

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)

Институт математики и информационных технологий
Кафедра алгебраических и информационных систем

**ОТЧЕТ
ПО УЧЕБНОЙ ПРАКТИКЕ**

Студенты 2 курса очного отделения

Группа 02272–ДБ

Яманова Ульяна Сергеевна

Агеева Александра Викторовна

Руководители:

ст. преподаватель Ильин Б.П.

Иркутск 2025

Оглавление

| | |
|---|----|
| ВВЕДЕНИЕ | 3 |
| Раздел 1. Исследование предметной области | 4 |
| 1.1 Описание предметной области | 4 |
| 1.2 Постановка задачи..... | 6 |
| Раздел 2. Обзор технологий разработки..... | 7 |
| 2.1 Visual Studio Code..... | 7 |
| 2.2 HTML..... | 8 |
| 2.3 CSS | 9 |
| 2.4 JavaScript | 9 |
| 2.4 Node.js..... | 10 |
| 2.5 Express..... | 10 |
| 2.6 MAMP..... | 11 |
| 2.7 SQLyog..... | 11 |
| 2.8 MySQL | 12 |
| Раздел 3. Описание реализации проекта | 13 |
| 3.1 Структура сайта..... | 13 |
| 3.1.1 Главная страница | 14 |
| 3.1.2 Минералы и их классификации | 14 |
| 3.1.3 Тесты | 16 |
| 3.1.4 Справочная информация и страница с параллакс-скроллингом | 20 |
| 3.1.5 Мини-игра «Memory cards»..... | 21 |
| 3.2 Серверная часть | 25 |
| 3.2.1 Создание схемы базы данных..... | 25 |
| 3.2.2 Локальный сервер на Express.js..... | 26 |
| 3.2.3 Подключение баз данных к сайту | 27 |
| 3.3 Адаптивная верстка..... | 29 |
| 3.4 Цветовые темы..... | 31 |
| ЗАКЛЮЧЕНИЕ..... | 33 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 34 |
| ПРИЛОЖЕНИЕ 1. SQL-скрипт создания базы данных..... | 36 |
| ПРИЛОЖЕНИЕ 2. Реализация параллакс-скроллинга | 38 |
| ПРИЛОЖЕНИЕ 3. Настройка маршрутов для GET-, POST-, PUT-запросов | 42 |

ВВЕДЕНИЕ

В эпоху цифровизации и активного развития интернет-технологий особую актуальность приобретают специализированные информационные веб-ресурсы, объединяющие научно-популярные и образовательные материалы. Такие ресурсы могут быть полезными при, например, написании различных учебных работ, подготовке тематических мероприятий или расширении общего кругозора. При этом тематика этих цифровых порталов может быть совершенно любой.

Одной из довольно популярных тем научно-популярных информационных ресурсов является тема минералов и драгоценных камней. Однако немалое количество сайтов данной тематики либо содержат выжимки фактов и научной информации (например, Mindat.org), либо же наоборот предоставляют ненаучную информацию, к примеру, о мистических и эзотерических свойствах минералов. Также такие сайты редко содержат инструменты для закрепления изученного материала.

Таким образом, целью реализуемого проекта является разработка научно-популярного сайта «Синюшкин колодец», посвященного тематике минералов и драгоценных камней. Ресурс будет включать в себя как элементы энциклопедии, так и образовательные компоненты, как, например, тестирование по изученной в статьях информации.

Исходя из данной цели, были поставлены следующие задачи:

- 1) разработать структуру и интерфейс сайта;
- 2) реализовать энциклопедическую часть сайта — страницы классификации, отдельных минералов, справочной информации;
- 3) разработать раздел тестов;
- 4) геймифицировать одну из страниц сайта — добавить секретную мини-игру и реализовать параллакс-скроллинг;
- 5) подключить базу данных MySQL;
- 6) реализовать адаптивную верстку и темную тему.

Раздел 1. Исследование предметной области

1.1 Описание предметной области

Интернет — коммуникационная сеть и всемирная система объединённых компьютерных сетей для хранения и передачи информации.

Раньше упоминался как Всемирная сеть и Глобальная сеть, а также просто Сеть. Построена на базе стека протоколов TCP/IP. На основе интернета работает Всемирная паутина (World Wide Web, WWW) и множество других систем передачи данных. К началу 2023 года число пользователей достигло 5,16 млрд человек, что составляет более 50 % от всех жителей планеты Земля.

История интернета началась в 1960-х годах в США как военный и научный проект. Агентство DARPA (Defense Advanced Research Projects Agency) разработало ARPANET — первую сеть, использующую технологию пакетной передачи данных. В 1969 году было отправлено первое сообщение между компьютерами в Калифорнийском университете и Стэнфордском исследовательском институте. К 1980-м годам ARPANET расширилась, а появление протокола TCP/IP стандартизировало обмен данными, заложив основу современного интернета.

В 1990-х годах интернет стал общедоступным благодаря изобретению Всемирной паутины Тимом Бернерсом-Ли. Веб-страницы и браузеры сделали сеть удобной для пользователей. Коммерческие компании начали развивать онлайн-сервисы, а к 2000-м годам интернет превратился в глобальную среду для общения, бизнеса и развлечений. [1]

Интернет-ресурс (веб-ресурсы, веб-сайты, веб-сервисы, сайты) – это совокупность интегрированных средств технического и программно-аппаратного характера, а также информации, предназначенной для публикации во Всемирной паутине. Интернет-ресурс может содержать информацию в текстовой, графической и мультимедийной форме. Каждый интернет-ресурс должен иметь уникальный адрес, который позволяет найти его в Сети.

К Интернет-ресурсам относятся веб-сайты, социальные сети, онлайн-энциклопедии, образовательные платформы, видеохостинги, интернет-магазины и многое другое. Эти ресурсы предоставляют пользователям неограниченные возможности для поиска данных, обучения, общения, работы и развлечений.

В современном мире цифровые сервисы стали неотъемлемой частью повседневной жизни. Они предоставляют уникальные возможности для обучения, работы и развлечений, позволяя нам экономить время и ресурсы. К основным преимуществам использования цифровых платформ и приложений относят доступность информации, удобство использования, экономию времени, интерактивность и многофункциональность. [2]

Одними из самых популярных типов ресурсов стали сайты-энциклопедии.

Сетевая энциклопедия, интернет-энциклопедия или онлайн-энциклопедия — цифровая энциклопедия, к которой открыт доступ из сети Интернет.

Виртуальная форма представления энциклопедической информации открыла новые перспективы в развитии изданий данного жанра, разрешила изменение формата (поиск по запросу без листания страниц, поиск по изображениям, больший объём текста, ссылки, отсутствие необходимости в сокращениях ради экономии бумаги) и увеличила доступность для широкой общественности, как в части свободного доступа к информации, так и в части участия общественности в формировании контента.

Сетевые энциклопедии, как и бумажные, могут быть как универсальными, так и предметными. По способу создания сетевые энциклопедии можно разделить на две большие группы: создаваемые с нуля для интернета и оцифрованные и выложенные в сеть копии бумажных энциклопедий. Как правило, вторые ориентированы только на предъявление информации, в то время как первые интерактивны и позволяют своим читателям быть также и редакторами.

По данным на апрель 2025 года, самый популярный сайт в категории «Словари и энциклопедии» — wikipedia.org. [3]

1.2 Постановка задачи

Разрабатываемый ресурс должен соответствовать следующим требованиям:

- 1) сайт должен содержать информацию о минералах в общем, классификации минералов, каждом классе и каждом минерале в виде таблиц или энциклопедических статей;
- 2) на ресурсе должна присутствовать страница справочной информации, где пользователь может самостоятельно изучить материал на предоставленных ресурсах и порталах;
- 3) ресурс должен предоставлять возможность закрепления материала с помощью тестов по каждой из представленных на сайте тем, причем содержимое тестов должно находиться в базе данных, привязанной к серверу;
- 4) в целях геймификации портала на сайте должна присутствовать страница с интерактивным параллакс-скроллингом, а также секретная мини-игра, рейтинг которой также сохраняется в базу данных;
- 5) на сайте должна быть реализована адаптивная верстка и минимум 2 цветовые темы — светлая и темная.

Раздел 2. Обзор технологий разработки

В качестве инструментов фронтенд-разработки сайта «Синюшкин колодец» использовались язык разметки гипертекстовых документов HTML5, формальный язык описания внешнего вида документа CSS, а также скриптовый язык сценариев JavaScript.

Средой разработки выступало приложение Visual Studio Code, в качестве системы управления базами данных использовалась MySQL.

Для реализации серверной части применялись инструменты Node.js, фреймворк Express, набор ПО для управления локальным веб-сервером MAMP, а также графический интерфейс для системы баз данных MySQL SQLyog.

Далее будут подробнее описаны упомянутые выше программные средства.

2.1 Visual Studio Code

Visual Studio Code (VS Code) — это текстовый редактор, разработанный Microsoft для Windows, Linux и macOS. Он позиционируется как удобный инструмент для кроссплатформенной разработки веб- и облачных приложений. Редактор включает встроенный отладчик, инструменты для работы с Git, подсветку синтаксиса, интеллектуальное автодополнение IntelliSense и средства для рефакторинга кода.

VS Code отличается высокой степенью кастомизации: пользователи могут настраивать темы, сочетания клавиш и конфигурационные файлы под свои нужды. Редактор распространяется бесплатно и разрабатывается как проект с открытым исходным кодом, хотя официальные сборки выпускаются под проприетарной лицензией.

VS Code позволяет решать разные задачи: от простого редактирования файлов до разработки сложных веб- и мобильных приложений. Он поддерживает работу с такими языками программирования как JavaScript, HTML, CSS, Python,

C, C++, Java, SQL и др. Кроме того, можно добавить поддержку дополнительных языков и фреймворков через расширения.

Редактор используют разработчики для создания веб-приложений, API и других программных продуктов; системные администраторы для написания и выполнения скриптов автоматизации; и аналитики данных для анализа больших объемов данных и машинного обучения. [7]

2.2 HTML

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — это стандартизированный язык гипертекстовой разметки, используемый для создания веб-страниц в интернете. Он представляет собой набор элементов, которые служат скелетом любой веб-страницы и задают ее базовую структуру.

Элементы HTML являются строительными блоками HTML страниц. С помощью HTML разные конструкции, изображения и другие объекты, такие как интерактивная веб-форма, могут быть встроены в отображаемую страницу. HTML предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Элементы HTML выделяются тегами, записанными с использованием угловых скобок.

Сам термин «гипертекстовый» означает, что текст содержит в себе ссылки на другие документы или части одного документа. Именно эта особенность стала революционной при создании всемирного интернета, так как позволила связать множество документов в единую систему.

2.3 CSS

CSS (от англ. Cascading Style Sheets — «каскадные таблицы стилей») — формальный язык декорирования и описания внешнего вида веб-страницы, написанного с использованием языка разметки (чаще всего HTML или XHTML).

CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц.

Основной целью разработки CSS является ограждение и отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

2.4 JavaScript

JavaScript (JS) — скриптовый язык программирования, предназначенный для написания сценариев для придания интерактивности HTML-страницам.

JavaScript разработан фирмой Netscape Communication Corporation. Первоначальное название языка — LiveScript. После завоевания языком Java всемирной известности LiveScript из коммерческих соображений переименовали в JavaScript.

Важная особенность JavaScript — объектная ориентированность: программисту доступны многочисленные объекты, такие как документы, гиперссылки, формы, фреймы и т.д. Эти объекты характеризуются описательной информацией (свойствами), а также возможными действиями (функциями). [4]

2.4 Node.js

Node или Node.js — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода.

Платформа Node.js была представлена в 2009 году — ее создал инженер Райан Дал, а спонсором разработки выступила компания Joyent.

Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения и даже программировать микроконтроллеры. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом. [5]

2.5 Express

Express — самый популярный веб-фреймворк для Node. Он является базовой библиотекой для ряда других популярных веб-фреймворков Node. Он предоставляет следующие механизмы:

- написание обработчиков для запросов с различными HTTP-методами в разных URL-адресах (маршрутах);
- интеграция с механизмами рендеринга «view», для генерации ответов, вставляя данные в шаблоны;
- установка общих параметров веб-приложения, такие как порт для подключения, и расположение шаблонов, которые используются для отображения ответа. [6]

2.6 MAMP

MAMP – это бесплатный локальный серверный пакет, который позволяет разработчикам запускать и тестировать веб-приложения на своих компьютерах без необходимости подключения к интернету или использования удалённого хостинга.

Название является аббревиатурой от Macintosh, Apache, MySQL и PHP, хотя сейчас MAMP доступен не только для macOS, но и для Windows. Пакет включает веб-сервер Apache, систему управления базами данных MySQL и интерпретатор PHP.

Одним из ключевых преимуществ MAMP является простота настройки: пользователи могут быстро развернуть локальную среду для тестирования сайтов, не тратя время на ручную установку и конфигурацию компонентов. Пакет также предлагает удобный графический интерфейс для управления серверами, базами данных и настройками.

2.7 SQLyog

SQLyog — графический интерфейс пользователя для популярной системы реляционных баз данных MySQL. Программа создана компанией Webyog Softworks Pvt. Ltd.

SQLyog предоставляет такие функции как визуальное проектирование схем баз данных, SQL-редактор с подсветкой синтаксиса, возможность выполнения сложных запросов и управления пользовательскими правами. Одним из ключевых преимуществ SQLyog является встроенный мастер синхронизации данных, который позволяет сравнивать и обновлять структуру и содержимое баз данных между разными серверами. Программа также поддерживает экспорт и импорт данных в различные форматы, включая CSV, XML и SQL-дампы. [12]

2.8 MySQL

MySQL — это свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы. [8]

Раздел 3. Описание реализации проекта

3.1 Структура сайта

До начала работы над проектом существовал черновой вариант сайта, содержащий главную страницу, страницу классификации минералов, а также три страницы отдельных минералов.

В процессе работы была разработана следующая структура сайта:

- главная страница;
- страница общей классификации минералов;
- страницы классов минералов;
- страницы отдельных минералов;
- раздел тестов, содержащий тестовые вопросы по каждому из представленных на сайте минералов;
- страница справочной информации.

Также с целью геймификации сайта были разработаны две секретные страницы:

- страница-шахта, реализованная с помощью параллакс-скроллинга, где визуально показано расположение минералов по высоте их добычи;
- страница с секретной мини-игрой (memory cards), переход на которую находится в самом низу «шахты».

3.1.1 Главная страница

Главная страница (рис. 1) включает в себя информацию о минералах в целом, картинки-переходы на другие страницы сайта, а также краткий вариант сказки П. Бажова «Синюшкин колодец». В шапке сайта расположены переходы на основные разделы. В подвале сайта находится интерактивный раскрывающийся блок контактной информации и кнопка переключения темы.

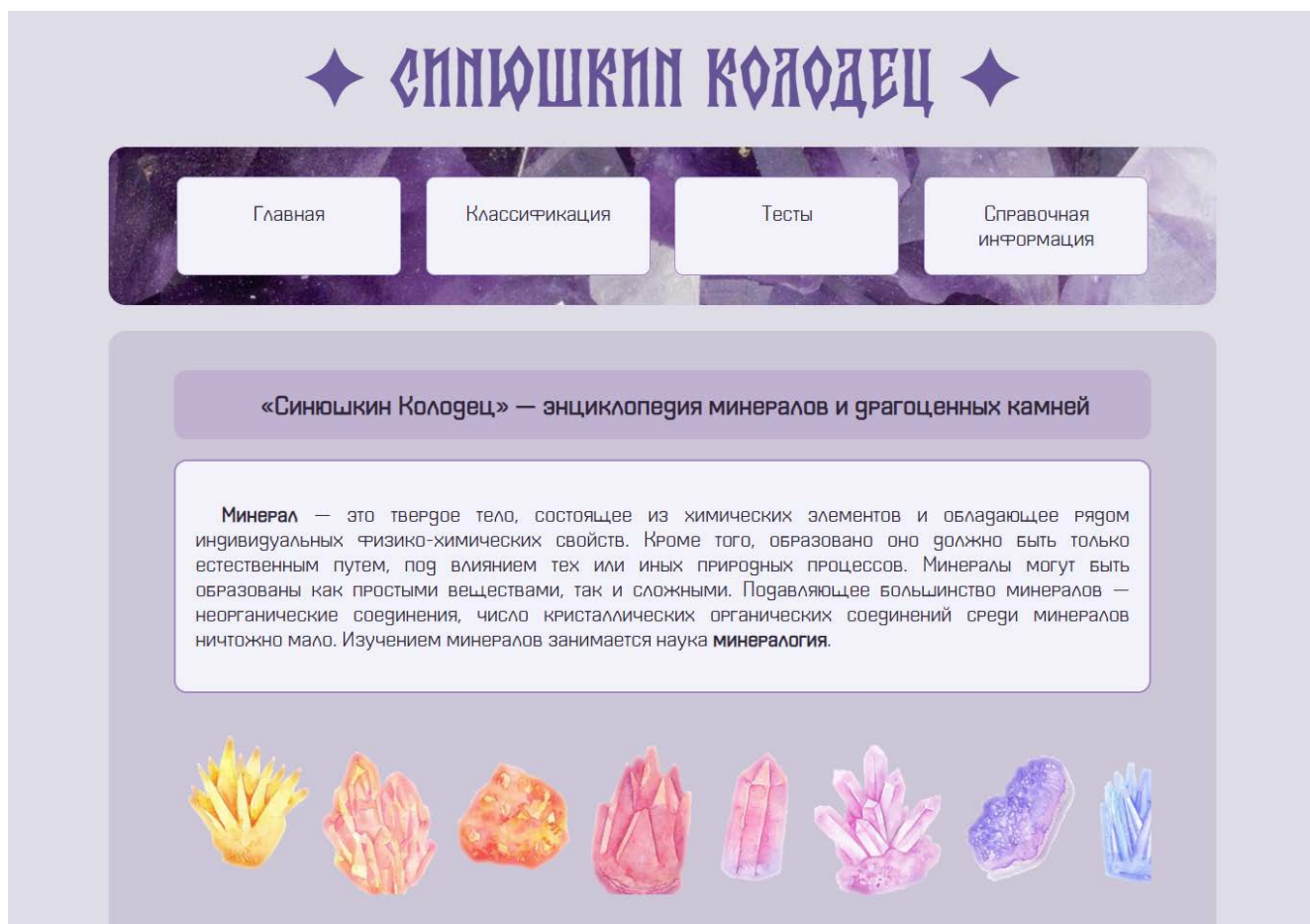


Рисунок 1. Главная страница

3.1.2 Минералы и их классификации

На странице классификации минералов расположена таблица с классами минералов (рис. 2). Из таблицы можно перейти либо на страницу какого-либо минерала, либо на страницу классов. Каждая страница класса содержит описание этого класса и минералы, к нему относящиеся. Каждая страница отдельного минерала содержит краткую информацию о самом минерале (описание, формула, твердость по шкале Мооса), его месторождениях, синтетических аналогах и применении (рис. 3).

| Классификация минералов | | | | | | | | |
|---|---|---|---|--|--|---|---|--|
| Самородные | | | | | | | | |
|  | |  | |  | |  | | |
| Алмаз | | Висмут | | Золото | | Серебро | | |
| Оксиды и гидроксиды | | | | | | | | |
| Кремния |  | |  | |  | |  | |
| | Авантюрин | | Горный хрусталь | | Цитрин | | Яшма | |

Рисунок 2. Общая классификация

Сапфир

← Вернуться к классификации

Сапфир — драгоценный камень, одна из разновидностей корунда. Обычно минералогии называют сапфирами темно синие-корунды, хотя в природе также встречаются экземпляры другой окраски: желтые сапфиры, оранжевые, зеленые, бесцветные, розовые сапфиры, голубые и других цветов. Они имеют одну и ту же химическую формулу, но отличаются содержанием примесей. Своим насыщенно-синим цветом камень обязан наличию соединений титана и железа в молекулярной структуре.

Минерал: Корунг

Твердость по шкале Мооса: 9

Формула: Al_2O_3

Месторождения сапфиров

Наиболее богатые сапфировые россыпи расположены в Юго-Восточной Азии (Таиланд, Мьянма, Китай, Вьетнам), в Индии, Австралии и на территории США. В России промышленная добыча не ведется ввиду отсутствия крупных месторождений. Самыми ценными и красивыми, признанными эталоном, считаются кашмирские сапфиры, получившие свое название от места их добычи — Кашмира (спорная область, занимаемая Индией и Пакистаном). Они имеют сочный сине-васильковый оттенок и не меняют цвета в зависимости от освещения, а за счет микрокристаллических включений малопрозрачны. В кашмирских сапфирах наиболее ярко выражена структура минерала, под определенным углом можно рассмотреть параллельные слои, образующие кристалл.

Натуральные и синтетические сапфиры

Первый искусственный сапфир был синтезирован в США в середине XX века и с тех пор используется в массовом производстве




Рисунок 3. Страница минерала

15

3.1.3 Тесты

В процессе работы над проектом был создан раздел тестов, содержащий набор тестовых заданий по каждой из статей минералов на сайте (рис. 4, 5). Пройденный на 100% тест помечается в списке тестов соответствующей надписью, пометки реализованы с использованием localStorage.



Рисунок 4. Раздел тестов

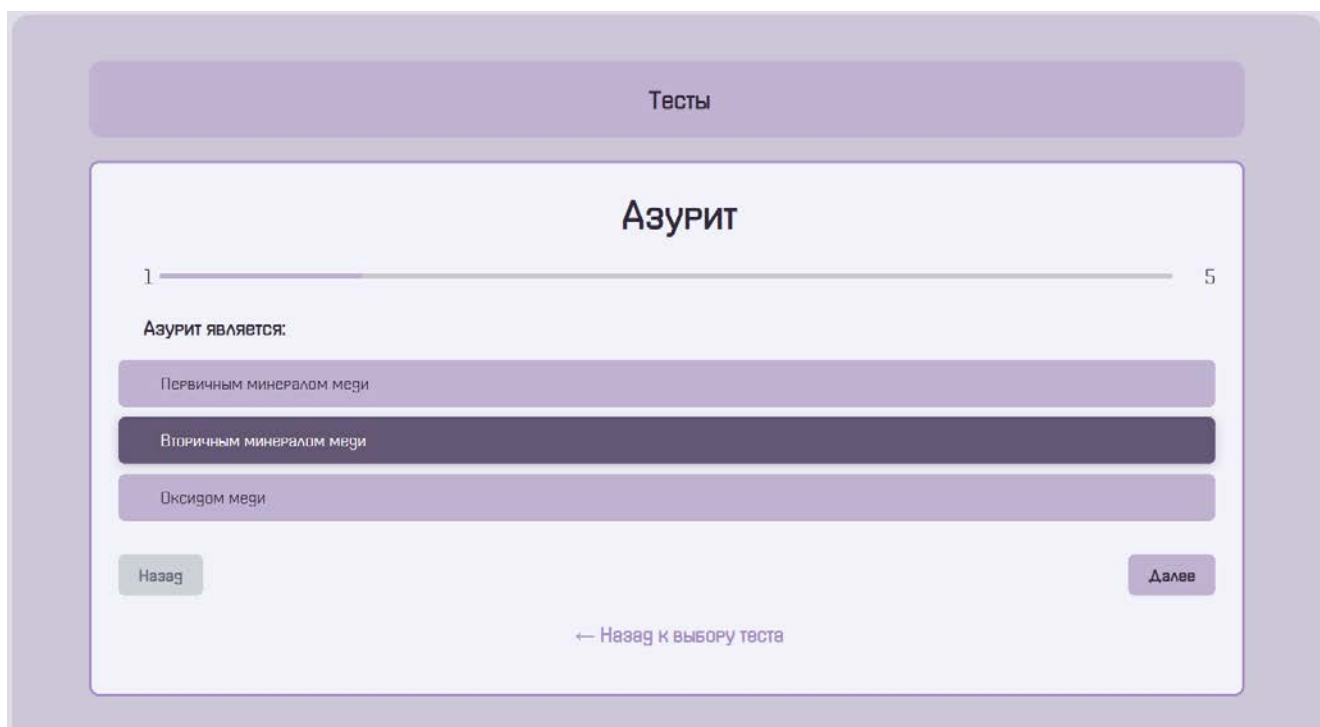


Рисунок 5. Пример тестового задания

Содержимое тестов находится в базе данных. Структура соответствующей базы данных описана в подразделе «Серверная часть».

Интерактивность тестов была полностью реализована с помощью JavaScript. Смена вопросов, возврат на экран выбора теста, отображение результатов обеспечивается сменой CSS-параметра display между «none» (скрытый контент) и «grid» (отображаемый в данный момент контент).

Листинг 1. Функция загрузки страницы выбора теста

```
function initTestSelection() {  
  
    testCategories.innerHTML = "";  
  
    for (const [testId, testData] of Object.entries(testsDatabase)) {  
  
        const testElement = document.createElement('div');  
  
        testElement.className = 'test-category';  
  
        // Если тест был пройден на 100%, добавляем соответствующую  
        надпись  
  
        testElement.innerHTML = `  
  
            <h3>${testData.title}${localStorage.getItem(testData.title) == 1 ? '  
<span class="correct">(100%)</span>' : ''}</h3>  
  
            <p style="font-size:20px;">Вопросов: ${testData.questions.length}</p>  
  
            `;  
  
        testElement.addEventListener('click', () => startTest(testId));  
  
        testCategories.appendChild(testElement);  
  
    }  
}
```

Листинг 2. Функция загрузки вопроса

```
function loadQuestion(index) {  
  
    const test = testsDatabase[appState.currentTest];  
  
    const question = test.questions[index];  
  
    questionText.textContent = question.question;  
  
    optionsContainer.innerHTML = "";  
  
    question.options.forEach((option, i) => {  
  
        const optionElement = document.createElement('div');  
  
        optionElement.className = 'option';  
  
        optionElement.textContent = option;  
  
        if (appState.answers[index] !== undefined && appState.answers[index] === i) {  
            optionElement.classList.add('selected');  
        }  
  
        optionElement.addEventListener('click', () => selectOption(index, i));  
  
        optionsContainer.appendChild(optionElement);  
  
    });  
  
    currentQuestionSpan.textContent = index + 1;  
  
    totalQuestionsSpan.textContent = test.questions.length;  
  
    updateProgressBar(index, test.questions.length);  
  
    updateNavigationButtons(index, test.questions.length);  
  
    resultsContainer.style.display = 'none';  
  
}
```

Листинг 3. Функция загрузки результата теста

```
function showResults() {

  const test = testsDatabase[appState.currentTest];

  let correctAnswers = 0;

  let resultsHTML = `<ol>`;

  test.questions.forEach((question, index) => {

    const userAnswerIndex = appState.answers[index];

    const isCorrect = userAnswerIndex === question.correctAnswer;

    if (isCorrect) correctAnswers++;

    resultsHTML += `<li><p><strong>Вопрос:</strong> ${question.question}</p>

      <p><strong>Ваш ответ:</strong><span class=${isCorrect ? 'correct' :
'incorrect'}>${userAnswerIndex !== undefined ? question.options[userAnswerIndex] :
'Нет ответа'}</span></p>${!isCorrect ? `<p><strong>Правильный ответ:</strong>
${question.options[question.correctAnswer]}</p>` : ''}<p></li>`;

    resultsHTML += `</ol>`;

    resultsHTML += `<h3><p id="res"><strong>Итого:</strong> ${correctAnswers} из
${test.questions.length} `;

    resultPercent = Math.round(correctAnswers / test.questions.length * 100);

    resultsHTML += `( + resultPercent + `%)</p></h3>`;

    if(resultPercent === 100)
localStorage.setItem(testsDatabase[appState.currentTest].title, 1);

    testContainer.style.gap=0;

    resultsContent.innerHTML = resultsHTML;

  }
}
```

3.1.4 Справочная информация и страница с параллакс-скроллингом

Страница справочной информации содержит список полезных ресурсов и ссылок, а также переход в «шахту» (рис. 6) — страницу с параллакс-скроллингом, где расположены минералы по глубине их добычи. При нажатии на минерал во всплывающем окне можно увидеть его название и глубину (рис. 7).

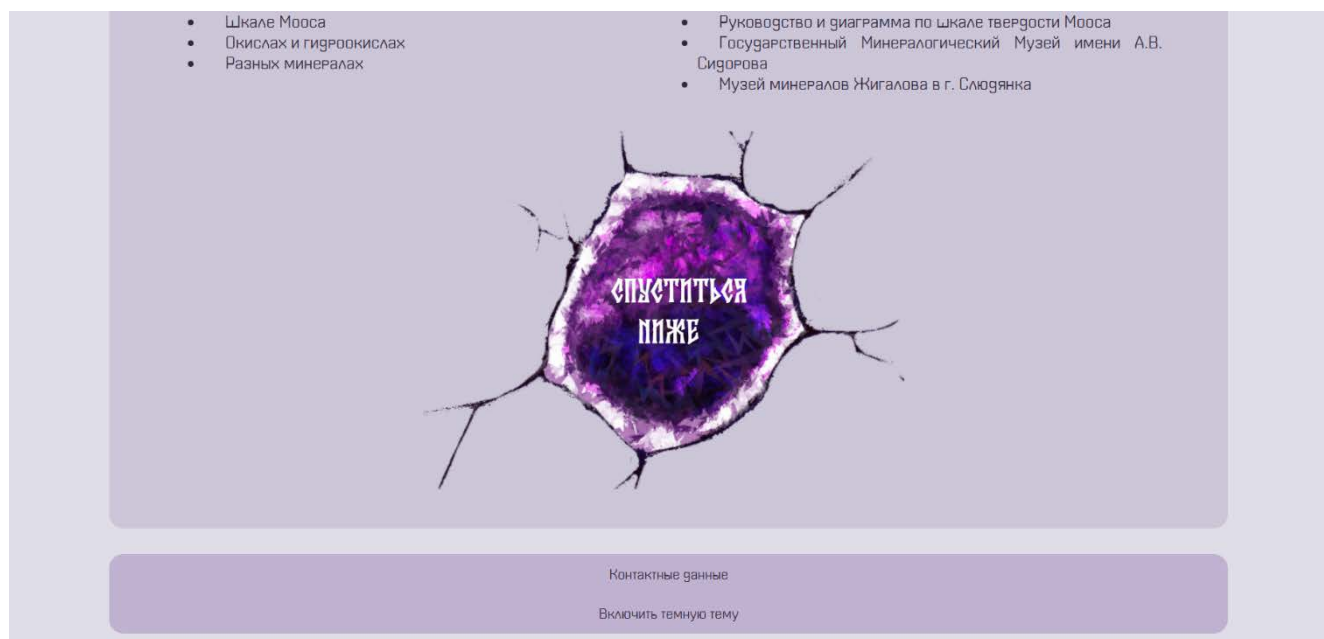


Рисунок 6. Переход в «шахту»



Рисунок 7. Минерал в «шахте»

Параллакс-скроллинг и интерактивность минералов была полностью реализована с помощью инструментов CSS и JavaScript без использования дополнительных библиотек (см. прил. 2).

В самом низу «шахты» расположен стилизованный переход на секретную мини-игру (рис. 8).



Рисунок 8. Секретный переход на мини-игру

3.1.5 Мини-игра «Memory cards»

«Memory cards», или просто «Мемо» — это игра на запоминание, где нужно находить парные картинки среди перевернутых карточек. В начале игры все карточки лежат рубашкой вверх, и игрок открывает по две за ход. Если картинки совпадают — они остаются открытыми, если нет — переворачиваются обратно, и игрок пробует снова. Цель — открыть все пары, запоминая расположение карт.

Данная мини-игра реализована в тематике сайта, поэтому в качестве картинок используются изображения различных кристаллов.

За каждую неудачную попытку нахождения пары игроку прибавляется 2 штрафных секунды. В качестве очков игрока выступает итоговое время прохождения.

После окончания игры пользователю предлагается ввести имя (рис. 9), и после этого его результат добавляется в базу данных. Рейтинговая таблица содержит результаты 10 лучших игроков.

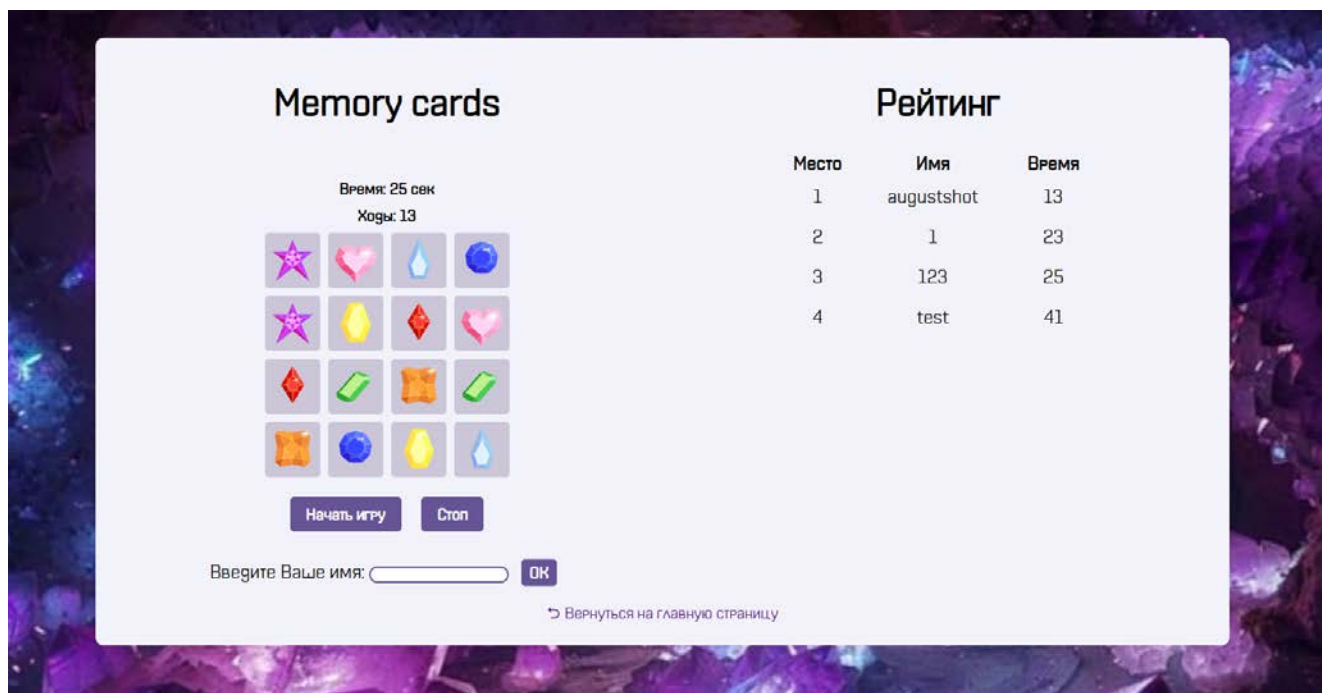


Рисунок 9. Страница мини-игры

Добавление нового результата в таблицу происходит следующим образом: вызывается функция `addRating()`, которая сначала проверяет, содержится ли имя игрока в базе данных. Если такой пользователь есть, то очки пользователя обновляются в базе данных с помощью PUT-запроса, в противном случае в базу данных с помощью POST-запроса добавляется новый пользователь. Затем вызывается функция `getRating()`, которая с помощью GET-запроса получает данные из базы данных и выводит их в рейтинговую таблицу.

Листинг 4. Функция загрузки рейтинговой таблицы

```
async function getRating() {  
  try {  
    const response = await fetch('/rating');  
    if (!response.ok) throw new Error('Не удалось загрузить рейтинг');  
    ratingDatabase = await response.json();  
  } catch (error) console.error('Ошибка загрузки рейтинга:', error);  
  const sortedParticipants = [...ratingDatabase].sort((a, b) => a.score - b.score);  
  const top10 = sortedParticipants.slice(0, 10);  
  let tableHTML =  
  `<table><thead><tr><th>Место</th><th>Имя</th><th>Время</th></tr></thead><tbody>`;   
  if(ratingDatabase.length == 0)  
    tableHTML += `<tr><td colspan=3 >Здесь пока никого нет!</td></tr>`;   
  else{  
    top10.forEach((participant, index) => {  
      tableHTML += `<tr><td>${index + 1}</td><td>${participant.name}</td><td>${participant.score}</td></tr>`;   
    })  
    tableHTML += `</tbody></table>`;   
    document.getElementById("table").innerHTML = tableHTML;  
  }  
}
```

Листинг 5. Функция добавления результата в базу данных

```
async function addRating(e) {  
  e.preventDefault();  
  
  form.classList.add("hidden");  
  
  const name = document.getElementById('name').value;  
  
  const score = seconds;  
  
  try {  
    const names = ratingDatabase.map(item => item.name);  
  
    if(names.includes(name)){ // есть пользователь - обновляем данные  
  
      const response = await  
fetch(`http://localhost:3000/rating/${encodeURIComponent(name)}`, {  
  method: 'PUT', headers: {'Content-Type': 'application/json'},  
  body: JSON.stringify({score: score})});  
  
  if (!response.ok) throw new Error('Ошибка при обновлении данных'); }  
else { // нет пользователя - создаем нового  
  
  const response = await fetch('http://localhost:3000/rating', {  
    method: 'POST', headers: {'Content-Type': 'application/json'}, },  
    body: JSON.stringify({name: name, score: score}));  
  
    if (!response.ok) throw new Error('Ошибка при добавлении пользователя'); }  
  
  document.getElementById('addUserForm').reset();  
  
  getRating();  
  
  } catch (error) console.error('Ошибка:', error);  
}
```


3.2 Серверная часть

При разработке сайта возникла необходимость использования баз данных в двух разделах — разделе тестов и секретной мини-игре.

Содержимое тестов необходимо было поместить в базу данных для более удобной корректировки вопросов, добавления новых или удаления старых тестов. В мини-игре требовалось создать систему отслеживания рейтинга игроков, которая позволяла бы добавлять в рейтинговую таблицу новых игроков или изменять данные уже существующих.

3.2.1 Создание схемы базы данных

Для работы с данными была выбрана база данных MySQL, для создания схемы базы использовался сервис drawDB.

Структура базы данных тестов выглядит следующим образом (рис. 10):

- таблица `tests`, содержащая поля `id` и `title` (тема теста);
- таблица `questions`, содержащая поля `id`, `test_id` (`id` теста, к которому относится этот вопрос), `question` (текст вопроса) и `correct_ans` (индекс правильного ответа в массиве вариантов);
- таблица `variants`, содержащая поля `id`, `question_id` (`id` вопроса, к которому относится этот вариант ответа) и `text` (текст варианта ответа).

Все поля `id` в таблицах являются первичными ключами, то есть являются уникальными.

Между таблицами были настроены следующие связи:

- `tests (id)` — `questions (test_id)` (один ко многим);
- `questions (id)` — `variants (question_id)` (один ко многим).

Таблица `rating`, хранящая данные о рейтинге игроков мини-игры, содержит поля `id`, `name` (имя игрока) и `score` (очки игрока).

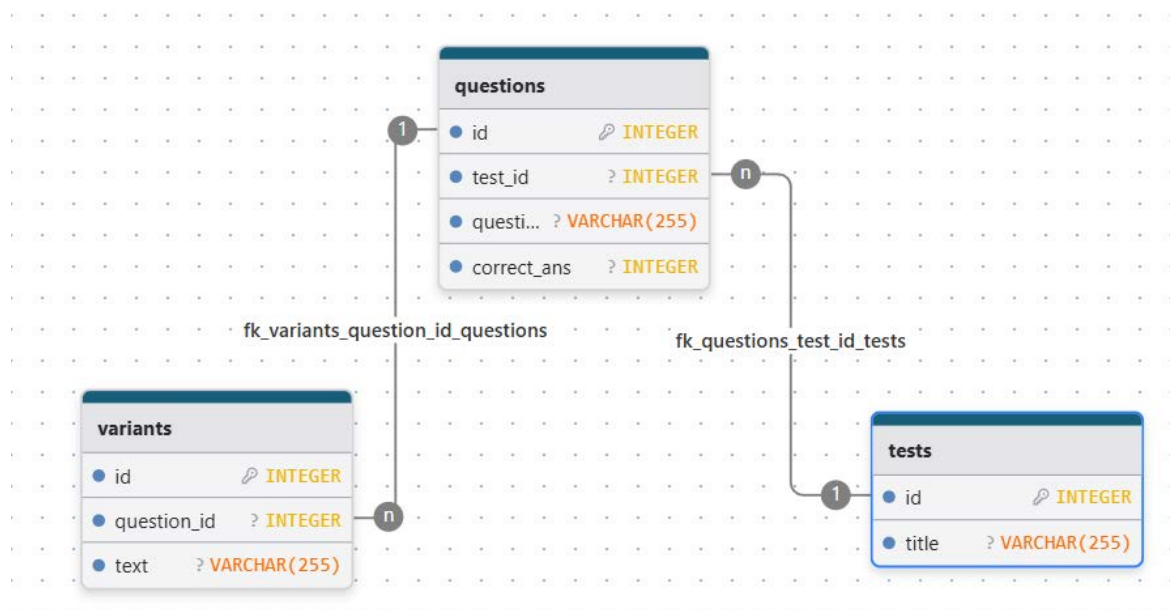


Рисунок 10. Схема базы данных тестов

Далее таблицы были перенесены в базу данных с помощью SQL-запросов, сгенерированных при экспорте созданных ранее схем из drawDB (прил. 1). В процессе создания и редактирования баз данных использовался графический интерфейс SQLyog.

3.2.2 Локальный сервер на Express.js

Для создания локального сервера использовались следующие программы и инструменты:

- MAMP — необходим для создания локальной среды разработки, обеспечивает работу локального сервера;
- Node.js — упрощает создание локального сервера и позволяет создать API для взаимодействия сайта с базой данных;
- Express.js — упрощает настройку маршрутов для HTTP-запросов.

Локальный сервер запускается через командную строку с помощью команды `node server.js`, где `server.js` — серверный файл.

Листинг 6. Настройка сервера

```
const pool = require('./db');

const bodyParser = require('body-parser');

const express = require('express');

const port = 3000;

const app = express();

const path = require('path');

app.use(express.static(path.join(__dirname, 'public')));

const server = app.listen(port, (error) => {

  if (error) return console.log(`Error: ${error}`);

  console.log(`Server listening on port ${server.address().port}`);

});
```

3.2.3 Подключение баз данных к сайту

Для подключения базы данных на локальном сервере потребовалось создать дополнительный файл конфигурации базы данных.

Также в серверном файле были настроены маршруты для GET-, POST- и PUT-запросов (прил. 3).

Благодаря этой настройке данные из базы данных представлялись по заданным маршрутам в виде JSON-файлов, а затем использовались в JS-файлах для работы скриптов соответствующих JSON-страниц с помощью fetch-запросов.

Листинг 7. Файл конфигурации базы данных db.js

```
const mysql = require('mysql');

const config = {

  host: 'localhost',

  user: 'root',

  password: 'root',

  database: 'api',

};

const pool = mysql.createPool(config);

module.exports = pool;
```

Листинг 8. Пример fetch-запроса (получение тестов из базы данных)

```
async function loadTests() {

  try {

    const response = await fetch('/tests');

    if (!response.ok) throw new Error('Не удалось загрузить тесты');

    testsDatabase = await response.json();

    testsDatabase = SQLtoJson(testsDatabase);

    initTestSelection();

  } catch (error) {

    console.error('Ошибка загрузки тестов:', error);

    testCategories.innerHTML = '<p class="error">Не удалось загрузить тесты. Пожалуйста, попробуйте позже.</p>';
  }
}
```

3.3 Адаптивная верстка

Для всех страниц сайта была реализована адаптивная верстка с использованием медиа-запросов в CSS-файлах проекта. Верстка на grid значительно упростила реализацию адаптивного интерфейса.

Листинг 9. Пример реализации адаптивного интерфейса (раздел тестов)

```
@media screen and (width<=750px){

    .test-categories {

        display: grid;

        grid-template-columns: repeat(1, 1fr);

    }

}

@media screen and (width>750px) and (width<=1400px) {

    .test-categories {

        display: grid;

        grid-template-columns: repeat(2, 1fr);

    }

}

@media screen and (width>1400px) {

    .test-categories {

        display: grid;

        grid-template-columns: repeat(3, 1fr);

    }

}
```

Ниже представлены примеры внешнего вида сайта на устройствах с различной шириной экрана (рис. 11, 12, 13).

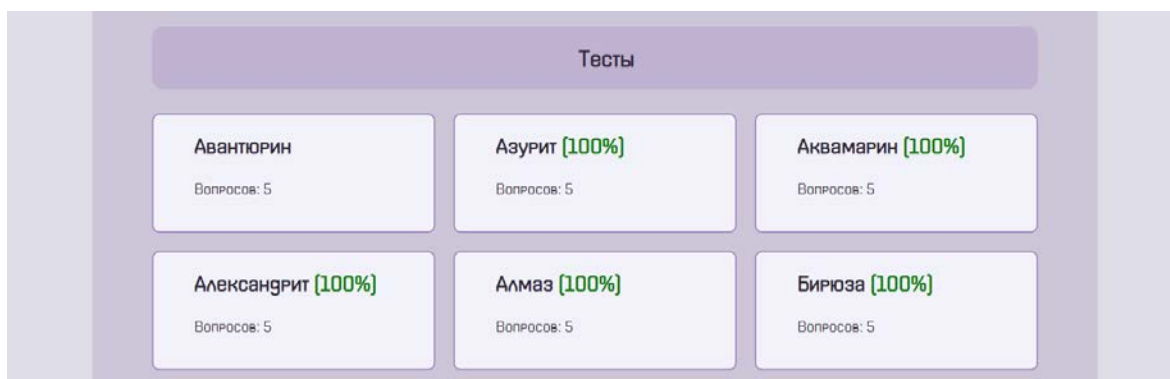


Рисунок 11. Вид страницы на устройстве с большой шириной экрана



Рисунок 12. Вид страницы на устройстве со средней шириной экрана

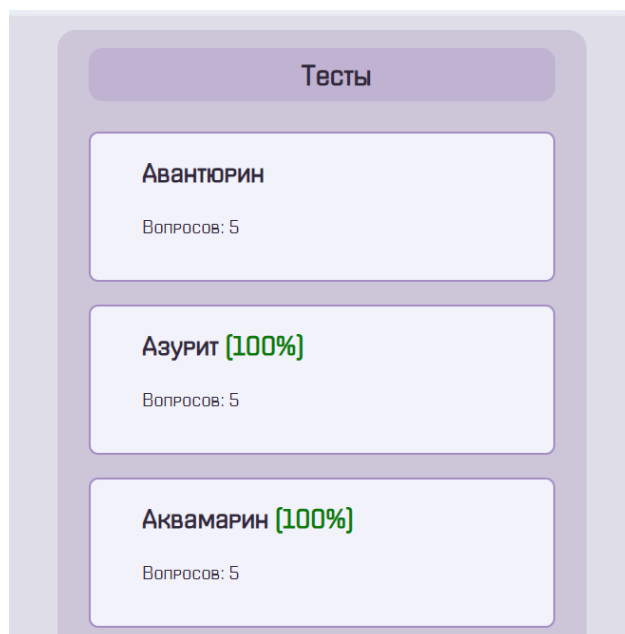


Рисунок 13. Вид страницы на устройстве с небольшой шириной экрана

3.4 Цветовые темы

На сайте реализованы две цветовые темы — светлая и темная. Необходимые переменные, содержание цвета или фоны, были заданы в файле `index.css`, подключенном ко всем страницам сайта (кроме «шахты» и мини-игры), а переключения темы с использованием `localStorage` было реализовано в файле `theme.js`.

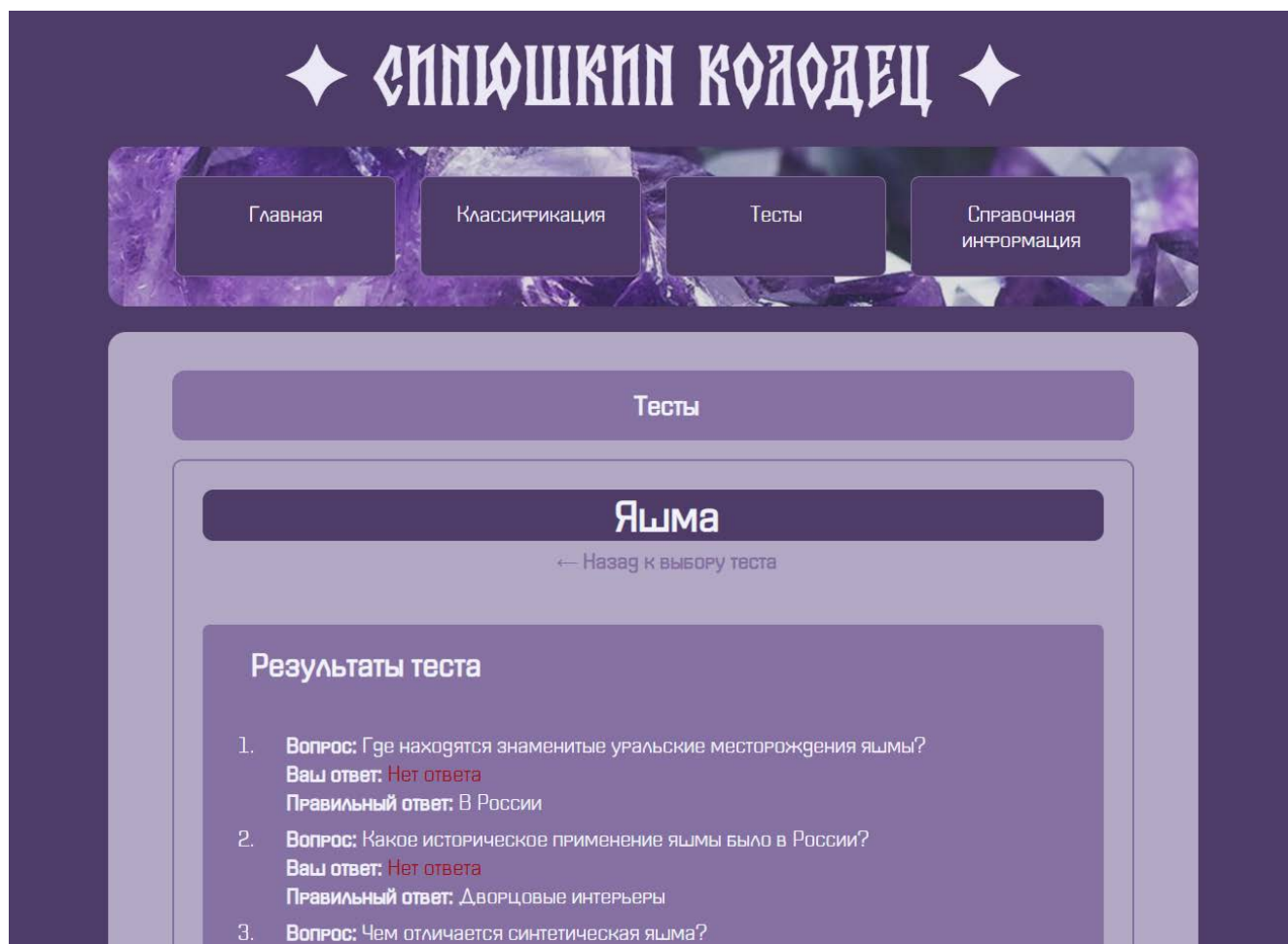


Рисунок 14. Темная тема

Листинг 10. Переключение цветовой темы

```
const html = document.querySelector("html");

(function () {

    window.addEventListener("load", init);

    function init() {

        document.getElementById("theme").addEventListener("click", setTheme);

        if (window.localStorage) {

            if (localStorage.getItem("data-theme") == "dark") {

                html.setAttribute("data-theme", "dark");

                document.getElementById("theme").innerHTML = "Включить светлую тему";

            } else

                document.getElementById("theme").innerHTML = "Включить темную тему";

        }

    }

})();

function setTheme() {

    if (window.localStorage) {

        if (localStorage.getItem("data-theme") == "dark") {

            html.setAttribute("data-theme", "light");

            document.getElementById("theme").innerHTML = "Включить темную тему";

            localStorage.setItem("data-theme", "light");

        } else {

            html.setAttribute("data-theme", "dark");

            document.getElementById("theme").innerHTML = "Включить светлую тему";

            localStorage.setItem("data-theme", "dark");

        }

    }

}
```


ЗАКЛЮЧЕНИЕ

В процессе реализации проекта были достигнуты следующие задачи:

- разработка структуры и интерфейса сайта;
- реализация страницы общей классификации, 4 страниц классов минералов и 24 страниц минералов;
- разработка раздела тестов;
- реализация параллакс-скроллинга;
- реализация секретной мини-игры и рейтинговой таблицы;
- подключение базы данных и использование ее содержимого из JS-файлов;
- реализация адаптивной верстки и темной темы.

Цель проекта была достигнута — был успешно создан сайт-энциклопедия минералов и драгоценных камней «Синюшкин колодец», содержащий также систему тестов для проверки изученных знаний и геймифицированные страницы.

Из идей дальнейшего развития проекта можно выделить следующие:

- размещение сайта на хостинге;
- расширение содержимого сайта и базы данных тестов;
- добавление поощрений за некоторые достижения (например, новая цветовая тема как награда за стопроцентное прохождение всех тестов);
- расширение игровой площадки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Интернет: что это такое, когда его придумали и как он устроен [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://hi-tech.mail.ru/review/125020-internet/> (дата обращения 30.05.2025).
2. Что такое интернет-ресурс? [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: https://www.glossary-internet.ru/terms/И/internet_resurs/ (дата обращения 30.05.2025).
3. Рейтинг лучших сайтов словарей и энциклопедий за апрель 2025 г. [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://www.similarweb.com/ru/top-websites/reference-materials/dictionaries-and-encyclopedias/> (дата обращения 30.05.2025).
4. JavaScript [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://developer.mozilla.org/ru/docs/Web/JavaScript> (дата обращения 30.05.2025).
5. Документация Node.js [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://nodejs.org/api/all.html> (дата обращения 30.05.2025).
6. Express routing [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://expressjs.com/en/guide/routing.html> (дата обращения 30.05.2025).
7. Документация Visual Studio Code [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://code.visualstudio.com/docs> (дата обращения 30.05.2025).
8. David Stokes MySQL and JSON: A Practical Programming Guide [Текст] / David Stokes — 1. — : McGraw Hill, 2018 — 128 с.
9. Code Your First API With Node.js and Express: Understanding REST APIs [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://code.tutsplus.com/code-your-first-api-with-nodejs-and-express-understanding-rest-apis--cms-31697t> (дата обращения 01.06.2025).

10. Создаем наш первый API при помощи Node.js и Express: Создаем сервер [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://code.tutsplus.com/ru/code-your-first-api-with-nodejs-and-express-set-up-the-server--cms-31698t> (дата обращения 01.06.2025).
11. Создаем наш первый API при помощи Node.js и Express: Подключаем базу данных [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://code.tutsplus.com/ru/code-your-first-api-with-nodejs-and-express-connect-a-database--cms-31699t> (дата обращения 01.06.2025).
12. Resource Central - Webyog [Электронный ресурс]. – [Б. м. : б.и], 2025. – URL: <https://webyog.com/resource-central/> (дата обращения 01.06.2025).

ПРИЛОЖЕНИЕ 1.

SQL-скрипт создания базы данных

```
CREATE TABLE `tests` (  
    `id` INTEGER NOT NULL AUTO_INCREMENT UNIQUE,  
    `title` VARCHAR(255),  
    PRIMARY KEY(`id`)  
);
```

```
CREATE TABLE `questions` (  
    `id` INTEGER NOT NULL AUTO_INCREMENT UNIQUE,  
    `test_id` INTEGER,  
    `question` VARCHAR(255),  
    `correct_ans` INTEGER,  
    PRIMARY KEY(`id`)  
);
```

```
CREATE TABLE `variants` (  
    `id` INTEGER NOT NULL AUTO_INCREMENT UNIQUE,  
    `question_id` INTEGER,  
    `text` VARCHAR(255),  
    PRIMARY KEY(`id`)  
);
```

```
CREATE TABLE `rating` (  
    `id` INTEGER NOT NULL AUTO_INCREMENT UNIQUE,  
    `name` VARCHAR(255),  
    `score` INTEGER,  
    PRIMARY KEY(`id`)  
);
```

```
ALTER TABLE `questions`  
ADD FOREIGN KEY(`test_id`) REFERENCES `tests`(`id`)  
ON UPDATE NO ACTION ON DELETE NO ACTION;  
ALTER TABLE `variants`  
ADD FOREIGN KEY(`question_id`) REFERENCES `questions`(`id`)  
ON UPDATE NO ACTION ON DELETE NO ACTION;
```

ПРИЛОЖЕНИЕ 2.

Реализация параллакс-скроллинга

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

  const maxDepth = 50000;

  let lastScroll = 0;

  let ticking = false;

  //установка скорости для движения слоев

  const layers = [

    { element: document.getElementById('layer1'), speed: 0.3, height: 600 },

    { element: document.getElementById('layer2'), speed: 1, height: 800 },

    { element: document.getElementById('layer3'), speed: 1.6, height: 1000 }

  ];

  //настройки отображения для камней - минимальная и максимальная высота
  при которой они видны

  const stones = Array.from({length: 23}, (_, i) => ({

    element: document.querySelector(`.stone-${i+1}`),

    minDepth: document.querySelector(`.stone-${i+1}`).offsetTop-2000,

    maxDepth: document.querySelector(`.stone-${i+1}`).offsetTop +

document.documentElement.clientHeight*3

  }));

  //StoneData – массив значений минерал-информация

  const stonesData = [<...>];

  //ищем окно с описанием камней
```

```

const tooltip = document.getElementById('tooltip');

let activeTooltip = null;

// Обработчики для камней

stones.forEach((stone, index) => {

  stone.element.addEventListener('click', (e) => {

    // Закрываем предыдущее окно

    if (activeTooltip) activeTooltip.classList.remove('visible');

    // Позиционируем новое окно

    const rect = stone.element.getBoundingClientRect();

    const x = stone.element.offsetLeft;

    const y = stone.element.offsetTop - stone.element.offsetHeight;

    tooltip.style.left = `${x}px`;

    tooltip.style.top = `${y}px`;

    // Заполняем данными

    tooltip.innerHTML = `<h3>${stonesData[index].name}</h3>

      <p><strong>Глубина:</strong> ${stonesData[index].depth} м</p>`;

    tooltip.classList.add('visible');

    activeTooltip = tooltip;

    // Закрытие при клике вне окна

    e.stopPropagation();

  });

});

```

```

// Заккрытие при клике в любом есте
document.addEventListener('click', () => {

  if (activeTooltip) {

    activeTooltip.classList.remove('visible');

    activeTooltip = null;

  });

// Оптимизация: предварительный расчет движения слоев при прокрутке
layers.forEach(layer => {

  layer.element.style.height = `${layer.height * 3}px`;

  layer.element.style.willChange = 'transform';

});

stones.forEach(stone => {

  stone.element.style.height = `${stone.height * 3}px`;

  stone.element.style.willChange = 'transform, opacity';

});

//двигаем

function updateParallax(scrollPos) {

  const depthPercent = (scrollPos+document.documentElement.clientHeight) /
maxDepth * 100;

  // Оптимизированный параллакс

  layers.forEach(layer => {

    const yPos = scrollPos * layer.speed;

    layer.element.style.backgroundPosition = `0 ${-yPos}px`; });

```



```

// Индикатор глубины

document.getElementById('depthFill').style.height = `${100-depthPercent}%`;

document.getElementById('depthText').textContent = `Глубина:
${Math.floor((scrollPos+document.documentElement.clientHeight)/50)} м`;

// Камни с оптимизированными проверками

stones.forEach(stone => {

    const isVisible = scrollPos > stone.minDepth && scrollPos < stone.maxDepth;

    stone.element.style.opacity = isVisible ? '1' : '0';

});

const finalImage = document.getElementById('deepFinalImage');

const finalImageStartDepth = 45000; // Когда начинать показ низа

if (scrollPos >= finalImageStartDepth) finalImage.style.opacity = 1;

else finalImage.style.opacity = 0; }

//прослушка на скролл

window.addEventListener('scroll', () => {

    lastScroll = window.scrollY;

    if (!ticking) {

        window.requestAnimationFrame(() => {

            updateParallax(lastScroll);

            ticking = false;

        });

        ticking = true; } });

updateParallax(0);});

```

ПРИЛОЖЕНИЕ 3.

Настройка маршрутов для GET-, POST-, PUT-запросов

```
const select_query = 'SELECT t.title as test_title, q.question AS question_text,  
q.correct_ans AS correct_answer, GROUP_CONCAT(v.text ORDER BY v.id) AS  
options FROM tests t JOIN questions q ON t.id = q.test_id' +  
"  
" JOIN variants v ON q.id = v.question_id GROUP BY t.title, q.question,  
q.correct_ans"
```

```
// получить json-файл базы данных тестов
```

```
app.get('/tests', (request, response) => {  
  pool.query(select_query, (error, result) => {  
    if (error) throw error;  
    response.send(result);  
  });  
});
```

```
// получить json-файл рейтинга
```

```
app.get('/rating', (request, response) => {  
  pool.query("SELECT * FROM rating", (error, result) => {  
    if (error) throw error;  
    response.send(result);  
  });  
});
```

```
// добавить нового пользователя в рейтинг
```

```
app.post('/rating', (request, response) => {  
  pool.query('INSERT INTO rating SET ?', request.body, (error, result) => {  
    if (error) throw error;  
    response.status(201).send(`User added with ID: ${result.insertId}`);  
  });  
});
```

```
// обновить счет существующего пользователя в рейтинге
```

```
app.put('/rating/:name', (request, response) => {  
  const name = request.params.name;  
  pool.query('UPDATE rating SET ? WHERE name = ?', [request.body, name], (error,  
result) => {  
    if (error) throw error;  
    response.send('User updated successfully.');  });  
});
```