

Система автоматизированного тестирования знаний на основе HTML-форм с динамической генерацией вопросов

Екимова Яна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматриваются методика создания, разработка и описание функциональной системы автоматизированного тестирования знаний, построенной на основе HTML-форм с возможностью динамической генерации вопросов. Представлена практическая реализация клиентской части веб-приложения с использованием современных веб-технологий (HTML5, CSS3, JavaScript), обеспечивающая интерактивное взаимодействие с пользователем, корректную проверку ответов, отображение результатов и пояснений, а также возможность повторного прохождения теста. Материал предназначен для преподавателей и студентов, заинтересованных в применении современных технологий в обучении и стремящихся сделать процесс образовательного взаимодействия более эффективным и увлекательным.

Ключевые слова: автоматизированное тестирование, HTML-формы, динамическая генерация вопросов, HTML5, CSS, JavaScript.

Automated knowledge testing system based on HTML forms with dynamic question generation

Ekimova Yana Sergeevna

Sholom Aleichem Priamurskiy State University

Student

Abstract

The article discusses the methodology for creating, developing and describing a functional automated knowledge testing system based on HTML forms with the ability to dynamically generate questions. The practical implementation of the client part of the web application using modern web technologies (HTML5, CSS3, JavaScript) is presented, providing interactive interaction with the user, correct verification of answers, display of results and explanations, as well as the possibility of retesting the test. The material is intended for teachers and students interested in using modern technologies in teaching and striving to make the process of educational interaction more effective and exciting.

Keywords: automated testing, HTML forms, dynamic question generation, HTML5, CSS, JavaScript.

1 Введение

1.1 Актуальность

Современные образовательные технологии активно интегрируют цифровые решения для повышения эффективности обучения, объективности оценки знаний и персонализации образовательного процесса. Одним из ключевых инструментов в этой сфере являются системы автоматизированного тестирования, которые позволяют оперативно проверять уровень усвоения материала, минимизировать человеческий фактор при оценке и адаптировать задания под индивидуальные потребности учащихся.

В контексте цифровизации образования автоматизированное тестирование играет важную роль, обеспечивая масштабируемость, мгновенную обратную связь и возможность анализа результатов в режиме реального времени. Использование современных веб-технологий, таких как HTML5, CSS3 и JavaScript, позволяет создавать гибкие и интерактивные системы тестирования с динамической генерацией вопросов, что повышает вариативность заданий и снижает риски списывания.

1.2 Обзор исследований

В статье рассматривается организация подготовки педагогических работников к использованию технологий HTML5 для разработки электронных образовательных ресурсов в формате веб-приложений. Отмечена актуальность использования мультимедийных интерактивных электронных образовательных ресурсов в современной школе и обоснована эффективность разработки подобных электронных образовательных ресурсов именно в формате офлайновых или онлайновых веб-приложений. [1]. Показала процесс создания и использования мультимедийных интерактивных плакатов по информатике К.А. Цилинская. Представила опыт разработки мультимедийных интерактивных плакатов средствами языка HTML5 и Интернет-сервисов [2]. Я. А. Королюк, Н. Ю. Куликова показали опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайн-обучения школьников информатике [3]. В работе рассмотрели возможности языка разметки гипертекста HTML5 для создания макета электронного учебника Д.М. Размологов, А.С. Панина, П.В. Захаров. Привели разработанный макет, а также часть таблицы стилей для дизайна макета [4]. И.С. Шегаев, А.Г. Полозов разработали новейший политологический глоссарий: основные категории, понятия, термины [5].

1.3 Цель исследования

Целью исследования является создание системы автоматизированного тестирования знаний на основе HTML-форм с динамической генерацией вопросов с использованием языка HTML5.

2 Материалы и методы

Для создания приложения используется язык программирования HTML5. В качестве среды разработки используется текстовым редактор Notepad++, оптимизированным для языков программирования.

3 Результаты

Открываем Notepad++ и создаём файл index.html, для этого в меню выбираем файл → новый. Далее нажимаем сочетание клавиш "Ctrl+S" и в поле "Имя файла" вводим название файла "index" с расширением .html, сохраняем данный файл на рабочем столе.

Создадим основную структуру HTML-документа для веб-приложения «Система тестирования знаний», которая включает подключение метатегов для корректного отображения и адаптации на разных устройствах, заголовков страницы, стилизованные блоки для отображения названия, настроек теста (с выбором количества вопросов и кнопкой их генерации), индикатора прогресса прохождения теста, формы для ответов на вопросы и кнопки завершения тестирования, а также область для вывода результатов, что обеспечивает удобный и понятный интерфейс для пользователя.

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Система тестирования знаний</title>
  <style>

</style>
</head>
<body>
  <div class="container">
    <header>
      <h1>Система тестирования знаний</h1>
      <p>Проверьте свои знания в различных областях</p>
    </header>

    <div class="settings">
      <div>
        <label for="questionCount">Количество вопросов:</label>
        <input type="number" id="questionCount" min="1" max="10" value="5">
      </div>
      <button id="generateBtn">Сгенерировать тест</button>
    </div>

    <div class="progress-container">
      <div class="progress-header">
        <span>Прогресс тестирования</span>
        <span id="progressText">0/0</span>
      </div>
      <div class="progress-bar">
        <div class="progress-fill" id="progressBar"></div>
      </div>
    </div>

    <div class="test-wrapper">
```

```
<form id="testForm"></form>
</div>

<div class="buttons">
  <button id="submitBtn">Завершить тест</button>
</div>

<div id="resultSection" class="result-section"></div>
</div>

<script>

</script>
</body>
</html>
```

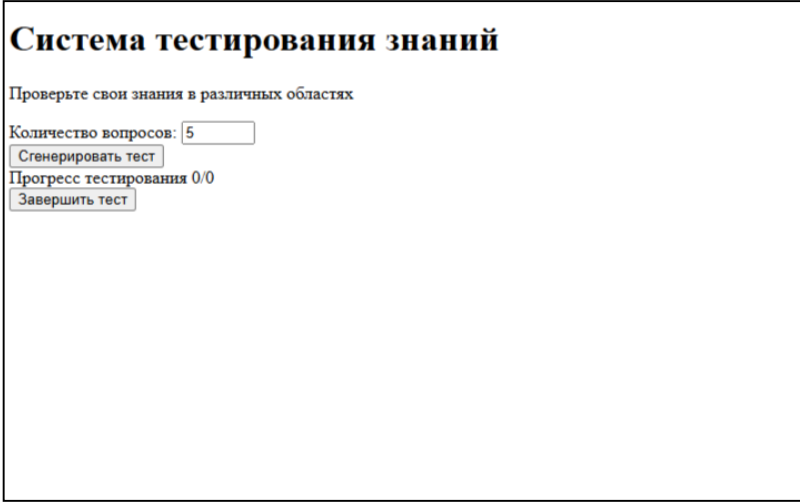


Рисунок 1. Основная структура HTML-документа

Далее добавим блок CSS-стилей, который используется для оформления веб-страницы с интерфейсом тестирования: он задаёт общий стиль страницы с градиентным фоном и центрированным контейнером, где глобальные стили сбрасывают отступы и устанавливают единую систему измерений и шрифт; контейнер имеет фиксированную ширину, закруглённые углы, тень и содержит внутри себя секции заголовка, настроек теста с полями ввода и кнопкой, индикатора прогресса, блока вопросов с вариантами ответов, кнопки отправки результатов и секции отображения итогов тестирования. Все элементы имеют продуманное оформление — от стилизованных заголовков и радио-кнопок до анимаций при наведении и фокусе, а также цветовых акцентов и теней для улучшения пользовательского опыта. В адаптивной части стилей предусмотрено изменение расположения элементов для экранов меньше 768 пикселей, что делает интерфейс удобным как на десктопах, так и на мобильных устройствах.

Данный код пропишем на следующей строчке после тега `<title>Интерактивная временная шкала</title>` в секции `<head>` HTML-документа.

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body {
  background: linear-gradient(135deg, #6a11cb 0%, #2575fc 100%);
  color: #333;
  min-height: 100vh;
  padding: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
}

.container {
  width: 100%;
  max-width: 900px;
  background: white;
  border-radius: 15px;
  box-shadow: 0 15px 30px rgba(0, 0, 0, 0.2);
  overflow: hidden;
}

header {
  background: #4361ee;
  color: white;
  padding: 25px;
  text-align: center;
}

header h1 {
  font-size: 2.2rem;
  margin-bottom: 10px;
}

.settings {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 20px;
  background: #f8f9fa;
  border-bottom: 1px solid #dee2e6;
}

.settings label {
  font-weight: 600;
  margin-right: 10px;
}

.settings input {
  padding: 10px;
  border: 1px solid #ced4da;
  border-radius: 8px;
  width: 70px;
  text-align: center;
  font-weight: bold;
  font-size: 1rem;
}
```

```
.settings button {
  background: #3a0ca3;
  color: white;
  border: none;
  padding: 12px 25px;
  border-radius: 8px;
  cursor: pointer;
  font-weight: 600;
  transition: all 0.3s;
}

.settings button:hover {
  background: #2a0a8a;
  transform: translateY(-2px);
}

.progress-container {
  padding: 15px 20px;
  background: white;
  border-bottom: 1px solid #dee2e6;
}

.progress-header {
  display: flex;
  justify-content: space-between;
  margin-bottom: 10px;
  font-weight: 600;
  color: #495057;
}

.progress-bar {
  height: 12px;
  background: #e9ecef;
  border-radius: 10px;
  overflow: hidden;
}

.progress-fill {
  height: 100%;
  background: #4cc9f0;
  border-radius: 10px;
  transition: width 0.5s ease;
  width: 0%;
}

.test-wrapper {
  padding: 20px;
  max-height: 60vh;
  overflow-y: auto;
}

.question {
  background: white;
  padding: 25px;
  margin-bottom: 25px;
  border-radius: 10px;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.05);
  border-left: 4px solid #4361ee;
}

.question h3 {
  margin-bottom: 20px;
}
```

```
        color: #3a0ca3;
        display: flex;
        align-items: center;
        font-size: 1.3rem;
    }

    .question h3 span {
        display: inline-flex;
        align-items: center;
        justify-content: center;
        width: 35px;
        height: 35px;
        background: #4361ee;
        color: white;
        border-radius: 50%;
        margin-right: 15px;
        font-size: 1.1rem;
    }

    .options {
        display: grid;
        grid-template-columns: 1fr;
        gap: 15px;
        margin: 20px 0;
    }

    .option {
        display: flex;
        align-items: center;
        padding: 15px;
        border: 2px solid #e9ecef;
        border-radius: 10px;
        cursor: pointer;
        transition: all 0.2s;
    }

    .option:hover {
        border-color: #4361ee;
        background-color: #f0f4ff;
        transform: translateY(-2px);
    }

    .option input {
        margin-right: 15px;
        width: 20px;
        height: 20px;
        cursor: pointer;
    }

    .option label {
        cursor: pointer;
        font-size: 1.05rem;
        flex-grow: 1;
    }

    .number-input {
        padding: 15px;
        width: 100%;
        border: 2px solid #e9ecef;
        border-radius: 10px;
        font-size: 1.1rem;
        transition: all 0.2s;
    }
```

```
.number-input:focus {
  border-color: #4361ee;
  outline: none;
  box-shadow: 0 0 0 3px rgba(67, 97, 238, 0.2);
}

.question-meta {
  display: flex;
  gap: 20px;
  margin-top: 15px;
  color: #6c757d;
  font-size: 0.95rem;
}

.buttons {
  text-align: center;
  padding: 20px;
  background: #f8f9fa;
  border-top: 1px solid #dee2e6;
}

#submitBtn {
  background: #4361ee;
  color: white;
  border: none;
  padding: 16px 50px;
  border-radius: 30px;
  cursor: pointer;
  font-size: 1.2rem;
  font-weight: 600;
  transition: all 0.3s;
  box-shadow: 0 5px 15px rgba(67, 97, 238, 0.4);
}

#submitBtn:hover {
  background: #3a0ca3;
  transform: translateY(-3px);
  box-shadow: 0 8px 20px rgba(67, 97, 238, 0.6);
}

.result-section {
  padding: 30px;
  background: #f8f9fa;
  display: none;
}

.result-header {
  text-align: center;
  margin-bottom: 30px;
}

.result-header h2 {
  color: #3a0ca3;
  margin-bottom: 15px;
}

.score {
  font-size: 3rem;
  font-weight: 800;
  color: #4361ee;
  margin: 15px 0;
}
```

```
.result-item {
  margin-bottom: 30px;
  padding: 25px;
  border-radius: 10px;
  background: white;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.05);
}

.correct {
  border-left: 5px solid #2ecc71;
}

.incorrect {
  border-left: 5px solid #e74c3c;
}

.answer-info {
  display: flex;
  gap: 20px;
  margin-bottom: 20px;
}

.answer-box {
  flex: 1;
  padding: 20px;
  border-radius: 8px;
  background: #f0f4ff;
}

.answer-box h5 {
  font-size: 1rem;
  color: #6c757d;
  margin-top: 0;
  margin-bottom: 15px;
}

.explanation {
  padding: 20px;
  background: #e8f4fd;
  border-radius: 8px;
  margin-top: 20px;
}

.retry-btn {
  background: #3a0ca3;
  color: white;
  border: none;
  padding: 16px 50px;
  border-radius: 30px;
  cursor: pointer;
  font-size: 1.2rem;
  font-weight: 600;
  display: block;
  width: 100%;
  margin-top: 30px;
  transition: all 0.3s;
  box-shadow: 0 5px 15px rgba(58, 12, 163, 0.4);
}

.retry-btn:hover {
  background: #2a0a8a;
  transform: translateY(-3px);
}
```

```

    box-shadow: 0 8px 20px rgba(58, 12, 163, 0.6);
  }

  @media (max-width: 768px) {
    .settings {
      flex-direction: column;
      gap: 15px;
    }

    .answer-info {
      flex-direction: column;
    }

    .question {
      padding: 20px;
    }

    header h1 {
      font-size: 1.8rem;
    }
  }
}

```

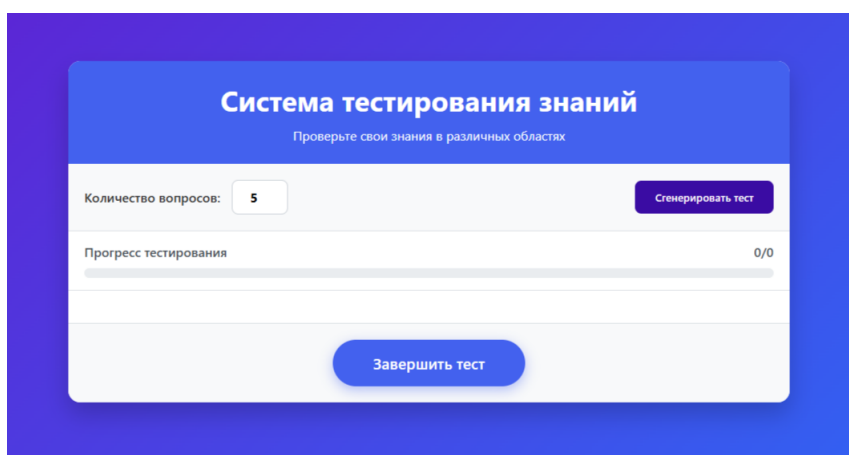


Рисунок 2. Добавление CSS-стилей

Далее пропишем код, который определяет массив вопросов `questionBank` с разными типами заданий — одиночный выбор (`radio`), множественный выбор (`checkbox`) и численный ответ (`number`), где каждый вопрос содержит текст, варианты ответов (если применимо), правильный ответ, объяснение, категорию и уровень сложности; также инициализируются переменные `currentQuestions` для хранения активных вопросов, выбираемых из банка, и `userAnswers` — объект для записи ответов пользователя во время прохождения теста.

Вставьте данный код после закрывающего тега `</div>` и после открывающего тега `<script>`:

```

const questionBank = [
  {
    id: 1,
    type: 'radio',
    question: "Что такое сложность алгоритма?",
    options: [
      "Время написания кода",
      "Объем используемой памяти",

```

```
"Зависимость времени работы от размера входных данных",
"Количество строк кода"
],
answer: "Зависимость времени работы от размера входных данных",
explanation: "Сложность алгоритма измеряет, как время выполнения или объем памяти
алгоритма растут с увеличением размера входных данных.",
category: "Программирование",
difficulty: "medium"
},
{
  id: 2,
  type: 'number',
  question: "Сколько сравнений потребуется для бинарного поиска в массиве из 1024
элементов?",
  answer: 10,
  explanation: "Бинарный поиск имеет сложность  $O(\log n)$ .  $\log_2(1024) = 10$ .",
  category: "Программирование",
  difficulty: "hard"
},
{
  id: 3,
  type: 'checkbox',
  question: "Какие из перечисленных методов относятся к алгоритмам сортировки?",
  options: [
    "Быстрая сортировка",
    "Двоичный поиск",
    "Сортировка слиянием",
    "Обход в глубину"
  ],
  answer: ["Быстрая сортировка", "Сортировка слиянием"],
  explanation: "Быстрая сортировка и сортировка слиянием — алгоритмы сортировки.
Двоичный поиск — алгоритм поиска, обход в глубину — алгоритм обхода графов.",
  category: "Программирование",
  difficulty: "easy"
},
{
  id: 4,
  type: 'radio',
  question: "Какой тег используется для создания навигации в HTML5?",
  options: [
    "<div>",
    "<nav>",
    "<menu>",
    "<header>"
  ],
  answer: "<nav>",
  explanation: "Тег <nav> предназначен для создания навигационных блоков на веб-странице
согласно спецификации HTML5.",
  category: "Веб-технологии",
  difficulty: "easy"
},
{
  id: 5,
  type: 'radio',
  question: "Что такое HTTP протокол?",
  options: [
    "Протокол передачи файлов",
    "Протокол передачи гипертекста",
    "Протокол безопасного соединения",
    "Протокол электронной почты"
  ],
  answer: "Протокол передачи гипертекста",
```

```
        explanation: "HTTP (HyperText Transfer Protocol) — протокол прикладного уровня для
передачи гипертекстовых документов.",
        category: "Веб-технологии",
        difficulty: "easy"
    }
];

let currentQuestions = [];
let userAnswers = {};
```

Теперь пропишем код, который реализует функцию `escapeHtml` для безопасного экранирования специальных HTML-символов, заменяя такие символы, как `&`, `<`, `>`, `"` и `'`, на соответствующие HTML-сущности, чтобы предотвратить выполнение потенциально опасного кода при выводе пользовательского ввода на страницу, что особенно важно для защиты от XSS-атак.

Вставьте данный код в продолжение тега `<script>`:

```
// Функция для экранирования HTML
function escapeHtml(unsafe) {
    if (unsafe === undefined || unsafe === null) return "";
    return unsafe.toString()
        .replace(/&/g, "&amp;")
        .replace(/</g, "&lt;")
        .replace(/>/g, "&gt;")
        .replace(/"/g, "&quot;")
        .replace(/'/g, "&#039;");
}
```

Далее займемся функцией `generateTest`, которая отвечает за генерацию теста на основе указанного пользователем количества вопросов, очищает форму теста, выбирает случайные вопросы из банка, перемешивая их и ограничивая нужным количеством, после чего динамически создаёт HTML-элементы для каждого вопроса с учётом его типа (`radio`, `checkbox` или `number`), применяя экранирование через функцию `escapeHtml` для безопасности; также добавляются метаданные вопроса — категория и сложность, отображается кнопка отправки, скрывается секция результатов, вызывается функция обновления прогресса и устанавливаются обработчики событий для взаимодействия с элементами теста.

Вставьте данный код в продолжение тега `<script>`:

```
// Генерация теста
function generateTest() {
    const count = parseInt(document.getElementById('questionCount').value) || 5;
    const testForm = document.getElementById('testForm');
    testForm.innerHTML = "";
    userAnswers = {};

    // Выбираем случайные вопросы
    currentQuestions = [...questionBank]
        .sort(() => Math.random() - 0.5)
        .slice(0, Math.min(count, questionBank.length));

    // Генерируем HTML для каждого вопроса
    currentQuestions.forEach((q, index) => {
```

```

const questionDiv = document.createElement('div');
questionDiv.className = 'question';
questionDiv.dataset.id = q.id;

let optionsHtml = "";
if (q.type === 'radio') {
  q.options.forEach((opt, optIndex) => {
    optionsHtml += `
      <div class="option">
        <input type="radio" name="q${q.id}" value="${escapeHtml(opt)}"
          id="q${q.id}_${optIndex}">
        <label for="q${q.id}_${optIndex}">${escapeHtml(opt)}</label>
      </div>
    `;
  });
} else if (q.type === 'checkbox') {
  q.options.forEach((opt, optIndex) => {
    optionsHtml += `
      <div class="option">
        <input type="checkbox" name="q${q.id}" value="${escapeHtml(opt)}"
          id="q${q.id}_${optIndex}">
        <label for="q${q.id}_${optIndex}">${escapeHtml(opt)}</label>
      </div>
    `;
  });
} else if (q.type === 'number') {
  optionsHtml = `
    <input type="number" class="number-input"
      placeholder="Введите ответ">
  `;
}

questionDiv.innerHTML = `
  <h3><span>${index + 1}</span> ${escapeHtml(q.question)}</h3>
  <div class="options">${optionsHtml}</div>
  <div class="question-meta">
    <span><strong>Категория:</strong> ${escapeHtml(q.category)}</span>
    <span><strong>Сложность:</strong> ${escapeHtml(q.difficulty)}</span>
  </div>
`;

testForm.appendChild(questionDiv);
});

document.getElementById('submitBtn').style.display = 'block';
document.getElementById('resultSection').style.display = 'none';
updateProgress();

// Добавляем обработчики событий
addEventListener();
}

```

Теперь пропишем код, который реализует логику взаимодействия с тестом: добавляет обработчики событий на ввод ответов, отслеживая изменения в радио-кнопках, чекбоксах и числовых полях и сохраняя их в объект `userAnswers`, а также обновляя индикатор прогресса; функция `checkAnswers` при отправке формы проверяет все ответы, используя функцию `compareAnswers`, которая корректно сравнивает значения для каждого типа вопроса — как по точному совпадению для радио, так и по полному

соответствию массива для чекбоксов и чисел; результаты выводятся в стилизованной секции `showResults`, где отображаются все вопросы, правильные и пользовательские ответы, объяснения и итоговый счёт, а также кнопка для повторного прохождения теста; дополнительно настроены слушатели событий для кнопок генерации и отправки теста, а при загрузке страницы автоматически создаётся первый тест.

Вставьте данный код в продолжение тега `<script>`:

```
// Добавление обработчиков событий
function addEventListeners() {
  const testForm = document.getElementById('testForm');

  // Обработчики для радио-кнопок
  testForm.querySelectorAll('input[type="radio"]').forEach(radio => {
    radio.addEventListener('change', function() {
      const questionId = this.name.replace('q', '');
      userAnswers[questionId] = [this.value];
      updateProgress();
    });
  });

  // Обработчики для чекбоксов
  testForm.querySelectorAll('input[type="checkbox"]').forEach(checkbox => {
    checkbox.addEventListener('change', function() {
      const questionId = this.name.replace('q', '');

      if (!userAnswers[questionId]) {
        userAnswers[questionId] = [];
      }

      if (this.checked) {
        userAnswers[questionId].push(this.value);
      } else {
        userAnswers[questionId] = userAnswers[questionId].filter(v => v !== this.value);
      }

      updateProgress();
    });
  });

  // Обработчики для числовых полей
  testForm.querySelectorAll('input[type="number"]').forEach(input => {
    input.addEventListener('input', function() {
      const questionId = this.closest('.question').dataset.id;
      userAnswers[questionId] = [this.value];
      updateProgress();
    });
  });
}

// Обновление прогресса
function updateProgress() {
  const total = currentQuestions.length;
  let answered = 0;

  currentQuestions.forEach(q => {
    if (userAnswers[q.id] && userAnswers[q.id].length > 0) {
      answered++;
    }
  })
}
```

```
});

document.getElementById('progressText').textContent = `${answered}/${total}`;
document.getElementById('progressBar').style.width = `${(answered/total)*100}%`;
}

// Проверка ответов
function checkAnswers(event) {
    event.preventDefault();

    const results = [];
    let score = 0;

    currentQuestions.forEach(q => {
        const userAnswer = userAnswers[q.id] || [];
        const isCorrect = compareAnswers(q, userAnswer);

        if (isCorrect) score++;

        results.push({
            question: q.question,
            userAnswer: userAnswer.length ? userAnswer.join(', ') : 'Нет ответа',
            correctAnswer: Array.isArray(q.answer) ? q.answer.join(', ') : q.answer,
            isCorrect: isCorrect,
            explanation: q.explanation,
            category: q.category,
            difficulty: q.difficulty
        });
    });

    showResults(results, score, currentQuestions.length);
}

// Сравнение ответов
function compareAnswers(question, userAnswer) {
    if (question.type === 'checkbox') {
        if (userAnswer.length !== question.answer.length) return false;
        return question.answer.every(ans => userAnswer.includes(ans.toString()));
    }

    if (question.type === 'number') {
        return parseFloat(userAnswer[0]) === parseFloat(question.answer);
    }

    return userAnswer[0] === question.answer.toString();
}

// Показ результатов
function showResults(results, score, total) {
    const resultSection = document.getElementById('resultSection');
    resultSection.style.display = 'block';

    let html = `
        <div class="result-header">
            <h2>Результаты тестирования</h2>
            <div class="score">${score}/${total}</div>
            <p>${score >= total/2 ? 'Отличный результат!' : 'Попробуйте еще раз!'}</p>
        </div>
    `;

    results.forEach((res, index) => {
        html += `
            <div class="result-item ${res.isCorrect ? 'correct' : 'incorrect'}">

```

```
<h4>Вопрос ${index + 1}: ${escapeHtml(res.question)}</h4>
<div class="question-meta">
  <span><strong>Категория:</strong> ${escapeHtml(res.category)}</span>
  <span><strong>Сложность:</strong> ${escapeHtml(res.difficulty)}</span>
</div>
<div class="answer-info">
  <div class="answer-box">
    <h5>Ваш ответ</h5>
    <div>${escapeHtml(res.userAnswer)}</div>
  </div>
  <div class="answer-box">
    <h5>Правильный ответ</h5>
    <div>${escapeHtml(res.correctAnswer)}</div>
  </div>
</div>
<div class="explanation">
  <strong>Объяснение:</strong> ${escapeHtml(res.explanation)}
</div>
</div>
`);
});

html += `
  <button class="retry-btn" onclick="generateTest()">
    Пройти тест снова
  </button>
`;

resultSection.innerHTML = html;
document.getElementById('submitBtn').style.display = 'none';

// Прокрутка к результатам
resultSection.scrollIntoView({ behavior: 'smooth' });
}

// Инициализация
document.getElementById('generateBtn').addEventListener('click', generateTest);
document.getElementById('submitBtn').addEventListener('click', checkAnswers);

// Генерация теста при загрузке страницы
window.onload = generateTest;
```

Система тестирования знаний

Проверьте свои знания в различных областях

Количество вопросов: 5

Сгенерировать тест

Прогресс тестирования 0/5

1

Какие из перечисленных методов относятся к алгоритмам сортировки?

☐ Быстрая сортировка

☐ Двоичный поиск

☐ Сортировка слиянием

☐ Обход в глубину

Категория: Программирование Сложность: easy

2

Какой тег используется для создания навигации в HTML5?

☐ <div>

Завершить тест

Рисунок 3. Тест

Система тестирования знаний

Проверьте свои знания в различных областях

Количество вопросов: 5

Сгенерировать тест

Прогресс тестирования 5/5

☐ Объем используемой памяти

☐ Зависимость времени работы от размера входных данных

☒ Количество строк кода

Категория: Программирование Сложность: medium

5

Сколько сравнений потребуется для бинарного поиска в массиве из 1024 элементов?

10

Категория: Программирование Сложность: hard

Завершить тест

Рисунок 4. Прохождение теста

Результаты тестирования

1/5

Попробуйте еще раз!

Вопрос 1: Какие из перечисленных методов относятся к алгоритмам сортировки?

Категория: Программирование Сложность: easy

Ваш ответ

Двоичный поиск

Правильный ответ

Быстрая сортировка, Сортировка слиянием

Объяснение: Быстрая сортировка и сортировка слиянием — алгоритмы сортировки. Двоичный поиск — алгоритм поиска, обход в глубину — алгоритм обхода графов.

Вопрос 2: Какой тег используется для создания навигации в HTML5?

Категория: Веб-технологии Сложность: easy

Ваш ответ

<div>

Правильный ответ

<nav>

Объяснение: Тег <nav> предназначен для создания навигационных блоков на веб-странице согласно спецификации HTML5.

Вопрос 3: Что такое HTTP протокол?

Категория: Веб-технологии Сложность: easy

Рисунок 5. Результаты теста

Вопрос 4: Что такое сложность алгоритма?

Категория: Программирование Сложность: medium

Ваш ответ

Количество строк кода

Правильный ответ

Зависимость времени работы от размера входных данных

Объяснение: Сложность алгоритма измеряет, как время выполнения или объем памяти алгоритма растут с увеличением размера входных данных.

Вопрос 5: Сколько сравнений потребуется для бинарного поиска в массиве из 1024 элементов?

Категория: Программирование Сложность: hard

Ваш ответ

10

Правильный ответ

10

Объяснение: Бинарный поиск имеет сложность $O(\log n)$. $\log_2(1024) = 10$.

Пройти тест снова

Рисунок 6. Результаты теста

4 Выводы

В данной работе подробно рассмотрена методика создания, разработки и описания функциональной системы автоматизированного тестирования знаний, построенной на основе HTML-форм с возможностью динамической

генерации вопросов. Представлена практическая реализация клиентской части веб-приложения с использованием современных веб-технологий (HTML5, CSS3, JavaScript), обеспечивающая интерактивное взаимодействие с пользователем, корректную проверку ответов, отображение результатов и пояснений, а также возможность повторного прохождения теста. Предложенное решение ориентировано на преподавателей и студентов, заинтересованных в применении современных веб-технологий для повышения эффективности образовательного процесса и превращения обучения в более увлекательный и интерактивный опыт.

Библиографический список

1. Бакулевская С. С. Подготовка педагогических работников к использованию технологий html5 для разработки электронных образовательных ресурсов //Информатика и образование. 2019. №. 5. С. 32-40.
2. Цилинская К.А. Создание и использование мультимедийных интерактивных плакатов по информатике //Будущее науки-2019. 2019. С. 340-344.
3. Королюк Я. А., Куликова Н. Ю. Опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайн-обучения школьников информатике //XXV Всероссийская студенческая научно-практическая конференция Нижневартковского государственного университета. 2023. С. 48-55.
4. Размологов Д. М., Панина А. С., Захаров П. В. Использование html5 для разработки электронного учебника //Современные тенденции развития науки и технологий. 2016. №. 4-4. С. 92-94.
5. Шегаев И. С., Полозов А. Г. Новейший политологический глоссарий: основные категории, понятия, термины. Общество с ограниченной ответственностью Издательство Перо, 2015. С. 242-242.

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Система тестирования знаний</title>
</style>
  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  }

  body {
    background: linear-gradient(135deg, #6a11cb 0%, #2575fc 100%);
    color: #333;
    min-height: 100vh;
    padding: 20px;
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .container {
    width: 100%;
    max-width: 900px;
    background: white;
    border-radius: 15px;
    box-shadow: 0 15px 30px rgba(0, 0, 0, 0.2);
    overflow: hidden;
  }

  header {
    background: #4361ee;
    color: white;
    padding: 25px;
    text-align: center;
  }

  header h1 {
    font-size: 2.2rem;
    margin-bottom: 10px;
  }

  .settings {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px;
    background: #f8f9fa;
    border-bottom: 1px solid #dee2e6;
  }

  .settings label {
    font-weight: 600;
    margin-right: 10px;
  }

  .settings input {
    padding: 10px;
```

```
border: 1px solid #ced4da;
border-radius: 8px;
width: 70px;
text-align: center;
font-weight: bold;
font-size: 1rem;
}

.settings button {
background: #3a0ca3;
color: white;
border: none;
padding: 12px 25px;
border-radius: 8px;
cursor: pointer;
font-weight: 600;
transition: all 0.3s;
}

.settings button:hover {
background: #2a0a8a;
transform: translateY(-2px);
}

.progress-container {
padding: 15px 20px;
background: white;
border-bottom: 1px solid #dee2e6;
}

.progress-header {
display: flex;
justify-content: space-between;
margin-bottom: 10px;
font-weight: 600;
color: #495057;
}

.progress-bar {
height: 12px;
background: #e9ecef;
border-radius: 10px;
overflow: hidden;
}

.progress-fill {
height: 100%;
background: #4cc9f0;
border-radius: 10px;
transition: width 0.5s ease;
width: 0%;
}

.test-wrapper {
padding: 20px;
max-height: 60vh;
overflow-y: auto;
}

.question {
background: white;
padding: 25px;
margin-bottom: 25px;
```

```
border-radius: 10px;
box-shadow: 0 5px 15px rgba(0, 0, 0, 0.05);
border-left: 4px solid #4361ee;
}
```

```
.question h3 {
margin-bottom: 20px;
color: #3a0ca3;
display: flex;
align-items: center;
font-size: 1.3rem;
}
```

```
.question h3 span {
display: inline-flex;
align-items: center;
justify-content: center;
width: 35px;
height: 35px;
background: #4361ee;
color: white;
border-radius: 50%;
margin-right: 15px;
font-size: 1.1rem;
}
```

```
.options {
display: grid;
grid-template-columns: 1fr;
gap: 15px;
margin: 20px 0;
}
```

```
.option {
display: flex;
align-items: center;
padding: 15px;
border: 2px solid #e9ecef;
border-radius: 10px;
cursor: pointer;
transition: all 0.2s;
}
```

```
.option:hover {
border-color: #4361ee;
background-color: #f0f4ff;
transform: translateY(-2px);
}
```

```
.option input {
margin-right: 15px;
width: 20px;
height: 20px;
cursor: pointer;
}
```

```
.option label {
cursor: pointer;
font-size: 1.05rem;
flex-grow: 1;
}
```

```
.number-input {
```

```
padding: 15px;
width: 100%;
border: 2px solid #e9ecef;
border-radius: 10px;
font-size: 1.1rem;
transition: all 0.2s;
}

.number-input:focus {
border-color: #4361ee;
outline: none;
box-shadow: 0 0 3px rgba(67, 97, 238, 0.2);
}

.question-meta {
display: flex;
gap: 20px;
margin-top: 15px;
color: #6c757d;
font-size: 0.95rem;
}

.buttons {
text-align: center;
padding: 20px;
background: #f8f9fa;
border-top: 1px solid #dee2e6;
}

#submitBtn {
background: #4361ee;
color: white;
border: none;
padding: 16px 50px;
border-radius: 30px;
cursor: pointer;
font-size: 1.2rem;
font-weight: 600;
transition: all 0.3s;
box-shadow: 0 5px 15px rgba(67, 97, 238, 0.4);
}

#submitBtn:hover {
background: #3a0ca3;
transform: translateY(-3px);
box-shadow: 0 8px 20px rgba(67, 97, 238, 0.6);
}

.result-section {
padding: 30px;
background: #f8f9fa;
display: none;
}

.result-header {
text-align: center;
margin-bottom: 30px;
}

.result-header h2 {
color: #3a0ca3;
margin-bottom: 15px;
}
```

```
.score {
  font-size: 3rem;
  font-weight: 800;
  color: #4361ee;
  margin: 15px 0;
}

.result-item {
  margin-bottom: 30px;
  padding: 25px;
  border-radius: 10px;
  background: white;
  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.05);
}

.correct {
  border-left: 5px solid #2ecc71;
}

.incorrect {
  border-left: 5px solid #e74c3c;
}

.answer-info {
  display: flex;
  gap: 20px;
  margin-bottom: 20px;
}

.answer-box {
  flex: 1;
  padding: 20px;
  border-radius: 8px;
  background: #f0f4ff;
}

.answer-box h5 {
  font-size: 1rem;
  color: #6c757d;
  margin-top: 0;
  margin-bottom: 15px;
}

.explanation {
  padding: 20px;
  background: #e8f4fd;
  border-radius: 8px;
  margin-top: 20px;
}

.retry-btn {
  background: #3a0ca3;
  color: white;
  border: none;
  padding: 16px 50px;
  border-radius: 30px;
  cursor: pointer;
  font-size: 1.2rem;
  font-weight: 600;
  display: block;
  width: 100%;
  margin-top: 30px;
}
```

```
        transition: all 0.3s;
        box-shadow: 0 5px 15px rgba(58, 12, 163, 0.4);
    }

    .retry-btn:hover {
        background: #2a0a8a;
        transform: translateY(-3px);
        box-shadow: 0 8px 20px rgba(58, 12, 163, 0.6);
    }

    @media (max-width: 768px) {
        .settings {
            flex-direction: column;
            gap: 15px;
        }

        .answer-info {
            flex-direction: column;
        }

        .question {
            padding: 20px;
        }

        header h1 {
            font-size: 1.8rem;
        }
    }
</style>
</head>
<body>
    <div class="container">
        <header>
            <h1>Система тестирования знаний</h1>
            <p>Проверьте свои знания в различных областях</p>
        </header>

        <div class="settings">
            <div>
                <label for="questionCount">Количество вопросов:</label>
                <input type="number" id="questionCount" min="1" max="10" value="5">
            </div>
            <button id="generateBtn">Сгенерировать тест</button>
        </div>

        <div class="progress-container">
            <div class="progress-header">
                <span>Прогресс тестирования</span>
                <span id="progressText">0/0</span>
            </div>
            <div class="progress-bar">
                <div class="progress-fill" id="progressBar"></div>
            </div>
        </div>

        <div class="test-wrapper">
            <form id="testForm"></form>
        </div>

        <div class="buttons">
            <button id="submitBtn">Завершить тест</button>
        </div>
    </div>
</body>
</html>
```

```
<div id="resultSection" class="result-section"></div>
</div>

<script>
const questionBank = [
  {
    id: 1,
    type: 'radio',
    question: "Что такое сложность алгоритма?",
    options: [
      "Время написания кода",
      "Объем используемой памяти",
      "Зависимость времени работы от размера входных данных",
      "Количество строк кода"
    ],
    answer: "Зависимость времени работы от размера входных данных",
    explanation: "Сложность алгоритма измеряет, как время выполнения или объем памяти алгоритма растут с увеличением размера входных данных.",
    category: "Программирование",
    difficulty: "medium"
  },
  {
    id: 2,
    type: 'number',
    question: "Сколько сравнений потребуется для бинарного поиска в массиве из 1024 элементов?",
    answer: 10,
    explanation: "Бинарный поиск имеет сложность  $O(\log n)$ .  $\log_2(1024) = 10$ .",
    category: "Программирование",
    difficulty: "hard"
  },
  {
    id: 3,
    type: 'checkbox',
    question: "Какие из перечисленных методов относятся к алгоритмам сортировки?",
    options: [
      "Быстрая сортировка",
      "Двоичный поиск",
      "Сортировка слиянием",
      "Обход в глубину"
    ],
    answer: ["Быстрая сортировка", "Сортировка слиянием"],
    explanation: "Быстрая сортировка и сортировка слиянием — алгоритмы сортировки. Двоичный поиск — алгоритм поиска, обход в глубину — алгоритм обхода графов.",
    category: "Программирование",
    difficulty: "easy"
  },
  {
    id: 4,
    type: 'radio',
    question: "Какой тег используется для создания навигации в HTML5?",
    options: [
      "<div>",
      "<nav>",
      "<menu>",
      "<header>"
    ],
    answer: "<nav>",
    explanation: "Тег <nav> предназначен для создания навигационных блоков на веб-странице согласно спецификации HTML5.",
    category: "Веб-технологии",
    difficulty: "easy"
  },
]
```

```

    {
      id: 5,
      type: 'radio',
      question: "Что такое HTTP протокол?",
      options: [
        "Протокол передачи файлов",
        "Протокол передачи гипертекста",
        "Протокол безопасного соединения",
        "Протокол электронной почты"
      ],
      answer: "Протокол передачи гипертекста",
      explanation: "HTTP (HyperText Transfer Protocol) — протокол прикладного уровня для
передачи гипертекстовых документов.",
      category: "Веб-технологии",
      difficulty: "easy"
    }
  ];

  let currentQuestions = [];
  let userAnswers = {};
  // Функция для экранирования HTML
  function escapeHtml(unsafe) {
    if (unsafe === undefined || unsafe === null) return "";
    return unsafe.toString()
      .replace(/&/g, "&amp;")
      .replace(/</g, "&lt;")
      .replace(/>/g, "&gt;")
      .replace(/"/g, "&quot;")
      .replace(/'/g, "&#039;");
  }
  // Генерация теста
  function generateTest() {
    const count = parseInt(document.getElementById('questionCount').value) || 5;
    const testForm = document.getElementById('testForm');
    testForm.innerHTML = "";
    userAnswers = {};

    // Выбираем случайные вопросы
    currentQuestions = [...questionBank]
      .sort(() => Math.random() - 0.5)
      .slice(0, Math.min(count, questionBank.length));

    // Генерируем HTML для каждого вопроса
    currentQuestions.forEach((q, index) => {
      const questionDiv = document.createElement('div');
      questionDiv.className = 'question';
      questionDiv.dataset.id = q.id;

      let optionsHtml = "";
      if (q.type === 'radio') {
        q.options.forEach((opt, optIndex) => {
          optionsHtml += `
            <div class="option">
              <input type="radio" name="q${q.id}" value="${escapeHtml(opt)}"
                id="q${q.id}_${optIndex}">
              <label for="q${q.id}_${optIndex}">${escapeHtml(opt)}</label>
            </div>
          `;
        });
      }
      } else if (q.type === 'checkbox') {
        q.options.forEach((opt, optIndex) => {
          optionsHtml += `
            <div class="option">

```

```

        <input type="checkbox" name="q${q.id}" value="${escapeHtml(opt)}"
            id="q${q.id}_${optIndex}">
        <label for="q${q.id}_${optIndex}">${escapeHtml(opt)}</label>
    </div>
    `;
    });
    } else if (q.type === 'number') {
        optionsHtml = `
            <input type="number" class="number-input"
                placeholder="Введите ответ">
        `;
    }

    questionDiv.innerHTML = `
        <h3><span>${index + 1}</span> ${escapeHtml(q.question)}</h3>
        <div class="options">${optionsHtml}</div>
        <div class="question-meta">
            <span><strong>Категория:</strong> ${escapeHtml(q.category)}</span>
            <span><strong>Сложность:</strong> ${escapeHtml(q.difficulty)}</span>
        </div>
    `;

    testForm.appendChild(questionDiv);
});

document.getElementById('submitBtn').style.display = 'block';
document.getElementById('resultSection').style.display = 'none';
updateProgress();

// Добавляем обработчики событий
addEventListener();
}

// Добавление обработчиков событий
function addEventListeners() {
    const testForm = document.getElementById('testForm');

    // Обработчики для радио-кнопок
    testForm.querySelectorAll('input[type="radio"]').forEach(radio => {
        radio.addEventListener('change', function() {
            const questionId = this.name.replace('q', '');
            userAnswers[questionId] = [this.value];
            updateProgress();
        });
    });

    // Обработчики для чекбоксов
    testForm.querySelectorAll('input[type="checkbox"]').forEach(checkbox => {
        checkbox.addEventListener('change', function() {
            const questionId = this.name.replace('q', '');

            if (!userAnswers[questionId]) {
                userAnswers[questionId] = [];
            }

            if (this.checked) {
                userAnswers[questionId].push(this.value);
            } else {
                userAnswers[questionId] = userAnswers[questionId].filter(v => v !== this.value);
            }

            updateProgress();
        });
    });
}

```

```
});

// Обработчики для числовых полей
testForm.querySelectorAll('input[type="number"]').forEach(input => {
  input.addEventListener('input', function() {
    const questionId = this.closest('.question').dataset.id;
    userAnswers[questionId] = [this.value];
    updateProgress();
  });
});

// Обновление прогресса
function updateProgress() {
  const total = currentQuestions.length;
  let answered = 0;

  currentQuestions.forEach(q => {
    if (userAnswers[q.id] && userAnswers[q.id].length > 0) {
      answered++;
    }
  });

  document.getElementById('progressText').textContent = `${answered}/${total}`;
  document.getElementById('progressBar').style.width = `${(answered/total)*100}%`;
}

// Проверка ответов
function checkAnswers(event) {
  event.preventDefault();

  const results = [];
  let score = 0;

  currentQuestions.forEach(q => {
    const userAnswer = userAnswers[q.id] || [];
    const isCorrect = compareAnswers(q, userAnswer);

    if (isCorrect) score++;

    results.push({
      question: q.question,
      userAnswer: userAnswer.length ? userAnswer.join(', ') : 'Нет ответа',
      correctAnswer: Array.isArray(q.answer) ? q.answer.join(', ') : q.answer,
      isCorrect: isCorrect,
      explanation: q.explanation,
      category: q.category,
      difficulty: q.difficulty
    });
  });

  showResults(results, score, currentQuestions.length);
}

// Сравнение ответов
function compareAnswers(question, userAnswer) {
  if (question.type === 'checkbox') {
    if (userAnswer.length !== question.answer.length) return false;
    return question.answer.every(ans => userAnswer.includes(ans.toString()));
  }

  if (question.type === 'number') {
    return parseFloat(userAnswer[0]) === parseFloat(question.answer);
  }
}
```

```
}

return userAnswer[0] === question.answer.toString();
}

// Показ результатов
function showResults(results, score, total) {
  const resultSection = document.getElementById('resultSection');
  resultSection.style.display = 'block';

  let html = `
    <div class="result-header">
      <h2>Результаты тестирования</h2>
      <div class="score">${score}/${total}</div>
      <p>${score >= total/2 ? 'Отличный результат!' : 'Попробуйте еще раз!'}</p>
    </div>
  `;

  results.forEach((res, index) => {
    html += `
      <div class="result-item ${res.isCorrect ? 'correct' : 'incorrect'}">
        <h4>Вопрос ${index + 1}: ${escapeHtml(res.question)}</h4>
        <div class="question-meta">
          <span><strong>Категория:</strong> ${escapeHtml(res.category)}</span>
          <span><strong>Сложность:</strong> ${escapeHtml(res.difficulty)}</span>
        </div>
        <div class="answer-info">
          <div class="answer-box">
            <h5>Ваш ответ</h5>
            <div>${escapeHtml(res.userAnswer)}</div>
          </div>
          <div class="answer-box">
            <h5>Правильный ответ</h5>
            <div>${escapeHtml(res.correctAnswer)}</div>
          </div>
        </div>
        <div class="explanation">
          <strong>Объяснение:</strong> ${escapeHtml(res.explanation)}
        </div>
      </div>
    `;
  });

  html += `
    <button class="retry-btn" onclick="generateTest()">
      Пройти тест снова
    </button>
  `;

  resultSection.innerHTML = html;
  document.getElementById('submitBtn').style.display = 'none';

  // Прокрутка к результатам
  resultSection.scrollIntoView({ behavior: 'smooth' });
}

// Инициализация
document.getElementById('generateBtn').addEventListener('click', generateTest);
document.getElementById('submitBtn').addEventListener('click', checkAnswers);

// Генерация теста при загрузке страницы
window.onload = generateTest;
</script>
```

</body>
</html>