

ОГЛАВЛЕНИЕ

ВЕДОМОСТЬ ОБЪЕМА ДИПЛОМНОГО ПРОЕКТА	
ВВЕДЕНИЕ	7
1 ОБЗОР ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ	8
2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	10
3 ПОСТАНОВКА ЗАДАЧИ	14
3.1 Общие определения	14
3.2 Требования к пользовательскому интерфейсу.....	15
4 ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ	17
4.1 Логическая модель	17
4.1.1 Основные роли в системе.....	17
4.1.2 Варианты использования системы.....	17
4.2 Модель взаимодействия клиентского приложения и сервера.....	19
5 ФИЗИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ	21
5.1 Проектирование пользовательского интерфейса.....	21
5.1.1 Проектирование пользовательского интерфейса для роли «Пользователь»	21
5.1.2 Проектирование пользовательского интерфейса для роли «Администратор»	23
5.2 Моделирование веб-страниц	24
6 РЕАЛИЗАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ	28
6.1 Интегрированная среда разработки Visual Studio Code	28
6.2 Визуальная часть	29
6.3 Программная часть.....	30
6.3.1 JavaScript (ReactJs).....	30
6.3.2 NPM.....	35
6.3.3 Webpack и Babel.....	36
6.3.4 React Router	36
6.3.5 Material UI.....	37
6.3.6 HTML	40
6.3.7 CSS 3 (SASS)	40
7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	41
8 РАЗВЕРТЫВАНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ.....	46
8.1 Развертывание клиентской части	46
8.2 Тестирование программной системы.....	47
8.2.1 Критическое тестирование	47
8.2.2 Углубленное тестирование	50
9 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ	

ПРОГРАММНОЙ СИСТЕМЫ.....	52
9.1 Расчет сметы затрат, цены и прибыли на программное средство.....	52
9.1.1 Исходные данные.....	53
9.1.2 Общая характеристика разрабатываемого программного средства.....	54
9.1.3 Определение объема программного средства	54
9.1.4 Расчет трудоемкости выполняемой работы.....	55
9.1.5 Расчет основной заработной платы исполнителей	57
9.1.6 Расчет дополнительной заработной платы исполнителей	59
9.1.7 Расчет отчислений в фонд социальной защиты населения	59
9.1.8 Расчет налога на ликвидацию последствий чернобыльской катастрофы.....	59
9.1.9 Расчет расходов по статье «Материалы»	60
9.1.10 Расчет расходов по статье «Спецоборудование»	60
9.1.11 Расчет расходов по статье «Машинное время»	61
9.1.12 Расчет расходов по статье «Научные командировки».....	61
9.1.13 Расчет расходов по статье «Прочие затраты».....	62
9.1.14 Расчет расходов по статье «Накладные расходы»	62
9.1.15 Расчет общей суммы расходов на разработку	62
9.1.16 Расчет прибыли и отпускной цены	63
9.2 Расчет экономического эффекта от применения ПС у пользователя.....	64
9.2.1 Исходные данные.....	64
9.2.2 Расчет объема работ	65
9.2.3 Расчет капитальных затрат	65
9.2.4 Расчет экономии основных видов ресурсов в связи с использованием нового программного средства.....	66
9.2.5 Расчет экономического эффекта	68
10 ОХРАНА ТРУДА	71
10.1 Производственная санитария и техника безопасности	71
10.1.1 Метеоусловия.....	71
10.1.2 Вентиляция и отопление	72
10.1.3 Освещение	73
10.1.4 Шум.....	73
10.1.5 Электробезопасность.....	74
10.1.6 Излучение	74
10.1.7 Пожарная безопасность.....	76
10.2 Требования к ВДТ, ЭВМ, ПЭВМ и периферийным устройствам.....	77
ЗАКЛЮЧЕНИЕ.....	78
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	
ГРАФИЧЕСКАЯ ЧАСТЬ	
ПРИЛОЖЕНИЕ	

ВВЕДЕНИЕ

В наши дни сложно представить себе компанию в той или иной сфере, которая ведет проекты без системы управления проектами. В условиях командной работы часто приходится планировать и распределять задачи между членами команды, а также оптимизировать рабочий процесс. Устная постановка задач, в условиях даже небольшого проекта, неудобна и неэффективна в использовании.

На проектах, где работу выполняют несколько человек, часто возникают следующие проблемы:

- 1) сложно запомнить какие задачи необходимо выполнить, и кто закреплен за ними, обеспечив при этом прозрачность;
- 2) необходимость часто переключаться между задачами или даже проектами;
- 3) трата времени работником на различные организационные вопросы: уведомление о своем прогрессе по важной задаче, передача работником выполненной в рамках его сферы задачи другому работнику из другой сферы с посвящением во все детали этой задачи и т.п.;
- 4) рассинхронизация между некоторыми членами команды: некоторые работники могут не видеть общей картины проекта и не владеть достаточной информацией, в том числе о занятости своих коллег.

Лучшим решением перечисленных выше проблем является автоматизация управленческой деятельности, которая может быть реализована в виде системы управления проектами. Это может быть настольное или веб-приложение.

Отдельной темой любого приложения является пользовательский интерфейс, поскольку это лицо приложения. Непривлекательный или не эргономичный интерфейс может оттолкнуть потенциального клиента или пользователя. Дизайн пользовательского интерфейса должен соответствовать целевой аудитории и задачам, которые непосредственно решает приложение. Не многим готовым решениям на данное время удалось выполнить все эти пункты.

Целью данного дипломного проекта является решение упомянутых выше вопросов, и разработка современного и удобного пользовательского интерфейса веб-системы для управления программными проектами.

1 ОБЗОР ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

Книга «JavaScript. подробное руководство, 6-е издание» Дэвида Флэнагана охватывает язык программирования JavaScript и прикладные интерфейсы JavaScript, реализованные в веб-браузерах. Книга делится на четыре части. Часть I охватывает сам язык JavaScript. Часть II охватывает клиентский JavaScript: прикладные программные интерфейсы JavaScript, определяемые стандартом HTML5 и сопутствующими ему стандартами и реализованные в веб-браузерах. Часть III представляет собой справочник по базовому языку, включающий описания всех классов, объектов, конструкторов, методов, функций, свойств и констант, определенных в JavaScript 1.8, V8 3.0 и ECMAScript 5. Часть IV – справочник по клиентскому JavaScript. Здесь описываются API веб-броузеров, стандарт DOM API Level 3 и недавно вошедшие в стандарт HTML5 технологии WebSockets и WebWorkers, объекты localStorage и sessionStorage. Шестое издание книги охватывает стандарты ECMAScript 5 (последняя версия спецификации базового языка) и HTML5 (последняя версия спецификации веб-платформы). Положения стандарта с 5 рассматриваются на протяжении всей первой части. Нововведения, появившиеся в HTML5, в основном рассматриваются в конце второй части. Совершенно новыми в этом издании являются глава 11 "Подмножества и расширения JavaScript", глава 12 "Серверный JavaScript", глава 19 "Библиотека jQuery" и глава 22 "Прикладные интерфейсы HTML5".

Для первого ознакомления с ReactJs была прочтена книга С. Стефанова «React.js. Быстрый старт». В этой книге рассмотрены основные концепции высокопроизводительного программирования при помощи React, реальные примеры кода и доступные блок-схемы. Книга описывает как использовать собственные компоненты React наряду с универсальными компонентами DOM, писать компоненты для таблиц данных, позволяющие редактировать, сортировать таблицу, выполнять в ней поиск и экспортировать ее содержимое. Так же доступно описан новый синтаксис JSX в качестве альтернативы для вызовов функций и инструментами ESLint, Flow и Jest, позволяющими проверять и тестировать код по мере разработки приложения. Запускать низкоуровневый гибкий процесс сборки, который освободит время и поможет сосредоточиться на работе с React. Прочтение книги дает полный объем знаний для разработки полноценных веб-приложений, позволяющих сохранять данные на стороне клиента.

Вопросы относительно библиотеки ReactJs подробно изложены в книге «React и Redux. Функциональная веб-разработка» авторов А. Бенкс, Е. Порселло. Эта позволяет изучить библиотеку React и освоить передовые технологии, внедряемые в язык JavaScript. Экосистема этого языка стремительно пополняется новыми инструментами, синтаксисом и наработанными методиками, обещающими

разрешение множества проблем, стоящих перед разработчиками. Книга систематизирует новых технологий, позволяет сразу же приступить к работе с React. В книге рассмотрена библиотека Redux, маршрутизатор React Router, а также инструменты разработчик. Чтение книги не предполагает никаких предварительных знаний React. Все основы библиотеки представлены с самого начала. В ходе изучения материала можно обращаться к хранилищу GitHub, в котором находится весь код, позволяющий освоить практические примеры. Кроме того, книга освещает новые возможности ECMAScript 6. Прочтение вышеупомянутых книг по библиотеке ReactJs дают полную и структурированную информацию для разработки приложений, обрабатывающих большие объемы данных.

Книга Джона Дакетта «HTML и CSS. Разработка и дизайн веб-сайтов» содержит много примеров и советов по написанию качественного, адаптивного и кроссбраузерного кода на языках разметки гипертекста HTML и стилей CSS. Чтение книги не предполагает никаких предварительных знаний программирования и верстки. В книге описана работа с основными тегами, такими как: текст, ссылки, таблицы, изображения, формы, списки, блоки и другие. В ходе изучения материала можно знакомиться с приемами веб-дизайна, узнать о разных записях цвета и наиболее удачных шрифтов. Приводится процесс разработки веб-сайта с нуля: от дизайн-макета, до верстки и тестирования на различных устройствах и браузерах. В книге описаны некоторые подводные камни, с которыми можно столкнуться особенно на первых этапах изучения верстки, и даны подробные советы по их преодолению. Так же подробно освещены вопросы работы с инструментом разработчика и консолью в браузере. Прочтение данной дает полное представление о разработке современных адаптивных веб-сайтов.

2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Система управления проектами – является комплексом технологических и организационных методов и инструментов, поддерживающих управление проектами и повышающих эффективности их реализации. Такая система обычно соединяет в себе ряд других приложений, таких как:

- 1) планирование задач;
- 2) составление расписания;
- 3) распределение совместной работы;
- 4) общения;
- 5) документирование и администрирование системы.

Система управления проектами повышает эффективность командной работы, качество проектного менеджмента руководителями проекта. Предоставляет возможность планировать, выставлять приоритеты задач, обсуждать задачи по средствам комментариев между командой, просматривать историю выполненных задач.

Задачи систем управления проектами:

- 1) обеспечение проект-менеджера необходимыми инструментами планирования и контроля процесса реализации проекта;
- 2) предоставление участникам проекта понятных инструментов для решения задач и доступа к соответствующей информации;
- 3) предоставление руководителям подразделений инструментария по контролю загрузки исполнителей проектных и непроектных задач и информации для принятия решений о назначении сотрудников на новые проекты и перераспределении нагрузки;
- 4) снабжение директора удобными инструментами, позволяющими автоматизировать рутинные операции и устанавливать контроль состояния всего портфеля проектов и качества работы руководителей каждого проекта;
- 5) обеспечение руководителя проекта целостной моделью мониторинга портфеля проектов и анализа принимаемых решений и сопутствующих отклонений;
- 6) обеспечение акционеров компании инструментом для мониторинга соответствия проектного портфеля стратегическим целям компании.

Для привлечения и удержания пользователей иногда недостаточно хорошо продуманной логики приложения и широкой функциональности. Существенную роль так же играет юзабилити, т.е. такие важные аспекты как:

- 1) дизайн интерфейса, его корректная реализация и масштабируемость;
- 2) простота и ясность интерфейса – пользователь не должен тратить время на поиск той или иной кнопки, либо другой нужной для него информации;
- 3) выявление ошибок – пользователь должен быть уведомлен в случае совершения им ошибок при вводе текста или других неверных действиях.

Уведомление должно быть содержательным и ненавязчивым. Это может быть, например, всплывающая текстовая подсказка, либо окрашивание некоторого элемента в красный или любой другой цвет.

На мировом рынке существует несколько программных решений по управлению проектами. Далее будет представлен краткий обзор по наиболее популярным системам в данной предметной области.

Jira – это коммерческая система отслеживания ошибок, предназначенная для организации взаимодействия с пользователями и управления проектами. Система разработана компанией Atlassian. Веб-интерфейс предоставляет возможности по созданию задач, отслеживанию процесса работы над проектом и многое другое.

С точки зрения пользовательского интерфейса данная система имеет следующие преимущества:

1) широкий функционал – большое количество элементов управления и динамичности на страницах, благодаря которым покрывается множество поставленных задач для этого типа системы;

2) наличие адаптивной версии и мобильного приложения.

Но, как и большинство систем подобного рода имеет недостатки:

1) больше количество элементов управления на странице является главным недостатком. Это делает порог освоения приложением для новых пользователей более сложным. Из-за большого количества одновременно присутствующих на странице элементов управления и навигации, уменьшается концентрация пользователя на необходимом для него действии;

2) плохо выражены некоторые ключевые моменты, например, приоритет, тип и описание задачи. Описание находится внизу страницы, хотя это один из основных информативных блоков. За всем разнообразием элементов на странице список задач не отражает, некоторых ключевых моментов и для этого приходится переходить в детализацию конкретной задачи;

3) устаревший интерфейс. Пользовательский интерфейс довольно скучный и плохо разбит на логические блоки;

4) малое количество иконок. В современном дизайне существует огромное количество информативных иконок, которые делают интерфейс более привлекательным и за счет цвета и графической информации позволяют сократить текстовое содержание.

Таким образом можно сказать, что пользовательский интерфейс приложения Jira очень обширный и сложен в освоении. Большинство элементов навигации и вкладок можно было скрыть, когда пользователь переходит от страницы к странице, и вероятность использования некоторой навигации крайне мала.

На рисунке 2.1 приведен пользовательский интерфейс, который отражает список задач и детальное описание выбранной задачи.

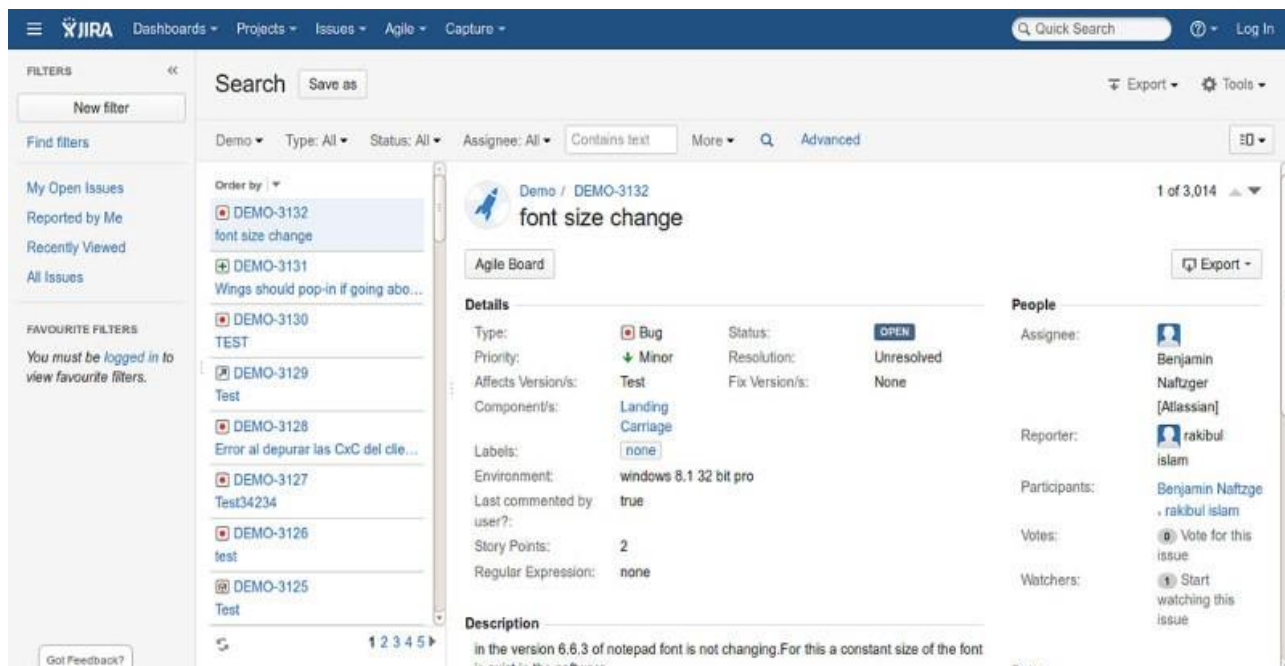


Рисунок 2.1 – Пользовательский интерфейс системы управления проектами Jira

Еще одним прим примером является Trello – программа для управления проектами в режиме онлайн, которая пользуется особенным спросом среди небольших компаний и стартапов. Система построена по японскому принципу управления «канбан», который предполагает последовательный контроль за этапами производства. В Trello систематизация задач осуществляется с помощью канбан-досок, которые можно увидеть на рисунке 2.2.

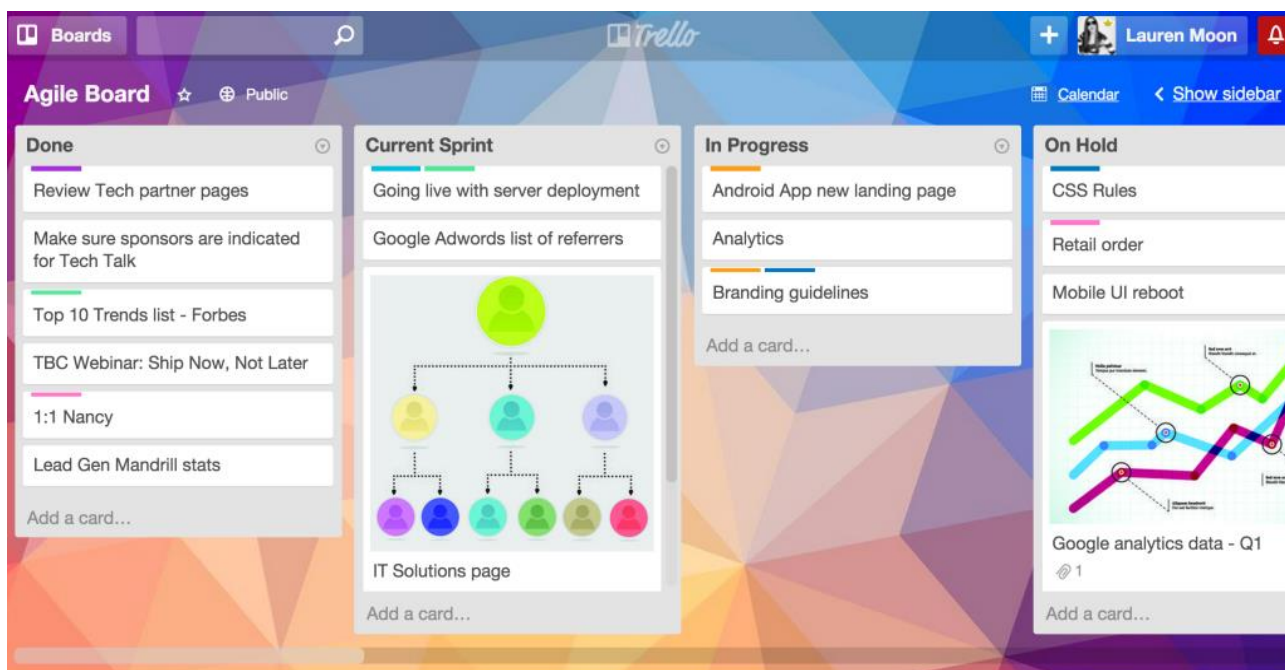


Рисунок 2.2 – Пользовательский интерфейс менеджера задач Trello

Канбан-досками могут быть различные проекты или процессы в компании. Доска состоит из списков – вертикальных колонок, расположенных слева направо, которые показывают этапы производства, от идей до реализации. Как правило задачи делятся на:

- 1) запланированные;
- 2) текущие;
- 3) выполненные.

Система имеет довольно инфантильный пользовательский интерфейс. Некоторым компаниям, в силу своей сферы деятельности, может быть не приемлем данный дизайн. И наоборот, для небольших стартапов или фиксирования обычных бытовых задач такой интерфейс может быть привлекателен. Как у любого приложения у Trello есть преимущества и недостатки. В силу своей простоты Trello имеет ряд преимуществ:

- 1) интуитивно понятный и привлекательный интерфейс, за счет небольшого количества навигации и вкладок на странице и множества однотипных логично сформированных блоков;
- 2) удобен для небольших команд и ежедневного планирования бытовых задач;
- 3) имеет адаптивный пользовательский интерфейс и мобильное приложение под различные платформы.

Пожалуй, главным недостатком является то, что данная система не подходит для фиксирования глобальных задач и работы над крупными проектами, из-за упрощенного функционала. Возможность добавления картинок, фоновых изображений и смайликов может служить как преимуществом, так и недостатком – отвлекать от основной функции приложения.

Подводя итоги можно сказать, что пользовательский интерфейс приложения Trello современен и прост в освоении. Подходит для более широкого круга пользователей, от домохозяек до небольших стартапов. Но тем не менее теряет клиентов в лице крупных фирм.

3 ПОСТАНОВКА ЗАДАЧИ

3.1 Общие определения

Пользователь – информация, связанная с реальным человеком, имеющим доступ к использованию данной системы. Набор функционала, которым может воспользоваться пользователь, зависит от назначенных ему прав доступа. Для различных проектов пользователь может иметь различные права доступа.

Администратор – специальный пользователь, который отвечает за управление пользовательскими аккаунтами в системы, выполняет конфигурацию системы на глобальном уровне (в отношении всех проектов) и имеющий непосредственный доступ к базе данных.

Права доступа – разрешение на определенные действия в рамках проекта для конкретного пользователя. В отношении проектов существуют следующие виды прав доступа:

- 1) просмотр задач – возможность просматривать основной информации по проектным задачам;
- 2) управление задачами – возможность создания и редактирования задач;
- 3) управление проектом – возможность изменять имя, описание и другие параметры проекта.

Проект – может означать идею, описание цели или чего-либо другого, по отношению к чему могут выполняться какие-либо задачи.

Задача – означает конкретную единицу работы, которую нужно выполнить конкретным исполнителем по отношению к определенному проекту.

Подзадача – дочерняя задача, которая необходима для детализации или разбиения некоторой задачи на несколько подзадач (подзадачи могут быть созданы для любого типа задачи кроме подзадачи).

Нововведение – новая задача, новая функция или новое свойство системы – задача, связанная с реализацией чего-то нового.

Дефект – задача, связанная с исправлением определенных проблем.

Иная задача – абстрактный тип задачи, обычно используется для тех случаев, когда другие типы задач не подходят.

Исполнитель – конкретный пользователь, ответственный за выполнение задачи.

Приоритет – приоритет задачи. Означает важность этой задачи по отношению к другим. Виды приоритетов:

- 1) тривиальный – означает минимальный приоритет задачи (например, изменение цвета кнопки, сортировка документов и т.п.);
- 2) низкий – более высокий приоритет, но все еще не критичный к исполнению (результат работы над проектом может отдаваться даже в случаях, когда не

выполнены некоторые задачи этого приоритета);

3) высокий – означает высокий приоритет и важность выполнения этой задачи для конечного клиента/заказчика;

4) критический – означает высокий приоритет и первоочередность выполнения такой задачи;

5) блокирующий – также высокий приоритет, отличающийся от других подобных тем, что задача с этим приоритетом будет препятствовать работе других участников команды.

Предполагаемое время – запланированное время необходимое для выполнения задачи.

Затраченное время – фактически потраченное время на выполнение задачи.

Статус – означает стадию, на котором находится задача. Статус задачи может быть следующих видов:

1) открыта – означает, задача создана и ей присвоен исполнитель;

2) выполняется – исполнитель принял задачу и находится в процессе ее выполнения;

3) тестирование – задача была выполнена исполнителем и сейчас находится в процессе тестирования;

4) завершена – задача была протестирована, ошибок не выявлено;

5) верификация – выполненная и протестированная задача находится на стадии проверки более квалифицированными работниками, для определения правильности ее решения, с точки зрения логичности применения тех или иных методов для ее исполнения.

3.2 Требования к пользовательскому интерфейсу

Необходимо разработать пользовательский интерфейс веб-системы для реализации удобного и эффективного управления проектами. Основные возможности, которые будут доступны для администратора:

1) создание проекта и снабжение его необходимой информацией и документацией;

2) добавление новых участников в проект, а также возможность ограничения доступа определенных участников в отношении определенного проекта.

Основные возможности, предоставляемые пользователю:

1) снабжение проекта необходимой информацией и документацией;

2) управление задачами: создание задачи, назначение исполнителя, установка сроков, приоритета перед другими задачами, типа задачи, редактирование;

3) возможность обсуждения проблем, предложений, вариантов улучшения в рамках какой-либо задачи любым участником проекта в любом количестве;

4) ведение журнала работ по каждой из задач с описанием выполненных работ и выставления затраченного времени для определения темпа разработки проекта и выполнения задачи, что может помочь выявить проблемы, препятствующие быстрому выполнению схожих задач;

5) сортировка задач в зависимости от сроков их завершения;

6) управление несколькими проектами одновременно;

7) управление списком задач для сотрудников и предоставление информации по распределения ресурсов;

8) обзор информации о сроках выполнения задач;

9) возможность раннего уведомления о возможных рисках, связанных с проектом;

10) обзор информации о рабочей нагрузке;

11) предоставление детальной информации о ходе проекта;

12) обсуждение и согласование рабочих вопросов проекта;

13) фиксация проблем проекта и запросов на изменения, их обработка;

14) ведение рисков проекта и управление ими;

15) валидация и обозначение ошибок при вводе данных и иных ошибочных действиях.

Система должна быть реализована в виде веб-приложения в виду того, что это наиболее предпочтительный вариант для удобного и эффективного использования в рамках большого количества людей (работники, менеджер, заказчики и т.п.) а так же любой операционной системы и веб-браузера. Одним словом, доступ в веб-приложение может быть осуществлен в любом месте. Веб-система упрощает обновление, так как это происходит только на стороне разработчиков. Такая система легка в разворачивании на рабочей машине или мобильном устройстве пользователя, поскольку необходим всего лишь веб-браузер и подключение к интернету.

Так же пользовательский интерфейс должен предусматривать валидацию вводимых данных и других ошибочных манипуляций для предотвращения неверного использования функционала. Приложение должно иметь простой и удобный пользовательский интерфейс для улучшения качества и эффективности своей работы.

Страницы приложения должны быть выдержаны в едином стиле и соответствовать современным стандартам веб-дизайна. Необходимо выбрать определенную цветовую гамму, которая будет создавать спокойную рабочую атмосферу.

4 ЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ

4.1 Логическая модель

4.1.1 Основные роли в системе

В системе должны быть определены некоторые логические группы пользователей, при попадании в которые пользователь будет иметь возможность пользоваться определенным набором функционалом. Отношение пользователя к той или иной группе будет означать то, что пользователь имеет определенную роль в системе.

В системе должны быть определены две основные роли пользователей:

1) пользователь – наделенные данной ролью пользователи системы могут пользоваться основным функционалом системы достаточным для осуществления деятельности по управлению проектами;

2) администратор – данная роль присваивается пользователем, ответственным за регистрацию новых пользователей в системе, назначение им ролей, управление доступом пользователей к определенным проектам, а также создание проектов в системе.

Между данными двумя ролями существует большая разница как в контексте использования функционала системы, так и в целом отношения к главной деятельности – управлению проектами.

4.1.2 Варианты использования системы

Пользователи, имеющие роль администратор по своей сути не имеют никакого отношения к управлению проектами. Их главная обязанность – введение новых пользователей и проектов в систему, а также разграничение прав доступа.

Таким образом, для администраторов должен быть предусмотрен и доступен следующий функционал в системе:

- 1) создание первоначальной учетной записи для пользователя;
- 2) назначение роли для пользователя;
- 3) создание проекта;
- 4) назначение для пользователя проектов, с которыми он может работать;
- 5) лишения доступа к определенным проектам для определенного пользователя.

В свою очередь пользователи, относящиеся к роли «User», не могут никаким образом воздействовать на учетные данные других пользователей либо на данные по проектам, но могут осуществлять основную деятельность в отношении проектов. Т.е.

пользователь с ролью «User» этот тот самый человек, который имеет прямое отношение к разработке и развитию программного проекта. Это может быть, как программист, ведущий разработку на проекте, так и проектный руководитель, организующий работу среди подчиненных, или же тестировщик, обеспечивающий качество выполненных работ, и др.

Таким образом, для осуществления своей деятельности, пользователям с данной ролью необходим следующий функционал в системе:

- 1) редактирование профиля с заданием всех ключевых персональных данных: должность, контакты, фото и т.д.;
- 2) просмотр информации о членах команды, включая их контакты;
- 3) получение списка текущих задач, всех задач в проекте, закрытые задачи в прошлом для определенного проекта, а также получение списка задач по множеству другим критериям для формирования статистики, изучения предыдущего опыта и для других целей;
- 4) создание задачи с указанием всех основных параметров: название, описание, сроки, тип, статус и другие данные;
- 5) возможность редактирования основных данных по задаче (за исключением некоторых данных, таких как номер задачи, создатель, дата создания и т.д.);
- 6) назначение исполнителя для определенной задачи;
- 7) возможность обсуждения задачи с остальными членами команды;
- 8) ведение и просмотр журнала работ по конкретной задаче;
- 9) просмотр информации по текущим проектам;
- 10) возможность загрузки документации, относящейся к проекту.

Диаграмма вариантов использования представлена на рисунке 4.1.

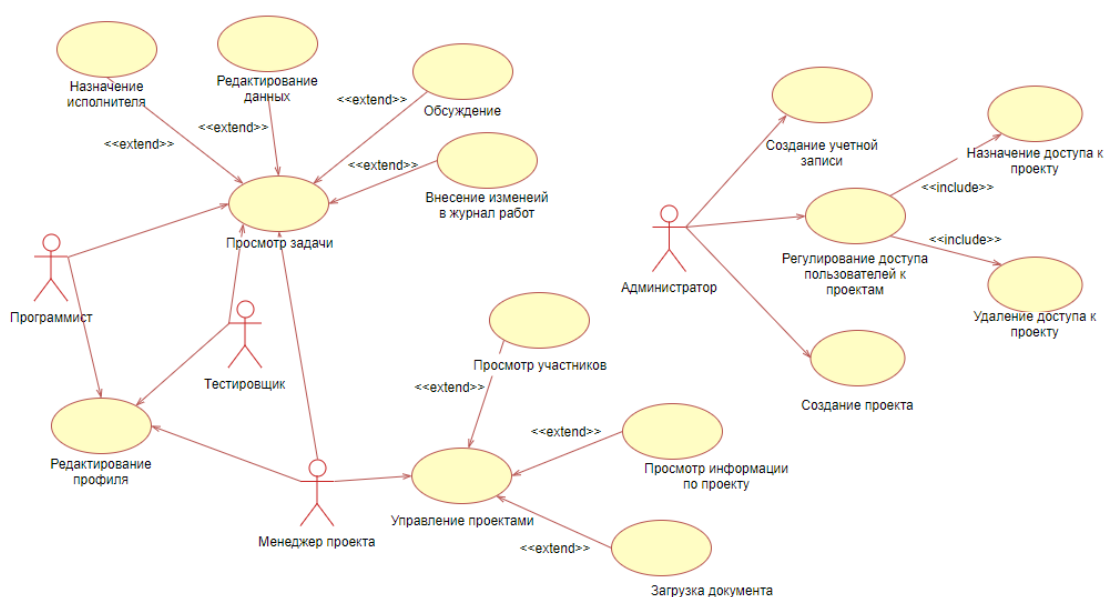


Рисунок 4.1 – Диаграмма вариантов использования веб-системы

4.2 Модель взаимодействия клиентского приложения и сервера

Для того чтобы веб-сервер мог взаимодействовать с клиентскими приложениями различных типов он должен предоставлять свой собственный API интерфейс.

API (программный интерфейс приложения, интерфейс прикладного программирования) – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Т.е. API интерфейс скрывает за собой некоторый функционал по обработке данных, предоставляя в открытом виде просто способ воспользоваться этим функционалом другим программам [1].

В контексте веба, взаимодействие между клиентским приложением и сервером должно осуществляться по протоколу HTTP, а значит должен соблюдаться некий набор правил форматов запроса, отправляемых клиентскими приложениями, и форматов ответов, отправляемых сервером в ответ на запрос [2].

В связи с этим, взаимодействие между клиентом и сервером должно соответствовать архитектуре, называемой REST. REST (сокращение от англ. Representational State Transfer – «передача состояния представления») – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

В сети Интернет обращение к API может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»; такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса. Пример REST-запроса представлен на рисунке 4.2.

```
GET http://ctld1.windowsupdate.com/msdownload/update/v3/static/trustedr/en/disallowedcertst1.cab?6ac2b6592d164b9e HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Microsoft-CryptoAPI/10.0
Host: ctld1.windowsupdate.com
```

Рисунок 4.2 – Формат REST-запроса

Основные секции запроса:

- 1) заголовки – здесь находится информация о типе запроса (Get, Post и др.), идентификационная информация и другие мета-данные;
- 2) адрес, по которому отправляется запрос (также может содержать параметры, если тип запроса GET);

3) тело запроса – необходимые данные и параметры (данная секция существует только для типа запроса POST).

Реализация API представляет собой совокупность API-методов доступных клиентским приложениям. Каждый API-метод имеет свой тип, адрес и конкретный способ обработки информации.

Таким образом, веб-сервер должен предоставлять информацию о всех реализованных API-методах с указанием следующей информации:

1) адрес API-метода – адрес конкретного метода в интерфейсе API, выполняющий определенную обработку (например `api/user/getallusers`), данный адрес должен объединяться с именем веб-сервера в Интернете и http протоколом, для того чтобы можно было обратиться к данному методу;

2) тип принимаемого запроса (Get, Post, Put или Delete);

3) перечень всех входных параметров и их форматы;

4) формат данных отправляемых в ответе.

На веб-сервер будет использоваться всего два типа http-запросов:

1) Get – для получения данных;

2) Post – для изменения данных либо запроса данных с передачей большого количества параметров.

Общая схема взаимодействия клиентского приложения и сервера приведена на рисунке 4.3.

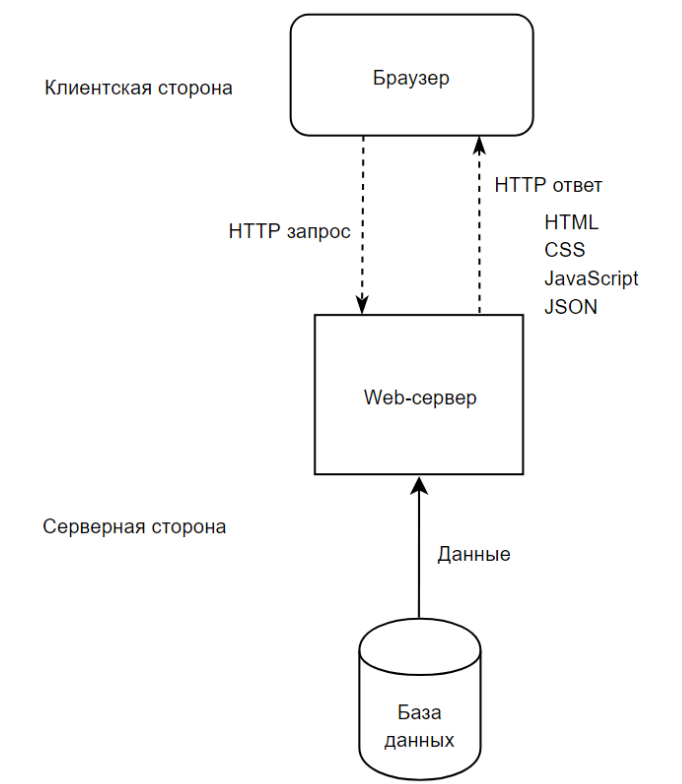


Рисунок 4.3 – Формат REST-запроса

5 ФИЗИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ

5.1 Проектирование пользовательского интерфейса

При проектировании пользовательского интерфейса необходимо учитывать специфику приложения, а также аудиторию на которую рассчитывается данное приложение.

Проектируемая система будет иметь две роли: «Пользователь» и «Администратор». При запуске веб-системы неавторизованный пользователь будет перенаправляться на страницу авторизации. После успешной авторизации должно осуществляться перенаправление на главную страницу. Здесь в зависимости от роли, пользователю системы будет доступен свой функционал.

Для двух ролей будет доступно меню в верхней части веб-системы, с помощью которого можно производить навигацию по основным страницам. Меню так же будет различаться для пользователя той или иной роли.

5.1.1 Проектирование пользовательского интерфейса для роли «Пользователь»

Для роли «Пользователь» главной страницей будет являться страница со списком задач. Здесь же должна быть предоставлена возможность создания новой задачи. Перейдя на выбранную задачу можно будет попасть на страницу с ее детальным описанием. На этой странице должны быть представлены следующие возможности:

- 1) редактирование задачи;
- 2) выбор ответственного за выполнение задачи;
- 3) подтверждение удаления задачи;
- 4) выбор статуса задачи;
- 5) комментирование;
- 6) журнал работ.

Страница со списком проектов должна быть доступна из верхнего меню веб-системы. При выборе определенного проекта пользователь будет перенаправляться на страницу с детальным описанием выбранного проекта. Страница описания должна предоставлять возможность обзора и добавления документации по проекту, но не редактирования.

На страницу профиля пользователя можно будет попасть кликнув по картинке с фото пользователя, которая должна находиться в верхней правой части веб-системы. Кроме обзора информации о профиле, эта страница будет предоставлять возможность

редактирования, а также выхода из системы.

Страница со списком участников команды будет доступна в верхнем меню веб-системы. Выбрав конкретного члена команды будет осуществлен переход на страницу его профиля.

На страницу с ошибкой можно попасть находясь на любой странице веб-системы, в случае возникновения той или иной ошибки. Веб система должна обрабатывать как минимум следующие типы ошибок и в виде текстовой информации уведомлять о них пользователя:

- 1) страница не найдена;
- 2) доступ запрещен;
- 3) ошибка сервера.

Для наглядности разработана карта пользовательского интерфейса веб-системы для роли «Пользователь», которая представлена на рисунке 5.1.

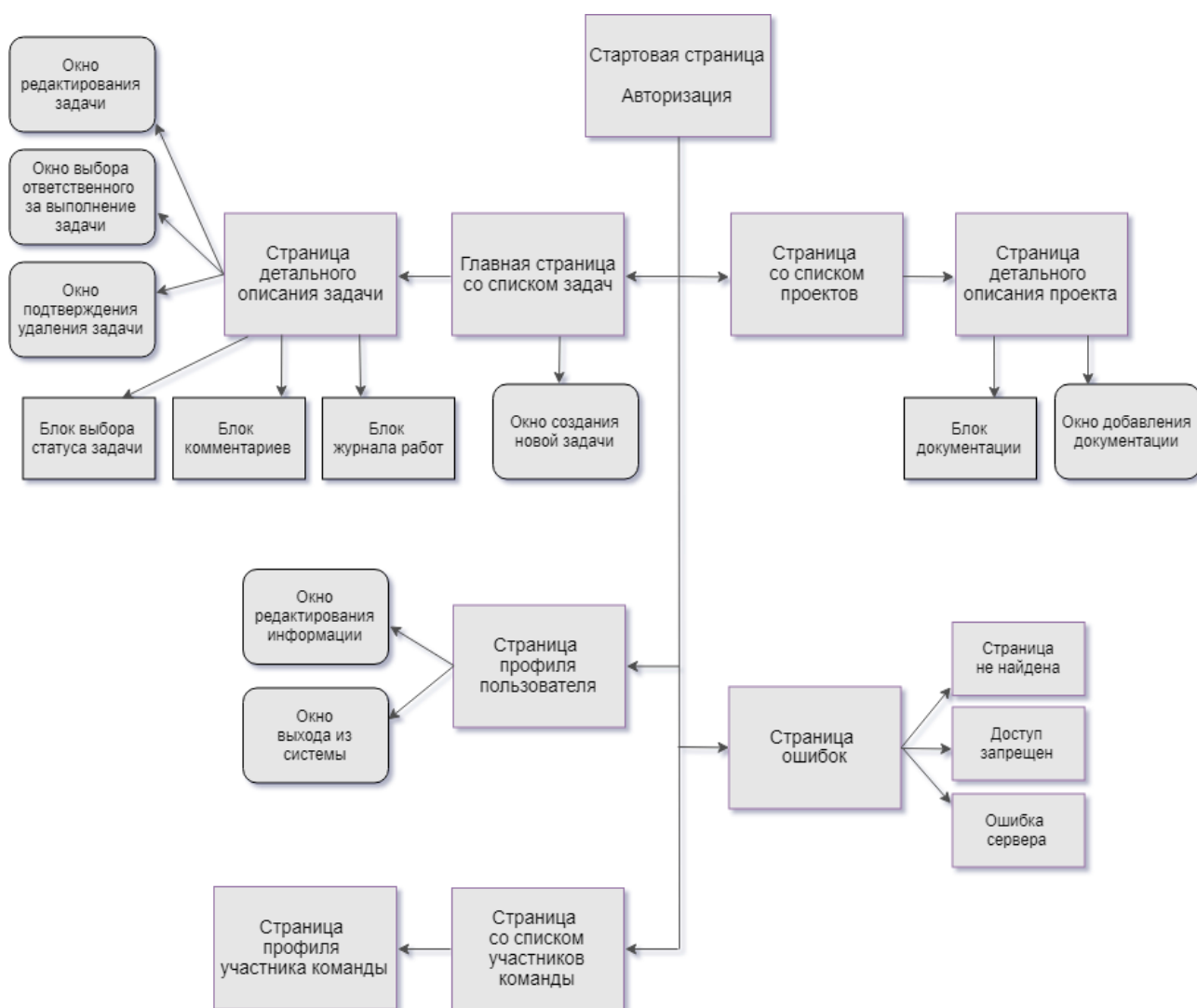


Рисунок 5.1 – Карта пользовательского интерфейса веб-системы для роли «Пользователь»

5.1.2 Проектирование пользовательского интерфейса для роли «Администратор»

Пользователь веб-системы с ролью «Администратор» будет обладать меньшими возможностями. При успешной авторизации будет происходить перенаправление на главную страницу со списком проектов. Данная страница будет предоставлять возможность создания нового проекта и редактирование существующих.

Из верхнего меню «Администратор» сможет попасть на страницу со списком пользователей, это будут все пользователи, зарегистрированные в системе. На этой странице должна быть возможность создания и редактирования нового пользователя, а также назначение проекта пользователю.

Возможность выхода из системы будет предоставлена в верхнем меню веб-системы.

На страницу с ошибкой можно попасть, находясь на любой странице веб-системы, также, как и для роли «Пользователь».

Карта пользовательского интерфейса веб-системы для роли «Администратор», представлена на рисунке 5.2.

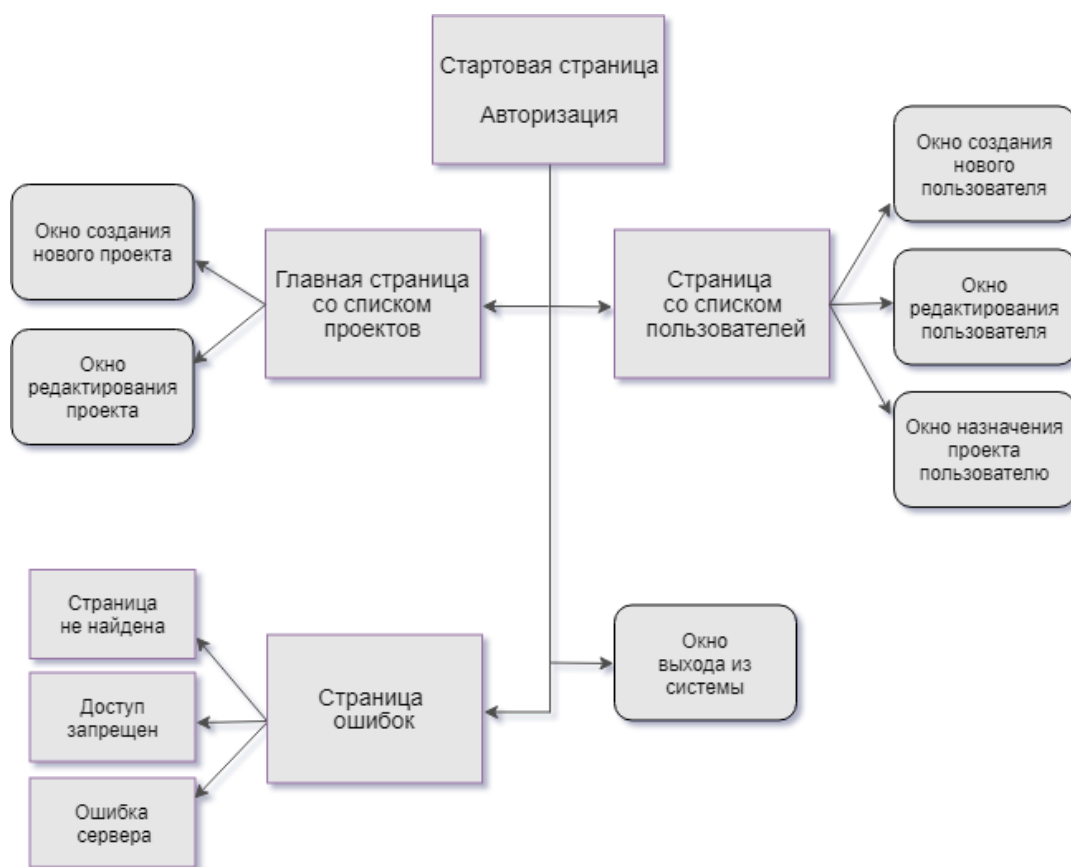


Рисунок 5.2 – Карта пользовательского интерфейса веб-системы для роли «Администратор»

5.2 Моделирование веб-страниц

Перед разработкой пользовательского интерфейса необходимо смоделировать макет веб-страницы. Учитывая критерии, приведенные в постановке задачи, каждая веб-страница должна содержать следующую структуру:

1) верхнюю часть сайта, в которой будет находиться логотип, меню и аватар (фото) авторизованного пользователя. Меню должно содержать навигацию по веб-приложению;

2) главная секция должна содержать основные данные, предоставленные приложением, с которыми будет взаимодействовать пользователь. Содержимое главной секции будет изменяться в зависимости от той страницы, на которой находится пользователь;

3) нижняя часть сайта должна носить информативный характер и содержать данные о защите авторских прав.

Веб-приложение должно содержать несколько страниц. Верхняя и нижняя части сайта будут одинаковыми на всех страницах.

Главная страница будет содержать список задач и краткую информацию о них для текущего авторизованного пользователя. Так же в правой части страницы должна размещаться кнопка, позволяющая создать новую задачу.

Макет главной страницы веб-приложения со списком задач представлен на рисунке 5.3.



Рисунок 5.3 – Макет главной страницы

При выборе определенной задачи должно происходить перенаправление на другую страницу, на которой будет детальное описание этой задачи. Макет детализации задачи приведен на рисунке 5.4.



Рисунок 5.4 – Макет страницы детализации задачи

Страница детализации задачи предоставляет возможность управления состоянием задачи по средствам кнопок. На странице должна быть предусмотрена возможность комментирования, в виде блока для ввода текста и кнопки отправить. Эта функция позволит участникам проекта задавать вопросы автору задачи относительно текущей задачи или проекта в целом. Так же должен быть предусмотрен вывод информация о ходе работы над задачей, который будет отражаться в журнале работ.

Страница проектов должна содержать таблицу с проектами и окном поиска для возможности быстрого нахождения нужного проекта. При клике по строке с проектом, пользователь будет переходить на страницу с детальным описанием проекта. На странице проекта будут представлены кнопки скачать или загрузить текстовые-документы, относящиеся к этому проекту. Страница с таблицей проектов и страница детализации проекта должны быть схожи по визуальному оформлению и логике манипуляций со страницами списка задач и детализации задачи.

Переход на страницу личного профиля пользователя должен осуществляться при клике по фото пользователя в верхней правой части сайта. На этой странице должна размещаться достаточная информация о работнике: имя, фамилия, должность,

контактные данные, навыки и другое. Кроме того, необходимо предусмотреть редактирование профиля, в виде формы с соответствующими полями для ввода данных. Так же из личного кабинета будет предоставлена возможность выхода из аккаунта. Макет страницы личного профиля пользователя представлен на рисунке 5.5.



Рисунок 5.5 – Макет страницы личного профиля пользователя

На странице входа должна быть простая форма, представленная на рисунке 5.6, с полями для ввода email и пароля пользователя, а также кнопкой войти.

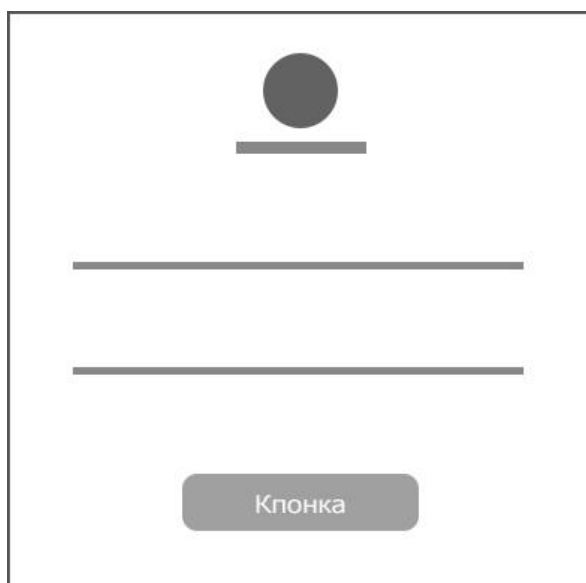


Рисунок 5.6 – Макет страницы входа

В случае если текущий пользователь имеет роль «Администратор», в меню скрываются некоторые пункты и добавляются новые, соответствующие роли. Администратору будет доступна страница с таблицей проектов, которая будет схожа с одноименной страницей для роли «Пользователь».

Так же будет доступна страница создания нового пользователя. Данная страница будет содержать форму с полями для ввода информации о пользователе: имя, фамилия, должность, проект, email. Типичная форма для ввода информации представлена на рисунке 5.7.

The image shows a wireframe of a web form for creating a new user. At the top, there is a header bar containing a logo placeholder labeled 'ЛОГОТИП', three menu items labeled 'Пункт меню 1', 'Пункт меню 2', and 'Пункт меню 3', and a circular profile picture placeholder labeled 'AVA'. Below the header is a horizontal line. The main content area contains a form with a title bar, followed by five input fields: three text fields and two dropdown menus. Below the input fields is a button labeled 'Кнопка'. At the bottom of the page is a footer bar labeled 'Нижняя панель сайта'.

Рисунок 5.7 – Макет формы для внесения информации

Таким образом, разрабатываемая веб-система имеет простой и удобный пользовательский интерфейс, который выдержан в минималистичном стиле. Интерфейс достаточно интерактивен за счет всплывающих окон, выпадающих списков, кнопок и полей для ввода информации, что делает его более привлекательным.

6 РЕАЛИЗАЦИЯ ПРОГРАММНОЙ СИСТЕМЫ

6.1 Интегрированная среда разработки Visual Studio Code

Visual Studio Code – редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом. Visual Studio Code поддерживает многие языки программирования, например, C, C++, C#, CSS, HTML, JavaScript, Json, React, PHP и другие. Скриншот интерфейса среды разработки приведен на рисунке 6.1.

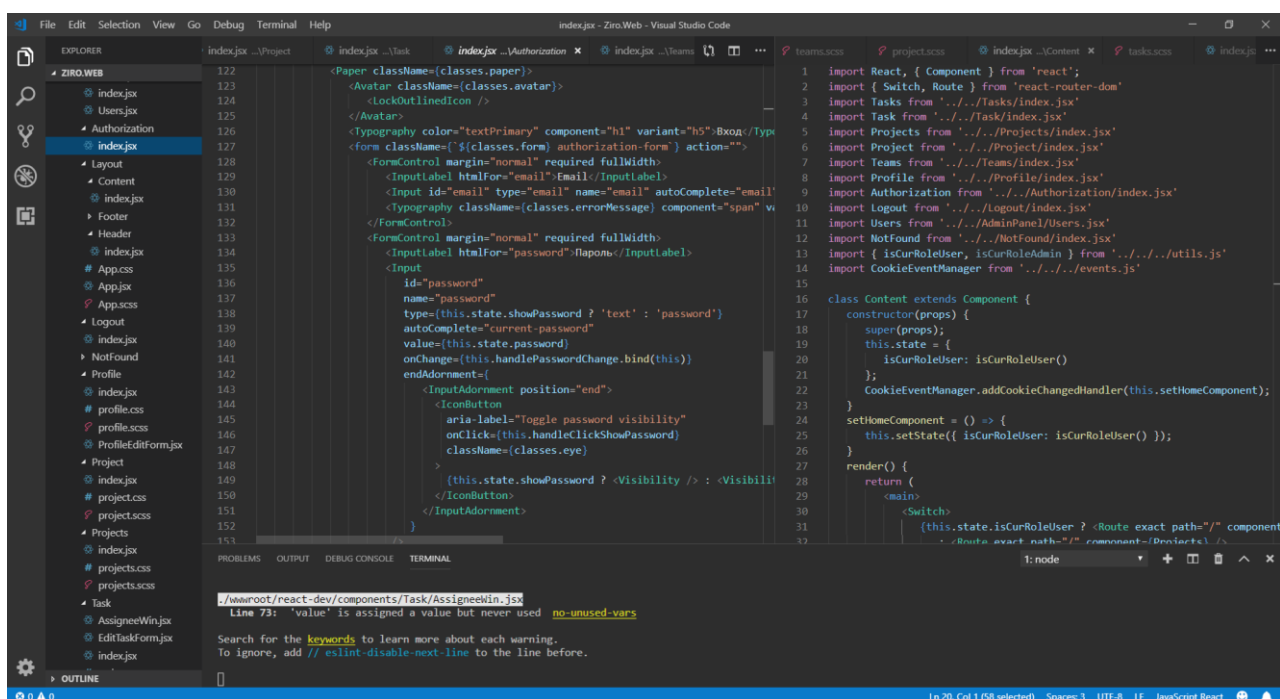


Рисунок 6.1 – Интерфейс среды разработки Visual Studio Code

Основными преимуществами для выбора данной среды разработки послужили:

- 1) бесплатная – полноценная версия программы доступна на официальном сайте;
- 2) легковесность, что выделяет Visual Studio Code над средой разработки того же семейства – Visual Studio;
- 3) встроенная консоль, которая позволяет избавиться от переключения между окнами и сконцентрироваться на пространстве редактора кода;
- 4) возможность кастомизации – позволяет настроить редактор кода под свои

нужды, скачивать и устанавливать различные темы и наборы иконок;

5) поддержка различных плагинов, позволяющих ускорить разработку приложений.

6.2 Визуальная часть

Пользовательский интерфейс системы управления проектами «ZIRO» выдержан в минималистичном стиле с элементами современного material дизайна и представляет собой одностраничное веб-приложение (SPA) с динамически изменяющимся контентом. Страница состоит из трех основных секций:

1) шапка – верхняя часть сайта, содержащая в себе логотип меню и аватар (фото) пользователя. Эта секция изменяется лишь в зависимости от роли авторизованного пользователя;

2) главная секция включает в себя основной контент сайта, с которым взаимодействует пользователь. Контент главной секции изменяется при переходе по ссылкам или при открытии всплывающих окон;

3) подвал – нижняя часть сайта. Содержит копирайт – защита авторских прав в интернете. Эта секция является статичной, и не изменяется ни при каких условиях.

Навигация в пользовательском интерфейсе была реализована согласно картам сайта для ролей «Пользователь» и «Администратор» представленным в разделе проектирования.

Важной составляющей дизайна веб-системы является цветовая гамма. Цвета влияют не только на внешний вид и стиль, но и на успех с точки зрения продолжительности посещения и эмоционального восприятия. Холодные цвета отлично передают спокойствие, профессионализм и уверенность. Лучшим выбором считаются оттенки синего цвета. Это цвет безопасности и надежности, в сочетании с белым и светло-серыми оттенками создает спокойную и дружелюбную рабочую атмосферу. На рисунке 6.2 приведена цветовая гамма разработанного пользовательского интерфейса веб-системы.

В верхней части рисунка вертикальными прямоугольниками показаны основные цвета, использованные при создании пользовательского интерфейса веб-системы. Белый и светло-серый цвета преобладают в фоне и крупных блоках. Темно-серый и черный цвет преимущественно используется для обычной текстовой информации. Голубой цвет используются для заголовков и важной информации, на которую следует обратить внимание. Синий цвет преобладает в логотипе, а также в большинстве обычных кнопок, этот цвет позволяет акцентировать внимание на важных участках веб-страницы.

В нижней части рисунка 6.2 расположены горизонтальные прямоугольники. В них отражены цвета, использованные для кнопок, и различных меток, отражающих

статусы и приоритеты задач и другое. Зеленый цвет используется преимущественно для кнопок создания или добавления чего-либо.



Рисунок 6.2 – Цветовая гамма пользовательского интерфейса

6.3 Программная часть

Пользовательский интерфейс веб-приложения написан с использованием современных технологий веб-разработки:

- 1) JavaScript (ReactJs);
- 2) Npm;
- 3) Webpack и Babel;
- 4) Router Dom;
- 5) Material UI;
- 6) HTML;
- 7) CSS 3 (SASS).

6.3.1 JavaScript (ReactJs)

JavaScript – это язык программирования, который изначально задумывался, как язык для взаимодействия с HTML-документом, который добавлял бы интерактивность на веб-сайт. Сейчас JavaScript единственный язык программирования для браузеров. Он работает под Windows, macOS, Linux, мобильных платформах, одним словом везде [3].

JavaScript стремительно развивается и сейчас это уже не только язык интерактивности веб-страниц, с помощью него можно взаимодействовать

пользователями, получать и отправлять запросы на сервер, а также писать полноценные приложения благодаря Node.js.

Для написания дипломного проекта была выбрана библиотека ReactJs. Это инструмент с открытым исходным кодом для создания пользовательских интерфейсов, разработанный Facebook в 2013 году и поддерживаемый им в настоящее время. Сейчас ReactJs пользуется большой популярностью благодаря своей лаконичности и легкости интеграции с существующими проектами. ReactJs представляет собой полноценный View модуль (Представление) из шаблона проектирования MVC (Модель-Представление-Контроллер), что подтверждает его независимость от серверной части проекта.

ReactJs использует компонентный подход. Компонент – это отдельная часть интерфейса, которую можно переиспользовать, наследовать, компоновать. Технически компонент строится с помощью функции либо класса. Для формирования веб-страницы происходит чтение и отрисовка компонентов в определенном порядке. Обычно ряд компонентов собирается в одном корневом компоненте и последний передается на отрисовку на странице [4].

Для решения проблем производительности ReactJs использует виртуальный DOM (Document Object Model) – организацию элементов html, которыми можно манипулировать, изменять, удалять или добавлять новые. Для взаимодействия с DOM применяется язык JavaScript. Манипуляции html-элементами с помощью JavaScript может привести к снижению производительности, особенно при изменении большого количества элементов. А операции над элементами могут занять некоторое время, что может замедлить работу приложения. Работа с объектами JavaScript, вместо реальных DOM-элементов, позволяет решить эту проблему, т.е. производить операции быстрее.

Виртуальный DOM представляет легковесную копию обычного DOM. Если приложению нужно узнать информацию о состоянии элементов, то происходит обращение к виртуальному DOM. Если необходимо изменить элементы веб-страницы, то изменения вначале вносятся в виртуальный DOM. Потом новое состояние виртуального DOM сравнивается с текущим состоянием. Если эти состояния различаются, то ReactJs находит минимальное количество манипуляций, которые необходимы до обновления реального DOM до нового состояния и производит их.

В итоге такая схема взаимодействия с элементами веб-страницы работает гораздо быстрее и эффективнее, в отличие от работы из JavaScript с DOM напрямую.

ReactJs использует JSX для описания визуального кода на JavaScript и разметки XML. JSX – это синтаксический сахар для функции создания элемента и компилируется в обычный синтаксис JavaScript. Использовать JSX рекомендуется для создания интерфейса [5].

ReactJs был выбран вместо JavaScript в силу следующих преимуществ:

1) компонентный подход – значительно улучшает организацию проекта, благодаря возможности переиспользовать и наследовать компоненты, а также разделять их на отдельные файлы и после собирать воедино;

2) использование виртуального DOM вместо обычного – ускоряющего отрисовку веб-страницы;

3) JSX синтаксис, благодаря которому разметка и логика размещаются вместе ускоряет и упрощает запись шаблонов. Так же JSX предлагает безопасность типов, большинство ошибок могут быть обнаружены во время компиляции.

При написании пользовательского интерфейса был использован компонентный подход. Для реализации функциональной части пользовательского интерфейса были разработаны следующие React-компоненты:

1) index – отвечает за отрисовку всего приложения (компонента App);

2) App – собирает воедино компоненты, представляющие три основные секции приложения (верхняя часть, главный контейнер и нижняя часть) и формирует общую структуру приложения;

3) Header – представляет верхнюю часть сайта, в которой расположены: логотип, меню и фото текущего пользователя;

4) Content – здесь подключаются все компоненты, отвечающие за основное содержимое главного контейнера. Этот компонент определяет логику по которой в зависимости от ссылки по которой перешел пользователь на отрисовку передается соответствующий компонент;

5) Footer – описывает нижний раздел приложения;

6) Authorization – содержит визуальное представление и логику формы входа;

7) Tasks – представляет компонент со списком задач (список компонентов Task);

8) AddTaskForm – содержит форму добавления новой задачи. Данный компонент подключается в компоненте Tasks.

9) Task – компонент содержит детальное описание задачи. Здесь сделано разделение на другие подкомпоненты – отдельные самостоятельные секции страницы:

– Comments – содержит блок комментариев и возможность добавления нового комментария;

– Log – содержит блок журнала работ и возможность добавления новой записи;

– AssigneeWin – компонент содержащий логику всплывающего окна, в котором есть выпадающий список с участниками проекта, на которых можно назначить текущую задачу;

– EditTaskForm – всплывающее окно формы редактирования текущей задачи;

– DeleteWin – компонент содержит окно подтверждения удаления.

10) Projects – компонент, который отображает список проектов для текущего пользователя системы, который состоит из компонента Project;

11) Project – компонент, который отображает список проектов для текущего пользователя системы;

12) Upload – используется в компоненте Project и содержит функционал позволяющий прикреплять текстовые-документы;

13) AddProjectForm – форма добавления проекта доступна только для роли «Администратор»;

14) EditProjectForm – форма редактирования проекта доступна только для роли «Администратор»;

15) DeactivateForm – форма деактивации проекта для роли «Администратор»;

16) Teams – компонент содержит логику, по которой страница разбивается на блоки, в которых указано название проекта и перечислены все пользователи, закрепленные на этом проекте;

17) AddUserForm – компонент доступен только для роли «Администратор». Содержит форму добавления нового пользователя;

18) DeleteUserWin – компонент доступен только для роли «Администратор». Содержит окно подтверждения удаления;

19) Error – компонент, который при возникновении ошибки на сервере, выводит соответствующее сообщение об ошибке.

Все React-компоненты, которые отображают данные о задаче, проекте и пользователях устроены по одной схеме. В момент инициализации компонента посылается запрос на сервер за необходимыми данными с помощью одного из реализованных методов:

1) FetchGetData – данный метод посылает get запрос на сервер, передавая параметры запроса в строке url-адреса;

2) FetchPostData – данный метод посылает post запрос на сервер, передавая параметры запроса в теле запроса.

Оба метода принимают в качестве параметра адрес api-метода сервера, по которому будет осуществляться запрос, объект с данными необходимыми для запроса, а также методы, срабатывающие при успешной и неуспешной обработке запроса на сервере.

Метод, срабатывающий при успешной обработке запроса назначает полученные данные состоянию компонента (state). Состояние представляет коллекцию значений, которые ассоциированы с компонентом, эти значения позволяют создавать динамические компоненты. Состояния определяются внутри компонента и доступны только из компонента. Метод, срабатывающий при успешной обработке запроса выводит сообщение об ошибке.

В разметку страницы подставляются актуальные данные из состояния. При совершении манипуляций пользователем данные могут изменяться. Манипуляции пользователя отслеживаются благодаря обработчикам событий, которые

навешиваются на кнопки, поля для ввода данных и другие элементы управления веб-страницей.

Специальный React-метод `setState`, который определяется в обработчике события, позволяет обновлять состояние компонента. Изменение состояния вызывает повторную отрисовку, в соответствии с чем только измененный участок страницы обновляется. Такое поведение доступно благодаря использованию React-ом виртуального DOM.

Архитектура React-компонентов для роли «Пользователь» можно увидеть на рисунке 6.3.

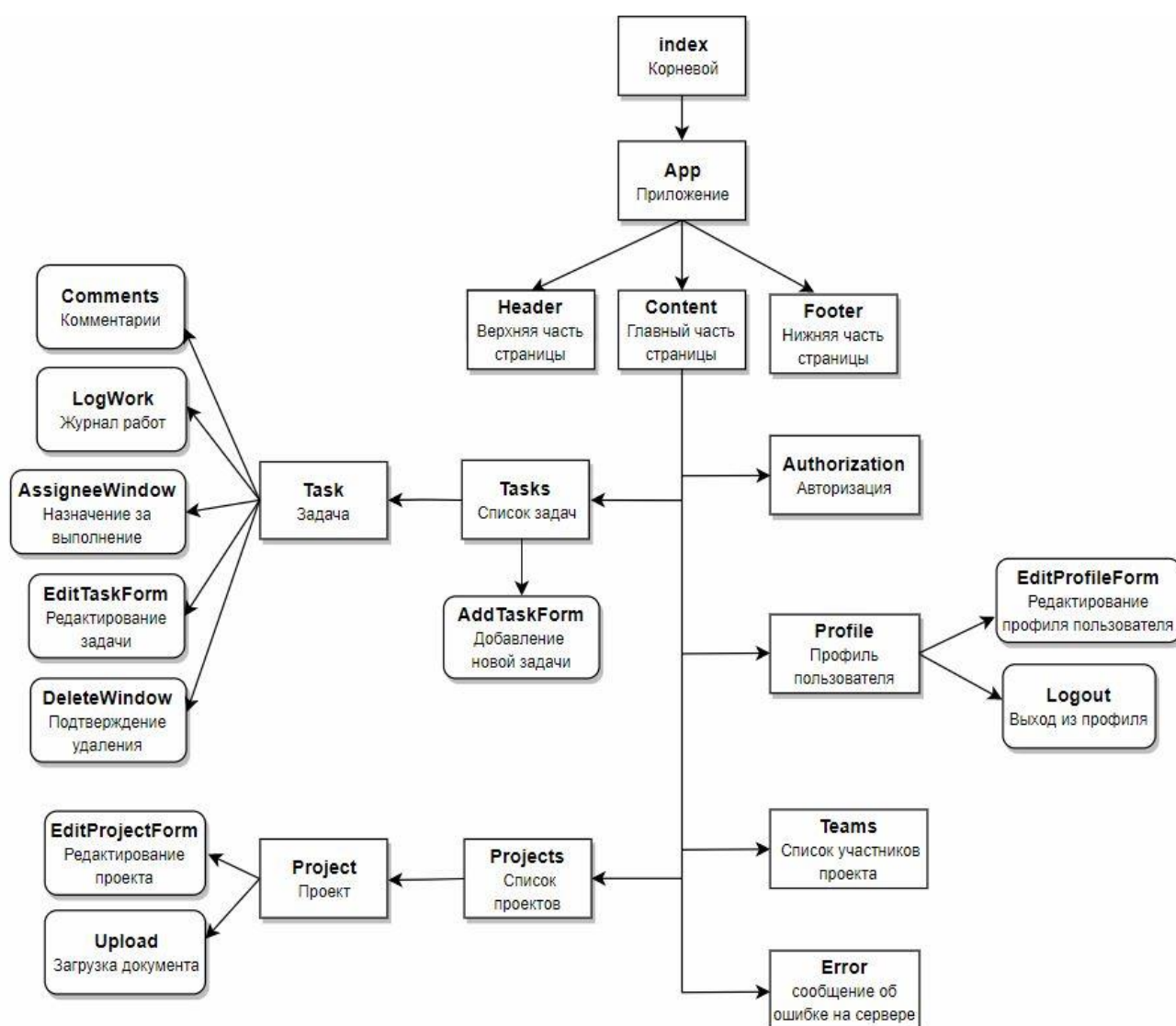


Рисунок 6.3 – Архитектура React-компонентов пользовательского интерфейса для роли «Пользователь»

Архитектура React-компонентов для роли «Администратор» немного отличается по своей структуре и имеет более упрощенный вариант, представлена на рисунке 6.4.

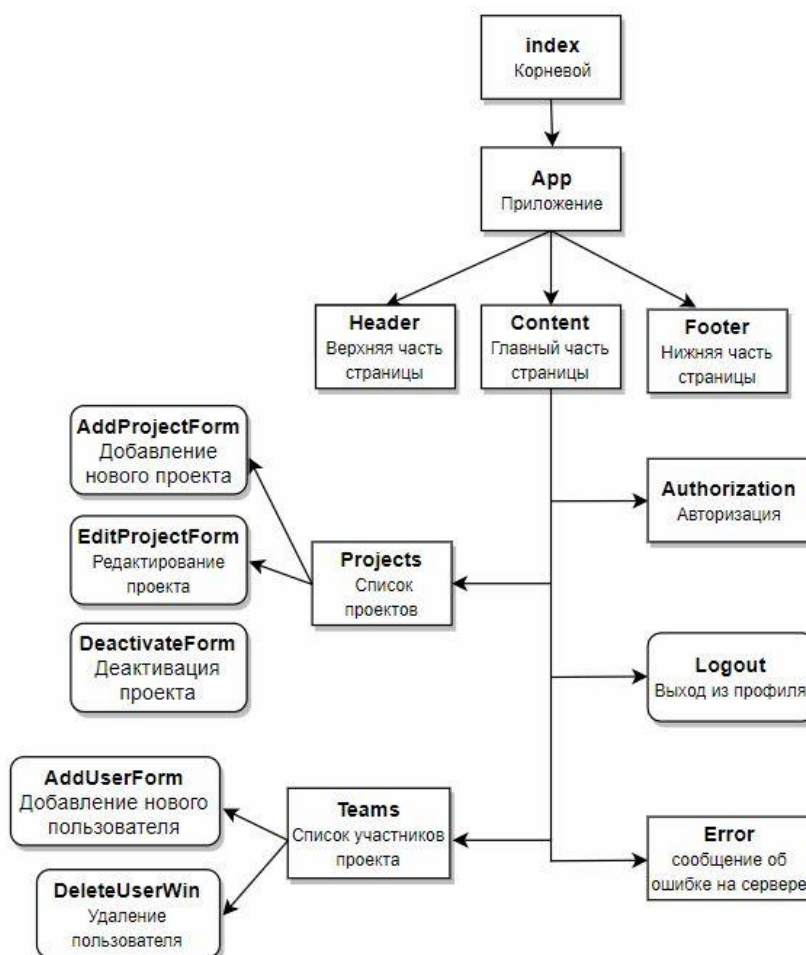


Рисунок 6.4 – Архитектура React-компонентов пользовательского интерфейса для роли «Администратор»

6.3.2 NPM

В 2009 году появился Node.js, который вывел JavaScript за пределы браузеров. Если JavaScript выполняет действия на стороне клиента, то Node.js – на сервере. С помощью Node можно писать полноценные приложения.

Для написания дипломного проекта Node.js использовался в качестве легковесного сервера для фронтенд разработки, который помогал в запуске и тестировании приложения.

NPM (Node Package Manager) – это стандартный менеджер пакетов, автоматически устанавливающийся вместе с Node.js. Он использовался для скачивания пакетов из облачного сервера npm, либо для загрузки пакетов на эти сервера. Все npm-команды прописываются в консоли (терминале) [6]. С помощью npm-скриптов можно создавать небольшие утилиты или описывать сложные системы сборки. Npm-скрипты позволяют запускать и тестировать приложения, настраивать слежение за изменением файлов.

6.3.3 Webpack и Babel

Приложения, написанные на JavaScript, постоянно усложняются, поэтому хорошим выходом является использование сборщика. Webpack – это модульный сборщик проектов, который позволяет объединять все ресурсы необходимые для проекта, в один файл. Это могут быть не только сторонние библиотеки, но и собственные файлы. Такая модульная система позволяет добиться лучшей организации проекта, поскольку он разбит на небольшие модули.

Для того чтобы React работал, необходимо вместе с ним установить Babel. Он предоставляет транспайлинг ES6 и JSX в ES5. Транспайлер – это тип компилятора, который использует исходный код программы, написанной на одном языке программирования, в качестве исходных данных и производит эквивалентный исходный код на другом языке программирования. Транспайлер переводит между языками программирования, которые работают примерно на одном и том же уровне абстракции, в то время как традиционный компилятор переводит с более высокого уровня языка программирования на язык более низкого уровня[5].

6.3.4 React Router

React – это не полноценный фреймворк, а библиотека, т.е. он не решает все потребности приложения. Он позволяет создавать компоненты и управлять состоянием приложения. Для разработки полноценного одностраничного приложения понадобился React Router – это маршрутизатор. В отличие от других подобных он использует JSX. React Router предоставляет несколько собственных компонентов:

1) Router – определяет набор маршрутов, и когда к приложению приходит запрос, Router выполняет сопоставление запроса с маршрутами. И если какой-то маршрут совпадает с URL запроса, то этот маршрут выбирается для обработки запроса;

2) Route – этот компонент не создает DOM элементы, а координирует работу внутренних компонентов, т.е. определяет правила, по которым работает приложение. В атрибутах сюда передается адрес ссылки, которая будет доступна в адресной строке браузера и название компонента, который будет отрисовываться по этому адресу. В Route можно передавать параметры для url-адресов, например, чтобы отобразить конкретную задачу из списка задач передается id [7];

3) Link – используется для создания ссылок и транслируется в обычный `` в DOM. В Link передается соответствующий адрес компонента, указанный в Route и задается название ссылки;

4) Switch – компонент группировки Route'ов. Итеративно проходит по дочерним компонентам и отрисовывает первый, который соответствует имени пути.

6.3.5 Material UI

Material UI – это набор компонентов React, который реализует Google Material Design. Эти компоненты работают изолированно, это означает, что они являются само-поддерживающимися и вводят только те стили, которые они должны отображать.

Данная библиотека доступна не только для веб-разработки, но и для мобильной разработки под iOS и Android. Material UI облегчает разработку и предоставляет множество готовых решений для дизайна интерфейсов. Основные компоненты, использовавшиеся в написании интерфейса:

1) `paper` – определяет блок, который напоминает лист белой бумаги и отдающий небольшую тень под собой. Имеет возможность изменять размер, компоноваться с другими подобными и быть вложенным. Обычно используется в качестве фона;

2) `table` – представляет таблицы, которые помогают отображать данные в удобочитаемом виде для пользователя, упрощают поиск нужной информации;

3) `dialog` – это всплывающие окна, могут использоваться как для простого вывода информации об успешном действии на странице, так и содержать в себе формы для обратной связи, либо поля для заполнения данных;

4) `icon` – определяет системный значок или значок пользовательского интерфейса, который символизирует команду, файл, действие. Google предоставляет широкий набор значков, которые можно использовать при создании интерфейса; Material-UI предоставляет два компонента для отображения системных значков: `SvgIcon` для отображения путей SVG и `Icon` для отображения значков шрифтов;

5) `textField` – текстовые поля, которые позволяют пользователю вводить текст в пользовательском интерфейсе. Обычно они используются в формах и диалогах, поисковых полях. `TextField` транслируется в `input` в HTML разметке;

6) `tabs` – это вкладки, похожие на те, что есть в браузере. `Tabs` помогают организовывать и перемещаться между группами контента, связанными на том же уровне иерархии;

7) `buttons` – компонент, отвечающий за отображение кнопок. Кнопки, обычно служат для сообщения о действиях, которые может выполнять пользователь. Библиотека Material UI дает широчайший выбор дизайна кнопок и анимации при наведении мышкой и клике. Внутри кнопки можно вставлять текст, а также иконки, отражающие действия. Есть несколько размеров кнопок, предоставленных библиотекой по умолчанию. Кнопки можно изменять под нужды собственного дизайна, а именно:

- менять цвет;
- задавать размер;
- задавать радиус закругления углов;
- делать кнопку прозрачной с рамкой или полностью окрашенной и другое.

На рисунке 6.5 приведен ряд кнопок, предоставленных библиотекой Material UI.

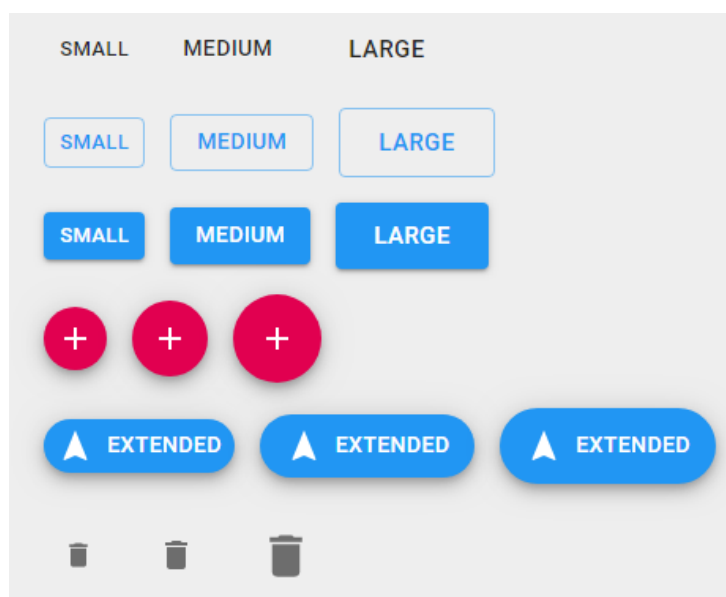


Рисунок 6.5 – Набор кнопок от библиотеки Material UI

8) Typography отвечает за размеры шрифтов и их начертания. Благодаря типографской шкале с ограниченным набором шрифтов, приведенной на рисунке 6.6, без дизайнера в команде можно легко управлять шрифтами на веб-странице.

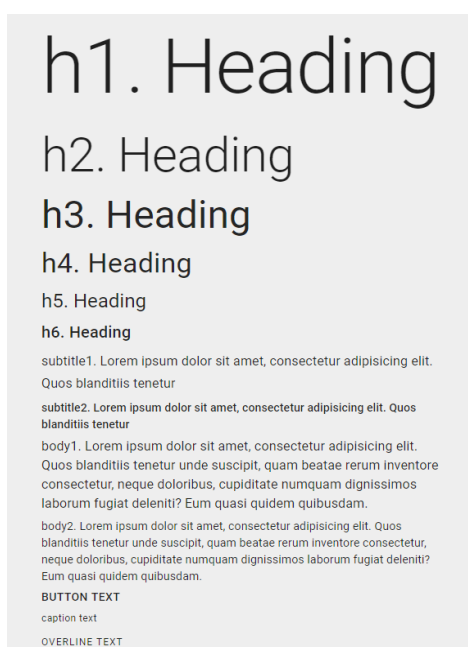


Рисунок 6.6 – Типографская шкала шрифтов Material UI

При реализации пользовательского интерфейса были так же использованы многие другие компоненты, предоставленные библиотекой Material UI. Ряд

перечисленных выше компонентов и других, использовавшихся в разработанном приложении приведены на рисунках 6.7 и 6.8.

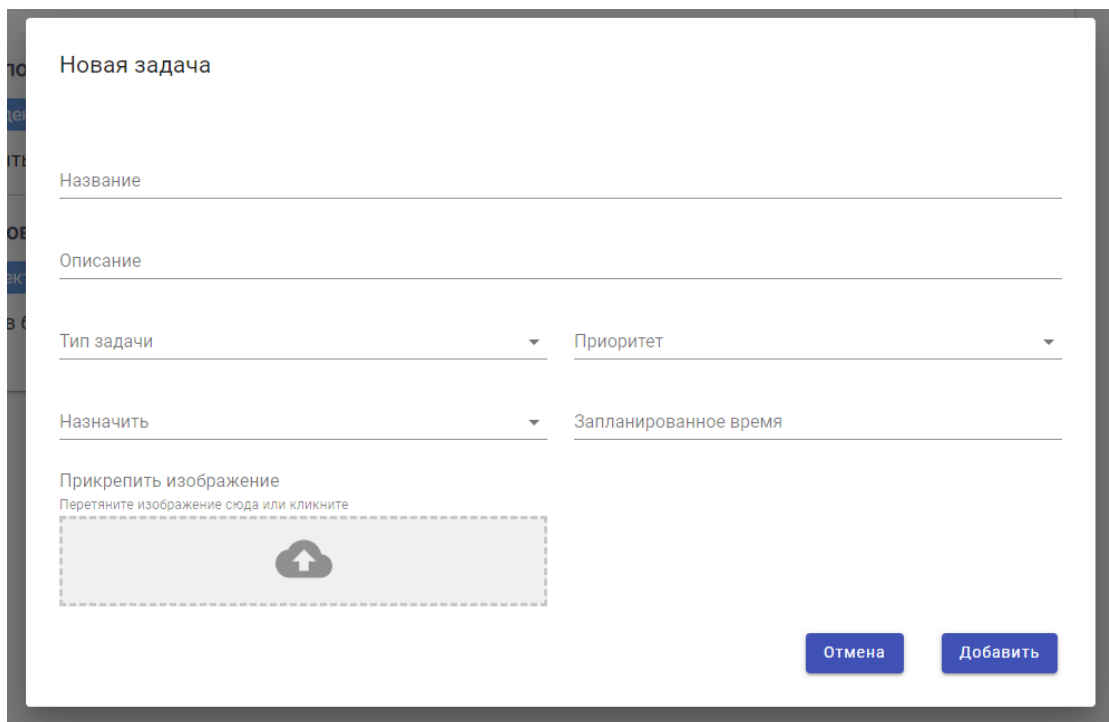


Рисунок 6.7 – Всплывающее окно с формой для создания новой задачи

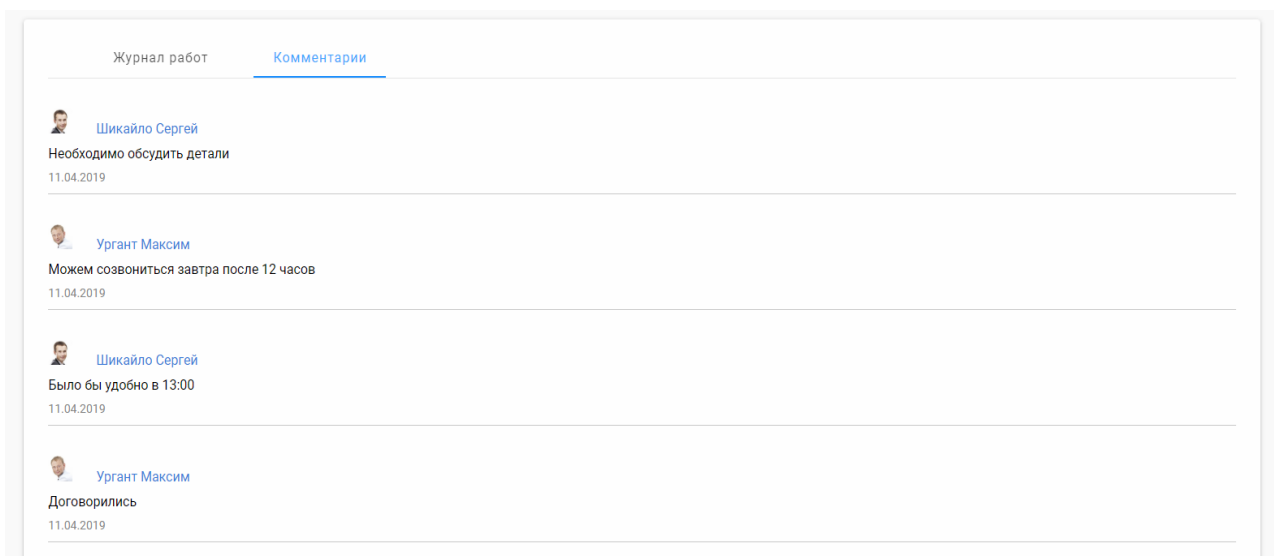


Рисунок 6.8 – Вкладки, содержащие журнал работ и комментарии

Как уже упоминалось выше, предоставленные по умолчанию библиотекой Material UI стили, можно изменять под собственные при помощи CSS (Cascading Style Sheets) – каскадных таблиц стилей. Стили можно писать, как в специальных файлах с разрешением css, так и в самих react-компонентах в формате объекта.

6.3.6 HTML

Несмотря на то, что для написания проекта HTML в чистом виде не использовался, а был заменен JSX, на выходе получилась полноценная HTML-страница. HTML – это язык гипертекстовой разметки, с помощью которого создается структура web-страниц и email-писем. Язык гипертекстовой разметки благодаря тегам позволяет сообщить браузеру смысловую нагрузку элементов на странице[7].

6.3.7 CSS 3 (SASS)

CSS – это язык стилей, которые описывают внешний вид HTML-документа. Им задаются цвета, шрифты, расположения отдельных блоков и другие аспекты представления внешнего вида веб-страниц. Стили могут быть написаны в самом HTML-документе, либо в файле с разрешением css и подключены в документ.

В наше время для больших и сложных проектов обычно приходится писать очень много стилей, что может затруднить читабельность кода и его сопровождение, многие участки в коде повторяются. Чтобы упростить задачу разработчикам были созданы множество библиотек и CSS-препроцессоров.

Язык написания стилей на препроцессорах очень схож с языком CSS, но предоставляет возможности, которые не доступны в CSS. Препроцессоры напрямую в браузере не работают и нуждаются в компиляции, результатом которой является CSS-файл. Компиляцию производят специальные утилиты, многие из них известны как сборщики проектов. В случае разработанного проекта, использовался сборщик Webpack. Его можно настроить на отслеживание изменений в файлах со стилями при их сохранении. Webpack собирает файлы воедино, если есть импортированные дочерние файлы и компилирует их в CSS-файл. А так же обновляет страницу в браузере при каждом сохранении файла, написанного на препроцессоре.

Для разработки интерфейса приложения был использован CSS-препроцессор SASS. Данный препроцессор предоставляет множество полезных функций, например:

1) переменные – позволяют хранить в себе информацию, которую можно использовать много раз на протяжении написания всех стилей проекта. В переменных можно хранить цвета, шрифты и любые другие значения;

2) вложенности – это подобно вложенностям при написании HTML. Благодаря этой возможности читаемость кода становится проще;

3) импорт – позволяет разбивать файлы на более мелкие, таким образом стили становится удобнее сопровождать. Обычный CSS так же имеет эту возможность, но у нее есть существенный недостаток: каждый раз, когда CSS использует @import, то в CSS создается еще один HTTP-запрос. SASS берет идею импорта, но вместо создания отдельного HTTP-запроса, он импортирует указанный файл в тот, где он вызывается.

7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для корректной работы пользовательского интерфейса необходимо, чтобы компьютер пользователя отвечал определенным требованиям. Таким образом, компьютер пользователя должен отвечать следующим аппаратным требованиям:

- 1) процессор Intel Pentium 4 / Athlon 64 или более поздней версии;
- 2) оперативная память объемом не менее 512 МБ;
- 3) свободное место на диске 350 Мб;
- 4) наличие сетевой карты.
- 5) наличие монитора, клавиатуры и мыши для взаимодействия с интерфейсом.

Также необходимо иметь следующее программное обеспечение на пользовательском компьютере:

- 1) операционная система Windows 7 (или более поздняя), Linux либо MacOS;
- 2) браузер (Google Chrome, Safari Firefox либо любой другой браузер);
- 3) в установленном браузере должно быть разрешено исполнение javascript кода.

Поскольку разработанный дипломный проект является полноценным веб-приложением для его использования достаточно наличия веб-браузера и подключения к интернету.

При переходе на сайт, не авторизованный пользователь автоматически перенаправляется на страницу авторизации, которая представлена на рисунке 7.1.

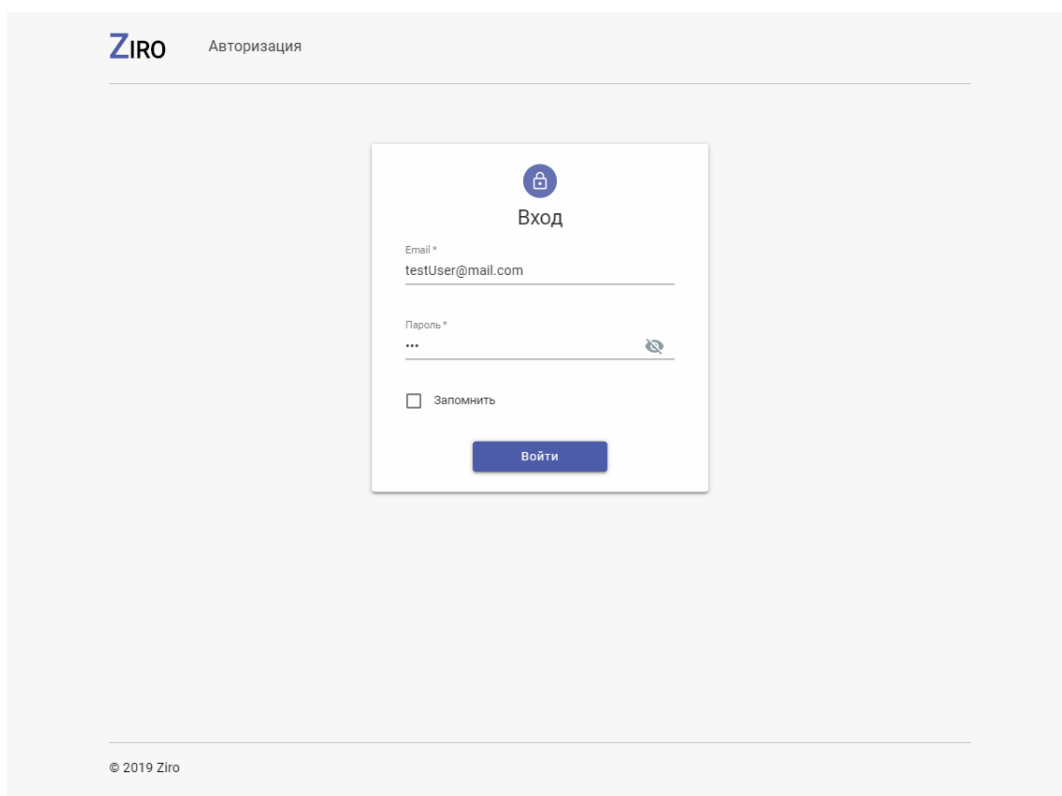


Рисунок 7.1 – Страница авторизации

На этой странице необходимо ввести email и пароль. Далее будет описан интерфейс для роли User.

В случае успешного входа, а именно, где email и пароль оказались верны, и пользователь существует в системе, происходит перенаправление на главную страницу со списком задач. Кроме того, авторизованному пользователю становится доступно меню с навигацией по сайту, доступ в личный кабинет, просмотр информации о задачах и проектах, а также другие функции, которые соответствуют его роли в системе. На главной странице отображается список задач текущего авторизованного пользователя. На этой же странице можно добавить новую задачу. При нажатии на кнопку «Создать задачу» отображается всплывающее окно с формой ввода информации. Главная страница приведена на рисунке 7.2.

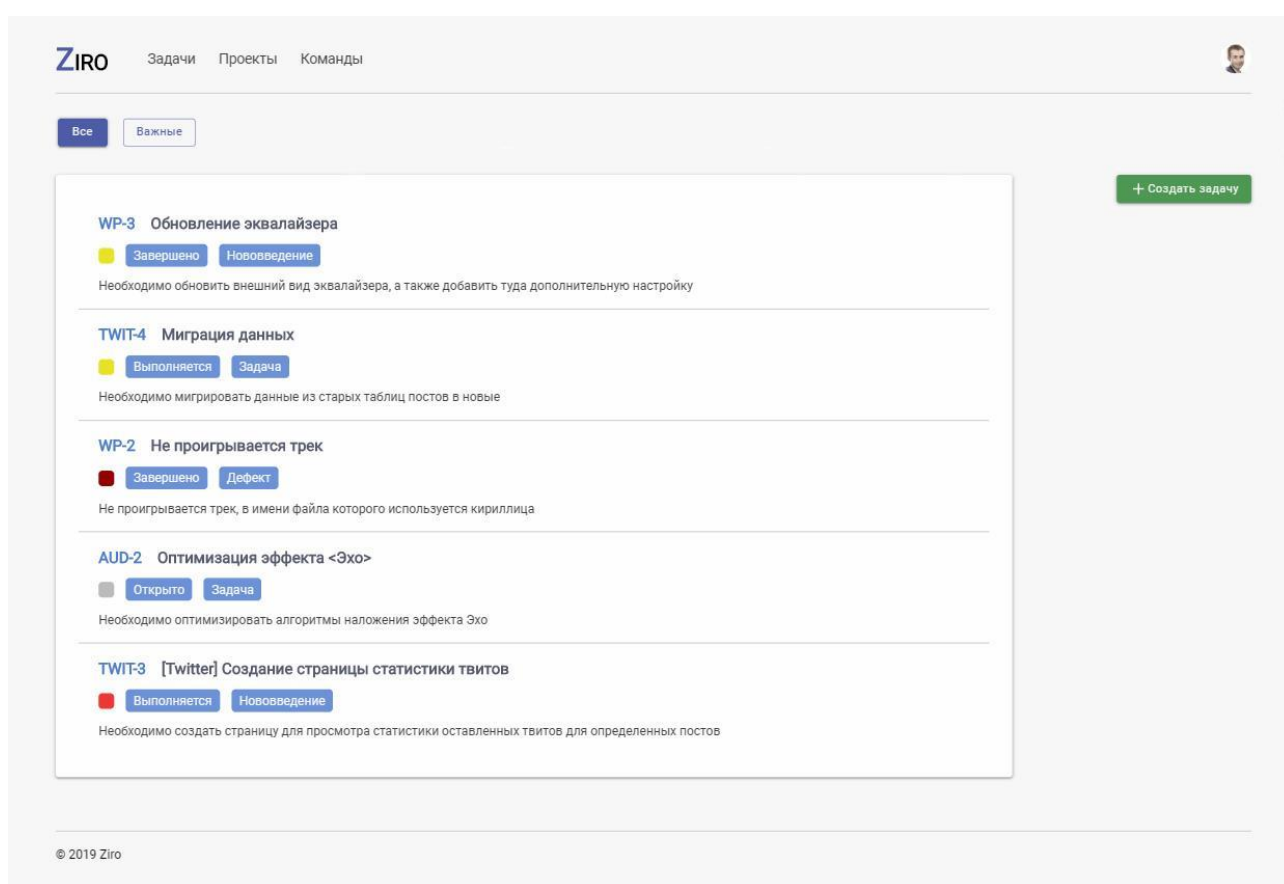


Рисунок 7.2 – Главная страница со списком задач

При клике по блоку с задачей, осуществляется переход на страницу с конкретной задачей. Страница конкретной задачи содержит ее детальное описание. В верхней части можно увидеть номер и название задачи. Ниже расположена информация о создателе задачи и количестве оставленных комментариев. Так же имеется горизонтальная панель кнопок, которая доступна только для создателя задачи. Это кнопки редактирования, назначения ответственного за выполнение

задачи, а также кнопка удаления задачу. Для пользователя, на которого назначена задача, перечисленные выше кнопки будут недоступны.

В правой части страницы находится вертикальная панель, где можно установить статус задачи.

Внизу страницы расположен блок вкладок. Во вкладке комментарии можно читать и писать комментарии. Имеется поле для ввода текста, и кнопка «отправить». Во вкладке журнал работ следует указывать затраченное время на выполнение задачи. Эта вкладка выглядит таким же образом, как и вкладка комментариев. На рисунке 7.3 приведен скриншот страницы детального описания задачи.

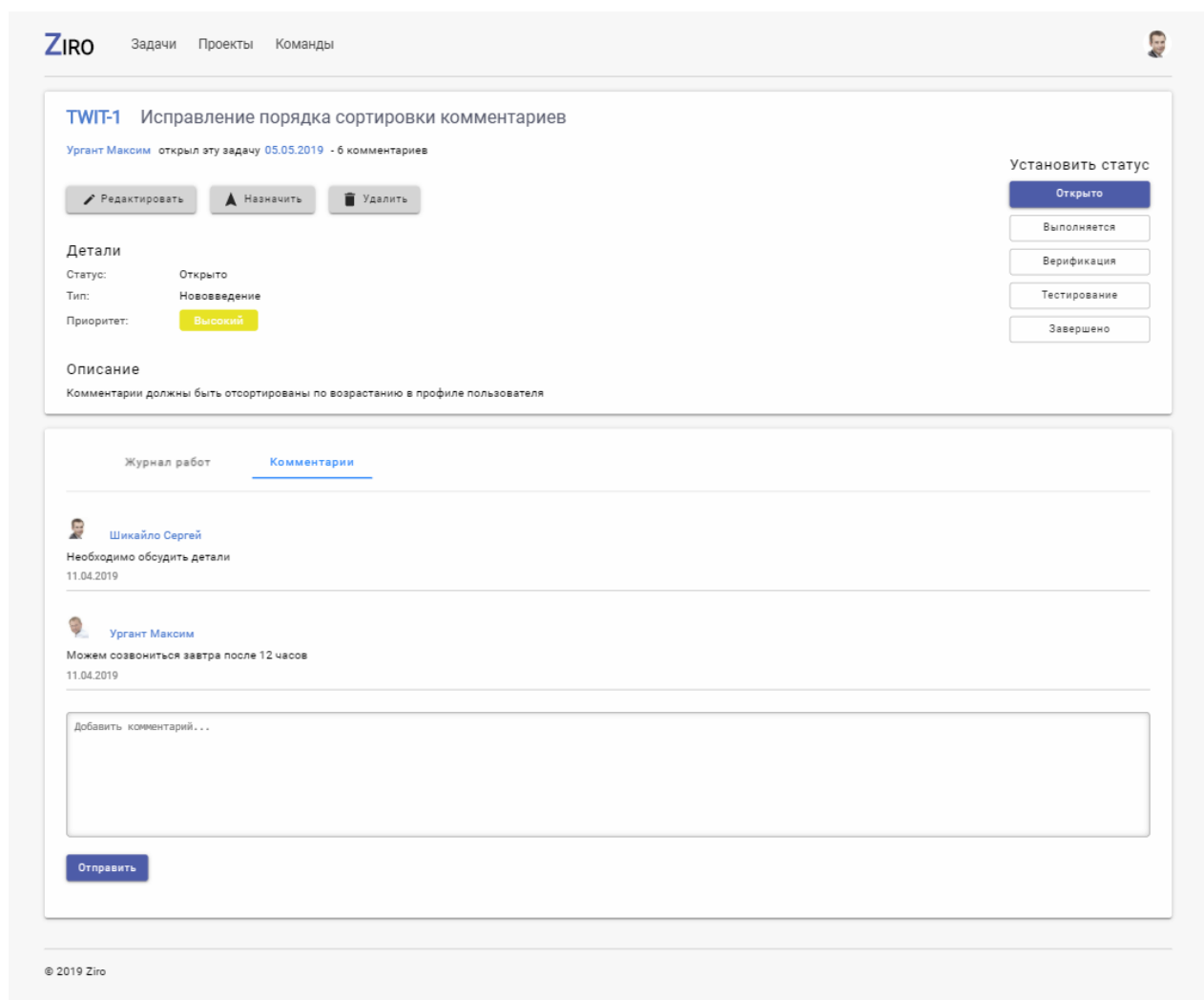


Рисунок 7.3 – Страница детального описания задачи

В профиль пользователя можно перейти, кликнув по фото в правом верхнем углу страницы, при условии, что текущая роль в системе – User. На странице профиля размещена информация о пользователе: имя, должность, контактные данные, если были указаны, дата рождения, список навыков пользователя, а также список проектов, над которыми он работает.

Кликнув по кнопке редактировать (отмена, когда форма открыта) раскрывается блок с формой редактирования, автоматически подставляются данные текущего профиля. Что удобно, в случае, если пользователю необходимо поменять или добавить всего одно поле. При открытии формы редактирования текст кнопки «редактировать» меняется на «отмена». При нажатии на кнопку «сохранить» новые данные перезаписывают старые, форма редактирования закрывается и текст кнопки «отмена» меняется на «редактировать».

При клике по кнопке «выйти» происходит выход из системы, и пользователь перенаправляется на страницу авторизации. Страница профиля пользователя приведена на рисунке 7.4.

The screenshot displays the ZIRO user interface. At the top, there is a navigation bar with the ZIRO logo and links for 'Задачи' (Tasks), 'Проекты' (Projects), and 'Команды' (Teams). A user profile card for 'Шикайло Сергей' (Shikailo Sergey) is shown, an Engineer-Programmer. It includes a profile picture, a 'Выйти' (Logout) button, and an 'Отмена' (Cancel) button. The profile details listed are: email: testUser@mail.com, skype: s.shikailo, phone number, date of birth: 11.03.1994, technologies and skills: C#, ASPNET, DB, javascript, and current projects: Ziro, Machine learning startup.

Below the profile card is the 'Редактирование профиля' (Edit profile) form. It contains the following fields:

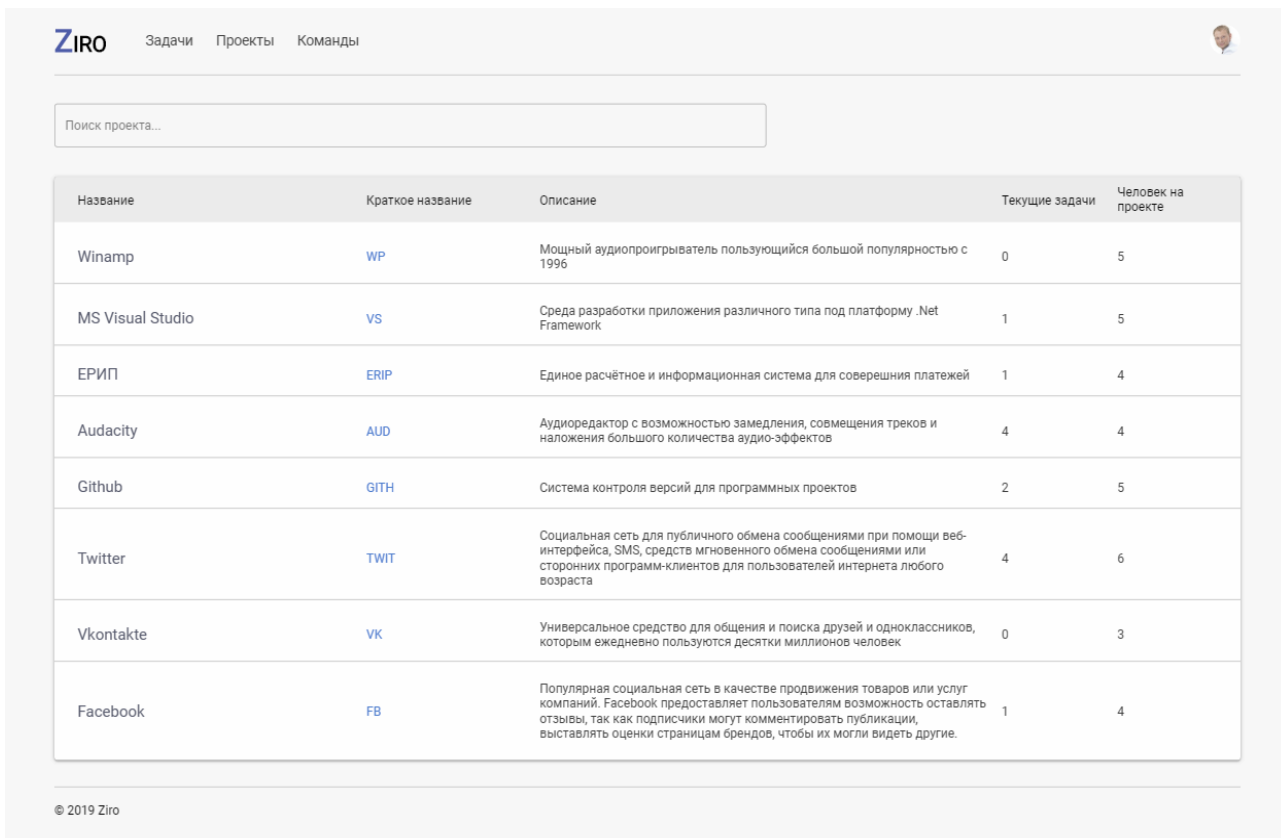
- Имя * (Name): Сергей (Sergey)
- Фамилия * (Surname): Шикайло (Shikailo)
- Отчество (Patronymic): [empty]
- Email *: testUser@mail.com
- Skype: s.shikailo
- Номер телефона * (Phone number): [empty]
- Должность * (Position): Инженер-программист (Engineer-programmer)
- Дата рождения * (Date of birth): 11.03.1994
- Выберите фото профиля (Select profile photo): A dashed box with an upload icon and the text 'Перетяните изображение сюда или кликните' (Drag the image here or click).

 At the bottom of the form is a 'Сохранить' (Save) button.

Рисунок 7.4 – Страница профиля пользователя

Страница «Проекты» содержит таблицу с проектами на которые назначен текущий пользователь системы. Эта страница приведена на рисунке 7.5.

В верхней части страницы расположено поле для поиска проекта. При клике по строке с проектом, пользователь переходит на страницу с детальным описанием проекта.



<div> <div>ZIRO</div> <div>Задачи</div> <div>Проекты</div> <div>Команды</div> </div> <div> <div>Поиск проекта...</div> </div>				
Название	Краткое название	Описание	Текущие задачи	Человек на проекте
Winamp	WP	Мощный аудиопроигрыватель пользующийся большой популярностью с 1996	0	5
MS Visual Studio	VS	Среда разработки приложения различного типа под платформу .Net Framework	1	5
ЕРИП	ERIP	Единое расчётное и информационная система для совершения платежей	1	4
Audacity	AUD	Аудиоредактор с возможностью замедления, совмещения треков и наложения большого количества аудио-эффектов	4	4
Github	GITH	Система контроля версий для программных проектов	2	5
Twitter	TWIT	Социальная сеть для публичного обмена сообщениями при помощи веб-интерфейса, SMS, средств мгновенного обмена сообщениями или сторонних программ-клиентов для пользователей интернета любого возраста	4	6
Vkontakte	VK	Универсальное средство для общения и поиска друзей и одноклассников, которым ежедневно пользуются десятки миллионов человек	0	3
Facebook	FB	Популярная социальная сеть в качестве продвижения товаров или услуг компаний. Facebook предоставляет пользователям возможность оставлять отзывы, так как подписчики могут комментировать публикации, выставлять оценки страницам брендов, чтобы их могли видеть другие.	1	4

© 2019 Ziro

Рисунок 7.5 – Страница с таблицей проектов

Страница детального описания проекта схожа со страницей детального описания задачи. Но также содержит список работников, закрепленных над проектом, предоставляет документацию, хранящуюся в файловых документах. Файлы можно скачать или загрузить новые текстовые-документы с помощью поля для загрузки.

На странице «Команды» отображен список групп пользователей, относящиеся к определенному проекту, сортированных по проектам. Для каждого пользователя указана краткая информация: имя, фамилия, должность и контактная информация. При клике по блоку с конкретным пользователем происходит перенаправление на страницу профиля этого пользователя.

Пользователь, имеющий роль – Administrator, в случае успешного входа в систему, перенаправляется на главную страницу со списком проектов. Эта страница имеет похожий интерфейс, как и одноименная страница для роли User, за тем исключением, что для администратора может создавать проекты. При нажатии на кнопку «создать проект» открывается всплывающее окно с формой ввода.

Администратору доступна страница со списком пользователей, которые сортированы по проектам. Также есть кнопка создания нового пользователя, клике по которой, открывается всплывающее окно для ввода и отправки информации. Администратор должен внести имя и фамилию пользователя, а также указать email, выбрать из выпадающего списка соответствующую должность и проект.

8 РАЗВЕРТЫВАНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОЙ СИСТЕМЫ

8.1 Развертывание клиентской части

Для развертывания клиентской части системы необходимо иметь компьютер или ноутбук, в котором обязательно наличие сетевой карты и доступа в интернет. Из программного обеспечения на нем должна быть установлена какая-либо операционная система, например, Windows, Linux, MacOS и т.д.

На компьютере или ноутбуке должен быть установлен браузер, например, Google Chrome, Opera, Mozilla FireFox, Safari, Edge и т.д. А так же для корректной работы веб-приложения в настройках браузера стоит включить модуль исполнения JavaScript-кода. Схемы аппаратного и программного развертывания представлены на рисунках 8.1 и 8.2 соответственно.

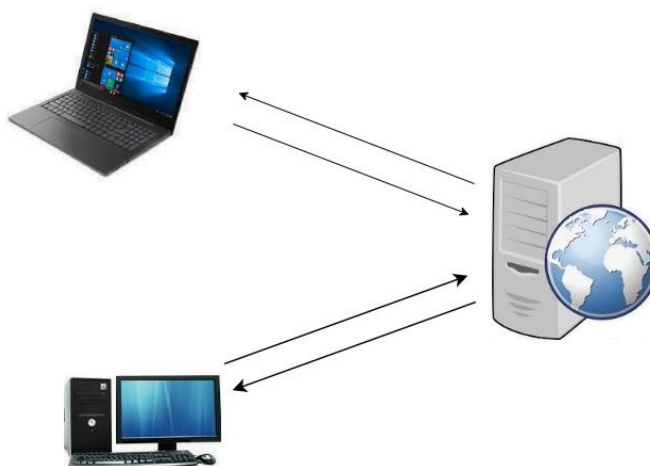


Рисунок 8.1 – Схема аппаратного развертывания



Рисунок 8.2 – Схема программного развертывания

8.2 Тестирование программной системы

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта.

Тестирование программного обеспечения будет проводится в рамках функционального тестирования, разделив его на критическое и углубленное.

Тестирование программы состоит из разработки тестовых случаев (тестов), их запуска и анализа полученных результатов.

Тестовые случаи – это алгоритмы проверки функциональности программы.

Каждый тестовый случай должен обладать следующими свойствами:

- 1) четкой целью проверки;
- 2) известными начальными условиями тестирования;
- 3) строго определенной средой тестирования;
- 4) тестовыми данными и ожидаемым результатом тестирования.

Тестирование было проведено в следующих версиях браузеров и операционных системах, представленных в таблице 8.1.

Таблица 8.1 – Матрица конфигурации веб-приложения

	Google Chrome v 74.0.3729.157	Opera v 11.16	Mozilla Firefox v 56.0	Microsoft Edge 42.17134.1.0
Windows 10	+	+	+	+
Windows 7	+	+	+	+

8.2.1 Критическое тестирование

Критическое тестирование – это процесс поиска ошибок в программе при стандартной ее работе (при правильной последовательности действий, при верном заполнении полей и т. д.).

Тест критического пути является одним из самых распространенных видов функционального тестирования, в частности, для веб-проектов. Частота данного тестирования обусловлена в первую очередь необходимостью периодической проверки всего приложения в сжатые сроки. Плюс ко всему позволяет выявить самые быстро находимые дефекты и исправить приложение в более сжатые сроки.

С помощью данного вида тестирования покрываются все сценарии стандартного использования приложения, исключая негативные сценарии. Данный тип также характеризует тестирование по глубине его проведения.

Это и есть критическое тестирование, благодаря которому можно проверить

правильность работы часто используемого функционала.

Тестовые случаи для критического тестирования представлены в таблице 8.2.

Таблица 8.2 – Тестовые случаи для критического тестирования

№	Название модуля/экрана	Описание тестового случая	Ожидаемые результаты	Тестовый случай пройден? Да/Нет	Комментарии
1	2	3	4	5	6
1	Запуск приложения	Запуск приложения в различных браузерах: 1) запуск браузера; 2) ввод адреса тестируемого веб-сайта.	1) открытие браузера; 2) загрузка страницы входа в систему	Да	
2	Аутентификация	Аутентификация 1) переход на тестируемый веб-сайт в браузере; 2) ввод в поле Email значения «test@mail.com»; 3) ввод в поле Пароль «123»; нажатие кнопки «Войти».	1) открытие стартовой страницы пользователя	Да	Также можно увидеть полученные с сервера куки в консоли браузера
3	Авторизация пользователя	Авторизация пользователя: 1) переход на тестируемый веб-сайт в браузере; 2) ввод в поле Email значения «user@mail.com»; 3) ввод в поле Пароль «user»; 4) нажатие кнопки «Войти».	1) открытие стартовой страницы пользователя со списком текущих задач; 2) показ верхнего меню, состоящего из задач, проектов и команд, а также отображение изображения пользователя в верхнем правом углу	Да	
4	Авторизация администратора	Авторизация администратора: 1) переход на тестируемый веб-сайт в браузере; 2) ввод в поле Email значения admin@mail.com; 3) ввод в поле Пароль «admin»; нажатие кнопки «Войти».	1) открытие стартовой страницы администратора со списком; проектов системы 2) показ верхнего меню, состоящего из пользователей и проектов	Да	

Продолжение таблицы 8.2

1	2	3	4	5	6
5	Создание задачи	Создание задачи 1) авторизация на веб-сайте как пользователь; 2) нажатие кнопки «Создать задачу» в открывшемся окне; 3) заполнить все поля и нажать кнопку создать.	1) обновление фотографии в верхнем правом углу; обновление информации о текущем пользователе.	Да	
6	Редактирование профиля пользователя	Редактирование профиля пользователя: 1) авторизация на веб-сайте как пользователь; 2) нажатие на изображение; пользователя в верхнем правом углу пользователя 3) нажатие кнопки «Редактировать»; ввод данных и загрузка новой фотографии.	1) обновление фотографии в верхнем правом углу; обновление информации о текущем пользователе.	Да	
7	Создание пользователя администратором	Создание пользователя администратором: 1) авторизация в приложении как администратор; 2) переход на вкладку «Пользователи» в верхнем меню; 3) нажатие кнопки «Создать»; 4) заполнение всех данных в открывшемся окне и нажатие кнопки «Сохранить».	1) переход на страницу с пользователями; 2) появление созданного пользователя в списке.	Да	
8	Создание проекта администратором	Создание проекта администратором: 1) авторизация в приложении как администратор; 2) переход на вкладку «Проекты» и нажатие кнопки «Создать» 3) заполнение всех данных и нажатие кнопки «Сохранить».	1) переход на страницу с проектами; 2) появление созданного проекта в списке.	Да	

Продолжение таблицы 8.2

1	2	3	4	5	6
9	Создание записи в журнале работ	Создание записи в журнале работ: 1) авторизация в приложении как пользователь; 2) переход на любую из задач в списке задач на появившейся странице; 3) выбор вкладки «Журнал работ» в отобразившейся странице информации о задаче; 4) нажатие кнопки «Добавить»; 5) ввод текста и потраченного времени в появившихся полях и нажатие кнопки «Добавить».	1) поля с вводом данных исчезают; 2) добавленная запись появляется в журнале работ, а также увеличивается значение в поле «Затраченное время» у задачи на размер введенного значения потраченного времени.	Да	
10	Комментирование задачи	Комментирование задачи: 1) авторизация в приложении как пользователь; 2) переход на любую из задач в списке задач на появившейся странице; 3) выбор вкладки «Комментарии» в отобразившейся странице информации о задаче; 4) нажатие кнопки «Добавить комментарий»; 5) ввод текста в появившемся поле и нажатие кнопки «Добавить».	1) поле с вводом комментария исчезает; 2) добавленный комментарий появляется в списке комментариев к задаче в самом конце.	Да	

8.2.2 Углубленное тестирование

Углубленное (расширенное) тестирование – это процесс поиска ошибок в программе в нестандартных, непредвиденных ситуациях при использовании веб-системы (например, при некорректно вводимых данных или специальных символов, границах переполнения массивов, данных).

Примеры тестовых случаев для углубленного тестирования для проверки функциональности работы с данными представлены ниже в таблице 8.3.

Таблица 8.3 - Тестовые случаи для углубленного тестирования

№	Название модуля/экрана	Описание тестового случая	Ожидаемые результаты	Тестовый случай пройден? Да/Нет	Комментарии
1	2	3	4	5	6
1	Авторизация пользователя	Аутентификация пользователя с незаполненными полем «Пароль» или «Email»: 1) открытие тестируемого веб-сайта в браузере; 2) ввод одного из полей; нажатие кнопки «Войти».	1) показ сообщения об ошибке о необходимости ввода пароля или Email	Да	
2	Создание пользователя другим пользователем	Попытка создание пользователя другим пользователем: 1) вход в систему под учетными данными пользователя; 2) ввод значения \users\create в адресной строке браузера в дополнение к имени тестируемого сайта; 3) нажатие клавиши «Ввод».	1) переход на страницу с показом ошибки о запрещении доступа и кодом ошибки 403	Да	
3	Оставление пустого комментария к задаче	Комментирование задачи: 1) авторизация в приложении как пользователь; 2) переход на любую из задач в списке задач на появившейся странице; 3) выбор вкладки «Комментарии» в отобразившейся странице информации о задаче; 4) нажатие кнопки «Добавить комментарий»; оставление пустым поле с текстом и нажатие кнопки «Добавить».	1) отображение сообщения об ошибке «Комментарий не может быть пустым».	Да	
4	Создание пользователя	Создание пользователя без указания Email: 1) авторизация в приложении как администратор; 2) переход на вкладку «Пользователи» в верхнем меню; 3) нажатие кнопки «Создать»; 4) заполнение всех данных в открывшемся окне кроме поля «Email».	1) отображение сообщения об ошибке «Поле Email не может быть пустым».	Да	Данные ошибки можно также увидеть в консоли разработчика браузера

Таким образом, в результате проведения критического и углубленного тестирования не было выявлено каких-либо ошибок в разработанном функционале. Система полностью соответствует поставленным требованиям и корректно ведет себя при неправильном использовании реализованного функционала.

9 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОЙ СИСТЕМЫ

9.1 Расчет сметы затрат, цены и прибыли на программное средство

В современных рыночных экономических условиях программное средство выступает преимущественно в виде продукции научно-технических организаций, представляющей собой функционально завершенные и имеющие товарный вид программные средства, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки программных средств являются научно-технической продукцией.

Выбор эффективных проектов связан с их экономической оценкой и расчетом экономического эффекта.

У разработчика экономический эффект выступает в виде чистой прибыли, а у пользователя – в виде экономии различных ресурсов, получаемой за счет:

- 1) снижения трудоемкости расчетов и алгоритмизации программирования;
- 2) отладки программ за счет использования программного средства в процессе разработки автоматизированных систем обработки данных;
- 3) снижения расходов на материалы (магнитные ленты, магнитные диски и прочие материалы);
- 4) улучшения показателей основной деятельности предприятий в результате использования программного средства.

Стоимостная оценка программного средства у разработчиков предполагает составление сметы затрат, которая включает следующие статьи:

- 1) заработная плата исполнителей основная (Z_o) и дополнительная (Z_d);
- 2) отчисления в фонд социальной защиты населения ($Z_{сз}$);
- 3) отчисления на развитие здравоохранения и охрану здоровья ($Z_{оз}$);
- 4) налоги, от фонда оплаты труда (H_e);
- 5) материалы и комплектующие (M);
- 6) спецоборудование (P_c);
- 7) машинное время (P_m);
- 8) расходы на научные командировки ($P_{нк}$);
- 9) прочие прямые затраты (Π_z);
- 10) накладные расходы (P_n).

Данное программное средство является программным средством общего назначения, представляющее собой функционально завершенное и имеющее товарный вид программное средство.

На основании сметы затрат рассчитывается себестоимость и отпускная цена программного средства.

9.1.1 Исходные данные

Исходные данные для расчета заносятся в таблицу (таблица 9.1).

Таблица 9.1 – Исходные данные для расчета

№ пп	Наименование показателя	Единица измерения	Условные обозначения	Нормат ив
1.	Коэффициент изменения скорости обработки информации	ед.	$K_{ск}$	0,5
2.	Численность разработчиков	чел.	$Ч_p$	2
3.	Тарифная ставка 1-го разряда в организации	руб.	$T_{м1}$	36,4
4.	Тарифный коэффициент	ед.	T_k	2,48
5.	Норматив дополнительной заработной платы рассчитывается по формуле	%	H_d	16
6.	Норматив отчислений в фонд социальной защиты населения	ед.	$H_{сз}$	35
7.	Норматив налога, уплачиваемого единым платежом	%	$H_{не}$	4
8.	Норматив расходов на командировки в целом по научной организации	%	$H_{рнк}$	30
9.	Норматив прочих затрат в целом по научной организации	%	$H_{пз}$	20
10.	Норматив накладных расходов в целом по научной организации	%	$H_{рн}$	100
11.	Уровень рентабельности программного средства	%	$У_{рн}$	53
12.	Норматив расходов на сопровождение и адаптацию	%	$H_{рса}$	4
13.	Норматив отчислений в местный и республиканский бюджеты	%	$H_{мр}$	3,9
14.	Норматив НДС	%	$H_{дс}$	20
15.	Налог на прибыль при отсутствии льгот	%	H_n	24
16.	Норматив расхода машинного времени на отладку 100 строк исходного кода	час	$H_{мв}$	12
17.	Количество дней в году	день	D_r	365
18.	Количество праздничных дней в году	день	$D_{п}$	9
19.	Количество выходных дней в году	день	$D_{в}$	104
20.	Количество дней отпуска	день	D_o	24

9.1.2 Общая характеристика разрабатываемого программного средства

Данное программное средство имеет общего назначения, основные функции:

- 1) организация ввода информации;
- 2) контроль, предварительная обработка и ввод информации;
- 3) управление вводом/выводом;
- 4) генерация структуры базы данных и манипулирование данными;
- 5) организация поиска и поиск в базе данных;
- 6) обработка ошибочных и сбойных ситуаций;

Разрабатываемое программное средство входит в третью группу сложности.

9.1.3 Определение объема программного средства

Общий объем программного средства рассчитывается по формуле:

$$V_o = \sum_{i=1}^n V_i, \quad (9.1)$$

где V_o – общий объем программного средства, строка исходного кода;

V_i – объем функций программного средства, строка исходного кода;

n – общее число функций.

Содержание и объем функций разрабатываемого программного средства приведено в таблице 9.2.

Таблица 9.2 – Содержание и объем функций разрабатываемого программного средства

№ функции	Содержание функции	Объем функций (строки исходного кода)
101	Организация ввода информации	150
102	Контроль, предварительная обработка и ввод информации	450
111	Управление вводом/выводом	2 400
201	Генерация структуры базы данных	4 300
207	Манипулирование данными	9 550
208	Организация поиска и поиск в базе данных	5 480
304	Обслуживание файлов	420
506	Обработка ошибочных и сбойных ситуаций	410
604	Справка и обучение	720
	ИТОГО	23 880

$$V_o = 150 + 450 + 2\,400 + 9\,550 + 5\,480 + 420 + 410 + 720 = 23\,880 \text{ строк кода.}$$

Объем программного средства определяется путем подбора аналогов на основании классификации типов программных средств, каталога функции программного средства и аналогов программного средства в разрезе функций, которые постоянно обновляются и утверждаются в установленном порядке, представленном выше.

9.1.4 Расчет трудоемкости выполняемой работы

На основании общего объема программного средства определяется нормативная трудоемкость (T_H). Нормативная трудоемкость устанавливается с учетом сложности программного средства. Выделяется три группы сложности, в которых учтены следующие составляющие программного средства:

- языковой интерфейс;
- ввод-вывод;
- организация данных;
- режим работы;
- операционная и техническая среда.

С учетом дополнительного коэффициента сложности $K_{СЛ}$ рассчитывается общая трудоемкость программного средства:

$$T_o = T_H \cdot K_{СЛ}, \quad (9.2)$$

где T_o – общая трудоемкость ПС, человеко-день;

T_H – нормативная трудоемкость ПС, человеко-день;

$K_{СЛ}$ – дополнительный коэффициент сложности ПС.

При 1 группе сложности нормативная трудоемкость T_H равна 442. ПС обладает характеристикой функционирования в расширенной операционной среде, поэтому коэффициент сложности $K_{сл}$ равен 0.08.

$$T_o = 442 * 0,08 = 35,36 \text{ человеко – дней.}$$

При решении сложных задач с длительным периодом разработки ПС трудоемкость определяется по стадиям разработки с учетом новизны, степени использования типовых программ и удельного веса трудоемкости стадий разработки ПС и общей трудоемкости разработки ПС. При 1 группе сложности нормативная трудоемкость T_H равна 442. ПС обладает характеристикой функционирования в

расширенной операционной среде. При этом на основании общей трудоемкости рассчитывается уточненная трудоемкость с учетом распределения по стадиям (T_y).

$$T_y = \sum_{i=1}^m T_i, \quad (9.3)$$

где T_i – трудоемкость разработки ПС на i -й стадии, человеко-день;

m – количество стадий разработки.

Трудоемкость ПС по стадиям определяется с учетом новизны и степени использования в разработке типовых программ и ПС.

$$T_{cTi} = d_{cTi} \cdot K_H \cdot K_T \cdot T_O, \quad (9.4)$$

где T_{cTi} – трудоемкость разработки ПС на i -й стадии (технического задания, эскизного проекта, технического проекта, рабочего проекта и внедрения), человеко-день;

K_H – поправочный коэффициент, учитывающий степень новизны ПС;

K_T – поправочный коэффициент, учитывающий степень использования в разработке типовых программ и ПС;

d_{cTi} – удельный вес трудоемкости i -й стадии разработки ПС в общей трудоемкости разработки ПС.

ПС относится к 3-ей группе сложности и имеет степень новизны В, при этом $K_{сл}=0,08$, а $K_H=0,7$. Коэффициенты удельных весов трудоемкостей для каждой стадии: ТЗ=0,09; ЭП=0,07; ТП=0,07; РП=0,61; ВН=0,16.

Исходя из этих данных, можно рассчитать трудоемкость ПС на каждой стадии:

$$T_{cТЗ} = 0,09 \cdot 0,7 \cdot 0,7 \cdot 35,36 = 1,56 \text{ человеко – дней.}$$

$$T_{cЭП} = 0,07 \cdot 0,7 \cdot 0,7 \cdot 35,36 = 1,21 \text{ человеко – дней.}$$

$$T_{cТП} = 0,07 \cdot 0,7 \cdot 0,7 \cdot 35,36 = 1,21 \text{ человеко – дней.}$$

$$T_{cРП} = 0,61 \cdot 0,7 \cdot 0,7 \cdot 35,36 = 10,57 \text{ человеко – дней.}$$

$$T_{cВН} = 0,16 \cdot 0,7 \cdot 0,7 \cdot 35,36 = 2,77 \text{ человеко – дней.}$$

Тогда уточненная трудоемкость с учетом распределения по стадиям равна:

$$T_y = 1,56 + 1,21 + 1,21 + 10,57 + 2,77 = 17,33 \text{ человеко – дня.}$$

Таким образом уточненная трудоемкость с учетом распределения по стадиям составляет 17,33 человеко-дня.

9.1.5 Расчет основной заработной платы исполнителей

Эффективный фонд времени работы одного работника ($\Phi_{\text{эф}}$) рассчитывается по формуле:

$$\Phi_{\text{эф}} = D_{\Gamma} - D_{\Pi} - D_{\text{в}} - D_{\text{о}}, \quad (9.5)$$

где D_{Γ} – количество дней в году, день;

D_{Π} – количество праздничных дней в году, день;

$D_{\text{в}}$ – количество выходных дней в году, день;

$D_{\text{о}}$ – количество дней отпуска, день.

При утверждении плановой численности разработчиков продолжительность разработки определяется по формуле:

$$T_{\text{р}} = \sum_{i=1}^m \frac{T_i}{\text{Ч}_{\text{р}i} \cdot \Phi_{\text{эф}}}, \quad (9.6)$$

где $T_{\text{р}}$ – срок разработки ПС (лет);

T_i – трудоемкость разработки ПС на i -й стадии (человеко-дней);

$\text{Ч}_{\text{р}i}$ – численность разработчиков ПС на i -й стадии (чел.);

m – число стадий.

Уточненная трудоемкость и общая плановая численность разработчиков служат базой для расчета основной заработной платы. По данным о специфике и сложности выполняемых функций составляется штатное расписание группы специалистов-исполнителей, участвующих в разработке ПС, с определением образования, специальности, квалификации и должности.

Таким образом, можно рассчитать эффективный фонд времени:

$$\Phi_{\text{эв}} = 365 - 9 - 104 - 24 = 228 \text{ дней.}$$

А также срок разработки ПС при количестве разработчиков на всех стадиях разработки ПС равным 2.

$$T_{\text{р}} = \frac{17,3264}{2 * 228} = 0,038 \text{ лет.}$$

В соответствии с «Рекомендациями по применению «Единой тарифной сетки» рабочих и служащих народного хозяйства» и тарифными разрядами и коэффициентами должностей руководителей научных организаций и

вычислительных центров, бюджетных учреждений науки непроизводственных отраслей народного хозяйства каждому исполнителю устанавливается разряд и тарифный коэффициент.

Месячная тарифная ставка каждого исполнителя (T_M) определяется путем умножения действующей месячной тарифной ставки 1-го разряда (T_{M1}) на тарифный коэффициент (T_K), соответствующий установленному тарифному разряду:

$$T_M = T_{M1} * T_K. \quad (9.7)$$

Часовая тарифная ставка рассчитывается путем деления месячной тарифной ставки на установленный при семичасовом рабочем дне фонд рабочего времени (Φ_p):

$$T_{\text{ч}} = \frac{T_M}{\Phi_p}, \quad (9.8)$$

где $T_{\text{ч}}$ – часовая тарифная ставка, руб.;

T_M – месячная тарифная ставка, руб.

Оба работника имеют одинаковую квалификацию и принадлежат 10-му разряду. Тарифная месячная и часовая ставки для них будут равны:

$$T_M = 36,4 * 2,48 = 90,27 \text{ руб.}$$

$$T_{\text{ч}} = \frac{90,72}{133} = 0,68 \text{ руб.}$$

Основная заработная плата исполнителей на конкретное ПС рассчитывается по формуле

$$Z_{oi} = \sum_{i=1}^n T_{\text{чи}} \cdot T_{\text{ч}} \cdot \Phi_{\text{эи}} \cdot K, \quad (9.9)$$

где n – количество исполнителей, занятых разработкой конкретного ПС;

$T_{\text{чи}}$ – часовая тарифная ставка i -го исполнителя (руб.);

$\Phi_{\text{эи}}$ – эффективный фонд рабочего времени i -го исполнителя (дней);

$T_{\text{ч}}$ – количество часов работы в день (ч);

K – коэффициент премирования.

$$Z_o = 2 * 0,68 * 133 * 1 = 180,54 \text{ руб.}$$

9.1.6 Расчет дополнительной заработной платы исполнителей

Дополнительная заработная плата на конкретное ПС ($З_{дi}$) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат), и определяется по нормативу в процентах к основной заработной плате, приведены в формуле 9.10:

$$З_{дi} = \frac{З_{oi} \cdot Н_{д}}{100}, \quad (9.10)$$

где $З_{дi}$ – дополнительная заработная плата исполнителей на конкретное ПС, руб.;
 $Н_{д}$ – норматив дополнительной заработной платы, %.

$$З_{д} = \frac{180,54 \cdot 16}{100} = 28,89 \text{ руб.}$$

9.1.7 Расчет отчислений в фонд социальной защиты населения

Отчисления в фонд социальной защиты населения ($З_{сzi}$) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной заработной платы исполнителей программного продукта:

$$З_{сzi} = \frac{(З_{oi} + З_{дi}) \cdot Н_{сз}}{100}, \quad (9.11)$$

где $Н_{сз}$ – норматив отчислений в фонд социальной защиты населения, %.

$$З_{сzi} = \frac{(180,544 + 28,88704) \cdot 35}{100} = 73,3 \text{ руб.}$$

9.1.8 Расчет налога на ликвидацию последствий чернобыльской катастрофы

Налоги рассчитываемые от фонда оплаты труда определяются в процентном отношении к сумме всей заработной платы, относимой на ПС (налог, уплачиваемый единым платежом, включая налог на ликвидацию последствий чернобыльской катастрофы и отчисления в фонд занятости ($Н_{ei}$)):

$$H_{ei} = \frac{(3_{oi} + 3_{di}) \cdot H_{не}}{100}, \quad (9.12)$$

где $H_{не}$ – норматив налога, уплачиваемого единым платежом (%).

$$H_{ei} = \frac{(180,544 + 28,88704) \cdot 4}{100} = 8,38 \text{ руб.}$$

9.1.9 Расчет расходов по статье «Материалы»

Расходы по статье «Материалы» (M) определяются на основании сметы затрат, разрабатываемой на ПС с учетом действующих нормативов. По статье «Материалы» отражаются расходы на магнитные носители, перфокарты, бумагу, красящие ленты и другие материалы, необходимые для разработки ПС. Нормы расхода материалов в суммарном выражении (H_M) определяются в расчете на 100 строк исходного кода. Сумма затрат материалов рассчитывается по формуле:

$$M_i = H_{Mi} \cdot \frac{V_{oi}}{100}, \quad (9.13)$$

где H_{Mi} – норма расхода материалов в расчете на 100 строк исходного кода ПС, руб.;

V_{oi} – общий объем ПС на конкретное ПС, строка исходного кода.

$$M_i = 0,038 \cdot \frac{23\,880}{100} = 9,07 \text{ руб.}$$

9.1.10 Расчет расходов по статье «Спецоборудование»

Расходы по статье «Спецоборудование» (P_{ci}) включает затраты средств на приобретение вспомогательных специального назначения технических и программных средств, необходимых для разработки конкретного ПС, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию. Затраты по этой статье определяются в соответствии со сметой расходов, которая составляется перед разработкой ПС. Данная статья включается в смету расходов на разработку ПС в том случае, когда приобретаются специальное оборудование или специальные программы, предназначенные для разработки данного ПС:

$$P_{ci} = \sum_{i=1}^n C_{ci}, \quad (9.14)$$

где Π_{Ci} – стоимость конкретного специального оборудования, руб.;

n – количество применяемого специального оборудования.

Необходимо учесть следующие расходы по данной статье:

- хостинг-план для размещения веб-сайта в интернете) – 28,9 рублей;
- VS Resharper – 55,23 рублей.

$$P_{ci} = 28,9 + 55,23 = 84,13 \text{ руб.}$$

9.1.11 Расчет расходов по статье «Машинное время»

Расходы по статье «Машинное время» (P_{Mi}) включают оплату машинного времени, необходимого для разработки и отладки ПС, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода (H_{MB}) машинного времени в зависимости от характера решаемых задач и типа ПЭВМ:

$$P_{Mi} = \Pi_{Mi} \cdot \frac{V_{oi}}{100} \cdot H_{MB}, \quad (9.15)$$

где Π_{Mi} – цена одного машино-часа, руб.;

V_{oi} – общий объем ПС, строка исходного кода;

H_{MB} – норматив расхода машинного времени на отладку 100 строк исходного кода, машино-час.

Цена машино-часа (Π_{Mi}) для обеспечения работы веб-сервера составляет 2 рубля.

$$P_{Mi} = 2 * \left(\frac{23\,380}{100} \right) * 12 = 5\,731,2 \text{ руб.}$$

9.1.12 Расчет расходов по статье «Научные командировки»

Расходы по статье «Научные командировки» (P_{Hki}) на конкретное ПС определяются по нормативу, разрабатываемому в процентах к основной зарплате:

$$P_{Hki} = \frac{3_{oi} \cdot H_{pнк}}{100}, \quad (9.16)$$

где $H_{pнк}$ – норматив расходов на командировки в целом по научной организации, %.

$$P_{Hki} = \frac{180,54 * 30}{100} = 54,16 \text{ руб.}$$

9.1.13 Расчет расходов по статье «Прочие затраты»

Расходы по статье «Прочие затраты» (Π_{zi}) на конкретное ПС включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по научной организации, в процентах к основной заработной плате:

$$\Pi_{zi} = \frac{3_{oi} \cdot H_{пз}}{100}, \quad (9.17)$$

где $H_{пз}$ – норматив прочих затрат в целом по научной организации.

$$\Pi_{zi} = \frac{180,544 \cdot 20}{100} = 36,11 \text{ руб.}$$

9.1.14 Расчет расходов по статье «Накладные расходы»

Затраты по статье «Накладные расходы» (P_{hi}), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды (P_{hi}), относятся на конкретное ПС по нормативу ($H_{рн}$) в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по научной организации:

$$P_{hi} = \frac{3_{oi} \cdot H_{рн}}{100}, \quad (9.18)$$

где P_{hi} – накладные расходы на конкретное ПС, руб.;

$H_{рн}$ – норматив накладных расходов в целом по научной организации.

$$P_{hi} = \frac{180,544 \cdot 100}{100} = 180,54 \text{ руб.}$$

9.1.15 Расчет общей суммы расходов на разработку

Общая сумма расходов по всем статьям сметы (C_{pi}) на ПС рассчитывается по формуле:

$$C_{pi} = 3_{oi} + 3_{di} + 3_{czi} + H_{ei} + M_i + P_{ci} + P_{mi} + P_{hki} + \Pi_{zi} + P_{hi}. \quad (9.19)$$

Кроме того, организация-разработчик осуществляет затраты на сопровождение и адаптацию ПС (P_{CAi}), которые определяются по нормативу (H_{PCA})

$$C_{pi} = 180,54 + 28,89 + 73,3 + 8,377 + 9,07 + 84,13 + 5731,2 + 54,16 + 36,11 + 180,544 = 6386,33 \text{ руб.}$$

$$P_{cai} = \frac{C_{pi} \cdot H_{pca}}{100}, \quad (9.20)$$

где H_{pca} – норматив расходов на сопровождение и адаптацию, %.

$$P_{cai} = \frac{6\,386,33 \cdot 4}{100} = 255,45 \text{ руб.}$$

Общая сумма расходов на разработку (с затратами на сопровождение и адаптацию) как полная себестоимость ПС (C_{Pi}) определяется по формуле:

$$C_{Pi} = C_{pi} + P_{cai}. \quad (9.21)$$

$$C_{Pi} = 6\,386,33 + 255,45 = 6\,641,78 \text{ руб.}$$

9.1.16 Расчет прибыли и отпускной цены

Рентабельность и прибыль по создаваемому ПС определяются исходя из результатов анализа рыночных условий, переговоров с заказчиком (потребителем) и согласования с ним отпускной цены, включающей дополнительно налог на добавленную стоимость и отчисления на содержание ведомственного жилого фонда. Прибыль рассчитывается по формуле:

$$P_{pci} = \frac{C_{Pi} \cdot Y_{pni}}{100}, \quad (9.22)$$

где P_{pci} – прибыль от реализации ПС, руб.;

Y_{pni} – уровень рентабельности ПС, %;

C_{Pi} – себестоимость ПС, руб.

$$P_{pci} = \frac{6\,641,78 \cdot 53}{100} = 3\,520,14 \text{ руб.}$$

Прогнозируемая цена ПС без налогов ($\Pi_{\text{пi}}$):

$$\Pi_{\text{пi}} = C_{\text{пi}} + \Pi_{\text{ci}}. \quad (9.23)$$

$$\Pi_{\text{пi}} = 6\,641,78 + 3\,520,14 = 10\,161,92 \text{ руб.}$$

Отчисления и налоги в местный и республиканский бюджеты единым платежом ($O_{\text{мрi}}$):

$$O_{\text{мрi}} = \frac{\Pi_{\text{пi}} \cdot H_{\text{мр}}}{100\%}, \quad (9.24)$$

$$O_{\text{мрi}} = \frac{(10\,161,92 \cdot 3,9)}{100} = 396,31 \text{ руб.}$$

где $H_{\text{мр}}$ – норматив отчислений в местный и республиканский бюджеты (%).

Налог на добавленную стоимость (НДС_i):

$$\text{НДС}_i = \frac{(\Pi_{\text{пi}} + O_{\text{мр}}) \cdot H_{\text{дс}}}{100\%}, \quad (9.25)$$

где $H_{\text{дс}}$ – норматив НДС, %.

$$\text{НДС}_i = \frac{(10\,161,92 + 396,31) \cdot 20}{100} = 2\,111,65 \text{ руб.}$$

Прогнозируемая отпускная цена (Π_{oi}):

$$\Pi_{\text{oi}} = \Pi_{\text{пi}} + O_{\text{мрi}} + \text{НДС}_i. \quad (9.26)$$

$$\Pi_{\text{oi}} = 10\,161,92 + 396,31 + 2\,111,65 = 12\,669,89 \text{ руб.}$$

9.2 Расчет экономического эффекта от применения ПС у пользователя

9.2.1 Исходные данные

Исходные данные для расчета экономического эффекта от применения разрабатываемого программного средства у пользователя сведены в таблице 9.3.

Таблица 9.3 – Исходные данные для расчета экономического эффекта

Наименование показателей	Обозначения	Единицы измерения	Значение показателя	
			в базовом варианте	в новом варианте
1. Средняя трудоемкость работ в расчете на 100 КБ	T_{c1} T_{c2}	человеко-час на 100 КБ	1,8	1,7
2.Средний расход машинного времени в расчете на 100 КБ	$M_{в1}$ $M_{в2}$	машино-час на 100 КБ	9,0	8,91
3.Средний расход материалов в расчете на 100 КБ	$M_{т1}$ $M_{т2}$	руб. на 100 КБ	0,032	0,028
4. Количество типовых задач, решаемых за год	$Z_{т2}$	задача	9	
5. Ставка налога на прибыль	$H_{п}$	%	18	

9.2.2 Расчет объема работ

Объем работ в зависимости от функциональной группы и назначения ПС можно определить по формуле:

$$A = V_{пс} \cdot K_{пс}, \quad (9.27)$$

где $V_{пс}$ – объем ПС в натуральных единицах измерения;

$K_{пс}$ – коэффициент применения ПС.

$$A = 23\,880 \cdot 0,5 = 11\,940.$$

9.2.3 Расчет капитальных затрат

Общие капитальные вложения (K_o) заказчика (потребителя), связанные с приобретением, внедрением и использованием ПС, рассчитываются по формуле:

$$K_o = K_{пр} + K_{ос} + K_{тс} + K_{об}, \quad (9.28)$$

где $K_{пр}$ – затраты пользователя на приобретение ПС по отпускной цене разработчика с учетом стоимости услуг по эксплуатации и сопровождению, руб.;

$K_{ос}$ – затраты пользователя на освоение ПС, руб.;

$K_{тс}$ – затраты на доукомплектацию ВТ техническими средствами в связи с

внедрением нового ПС, руб.;

$K_{об}$ – затраты на пополнение оборотных средств в связи с использованием нового ПС, руб.

$$K_o = 12\,669,89 + 126,7 + 0 + 126,7 = 12923,29 \text{ руб.}$$

Затраты на освоение ПС и на пополнение оборотных средств по формулам:

$$K_{ос} = K_{пр} * H_{кос}, \quad (9.29)$$

где $H_{кос}$ – норматив затрат пользователя на освоение ПС, равный 0,01.

$$K_{ос} = 12\,669,89 * 0,01 = 126,7 \text{ руб.}$$

$$K_{об} = K_{пр} * H_{коб}, \quad (9.30)$$

где $H_{коб}$ – норматив затрат на пополнение оборотных средств в связи с использованием нового ПС, равный 0,01.

$$K_{об} = 12\,669,89 * 0,01 = 126,7 \text{ руб.}$$

9.2.4 Расчет экономии основных видов ресурсов в связи с использованием нового программного средства

Экономия затрат на заработную плату при использовании нового ПС в расчете на объем выполненных работ:

$$C_3 = C_{зе} * A_2, \quad (9.31)$$

где $C_{зе}$ – экономия затрат на заработную плату при решении задач с использованием нового ПС в расчете на 100 КБ, руб.;

A_2 – объем выполненных работ с использованием нового ПС (100 КБ).

$$C_3 = 0,06787 * 107\,460 = 7\,293,70 \text{ руб.}$$

Экономия затрат на заработную плату в расчете на 100 КБ ($C_{зе}$):

$$C_{зе} = \frac{3_{см} \cdot (T_{c1} - T_{c2}) : T_{ч}}{D_p}, \quad (9.32)$$

где $Z_{см}$ – среднемесячная заработная плата одного программиста, руб.;

$T_{с1}$, $T_{с2}$ – снижение трудоемкости работ в расчете на 100 строк кода (человеко-часов);

$T_{ч}$ – количество часов работы в день (ч);

D_p – среднемесячное количество рабочих дней.

$$C_{зе} = \frac{90,3 \cdot (1,8 - 1,7)}{133} = 0,06787 \text{ руб.}$$

Объем выполненных работ с использованием нового ПС (100 КБ):

$$A_2 = A_o \cdot Z_{т2}, \quad (9.33)$$

где A_o – объем работ необходимый для решения одной задачи (100 КБ);

$Z_{т2}$ – количество типовых задач, решаемых за год (задач).

$$A_2 = 11\,940 \cdot 9 = 107\,460.$$

Экономия затрат на оплату машинного времени (C_m) в расчете на выполненный объем работ в результате применения нового ПС:

$$C_m = C_{ме} \cdot A_2, \quad (9.34)$$

где $C_{ме}$ – экономия затрат на оплату машинного времени при решении задач с использованием нового ПС в расчете на 100 КБ.

$$C_m = 0,18 \cdot 107\,460 = 19\,342,8 \text{ руб.}$$

Экономия затрат на оплату машинного времени в расчете на 100 КБ ($C_{ме}$):

$$C_{ме} = \Pi_m \cdot (M_{в1} - M_{в2}), \quad (9.35)$$

где Π_m – цена одного машино-часа работы ЭВМ;

$M_{в1}$, $M_{в2}$ – средний расход машинного времени в расчете на 100 КБ при применении соответственно базового и нового ПС.

$$C_{ме} = 2 \cdot (9 - 8,91) = 0,18 \text{ руб.}$$

Экономия затрат на материалы ($C_{мт}$) при использовании нового ПС в расчете на

объем выполненных работ:

$$C_{MT} = C_{MTe} \cdot A_2, \quad (9.36)$$

где C_{MTe} – экономия затрат на материалы в расчете на 100 КБ при использовании нового ПС.

$$\begin{aligned} C_{MT} &= 0,004 \cdot 107\,460 = 429,84 \text{ руб.} \\ C_{MTe} &= C_{M1} - C_{M2}, \end{aligned} \quad (9.37)$$

где C_{M1} , C_{M2} – средний расход материалов у пользователя в расчете на 100 КБ при использовании соответственно базового и нового ПС, руб.

$$C_{MTe} = 0,032 - 0,028 = 0,004 \text{ руб.}$$

Общая годовая экономия текущих затрат, связанных с использованием нового ПС (C_o):

$$C_o = C_3 + C_{oz} + C_M + C_{MT}. \quad (9.38)$$

$$C_o = 7\,293,7 + 19\,342,8 + 429,84 = 27\,066,35 \text{ руб.}$$

9.2.5 Расчет экономического эффекта

Экономический эффект – разность между результатами деятельности хозяйствующего субъекта и произведенными для их получения затратами на изменения условий деятельности. Различают положительный и отрицательный экономический эффект. Положительный экономический эффект достигается в случае, когда результаты деятельности предприятия превышают затраты. Если затраты превышают результаты, имеет место отрицательный экономический эффект, то есть убыток.

Внедрение нового ПС позволит пользователю сэкономить на текущих затратах. Для пользователя в качестве экономического эффекта выступает лишь чистая прибыль – дополнительная прибыль, остающаяся в его распоряжении ($\Delta\P_{\text{ч}}$), которые определяются по формуле:

$$\Delta\P_{\text{ч}} = C_o - \frac{C_o \cdot H_{\text{п}}}{100}, \quad (9.39)$$

где H_{π} – ставка налога на прибыль (%).

$$\Delta\Pi_{\text{ч}} = 27\,066,35 - \frac{27\,066,35 * 18}{100} = 22\,194,4 \text{ руб.}$$

В процессе использования нового программного средства чистая прибыль в конечном итоге возмещает капитальные затраты.

Норматив приведения разновременных затрат и результатов (E_n) для программных средств ВТ в существующей практике принимается в пределах 0,2–0,4

Полученные при этом суммы результатов (прибыли) и затрат (капитальных вложений) по годам приводят к единому времени. Единое время соответствует расчетному году (за расчетный год принят текущий год разработки программного средства 2019 год) путем умножения результатов и затрат за каждый год на коэффициент приведения ($ALFA_t$), который рассчитывается по формуле:

$$ALFA_t = (1 + E_n)^{t_p - t}, \quad (9.40)$$

где E_n – норматив приведения разновременных затрат и результатов;

t_p – расчетный год, $t_p=1$;

t – номер года, результаты и затраты которого приводятся к расчетному (2019-1, 2020-2, 2021-3, 2022-4).

Норматив приведения разновременных затрат и результатов (E_n) для программных средств ВТ в существующей практике принимается в пределах 0,2–0,4.

Например, при нормативе 0,4 коэффициентам приведения ($ALFA_t$) по годам будут соответствовать следующие значения:

$$ALFA_1 = (1 + 0,4)^{1-1} = 1 - \text{расчетный год};$$

$$ALFA_2 = (1 + 0,4)^{1-2} = 0,714 - 2020 \text{ год};$$

$$ALFA_3 = (1 + 0,4)^{1-3} = 0,510 - 2021 \text{ год};$$

$$ALFA_4 = (1 + 0,4)^{1-4} = 0,364 - 2022 \text{ год}.$$

Результаты расчета экономического эффекта от реализации программного средства сведены в таблицу 9.4. В которой можно наглядно определить положительный (деятельности предприятия превышают затраты), либо отрицательный (убыток) ли эффект предполагается от разрабатываемого программного средства.

Таблица 9.4 – Данные экономического эффекта от использования нового ПС

Показатели	Ед. измерения	2019	2020	2021	2022
Результаты:					
Прирост прибыли за счет экономии затрат (П _ч)	руб.	22 194,4	22 194,4	22 194,4	22 194,4
То же с учетом фактора времени	руб.	22 194,4	15 846,8	11 319,15	8 078,76
Затраты:					
Приобретение, адаптация и освоение ПС (К _{пр})	руб.	12 669,89	-	-	-
Освоение ПС (К _{ос})	руб.	126,7	-	-	-
Доукомплектование ВТ техническими средствами (К _{тс})	руб.	-	-	-	-
Пополнение оборотных средств (К _{об})	руб.	126,7	126,7	126,7	126,7
Всего затрат	руб.	12 923,29	126,7	126,7	126,7
То же с учетом фактора времени	руб.	12 923,29	90,46	64,62	46,12
Экономический эффект:					
Превышение результата над затратами	руб.	9 271,11	15 756,34	11 254,53	8 032,64
То же с нарастающим итогом	руб.	9 271,11	25 027,45	36 281,98	44 314,63
Коэффициент приведения	единиц	1	0,714	0,510	0,364

Таким образом, отпускная цена предлагаемой разработки составляет 12 700 рублей, окупаемость ее будет достигнута уже на первом году применения. При этом эффект от использования в течение четырех лет составит 44 315 рублей.

10 ОХРАНА ТРУДА

10.1 Производственная санитария и техника безопасности

Работающие с ПЭВМ могут подвергаться воздействию различных опасных и вредных производственных факторов, основными из которых являются повышенные уровни: электромагнитного, рентгеновского, ультрафиолетового и инфракрасного излучения; статического электричества; запыленности воздуха рабочей зоны; повышенный или пониженный уровень освещенности рабочей зоны, содержание в воздухе рабочей зоны оксида углерода, озона, аммиака, фенола; напряжение зрения, памяти, внимания; длительное статическое напряжение; большой объем информации, обрабатываемой в единицу времени; монотонность труда.

Работа с ПЭВМ проводится в соответствии с Санитарными нормами и правилами «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами» и Гигиеническим нормативом «Предельно-допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами», утвержденными постановлением Министерства здравоохранения от 28.06.2013 г. № 59 и Типовой инструкцией по охране труда при работе с персональными ЭВМ [9].

Согласно вышеуказанных документов площадь помещения на одного пользователя ПЭВМ на базе плоских дискретных экранов (жидкокристаллические, плазменные) составляет не менее 4,5 м².

10.1.1 Метеоусловия

В производственных помещениях, в которых работа с использованием ПЭВМ является основной обеспечиваются оптимальные параметры микроклимата для категории работ 1а и 1б (табл. 10.1) согласно вышеуказанным нормативам.

Таблица 10.1 – Оптимальные параметры микроклимата для помещений с ВДТ, ЭВМ и ПЭВМ

Период года	Категория работ	Температура воздуха, °С, не более	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	легкая-1а	22-24	40-60	0,1
	легкая-1б	21-23	40-60	0,1
Теплый	легкая-1а	23-25	40-60	0,1
	легкая-1б	22-24	40-60	0,2

Работа с компьютером относится к категории 1а (работы, производимые сидя и сопровождающиеся незначительным физическим напряжением, при которых расход энергии составляет до 120 ккал/ч, т.е. до 139 Вт).

Интенсивность теплового излучения работающих от нагретых поверхностей технологического оборудования, осветительных приборов, инсоляции на постоянных рабочих местах не превышает значений, указанных в табл. 10.2.

Таблица 10.2 – Предельно допустимые уровни интенсивности излучения в инфракрасном и видимом диапазоне излучения на расстоянии 0,5 м со стороны экрана ВДТ, ЭВМ и ПЭВМ

Диапазоны длин волн	400-760 нм	760-1050 нм	свыше 1050 нм
Предельно допустимые уровни	0,1 Вт/м ²	0,05 Вт/м ²	4,0 Вт/м ²

Для создания нормальных метеорологических условий наиболее целесообразно уменьшить тепловыделения от самого источника – монитора, что предусматривается при разработке его конструкции.

В производственных помещениях для обеспечения необходимых показателей микроклимата предусмотрены системы отопления, вентиляции воздуха.

10.1.2 Вентиляция и отопление

Воздух рабочей зоны помещения соответствует санитарно-гигиеническим требованиям по содержанию вредных веществ и частиц пыли, приведенным в Санитарных нормах и правилах «Требованию к контролю воздуха рабочей зоны», Гигиеническом нормативе.

В помещениях, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы.

Уровни положительных и отрицательных аэроионов, коэффициент униполярности в воздухе, соответствуют значениям, указанным в табл. 10.3.

Таблица 10.3 – Уровни ионизации и коэффициент униполярности воздуха помещений при работе с ВДТ, ЭВМ и ПЭВМ

Уровни	Число ионов в 1 см ³ воздуха		Коэффициент униполярности (У)
	n+	n-	
Минимально допустимые	400	600	0,4 ≤ У < 1,0
Оптимальные	1500-3000	3000-5000	
Максимально допустимые	50000	50000	

Одним из мероприятий по оздоровлению воздушной среды является устройство вентиляции и отопления. Задачей вентиляции является обеспечение чистоты воздуха и параметров метеорологических условий на рабочих местах. Для поддержания нормального микроклимата необходим достаточный объем вентиляции, для чего в вычислительном центре предусматривается кондиционирование воздуха, осуществляющее поддержание постоянных параметров микроклимата в помещении независимо от наружных условий.

Параметры микроклимата поддерживаются в холодный период года за счет системы водяного отопления с нагревом воды до 100°C, а в теплый - за счет кондиционирования, с параметрами, отвечающими требованиям СНБ 4.02.01-03.

10.1.3 Освещение

В помещении для эксплуатации ПЭВМ предусмотрены естественное и искусственное освещение. Естественное освещение на рабочих местах осуществляется через световые проемы, ориентированные преимущественно на север, северо-восток, восток, запад или северо-запад и обеспечивает коэффициент естественной освещенности не ниже 1,5 %. Оконные проемы оборудованы регулируемыми устройствами типа жалюзи, занавесей.

Для внутренней отделки интерьера помещений используются материалы с коэффициентом отражения для потолка – 0,7- 0,8; для стен – 0,5- 0,6; для пола – 0,3- 0,5 [10].

Искусственное освещение в помещениях осуществляется системой общего равномерного освещения. При работе с документами применяется система комбинированного освещения, а освещенность поверхности стола в зоне размещения рабочего документа должна составлять 300-500 люкс. Освещенность поверхности экрана не более 300 люкс. В качестве источников света применяем люминесцентные лампы типа ЛБ. Коэффициент запаса для осветительных установок общего освещения принимается равным 1,4, а коэффициент пульсации – не более 5 %.

10.1.4 Шум

Основными источниками шума в помещениях, оборудованных ЭВМ, являются принтеры, множительная техника и оборудование для кондиционирования воздуха, в самих ЭВМ – вентиляторы систем охлаждения и трансформаторы.

В табл. 10.4 приведены нормированные уровни шума согласно Санитарных норм и правил «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами» и Гигиенических нормативов «Предельно-допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и

электронно-вычислительными машинами», которые обеспечиваются за счет использования малошумного оборудования, применения звукопоглощающих материалов для облицовки помещений, а также различных звукопоглощающих устройств (перегородки и т. п.).

Таблица 10.4 – Предельно-допустимые уровни звука, эквивалентные уровни звука и уровни звукового давления в октавных полосах частот при работе с ВДТ, ЭВМ и ПЭВМ и периферийными устройствами

Категория нормы шума	Уровни звукового давления, дБ, в октавных полосах со среднегеометрическими частотами, Гц									Уровни звука и эквивалентные уровни звука, дБА
	31.5	63	125	250	500	1000	2000	4000	8000	
I	86	71	61	54	49	45	42	40	38	50
II	93	79	70	63	58	55	52	50	49	60
III	96	83	74	68	63	60	57	55	54	65
IV	103	91	83	77	73	70	68	66	64	75

10.1.5 Электробезопасность

Помещение вычислительного центра по степени опасности поражения электрическим током относится к помещениям без повышенной опасности.

Основные меры защиты от поражения током:

- 1) изоляция и недоступность токоведущих частей;
- 2) защитное заземление ($R_3 = 4 \text{ Ом}$ ГОСТ 12.1.030 - 81).

Первая помощь при поражениях электрическим током состоит из двух этапов: освобождение пострадавшего от действия тока и оказание ему доврачебной медицинской помощи. После освобождения пострадавшего от действия электрического тока необходимо оценить его состояние. Во всех случаях поражения электрическим током необходимо вызвать врача независимо от состояния пострадавшего.

10.1.6 Излучение

При работе с дисплеем могут возникать следующие опасные факторы: электромагнитные и электростатические поля, ультрафиолетовое и инфракрасное излучение.

Уровни физических факторов на рабочих местах пользователей, создаваемые ПЭВМ и периферийными устройствами, не превышают предельно-допустимые уровни: электромагнитных и электростатических полей (табл. 10.5, 10.6),

ультрафиолетового (табл. 10.7), установленных Гигиеническим нормативом «Предельно допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами».

Таблица 10.5 – Предельно допустимые уровни электромагнитных полей от экранов ВДТ, ЭВМ и ПЭВМ

Наименование параметра	Предельно-допустимые уровни
Напряженность электрического поля в диапазоне частот: 5 Гц-2 кГц 2-400 кГц	не более 25,0 В/м не более 2,5 В/м
Плотность магнитного потока магнитного поля в диапазоне частот: 5 Гц-2 кГц 2-400 кГц	не более 250 нТл не более 25 нТл
Напряженность электростатического поля	не более 15 кВ/м

Таблица 10.6 – Предельно допустимые уровни электромагнитных полей при работе с ВДТ, ЭВМ, ПЭВМ от клавиатуры, системного блока, манипулятора «мышь», беспроводных системам передачи информации и иных периферийных устройств

Диапазоны частот	0,3-300 кГц	0,3-3 МГц	3-30 МГц	30-300 МГц	0,3-300 ГГц
Предельно допустимые уровни	25 В/м	15 В/м	10 В/м	3 В/м	10 мкВт/см ²

Таблица 10.7 – Предельно допустимые уровни интенсивности излучения в ультрафиолетовом диапазоне на расстоянии 0,5 м со стороны экрана ВДТ, ЭВМ и ПЭВМ

Диапазоны длин волн	200-280 нм	280-315 нм	315-400 нм
Предельно допустимые уровни	не допускается	0,0001 Вт/м ²	0,1 Вт/м ²

Наиболее эффективным и часто применяемым методом защиты от электромагнитных излучений является установка экранов. Экранируют либо источник излучения, либо рабочее место. Часто экран устанавливают непосредственно на монитор.

При работе монитора на экране кинескопа накапливается электростатический заряд, создающий электростатическое поле. При этом персонал, работающий с

монитором, приобретают электростатический потенциал. Общее электростатическое поле создает электризующиеся от трения поверхности клавиатуры и мыши.

10.1.7 Пожарная безопасность

По взрывопожарной и пожарной опасности помещения и здания для ЭВМ относятся к категории Д согласно ТКП 474-2013. Здания для ВЦ и части зданий другого назначения, в которых предусмотрено размещение ЭВМ, относятся к 2 степени огнестойкости согласно ТКП 45-2.02-315-2018.

Для предотвращения распространения огня во время пожара с одной части здания на другую устраивают противопожарные преграды в виде стен, перегородок, дверей, окон. Особое требование предъявляется к устройству и размещению кабельных коммуникаций.

Нормы первичных средств пожаротушения для вычислительных центров приведены в табл. 10.8.

Таблица 10.8 – Примерные нормы первичных средств пожаротушения для вычислительного центра

Помещение	Площадь, м ²	Углекислотные огнетушители ручные	Порошковые огнетушители
Вычислительный центр	100	1	1

Для ликвидации пожаров в начальной стадии применяются первичные средства пожаротушения: внутренние пожарные водопроводы, огнетушители типа ОВП-10, ОУ-2, асбестовые одеяла и др.

Эвакуация персонала вычислительного центра осуществляется через эвакуационные выходы [11]. Количество и общая ширина эвакуационных выходов определяются в зависимости от максимального возможного числа эвакуирующихся людей и допустимого расстояния от наиболее удаленного места пребывания людей до ближайшего эвакуационного выхода согласно ТКП 45-2.02-315-2018.

Расчетное время эвакуации устанавливается по реальному расчету времени движения одного или нескольких потоков людей через эвакуационные выходы из наиболее удаленных мест размещения людей. Необходимое время эвакуации устанавливается на основе данных о критической продолжительности пожара с учетом степени огнестойкости здания, категории производства по взрывной и пожарной опасности. Для успешной эвакуации необходимо, чтобы расчетное время было меньше необходимого.

10.2 Требования к ВДТ, ЭВМ, ПЭВМ и периферийным устройствам

Уровни физических факторов (уровни электромагнитных и электростатических полей, уровни вибрации, уровни ультрафиолетового, инфракрасного, видимого и мягкого рентгеновского излучений), создаваемые ПЭВМ и периферийными устройствами, не превышают предельно-допустимые уровни, установленных Гигиеническим нормативом «Предельно-допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами».

Уровни шума, создаваемые ПЭВМ и периферийными устройствами, не превышают предельно допустимых уровней, предусмотренных Гигиеническим нормативом, и устанавливаются в зависимости от следующих категорий производимых работ (табл. 10.4):

1) категория I – выполнение основной работы в залах вычислительной техники, а также в помещениях с ПЭВМ всех типов учреждений образования;

2) категория II – выполнение работы с ПЭВМ в помещениях, где работают инженерно-технические работники, осуществляющие лабораторный, аналитический или измерительный контроль;

3) категория III – выполнение работы в помещениях операторов ЭВМ (без дисплеев);

4) категория IV – выполнение работы с ПЭВМ в помещениях для размещения шумных агрегатов.

В производственных помещениях, в которых работа с ПЭВМ является вспомогательной, уровни шума на рабочих местах не превышают значений, установленных для видов трудовой деятельности, осуществляемых в этих помещениях, в соответствии с Санитарными нормами и правилами, устанавливающими ПДУ шума на рабочих местах, в помещениях жилых и общественных зданий.

Допустимые визуальные эргономические параметры устройств отображения ВДТ, ЭВМ и ПЭВМ соответствуют допустимым значениям, установленным Гигиеническим нормативом.

Конструкция ЭВМ и ПЭВМ, дизайн и совокупность эргономических параметров обеспечивают надежное и комфортное считывание отображаемой информации в условиях эксплуатации. Конструкция оборудования ЭВМ и ПЭВМ обеспечивает возможность поворота корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана.

Дизайн ВДТ, ЭВМ, ПЭВМ и периферийных устройств должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света.

ЗАКЛЮЧЕНИЕ

В процессе выполнения дипломного проекта был придуман веб-дизайн и разработана клиентская часть веб-приложения для удобного и эффективного управления программными проектами, с учетом всех требований, описанных в постановке задачи.

В ходе проектирования и разработки интерфейса веб-приложения были проанализированы существующие аналоги. С учетом их достоинств и недостатков, а также с учетом специфики предметной области, были выбраны подходящие и новейшие технологии и инструменты, позволяющие решить поставленную задачу с наилучшими показателями качества, производительности и удобства использования.

Преимуществами созданного пользовательского интерфейса веб-приложения являются:

- 1) доступность на любом устройстве и любой операционной системе;
- 2) лаконичный и интуитивно понятный интерфейс;
- 3) создание проектов в режиме администратора;
- 4) контроль потраченного времени и ведение журнала работ;
- 5) возможность коммуникации между участниками команды;
- 6) охват всех необходимых потребностей для управления проектами;
- 7) загрузка документации для проекта;
- 8) возможность просмотра информации по текущей команде и проектам;
- 9) безопасность – сокрытие данных от неавторизованных пользователей.

Дизайн разработан с учетом основной аудитории веб-приложения и специфики их работы. Использована современная дизайна-концепция Material.

Пользовательский интерфейс разработан с использованием основных принципов программирования (ООП и компонентного подхода). А также имеет высокий уровень удобства использования, простоты вхождения для новых пользователей, а также покрывает все необходимые требования, касающиеся данного типа приложений.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- 1 API [Электронный ресурс]. – Режим доступа: <https://tproger.ru/articles/web-api/>;
- 2 Протокол HTTP [Электронный ресурс]. – <https://developer.mozilla.org/ru/docs/Web/HTTP>;
- 3 Дэвид Флэнаган, «JavaScript. подробное руководство, 6-е издание», 2012 г.;
- 4 Стоян Стефанов «React.js Быстрый старт», 2017г.;
- 5 А. Бенкс, Е. Порселло, «React и Redux. Функциональная веб-разработка», 2018г.;
- 6 Руководство по React [Электронный ресурс]. – Режим доступа: <https://metanit.com/web/react/>;
- 7 Руководство по React Router [Электронный ресурс]. – Режим доступа: <https://reacttraining.com/react-router/core/guides/quick-start>;
- 8 Джон Дакет, «HTML и CSS. Разработка и дизайн веб-сайтов»;
- 9 Санитарные нормы и правила «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами» и Гигиенический норматив «Предельно-допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами», утвержденные постановлением МЗ РБ от 28.06.2013 г. № 59;
- 10 Лазаренков, А. М. Охрана труда в машиностроении: учебное пособие / А. М. Лазаренков. – Минск: ИВЦ Минфина, 2017. – 446 с;
- 11 Лазаренков А.М., Ушакова И.Н. Охрана труда: Учебно-методическое пособие для практических занятий. – Мн.: БНТУ, 2011. – 205 с.

