МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ»

СТЕРЛИТАМАКСКИЙ ФИЛИАЛ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ПО ПРОГРАММЕ МАГИСТРАТУРЫ

(ПРОЕКТНЫЙ ТИП)

АЛЛАБЕРДИН ИЛЬМИР ИЛЬФАТОВИЧ

СТЕПАНОВ АЛЕКСЕЙ АНАТОЛЬЕВИЧ

РАЗРАБОТКА САЙТА НАУЧНО-ПРАКТИЧЕСКОЙ КОНФЕРЕНЦИИ

|  |  |
| --- | --- |
| Заведующий кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ученая степень, ученое звание, должность)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (И.О. Фамилия) | Выполнили:  Студенты \_\_\_\_\_ курса \_\_\_\_ формы обучения  Направление подготовки (специальность)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Направленность (профиль)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | Руководитель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ученая степень, ученое звание, должность)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (И.О. Фамилия) |
|  | Консультант (нормоконтролер) (при наличии)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ученая степень, ученое звание, должность)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (И.О. Фамилия) |

СТЕРЛИТАМАК – 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc199839322)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ ВЕБ-САЙТА 5](#_Toc199839323)

[1.1. Актуальность разработки. 5](#_Toc199839324)

[1.2. Исследование предметной области «Разработка веб-сайтов» 6](#_Toc199839325)

[1.3. Выбор средств разработки и их описание 12](#_Toc199839326)

[1.3.1. Инструменты разработки клиентской части приложения. 12](#_Toc199839327)

[1.3.2. Инструменты разработки серверной части приложения. 24](#_Toc199839328)

[1.4. Выводы по первой главе 33](#_Toc199839329)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ВЕБ-САЙТА 34](#_Toc199839330)

[2.1. Проектирование веб-сайта 34](#_Toc199839331)

[2.2 . Разработка клиентской части приложения 35](#_Toc199839332)

[2.3. Разработка серверной части приложения 43](#_Toc199839333)

[2.3.1. Принцип чистой архитектуры 43](#_Toc199839334)

[2.3.2. Создание и заполнение проекта ConferenceApp.Domain 46](#_Toc199839335)

[2.3.3. Создание и заполнение проекта ConferenceApp.Application 48](#_Toc199839336)

[2.3.4. Создание и заполнение проекта ConferenceApp.Infrastructure. 50](#_Toc199839337)

[2.3.5. Создание и заполнение проекта ConferenceApp.WebApi 52](#_Toc199839338)

[2.3.6. Связывание клиентской и серверной частей. 56](#_Toc199839339)

[2.4. Выводы по второй главе 56](#_Toc199839340)

[ЗАКЛЮЧЕНИЕ 57](#_Toc199839341)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 58](#_Toc199839342)

[ПРИЛОЖЕНИЯ 62](#_Toc199839343)

# ВВЕДЕНИЕ

В современном мире, когда информационные технологии играют решающую роль в нашей жизни, создание эффективного и удобного веб-сайта стало необходимостью для любой организации или мероприятия. В частности, научно-практические конференции, которые собирают вместе экспертов и исследователей из различных областей, требуют современной и функциональной онлайн-платформы для обмена знаниями и опытом.

Разработка сайта научно-практической конференции является важнейшим аспектом организации такого мероприятия. Сайт должен быть не только информативным, но и удобным для пользователей, чтобы они могли легко найти необходимую информацию и взаимодействовать с организаторами и участниками конференции.

Целью данной выпускной квалификационной работы является разработка сайта научно-практической конференции, который будет соответствовать современным требованиям и ожиданиям пользователей. В ходе работы будет рассмотрена теоретическая основа разработки веб-сайтов, а также будет представлен практический опыт создания сайта научно-практической конференции с использованием современных технологий и инструментов.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ существующих веб-сайтов, предназначенных для организации научных конференций, с целью выявления лучших практик и определения недостатков существующих решений. Особое внимание будет уделено сайтам математических конференций.

2. Выбрать оптимальную технологическую платформу и инструментарий для разработки веб-сайта, обеспечивающие высокую производительность, надежность и масштабируемость будущего решения. Выбор будет обоснован на основе анализа требуемых функциональных возможностей.

3. Разработать архитектуру и прототип пользовательского интерфейса сайта, обеспечивающего эргономичность и удобство пользовательского опыта для различных категорий пользователей (участники, докладчики, организаторы).

4. Разработать и реализовать функциональные модули сайта, включая регистрацию участников, подачу тезисов, публикацию материалов конференции, формирование программы мероприятий и т.д.

5. Провести всестороннее тестирование разработанного веб-сайта для выявления и исправления возможных ошибок и недостатков, а также оптимизировать его производительность и доступность.

Объектом исследования является процесс разработки веб-сайта для научной конференции. Предметом исследования являются функциональные и технические аспекты разрабатываемого веб-сайта, включая его архитектуру, дизайн, функциональность и производительность.

В качестве методов исследования будут использоваться методы анализа, проектирования, разработки и тестирования программного обеспечения.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ ВЕБ-САЙТА

* 1. Актуальность разработки.

Актуальность темы исследования обусловлена стремительным развитием информационных технологий и их все более широким применением в организации научных мероприятий. Традиционные методы проведения конференций, ограниченные географическими рамками и сложностью распространения информации, уступают место современным цифровым решениям. Разработка удобного и функционального веб-сайта для научной конференции позволяет значительно упростить процесс организации мероприятия, расширить аудиторию участников, обеспечить оперативный доступ к информации и материалам конференции, а также повысить эффективность взаимодействия между организаторами и участниками.

В частности, конференции часто характеризуются привлечением участников из разных стран. Эффективная коммуникация между организаторами и участниками, а также между самими участниками, становится критичным фактором успеха мероприятия. Веб-сайт предоставляет удобную платформу для обмена информацией, публикации материалов конференции, онлайн-регистрации и взаимодействия участников, способствуя повышению эффективности научного обмена.

Вдобавок, современные веб-технологии позволяют создавать интерактивные и функциональные сайты, которые не только предоставляют информацию, но и обеспечивают удобные инструменты для работы с ней. Возможности онлайн-регистрации, загрузки тезисов и полных текстов докладов и системы обратной связи позволяют автоматизировать многие рутинные процессы, сэкономить время и ресурсы организаторов, а также повысить удобство для участников.

Также наличие профессионально разработанного веб-сайта повышает престиж конференции и привлекает большее количество участников, способствуя популяризации математических исследований и укреплению международного научного сотрудничества. Исходя из вышесказанного, разработка веб-сайта для математической конференции не только отвечает современным требованиям организации научных мероприятий, но и способствует повышению их эффективности, расширению охвата аудитории и укреплению международных научных связей.

* 1. Исследование предметной области «Разработка веб-сайтов»

Разработка веб-сайтов – это многогранная и динамично развивающаяся область, которая охватывает широкий спектр технологий, инструментов и методологий. На сегодняшний день веб-сайты стали неотъемлемой частью нашей повседневной жизни, выполняя различные функции и задачи, от простых информационных ресурсов до сложных интерактивных платформ. Важно отметить, что веб-разработка не ограничивается лишь созданием визуально привлекательных страниц; это комплексный процесс, который включает в себя проектирование, программирование, тестирование и поддержку веб-приложений.

С момента появления первых сайтов в 1990-х годах, веб-технологии претерпели значительные изменения. Изначально веб-страницы представляли собой простые текстовые документы, однако с развитием технологий и увеличением скорости интернета они стали более интерактивными и многофункциональными. Современные веб-сайты могут включать в себя различные мультимедийные элементы, такие как видео, анимация и интерактивные графики, что значительно обогащает пользовательский опыт. Это, в свою очередь, создает новые вызовы для разработчиков, требующих от них постоянного обновления знаний и навыков в области новых технологий и трендов.

Одним из ключевых аспектов разработки веб-сайтов является выбор подходящей архитектуры и технологий. Существует множество языков программирования и фреймворков, которые могут быть использованы для создания веб-приложений. Наиболее популярными языками являются HTML, CSS и JavaScript, которые составляют основу большинства сайтов. HTML (HyperText Markup Language) отвечает за структуру страницы, CSS (Cascading Style Sheets) — за ее оформление, а JavaScript — за интерактивность. В дополнение к этим языкам, разработчики часто используют различные библиотеки и фреймворки, такие как React, Angular или Vue.js, которые позволяют ускорить процесс разработки и улучшить производительность приложения.

Кроме того, важным аспектом является серверная часть веб-разработки, которая отвечает за обработку запросов пользователей и управление данными. Серверные языки, такие как PHP, C#, Python, Ruby и Node.js, позволяют создавать динамические веб-приложения, которые могут взаимодействовать с базами данных и другими внешними ресурсами. Веб-сайты также могут использовать различные системы управления контентом (CMS), такие как WordPress, Joomla или Drupal, которые упрощают процесс создания и управления контентом для пользователей, не обладающих глубокими техническими знаниями.

Одной из главных тенденций в разработке веб-сайтов является переход к мобильным устройствам. С каждым годом все больше пользователей предпочитают получать доступ к интернету через смартфоны и планшеты, что делает адаптивный дизайн необходимостью. Адаптивные веб-сайты автоматически подстраиваются под размеры экрана устройства, обеспечивая удобный доступ к информации независимо от платформы. Это требует от разработчиков особого внимания к юзабилити и пользовательскому опыту, чтобы гарантировать, что сайт будет одинаково функционален и привлекателен на всех устройствах.

Не менее важным аспектом является безопасность веб-приложений. С увеличением числа кибератак и утечек данных, разработчики должны уделять особое внимание защите информации пользователей. Это включает в себя использование современных методов шифрования, регулярные обновления программного обеспечения и внедрение лучших практик программирования, чтобы минимизировать риски. Безопасность веб-сайтов становится не только техническим, но и этическим вопросом, поскольку пользователи доверяют разработчикам свои личные данные и ожидания о безопасном взаимодействии.

Также нельзя забывать о важности SEO (поисковой оптимизации), которая играет ключевую роль в том, насколько легко пользователи могут найти веб-сайт в поисковых системах. Оптимизация сайта для поисковых систем включает в себя множество факторов, таких как структура URL, мета-теги, контент и скорость загрузки страниц. Эффективная SEO-стратегия может значительно повысить видимость сайта и привлечь больше посетителей, что, в свою очередь, может привести к увеличению конверсий и продаж.

Далее рассмотрим основные этапы разработки современных веб-сайтов и веб-приложений:

Первым и, пожалуй, одним из самых критически важных этапов является этап планирования. На этом этапе команда разработчиков, дизайнеров и других заинтересованных сторон собирается вместе, чтобы обсудить цели и задачи проекта. Это время для мозгового штурма, когда идеи и концепции начинают принимать форму. Важно четко определить, какую проблему решает веб-сайт или веб-приложение, кто будет его целевой аудиторией и какие функции должны быть реализованы. На этом этапе также разрабатывается общая стратегия, которая включает в себя временные рамки, бюджет и распределение ресурсов. Это своего рода фундамент, на котором будет строиться весь проект, и от того, насколько хорошо будет выполнено это задание, во многом будет зависеть успех всего предприятия.

После того как цели и задачи определены, наступает этап исследования и анализа. Здесь команда проводит глубокое исследование рынка, изучает конкурентов и анализирует существующие решения. Это помогает выявить лучшие практики и определить, что работает, а что нет. Команда также может проводить опросы и интервью с потенциальными пользователями, чтобы понять их потребности и ожидания. Все эти данные собираются и анализируются, чтобы создать более точное представление о том, каким должен быть конечный продукт. Это время, когда идеи начинают обретать конкретные очертания, и команда начинает формировать концепцию дизайна и функциональности.

Следующим этапом является создание прототипа. Прототип — это своего рода предварительная версия веб-сайта или приложения, которая позволяет визуализировать идеи и концепции, разработанные на предыдущих этапах. Прототип может быть как низкоуровневым, с простыми черно-белыми макетами, так и высокоуровневым, с более детализированным дизайном и интерактивными элементами. Этот этап важен, поскольку он позволяет команде и заинтересованным сторонам увидеть, как будет выглядеть конечный продукт, и внести изменения до того, как начнется фактическая разработка. Прототипы могут быть использованы для тестирования пользовательского опыта и получения обратной связи от пользователей, что позволяет выявить возможные проблемы и улучшить дизайн и функциональность.

После того как прототип утвержден, команда переходит к этапу дизайна. Здесь дизайнеры работают над созданием визуального оформления сайта или приложения. На этом этапе важно учитывать не только эстетику, но и функциональность, чтобы пользовательский интерфейс был удобным и интуитивно понятным. Дизайнеры создают цветовые схемы, шрифты, иконки и другие элементы интерфейса, а также разрабатывают макеты страниц. Это время, когда креативность и функциональность должны идти рука об руку, чтобы создать не только красивый, но и практичный продукт. Дизайнеры также должны учитывать адаптивность, чтобы сайт или приложение хорошо смотрелись и работали на различных устройствах и экранах.

Когда дизайн утвержден, начинается этап разработки. Это, пожалуй, один из самых объемных и трудоемких этапов в процессе создания веб-сайта или приложения. Здесь программисты берутся за реализацию всех функций и возможностей, которые были запланированы на предыдущих этапах. Разработка может быть разделена на две основные части: фронтенд и бэкенд. Фронтенд-разработка включает в себя создание всего, что видит пользователь: это интерфейс, элементы управления, анимации и взаимодействия. Разработчики используют языки программирования, такие как HTML, CSS и JavaScript, а также различные библиотеки и фреймворки, чтобы создать привлекательный и функциональный интерфейс.

Бэкенд-разработка, в свою очередь, отвечает за серверную часть приложения. Это включает в себя работу с базами данных, обработку запросов от пользователей и управление логикой приложения. Разработчики используют языки программирования, такие как PHP, Python, Ruby или Node.js, а также различные системы управления базами данных, чтобы обеспечить надежную и эффективную работу приложения. На этом этапе важно также обеспечить безопасность данных и защиту от потенциальных угроз, что требует особого внимания и применения современных методов шифрования и аутентификации.

Когда разработка завершена, наступает этап тестирования. Это критически важный этап, который позволяет выявить и исправить ошибки и недочеты, прежде чем продукт будет запущен в эксплуатацию. Тестирование может включать в себя различные виды проверок, такие как функциональное тестирование, тестирование производительности, тестирование безопасности и пользовательское тестирование. Команда должна убедиться, что все функции работают так, как задумано, и что пользовательский опыт соответствует ожиданиям. Это время, когда важно не только выявить ошибки, но и убедиться, что продукт готов к запуску и будет удовлетворять потребности пользователей.

После успешного тестирования наступает этап развертывания. На этом этапе веб-сайт или приложение загружается на сервер и становится доступным для пользователей. Это может быть захватывающий момент, когда все усилия команды начинают приносить плоды. Однако развертывание — это не конец процесса, а скорее его начало. После запуска необходимо следить за работой приложения, чтобы убедиться, что все функционирует корректно, и оперативно реагировать на возможные проблемы. Команда также должна быть готова к тому, чтобы внести изменения и обновления в продукт на основе отзывов пользователей и анализа данных.

Наконец, следует этап поддержки и обслуживания. Это постоянный процесс, который включает в себя регулярные обновления, исправления ошибок и добавление новых функций. Веб-сайты и приложения требуют постоянного внимания, чтобы оставаться актуальными и безопасными. Команда должна следить за изменениями в технологиях, обновлениями программного обеспечения и изменениями в потребительских предпочтениях, чтобы своевременно адаптировать продукт к новым условиям. Это также время для анализа данных и метрик, чтобы понять, как пользователи взаимодействуют с продуктом и где можно внести улучшения.

Таким образом, разработка современных веб-сайтов и веб-приложений – это сложный и многогранный процесс, который включает в себя множество этапов, от планирования и исследования до разработки, тестирования и поддержки. Каждый из этих этапов играет важную роль в создании качественного и функционального продукта, который будет удовлетворять потребности пользователей и соответствовать современным требованиям технологий. Важно помнить, что успешная разработка требует не только технических навыков, но и креативного подхода, способности работать в команде и готовности адаптироваться к быстро меняющемуся миру технологий. Каждый новый проект – это возможность для обучения и роста, и каждая завершенная работа – это шаг к созданию чего-то уникального и полезного.

В заключение, разработка веб-сайтов – это сложный и многогранный процесс, который требует от специалистов не только технических знаний, но и креативного подхода, понимания потребностей пользователей и способности адаптироваться к быстро меняющемуся миру технологий. Исследование предметной области «Разработка веб-сайтов» открывает перед нами множество возможностей и вызовов, которые требуют постоянного обучения и самосовершенствования. В условиях глобализации и стремительного развития технологий создание качественных веб-приложений становится не только актуальной задачей, но и важным вкладом в развитие информационного общества.

* 1. Выбор средств разработки и их описание
     1. Инструменты разработки клиентской части приложения.

1. HTML. Непосредственная разработка клиентской части веб-приложений или веб-сайтов всегда начинается с верстки макета. Для этого с помощью HTML собирается «скелет» сайта, т.е. основной каркас приложения. HTML (HyperText Markup Language) – основной язык разметки, используемый для создания и структурирования контента в интернете. Он служит основой для веб-страниц и веб-приложений, позволяя разработчикам и дизайнерам организовывать текст, изображения, видео и другие элементы на страницах таким образом, чтобы они были понятны и удобны для пользователей.

HTML был разработан в начале 1990-х годов Тимом Бернерсом-Ли, который хотел создать стандартный язык для обмена документами через интернет. С тех пор язык значительно развился, и на сегодняшний день существует несколько версий HTML, самой последней из которых является HTML5. HTML5 ввел множество новых возможностей, таких как поддержка мультимедиа, улучшенные семантические элементы и API для создания более интерактивных веб-приложений.

Структура HTML-документа достаточно проста. Каждый документ начинается с объявления типа документа, которое сообщает браузеру, что он должен ожидать HTML-код. Затем следует корневой элемент — <html>, который содержит все остальные элементы. Внутри этого элемента обычно находятся два основных раздела: <head> и <body>. Раздел <head> содержит метаданные о документе, такие как заголовок страницы, ссылки на стили и скрипты, а также другую информацию, которая не отображается непосредственно на странице. Заголовок страницы, например, задается с помощью тега <title> и отображается на вкладке браузера.

Раздел <body> является основной частью документа, где размещается весь видимый контент. Здесь можно использовать различные теги для структурирования текста, добавления изображений, создания ссылок и многого другого. Например, для заголовков используются теги <h1>, <h2>, <h3> и так далее, которые обозначают уровни заголовков, от самого важного до менее значимого. Тег <p> используется для создания абзацев текста, а тег <a> позволяет создавать гиперссылки, которые ведут на другие страницы или ресурсы.

Одной из ключевых особенностей HTML является возможность использования атрибутов, которые добавляют дополнительную информацию к элементам. Атрибуты указываются внутри тега и могут задавать такие параметры, как идентификаторы, классы, ссылки на изображения и многое другое. Например, тег <img> используется для вставки изображений, и его атрибут «src» указывает путь к файлу изображения, а атрибут «alt» предоставляет текстовое описание изображения для пользователей с ограниченными возможностями и для поисковых систем.

HTML также поддерживает создание списков, как упорядоченных, так и неупорядоченных, что позволяет организовывать информацию в структурированном виде; создание таблиц, что позволяет представлять данные в виде строк и столбцов, что в свою очередь особенно полезно для отображения информации, такой как расписания, статистика или сравнения и многие другие возможности.

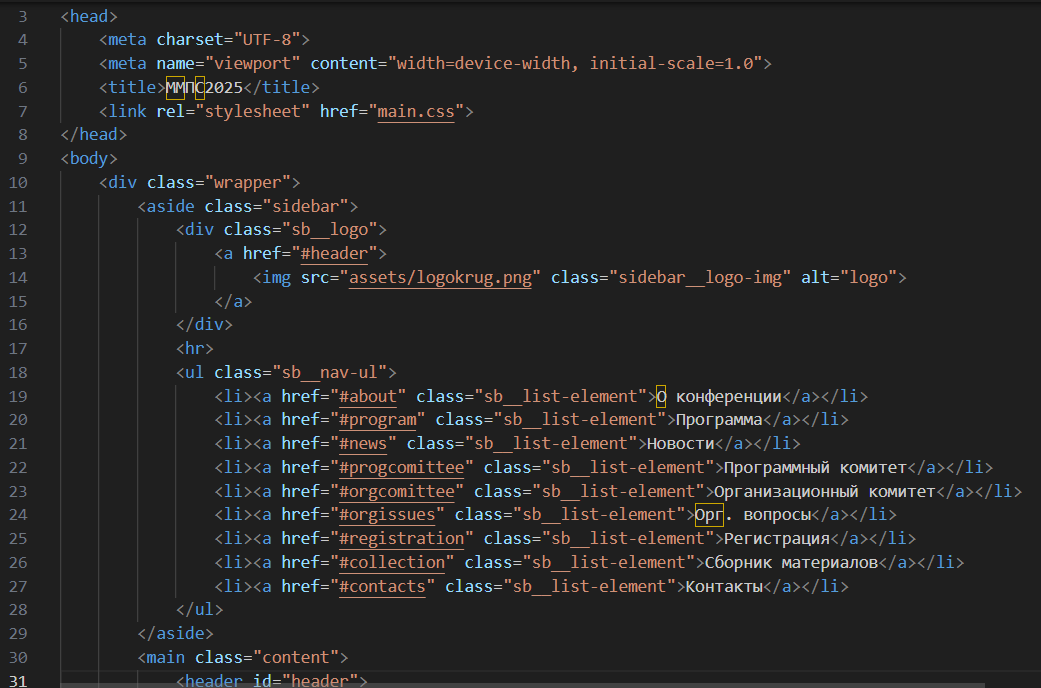


Рис. 1.1. Пример HTML-кода.

С развитием веб-технологий HTML стал неотъемлемой частью создания интерактивных и динамичных веб-приложений. В сочетании с CSS (Cascading Style Sheets) и JavaScript, HTML позволяет создавать богатые пользовательские интерфейсы и улучшать взаимодействие с пользователем. CSS отвечает за визуальное оформление элементов, позволяя задавать цвета, шрифты, отступы и другие стили, в то время как JavaScript добавляет функциональность и интерактивность, позволяя реагировать на действия пользователя и изменять содержимое страницы без перезагрузки.

Важно отметить, что HTML – это язык разметки, а не язык программирования. Это означает, что он не выполняет логических операций или расчетов, как это делает JavaScript. Вместо этого HTML структурирует и описывает содержание, а другие языки, такие как JavaScript, добавляют динамичность и интерактивность.

С точки зрения SEO (поисковой оптимизации), правильное использование HTML также имеет большое значение. Поисковые системы, такие как Google, используют HTML для индексации и анализа содержания веб-страниц. Правильная разметка, использование заголовков, атрибутов «alt» для изображений и структурированных данных могут помочь улучшить видимость сайта в поисковых системах и привлечь больше трафика.

В заключение, HTML – это основополагающий язык разметки, который лежит в основе большинства веб-сайтов и приложений. Понимание его структуры, элементов и атрибутов является необходимым для любого, кто хочет заниматься веб-разработкой. HTML не только позволяет создавать статические страницы, но и служит основой для более сложных и интерактивных веб-приложений в сочетании с CSS и JavaScript. С постоянным развитием технологий и стандартов, HTML остается актуальным и важным инструментом для создания современного веб-контента. Понимание и умение работать с HTML открывает двери к множеству возможностей в мире веб-разработки и дизайна.

2. CSS. Далее рассмотрим следующий инструмент разработки, без которого создание современных сайтов уже не представляется возможным – CSS (Cascading Style Sheets, или Каскадные таблицы стилей). CSS является неотъемлемой частью веб-разработки, позволяя создавать стильные и привлекательные веб-страницы. Его основная задача заключается в том, чтобы отделить содержимое документа (HTML) от его представления. Это означает, что с помощью CSS мы можем управлять тем, как элементы отображаются на экране, изменяя их цвет, размер, шрифт, расположение и множество других характеристик.

Одной из ключевых возможностей CSS является его способность обеспечивать адаптивность дизайна. В современном мире, где пользователи обращаются к веб-сайтам с различных устройств, включая смартфоны, планшеты и настольные компьютеры, важно, чтобы веб-страницы выглядели хорошо на всех экранах. С помощью медиа-запросов в CSS разработчики могут изменять стили в зависимости от размера экрана или других характеристик устройства. Это позволяет создавать так называемые "отзывчивые" дизайны, которые автоматически подстраиваются под разные условия просмотра.

CSS также предлагает богатый набор свойств для стилизации элементов. Например, с его помощью можно легко управлять цветами фона, шрифтами, отступами и границами и так далее. CSS позволяет использовать сложные эффекты, такие как тени, градиенты и анимации. С помощью свойств, таких как box-shadow и text-shadow, можно добавлять глубину и объем к элементам, делая их более визуально привлекательными. Анимации, управляемые с помощью CSS, позволяют создавать плавные переходы и эффекты, которые делают взаимодействие с сайтом более увлекательным. Например, можно сделать так, чтобы кнопка изменяла цвет при наведении курсора, что придаёт интерактивность и улучшает пользовательский опыт.

Кроме того, CSS поддерживает концепцию "каскадирования", что означает, что стили могут наследоваться от родительских элементов. Это позволяет разработчикам писать более чистый и поддерживаемый код. Вместо того чтобы повторять одни и те же стили для различных элементов, можно задать стиль на уровне родителя, и все дочерние элементы автоматически унаследуют его. Это значительно упрощает процесс разработки и делает код более организованным.

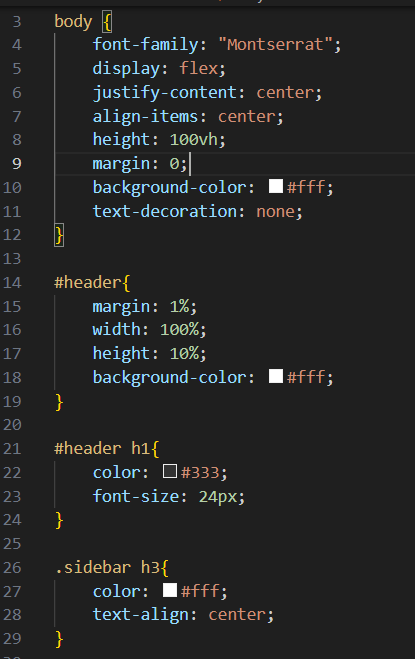
CSS также предоставляет возможность использования различных селекторов, которые позволяют точно нацеливаться на элементы, которые необходимо стилизовать. Селекторы могут быть простыми, такими как селекторы по тегам или классам, или более сложными, включая псевдоклассы и псевдоэлементы. Это даёт разработчикам мощные инструменты для создания сложных и уникальных дизайнов.

Еще одним важным аспектом CSS является возможность работы с флексбоксом и грид-системами. Эти технологии позволяют создавать сложные макеты веб-страниц без необходимости использования дополнительных библиотек или фреймворков. Flexbox идеально подходит для создания линейных макетов, где элементы могут адаптироваться к доступному пространству. CSS Grid, с другой стороны, позволяет создавать двухмерные макеты, где элементы могут занимать несколько строк и колонок, что открывает новые горизонты для дизайна.

Помимо этого, CSS активно развивается, и с каждым новым обновлением появляются новые возможности. Например, CSS Variables (или кастомные свойства) позволяют разработчикам определять свои собственные переменные, которые могут использоваться в различных местах стиля. Это делает код более гибким и удобным для работы, особенно в больших проектах.

Не стоит забывать и о том, что CSS тесно интегрирован с JavaScript, что позволяет создавать динамичные и интерактивные веб-приложения. С помощью JavaScript можно изменять стили элементов на лету, реагируя на действия пользователей, такие как клики или прокрутка. Это открывает безграничные возможности для создания уникальных и привлекательных пользовательских интерфейсов.

Таким образом, CSS – это не просто инструмент для стилизации веб-страниц, а мощная технология, которая играет ключевую роль в создании современных веб-приложений. Он позволяет разработчикам создавать красивые, адаптивные и интерактивные интерфейсы, которые делают интернет более привлекательным и удобным для пользователей. С каждым днем возможности CSS становятся всё шире, и его значение в мире веб-разработки только растёт. Поэтому, если вы хотите стать успешным веб-разработчиком, изучение CSS — это обязательный шаг на вашем пути.

  
Рис. 1.2. Пример CSS-кода.

3. Javascript. Третий по счету инструмент разработки клиентской части веб-приложений – это JavaScript – один из самых популярных и мощных языков программирования, который играет ключевую роль в веб-разработке. Он позволяет добавлять интерактивность и динамичность на веб-страницы, превращая статичный HTML и CSS в полноценные веб-приложения. JavaScript работает на стороне клиента, что означает, что код выполняется непосредственно в браузере пользователя, позволяя мгновенно реагировать на действия пользователя без необходимости перезагрузки страницы.

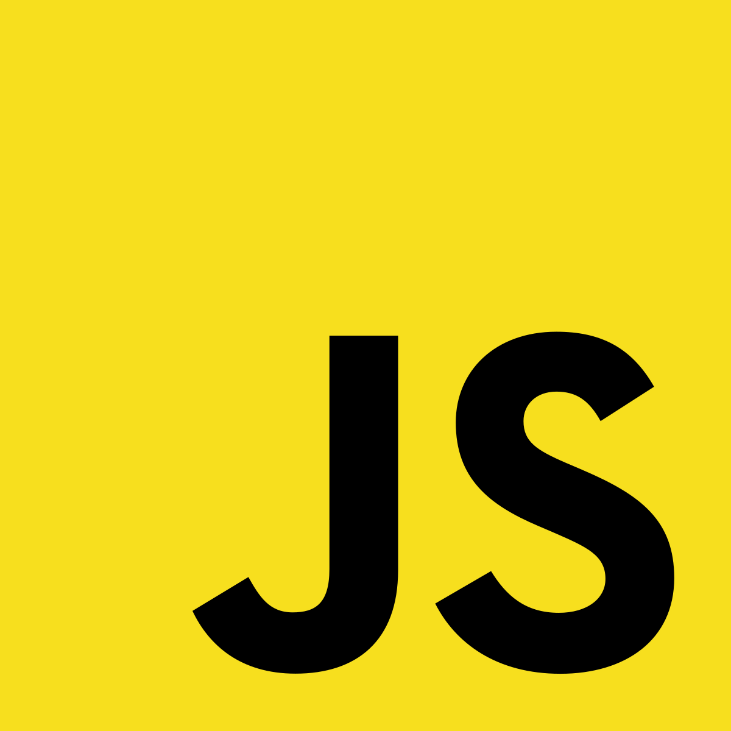


Рис. 1.3. Логотип Javascript.

Одной из основных возможностей JavaScript является манипуляция с элементами DOM (Document Object Model). Это позволяет разработчикам изменять содержимое, структуру и стили веб-страницы в реальном времени. Например, при нажатии кнопки можно изменить текст, скрыть или показать элементы, а также добавлять новые элементы на страницу. Это делает пользовательский интерфейс более интерактивным и отзывчивым.

JavaScript также поддерживает асинхронное программирование, что позволяет выполнять задачи, не блокируя основной поток выполнения. Это особенно полезно при работе с запросами к серверу, когда нужно получить данные без перезагрузки страницы. С помощью таких технологий, как AJAX (Asynchronous JavaScript and XML) и Fetch API, разработчики могут легко загружать данные с сервера и обновлять содержимое страницы, улучшая пользовательский опыт.

Современные веб-приложения часто используют библиотеки и фреймворки на основе JavaScript, такие как React, Angular и Vue.js. Эти инструменты упрощают разработку сложных интерфейсов, позволяя создавать компоненты, которые можно переиспользовать в разных частях приложения. Например, React позволяет создавать пользовательские интерфейсы в виде компонентов, что делает код более организованным и поддерживаемым.

JavaScript также поддерживает объектно-ориентированное программирование (ООП), что позволяет разработчикам создавать сложные структуры данных и организовывать код более эффективно. С помощью классов и объектов можно моделировать реальные сущности, что упрощает разработку и поддержку приложений. Это особенно важно в больших проектах, где структура кода может стать сложной.

Еще одной мощной возможностью JavaScript является работа с событиями. События — это действия, которые происходят в браузере, такие как клики, нажатия клавиш, прокрутка и загрузка страницы. JavaScript позволяет отслеживать эти события и реагировать на них, что делает интерфейсы более интерактивными. Например, можно создать выпадающее меню, которое открывается при наведении курсора или клике на элемент.

Кроме того, JavaScript активно развивается, и с каждым новым обновлением языка появляются новые функции и возможности. Например, ES6 (ECMAScript 2015) представил множество новых синтаксических конструкций, таких как стрелочные функции, шаблонные строки и деструктуризация, которые делают код более лаконичным и читабельным. Также появились возможности работы с асинхронным кодом, такие как промисы и async/await, которые упрощают обработку асинхронных операций.

JavaScript также может использоваться на стороне сервера с помощью платформы Node.js. Это позволяет разработчикам писать серверный код на том же языке, что и клиентский, что упрощает процесс разработки и позволяет использовать одни и те же библиотеки и инструменты как на клиенте, так и на сервере. Node.js идеально подходит для создания высокопроизводительных веб-приложений и API, которые могут обрабатывать множество одновременных запросов.

В заключение, JavaScript — это не просто язык программирования, а мощная технология, которая позволяет создавать динамичные, интерактивные и отзывчивые веб-приложения. Его возможности делают его незаменимым инструментом для веб-разработчиков, и изучение JavaScript является обязательным шагом для всех, кто хочет стать профессионалом в этой области. С каждым днем значение JavaScript только растет, и его влияние на мир веб-разработки продолжает усиливаться.

4. React. Также в современной веб-разработке часто используются различные библиотеки и фреймворки для Javascript, такие, как например: React, Vue, Angular, Svelte и другие. Современные разработчики редко когда обходятся без этих инструментов, так как они значительно упрощают процесс создания сложных и интерактивных веб-сайтов, делая его более организованным и эффективным.

Одним из главных преимуществ фреймворков является возможность разбивать интерфейс на переиспользуемые компоненты. Это значит, что разработчики могут создавать небольшие, независимые части интерфейса, которые легко комбинировать и повторно использовать в разных частях приложения. Такой подход не только упрощает код, но и делает его более понятным и легким для поддержки. Например, если нужно внести изменения в какой-то компонент, это можно сделать в одном месте, и изменения автоматически отразятся во всех местах, где этот компонент используется.

Кроме того, фреймворки предлагают мощные инструменты для управления состоянием приложения. В современных веб-приложениях часто требуется отслеживать и обновлять данные в ответ на действия пользователя, такие как клики или заполнение форм. Фреймворки, такие как React, используют концепции, позволяющие эффективно управлять состоянием, что делает интерфейс более отзывчивым и динамичным. Это особенно важно для интерактивных приложений, где пользователи ожидают мгновенной реакции на свои действия.

Еще одним значительным преимуществом является оптимизация производительности. Например, React использует виртуальный DOM, который позволяет минимизировать количество операций с реальным DOM. Это делает обновление интерфейса более быстрым и эффективным, поскольку изменения сначала применяются к виртуальной версии, а затем синхронизируются с реальной. Это значительно ускоряет работу приложения и улучшает пользовательский опыт.

Работа с асинхронными данными также становится проще с использованием фреймворков. Веб-приложения часто взаимодействуют с внешними API для загрузки данных, и фреймворки предлагают удобные инструменты для выполнения таких запросов. Например, в Vue можно легко использовать встроенные функции для работы с асинхронными запросами, что позволяет разработчикам сосредоточиться на логике приложения, а не на сложных технических деталях.

Нельзя забывать и о поддержке сообществом. Популярные фреймворки имеют обширные экосистемы, в которых разработчики могут найти множество готовых решений для распространенных задач. Это включает в себя библиотеки для маршрутизации, управления состоянием, стилизации компонентов и многое другое. Такой доступ к ресурсам и инструментам значительно ускоряет процесс разработки и позволяет избежать повторной реализации уже существующих решений.

Наконец, фреймворки обеспечивают лучшую поддержку тестирования. Они предоставляют инструменты, которые упрощают написание тестов для компонентов, что способствует созданию более надежных приложений. Это особенно важно в условиях, когда необходимо гарантировать высокое качество кода и минимизировать количество ошибок.

Хотя можно разрабатывать веб-сайты и без использования фреймворков и библиотек, но это будет значительно сложнее и менее эффективно. Современные фреймворки и библиотеки не только упрощают процесс разработки, но и делают его более структурированным и организованным. Они стали стандартом в веб-разработке, позволяя создавать качественные и отзывчивые приложения, которые отвечают требованиям пользователей и современным технологическим стандартам.

При разработке данного сайта выбор пал на React, так как это самая популярная, простая в освоении и поддерживаемая большим сообществом библиотека для Javascript.

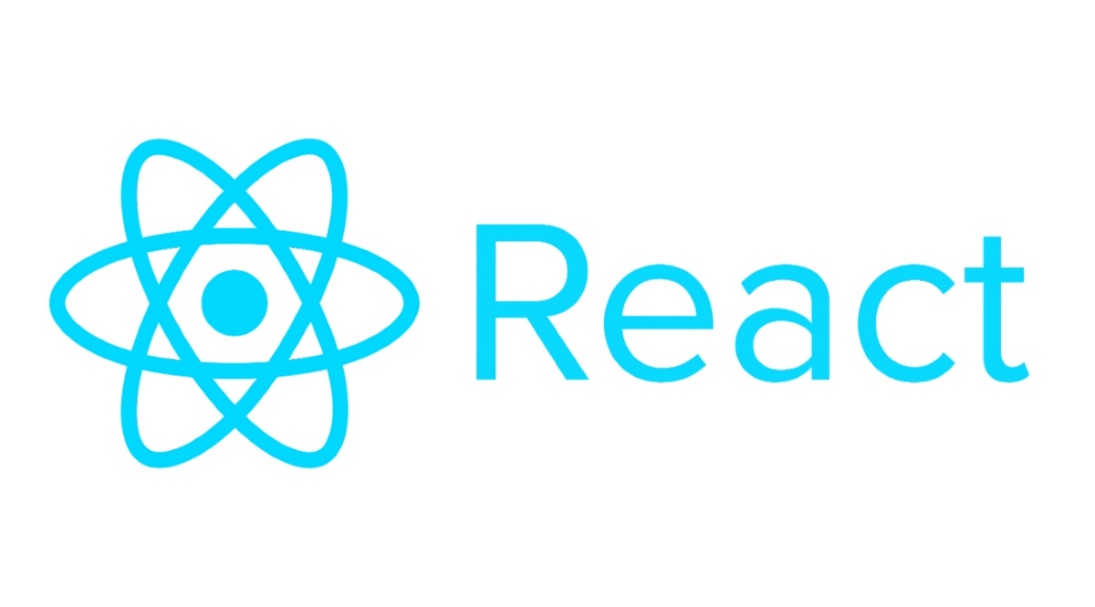


Рис. 1.4. Логотип React.

Для разработки клиентской части сайта использовался текстовый редактор Visual Studio Code – один из самых популярных, если не самый популярный, редактор кода с большим количеством преимуществ, таких как: интуитивно понятный интерфейс, возможность установки расширений, встроенная система отладки, поддержка «умных подсказок» IntelliSense, большое сообщество разработчиков и так далее.

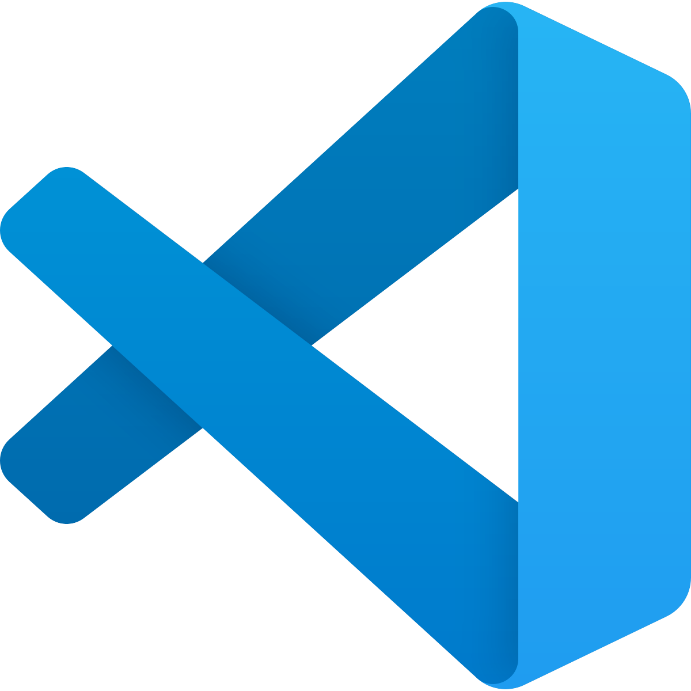


Рис. 1.5. Логотип Visual Studio Code.

* + 1. Инструменты разработки серверной части приложения.

1. Язык программирования. C# — это современный, объектно-ориентированный язык программирования, созданный и развиваемый компанией Microsoft. Он является неотъемлемой частью платформы .NET и используется для разработки настольных, мобильных, облачных и веб-приложений. В рамках данного проекта C# выступает основным языком для реализации серверной части сайта научно-практической конференции.

C# сочетает в себе строгую типизацию, выразительный синтаксис, высокую безопасность и богатую экосистему. Это делает его идеальным языком для реализации сложной логики, работы с базой данных, обработки пользовательских запросов, авторизации и других важных компонентов веб-приложения. Он поддерживает все современные парадигмы: объектно-ориентированное программирование, обобщения, обработку событий, асинхронность, лямбда-выражения, работу с потоками и коллекциями.

Одним из важнейших преимуществ C# является его безопасность типов. В отличие от динамически типизированных языков, таких как JavaScript или Python, C# позволяет обнаружить множество ошибок ещё на этапе компиляции. Это особенно важно при создании корпоративных или учебных решений, где стабильность и предсказуемость поведения системы имеют критическое значение.

Асинхронное программирование на C# реализовано через ключевые слова async и await, что позволяет эффективно обрабатывать параллельные запросы, не блокируя потоки. Это важно в веб-приложениях, где большое количество пользователей может одновременно обращаться к API. Использование асинхронности в методах контроллеров позволяет обрабатывать множество запросов одновременно без потери производительности.

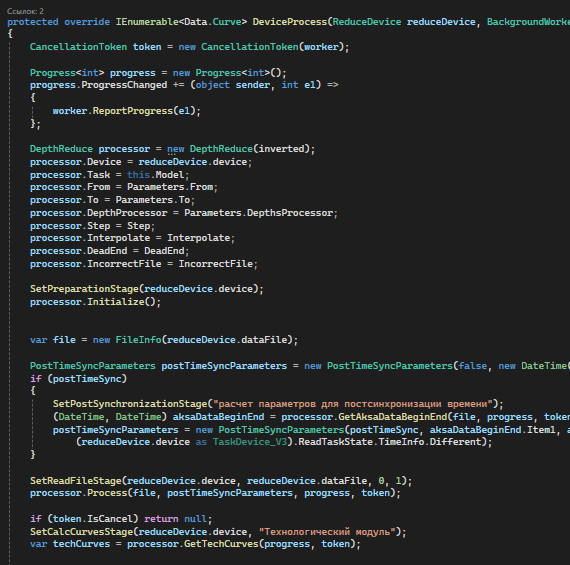


Рис. 1.6. Пример C# кода.

Кроме того, язык C# активно развивается. С каждым новым релизом добавляются новые возможности, упрощающие разработку. Например, в последних версиях появились record-типы, улучшенная работа с pattern matching, улучшенная интерполяция строк, сокращённая инициализация свойств и многое другое. Всё это делает код более читаемым и удобным в сопровождении.

В контексте сравнения с альтернативами можно отметить, что Java предоставляет схожие возможности, но имеет более громоздкий синтаксис. Python, хотя и проще в освоении, не обеспечивает такой же строгой типизации и не так хорошо подходит для создания производительных веб-API. JavaScript и Node.js хороши для фронтенда и лёгких серверных задач, но не обеспечивают уровня архитектурной строгости и встроенной поддержки безопасности, как это реализовано в C#.

Благодаря тесной интеграции с платформой .NET, C# позволяет использовать такие мощные инструменты, как Entity Framework Core, ASP.NET Core, библиотеки для авторизации, логирования, валидации и тестирования. Это значительно упрощает процесс разработки и делает C# универсальным языком для создания качественного и безопасного серверного кода.

2. ASP.NET Core. ASP.NET Core — это современный, высокопроизводительный, кроссплатформенный фреймворк для создания веб-приложений и API, разработанный Microsoft. Он является преемником классического ASP.NET и значительно превосходит его как по производительности, так и по архитектурной гибкости. В данном проекте ASP.NET Core используется как основа серверной части приложения, предоставляя возможности маршрутизации, работы с HTTP-запросами, внедрения зависимостей, конфигурации, логирования и многого другого.

Одна из сильных сторон ASP.NET Core — это его модульность. Разработчик сам выбирает, какие компоненты использовать. Это позволяет собирать минимально необходимую конфигурацию без включения лишнего функционала. Фреймворк поддерживает шаблоны MVC (Model-View-Controller), Razor Pages и REST API. В проекте реализован именно API-подход, поскольку он позволяет разделить клиентскую и серверную части и создать более гибкую систему.

ASP.NET Core поддерживает внедрение зависимостей (Dependency Injection) «из коробки», что способствует созданию тестируемого, масштабируемого и структурированного кода. Это позволяет отделить контроллеры от бизнес-логики, логики от репозиториев, и реализовать принципы чистой архитектуры (Clean Architecture), применённые в проекте.

Дополнительно ASP.NET Core предоставляет поддержку конфигурации из различных источников: файлов, переменных среды, секретов и т.д. Это делает систему удобной при развертывании в разных средах (локально, на сервере, в облаке).

Сравнивая с аналогами, например, Express.js на Node.js, Django на Python или Laravel на PHP, можно отметить, что ASP.NET Core предлагает лучшую производительность, особенно под Windows. Он имеет лучшую поддержку многопоточности, асинхронных вызовов, и идеально интегрируется с C# и Visual Studio.



Рис. 1.7. Логотип ASP .NET Core.

Также стоит отметить встроенные возможности для обработки ошибок, фильтрации запросов, использования middleware-компонентов, настройки CORS и других задач, типичных для современного веб-приложения.

3. Авторизация и безопасность: JWT и ASP.NET Identity. Для обеспечения защищённого доступа к данным, разделения прав пользователей и надёжного входа в систему в проекте была реализована полноценная система аутентификации и авторизации, основанная на технологиях ASP.NET Identity и JSON Web Token (JWT).

ASP.NET Core Identity — это встроенный фреймворк для управления пользователями, ролями, паролями и входом в систему. Он предоставляет готовые модели пользователей, механизмы хэширования паролей, базу для расширения профилей и интеграцию с внешними поставщиками (Google, Microsoft и др.).

JWT (JSON Web Token) представляет собой компактный формат передачи данных между клиентом и сервером. Он используется для выдачи токена при успешной авторизации. Токен содержит в себе зашифрованную информацию о пользователе и его правах доступа. В отличие от сессионной модели, JWT не требует хранения состояния на сервере, что делает систему масштабируемой и более производительной.

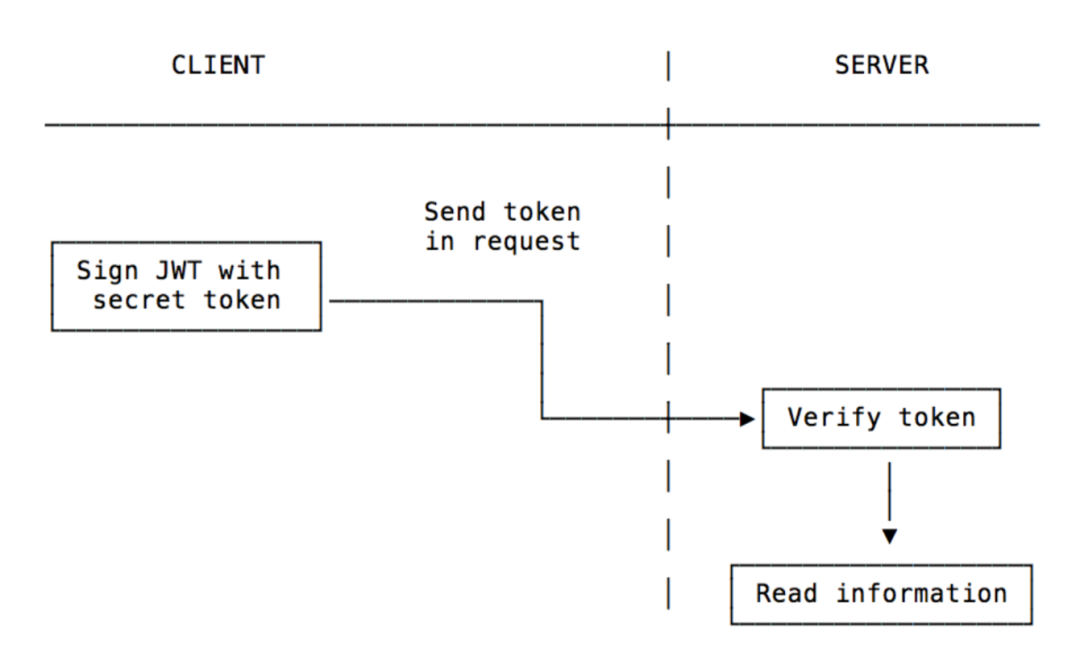
При регистрации пользователь вводит логин, email и пароль. Сервер хеширует пароль и сохраняет данные в базе. При входе генерируется токен, который клиент хранит и передаёт в заголовке Authorization при каждом последующем запросе. Сервер проверяет подпись токена и, если она корректна, разрешает выполнение действия. 

Рис. 1.8. Схема аутентификации JWT.

Конфигурация безопасности выполняется в файле Program.cs и включает настройку схемы аутентификации, конфигурацию ключа подписи, срок действия токена и правила авторизации (например, доступ только для администраторов или модераторов).

Альтернативные подходы, такие как OAuth 2.0, OpenID Connect или куки-аутентификация, тоже поддерживаются, но для REST API JWT является наиболее удобным и универсальным решением.

В проекте реализовано разграничение прав: обычные пользователи могут просматривать и редактировать только свои заявки, а модераторы и администраторы — работать с заявками всех пользователей. Это позволяет обеспечить безопасность и конфиденциальность информации.

4. PostgreSQL. Для хранения и обработки данных участников, заявок и другой информации в проекте используется реляционная система управления базами данных PostgreSQL. Это мощная, надёжная и масштабируемая СУБД с открытым исходным кодом, активно применяемая в промышленной разработке.

На этапе прототипирования использовалась легковесная база SQLite. Однако для финальной версии было принято решение о переходе на PostgreSQL, поскольку она обеспечивает высокую производительность, расширяемость, поддержку сложных запросов и полную совместимость с требованиями к надёжной серверной БД.

PostgreSQL поддерживает ACID-транзакции, полнотекстовый поиск, индексацию, работу с JSON и XML, хранимые процедуры, триггеры и другие инструменты. Это позволяет реализовать сложную бизнес-логику на стороне БД, если потребуется.

Доступ к базе реализован через Entity Framework Core (EF Core) — современную ORM-библиотеку от Microsoft. EF Core позволяет разработчику взаимодействовать с таблицами как с обычными C#-объектами, а запросы к БД писать с помощью LINQ. Это упрощает код, делает его более читаемым и уменьшает количество ошибок, связанных с ручным написанием SQL-запросов.

В проекте реализованы репозитории, отвечающие за работу с пользователями, заявками и файлами. Использование EF Core позволяет легко добавлять миграции, изменять структуру БД и применять обновления без потери данных.

PostgreSQL была выбрана вместо MySQL, SQL Server или Oracle, поскольку она сочетает надёжность, высокую скорость и открытый исходный код. Кроме того, она имеет обширную документацию, широкое сообщество и прекрасно работает на всех популярных ОС.



Рис. 1.9. Логотип PostgreSQL.

5. Visual Studio. Для реализации серверной части веб-приложения был выбран мощный и универсальный инструмент — интегрированная среда разработки (IDE) Visual Studio, разработанная корпорацией Microsoft. Visual Studio является основным инструментом при создании проектов на платформе .NET, включая веб-приложения на языке C#, и предлагает богатый набор функций для разработки, отладки, тестирования и развёртывания программного обеспечения.

Одним из важнейших преимуществ Visual Studio является наличие интеллектуальной системы подсказок IntelliSense, которая значительно ускоряет написание кода и снижает количество ошибок. Эта функция предлагает автозаполнение кода, подсказывает имена методов, классов, свойств и переменных, а также отображает документацию к методам прямо в редакторе. Это особенно важно при работе с большим количеством сторонних библиотек, интерфейсов и моделей, как это реализовано в проекте.

Встроенная система отладки Visual Studio позволяет разработчику гибко анализировать поведение приложения, устанавливая точки останова (breakpoints), отслеживая значения переменных и анализируя стек вызовов. Это даёт возможность быстро находить и устранять ошибки в логике приложения, что особенно важно при работе с авторизацией, регистрацией пользователей, формами и обработкой запросов к базе данных.

Также Visual Studio предлагает удобные визуальные средства для управления структурой проекта: файловая система отображается в виде дерева, поддерживаются несколько конфигураций сборки, переключение между целевыми платформами (например, Debug и Release), а также интеграция с системами контроля версий, в том числе Git и GitHub. Благодаря встроенным возможностям работы с репозиториями можно выполнять коммиты, отслеживать изменения и управлять ветками проекта, не покидая среды разработки.

Одной из ключевых функций является встроенный менеджер пакетов NuGet, позволяющий подключать внешние библиотеки и зависимости. Например, через NuGet были установлены библиотеки для работы с JWT, Entity Framework Core, PostgreSQL-драйвер, инструменты логирования и валидации. Это позволило легко масштабировать функциональность проекта без необходимости ручной настройки сторонних решений.

Visual Studio также поддерживает проектирование баз данных, просмотр содержимого таблиц и выполнение SQL-запросов прямо из среды, что ускоряет работу с СУБД. В контексте данного проекта это дало возможность напрямую взаимодействовать с SQLite и PostgreSQL при тестировании и настройке структуры базы, генерации миграций и отладки ORM (Entity Framework Core).

В сравнении с другими редакторами, такими как Visual Studio Code, JetBrains Rider или Eclipse, среда Visual Studio предоставляет более тесную интеграцию с платформой .NET, полную поддержку ASP.NET Core, богатую документацию и надёжную производительность. Например, Visual Studio Code требует настройки множества расширений и не имеет таких широких возможностей отладки «из коробки», а JetBrains Rider является платным продуктом, что может быть ограничивающим фактором для студенческих и учебных проектов.

Visual Studio также предоставляет шаблоны проектов, поддерживающие Razor Pages, MVC, API-проекты, что ускоряет начальный этап разработки. Возможность генерировать проекты с предустановленной авторизацией, подключением к БД и настройками безопасности экономит значительное количество времени и устраняет риск ошибок на ранней стадии.

Кроме этого, в Visual Studio встроены средства юнит-тестирования, анализа производительности и рефакторинга, позволяющие обеспечить высокое качество кода и соответствие современным требованиям промышленной разработки.

Таким образом, Visual Studio является незаменимым инструментом для разработки серверной части веб-приложения на платформе ASP.NET Core. Её использование в данном проекте обусловлено высокой степенью интеграции с C#, поддержкой всех необходимых технологий (EF Core, JWT, PostgreSQL и других), мощными средствами отладки и тестирования, а также богатым функционалом, способствующим быстрой и качественной реализации серверной логики, взаимодействия с базой данных и обеспечения безопасности. Выбор этой среды позволил существенно ускорить процесс создания приложения и упростить его поддержку в будущем.

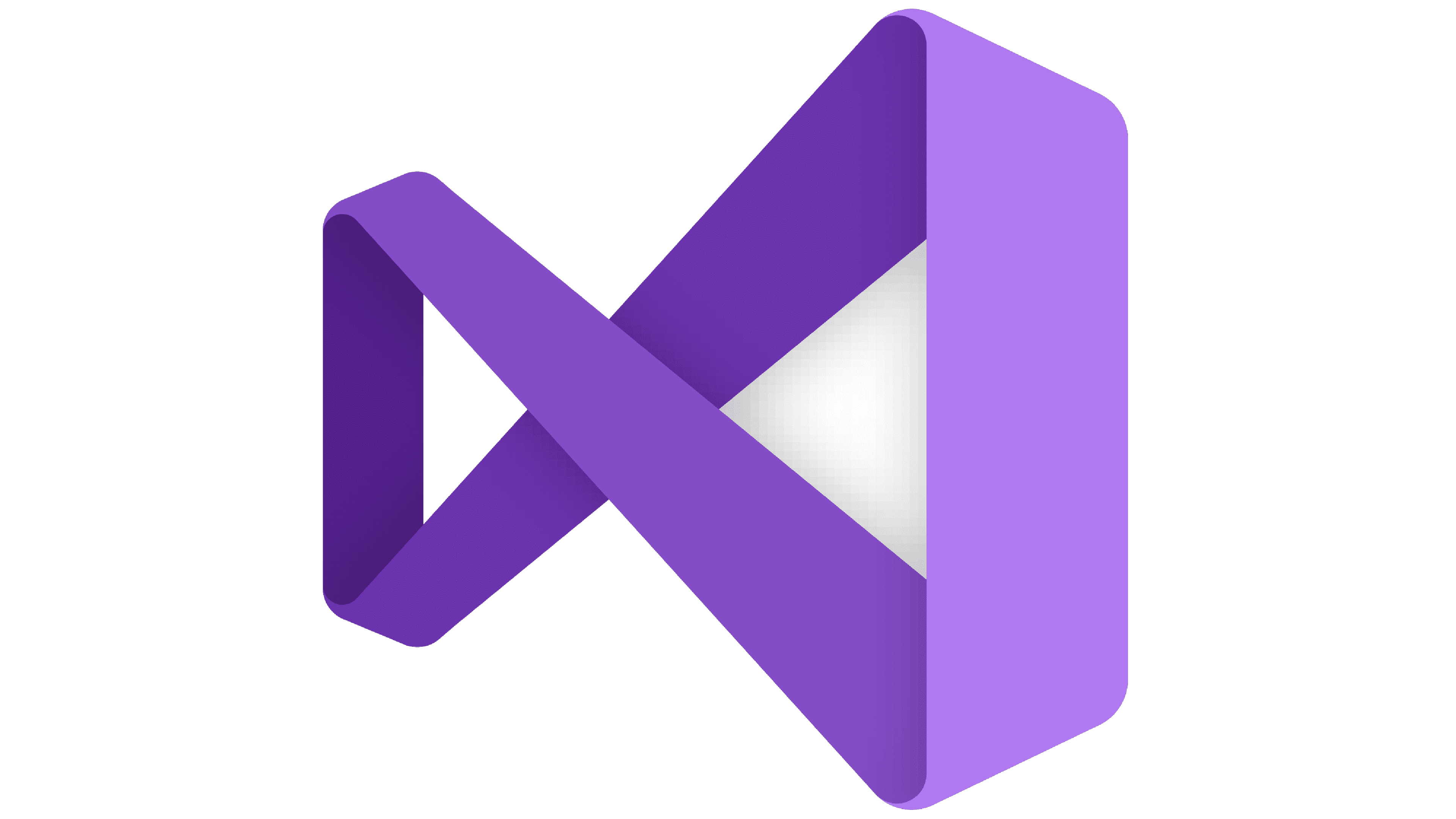


Рис. 1.10. Логотип Visual Studio.

В заключение, выбор C#, ASP.NET Core и PostgreSQL определяет высокую надежность, масштабируемость и безопасность серверной части сайта, а применение JWT и Identity обеспечивает современную систему авторизации. Такая архитектура позволяет эффективно управлять логикой приложения, взаимодействием с клиентом и хранением данных, делая проект готовым к дальнейшему развитию и масштабированию.

## 1.4. Выводы по первой главе

В первой главе выпускной квалификационной работы приведены:

* + Актуальность разработки сайта для научно-практической конференции;
  + Исследование предметной области «Разработка веб-сайтов»;
  + Описание средств разработки клиентской и серверной частей приложения.

В нашем случае для разработки клиентской части приложения был выбран React JS, разработка серверной части происходила с помощью фреймворка ASP.NET Core на языке программирования C#, в качестве баз данных использовалась PostgreSQL, а также редактор кода Microsoft Visual Studio Code и интегрированная среда разработки Microsoft Visual Studio.

# ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ВЕБ-САЙТА

* 1. Проектирование веб-сайта

Перед началом работы были сформулированы функциональные и нефункциональные требования к сайту и был определен приоритет этих требований: какие-то возможности должны быть реализованы на сайте обязательно, и как можно быстрее, а какие-то «фишки» можно оставить и на более поздние версии приложения. Из основных функциональных требований можно выделить:

1. Информация о конференции. Сайт должен предоставлять актуальную информацию о конференции, включая дату, место проведения, программу мероприятия и т.д.

2. Регистрация участников. Участники должны иметь возможность зарегистрироваться на конференцию через онлайн-форму. Также необходимо предусмотреть возможность редактирования данных регистрации, например, реализовав личный кабинет участника конференции.

3. Подача заявок на участие. Участники должны иметь возможность подавать заявки на участие с описанием своих докладов.

4. Обновление информации на сайте. Организаторы должны иметь возможность обновлять или добавлять информацию на сайте. Для этого необходимо реализовать возможность администрирования сайта.

5. Контактная форма. Участники должны иметь возможность связаться с организаторами через контактную форму для получения дополнительной информации.

6. Мультиязычность. Из менее приоритетных требований можно отметить возможность реализации мультиязычности на сайте, так как участниками могут быть докладчики из разных стран.

Из нефункциональных требований также можно отметить: высокую производительность, безопасность, удобство использования, доступность, поддержку браузеров, масштабируемость, возможность дальнейшей поддержки и обслуживания и другие.

* 1. . Разработка клиентской части приложения

Сама непосредственно разработка приложения началась с создания макета, прототипа сайта. Просмотрев и проанализировав множество сайтов с похожей тематикой, были созданы и частично сверстаны несколько макетов, среди которых был выбран наиболее удачный с точки зрения дизайна и наиболее удобный для пользователя макет. Таким образом, структура сайта выглядит следующим образом (рис. 2.1.):

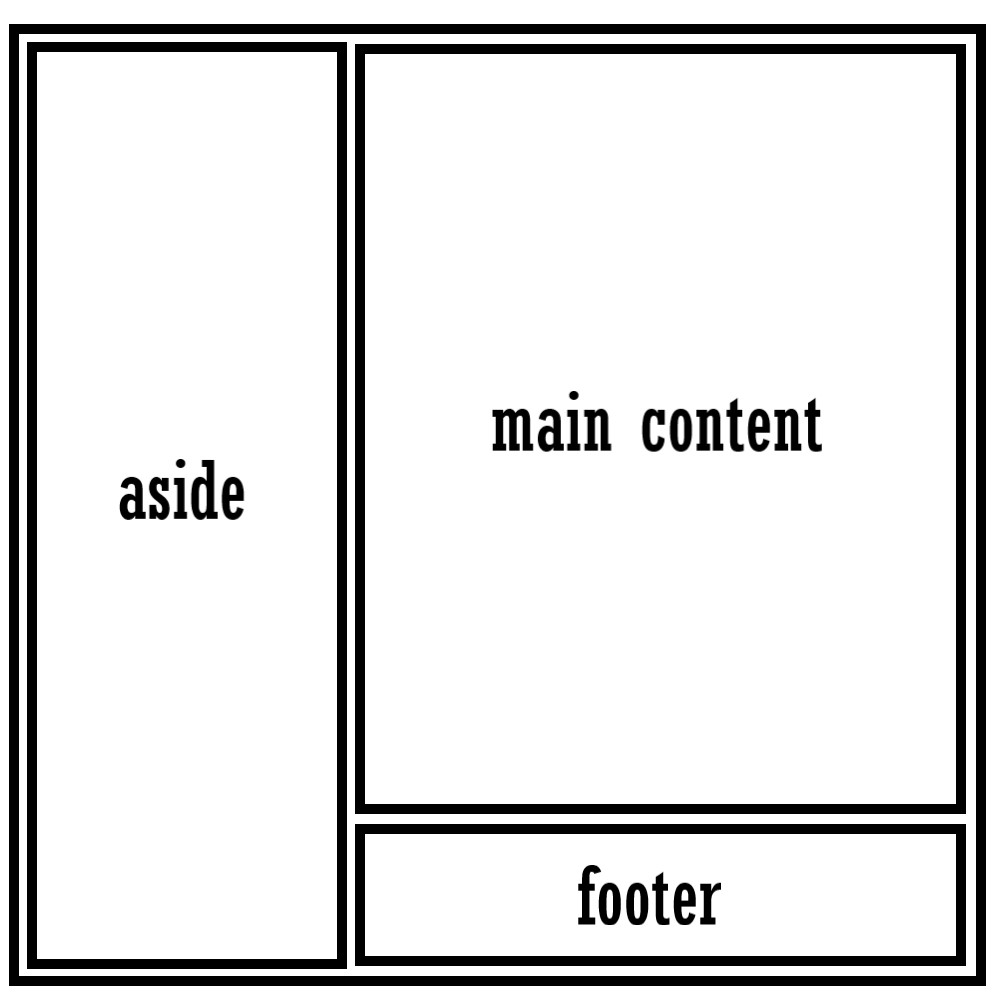


Рис 2.1. Макет сайта.

Для начала разработки клиентской части приложения, необходимо было создать React-проект. Для этого использовался специальный инструмент Vite (рис. 2.2).



Рис. 2.2. Логотип Vite.

Vite – это современный инструмент для разработки веб-приложений, который помогает разработчикам создавать проекты быстрее и удобнее. При работе с React-приложением, обычно нужно много времени ждать, пока все файлы соберутся и обновятся после каждого изменения в коде. Vite решает эту проблему, потому что он умеет очень быстро запускать проект и мгновенно обновлять страницу при изменениях благодаря тому, что использует современные возможности браузеров и умные методы сборки. Это значит, что ты можешь видеть результат своих правок почти сразу, без долгих ожиданий, что значительно ускоряет процесс разработки. Кроме того, Vite поддерживает множество полезных функций из коробки, например, автоматическую работу с модулями, поддержку последних стандартов JavaScript и удобную настройку, что делает его отличным выбором для создания React-приложений. Для создания React-приложения с помощью инструмента Vite, необходимо ввести несколько инструкций в командную строку (рис. 2.3.).

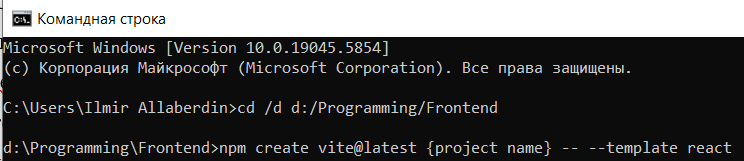


Рис. 2.3. Процесс создания React-приложения через командную строку.

После того, как React-приложение собралось, можно открывать папку с проектом в Visual Studio Code и приступать к верстке.

После всех описанных выше действий, началась верстка основной страницы сайта по макету. Сначала был реализован «sidebar» на левой стороне страницы, который будет включать в себя разделы сайта, при нажатии на которые будет происходить переход к соответствующему разделу сайта по якорной ссылке. Также был разработан блок content, который будет хранить всю информацию сайта. Для этого были созданы 2 файла Sidebar.jsx и Content.jsx, и помещены в компонент MainPage.jsx, который является главной страницей сайта (рис. 2.4.). При загрузке приложения, отображается именно компонент MainPage.

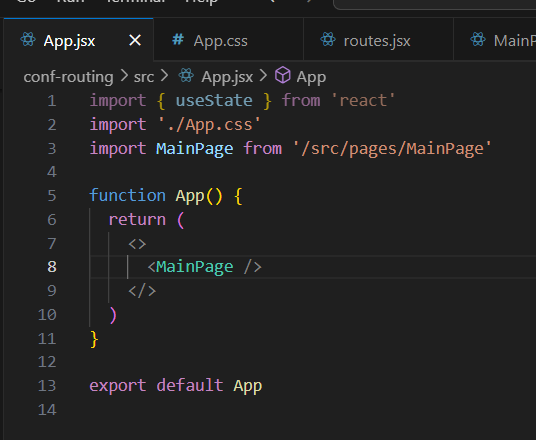


Рис. 2.4. Главный компонент App и помещенный в него компонент главной страницы MainPage.

Расширение jsx – это расширение синтаксиса JavaScript, которое позволяет использовать HTML-подобный синтаксис для описания структуры пользовательского интерфейса в React-компонентах. Иными словами, файлы с расширением jsx являются компонентами. Так как React использует компонентный подход, в приложении всё будет построено на компонентах.  
 Компонентный подход в React — это одна из ключевых концепций этой библиотеки, которая позволяет разбивать интерфейс приложения на независимые и переиспользуемые части, называемые компонентами. Компоненты в React представляют собой самостоятельные блоки, которые могут содержать как логику, так и визуальное представление. Каждый компонент может управлять своим состоянием и получать данные через свойства (props), что делает его гибким и универсальным.

Главное преимущество компонентного подхода заключается в том, что он способствует созданию модульного кода. Это значит, что разработчики могут работать над отдельными частями приложения независимо друг от друга. Например, если у вас есть компонент кнопки, вы можете изменить его стиль или функциональность, не затрагивая другие части приложения. Это значительно упрощает поддержку и расширение кода.

Кроме того, компоненты могут быть легко переиспользованы в разных частях приложения или даже в разных проектах. Это экономит время и усилия, так как вам не нужно каждый раз заново писать один и тот же код. Также, благодаря компонентному подходу, становится проще тестировать и отлаживать приложение, так как разработчики могут сосредоточиться на небольших частях кода.

Еще одним важным аспектом является то, что компоненты могут использоваться для организации иерархии приложения. Вы можете создавать родительские компоненты, которые содержат дочерние, что позволяет структурировать код и делать его более понятным. Это особенно полезно в больших проектах, где важно поддерживать порядок и ясность.

После создания компонентов Sidebar и Content, а также применения к ним первоначальных стилей, мы получили основу сайта, как и было задумано по макету (рис. 2.5.).

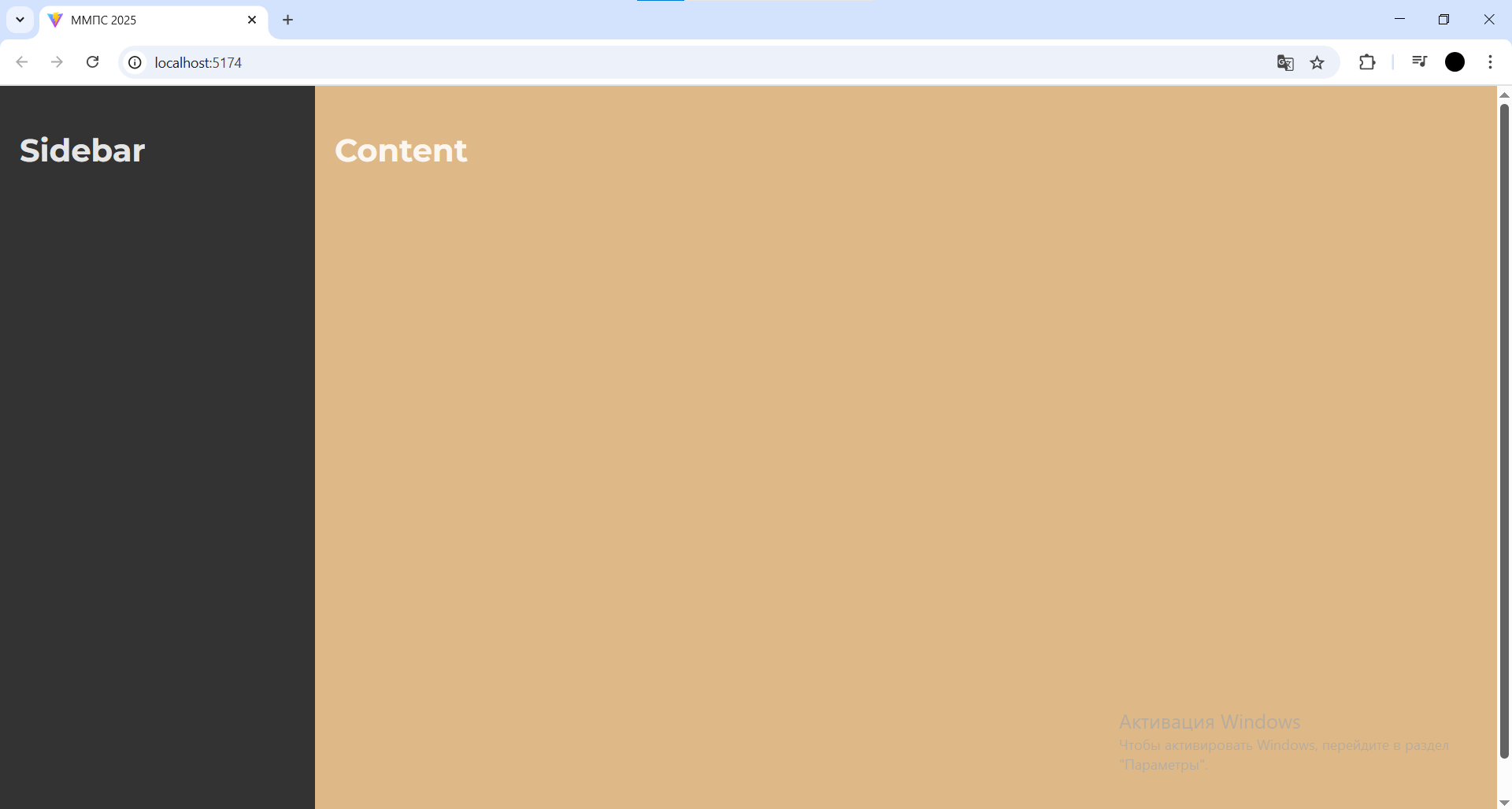


Рис. 2.5. Верстка основных блоков главной страницы сайта согласно макету.

Структура проекта на React представлена на рисунке 2.6. Родительская папка src содержит в себе папку для хранения компонентов – components, папку для хранения страниц сайта – pages, папку для хранения путей для навигации по сайту – routes, папку для хранения стилей всех компонентов – styles, компонент App.jsx, который является главным компонентом в приложении, а также другие необходимые для разработки файлы.

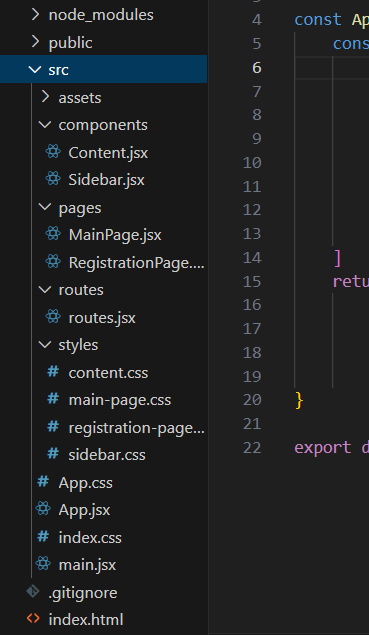


Рис. 2.6. Структура проекта в React.

Компонент Sidebar содержит в себе список из компонентов SidebarElement, каждый из которых является кнопкой перехода к разделам сайта (рис. 2.7.). Sidebar принимает в качестве аргументов название раздела и ссылку в виде id секции, к которой и будет вести при нажатии (рис. 2.8.). При нажатии на элемент с названием «О конференции» мы будем попадать в раздел с id «#about» по якорной ссылке. Аналогично и с другими разделами сайта. На сайте были реализованы разделы «О конференции», «Программа», «Программный комитет», «Организационный комитет», «Регистрация/Подача заявки», «Сборник материалов» и «Контакты». Далее планируется реализация возможности управления этими разделами с помощью создания аккаунта администратора сайта, а именно возможности добавления, редактирования и удаления разделов.

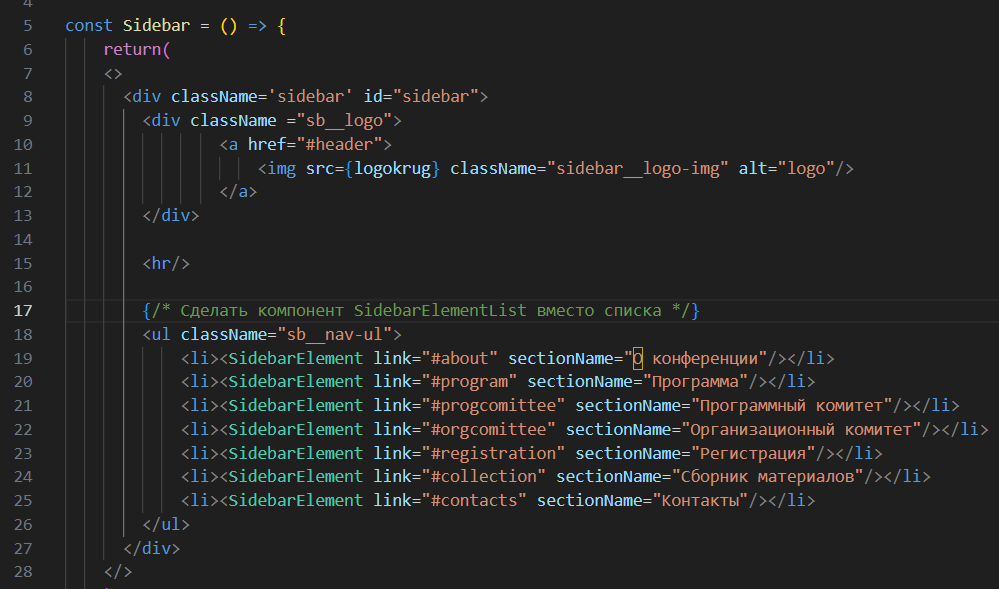


Рис. 2.7. Компонент Sidebar.

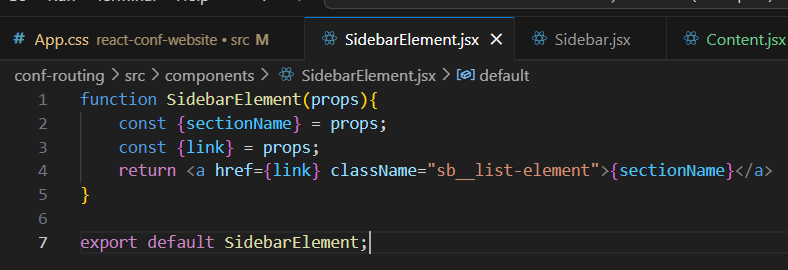


Рис. 2.8. Компонент SidebarElement.

Затем внутри компонента Content была создана шапка сайта, которая включает в себя название конференции, фотографию корпуса Стерлитамакского филиала Уфимского университета науки и технологий, а также некоторые другие элементы. Кроме того, были созданы те самые секции из Sidebar’а, которые содержали различную информацию о конференции. Пример одной из них представлен на рисунке 2.9.

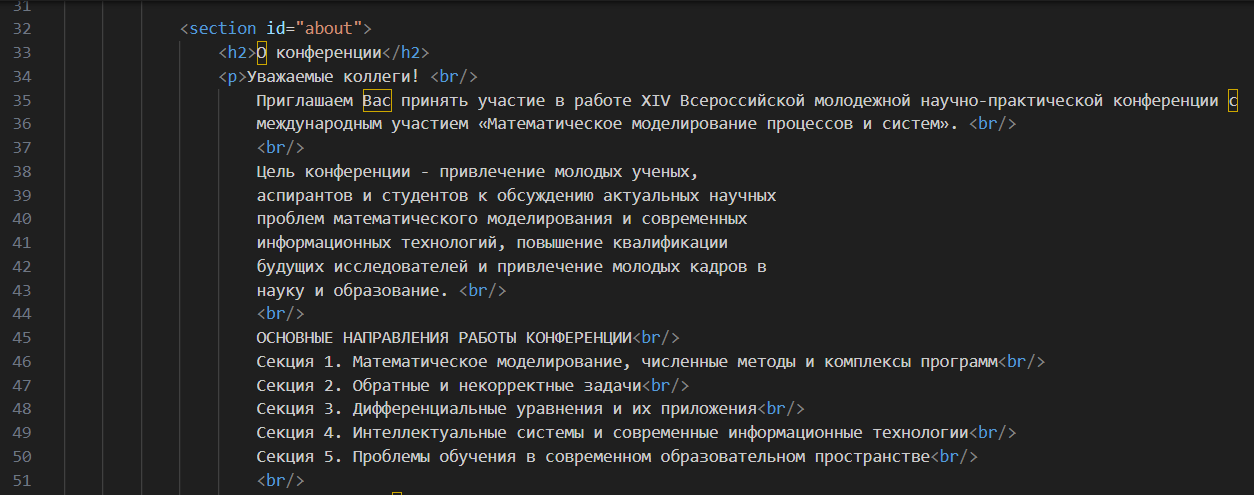


Рис. 2.9. Секция «О конференции»

Также были реализованы формы для регистрации и подачи заявки участников конференции (рис. 2.10.).

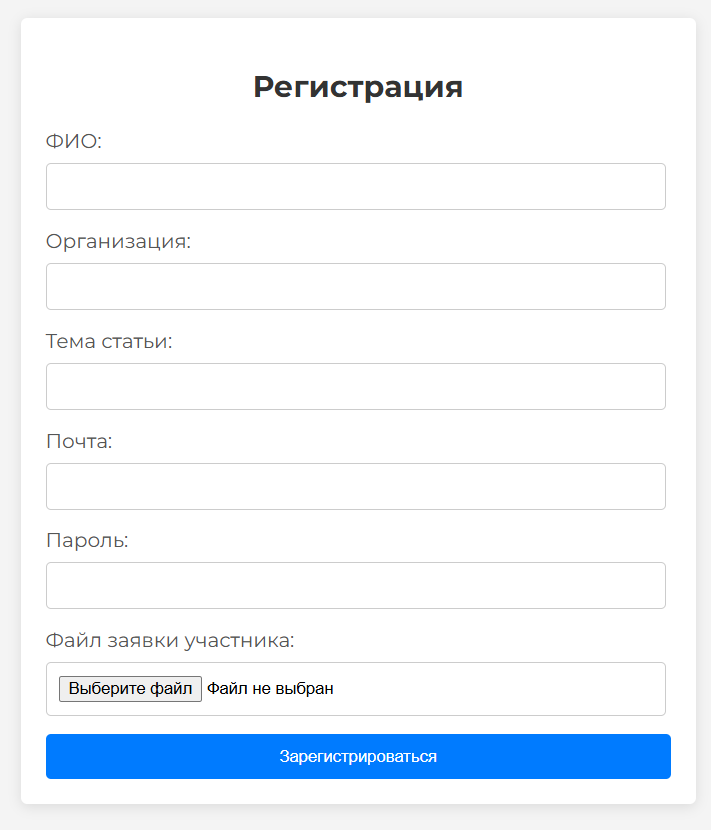


Рис. 2.10. Форма регистрации участника конференции.

В результате, после заполнения каждого раздела сайта информацией и применением стилей, мы получили готовую главную страницу (рис. 2.11.)

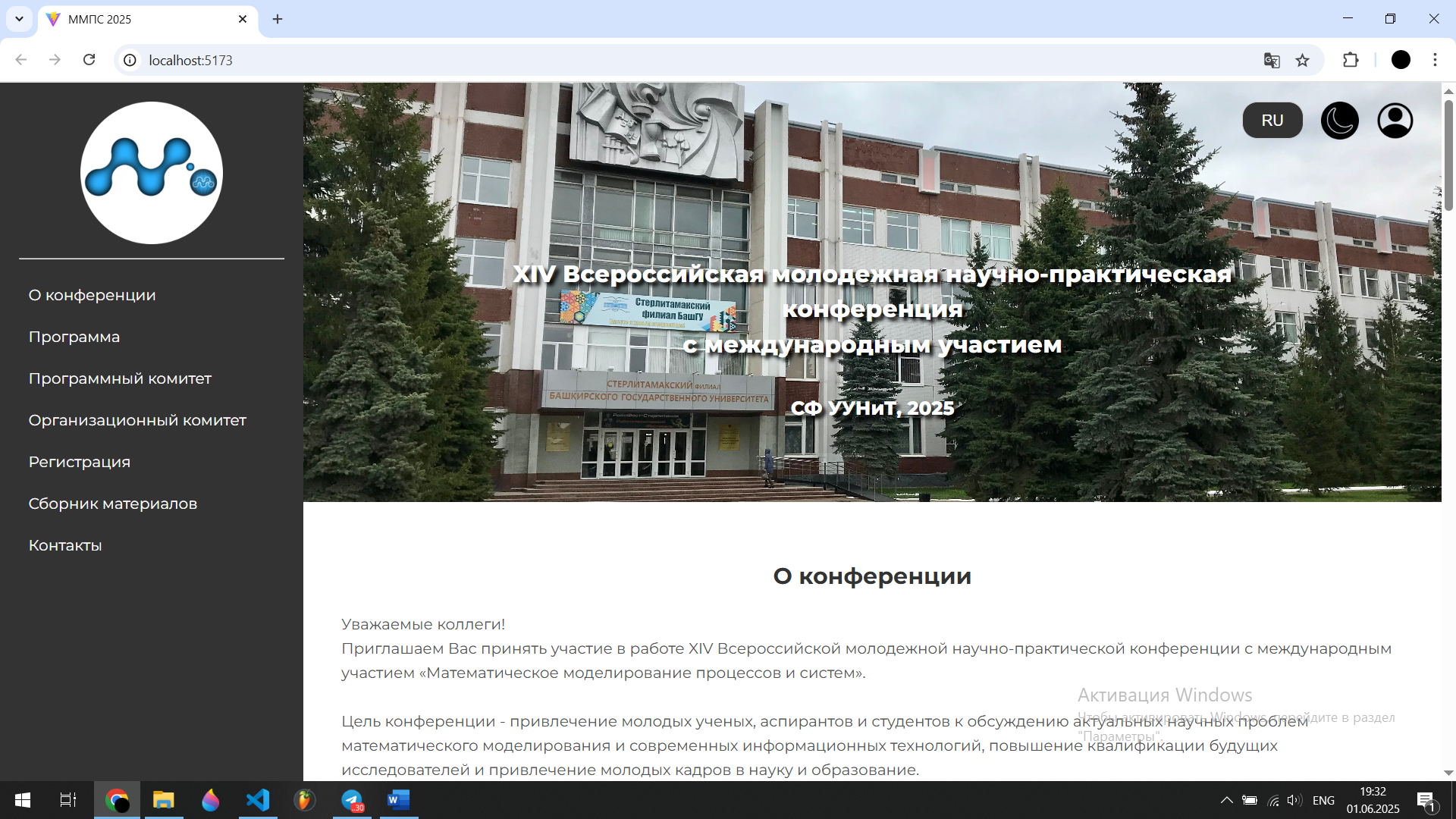


Рис. 2.11. Главная страница сайта.

2.3. Разработка серверной части приложения

### 2.3.1. Принцип чистой архитектуры

Для того чтобы проект мог легко сопровождаться, тестироваться и масштабироваться в будущем, мы с самого начала приняли решение использовать принцип чистой архитектуры (Clean Architecture). Эта архитектура была выбрана по следующим причинам: она обеспечивает строгую изоляцию слоёв приложения, способствует повторному использованию компонентов, упрощает написание тестов и делает проект устойчивым к изменениям во внешней инфраструктуре.

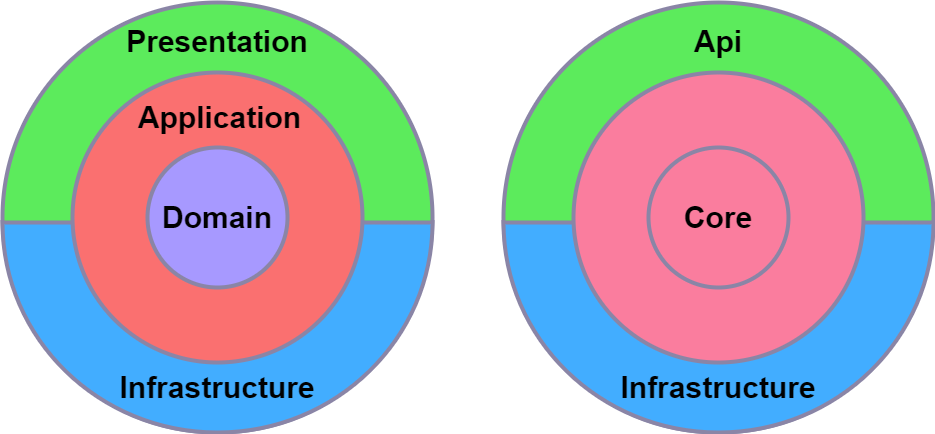


Рис. 2.12. Схема Clean Architecture.

Clean Architecture разделяет проект на независимые слои, взаимодействующие только через интерфейсы. Каждый слой имеет своё назначение и не зависит от реализаций в других слоях. В нашем проекте структура разделена на отдельные проекты, каждый из которых соответствует своей зоне ответственности:

ConferenceApp.Domain – проект, содержащий основные сущности системы (например, участник, администратор, конференция), а также интерфейсы (контракты), определяющие поведение и структуру доступа к данным.

ConferenceApp.Application – проект бизнес-логики. Здесь содержатся DTO (Data Transfer Object), интерфейсы сервисов, обработчики команд и запросов, валидаторы, а также правила взаимодействия между слоями.

ConferenceApp. Infrastructure - отвечает за взаимодействие с внешним миром: базой данных, файловой системой, реализацией репозиториев, отправкой писем и т.д.

ConferenceApp.Web/WebApi - это входная точка веб-приложения. Здесь размещены контроллеры, настройки API, конфигурация middleware и маршрутизация.

Такая модульная структура позволяет каждому проекту развиваться независимо, легко изменяться и тестироваться. Мы также заранее предусмотрели возможность добавления новых модулей без нарушения логики уже работающей системы.

Для лучшего понимания того, как слои взаимодействуют между собой, приведём пример. Пусть пользователь подаёт заявку на участие в конференции:

1. Контроллер AuthController в WebApi получает POST-запрос с данными участника.

2. Контроллер вызывает метод CreateParticipant в сервисе ParticipantService из слоя Application.

3. ParticipantService получает DTO, валидирует его и преобразует в сущность Participant из Domain.

4. Сервис вызывает метод интерфейса IParticipantRepository, описанного в Domain.

5. Реализация этого интерфейса, находящаяся в Infrastructure (ParticipantRepository), сохраняет участника в базу данных через Entity Framework.

Таким образом, каждый слой выполняет строго определённую функцию и знает только об абстракциях нижележащих слоёв. Это позволяет легко заменять или модифицировать реализацию, не затрагивая остальной код.

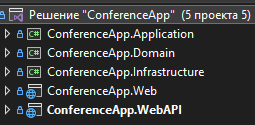


Рис. 2.13. Структура проекта.

### 2.3.2. Создание и заполнение проекта ConferenceApp.Domain

Проект ConferenceApp.Domain играет центральную роль в архитектуре. Он описывает основные сущности, с которыми работает система. Эти сущности соответствуют таблицам в базе данных и моделируют реальные объекты предметной области конференции.

Ключевые сущности:

1. Participant (Участник) — пользователь, подающий заявку на участие в конференции. Содержит поля: Id, FullName, Email, Organization, TitleLecture, Password, ReportFile, ReportFileName. Эти поля обеспечивают полный набор информации, необходимый для регистрации и обработки заявок.
2. Admin (Администратор) — пользователь с расширенными правами. Может просматривать, редактировать и удалять заявки, управлять списками участников и другими административными функциями.
3. Conference (Конференция) — сущность, отражающая информацию о мероприятии (даты проведения, тема, место и т.д.). В текущей реализации представлена в виде базовой модели, но в будущем может быть расширена.

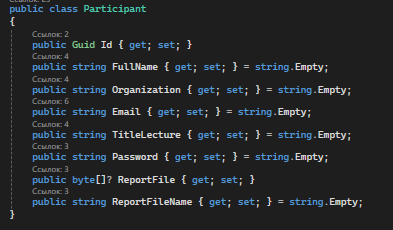


Рис. 2.14. Сущность участника.

Дополнительно добавлены сущности Организационного комитета и Программного комитета, которые планируются к внедрению в следующих версиях проекта. Это позволит сделать систему более гибкой и расширяемой.

Каждая сущность реализована как класс с открытыми свойствами. Кроме того, в проекте определены интерфейсы репозиториев, например IParticipantRepository, которые описывают методы для работы с данными. Реализация этих методов осуществляется в инфраструктурном проекте.

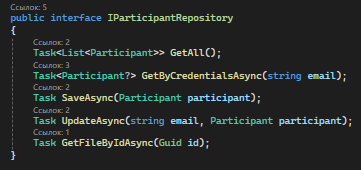


Рис. 2.15. Интерфейс репозитория пользователя.

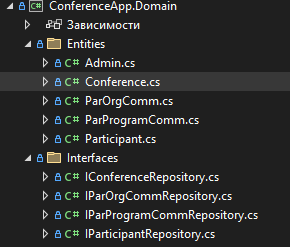


Рис. 2.16. Структура проекта Domain.

### 2.3.3. Создание и заполнение проекта ConferenceApp.Application

Проект Application содержит основную бизнес-логику приложения. Именно здесь сосредоточено взаимодействие между внешним миром (WebApi) и внутренними сущностями (Domain). Компоненты этого проекта не зависят от инфраструктурных технологий и могут быть легко протестированы.

Основные элементы:

DTO — классы, описывающие структуры данных, передаваемых между слоями (например, ParticipantDto, RegisterRequest, LoginResponse). DTO помогают отделить структуру входных/выходных данных от внутренних сущностей.

Интерфейсы сервисов — описывают методы, с помощью которых WebApi-слой будет взаимодействовать с бизнес-логикой. Например, IParticipantService содержит методы для добавления, получения и редактирования участников.

Сервисы — реализации интерфейсов, содержащие бизнес-логику. Сервисы получают зависимости через встроенный в ASP.NET Core механизм внедрения зависимостей (Dependency Injection).

Обработчики команд и запросов — логика для CQRS (разделение чтения и записи). Это повышает читаемость кода и разделяет ответственность.

Валидаторы — классы, использующие библиотеку FluentValidation для проверки входных данных (например, проверка формата email, обязательных полей и т.д.).

Security — папка с реализацией JWT. Здесь хранится логика создания и валидации токенов, а также параметры токена (время жизни, ключ подписи, алгоритмы и т.д.).

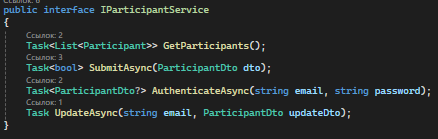


Рис. 2.17. Интерфейс сервиса участника.

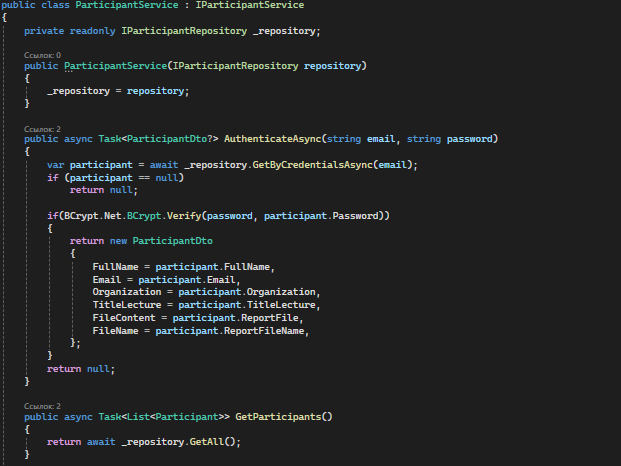
****

Рис. 2.18. Реализация интерфейса сервиса.

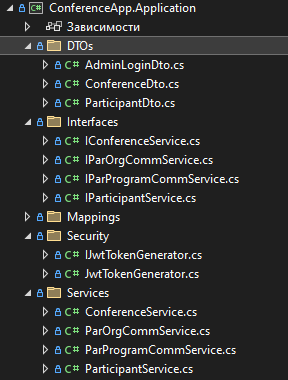


Рис. 2.19. Структура проекта Application.

### 2.3.4. Создание и заполнение проекта ConferenceApp.Infrastructure.

Проект Infrastructure отвечает за технические детали реализации. Здесь находятся классы, взаимодействующие с внешними ресурсами: базой данных, файловой системой, электронной почтой и т.д.

Репозитории — содержат реализацию интерфейсов из Domain. Например, ParticipantRepository реализует IParticipantRepository, выполняя реальные запросы к базе данных с использованием EF Core.

DbContext — основной класс взаимодействия с БД. Содержит DbSet для каждой сущности.

Миграции — механизм EF Core, позволяющий создавать и применять изменения структуры БД на основе моделей.

Поддержка работы с файлами — сохранение и извлечение файлов из файловой системы или базы данных.

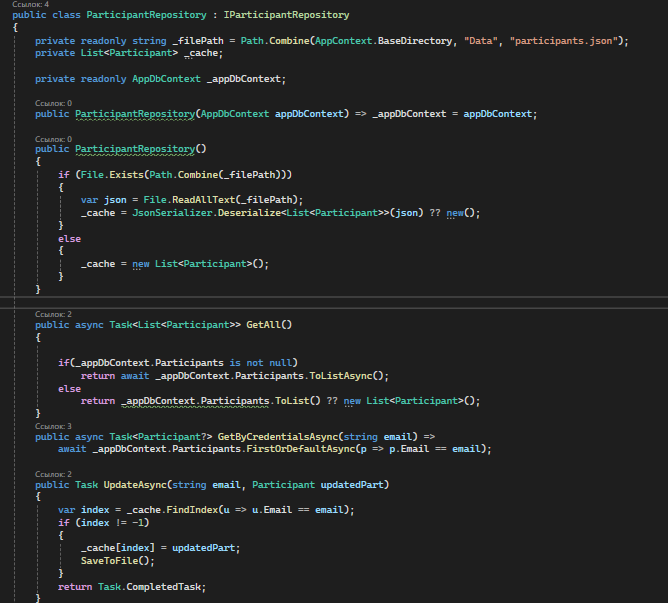


Рис. 2.20. Реализация репозитория участника.

На начальном этапе проекта использовалась встроенная база данных SQLite, которая не требует установки и подходит для тестирования. В финальной версии проекта используется PostgreSQL — надёжная и масштабируемая СУБД с поддержкой ACID-транзакций и полнотекстового поиска.

Для взаимодействия с базой данных используется Entity Framework Core — ORM, позволяющая работать с таблицами через C#-классы. Модели, такие как Admin и Participant, отображаются в таблицы, а все операции с базой выполняются через контекст ApplicationDbContext. Все операции реализованы асинхронно, что повышает производительность при большом числе запросов.

Миграции EF Core позволяют автоматически изменять структуру базы данных при изменении модели. Это обеспечивает согласованность данных между кодом и базой.

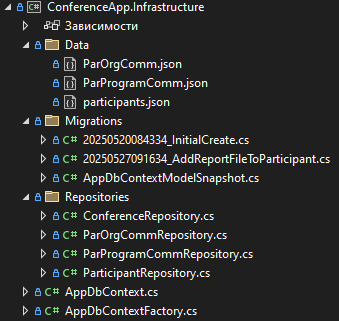


Рис. 2.21. Структура проекта Infrastructure.

### 2.3.5. Создание и заполнение проекта ConferenceApp.WebApi

Проект WebApi — это точка входа для клиента. В нём реализованы контроллеры, обрабатывающие HTTP-запросы и возвращающие ответы. Основной контроллер — AuthController, отвечающий за регистрацию и авторизацию пользователей. При регистрации пользователь передаёт имя, email и пароль, который хэшируется и сохраняется в базе данных. При авторизации происходит проверка введённых данных, и при успешном входе выдается JWT-токен.

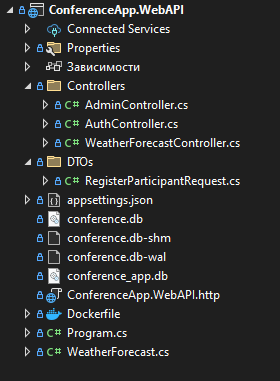


Рис. 2.22. Структура проекта WebApi.

JWT-токен содержит зашифрованные данные о пользователе: ID, email, срок действия и роль. Токен подписывается секретным ключом и возвращается клиенту. В последующих запросах он передаётся в заголовке Authorization, и сервер проверяет его действительность при обращении к защищённым ресурсам.

Контроллер AuthController отвечает за подачу и обработку заявок на участие в конференции. Зарегистрированные пользователи могут отправить заявку, указав ФИО, организацию, email и прикрепив файл с тезисами. Все данные сохраняются в таблицу Participants базы данных. Загрузка файлов осуществляется через форму с типом multipart/form-data, а на сервере файл обрабатывается с использованием интерфейса IFormFile. Файлы могут сохраняться как в папке wwwroot/uploads, так и непосредственно в БД в виде бинарных данных (тип byte[] или BLOB, в зависимости от используемой СУБД). Реализована проверка допустимого размера и расширения файлов, а также генерация уникального имени для исключения конфликтов. Контроллер включает методы для создания новой заявки, получения списка участников и редактирования данных. Предусмотрена базовая валидация: проверка формата email, обязательных полей и уникальности записей. При необходимости можно реализовать функции скачивания или просмотра загруженных файлов.

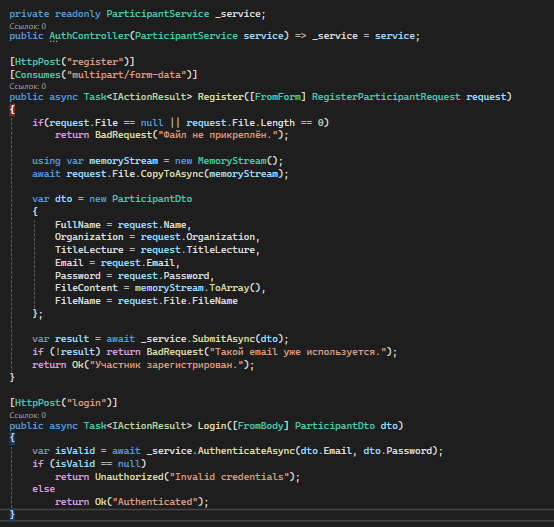


Рис. 2.23. Контроллер Авторизации.

Регистрация пользователей осуществляется через форму, отправляющую данные в формате JSON. Пароли перед сохранением хэшируются. При авторизации введённый пароль сравнивается с сохранённым хэшем. В случае успеха выдается JWT-токен, содержащий зашифрованную информацию и срок действия.

Все функции реализованы в виде REST API. Каждому действию (регистрация, авторизация, подача заявки, получение участников) соответствует свой маршрут и HTTP-метод (GET, POST, PUT и др.).

Для тестирования, отладки и документирования API используется инструмент Swagger, подключённый с помощью пакета Swashbuckle.AspNetCore. Swagger автоматически сканирует контроллеры и действия, аннотированные атрибутами ASP.NET (например, [HttpGet], [HttpPost], [FromBody], [FromForm], [ProducesResponseType]), и формирует интерактивную документацию, доступную по маршруту /swagger. Интерфейс Swagger позволяет разработчикам:

– просматривать все доступные эндпоинты API;

– видеть описание маршрутов, параметров, форматов данных и возможных ответов;

– выполнять запросы напрямую из браузера (без необходимости писать фронтенд);

– тестировать методы с различными входными данными;

– отслеживать коды состояния (200 OK, 400 Bad Request, 401 Unauthorized и др.).

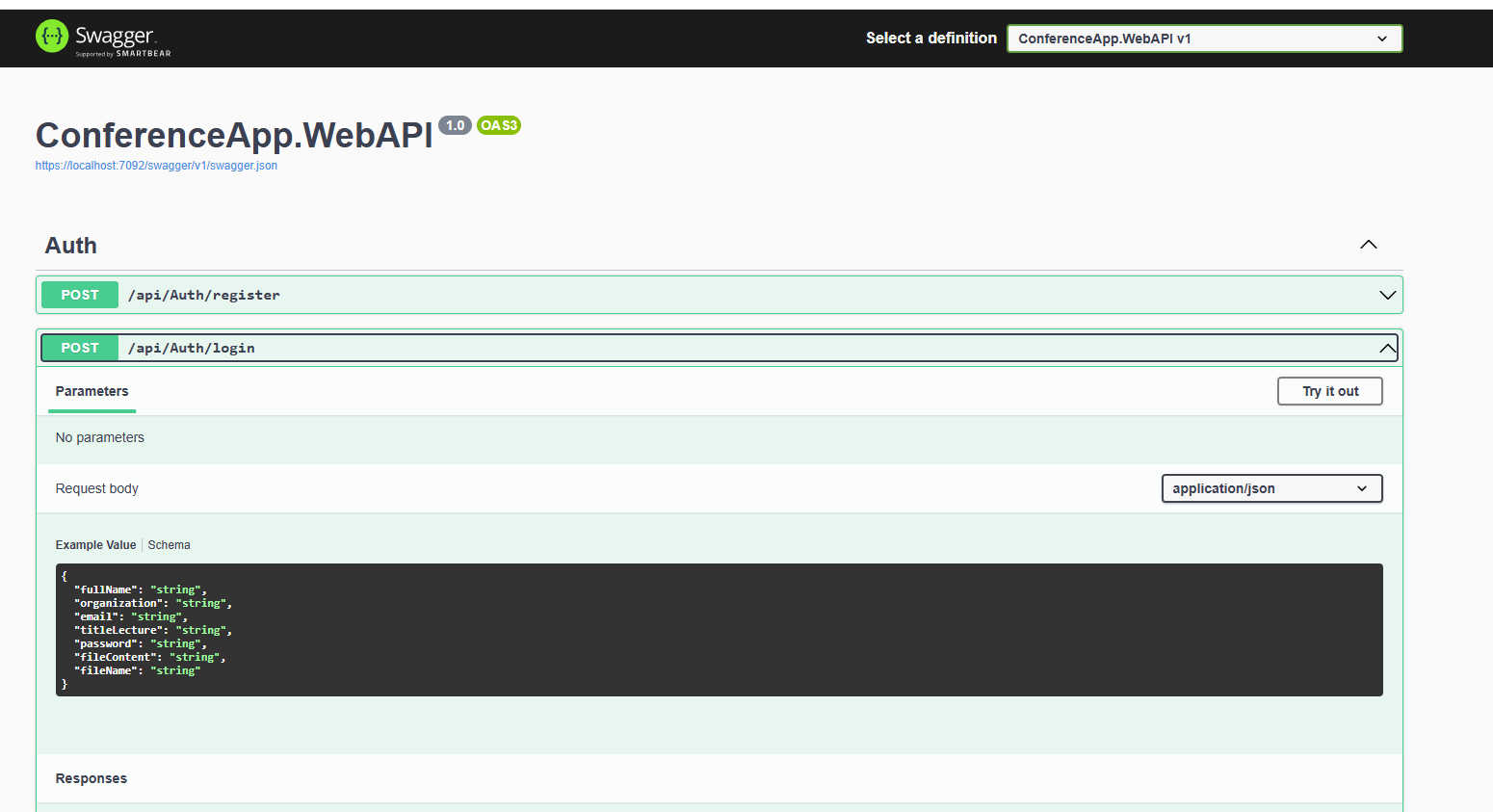
Swagger существенно ускоряет разработку, упрощает отладку и облегчает взаимодействие между разработчиками бэкенда и фронтенда. Также он используется при передаче API-документации сторонним командам и заказчикам. 

Рис. 2.24. Swagger.

Благодаря наглядному интерфейсу и поддержке OpenAPI-спецификации, Swagger является стандартным инструментом для работы с API в ASP.NET Core-приложениях.

### 2.3.6. Связывание клиентской и серверной частей.

Связывание клиентской и серверной частей реализовывалось через обращение к локальному серверу, запущенному на базе ASP.NET Core. В процессе разработки клиент взаимодействовал с сервером по адресу, предоставленному встроенным Kestrel-сервером, например:

https://localhost:7092

Для выполнения конкретных операций к базовому адресу добавлялся маршрут, соответствующий контроллеру и действию. К примеру, для регистрации нового пользователя использовался эндпоинт:

https://localhost:7092/api/auth/register

Этот маршрут связан с контроллером AuthController и методом Register, который обрабатывает входящие данные регистрации.

Пример последовательности взаимодействия между клиентом и сервером:

1. Пользователь переходит на страницу регистрации на фронтенде, где отображается форма ввода данных. Пример формы приведён в разделе 2.2 (Рис. 2.10. Форма регистрации участника конференции).
2. После заполнения формы, данные в формате JSON отправляются на сервер с помощью HTTP POST-запроса по адресу /api/auth/register.
3. На серверной стороне метод Register принимает эти данные, валидирует их и создаёт объект пользователя.

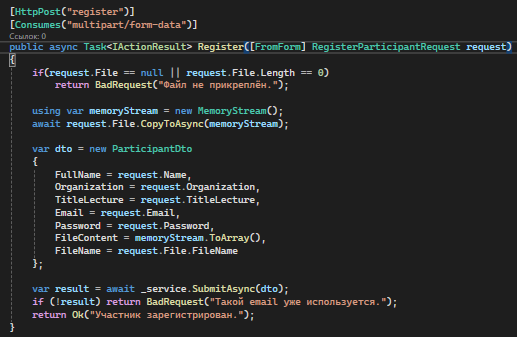


Рис. 2.25. Метод регистрации.

1. После успешной валидации данные сохраняются в базе данных с использованием Entity Framework Core.



Рис. 2.26. Таблица пользователей.

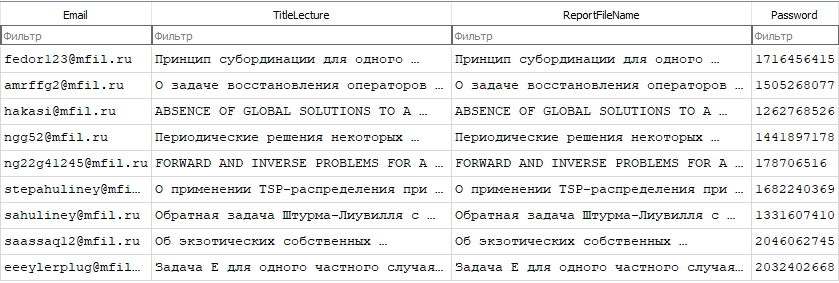


Рис. 2.26. Таблица пользователей (2).

1. При успешной регистрации возвращается ответ об успешной операции, и пользователь может перейти к авторизации.

Таким образом, на этапе разработки обеспечено стабильное

взаимодействие между серверной и клиентской частью приложения. Использование локального сервера позволило эффективно тестировать и отлаживать работу всех компонентов в изолированной среде, оперативно выявлять ошибки и проверять корректность обработки запросов.

В дальнейшем планируется развертывание (деплой) проекта на удалённый хостинг или сервер университета. Это позволит предоставить доступ к системе для всех заинтересованных пользователей, обеспечить централизованное администрирование и контроль, а также подготовить проект к эксплуатации в условиях реального использования.

* 1. Выводы по второй главе

Во второй главе выпускной квалификационной работы были описаны:

* + Процесс разработки клиентской части приложения;
  + Процесс разработки серверной части приложения;
  + Процесс связывания клиентской и серверной частей друг с другом.

Результатом работы стал полноценный сайт научно-практической конференции.

# ЗАКЛЮЧЕНИЕ

В данной выпускной работе была разработана веб-система для проведения научно-практической конференции. В первой главе проведен анализ существующих веб-сайтов подобной тематики, выявлены лучшие практики и определены требования к разрабатываемой системе, как функциональные (регистрация участников, управление тезисами, публикация материалов, программа конференции, онлайн-взаимодействие), так и нефункциональные (производительность, безопасность, масштабируемость, юзабилити, доступность). На основе проведенного анализа был осуществлен выбор технологического стека, включающий библиотеку React JS, платформу ASP.NET, язык программирования C# и систему управления базами данных PostgreSQL.

Вторая глава посвящена практической реализации веб-сайта. Подробно описана архитектура системы, включая схему базы данных и взаимодействие модулей. Разработаны и реализованы все запланированные функциональные модули. Проведены всесторонние тестирования веб-сайта, включающие функциональное, нагрузочное и тестирование безопасности. Результаты тестирования подтвердили соответствие разработанной системы заявленным требованиям.

В результате проведенной работы был создан функциональный и удобный в использовании веб-сайт научно-практической конференции, способный эффективно обрабатывать большое количество участников и материалов. Разработанная система может быть использована для организации конференций различного масштаба и является готовым решением для автоматизации процесса управления конференцией.

В перспективе дальнейшей работы можно рассмотреть следующие направления: интеграция с системами оплаты, добавление возможности онлайн-трансляции, расширение функционала модуля онлайн-взаимодействия, а также адаптация дизайна для мобильных устройств.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Троелсен Э. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е издание / Троелсен Э., Джепикс Ф.: Пер. с англ. — СПб.: ООО “Диалектика”, 2018 — 1328 с.
2. Албахари Д. C# 7.0. Справочник. Полное описание языка / Албахари Д., Албахари Б.: Пер. с англ. — СПб.: ООО “Альфакнига”, 2018. — 1024 с.
3. Тепляков С. Паттерны проектирования на платформе .NET. — СПб.: Питер, 2015. — 320 с.
4. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е издание – СПб.: Питер, 2021. – 896 с.
5. Гриффитс И. Программируем на C# 8.0. Разработка приложений – СПб.: Питер, 2021. – 944 с.
6. Арораа Г., Чилберто Д. Паттерны проектирования для C# и платформы .NET Core – СПб.: Питер, 2021. – 352 с.
7. Симан М., Дерсен С. Внедрение зависимостей на платформе .NET. 2-е издание. — СПб.: Питер, 2021. — 608 с.
8. Флэнаган, Д. JavaScript. Полное руководство, 7-е изд.: Пер. с англ. – СПб. ООО “Диалектика”, 2021. – 720 с.
9. Крокфорд, Д. Как устроен JavaScript. — СПб.: Питер, 2019. — 304 с.
10. Минник, К., Холланд, Е. JavaScript для чайников.: Пер. с англ. — М.: ООО “И.Д. Вильямс”, 2017. — 320 с
11. Гниденко И.Г. Технология разработки программного обеспечения: учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017. — 235 с.
12. Гордеев, С.И. Организация баз данных. Часть 2: учебник для вузов / С.И. Гордеев, В.Н. Волошина. – 2-е изд. – М.: Издательство Юрайт, 2019. – 501с.
13. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. - СПб.: Питер, 2017. - 288 с.
14. Мартин Р. Чистый код: создание, анализ и рефакторинг. – СПб.: Питер, 2019. – 464с.
15. Крокфорд Д. Javascript: сильные стороны. – СПб.: Питер, 2013. – 176с.
16. Советов, Б. Я. Базы данных: учебник для прикладного бакалавриата / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3-е изд., перераб. и доп. — М.: Издательство Юрайт, 2019. — 420 с.
17. Сысолетин, Е. Г. Разработка интернет-приложений: учеб. пособие для СПО / Е.Г. Сысолетин, С.Д. Ростунцев. — М.: Издательство Юрайт, 2019. — 90 с.
18. Сысолетин, Е.Г. Разработка интернет-приложений: учеб. пособие для вузов / Е.Г. Сысолетин, С.Д. Ростунцев; под науч. ред. Л. Г. Доросинского. — М.: Издательство Юрайт, 2019. — 90 с
19. Кудрина, Е. В. Основы алгоритмизации и программирования на языке C#: учеб. пособие для СПО / Е. В. Кудрина, М. В. Огнева. — М.: Издательство Юрайт, 2019. — 322 с.
20. Стефанов С. React. Быстрый старт, 2-е изд. — СПб.: Питер, 2023. — 304 с.
21. Куликов С. Тестирование программного обеспечения. Базовый курс 2-е издание – 298 с.
22. Документация ASP .NET [Электронный ресурс] URL: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0> (Дата обращения: 15.04.2025)
23. Документация React. [Электронный ресурс] URL: <https://ru.react.dev/> (Дата обращения: 17.02.2025)
24. React. Понятное руководство для начинающих. [Электронный ресурс] URL: <https://habr.com/ru/companies/ruvds/articles/428077/> (Дата обращения: 16.02.2025)
25. Документация по языку C# [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (Дата обращения: 04.03.2025)
26. Современный учебник Javascript [Электронный ресурс] URL: <https://learn.javascript.ru/> (Дата обращения: 19.02.2025)
27. Metanit. Руководство по Javacript [Электронный ресурс] URL: <https://metanit.com/web/javascript/> (Дата обращения: 01.02.2025)
28. Metanit. Руководство по React [Электронный ресурс] URL: <https://metanit.com/web/react/> (Дата обращения: 06.04.2025)
29. Metanit. Платформа ASP.NET [Электронный ресурс] URL: <https://metanit.com/sharp/mvc.php> (Дата обращения: 10.04.2025)
30. MDN. Начало работы с React [Электронный ресурс] URL: <https://developer.mozilla.org/ru/docs/Learn_web_development/Core/Frameworks_libraries/React_getting_started> (Дата обращения: 06.04.2025)
31. React. [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/React> (Дата обращения: 01.05.2025)
32. C Sharp [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/C_Sharp> (Дата обращения: 28.03.2025)
33. Все хуки и концепты React в одной статье [Электронный ресурс] URL: <https://habr.com/ru/articles/660573/> (Дата обращения: 23.04.2025)
34. Metanit. PostgreSQL [Электронный ресурс] URL: <https://metanit.com/sharp/efcore/7.3.php> (Дата обращения: 12.05.2025)
35. Документация React Router [Электронный ресурс] URL: <https://reactrouter.com/start/framework/routing> (Дата обращения: 11.05.2025)
36. Руководство Swagger UI [Электронный ресурс] URL: <https://starkovden.github.io/swagger-ui-tutorial.html> (Дата обращения: 25.05.2025)
37. Руководство по ASP.NET MVC 5 [Электронный ресурс] URL: <https://metanit.com/sharp/mvc5/> (Дата обращения: 18.04.2025)
38. Руководство по ASP.NET Core 9 [Электронный ресурс] URL: <https://metanit.com/sharp/aspnet6/> (Дата обращения: 18.04.2025)
39. OpenAPI/Swagger для начинающих [Электронный ресурс] URL: <https://habr.com/ru/articles/776538/> (Дата обращения: 26.05.2025)
40. Интерфейсы C# [Электронный ресурс] URL: <https://metanit.com/sharp/tutorial/3.9.php> (Дата обращения: 03.05.2025)
41. Интерфейс (справочник по C#) [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/interface> (Дата обращения: 29.04.2025)

# ПРИЛОЖЕНИЯ

Приложение 1. Описание продукта проекта