Частное учреждение образование

«Колледжа бизнеса и права»

WEB-ПРИЛОЖЕНИЕ ФОРУМ «PADZIEI»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

КП Т.093026.401 ПЗ

Руководитель проекта (Кривошеина А. В.)

Учащийся (Черняков    В.  А.)

2023

СОДЕРЖАНИЕ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

КП Т.093026.401 ПЗ

Разраб.

Черняков В.А.

Провер.

Кривошеина А.В.

Реценз.

Н. Контр.

Утверд.

*Веб-приложение форум «Padziei»*

Лит.

Листов

КБиП

61

У

[Введение 4](#_Toc138667760)

[1 Объектно-ориентированный анализ и проектирование приложения 5](#_Toc138667761)

[1.1 Назначение и цели создания Веб-приложения 5](#_Toc138667762)

[1.2 Проектирование модели 5](#_Toc138667763)

[2 Проектирование Веб-приложения 7](#_Toc138667764)

[2.1 Требование к Веб-приложению 7](#_Toc138667765)

[2.2 Структура Веб-приложения 8](#_Toc138667766)

[2.3 Проектирование макета Веб-приложения 8](#_Toc138667767)

[2.4 Программно-техничесие средства необходимые для разработки приложения 9](#_Toc138667768)

[2.5 Защита и сохранность данных 10](#_Toc138667769)

[2.6 Организация и ведение информационной базы 11](#_Toc138667770)

[3 Реализация Веб-приложения 12](#_Toc138667771)

[3.1 Разработка клиентской части приложения 12](#_Toc138667772)

[3.2 Разработка клиентской части приложения 12](#_Toc138667773)

[3.3 Описание используемых функций и процедур 13](#_Toc138667774)

[4 Описание Веб-приложения 14](#_Toc138667775)

[4.1 Общие сведения 14](#_Toc138667776)

[4.2 Функциональное назначение 14](#_Toc138667777)

[4.3 Описание разделов сайта 15](#_Toc138667778)

[5 Методика испытаний 16](#_Toc138667779)

[5.1 Технические требования 16](#_Toc138667780)

[5.2 Функциональное тестирование 16](#_Toc138667781)

[6 Применение 18](#_Toc138667782)

[6.1 Назначение программы 18](#_Toc138667783)

[6.2 Программно-аппаратное обеспечение сервера и клиента 18](#_Toc138667784)

[Заключение 19](#_Toc138667785)

[Список информационных источников 20](#_Toc138667786)

[Приложение А Структура Веб-приложения 21](#_Toc138667787)

[Приложение Б Текст модулей Веб-приложения 22](#_Toc138667788)

[Приложение В Результаты работы приложения 59](#_Toc138667789)

[Приложение Г (Обязательное) Блок-схемы работы алгоритмов 61](#_Toc138667790)

# Введение

Данная записка составлена с целью предоставить полное описание процесса создания и функциональности данного веб-форума. Проект "Padziei" представляет собой современное веб-приложение, разработанное с учетом потребностей и требований пользователей, для обеспечения коммуникации и обмена информацией в онлайн-среде.

Настоящая пояснительная записка включает в себя различные разделы, в которых описываются основные аспекты создания и функциональности веб-приложения "Padziei". В разделе "Назначение и цели создания Веб-приложения" представлены основные цели создания приложения, его связь с задачами конкретного предприятия, а также краткая характеристика предприятия. Здесь также указаны задачи, которые планируется решать с помощью данного веб-форума, аудитория, для которой предназначено приложение, периодичность его использования, а также описание существующих аналогов.

В дальнейших разделах данной записки детально рассмотрены требования к веб-приложению, его стилистическое оформление, графический дизайн, шрифтовое оформление, средства просмотра, контент и система управления контентом. Также приведено описание структуры приложения, включая информационную структуру и содержание главной страницы. Обоснован выбранный тип дизайна и представлены графические представления макетов страниц приложения.

Важным аспектом разработки приложения "Padziei" является выбор программно-технических средств, необходимых для его создания. В соответствующем разделе обоснован выбор инструмента разработки сайта, а также применяемых технологий, таких как HTML, CSS, JavaScript, ASP.NET и DOTNET.

Другие разделы данной пояснительной записки охватывают организацию и ведение информационной базы, защиту и сохранность данных, а также разработку административной и клиентской частей приложения. Описание используемых функций и процедур, а также проведение технического и функционального тестирования также представлены в соответствующих разделах данного документа.

Заключительные разделы включают общие сведения о проекте, технические требования, а также результаты функционального тестирования. Здесь также представлено описание назначения программы, областей ее применения, класса решаемых задач, а также организации диалога с пользователем и способов ввода информации.

Все представленные разделы данной пояснительной записки освещают основные аспекты разработки и функциональности веб-приложения "Padziei" и призваны предоставить вам полное представление о данном проекте.

# Объектно-ориентированный анализ и проектирование приложения

## Назначение и цели создания Веб-приложения

Целью создания веб-приложения "Padziei" является исследование предметной области с учетом связи этого проекта с конкретными задачами предприятия. Наше предприятие — это форум, предоставляющий платформу для обмена знаниями, идеями и опытом в различных сферах. Основной задачей приложения является обеспечение удобного и эффективного пространства для взаимодействия участников форума.

"Padziei" стремится решить следующие задачи:

– Предоставление возможности пользователям создавать темы для обсуждения, задавать вопросы и выражать свои мысли;

– Содействие в формировании сообщества экспертов, специалистов и любознательных пользователей;

– Обеспечение удобства поиска информации и доступа к уже существующим обсуждениям;

– Способствование взаимодействию и обмену знаниями между участниками форума;

– Создание возможности для пользователей оценивать и комментировать контент, повышая его качество и полезность.

Веб-приложение "Padziei" ориентировано на широкую аудиторию, состоящую из активных участников, желающих делиться своим опытом и обсуждать различные темы. Нашей целью является привлечение студентов, профессионалов и всех, кто интересуется обменом знаниями и развитием в своей сфере деятельности.

Предполагается, что пользователи будут активно использовать приложение "Padziei" на протяжении всего времени своего участия на форуме. Периодичность использования будет определяться потребностями и интересами пользователей и стремиться создать привлекательную и полезную платформу, стимулирующую регулярное взаимодействие и обмен мнениями.

На данный момент существуют несколько аналогичных веб-форумов, таких как "ForumHub", "DiscussIt" и "InteractZone". Однако, "Padziei" представляет собой уникальное приложение, в котором уделяется особое внимание удобству использования, интуитивно понятному интерфейсу, разнообразию предоставляемых функций и возможностям взаимодействия между участниками.

## Проектирование модели

Во время разработки приложения были разработаны различные графические схемы и модели, которые способствовали эффективности и продуктивности процесса разработки приложения.

В приложении Г представленные такие диаграммы как: диаграмма классов и диаграмма развертывания приложения.

Диаграмма классов (англ. class diagram) — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

# Проектирование Веб-приложения

## Требование к Веб-приложению

Веб-приложение "Padziei" имеет определенные требования к своему стилистическому оформлению. Оно должно быть органично вписано в целостность форума, передавать атмосферу обмена знаниями и способствовать комфортному взаимодействию пользователей. Стиль и цветовая гамма приложения должны быть умеренными и не отвлекать от основного контента, подчеркивая его значимость и читаемость. Важным требованием является применение современных веб-стандартов и технологий, обеспечивающих удобство использования и доступность приложения.

Графический дизайн "Padziei" должен быть эстетически привлекательным и привычным для пользователей, не затрудняющим восприятие информации. Учебный характер форума подразумевает использование нейтральных и профессиональных изображений, способных иллюстрировать различные тематики и не вводить пользователей в заблуждение. Визуальные элементы, такие как логотип и иконки, должны быть согласованы и четко передавать идею форума.

Относительно шрифтового оформления, основным требованием является читаемость текста на всех уровнях масштабирования. Используемые шрифты должны быть простыми, универсальными и подходящими для чтения как на больших экранах, так и на мобильных устройствах. Также, важно соблюдать принципы типографики, подбирая соответствующие размеры и визуальные отступы для различных элементов контента.

Для обеспечения средств просмотра контента требуется поддержка различных браузеров и устройств, включая популярные на данный момент. Веб-приложение "Padziei" должно быть отзывчивым и адаптивным, что позволит пользователям комфортно просматривать и взаимодействовать с контентом на широком спектре платформ.

Относительно контента, важно обеспечить его разнообразие, актуальность и полезность для пользователей. Содержание форума должно быть представлено в понятном и структурированном формате, позволяющем пользователям легко находить интересующую их информацию. Требования также включают модерацию контента для обеспечения качества дискуссий и предотвращения нарушений.

Система управления контентом должна быть надежной, гибкой и интуитивно понятной для администраторов форума. Она должна обеспечивать удобное добавление, редактирование и удаление контента, а также управление пользователями, правами доступа и настройками форума. Это позволит эффективно поддерживать и развивать "Padziei" в соответствии с потребностями пользователей и требованиями времени.

Компоновка страниц веб-приложения должна быть логичной и удобной для использования. Основное внимание следует уделять навигации и структуре, чтобы пользователи могли быстро ориентироваться в форуме и легко переходить между различными разделами. Эффективное использование пространства и визуальное выделение ключевых элементов помогут сосредоточить внимание пользователей на наиболее важной информации.

С учетом указанных требований к стилистическому оформлению, графическому дизайну, шрифтовому оформлению, средствам просмотра, контенту, системе управления контентом и компоновке страниц, "Padziei" стремится обеспечить удобство, эффективность и привлекательность для своих пользователей.

## Структура Веб-приложения

Первоначальная информационная структура приложения "Padziei" представляет собой иерархическую систему организации контента. Главные уровни структуры включают категории тем, подкатегории и отдельные темы для обсуждения. Это позволяет пользователям легко найти нужную информацию и участвовать в дискуссиях, соответствующих их интересам.

Главная страница приложения "Padziei" представляет собой информационный центр, который обеспечивает пользователей ключевой информацией и навигационными элементами. На главной странице содержится общая информация о форуме, его целях и возможностях. Кроме того, на главной странице отображаются последние активные темы, популярные темы и предложения для новых обсуждений. Это позволяет пользователям быть в курсе текущих тенденций и активно участвовать в актуальных дискуссиях.

Структура главной страницы "Padziei" также включает основное меню навигации, позволяющее пользователям легко переходить между различными разделами и категориями форума. На главной странице также присутствуют элементы интерактивности, такие как возможность задать вопрос, создать новую тему или просмотреть последние обновления. В целом, главная страница является центром взаимодействия пользователей с приложением и предоставляет им необходимую информацию для активного участия в обсуждениях.

Структура "Padziei" стремится обеспечить удобство использования и интуитивно понятную навигацию для пользователей, чтобы они могли быстро ориентироваться в приложении, находить интересующий их контент и активно взаимодействовать с сообществом.

## Проектирование макета Веб-приложения

В процессе проектирования макета Веб-приложения "Padziei" была выбрана блочный тип дизайна сайта. Блочная компоновка позволяет гибко располагать различные элементы на странице и адаптировать их под различные экраны и разрешения. Это обеспечивает удобство использования приложения как на больших мониторах, так и на мобильных устройствах.

Графическое представление макета главной страницы сайта "Padziei" включает в себя четкую и привлекательную композицию. В верхней части страницы располагается логотип и название форума, создающие узнаваемую идентичность приложения. Далее следует информационный блок с описанием форума, его целями и возможностями. Затем на странице отображаются последние активные темы и популярные обсуждения, привлекающие внимание пользователей и стимулирующие их активность. Меню навигации размещается в верхней части страницы для удобного доступа к различным разделам и категориям форума.

Макет второстепенной страницы "Padziei" основан на принципах согласованного и последовательного дизайна. На этой странице пользователи могут углубиться в конкретную тему или обсуждение. Макет включает в себя заголовок с названием темы, а также разделы с сообщениями пользователей, где отображаются их аватары, имена и текстовые комментарии. Возможно, на странице также присутствуют элементы для оценки сообщений или добавления новых комментариев. Графическое оформление страницы сосредоточено на читаемости текста и удобстве взаимодействия с контентом.

Макет Веб-приложения "Padziei" стремится обеспечить понятность, легкость использования и эстетическое удовлетворение для пользователей. Он основан на блочной компоновке, предоставляющей гибкость и адаптивность, и содержит графические элементы, подчеркивающие ключевую информацию и стимулирующие активность пользователей.

## Программно-техничесие средства необходимые для разработки приложения

При разработке веб-приложения "Padziei" была выбрана инструмент разработки сайта - интегрированную среду разработки (IDE) под названием "Visual Studio Code". Этот инструмент обеспечивает удобство и эффективность разработки благодаря своим расширенным функциональным возможностям, включая подсветку синтаксиса, автозаполнение кода и интеграцию с различными инструментами и плагинами. "Visual Studio Code" также обладает широкой поддержкой различных языков программирования и технологий, что позволяет нам легко реализовывать требуемый функционал приложения.

Одной из основных технологий, которая применялась при разработке "Padziei", является комбинация HTML и CSS. HTML (HyperText Markup Language) используется для структурирования и разметки контента веб-страницы, а CSS (Cascading Style Sheets) - для оформления и визуального оформления элементов страницы. Была выбрана эта технологическая комбинация, так как она является основой веб-разработки и позволяет нам создавать гибкие и привлекательные визуальные элементы интерфейса.

Для обеспечения совместимости и кросс-браузерной поддержки также необходимо следовать строгому стандарту XHTML (eXtensible HyperText Markup Language). XHTML позволяет нам создавать веб-страницы, совместимые с различными браузерами, обеспечивая стабильность и надежность работы приложения.

В дополнение к HTML и CSS используется JavaScript для добавления интерактивности и динамического поведения на страницах "Padziei". JavaScript является широко распространенным языком программирования, который позволяет нам создавать интерактивные элементы, обрабатывать пользовательские события и взаимодействовать с сервером для обновления данных без необходимости полной перезагрузки страницы.

Применение указанных технологий и языков программирования обеспечивает нам широкий функционал и возможности разработки, позволяя создать удобное и привлекательное веб-приложение "Padziei".

## Защита и сохранность данных

Веб-приложение "Padziei" обеспечивает надлежащую защиту и сохранность данных путем применения следующих мер и механизмов:

Ограничение доступа к данным: Доступ к данным в приложении ограничен и контролируется. Пользователи могут получать доступ только к тем данным, на которые у них есть соответствующие права. Аутентификация и авторизация осуществляются с использованием защищенных методов, таких как хеширование паролей и проверка токенов аутентификации. Также применяются механизмы контроля доступа и разграничения ролей пользователей для предотвращения несанкционированного доступа к конфиденциальным данным.

Защита информации от несанкционированного использования: для обеспечения безопасности данных приложение "Padziei" применяет шифрование важных информационных элементов, таких как пароли пользователей или другие конфиденциальные данные. Используются криптографические протоколы и алгоритмы, которые обеспечивают сохранность данных и предотвращают их несанкционированное использование или изменение. Также применяются меры по защите от внешних атак, таких как фильтрация входящих данных и предотвращение внедрения вредоносного кода.

Механизм защиты проектируемой задачи: в рамках разработки "Padziei" реализуется механизм контроля целостности данных и резервного копирования. Это позволяет обнаруживать и предотвращать случайные или злонамеренные изменения данных, а также обеспечивает возможность восстановления данных в случае их потери или повреждения. Регулярное создание резервных копий базы данных и системы, а также проверка и обновление программного обеспечения для обнаружения и исправления возможных уязвимостей, являются частью механизма защиты и сохранности данных.

Все эти меры и механизмы направлены на обеспечение безопасности данных и сохранность информации веб-приложения "Padziei". Они способствуют защите данных от несанкционированного доступа, использования или модификации, а также обеспечивают надежность и непрерывность работы приложения.

## Организация и ведение информационной базы

В информационной базе приложения "Padziei" содержатся следующие таблицы, которые описывают основные информационные сущности:

– таблица "Пользователь": эта таблица содержит информацию о зарегистрированных пользователях форума. В ней хранятся данные о каждом пользователе, включая идентификатор пользователя, имя, электронную почту, пароль и другие атрибуты, необходимые для аутентификации и управления учетными записями;

– таблица "Ветка обсуждения": в данной таблице хранится информация о созданных ветках обсуждения. Каждая ветка обсуждения имеет уникальный идентификатор, название, описание и дату создания. Эта таблица отражает структуру форума, позволяя пользователям создавать новые ветки для обсуждения определенных тем или категорий;

– таблица "Комментарий к ветке": данная таблица содержит комментарии, оставленные пользователями в рамках конкретной ветки обсуждения. Каждый комментарий связан с определенной веткой обсуждения и содержит информацию о пользователе, дате и тексте комментария. Эта таблица позволяет пользователям активно участвовать в обсуждениях и делиться своими мнениями и замечаниями;

– таблица "Токен пользователя": в этой таблице хранятся данные о токенах пользователей, используемых для аутентификации и авторизации. Каждый пользователь имеет свой уникальный токен, который используется для проверки их подлинности при запросах к защищенным ресурсам приложения.

Модель информационной базы данных "Padziei" отражает взаимосвязи между указанными таблицами, обеспечивая структурированное хранение и управление информацией о пользователях, ветках обсуждения и комментариях. Это позволяет приложению эффективно работать с данными и обеспечивать удобство использования для пользователей.

# Реализация Веб-приложения

## Разработка клиентской части приложения

Логическая и физическая структура серверной части Веб-приложения "Padziei" была разработана с учетом эффективного управления и администрирования системой.

Логическая структура приложения представлена в виде карты, которая отражает основные компоненты и модули приложения. Она включает разделы, такие как "Управление пользователями", "Управление ветками обсуждения", "Управление комментариями" и другие функциональные области. Каждый раздел соответствует определенным административным задачам и операциям.

Физическая структура серверной части приложения представлена в виде каталога, который содержит файлы и папки, необходимые для функционирования и обслуживания приложения. В этом каталоге расположены скрипты, модули, статические файлы, шаблоны и другие компоненты, которые обеспечивают функциональность административной части.

Навигация в административной части приложения осуществляется с помощью удобного и интуитивно понятного меню. Меню предоставляет администратору доступ к различным разделам и функциям приложения. Кроме того, на страницах присутствуют гиперссылки, которые позволяют быстро переходить между различными модулями и функциональными областями.

В целом, административная часть приложения "Padziei" разработана с учетом удобства использования и эффективного управления системой. Логическая и физическая структура, а также навигация обеспечивают администратору удобный доступ к функциям приложения и позволяют эффективно выполнять административные задачи.

## Разработка клиентской части приложения

Логическая и физическая структура клиентской части Веб-приложения "Padziei" была разработана с целью обеспечить удобство использования и навигацию для пользователей.

Логическая структура клиентской части представлена в виде карты, которая описывает основные модули и функциональные блоки приложения. Ключевые разделы включают "Главную страницу", "Форумы и ветки обсуждения", "Профиль пользователя" и другие соответствующие разделы. Каждый раздел предоставляет определенные функциональные возможности и возможность взаимодействия пользователей с приложением.

Физическая структура клиентской части представлена в виде каталога, который содержит файлы и ресурсы, необходимые для отображения интерфейса приложения. В этом каталоге находятся файлы HTML, CSS, JavaScript и другие статические ресурсы, которые обеспечивают визуальное представление и функциональность клиентской части.

Навигация в клиентской части приложения обеспечивается через удобное меню и гиперссылки. Меню предоставляет пользователю доступ к основным разделам и функциональным блокам приложения. Гиперссылки располагаются на страницах и позволяют пользователям переходить между различными разделами, ветками обсуждения, профилями пользователей и другими сущностями приложения.

В целом, клиентская часть приложения "Padziei" разработана с учетом удобства использования и интуитивной навигации для пользователей. Логическая и физическая структура, а также навигационные элементы, способствуют понятному и эффективному взаимодействию пользователей с приложением.

## Описание используемых функций и процедур

В рамках среды разрабатываемого приложения "Padziei" используются различные функции и процедуры, которые связаны с определенными элементами управления, событиями, страницами и ссылками. Они обеспечивают функциональность и взаимодействие пользователей с приложением.

Одной из основных функций является функция "Регистрация пользователя". Она позволяет новым пользователям создавать учетные записи в приложении и получать доступ к его функционалу. Данная функция связана с кнопкой "Регистрация" на главной странице и обрабатывает информацию, предоставленную пользователем для создания учетной записи.

Функция "Создание новой ветки обсуждения" позволяет пользователям создавать новые темы для обсуждения на форуме. Она связана с кнопкой "Создать ветку" в соответствующем разделе и обрабатывает информацию, введенную пользователем, для создания новой ветки обсуждения.

Процедура "Добавление комментария к ветке" позволяет пользователям оставлять комментарии к существующим веткам обсуждения. Она связана с кнопкой "Добавить комментарий" на странице ветки обсуждения и обрабатывает введенный пользователем текст комментария.

Взаимосвязь функций проявляется, например, в процессе создания новой ветки обсуждения. После успешного создания ветки, функция "Создание новой ветки обсуждения" вызывает функцию "Отображение списка веток обсуждения", которая обновляет список доступных веток на форуме, чтобы новая ветка была видима для пользователей.

Листинг всех модулей с комментариями и указанием элементов управления представлен в приложении Б.

# Описание Веб-приложения

## Общие сведения

Цель разработки Веб-приложения "Padziei" заключается в создании удобного и функционального форума, который позволит пользователям обмениваться информацией, задавать вопросы, обсуждать темы и находить интересующую их информацию. Главной задачей является обеспечение простоты использования и понятной навигации для пользователей, а также предоставление возможности взаимодействия между пользователями.

При установке Веб-приложения "Padziei" необходимо развернуть все файлы, относящиеся к приложению, в соответствующем каталоге на сервере. Дерево файлов включает файлы HTML, CSS, JavaScript, серверные скрипты ASP.NET, DOTNET и другие ресурсы, необходимые для полноценного функционирования приложения.

Для запуска Веб-приложения "Padziei" требуется наличие веб-сервера, поддерживающего ASP.NET и DOTNET, а также настроенного окружения для выполнения серверных скриптов. После успешного развертывания файлов на сервере и настройки окружения, приложение становится доступным для пользователей через соответствующий URL-адрес.

Верстка страниц Веб-приложения "Padziei" основана на использовании блочной модели и таблиц для структурирования и расположения элементов на страницах. Цветовая разметка страниц определена в CSS-файлах, где применяются соответствующие стили и цветовые схемы для обеспечения согласованного визуального представления.

Применяемые технологии в Веб-приложении "Padziei" включают HTML, JavaScript, ASP.NET и DOTNET. HTML используется для разметки содержимого страниц, JavaScript - для добавления интерактивности и динамического поведения, а ASP.NET и DOTNET - для разработки серверной логики и взаимодействия с базой данных.

## Функциональное назначение

Назначение Веб-приложения "Padziei" заключается в создании платформы для обмена информацией и общения между пользователями. Приложение предоставляет возможность пользователям задавать вопросы, выражать свои мнения, обсуждать различные темы, делиться опытом и находить интересующую информацию. Целью приложения является создание активного и динамичного сообщества, где пользователи могут взаимодействовать и получать нужную помощь или информацию.

Потенциальная аудитория Веб-приложения "Padziei" включает широкий круг пользователей, заинтересованных в общении, обмене опытом и получении информации по различным тематикам. Это могут быть студенты, профессионалы, исследователи, учителя и люди, интересующиеся определенной предметной областью.

Для обеспечения защиты и разграничения доступа к данным в приложении "Padziei" применяются различные средства и механизмы. В первую очередь, используются механизмы аутентификации и авторизации пользователей, которые позволяют контролировать доступ к определенным функциям и ресурсам приложения. Кроме того, применяются механизмы шифрования для защиты конфиденциальных данных и обеспечения безопасности передачи информации между клиентом и сервером.

Веб-приложение "Padziei" разрабатывается с учетом кроссплатформенности, что означает его работоспособность на различных операционных системах, таких как Windows, macOS, Linux, а также на мобильных платформах. Кроссбраузерность также учитывается, что означает поддержку различных веб-браузеров, таких как Google Chrome, Mozilla Firefox, Safari, Microsoft Edge и других, с целью обеспечения корректного отображения и функционирования приложения на разных платформах и веб-браузерах.

## Описание разделов сайта

Главная страница административной части Веб-приложения "Padziei" предоставляет доступ к функциональным возможностям, предназначенным для управления и администрирования приложения. На главной странице административной части пользователь может войти в систему, используя свои учетные данные, чтобы получить доступ к административным функциям. Здесь также могут быть представлены общие сведения о приложении, статистика использования и другие сведения, которые могут быть полезны для администраторов.

Главная страница клиентской части Веб-приложения "Padziei" предназначена для представления информации и функциональных возможностей, предназначенных для пользователей. Здесь пользователи могут видеть список доступных категорий или разделов форума, наиболее популярные обсуждения или последние добавленные комментарии. Также могут быть представлены функции поиска, регистрации, авторизации и другие интерактивные элементы, обеспечивающие удобную навигацию и использование приложения.

Дополнительная схема разделов сайта предоставлена в приложении А.

# Методика испытаний

## Технические требования

Для проведения испытаний и проверки работоспособности Веб-приложения "Padziei" необходимы следующие технические средства:

– компьютер или ноутбук: для запуска и тестирования приложения требуется компьютер или ноутбук с достаточными ресурсами для обработки и отображения веб-страниц, выполнения скриптов и обеспечения плавной работы интерфейса.

– операционная система: веб-приложение "Padziei" должно быть совместимо с различными операционными системами, такими как Windows, macOS, Linux и другими популярными операционными системами;

– браузеры: Приложение должно поддерживать работу с различными веб-браузерами, включая Google Chrome, Mozilla Firefox, Microsoft Edge, Safari и другие популярные браузеры. Требуется проверка и обеспечение совместимости приложения с разными версиями браузеров.

– интернет-соединение: для полноценного использования Веб-приложения "Padziei" необходимо стабильное и быстрое интернет-соединение, чтобы обеспечить доступность и быстродействие при работе с приложением;

– тестовые данные: для проведения испытаний и проверки функциональности приложения требуются тестовые данные, включающие различные сценарии использования, типичные для предполагаемой аудитории приложения.

Что касается совместимости сайта с различными браузерами и настройками, особое внимание должно быть уделено обеспечению кроссбраузерности. Сайт должен корректно отображаться и функционировать в различных браузерах с разными настройками, включая разрешение экрана, размеры окна браузера, блокировку JavaScript или изображений и другие возможные ограничения.

При разработке и тестировании приложения необходимо учитывать эти требования и обеспечить, чтобы оно работало стабильно и корректно в широком спектре технических средств и настроек браузеров.

## Функциональное тестирование

В рамках функционального тестирования Веб-приложения "Padziei" были проведены проверки каждого пункта меню и каждой операции, выполняемой приложением, с использованием тестовых примеров. Целью функционального тестирования было гарантировать работу всех элементов управления в автономном режиме и проверить соответствие результатов работы приложения ожидаемым результатам.

Одним из тестовых примеров была проверка функции создания новой ветки обсуждения. Входными данными были указаны заголовок ветки, текст сообщения и данные автора. Ожидалось, что после создания ветки она будет отображаться на главной странице, а данные об авторе и текст сообщения будут корректно сохранены в базе данных. Результаты работы приложения, подтверждающие успешное создание ветки обсуждения, предоставлены в Приложении В.

Также были проведены тесты других операций и функций приложения, таких как добавление комментария к ветке, редактирование сообщения, поиск по форуму и другие. Каждый тестовый пример включал конкретные входные данные и ожидаемые результаты. Результаты работы приложения, включая копии экранов с подтверждением выполнения определенных действий, предоставлены в Приложении В.

Функциональное тестирование позволило проверить корректность работы всех элементов управления приложения и соответствие его функциональности требованиям и ожиданиям пользователей.

# Применение

## Назначение программы

Назначение сайта "Padziei" заключается в предоставлении платформы для обмена информацией, обсуждения и взаимодействия пользователей в рамках конкретной предметной области. Он предназначен для создания и поддержки онлайн-форума, где пользователи могут задавать вопросы, делиться знаниями, обсуждать темы и находить ответы на интересующие их вопросы.

Сайт "Padziei" может быть применен в различных областях, в зависимости от его конкретной тематики и предназначения. Он может использоваться для обсуждения академических вопросов, научных исследований, обмена опытом в профессиональной сфере, обсуждения новостей и событий, обучения и других сфер, где требуется взаимодействие и обмен информацией между пользователями.

Однако, следует отметить, что сайт "Padziei" имеет ограничения, накладываемые на его область применения. Он не предназначен для проведения коммерческой деятельности, распространения незаконной информации, оскорблений, нарушения авторских прав или иных негативных действий. Администрация сайта осуществляет контроль и модерацию контента, чтобы обеспечить безопасную и полезную среду общения для пользователей.

Для организации диалога с пользователями сайта "Padziei" предусмотрены различные способы ввода информации. Пользователи могут оставлять комментарии, задавать вопросы, отвечать на вопросы других пользователей, создавать новые ветки обсуждения и активно участвовать в обмене информацией. Ввод информации осуществляется с помощью текстовых полей, кнопок, ссылок и других элементов управления, обеспечивающих удобство и простоту взаимодействия с сайтом.

## Программно-аппаратное обеспечение сервера и клиента

Для корректной работы Веб-приложения "Padziei" требуется определенное программно-аппаратное обеспечение на стороне сервера и клиента. Ниже приведены требования к каждой из этих составляющих:

– один процессор частоты работы от 4 ГГц, архитектуры x64;

– оперативная память от 8 ГиБ;

– место на диске от 20 ГиБ;

– операционная система MS Windows или Linux;

– интернет-соединение от скорости 1МиБ/с;

– современный веб-браузер;

– клавиатура проводная;

– мышь проводная;

– видеопамять от 2 ГиБ.

# Заключение

В ходе разработки веб-приложения "Padziei" был выполнен значительный объем работы, направленной на создание инновационного и функционального веб-форума. В этом разделе подведены итоги и подчеркнуть ключевые моменты, связанные с разработкой и реализацией проекта.

Процесс разработки "Padziei" начался с определения целей и назначения приложения. Наша цель состояла в создании удобного и эффективного средства коммуникации и обмена информацией, аудиторией которого стали пользователи, заинтересованные в обсуждении различных тем и получении качественной информации. В процессе разработки было уделено особое внимание функциональности, интерфейсу и безопасности приложения, чтобы обеспечить удовлетворение потребностей пользователей и защитить их данные.

Основываясь на требованиях пользователей и анализе существующих решений, была определена область применения веб-приложения "Padziei". Оно предоставляет возможность общения, обмена мнениями и опытом, а также получения информации по различным темам. Приложение может использоваться в различных сферах, включая образование, бизнес, науку и многие другие.

Важным аспектом разработки "Padziei" является защита данных пользователей и разграничение доступа к информации. Были приняты все необходимые меры для обеспечения безопасности и конфиденциальности данных. Также было уделено внимание совместимости приложения с различными браузерами и настройками, чтобы обеспечить комфортное использование для всех пользователей.

Результаты технического и функционального тестирования подтвердили работоспособность и эффективность веб-приложения "Padziei". Были проведены проверки каждого пункта меню и операции, выполняемой приложением, и убедились в их корректной работе. Тестовые примеры, представленные в приложении В, подтверждают успешное выполнение задач и достижение ожидаемых результатов.

Веб-приложение "Padziei" было разработано с использованием таких технологий, как HTML, CSS, JavaScript, ASP.NET и DOTNET. Эти инструменты и технологии обеспечили надежность, гибкость и функциональность приложения.

В заключение, разработка и реализация веб-приложения "Padziei" была успешно завершена. Оно представляет собой инновационный инструмент для общения и обмена информацией, который может быть эффективно применен в различных сферах.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых работ, дипломных проектов и отчётов для учащихся специальности 2-40 01 01 «ПОИТ» / Т.Г. Багласова. – Минск : КБП , 2016 . – 30 с.
2. Объектно-ориентированный анализ и проектирование с примерами приложений / Гради Буч [и др.]. – 3-е изд. - М. : ООО «И.Д. Вильямс» , 2008 . – 720 с.
3. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996 . – Минск : Межгос. совет по стандартизации, метрологии и сертификации , 1995 . – 84 с.
4. Описание программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.402-2000 . – Введ. 01.09.2001 . – Минск : Межгос. совет по стандартизации, метрологии и сертификации , 2000 . – 14 с.
5. Текст программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.401-2000 . – Введ. 01.09.2001 . – Минск : Межгос. совет по стандартизации, метрологии и сертификации , 2000 . – 16 с.

# Приложение А Текст модулей Веб-приложения

using System.Reflection;

using System.Text.RegularExpressions;

using System.Diagnostics.CodeAnalysis;

public class Program

{

public static string WEB\_DIR = @"C:/Users/VloBo/Documents/code/padziei-webpages";

[NotNull]

public static WebApplication app;

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

app = builder.Build();

\_ = ProxyServer.Start();

app.Run(async (context) =>

{

try

{

app.Logger.LogWarning(context.Connection.LocalIpAddress + " " + context.Connection.RemoteIpAddress);

foreach (var method in typeof(PathHandler).GetMethods(BindingFlags.Public | BindingFlags.Instance))

{

var attribute = method.GetCustomAttribute<PathAttribute>();

if (attribute != null && new Regex(attribute.Regex).IsMatch(context.Request.Path))

{

Task? task = (Task?)(method.Invoke(new PathHandler(), new Object[] { context }));

if (task is not null)

{

await task;

return;

}

}

}

}

catch (Exception e)

{

app.Logger.LogError("КРИТИЧЕСКАЯ ОШИБКА ОБРАБОТКИ ЗАПРОСА\n" + e.Message + "\n\n" + e.StackTrace);

}

return;

});

app.Run();

}

}  
  
using System.Reflection;

using System.Text;

using System.Text.Json;

public class PathHandler

{

/// <summary>

/// Обработчик пути для файлов JavaScript.

/// </summary>

[Path(@"^\/(js).\*$")]

public async Task fun1(HttpContext context)

{

var response = context.Response;

response.Headers.ContentType = "text/javascript; charset=utf-8";

await context.Response.SendFileAsync(Program.WEB\_DIR + context.Request.Path);

return;

}

/// <summary>

/// Обработчик пути для файлов CSS.

/// </summary>

[Path(@"^\/(css).\*$")]

public async Task fun0(HttpContext context)

{

var response = context.Response;

response.Headers.ContentType = "text/css; charset=utf-8";

await context.Response.SendFileAsync(Program.WEB\_DIR + context.Request.Path);

return;

}

/// <summary>

/// Обработчик пути для API.

/// </summary>

[Path(@"^\/api.\*$")]

public async Task fun2(HttpContext context)

{

string requestBody;

using (var reader = new StreamReader(context.Request.Body, Encoding.UTF8))

{

requestBody = await reader.ReadToEndAsync();

}

JsonDocument jd = JsonDocument.Parse(requestBody);

int code = Convert.ToInt32(jd.RootElement.GetProperty("code").ToString());

foreach (var method in typeof(ApiCodeHandler).GetMethods(BindingFlags.Public | BindingFlags.Instance))

{

var attribute = method.GetCustomAttribute<ApiCodeAttribute>();

if (attribute != null && attribute.Code == code)

{

Task? task = (Task?)(method.Invoke(new ApiCodeHandler(), new Object[] { context, jd }));

if (task is not null)

{

await task;

return;

}

}

}

return;

}

/// <summary>

/// Обработчик пути для файлов с расширениями.

/// </summary>

[Path(@"^\/.\*\..\*$")]

public async Task fun3(HttpContext context)

{

try

{

await context.Response.SendFileAsync(Program.WEB\_DIR + context.Request.Path);

}

catch (FileNotFoundException)

{

context.Response.StatusCode = 404;

await context.Response.SendFileAsync(Program.WEB\_DIR + "/404.html");

}

return;

}

/// <summary>

/// Обработчик пути по умолчанию.

/// </summary>

[Path(@"\/?.\*$")]

public async Task fun4(HttpContext context)

{

try

{

await context.Response.SendFileAsync(Program.WEB\_DIR + context.Request.Path + "/index.html");

}

catch (FileNotFoundException)

{

context.Response.StatusCode = 404;

await context.Response.SendFileAsync(Program.WEB\_DIR + "/404.html");

}

return;

}

}

[AttributeUsage(AttributeTargets.Method)]

public class PathAttribute : Attribute

{

private string \_regex;

public PathAttribute(string regex)

{

this.\_regex = regex;

}

public string Regex

{

get

{

return this.\_regex;

}

}

}

using System;

using System.Net;

using System.Net.Sockets;

using System.Threading;

class ProxyServer

{

public static async Task Start()

{

try

{

TcpListener listener = new TcpListener(IPAddress.Any, 80);

listener.Start();

Program.app.Logger.LogInformation("Proxy-server is running on: port 80 to 1984");

while (true)

{

TcpClient client = await listener.AcceptTcpClientAsync();

Program.app.Logger.LogDebug("Proxy connection opened");

\_ = HandleClient(client);

}

}

catch (Exception ex)

{

Program.app.Logger.LogError("Failed to start proxy server: \n" + ex.Message + "\n" + ex.StackTrace);

}

}

static async Task HandleClient(TcpClient client)

{

using (client)

{

try

{

// Создаем подключение к целевому серверу на порту 1984

TcpClient targetServer = new TcpClient();

await targetServer.ConnectAsync("localhost", 1984);

// Получаем потоки для чтения и записи данных между клиентом и сервером

using (NetworkStream clientStream = client.GetStream())

using (NetworkStream targetStream = targetServer.GetStream())

{

// Асинхронно перенаправляем данные от клиента к серверу и наоборот

Task clientToServer = clientStream.CopyToAsync(targetStream);

Task serverToClient = targetStream.CopyToAsync(clientStream);

// Ожидаем завершения обоих задач

await Task.WhenAll(clientToServer, serverToClient);

}

}

catch (Exception ex)

{

Program.app.Logger.LogError("Failed to send information on proxy server: \n" + ex.Message + "\n" + ex.StackTrace);

}

Program.app.Logger.LogDebug("Proxy connection closed");

}

}

}

using Npgsql;

using System.Diagnostics;

using System.Security.Cryptography;

using System.Text;

class Database

{

private static Database? HINSTANCE;

private NpgsqlConnection \_connection;

private const string CONNECTION\_STRING = "Server=localhost;Port=5432;Database=padziei;User Id=postgres;Password=postgres1;";

private Database()

{

this.\_connection = new NpgsqlConnection(CONNECTION\_STRING);

this.\_connection.Open();

}

~Database()

{

this.\_connection.Close();

}

public static Database Hinstance

{

get

{

if (HINSTANCE is null)

{

HINSTANCE = new Database();

}

return HINSTANCE;

}

}

public Guid? FindUserByUsername(string username)

{

string sqlstr = $"SELECT id FROM Users WHERE Users.username = '{username}';";

Guid? ret;

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

ret = (Guid?)command.ExecuteScalar();

}

}

return ret;

}

public int CreateUser(string username, string password, string email)

{

string hashPassword = password;

using (var sha256 = SHA256.Create())

{

hashPassword = Convert.ToHexString(sha256.ComputeHash(Encoding.UTF8.GetBytes(password)));

}

try

{

string sqlstr = $"INSERT INTO Users VALUES ('{Guid.NewGuid().ToString()}', '{username}', '{hashPassword}', '{DateTime.Now.ToUniversalTime().ToString()}', '{email}', 'normal');";

Program.app.Logger.LogDebug(sqlstr);

int ret = 0;

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

ret = command.ExecuteNonQuery();

}

}

return ret;

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message);

return -1;

}

}

public Guid? CreateToken(string username, string password)

{

string hashPassword = password;

using (var sha256 = SHA256.Create())

{

hashPassword = Convert.ToHexString(sha256.ComputeHash(Encoding.UTF8.GetBytes(password)));

}

try

{

string sqlstr = $"SELECT id FROM Users WHERE username = '{username}' AND password = '{hashPassword}';";

Program.app.Logger.LogDebug(sqlstr);

Guid? guid;

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

guid = (Guid?)command.ExecuteScalar();

}

}

if (guid is null)

{

return null;

}

Guid token\_guid = Guid.NewGuid();

sqlstr = $"INSERT INTO tokens VALUES ('{token\_guid.ToString()}', '{guid.ToString()}', '{DateTime.Now.ToUniversalTime().ToString()}', '30 day');";

int i = 0;

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

i = command.ExecuteNonQuery();

}

}

if (i > 0)

{

return token\_guid;

}

else

{

return null;

}

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message);

return null;

}

}

public Guid? GetUserByToken(Guid userToken)

{

Guid? ret;

try

{

string sqlstr = $"SELECT user\_id FROM tokens WHERE id = '{userToken.ToString()}';";

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

ret = (Guid?)command.ExecuteScalar();

}

}

return ret;

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message);

return null;

}

}

public string? GetUserInfo(Guid userGuid, bool email)

{

try

{

string sqlstr = $"SELECT \* FROM Users WHERE id = '{userGuid.ToString()}';";

string outt = "";

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

using (var reader = command.ExecuteReader())

{

reader.Read();

outt = "{" +

"\"id\":\"" + reader.GetGuid(0) + "\"," +

"\"username\":\"" + reader.GetString(1) + "\"," +

(email ? ("\"email\":\"" + reader.GetString(4) + "\",") : "") +

"\"date\":\"" + reader.GetDateTime(3).ToString("dd.MM.yyyy hh:mm") + "\"," +

"\"role\":\"" + reader.GetString(5) + "\"" +

"}";

}

}

}

return outt;

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message + "\n" + e.StackTrace);

return null;

}

}

public string? GetThredInfo(Guid threadGuid)

{

try

{

string sqlstr = $"SELECT \* FROM Threads WHERE id = '{threadGuid.ToString()}';";

string outt = "";

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

using (var reader = command.ExecuteReader())

{

reader.Read();

outt = "{" +

"\"id\":\"" + reader.GetGuid(0) + "\"," +

"\"author\":\"" + reader.GetGuid(1) + "\"," +

"\"date\":\"" + reader.GetDateTime(2).ToString("dd.MM.yyyy hh:mm") + "\"," +

"\"title\":\"" + reader.GetString(3) + "\"," +

"\"body\":\"" + reader.GetString(4) + "\"," +

"\"karma\_count\":\"" + ((Guid[])reader["karma"]).Length + "\"" +

"}";

}

}

}

return outt;

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message + "\n" + e.StackTrace);

return null;

}

}

public Guid[] SelectTopThreads(int count)

{

try

{

string sqlstr = $"SELECT id FROM Threads LIMIT {count};";

List<Guid> guids = new List<Guid>();

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

using (var reader = command.ExecuteReader())

{

while (reader.Read())

{

guids.Add((Guid)reader["id"]);

}

}

}

}

return guids.ToArray();

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message + "\n" + e.StackTrace);

return new Guid[0];

}

}

public Guid? CreateThread(Guid token, string title, string content)

{

Guid? user = GetUserByToken(token);

if (user is null)

{

return null;

}

Guid? ret;

try

{

string sqlstr = $"INSERT INTO Threads (id, author, data\_create, title, body, karma) VALUES ('{Guid.NewGuid().ToString()}', '{user.ToString()}', '{DateTime.Now.ToUniversalTime().ToString()}', '{title}', '{content}', ARRAY[uuid('{user.ToString()}')]) RETURNING id;";

Program.app.Logger.LogWarning(sqlstr);

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

using (var reader = command.ExecuteReader())

{

reader.Read();

ret = (Guid)reader["id"];

}

}

}

return ret;

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message + "\n" + e.StackTrace);

return null;

}

}

public Guid[] GetCommentsFromThread(Guid thread)

{

try

{

string sqlstr = $"SELECT id FROM Comments WHERE thread\_id = '{thread.ToString()}';";

List<Guid> guids = new List<Guid>();

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

using (var reader = command.ExecuteReader())

{

while (reader.Read())

{

guids.Add((Guid)reader["id"]);

}

}

}

}

return guids.ToArray();

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message + "\n" + e.StackTrace);

return new Guid[0];

}

}

public string? GetCommentInfo(Guid id)

{

try

{

string sqlstr = $"SELECT \* FROM Comments WHERE id = '{id.ToString()}';";

string ret = "";

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

using (var reader = command.ExecuteReader())

{

reader.Read();

ret = "{" +

"\"id\":\"" + reader.GetGuid(0) + "\"," +

"\"thread\":\"" + reader.GetGuid(1) + "\"," +

"\"author\":\"" + reader.GetGuid(2) + "\"," +

"\"date\":\"" + reader.GetDateTime(3).ToString("dd.MM.yyyy hh:mm") + "\"," +

"\"content\":\"" + reader.GetString(4) + "\"" +

"}";

}

}

}

return ret;

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message + "\n" + e.StackTrace);

return null;

}

}

public Guid? CreateComment(Guid token, Guid thread, string content)

{

Guid? user = GetUserByToken(token);

if (user is null)

{

return null;

}

Guid? ret;

try

{

string sqlstr = $"INSERT INTO Comments (id, thread\_id, author, data\_create, body) VALUES ('{Guid.NewGuid().ToString()}', '{thread.ToString()}', '{user.ToString()}', '{DateTime.Now.ToUniversalTime().ToString()}', '{content}') RETURNING id;";

Program.app.Logger.LogWarning(sqlstr);

using (var con = new NpgsqlConnection(CONNECTION\_STRING))

{

con.Open();

using (var command = new NpgsqlCommand(sqlstr, con))

{

using (var reader = command.ExecuteReader())

{

reader.Read();

ret = (Guid)reader["id"];

}

}

}

return ret;

}

catch (Exception e)

{

Program.app.Logger.LogError(e.Message + "\n" + e.StackTrace);

return null;

}

}

}

using System.Text.Json;

public class ApiCodeHandler

{

/// <summary>

/// Обработчик кода 12.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(12)]

public async Task code12(HttpContext context, JsonDocument jd)

{

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

await response.WriteAsync("{\"code\":\"13\",\"body\":{\"content\":\"hello padziei\"}}");

}

/// <summary>

/// Обработчик кода 20.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(20)]

public async Task code20(HttpContext context, JsonDocument jd)

{

// Извлечение данных из JSON-документа

string username = jd.RootElement.GetProperty("body").GetProperty("username").ToString();

string email = jd.RootElement.GetProperty("body").GetProperty("email").ToString();

string password = jd.RootElement.GetProperty("body").GetProperty("password").ToString();

// Создание пользователя в базе данных

string status = -1 != Database.Hinstance.CreateUser(username, password, email) ? "OK" : "NOT";

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

await response.WriteAsync("{\"code\":\"1\",\"body\":{\"status\":\"" + status + "\"}}");

}

/// <summary>

/// Обработчик кода 30.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(30)]

public async Task code30(HttpContext context, JsonDocument jd)

{

// Извлечение данных из JSON-документа

string username = jd.RootElement.GetProperty("body").GetProperty("username").ToString();

string password = jd.RootElement.GetProperty("body").GetProperty("password").ToString();

// Создание токена и проверка наличия пользователя в базе данных

Guid? g = Database.Hinstance.CreateToken(username, password);

string status = g is null ? "NOT" : "OK";

var response = context.Response;

if (g is not null)

{

response.Cookies.Append("\_t", g.ToString() ?? "\_");

}

response.Headers.ContentType = "application/json; charset=utf-8";

await response.WriteAsync("{\"code\":\"1\",\"body\":{\"status\":\"" + status + "\"}}");

}

/// <summary>

/// Обработчик кода 42.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(42)]

public async Task code42(HttpContext context, JsonDocument jd)

{

Program.app.Logger.LogWarning(jd.RootElement.ToString());

Guid? token;

Guid? id;

try

{

token = new Guid(jd.RootElement.GetProperty("body").GetProperty("token").ToString());

}

catch (KeyNotFoundException)

{

token = null;

}

try

{

id = new Guid(jd.RootElement.GetProperty("body").GetProperty("id").ToString());

}

catch (KeyNotFoundException)

{

id = null;

}

if (id is null)

{

if (token is not null)

{

id = Database.Hinstance.GetUserByToken(token ?? new Guid());

}

}

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

if (id is null)

{

await response.WriteAsync("{\"code\":\"43\",\"body\":{\"status\":\"NOT\"}}");

return;

}

else

{

string? res = Database.Hinstance.GetUserInfo(id ?? new Guid(), (token is null));

if (res is null)

{

await response.WriteAsync("{\"code\":\"43\",\"body\":{\"status\":\"NOT\"}}");

Program.app.Logger.LogWarning("3");

return;

}

else

{

await response.WriteAsync("{\"code\":\"43\",\"body\":{\"status\":\"OK\",\"user\":" + res + "}}");

Program.app.Logger.LogWarning(res);

return;

}

}

}

/// <summary>

/// Обработчик кода 52.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(52)]

public async Task code52(HttpContext context, JsonDocument jd)

{

int count = Convert.ToInt32(jd.RootElement.GetProperty("body").GetProperty("count").ToString());

// Выборка верхних потоков из базы данных

Guid[] guids = Database.Hinstance.SelectTopThreads(count);

string asnwer = "[";

for (int i = 0; i < guids.Length - 1; i++)

{

asnwer += "\"" + guids[i].ToString() + "\",";

}

asnwer += "\"" + guids[guids.Length - 1].ToString() + "\"]";

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

await response.WriteAsync("{\"code\":\"53\",\"body\":{\"threads\":" + asnwer + "}}");

}

/// <summary>

/// Обработчик кода 62.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(62)]

public async Task code62(HttpContext context, JsonDocument jd)

{

Guid id = new Guid(jd.RootElement.GetProperty("body").GetProperty("id").ToString());

// Получение информации о потоке из базы данных

string str = Database.Hinstance.GetThredInfo(id) ?? "{}";

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

await response.WriteAsync("{\"code\":\"63\",\"body\":" + str + "}");

Program.app.Logger.LogWarning("{\"code\":\"63\",\"body\":" + str + "}");

}

/// <summary>

/// Обработчик кода 70.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(70)]

public async Task code70(HttpContext context, JsonDocument jd)

{

Guid token = new Guid(jd.RootElement.GetProperty("body").GetProperty("token").ToString());

string title = jd.RootElement.GetProperty("body").GetProperty("title").ToString();

string content = jd.RootElement.GetProperty("body").GetProperty("content").ToString();

// Создание нового потока в базе данных

Guid? id = Database.Hinstance.CreateThread(token, title, content);

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

if (id is null)

{

await response.WriteAsync("{\"code\":\"71\",\"body\":{\"id\":null}}");

return;

}

await response.WriteAsync("{\"code\":\"71\",\"body\":{\"id\":\"" + id + "\"}}");

Program.app.Logger.LogWarning("{\"code\":\"71\",\"body\":{\"id\":\"" + id + "\"}}");

}

/// <summary>

/// Обработчик кода 82.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(82)]

public async Task code82(HttpContext context, JsonDocument jd)

{

Guid thread = new Guid(jd.RootElement.GetProperty("body").GetProperty("thread").ToString());

// Получение комментариев из базы данных для указанного потока

Guid[] guids = Database.Hinstance.GetCommentsFromThread(thread);

string asnwer = "[";

for (int i = 0; i < guids.Length - 1; i++)

{

asnwer += "\"" + guids[i].ToString() + "\",";

}

if (guids.Length == 0)

{

asnwer += "]";

}

else

{

asnwer += "\"" + guids[guids.Length - 1].ToString() + "\"]";

}

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

await response.WriteAsync("{\"code\":\"83\",\"body\":{\"comments\":" + asnwer + "}}");

}

/// <summary>

/// Обработчик кода 92.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(92)]

public async Task code92(HttpContext context, JsonDocument jd)

{

Guid id = new Guid(jd.RootElement.GetProperty("body").GetProperty("id").ToString());

// Получение информации о комментарии из базы данных

string str = Database.Hinstance.GetCommentInfo(id) ?? "{}";

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

await response.WriteAsync("{\"code\":\"93\",\"body\":" + str + "}");

Program.app.Logger.LogWarning("{\"code\":\"93\",\"body\":" + str + "}");

}

/// <summary>

/// Обработчик кода 100.

/// </summary>

/// <param name="context">Контекст HTTP-запроса.</param>

/// <param name="jd">JSON-документ.</param>

[ApiCode(100)]

public async Task code100(HttpContext context, JsonDocument jd)

{

Guid token = new Guid(jd.RootElement.GetProperty("body").GetProperty("token").ToString());

Guid thread = new Guid(jd.RootElement.GetProperty("body").GetProperty("thread").ToString());

string content = jd.RootElement.GetProperty("body").GetProperty("content").ToString();

// Создание нового комментария в базе данных

Guid? id = Database.Hinstance.CreateComment(token, thread, content);

var response = context.Response;

response.Headers.ContentType = "application/json; charset=utf-8";

if (id is null)

{

await response.WriteAsync("{\"code\":\"101\",\"body\":{\"id\":null}}");

return;

}

await response.WriteAsync("{\"code\":\"101\",\"body\":{\"id\":\"" + id + "\"}}");

Program.app.Logger.LogWarning("{\"code\":\"101\",\"body\":{\"id\":\"" + id + "\"}}");

}

}

[AttributeUsage(AttributeTargets.Method)]

public class ApiCodeAttribute : Attribute

{

private int \_code;

public ApiCodeAttribute(int code)

{

this.\_code = code;

}

public int Code

{

get

{

return this.\_code;

}

}

}

async function authorization() {

let username = document.getElementById("username").value;

let password = document.getElementById("password").value;

let regex\_user = /^[a-z0-9\_.]{4,32}$/;

let regex\_pass = /^.{8,64}$/;

if (!regex\_user.test(username)) {

registration\_er("Имя пользователя должно состоять только из латинских символов нижнего регистра, цифр, точек и нижних подчеркиваний. От 4 до 32 символов");

return;

}

if (!regex\_pass.test(password)) {

registration\_er("Пароль должен состоять из 8 до 64 символов");

return;

}

let data =

{

code: 30,

body: {

username: username,

password: password

}

};

data = await toapi(data);

if (data.body.status == "OK") {

document.location = '/auth/success.html';

}

else {

registration\_er("Введены неверные данные");

}

}

function registration\_er(msg) {

let el = document.getElementById("error\_field");

el.style.color = "rgba(255, 0, 0, 0.83)";

el.textContent = msg;

}

async function toapi(data) {

let ret = {};

await fetch('/api', {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify(data)

})

.then(response => response.json())

.then(data => ret = data)

.catch(error => {

console.error(error);

ret = {};

});

return ret;

}

async function createc() {

let token = getCookie('\_t');

let id = getParametr("id");

if (token === null) {

registration\_er("Вы не авторизированы");

return;

}

let content = document.getElementById("content").value;

let regex\_content = /^.{2,2048}$/;

if (!regex\_content.test(content)) {

registration\_er("Размер содержимого должно быть от 2 до 2048 символов");

return;

}

let data =

{

code: 100,

body: {

token: token,

thread: id,

content: content

}

};

alert(JSON.stringify(data));

data = await toapi(data);

alert(JSON.stringify(data));

if (data === {}) {

registration\_er("Не удалось создать комментарий");

}

if (!(data.body.id === null)) {

location.reload()

}

else {

registration\_er("Не удалось создать комментарий");

}

}

function getCookie(name) {

const cookies = document.cookie.split(';');

for (let i = 0; i < cookies.length; i++) {

const cookie = cookies[i].trim();

if (cookie.startsWith(name + '=')) {

return cookie.substring(name.length + 1);

}

}

return null;

}

function registration\_er(msg) {

let el = document.getElementById("error\_field");

el.style.color = "rgba(255, 0, 0, 0.83)";

el.textContent = msg;

}

function getParametr(name) {

return new URLSearchParams(window.location.search).get(name);

}

async function toapi(data) {

let ret = {};

await fetch('/api', {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify(data)

})

.then(response => response.json())

.then(data => ret = data)

.catch(error => {

console.error(error);

ret = {};

});

return ret;

}

export function getCookie(name) {

const cookies = document.cookie.split(';');

for (let i = 0; i < cookies.length; i++) {

const cookie = cookies[i].trim();

if (cookie.startsWith(name + '=')) {

return cookie.substring(name.length + 1);

}

}

return null;

}

export function deleteCookie(name) {

document.cookie = name + '=; Path=/; Expires=Thu, 01 Jan 1970 00:00:01 GMT;';

}

export function getParametr(name) {

return new URLSearchParams(window.location.search).get(name);

}

export function sleep(ms) {

return new Promise(resolve => setTimeout(resolve, ms));

}

export async function toapi(data) {

let ret = {};

await fetch('/api', {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify(data)

})

.then(response => response.json())

.then(data => ret = data)

.catch(error => {

console.error(error);

ret = {};

});

return ret;

}

async function createt() {

let token = getCookie('\_t');

if (token === null) {

registration\_er("Вы не авторизированы");

return;

}

let title = document.getElementById("title").value;

let content = document.getElementById("content").value;

let regex\_title = /^.{2,64}$/;

let regex\_content = /^.{2,2048}$/;

if (!regex\_title.test(title)) {

registration\_er("Размер залоговка должна быть от 2 до 64 символов");

return;

}

if (!regex\_content.test(content)) {

registration\_er("Размер Содержимого должно быть от 2 до 2048 символов");

return;

}

let data =

{

code: 70,

body: {

token: token,

title: title,

content: content

}

};

alert(JSON.stringify(data));

data = await toapi(data);

alert(JSON.stringify(data));

if (data === {}) {

registration\_er("Не удалось создать ветку");

}

if (!(data.body.id === null)) {

document.location = '/t?id=' + data.body.id;

}

else {

registration\_er("Не удалось создать ветку");

}

}

function getCookie(name) {

const cookies = document.cookie.split(';');

for (let i = 0; i < cookies.length; i++) {

const cookie = cookies[i].trim();

if (cookie.startsWith(name + '=')) {

return cookie.substring(name.length + 1);

}

}

return null;

}

function registration\_er(msg) {

let el = document.getElementById("error\_field");

el.style.color = "rgba(255, 0, 0, 0.83)";

el.textContent = msg;

}

async function toapi(data) {

let ret = {};

await fetch('/api', {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify(data)

})

.then(response => response.json())

.then(data => ret = data)

.catch(error => {

console.error(error);

ret = {};

});

return ret;

}

function loading() {

loadyou();

}

async function loadyou() {

let token = getCookie('\_t');

if (token == "\_") {

return;

}

let data =

{

code: 42,

body: {

token: token

}

};

data = await toapi(data);

if (data.body.status == "OK") {

elem = document.getElementById("you-user");

elem.innerHTML = data.body.user.username;

elem.setAttribute('href', '/u?id=' + data.body.user.id);

}

}

function getCookie(name) {

const cookies = document.cookie.split(';');

for (let i = 0; i < cookies.length; i++) {

const cookie = cookies[i].trim();

if (cookie.startsWith(name + '=')) {

return cookie.substring(name.length + 1);

}

}

return null;

}

async function toapi(data) {

let ret = {};

await fetch('/api', {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify(data)

})

.then(response => response.json())

.then(data => ret = data)

.catch(error => {

console.error(error);

ret = {};

});

return ret;

}

loading();

import \* as common from "/js/common.js";

async function loading() {

let data = {

code: 52,

body: {

count: 10

}

};

data = await common.toapi(data);

if (data == {}) {

return;

}

let threads = data.body.threads;

let main\_block = document.getElementById("mainmainmain");

for (let i = 0; i < threads.length; i++) {

// alert(threads[i]);

main\_block.insertAdjacentElement('afterbegin', await makeblock(threads[i]));

}

}

async function makeblock(id) {

let data = {

code: 62,

body: {

id: id

}

};

data = await common.toapi(data);

if (data == {}) {

return;

}

let elem\_block = document.createElement("div");

elem\_block.className = "block";

let str = ' <a href="/t?id=' + data.body.id + '">' + //

' <div class="title">' +

data.body.title + //

'</div>' +

'</a>' +

'<div class="second-block">' +

'<a href="/u?id=' + data.body.author + '">' +

'<div class="sub-title author">' +

'Автор' +

'</div>' +

'</a>' +

'<div class="space-point">•</div>' +

'<div class="sub-title" style="min-width: 300px;">' +

data.body.date +

'</div>' +

'<div class="sub-title-space"> </div>' +

'<a onclick="alert(" test");">' +

'<div class="sub-title">' +

'Лайки&nbsp;' + data.body.karma\_count +

'</div>' +

'</a>' +

'</div>';

elem\_block.innerHTML = str;

return elem\_block;

// let elem1 = document.createElement("a");

// elem1.href = "/t?101010";

// let elem2 = document.createElement("div");

// elem2.className = "title";

// elem2.innerHTML = data.body.title;

// elem1.insertAdjacentElement('afterbegin', elem2);

// elem\_block.insertAdjacentElement('afterbegin', elem1);

// elem1 = document.createElement("div");

// elem1.className = "content";

// elem1.innerHTML = data.body.body;

// elem\_block.insertAdjacentElement('afterbegin', elem1);

// elem1 = document.createElement("div");

// elem1.className = "second-block";

// elem2 = document.createElement("a");

// elem2.href = "/u?id=1";

// let elem3 = document.createElement('div');

// elem3.className = "sub-title author";

// elem3.innerHTML = "author";

// elem2.insertAdjacentElement('afterbegin', elem3);

// elem1.insertAdjacentElement('afterbegin', elem2);

}

loading();

async function registration() {

let username = document.getElementById("username").value;

let email = document.getElementById("email").value;

let password = document.getElementById("password").value;

let password2 = document.getElementById("password2").value;

if (password != password2) {

registration\_er("Пароли не совпадают");

}

let regex\_user = /^[a-z0-9\_.]{4,32}$/;

let regex\_pass = /^.{8,64}$/;

if (!regex\_user.test(username)) {

registration\_er("Имя пользователя должно состоять только из латинских символов нижнего регистра, цифр, точек и нижних подчеркиваний. От 4 до 32 символов");

return;

}

if (!regex\_pass.test(password)) {

registration\_er("Пароль должен состоять из 8 до 64 символов");

return;

}

let data =

{

code: 20,

body: {

username: username,

email: email,

password: password

}

};

alert()

data = await toapi(data);

if (data == {}) {

registration\_er("Введены неверные данные");

}

if (data.body.status == "OK") {

document.location = '/reg/success.html';

}

else {

registration\_er("Введены неверные данные");

}

}

function registration\_er(msg) {

let el = document.getElementById("error\_field");

el.style.color = "rgba(255, 0, 0, 0.83)";

el.textContent = msg;

}

async function toapi(data) {

let ret = {};

await fetch('/api', {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify(data)

})

.then(response => response.json())

.then(data => ret = data)

.catch(error => {

console.error(error);

ret = {};

});

return ret;

}

import \* as common from "/js/common.js";

async function loading() {

let main\_block = document.getElementById("mainmainmain");

let id = common.getParametr("id")

let data = {

code: 82,

body: {

thread: id

}

};

data = await common.toapi(data);

if (data === {}) {

return;

}

let comments = data.body.comments;

for (let i = comments.length - 1; i >= 0; i--) {

// alert(threads[i]);

main\_block.insertAdjacentElement('afterbegin', await makeblock2(comments[i]));

}

if (comments.length != 0) {

main\_block.insertAdjacentElement('afterbegin', document.createElement('hr'));

}

main\_block.insertAdjacentElement('afterbegin', await makeblock(id));

}

async function makeblock(id) {

let data = {

code: 62,

body: {

id: id

}

};

data = await common.toapi(data);

if (data === {}) {

return;

}

let elem\_block = document.createElement("div");

elem\_block.className = "block";

let str =

' <div class="title">' +

data.body.title + //

'</div>' +

'<div class="content">' +

data.body.body +

'</div>' +

'<div class="second-block">' +

'<a href="/u?id=' + data.body.author + '">' +

'<div class="sub-title author">' +

'Автор' +

'</div>' +

'</a>' +

'<div class="space-point">•</div>' +

'<div class="sub-title" style="min-width: 300px;">' +

data.body.date +

'</div>' +

'<div class="sub-title-space"> </div>' +

'<a onclick="alert(" test");">' +

'<div class="sub-title">' +

// 'Лайки&nbsp;' + data.body.karma\_count +

'</div>' +

'</a>' +

'</div>';

elem\_block.innerHTML = str;

return elem\_block;

}

async function makeblock2(id) {

let data = {

code: 92,

body: {

id: id

}

};

data = await common.toapi(data);

if (data === {}) {

return;

}

let elem\_block = document.createElement("div");

elem\_block.className = "block";

let str =

'<div class="content">' +

data.body.content +

'</div>' +

'<div class="second-block">' +

'<a href="/u?id=' + data.body.author + '">' +

'<div class="sub-title author">' +

'Автор' +

'</div>' +

'</a>' +

'<div class="space-point">•</div>' +

'<div class="sub-title" style="min-width: 300px;">' +

data.body.date +

'</div>' +

'<div class="sub-title-space"> </div>' +

'<a onclick="alert(" test");">' +

'<div class="sub-title">' +

// 'Лайки&nbsp;' + data.body.karma\_count +

'</div>' +

'</a>' +

'</div>';

elem\_block.innerHTML = str;

return elem\_block;

}

loading();

import \* as common from "/js/common.js";

async function loading() {

let token = common.getCookie('\_t');

await loadyou(token);

//await common.sleep(1500);

let id = common.getParametr("id");

await loaduser(id, false);

if (document.getElementById("you-user").innerHTML == document.getElementById("username").innerHTML) {

let elem = document.getElementById("you-user");

elem.removeAttribute("href");

elem.setAttribute("onclick", "exit()");

elem.innerHTML = "Выйти из аккаунта";

}

}

async function loadyou(token) {

if(token === null){

return;

}

let data =

{

code: 42,

body: {

token: token

}

}

data = await common.toapi(data);

if (data === {}) {

return;

}

if (data.body.status == "OK") {

let elem = document.getElementById("you-user");

elem.innerHTML = data.body.user.username;

elem.setAttribute('href', '/u?id=' + data.body.user.id);

}

}

async function loaduser(id, email) {

let data =

{

code: 42,

body: {

id: id

}

}

data = await common.toapi(data);

if (data == {}) {

return;

}

if (data.body.status == "OK") {

let elem = document.getElementById("username");

elem.innerHTML = data.body.user.username;

elem = document.getElementById("date");

elem.innerHTML = data.body.user.date;

elem = document.getElementById("role");

if (data.body.user.role == "normal") {

elem.innerHTML = "Пользователь";

}

if (email) {

elem = document.getElementById("email");

elem.innerHTML = data.body.user.email;

}

}

}

loading();

# Приложение Б Результаты работы приложения

Проведенные ручные тестирования веб-приложения.

1. Тест на регистрацию нового пользователя:
   * Входные данные: заполненные поля формы регистрации с уникальными данными (имя, электронная почта, пароль).
   * Ожидаемый результат: успешная регистрация, создание нового пользователя в базе данных, переход на главную страницу.
2. Тест на добавление новой ветки обсуждения:
   * Входные данные: заполненные поля формы добавления новой ветки обсуждения (название ветки, описание, теги).
   * Ожидаемый результат: успешное добавление ветки обсуждения, отображение новой ветки на странице со списком веток обсуждения.
3. Тест на добавление комментария к ветке обсуждения:
   * Входные данные: текст комментария, выбранная ветка обсуждения.
   * Ожидаемый результат: успешное добавление комментария, отображение комментария на странице ветки обсуждения.
4. Тест на редактирование профиля пользователя:
   * Входные данные: изменение данных в профиле пользователя (имя, электронная почта, пароль).
   * Ожидаемый результат: успешное сохранение изменений, обновление данных в базе данных, отображение обновленных данных в профиле пользователя.
5. Тест на поиск ветки обсуждения по ключевым словам:
   * Входные данные: ввод ключевых слов в поле поиска.
   * Ожидаемый результат: отображение списка веток обсуждения, содержащих указанные ключевые слова.

КП Т.091020.401 ГЧ

КБиП

У

Т. Контр.

Разраб.

Черняков В.А.

Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ Докум.

Подпись

Дат

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 1

Инв.№подл.

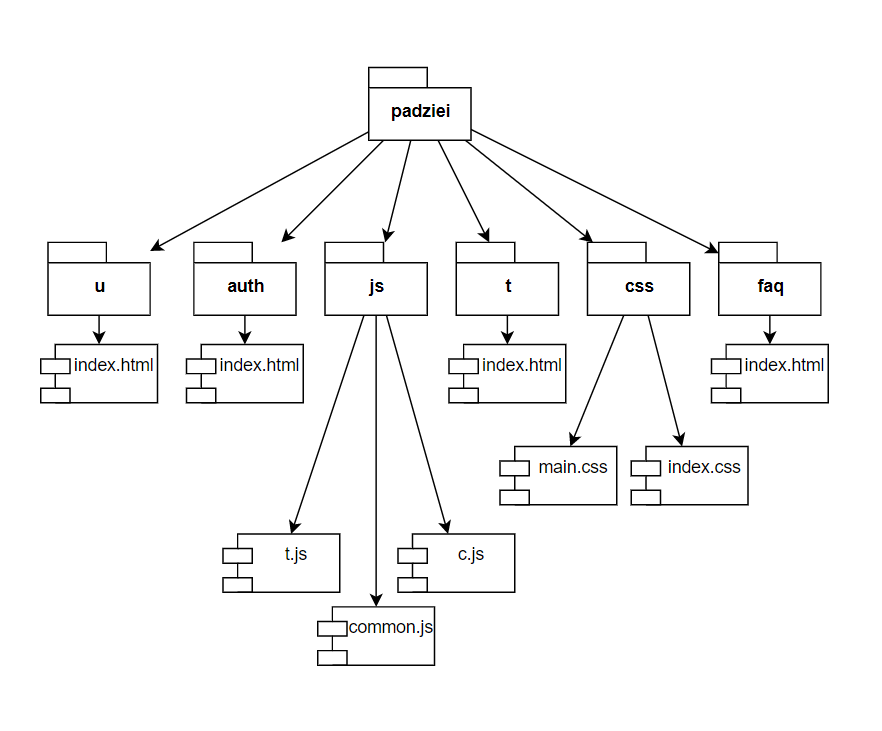
Подп. и дата

Взам.инв.№в.

Инв.№ дубл.

Подп. и дата

КП Т.093026.401 ГЧ



*Диаграмма развертывания*

*Веб-приложение форум «Padziei»*

# Приложение Г В (Обязательное) Блок-схемы работы алгоритмов

КБиП

У

Т. Контр.

Разраб.

Черняков В.А.

Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ Докум.

Подпись

Дат

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 1

Инв.№подл.

Подп. и дата

Взам.инв.№в.

Инв.№ дубл.

Подп. и дата

КП Т.093026.401 ГЧ

КП Т.091020.401 ГЧ

*Диаграмма классов*

*Веб-приложение форум «Padziei»*

КП Т.091020.401 ГЧ

КБиП

У

Т. Контр.

Разраб.

Черняков В.А.

Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ Докум.

Подпись

Дат

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 1

Инв.№подл.

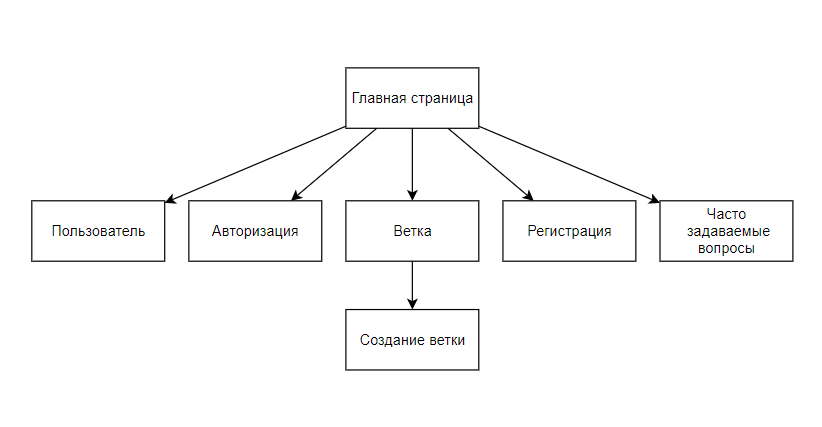
Подп. и дата

Взам.инв.№в.

Инв.№ дубл.

Подп. и дата

КП Т.093026.401 ГЧ



*Диаграмма структуры сайта*

*Веб-приложение форум «Padziei»*

КП Т.091020.401 ГЧ

КБиП

У

Т. Контр.

Разраб.

Черняков В.А.

Провер.

Кривошеина А.В.

Реценз.

Утверд.

Н. Контр.

№ Докум.

Подпись

Дат

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 1

Инв.№подл.

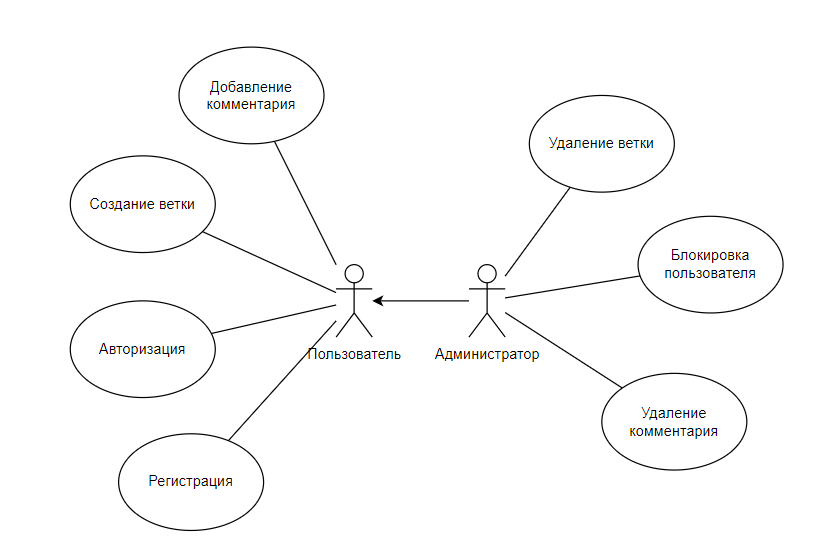
Подп. и дата

Взам.инв.№в.

Инв.№ дубл.

Подп. и дата

КП Т.093026.401 ГЧ



*Диаграмма вариантов использования*

*Веб-приложение форум «Padziei»*

Разраб.