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](#_Toc138832308)

[1 Описание задачи 5](#_Toc138832309)

[1.1 Анализ предметной области 5](#_Toc138832310)

[1.2 Постановка задачи 6](#_Toc138832311)

[2 Проектирование веб-приложения 7](#_Toc138832312)

[2.1 Проектирование модели 7](#_Toc138832313)

[2.2 Требования к веб-приложению 7](#_Toc138832314)

[2.3 Структура веб-приложения 9](#_Toc138832315)

[2.4 Проектирование макета веб-приложения 9](#_Toc138832316)

[2.5 Программно-технические средства, необходимые для разработки приложения 10](#_Toc138832317)

[2.6 Защита и сохранность данных 11](#_Toc138832318)

[2.7 Организация и ведение информационной базы 11](#_Toc138832319)

[3 Реализация веб-приложения 13](#_Toc138832320)

[3.1 Описание разделов веб-приложения 13](#_Toc138832321)

[3.2 Разработка разделов веб-приложения 14](#_Toc138832322)

[3.3 Разработка клиентской части приложения 15](#_Toc138832323)

[3.4 Описание используемых функций и процедур 15](#_Toc138832324)

[3.5 Функциональное тестирование 16](#_Toc138832325)

[4 Применение 17](#_Toc138832326)

[4.1 Назначение веб-приложения 17](#_Toc138832327)

[4.2 Программно-аппаратное обеспечение сервера и клиента 17](#_Toc138832328)

[Заключение 18](#_Toc138832329)

[Список информационных источников 20](#_Toc138832330)

[ПРИЛОЖЕНИЕ А (Обязательное) Текст модулей Веб-приложения 21](#_Toc138832331)

[ПРИЛОЖЕНИЕ Б (Обязательное) Результаты работы приложения 58](#_Toc138832332)

[ПРИЛОЖЕНИЕ В (Обязательное) Блок-схемы работы алгоритмов 60](#_Toc138832333)

# Введение

В рамках курсового проекта представляется пояснительная записка для разработки веб-приложения "Padziei" - форума, предоставляющего возможность регистрации, создания веток и комментариев, а также функций администрирования. Задача заключается в создании функционального и удобного веб-приложения, способного обеспечить активное взаимодействие и обмен информацией между участниками форума.

Актуальность поставленной задачи обусловлена необходимостью предоставления платформы, на которой пользователи смогут обсуждать интересующие их темы, делиться мнениями и получать информацию от других участников. Форумы являются важным инструментом социального взаимодействия в сети, способствующим обмену опытом и знаниями. Разработка функционального веб-приложения для форума предоставляет возможность участникам активно участвовать в обсуждениях и находить интересующую информацию в удобной и доступной форме.

Пояснительная записка для курсового проекта включает анализ предметной области, где проводится исследование бизнес-процессов и происходящих в них событий, а также обосновывается необходимость компьютерной обработки информации и процессов. Затем формулируется постановка задачи, в которой определяются функции, подлежащие автоматизации, и описываются отличия разрабатываемого веб-приложения от существующих аналогов. Далее представляется проектирование модели, где приводятся диаграммы вариантов использования, классов, структуры веб-приложения и диаграммы развертывания.

Также в пояснительной записке описываются разделы веб-приложения, включая главную страницу, навигационное меню и административную часть. Отдельно рассматривается разработка административной части, где описываются логическая и физическая структура серверной части, а также навигация в этой части.

В заключении подводятся итоги выполненной работы, отражается поставленная задача и ее выполнение, описываются разработанные функции, а также преимущества и недостатки разработанного веб-приложения. В целом, пояснительная записка представляет собой полное описание разработки веб-приложения "Padziei" и включает в себя все необходимые аспекты для оценки и понимания работы.

# Описание задачи

## Анализ предметной области

Предметная область, связанная с веб-форумами, представляет собой виртуальное пространство, где пользователи сходятся, чтобы обмениваться информацией, делиться опытом, задавать вопросы и находить ответы на них. Бизнес-процессы, происходящие в данной предметной области, включают регистрацию и авторизацию пользователей, создание новых веток для обсуждения, добавление комментариев к веткам, а также управление содержимым форума администратором.

Для обеспечения функциональности "Padziei" необходимы следующие алгоритмы:

– aлгоритм регистрации. Пользователи предоставляют свои данные, которые проверяются на валидность и уникальность. После успешной регистрации пользователю присваивается уникальный идентификатор;

– aлгоритм авторизации. Пользователи, уже зарегистрированные в системе, предоставляют свои учетные данные для входа в приложение. Данные проверяются на соответствие с зарегистрированными данными, и в случае успеха пользователь получает доступ к своему аккаунту;

– aлгоритм создания веток. Пользователи могут создавать новые ветки для обсуждения, указывая заголовок и содержание ветки;

– aлгоритм добавления комментариев. Пользователи могут добавлять комментарии к существующим веткам, указывая свои мнения и отзывы по теме;

– aлгоритм управления содержимым форума администратором. Администратор имеет возможность удаления веток и комментариев, а также блокировки участников, нарушающих правила форума.

Компьютерная обработка информации и процессов в веб-приложении форума "Padziei" обеспечивает эффективную и удобную работу с данными для пользователей и администратора. Она позволяет автоматизировать процессы регистрации, авторизации, создания и управления ветками, добавления комментариев и контроля за содержимым форума. Кроме того, компьютерная обработка информации обеспечивает сохранность данных, облегчает навигацию и обеспечивает быстрый доступ к нужным разделам форума.

Веб-приложение форума "Padziei" предназначено для широкой аудитории пользователей, интересующихся обменом знаниями, обсуждением тем и получением ответов на свои вопросы. Потенциальные пользователи включают студентов, ученых, практикующих специалистов, преподавателей и всех, кто желает вести активную дискуссию и делиться своими мыслями в различных областях знаний. Удобство использования веб-приложения и разнообразие тематик на форуме делают его привлекательным для широкого круга пользователей, стремящихся получить и предоставить информацию, опыт и экспертизу.

## Постановка задачи

На основе проанализированных бизнес-процессов и бизнес-задач, выделены определенные функции, которые требуется автоматизировать в разрабатываемом веб-приложении форума "Padziei". Целью разработки является создание удобного и функционального инструмента, который обеспечит эффективное взаимодействие пользователей и управление содержимым форума.

Определенные функции, подлежащие автоматизации, включают:

– регистрация и авторизация пользователей: создание механизма, позволяющего пользователям зарегистрироваться в системе и получить доступ к своему персональному аккаунту. Это обеспечит индивидуальные настройки, возможность управления созданными ветками и комментариями, а также сохранение истории взаимодействия с форумом;

– создание новых веток: предоставление пользователю возможности создания новых тематических веток для обсуждения. Это позволит организовать структурированное пространство для обмена информацией и суждениями, а также обеспечит пользователей возможностью участвовать в активных дискуссиях;

– добавление комментариев: обеспечение пользователей возможностью добавлять комментарии к существующим веткам. Это сделает обсуждение более интерактивным, позволит выражать мнения, отвечать на вопросы и поддерживать взаимодействие между участниками;

– управление содержимым форума администратором: предоставление администратору инструментов для удаления веток и комментариев, а также блокировки участников, нарушающих правила форума. Это обеспечит поддержание порядка и качества содержимого, а также защиту от нежелательной активности и нарушений правил использования форума.

При анализе существующих аналогов было выявлено несколько веб-приложений форумов, предоставляющих схожий функционал. Однако, разрабатываемое программное средство "Padziei" имеет ряд отличий и улучшений, которые делают его привлекательным для пользователей. В отличие от некоторых аналогов, "Padziei" обладает следующими преимуществами:

– интуитивно понятный и привлекательный интерфейс, обеспечивающий удобство использования и навигацию по форуму;

– улучшенная система управления содержимым, позволяющая администратору более гибко контролировать форум и поддерживать высокое качество содержания;

– современные механизмы безопасности и защиты данных, обеспечивающие сохранность информации пользователей.

Таким образом, разрабатываемое программное средство "Padziei" будет предоставлять возможность автоматизации ключевых функций, необходимых для эффективного функционирования веб-форума. Оно отличается от существующих аналогов своим удобством использования, персонализированным опытом, улучшенной системой управления содержимым и обеспечением безопасности данных пользователей.

# Проектирование веб-приложения

## Проектирование модели

В процессе проектирования модели веб-приложения форума "Padziei" были разработаны следующие диаграммы, которые являются важными инструментами для понимания структуры и взаимодействия компонентов системы. Графические представления этих диаграмм находятся в приложении В.

Первой разработанной диаграммой является диаграмма вариантов использования. На данной диаграмме представлены основные сценарии взаимодействия пользователей с веб-приложением. Для каждого из этих прецедентов в тексте пояснительной записки будут представлены соответствующие сценарии, описывающие последовательность действий пользователя и системы.

Следующая разработанная диаграмма - диаграмма классов. Она иллюстрирует модель данных, используемых в веб-приложении "Padziei". Каждый класс имеет свои атрибуты и/или методы, которые определяют их характеристики и функциональность. Эта диаграмма позволяет лучше понять структуру данных и связи между различными компонентами системы.

Структура веб-приложения также была отражена в разработанной диаграмме. Эта диаграмма показывает компоненты веб-приложения и их взаимосвязи. Каждый компонент выполняет определенные функции и взаимодействует с другими компонентами для обеспечения работы приложения. Данная диаграмма помогает визуализировать структуру системы и понять, как компоненты взаимодействуют друг с другом.

Наконец, была разработана диаграмма развертывания, которая отображает физическое размещение компонентов системы и их взаимодействие. Взаимодействие между этими узлами позволяет пользователям получать доступ к веб-приложению и осуществлять необходимые операции. Данная диаграмма позволяет лучше понять архитектуру системы и распределение компонентов.

Таким образом, разработанные диаграммы играют важную роль в проектировании модели веб-приложения "Padziei". Они предоставляют понятное и наглядное представление о вариантах использования, модели данных, структуре веб-приложения и развертывании компонентов. Эти диаграммы можно найти в графической части работы, где они более подробно иллюстрируют суть спроектированных компонентов системы.

## Требования 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-технические 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

## Описание разделов веб-приложения

Главная страница веб-приложения "Padziei" состоит из двух разделов: административной и клиентской части. Каждый из этих разделов обладает своей спецификой и функциональностью, предоставляя пользователям соответствующие возможности и инструменты.

Административная часть главной страницы предназначена для администраторов форума. Здесь администраторы могут осуществлять управление содержимым и пользователями. На административной странице предоставляются функции, такие как удаление веток и комментариев, блокировка участников, а также просмотр и анализ статистических данных о форуме. Административная часть обеспечивает администраторам полный контроль и возможность поддержания порядка на форуме.

Клиентская часть главной страницы предназначена для обычных пользователей, зарегистрированных на форуме. Здесь пользователи могут получить доступ к обсуждениям, создавать новые ветки и оставлять комментарии. Клиентская часть предоставляет удобный интерфейс для навигации по форуму, поиска интересующих тем и участия в дискуссиях. Пользователи могут просматривать последние обновления, следить за активностью веток и обмениваться мнениями с другими участниками.

В навигационном меню веб-приложения "Padziei" предоставляются различные пункты, которые обеспечивают навигацию и доступ к различным функциям приложения. При выборе каждого пункта меню выполняются определенные действия, направленные на удовлетворение потребностей пользователей.

Пункты меню в клиентской части включают:

– переход на главную страницу форума, где отображаются последние обновления и активные ветки;

– позволяет просмотреть список доступных форумов и выбрать интересующий для участия в обсуждениях;

– при выборе этого пункта меню пользователь перенаправляется на страницу создания новой ветки для начала собственной темы обсуждения;

– предоставляет доступ к персональному профилю пользователя, где можно настроить информацию, просмотреть созданные ветки и комментарии, а также управлять настройками аккаунта.

Пункты меню в административной части включают:

– переход на административную главную страницу с обзором основных статистических данных и функций управления форумом;

– позволяет администраторам просматривать и удалять созданные ветки, поддерживать порядок и качество содержания форума;

– предоставляет возможность администраторам просматривать и удалять комментарии, контролируя обсуждения и поддерживая приемлемые нормы общения;

– позволяет администраторам управлять аккаунтами пользователей, блокировать или разблокировать участников форума.

Таким образом, главная страница веб-приложения "Padziei" разделена на административную и клиентскую части, предоставляющие соответствующие функциональные возможности. Навигационное меню обеспечивает удобную навигацию и доступ к основным функциям приложения, в зависимости от статуса пользователя (администратор или обычный участник).

## Разработка разделов веб-приложения

Разработка административной части приложения "Padziei" включает описание логической и физической структуры серверной части веб-приложения, а также навигации внутри приложения.

Логическая структура административной части приложения представляет собой карту функциональных модулей и компонентов, которые обеспечивают управление содержимым и пользователями. Она состоит из различных модулей, таких как "Управление ветками", "Управление комментариями" и "Управление пользователями". Каждый модуль выполняет определенные задачи и предоставляет соответствующие функции для администрирования форума. Логическая структура позволяет администраторам эффективно управлять различными аспектами форума и поддерживать его работоспособность.

Физическая структура административной части приложения определяет организацию файлов и каталогов на сервере. Веб-приложение "Padziei" состоит из различных файлов, которые хранятся в определенных каталогах на сервере. Например, файлы модулей управления ветками, комментариями и пользователями располагаются в соответствующих каталогах. Физическая структура обеспечивает правильное размещение файлов и удобный доступ к ним для функционирования административной части приложения.

Навигация в административной части приложения осуществляется с помощью меню и гиперссылок. Меню предоставляет администраторам набор пунктов, каждый из которых соответствует определенному модулю или функциональности. При выборе пункта меню администратор перенаправляется на соответствующую страницу или модуль для выполнения требуемых операций. Гиперссылки также используются для обеспечения быстрого перехода между различными разделами и модулями административной части приложения.

Таким образом, разработка административной части приложения "Padziei" включает описание логической и физической структуры серверной части, которые обеспечивают функциональность управления содержимым и пользователями. Навигация осуществляется через меню и гиперссылки, обеспечивая удобство использования и эффективное администрирование форума.

## 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 А.

## 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 Б.

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

# Применение

## Назначение веб-приложения

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" была поставлена задача создания функционального форума, предоставляющего возможности регистрации, авторизации, создания новых веток и добавления комментариев. Также важными аспектами были функции администрирования, включающие удаление веток и комментариев, блокировку участников.

Для разработки веб-приложения были использованы современные технологии веб-разработки, включая фреймворк для создания серверной части, базу данных для хранения информации о форуме и клиентский интерфейс для взаимодействия с пользователями. В результате выполнения поставленной задачи было разработано функциональное веб-приложение "Padziei", способное обеспечить полноценный форум с широкими возможностями для пользователей и администраторов.

Веб-приложение "Padziei" успешно реализует все поставленные задачи. Пользователи могут зарегистрироваться, авторизоваться и создавать новые ветки для обсуждения интересующих тем. Они также имеют возможность добавлять комментарии к существующим веткам и взаимодействовать с другими участниками форума. Администраторы имеют доступ к функциям управления, позволяющим поддерживать порядок на форуме, удалять ветки и комментарии, а также блокировать участников.

Разработанное веб-приложение "Padziei" обладает следующими функциями:

– регистрация и авторизация пользователей;

– создание новых веток и добавление комментариев;

– удаление веток и комментариев администраторами;

– блокировка участников форума администраторами.

Преимущества разработанного веб-приложения включают его удобный и интуитивно понятный интерфейс, который облегчает навигацию и использование форума для пользователей всех уровней. Реализованные функции администрирования позволяют эффективно поддерживать порядок и контролировать содержимое форума. Кроме того, веб-приложение предоставляет возможность взаимодействия и обмена мнениями между участниками, способствуя активному общению и обмену информацией.

Однако у разработанного веб-приложения также есть некоторые недостатки. Например, в настоящей версии отсутствуют некоторые дополнительные функции, такие как возможность прикрепления файлов к комментариям или веткам. Также можно отметить, что безопасность и защита данных пользователей являются важными аспектами, которые требуют дополнительного внимания и обеспечения соответствующих механизмов защиты.

В целом, разработанное веб-приложение "Padziei" успешно выполняет поставленную задачу, предоставляя функциональный форум с возможностями регистрации, создания веток и комментариев, а также функции администрирования. Он обладает удобным интерфейсом и способствует активному общению между участниками. Однако, для дальнейшего развития и совершенствования приложения, рекомендуется добавить дополнительные функции и обеспечить высокий уровень безопасности данных пользователей.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2017. – 30 c.
2. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.
3. Бондарь, А.Г. Microsoft SQL Server 2012 / А.Г. Бондарь. – СПб. : БХВ-Петербург, 2013. – 608 с.
4. Крум, Э.В. Экономика предприятия. Практикум / Э. В. Крум. – Минск : Издательство Гревцова, 2009. – 360 с.
5. Мацяшек, Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0 / Лешек А. Мацяшек. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2008. – 816 с.
6. Михнюк, Т.Ф. Охрана труда / Т.Ф. Михнюк. – Минск : ИВЦ Минфина, 2009. – 365 с.
7. Объектно-ориентированный анализ и проектирование с примерами приложений / Гради Буч [и др.]. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2008. – 720 с.
8. Тепляков, С. Паттерны проектирования на платформе .NET / С.Тепляков. – СПб. : Питер, 2015. – 320 с.
9. Экономика предприятия. Практикум / Э. В. Крум [и др.] ; под ред. Э. В. Крум. – Минск : Издательство Гревцова, 2009. – 355 с.
10. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
11. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
12. 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.
13. Руководство по программированию на C# [Электронный ресурс]. – Microsoft, 2019. – Режим доступа : http://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/. – Дата доступа : 25.04.2019.
14. Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2019. – Режим доступа : https://www.microsoft.com/ru-ru/windows/features. – Дата доступа : 22.05.2019.

# ПРИЛОЖЕНИЕ А (Обязательное) 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 (Обязательное) 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.

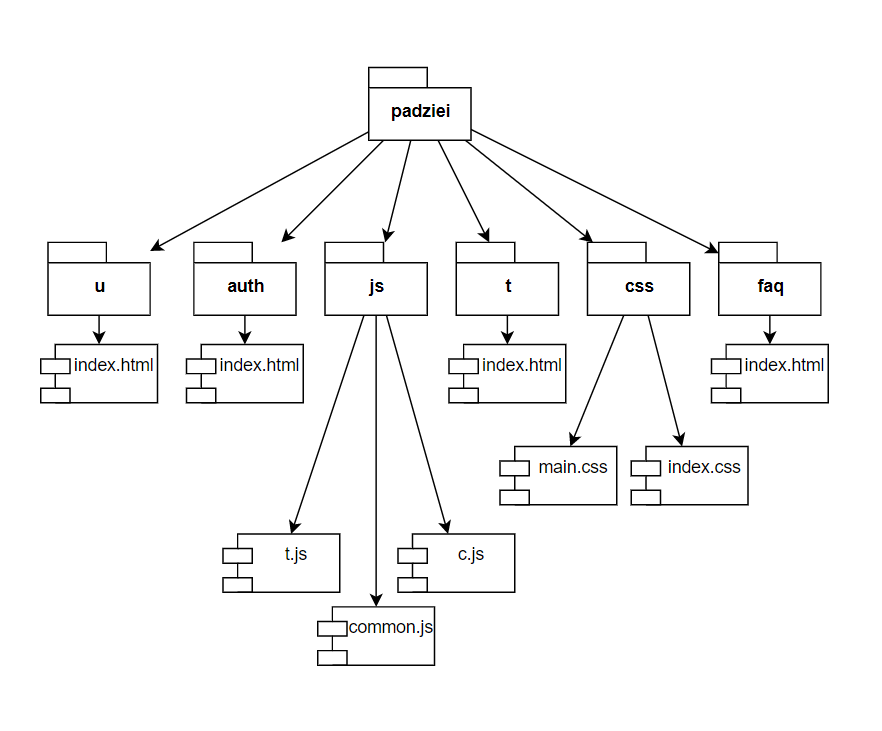
Подп. и дата

Взам.инв.№в.

Инв.№ дубл.

Подп. и дата

КП Т.093026.401 ГЧ



*Диаграмма развертывания*

*Веб-приложение форум «Padziei»*

# ПРИЛОЖЕНИЕ 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.

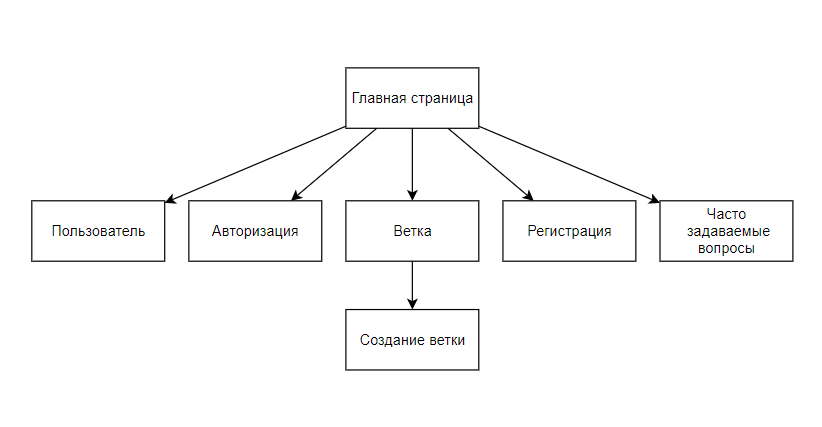
Подп. и дата

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.

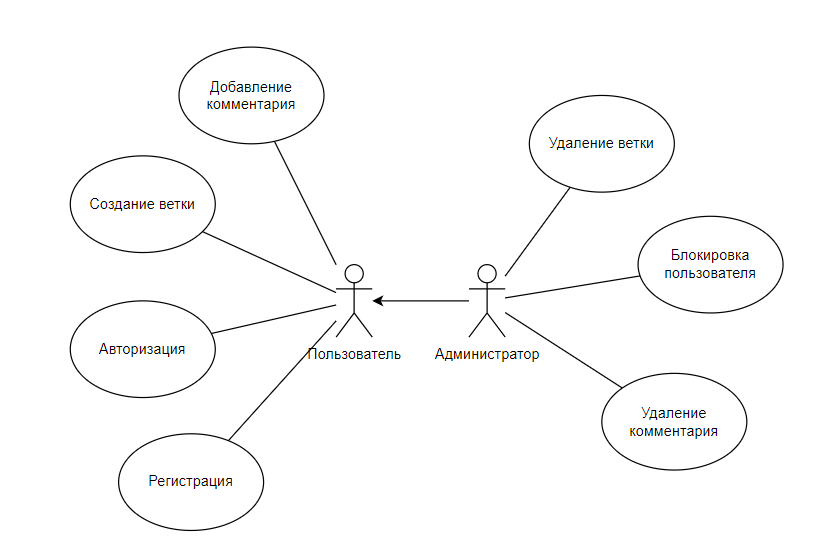
Подп. и дата

Взам.инв.№в.

Инв.№ дубл.

Подп. и дата

КП Т.093026.401 ГЧ



*Диаграмма вариантов использования*

*Веб-приложение форум «Padziei»*

Разраб.