Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Кафедра прикладной информатики

Утверждена распоряжением по институту
от _____ № _____

Допущена к защите
«_____» _____ 20___ г.
Зав. кафедрой прикладной информатики

к.э.н., доцент Азаров Иван Валерьевич

(уч. степень, уч звание, ФИО зав. каф.)

(подпись зав. кафедрой)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Исследование и разработка системы управления проектами
(название работы)

Рецензенты:
Костюков Константин Иванович
Ротачев Глеб Александрович
(ФИО)

к.э.н., заведующий кафедрой экономики,
управления, финансового права и
информационных технологий

(ученая степень, звание, должность)

Выполнил:
(ФИО)

студент 2 курса, группы ПИН-м-о-18-2
направле
«Прикладная информатика», очной формы обучения
Ставропольского филиала Московского

(подпись)

педагогического государственного университета

Нормоконтролер:

Руководитель:

Тарасевич Павел Петрович
(ФИО)

ассистент кафедры прикладной информатики
(ученая степень, звание, должность)

Орлинская Оксана Геннадьевна
(ФИО)

к.э.н., доцент, доцент кафедры прикладной информатики
(ученая степень, звание, должность)

Дата защиты
«__» _____ 20___ г.

Оценка _____

_____
(подпись)

Ставрополь, 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Кафедра прикладной информатики
Направление 09.04.03 «Прикладная информатика»
Направленность (профиль) «Математическое и информационное обеспечение экономической деятельности»

«УТВЕРЖДАЮ»
Зав. кафедрой
_____ Азаров И.В.
подпись, инициалы, фамилия
«_____»_____20___г

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ (ДИПЛОМНУЮ РАБОТУ)

Студент Ротачев Глеб Александрович группа ПИН-м-о-18-2
фамилия, имя, отчество
1.Тема Исследование и разработка системы управления проектами
Утверждена распоряжением по институту №_____ от «___»_____20___ г.
2.Срок представления работы к защите «____»_____20___г.
3. Исходные данные для исследования научные статьи, учебные и учебно-методические пособия, периодические издания, материалы преддипломной практики
4. Содержание дипломной работы:
4.1 Аналитический раздел
4.2 Исследовательский раздел
4.3 Проектный раздел
4.4 Организационно-экономический раздел
4.5 Раздел безопасности и экологичности выпускной квалификационной работы
Дата выдачи задания
Руководитель дипломной работы _____ О.Г. Орлинская
подпись *инициалы, фамилия*

Консультанты по разделам
Аналитический раздел _____ И.В. Азаров
подпись *инициалы, фамилия*
Исследовательский раздел _____ Н.В. Кононова
подпись *инициалы, фамилия*
Проектный раздел _____ Н.В. Кононова

подпись *инициалы, фамилия*

Организационно-экономический раздел_____О.Г. Орлинская_____

подпись *инициалы, фамилия*

Раздел безопасности и экологичности ВКР_____С.Р. Амироков

подпись *инициалы, фамилия*

Нормоконтроль_____П.П. Тарасевич_____

подпись *инициалы, фамилия*

Задание к исполнению принял

«___»_____20__г._____

подпись

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Кафедра прикладной информатики
Направление 09.04.03 «Прикладная информатика»
Направленность (профиль) «Математическое и информационное обеспечение экономической деятельности»

КАЛЕНДАРНЫЙ ПЛАН

Фамилия, имя, отчество (полностью) Ротачев
Тема ВКР Исследование и разработка системы управления проектами
Руководитель Орлинская Оксана Геннадьевна
Консультанты: Орлинская Оксана Геннадьевна, Азаров Иван Валерьевич, Кононова Наталия Владимировна, Амироков Станислав Рауфович,

№	Наименование этапов выполнения выпускной квалификационной работы	Срок выполнения работы	Примечание
1.	1. Выдача задания	30.03.2020	
2.	2. Начало проектирования	30.03.2020	
3.	3. Разделы ВКР	30.03.2020-25.06.2020	
4.	3.1. Аналитический раздел	30.03.2020-11.04.2020	
5.	3.2 Исследовательский раздел	13.04.2020-09.05.2020	
6.	3.3. Проектный раздел	11.05.2020-23.05.20120	
7.	3.4. Экономический раздел	25.05.2020-30.05.2020	
8.	3.5 Безопасность и экологичность проекта	01.06.2020-06.06.2020	
9.	4. Предзащита	08.06.2020-13.06.2020	
10.	5. Рецензирование, нормоконтроль, проверка в системе антиплагиат	15.06.2020-20.06.2020	
11.	6. Ознакомление с отзывом и рецензией	22.06.2020-25.06.2020	
12.	6.Сдача ВКР, отзыва и рецензии в ГЭК	25.06.2020-27.06.2020	
13.	7. Защита в ГЭК	30.06.2020	

Научный руководитель _____О.Г. Орлинская_____

подпись, Ф.И.О.

Зав. кафедрой_____И.В. Азаров

подпись, Ф.И.О.

«___»_____20____г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С каждым годом информационные технологии играют все большую роль в жизни людей и все больше компаний обращают пристальное внимание, внедряя принципы проектного управления в работу как отдельных подразделений, так и организации в целом. Именно эффективное управление проектами позволяет компании грамотно планировать и успешно реализовывать проекты, оптимизируя затраты временных, денежных и человеческих ресурсов, но при этом не отклоняясь от запланированного качества конечного продукта проекта. Использование принципов и методов управления проектами позволяет организации достигать в бизнесе новых конкурентных преимуществ и повышать результативность своей деятельности. Для каждого предприятия использование информационных технологий в деятельности организации может быть разнообразно, а так или иначе, организации любого масштаба и любой сферы деятельности стараются внедрять информационные технологии в своей деятельности и в полной мере использовать преимущества, которые они получают при их использовании. Так же с развитием информационных технологий веб-приложения обрели сложную архитектуру и их уже не под силу разрабатывать одному человеку. К процессу подключается команда специалистов, каждый из которых отвечает за определенный «участок» разработки. Неотъемлемой частью команды является back-end разработчик – специалист, который отвечает за программирование серверной части веб-приложения.

Актуальность данной темы состоит в том, что система управления проектами (СУП) позволит компании грамотно планировать и успешно реализовывать проекты, оптимизируя затраты временных, денежных и человеческих ресурсов, но при этом не отклоняясь от запланированного качества конечного продукта проекта.

Целью дипломной работы является теоретико-методологические и организационно-методологические положения контроля и анализа эффективности работы сотрудников организации с помощью системы управления проектами для повышения эффективности деятельности данной компании.

Задачами дипломной работы являются:

- выявить проблему ведения системы управления проектами в организации;
- систематизировать и централизовать работу над проектами;
- раскрыть понятия и требования к созданию системы управления проектами;
- описать состояние работы предприятия ООО «МИРТЕК» и его отрасли в целом;
- описать модуль деятельности ООО «МИРТЕК»
- рассмотреть критерии для оценки качества условий оказания услуг, на основе которых будут проводиться вычисления;
- описать методы оценки экономической эффективности;
- спроектировать структуру и разработать систему управления проектами.

В качестве объекта исследования в данной работе выступает предприятие ООО «МИРТЕК», а предметом исследования является внедрение системы управления проектами.

Практическая значимость разработанного модуля в том, что он может применяться не только в данной организации, но при его модернизации, интегрироваться в любую другую информационную систему, что показывает его универсальность.

Структура выпускной квалификационной работы. Работа состоит из введения, пяти разделов, заключения и списка использованных источников.

1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ

1.1 Постановка задачи исследования

Разработка системы управления проектами (СУП) является важной проблемой для руководителя и сотрудников предприятия ООО «МИРТЕК», которые хотят контролировать процесс работы.

Исследуя и разрабатывая программно-административную часть, а так же внутренние содержание системы и серверную часть СУП повлияет на упрощение работы руководства, отдела разработки и тестирования.

Разрабатываемая система управления проектами поможет централизовать работу отделов, вести контроль проектов, упростить хранение документации и ускорить время разработки.

Основными целями и задачами исследования является обеспечение контроля проектов по срокам и организация оперативной обратной связи между отделами.

1.2 Анализ организационной структуры ООО «МИРТЕК»

Объектом исследования является организация ООО «МИРТЕК», основанное в 2012 году. Основным видом деятельности которого является производство и реализация многофункциональных однофазных и трёхфазных приборов учёта (электроэнергии, воды, тепла и газоснабжения). Организация осуществляет полный производственный цикл: сборку, программирование, поверку и упаковку.

Организационная структура предприятия представлена на рисунке 1.1, где высшим органом управления предприятия является Генеральный директор. Заместители генерального директора контролируют организационную структуру, которая состоит из восьми подразделений: вспомогательного отдела, отдела продаж, отдела внешнеэкономической деятельности, отдела администрации, отдела технической поддержки, конструкторского отдела, отдела прикладного программного обеспечения, отдела Web-разработки и отдела бухгалтерии.

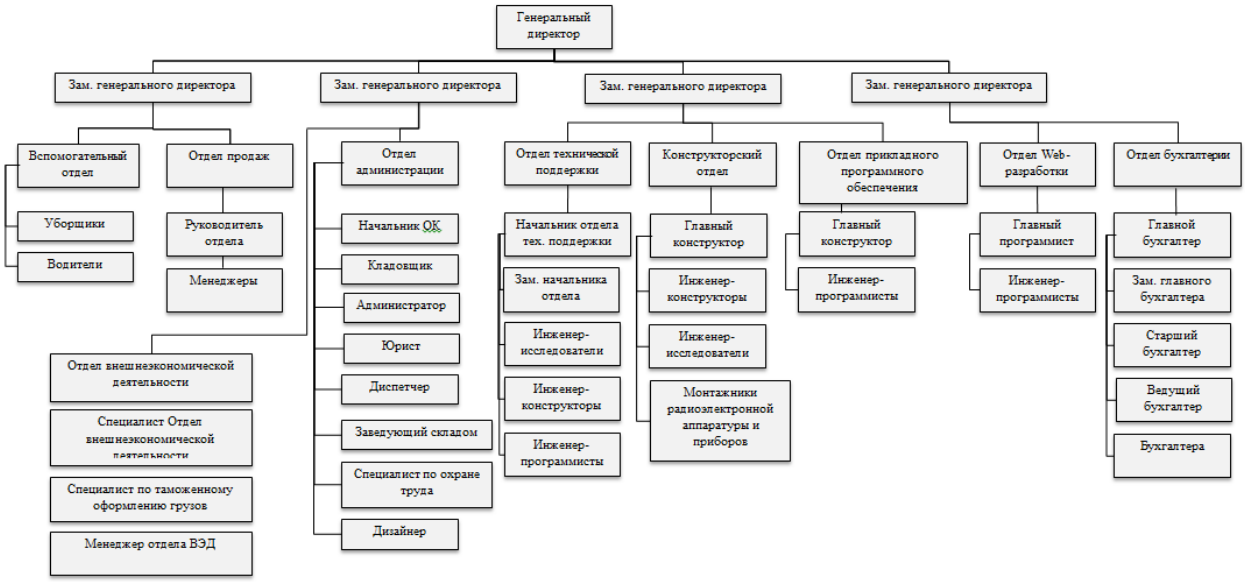


Рисунок 1.1 – Организационная структура организации ООО «МИРТЕК»

Каждый отдел представленной организационной структуры организации выполняет свою важную роль, а слаженная деятельность всех работников организации в каждом из отделов обеспечивает достижение целей, поставленных перед ООО «МИРТЕК». Предприятие располагает определенными материалами, которые

позволяют успешно функционировать и эффективно вести свою деятельность.

В условиях высокой рыночной конкуренции для руководителя организации необходимо иметь преимущество перед другими фирмами-конкурентами. Для этого предприятие должно работать бесперебойно, типовые задачи выполняться быстро и с минимальными затратами ресурсов. Достижение этих целей возможно при внедрении системы управления проектами [33].

В наше время многие руководители малых предприятий полагают, что внедрение систем автоматизации – дорогостоящие услуги, однако при анализе ресурсов компаний, приходят к выводу, что сроки окупаемости, а затем и получение выгоды говорят о необходимости внедрения систем управления проектами.

Основной целью использования вычислительной техники является снижение трудозатрат, улучшение качества и продуктивности работы.

На основании проведенного анализа и опросов руководителей и работников ООО «МИРТЕК», была выявлена основная проблема, такая как отсутствие возможности системы управления проектами.

Решением данной проблемы стала разработка системы управления проектами, которая содержала бы в себе весь необходимый функционал.

Результатом разработки является улучшение качества оказываемых услуг и повышение эффективности работы организации в целом.

1.3 Обзор научной литературы по теме исследования

В мире существует достаточно много стандартов по управлению проектами. Стандарт PMBOK, разработанный Институтом управления проектами (Project Management Institute - PMI), является одним из самых распространенных в России и США [6-9].

Степень научной разработанности темы, а так же основной вклад в развитие теоретических и методологических вопросов управления проектами внесли российские и зарубежные авторы: Д. Дипроуз, А.В. Цветкова, Ю.И. Попов, А.Н. Павлов, А.С. Товб, И. Кендалл [10-13].

Среди российских авторов большой вклад в развитие представлений об электронной коммерции внесли А.Н. Павлов и А.В. Цветкова, их труды содержат подробное описание каждого из процессов управления проектом, примеры реализации типовых задач.

Из зарубежных авторов можно выделить Д. Дипроуз, которая очень основательно опирается на стандарт PMBOK, что, в совокупности с многочисленными примерами использования проектного подхода, как в крупных корпорациях, так и в рядовых компаниях, может послужить отличным пособием для изучения данной темы, как в теоретическом направлении, так и для специалистов-практиков.

Изучение трудов российских и зарубежных авторов показало, что, несмотря, на то, что основы управления проектами начали формироваться еще в шестидесятых

годах прошлого века, на данный момент проектный подход еще не распространен повсеместно, однако его потенциал очень велик, он пригоден для использования в организациях любого рода деятельности с любым количеством сотрудников.

Методологической и информационной базой для написания дипломной работы является информация в работах отечественных и зарубежных авторов в области экономической теории, управления проектами, программирования, автоматизации деятельности компании и применении информационных систем.

1.4 Выбор метода анализа моделей

Системный анализ - это процесс изучения процедуры или бизнеса с целью определения его целей и задач и создания систем и процедур, которые будут эффективно их достигать. Другая точка зрения рассматривает системный анализ, как метод решения проблем, который разбивает систему на составляющие ее части с целью изучения того, насколько хорошо эти составляющие работают и взаимодействуют для достижения своей цели [14].

Область системного анализа тесно связана с анализом требований или операционным исследованием. Это также явное официальное расследование, проведенное, чтобы помочь лицу, принимающему решение, определить лучший курс действий и принять лучшее решение, чем он мог бы в противном случае принять.

Системный анализ используется в каждой области, где что-то разрабатывается. Анализ также может представлять

собой ряд компонентов, которые совместно выполняют органические функции, такие как системная инженерия. Системная инженерия - это междисциплинарная область проектирования, которая фокусируется на том, как сложные инженерные проекты должны разрабатываться и управляться.

Система - это группа взаимодействующих и взаимосвязанных объектов, которые образуют единое целое. Система описывается своими пространственными и временными границами, окружена и находится под влиянием окружающей среды, описывается своей структурой и назначением и выражается в ее функционировании. Системы являются предметом изучения теории систем[15].

Теория систем рассматривает мир, как сложную систему взаимосвязанных частей. Вы определяете систему, определяя ее границу; это означает выбор того, какие объекты находятся внутри системы, а какие - вне среды. Можно сделать упрощенные представления (модели) системы, чтобы понять ее и предсказать или повлиять на ее будущее поведение. Эти модели могут определять структуру и поведение системы

Существуют природные и созданные человеком системы. Природные системы могут не иметь очевидной цели, но их поведение может быть интерпретировано наблюдателем как целенаправленное. Созданные человеком системы создаются с переменными целями, которые достигаются с помощью некоторых действий, выполняемых системой или вместе с ней. Части системы

должны быть связаны; они должны быть «спроектированы, чтобы работать как единое целое» - иначе это были бы две или более различных системы.

Открытая система также может рассматриваться как ограниченный процесс преобразования, то есть черный ящик, который представляет собой процесс или совокупность процессов, которые преобразуют входные данные в выходные. Входы потребляются; выходы производятся. Концепция ввода и вывода здесь очень широка. Например, выход пассажирского судна - это движение людей от отправления до места назначения.

Подсистема - это набор элементов, представляющих собой саму систему и компонент более крупной системы. Основными общими элементами являются компоненты, которые обрабатывают ввод, планирование, буферизацию и вывод; они также могут взаимодействовать с местными и удаленными операторами.

Исследование состоит из нескольких этапов.

На первом этапе определяются исходные положения исследования, проводится поиск путей повышения эффективности ведения проектов и системы предоставления информации.

На втором этапе проводится разработка системы серверного хранения, обработки информации.

На третьем этапе осуществляется внедрение разработанной системы управления проектами и предоставления API для различных интерфейсов от сайтов до мобильных приложений.

При выборе способа предоставления API остановимся на методологии REST API. Этот выбор обусловлен тем, что: во-первых, существует много хороших фреймворков для различных языков программирования, таких как Django Rest Framework, Express, .Net Web API. Во-вторых, это независимость сервера от клиента — серверы и клиенты могут быть мгновенно заменены другими независимо друг от друга, так как интерфейс между ними не меняется. Сервер не хранит состояний клиента.

Уникальность адресов ресурсов — каждая единица данных (любой степени вложенности) имеет свой собственный уникальный URL, который, по сути, целиком является однозначным идентификатором ресурса.

Независимость формата хранения данных от формата их передачи — сервер может поддерживать несколько различных форматов для передачи одних и тех же данных (JSON, XML и т.д.), но хранит данные в своем внутреннем формате, независимо от поддерживаемых.

Присутствие в ответе метаданных в полном объеме — помимо самих данных сервер должен возвращать детали обработки запроса, например, сообщения об ошибках, различные свойства ресурса, необходимые для дальнейшей работы с ним.

1.5 Анализ существующих разработок в области системы управления проектами

Программное обеспечение для управления проектами (УП) в последние годы стало чрезвычайно популярным, и это означает, что есть множество вариантов на выбор. От

простого способа организации задач до мощной системы корпоративного уровня [16-19]:

- LiquidPlanner - это решение для управления проектами, в котором используется подход, основанный на использовании ресурсов, для установки прогнозируемых дат завершения;
- инструменты повышения производительности Monday.com, такие как встроенное отслеживание времени и просмотр временных шкал, зарекомендовали себя среди широкого круга пользователей;
- Jira - это система управления проектами на основе рабочих процессов, которая лучше всего работает при использовании с другими продуктами Atlassian.

При анализе систем, можно выделить, что они либо ограничены в функционале, либо перегружены. За последние десять лет управление проектами становилось все более сложным. Это часто приводит к крупным проектам, особенно к информационным технологиям, с просрочкой, превышением бюджета и с более низкой, чем прогнозировалось, окупаемостью инвестиций.

Программное обеспечение для управления проектами охватывает целый ряд платформ, каждая из которых имеет несколько разную функциональность. Крайне важно, чтобы выбранный поставщик упростил управление проектами и не добавил ненужной сложности. Переход должен быть максимально плавным. Рассмотрим три основных столпа управления проектами: планирование, отслеживание и сотрудничество. Для каждого компонента рассмотрим, как программное обеспечение для управления проектами и

перечислим некоторые из самых популярных инструментов на рынке. а также самые популярные инструменты на рынке.

Первым столпом управления проектами является планирование. Планирование включает в себя поиск членов команды с необходимыми навыками и определение того, какие ресурсы требуются для проекта. Собрав команду и составив список ресурсов, вы можете использовать систему управления проектами для планирования задач, прогнозирования даты завершения и надлежащего распределения ресурсов.

Планирование сроков завершения для всех движущихся частей в крупном проекте позволяет менеджерам составлять временные рамки проекта и делегировать задачи. Программное обеспечение для управления проектами помогает менеджерам и членам команды планировать задачи, предлагая пространство для записи и сохранения их в виде списков, календарей и многого другого. Автоматические оповещения уведомляют членов команды о приближении сроков или о том, что задача не выполнена в срок. Более надежные интерфейсы включают в себя автоматические отчеты, которые показывают, насколько далеко проект от завершения, на основе количества подзадач и уже достигнутых целей. Пример решения по управлению проектами - Kanbanize, который позволяет пользователям визуализировать состояние завершения задачи, назначая задачи карточкам, которые пользователи могут перемещать между несколькими полосами прогресса.

Даже самые продуманные планы нарушаются на протяжении всего жизненного цикла проекта. Лучшее программное обеспечение для управления проектами обеспечивает достаточную степень детализации и мониторинга, чтобы поддерживать эффективность проектов и следить за их ходом. Как следует из названия, программное обеспечение для отслеживает количества времени, которое каждый участник проекта тратит на свои назначенные задачи. Помимо простого измерения производительности, программное обеспечение для отслеживания времени также создает архив ценных данных, которые могут помочь компаниям прогнозировать сроки завершения аналогичных задач или проектов в будущем. Отслеживание времени особенно полезно при управлении командой удаленных или частично занятых сотрудников. На рисунке 1.2 показан интерфейс отслеживания времени Wrike.

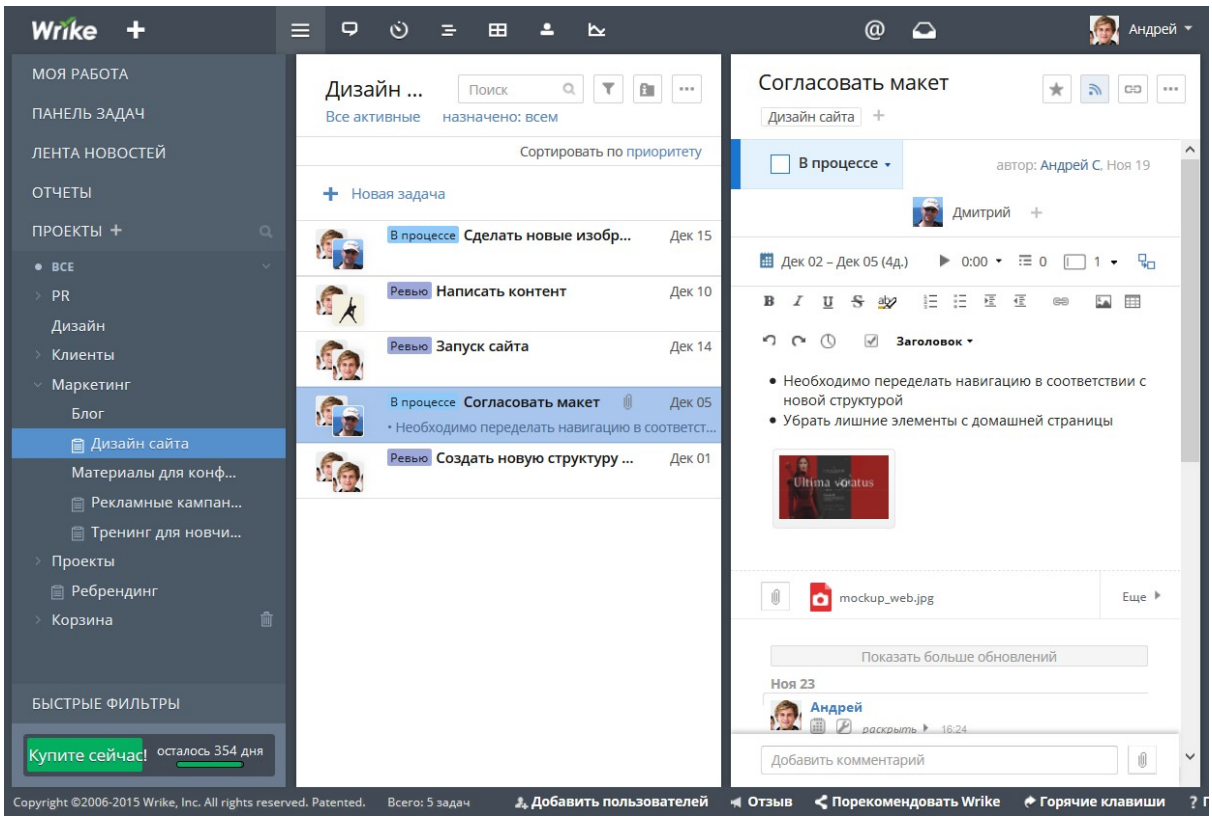


Рисунок 1.2 – Интерфейс отслеживания времени Wrike

Как только проекты начнутся, поддержание связи между движущимися частями может стать самой большой проблемой. Инструменты совместной работы помогают соединять различные отделы и упрощают включение распределенных групп. Обширные функции совместной работы более распространены среди облачного программного обеспечения, чем локальные решения.

Как правило, электронная почта является наиболее популярным способом обмена документами между командами. К сожалению, это часто приводит к избыточной связи, так как отправляется несколько сообщений об одной и той же проблеме. Основная структура электронной почты также затрудняет поиск документов. Программное обеспечение, позволяющее членам группы обмениваться документами, может повысить производительность и

эффективность. Многие решения по управлению проектами позволяют пользователям загружать документы в конкретные проекты или задачи, упрощая поиск ресурсов.

Выводы

Таким образом, исходя из первого раздела дипломной работы, была поставлена цель и задачи исследования, проанализирована общая деятельность ООО «МИРТЕК», представлена организационная структура предприятия, рассмотрены существующие разработки в области системы управления проектами, рассмотрена научная литература по теме работы, определена проблемная ситуация при анализе, а также были рассмотрены методы для анализа модели.

2 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

2.1 Анализ моделей, методов и алгоритмов

Говоря об архитектурах API (интерфейса прикладного программирования), обычно сравнивают SOAP и REST – это две наиболее распространенные парадигмы API. Не смотря на то, что они очень схожи, они по своей сути разные технологии и их нелегко сравнить на детальном уровне [20, 24].

API – это часть программного обеспечения, которая подключает одно приложение непосредственно к данным и услугам другого, предоставляя ему доступ к определенным частям сервера. API-интерфейсы позволяют взаимодействовать двум программным продуктам, они являются основой всего. Он позволяет оптимизировать ИТ-архитектуру, оптимизировать энергопотребление и упростить обмен наборами данных.

REST – это сокращение от REpresentational State Transfer. Это архитектурный стиль для распределенных гипермедиа систем, и он был впервые представлен Роем Филдингом в 2000 году в своей знаменитой диссертации. Как и любой другой архитектурный стиль, REST также имеет свои 6 направляющих принципов перечислены ниже.

Можно выделить руководящие принципы REST:

- клиент-сервер – отделяя проблемы пользовательского интерфейса от проблем хранения данных, мы улучшаем переносимость пользовательского интерфейса на несколько платформ и улучшить

масштабируемость за счет упрощения компонентов сервера;

- без сохранения состояния – каждый запрос от клиента к серверу должен содержать всю информацию, необходимую для понимания запроса, и не может использовать какой-либо сохраненный контекст на сервере. Поэтому состояние сеанса полностью сохраняется на клиенте;

- Cacheable –ограничения Cache требуют, чтобы данные в ответе на запрос были неявно или явно помечены как кешируемые или не кешируемые. Если ответ кешируется, клиентскому кешу предоставляется право повторно использовать эти данные ответа для последующих эквивалентных запросов;

- унифицированный интерфейс – благодаря применению принципа общности разработки программного обеспечения к интерфейсу компонента упрощается общая архитектура системы и улучшается видимость взаимодействий. Чтобы получить унифицированный интерфейс, необходимо несколько архитектурных ограничений для управления поведением компонентов. REST определяется четырьмя интерфейсными ограничениями: идентификация ресурсов; манипулирование ресурсами через представления; информативные сообщения; и гипермедиа как двигатель состояния приложения;

- многоуровневая система. Стиль многоуровневой системы позволяет архитектуре состоять из иерархических слоев, ограничивая поведение компонента таким образом,

что каждый компонент не может «видеть» за пределами непосредственного уровня, с которым они взаимодействуют;

- код по запросу (необязательно) – REST позволяет расширять функциональность клиента путем загрузки и выполнения кода в форме апплетов или скриптов. Это упрощает работу клиентов за счет уменьшения количества функций, необходимых для предварительной реализации.

SOAP (Simple Object Access Protocol) – это собственный протокол, который немного сложнее, поскольку определяет больше стандартов, чем REST, таких как безопасность и способ отправки сообщений. Эти встроенные стандарты несут в себе немного больше накладных расходов, но могут стать решающим фактором для организаций, которым требуются более полные функции в отношении безопасности, транзакций и соответствия требованиям ACID (атомарность, согласованность, изоляция, долговечность). Ради этого сравнения следует отметить, что многие из причин, по которым SOAP является хорошим выбором, редко применяются к сценариям веб-сервисов, что делает его более идеальным для ситуаций корпоративного типа.

У SOAP более строгая защита. WS-Security, в дополнение к поддержке SSL, является встроенным стандартом, который предоставляет SOAP некоторые дополнительные функции безопасности на уровне предприятия, если у вас есть к ним требования.

Успешная / повторная логика для надежного обмена сообщениями. Rest не имеет стандартной системы обмена

сообщениями и может только устранить сбои связи, повторив попытку. SOAP имеет встроенную логику успешных / повторных попыток и обеспечивает сквозную надежность даже через посредников SOAP.

SOAP имеет встроенную совместимость с ACID. Соответствие требованиям ACID уменьшает аномалии и защищает целостность базы данных, точно предписывая, как транзакции могут взаимодействовать с базой данных. ACID является более консервативным, чем другие модели согласованности данных, поэтому его обычно предпочитают при обработке финансовых или других чувствительных транзакций [25].

Основные отличия технологий:

- SOAP – это протокол. REST – это архитектурный стиль. API предназначен для предоставления определенных аспектов бизнес-логики приложения на сервере, и SOAP использует интерфейс службы для этого, в то время как REST использует URI;

- API REST получают доступ к ресурсу для данных (URI); API-интерфейсы SOAP выполняют операцию. REST – это архитектура, которая в большей степени основана на данных; SOAP – это стандартизированный протокол для передачи структурированной информации, который в большей степени зависит от функций;

- безопасность обрабатывается по-разному. SOAP поддерживает WS-Security, который великолепен на транспортном уровне и немного более полон, чем SSL, и более идеален для интеграции с инструментами безопасности уровня предприятия. Оба поддерживают SSL

для сквозной безопасности, в свою очередь REST может использовать безопасную версию протокола HTTP, HTTPS;

- SOAP требует большей пропускной способности; REST требует меньше ресурсов (в зависимости от API). Из-за нестабильного транспорта полезных нагрузок SOAP выходит за пределы шлюза. Поскольку REST используется в основном для веб-служб, его легкость является преимуществом в этих сценариях;

- API создан для обработки полезной нагрузки вашего приложения, а REST и SOAP делают это по-разному. Полезная нагрузка – это данные, отправляемые через Интернет, а когда полезная нагрузка «тяжелая», она требует больше ресурсов. REST имеет тенденцию использовать HTTP и JSON, которые облегчают работу; SOAP больше полагается на XML.

Подводя итоги, можно выделить ключевые различие между SOAP и REST. В REST проще вносить обновления, не разрушая целые отношения. для рассмотрения. SOAP очень тесно связан с сервером, имея с ним строгий коммуникационный контракт, что затрудняет внесение изменений или обновлений. Клиент, взаимодействующий с REST API, не нуждается в знании API, но клиент, взаимодействующий с SOAP API, должен знать обо всем, что он будет использовать, прежде чем он сможет даже инициировать взаимодействие.

После выбора метода предоставления API, необходимо выбрать технологию для его реализации.

ASP.Net является производным продуктом от Active Server Pages от Microsoft вместе с поддержкой языка

программирования .net, используемого для разработки веб-приложений. PHP – это серверный язык сценариев общего назначения от Rasmus Lerdarf, который можно расширить как Hypertext PreProcessor, который также используется для разработки веб-приложений. ASP.Net можно приобрести как платный лицензионный продукт и использовать его для средних и крупных корпоративных приложений. С другой стороны, PHP доступен бесплатно как продукт с открытым исходным кодом и может использоваться только для небольших и средних веб-приложений [26].

Основные характеристики для ASP.NET:

- ASP.NET также предоставляет модель программирования, комплексную программную инфраструктуру и различные другие службы, которые являются обязательными для создания надежного веб-приложения для компьютеров и мобильных устройств, помимо платформы веб-разработки. Впервые он был выпущен в январе 2002 года и является преемником технологии Microsoft Active Server Pages (ASP). ASP.NET не зависит от языка, поэтому разработчики могут использовать любой язык, поддерживаемый .NET, для создания приложений .NET;

- C# и VB.NET являются двумя наиболее распространенными языками для написания приложений. VB.NET напрямую основан на Visual Basic, тогда как C# был представлен вместе с .NET Framework. Платформа ASP.NET обеспечивает отличную поддержку HTML, CSS и JavaScript. Поскольку он построен на Common Language Runtime

(CLR), он позволяет программистам писать код, используя любой поддерживаемый язык .NET;

- ASP.NET широко используется для создания динамических веб-страниц. Он обеспечивает легкий и быстрый способ объединения серверного кода с HTML. Разработчики могут записывать элегантные сайты, которые соответствуют новейшим веб-стандартам. Также помогает добавлять видео, ссылки на сайты социальных сетей. ASP.NET – это надежная структура, с помощью которой разработчики могут писать приложения любого типа. Более того, мы можем использовать любой стиль для создания приложения.

Основные характеристики для PHP:

- PHP начал свой путь как небольшой проект с открытым исходным кодом, который со временем развивался. Существует множество популярных баз данных, которые можно эффективно интегрировать с PHP-кодом. Его код обычно обрабатывается интерпретатором PHP, который реализован в виде модуля в веб-сервере;

- веб-сервер объединяет результаты интерпретируемого и исполняемого кода PHP, который может быть любого типа, включая изображения. PHP самодовольно быстр в своем исполнении. Его выполнение работает даже более плавно, когда он скомпилирован как модуль Apache на стороне Unix. PHP также поддерживает значительное количество основных протоколов, таких как POP3, IMAP и LDAP;

- добавленная поддержка Java и распределенных объектных архитектур впервые делает реальностью

многоуровневую разработку в экосистеме PHP.PHP включает в себя множество бесплатных библиотек с открытым исходным кодом в своем исходном коде. По сути, это интернет-система со встроенными модулями для доступа к FTP-серверам и многим серверам баз данных. Существует множество функций, знакомых программистам на Си, например, в семействе «stdio», которые доступны в стандартных сборках PHP.

Можно выделить ключевые различия между ASP.NET и PHP:

- ASP.NET – это платная платформа веб-приложений, предоставляемая Microsoft, тогда как PHP - это серверный язык сценариев с открытым исходным кодом;
- ASP.NET лучше подходит для крупных и средних организаций, в то время как PHP лучше подходит для начинающих и небольших организаций;
- ASP.NET хорошо оснащен для обслуживания и создания настольных приложений, тогда как PHP работает медленнее, чем ASP.NET для настольных приложений;
- ASP.NET лучше подходит для приложений, где ключевыми вопросами являются безопасность и функциональность, тогда как PHP лучше подходит для приложений, в которых основное внимание уделяется пользовательским интерфейсам;
- платформа ASP.NET информирует разработчиков, если они допустили какую-либо ошибку в кодировании перед компиляцией, что делает ее более безопасной и менее подверженной ошибкам, тогда как в PHP нет такой

возможности, позволяющей разработчику узнать о плохом коде на этапе предварительной компиляции;

- ASP.NET может быть непростой задачей для изучения и понимания для новичка и требует времени для освоения, в то время как PHP, являющийся языком сценариев, легче выучить и понять;

- ASP.NET не допускает каких-либо нарушений, хотя все еще может работать, в то время как PHP очень настраиваемый, следовательно, более подвержен ошибкам, хотя веб-скрипты могут быть эффективно написаны с его помощью.

Из всего выше сказанного можно сделать вывод, что фреймворк ASP.NET имеет самый удивительный набор библиотек. Он поставляется с большим количеством функций, что позволяет разработчику создавать веб-сайт со встроенными функциями перетаскивания. Все эти качества поставляются с ценником в виде лицензионного сбора. У разработчиков PHP нет возможностей работать с широко используемыми веб-фреймворками, такими как ASP.NET. Программист может писать код на любом языке, таком как C#, VB и F# в экосистеме ASP.NET. Но у программиста нет вариантов написания кода на PHP. Таким образом, PHP позволяет разработчикам гибко создавать приложения, позволяя им выбирать из широкого спектра веб-фреймворков.

2.2 Апробация научных методов и алгоритмов системы управления проектами

Программное обеспечение для управления проектами (УП) используется для всех аспектов управления, мониторинга и контроля проекта, включая планирование проекта, планирование, распределение ресурсов и управление изменениями. Программное обеспечение для управления проектами также обеспечивает центральную базу знаний для проектной документации и информации.

Это программное обеспечение является основой успеха управления проектами, поскольку его функциональные возможности базы знаний позволяют членам проектной группы сотрудничать и общаться друг с другом, а также с другими заинтересованными сторонами проекта.

Программное обеспечение для управления проектами также позволяет каждому оптимизировать процессы и работать вместе наиболее эффективным способом. Программное обеспечение поддерживает планирование, управление и контроль проектов в одном централизованном виртуальном местоположении. Кроме того, это позволяет командам оставаться в курсе статусов проектов, рабочих процессов и задач в режиме реального времени.

Программное обеспечение для управления проектами также позволяет каждому оптимизировать процессы и работать вместе наиболее эффективным способом. Программное обеспечение поддерживает планирование, управление и контроль проектов в одном централизованном виртуальном местоположении. Кроме того, это позволяет

командам оставаться в курсе статусов проектов, рабочих процессов и задач в режиме реального времени.

Независимо от того, управляете ли вы одним проектом или несколькими проектами, программами и портфелями, программное обеспечение для управления проектами позволяет вам контролировать, управлять и координировать все и всех. Приняв программное обеспечение для управления проектами, команды могут заменить более элементарные инструменты, такие как электронная почта и электронные таблицы.

В отличие от этих простых инструментов, программное обеспечение для управления проектами является гибким, требует небольшого ручного труда и повышает производительность. Это решение, которое помогает менеджерам проектов и командам управлять проектной работой, вместо того, чтобы тратить много времени на управление самим инструментом.

Если взглянуть на список последних статистических данных по управлению проектами

56% организаций использовали только одну систему управления проектами. В среднем организации тратят 59 499 рублей в месяц на программное обеспечение УП. Большинство - 54% - используют локальное программное обеспечение УП, хотя это быстро меняется.

77% высокоэффективных проектов используют программное обеспечение для управления проектами. Несмотря на это, уровень внедрения программного обеспечения УП остается низким (22% - см. Выше). 66% руководителей проектов говорят, что они будут

использовать программное обеспечение УП более широко, если у них будет адекватная поддержка со стороны их организации.

В период с 2018 по 2019 год доля организаций, использующих электронные таблицы для управления гибкими проектами, снизилась с 74% до 57%. Вместо этого эти организации перешли на специализированные инструменты УП.

Эти данные показывают, что системы УП все чаще внедряются в крупные и мелкие компании.

Практики управления проектами редко бывают единообразными в разных организациях. Различные практики также дают разные результаты для разных предприятий.

В этой статистике управления проектами мы рассмотрим принятие различных практик УП:

- старшие руководители гораздо лучше понимают ценность управления проектами, чем рядовые сотрудники. Среди старших руководителей 87% говорят, что они «полностью» понимают важность практики УП;

- только 32% организаций говорят, что они удовлетворены текущим уровнем зрелости УП. 67% оценили бы уровень зрелости УП своего департамента на 3 или более из 5. Тем не менее, только 47% оценили бы уровень зрелости своих менеджеров на 3 или выше. Это показывает, что существует значительный разрыв между зрелостью УП на уровне департамента и зрелостью на уровне организации.

Выводы

В данном разделе был осуществлен сравнительный анализ моделей, методов и алгоритмов системы управления проектами, выполнены теоретические исследования, рассмотрены архитектуры API и технологии для его реализации.

Собрана статистика по результатам внедрения системы управления в организации, которые позволят увеличить показатель качества обслуживания.

3 ПРОЕКТНЫЙ РАЗДЕЛ

3.1 Проектирование предметной области и анализ результатов проектирования системы управления проектами

Чтобы разработать систему управления проектами, необходимо разделить всю систему на объекты.

Проанализировав то, как проекты контролировались и что они отслеживали в системе, можно выделить основные сущности:

- проект;
- статус проекта;
- задача;
- тип задачи;
- статус задачи;
- приоритет задачи;
- исполнитель задачи.

В таблице 3.1 представлено описание сущности объекта «Проект».

Таблица 3.1 – Сущность объекта «Проект»

Наименование	Тип данных
1	2
ID проекта	int
Название проекта	string
Описание проекта	string
ID пользователя, создавшего проект	int
ID пользователя, ответственного за проекта	int
Дата начала проекта	DateTime

Дата окончания проекта	DateTime
ID статуса проекта	int

Продолжение таблицы 3.1

1	2
ID родительского проекта	int
Флаг - проект помещен в корзину	boolean
Флаг - проект является шаблоном	boolean

В таблице 3.2 представлено описание сущности объекта «Статус проекта».

Таблица 3.2 – Сущность объекта «Статус проекта»

Наименование	Тип данных
1	2
ID статуса	int
Название статуса	string
Описание статуса	string

В таблице 3.3 представлено описание сущности объекта «Задача проекта».

Таблица 3.3 – Сущность объекта «Задача проекта»

Наименование	Тип данных
1	2
ID задачи	int
Название задачи	string
Описание задачи	string
Дата начала задачи	DateTime

Дата окончания задачи	DateTime
ID пользователя, поставившего задачу	int

Продолжение таблицы 3.3

1	2
ID родительской задачи (для подзадачи)	int
ID статуса задачи	int
ID приоритета задачи	int
ID проекта, к которому относится задача	int
ID типа задачи	int

В таблице 3.4 представлено описание сущности объекта «Тип задачи».

Таблица 3.4 – Сущность объекта «Тип задачи»

Наименование	Тип данных
1	2
ID типа	int
Название типа	string
Описание типа	string
HTML-код иконки типа в нотации иконок	string

В таблице 3.5 представлено описание сущности объекта «Статус задачи».

Таблица 3.5 – Сущность объекта «Статус задачи»

Наименование	Тип данных
1	2
ID статуса	int

Название статуса	string

В таблице 3.6 представлено описание сущности объекта «Приоритет задачи».

Таблица 3.6 – Сущность объекта «Приоритет задачи»

Наименование	Тип данных
1	2
ID приоритета	int
Название приоритета	string
Значение цвета приоритета в шестнадцатеричной системе	string

В таблице 3.7 представлено описание сущности объекта «Исполнитель задачи».

Таблица 3.7 – Сущность объекта «Исполнитель задачи»

Наименование	Тип данных
1	2
ID записи	int
ID задачи	int
ID исполнителя	int
ID проекта, к которому относится задача	int

Раньше жизненный цикл проекта велся не централизованно и можно было упустить важные детали при его разработки. Для увеличения возможности контроля и улучшения процессов, были введены новые сущности:

- комментарий к задаче;
- изображение комментария;
- уведомление;

- группа проектов;
- проект включенный в группу;
- файл;
- запись действий.

В таблице 3.8 представлено описание новой сущности объекта «Комментарий к задаче».

Таблица 3.8 – Сущность объекта «Комментарий к задаче»

Наименование	Тип данных
1	2
ID комментария	int
Текст комментария	string
Дата записи комментария	DateTime
ID пользователя, оставившего комментарий	int
ID задаче, к которой относится комментарий	int
Флаг - если установлен в True, то комментарий важен	boolean
ID проекта, к которому относится комментарий	int

В таблице 3.9 представлено описание новой сущности объекта «Изображение прикрепленное к комментарию».

Таблица 3.9 – Сущность объекта «Изображение прикрепленное к комментарию»

Наименование	Тип данных
1	2
ID изображения	int
Название изображения	string

Байтовое представление изображения	byte
Формат изображения в виде строки Base64	string

Продолжение таблицы 3.9

1	2
ID комментария, к которому относится изображение	int
ID проекта, к которому относится изображение	int

В таблице 3.10 представлено описание новой сущности объекта «Уведомление».

Таблица 3.10 – Сущность объекта «Уведомление»

Наименование	Тип данных
1	2
ID типа уведомления	int
Название типа уведомления	string
Количество дней (уведомить за N дней до окончания)	int
Флаг - если True, то уведомление включено и этот тип уведомлений необходимо отправлять пользователю, при возникновении соответствующего действия	bool
Числовое значение типа уведомления	int

В таблице 3.11 представлено описание новой сущности объекта «Проекты включенные в группу».

Таблица 3.11 – Сущность объекта «Проекты включенные в группу»

Наименование	Тип данных
1	2
ID записи	int
ID проекта, назначенного группе	int
ID группы	int

В таблице 3.12 представлено описание новой сущности объекта «Группа проектов».

Таблица 3.12 – Сущность объекта «Группа проектов»

Наименование	Тип данных
1	2
ID группы	int
Название группы	string
Описание группы	string
ID пользователя, создавшего группу	int

В таблице 3.13 представлено описание новой сущности объекта «Папка проекта».

Таблица 3.13 – Сущность объекта «Папка проекта»

Наименование	Тип данных
1	2
ID папки	int
Описание папки	string
Ссылка на папку	string
Полный путь к папке на сервере	string
Название папки	string
Дата создания папки	DateTime

ID пользователя, создавшего папку	int
ID проекта, к которому относится папка	int

В таблице 3.14 представлено описание новой сущности объекта «Запись действий».

Таблица 3.14 – Сущность объекта «Запись действий»

Наименование	Тип данных
1	2
ID записи	int
Название действия	string
Дата формирования действия	DateTime
ID пользователя, создавшего определенное действие	int
Описание действия в формате XML	XDocument
Описание действия	string
ID проекта, если действие направлено на проект	int
ID задачи, если действие направлено на задачу	int
ID комментария, если действие направлено на комментарий	int
ID группы проектов, если действие направлено на группу проектов	int
ID папки, если действие направлено на папку	int
ID файла, если действие направлено на файл	int
Числовое значение типа действия	int

После чего появится возможность, мгновенно уведомлять руководство и менеджера проекта, а так же делать комментарии к задаче, прикреплять сопутствующие документы, а работа со схожими проектами станет проще, формируя группы проектов.

Все это упростит работу руководства, отдела разработки и тестирования. Таким образом, все действия будут происходить централизованно, упроститься хранение документации и ускорить время разработки.

Для поддержки разработанной системы была разработана еще одна сущность, которая должна помочь понять причину и место возникновения ошибки при работе с системой.

В таблице 3.15 представлено описание новой сущности объекта «Запись ошибок системы».

Таблица 3.15 – Сущность объекта «Запись ошибок системы»

Наименование	Тип данных
1	2
ID записи	int
Имя класса, в котором произошла ошибка	string
Имя метода, в котором произошла система	string
Дата возникновения ошибки	DateTime
ID пользователя, у которого возникла ошибка при работе с системой	int

После того как все сущности описаны, необходимо спроектировать интерфейс и посмотреть, как все должно выглядеть и каким правилам соответствовать.

Очень важно при разработке API REST выбрать правильные сущности и смоделировать ресурсы с правильной гранулярностью, чтобы потребители API получали желаемую функциональность, API работала правильно и поддерживалась [28-29, 32].

Так как сервер API разрабатывается не для конечного пользователя, а для разработчиков внешнего интерфейса, необходимо соблюсти несколько требований:

- API должен быть простой для понимания;
- простая аутентификация для получения данных;
- система должна быть расширяемая.

Так как данная система является частью микросервисов, аутентификация осуществляется сторонним сервером, которая выдает специальный токен.

Принципы RESTful предоставляют стратегии для обработки действий CRUD с использованием методов HTTP, отображаемых следующим образом:

- GET /api/v1/projects/ - получить все проекты пользователя;
- POST/api/v1/projects/ - создать новый проект;
- GET /api/v1/projects/{id} - получить конкретный проект по идентификатору;
- DELETE /api/v1/projects/{id} - удалить проект по идентификатору;
- PUT /api/v1/projects/{id} - изменить проект по идентификатору;
- GET /api/v1/projects/{id}/comments/ - получить список комментариев принадлежащий к проекту;

- DELETE /api/v1/projects/{id}/comments/{id} - удалить комментарий по идентификатору;
- GET /api/v1/projects/{id}/actions/ - получить список действий связанных с проектом.

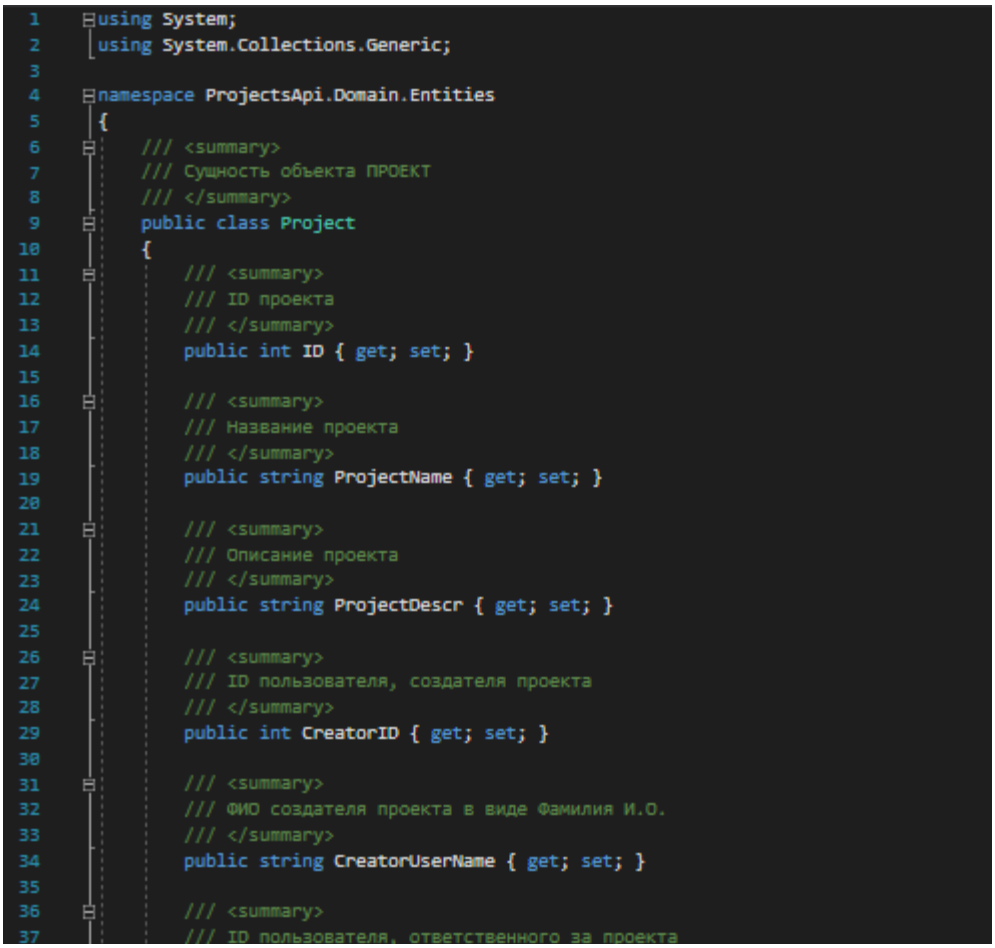
По аналогии реализованы остальные пути API, представлены без CRUD описания:

- /api/v1/tasks/
- /api/v1/tasks/{id}
- /api/v1/tasks/{id}/actions/
- /api/v1/tasks/{id}/comments/
- /api/v1/tasks/{id}/comments/{id}
- /api/v1/files/
- /api/v1/files/{id}/

Потребитель API не всегда нуждается в полном представлении ресурса. Возможность выбирать и выбирать возвращаемые поля имеет большое значение, позволяя потребителю API минимизировать сетевой трафик и ускорить собственное использование API.

Для примера используя параметр запроса name, который принимает строку, можно отфильтровать проекты по имени, упрощая тем самым пользователю поиск проекта. Например, следующий запрос вернет задачи, в имени которых присутствует слово «сделать» и статус задачи в состоянии выполняется (передать id статуса «выполняется», который равен «2»): GET /api/v1/tasks?name=сделать&status=2. Реализуем на примере проекта. Для этого решим следующую задачу: описать сущность, модель и обработчик запроса. Создадим файл сущности

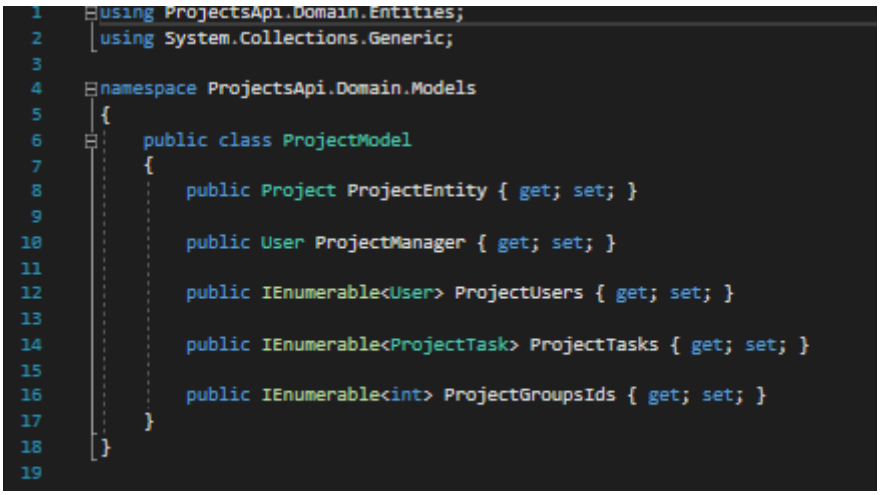
«Проект» (рисунок 3.1), полная реализация сущности представлена в приложении А.



```
using System;
using System.Collections.Generic;

namespace ProjectsApi.Domain.Entities
{
    /// <summary>
    /// Сущность объекта ПРОЕКТ
    /// </summary>
    public class Project
    {
        /// <summary>
        /// ID проекта
        /// </summary>
        public int ID { get; set; }

        /// <summary>
        /// Название проекта
        /// </summary>
        public string ProjectName { get; set; }

        /// <summary>
        /// Описание проекта
        /// </summary>
        public string ProjectDescr { get; set; }

        /// <summary>
        /// ID пользователя, создателя проекта
        /// </summary>
        public int CreatorID { get; set; }

        /// <summary>
        /// ФИО создателя проекта в виде Фамилия И.О.
        /// </summary>
        public string CreatorUserName { get; set; }

        /// <summary>
        /// ID пользователя, ответственного за проекта
```

Рисунок 3.1 – Файл сущности «Проект»

После того, как сущность «Проект» описана, необходимо описать его класс модели (рисунок 3.2).



```
using ProjectsApi.Domain.Entities;
using System.Collections.Generic;

namespace ProjectsApi.Domain.Models
{
    public class ProjectModel
    {
        public Project ProjectEntity { get; set; }

        public User ProjectManager { get; set; }

        public IEnumerable<User> ProjectUsers { get; set; }

        public IEnumerable<ProjectTask> ProjectTasks { get; set; }

        public IEnumerable<int> ProjectGroupsIds { get; set; }
    }
}
```

Рисунок 3.2 – Модель сущности «Проект»

Далее когда все описано можно приступать к написанию обработчиков, для этого необходимо создать «Контроллер» и проверить права доступа пользователя, после чего достать все проекты, принадлежащие этому пользователю и сформировать ответ. Если в обработчике происходит ошибка, то она логируется и возвращается код ошибки клиенту сервиса (рисунок 3.3.). Полный метод обработки запроса представлен в приложении Б.

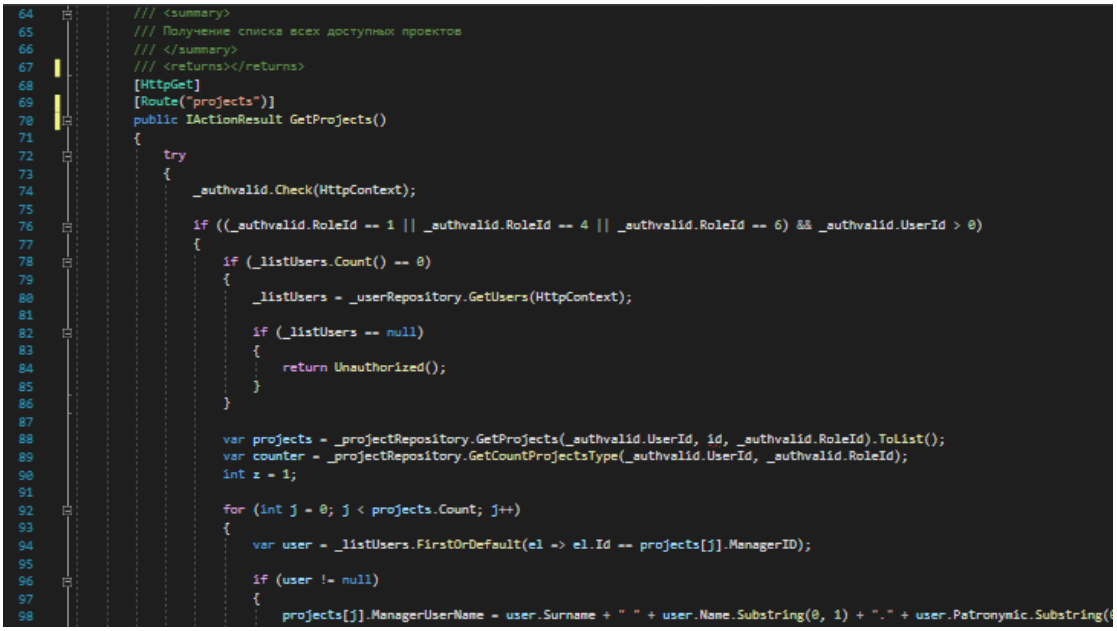


Рисунок 3.3 – Метод обработки запроса

3.2 Обоснование выбора средств разработки

Один из самых популярных типов API – это REST или, как их иногда называют, API-интерфейсы RESTful. API REST или RESTful были разработаны для использования преимуществ существующих протоколов. Хотя REST – или передача состояния представления – может использоваться практически для любого протокола, при использовании для веб-API он обычно использует преимущества HTTP. Это означает, что разработчикам не нужно устанавливать дополнительное программное обеспечение или библиотеки при создании REST API.

Одним из ключевых преимуществ API REST является то, что они обеспечивают большую гибкость. Данные не привязаны к ресурсам или методам, поэтому REST может обрабатывать несколько типов вызовов, возвращать разные форматы данных и даже структурно изменяться при правильной реализации гипермедиа. Такая гибкость позволяет разработчикам создавать API, отвечающий любым потребностям, а также потребностям самых разных клиентов.

Существует 6 плюсов при выборе RESTful API:

- клиент-сервер: клиент и сервер отделены друг от друга что позволяет развиваться индивидуально;
- отсутствие состояния: API REST не имеют состояния, что означает, что вызовы могут выполняться независимо друг от друга, и каждый вызов содержит все данные, необходимые для успешного завершения;
- кэширование: поскольку API без сохранения состояния может увеличить накладные расходы на запросы, обрабатывая большие нагрузки входящих и исходящих вызовов, REST API может быть спроектирован так, чтобы стимулировать хранение кэшируемых данных;
- многоуровневая система. API-интерфейсы REST имеют разные уровни своей архитектуры, которые работают вместе для создания иерархии, которая помогает создавать более масштабируемое и модульное приложение;
- код по требованию: Код по требованию позволяет передавать код или апплеты через API для использования в приложении.

В отличие от SOAP, REST не ограничивается XML, но вместо этого может возвращать XML, JSON, YAML или любой другой формат в зависимости от того, что запрашивает клиент. И в отличие от RPC, пользователи не обязаны знать имена процедур или конкретные параметры в определенном порядке. Одним из недостатков API RESTful является то, что вы можете потерять способность поддерживать состояние в REST, например, в сеансах. Это также может быть более сложным для использования новыми разработчиками.

Почему ASP.net, когда рынок переполнен разными программными средствами, такими как Djnago, Spring, NodeJS и т.д. В пользу выбора программного средства, можно выделить существенные плюсы:

- ASP.Net – это платформа веб-приложений с открытым исходным кодом, разработанная Microsoft. Это дает поддержку и развитие технологии как со стороны большой компании так и со стороны сообщество;
- Asp.net очень прост в освоении по сравнению с другими языками программирования, разработка веб-сайтов с использованием Asp.net очень проста;
- средства разработки, разрабатываются также Microsoft, что дает идеальную среду разработки для данной технологии;
- исправление ошибок и обслуживание легко в случае asp.net;
- читаемость кода проста в случае asp.net, поэтому мы можем легко освоить.

3.3 Описание реализации предметной области

API REST с .NET и C# ASP.NET позволяет легко создавать сервисы, которые охватывают широкий круг клиентов, включая браузеры и мобильные устройства. В ASP.NET вы используете одну и ту же структуру и шаблоны для создания веб-страниц и сервисов параллельно в одном проекте.

Данный подход позволяет более гибкое управление данными. Что Позволяет сделать любые выгружаемые документы: от статистики, до отчетов по проектам, так как такой API позволяет отдавать данные любых форматов.

ASP.NET имеет автоматическую сериализацию классов для правильного форматирования JSON из коробки. Никаких специальных настроек не требуется. Конечно, сериализация может быть настроена для уникальных требований.

Маршрутизация вместе с кодом ASP.NET позволяет определять маршруты и глаголы, встроенные в код, используя атрибуты. Данные из пути запроса, строки запроса и тела запроса автоматически привязываются к параметрам метода.

Без труда можно сделать автоматизирование создание проектов, назначить команду и создать первоначальные задания. Для этого необходимого, всего лишь добавить uri ветку API и сделать соответствующие обработчики.

Отличный инструмент для любой платформы сборка, отладка и развертывание с любой платформы на любую платформу.

3.4 Обеспечение информационной безопасности при эксплуатации системы управления проектами

Аутентификация и авторизация Защищенные конечные точки API со встроенной поддержкой отраслевых стандартных токенов JSON (JWT). Авторизация на основе политик позволяет гибко определять мощные правила контроля доступа.

SP.NET обеспечивает первоклассную поддержку HTTPS из коробки. Автоматически генерируется тестовый сертификат который просто нужно импортировать, чтобы включить локальный HTTPS, Это позволяет запускать и отлаживать код, так как бы он работал в рабочем режиме, обеспечивая безопасность даже во время разработки, таким образом, можно отлаживать код не только на локальном компьютере, но и на удаленном сервере.

Выводы

Данная глава посвящена проектированию программного модуля, обоснованию выбора средств разработки, основному описанию предметной области и информационной безопасности разрабатываемой системы управления проектами.

Были сделаны следующие выводы:

- учитывая специфику разрабатываемого программного модуля и обеспечиваю простоту управления, было решено использовать фреймворк ASP.net;

- был разработан интерфейс прикладного программирования (API), а именно набор классов, процедур и функций;

- проанализирована безопасность модуля и определены угрозы безопасности.

4 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

Расчет затрат на работу целесообразно проводить точным методом на основе нормативных материалов и трудовых затрат.

Исходными данными для расчета являются: нормы трудоемкости по выполнению отдельных видов работ, часовые тарифные ставки специалистов различной квалификации, спецификации оборудования и материалов, используемых при изготовлении товара, цены на материалы и комплектующие изделия, норматив отчислений на социальное страхование, тариф на электроэнергию.

4.1. Трудоемкость выполняемых работ

При реализации информационной подсистемы, которая предполагает разработку системы управления проектами и всю необходимую программную документацию, предусмотренную техническим заданием, необходимо учитывать трудоемкость разработки программного обеспечения.

Трудоемкость разработки программного обеспечения $T_{ПО}$, чел.ч, определяется по формуле 4.1.

$$T_{ПО} = T_{О} + T_{И} + T_{А} + T_{П}, \quad (4.1)$$

где $T_{О}$ – затраты труда на описание задачи, чел.ч;

$T_{И}$ – затраты на исследование предметной области, чел.ч;

$T_{А}$ – затраты на разработку блок-схем, чел.ч;

$T_П$ – затраты на программирование, чел.ч.

Большинство составляющих в правой части формулы (4.1) определяются через общее число операторов (D).

$$D = \alpha \times c \times (1 + p), \quad (4.2)$$

где α – исходное число строчек кода в тексте программы, (α = 3643 ед.);

c – коэффициент сложности задачи;

p – коэффициент коррекции программы, учитывающий новизну проекта.

Коэффициент сложности задачи (c) характеризует относительную сложность программы по отношению к так называемой типовой задаче, реализующей стандартные методы решения, сложность которой принята равной единице (величина коэффициента «c» лежит в пределах от 1,25 до 2). Для рассматриваемого программного продукта, включающего в себя алгоритмы учета, отчетности, поиска – коэффициент сложности задачи примем равным 1,5 (c = 1,5).

Коэффициент коррекции программы (p), учитывающий новизну проекта, количественно характеризует увеличение объема работ по реализации программного продукта, возникающего за счет внесения изменений в алгоритм или в тексте программы по результатам её тестирования и отладки, с учетом коррекций требований к прецедентам, поддерживаемым программным продуктом, со стороны заказчика. В данном случае заказчик, недостаточно хорошо представлял себе, полный перечень прецедентов, которые должен поддерживать программный продукт, а это

приводило к многочисленным корректировкам и доработкам текста программного кода. Поэтому примем коэффициент коррекции программы равным 0,1 (p = 0,1).

В результате подстановки численных значений коэффициентов и параметров в формулу (4.2) получим следующее общее число строчек кода в тексте программы.

$$D = 3643 \times 1,5 \times (1 + 0,1) = 6010,9 \text{ (ед.)}$$

Затраты труда на описание задачи принимаем: T_O = 40 чел.ч. Работу по описанию задачи и все другие работы по созданию системы управлния проектами оплачивается в размере оклада 9450 руб. в месяц и коэффициентом квалификации k_K = 0,8 (опыт работы по специальности до 2 лет).

Затраты труда на изучение задачи $T_И$, чел.ч, с учетом уточнения описания и квалификации программиста могут быть определены по формуле 4.3.

$$T_И = \frac{D \times \beta}{s_u \times k_к},$$

(4.3)

где D – общее число строчек кода в тексте программы, ед. (формула 4.2);

β – коэффициент увеличения затрат труда, вследствие недостаточного описания задачи;

s_u – количество строчек кода в тексте программы, приходящееся на один чел.ч, (ед./ чел.ч);

$k_к$ – коэффициент квалификации работника (определяется в зависимости от стажа работы).

В связи с тем, что решение рассматриваемой задачи потребовало уточнения и доработок, примем коэффициент β = 1,5.

Количество строчек кода в тексте программы, приходящееся на один чел.ч, примем равным s_u = 75 ед./ чел.ч

Таким образом, на основании формулы (4.3) получим.

$$T_И = \frac{6010,9 \times 1,5}{75 \times 0,8} = 150,27 \text{ (чел.ч)}$$

Затраты труда на разработку алгоритма решения задачи T_A, чел.ч, рассчитываются по формуле 4.4.

$$T_A = \frac{D}{s_a \times k_к}, \quad (4.4)$$

где D – общее число строчек кода в тексте программы, ед. (формула 4.2);

s_a – количество строчек кода в тексте программы, приходящееся на один чел.ч, (ед./ чел.ч);

k_K – коэффициент квалификации работника (определяется в зависимости от стажа работы).

Для расчета по формуле (4.4) примем s_a = 20 ед./ чел.ч.

Подставив численные значения параметров и коэффициентов в формулу (4.4), получим.

$$T_A = \frac{6010,9}{20 \times 0,8} = 375,68 \text{ (чел.ч)}$$

Затраты труда на составление программы по готовой блок-схеме $T_П$, чел.ч, определяется по формуле 4.5.

$$T_П = \frac{D}{s_a \times k_к}, \quad (4.5)$$

где D – общее число строчек кода в тексте программы, ед. (формула 4.2);

s_a – количество строчек кода в тексте программы, приходящееся на один чел.ч, (ед./ чел.ч);

$k_К$ – коэффициент квалификации работника (определяется в зависимости от стажа работы).

Для расчетов по формуле (4.5) примем s_a = 20 ед./ чел.ч.

Подставив численные значения параметров и коэффициентов в формулу (4.5), получим.

$$Т_П = \frac{6010,9}{20 \times 0,8} = 375,68 \text{ (чел.ч)}$$

Подставив все полученные данные, составляющие трудоемкость разработки программного обеспечения в формулу (4.1), получим.

$Т_{ПО}$ = 40,00 + 150,27 + 375,68 + 375,68 = 941,63 (чел.ч)

С учетом уровня языка программирования трудоемкость разработки программы может быть скорректирована следующим образом 4.6.

$$Т_{КОР} = Т_{ПО} \times k_{КОР}, \quad (4.6)$$

где $Т_{КОР}$ – скорректированная трудоемкость разработки программного обеспечения, чел.ч;

$Т_{ПО}$ – трудоемкость разработки программного обеспечения, чел.ч (формула 4.1),

$k_{КОР}$ – коэффициент коррекции, учитывающий изменения трудоемкости разработки программного обеспечения в зависимости уровня языка программирования [24-25].

Использованный для разработки программного обеспечения язык программирования (C#) относится к

алгоритмическим языкам высокого уровня, с учетом этого примем $k_{КОР} = 0,9$.

Таким образом, получим по формуле (4.6) итоговую откорректированную трудоемкость разработки программы:

$$Т_{КОР} = 941,63 \times 0,9 = 847,47 \text{ (чел.ч)}$$

4.2. Расчет себестоимости системы управления проектами

Себестоимость создания системы управления проектами (З, руб.), определяется по следующей формуле 4.7.

$$З = З_О + З_Д + З_С + З_Э + З_М + З_П + З_{АО}, \quad (4.7)$$

где $З_О$ – основная заработная плата производственного персонала, руб.;

$З_Д$ – дополнительная заработная плата производственного персонала, руб.;

$З_С$ – отчисления на страховые взносы, руб.;

$З_Э$ – затраты на потребляемую электроэнергию, руб.;

$З_М$ – расходы на материалы и запасные части, руб.;

$З_П$ – затраты на техническое обслуживание и текущий ремонт вычислительной техники, руб.;

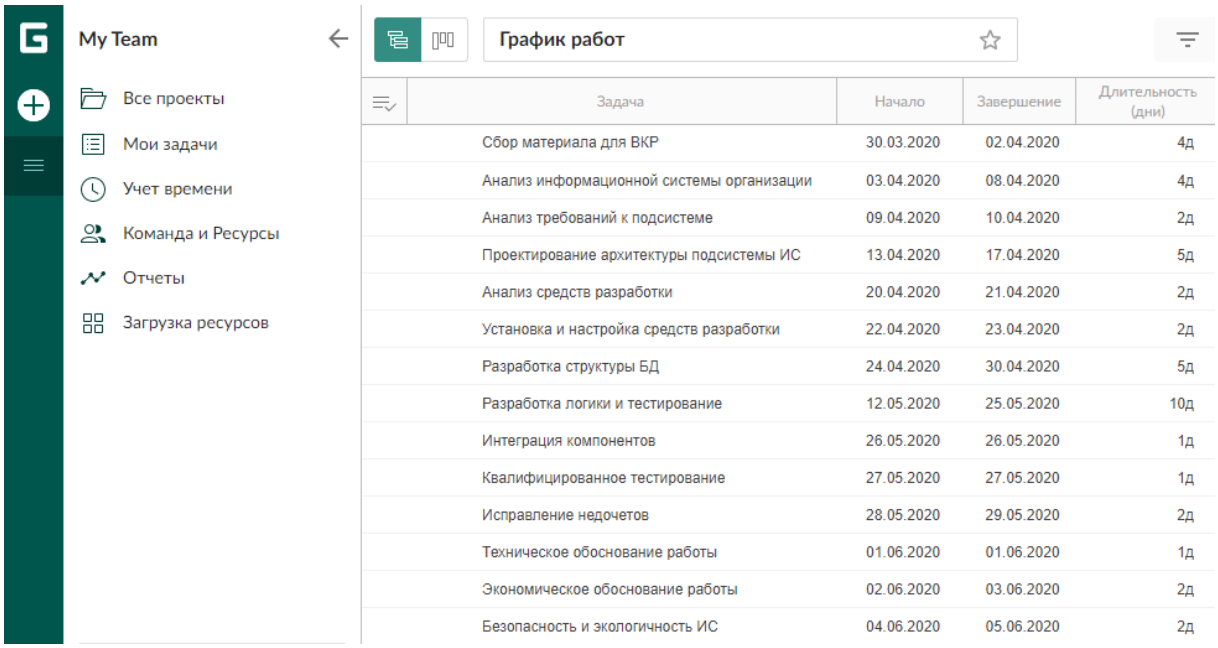
$З_{АО}$ – затраты на амортизацию вычислительной техники, руб.

Для выявления занятости и общего количества рабочих дней (планового фонда рабочего времени $Т_{ПФ}$) был составлен календарный план в онлайн платформе Ganttpro. Первым делом были определены свойства и параметры проекта разработки подсистемы регистрации. Далее был

составлен календарь рабочего времени – 8 часовой рабочий день, суббота и воскресенье – выходные дни, но в связи с Указами Президента РФ о нерабочих днях с 30.03.2020 по 29.04.2020 от 25.03.2020 № 206, от 02.04.2020 № 239, от 28.04.2020 № 294, работа была выстроена удаленно, что бы успеть в поставленные сроки.

После выполнения указанных действий была определена последовательность и продолжительность работ в рамках дипломного проектирования. Начало построения диаграммы Ганта представлено на рисунке 4.1.

На рисунке 4.2 можно увидеть последовательность выполнения работ с указанием длительности, дат начала и окончания выполнения и исполнителя.



Задача	Начало	Завершение	Длительность (дни)
Сбор материала для ВКР	30.03.2020	02.04.2020	4д
Анализ информационной системы организации	03.04.2020	08.04.2020	4д
Анализ требований к подсистеме	09.04.2020	10.04.2020	2д
Проектирование архитектуры подсистемы ИС	13.04.2020	17.04.2020	5д
Анализ средств разработки	20.04.2020	21.04.2020	2д
Установка и настройка средств разработки	22.04.2020	23.04.2020	2д
Разработка структуры БД	24.04.2020	30.04.2020	5д
Разработка логики и тестирование	12.05.2020	25.05.2020	10д
Интеграция компонентов	26.05.2020	26.05.2020	1д
Квалифицированное тестирование	27.05.2020	27.05.2020	1д
Исправление недочетов	28.05.2020	29.05.2020	2д
Техническое обоснование работы	01.06.2020	01.06.2020	1д
Экономическое обоснование работы	02.06.2020	03.06.2020	2д
Безопасность и экологичность ИС	04.06.2020	05.06.2020	2д

Рисунок 4.1 – Построение диаграммы Ганта

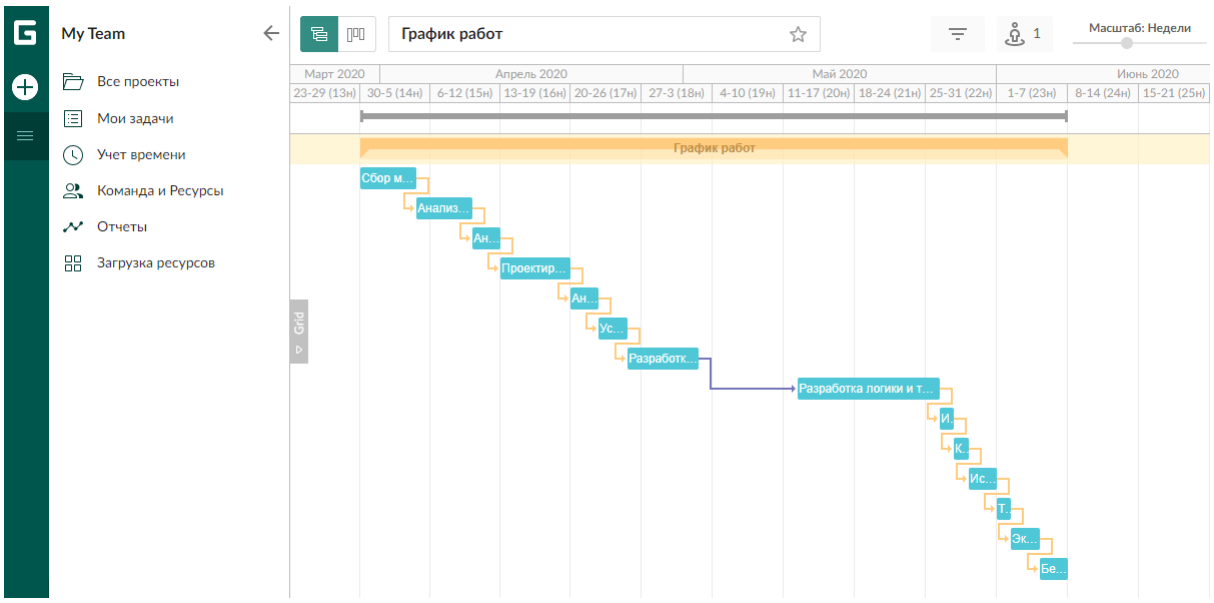


Рисунок 4.2 – Построение диаграммы Ганта

Таким образом, на выполнение работы было затрачено 43 дня на разработку системы управления проектами, что соответствует 344 ч при восьмичасовом рабочем дне, с учетом праздничных дней.

Таким образом, часовая тарифная ставка $s_ч$, руб./ч, составляет.

$$s_ч = \frac{9450}{344} = 27{,}47 \text{ (руб./ч)}$$

Основная заработная плата ($З_О$, руб.), производственного персонала инженера-программиста первой категории определяется по формуле 4.8.

$$З_О = s_ч \times Т_{КОР}. \quad (4.8)$$

Подставив все численные значения параметров в формулу (4.8) получим, что основная заработная составит.

$$З_О = 27{,}47 \times 847{,}47 = 23280 \text{ (руб.)}$$

Дополнительная заработная плата ($З_Д$, руб.), производственного персонала инженера-программиста первой категории определяется по формуле 4.9.

$$З_Д = З_О \times \eta_Д, \quad (4.9)$$

где $\eta_Д$ – коэффициент дополнительной заработной платы.

Коэффициент дополнительной заработной платы инженера-программиста первой категории составляет $\eta_Д = 0,2$. Таким образом, дополнительная заработная плата ($З_Д$, руб.), инженера-программиста первой категории, вычисленная по формуле (4.9), равна.

$$З_Д = 23280 \times 0,2 = 4656 \text{ (руб.)}$$

Отчисления в Пенсионный фонд Российской Федерации, Фонд социального страхования Российской Федерации и фонды обязательного медицинского страхования Российской Федерации в соответствии с НК РФ ($З_С$, руб.), вычислим по формуле 4.10.

$$З_С = \frac{(З_О + З_Д)}{100} \times \eta_С \quad (4.10)$$

где $\eta_с$ – норматив страховых взносов, %.

В соответствии с НК РФ с предприятия от фонда оплаты труда для направления во внебюджетные фонды в следующем размере:

- в Пенсионный фонд России (ПФР) – 22%;
- в Фонд социального страхования (ФСС) – 2,9%;
- в Фонд обязательного медицинского страхования (ФОМС) – 5,1%;

Кроме того, Фонд социального страхования так же осуществляет обязательное страхование от несчастных случаев в соответствии с тарифами по отраслям (предприятия IT-сферы относятся к I классу риска, для которого начисляется минимальная ставка 0,2% к величине фонда оплаты труда) [25].

Норматив страховых взносов для расчета, учитывая все вышеперечисленное, составляет 30,2% ($\eta_С$ = 30,2%).

Подставив все численные значения в формулу (4.10) получим, что отчисления на страховые взносы равны.

$$З_С = \frac{(23280+4656)}{100} \times 30,2 = 8436,67 \text{ (руб.)}$$

Таким образом, размер страховых взносов составит 8436,67 руб.

Затраты на потребляемую электроэнергию ($З_Э$, руб.).

$$З_Э = P_В \times t_В \times c_Э, \quad (4.11)$$

где $P_В$ – мощность ЭВМ, кВт;

$t_В$ – время работы вычислительного комплекса, ч;

$c_Э$ – стоимость 1 кВтч электроэнергии, руб./ кВтч.

Мощность ЭВМ, на которой работает инженер-программист первой категории, равна $P_В$ = 0,3 кВт.

Время работы вычислительного комплекса ($t_В$, ч), при создании программного продукта вычислим по формуле 4.12.

$$t_В = \gamma_{пр} \times T_П \times k_{КОР}, \quad (4.12)$$

где $\gamma_{пр}$ – коэффициент, учитывающий затраты времени на профилактические работы на ЭВМ;

$k_{КОР}$ – коэффициент коррекции времени работы вычислительного комплекса.

Для расчетов по формуле (4.12) примем $\gamma_{пр}$ = 1,15 и $k_{КОР}$ = 0,9. Подставив все численные значения параметров в формулу (4.12) получим.

$$t_В = 1,15 \times 375,68 \times 0,9 = 388, 83 \text{ (ч.)}$$

Стоимость 1 кВт электроэнергии составляет $c_Э$ = 4,64 руб./ кВтч (с учетом НДС).

Подставив все численные значения параметров в формулу (4.11) получим, что затраты на потребляемую электроэнергию составят:

$$З_Э = 0,3 \times 388,83 \times 4,64 = 541,25 \text{ (руб.)}$$

Для расчета затрат на материалы и запасные части потребуется ($З_М$, руб.) 10500,00 руб.

Затраты на техническое обслуживание и текущий ремонт вычислительной техники ($З_П$, руб.).

$$З_П = К_В \times \frac{\alpha}{100} \times \frac{t_В}{t_{ВГ}},$$

(4.12)

где $К_В$ − балансовая стоимость вычислительной техники, руб.

α – норма отчислений на ремонт, %;

$t_{В.Г}$ – годовой фонд времени работы вычислительной техники, ч.

Для расчетов по формуле (4.12) примем:

- балансовая стоимость вычислительной техники $К_В$= 23000,00 руб.;
- норма отчислений на ремонт α = 4%;

- годовой фонд времени работы вычислительной техники при 40-часовой рабочей неделе в текущем году $t_{В.Г}$ = 1970 ч.

Подставив все численные значения параметров в формулу (4.12) получим, что затраты на техническое обслуживание и текущий ремонт вычислительной техники составят:

$$З_П = 23000 \times \frac{4}{100} \times \frac{388,83}{1970} = 181,59 \text{ (руб.)}$$

Затраты на амортизацию вычислительной техники ($З_{АО}$, руб.):

$$З_{АО} = К_В \times \frac{\beta}{100} \times \frac{t_В}{t_{ВГ}},$$

(4.13)

где $К_В$ – балансовая стоимость вычислительной техники, руб.

β – норма отчислений на амортизацию вычислительной техники, %;

$t_{В.Г}$ – годовой фонд времени работы вычислительной техники, ч.

Для расчетов по формуле (4.13) примем:

- балансовая стоимость вычислительной техники $К_В$= 23000,00 руб.;

- норма отчислений на амортизацию вычислительной техники β = 33,3%;

- годовой фонд времени работы вычислительной техники при 40-часовой рабочей неделе в текущем году $t_{В.Г}$ = 1970 ч.

Подставив все численные значения параметров в формулу (4.13) получим, что затраты на амортизацию вычислительной техники ($З_{АО}$, руб.) составят:

$$З_{АО} = 23000 \times \frac{33,3}{100} \times \frac{388,83}{1970} = 1511,69 \text{ (руб.)}$$

Все расчеты по статьям калькуляции затрат, составляющих себестоимость системы управления проектами сведены в таблицу 4.3.

Таблица 4.3 – Величины затраты, составляющих себестоимость системы управления проектами

Статья расхода	Сумма, руб.
1	2
Основная заработная плата производственного персонала	23280,00
Дополнительная заработная плата производственного персонала	4656,00
Отчисления на страховые взносы	8436,67
Затраты на потребляемую электроэнергию	541,25
Расходы на материалы и запасные части	10500,00
Затраты на техническое обслуживание и ремонт вычислительной техники	181,59
Затраты на амортизацию вычислительной техники	1511,69
Итого	48907,20

Таким образом, себестоимость затраты на создание программного продукта составляют 48907,20руб.

Разработка системы управления проектами рассчитана с учетом оптовой цены и рассчитана по формуле:

$$Ц = З \times (1 + Н_Р), \quad (4.14)$$

где $Н_Р$ – норма рентабельности, %.

Для расчетов по формуле (4.14) примем $Н_Р$ = 15%. Подставив численное значение параметров в формулу (4.14) получим:

Ц = 48907,20×(1+0,15) = 56243,28 (руб.)

Капиталовложения при внедрении системы управления проектами равняются его оптовой цене:

К = Ц = 56243,28 руб.

4.2 Оценка экономической эффективности и срока окупаемости от реализации проекта

Показатель эффективности обуславливает все положительные результаты, достигаемые при использовании системы управления проектами. Прибыль от использования программного продукта за год эксплуатации (П, руб.), определяется по формуле 4.15.

$$П = Э - З_{исп.}, \quad (4.15)$$

где Э – стоимостная оценка результатов применения программного продукта в течение года, руб.;

$З_{исп.}$ – стоимостная оценка затрат при использовании программного продукта в течение года, руб.

Приток денежных средств из-за использования информационной системы (Э, руб.), в течение года может составить (4.16).

$$Э = (З_{РУЧ} - З_{АВТ}) + Э_{ДОП}, \quad (4.16)$$

где $З_{РУЧ}$ – затраты на ручную обработку информации, руб.;

$З_{АВТ}$ – затраты на автоматизированную обработку информации, руб.;

$Э_{ДОП}$ – дополнительный экономический эффект, связанный с уменьшением числа используемых бланков, высвобождением рабочего времени и т. д., руб.

За системой управления проектами следит инженер-программист, оклад которого составляет 25000 руб. Тогда, цена одного часа работы ($ц_ч$, руб./ч), с учетом восьми часовой работы и количеством рабочих дней за месяц = 20 составит.

$$ц_ч = \frac{25000}{160} = 156{,}25 \text{ (руб./ч.)}$$

Из данных, полученных в ходе тестирования системы управления проектами, о времени, затрачиваемом на обработку информации и контроль проектов не в вручную, а при использовании программного продукта за месяц, сократилось с $t_{ОБЩ.\ Р}$ = 45 ч. до $t_{ОБЩ.\ А}$ = 5 ч., что в 9 раз меньше.

Годовые затраты при ручной обработке информации вычислим по формуле 4.17.

$$З_{РУЧН} = t_{ОБЩ.\ Р} \times 12 \times ц_ч. \quad (4.17)$$

Тогда годовые затраты при ручной обработке информации составят:

$$З_{РУЧН} = 45 \times 12 \times 156{,}25 = 84375 \text{ (руб.)}$$

Годовые затраты при автоматизированной обработке информации вычислим по формуле 4.18.

$$З_{АВТ} = t_{ОБЩ.\ А} \times 12 \times ц_ч. \quad (4.18)$$

При автоматизированной обработке информации затраты составят:

$$З_{АВТ} = 5 \times 12 \times 156,25 = 9375 \text{ (руб.)}$$

Следовательно, годовой эффект от внедрения программного продукта, даже без учета дополнительный экономический эффекта ($Э_{ДОП} = 0$), на основании формулы (4.16), получится равным:

$$Э = 84375 – 9375 = 75000 \text{ (руб.)}$$

Эксплуатационные затраты при использовании программного продукта состоят из затрат на электроэнергию, техническое обслуживание, текущий ремонт вычислительно техники и затрат на амортизацию вычислительной техники.

На основании формулы (4.11), за 12 месяцев затраты на электроэнергию при потребляемой мощности компьютера $Р_В = 0,3$ кВт составят (стоимость электроэнергии $с_Э = 4,64$ руб./ кВт с учетом НДС).

$$З_Э = 0,3 \times 388,83 \times 4,64 \times 12 = 6495,02 \text{ (руб.)}$$

Балансовая стоимость вычислительной техники $К_В$= 23000,00 руб. Тогда, на основании формулы (4.12), за 12 месяцев затраты на техническое обслуживание и текущий ремонт составят:

$$З_П = 23000 \times \frac{4}{100} \times \frac{388,83}{1970} \times 12 = 2179,08 \text{ (руб.)}$$

Затраты на амортизацию вычислительной техники по формуле (4.13) составят.

$$З_{АО} = 23000 \times \frac{33,3}{100} \times \frac{388,83}{1970} \times 12 = 18140,28 \text{ (руб.)}$$

Тогда, эксплуатационные затраты при использовании программного продукта составят:

$$З_{исп.} = З_{Э} + З_{П} + З_{АО} = 6495,02 + 2179,08 + 18140,28 = 26814,38 \text{ (руб.)}$$

Прибыль от использования программного продукта за год рассчитаем по формуле (4.15):

$$П = Э - З_{исп} = 75000 - 26814,38 = 48185,62 \text{ (руб.)}$$

Таким образом, имеем следующий денежный поток:

0 шаг (капиталовложения) – 56243,28 руб.;

1 шаг – 48185,62 руб.;

2 шаг – 48185,62 руб.;

3 шаг – 48185,62 руб.

Чистый дисконтированный доход (ЧДД, руб.), от использования программного продукта определим по формуле:

$$ЧДД = \sum_{k=1}^{N} \frac{П_k}{\left(1+\frac{E}{100}\right)^k} - K \quad (4.19)$$

где N – расчетный период, год;

$П_k$ – прибыль от использования программного продукта за k-й год его эксплуатации, руб.;

E – норма дисконта, %;

K – капиталовложения при внедрении программного продукта, руб.

Следовательно, ЧДД, руб., при N = 3, т. е. за три года использования программного продукта (срок до морального старения рассматриваемой информационной подсистемы) при норме дисконта E = 7,25% в соответствии с формулой (4.19) составит:

$$ЧДД = \frac{48185,62}{1+0,0725} + \frac{48185,62}{(1+0,0725)^2} + \frac{48185,62}{(1+0,0725)^3} - 56243,28 = 44928,317 +$$

+ 41891,2047 + 39059,3983 - 56243,28 = 439635,64 (руб.)

Приходим к выводу, что ЧДД − положителен, т. е. проект эффективен.

Внутреннюю норму доходности проекта ($E_{вн}$, %), определим по формуле:

$$E_{ВН} = E_{ВН.\ MAX+} + ЧДД \vee \frac{¿_{E_{ВН.MAX}+¿}}{ЧДД \vee ¿_{E_{ВН.MAX+¿}} ЧДД \vee ¿_{E_{ВН.MIN-¿}} ¿¿¿} ¿¿$$

(4.20)

где $E_{ВН.\ MAX\ +}$ – максимальное значение внутренней нормы дисконта, %, при которой ЧДД является положительной величиной (ЧДД > 0);

$E_{ВН.\ MIN}$ - – минимальное значение внутренней нормы дисконта, %, при которой ЧДД является отрицательной величиной (ЧДД < 0);

$ЧДД \vee ¿_{E_{ВН.MAX+¿}¿}$ – ЧДД, руб., вычисленный по формуле (4.19) при подстановке нормы дисконта Е = $ЧДД \vee ¿_{E_{ВН.MAX+¿}¿}$;

$ЧДД \vee ¿_{E_{ВН.MIN-¿}¿}$ – ЧДД, руб., вычисленный по формуле (4.19) при подстановке нормы дисконта Е = $ЧДД \vee ¿_{E_{ВН.MIN-¿}¿}$.

Предполагаем, что $E_{ВН}$ лежит в диапазоне 67...68 %. При норме дисконта $E_{ВН}$ = 67% получаем ЧДД = 233,91 руб. Таким образом, при норме дисконта $E_{ВН}$ = 67% ЧДД – положителен.

При норме дисконта $E_{ВН}$ = 68% получаем ЧДД = – 326,55 руб. Таким образом, при норме дисконта $E_{ВН}$ = 68% ЧДД – отрицателен.

Следовательно, по формуле (4.20) имеем:

$$E_{ВН} = 67 + \frac{233,91}{233,91-(-326,55)} (68 - 67) = 67,4173536\%$$

Рассчитаем срок окупаемости проекта. Срок окупаемости проекта $T_{ок}$, год, найдем по формуле:

$$T_{ок} = N + \frac{\sum_{j=1}^{N+1} Э_j - \sum_{j=1}^{N} Э_j}{Э_N}, \quad (4.21)$$

где N – максимальное количество лет, прошедших с начала эксплуатации программного продукта, в течение которых величина дохода от его использования не превысила величины капиталовложения при внедрении программного продукта;

$Э_j$ – величины приведенных (дисконтированных) годовых эффектов за j-й год, руб., прошедший с начала эксплуатации программного продукта, вычисленные по формуле (4.19) при подстановке нормы дисконта E = 7.25%.

Величина приведенного (дисконтированного) годового эффекта за первый год расчетного периода по формуле (4.19) равна:

$$Э_1 = \frac{48185,62}{1+0,0725} = 44928,317 \text{ (руб.)},$$

что меньше величины капиталовложений (K = 56243,28 руб.).

Величина приведенного (дисконтированного) годового эффекта за второй год расчетного периода по формуле (4.19) равна:

$$Э_2 = \frac{48185,62}{(1+0,0725)^2} = 41891,2047 \text{ (руб.)},$$

Сумма приведенных (дисконтированных) годовых эффектов за первый и второй год расчетного периода составят:

$$44928,317 + 41891,2047 = 86819,5217 \text{ (руб.)}$$

Тогда, в формуле (4.21) имеем N = 1 и срок окупаемости составит

$$T_{ок} = 1 + \frac{56243,28 - 44928,317}{41891,2047} = 1,27 \text{ (года)}$$

4.3 Основные экономические показатели проекта

Для удобства анализа, все основные технико-экономические показатели проекта сведены в таблицу 4.5.

Таблица 4.5 − Основные экономические показатели проекта

Основные характеристики	Единицы измерения	Проект
1	2	3
Итоговая трудоемкость разработки	чел.ч	941,63
Полные затраты на создание программного продукта	руб.	48907,20
Оптовая цена программного продукта	руб.	56243,28
Годовой экономический эффект от внедрения программного продукта	руб.	75000
Чистый дисконтированный доход	руб.	439635,64

Продолжение таблицы 4.5

1	2	3
Внутренняя норма доходности	%	67,4173536
Срок окупаемости проекта	год	1,27

Выводы

Целью деятельности любого предприятия является эффективное функционирование как с точки зрения рационального использования ресурсов, так и с точки зрения неуклонного ускорения научно-технического

процесса и полного удовлетворения нужд производителей и потребителей. Поэтому в настоящее время предъявляются повышенные требования к уровню экономического образования специалистов, к их умению применять теоретические знания на практике.

В ходе выполнения четвертой главы было проведено технико-экономическое обоснование проведенной работы.

Был рассчитан показатель экономической эффективности, а также показатель окупаемости, видно, что срок окупаемости проекта составляет чуть больше года. Небольшой срок окупаемости проекта доказывает, что разработка системы управления проектами, является экономически обоснованной и эффективной.

5 РАЗДЕЛ БЕЗОПАСНОСТИ И ЭКОЛОГИЧНОСТИ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Важным моментом в комплексе мероприятий направленных на совершенствование условий труда являются мероприятия по охране труда. Имеющийся в настоящее время комплекс разработанных организационных мероприятий и технических средств защиты, накопленный опыт работы ряда вычислительных центров показывает, что имеется возможность добиться значительно больших успехов в деле устранения воздействия на работающих опасных и вредных производственных факторах.

В рамках реализации разработки модуля задействовано оборудование, входящее в состав автоматизированного рабочего места администратора по информационной безопасности:

- компьютер;
- программное обеспечение.

Согласно ГОСТ 12.0.003-74 (СТ СЭВ 790-77) «ССБТ» [5]. Опасные и вредные производственные факторы. Классификация при эксплуатации вычислительных машин можно выделить четыре типа опасных и вредных факторов: физические, химические, биологические и психофизиологические.

При работе с ЭВМ в основном сталкиваются с физическими и психофизиологическими опасными и вредными производственными факторами, которые будут

рассмотрены ниже. Биологические и химические опасные и вредные факторы при таком виде работ не встречаются.

Для создания комфортных условий труда на рабочем месте с ЭВМ необходимо обеспечить санитарно-гигиенические, технические, эргономические, медико-профилактические требования в соответствии с СанПиН 2.2.2 / 2.4.1340–03 [6].

Можно выделить основные требования:

- оптимальный выбор эргономических параметров мониторов;
- организация рабочего места пользователя ЭВМ для обеспечения электромагнитной безопасности и требований эргономики;
- оценка естественного или искусственного освещения;
- профилактика по снижению влияния психофизиологических факторов;
- профилактика зрительного утомления пользователя ЭВМ;
- обеспечение аэроионного режима на рабочем месте;
- обеспечение пожаробезопасности в помещении и т.д.

Конструкция компьютера, дизайн и совокупность эргономических параметров должны обеспечивать надежное и комфортное считывание отображаемой информации в условиях эксплуатации:

- конструкция компьютера должна обеспечивать возможность поворота корпуса в горизонтальной и

вертикальной плоскостях с фиксацией в заданном положении для обеспечения фронтального наблюдения экрана;

- дизайн компьютера должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света, т.е. рассеиванием света во всевозможных направлениях;

- корпус компьютера должен иметь матовую поверхность с коэффициентом отражения (0,4–0,6) и не иметь блестящих деталей, способных создавать блики;

- конструкция компьютера должна предусматривать регулирование яркости и контрастности.

Помещения для эксплуатации компьютеров должны иметь естественное и искусственное освещение, но оно не должно создавать бликов на поверхности экрана. Для обеспечения нормируемых значений освещенности в помещениях для использования компьютеров следует проводить чистку стекол оконных рам и светильников со своевременной заменой перегоревших ламп. Так же рабочая зона должна удовлетворять нормам СанПиН 2.2.2.542-03, предусматривающей не менее 6 м^2 свободной от оборудования площади на одного человека. Оптимальные и допустимые микроклиматические параметры должны учитывать специфику технологического процесса в отделе радиоконтроля, в частности, условия по обеспечению надежной работы оборудования АРМ. Согласно требованиям СанПиН 2.2.2/2.4.1340-03 [6] в помещениях с компьютерами, должна проводиться ежедневная влажная уборка и систематическое

проветривание после каждого часа работы с компьютерами.

Для ЭВМ уровень шума, согласно СанПиН 2.2.2/2.4.1340-03, должна не превышать50 дБ. Клавиатура ноутбука должны быть расположены на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю. Расстояния от глаз оператора до монитора должно составлять 500-600 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Рабочее место представляет собой целостную единицу производства, все его элементы должны соответствовать антропометрическим, гигиеническим, физическим и психологическим требованиям. Немалое значение имеет и характер работы.

К гигиеническим требованиям, можно отнести показатели, которые определяют условия жизнедеятельности и работоспособности оператора: температура и влажность; уровень освещения, вибрации или шума и т.д.

К антропометрическим требованиям, показатели для соответствия строения оборудования характеристикам человека: возможность свободного движения; поза трудящегося.

К физическим и психологическим относится восприятие и переработка информации [30].

Под организацией рабочего места понимается проведение системы мероприятий по его оборудованию способами, предметом труда и размещения их в определенном порядке с целью достижения:

- оптимизации условий деятельности;
- безопасности труда;
- максимальной эффективности;
- комфортности работы человека.

К рабочему месту предъявляются следующие требования*:*

- оператор должен иметь рабочее пространство, которое позволило бы ему делать необходимые движения и перемещения;
- достаточные физические, зрительные и слуховые связи между человеком и оборудованием, а также между людьми во время выполнения общей задачи;
- необходимый уровень освещения;
- допустимые уровни шума, вибрации;
- наличие необходимых способов защиты;
- оптимальное размещение рабочих мест, а также безопасные и достаточные проходы для работающих [31].

Эргономическими аспектами проектирования рабочих мест также являются: высота рабочей поверхности, размеры пространства для ног, требования к расположению документов на рабочем месте, характеристики рабочего кресла, требования к поверхности рабочего стола, регулируемость элементов рабочего места. Главными элементами рабочего места являются стол и кресло. Основным рабочим положением является положение сидя.

Рабочая поза сидя вызывает минимальное утомление оператора ПК. Рациональная планировка рабочего места

предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации.

Важно так же периодически прерывать работу за экраном монитора на регламентированные перерывы, которые устанавливаются для обеспечения работоспособности и сохранения здоровья, или заменять другой работой с целью сокращения рабочей нагрузки у экрана. А так же во время регламентированных перерывов с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития статического утомления необходимо выполнять физические упражнения и упражнения для глаз.

Выводы

В данном разделе описаны основные нормы и требования по СанПИНу при работе с компьютером. Также описана организация рабочего места, что она из себя представляет, какие требования, учтены основные нормы освещения, микроклимат помещения, допустимый уровень шума, площадь помещения, создание благоприятных условий в помещении и т.д. Созданные условия должны обеспечивать комфортную работу.

Соблюдение условий, определяющих оптимальную организацию рабочего места оператора ПК, позволит сохранить хорошую работоспособность в течение всего рабочего дня, повысит как в количественном, так и в качественном отношениях производительность труда оператора ПК.

ЗАКЛЮЧЕНИЕ

В результате дипломной работы были выявлены, сформированы и выполнены цели и задачи исследования описанные в введении, для систематизации и централизованной работы приложения системы управления проектами, были раскрыты понятия и требования к созданию приложения, описан метод оценки экономической эффективности, спроектирована структура и разработана система управлениями проектами. Была так же проанализированная и представлена организационная структура ООО «МИРТЕК», рассмотрены существующие разработки системы управления проектами и определена проблемная ситуация при анализе предприятия. Так же была рассмотрена российская и зарубежная научная литература по теме исследования.

Был осуществлен сравнительный анализ моделей, методов и алгоритмов системы управления проектами, на основе которого был выбран интерфейс API, так как он является самым быстрым для представления данных. Учитывая специфику разрабатываемого программного модуля, было решено использовать фреймворк ASP.net.

Одним из важных разделов дипломной работы это расчет затрат на разработку в ходе которой был рассчитан показатель экономической эффективности, а также показатель окупаемости, который доказывает, что разработка системы управления проектами, является экономически обоснованной и эффективной, так как срок окупаемости составит 1 год и 27 дней. Для удобства работы

все основные технико-экономические показатели проекта приведены в таблицу 4.5.

Были расписаны нормы и требования по СанПИНу при работе с компьютером и организации рабочего места для комфорного условия работы.

Из всего выше сказанного можно сделать вывод, что исследование и разработка системы управления проектами для предприятия является экономически обоснованной и эффективной. И отношение к системе управлению проектов, как конкурентному преимуществу становится мировой практикой, ведь внедрение такой системы является важной стратегией и критическим фактором успеха, так как качественный сервис значительно увеличивает уровень производительности команды что, в свою очередь, влияет на прибыльность предприятий за счет контроля проектов по срокам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.Федеральный государственный образовательный стандарт высшего образования по направлению подготовки 09.04.03 – Прикладная информатика (квалификация (степень) «магистр»). – Утв. 2014–10–30. – М., 2014.

2.ГОСТ 7.9–95. Реферат и аннотация. Общие требования. – Введ. 1997–07–01. – М. : Изд-во стандартов, 1996. – 7 с. – (Система стандартов по информатизации, библиотечному и издательскому делу).

3.ГОСТ 7.11–2004. Сокращение слов и словосочетаний на иностранных европейских языках в библиографическом описании. – Введ. 2005–09–01. – М. : Стандартинформ, 2005. – 82 с. – (Система стандартов по информатизации, библиотечному и издательскому делу).

4.ГОСТ 7.12–93. Сокращение слов на русском языке. Общие требования и правила. – Введ. 1995–07–01. – М. : Изд-во стандартов, 1995. – 18 с. – (Система стандартов по информатизации, библиотечному и издательскому делу).

5. ГОСТ 12.0.003-74 Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы. Классификация (с Изменением N 1) Режим доступа: http://docs.cntd.ru/document/5200224 (дата обращения: 10.05.2020).

6. СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работ Режим доступа: https://ceut.ru/sanpin-2-2-2-2-4-1340-03/ (дата обращения: 10.05.2020).

7.Статья 79.1. Независимая оценка качества условий оказания услуг организациq [Электронный ресурс]. Режим доступа: http://base.garant.ru/12191967/c4f5bccbbdb9e1196efc46ad1b72a13e/ (дата обращения: 25.05.2020).

8.Кузин Ф. А. Магистерская диссертация. Методика написания, правила оформления и процедура защиты: практ. пособие для магистров-магистрантов. – М. : Ось – 89, 1997. – 304 с.

9.Алексеев Ю.В., Казачинский В.П., Никитина Н.С. Научно-исследовательские работы. Курсовые, дипломные, диссертации. Общая методология, методика подготовки и оформления: Учебное пособие. – Москва, 2006.

10. Волков Ю.Г. Диссертация: Подготовка, защита, оформление: Практ. пособие. - М.: Гардарики, 2004. – 185 с.

11. Райзберг Б.А. Диссертация и ученая степень: Пособие для соискателей. – М.: ИНФРА-М, 2008.

12. Стрельникова А. Г. Правила оформления диссертации: Пособие для аспирантов и соискателей. – Москва, 2009.

13. Князева В.В. Как работать над диссертацией и защищать ее: Практ. советы с точки зрения соискателя и эксперта. – Оренбург: ОГПУ, 2002.

14. Сстемный анализ [Электронный ресурс]. Режим доступа: http://systems-analysis.ru/systems_analysis.html (дата обращения: 22.04.2020).

15. Колесникова Н.И. От конспекта до диссертации: Учебное пособие по развитию навыков письменной речи. – М.: Флинта, 2003. – 288 с.

16. Марьянович А.Т. Эрратология или как избежать наиболее неприятных ошибок при подготовке диссертации. – 3-е изд., испр. – М.: Вуз. кн., 2001.

17. Новиков А.М. Как работать над диссертацией? – Москва, 2003.

18. Серова Г.А. Компьютер – помощник в оформлении диссертации: Практ. руководство для тех, кто хочет быстро научиться работать на компьютере. – М.: Финансы и статистика, 2002.

19. Шаршунов В.А., Гулько Н.В. Как подготовить и защитить диссертацию: История, опыт, методика и рекомендации. – М.: УП «Техноприн», 2003, 459 с.

20. Ярская В.Н. Методология диссертационного исследования: В помощь соискателю. – Саратов, 2000.

21. . Kohonen T. “The self-organizing map”, Proceedings of the Institute of Electrical and Electronics, 1990, vol. 78, p. 1464 – 1480

22. Nordstrom T. Designing parallel computers for self-organizing maps. Forth Swedish Workshop on Computer System Architecture, Linkoping, 1992. [Труды конференции]

23. Дэвид Макфарланд Большая книга CSS3 - М.:Питерское Издательство, 2016

24. ASP.NET [Электронный ресурс]. Режим доступа: https://dotnet.microsoft.com/apps/aspnet (дата обращения: 22.04.2020).

25. Лекция 5 Стандарт SOAP [Электронный ресурс]. Режим доступа: http://khpi-iip.mipk.kharkiv.edu/library/sotii/lectures/Lecture5.pdf (дата обращения: 22.04.2020).

26. API (Application Programming Interface) [Электронный ресурс]. Режим доступа: https://ru.bmstu.wiki/API_(Application_Programming_Interface) (дата обращения: 22.04.2020).

27. Об утверждении типовых отраслевых норм времени на выполнение работ, связанных с посещением одним пациентом врача-кардиолога, врача-эндокринолога, врача-стоматолога-терапевта: приказ Министерства здравоохранения Российской Федерации (Минздрав России) от 19 декабря 2016 г. [Электронный ресурс]. Режим доступа: https://minjust.consultant.ru/documents/22243 (дата обращения: 22.04.2020).

28. Эргономические требования к организации рабочего мечта [Электронный ресурс]. Режим доступа: https://spmag.ru/articles/pravila-organizacii-rabochego-mesta (дата обращения: 03.05.2020).

29. Основная цель и предмет эргономики. [Электронный ресурс]. Режим доступа: https://studopedia.su/13_92139_ergonomicheskie-trebovaniya-k-organizatsii-rabochego-mesta.html (дата обращения: 12.05.2020).

30. Методы системного анализа и синтеза [Электронный ресурс]. Режим доступа: https://all-sci.net/sistem-upravleniya-issledovanie/metodyi-sistemnogo-analiza-222842.html (дата обращения: 21.05.2020).

31. Понятие системы. Классификация систем [Электронный ресурс]. Режим доступа: https://studfiles.net/preview/7152779/ (дата обращения: 21.05.2020).

32. Системы, сущность и свойства [Электронный ресурс]. Режим доступа: https://creativeconomy.ru/lib/9298 (дата обращения: 30.06.2020).

33. Организация, как система [Электронный ресурс]. Режим доступа: https://studopedia.org/8-105291.html (дата обращения: 21.05.2020).

Приложение А

Создание сущности системы управления проектами

Листинг 1. Описание класса сущности «Проект»

```
using System;
using System.Collections.Generic;

namespace ProjectsApi.Domain.Entities
{
    /// <summary>
    /// Сущность объекта ПРОЕКТ
    /// </summary>
    public class Project
    {
        /// <summary>
        /// ID проекта
        /// </summary>
        public int ID { get; set; }

        /// <summary>
        /// Название проекта
        /// </summary>
        public string ProjectName { get; set; }

        /// <summary>
        /// Описание проекта
        /// </summary>
        public string ProjectDescr { get; set; }

        /// <summary>
        /// ID пользователя, создателя проекта
        /// </summary>
        public int CreatorID { get; set; }

        /// <summary>
        /// ФИО создателя проекта в виде Фамилия И.О.
        /// </summary>
        public string CreatorUserName { get; set; }

        /// <summary>
        /// ID пользователя, ответственного за проекта
        /// </summary>
        public int ManagerID { get; set; }

        /// <summary>
        /// ФИО ответственного за проекта в виде Фамилия И.О.
        /// </summary>
        public string ManagerUserName { get; set; }

        /// <summary>
```

```
        /// Дата начала проекта
        /// </summary>
        public DateTime? DateStart { get; set; }

        /// <summary>
        /// Строковое представление даты начала проекта в виде
dd.mm.yyyy
        /// </summary>
        public string DateStartStr { get; set; }

        /// <summary>
        /// Дата окончания проекта
        /// </summary>
        public DateTime? DateFinish { get; set; }

        /// <summary>
        /// Строковое представление даты окончания в виде
dd.mm.yyyy
        /// </summary>
        public string DateFinishStr { get; set; }

        /// <summary>
        /// ID статуса проекта
        /// </summary>
        public int StatusID { get; set; }

        /// <summary>
        /// ID группы проекта
        /// </summary>
        public int GroupID { get; set; }

        public List<int> GroupListIds { get; set; }

        /// <summary>
        /// ID родительского проекта
        /// </summary>
        public int ParentID { get; set; }

        /// <summary>
        /// Уровень вложености сущности в дереве проектов
        /// </summary>
        public int Level { get; set; }

        /// <summary>
        /// Принадлежность к проекту, у которого Level = 0
        /// </summary>
        public int Parent { get; set; }

        /// <summary>
        /// Название родительского проекта
        /// </summary>
```

```
        public string ParentName { get; set; }

        /// <summary>
        /// Количество дочерних записей
        /// </summary>
        public int CountChilds { get; set; }

        /// <summary>
        /// В массиве проектов имеются проекты с уровнем
вложенности равной 1
        /// </summary>
        public bool IsHasChilds { get; set; }

        /// <summary>
        /// Положение отображения потомков проекта
        /// </summary>
        public bool IsChildsView { get; set; }

        /// <summary>
        /// Флаг - проект помещен в корзину
        /// </summary>
        public bool IsTrash { get; set; }

        /// <summary>
        /// Флаг -проект является шаблоном
        /// </summary>
        public bool IsTemplate { get; set; }

        /// <summary>
        /// Количество задач в проекте всего
        /// </summary>
        public int CountTaskAll { get; set; }

        /// <summary>
        /// Количество выполненных задач
        /// </summary>
        public int CountTaskDone { get; set; }

        /// <summary>
        /// Количество заверешнных задач
        /// </summary>
        public int CountTaskComplete { get; set; }

        /// <summary>
        /// Количество оставшихся в работе задач
        /// </summary>
        public int CountTaskDiff { get; set; }

        /// <summary>
        /// Количество проектов всего
        /// </summary>
```

```
        //public int ProjectsAll { get; set; }

        /// <summary>
        /// Количество проектов с ролью "Участник"
        /// </summary>
        //public int ProjectsUser { get; set; }

        /// <summary>
        /// Количество проектов с ролью "Ответственный"
        /// </summary>
        //public int ProjectsCreate { get; set; }

        /// <summary>
        /// Количество проектов помещенных в Корзине
        /// </summary>
        //public int ProjectsTrash { get; set; }

        /// <summary>
        /// Проверка для корзины условия: Родительский проект
не находится в корзине, а дочерний - находится, если да, то
поле примет значение False
        /// </summary>
        public bool CheckTrashParent { get; set; }

        /// <summary>
        /// Дни после даты завершения проекта
        /// </summary>
        public int DayCountExp { get; set; }

        /// <summary>
        /// Количество выполненных задач
        /// </summary>
        public int CountTaskClose { get; set; }

        /// <summary>
        /// Количество задач в работе
        /// </summary>
        public int CountTaskInWork { get; set; }

        /// <summary>
        /// Количество не начатых задач
        /// </summary>
        public int CountTaskNoWork { get; set; }

        /// <summary>
        /// Количество  all задач
        /// </summary>
        public int CountTaskAlls { get; set; }
        /// <summary>
        /// Порядковый номер проекта
        /// </summary>
```

```
        public int Counter { get; set; }

        /// <summary>
        /// Выбран для перемещения как проект-приемник
        /// </summary>
        public bool RelocationSelected { get; set; }
    }
}
```

Приложение Б

Метод обработки запроса

Листинг 1. Описание метода обработки запроса для получения списка всех доступных проектов

```
/// <summary>
        /// Получение списка всех доступных проектов
        /// </summary>
        /// <returns></returns>
        [HttpGet]
        [Route("projects")]
        public IActionResult GetProjects()
        {
            try
            {
                _authvalid.Check(HttpContext);

                if ((_authvalid.RoleId == 1 ||
_authvalid.RoleId == 4 || _authvalid.RoleId == 6) &&
_authvalid.UserId > 0)
                {
                    if (_listUsers.Count() == 0)
                    {
                        _listUsers =
_userRepository.GetUsers(HttpContext);

                        if (_listUsers == null)
                        {
                            return Unauthorized();
                        }
                    }

                    var projects =
_projectRepository.GetProjects(_authvalid.UserId,
_authvalid.RoleId).ToList();
                    var counter =
_projectRepository.GetCountProjectsType(_authvalid.UserId,
_authvalid.RoleId);
                    int z = 1;

                    for (int j = 0; j < projects.Count; j++)
                    {
                        var user =
_listUsers.FirstOrDefault(el => el.Id ==
projects[j].ManagerID);

                        if (user != null)
                        {
```

```
                            projects[j].ManagerUserName =
user.Surname + " " + user.Name.Substring(0, 1) + "." +
user.Patronymic.Substring(0, 1) + ".";
                        }

                        projects[j].Counter = z;
                        z++;
                    }

                    var result = new ProjectDTO();
                    result.Projects = projects;

                    result.ProjectsAll = counter.ProjectsAll;
                    result.ProjectsCreate =
counter.ProjectsCreate;
                    result.ProjectsUser =
counter.ProjectsUser;
                    result.ProjectsTrash =
counter.ProjectsTrash;

                    result.UserId = _authvalid.UserId;
                    result.RoleId = _authvalid.RoleId;

                    if (projects.Count() > 0 &&
projects.Where(el => el.Level > 0).Count() > 0 &&
projects.Where(el => el.Level == 0).Count() > 0)
                    {
                        result.IsHasChilds = true;
                    }
                    else
                    {
                        result.IsHasChilds = false;
                    }

                    return Ok(result);
                }

                return Forbid();
            }
            catch (Exception ex)
            {
                _logger.LogError("ApplicationContorller",
"GetProjects", ex.Message, DateTime.Now, _authvalid.UserId);

                return BadRequest(ex.Message);
            }
        }
```