МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка информационной системы ведения проектов

Обучающийся Д. А. Ярнов _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд. техн. наук Д. Г. Токарев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема выпускной квалификационной работы «Разработка информационной системы ведения проектов».

Информационная система ведения проектов используется в IT-компании, специализирующейся на разработке и внедрении программного обеспечения.

В первой главе проведен бизнес-анализ процессов IT компании, на основе которого выполнено функциональное моделирование по методологии SADT (англ. structured analysis and design technique), нотации IDEF0 AS IS («Как есть»). Нотация IDEF0 позволяет выделить проблемные места процесса и спроектировать процессы в нотации IDEF0 TO BE («Как должно быть»). Описаны варианты использования с помощью диаграмм UML (англ. Unified Modeling Language – унифицированного языка моделирования).

Во второй главе выполнено концептуальное и логическое проектирование информационной системы ведения проектов. Описаны входящая и исходящая информация, с которой работают сотрудник ИТ-компании в процессах ведения проектов. Описывается проектирование информационной системы с помощью диаграмм UML.

В третьей главе описывается ИТ-архитектура системы ведения проектов и методы разработки. Выполнен обзор инструментов разработки и выбраны наиболее подходящие. Рассчитывается экономическая эффективность от разработки информационной системы ВП.

В заключении описываются результаты разработки информационной системы ведения проектов.

Информационная система ведения проектов проектируется как AGILE ориентированная информационная система для использования методологии Scrum и Канбан в разработке ПО.

Оглавление

Введение

Современный мир не может существовать без информационных технологий, и IT-индустрия становится все более значимой в различных сферах бизнеса и жизни людей. Компании в области информационных технологий нуждаются в эффективных инструментах для ведения бизнеса и управления проектами. В связи с этим, разработка информационной системы для ведения IT-проектов является важной задачей для компаний, работающих в этой сфере [2].

Современной методологией ведения проектов при разработке ПО является Scrum с использованием досок Канбан. Scrum и Канбан относится к гибким моделям разработки Agile [20]. Разрабатываемое ПО должно отвечать современным методологиям разработки ПО [9].

Информационная система ведения проектов позволит упростить и автоматизировать процессы, связанные с управлением проектами, сбором, хранением и обработкой данных, а также улучшить коммуникацию между участниками проекта и повысить эффективность работы. Важными элементами такой системы являются управление задачами, планирование, отслеживание времени, контроль версий, документирование процессов и многие другие функции.

Без использования специальных систем для управления IT-проектами, компании могут столкнуться с рядом проблем, таких как: неэффективное использование времени и ресурсов, потеря данных, недостаточная прозрачность и согласованность процессов, а также отсутствие возможности для точного прогнозирования времени и бюджета проекта. Информационная система для ведения IT-проектов поможет избежать этих проблем и повысить эффективность бизнес-процессов в компании [6].

Тема ВКР предложена кафедрой Прикладной математики и информатики ТГУ.

Задача ВКР сводится к разработке информационной системы ведения проектов для ИТ -компании занимающейся разработкой ПО.

Задачи при разработке информационной системы ведения проектов:

- изучение процессов ведения проектов в ИТ-компании;
- анализ бизнес–процессов;
- определить процесс, нуждающийся в реинжиниринге;
- концептуальное и логическое проектирование;
- разработка Agile ориентированной информационной системы.

ВКР включает введение, три главы, заключение, список использованных источников:

- В первой главе исследуются процессы ведения проектов.
- Во второй главе моделируется информационная система.
- В третьей главе описывается ИТ-архитектура разрабатываемой системы ведения проектов и методы разработки. Проводится анализ инструментов разработки и выбран наиболее подходящий. Разрабатывается информационная система и приводятся контрольные примеры. Проводится расчёт экономической эффективности внедрения информационной системы ведения проектов.

В заключении проведено описание решенных задач во время проектирования информационной системы ведения проектов.

Объект исследования – деятельность IT-компании, специализирующейся на разработке и внедрении программных продуктов.

Предмет исследования – автоматизация процесса ведения проектов IT-компании.

Цель исследования – разработка информационной системы ведения проектов.

Задачи исследования – анализ, моделирование, проектирование БД и программного комплекса используя нотации IDEF0, IDEF1X, UML. Выбор подходящих средств и языка программирования для разработки информационной системы [15].

Глава 1 Функциональное моделирование предметной области

1.1 Технико-экономическая характеристика предметной области

IT-компания специализируется на обследовании, проектировании и разработке программного обеспечения, баз данных и интерфейсов. Деятельность IT-компании может включать в себя следующие аспекты:

Разработка программного обеспечения: IT-компания может заниматься созданием программных продуктов для различных сфер бизнеса, включая веб-приложения, мобильные приложения, программы для настольных компьютеров и другие.

Интеграция систем и сервисов: IT-компания может заниматься интеграцией различных систем и сервисов для повышения эффективности бизнес-процессов в компаниях.

Консалтинг: аудит информационных систем и ИТ-инфраструктуры предприятия, а также разработку плана по перестройке ИТ-ландшафта или внедрению новых платформ и сервисов.

Техническая поддержка: IT-компания предоставляет аутсорсинг информационно-технологического сопровождения.

Исследования и разработки: IT-компания может заниматься исследованием и разработкой новых технологий и инновационных продуктов в области информационных технологий.

Продвижение и маркетинг: IT-компания может заниматься продвижением своих продуктов и услуг, а также проводить маркетинговые исследования для определения потребностей рынка и трендов.

Основные подразделения организационная структура IT-компании:

- руководство компании;
- отдел разработки;
- отдел технической поддержки;
- отдел управления проектами;

– финансовый отдел;

Тип организационной структуры относится к линейно-функциональной структуре.

На рисунке 1 представлена организационная структура.

Рисунок 1 – линейная организационная структура IT-компании

Каждое подразделение выполняет свои функции в деятельности компании.

Руководство компании: включает в себя генерального директора (CEO), который отвечает за общее управление компанией, принятие стратегических решений и развитие бизнеса.

Отдел продаж: занимается продажами и маркетингом, поиском новых клиентов и развитием бизнеса.

Отдел разработки: разрабатывает программные продукты.

Отдел технической поддержки: предоставляет информационно-технологическое сопровождение.

Отдел управления проектами: занимается планированием, координацией и управлением проектами, в том числе распределением ресурсов, управлением бюджетом и сроками.

Отдел качества: отвечает за контроль качества продуктов и услуг, тестирование и анализ результатов, а также разработку и внедрение методик и стандартов.

Отдел управления персоналом: занимается управлением кадрами, включая подбор и найм сотрудников, оценку и развитие персонала, а также создание политик и процедур по управлению персоналом.

Финансовый отдел: отвечает за управление финансами компании, включая учет и отчетность, бюджетирование, финансовый анализ и планирование.

Юридический отдел: занимается юридическими вопросами, связанными с бизнес-операциями компании, включая согласование договоров, защиту прав и интересов компании, обеспечение соответствия законодательству и т.д [1].

1.2 Концептуальное моделирование предметной области

При ведении проектов каждый этап может потребовать значительных усилий, времени и ресурсов. Например, планирование проекта может включать в себя определение целей, задач и сроков выполнения, а также назначение ответственных за каждую задачу. Это может потребовать значительного количества времени и нескольких встреч с участниками команды проекта.

Во время выполнения проекта участники команды могут работать над своими задачами вручную, что может занять много времени и не дать возможности для быстрого контроля процесса выполнения задач. Контроль выполнения проекта может включать в себя регулярные отчеты о состоянии проекта, которые могут быть подготовлены вручную, что также потребует много времени и усилий.

Наконец, завершение проекта требует создание отчета о выполнении проекта, проведение встреч с заказчиком для утверждения результата, а также

анализ проекта для выявления сильных и слабых сторон процесса ведения проекта. Это может занять много времени и ресурсов.

Концептуальное моделирование информационной системы — это первый этап процесса проектирования, на котором создаются общие представления о системе, ее структуре, функциональности и возможных ограничениях. На этом этапе определяются ключевые сущности и их связи, функциональные требования к системе, ее основные компоненты и интерфейсы между ними.

Концептуальное моделирование ИС имеет несколько целей:

Определение требований к системе: на этом этапе определяются функциональные и нефункциональные требования к системе, ее цели, ограничения и критерии качества. Это позволяет разработчикам определить, какие компоненты и технологии будут необходимы для создания ИС.

Создание общего представления о системе: на этом этапе создается общее представление о системе, ее структуре и связях между ее компонентами. Это позволяет лучше понимать потребности пользователей и участников проекта, а также определять ключевые компоненты ИС.

Минимизация рисков: концептуальное моделирование позволяет идентифицировать риски, связанные с проектом, и предпринимать меры для их минимизации. Например, на этом этапе можно определить, что некоторые компоненты ИС могут быть слишком сложными для реализации в заданный срок или бюджет.

Создание основы для дальнейшего проектирования: концептуальное моделирование создает основу для дальнейшего проектирования ИС. На его основе можно создавать более детальные модели, проводить анализы и тестирование, а также разрабатывать конкретные компоненты системы [1][15].

1.2.1 Выбор технологии концептуального моделирования предметной области

Рассмотрим несколько популярных технологий концептуального моделирования, которые могут быть использованы в проекте:

UML (Unified Modeling Language) — это язык, который позволяет описывать архитектуру и поведение системы в виде диаграмм. UML является стандартом индустрии и широко используется в проектировании различных типов систем.

ER-диаграммы (Entity-relationship diagrams) — это графическое представление сущностей и их отношений в базе данных. Они позволяют описывать структуру базы данных и отношения между таблицами.

IDEF0 (Integration Definition for Function Modeling) — это методология функционального моделирования, разработанная для формализации и описания функциональных аспектов системы. IDEF0 используется для создания функциональных блок-схем, которые описывают структуру процесса, его входные и выходные данные, а также потоки данных между блоками.

Для проведения бизнес-анализа информационной системы будет использоваться методология моделирования IDEF0.

Нотация IDEF0 — это графический язык моделирования процессов, который используется для анализа, проектирования и документирования бизнес-процессов и систем. Он основан на диаграммах функционального моделирования и позволяет описать, как функции взаимодействуют между собой, чтобы выполнять определенные цели.

IDEF0 представляет собой язык, который используется для создания диаграмм, называемых функциональными блок-схемами (Functional Flow Diagrams, FFDs). Каждый блок на диаграмме представляет собой функцию, которая может принимать входные данные, выполнять определенные

операции и выдавать выходные данные. Связи между блоками представляют собой потоки данных, которые передаются от одного блока к другому.

В целом, нотация IDEF0 позволяет моделировать процессы на различных уровнях детализации, начиная от общей картины и заканчивая детальными описаниями функций и потоков данных. Это помогает улучшить понимание процесса и идентифицировать возможные проблемы или улучшения в работе системы [19].

Основными элементами IDEF0 являются:

- функциональные блоки - представляют отдельные функции системы;
- входные и выходные потоки данных - представляют данные, которые вводятся в систему и данные, которые выводятся из системы;
- механизмы управления - представляют управляющие элементы, такие как решения и контрольные точки, которые используются для управления потоками данных;
- объекты - представляют объекты, которые используются в системе.

IDEF0 позволяет описывать бизнес-процессы на разных уровнях детализации, начиная от высокоуровневого описания общих целей и задач системы, и заканчивая детальным описанием операций и процедур. Это делает IDEF0 полезным инструментом для анализа и оптимизации бизнес-процессов в различных отраслях, таких как производство, ит-организации, здравоохранение и т.д [1].

1.2.2 Разработка и анализ модели бизнес-процесса "как есть"

При помощи моделирования в нотации IDEF0 рассмотрим процесс ведения проектов «AS IS (Как есть)» используя CASE (англ. computer-aided software engineering) инструмент программное обеспечение Ramus Educational. На рисунке 2 представлена диаграмма IDEFO (Как есть).

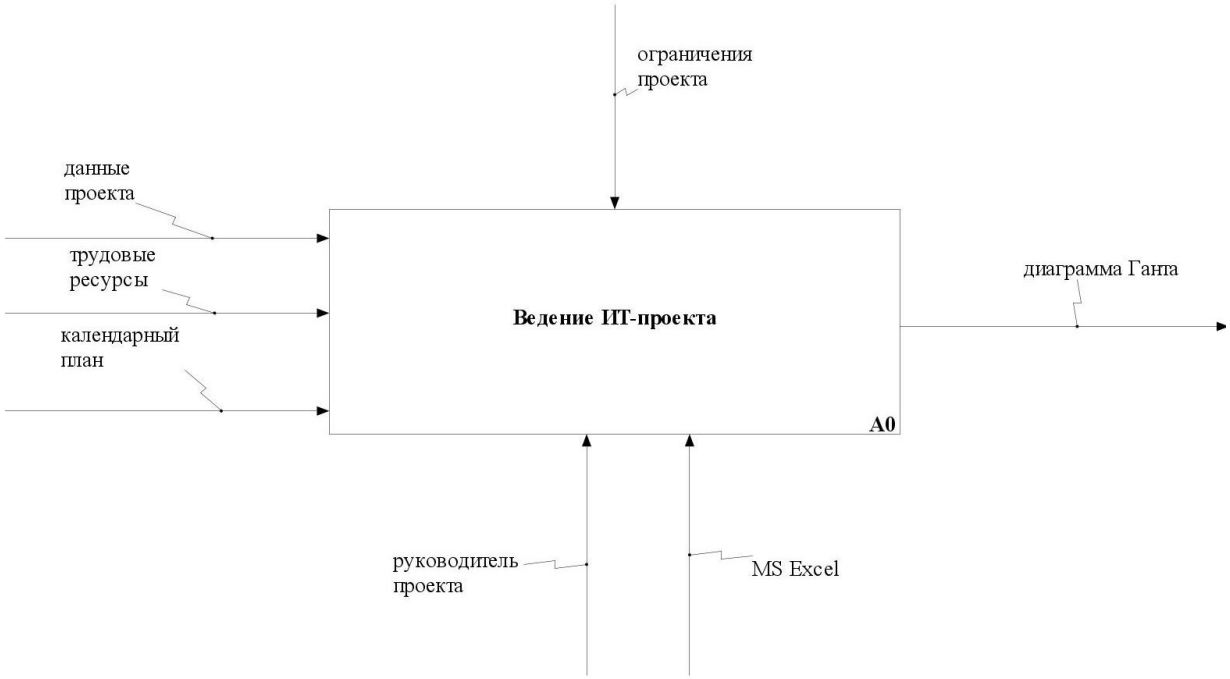

Рисунок 2 – Модель «AS IS (Как есть)» в нотации IDEF0

Далее выполняем декомпозицию процесса [16]. Необходимо:

- разбить проект на задачи;
- назначить ресурсы на задачи;
- опредлеить длительность каждой работы;
- построить график исполнения проекта.

Диаграмма в нотации IDEF0 «AS IS (Как есть)» декомпозиция процесса показана на рисунке 3.

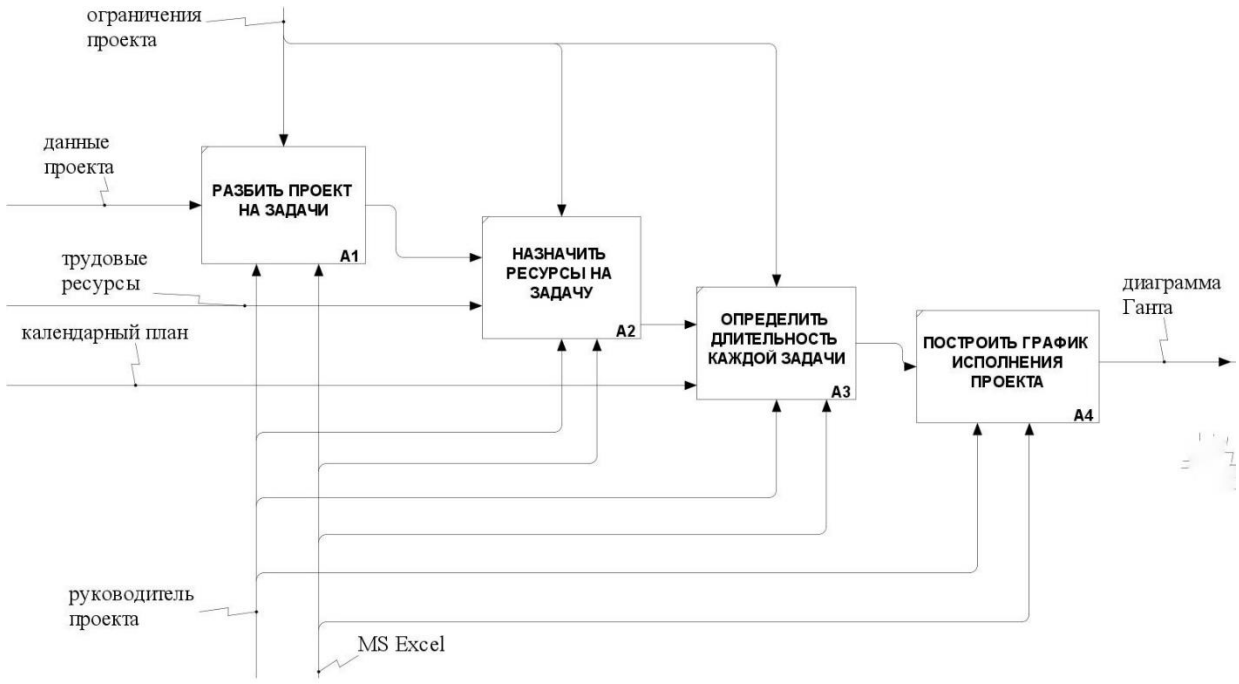

Рисунок 3 – Декомпозиция процесса «Как есть»

Декомпозиция процесса показывает, как происходит процесс ведения проектов.

- Разбить проект на задачи: этот этап начинается с идеи о проекте и заканчивается утверждением проекта. На этом этапе определяются цели и задачи проекта, определяется бюджет, оцениваются риски и выбирается команда проекта;
- Назначить ресурсы на задачу: на этом этапе проекта разрабатывается детальный план, включающий в себя задачи, сроки, бюджет, ресурсы и оценки рисков. В этом плане также определяются роли и обязанности каждого члена команды проекта, а также создаются планы управления качеством, коммуникацией и изменениями;
- Определить длительность каждой задачи: этот этап включает в себя фактическое выполнение работ по проекту в соответствии с планом;
- Построить график исполнения проекта: Диаграмма Ганта;
- Каждый этап управления IT проектом является важным и необходимым для успешного выполнения проекта. Важно уделить

достаточно времени и внимания каждому этапу, чтобы обеспечить качественное и своевременное выполнение проекта.

Процесс «ведение IT-проекта» происходит не всегда корректным образом, руководитель должен постоянно отслеживать выполнение каждой задачи согласно построенному графику. График исполнения проекта постоянно требуется перестраивать и назначать ресурсы по новой. Для ускорения этого принято решение автоматизировать его путем внедрения системы, которая будет автоматически планировать график выполнения проекта и выравнивать ресурсы, если какие-то окажутся загружены на работах, также система будет позволять руководителю проекта отслеживать проект на прогнозные даты, чтобы заранее предпринять меры при угрозе срыва сроков исполнения проекта или увеличении бюджета.

Чтобы проект был успешным и был выполнен в срок, необходимо руководителю проекта вести постоянный мониторинг и контроль исполнения проекта.

Проекты моделируются с помощью диаграммы Ганта в Excel, параллельно с этим выполняются ряд работ:

- Регулярные встречи с командой проекта: проведение ежедневных или еженедельных собраний с командой проекта, на которых обсуждаются текущие задачи, проблемы и прогресс выполнения работ. Это поможет выявлять проблемы вовремя и принимать меры по их решению;
- Ведение журнала проекта: создание документа, в котором отслеживаются все изменения, проблемы и решения, принятые в процессе выполнения проекта. Этот журнал может помочь понять, какие задачи были выполнены и сколько времени заняло их выполнение;
- Оценка прогресса выполнения задач: регулярная проверка выполнения задач и сравнение фактического прогресса с

планируемым. Если обнаруживается отставание от графика, можно принять меры для устранения этой проблемы;

- Контроль за бюджетом проекта: следить за расходами и доходами проекта, чтобы убедиться, что не превышается бюджет и есть достаточно средств для выполнения проекта [15][16].

1.2.3 Обоснование необходимости автоматизированного варианта решения

Автоматизация процесса управления проектами повысит эффективность работы команды и улучшить качество проекта:

- Слабости в процессе управления проектами могут включать: Недостаточная видимость процессов - когда команда проекта не имеет полной картины о том, какие задачи выполняются, кто отвечает за каждую задачу и какие ресурсы используются;
- Неполное использование инструментов - когда команда проекта не использует все возможности современных инструментов для управления проектами, таких как онлайн-коллаборационные платформы. Онлайн-коллаборационные платформы позволяют участникам проекта обмениваться информацией и документами, и работать вместе над проектом. Это улучшает коммуникацию и упрощает процесс совместной работы;
- Неэффективное планирование - когда команда проекта не может правильно распределить ресурсы и установить реалистичные сроки выполнения задач;
- Низкая коммуникация - когда команда проекта не общается достаточно часто и не учитывает мнения всех участников проекта;
- Системы управления проектами позволяют команде проекта просматривать задачи, назначать ответственных и контролировать выполнение задач.

Таким образом, автоматизация процесса управления проектами может помочь решить многие проблемы, связанные с неэффективным управлением проектами и повысить эффективность работы команды [8].

1.3 Анализ существующих разработок

Существует множество программ для ведения проектов. Некоторые из них имеют большой набор функций, другие - более простые и простые в использовании. Приведем некоторые из самых популярных программ для управления проектами и их функции:

Trello — это облачная программа ведения проектов с возможностью создания досок, на которых можно создавать карточки с задачами и перемещать их между списками для отслеживания прогресса проекта.

Функции:

- возможность прикрепления файлов;
- установки дедлайнов;
- комментирования карточек;
- создания чек-листов;
- настройки оповещений и т.д.

Достоинства:

- простота использования;
- бесплатность;
- возможность работы в команде;
- поддержка мобильных устройств.

Недостатки:

- ограниченный функционал;
- отсутствие функции гант-чарта;
- ограниченный доступ к карточкам для гостей.

Цена: бесплатно для базовой версии, от $6 в месяц за расширенную версию.

Microsoft Project — это платная программа ведения проектов с возможностью создания гант-чартов и управления ресурсами проекта.

Функции:

- создание гант-чартов;
- управление задачами и ресурсами;
- создание отчетов;
- интеграция с другими программами и т.д.

Достоинства:

- множество функций и возможностей;
- возможность работы в команде;
- интеграция с другими программами.

Недостатки:

- высокая цена;
- сложность настройки и использования;
- отсутствие возможности работы в облаке.

Цена: от $10 в месяц за базовую версию, от $30 в месяц за расширенную версию.

Jira — это программа управления проектами, разработанная компанией Atlassian. Она широко используется в IT-компаниях для управления разработкой программного обеспечения, но также может быть использована для управления проектами в других отраслях.

Функции Jira включают в себя:

- возможность создания задач и подзадач;
- назначения ответственных и сроков;
- создания досок для отслеживания прогресса проекта;
- создания отчетов и интеграцию с другими программами.

Достоинства Jira включают в себя:

- Функции включают планирование проекта, управление задачами, управление ресурсами, отслеживание прогресса, анализ данных и т.д.;
- интеграция с Confluence и Bitbucket.

Недостатки Jira включают в себя:

- сложность настройки и использования;
- высокая цена для расширенных версий;
- меньшая гибкость в сравнении с другими программами, такими как Trello.

Цена Jira начинается от $10 в месяц за базовую версию и может достигать $14,300 в год за расширенную версию для больших команд и организаций. Jira также предоставляется в виде облачного решения или как программное обеспечение для установки на сервер.

Эти программы имеют множество функций для управления проектами и задачами, что позволяет управлять проектами эффективно и организованно. Однако, выбор программы зависит от потребностей и бюджета, а также от того, какие функции наиболее важны. Если необходимо чтобы программа выполняла только те функции, которые нужны для управления проектом или расширить определенным функционалом, тогда требуется разработать свой программный продукт под свои нужды.

По рассмотренным системам можно с уверенностью сказать, что ни одна система не подходит для рассматриваемой компании, все системы являются зарубежными, что значительно затрудняет ее внедрение и дальнейшее сопровождение.

1.4 Постановка задачи на разработку информационной системы ведения проектов

Компания имеет несколько проектов в работе, каждый проект состоит из нескольких задач, которые должны быть выполнены в определенные сроки. В настоящее время используется Excel-таблицы и диаграммы Ганта для отслеживания прогресса проектов и задач, однако это неэффективно и занимает много времени. Создание системы позволит эффективно управлять каждым проектом и отслеживать прогресс. Система должна быть легко

настраиваемой и гибкой, чтобы сотрудники компании могли адаптировать ее к своим потребностям.

Функции системы должны включать:

- создание и управление задачами и подзадачами для каждого проекта;
- назначение ответственных за выполнение задач и установка сроков;
- создание графиков и диаграмм для отслеживания прогресса проектов;
- Доска Kanban для визуализации всех задач;
- Scrum шаблоны;
- отчеты о статусе проектов и задач;
- возможность прикреплять файлы и комментарии к задачам;
- интеграция с другими программами, такими как Google Calendar и MS Outlook.

Функциональные требования проекта определяют основные функции и возможности, которые должен обладать проект для того, чтобы соответствовать потребностям пользователей и целям заказчика. Они описывают, как система должна работать и какие действия и функции должна выполнять. Приведем примеры функциональных требований для проекта веб-приложения [18]:

- Регистрация пользователей: Пользователи должны иметь возможность зарегистрироваться в приложении и создать свой аккаунт. Данные, запрашиваемые при регистрации, должны включать в себя имя пользователя, электронную почту и пароль;
- Авторизация пользователей: Зарегистрированные пользователи должны иметь возможность входа в приложение, используя свой логин и пароль;
- Создание профиля пользователя: Пользователи должны иметь возможность создавать свой профиль в приложении, указывая информацию о себе, такую как имя, фотографии, контактные данные и т.д.;
- Создание проекта: Пользователь должен иметь возможность создать проект, назначить сроки и исполнителя;

− Создание задач: пользователь должен иметь возможность создать задачу и подзадачу;

− Просмотр списка задач: Пользователи должны иметь возможность просмотреть список задач, доступных в приложении;

− Доска Kanban: визуализация всех задач;

− Scrum шаблоны: бэклог задач;

− Администрирование приложения: Администраторы приложения должны иметь возможность управлять содержимым приложения, включая добавление, удаление и изменение задач, управление пользователями и т.д.

Целью разработки системы является упрощение управления проектами, повышение эффективности работы команды и улучшение контроля над выполнением задач. Также требуется, чтобы система была доступна для использования в любом месте и в любое время, что позволит быть более гибкими и быстрыми в принятии решений.

1.5 Модель бизнес-процесса «TO BE (Как должно быть)»

Бизнес-анализ модели «AS IS (Как есть)» выявляет недостатки в процессе ведения проекта, которые являются наиболее важным в деятельности компании и не допускает ошибок.

Проведем моделирование процессов с внедренной системой ведения проектов, которая поможет руководителю и другим участникам проекта вести проекты в компании более простым и надежным способом.

На рисунке 4 представлена диаграмма TO BE («Как должно быть»).

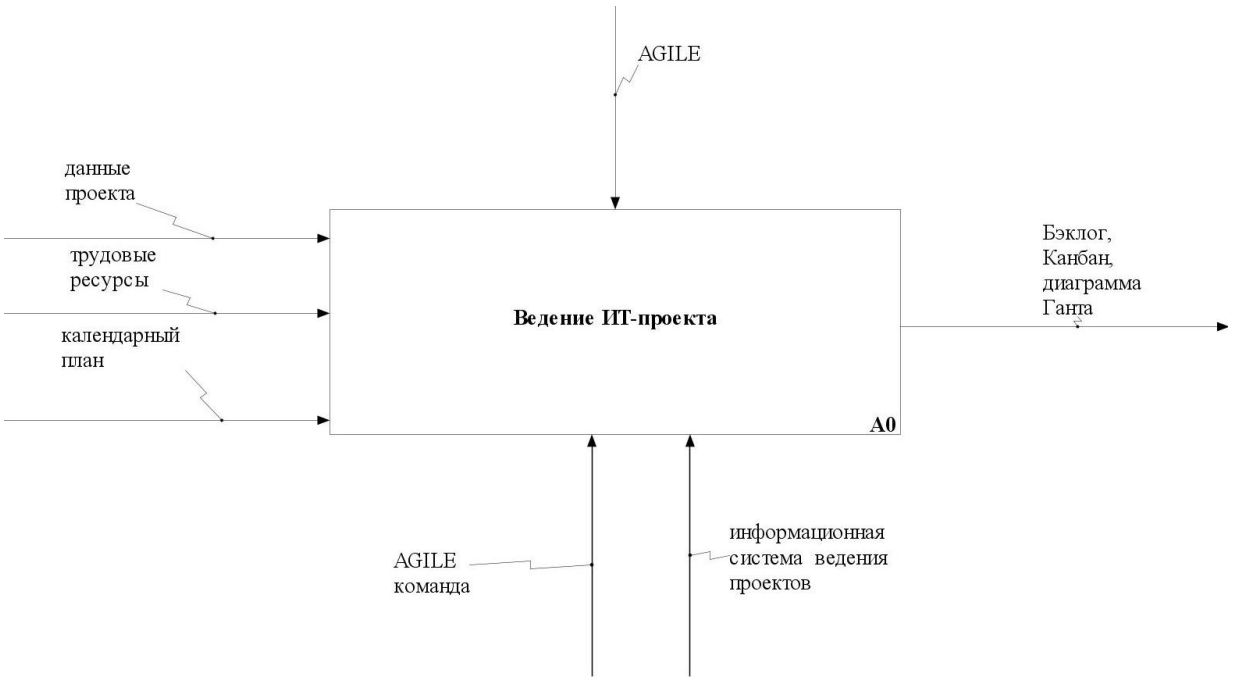

Рисунок 4 – Диаграмма TO BE («Как должно быть»)

В диаграмму включена разрабатываемая информационная система.

На рисунке 5 представлена декомпозиция TO BE («Как должно быть»).

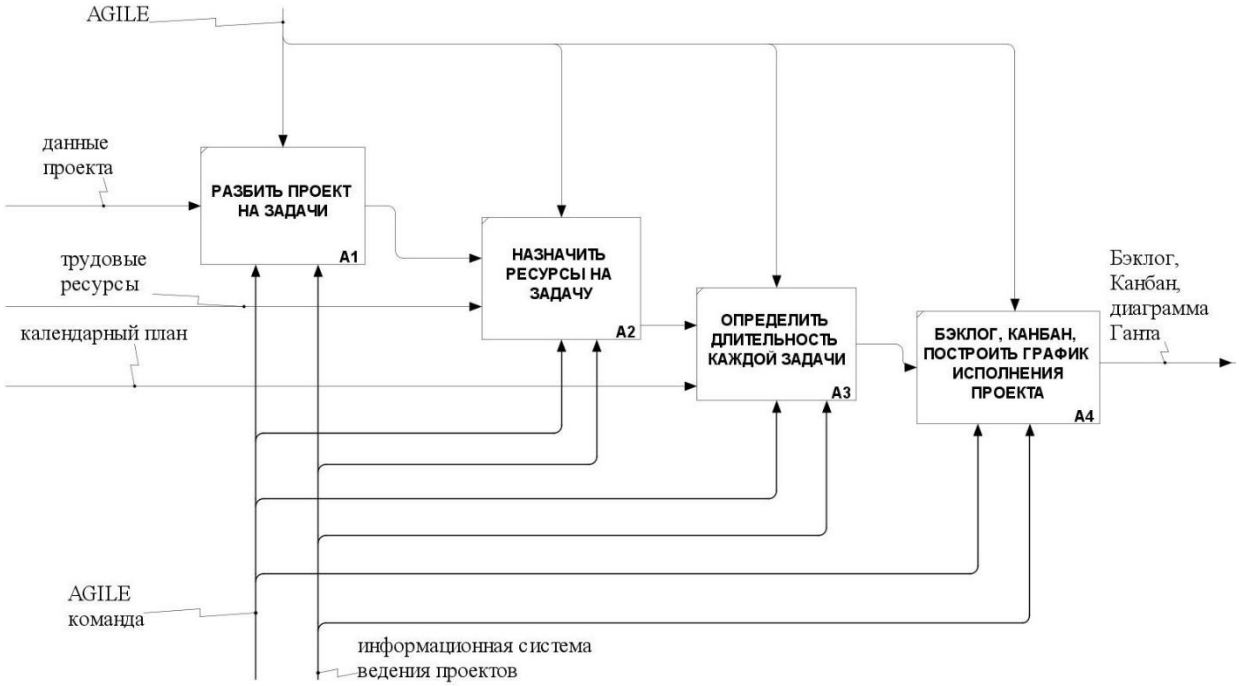

Рисунок 5 – Декомпозиция TO BE («Как должно быть»)

Слабое место в процессе управления проектами вызвано различными факторами, такими как неэффективная коммуникация, отсутствие стандартов и методологий управления проектами, неэффективное использование ресурсов и т.д.

Без автоматизации процесс ведения проекта может быть не только неэффективным, но и подвержен ошибкам и задержкам. Например, возможны ошибки при назначении ответственных за задачи, упущение важных сроков выполнения задач или проблемы в связи с недостаточной коммуникацией между участниками проекта.

В целом, автоматизация процесса ведения проектов может значительно упростить процесс и повысить эффективность команды проекта [15][16].

Выводы по главе 1

В первой главе работы проведен бизнес-анализ процессов разработки ПО в IT-компании, спроектированы диаграммы IDEF0 AS IS («Как есть») и TO BE («Как должно быть»). Для дальнейшего проектирования системы были рассмотрены аналогичные системы, представленные на рынке в готовом варианте, что позволило выявить, что все системы являются зарубежными что значительно затрудняет ее внедрение и дальнейшее сопровождение. На основе выделенных основных бизнес-процессов были выявлены слабые места и бизнес-процесс, который нуждается в реинжиниринге. Также выбран метод реинжиниринга, который будет применен на втором этапе, где будет осуществлено моделирование бизнес-процессов. Была разработана постановка задачи на разработку AGILE системы с методологией Scrum и Канбан.

Глава 2 Логическое проектирование ситсемы

2.1 Выбор технологии логического моделирования

Некоторые из популярных технологий логического моделирования включают в себя:

- ER-моделирование (англ. Entity-Relationship Modeling): это техника моделирования баз данных, которая используется для описания сущностей и их взаимоотношений;
- UML (англ. Unified Modeling Language): используется для проектировки программных систем и описания бизнес-процессов;
- DFD (Data Flow Diagrams): это техника моделирования, которая используется для описания потоков данных в системе.
- UML (Unified Modeling Language) - это стандартный язык моделирования, который позволяет разработчикам создавать графические модели различных типов систем, включая программное обеспечение, бизнес-процессы, аппаратное обеспечение и т.д. UML обладает множеством преимуществ, которые делают его одним из наиболее популярных языков моделирования:
- Универсальность: UML может использоваться для моделирования различных типов систем, что делает его универсальным языком для разработки различных проектов;
- Стандартизация: UML — это стандарт, что означает, что он широко распространен и поддерживается многими инструментами и библиотеками;
- Графический подход: UML позволяет использовать графические диаграммы для моделирования системы, что упрощает восприятие и понимание модели;
- Простота и понятность: большинство диаграмм UML имеют интуитивно понятное название, что упрощает понимание модели;

- Расширяемость: UML может быть расширен с помощью профилей, что позволяет создавать собственные элементы моделирования;
- Возможность автоматической генерации кода: многие инструменты для работы с UML позволяют автоматически генерировать код на основе модели, что упрощает и ускоряет процесс разработки.

UML включает в себя несколько видов диаграмм, включая:

- диаграммы классов: используются для описания структуры классов и их взаимоотношений в системе;
- диаграммы объектов: используются для описания объектов и их связей в системе;
- диаграммы случаев использования: используются для описания взаимодействия пользователей с системой;
- диаграммы активностей: используются для описания бизнес-процессов и действий, которые выполняет система;
- диаграммы последовательностей: используются для описания взаимодействия между объектами и компонентами системы во времени.

В работе будет использована технология моделирования UML, которая помогает достичь нескольких целей в процессе проектирования информационных систем. Вот некоторые из них:

- позволяет анализировать требования пользователя и создавать модели, которые являются понятными для разных участников проекта;
- предоставляет стандартный набор диаграмм, которые помогают анализировать и проектировать систему;
- обеспечивает стандартизированный язык для коммуникации в команде, что позволяет лучше понимать друг друга и снижает вероятность недопонимания;
- позволяет выявить проблемы и ошибки в проектируемой системе на ранних стадиях, что способствует улучшению качества системы.

2.2 Логическая модель

Логическая модель системы ведения проектов компании является важной частью процесса разработки программного обеспечения и позволяет создать абстрактное представление о системе. Эта модель описывает основные компоненты, связи между ними и их взаимодействие в рамках системы.

Логическая модель системы ведения проектов выполняет следующие функции:

- Определение основных компонентов системы: модель определяет, какие компоненты будут входить в систему и как они будут взаимодействовать друг с другом;
- Описание процессов, которые выполняет система: модель описывает, какие задачи и операции система должна выполнять, чтобы управлять проектами;
- Определение интерфейсов и взаимодействия с пользователем: модель определяет, как пользователи будут взаимодействовать с системой и какие интерфейсы будут использоваться;
- Описание логики работы системы: модель описывает, какие правила и условия используются при выполнении задач;
- Определение данных и их структуры: модель определяет, какие данные будут использоваться системой и как они будут структурированы;
- Модель служит основой для разработки более подробной модели и создания рабочей системы;
- Диаграмма вариантов использования (прецедентов) — это графическое представление функциональности системы, основанное на взаимодействии актеров (пользователей) и системы. Диаграмма прецедентов позволяет определить, какие действия могут выполнять

пользователи в системе и как система должна отвечать на эти действия, диаграмма представлена на рисунке 6.

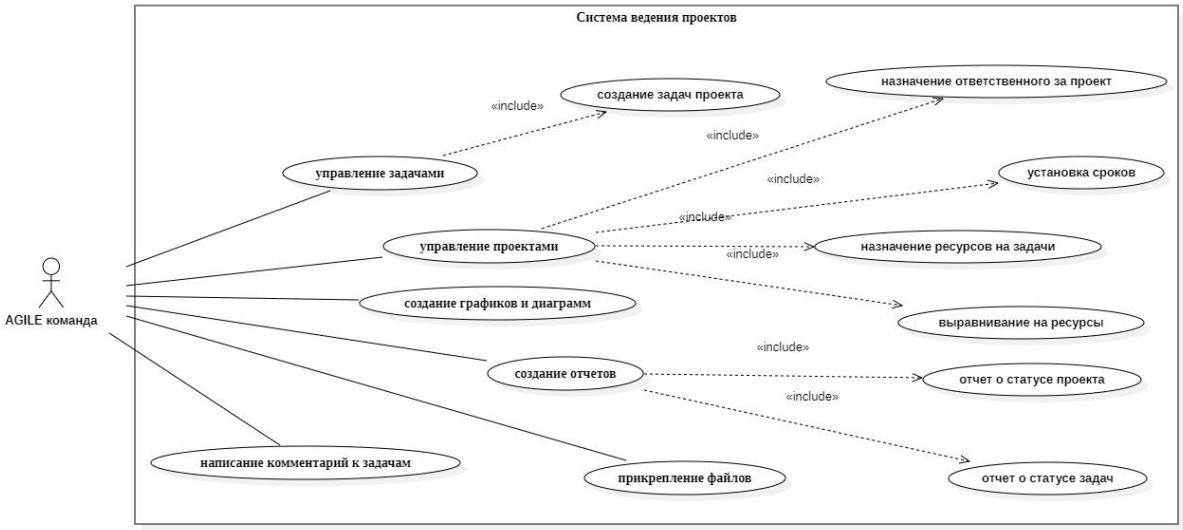

Рисунок 6 – UML диаграмма вариантов использования (англ. USE CASE DIAGRAM)

Основными пользователями системы будет Agile команда, которая будет вести несколько проектов одновременно с помощью системы.

Представим сценарии для наиболее важных вариантов использования.

– Сценарий к варианту использования «Создание задач проекта»:

Пользователь заходит в систему ведения проектов и выбирает проект, для которого нужно создать задачу.

Пользователь переходит на страницу создания задачи и заполняет необходимые поля, такие как название задачи, описание, дата начала и окончания, приоритет и ответственный за задачу.

Пользователь нажимает на кнопку «Создать задачу».

Система проверяет корректность введенных данных и сохраняет задачу в базе данных проекта.

Система отправляет уведомление ответственному за задачу о том, что ему назначена новая задача.

Пользователь получает уведомление об успешном создании задачи и перенаправляется на страницу проекта.

В случае ошибки при заполнении полей система должна уведомить пользователя о необходимости исправить данные и повторить попытку создания задачи. Если данные заполнены корректно, задача должна быть создана и доступна для просмотра и редактирования в рамках проекта;

- Сценарий к варианту использования «Управление проектами»:

Пользователь заходит в систему ведения проектов и выбирает проект, который он хочет управлять.

Пользователь переходит на страницу проекта, где ему доступны различные функции управления проектом, такие как создание задач, установка сроков выполнения, назначение ответственных и т.д.

Пользователь создает задачи, определяет их приоритеты, назначает ответственных, устанавливает сроки выполнения и отслеживает их выполнение.

Пользователь также может добавлять новых участников проекта, определять их роли и управлять доступом к проекту.

Пользователь получает уведомления о задачах, требующих его внимания, и может отслеживать прогресс выполнения задач в рамках проекта.

Пользователь может просматривать отчеты о ходе выполнения проекта, анализировать эффективность работы команды и принимать необходимые меры для улучшения результатов.

Пользователь также может выгружать данные о проекте в различных форматах, например, в таблицы Excel, для дальнейшего анализа или использования в других системах.

В случае возникновения ошибок при управлении проектом, система должна уведомлять пользователя о причинах и предоставлять рекомендации по их устранению. Если все действия пользователя выполнены корректно, система должна обновлять информацию о проекте и сохранять изменения в базе данных для последующего использования;

- Сценарий к варианту использования «Доска Kanban»:

Пользователь может выбрать доску Kanban, в которой будут собраны все задачи по проекту и статус их выполнения;

- Сценарий к варианту использования «Создание графиков и диаграмм»:

Пользователь заходит в систему ведения проектов и выбирает проект, для которого он хочет создать графики или диаграммы.

Пользователь переходит на страницу проекта, где ему доступны различные функции управления проектом, включая создание графиков и диаграмм.

Пользователь выбирает тип графика или диаграммы, который он хочет создать, например, гистограмму, круговую диаграмму или диаграмму Ганта.

Пользователь выбирает параметры для создаваемого графика или диаграммы, например, временной диапазон или выбор определенных задач проекта.

Пользователь может настроить внешний вид графика или диаграммы, например, изменить цвета, размеры, подписи и т.д.

Пользователь сохраняет созданный график или диаграмму и может просматривать его на странице проекта, а также выгружать в различных форматах для использования в других системах или для печати.

Пользователь может обновлять созданный график или диаграмму в соответствии с изменениями в проекте, например, добавление новых задач или изменение сроков выполнения.

В случае возникновения ошибок при создании графика или диаграммы, система должна уведомлять пользователя о причинах и предоставлять рекомендации по их устранению. Если все действия пользователя выполнены корректно, система должна сохранять созданные графики и диаграммы в базе данных для последующего использования;

- Сценарий к варианту использования «Создание отчетов»:

Пользователь заходит в систему ведения проектов и выбирает проект, для которого необходимо сформировать отчет.

Пользователь выбирает тип отчета, который необходимо создать. Например, это может быть отчет о статусе проекта, отчет о прогрессе работ, отчет о затратах на проект и т.д.

Пользователь указывает параметры для формирования отчета, такие как диапазон дат, список задач, участников проекта и другие необходимые фильтры.

Система формирует отчет на основе выбранных пользователем параметров. В отчете может быть представлена информация о состоянии проекта, задачах и их статусе, прогрессе работ, затратах на проект и других показателях.

Пользователь может просмотреть сформированный отчет и сохранить его в различных форматах, например, в формате PDF или Excel, для дальнейшего использования или отправки другим участникам проекта.

В случае необходимости пользователь может повторить процесс создания отчета, изменяя параметры или выбирая другой тип отчета. Также система может предоставлять возможность автоматического формирования отчетов по расписанию или на основе заданных событий, например, при достижении определенного этапа проекта или при изменении статуса задачи.

На основе проведенного анализа функций и требований к системе можно выделить объекты системы ведения проектов.

Проекты: объекты, которые содержат информацию о проекте, такую как название проекта, описание, даты начала и окончания проекта и другие атрибуты.

Задачи: объекты, которые содержат информацию о задачах, которые необходимо выполнить в рамках проекта. Задачи могут иметь атрибуты, такие как название, описание, статус, дата начала и окончания, а также ответственного за выполнение.

Пользователи: объекты, которые представляют пользователей системы ведения проектов. Они могут иметь атрибуты, такие как имя пользователя, адрес электронной почты, роль в проекте и другие.

Команды: объекты, которые представляют команды, назначенные на выполнение задач в проекте. Команды могут иметь атрибуты, такие как название, описание и список пользователей, которые составляют команду.

Отчеты: объекты, которые содержат информацию о продвижении проекта, такую как отчеты о выполнении задач и общий прогресс проекта [18].

После выделения объектов системы можно построить диаграмму классов (Class diagram) [3].

Диаграмма классов представлена на рисунке 7.

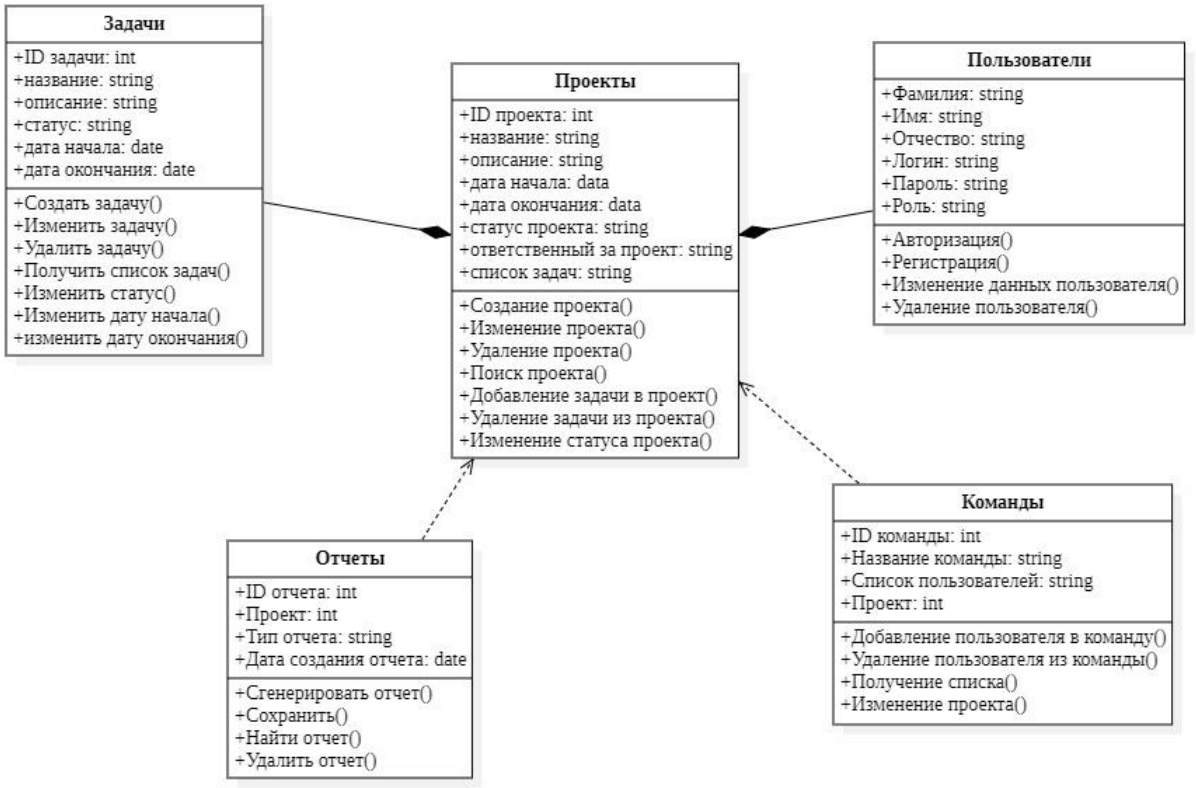

Рисунок 7 – UML Диаграмма классов (англ. class diagram)

Диаграмма классов показывает классы, атрибуты и методы проектируемой ИС.

2.3 Информационное обеспечение

В разработке необходимо придерживаться стандартам разработки информационных системы (нормативная документация разработки программного обеспечения):

- Стандарт ISO/IEC 12207:2008 «Процессы жизненного цикла программного обеспечения»;
- Стандарт ISO/IEC 15504:2012 «Оценка процессов жизненного цикла программного обеспечения»;
- Стандарт ISO/IEC 9126-1:2001 «Характеристики качества программного обеспечения. Часть 1: Внешние характеристики»;
- Стандарт ISO/IEC 9126-2:2001 «Характеристики качества программного обеспечения. Часть 2: Внутренние характеристики»;
- Стандарт ISO/IEC 25010:2011 «Системы и программные продукты. Оценка качества и аттестация качества продуктов программного обеспечения»;
- Стандарт ISO/IEC 27001:2013 «Информационная технология. Методы обеспечения информационной безопасности. Системы менеджмента информационной безопасности»;
- Стандарт IEEE 830-1998 «Рекомендации по написанию требований к программному обеспечению»;
- Стандарт IEEE 1062-2015 «Рекомендации по проведению проверок и аудитов программного обеспечения»;
- Стандарт IEEE 1012-2016 «Стандарт для верификации и валидации программного обеспечения»;
- Стандарт IEEE 829-2008 «Стандарт для документации тестирования программного обеспечения»[12].

2.4 Проектирование базы данных

2.4.1 Выбор модели данных базы данных

База данных — это упорядоченный набор структурированной информации или данных в электронном виде. Для управления БД используют систему управления базой данных (СУБД).

При проектировании базы данных следует учитывать следующие факторы:

- Требования к производительности: если система должна обрабатывать большие объемы данных и обеспечивать быстрый доступ к ним, то необходимо выбирать технологии, обладающие высокой производительностью;
- Требования к безопасности: если в системе хранятся конфиденциальные данные, необходимо выбирать технологии, обеспечивающие высокий уровень защиты данных;
- Наличие необходимых функциональных возможностей: если система должна обеспечивать определенный функционал, необходимо выбирать технологии, позволяющие реализовать этот функционал;
- Опыт разработки: если команда разработчиков имеет опыт работы с определенными технологиями, то разумно выбрать технологию, с которой они работали ранее.

Некоторые из наиболее популярных моделей данных баз данных:

1. Реляционные базы данных (Relational Database Management Systems - RDBMS), такие как MySQL, PostgreSQL, Oracle, Microsoft SQL Server и другие. Они используют язык SQL для хранения и управления данными, а также поддерживают транзакции, целостность данных, безопасность и другие функции. Данные хранятся в виде строк и столбцов, формирующих таблицы. Для создания, модификации и управления данными используется язык

структурированных запросов SQL (англ. Structured Query Language — «язык структурированных запросов»). Реляционные базы данных в настоящее время имеют наибольшее распространение [12].

2. NoSQL базы данных (Not Only SQL), такие как MongoDB, Cassandra, Redis, CouchDB и другие. Они обычно используют документы, ключи и значения, графы или другие форматы для хранения данных, и могут обеспечивать высокую масштабируемость и производительность [12].

3. Графовые базы данных (Graph Database Management Systems), такие как Neo4j, OrientDB, ArangoDB и другие. Они специализируются на хранении и обработке графовых данных, таких как социальные сети, связи между пользователями, карты дорог и т.д [12].

Для информационной системы ведения проектов будет использована реляционная база данных.

Для проектирования концептуальной модели базы данных используют модель данных Entity-Relationship model (модель «сущность-связь»), которая позволяет описать проектируемую БД.

2.4.2 Разработка концептуальной модели данных системы

Генерация схем данных всех уровней относится к процессу создания модели данных, которая представляет собой логическую структуру данных, используемых в информационной системе. Для моделирования базы данных используется семантическая модель данных в нотации IDEF1X.

Схема данных для уровня бизнес-логики описывает данные, которые используются в бизнес-процессах, связи между ними и правила, которые используются для их обработки. Она может быть использована для создания бизнес-моделей и проектирования бизнес-процессов.

Генерация схем данных всех уровней включает в себя определение сущностей и атрибутов, которые необходимы для каждого уровня, и создание соответствующих связей между ними. Это обеспечивает целостность и

согласованность данных на всех уровнях информационной системы и помогает обеспечить эффективную работу системы.

Разработка концептуальной модели данных для системы ведения проектов включает следующие этапы:

- идентификация сущностей;
- определение атрибутов;
- определение связей;
- определение ограничений целостности.

В сущности, Задачи вводится суррогатный ключ ID задачи так как задачи могут повторяться, он же и становится ключевым атрибутом (первичный ключ).

В сущности, Проекты вводится суррогатный ключ ID проекта, так как проекты могут повторяться, он же и становится ключевым атрибутом (первичный ключ) и добавляется внешний ключ id пользователя так как он является ключевым атрибутом сущности Пользователя.

В сущности, Пользователи так же вводится суррогатный ключ id так как фамилии могут повторяться, он же и становится ключевым атрибутом (первичный ключ), добавляется внешний ключ id должности.

В сущности, Должность вводится суррогатный ключ id должности, он же и становится ключевым атрибутом (первичный ключ).

В сущности, Команды ключевой атрибут id команды, он и становится ключевым атрибутом (первичный ключ). Добавляется внешний ключ id проекта, который обеспечивает связь с сущностью Проекты.

В сущности, Отчеты ключевой атрибут id отчета, он и становится ключевым атрибутом (первичный ключ). Добавляется внешний ключ id проекта, который обеспечивает связь с сущностью Проекты.

Концептуальная модель данных описывает сущности и их связи без деталей реализации в базе данных. Разработка концептуальной модели данных для системы ведения проектов включает следующие этапы:

Идентификация сущностей: основных объектов, таких как проекты, задачи, пользователи и т.д.

Определение атрибутов: характеристик сущностей, таких как название проекта, статус задачи, имя пользователя и т.д.

Определение связей: определение связей между сущностями, которые будут храниться в базе данных, таких как связь между проектами и задачами, связь между пользователями и задачами и т.д.

Определение ограничений целостности: определение правил, которые гарантируют корректность данных в базе данных, таких как правило, что задача не может быть назначена на несуществующий проект или правило, что пользователь не может быть назначен на несуществующую задачу [7][21].

Для сущностей необходимо ввести суррогатный ключ.

На рисунке 8 представлена ER-диаграмма.

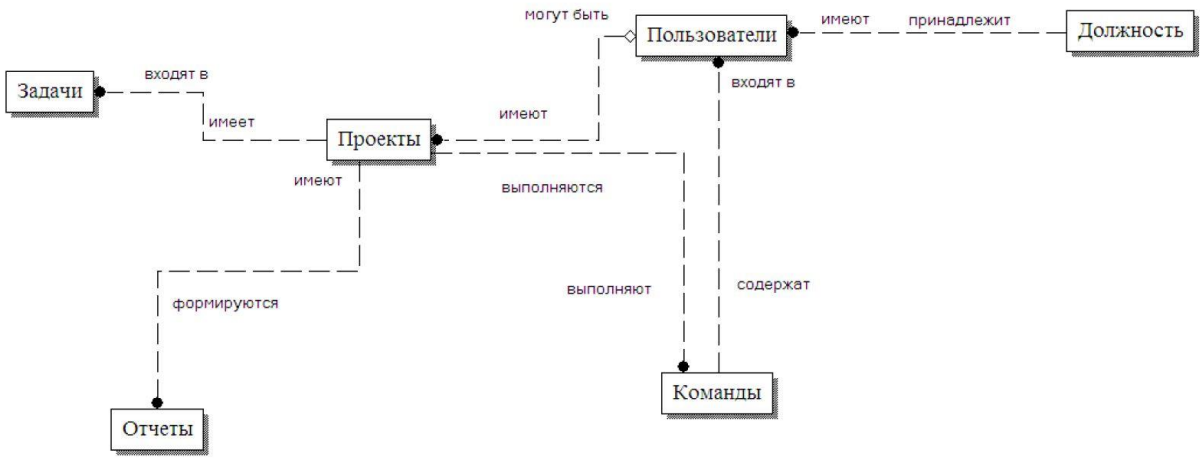

Рисунок 8 – ER-диаграмма

В таблице 2.1 отражены сущности и атрибуты.

Таблица 2.1 – Сущности и атрибуты базы данных

Имя сущности	Описание	Тип
Пользователь	Id пользователя	Primary key
	Фамилия	Текстовый
	Имя	Текстовый
	Отчество	Текстовый
	Логин	Текстовый
	Пароль	Текстовый
	Id команды	Внешний ключ
	Id должности	Внешний ключ
Должность	id должности	Primary key
	Название	Текстовый
	Оклад	Денежный
Команды	Id команды	Primary key
	Название	Текстовый
	Id проекта	Внешний ключ
Отчеты	Id отчета	Primary key
	Тип отчета	Текстовый
	Дата создания	Дата
	Id проекта	Внешний ключ
Задачи	Id задачи	Primary key
	Название	Дата
	Id проекта	Primary key
	Описание	Текстовый
	Статус	Текстовый
	Дата начала	Дата
	Дата окончания	Дата
Проекты	Id проекта	Primary key
	Название	Текстовый
	Описание	Текстовый
	Ответственный	Внешний ключ
	Дата начала	Дата
	Дата окончания	Дата
	Статус проекта	Текстовый

Логическая модель базы данных информационной системы ведения проектов представлена на рисунке 9.

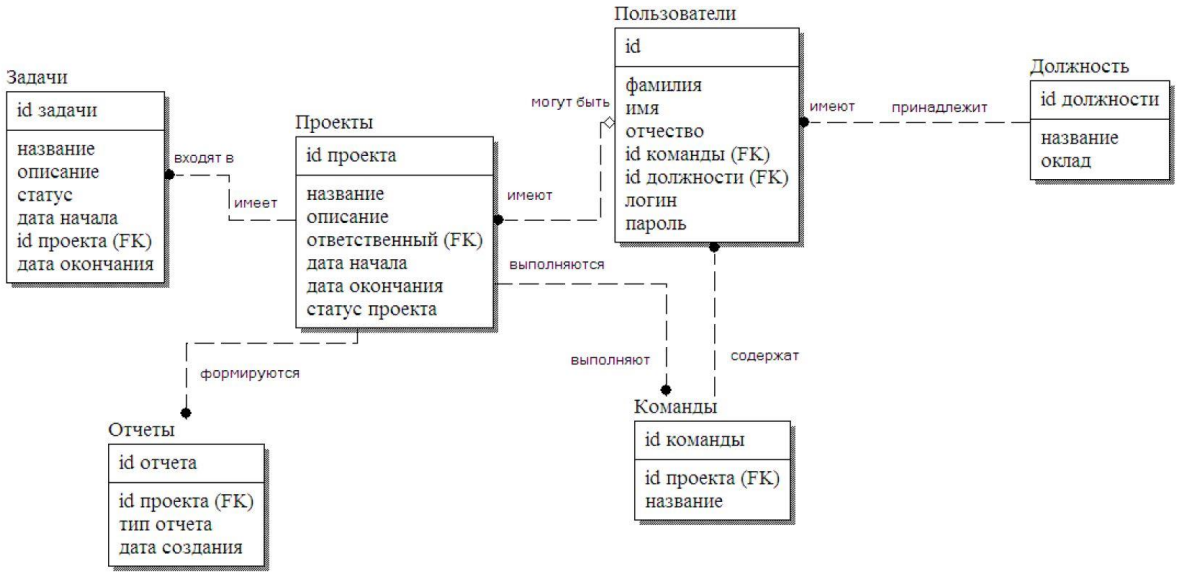

Рисунок 9 – Логическая модель базы данных

Логическая модель базы данных не привязана к конкретной СУБД.

2.5 Требования к аппаратно-программному обеспечению информационной системы

Для эффективной работы системы ведения проектов необходимо обеспечить определенные требования к аппаратно-программному обеспечению:

- Процессор. Желательно использовать мощный процессор, так как система ведения проектов требует вычислительной мощности для обработки больших объемов данных и быстрой генерации отчетов;
- Оперативная память. Рекомендуется использовать не менее 8 ГБ оперативной памяти, чтобы обеспечить быструю обработку данных и их эффективную передачу;
- Хранение данных. Для системы ведения проектов требуется большой объем хранения данных, поэтому необходимо использовать надежный жесткий диск или сетевое хранилище;

- ОС и программное обеспечение. Система ведения проектов должна работать на современной операционной системе, например, Windows 10 или macOS, и поддерживать последние версии браузеров, таких как Google Chrome, или Safari. Также необходимо установить необходимое программное обеспечение, такое как базы данных, серверное ПО и т.д.;
- Среда разработки. Для разработки системы ВП необходима среда разработки, такая как MS Visual Studio или MS Visual Studio Code, а также необходимы инструменты для создания и управления базами данных, такие как MS SQL Server Management Studio или MySQL Workbench;
- Безопасность. Система ведения проектов должна обеспечивать безопасность данных и доступа к системе. Для этого необходимо использовать антивирусное ПО и настроить права доступа пользователей;
- Интернет-соединение. Для доступа к системе ведения проектов из удаленных мест необходимо иметь высокоскоростное интернет-соединение. Также необходимо обеспечить безопасную передачу данных через интернет, например, используя протокол HTTPS.

В целом, необходимо обеспечить высокую производительность и безопасность системы ведения проектов, чтобы обеспечить эффективную работу пользователей и защитить данные компании.

Выводы по главе 2

Во второй главе выбрана модель данных базы данных. Выполнено концептуальное и логическое проектирование базы данных. На основании представленного моделирования можно разработать базу данных для информационной системы ведения проектов. Были выдвинуты основные требования к аппаратно-программному обеспечению системы, на которые также будет основываться программист при разработке.

Глава 3 Физическое проектирование информационной системы

3.1 Выбор архитектуры информационной системы

Существует несколько видов системных архитектур, которые используются в различных областях и для различных целей.

Монолитная архитектура: это классический подход, при котором весь функционал приложения находится в одном целом, неразделенном блоке кода. Это означает, что приложение является единым модулем, который запускается на одном или нескольких серверах. Монолитная архитектура хорошо подходит для маленьких проектов, но может стать проблемой при увеличении сложности приложения и объемах работы.

Клиент-серверная архитектура: это модель, в которой приложение разделено на две части: клиентскую и серверную. Клиент обычно представляет из себя пользовательский интерфейс, а сервер выполняет логику приложения и управляет базой данных. Клиент-серверная архитектура позволяет создавать распределенные системы, где клиенты и серверы могут находиться в разных местах.

Service-oriented architecture (SOA): это подход, который использует сервисы для обеспечения функционала приложения. Сервисы могут быть созданы для выполнения конкретных задач и могут быть вызваны другими сервисами или клиентами. SOA позволяет создавать более гибкие системы, которые могут легко масштабироваться и модифицироваться.

Микросервисная архитектура: это подход, который использует множество небольших сервисов, каждый из которых выполняет определенную функцию. Микросервисы могут работать вместе, чтобы обеспечить полный функционал приложения. Этот подход обеспечивает более высокую гибкость и масштабируемость, но может быть более сложным в управлении.

Event-driven архитектура: это подход, при котором приложение реагирует на события, которые происходят в системе, такие как изменение

данных в базе данных или прибытие новых данных из внешней системы. События могут инициироваться в любой части системы, и приложение должно быть способно быстро реагировать на них и выполнять соответствующие действия.

Для информационной системы ведения проектов больше подойдет клиент-серверная архитектура.

Клиент-серверная архитектура включает в себя сервер, на котором работает база данных и обрабатываются запросы от клиентов. Клиенты могут быть как настольными, так и веб-приложениями. Эта архитектура обеспечивает высокую производительность и масштабируемость, но требует больших затрат на обеспечение надежности и безопасности.

Клиент-серверная архитектура может быть двух или трехзвенной. Если приложение обращается напрямую к СУБД - это двухзвенная архитектура, в случае информационной системы ведения проектов - это трехзвенная архитектура, когда между СУБД и клиентом существе веб-сервер, который обрабатывает запросы пользователи. Клиент-серверная архитектура включает в себя следующие основные компоненты:

Клиент — это приложение, работающее на компьютере пользователя, которое позволяет пользователю взаимодействовать с сервером. Клиент может быть как программным обеспечением, установленным на компьютере пользователя, так и веб-браузером, который обращается к серверу через интернет.

Сервер — это центральный компонент, который обрабатывает запросы от клиентов, обеспечивает доступ к данным и выполняет бизнес-логику. Сервер может быть как физическим сервером, так и виртуальной машиной.

Сеть — это среда передачи данных, которая связывает клиентов и серверы между собой. Для передачи данных между клиентом и сервером используются различные протоколы, такие как HTTP, TCP/IP, FTP и другие.

База данных — это хранилище данных, к которому имеет доступ сервер. База данных может быть как локальной, находящейся на сервере, так и удаленной, находящейся на другом сервере.

Преимущества клиент-серверной архитектуры:

– Высокая отказоустойчивость: приложение может продолжать работу, если один из компонентов не работает;
– Безопасность: сервер может контролировать доступ к данным и ресурсам;
– Масштабируемость: система может быть масштабирована путем добавления новых серверов.

Серверная часть системы ведения проектов обрабатывает запросы, отправленные пользователем, извлекая информацию базы данных, и выводит ее в интерфейсе приложения.

Компоненты ИТ-системы - это база данных или пользовательский интерфейс. Компонентами могут быть программные компоненты рисунок 10.

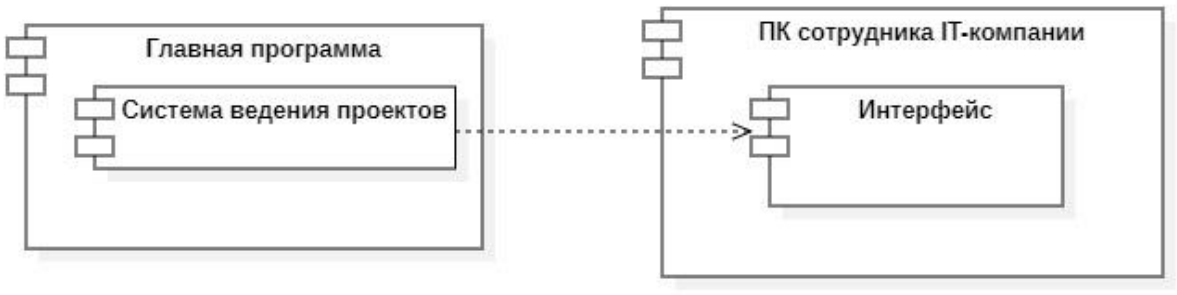

Рисунок 10 – UML Диаграмма компонентов

Диаграмма компонентов позволяет разработчикам понять, как компоненты системы взаимодействуют друг с другом, и определить, какие компоненты необходимо разработать или изменить для реализации требуемой функциональности. Она также может быть использована для определения технических требований к инфраструктуре, таких как серверы и базы данных, необходимых для поддержки системы.

На рисунке 11 представлена UML диаграмма размещения. Диаграмма размещения показывает физическое размещение трехзвенной клиент-серверной архитектуры.

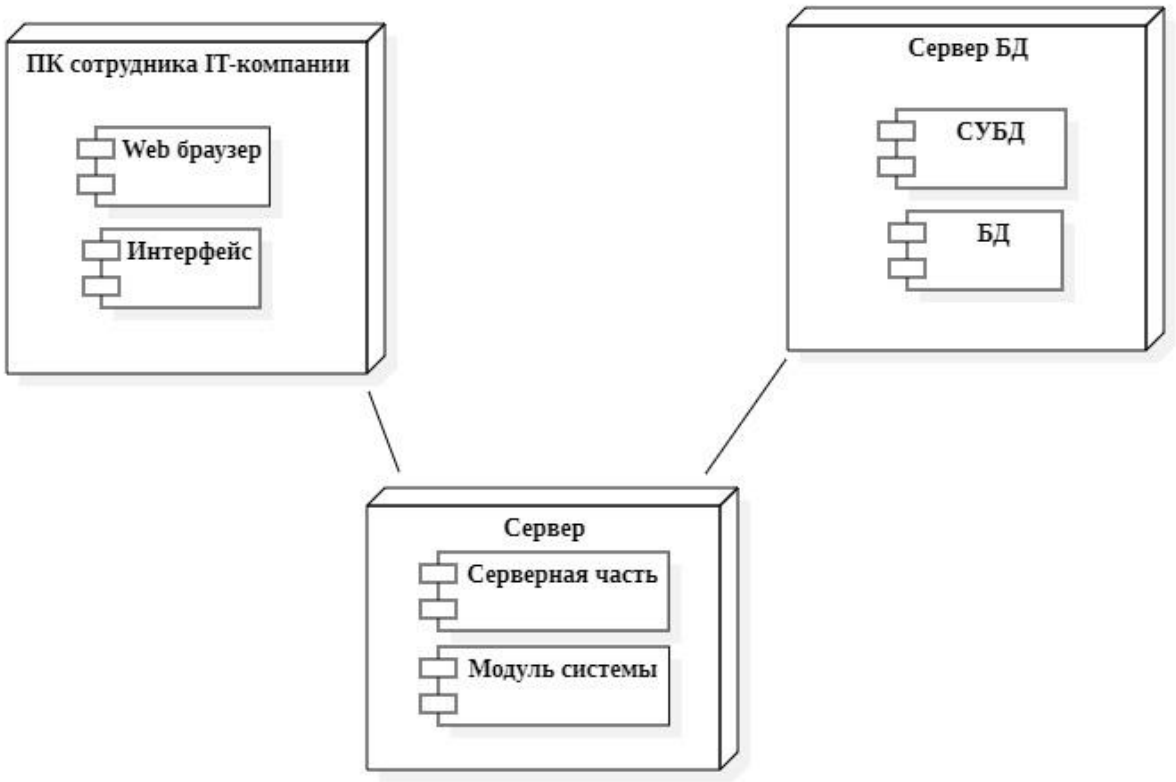

Рисунок 11 – UML Диаграмма размещения

Трехзвенная клиент-серверная архитектура является наиболее функциональной за счет веб-сервера.

3.2 Выбор технологии разработки информационной системы

Для разработки системы ведения проектов рекомендуется использовать гибкие (Agile) методологии разработки, такие как Scrum и Канбан. Они позволяют быстро адаптироваться к изменяющимся требованиям и эффективно управлять процессом разработки.

Scrum — это гибкая методология управления проектами, которая позволяет команде разработчиков быстро адаптироваться к изменениям в процессе разработки и добиться наилучшего результата.

Выбранная Методология scrum позволяет:

- команде быстро адаптироваться к изменениям в процессе разработки. Каждый спринт (итерация) может быть адаптирован в соответствии с новыми требованиями или ограничениями;
- инкрементальную разработку, что означает, что функциональность добавляется к проекту постепенно, в итерациях. Это позволяет быстро получать обратную связь и улучшать процесс разработки;
- команде быстро выявлять и решать проблемы, связанные с разработкой. Это позволяет управлять рисками и предотвращать их на ранних стадиях;
- регулярные встречи, обзоры и ретроспективы помогают улучшить коммуникацию и убедиться, что проект развивается в правильном направлении;
- работу в команде, что позволяет максимально эффективно использовать ресурсы и навыки каждого члена команды;
- прозрачность в процессе разработки. Каждый член команды должен знать, что происходит на проекте, какие задачи выполняются, какие проблемы возникают и т.д. Это позволяет всей команде работать вместе и достигать лучших результатов;
- помогает команде разработчиков быть стрессоустойчивой [14].

Для разработки системы в качестве редактора кода Python выбрана IDE (англ. integrated development environment — IDE) PyCharm — это кроссплатформенная среда разработки для языка программирования Python, предоставляет пользователю возможность написания кода и отладчик. Его возможности обширны: автозавершение кода, подсветка синтаксиса, подбор отступа.

PyCharm - разработана компанией JetBrain. Существует платная и бесплатная версия. Для профессиональной разработки необходимо использовать платную PRO версию, существует гибкая система скидок.

Основные функции и плюсы PyCharm:

Возможность создавать проекты.

Редактор предлагает подсветку синтаксиса, автодополнение кода, проверку ошибок и предупреждений. Рефакторинг кода.

Потребляет немного памяти по сравнению с другими громоздкими инструментами разработки и прост в использовании.

Поддерживает систем управления версиями Git.

PyCharm поддерживает языки веб-программирования JavaScript, TypeScript и популярные фреймворки для веб разработки (в платной версии).

После проектирования системы ведения проектов рекомендуется провести следующие действия:

- Разработка: на основе концептуальной и логической модели данных, разработать физическую модель базы данных и начать программирование системы;
- Тестирование: провести функциональное тестирование системы, чтобы убедиться, что все функции работают правильно и нет ошибок;
- Внедрение: установить систему на сервере и обеспечить доступ к ней для пользователей;
- Обучение пользователей: обучить пользователей использованию системы и предоставить им руководство пользователя;
- Поддержка: обеспечить поддержку и обслуживание системы, включая исправление ошибок и обновление программного обеспечения;
- Также рекомендуется постоянно улучшать и совершенствовать систему на основе обратной связи от пользователей, чтобы улучшить ее функциональность и удобство использования.

3.3 Выбор СУБД информационной системы

Для хранения информации о проектах должна быть база данных. Для управления базой данных используются системы управления баз данных.
Сравним такие СУБД как MSSQL Server, PostgreSQL и Oracle Database.
MS SQL Server — это реляционная система управления базами данных (СУБД), разработанная корпорацией Microsoft. SQL Server поддерживает язык T-SQL (Transact Structured Query Language) и широкий набор инструментов для разработки, управления и обслуживания баз данных.

Некоторые из основных возможностей SQL Server:

- MS SQL Server может обрабатывать миллионы запросов в день и хранить терабайты данных;
- Безопасность: SQL Server обеспечивает высокий уровень безопасности данных, включая авторизацию, аутентификацию, шифрование и аудит. Это позволяет защитить данные от несанкционированного доступа и использования;
- Интеграция с другими продуктами Microsoft: MS SQL Server хорошо интегрируется с другими продуктами Microsoft, такими как MS Excel, Power BI и Visual Studio. Это позволяет использовать данные из различных источников и упрощает разработку и анализ данных;
- Аналитика данных: SQL Server включает в себя мощный инструментарий для аналитики данных, включая инструменты для создания OLAP-кубов, анализа данных и визуализации данных. Это позволяет быстро и легко анализировать данные и принимать важные бизнес-решения;
- Высокая доступность: SQL Server обеспечивает высокую доступность данных, включая резервное копирование, восстановление после сбоев и репликацию данных. Это позволяет минимизировать время простоя системы и сохранять данные в безопасности;

- Поддержка облачных вычислений: SQL Server можно использовать как локально, так и в облаке, с помощью Microsoft Azure. Это позволяет легко масштабировать базы данных и управлять ими с помощью облачных сервисов.

СУБД PostgreSQL:

- Бесплатное и открытое ПО;
- Установка на бесплатный Linux дистрибутив;
- Существуют российская версия Postgres PRO;
- Хорошо подходит для крупных проектов с высокими требованиями к масштабируемости;
- Полностью поддерживает стандарт SQL;
- Обладает мощными функциями, такими как триггеры, процедуры и внешние ключи;
- Существует активное сообщество пользователей и хорошая документация;
- Установка и настройка может быть более сложной, чем MS SQL сервер.
- СУБД Oracle Database:
- Предоставляет широкий спектр возможностей и масштабируется для больших проектов;
- Полностью поддерживает стандарт SQL;
- Обладает мощными функциями, такими как кластеризация и репликация данных;
- Предоставляет высокую степень безопасности и управления доступом;
- Существует широкое сообщество пользователей и большое количество документации.

Однако Oracle Database является платным ПО, и установка и настройка могут быть сложными.

Разработка на фреймворке Django универсальная и подходит для разных СУБД систему, написанную на этом фреймворке, можно переносить на разные СУБД.

Встраиваемые СУБД, такие как SQLite, подходят для веб-проекта, но так они имеет ряд ограничений рекомендуется использовать полноценную СУБД. СУБД SQLite по умолчанию встроена в проект Django, соответственно небольшой проект можно развернуть без подключения полновесных СУБД.

В целом, PostgreSQL является мощной и надежной СУБД, которая может удовлетворить потребности большинства организаций в управлении и анализе данных. СУБД PostgreSQL выбрана для реализации проекта.

3.4 Разработка физической модели данных

Для физической реализации БД используется язык SQL.

Примеры скрипта для создания основной таблицы бд панели Канбан представлен на рисунке 12.

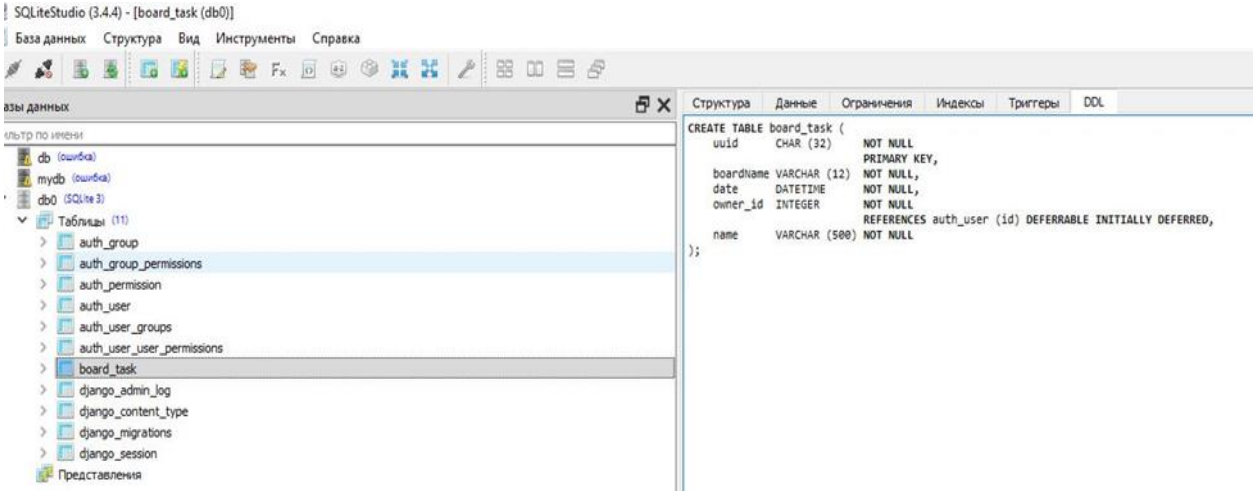

Рисунок 12 – SQL скрипт создания таблицы доски Канбан

Список всех таблиц доски Канбан:

- auth_group;
- auth_group_permissions;
- auth_permission;
- auth_user_groups;
- auth_user_user_permissions;
- django_migrations;
- django_session;
- django_content_type;
- django_admin_log;
- board_task;
- auth_user.

3.5 Разработка информационной системы

Для реализации информационной системы нужно выбрать наиболее подходящий язык программирования [11].

Рассмотрим C#, Python и JavaScript.

Python, C# и JavaScript — это популярные языки программирования, используемые для разработки различных типов приложений и систем. Каждый из этих языков имеет свои сильные и слабые стороны, а также области применения.

Python - интерпретируемый язык программирования, который часто используется для научных вычислений, анализа данных, машинного обучения и разработки веб-приложений. Python известен своей простотой и удобством в использовании, что делает его очень популярным среди начинающих программистов. Python также имеет большое сообщество пользователей и богатую библиотеку сторонних модулей и пакетов, что делает его мощным

инструментом для быстрой разработки. Для реализации веб-приложения существует очень популярный в настоящее время фреймворк Django с открытым исходным кодом на языке Python [4].

C# - объектно-ориентированный язык программирования, разработанный Microsoft. C# часто используется для создания Windows-приложений, игр и веб-приложений на платформе .NET. Он имеет строгую типизацию и обширную стандартную библиотеку, что делает его мощным языком для разработки крупных проектов. C# также поддерживает асинхронное программирование, что делает его очень эффективным для работы с большими объемами данных.

JavaScript - язык программирования, который широко используется для разработки интерактивных веб-приложений и сайтов. JavaScript может использоваться для создания клиентской и серверной части веб-приложений, а также для разработки мобильных приложений. Он имеет простой синтаксис и широко распространенный веб-фреймворк Node.js, что делает его очень популярным для быстрой разработки веб-приложений [13].

В разработке можно использовать готовые решения с открытым исходным кодом и адаптировать под свой проект, дописывая собственные доработки в проект.

Так как проект направлен на разработку WEB-приложения, то Python и фреймворк Django может быть лучшим выбором. Веб проект может быть написан на нескольких языках библиотеках или фреймворках. Python фреймфорк Django позволяет управлять базой данных, формами, шаблонами, системой аутентификации и авторизации [5].

Автоматизированная система представляет собой полноценную программу, которая позволяет пользователю системы вносить необходимые данные, а также находить необходимые информацию по проектам, задачам, срокам выполнения задач. Система облегчает рабочий процесс сотрудников отдела разработки.

Для доски Kanban возьмем готовый проект с подробной инструкцией по развертыванию и открытым исходным на Github https://github.com/natkaida/kanban. Фронтенд доски Канбан написан на Alpine.js JS-фреймворке. Дизайн сделан на CSS-фреймворке Tailwind, а для HTTP-запросов к бэкенду используется библиотека Axios [22].

Графический интерфейс приложения предназначен для ввода и отображения данных системы, на рисунке 13 показан контрольный пример доска Kanban.

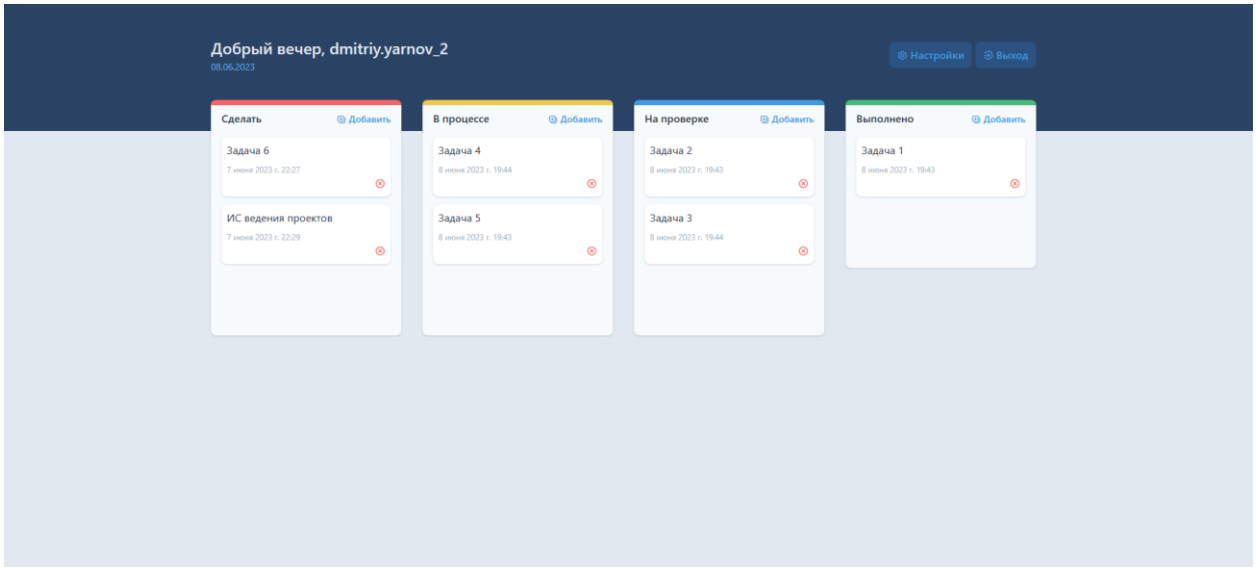

Рисунок 13 – панель Канбан

В программе ведется учет проектов, которые были выполнены ранее и выполняются в данный момент. В доске канбан карточки можно перетаскивать из одной колонки в другую, редактировать и удалять без перезагрузки страницы.

На рисунке 14 в качестве контрольного примера показана возможность редактировоания данных проетка.

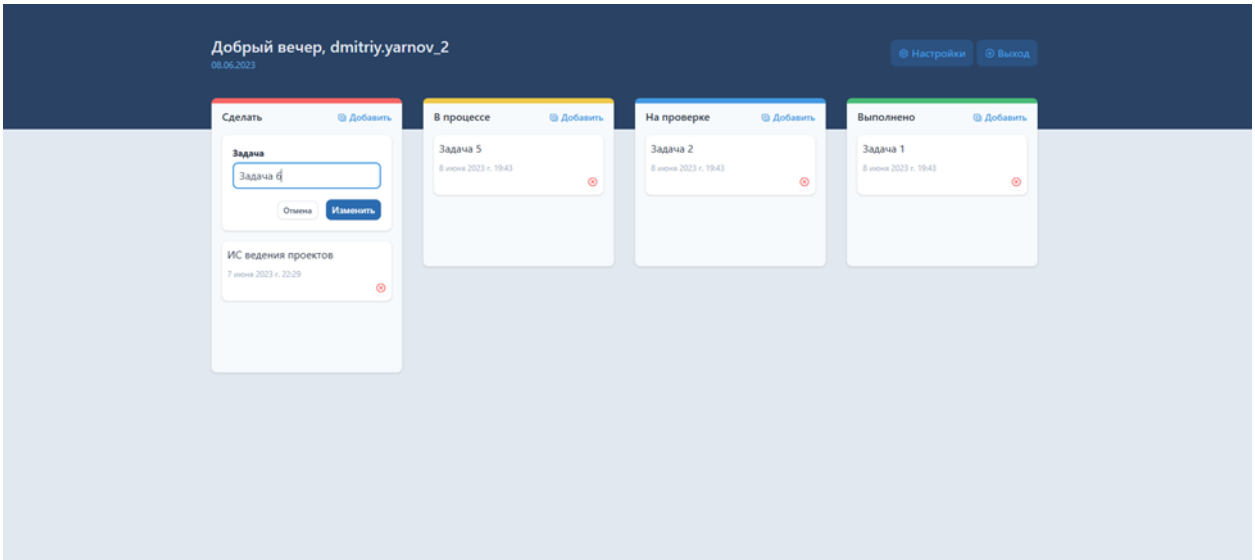

Рисунок 14 – редактирование доски Канбан

Имеется возможность управлять дизайном рисунок 15.

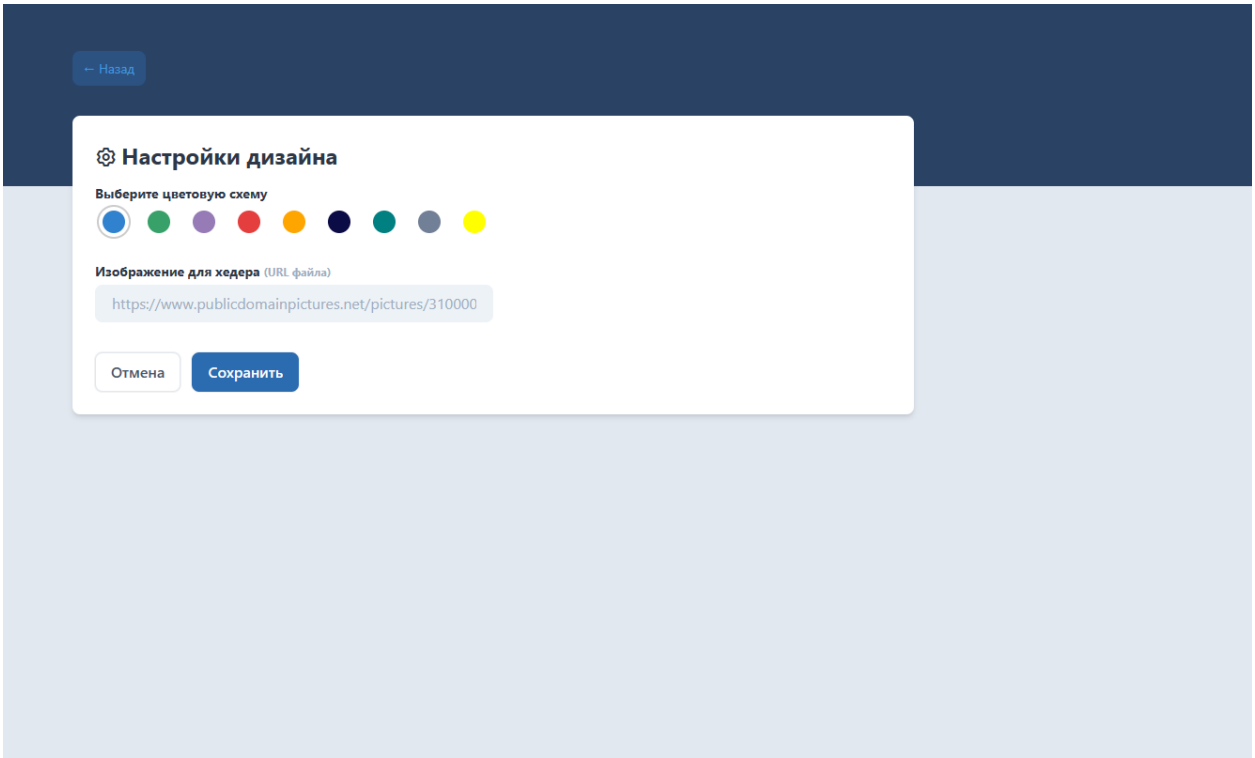

Рисунок 15 – настройки дизайна

Проекты на Django имеют страницу для администрирования, из коробки можно получить администрирование пользователей, управление контентом. На рисунке 16 панель администрирования Django.

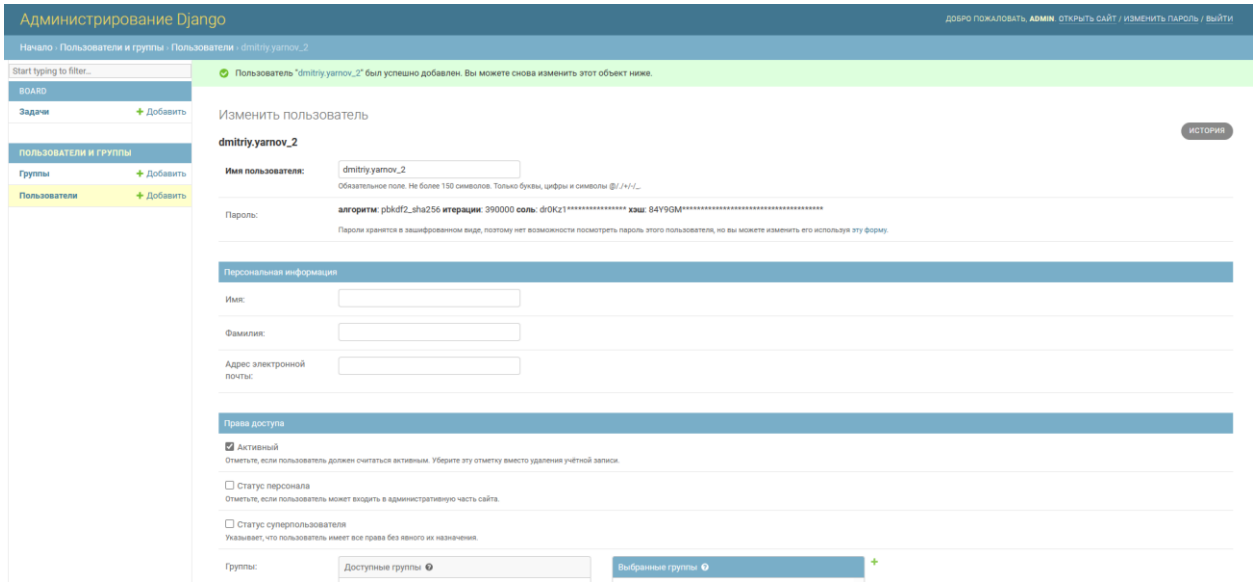

Рисунок 16 – панель администрирование Django

Для построения диаграмм Ганта можно использовать Python библиотеку Plotly или Matplotlib [17].

Информационная система должна обеспечивать выполнение таких функций, как:

– создание проектов;
– создание задач;
– шаблон Scrum;
– доска Kanban;
– диаграмму Ганта;
– учет выполнения;
– учет работы сотрудников по задачам;
– поиск необходимой информации.

Шаблоны Scrum и некоторые компоненты информационной системы необходимо написать самостоятельно.

В базе данных должны быть сущности с названием: «Проект», «Задачи», «Сотрудник», «Клиент». В каждой сущности должны быть атрибуты, один из которых должен иметь уникальный ключ.

Добавление, поиск и обработка информации в базе данных должно осуществляться посредством SQL запросов.

Пользовательский интерфейс должен выглядеть в виде главного меню с выбором функций, каждая функция должна открывать форму по запросу либо формировать отчет, согласно запрашиваемой информации.

Наиболее важной частью Agile системы является доска Kanban и scrum шаблоны.

Выходные данные системы должны быть в виде отчетов или специальной формы.

Доступ к системе будут иметь сотрудники ИТ-компании, участвующие в разработке программных продуктов.

Срок разработки проекта 2 месяца.

На рисунке 17 представлена диаграмма последовательности разработки информационной системы.

Рисунок 17 – UML диаграмма последовательности разработки проекта

На этапе разработки для AGILE подхода доску Канбан можно рисовать на обычной доске для презентаций.

3.6 Расчет экономической эффективности разработки

Экономическая эффективность разработки складывается из внедрения лучших практик Agile для ведения проектов с использованием информационной системы ведения проектов. Как известно каскадные методики разработки часто на выходе нарушают запланированные сроки на внедрение, и финансовые затраты оказываются выше, оказывая негативное влияние на прямой экономический эффект. Каскадные методики рассчитаны на очень крупные проекты. Косвенный экономический эффект также падает без AGILE подхода, падает привлечение большего числа клиентов за счет увеличения сроков разработки, возможны штрафы и неустойка за счет плохого контроля времени реализации проектов.

Существует много методик оценки экономической эффективности, например, стандартные экономические расчет NPV (чистый дисконтированный доход), IRR (внутренняя норма доходности), ROI (рентабельность инвестиций), PI (индекс доходности) и срок окупаемости.

Помимо стандартных экономических методов оценки существуют отраслевой метод оценки экономической эффективности для ИТ отрасли TCO (Total cost of ownership) или совокупная стоимость владения — общие расходы, которые возникают у компании из-за владения IT-инфраструктурой.

Для экономического обоснования разработки информационной системы рассчитаем прямую экономическую эффективность в сравнении с существующим вариантом ведения проектов.

Прямой экономический эффект от внедрения информационной системы рассчитывается на основании трудовых и стоимостных затрат.

В первую очередь улучшаются трудовые показатели за счет оптимизации работы создания проекта разработки и контроля над выполнением.

Абсолютное снижение трудовых затрат ΔT рассчитывается по формуле:

$$\Delta T = T_0 - T_1, (1)$$

где T_0 – время, затраченное ведение проекта в базовом варианте;

T_1- время, затраченное на ведение проекта в информационной системе ведения проектов.

Коэффициент относительного снижения трудовых затрат K_t рассчитывается по формуле:

$$K_t = (\Delta T/T_0) \cdot 100\%, (2)$$

Индекс снижения трудовых Y_T рассчитывается по формуле:

$$Y_T = T_0/T_1, (3)$$

Помимо трудовых экономических показателей улучшаются стоимостные экономические показатели за счет снижения заработной платы, которую сотрудники получают за время работы над ведением проектов.

Абсолютное снижение стоимости ΔC рассчитывается по формуле:

$$\Delta C = C_0 - C_1, (4)$$

где C_0 – стоимостные затраты на обработку информации по базовому варианту;

C_1- стоимостные затраты на обработку информации по предлагаемому варианту.

Коэффициент относительного снижения стоимости K_c рассчитывается по формуле:

$$K_c = (\Delta C/C_0) \cdot 100\%, (5)$$

Индекс снижения стоимостных затрат Y_c рассчитывается по формуле:

$$Y_c = C_0/C_1, (6)$$

Расчет показателей экономической эффективности приведен в таблице 3.1

Таблица 3.1 – Расчет показателей экономической эффективности

	Затраты		Абсолютное изменение затрат	Коэффициент изменения затрат	Индекс изменения затрат
	Базовый вариант	Проектный вариант			
Трудоемкость	T_0 (час)	T_1 (час)	$\Delta T = T_0 - T_1$	$K_t = (\Delta T/T_0) \cdot 100\%$	$Y_T = T_0/T_1$
	24	8	16	66%	3
Стоимость	C_0 (руб.)	C_1(руб.)	$\Delta C = C_0 - C_1$ (руб.)	$K_c = (\Delta C/C_0) \cdot 100\%$	$Y_c = C_0/C_1$
	24000	8000	16000	66%	3

Рассчитаем срок окупаемости T_{ok} по формуле:

$$T_{ok} = K_п/\Delta C. (7)$$

где $K_п$ - капитальные затраты на проект [10].

Для расчета срока окупаемости составим список затрат на реализацию проекта, исходим с того, что компании, действующей с существующей стандартной на сегодняшний день ит-инфраструктурой, нет необходимости закупать серверное оборудование. В проект задействовано 4 сотрудника.

Список затрат:

- серверное оборудование и лицензии на серверное ПО: виртуальный сервер в существующем кластере на базе Ubuntu 20.04 LTS 64-bit= 0 рублей за лицензию;

- IDE для разработки PyCharm: $29.88 в месяц (курс на 11.06.23 82.70 за USD = 2471 рублей в месяц);
- сертификат/домен для сайта: 2000 рублей в год;
- лицензии на СУБД: создание базы данных на существующем сервере СУБД PostgreSQL: 0 рублей;
- система контроля версий GIT: бесплатно;
- зарплаты сотрудников, участвующих в проекте разработки: 320 000 рублей за 2 месяца разработки.

  Итоговая сумма первоначальных затрат: 326 942 рублей.

T_{ok}= 20 месяцев.

Как видно, основная часть затрат уходит на зарплаты сотрудникам участвующих в разработке, соответственно, при сокращении времени работы над ведением проектов, мы уменьшим этот показатель в будущих проектах разработки и повысим экономическую эффективность компании с незначительным увеличением стоимости владения на старте разработки.

Выводы по главе 3

В третьей главе описывается физическое проектирование информационной системы с помощью языка графического моделирования UML, построены диаграммы компонентов и размещения. Выбрана СУБД PostgreSQL для базы данных проекта. Выбран язык программирования Python и веб-фреймворк Django. Выполнен обзор инструментов разработки и выбран наиболее подходящие IDE PyCharm. Показаны контрольные примеры работы программы на примере доски Канбан. Показана экономическая эффективность разработки собственного программного продукта.

Заключение

В ходе выполнения выпускной квалификационной работы описан полный цикл разработки системы ведения проектов для ит-компании занимающейся разработкой и поддержкой программного обеспечения.

В ходе исследования и проектирования системы ведения проектов были проведены анализ требований, выбрана технология UML для создания логической модели, определены основные компоненты и архитектура системы, разработаны диаграммы классов и компонентов, выбрана СУБД PostgreSQL и язык программирования Python, выбрана методология Scrum и Канбан для управления жизненным циклом проекта.

Кроме того, были определены требования к аппаратному и программному обеспечению, описаны функциональные и нефункциональные требования, разработаны сценарии использования и прототипы пользовательского интерфейса.

На основе проведенного анализа и выделенного процесса были определены основные объекты и разработаны контрольные примеры интерфейса пользователя и основных форм системы.

Использование современных технологий и инструментов, таких как IDE PyCharm и PostgreSQL сервер, позволит разработать надежную и эффективную систему, удовлетворяющую требованиям бизнеса и пользователей. Методология Scrum и Канбан обеспечит гибкий Agile подход к управлению проектом, учитывая изменения и дополнения в процессе его разработки.

На завершающем этапе были проведены расчеты экономической эффективности проекта.

Результаты работы могут быть использованы айти компаниями, планирующих внедрять AGILE подходы в разработке программного обеспечения, и автоматизации ведения проектов с использование Scrum и Канбан.

Список используемой литературы и используемых источников

Блинов, А.О Реинжиниринг бизнес-процессов : учебное пособие для студентов вузов, обучающихся по специальностям экономики и управления / А. О. Блинов, О. С. Рудакова, В. Я. Захаров, И. В. Захаров ; под редакцией А. О. Блинова. — Москва : ЮНИТИ-ДАНА, 2017. — 343 с.

2. Балдин, К. В. Информационные системы в экономике : учебник / К. В. Балдин, В. Б. Уткин. — 8-е изд. — Москва : Дашков и К, 2019. — 395 с.

3. Буч, Г. Введение в UML от создателей языка. Гради Буч, Джеймс Рамбо, Ивар Якобсон — 2-е издание — М.: ДМК Пресс, 2015. — 496 с.

4. Васильев, А. Н. Python на примерах : практический курс по программированию / А. Н. Васильев. — 2-е изд. — Санкт-Петербург : Наука и Техника, 2017. — 432 с.

5. Головатый, А. Django. Подробное руководство. Адриан Головатый, Джейкоб Каплан-Мосс — Второе издание — СПб.: Символ-плюс, 2010. — 560 с.

6. Горбенко, А. О. Информационные системы в экономике / А. О. Горбенко. — 4-е изд. — Москва : Лаборатория знаний, 2020. — 295 с.

7. Емельянова, Т. В. Моделирование баз данных : учебное пособие / Т. В. Емельянова, А. М. Кольчатов, Н. Ю. Зюзина. — Саратов : Ай Пи Эр Медиа, 2018. — 62 с.

8. Золотов С.Ю. Проектирование информационных систем : учебное пособие / Золотов С.Ю.. — Томск : Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2013. — 88 с.

9. Майк, Кон Agile: оценка и планирование проектов / Кон Майк ; перевод В. Ионов. — Москва : Альпина Паблишер, 2018. — 424 с.

10. Мкртычев, С. В. Прикладная информатика. Бакалаврская работа : электрон. учеб.-метод. пособие / С. В. Мкртычев, О. М. Гущина, А. В. Очеповский ; Тольяттинский государственный университет. – Тольятти : Изд-во ТГУ, 2019.

11. Мартин, Р. Чистый код: создание, анализ и рефакторинг. Р. Мартин. — 1-е изд. — СПб : Питер, 2013. — 464 с.

12. Мартишин С. А. Базы данных [Электронный ресурс] : Практическое применение СУБД SQL и NoSOL-типа для применения проектирования

информационных систем : учеб. пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. - Москва : Форум : ИНФРА-М, 2018. 367 с.

13. Рындин, Н. А. Технологии разработки клиентских WEB-приложений на языке JavaScript : учебное пособие / Н. А. Рындин. — Воронеж : Воронежский государственный технический университет, ЭБС АСВ, 2020. — 54 с.

14. Смирнов, А. А. Разработка прикладного программного обеспечения : учебное пособие / А. А. Смирнов. — Москва : Евразийский открытый институт, Московский государственный университет экономики, статистики и информатики, 2003. — 101 с.

15. Силич, В. А. Реинжиниринг бизнес-процессов : учебное пособие / В. А. Силич, М. П. Силич. — Томск : Томский государственный университет систем управления и радиоэлектроники, 2007. — 200 с.

16. Силич, В. А. Моделирование и анализ бизнес-процессов : учебное пособие / В. А. Силич, М. П. Силич. — Томск : Томский государственный университет систем управления и радиоэлектроники, 2011. — 212 с.

17. Титов, А. Н. Визуализация данных в Python. Работа с библиотекой Matplotlib : учебно-методическое пособие / А. Н. Титов, Р. Ф. Тазиева. — тельство КНИТУ, 2022. — 92 с.

18. Фаулер, М. Разработка требований к программному обеспечению. Вигерс Карл И., Битти Джой — 3-е издание — СПб : БХВ-Петербург, 2019. — 736 с.

19. Цуканова О. А. Методология и инструментарий моделирования бизнес процессов: учебное пособие – СПб.: Университет ИТМО, 2015. – 100 с.

20. Юрген, Аппело Agile-менеджмент: Лидерство и управление командами / Аппело Юрген ; перевод А. Олейник. — Москва : Альпина Паблишер, 2018. —

21. Якимов, В. Н. Проектирование реляционных баз данных : учебное пособие по курсовому проектированию / В. Н. Якимов. — 2-е изд. — Самара : Самарский государственный технический университет, ЭБС АСВ, 2018. — 96

22. Янцев, В. В. Web-программирование на Python / В. В. Янцев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2023. — 180 с.