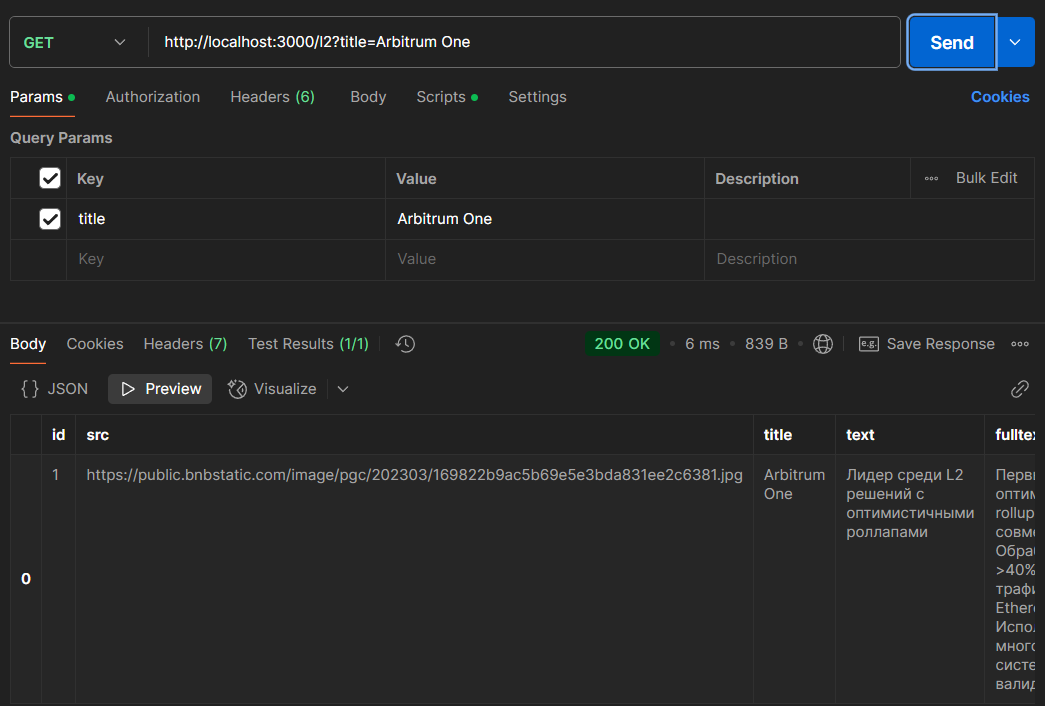


Рисунок 29 –GET-запрос для получения всех карточек

Теперь продемонстрируем работу добавления новой карточки. (см. Рисунок 30)

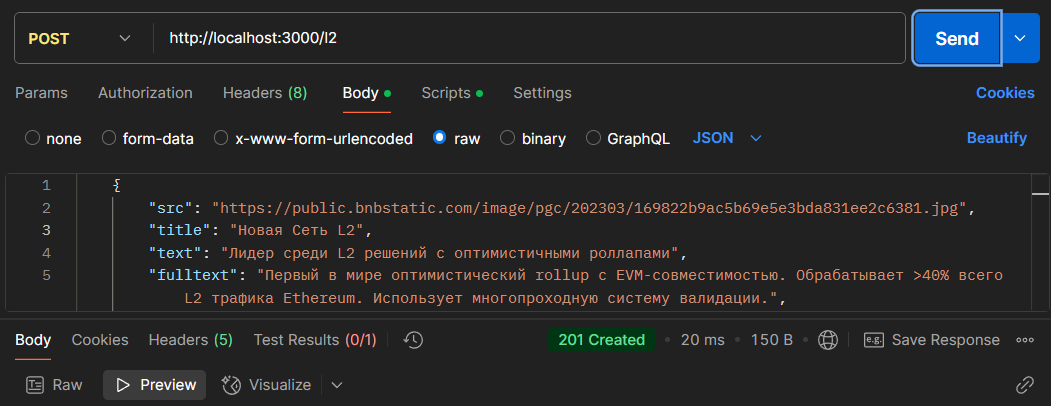


Рисунок 30 – POST-запрос для создания новой карточки

В результате видим, что была добавлена новая карточка с id=9 (см. Рисунок 31).

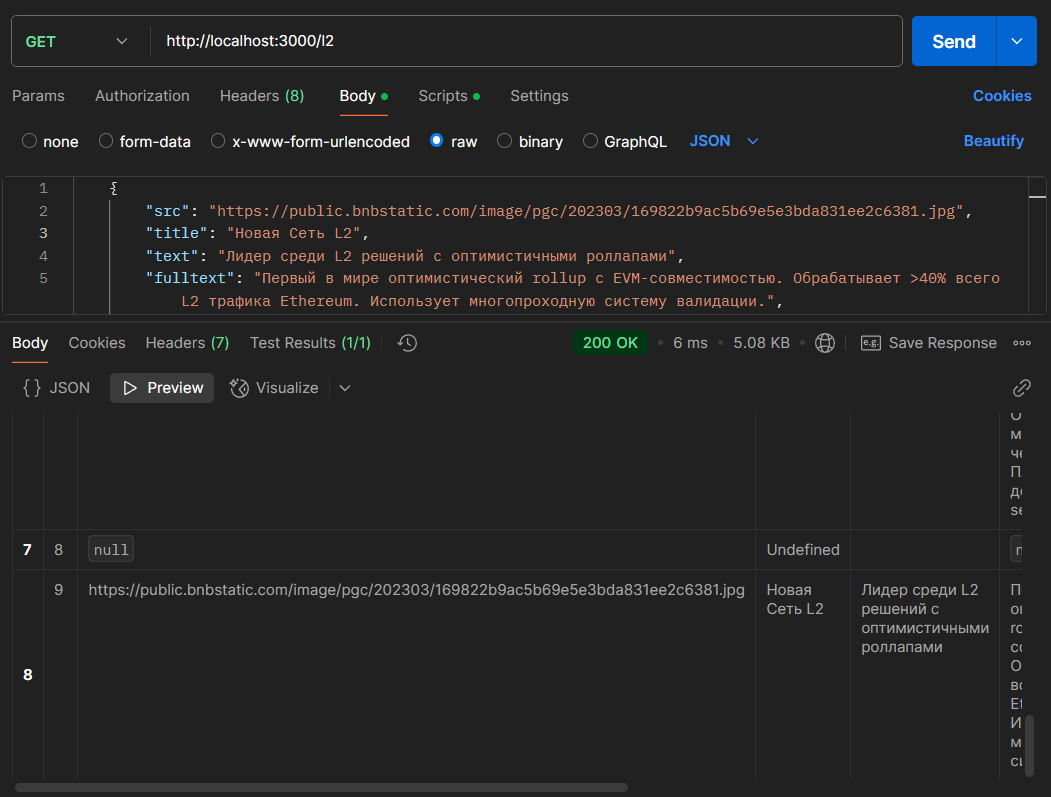


Рисунок 31 – GET-запрос для получения всех карточек

Дальше покажем, как работает изменение карточки (см. Рисунок 32, 33).

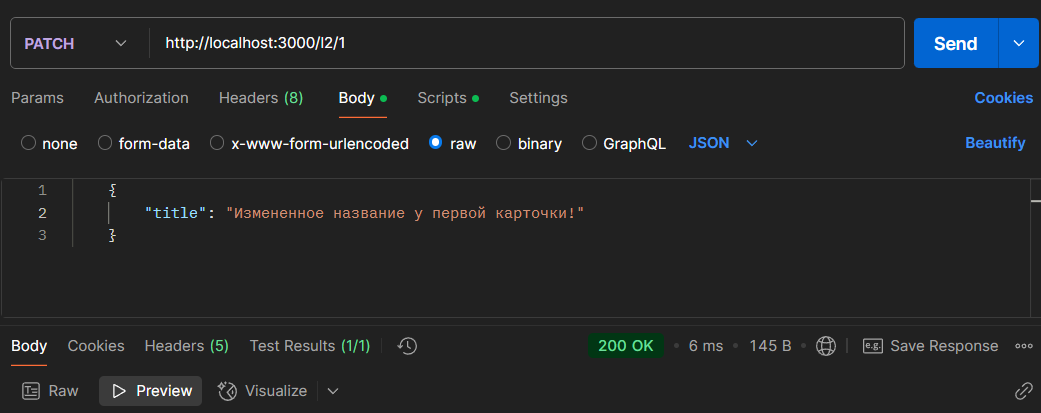


Рисунок 32 – PATCH-запрос для изменения данных карточки

В результате видим, что карточка с id=1 изменила название.

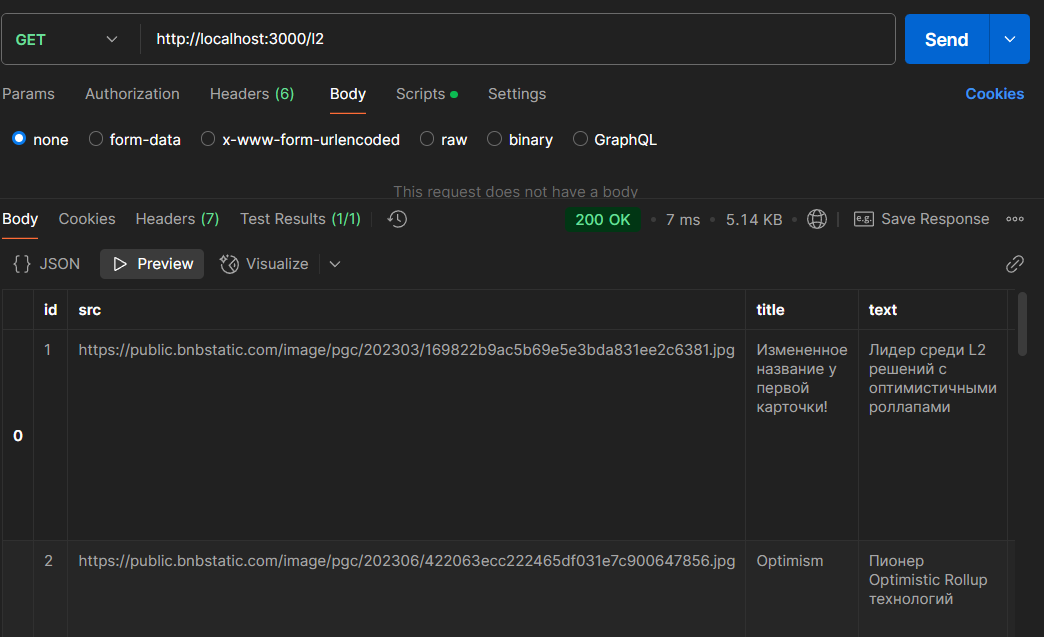


Рисунок 33 – GET-запрос для получения всех карточек

Осталось продемонстрировать работу удаления карточки (см. Рисунок 34, 35).

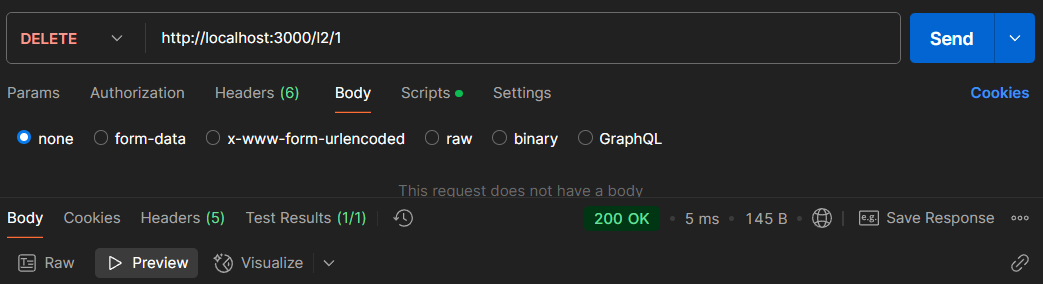


Рисунок 34 – DELETE-запрос для удаления данных первой карточки

В результате видим, что карточка с id=1 была удалена.

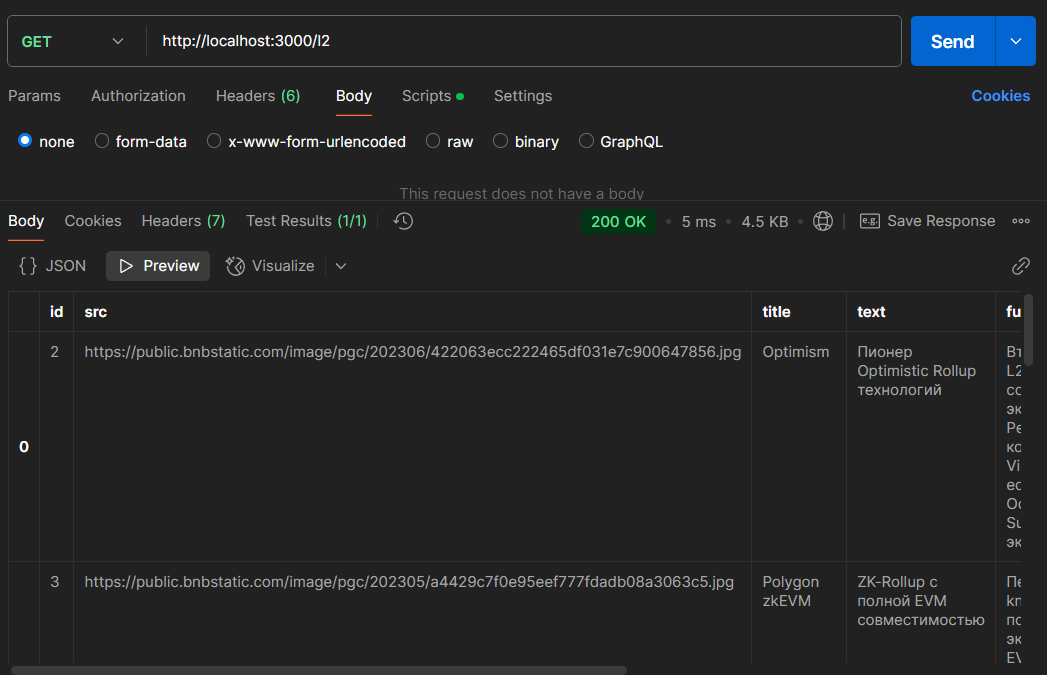


Рисунок 35 – GET-запрос для получения всех карточек

ЗАКЛЮЧЕНИЕ

В ходе работы были достигнуты следующие результаты:

1. Оформлен калькулятор в стилистике веб-ресурса [1]. Github: https://github.com/dranniks/PNA/tree/lab1
2. В калькуляторе реализованы основные операции и операция расчета силы тока. Github: https://github.com/dranniks/PNA/tree/lab2
3. Реализовано двухстраничное приложение, оформленное согласно варианту. Github: https://github.com/dranniks/PNA/tree/lab3
4. Реализована функция поиска анаграмм, поиса по префиксу, кнкатенации названий монет и удаление нежелательных данных среди карточек лабораторной работы №3. Github: https://github.com/dranniks/PNA/tree/lab3\_hw
5. Реализован собственный API на Node.js. Проведено тестирование через Postman/Insomnia 5 методов: список с фильтрацией, получение одной записи, добавление, редактирование, удаление. Github: https://github.com/dranniks/PNA/tree/lab4

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ethereum [Электронный ресурс]. – URL: <https://ethereum.org/ru/> (дата обращения: 9.03.2025)