# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_Toc12004184)

1. [РАЗРАБОТКА И АНАЛИЗ СИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ](#_bookmark1)…………………………………………………………………………….4

[1.1. Основные понятия……………………………………………….………………6](#_Toc12004186)

[1.2. Модели жизненных циклов проектов и методы управления проектами……..7](#_Toc12004187)

1.3. Обзор аналогов…………………………………………………………………….9

1.3.1. Scrum……………………………………………………………………..10

1.3.2. Kanban………………………………………………………………..……10

1.3.3. Trello ……………………………………………………………………..…10

1.3.4. Сравнение аналогов…………………………………………………..……11

[1.4. Анализ инструментов управления проектами ………………………………..12](#_Toc12004192)

[1.5. Выводы…………………………………………………………………………..16](#_Toc12004196)

[[2. МЕТОДЫ И АЛГОРИТМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ В](#_bookmark6) [КОМПАНИИ «ЖК Небо»](#_bookmark6)](#_Toc12004197)………………………………………………………………………………………6

* 1. [Алгоритмы процесса разработки программных продуктов и методы](#_bookmark7) [управления проектами на предприятии «ЖК Небо»…………………………….. 42](#_bookmark7)
  2. [[Организация процесса исследовательской деятельности на предприятии](#_Toc12004201)](#_bookmark8)[[«ЖК Небо» 14](#_Toc12004201)](#_bookmark8)

[2.3. Выводы…………………………………………………………………………..24](#_Toc12004202)

[3.](#_Toc12004203)  [[ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ](#_Toc12004203)](#_bookmark11)[[УПРАВЛЕНИЯ ПРОЕКТАМИ В КОМПАНИИ «ЖК Небо» 65](#_Toc12004203)](#_bookmark11)

[3.1. Архитектура приложение(сайт)……………………………………………….25](#_Toc12004204)

[3.2. Описание средств разработки………………………………………………….25](#_Toc12004205)

[3.3. Описание СУБД …………………………………………………………………26](#_Toc12004206)

[3.4. Проектные модели………………………………………………………………31](#_Toc12004208)

[4. ТЕСТИРОВАНИЕ…………………………………………………………………..36](#_Toc12004212)

[4.1. План тестирования………………………………………………………………36](#_Toc12004213)

[ЗАКЛЮЧЕНИЕ………………………………………………………………………..60](#_Toc12004237)

[СПИСОК ЛИТЕРАТУРЫ……………………………………………………………..78](#_Toc12004238)

# ВВЕДЕНИЕ

Общественные и технологические условия современного мирового развития характеризуются постоянным возрастанием объема цифровых технологий, основывающихся на использовании искусственного интеллекта (ИИ), инновационными процессами в обществе в целом, а также в сферах производства, науки, бизнеса.

На сегодняшний день наиболее распространенными и популярными методологиям является Agile, Kanban, Trello и другие. Но они не обеспечивают в полной мере запросы управления проектами, как наукоемкими и оперирующими большими массивами данных.

Современный рынок информационных технологий предлагает большой выбор платформ для автоматизированного управления проектами. Ассортимент таких систем очень велик. Насчитывается около 100 автоматизированных систем управления процессом создания программных продуктов. Самыми популярными из них являются Trello, Jira и Asana. Каждый из предлагаемых продуктов имеет свои преимущества и недостатки. Так, Trello ориентируется на малые и краткосрочные проекты, хорошо зарекомендовала себя в работе с легко формализуемыми проектами; Jira удобна в использовании в больших командах, но при условии предварительной настройки и расширения базового набора функций с помощью плагинов, и надстроек. Данный процесс является достаточно трудоемким, требует много усилий. Сама Jira имеет сложную структуру и трудна в понимании для новых пользователей.

Поэтому становятся актуальными вопросы адаптации существующих моделей управления проектами к конкретной компании.

С точки зрения менеджмента, исследовательская работа является одной из самых сложных объектов управления. Это обусловлено особенностями творческого процесса мышления, неопределенностью и трудной предсказуемостью конечных результатов, невозможностью полной алгоритмизации процесса поиска новых решений.

Поиск путей разрешения указанного противоречия обусловил выбор **темы** диссертационного исследования: «Разработка информационной системы по постановке и контролю выполнения задач предприятия».

Таким образом, следующие факторы: постоянные и достаточно быстрые изменения в информационных технологиях; возникновение большого количества компаний, конкурирующих на рынке программных продуктов; усложнение разработок, увеличение доли исследовательского, творческого труда в работе IT-специалиста; увеличение степени неопределенности и риска повышают ответственность менеджмента всех уровней компании, обуславливают необходимость постоянного совершенствования методов управления проектами и **актуализируют** данное исследование.

**Объект исследования**: система управления проектами, для использовании задач предприятии.

**Предмет:** методы и средства моделирования автоматизированной системы управления проектами в компании, ориентированной на выпуск ИИ-

содержащих продуктов.

**Цель работы**: теоретическое обоснование и построение модели автоматизированной системы управления проектами на предприятии «ЖК Небо».

В соответствии с объектом, предметом и целью работы была сформулирована **гипотеза**: для разработки модели автоматизированной системы управления проектами на предприятии «ЖК Небо» могут быть использованы существующие методы и средства управления проектами.

Цель и гипотеза исследования обусловили решение таких **задач**:

* + 1. изучить современные методы и инструменты управления проектами, выяснить их преимущества и недостатки, особенности применения в различных IT-компаниях;
    2. проанализировать и сравнить существующие информационные системы по управлению проектами;
    3. выяснить особенности управления проектами в компании

«ЖК Небо», применяемые методологии и инструменты управления проектами;

* + 1. проектирование веб-сервиса;
    2. тестировать программу;

В структуру работы входят: введение, 4 главы, заключение и список

используемой литературы.

Работа изложена на 60 страницах и включает 23 рисунков, 6 таблиц, 30

источников.

# РАЗРАБОТКА И АНАЛИЗ СИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ

## Основные понятия, использующиеся в управлении проектами

*Проект* – это целенаправленное, ограниченное по времени и ресурсам мероприятие, ориентированное на создание уникального продукта или услуги. Менее формально, это некая новая идея, которая в результате определенной деятельности воплощается в жизнь в виде реального продукта (объекта), материального или интеллектуального. Соответственно, *программный проект* – предприятие по созданию уникальных программных продуктов.

*Управление проектами* – область деятельности, в ходе которой определяются и достигаются четкие цели проекта при балансировании между объемом работ, ресурсами (такими как деньги, труд, материалы, энергия, пространство и др.), временем, качеством и рисками. Ключевым фактором успеха проектного управления является наличие четкого заранее определенного плана, минимизации рисков и отклонений от плана, эффективного управления изменениями. Главной целью управления проектами является *решение задач проектов* в кратчайшие сроки, с наименьшими затратами, получением продуктов наилучшего качества.

Таким образом, *проект* – динамическая система действий, направленных на получение заданных результатов в многокритериальном поле в течение установленного срока и в рамках выделенных ресурсов с привлечением исполнителей, обладающих необходимыми навыками и знаниями.

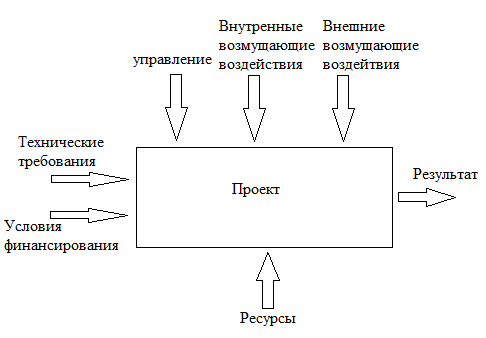


Рисунок 1.1 - Проект как система

Контекстная диаграмма (рис. 1.1) представляет проект как систему. Входными данными являются технические требования и условия финансирования; целью является достижение требуемого результата. Наличие ресурсов в виде материалов, финансов, человеческого ресурса обеспечивает выполнение работ. Эффективность определяется управлением процессом реализации проекта. В функции управления входит распределение ресурсов, координация выполняемой последовательности операций, компенсация возмущающих внутренних и внешних воздействий.

В таблице 1.1 раскрывается содержание составных элементов проекта. Таблица 1.1  Описание элементов системы «проект»

|  |  |
| --- | --- |
| **Элемент** | **Содержание** |
| Цель | Описываются новые продукты или услуги, которые получит заказчик в результате реализации проекта |
| Стоимость проекта | Финансовые затраты, требуемые для выполнения работ проекта |
| Технические требования | Объемы работ (количественные показатели объема работ проекта); сроки выполнения; качество (соответствие характеристик проекта и его продукции установленным заранее параметрам качества) |
| Ресурсы | Оборудование, материалы, персонал, программное обеспечение, информационные системы, производственные площади; специалисты и организации, привлеченные к выполнению работ проекта, их квалификация |
| Внутренние возмущающие воздействия | Стиль руководства проектом; организация проекта с точки зрения коммуникации между основными участниками проекта, распределения прав, ответственности и обязанностей; методы и средства взаимодействия между сотрудниками всех уровней на проекте; условия труда и техники безопасности, страхование и социальное обеспечение и т.п. |
| Внешние возмущающие воздействия | Взаимодействие с заказчиком и конкурентами; ситуация на рынке и связанные с этим риски; непредвиденные обстоятельства |

Таким образом, существует разнообразие в определении понятия

«управление проектами».

Однако отличительной чертой любого проекта является организация взаимодействия между участниками проекта, осуществляемая через project manager. В процессе коммуникации обозначается круг проблем, возникающих при реализации проекта и их возможные решения.

В результате обобщения существующих в источниках классификаций, построим таблицу 1.2, в левом столбце которой указано основание классификации, в правом – соответствующий вид проекта.

Таблица 1.2  Виды проектов

|  |  |
| --- | --- |
| Критерий | Вид проекта |
| по составу и структуре проекта | монопроект, мультипроект, мегапроект |
| по основным сферам деятельности, в которых осуществляется проект | технический, организационный, экономический, социальный, смешанный |
| по характеру предметной области проекта | инвестиционный, инновационный, научно- исследовательский, учебно-образовательный, смешанный |
| по продолжительности периода осуществления проекта | краткосрочные (до 2-х лет), среднесрочные (до 5-ти лет), долгосрочные (свыше 5-ти лет) |
| по масштабу (по размерам бюджета, количеству участников и степени влияния на окружающий мир) | мелкие, средние, крупные |
| степени охвата этапов инновационного процесса | полные инновационные проекты, включающий НИР, ОКР, освоение новшества и его коммерциализацию, неполные инновационные проекты, включающие отдельные этапы инновационного процесса |
| по сложности | простые, сложные, очень сложные |
| по отраслям экономики и социальной сферы | промышленность, строительство, транспорт, здравоохранение, туризм |
| по результирующему продукту | продукт (часть другого изделия, усовершенствование изделия или конечное изделие); услуга (бизнес-функция, оптимизирующая производственные процессы); улучшение (существующей линейки продуктов или услуг); нематериальный продукт (например, исследовательский проект приносит новые знания) |

Классификация проектов позволяет достаточно четко ранжировать перспективные и реализуемые проекты, и как следствие, ставить выполнимые цели, задавать реальные сроки достижения целей, и привлекать оптимально необходимые ресурсы для их успешной реализации.

Понятие ***жизненного******цикла***проекта подразумевает под собой определенную последовательность этапов по реализации той или иной идеи касательно производственного или управленческого процесса. Важность данного понятия обуславливается тем, что оно фиксирует продолжительность проекта, четко обозначая сроки его выполнения; позволяет детализировать процесс реализации замысла, разбивая его на конкретные фазы; дает возможность четко определить количество задействованного персонала, а также необходимые ресурсы; облегчает процедуру контроля.

Жизненный цикл проекта – это совокупность фаз, через которые реализуется первоначальный замысел. Такое разделение важно не только с теоретической, но также и с практической точки зрения, ведь оно дает возможность лучше контролировать процесс производства программного продукта. Так, принято выделять следующие этапы жизненного цикла IT- проекта: инициация, планирование, исполнение, завершение (рис. 1.2).

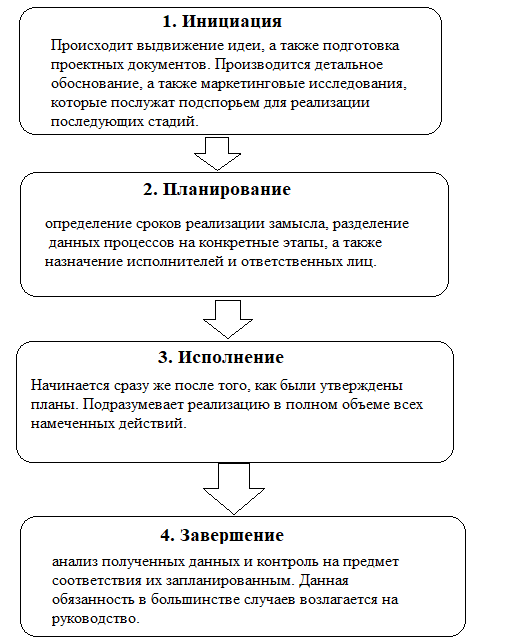


Рисунок 1.2 - Этапы жизненного цикла проекта

При изучении жизненных циклов проектов было замечено, что, несмотря на их различную направленность, всем проектам присущи следующие общие свойства:

* самые высокие показатели затрат и количество единиц персонала, присутствующего на проекте, присущи середине цикла; начало и конец данного процесса характеризуются невысокими показателями;
* уровень риска наиболее высок в начале жизненного цикла;
* именно в начале жизненного цикла проекта сотрудники имеют большую свободу по внесению изменений и совершенствования технологии реализации проекта, с течением времени это становится сделать все сложнее.

Как указывает Султанов И.А., «каждый project manager, набирая опыт, все больше понимает значимость жизненного цикла для того, чтобы проектная реализация с каждым разом проводилась все безопаснее и с более прогнозируемым результатом. В этом помогает не только система оценки рисков. Большое значение имеет планирование проекта по фазам его ЖЦ.

После каждого этапа намечаются вехи.

В эти моменты руководители обязаны остановиться, оценить достигнутый результат, осуществить прогнозный анализ и решить дальнейшую судьбу уникальной задачи.

Опыт, знания и управленческая интуиция одного из лидеров бизнеса позволяют доверить ему столь ответственные решения».

Методологическая ценность модели жизненного цикла в том, что она помогает понять особенности исполнения работы, вовремя принять нужные управленческие решения, поскольку модель – это подробное описание последовательности необходимых действий, что позволяет не терять контроль над разработкой проекта ни на одной стадии жизненного цикла.

## Модели жизненных циклов проектов и методы управления проектами

Управление проектами как вид профессиональной деятельности и как объект научных исследований получает существенное развитие в 1980-х годах, когда мировая экономика выходит из кризиса, растет насыщение рынка и возникает необходимость решения новых, больших задач. Именно тогда выходит первая значительная работа PMBoК (A Guide to the Project Management Body of Knowledge), выполненная в Project Management Institute (PMI). Данная организация на сегодняшний день является наиболее авторитетной профессиональной ассоциацией, разрабатывающей стандарты в области управления проектами. Среди других институтов, занимающихся стандартизацией проектного менеджмента, следует назвать IPMA (International Project Management Association), OGC (The Office of Government Commerce, стандарты PRINCE2), ISO (International Standartization Organization), APM (Association for Project Management) и др. Следует также отметить, что, кроме международных стандартов, существуют также национальные системы стандартизации, отраслевые и корпоративные.

Развитие IT-сферы, интенсивная работа в области программного обеспечения привели к накоплению (и этот процесс не останавливается) большого практического опыта («best practice»). Комплекс таких «лучших практик», реализуемых на различных стадиях жизненного цикла проекта и базирующихся на общей идеологии, стандарт SWEBOK называет

«методология разработки программного обеспечения». Методологии, или методы разработки программных продуктов на сегодняшний день являются самой быстро развивающейся областью информационных технологий, так как опираются на реальные практические знания.

Методы управления проектами тесно связаны с жизненным циклом проекта. Можно сказать, что они взаимно обуславливают друг друга. Фазы жизненного цикла проекта, представленные на рис. 1.2, являются весьма обобщенными и одинаково присущими проектам, выполняемым, например, в

строительной сфере, где все этапы четко обозначены и понятна последовательность разных видов работ, или, к примеру, в авиационной или космической области, где возможно распараллеливание операций, или повторение (если есть необходимость) одних и тех же видов работ.

Поэтому при разработке конкретного программного продукта необходима подробная детализация его жизненного цикла и адекватная

(содержанию проекта) модель.

«В общем случае, жизненный цикл определяется моделью и описывается в форме методологии (метода). Модель или парадигма жизненного цикла определяет концептуальный взгляд на организацию жизненного цикла и, часто, основные фазы жизненного цикла и принципы перехода между ними. Методология (метод) задает комплекс работ, их детальное содержание и ролевую ответственность специалистов на всех этапах выбранной модели жизненного цикла, обычно определяет и саму модель, а также рекомендует практики (best practices), позволяющие максимально эффективно воспользоваться соответствующей методологией и ее моделью».

Традиционными и хронологически первыми были разработаны каскадная (водопадная, waterfall) модель, впервые описанная в 1970 году в работе Уинстона Ройса и спиральная (Spiral), предложенная Барри Боэмом в 1986 г. [40]. Позже, с усложнением информационных технологий и нарастающим функционалом программных продуктов возникли так называемые гибкие (Agile) модели.

В настоящее время они получили наибольшее распространение.

Рассмотрим современные методы и алгоритмы управления проектами и проведем их сравнительный анализ по наиболее существенным параметрам.

1. Каскадная модель (Waterfall).

Самой старой и известной моделью построения многоуровневого процесса разработки является каскадная (или водопадная) модель: в ней каждый этап разработки, соответствующий стадии жизненного цикла ПО,

продолжает предыдущий. То есть новый этап начинается только после полного завершения текущего.

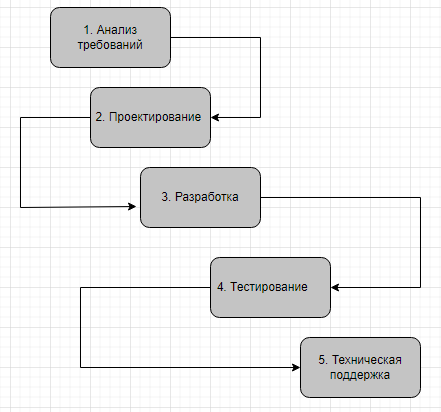


Рисунок 1.3 – Каскадная модель жизненного цикла

Каскадная модель проста и понятна, но не так практична, как раньше. В условиях динамично изменяющихся требований, строго структурированный процесс может из преимущества превратиться в помеху на пути успешного завершения разработки системы. Поэтому сегодня водопадная модель применяется преимущественно крупными компаниями для больших и сложных проектов, которые предполагают всеобъемлющий контроль рисков.

Преимущества каскадной модели:

* понятная и чёткая схема рабочего процесса;
* возможность просчёта точного количества затраченных на проект ресурсов;
* не требует затрат по налаживанию коммуникаций между всеми членами команды.

Недостатки каскадной модели:

* приоритет формального подхода к последовательности процесса работы;
* невозможность внесения изменений заказчиком до окончания разработки продукта;
* в случае нехватки ресурсов страдает качество проекта из-за сокращения этапа тестирования.

Несмотря на то, что каскадная модель все еще используется, она уже утратила былые позиции. Сегодня ей на смену приходят более продвинутые модели и методологии разработки программного обеспечения.

1. Спиральная модель.

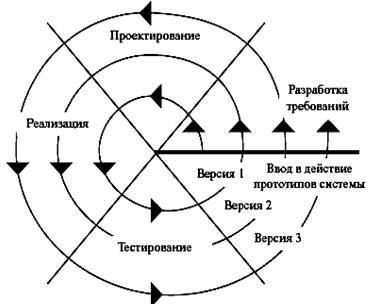


Рисунок 1.4 – Спиральная модель жизненного цикла

Спиральная модель использует разбиение проекта на итерации. Повышенное внимание уделяется начальным этапам разработки – анализу и проектированию. Основная функция начального этапа – обоснование возможности реализации программного решения; для этого создается прототип будущего продукта.

Свое название спиральная модель получила вследствие того, что жизненный цикл проекта в этой модели благодаря итерационному (то есть постепенно приближающему) подходу создает образ спирали – с преодолением каждой итерации, или отдельного цикла, будущее программное изделие приобретает требуемые формы и содержание.

Такой подход позволяет на каждой итерации уделять время уточнению целей и параметров проекта, выяснению возникающих затруднений, планированию работ на следующем витке.

Благодаря такому подходу к разработке время выполнения всех поставленных задач уменьшается.

Преимущества спирального подхода:

* преодоление жесткого, регламентированного процесса разработки в каскадной модели; создание дополнительных возможностей;
* облегчение процесса внесения корректив в проект, если требования заказчика изменились;
* постоянная связь пользователя будущего продукта с разработчиком, что позволяет оперативно отслеживать возникающие ошибки или слабые места проекта;
* более эффективное управление проектом благодаря наличию итераций;
* снижение уровня рисков.

Трудности использования спиральной модели:

* отсутствие ограничений времени на каждом из этапов жизненного цикла;
* то преимущество, что имеется возможность постоянно вносить изменения в требования к проекту, оборачивается тем, что количество циклов может расти, следовательно, удлиняются сроки разработки;
* довольно сложная структура модели, что создает затруднения для разработчиков и менеджеров при ее применении.

1. Гибкие (итеративные) модели (Agile).

Кроме последовательных моделей, существуют итеративные (инкрементальные). Это целое семейство методологий, объединенных общим подходом в управлении проектами:

* люди и взаимодействие важнее процессов и инструментов;
* работающий продукт важнее исчерпывающей документации;
* сотрудничество с заказчиком важнее согласования условий контракта;
* готовность к изменениям важнее следования первоначальному плану.

В таких моделях жизненный цикл проекта разбит на некоторое количество мини-циклов, каждый из которых состоит из базовых стадий модели жизненного цикла. Эти мини-циклы называются итерациями. Разработка отдельного компонента систем происходит внутри итерации, затем этот компонент добавляется к ранее разработанному функционалу.

Итеративная модель не нуждается в полном объеме требований для начала работ над продуктом. Разработка программы может начинаться с требований к части функционала, причем впоследствии требования могут претерпевать изменения и дополнения. Путем повторения процесса происходит обновление версии продукта на каждом цикле.

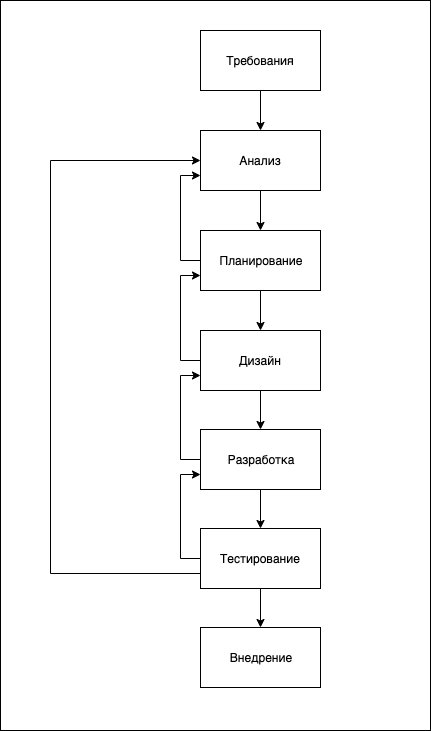


Рисунок 1.5 - Итеративная модель жизненного цикла

Решение об использовании результатов предыдущей итерации в качестве входных для следующей принимается по окончании предыдущей (т.н. инкрементальное прототипирование). В конечном итоге, достигается точка, в которой все требования были воплощены в продукте – происходит релиз.

Используя математическую терминологию, итеративная модель реализует последовательную аппроксимацию, то есть приближение к заданным параметрам, т.е. готовому продукту.

Успех использования этой модели определяется строгой валидацией требований и верификацией разрабатываемой функциональности на каждой итерации.

Основные стадии процесса разработки в итеративной модели фактически повторяют модель водопада. В каждой итерации создается программное обеспечение, требующее тестирования на всех уровнях.

Достоинства итеративной модели:

* высокий уровень взаимодействия между членами команды проекта;
* быстрый результат (рабочий код) в итоге «спринтов»;
* стимулирование изменения и улучшений продукта во время его разработки;
* непосредственное вовлечение заказчика в рабочий процесс;
* высокая способность к адаптивности под разные процессы и условия;
* быстрый отклик на изменения внешних и внутренних условий проекта;
* приспособлен для разработки инновационных продуктов с высокой степенью неопределенности и недостаточной информативностью;

Недостатки:

* риск бесконечных изменений продукта;
* большая зависимость от уровня квалификации и опыта команды;
* практически невозможно точно подсчитать итоговую стоимость проекта;
  1. **Обзор аналогов**

Рассмотрим некоторые существующие сервисы управления проектами и задачами проектов. Выявим преимущества и недостатки рассматриваемых сервисов, выполним их сравнение.

Наиболее востребованными представителями семейства Agile- методологии являются Scrum, Lean, Trello, Kanban, Six Sigma, PRINCE2. Рассмотрим подробнее Scrum, Trello и Kanban.

* + 1. **Scrum**

**Scrum** - наиболее структурированный в группе Agile-методов.

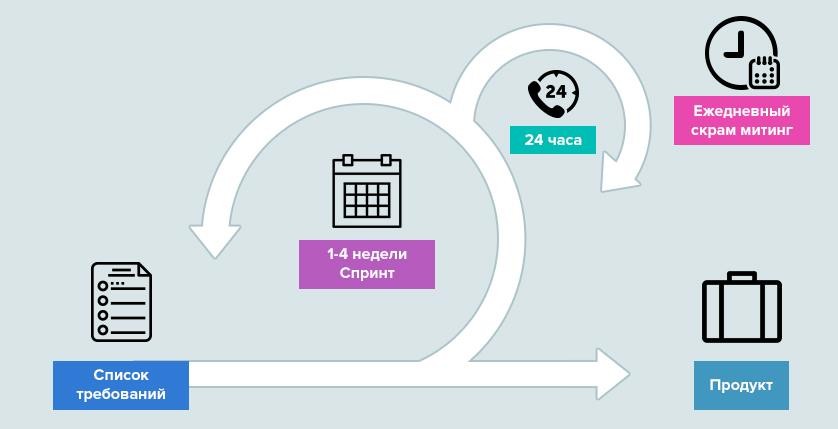


Рисунок 1.6 - Scrum модель жизненного цикла

Упрощенная схема работы по Scrum состоит в следующем. Согласно технологии Scrum, проект разбивается на части - задачи, которые будут создаваться в течение проекта. После этого полученным частям присваивается свой приоритет.

Хронологически проект разбивается на итерации - определенные промежутки времени, устанавливаемые командой и согласованные с заказчиком для текущего контроля и управления проектом.

Наиболее важные задачи первыми отбираются для выполнения в итерации.

По итогам итерации заказчик получает рабочий промежуточный вариант продукта. Чтобы убедиться в соответствии проекта требованиям заказчика, перед началом любого спринта нужно выполнять переоценку еще не выполненного содержания проекта и вносить в него изменения.

Участие в этом процессе принимает руководитель проекта, команда и инициатор. Ответственность распределяется на всех участников.

Процесс работы в среде Scrum опирается на пять производственных совещаний:

* совещание-упорядочивание; в первый день новой итерации происходит обсуждение того, что уже сделано по проекту, что еще предстоит выполнить;
* совещание-планирование; команда выбирает из списка приоритетных задач те, которые она будет выполнять на протяжении текущей итерации;
* совещания-летучки; происходят ежедневно, оперативно (до 15 минут) для обмена сведениями о текущем состоянии выполнения индивидуальных задач каждого члена команды;
* итоговые совещания; команда демонстрирует свой результат всем заинтересованным лицам; выясняется, насколько продукт соответствует целям и ожиданиям;
* совещание-ретроспектива. Проводится анализ того, насколько слаженно работала команда, обсуждаются возникшие проблемы методов и коммуникации, делаются определенные выводы.

***Преимущества Scrum:***

* удобен для проектов, в которых возможен быстрый, оформленный и пригодный для первоначального использования, результат;
* тесная и постоянная коммуникация сотрудников допускает присутствие в команде малоопытного разработчика;
* быстрое исправление ошибок благодаря мгновенной обратной связи между всеми участниками проекта.

***Недостатки Scrum:***

* подходит не каждому темпераменту; наличие постоянной и очень тесной коммуникации всех участников проекта выдвигает требования к психологическим свойствам и социальным навыкам участников проекта;
* не подходит для проектов из строго регламентированных и нормированных областей (например, юриспруденция).
  + 1. **Kanban**

В отличие от Scrum, Kanban-метод не предполагает наличие четких временных итераций. Процесс создания продукта не разделяется на итерации, а предполагает ограничение по количеству одновременно выполняемых задач.

Особенности метода состоят в том, что нет ограниченных по времени итераций, регламентированных производственных совещаний, каждый сотрудник команды может решать несколько задач одновременно. Отсутствие временных рамок каждого этапа позволяет приостановить выполнение текущей задачи, если изменился ее приоритет или появились иные, более важные задачи.

***Преимущества Kanban:***

* отсутствие дедлайнов;
* экономически эффективен вследствие разумного распределения количества задач на каждого члена команды.

***Недостатки Kanban:***

* наиболее эффективен в командах, члены которых взаимозаменяемы;
* не пригоден для проектов, сроки выполнения которых строго оговорены.
  + 1. **Trello**

Программное обеспечение **Trello** для управления Agile-проектами. Trello

-облачная программа для управления проектами небольших групп, разработанная Fog Creek Software.

Это бесплатная многоплатформенная система управления проектами (рис. 1.9, 1.10).

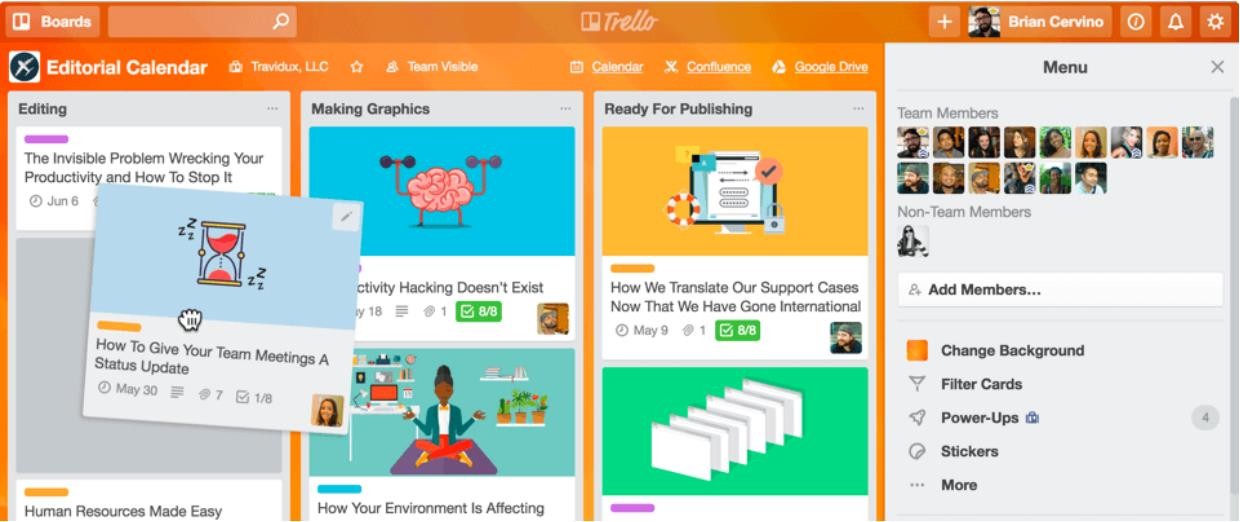


Рисунок 1.9 – Trello Board

Она использует парадигму управления проектами, известную как канбан. Проекты изображаются досками, содержащих списки.

Списки содержат карты, которыми изображаются задачи.

Карточки должны переходить из предварительного списка к следующему (с помощью перетаскивания), таким образом изображая движение какой-то функции от идеи, до тестирования.

Карточке может быть присвоено имя ответственных за нее пользователей.

Пользователи и доски могут объединяться в команды.

Trello имеет ограниченную поддержку тегов в виде шести цветных меток.

Карточки могут содержать комментарии, вложения, дату завершения и перечни (списки подзадач).

Форматируются карточки разметкой Markdown.

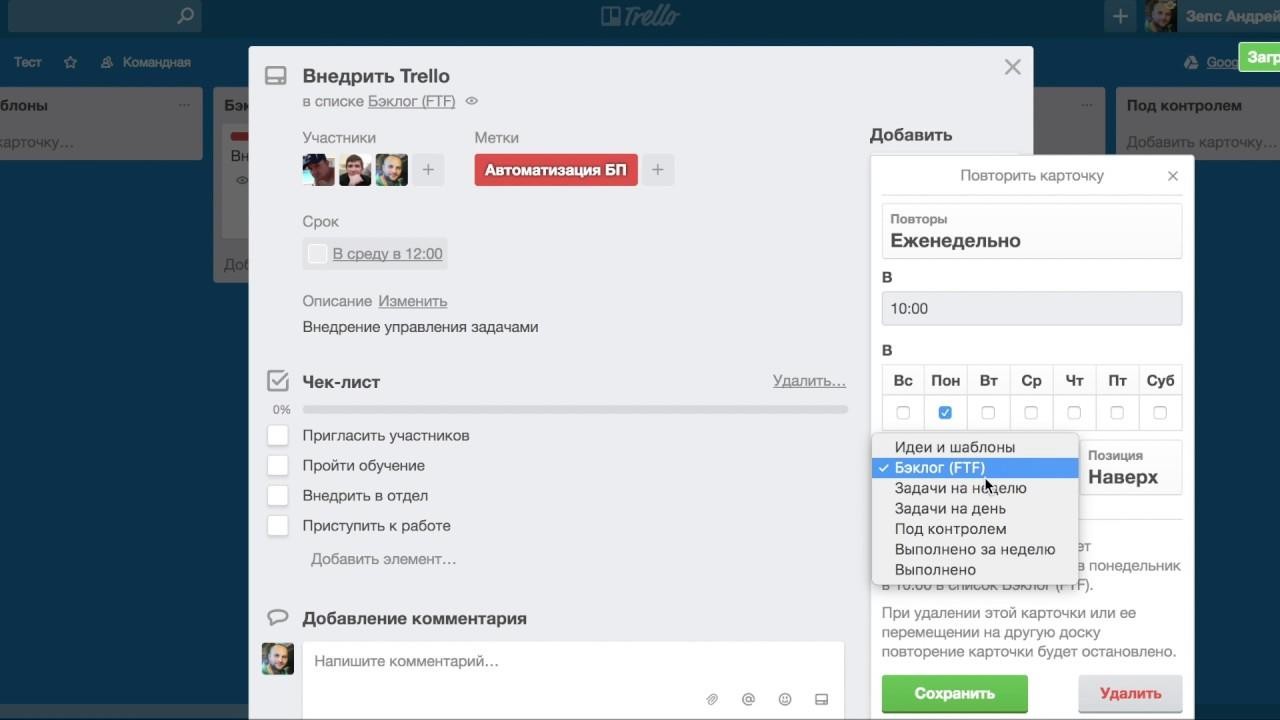


Рисунок 1.10 - Карточки задачи в Trello

***Преимущества работы с Trello*** заключаются в следующем:

* наличие функциональных возможностей, упрощающих процесс управления проектом, а именно: обмен файлами (включая фотографии и видео)

со своими членами команды; комментарии к карте; слежение за to-do list; установление цветных меток в соответствии с приоритетом; неограниченное создание карт под проект;

* наличие функциональных возможностей, упрощающих процесс управления проектом, а именно: обмен файлами (включая фотографии и видео) со своими членами команды; комментарии к карте; слежение за to-do list; установление цветных меток в соответствии с приоритетом; неограниченное создание карт под проект;
* стоимость. По сравнению с другими инструментами управления проектами, Trello имеет менее сложную структуру ценообразования. Более простой вариант версии позволяет пригласить неограниченное количество участников, создать доски, карты и списки. Версия бизнес-класса, стоимость которой 25 долларов США в месяц, предоставляет целый ряд функций, таких, как интеграция в Службы Google, легкий объемный экспорт, а также возможность доступа и управления всеми досками;
* понятный мониторинг времени. Вы никогда не пропустите дату завершения проекта или важную встречу с Trello Борд. При создании Cards можно добавить соответствующие даты (deadline). Карточка станет желтой за 1- 2 дня до окончания промежутка времени, отведенного на выполнение, карточка станет полностью красной в день сдачи проекта;
* mobile friendly. Trello работает на каждой платформе. Независимо от того, находится ли он на компьютере, планшете или телефоне, инструмент может адаптироваться под размер экрана;
* instant notifications. Вы всегда будете в курсе новостей, так как Trello оснащена функцией мгновенного оповещения при обновлении, комментировании или удалении задачи. Также можно получать оповещения по электронной почте.

Таблица 1.3  Обзор Trello

|  |  |
| --- | --- |
| Назначение продукта | Отслеживание задач проекта и их статусов. Управление небольшими проектами. |
| Users | Software developers Project managers SCRUM masters |
| Use Cases | Управление проектами Менеджмент продукта Управление задачами  Разработка программного обеспечения Agile разработка программного обеспечения |
| Интеграция | Confluence JIRA  Slack GitHub |
| Hosting options | Cloud |

***Недостатки работы с Trello*** следующие:

* нет offline-доступа. Trello не будет работать, если нет wifi. Если вы путешествуете или ваше электронное устройство переключено на режим «в самолете», вы не получите доступа к Trello . Доступ к сайту Trello требует подключение к Interwebs. Эти особенности, присущие Trello, создают определенный дискомфорт для пользователя, находящегося офлайн;
* данная платформа не предназначена для работы над крупными проектами. Оптимальное ее применение - стартапы, домашние проекты или некоторая идея, которая требует проверки. Trello - неудобный инструмент управления масштабными проектами с несколькими командами со всего мира;
* ограниченный объем файлохранилища. Количество вложенных файлов на карте Trello может быть любым, но для каждого вложенного файла существует лимит загрузки файла до 10 МБ. Члены бизнес-класса и Trello Gold могут пользоваться ограничением загрузки файлов размером до 250 Мб. Лимита хранения данных учетной записи не существует;
* неудобство комментирования. После опубликования и сохранения комментария на карте редактировать его в случае необходимости невозможно, придется писать новый комментарий;
* недостаточный уровень визуализации процесса работы всех членов команды одновременно;
* невозможно создать долгосрочные планы. Доска проста и удобна для разработки программного обеспечения небольших проектов, но для проектов длительностью даже в две недели создать дорожную карту оказывается невозможным;
* нет возможности просматривать итерации. Эффективность работы повышается при условии регулярного наблюдения за выполнением текущих задач, продуманного плана контроля предстоящих заданий, а также возможности анализа и корректирования уже выполненной работы. Trello не располагает встроенными инструментами для просмотра и анализа реализованных задач для успешного проведения ретроспектив;
* нет возможности вносить данные о затраченном на задачу времени. Отсутствие возможности для менеджеров проекта и его участников отслеживать количество времени, затраченное на выполнение каждой задачи. Это затрудняет составление ежемесячной отчетности, а также осложняет контроль за выполнением задачи.

**1.3.4 Сравнение аналогов**

В табл. 1.1 приведены характеристики аналогов, а также планируемые показатели разрабатываемого сервиса, который предполагается представить под названием *Itrack*. Предполагаемое преимущество в том, что предусматривается широкий функционал, но без ущерба простоте использования.

***Сравнительная характеристика программ-аналогов***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Характеристика | Scrum | Kanban | Trello | Itrack  (план.) |
| Ведение списка исполнителей по задачам | Да | Да | Нет | Да |
| Прикрепление файлов к проекту | Да | Да | Да | Да |
| Отслеживание просрочек | Нет | да | Да | Да |
| Пользовательские роли | Нет | Да | Нет | Да |
| Возможность сдвига сроков задач | Да | Да | Нет | Да |
| Учёт приоритетов задач | Нет | Да | Да | Да |
| Русскоязычный интерфейс | Да | Нет | Да | Да |
| Визуализация связей задач | Да | Нет | Да | Нет |
| Временные задержки отклика | Да | Нет | Нет | Нет |
| Бесплатность | Условная | Условная | Да | Да |
| deadline | Нет | Нет | Да | Да |

Мы вынуждены признать, что разрабатываемый веб-сервис будет учитывать далеко не все возможные характеристики задач проектов, однако он может предоставить наиболее важные функции управления и задачами так, чтобы в большинстве случаев было проще использовать наш продукт, нежели другие. По крайней мере, наш сервис должен быть проще в использовании при управлении типовыми программными проектами (например, учётными системами на предприятиях, обучающими системами). Заметим, что в отличие от рассмотренных аналогов, *Itrack* не будет иметь критических недостатков, например, проблем с отслеживанием просрочек, установка время начала и окончания проекта.

## Анализ инструментов управления проектами

Инструменты, используемые менеджером проекта, могут применяться как в течение всего проекта, так и на определенном этапе (фазе) жизненного цикла.

Под *инструментами управления* мы будем понимать законченные формализованные методики, процедуры, а также шаблоны необходимых проектных документов.

Программное обеспечение **ASANA** для управления Agile-проектами. Asana (Асана) — одно из ведущих мобильных и веб-приложений для управления проектами в небольших командах.

Это универсальное решение для управления проектами и задачами, которое позволит автоматизировать некоторые из наиболее сложных задач, таких, как коммуникация и удобное сотрудничество. Многие компании хвалят Asana за точность и эффективное использование времени, особенно отмечая удобство отслеживания задач и обсуждения их в реальном времени (рис. 2.9).

Основной акцент создатели сервиса делают на том, что теперь управлять проектами возможно и без использования электронной почты. Разработчики

Asana предоставили пользователям возможность обсуждать рабочие процессы на той же платформе, где они были созданы, и усовершенствовали программу дополнительным функционалом: notes, groups, combined tasks, and followers.

Asana мгновенно сообщает о каждом изменении, которое может оказать влияние на проект. Программное обеспечение помогает сделать качественную оценку проекта и управлять рисками до момента подписания контракта с клиентом.

Сервис предлагает синхронизацию в режиме реального времени для различных устройств; проверку вашего почтового ящика; создание плана на грядущий день; просмотр, редактирование и создание задач и проектов с приоритетами; структурирование проектов с помощью разделов; лайки к задачам; обсуждения по проекту и задачам; настраиваемые дэшборды; календари; поиск задач, проектов, людей и тегов; прикрепление файлов из Dropbox, Box или Google Диска; канбан-доски.

Основные характеристики Asana представлены в таблице 1.4. Таблица 1.4  Обзор Asana

|  |  |
| --- | --- |
| Назначение продукта | Планирование, документирование, отслеживание и выпуск программного обеспечения. |
| Users | Software developers Project managers SCRUM masters |
| Use Cases | Управление проектами Менеджмент продукта Управление задачами  Разработка программного обеспечения Agile разработка программного обеспечения |
| Интеграция | JIRA  Slack GitHub |
| Hosting options | Cloud |

Рассмотрим *преимущества* Asana:

* расширенный список функций управления;
* существует бесплатная пробная версия. Сервис также бесплатен для команды до 15 человек;
* обозначение цветом наиболее важных задач. Удобный функционал для making priority for task (рис. 1.11);
* неограниченный функционал добавления более одного тега к каждому элементу (рис. 1.12);
* поддержка мобильной версии для различных платформ;
* одна история / элемент / задача может иметь ссылки на более чем один проект и тег;
* есть поддержка горячих клавиш;
* добавление новых задачи без необходимости заполнения многих полей;
* интеграция электронной почты;
* четкое дерево задач, которые нужно выполнить. Возможность увидеть все задачи в одном месте.
* платформа «Премиум» стоит 10 долларов США за каждого члена/в месяц (в случае, если вы хотите иметь дополнительный и расширенный функционал).

*Недостатки* Asana следующие:

* можно назначить историю/задачи только одному из членов вашей команды;
* графические детали перегружают интерфейс, делая его медленным при перемещении;
* трудно работать с задачами в задании (sub-tasks). Есть возможность потерять связь при перемещении элементов;
* чрезмерное количество функций, которые отображаются на экране, затрудняют и замедляют процесс обучения.
* сообщения через программу зашифрованы, но, к сожалению, не существует двухэтапной проверки. Отсутствие two-step verification делает программное обеспечение небезопасным для ваших личных данных [37].

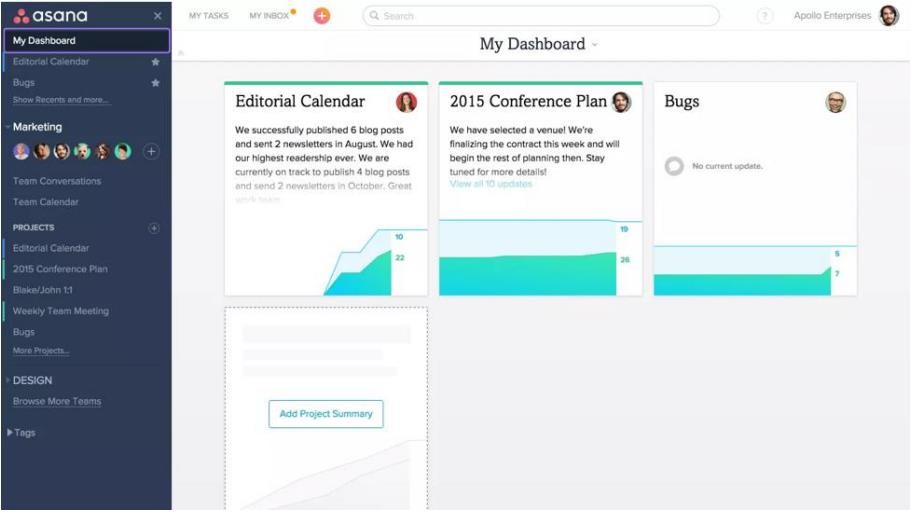


Рисунок 1.11 - Asana Dashboard

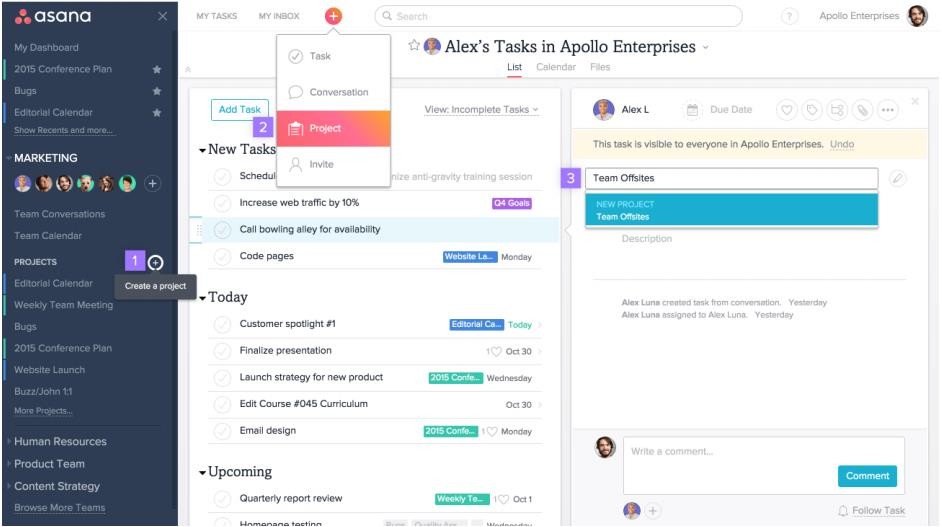
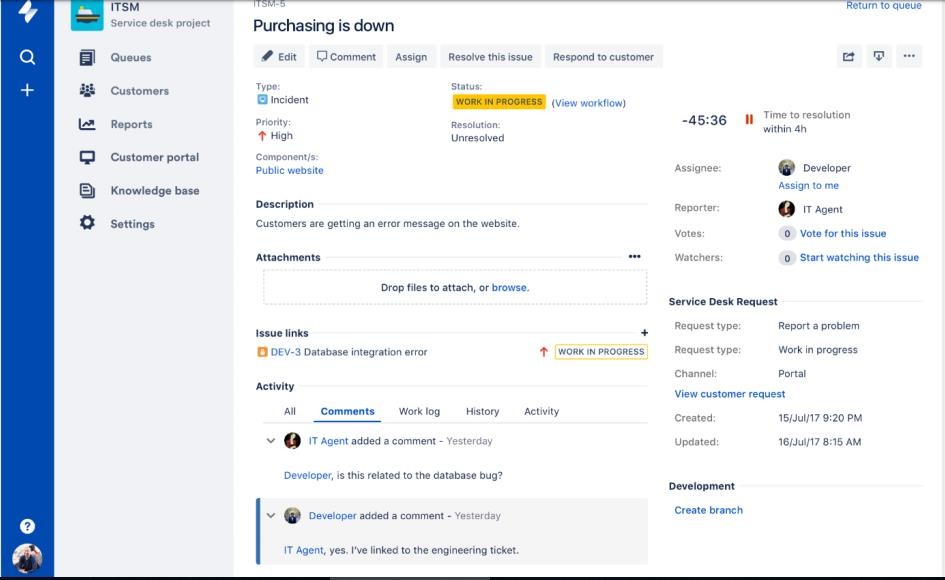


Рисунок 1.12 - Открытая задача в Asana

Программное обеспечение **JIRA** для управления Agile-проектами. JIRA - система отслеживания ошибок. Данный продукт разработан для организации взаимодействия с пользователями, также ее можно использовать и для управления проектами. Разработчик указанной системы - компания Atlassian.

Jira является одним из двух её основных продуктов (наряду с вики-системой Confluence). Одним из достоинств Jira является наличие веб-интерфейса [16].

JIRA — это комплекс систем, позволяющий координировать работу всех команд, занятых разработкой нового продукта. JIRA предлагает несколько продуктов и вариантов развертывания, предназначенных специально для программного обеспечения, ИТ, бизнеса, команд операторов и тому подобное.



Риcунок 1.13 - Открытая задача в JIRA

Продукты и программы, построенные на платформе JIRA, помогают командам планировать, назначать, отслеживать, отчитываться и управлять работой. Платформа JIRA объединяет команды во всем: от гибкой разработки программного обеспечения и поддержки клиентов до управления списками покупок и семейными делами (рис.1.13).

Четыре продукта построены на платформе JIRA: JIRA Software, JIRA Service Desk, JIRA Ops и JIRA Core. Каждый продукт поставляется со встроенными шаблонами для различных случаев использования и легко интегрируется, поэтому команды из разных организаций могут работать лучше вместе.

Рассмотрим основные характеристики JIRA Software.

Рабочие процессы представляют собой последовательный путь от зарождения идеи до ее реализации.

Схема базового рабочего процесса показана на рис. 1.14.

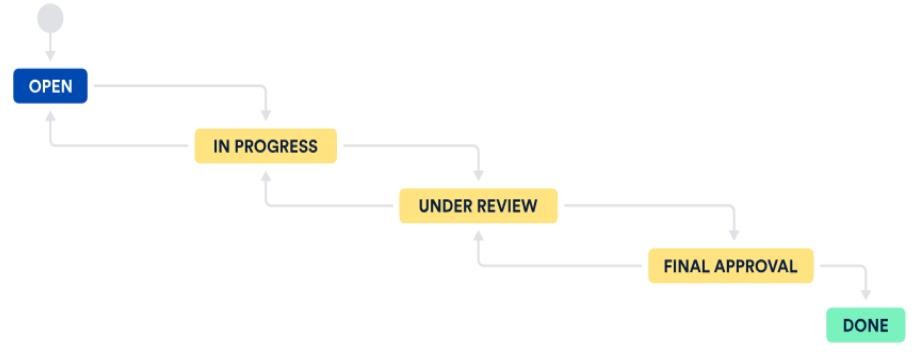


Рисунок 1.14 - Jira Workflow

В этом случае Open, Done и метки между ними отражают статус, в котором может возникнуть проблема, а стрелки - потенциальные переходы от одного состояния к другому.

Рабочие процессы могут быть простыми или сложными, с условиями, триггерами, валидаторами и функциями сообщение.

Рекомендуется для администраторов, начинающих использовать JIRA Software, сохранять максимально простой вариант своих рабочих процессов до тех пор, пока при реализации проекта не возникнет необходимость в усложнении конфигурации рабочего процесса.

Таблица 1.5 содержит сведения о важных характеристиках продукта Jira Software, а именно:

* назначение;
* потенциальных пользователях;
* сфере применения;
* возможности интегрирования с другими системами;
* доступных опциях для хостинга.

Таблица 1.5  Обзор JIRA Software

|  |  |
| --- | --- |
| Назначение продукта | Планирование, отслеживание и выпуск программного обеспечения мирового уровня. |
| Users | Software developers Project managers SCRUM masters |
| Use Cases | Отслеживание ошибок Управление проектами Менеджмент продукта Управление процессами Управление задачами  Разработка программного обеспечения Agile разработка программного обеспечения |
| Интеграция | Confluence Bitbucket Slack GitHub |
| Hosting options | Cloud, Server, Data Center |

Рассмотрим *положительные стороны* использования JIRA Software:

* хорошая видимость (visibility). Одним из факторов, замедляющих осуществление любого проекта (не только в области разработки программного обеспечения, но и в других профессиональных сферах, да и просто в жизни), является отсутствие четкой постановки задач и выстраивания иерархии их выполнения. JIRA устраняет эту проблему, поскольку она объединяет команды таким образом, что все члены команды получают возможность видеть продвижение в выполнении задач другими сотрудниками в режиме реального времени, так как задачи имеют теги «started» и «completed». Это помогает всем членам команды знать, на какой стадии находится проект, что ускоряет работу над программным продуктом;
* удобное определение приоритетов. Еще одно преимущество использования JIRA заключается в том, что она позволяет лучше определить порядок приоритетов задач, стоящих перед всеми членами команды, вследствие чего можно увидеть, какие задачи необходимо выполнить немедленно и какие

могут быть решены позже. Это очень полезно с точки зрения соблюдения сроков, особенно при работе над различными проектами с разными датами завершения;

* повышение производительности. Используя JIRA, члены команды получают возможность в любой момент времени видеть последовательность задач в списке (backlog), вследствие чего уменьшается время простоя, потраченное на обсуждение текущих проблем, и повышается производительность работы. Хотя время простоев может показаться незначительным, тем не менее общее их количество может привести к превышению времени, отведенного на выполнение задачи. Таким образом, JIRA способствует устранению этой проблемы и повышению производительности труда;
* обеспечение непрерывности взаимодействия между членами команды, где бы они ни находились. Еще одним преимуществом программного обеспечения JIRA является то, что JIRA поставляется с доступными мобильными приложениями. Это означает, что все члены команды могут оставаться на связи не только в офисе или дома через ноутбук, но и используя мобильные телефоны и планшеты;
* более 1000 плагинов. JIRA поставляется с более чем 1000 приложениями, которые помогут сделать программное обеспечение еще более полезным для команд гибкой разработки, два наиболее популярных из них - GreenHopper и Bonfire.

Рассмотрим *недостатки* использования JIRA:

* большая насыщенность функциональными возможностями, что создает дополнительные трудности не только для начинающих пользователей, но и для продвинутых, так как процесс кастомизации продукта оказывается достаточно трудоемким в силу большого количества настраиваемых параметров системы. Таким образом, требуется некоторое время, чтобы понять архитектуру программного обеспечения;
* сложный UX/UI. Обилие визуальной информации, а также наличие не всегда интуитивно понятных элементов интерфейса создает пользователю трудности взаимодействия с JIRA;
* требование наличия навыков DevOps, знания Scrum, уметь строить спринт для создания структуры работы;
* отсутствие встроенной функциональности управления календарем;
* высокая стоимость, которая составляет для команды до 10 человек 10 долларов США в месяц, до 100 человек - 7 долларов США за каждого члена в месяц.

Для лучшей организации совместной работы вместе с Jira в IT-компаниях обычно используют Confluence, так как это продукты одной компании Atlassian и они хорошо интегрированы друг с другом. Рассмотрим достоинства и недостатки работы с Confluence.

Confluence – платформа, позволяющая публиковать web-страницы и документов в стиле wiki, а также обмениваться контентом между участниками команды и организовать обсуждение.

Назначение Confluence:

* централизация информационных потоков;
* полное обслуживание файловой системы и документации (создание, хранение, просмотр, обмен, редактирование);
* функция поиска документов или другой рабочей информации, например, писем;
* встроенный корпоративный календарь;
* постановка задач и контроль процесса исполнения;
* корпоративные чаты.

Среди недостатков данного программного продукта пользователи указывают следующие:

* неудобная система комментариев;
* плохо организованная система одновременного редактирования текстов;
* не налажено отслеживание развития проекта;
* неудобный тайм-менеджмент;
* отсутствует управление ресурсами;
* система ценообразования; цена на продукт растет с ростом количества пользователей .

Как видно, возможности Confluence достаточно большие. Разработчики постарались учесть все потребности работы в офисе.

## 1.4 Выводы

На основании рассмотренных моделей жизненных циклов проекта, моделей и средств управления проектами можно сделать следующие выводы. Модель жизненного цикла проекта Trello наиболее адекватно отражает процессы разработки. Для управления процессами разработки программного продукта с использованием современной технологией наиболее оптимальным будет использование комбинации гибкой модели и модели Trello.

Гибкая модель является наиболее подходящей для IT-компаний, стартапов, проектов в инновационных сферах, модель Trello позволяет управлять деятельностью отдела R&D; каскадная модель удобна в проектах, где ключевым ограничителем является срок реализации проекта, а не финансы.

Параллельное использование данных методологий позволит оптимизировать процессы и синхронизировать принципиально разную деятельность команд, работающих над созданием одного продукта.

1. **МЕТОДЫ И АЛГОРИТМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ В КОМПАНИИ «ЖК НЕБО»**

## Алгоритмы процесса разработки программных продуктов и методы управления проектами на предприятии «ЖК Небо»

Компания Одна из крупнейших российских девелоперских компаний. На протяжении 28 лет специализируется на строительстве самых сложных и знаковых объектов Москвы – многофункциональных комплексов, жилой и коммерческой недвижимости. Профессиональная экспертиза высокого профиля, накопленный опыт, использование качественных и современных технологий обеспечили строительным проектам холдинга Capital Group признание в профессиональной и архитектурной среде. В портфеле компании – 9 млн кв. м реализованных, строящихся и проектируемых объектов. Capital Group – признанный лидер в области строительства жилья класса бизнес, элит и премиум, а также высотного строительства. Среди знаковых проектов компании – «Лица», «Небо», «Легенда Цветного», «Mon Cher», «Город Столиц», «ОКО», «Кутузовский 12», «Capital Towers», «The Book», «White Khamovniki», проект редевелопмента на территории Бадаевского пивоваренного завода, первый в России отель Mandarin Oriental Hotel, Москва и первые в стране брендированные резиденции с юридическим статусом квартир на Софийской набережной и другие. Проекты Capital Group неоднократно отмечались как лучшие на профессиональных российских и международных премиях.

В настоящее время компания работает над проектом для сферы информационной технологии.

Процесс изготовления прикладного программного обеспечения, использующего новых технологий, подразумевает более сложную структуру по сравнению с процессом изготовления типичного программного продукта.

Ниже представлена укрупненная блок-схема алгоритма создания любого программного продукта - как содержащего современных технологий, так и стандартного (рис. 2.1).

Данный процесс включает в себя пять стадий: от анализа условий поставленной задачи до сопровождения готового изделия в процессе его эксплуатации заказчиком.

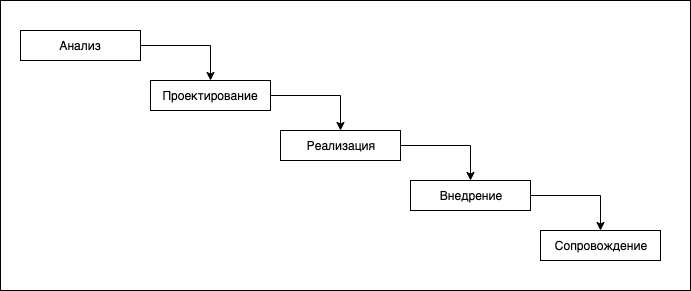


Рисунок 2.1  Блок-схема рабочего процесса разработки программных

продуктов

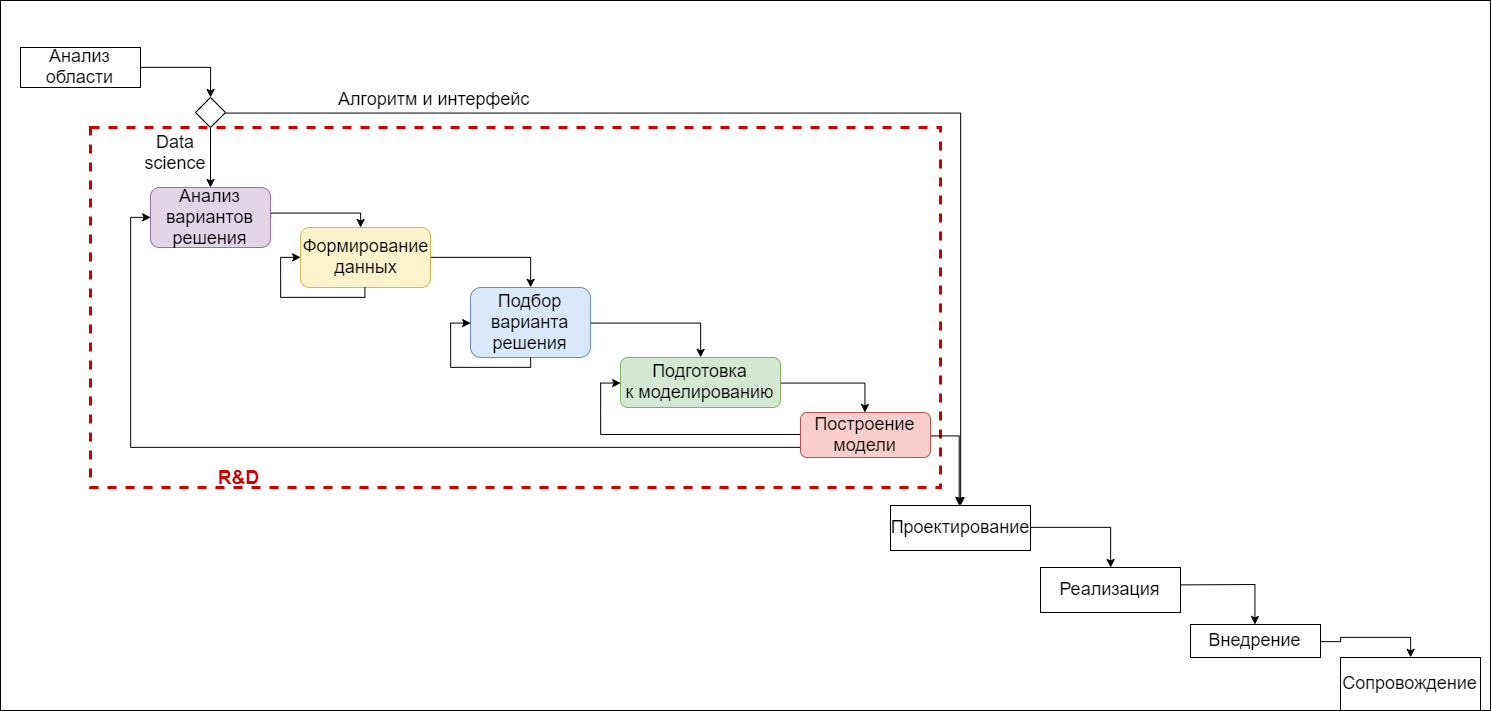


Рисунок 2.2  Блок-схема рабочего процесса разработки программного продукта, содержащего современных технологий

Однако более подробная детализация блок-схемы (рис. 2.2) в случае работы над созданием сервисов, программ, платформ с использованием современных технологий позволяет выделить в рабочем процессе целый этап, который предшествует непосредственно проектированию программного решения - исследовательскую деятельность (Research and Development - R&D), на рисунке 2.2 обозначен красным.

Такой тип деятельности присущ Data Science-проектам и направлен на исследование возможных вариантов решения поставленной задачи, а также разработку и обучение модели для каждого из вариантов.

Особенностью такой деятельности является то, что после этапа анализа предметной области невозможно однозначно переходить к этапу проектирования конечного программного продукта (как это происходит при решении стандартных задач), так как выходные данные, полученные после этапа анализа, могут быть многократно пересмотрены и уточнены.

Уточненный алгоритм этапа R&D представлен на рисунке 2.3.

Как показано на рисунке 2.3, в отличие от процесса реализации традиционных решений, жизненный цикл проектов усложняется необходимостью проведения ряда подготовительных работ, которые имеют циклический характер. Количество этих циклов заранее неизвестно и зависит как от степени сложности поставленной задачи, так и от квалификации сотрудников на проекте и уровня развития их творческого мышления.

Более того, возможен исход, при котором детальное изучение поставленного задания показывает, что оно не имеет решения.

Узел принятия решения, обозначенный \*1, отражает тот момент исследования, когда команда, основываясь на своем опыте и интуиции, выбирает направление, связанное либо с изменением параметров модели, либо с изменением «features», либо с поиском нового варианта решения поставленной задачи.

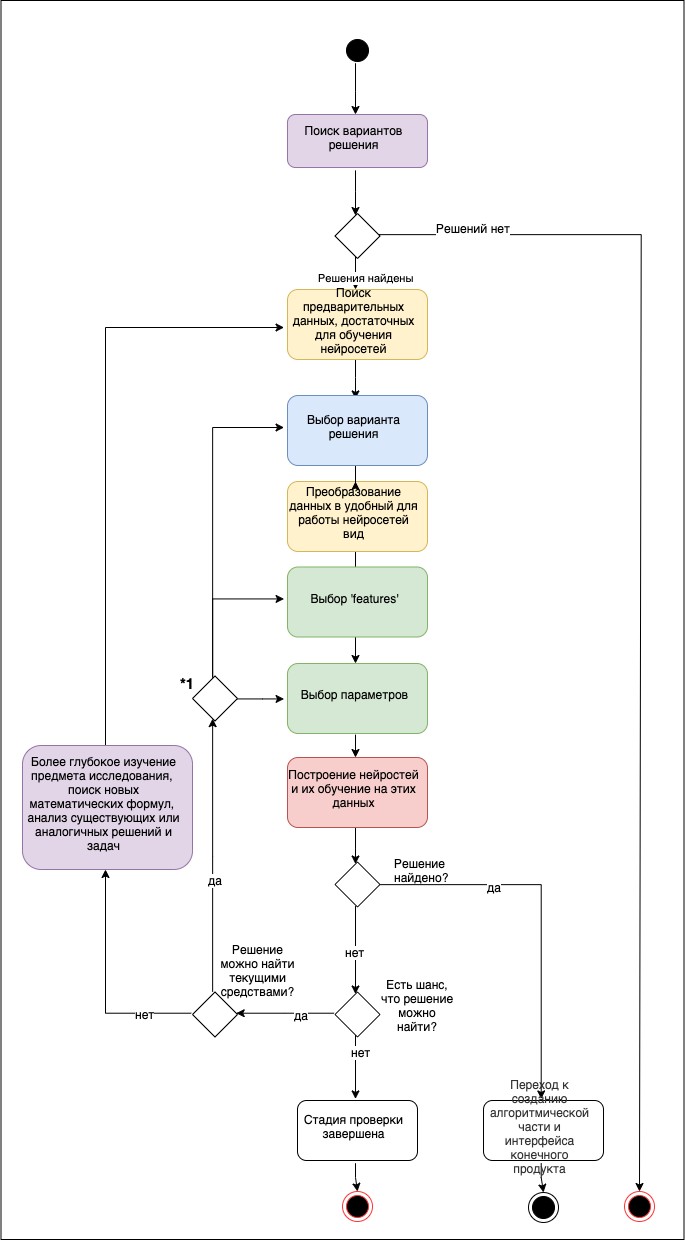


Рисунок 2.3 - Блок-схема алгоритма R&D

Циклический характер процесса создания продукта существенно отличается от Agile-моделей (Scrum, Kanban и др.), основу которых составляют итерации, на каждой из которых мы получаем некоторую законченную часть целевого программного продукта.

В случае с проектами завершение каждого цикла не всегда означает создание части или функции конечного продукта; напротив, нередко бывает, что исполнители проекта вынуждены вернуться к точке нулевого отсчета и начать заново работу с исходной информацией.

Кроме того, деятельность R&D основывается на работе с большими массивами данных, причем на протяжении всего жизненного цикла проекта.

Это ключевое обстоятельство вынудило специалистов Data Science искать специализированные методы работы с Big Data.

На данный момент в компаниях, специализирующихся на создании Data Science-продуктов, наиболее распространенной является CRISP-DM - методология по исследованию данных.

Таким образом, деятельность команды в процессе создания продукта, состоит из исследовательских действий, которые всегда носят уникальный, оригинальный характер, и стандартных действий по проектированию, реализации и внедрению.

## Организация процесса исследовательской деятельности на предприятии «ЖК Небо»

Рассмотрим организационную структуру компании «ЖК Небо». В компании работает 15 человек, распределенных между такими отделами: финансовый -2 чел., отдел DevOps - 6 чел., специалисты по данным (Data Engineering) - 2 чел., отдел R&D - 4 чел., технический директор CTO, исполнительный директор CEO, Product Owner; разработчик сайтов готового продукта - 1 чел.

Сотрудники отдела R&D владеют следующими знаниями и умениями.

Высшее образование по прикладной математике, алгоритмы и структуры данных, финансово-экономические знания, языки программирования и фреймворки Laravel, Docker Swarm, PHP, JS, Git, NginX, DevOps , C++, NodeJS, MySQL, Redis, Bash script.

Директора компании имеют высшее техническое образование, владеют некоторыми из перечисленных языков программирования и фреймворков.

Начальник R&D-отдела имеет степень PhD по математике.

Кроме того, технический директор выполняет обязанности архитектора проектов, а также непосредственно руководит отделами.

Исполнительный директор CEO совмещает обязанности Project Manager. Его задачи следующие:

* постановка заданий для проектной команды;
* назначение исполнителей для конкретных задач;
* отслеживание выполнения;
* создание проектной документации и т.д.

R&D-процесс подразумевает большую вовлеченность проектной команды в задачи, которые по Agile-методологиям (и другим распространенным общепринятым методологиям) выполняет проектный менеджер.

Как видно из приведенных сведений, сотрудники компании представляют собой высококвалифицированных специалистов, и обладают перекрестными умениями, что является необходимым условием работы в сфере производства ИТ-продуктов.

В процессе работы над программным продуктом все сотрудники, включая менеджмент, тесно связаны между собой, и поддерживают постоянную связь с заказчиком. Каждое утро проводятся стендап-митинги, на которых обсуждаются результаты прошедшего дня и планируются задачи на текущий день.

При этом также принимаются решения об отклонении промежуточных задач или изменении постановки задачи, если не удалось найти ее решение. Каждую неделю проводятся брейн-штормы, где на повестку выносятся задачи, выполнение которых кажется невозможным. В обсуждении поиска возможных вариантов решения принимает участие весь состав компании, в особо затруднительных случаях могут привлекаться сторонние специалисты.

Следует заметить, что результаты проводимых митингов практически не документируются.

Возникающие в процессе обсуждения мысли, идеи, выводы, предложения записываются на листочках, с помощью стикеров на доске, а порой вообще не фиксируются. Пока компания небольшая, подобная практика не является катастрофичной, но тем не менее на сегодняшний день существует большое упущение в процессах управления проектами.

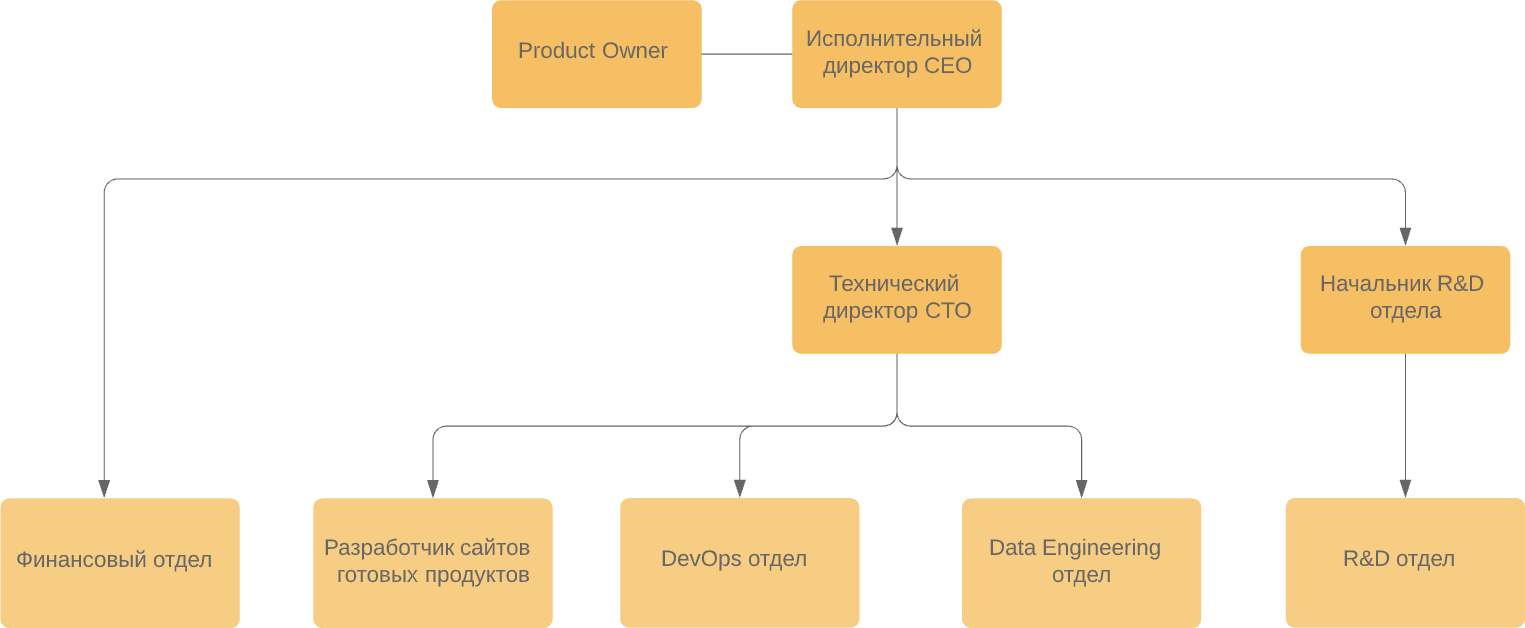
Схема организационной структуры предприятия в общем виде представлена на рисунке 2.4.

Рисунок 2.4 - Общая организационная структура предприятия «ЖК Небо»

Рассмотрим, как происходит создание программного продукта в компании с точки зрения разделения труда.

Анализ организации процесса разработки проектов в компании «ЖК Небо» на этапе R&D показал, что она имеет сложную нелинейную структуру (рис. 2.5).

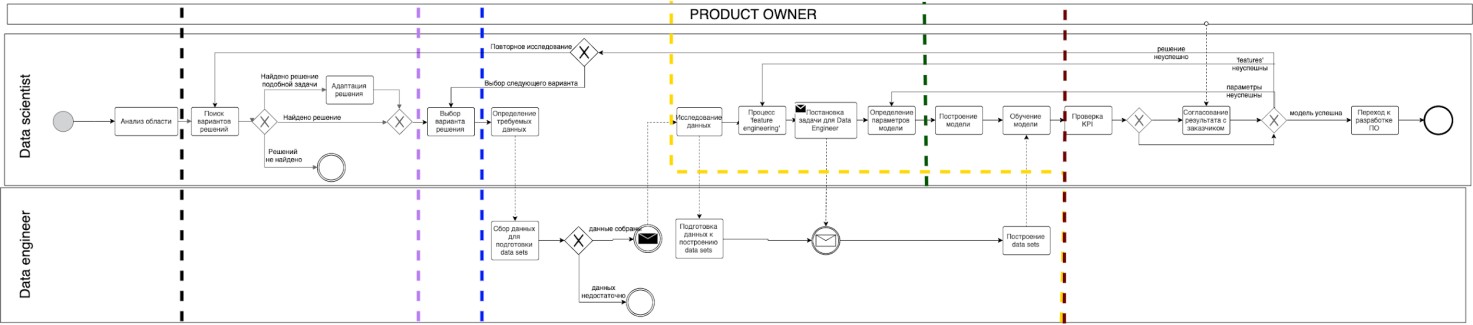


Рисунок 2.5 - Модель бизнес-процессов этапа R&D на предприятии «ЖК Небо»

Представленная модель бизнес-процессов отражает развертывание процесса разработки продукта во времени и распределение работ между исполнителями, а также связи между ними.

Рисунок 2.5 представляет собой следующий уровень декомпозиции блок- схем алгоритма R&D, выполненных на рисунках 2.2 и 2.3. Поскольку более

подробная детализация усложняет схему исследовательской деятельности, выделение четких этапов (как это было сделано в рисунке 2.2), становится весьма затруднительным, так как реальный, физический процесс создания программного продукта является нелинейным, ситуативным, а его подпроцессы могут быть параллельны или пересекаться.

Как было ранее показано на рисунке 2.3, R&D-деятельности предшествует анализ предметной области. Результатами этого анализа должны быть:

* определение цели продукта;
* определение бизнес-требований;
* определение KPI для каждого бизнес-требования;
* определение возможных способов удовлетворения бизнес-требований, а именно: определение проектной командой стратегии решения каждого бизнес-требования - с помощью построения современных технологий или алгоритмических методов.

Предварительная ориентировочная деятельность в проектах с современными технологиями имеет значение большее, чем в любых других видах проектов. На этой стадии происходит формулирование критериев, которым в дальнейшем будет соответствовать готовый продукт.

Реализация любого проекта всегда начинается с формулирования потребности заказчика/рынка в нем. Product Owner должен иметь представление о том, как этот продукт или его новая функция должны работать в конечном итоге. Например, внедрение некоторого сервиса предотвратит отток посетителей или простимулирует продажи других продуктов.

Здесь необходимо изложить цели продукта, на основании которых формируются бизнес-требования, т.е. набор той функциональности, которая определяет назначение ПО.

На этом этапе все участники проекта предлагают идеи, которые можно использовать при решении сформулированной проблемы, происходит создание некоторых первоначальных набросков возможного решения. Этим этапом обычно руководит специалист Data Science, но важно, чтобы принимали участие абсолютно все участники работы на проекте, включая менеджмент, так как возможные решения и продуктивные идеи могут лежат в совершенно различных областях знаний.

В результате этого этапа обычно становится ясно, насколько глубоким будет процесс исследования данных.

Затем следует определение количественных показателей, которыми будет определяться, достигнуто ли бизнес-требование. Этот этап заключается в совместном определении объема и ключевых показателей эффективности проекта (KPI). Затем эти KPI должны уточнены и переведены в измеримые. Возможно, удастся получить очень четкие показатели, такие как

*«прогнозирование ожидаемого CTR объявления с приближением не менее X% в по меньшей мере Y% случаев, для любого объявления, показанного не менее недели, и для любого клиента с более чем двумя месяцами временных данных»* . Однако в некоторых случаях необходимо использовать качественные показатели, например, *«время, необходимое для изучения темы с использованием сгенерированных расширенных запросов, будет сокращено и/или улучшится качество результатов по сравнению с исходными запросами»*.

Это особенно верно, когда модель предназначена для оказания помощи человеку в какой-то сложной его деятельности.

Ключевые показатели могут быть уточнены в связи с имеющимися временными рамками и ресурсами. Для их уточнения нужен дальнейший сбор данных.

Обычно эти метрики (KPI) определяются проектной командой совместно с Product Owner в точных количественных показателях. Но в зависимости от ресурсов и временных ограничений они могут быть подвергнуты пересмотру после консультаций с заказчиком. Поскольку любая обратная связь означает дополнительный расход времени, то проектная команда может попытаться найти дополнительные жесткие метрики, на которые можно опереться для продолжения работы над продуктом, и избежать дополнительных консультаций с заказчиком.

Анализ решения (обозначен сиреневым цветом на рисунках 2.2, 2.3 и 2.5).

На данном этапе перед проектной командой стоят задачи изучения литературы и существующих готовых решений.

Исследователь анализирует как академическую литературу, так и существующие программные решения и инструменты.

Глубина погружения в литературу зависит от особенностей проекта: является ли он фундаментальным, определяющим дальнейшую стратегию компании или будущий программный продукт предназначен для решения разовой проблемы. Также необходимо ответить на вопросы: планируется ли публикация результатов исследования по этому вопросу в академических изданиях; планирует ли исследователь стать экспертом команды по этой теме и т. п.

Например, проект предназначен для того, *чтобы помочь отделу продаж лучше прогнозировать отток клиентов и убытки от оттока. Для решения поставленной задачи требуется хорошее знание теории случайных процессов, на которой строятся многие общие решения подобных проблем.*

Необходимая глубина понимания зависит от технических аспектов проблемы. Некоторые из аспектов могут быть предвидены заранее, а некоторые обнаружены только позже. *Например, если разрабатываемый продукт должен быть интегрирован в готовую программную среду, написанную на языках Java и Scala, то приходится использовать те технологии, которые применяются в готовом продукте. И если разработчики не владеют этими языками, то их придется изучить.*

При изучении литературы специалист Data Science должен подготовить для обсуждения в команде не только выбранные варианты решения проблемы, но сделать краткий обзор всей области и всех рассмотренных решений, объясняя преимущества и недостатки каждого решения и обосновать свой выбор.

После того, как найден подходящий математический аппарат, найденные направления решений необходимо оценить с точки зрения способа реализации и сложности этого способа в производстве.

Цели продукта и бизнес-требования к нему, а также структура и характеристики предлагаемых вариантов решений служат обоснованием для выбора способа хранения и обработки данных, способность к масштабированию по горизонтали и вертикали (scalability) и приблизительную оценку стоимости проекта.

Это важная проверка, которую необходимо выполнить на этом этапе, поскольку обработка данных и разработка программного обеспечения могут начинаться параллельно с разработкой модели. Кроме того, предлагаемое решение может оказаться неадекватным или слишком дорогостоящим с технической точки зрения, и в этом случае это должно быть выявлено и устранено как можно скорее. Когда технические вопросы рассматриваются до начала разработки модели, знания, полученные на этапе исследования, могут затем использоваться для предложения альтернативного решения, которое могло бы лучше соответствовать техническим ограничениям. Это еще одна причина, по которой этап исследования должен закончиться выработкой спектра решений, а не ограничиваться одним. Подготовка данных и выбор решения (синий и желтый цвет на рисунках 2.2, 2.3, 2.5);

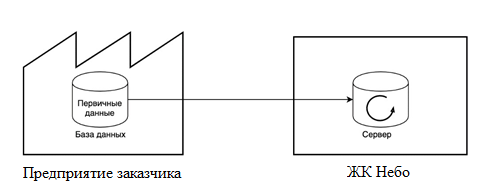
На этом этапе проектная команда определяет набор данных, которые потребуются в дальнейшем для обучения нейронных сетей; сбор первичных данных, обработка их (очистка от мусорной информации), преобразование в требуемый формат. Хранятся подготовленные данные вместе с запросами в базах данных компании.

Рисунок 2.6 – Схема взаимодействия баз данных

Процесс подготовки данных часто сопровождается выгрузкой больших массивов данных (Big Data) из баз данных заказчика на сервер компании (если позволяют условия конфиденциальности). В дальнейшем происходит обработка Big Data в формат, удобный для быстрых запросов и сложных вычислений. Эти действия необходимы для начала процесса исследования данных и иногда занимают больше времени, чем ожидалось.

## Выводы

Была произведена качественная оценка разработанной модели с применением таких метод экспертных оценок и сравнительного анализа уровня автоматизации бизнес-процессов предприятия при использовании моделей управления «As Is» и «To Be». Было показано, что модель «To Be» позволяет увеличить степень автоматизации создания задач на 85,7%, а создания документа на 50%.

SWOT-анализ модели «To Be» показал сильные и слабые стороны новой модели, а также возможные риски и потенциал дальнейшего развития.

В целом, **новизна** исследования и его **теоретическую значимость** заключается в том, что предложенная модель системы управления проектами направлена в первую очередь на управление этапом R&D, который наиболее трудно поддается управлению в силу специфики научно- исследовательской деятельности; для создания адекватной модели управления был разработан алгоритм процесса проектной деятельности на этапе R&D; предложен способ представления жизненного цикла проекта и фаз проектной деятельности в виде дерева решений, явным преимуществом которого перед иными моделями жизненных есть визуальное отражение исходных точек начала следующего цикла.

Рекомендации по практическому применению разработанной модели и дальнейшее развитие темы диссертационной работы заключаются в следующем:

1. воплощение модели «To Be» в виде плагина R&D и внедрении его в систему управления проектами компании «ЖК Небо»;
2. управление временем. На данный момент предложенная модель нигде не учитывает такой важный фактор, как время. Нигде в модели не присутствует контроль времени, то есть не установлен параметр и не разработана функция,
3. регулирующая количество времени, необходимое для:

* изучения предметной области проекта,
* поиска адекватных и доступных для реализации решений формулируемых задач;
* повторения цикла исследования (если не повезло найти сразу нужное решение).

Также не определена протяженность временного интервала обучения модели.

Тем не менее управление временем в компании выполняется - каждым сотрудником, проектным менеджером и Product Owner-ом.

Следовательно, можно попробовать разработать некоторые плавающие временные коэффициенты, управляющие процессом создания проекта. В процессе накопления опыта выполнения проектов компанией будет происходить настройка временного коэффициента и совершенствоваться система управления проектами.

1. в процессе расширения «библиотеки готовых решений» по профилю компании возможна ее интеграция с внешними подобными библиотеками;
2. создание рекомендательной подсистемы, наблюдающей за работой исследователя на этапе «анализ решений». Изучая запросы исследователя в конкретной области, такая подсистема сможет помогать сотруднику в поиске необходимого решения, осуществляя поиск источников.
   * повышение комфорта и скорости взаимодействия между сотрудниками.

Достоверность и обоснованность результатов обеспечивается выбранной теоретической и методологической основой исследования; аргументированностью выводов, подтвержденных оценками экспертов - директоров компании «ЖК Небо»; соответствием методов исследования цели, предмету и задачам.

Изучение методов управления проектами, принятых в компании

«ЖК Небо», а также алгоритма R&D-деятельности позволили сделать следующие наблюдения и выводы:

Компания сталкивается с такими проблемами, как -

* неполная документация.

Эта проблема возникает в связи с тем, в связи с тем, что создание документации традиционно является кругом обязанностей менеджерского состава и бизнес-аналитиков. В условиях исследовательской деятельности документация является технической и проектный менеджер не в состоянии ее корректно заполнить без помощи технических специалистов. Поэтому разработчики создают ее самостоятельно. Но поскольку подобного рода обязанности не являются традиционными для них, необходимая документация формируется некачественно, и иной раз даже вообще не создается, так как исполнители не хотят тратить на нее время и полагаются на свою память. Таким образом, ценные мысли, идеи, творческие решения могут быть утеряны. По причине несохраненной документации в некоторых случаях проектной команде приходится тратить дополнительное время на повторное обсуждение задач, способов решения, а также на повторные исследования и т. д.

* неунифицированный формат.

По причине, описанной выше, документация создается в разных форматах. Это затрудняет процесс ее совместного использования и т. д.

* слабая визуализация.

Дерево решений не визуализируется и нигде не сохраняется. Это может влиять на общее представление о стадиях проекта, а также затрудняет проектному менеджеру задачу отчетности заказчикам о проделанной работе;

* неполный учет времени.

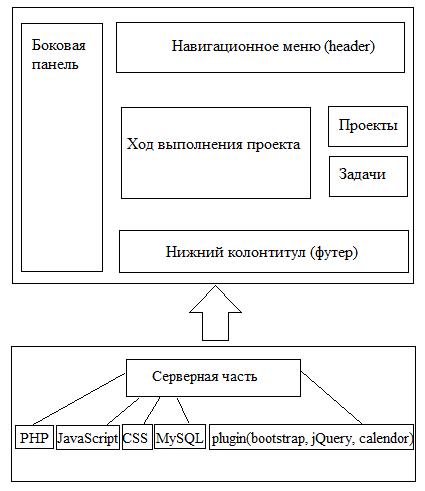
Задачи повторяются от ветки к ветке, некоторые задачи являются незначительными (с точки зрения времени), проектная команда к компании небольшая и каждый знает свои обязанности, по этим причинам некоторые задачи могут не создаваться в инструменте управления проектами. Это затрудняет учет времени, так как в Jira будет храниться неполная информация о задачах. Также это может породить проблему «потери» некоторых задач.

Отсутствие критерия времени. Модель не подразумевает учет времени или ограничение времени каждого цикла.

1. **ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ В КОМПАНИИ «ЖК Небо»**

**3.1 Архитектура приложение (сайт)**

Для рассматриваемого приложения используем клиент-серверную архитектуру. Это является вполне естественным выбором архитектуры для сайта. На рис. 3.1 представлена архитектура в виде диаграммы развертывания. Заметим, что все необходимые данные хранятся на серверной часть, а на клиентской не требуется хранить никаких данных.



***Рис. 3.1. Архитектура системы. Диаграмма развертывания***

Увеличение уровня автоматизации процессов приводит также к росту некоторых качественных показателей управления, среди которых можно указать улучшение визуального представления циклов решения, потенциальное сохранение большего количества данных о создаваемых программных решениях и многие другие.

## 3.2. Описание средств разработки приложения

*Язык программирования*. В данном случае целесообразно выбрать язык PHP, исходя из следующих его преимуществ:

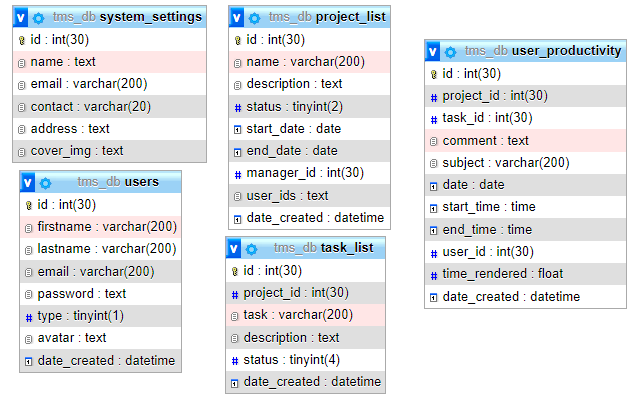
* простота программирования;
* эффективность – достаточно высокая скорость обработки скриптов, по некоторым оценкам, в большинстве случаев скрипты на PHP выполняются быстрее, чем на другие языки;
* расширяемость – при хорошем структурировании кода скрипты легко дорабатываются;
* возможность работы на большинстве популярных серверов.

## 

## 3.3. Описание СУБД

*СУБД*. В данном проекте целесообразно выбрать *MySQL*:

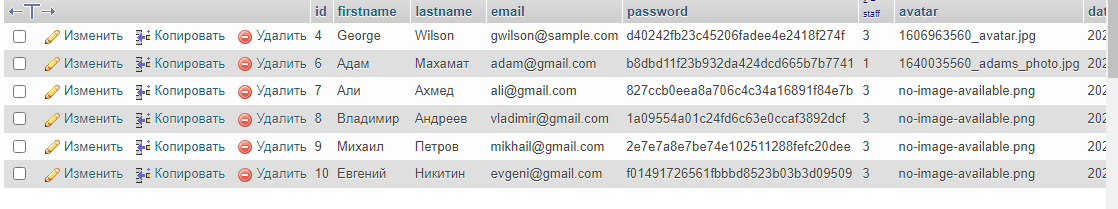
* эта СУБД характеризуется высокой производительностью, что очень важно в нашем случае, поскольку по мере использования сервиса(приложения) будут накапливаться тысячи записей о проектах и миллионы записей о задачах проектов;
* простота в использовании, причём наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных;
* безопасность: система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию; MySQL снабжена развитой системой защиты от несанкционированного доступа;
* кроссплатформенность: MySQL отлично работает как под управлением разных версий UNIX-систем, так и под управлением Windows, что позволяет нам не выдвигать излишних требований к программному обеспечению серверной ЭВМ.



***Рис. 3.2. Связь между таблицами***

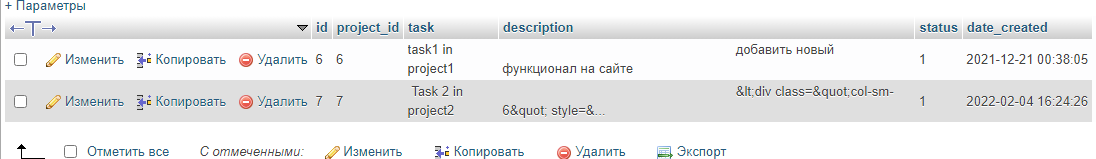
Структура нашей базы, в нем хранится информации по всему проекту.

Состоит из несколько таблиц рассмотрим пример:

**users** – это таблица содержит информации всех пользователей

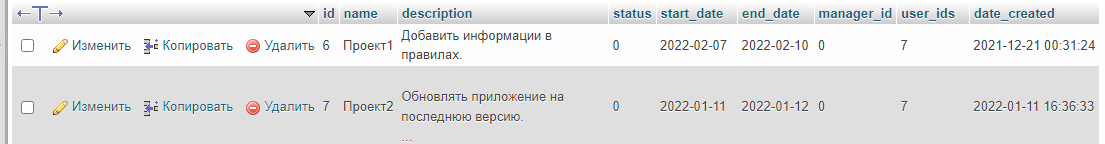
***Рис. 3.3. Таблица пользователей***

**Task\_list** – Это таблица хранит информации о задачах



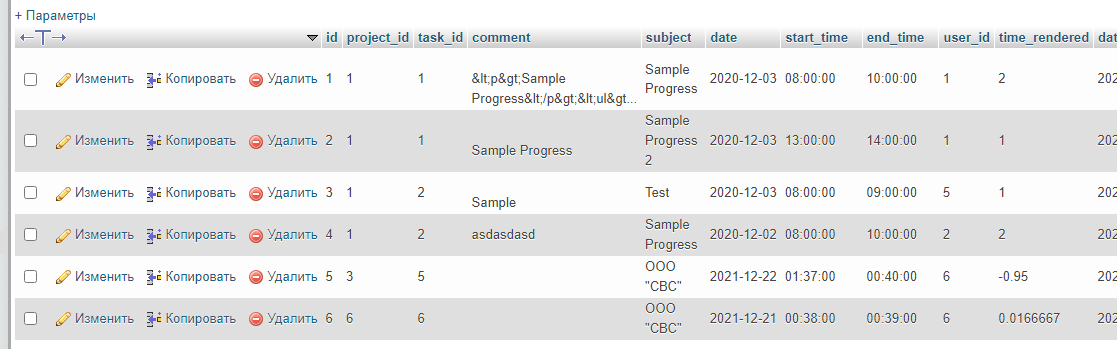
***Рис. 3.3. Таблица задачей***

**Project\_list** – Это таблица хранит информации о проектах

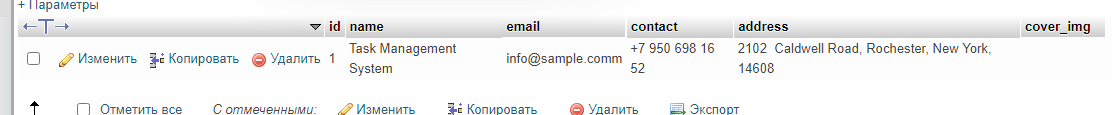


***Рис. 3.4. Таблица проектов***

**User\_productivity** – Это таблица хранит информации о комментариях, статусах, время начала и время окончания задачи и проекта.

 ***Рис. 3.5. Таблица времени.***

**system\_settings** – Это таблица хранит информации о настройке сайта.



***Рис. 3.6. Таблица хранить информации о настройке сайта***

## 3.4. Проектирование интерфейса сайта

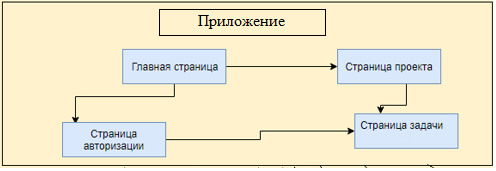
Выдвинем следующие требования к интерфейсу рассматриваемого сервиса:

* возможность перехода с любой страницы сайта на любую другую максимум за 2 щелчка по ссылкам;
* минимизация использования полос прокрутки;
* соответствие расположения элементов и системы навигации ожиданиям посетителей (например, расположение меню столбиком в правой части страницы – явное нарушение такого требования);
* соответствие графических образов и терминологии общепринятым в рамках программ и сервисов управления проектами;
* отсутствие перегруженности элементами управления;
* учёт специфики проектов по разработке программных продуктов.

В нашем случае целесообразно выделить следующие веб-страницы:

* главная страница – содержит краткую информацию о сервисе, обеспечивает доступ к регистрации и входу под учётной записью;
* страница авторизации – содержит элементы для заполнения личных данных, отправки запроса на авторизацию;
* страница «Проекты» - содержит информацию о проектах, связанных с текущим пользователем;
* страница «Задачи проекта» - в случае входа в роли руководителя содержит информацию обо всех задачах проекта, в роли исполнителя – только о задачах текущего исполнителя;

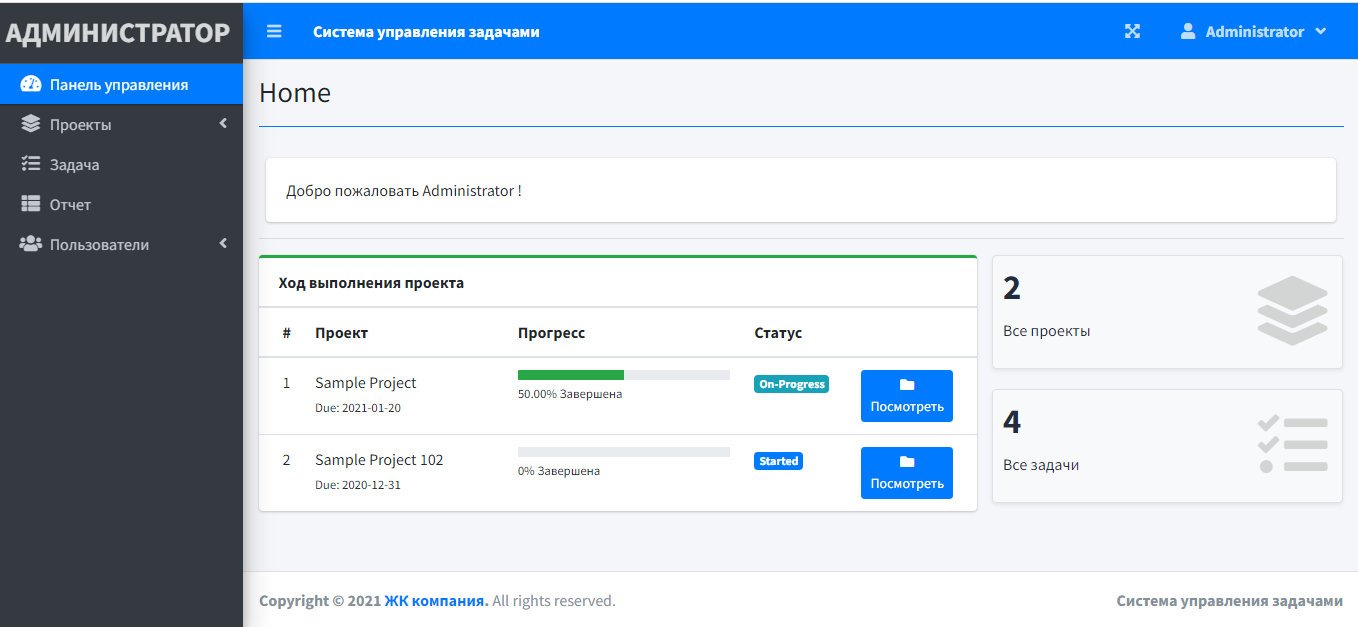
На рис. 3.2 представлен граф сайта. Все переходы двусторонние – если можно перейти за единственный щелчок по ссылке с одной страницы на вторую, то можно и со второй на первую страницу. Легко заметить, что кратчайший путь между любой парой вершин не превышает 2 ребра.



***Рис. 3.5. Граф сайта***

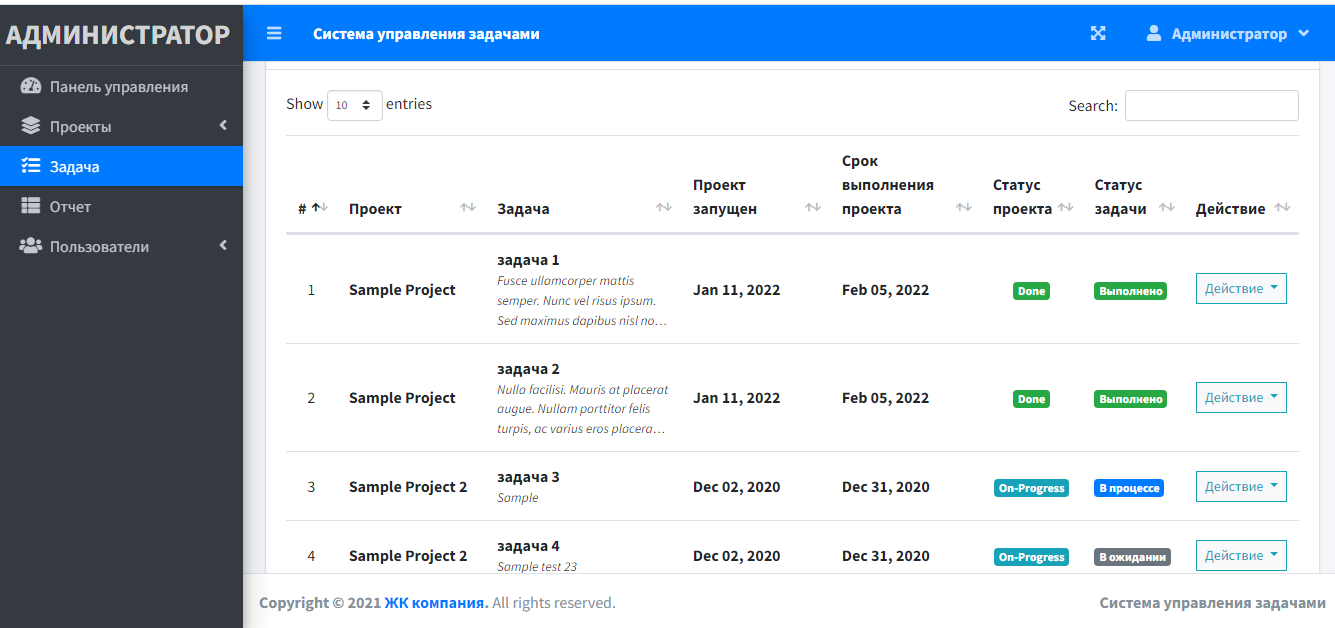
Рассмотрим структуру и содержимое некоторых страниц. Страница «Проекты» содержит (рис. 3.6):

* меню для перехода к другим страницам (Главная и задачи);
* таблицу проектов и вложений для текущего пользователя-руководителя (название проекта реализуется в виде гиперссылки – при переходе по нему мы получим доступ к задачам);
* элемент для сортировки таблицы (полезно предусмотреть сортировку по названию, статусу, срокам);
* панель добавления проекта.



***Рис. 3.6. Схема страницы проектов***

На панели добавления проекта размещаются элементы задания характеристик проекта, элементы для добавления файлов к проекту.

Следует минимизировать ввод с клавиатуры, заменив его элементами выбора из конечного множества там, где это возможно. Например, множества распространённых методологий программных проектов (каскадная, RAD и т.д.) и классов программных продуктов (веб-сайты, настольные приложения и т.д.) известны. Мы можем предоставить выбор из раскрывающегося списка, предусмотрев вариант «Прочее» и сделав доступным текстовое поле ввода при выборе этого особого варианта (рис. 3.7).

***Рис. 3.7. Пример предоставления задачи***

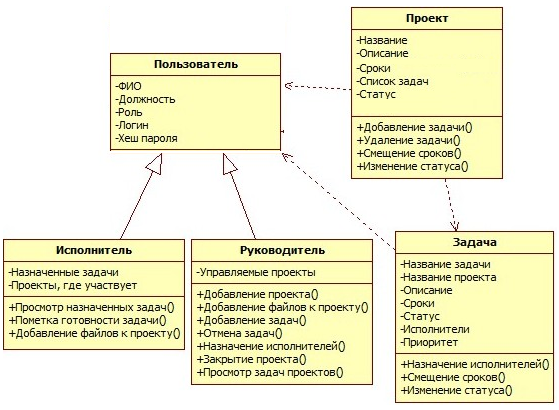
Страница задач проекта во многом похожа – она также должна содержать таблицу данных о задачах и элементы добавления, редактирования, удаления задач. Однако следует учесть, что страница проектов доступна только руководителя и администратор, а страница задач – также и исполнителем: если совершён вход под исполнителем, пользователь не должен видеть чужие задачи, а также не должен иметь возможность сдвинуть сроки и выполнить прочие действия, превышающие его полномочия.

## 3.5. Проектные модели

### *3.5.1. Диаграмма классов*

Для написания PHP-кода в скриптах выбран объектно-ориентированный подход, поскольку он позволяет эффективно структурировать большой код, сохраняя контроль над ним. Программист может абстрагироваться от тех деталей реализации и сущностей, которые в данный момент несущественны, сосредоточив свое внимание на том алгоритме, который сейчас реализуется.

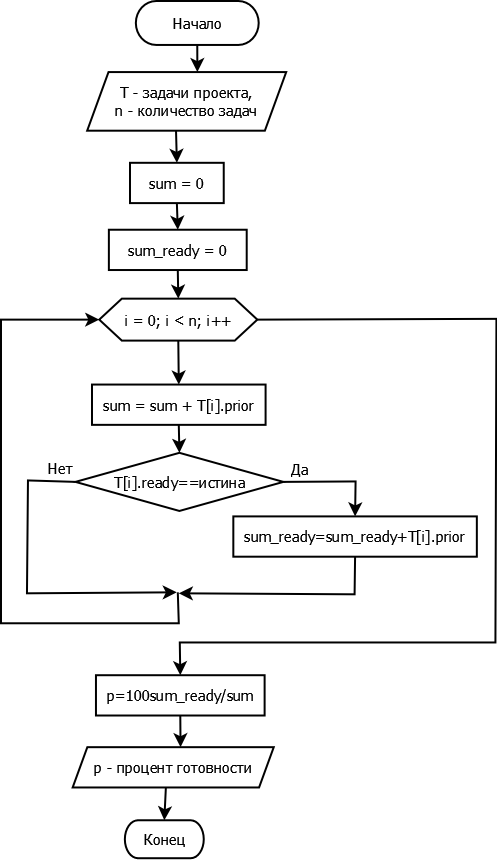
На рис. 3.8 представлена диаграмма классов. Указаны классы и их отношения, а также важнейшие поля и методы. Стрелка с маленьким наконечником от класса *X* к классу *Y* означает отношение зависимости – в коде класса *X* используется имя класса *Y*, стрелка с большим наконечником идет от производного класса к базовому. Поскольку реализация наследования в PHP не проблематична, составленная диаграмма классов адекватна средствам разработки веб-сервиса.



***Рис. 3.8. Диаграмма классов в скриптах сайта***

### *3.5.2. Алгоритмическое конструирование*

Одной из интересных проблем, решаемых при разработке сайта по постановке и контролю выполнения задач предприятия, является проблема оценки степени готовности проекта. Простейший способ, как это сделать, - оценить соотношение числа задач, помеченных как выполненные, и общее число задач проекта. Однако он не учитывает такой важной особенности, как приоритеты задач. Если наиболее приоритетные (важные) задачи выполнены, а наименее приоритетные – нет, нельзя оценивать готовность так же, как если выполнены только наименее приоритетные задачи. На рис. 3.8 представлен алгоритм, частично решающий проблему разницы в приоритетах.



***Рис. 3.9. Блок-схема расчёта процента готовности проекта***

Уровень готовности определяется отношением суммы приоритетов выполненных задач к сумме приоритетов всех задач. Предполагается, что приоритет задачи – некоторое натуральное число. Если бы приоритеты задач могли отличаться на порядок и гипотетически в проекте могли бы оказаться шесть задач приоритета 1 и одна задача приоритета 10, при выполнении только самой важной задачи алгоритм выдал бы абсурдный результат – 63,5% готовности. И это при том, что выполнена одна задача из семи. Однако мы планируем ограничиться 2-3 уровнями важности задач, вследствие чего алгоритм оказывается приемлемым. Обратим также внимание, что если приоритеты задач равны, процент готовности будет определяться соотношением числа выполненных задач и общего числа задач.

## 3.6. Руководство по установке

Laravel – это высокопроизводительный компонентный PHP Фреймворк. Подходит для разработки всех типов веб-приложений: порталы, форумы, CMS, магазины и т д. Laravel отлично расширяем и производителен.

**Установка Laravel**

Установить **Laravel** можно 2 способами: используя [composer](https://getcomposer.org/) (предпочтительно) или скачав архив. Также можно установить basic или advanced шаблон, соответственно для простого и более продвинутого приложения.

**Запуск приложения**

Установленный шаблон уже содержит простой пример и несколько страничек. Работает простая авторизация пользователя администратора. Для начала нужно запустить команду php artisan serve .

**Структура приложения**

Laravel -приложения организованы по MVC шаблону проектирования. Модели – это данные, бизнес-логика и бизес-правила. Представления отвечают за отображение информации. Контроллеры принимают входные данные от пользователя и преобразуют их в понятный для моделей формат и команды, также отвечают за отображение нужного представления.

**Входные скрипты**

Это первое звено в процессе начальной загрузки приложения. Входные скрипты для веб-приложений - index.php, для консольных приложений – init.php

Приложения — это объекты, которые управляют всей структурой и жизненным циклом прикладной системы laravel. Во входном скрипте создаётся один объект приложения, и он доступен глобально через \ laravel:$app. Существуют 2 вида приложений - веб-приложения и консольные приложения. Первые занимаются обработкой веб-запросов, вторые - консольных команд.

Входной скрипт при создании приложения, загружает конфигурацию и применяет её к приложению.

* [Компоненты приложения](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-application-components.md).
* [Контроллеры](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-controllers.md).
* [Модели](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-models.md).
* [Представления](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-views.md).
* [Модули](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-modules.md).
* [Фильтры](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-filters.md).
* [Виджеты](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-widgets.md).
* [Ресурсы](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-assets.md).
* [Расширения](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/structure-extensions.md).

**Работа с базами данных**

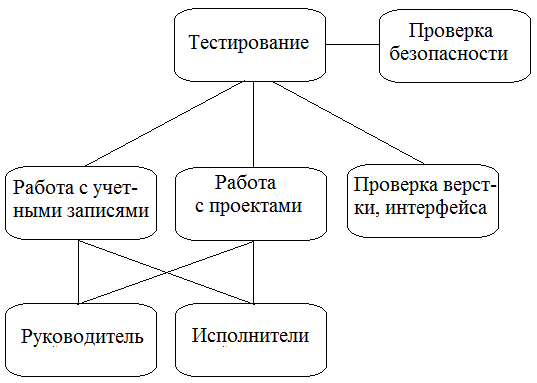
* [Объекты доступа к данным (DAO)](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/db-dao.md) - Соединение с базой данных, простые запросы, транзакции и работа со схемой.
* [Миграции](https://github.com/yiisoft/yii2/blob/master/docs/guide-ru/db-migrations.md) - контроль версий схемы данных при работе в команде.
* [Sphinx](https://github.com/yiisoft/yii2-sphinx/blob/master/docs/guide-ru/README.md).
* [Redis](https://github.com/yiisoft/yii2-redis/blob/master/docs/guide-ru/README.md).
* [MongoDB](https://github.com/yiisoft/yii2-mongodb/blob/master/docs/guide-ru/README.md).

# ТЕСТИРОВАНИЕ

## 4.1. План тестирования

Тестирование сайта по постановке и контролю задач предприятии целесообразно проводить по следующим направлениям (рис. 3.1):

* тестирование функций ведения учётных записей (авторизации, ведения личных данных);
* тестирование работы с проектами (функций, связанных с управлением проектами и задачами проектов со стороны руководителя и с участием в задачах проекта со стороны исполнителя);
* тестирование интерфейса и вёрстки;
* тестирование обеспечения безопасности – поскольку данные о проектах и о пользователях не должны оказаться в посторонних руках, требует особо ответственного отношения к себе.



***Рис. 4.1. План тестирования сайта***

## 

## 4.2. Подробное описание тестирования

### 4.2.1. Признаки ошибок в реализации веб-приложение

Некоторые признаки ошибок в функциях работы с учётными записями:

* возможность доступа с учётной записи к данным, к которым не должно быть доступа (например, доступ к задачам, не назначенным данному исполнителю);
* отличие в данных пользователя после нового входа в сервис по сравнению с данными, которые были актуальны при предыдущем входе.

Некоторые признаки ошибок в функциях работы с проектами:

* допущение противоречий в данных проекта (например, срок выполнения задачи проекта позже срока выполнения всего проекта);
* сбои при добавлении файлов к проекту или попытке просмотра;
* отсутствие предупреждений при попытках загрузить сверхбольшой или пустой файл;

Некоторые признаки ошибок вёрстки, интерфейса:

* интерфейсные элементы имеют стиль или расположение, не совпадающие с задуманными, хотя бы на некоторых из браузеров, предусмотренных техническим заданием (ситуация, известная как «слетает вёрстка», - частный случай проблемы);
* система не оповещает пользователя о трудно обратимых операциях (в частности, операциях удаления данных – проекта, задач, учётных записей пользователей).

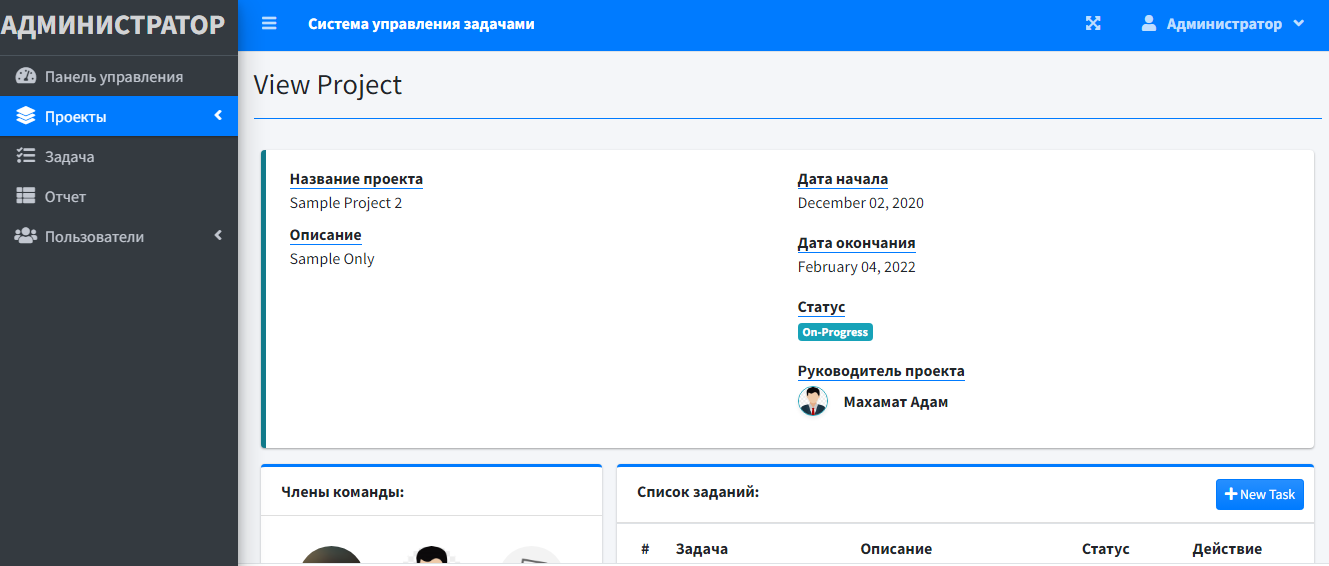
Некоторые признаки ошибок в обеспечении безопасности:

* нет предупреждений при вводе заведомо неэффективного пароля (короткого или распространённого пароля, а также пароля, основанного на имени, дате рождения и т.п.);
* нет предупреждений руководителя при загрузке исполнителями скриптов и исполняемых приложений в качестве файлов проекта;
* неограниченная возможность повторных попыток ввода пароля при неуспешном входе.

### *4.2.2. Описание тестов*

Рассмотрим некоторые тесты из тех, которые планируется осуществить для проверки работоспособности веб-приложения.

*Тестирование функций отслеживания сроков*. В разрабатываемом веб- приложение важна корректная реализация оповещения о просрочке, а также о приближении срока завершения проекта или отдельных задач. Оповещение должно выполняться за число дней, заданное руководителем проекта. На рис. 4.2 представлена схематически серия тестов для проверки, корректно ли отслеживается просрочка задач и приближение дедлайна. В белую рамку обведены задачи, которые должны отмечаться в качестве задач с приближающимся дедлайном(deadline), в серую – которые требуется отмечать как просроченные.



***Рис. 4.2. Тест отслеживания задач с приближающимся дедлайном***

Охватываются следующие ситуации для отслеживания задач, которые просрочены или скоро могут быть просрочены:

* наличие задач, для которых нет проблем со сроками;
* наличие задач, для которых приближается дедлайн;
* наличие задач, у которых есть проблемы со сроками, но только если не обращать внимание на пометку готовности;
* добавление задач, у которых близок дедлайн (например, руководитель желает, чтобы при входе в сервис его оповещали, если остаётся до 4 дней до срока завершения задачи, и он добавил задачу с трёхдневным сроком).

Процесс, схематически изображённый на рис. 3.2, можно дополнить, попытавшись добавить задачу, у которой срок завершения в прошлом, чтобы проверить, как отреагирует веб-приложения.

*Тестирование функции добавления задачи*. При тестировании функции добавления задачи целесообразно охватить следующие ситуации:

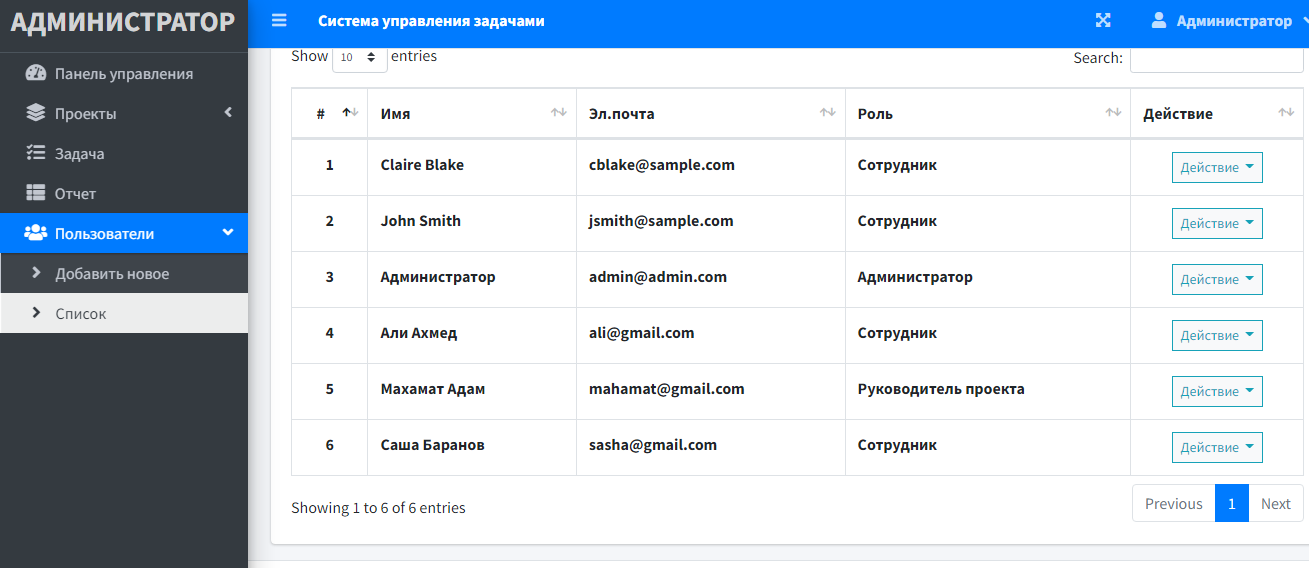
* наличие незаполненных полей (например, не введено название или описание задачи);
* наличие в проекте задачи с таким же названием или описанием;
* противоречия в сроках (дата планируемого завершения – из прошлого, дата завершения после даты планируемого завершения проекта).

Целесообразно выполнить тестирование следующим образом:

* сначала попытаться добавить задачу с различными ошибками в данных, указанными выше;
* далее попытаться добавить задачу, исправив ошибки в заполняемых полях.

Сделали несколько тестов с разными пользователями и разные сроки, так же добавили их статус.

При выполнении задачи можно увидеть точную дату и время.



***Рис. 4.3. Несколько тестов с разными пользователями***

# ЗАКЛЮЧЕНИЕ

В результате выпускной квалификационной работыбыло достигнуто следующее.

* рассмотрены преимущества автоматизации управления программными проектами и их задачами;
* выполнен анализ существующих веб-приложение управления проектами, определены их достоинства и недостатки;
* на основе результатов анализа существующих веб- приложение по постановке и контролю задач предприятии выдвинуты требования к веб- приложению, позволяющему осуществлять управление проектами по постановке и контролю задач предприятии;
* спроектирован и реализован веб-приложение по постановке и контролю задач предприятии: в частности, выбрана архитектура и средства разработки, спроектирован пользовательских интерфейс, рассмотрено использование объектно-ориентированного подхода к разработке веб-приложения;
* проведено тестирование веб- приложения на предмет соответствия функциональным требованиям.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

*Научная и методическая литература*

1. Александрова, Т.В. Повышение эффективности проектного управления в организации на основе гибкой методологии Agile // Экономика и бизнес: теория и практика. 2019. №9.
2. Баканов, А. Б., Дрождин, В. В., Зинченко, Р. Е., Кузнецов, Р. Н. - Методы адаптации и поколения развития программного обеспечения // Известия ПГПУ им. В. Г. Белинского. 2009. № 13 (17). С. 66-69
3. Бэгьюли, Ф. Управление проектом: пер. с англ. / Ф. Бэгьюли – М.: Гранд ФАИР-ПРЕСС, 2002. – 202 с.
4. Вареникова, О.В., Бобылева, А.А., Голубев, Д.В. Управление проектами в электроэнергетике/ Colloquium-journal. – 2019. - №13(37). – С.43-56
5. Вертакова, Ю. В. Управленческие решения: разработка и выбор: учеб. пособие / Ю. В. Вертакова, И. А. Козьева, Э. Н. Кузьбожев; под общ. Ред. Проф. Э. Н. Кузьбожева. — М.: КНОРУС, 2005. — 352 с.
6. Голубев, С.А. Управление венчурными проектами.-СПб, СпбГТУ, 2009
7. Дерябкин, В.П. Проектирование автоматизированных систем обработки информации и управления: курс лекций / В.П. Дерябкин. - Самара: СГАУ, 2001. – 120 с.
8. Заренков, В. А. Управление проектами: учебное пособие. - 2 изд. - СПб: АСВ, 2010. - 312 с.
9. Ивасенко, А. Г. Управление проектами / А. Г. Ивасенко, Я. И. Никонова, М. В. Каркавин – Ростов-на-Дону : Феникс, 2009. – 327 с.
10. Кораблев, И. Г. Оценка уровня автоматизации бизнес-процессов предприятия // Вестник Череповецкого государственного университета. 2016. №1 (70). URL: https://cyberleninka.ru/article/n/otsenka-urovnya- avtomatizatsii-biznes-protsessov-predpriyatiya (дата обращения: 16.12.2019)
11. Кузнецова, Н. В. Анализ подготовки и принятия управленческих решений в практике: к вопросу выбора оптимального метода // Молодой ученый. — 2016. — №27. — С. 425-433.
12. Мазур, И. И. Управление инвестиционно-строительными проектами: международный подход / И. И. Мазур. М. : Омега-Л, 2011. 736 с.
13. Макашов, П. Л., Романенко, Н. А. Сервис-ориентированный подход к управлению ИТ проектами на примере использования программного продукта "Jira" // Современные информационные технологии и ИТ- образование. - 2015. - №11. – С.12-16.
14. Николаенко, В. С. Разработка принципов управления ИТ-проектом // Вестн. Том. гос. ун-та. - 2015. -№390. –С.56-60.
15. Озерова, Т. Системная триада как основа управления проектами на предприятиях общественного питания / Т. Озерова // РИСК : ресурсы, информация, снабжение, конкуренция. 2011. № 2, ч. 1. С. 35–38.
16. Скотт, А. Гибкие технологии: экстремальное программирование и унифицированный процесс разработки. Wiley (ISBN 0-471-20282-7), 2002 - Scott W. Ambler, Agile Modeling: Effective Practices for Extreme Programming, 2002; перевод и издание на русском языке: ЗАО Издательский дом “Питер” (ISBN 5-94723-545-5), 2005
17. Управление проектами / В. Д. Шапиро и др. СПб.: ДваТрИ, 1996. 610 с.
18. Фунтов, В. Н. Основы управления проектами в компании. / В. Н. Фунтов – СПб.: Питер, 2011. – 393 с.
19. Чуланова, О. Л. Технология управления проектами и проектными командами на основе методологии гибкого управления проектами Agile // Вестник евразийской науки. - 2018. - №1.-С 31-35.
20. Шайхулова, А.Ф. Автоматизация и управление инновационными проектами технического перевооружения авиадвигателестроительного производства на основе каскадного метода оптимизации: Автореф...дис. кан. тех. наук. – Уфа, 2018. –20 с.

*Электронные ресурсы*

1. Agile-манифест разработки программного обеспечения

//<https://agilemanifesto.org/iso/ru/manifesto.html>

1. Понятие «проект» // Pandia URL: <http://www.pandia.ru/365896/>
2. Confluence //<https://startpack.ru/application/confluence>
3. PMBOK® Guide – Sixth Edition // Project Management Institute: https://[www.pmi.org/pmbok-guide-standards/foundational/pmbok](http://www.pmi.org/pmbok-guide-standards/foundational/pmbok)
4. Жизненный цикл проектной задачи // Projectimo: <http://projectimo.ru/upravlenie-proektami/zhiznennyj-cikl-proekta.html>
5. Орлик, С. Основы программной инженерии по SWEBOK

//<https://ligurio.github.io/swebok-ru/>

1. Advantages and Disadvantages of Trello // SoftwareDeveloperIndia: https://[www.software-developer-india.com/advantages-and-disadvantages-of-trello.](http://www.software-developer-india.com/advantages-and-disadvantages-of-trello)
2. What is Asana? // CompareCamp: <http://comparecamp.com/asana-reviews-> pricing-benefits-and-features-analysis/
3. 6 Benefits of Using JIRA // Dumbfunded: https://[www.dumbfunded.co.uk](http://www.dumbfunded.co.uk/)/ guides/6-benefits-of-using-jira/
4. Trello vs Asana: The Best Project Management App in 2017? // Process.st: <https://www.process.st/trello-vs-asana/>
5. Product Guides & Tutorials // Atlassian: <https://www.atlassian.com/software/jira/>guides/getting-started/overview
6. Project management. A Guide to the Project Management Body of Knowledge: PMBOK guide. 3rd ed. 2004. 506 p. [Electronic resource]: <http://rutracker.org/forum/viewtopic.php?t=2831128>

*Литература на иностранном языке*

1. A Spiral Model of Software Development and Enhancement. Barry W. Boehm, TRW Defense System Group, 1988
2. Frohm J., Stahre Jo, Winroth M. P. Levels of Automation in Manufacturing // International Journal of Ergonomics and Human Factors, Vol. 30 Issue 3, 2008.
3. Managing the Development of Large Software Systems: concepts and techniques. Rouce Winston, 1970, ICSE '87 Proceedings of the 9th international conferense on Software Engineering
4. Oberlender Garold D. Project management for engineering and construction / 2nd ed. New York: McGrawHill, 2000. 368 p.
5. "Software Engineering — Guide to the software engineering body of knowledge". ISO/IEC TR 19759:2015
6. Walker Anthony. Project management in construction / 4th ed. Oxford: Blackwell Science, 2002. 289 p.

Приложение А

## Модель проектирования информационной системы

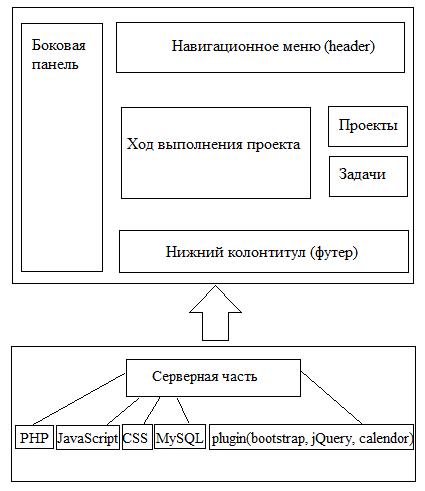


Рисунок А.. Архитектура системы. Диаграмма развертывания

Приложение Б

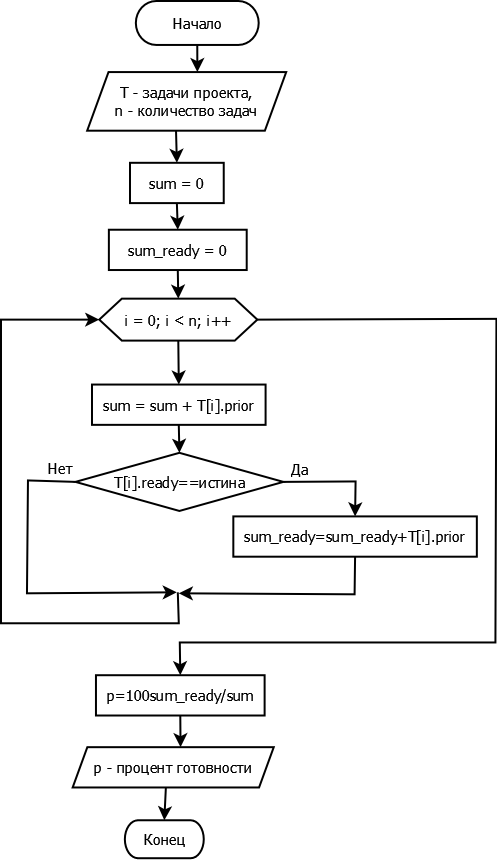


Рисунок Б. Блок-схема расчёта процента готовности проекта

Приложение В

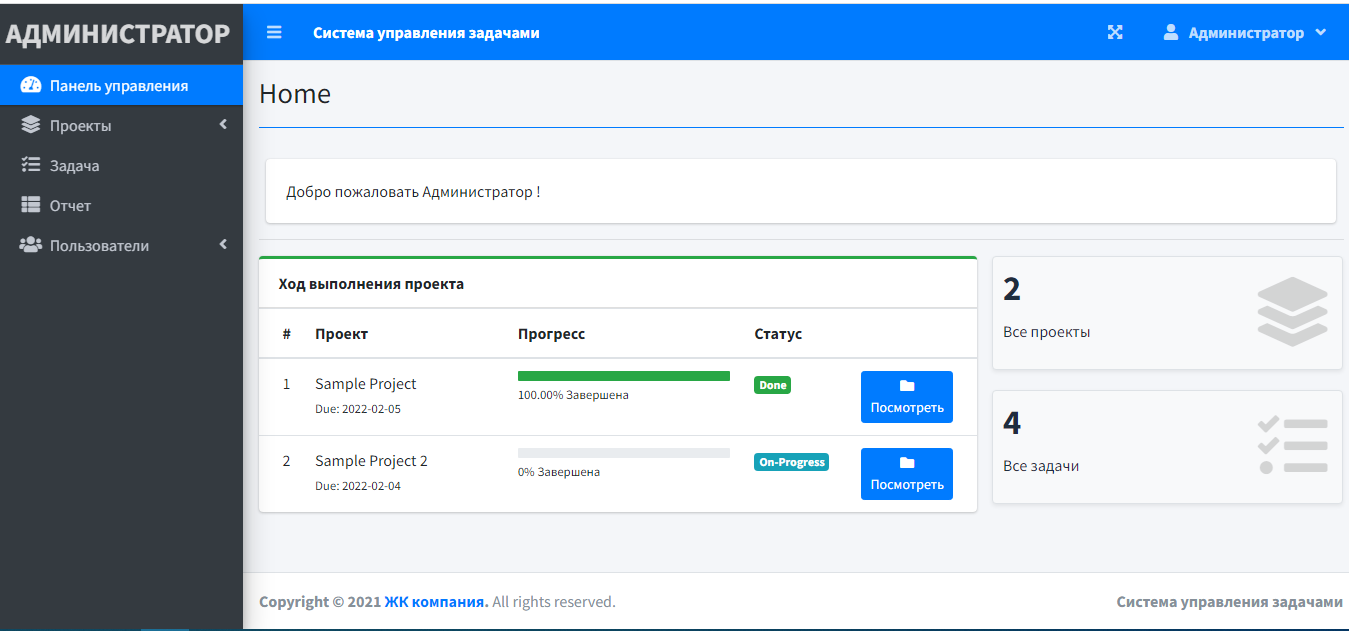


Рисунок В. Интерфейс веб-приложения