СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc200058047)

[1 ОБЩАЯ ЧАСТЬ 4](#_Toc200058048)

[1.1 Цель разработки 4](#_Toc200058049)

[1.2 Анализ использования разработки 4](#_Toc200058050)

[1.3 Анализ методов решения 5](#_Toc200058051)

[1.4 Обзор средств программирования 8](#_Toc200058052)

[1.5 Описание языка программирования 21](#_Toc200058053)

# ВВЕДЕНИЕ

Цифровые технологии получили широкое распространение и значительно изменили структуру образовательного процесса. Современные студенты все чаще осваивают учебный материал вне аудиторий, взаимодействуя с преподавателями в онлайн-режиме, выполняя задания дистанционно и проходя тестирование через интернет. Увеличение популярности удалённого обучения обусловило рост потребности в удобных и функциональных образовательных платформах.

Для обучающихся, осваивающих новые профессии, важна возможность обучения в удобное время и с индивидуальным темпом. В этом случае актуальным становится создание онлайн-ресурсов, которые могут эффективно дополнять или заменять традиционные формы обучения.

Разрабатываемая платформа обеспечивает эффективное взаимодействие между студентами и преподавателями посредством интуитивно понятного интерфейса и необходимого функционала. Пользователи получают возможность изучать материалы и проходить тесты без необходимости личного присутствия, что способствует экономии времени преподавателей и повышает гибкость образовательного процесса.

Итоговая версия проекта представляет собой универсальное решение для дистанционного обучения, пригодное как для самостоятельного изучения и проверки знаний студентами, так и для автоматизации учебного процесса преподавателями – от загрузки материалов до анализа успеваемости.

## 1 ОБЩАЯ ЧАСТЬ

### 1.1 Цель разработки

Целью данной дипломной работы является разработка образовательного ресурса для изучения дисциплины «Архитектура компьютерных систем», обеспечивающего доступ к лекциям, практическим заданиям и автоматизированному тестированию.

В рамках работы поставлены следующие задачи: создать удобный пользовательский интерфейс для студентов и преподавателей; подготовить и интегрировать учебный контент, включающий лекции, практические материалы и тесты; реализовать систему автоматической проверки тестовых заданий с сохранением результатов; обеспечить функционал управления контентом для преподавателей.

### 1.2 Анализ использования разработки

Разрабатываемый образовательный ресурс предназначен для эффективной поддержки изучения дисциплины «Архитектура компьютерных систем». Платформа предоставляет доступ к структурированным учебным материалам, интерактивным заданиям и проверочным тестам, что способствует повышению качества образования.

Ресурс оснащён инструментами для преподавателей, обеспечивающими проверку результатов тестирования и добавление дополнительных материалов. Преподаватели могут управлять содержанием сайта, редактировать и обновлять информационные блоки, поддерживая актуальность и полноту учебного контента.

Программный продукт ориентирован на две основные категории пользователей:

Студенты: имеют возможность изучать лекции, выполнять практические задания и проходить тестирование; просматривать оценки и отслеживать прогресс; использовать платформу для самоподготовки и повторения материала.

Преподаватели: загружают и обновляют учебные материалы, создают и назначают задания; получают доступ к статистическим данным успеваемости студентов для анализа эффективности обучения.

Разработанная онлайн-платформа обеспечивает самостоятельное изучение дисциплины «Архитектура компьютерных систем» студентами, упрощает организацию учебного процесса для преподавателей и повышает оперативность управления контентом.

### 1.3 Анализ методов решения

Для создания образовательного ресурса, соответствующего поставленным целям, необходимо подобрать оптимальные технологии и инструменты. Выбор средств разработки определяется требованиями к функциональности, масштабом проекта и доступными ресурсами.

Рассматриваются следующие методы разработки:

1. Конструкторы сайтов.

Конструкторы сайтов (website builders) – это онлайн-платформы, позволяющие создавать веб-ресурсы через визуальные редакторы и готовые шаблоны без необходимости программирования. Используя интерфейс drag-and-drop, пользователи быстро и интуитивно размещают и настраивают элементы страницы (тексты, изображения, кнопки и формы).

Популярные примеры: Tilda, Wix, Squarespace.

Преимущества конструкторов включают простоту использования, отсутствие необходимости знания кода, быструю разработку благодаря готовым шаблонам, наличие встроенного хостинга, домена, SSL-сертификатов и базовых SEO-инструментов. Платформы регулярно обновляют программное обеспечение и обеспечивают базовую защиту сайта.

Однако конструкторы ограничены функционально: сложно реализовать уникальные решения, глубоко изменить код или архитектуру сайта невозможно. Часто используется избыточный код, а шаблонность может затруднять масштабирование. При росте бизнеса или трафика может потребоваться переход на более гибкие технологии, так как перенос сайта на другую платформу зачастую невозможен без полной переработки.

Конструкторы широко применяются для быстрого создания лендингов, портфолио, презентационных сайтов и небольших интернет-магазинов. Например, стартапы и малые предприятия используют Wix или Tilda для запуска первых сайтов без привлечения специалистов. Творческие профессионалы – дизайнеры, фотографы, копирайтеры – создают простые и привлекательные сайты для демонстрации работ. В качестве примера: небольшой магазин одежды может запустить сайт на Wix с каталогом товаров и встроенной системой оплаты, что позволяет начать онлайн-продажи с минимальными затратами.

2. Системы управления контентом (CMS).

Системы управления контентом (CMS) – это программные платформы для создания, редактирования и организации содержимого веб-сайтов, предоставляющие удобный интерфейс для работы с текстами, изображениями, видео и структурой сайта без глубоких технических знаний. В отличие от конструкторов, CMS обеспечивают более широкие возможности кастомизации и расширения функционала благодаря плагинам, модулям и темам оформления.

Примеры CMS: WordPress, Joomla, Drupal.

CMS широко применяются для создания сложных и масштабируемых проектов: корпоративных сайтов, новостных порталов, образовательных платформ и интернет-магазинов. Например, WordPress популярен для ведения блогов и новостных ресурсов благодаря простоте использования и большому количеству дополнений. Joomla отличается более сложной архитектурой и расширенными возможностями управления пользовательскими правами и структурой сайта. Drupal используется для крупных и нагруженных проектов, таких как государственные порталы и университетские сайты, благодаря высокой безопасности и гибкости управления данными.

Преимущества CMS включают модульность, масштабируемость и возможность интеграции с внешними сервисами. Однако для эффективного использования требуется базовое понимание веб-технологий и навыки настройки серверного окружения.

Пример использования: университет может внедрить Drupal для создания сайта с разделами для студентов, преподавателей и сотрудников, интегрируя системы управления учебными материалами и расписанием, что обеспечивает удобство и функциональность образовательного процесса.

3. Ручная разработка (Custom Development).

Ручная разработка представляет собой создание веб-ресурсов с нуля с использованием языков программирования и фреймворков, что обеспечивает максимальную гибкость и позволяет реализовать уникальный дизайн и функциональность, полностью соответствующие требованиям проекта. Основные технологии включают HTML, CSS, JavaScript для фронтенда, а также серверные языки и фреймворки, такие как Python с Django, PHP с Laravel, Node.js. Для создания динамичного интерфейса часто применяются библиотеки React и Vue.js.

Данный подход подходит для проектов с нестандартной логикой, высокой нагрузкой и необходимостью интеграции с внешними системами – сложных веб-приложений, корпоративных порталов, социальных сетей и крупных интернет-магазинов. Ручная разработка требует высокой квалификации специалистов, знаний архитектуры веб-приложений, безопасности и оптимизации производительности.

Преимущества ручной разработки включают максимальную гибкость в реализации функций и дизайна, полный контроль над архитектурой, безопасностью и масштабируемостью, возможность оптимизации скорости работы и SEO, а также легкость расширения функционала по мере роста проекта. Однако этот метод отличается высокой трудоёмкостью, требует опытных разработчиков и значительных затрат на запуск и поддержку. Ответственность за обновления и безопасность полностью лежит на команде разработчиков.

Пример использования: крупная компания может заказать разработку корпоративного портала с интеграцией внутренней системы, системы управления проектами и аналитики, реализованной с помощью React и Django. Такой сайт будет соответствовать бизнес-целям и обеспечит надежную работу при высокой нагрузке.

В результате анализа доступных методов был выбран подход ручной разработки, так как он позволяет создать уникальный функционал и дизайн, невозможные при использовании конструкторов или стандартных CMS. Этот подход даёт полную свободу действий, позволяя реализовывать сложные сценарии взаимодействия пользователей и интегрироваться с внешними сервисами. Ручная разработка обеспечивает высокую гибкость и масштабируемость, облегчая расширение функционала и модернизацию архитектуры. Она также позволяет существенно повысить производительность и безопасность благодаря точной настройке кода и внедрению специализированных защитных механизмов. Несмотря на большие начальные затраты времени и ресурсов, этот метод наиболее эффективен для крупных проектов, требующих устойчивости, гибкости и свободы от ограничений типовых платформ.

### 1.4 Обзор средств программирования

При разработке образовательного ресурса важно не только чётко определить, какие функции должна выполнять система, но и подобрать оптимальные технологии для её реализации. После того как я решила реализовывать образовательный ресурс вручную, следующим шагом стало определение подходящих технологий и инструментов для разработки. От правильного выбора языков программирования и фреймворков зависит не только эффективность и качество работы, но и дальнейшая поддержка, масштабируемость и безопасность системы. Я рассмотрела основные варианты, которые могут быть использованы для создания такой системы:

1. Языки и технологии для клиентской части (Frontend):

HTML (HyperText Markup Language).

Язык HTML, являющийся сокращением от HyperText Markup Language, представляет собой стандартный язык разметки, используемый для структурирования веб-страниц и определяющий, как контент будет отображаться в веб-браузере. Он является фундаментом всемирной паутины, обеспечивая возможность создания веб-сайтов, начиная от простых статических страниц и заканчивая сложными веб-приложениями. HTML использует теги для определения различных элементов страницы, таких как заголовки (<h1> - <h6>), абзацы (<p>), изображения (<img>), ссылки (<a>), списки (<ul>, <ol>, <li>) и таблицы (<table>). Эти элементы позволяют организовать контент в логическую иерархию, обеспечивая удобство навигации и восприятия информации. HTML-код, выступая в качестве структурной основы, определяет расположение элементов на странице. Например, <p>Это абзац текста</p> сообщает браузеру, что данный текст должен быть отображен в виде абзаца. Для визуального оформления веб-страниц используется язык CSS (Cascading Style Sheets), позволяющий задавать цвета, шрифты, размеры и другие параметры отображения элементов. Для добавления интерактивности и динамического поведения, например, для обработки пользовательских действий или обновления контента без перезагрузки страницы, используется язык JavaScript. HTML характеризуется высокой степенью совместимости с различными браузерами и операционными системами, что позволяет пользователям получать доступ к контенту с любого устройства.

Преимущества: ключевыми достоинствами являются его простота и доступность, что делает его привлекательным для начинающих разработчиков; структура HTML-кода интуитивно понятна и легко читаема, что упрощает процесс разработки и отладки; для создания HTML-документов не требуется специализированных IDE (Integrated Development Environment), достаточно простого текстового редактора, что снижает порог входа для начинающих; обеспечивает широкую совместимость, позволяя веб-страницам корректно отображаться на различных платформах и в различных браузерах; благодаря своей структурированности, позволяет быстро разрабатывать прототипы и создавать базовую структуру веб-сайта; является открытым стандартом, что означает отсутствие лицензионных ограничений и свободный доступ к документации и ресурсам; легко интегрируется с CSS для стилизации и JavaScript для добавления интерактивности, образуя мощный инструмент для веб-разработки.

Недостатки: основным ограничением является его статичный характер; для реализации динамических функций и интерактивности требуется использование дополнительных языков программирования, таких как JavaScript; не предназначен для непосредственного управления базами данных или выполнения сложных алгоритмов; существует вероятность различий в отображении некоторых HTML-элементов в разных браузерах, что может потребовать применения кросс-браузерных техник для обеспечения единообразного пользовательского опыта.

CSS (Cascading Style Sheets).

Каскадные таблицы стилей представляют собой язык, предназначенный для описания визуального представления документов, написанных на языке разметки, таком как HTML. В то время как HTML определяет структуру и содержание веб-страницы, CSS отвечает за её оформление, определяя такие аспекты, как цветовая схема, шрифты, размеры элементов, отступы и расположение контента. Эта концепция разделения структуры и представления позволяет разработчикам поддерживать чистоту и организованность кода, упрощая процесс внесения изменений и обновлений. CSS позволяет создавать адаптивные веб-сайты, которые корректно отображаются на различных устройствах, от настольных компьютеров до мобильных телефонов, обеспечивая оптимальный пользовательский опыт независимо от размера экрана. Благодаря возможностям CSS, разработчики могут реализовывать сложные анимации и переходы, обогащая интерфейс и делая его более привлекательным.

Преимущества: обеспечивает четкое разделение между структурой веб-страницы (определенной HTML) и её визуальным оформлением, это упрощает поддержку и внесение изменений, поскольку модификации стиля не затрагивают структуру документа; позволяет задавать единый стиль для всех страниц веб-сайта с помощью одного или нескольких внешних файлов, это обеспечивает согласованный внешний вид и удобство в управлении дизайном; изменение CSS-файла позволяет оперативно обновить визуальный стиль всего веб-сайта, что значительно ускоряет процесс дизайна и адаптации к новым требованиям; предоставляет мощные инструменты для создания адаптивных веб-сайтов, которые автоматически подстраиваются под различные размеры экранов и устройства, обеспечивая оптимальное отображение на любом устройстве; использование медиа-запросов (@media) позволяет применять различные стили в зависимости от характеристик устройства; позволяет создавать плавные анимации и переходы между состояниями элементов, добавляя динамику и интерактивность веб-странице без использования JavaScript, например, можно изменить цвет кнопки при наведении курсора или создать эффект плавного появления элементов при загрузке страницы; за счет повторного использования стилей, определенных в CSS-файле, уменьшается объем данных, передаваемых по сети, что приводит к ускорению загрузки веб-страниц.

Недостатки: отображение CSS-стилей может варьироваться в разных веб-браузерах, что требует тщательного тестирования и применения специальных техник для обеспечения единообразного отображения на всех платформах; реализация сложных и нетрадиционных макетов с использованием только CSS может потребовать применения нестандартных подходов и сложных конструкций, что усложняет процесс разработки и поддержки; большие CSS-файлы, содержащие избыточные или неоптимизированные стили, могут замедлять загрузку веб-сайта, особенно на мобильных устройствах с медленным интернет-соединением. сжатие CSS-файлов помогают снизить эту проблему; в крупных веб-проектах CSS-код может становиться сложным и запутанным, что затрудняет его поддержку и внесение изменений, использование методологий CSS, таких как BEM (Block Element Modifier) или OOCSS (Object-Oriented CSS), помогает организовать CSS-код и упростить его поддержку.

JavaScript (JS).

JavaScript представляет собой динамический, интерпретируемый язык программирования высокого уровня, который играет ключевую роль в создании интерактивных и функциональных веб-приложений. Изначально разработанный для добавления динамического поведения в веб-страницы на стороне клиента (в браузере), JavaScript эволюционировал и теперь используется в широком спектре приложений, включая серверную разработку, разработку мобильных приложений и создание настольных приложений. JavaScript позволяет разработчикам создавать сложные интерфейсы, реагировать на действия пользователя в реальном времени, взаимодействовать с сервером для получения и отправки данных, а также манипулировать DOM (Document Object Model) веб-страницы для динамического изменения её содержимого и внешнего вида. Он является одним из трех основных языков веб-разработки, наряду с HTML (для структуры) и CSS (для стилизации).

Преимущества: поддерживается всеми современными веб-браузерами без необходимости установки дополнительных плагинов или расширений; обеспечивает широкую доступность веб-приложений для пользователей на различных платформах; позволяет создавать интерактивные и динамичные веб-сайты; поддерживает анимации, обработку событий, валидацию форм и динамическую загрузку контента; улучшает пользовательский опыт с помощью всплывающих подсказок и автоматического заполнения полей; имеет огромную и активную экосистему готовых библиотек и фреймворков; библиотеки, такие как React, Angular и Vue.js, обеспечивают инструменты для создания сложных интерфейсов; Node.js позволяет использовать JavaScript для разработки серверной логики; доступные инструменты для автоматизации задач, тестирования и оптимизации; является универсальным языком программирования для клиентской и серверной разработки; упрощает процесс использования одних и тех же навыков и инструментов; позволяет создавать масштабируемые и высокопроизводительные серверные приложения; широко распространен и поддерживается большим количеством разработчиков по всему миру; наличие широких ресурсов и инструментов от компаний для JavaScript-разработчиков; относительно легкость изучения для начинающих; множество онлайн-ресурсов и курсов для освоения JavaScript.

Недостатки: код выполняется на стороне клиента, что позволяет пользователям просматривать и изменять исходный код, это создает угрозу безопасности, так как злоумышленники могут внедрять вредоносный код или красть конфиденциальную информацию, необходимо следовать рекомендациям по безопасности, включая валидацию входных данных, экранирование данных и использование HTTPS; различия в реализации JavaScript в разных браузерах могут приводить к нестабильной работе кода; требует дополнительного тестирования и отладки; может быть медленным для сложных вычислений и больших объемов данных; иногда требуется оптимизация кода или использование других языков для ресурсоемких задач; большое количество JavaScript-кода может замедлить загрузку страницы и снизить производительность; в крупных веб-проектах код становится трудным для поддержки и отладки; рекомендуется использовать модульную архитектуру, следовать принципам SOLID и применять инструменты для автоматического тестирования и анализа кода.

React (React.js или ReactJS).

React представляет собой декларативную, эффективную и гибкую JavaScript-библиотеку с открытым исходным кодом, предназначенную для создания пользовательских интерфейсов. Разработана компанией Facebook (в настоящее время Meta), React позволяет разработчикам создавать сложные и интерактивные интерфейсы, разбивая их на небольшие, независимые и многократно используемые компоненты. Эти компоненты, подобно строительным блокам, могут быть собраны вместе для создания полноценного пользовательского интерфейса. React использует концепцию виртуального DOM (Document Object Model) для повышения производительности и оптимизации процесса обновления пользовательского интерфейса. Вместо того, чтобы напрямую манипулировать реальным DOM браузера, React создает виртуальную копию DOM, в которой происходят все изменения. Затем React сравнивает виртуальный DOM с реальным DOM и применяет только необходимые изменения, что значительно ускоряет процесс рендеринга и повышает производительность приложения.

Преимущества: ускоряет разработку сложных интерфейсов; позволяет создавать приложения, которые быстро реагируют на действия пользователя; множество готовых компонентов; хорошо подходит для масштабируемых проектов; легко интегрируется с другими библиотеками и инструментами; использует компонентный подход, что упрощает повторное использование кода и поддержку приложения; виртуальный DOM повышает производительность и оптимизирует процесс рендеринга; декларативный подход упрощает написание и понимание кода; JSX (JavaScript XML) позволяет писать HTML-подобный код внутри JavaScript, что делает разработку более интуитивной; однонаправленный поток данных упрощает отладку и управление состоянием приложения; React Native позволяет разрабатывать мобильные приложения для iOS и Android, используя JavaScript и React; большая экосистема инструментов и библиотек упрощает разработку и тестирование React-приложений.

Недостатки: требует изучения дополнительных концепций (JSX, виртуальный DOM, управление состоянием); быстро меняется, приходится постоянно учиться новому; для небольших сайтов может быть избыточным; JSX может быть сложным для понимания для разработчиков, незнакомых с XML/HTML; интеграция с другими библиотеками может потребовать дополнительных усилий; SEO-оптимизация (Search Engine Optimization) может быть сложной из-за рендеринга на стороне клиента; виртуальный DOM может потреблять больше памяти, чем прямой DOM; отладка React-приложений может быть сложной из-за использования JSX и виртуального DOM.

Angular.

Angular – это фреймворк для создания динамических и интерактивных веб-приложений на языках программирования TypeScript и JavaScript. Он представляет собой полноценный фреймворк, предоставляющий комплексное решение для создания сложных клиентских приложений. Angular разрабатывается и поддерживается компанией OpenAI и широко используется в корпоративной среде для создания масштабируемых и поддерживаемых приложений. Angular использует компонентно-ориентированную архитектуру, где пользовательский интерфейс разбивается на переиспользуемые компоненты. Каждый компонент Angular состоит из шаблона (HTML), класса (TypeScript) и метаданных, определяющих его поведение и внешний вид. Angular предоставляет мощные инструменты для управления данными, маршрутизации, тестирования и развертывания приложений. Фреймворк следует принципам Model-View-Controller (MVC) или, точнее, его варианту Model-View-ViewModel (MVVM), что способствует разделению ответственности и упрощает разработку и поддержку кода.

Преимущества: предоставляет комплексное решение для разработки клиентских приложений; использует TypeScript, что обеспечивает статическую типизацию и улучшает качество кода; компонентно-ориентированная архитектура упрощает повторное использование кода и поддержку приложения; двустороннее связывание данных упрощает синхронизацию данных между моделью и представлением; мощная система маршрутизации; предоставляет инструменты для тестирования и отладки приложений; поддерживается компанией OpenAI и имеет большое сообщество разработчиков; хорошо подходит для разработки больших и сложных приложений; Angular CLI (Command Line Interface) упрощает создание, сборку и развертывание приложений; Dependency Injection (DI) упрощает тестирование и поддержку компонентов; RxJS (Reactive Extensions for JavaScript) предоставляет мощные инструменты для работы с асинхронными данными и событиями.

Недостатки: требует изучения TypeScript, RxJS и других дополнительных концепций; может быть сложным для начинающих разработчиков; большой размер фреймворка может влиять на производительность приложения; SEO-оптимизация может быть сложной из-за рендеринга на стороне клиента; сложное обучение может быть тяжелым для начинающих разработчиков; двустороннее связывание данных может привести к проблемам с производительностью в сложных приложениях; отладка Angular-приложений может быть сложной из-за использования TypeScript и RxJS; версии Angular могут быстро меняться, что требует постоянного обновления знаний; рендеринг на стороне сервера требует дополнительной настройки.

Vue.js.

Vue – это прогрессивный JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов и одностраничных приложений. Разработанный с акцентом на постепенное внедрение, Vue легко интегрируется в существующие проекты, позволяя разработчикам добавлять интерактивность и динамическое поведение в существующие веб-страницы без необходимости полной переработки архитектуры. Vue также предоставляет мощные инструменты для создания сложных одностраничных приложений с использованием компонентного подхода, где пользовательский интерфейс разбивается на переиспользуемые компоненты. Vue использует виртуальный DOM для повышения производительности и предоставляет декларативный синтаксис для управления данными и отображением информации. Простота, гибкость и производительность делают Vue популярным выбором для широкого круга задач, от небольших интерактивных элементов до крупных веб-приложений.

Преимущества: очень прост в освоении для новичков; легко интегрируется в существующие проекты; позволяет быстро создавать прототипы и готовые приложения; хорошо документирован и поддерживается сообществом; легкий и быстрый; прогрессивный фреймворк, позволяющий постепенно внедрять Vue в существующие проекты; компонентно-ориентированная архитектура упрощает повторное использование кода и поддержку приложения; виртуальный DOM повышает производительность и оптимизирует процесс рендеринга; декларативный синтаксис упрощает написание и понимание кода; Vue CLI (Command Line Interface) упрощает создание, сборку и развертывание приложений; Vue Router предоставляет инструменты для создания одностраничных приложений; Vuex предоставляет инструменты для управления состоянием приложения.

Недостатки: меньше крупных компаний используют Vue по сравнению с React; некоторые крупные библиотеки и инструменты выходят позже, чем у конкурентов; меньшее сообщество разработчиков по сравнению с React и Angular; меньше доступных вакансий для Vue-разработчиков; SEO-оптимизация может быть сложной из-за рендеринга на стороне клиента; возможны проблемы с масштабируемостью в очень больших и сложных приложениях; относительная молодость фреймворка может означать меньшую зрелость некоторых инструментов и библиотек.

2. Языки и фреймворки для серверной части (Backend).

Python.

Python – это интерпретируемый, объектно-ориентированный язык программирования высокого уровня с динамической типизацией. Он разработан с акцентом на читаемость кода и простоту синтаксиса, что делает его отличным выбором для начинающих программистов и для быстрой разработки прототипов. Python является универсальным языком, используемым в широком спектре приложений, включая веб-разработку, анализ данных, машинное обучение, искусственный интеллект, научные вычисления, автоматизацию задач и разработку скриптов. Благодаря своей гибкости и обширной экосистеме библиотек и фреймворков, Python стал одним из самых популярных языков программирования в мире. Он также используется для обучения программированию в школах и университетах.

Преимущества: очень простой и понятный синтаксис, подходит для детей и начинающих; используется в самых разных областях: от веб-разработки до науки и машинного обучения; огромное количество библиотек и фреймворков; легко читать и поддерживать код; кроссплатформенный, работает на различных операционных системах (Windows, macOS, Linux); интерпретируемый язык, что упрощает разработку и отладку кода; динамическая типизация упрощает написание кода, но может потребовать большего внимания к тестированию; поддерживает различные парадигмы программирования (объектно-ориентированное, процедурное, функциональное); имеет встроенные инструменты для работы со строками, числами, списками и другими типами данных; активно развивается и поддерживается сообществом.

Недостатки: работает медленнее, чем языки вроде C++ или Java; не всегда подходит для мобильных приложений и игр с высокой производительностью; иногда бывает сложно развернуть большие проекты на сервере; динамическая типизация может приводить к ошибкам во время выполнения, которые не обнаруживаются на этапе компиляции; глобальная блокировка интерпретатора (GIL) может ограничивать производительность многопоточных приложений; некоторые библиотеки могут быть плохо документированы или иметь устаревший код; может потреблять больше памяти, чем языки с более строгим управлением памятью; управление зависимостями может быть сложным в больших проектах.

Django.

Django – это высокоуровневый Python веб-фреймворк с открытым исходным кодом, который способствует быстрой разработке безопасных и поддерживаемых веб-сайтов и приложений. Django предоставляет широкий спектр встроенных функций и инструментов, избавляя разработчиков от необходимости писать код для общих задач, таких как управление пользователями, аутентификация, работа с базами данных, администрирование контента и обеспечение безопасности. Django придерживается архитектурного паттерна Model-Template-View (MTV), который способствует разделению ответственности и упрощает структуру веб-приложения. Фреймворк разработан для создания сложных веб-приложений, ориентированных на работу с данными, и широко используется для разработки веб-сайтов.

Преимущества: позволяет быстро создавать сложные сайты; встроенная система безопасности; множество готовых инструментов (администрирование, работа с базой данных, аутентификация); хорошо документирован; подходит для крупных проектов; предоставляя широкий спектр встроенных функций; использует архитектурный паттерн Model-Template-View (MTV) для разделения ответственности; Django ORM (Object-Relational Mapper) упрощает работу с базами данных; Django Admin предоставляет готовую панель администрирования для управления контентом; обеспечивает защиту от распространенных веб-уязвимостей; имеет масштабируемую архитектуру, подходящую для больших и сложных приложений.

Недостатки: может быть сложным для новичков; не всегда подходит для нестандартных задач; требует знаний Python; может быть менее гибким, чем микрофреймворки, для небольших и простых проектов; архитектура MTV может быть сложной для понимания для разработчиков, незнакомых с этим паттерном; производительность может быть ниже, чем у фреймворков, написанных на других языках (например, Node.js или Go), особенно для задач, требующих высокой производительности; может быть избыточным для простых веб-сайтов; развертывание Django-приложений может быть сложным, особенно на серверах с ограниченными ресурсами.

PHP (Hypertext Preprocessor).

PHP – это широко используемый язык программирования общего назначения с открытым исходным кодом, который особенно хорошо подходит для разработки веб-приложений и создания динамического контента. PHP выполняется на сервере и генерирует HTML-код, который затем отправляется в веб-браузер пользователя. PHP часто используется в сочетании с базами данных, такими как MySQL, для создания веб-сайтов и приложений, требующих хранения и обработки данных. Благодаря своей простоте, гибкости и широкой доступности, PHP является популярным выбором для веб-разработчиков, особенно для создания небольших и средних веб-сайтов и приложений. Он также является основой для многих популярных систем управления контентом (CMS), таких как WordPress, Joomla и Drupal.

Преимущества: работает на большинстве серверов; прост в изучении; много готовых решений и библиотек; поддерживает работу с базами данных; используется для создания популярных систем (WordPress, Joomla); имеет большое и активное сообщество разработчиков; легко интегрируется с HTML, CSS и JavaScript; предоставляет инструменты для работы с формами, сессиями и cookie; поддерживает различные архитектурные паттерны; имеет низкий порог входа для начинающих разработчиков; хорошо подходит для создания простых и средних веб-сайтов.

Недостатки: считается устаревшим по сравнению с современными языками; сложно поддерживать большой и запутанный код; безопасность зависит от опыта разработчика; имеет непоследовательный синтаксис, что может приводить к ошибкам; динамическая типизация может усложнять отладку кода; отсутствие строгой структуры может приводить к плохо организованному коду; медленнее, чем некоторые другие языки программирования (например, Node.js или Go); не всегда подходит для разработки сложных веб-приложений с высокой производительностью; управление зависимостями может быть сложным в больших проектах.

Laravel:

Laravel – это элегантный и мощный PHP веб-фреймворк с открытым исходным кодом, предназначенный для упрощения разработки веб-приложений. Он предоставляет богатый набор инструментов и функций, которые позволяют разработчикам создавать современные, безопасные и масштабируемые веб-приложения с меньшими усилиями. Laravel акцентирует внимание на элегантном синтаксисе, удобстве для разработчиков и высокой производительности. Фреймворк предоставляет встроенную поддержку для общих задач, таких как маршрутизация, аутентификация, работа с базами данных, шаблонизация, тестирование и развертывание. Laravel также имеет активное и поддерживающее сообщество, что делает его популярным выбором для PHP-разработчиков.

Преимущества: упрощает работу с PHP; встроенные инструменты для обеспечения безопасности, работы с базой данных, почтой и т.д.; современный подход к разработке; большое сообщество и хорошая документация; элегантный и выразительный синтаксис, облегчающий чтение и написание кода; обеспечивает разделение ответственности и упрощает структуру приложения; предоставляет мощный и удобный способ создания динамических HTML-страниц; упрощает взаимодействие с базами данных; предоставляет инструменты для автоматизации задач разработки; встроенная поддержка тестирования, включая модульные тесты и функциональные тесты; обеспечивает защиту от распространенных веб-уязвимостей; имеет масштабируемую архитектуру, подходящую для больших и сложных приложений.

Недостатки: требует знаний PHP; может быть избыточным для простых сайтов; иногда сложен для новичков; большой размер фреймворка может влиять на производительность приложения; обучение может быть сложным для начинающих разработчиков; архитектура может быть сложной для понимания для разработчиков, незнакомых с этим паттерном; может потребовать больше ресурсов сервера, чем более легковесные фреймворки; обновления фреймворка могут требовать значительных изменений в коде приложения.

Node.js.

Node.js – это кросс-платформенная среда выполнения JavaScript с открытым исходным кодом, которая позволяет выполнять JavaScript-код на стороне сервера. Она построена на движке V8 JavaScript, разработанном Google для браузера Chrome. Node.js использует неблокирующую, событийную модель ввода-вывода, что делает его легким и эффективным, особенно для приложений реального времени, таких как чаты, потоковые сервисы и онлайн-игры. Node.js позволяет разработчикам использовать JavaScript как для клиентской, так и для серверной части веб-приложения, что упрощает разработку и обслуживание. Благодаря своей масштабируемости и производительности, Node.js широко используется для разработки микросервисов, бэкендов для веб-приложений и других серверных решений.

Преимущества: позволяет использовать JavaScript как на стороне клиента, так и на стороне сервера; использует неблокирующую, событийную модель ввода-вывода, что обеспечивает высокую производительность; большое и активное сообщество разработчиков; огромное количество готовых модулей и пакетов; хорошо подходит для разработки приложений реального времени; легко масштабируется; кросс-платформенный, работает на различных операционных системах (Windows, macOS, Linux); подходит для разработки микросервисов; относительно легко начать учить, особенно для разработчиков, знакомых с JavaScript.

Недостатки: может быть сложным для начинающих разработчиков, незнакомых с асинхронным программированием; не всегда подходит для вычислительно-интенсивных задач из-за однопоточной архитектуры; callback hell (проблема вложенных колбэков) может усложнять чтение и отладку кода; требует опыта в управлении зависимостями; некоторые модули могут быть плохо документированы или иметь устаревший код; обработка ошибок может быть сложной в асинхронном коде; однопоточная модель может ограничивать производительность приложений, требующих многопоточности; может потреблять больше памяти, чем некоторые другие серверные технологии.

В результате проведенного анализа различных средств программирования для разработки веб-приложения было принято решение использовать следующий стек технологий: HTML, CSS и JavaScript для клиентской части и PHP для серверной стороны. Данный выбор обусловлен рядом ключевых факторов, связанных с функциональностью, производительностью, удобством разработки и стоимостью владения. HTML и CSS обеспечивают основу для структуры и внешнего вида веб-страниц. Современные методы CSS, такие как Flexbox и Grid Layout, позволят создать адаптивный и привлекательный пользовательский интерфейс, который корректно отображается на разных устройствах.. JavaScript предоставляет необходимые инструменты для добавления интерактивности и динамического поведения в клиентскую часть приложения. PHP выбран для работы на сервере благодаря своей распространенности, развитой среде, простоте развертывания и совместимости с базами данных. PHP подходит для обработки запросов, взаимодействия с базой данных, управления пользователями и обеспечения безопасности.

Выбранный стек технологий предоставляет оптимальный баланс между производительностью, удобством разработки, стоимостью владения и возможностью реализации всех необходимых функций веб-приложения.

### 1.5 Описание языка программирования

В предыдущем разделе был выбран стек технологий: : HTML, CSS и JavaScript для клиентской части и PHP для серверной стороны.

HTML (HyperText Markup Language).

HTML является основой для создания структуры веб-страниц. Теги используются для определения различных элементов контента, таких как заголовки, параграфы, изображения и ссылки. В данном проекте HTML используется для организации контента и создания макета страниц.

<!-- Фон затемнения для модального окна -->

<div id="modalBackground" class="modal-background"></div>

<!-- Модальное окно с формами авторизации и регистрации -->

<div id="authModal" class="modal hidden">

<!-- Основное содержимое модального окна -->

<div class="modal-content auth-container">

<!-- Кнопка закрытия модального окна -->

<span class="close">&times;</span>

<!-- Форма авторизации -->

<div class="form-container sign-in-container">

<form action="actions/action\_authorization.php" id="authorizationForm" method="POST">

<h1>Войти в аккаунт</h1>

<!-- Поле ввода фамилии -->

<input type="text" placeholder="Фамилия" name="surname" required minlength="2" maxlength="50" title="Только буквы, длина от 2 до 50 символов" autocomplete="off"/>

<!-- Поле ввода пароля -->

<input type="password" placeholder="Пароль" name="password" required minlength="6" maxlength="15" title="Пароль должен содержать минимум 6 символов максимум 15" autocomplete="off"/>

<!-- Кнопка отправки формы -->

<button type="submit" name="button\_authorization">Войти</button>

<!-- Блок для вывода сообщений об ошибках или успехе -->

<p id="message\_a"></p>

</form>

</div>

<!-- Форма регистрации -->

<div class="form-container sign-up-container">

<form action="actions/action\_registration.php" id="registrationForm" method="POST">

<h1>Создать аккаунт</h1>

<!-- Поле ввода фамилии -->

<input type="text" placeholder="Фамилия" name="surname" required minlength="2" maxlength="50" title="Только буквы, длина от 2 до 50 символов" autocomplete="off"/>

<!-- Поле ввода имени -->

<input type="text" placeholder="Имя" name="name" required minlength="2" maxlength="50" title="Только буквы, длина от 2 до 50 символов" autocomplete="off"/>

<!-- Поле ввода группы -->

<input type="text" placeholder="Группа" name="group" required minlength="3" maxlength="4" title="Только буквы и цифры, длина 3 или 4 символа" autocomplete="off"/>

<!-- Поле ввода пароля -->

<input type="password" placeholder="Пароль" name="password" required minlength="6" maxlength="15" title="Пароль должен содержать минимум 6 символов максимум 15" autocomplete="off"/>

<!-- Подтверждение пароля -->

<input type="password" placeholder="Повторите пароль" name="password\_check" required minlength="6" maxlength="15" title="Пароль должен совпадать с предыдущим полем" autocomplete="off"/>

<!-- Кнопка отправки формы регистрации -->

<button type="submit" name="button\_registration">Зарегистрироваться</button>

<!-- Блок для вывода сообщений об ошибках или успехе -->

<p id="message\_r"></p>

</form>

</div>

<!-- Контейнер с информацией на боковой панели модального окна -->

<div class="overlay-container">

<div class="overlay">

<!-- Панель с информацией о входе -->

<div class="overlay-panel overlay-left">

<h1 class="white">Добро пожаловать!</h1>

<p class="white">Пожалуйста, войдите в аккаунт</p>

<!-- Кнопка переключения на форму авторизации -->

<button class="overlay-button" id="signIn">Войти</button>

</div>

<!-- Панель с информацией о регистрации -->

<div class="overlay-panel overlay-right">

<h1 class="white">Вы еще не с нами?</h1>

<p class="white">Скорее регистрируйтесь!</p>

<!-- Кнопка переключения на форму регистрации -->

<button class="overlay-button" id="signUp">Зарегистрироваться</button>

</div>

</div>

</div>

</div>

</div>

CSS используется для стилизации HTML-элементов и определения их внешнего вида. Цвет, шрифт, расположение элементов и другие визуальные характеристики веб-страницы управляются с помощью CSS. В проекте CSS применяется для создания привлекательного и удобного пользовательского интерфейса.

/\* стиль для контейнера с темами \*/

.home\_\_topics-container {

display: grid; /\* Активирует режим сетки \*/

grid-template-columns: repeat(5, 230px); /\* 5 колонок по 300px \*/

grid-template-rows: repeat(2, 230px); /\* 2 строки высотой 200px \*/

gap: 10px; /\* Пространство между ячейками (необязательно) \*/

}

/\* Первый блок занимает 2 строки вертикально \*/

.home\_\_topics-container > :nth-child(1) {

grid-row: 1 / 3; /\* Занимает строки с 1 по 3 (включая 2 строки) \*/

grid-column: 1 / 2; /\* Занимает одну колонку \*/

flex-direction: column;

}

/\* Второй блок занимает 2 колонки на первой строке \*/

.home\_\_topics-container > :nth-child(2) {

grid-row: 1 / 2; /\* Занимает первую строку \*/

grid-column: 2 / 4; /\* Занимает две колонки (с 2 по 4) \*/

}

/\* Третий блок занимает 2 колонки на первой строке \*/

.home\_\_topics-container > :nth-child(3) {

grid-row: 1 / 2; /\* Занимает первую строку \*/

grid-column: 4 / 6; /\* Занимает две колонки (с 4 по 6) \*/

}

/\* Четвертый блок занимает 4 колонки на второй строке \*/

.home\_\_topics-container > :nth-child(4) {

grid-row: 2 / 3; /\* Занимает вторую строку \*/

grid-column: 2 / 6; /\* Занимает четыре колонки (с 2 по 6) \*/

}

/\* стили для блоков \*/

.home\_\_topic-card {

display: flex;

background-color: #fff; /\*Легкий зеленый фон \*/

border-radius: 10px; /\* Закругленные углы \*/

padding: 20px;/\* Внутренние отступы \*/

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);/\* Тень для эффекта карточки \*/

}

.home\_\_topic-card h3{

text-align: left; /\* Выравнивание текста по левому краю \*/

}

.home\_\_topic-card img {

width: 190px; /\* Изображение занимает всю ширину блока \*/

height: 190px; /\* Сохраняем пропорции изображения \*/

margin-bottom: 10px; /\* Отступ снизу от изображения \*/

margin-left: 10px;

border-radius: 10px;

}

.home\_\_topic-card h3 {

font-size: 18px; /\* Размер заголовка темы \*/

color: #333; /\* Цвет заголовка \*/

margin: 0 0 10px; /\* Отступы вокруг заголовка \*/

}

.home\_\_topic-card p {

font-size: 16px; /\* Размер текста \*/

color: #555; /\* Цвет текста \*/

line-height: 1.6; /\* Межстрочный интервал \*/

}

JavaScript – это язык программирования, который используется для добавления интерактивности на веб-страницы. В данном проекте JavaScript применяется для обработки пользовательского ввода, отправки данных на сервер и динамического обновления контента.

// Слушаем клик по всей странице

document.body.addEventListener("click", function (e) {

// Проверяем, был ли клик по кнопке с классом "submit-test"

if (e.target.classList.contains("submit-test")) {

// Находим ближайшую форму теста

const form = e.target.closest(".test-form");

if (!form) return; // Если форма не найдена — выходим

// Ищем индикатор загрузки внутри формы

const loadingIndicator = form.querySelector(".test-loading-indicator");

// Проверяем, все ли вопросы в тесте отвечены

const questions = form.querySelectorAll(".question-block");

let allAnswered = true;

questions.forEach((questionBlock) => {

// Получаем все радиокнопки текущего вопроса

const radioButtons = questionBlock.querySelectorAll('input[type="radio"]');

// Проверяем, выбран ли хотя бы один вариант ответа

const isChecked = Array.from(radioButtons).some(radio => radio.checked);

if (!isChecked) allAnswered = false;

});

// Если есть незаполненные вопросы — выводим предупреждение

if (!allAnswered) {

alert("Пожалуйста, ответьте на все вопросы перед отправкой теста.");

return;

}

// Отображаем индикатор загрузки

loadingIndicator.style.display = "block";

// Получаем ID лекции из скрытого поля или dataset формы

const lectureIdInput = form.querySelector("[name='lecture\_id']");

const lectureId = lectureIdInput ? lectureIdInput.value : form.dataset.lectureId;

// Собираем данные формы для отправки

const formData = new FormData(form);

formData.append("lecture\_id", lectureId); // Добавляем ID лекции в данные

// Отправляем данные на сервер

fetch("actions/action\_course-getTest.php", {

method: "POST",

body: formData

})

.then(response => response.json()) // Ожидаем JSON-ответ

.then(data => {

// Скрываем индикатор загрузки

loadingIndicator.style.display = "none";

if (data.success) {

// Если тест успешно обработан — показываем результаты

alert(`Тест успешно отправлен! Ваша оценка: ${data.grade}`);

updateAttemptsDisplay(lectureId, data.attemptsLeft); // Обновляем количество попыток

showTestResults(form, data); // Отображаем результаты теста

} else {

// Выводим сообщение об ошибке

alert(data.message || "Ошибка при отправке теста");

}

})

.catch(error => {

// Обработка сетевых ошибок

loadingIndicator.style.display = "none";

console.error("Ошибка сети:", error);

alert("Произошла ошибка при отправке теста.");

});

}

});

PHP – это скриптовый язык программирования, который выполняется на сервере. Он используется для обработки запросов от клиента, взаимодействия с базой данных и генерации динамического HTML-контента. В данном проекте PHP применяется для обработки данных, отправленных с клиентской стороны, и управления данными учебника.

function getCache($key)

{

// Формируем путь к файлу кэша на основе MD5-хэша ключа

$cacheFile = \_\_DIR\_\_ . '/cache/' . md5($key) . '.cache';

// Проверяем существование файла и его актуальность (не старше 1 часа)

if (file\_exists($cacheFile) && (time() - filemtime($cacheFile)) < 3600) {

// Читаем содержимое кэш-файла

$data = file\_get\_contents($cacheFile);

// Если чтение не удалось — записываем ошибку в лог и возвращаем false

if ($data === false) {

error\_log("Cache read error for key: $key");

return false;

}

// Декодируем JSON-данные

$decodedData = json\_decode($data, true);

// Проверяем, успешно ли произошла декодировка JSON

if (json\_last\_error() !== JSON\_ERROR\_NONE) {

error\_log("JSON decode error for key: $key - " . json\_last\_error\_msg());

return false;

}

// Возвращаем успешно прочитанные и обработанные данные

return $decodedData;

}

// Если кэш отсутствует или устарел — возвращаем false

return false;

}