СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc199451783)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ 2](#_Toc199451784)

[1.1 Обзор современных подходов к управлению задачами 2](#_Toc199451785)

[1.2 Анализ существующих систем ведения задач 2](#_Toc199451786)

[1.3 Формулировка требований к будущему решению 2](#_Toc199451787)

[1.3.1. Функциональные требования: 2](#_Toc199451788)

[1.3.2. Нефункциональные требования: 2](#_Toc199451789)

[1.4 Выбор средств разработки 2](#_Toc199451790)

[1.4.1. Выбор системы управления базами данных 2](#_Toc199451791)

[1.4.2. Выбор языка программирования 2](#_Toc199451792)

[1.4.3. Выбор технологий для фронтенда 2](#_Toc199451793)

[1.5 Вывод по главе 1 2](#_Toc199451794)

[2 ПРОЕКТНАЯ ЧАСТЬ 2](#_Toc199451795)

[2.1 Архитектура системы 2](#_Toc199451796)

[2.2 Диаграмма вариантов использование 2](#_Toc199451797)

[2.3 Структура автоматизированной системы 2](#_Toc199451798)

[2.4 Проектирование физической модели базы данных 2](#_Toc199451799)

[2.5 Реализация модулей 2](#_Toc199451800)

[2.5.1. Управление задачами (CRUD) 2](#_Toc199451801)

[2.5.2. Статусы и метки 2](#_Toc199451802)

[2.5.3. Доска задач и drag-and-drop 2](#_Toc199451803)

[2.5.4. Логирование действий 2](#_Toc199451804)

[2.5.5. Комментарии и вложения 2](#_Toc199451805)

[2.5.6. Связь между задачами 2](#_Toc199451806)

[2.5.7. Создание задач на основе других задач 2](#_Toc199451807)

[2.6 Пользовательский интерфейс 2](#_Toc199451808)

[2.6.1. Базовый шаблон 2](#_Toc199451809)

[2.6.2. Доска задач 2](#_Toc199451810)

[2.6.3. Просмотр, добавление и редактирование карточки задачи 2](#_Toc199451811)

[2.6.4. Список задач 2](#_Toc199451812)

[2.6.5. Страница документации 2](#_Toc199451813)

[2.7 Выводы по главе 2 2](#_Toc199451814)

[3 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И ВНЕДРЕНИЯ ПРОГРАММНОГО ПРОДУКТА 2](#_Toc199451815)

[3.1 Определение факторов эффективности разработки и внедрения программного продукта 2](#_Toc199451816)

[3.2 Определение трудоёмкости разработки отдельных стадий программного продукта и затрат на оплату труда разработчиков 2](#_Toc199451817)

[3.3 Расчет амортизационных отчислений от стоимости основных средств и нематериальных активов, участвующих в процессе разработки программного продукта 2](#_Toc199451818)

[3.4 Расчёт стоимости оборотных средств на проектирование и внедрение программного продукта 2](#_Toc199451819)

[3.5 Расчёт текущих затрат на проектирование и внедрение программного продукта 2](#_Toc199451820)

[3.6 Расчет прочих затрат на проектирование и внедрение программного продукта 2](#_Toc199451821)

[3.7 Расчет накладных затрат (коммерческие и управленческие расходы) на проектирование и внедрение программного продукта: 2](#_Toc199451822)

[3.8 Определение сметы затрат на разработку программного продукта 2](#_Toc199451823)

[3.9 Расчет экономической эффективности разработки программного продукта 2](#_Toc199451824)

[3.10 Экономические показатели разработки программного продукта 2](#_Toc199451825)

[ЗАКЛЮЧЕНИЕ 2](#_Toc199451826)

ВВЕДЕНИЕ

В современных условиях эффективное управление представляет собой ценный ресурс организации. Следовательно, повышение эффективности управленческой деятельности становится основным направлением совершенствования деятельности предприятия в целом. Однако, современные компании всё чаще сталкиваются с проблемой, связанной с рутинными операциями по постановке, контролю выполнения и проверке задач. Эти процессы требуют значительных временных затрат, особенно при участии тестировщиков, которым необходимо отслеживать статусы задач, находить связанные элементы и документировать обнаруженные проблемы.

Самым очевидным способом решения данной проблемы станет разработка собственного модуля для управления задачами внутри проектов предприятия с помощью визуального инструмента Kanban-доски.

Kanban-доска – это таблица с несколькими столбцами, внутри которых находятся стикеры с задачами. Все задачи, выполняемые в их рамках, можно вручную распределять по стадиям. Перемещать задачи смогут исполнители. Можно оперативно отслеживать, на каком этапе работы находится та или иная задача. Легко поддается контролю остаток времени до завершения выполнения задачи, что позволяет эффективно распоряжаться трудовыми и материальными ресурсами компании.

Хотя популярные системы управления задачами проекта предлагают комплексные решения, их использование так же имеет недостатки: излишняя перегруженность ненужными функциями для отдельных предприятий, необходимость доработки при отсутствии конкретных особенностей, соответствующих нуждам компании. Однако, что не маловажно, существует значительный риск, связанный с использованием зарубежных решений, который проявляется в потенциальных ограничениях доступа, изменениях условий использования или полном уходе сервисов с отечественного рынка.

Актуальность представленного исследования обусловлена потребностью в специализированном модуле, ориентированного на внутренние особенности компании. Такая система позволит снизить трудозатраты сотрудников, повысить прозрачность рабочих процессов и минимизировать влияние человеческого фактора за счёт автоматизации повторяющихся действий.

Целью проекта является разработка модуля системы управления задачами, обеспечивающего автоматизацию процессов. Особое внимание уделено возможности интеграции тестирования в жизненный цикл задачи, а также гибкой настройке меток, фильтров и связи между задачами.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Провести анализ существующих решений и определить ключевые требования к внутренней системе управления задачами.
2. Определить архитектуру модуля и выбрать подходящие технологии для его реализации.
3. Создать прототип системы, включающий базовый функционал:

* создание, редактирование и просмотр как отдельных задач, так и их перечня;
* управление особыми метками для идентификации статуса задачи – лейблами;
* комментирование задач;
* возможность создания задач на основе уже существующих задач по шаблону баг-репорта.

1. Провести тестирование модуля и оценить его эффективность.

Объектом исследования выступает процесс управления и тестирования задач в рамках IT-проектов компании. Предметом исследования выступает методология интеграции тестирования в систему управления задачами с целью повышения её эффективности и снижения доли ручной работы.

Методологическая основа работы включает в себя анализ существующих решений, проектирование архитектуры системы, разработку программного обеспечения с использованием современных web-технологий, а также тестирование функциональности полученного продукта.

Практическая значимость работы в том, что создаваемая система может помочь сотрудникам работать эффективнее, не допуская ошибок, связанных с человеческим фактором, автоматизирует определенные рутинные операции при проверке и создании новых задач, что ускорит процесс тестирования и сэкономит время сотрудников. А самое главное - компания перестанет зависеть от зарубежных сервисов, которые могут в любой момент стать недоступными.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ

## Обзор современных подходов к управлению задачами

Процесс управления задачами является одной из ключевых составляющих успешной реализации любого программного проекта. Он включает в себя планирование, распределение, отслеживание и контроль выполнения задач между участниками команды. Современные подходы к управлению задачами можно условно разделить на два типа: традиционные - каскадные и гибкие - Agile.

Agile — это гибкий подход к управлению проектами по разработке программного обеспечения, который часто применяют в небольших командах. Он возник после того, как в сфере IT устали от излишней бюрократии и строгости.

В основе данного подхода лежат 12 принципов, из которых выделяют четыре ценности системы Agile:

* люди и взаимодействия важнее процессов и инструментов;
* работающий продукт важнее точной и подробной документации;
* сотрудничество с заказчиком важнее условий договора;
* готовность к изменениям важнее следования изначальному плану.

Эти ценности легли в основу большинства современных методологий и позволили создать гибкие фреймворки, которые сегодня широко используются в IT-среде.

Основные гибкие методологии:

Scrum - одна из самых распространённых методологий, ориентированная на итеративную разработку. Она предполагает работу в рамках спринтов - фиксированных временных отрезков, в течение которых выполняется определённый объём задач.

Kanban же отличается меньшей формализацией и ориентирован на непрерывное выполнение задач, визуализируемое в виде доски с колонками статусов. Это дает возможность управлять потоком задач без жёсткой привязки к спринтам.

Scrumban - гибрид Scrum и Kanban. Встречается во многих современных системах. Такая методология выражается в поддержке таких элементов, как доски задач, напоминающие интерфейс Kanban, а также механизмы спринтов, характерные для Scrum. Важной частью является автоматизация перехода задач между статусами, позволяющая снизить ручную работу и повысить прозрачность процессов.

## Анализ существующих систем ведения задач

Современные системы управления задач играют ключевую роль в организации работы IT-команд и позволяют эффективно управлять задачами, отслеживать прогресс выполнения и обеспечивать прозрачность процессов. На рынке представлено множество решений, которые условно можно разделить на три категории:

* корпоративные платформы (Jira, YouTrack) — ориентированы на крупные команды, предоставляют широкие возможности по настройке рабочих процессов;
* devOps-интегрированные системы (GitLab, Azure DevOps) — тесно связаны с циклом разработки ПО и обеспечивают сквозное управление задачами и кодом;
* облегчённые решения (Trello, ClickUp, Notion) — подходят для небольших команд или стартапов, где важна простота и скорость внедрения.

При детальном разборе отличия становятся видны:

1. Jira

Jira — одна из самых известных и гибко настраиваемых систем ведения задач. Изначально созданная как баг-трекинговая система, она со временем превратилась в полноценную платформу, поддерживающую Scrum, Kanban и другие методологии.

Среди основных особенностей отмечают мощную систему фильтров, меток и пользовательских полей, расширенную аналитику и отчетность, интеграцию с Git и другими DevOps инструментами.

Особо популярна в крупных компаниях и командах, которым нужна глубокая настройка процессов. Однако, из этого преимущества вытекает и минус - интерфейс зачастую слишком перегруженным для небольших организаций.

1. Trello

Trello — это простой, но в то же время удобный инструмент управления задачами, основанный на концепции Kanban. Работа с задачами тут организована через визуальные доски, разделённые на колонки и карточки. Интерфейс системы интуитивно понятен, что делает ее делает её популярной среди небольших команд и проектов, где важна скорость настройки и наглядность процессов.

Главным плюсом является доступность и универсальность - большая часть функций работает без ограничений по количеству участников, срокам пользования и другим параметрам

Идеально подходит для быстрого старта, когда нужен наглядный инструмент без сложной настройки. Часто используется вне программной разработки, например, в маркетинговых или стартап-командах.

1. GitLab Issues

Платформа GitLab обеспечивает комплекс всевозможных DevOps-функций, таких как управление кодом, непрерывная интеграция и развертывание, процедуры безопасности и управление задачами. Одним из ключевых инструментов в этой системе являются Issues — механизм создания, обсуждения и контроля задач.

Главное преимущество GitLab заключается в тесной интеграции с Git-репозиториями, что позволяет управлять задачами при помощи коммитов, а также удобно фильтровать и группировать задачи по различным критериям.

Именно эти возможности и делают платформу особенно полезной для технических специалистов, активно использующих автоматизированные процессы и репозитории. Но, в то же время, система может представлять сложности для пользователей, которые плохо знакомы с DevOps-процессами.

Сравнительный анализ рассмотренных решений и разрабатываемого продукта отражен в таблице 1.

1. Сравнительный анализ рассмотренных приложений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Jira | GitLab Issues | Trello | Разрабатываемое решение |
| Основное назначение | управление проектами и баг-трекинг | devops-платформа с задачами | визуальное управление задачами | локальная система управления задачами |
| Поддержка Agile | + | + | + | минималистичная реализация |
| Интеграция с Git | только через плагины | + | - | - |
| Метки и фильтры | + | + | + | + |
| Комментарии к задачам | + | + | + | + |
| История изменений | + | + | - | базовая история действий |
| Создание задач из комментов | + | + | - | в виде создания связанных задач |
| Простота интерфейса | - | - | + | + |
| Наличие бесплатной версии | ограниченная | + | ограниченная | + |
| Зависимость от зарубежного ПО | + | + | + | - |

## Формулировка требований к будущему решению

На основе анализа существующих систем управления задачами были сформулированы основные требования к проекту, ориентированному на потребности ООО «НПЦ «БизнесАвтоматика». К ним относится как функциональная часть системы, так и её нефункциональные характеристики.

### Функциональные требования:

* создание, редактирование и просмотр задач;
* отображение доски с текущими задачами;
* управление метками задач;
* организация зависимостей;
* комментирование задач;
* поддержка вложений;
* возможность создания новых задач на основе существующих.

### Нефункциональные требования:

* простота интерфейса;
* возможность локального развертывания;
* возможность дальнейшей интеграции с корпоративными системами учета сотрудников;
* отсутствие зависимости от зарубежных сервисов;
* кроссплатформенность.

## Выбор средств разработки

Выбор технологий является критически важным этапом при разработке, так как он определяет её скорость, надёжность системы и дальнейшую её поддержку.

### Выбор системы управления базами данных

Средство управления базами данных (СУБД) является ключевым компонентом любого программного обеспечения, ориентированного на хранение и обработку данных. Оно представляет собой программный комплекс, обеспечивающий создание, модификацию, удаление и управление данными в соответствии с заданными правилами и структурой.

На сегодняшний день наиболее распространёнными являются реляционные СУБД, которые организуют данные в виде таблиц и обеспечивают гибкое взаимодействие между ними через связи. Этот подход позволяет эффективно управлять сложными системами и сохранять целостность информации.

Рынок СУБД довольно обширный. Присутствуют как платные продукты, так и продукты с открытым исходным кодом. Каждая из этих систем реализует основные механизмы управления базами данных, а также предлагает свои уникальные функции для работы с данными.

Рассмотрим следующие продукты:

1. MySQL

На сегодняшний день является одной из самых распространённых СУБД. Как правило, используется при разработке веб-сайтов и других веб-приложений. Система обладает открытым исходным кодом, что является одним из значимых факторов ее популярности. Активное сообщество пользователей постоянно улучшает ее, разрабатывая собственные плагины, добавляя новый функционал.

Большинство современных СУБД для своей работы использует язык SQL, однако MySQL использует его не в чистом виде, то есть некоторый функционал языка отсутствует.

К преимуществам данной системы можно отнести:

* СУБД практически полностью поддерживает язык SQL;
* MySQL может достаточно просто работать с большими объемами данных и при необходимости легко масштабироваться;
* некоторые упрощения некоторых стандартов языка SQL позволяют ускорить работу СУБД;

К недостаткам использования MySQL можно отнести:

* неполное соответствие стандартам языка SQL;
* проблемы с надежностью. MySQL из-за некоторых методов обработки информации в некоторых ситуациях уступает другим СУБД.

1. Postgre SQL

Представляет собой одну из наиболее мощных и профессиональных открытых СУБД. В отличие от MySQL, она стремится максимально точно следовать стандартам SQL, что делает её удобной для реализации сложных запросов и транзакций.

К достоинствам данной системы можно отнести:

* это система с открытым исходным кодом, которая соответствует стандартам SQL;
* достаточно большое количество дополнений, которые позволяют расширять функционал системы;
* в отличии от других СУБД она обеспечивает более высокий уровень целостности данных и транзакций;
* поддерживает формат json.

К недостаткам же можно отнести:

* при выполнении достаточно простых операций эта СУБД использует значительно больше ресурсов, чем ее конкуренты;
* несмотря на достаточно большую базу пользователей, трудно найти хостинг, который будет поддерживать Postgre.

1. Microsoft SQL Server

Microsoft SQL server – СУБД, разработанная компанией Microsoft. Так же как MySQL и Postgre, эта СУБД предназначена для управления реляционными базами данных. Microsoft SQL server использует язык Transact-SQL. Данный язык соответствует стандартам языка SQL. Данная СУБД может использоваться как с маленькими базами предприятия, так и с базами огромных компаний.

К преимуществам СУБД можно отнести:

* шифрование всей хранимой информации в БД. Шифрование и расшифровка информации происходит непосредственно перед чтением или записью информации;
* интуитивно понятный интерфейс;
* высокая производительность СУБД;
* легкая масштабируемость системы.

К недостаткам можно отнести:

* ограниченный выбор платформ. СУБД будет работать только под управлением операционных систем семейства windows;
* Microsoft SQL server является не бесплатным ПО.

Выше рассмотрены одни из самых популярных СУБД. Рынок данных систем достаточно обширен и не ограничивается рассматриваемыми СУБД. Результаты сравнения СУБД представлены в таблице 2.

1. Сравнение СУБД

|  |  |  |  |
| --- | --- | --- | --- |
|  | Postgre SQL | MySQL | Microsoft SQL Server |
| Доступность | открытый исходный код | открытый исходный код | платная лицензия |
| Полное соответствие SQL | + | - | + |
| Поддержка JSON | + | + | + |
| Производительность | средняя | высокая | высокая |
| Поддержка транзакций | полная | ограниченная | полная |
| Поддержка платформ | linux, windows | linux, windows | только windows |
| Защита данных | высокая | средняя | высокая |

При использовании MySQL в качестве системы управления базами данных обеспечивается простота внедрения и настройки, что особенно важно на начальном этапе. Поэтому для реализации была выбрана MySQL, как наиболее подходящая по уровню доступности, производительности и простоте использования.

### Выбор языка программирования

Для реализации проекта был выбран язык программирования Python. Такой выбор обусловлен тем, что разработка включает создание веб-интерфейса для управления задачами, где важны удобство реализации и наличие развитой экосистемы. Python активно применяется при разработке веб-приложений, и этому способствуют следующие факторы:

* код на Python легко переносится между различными операционными системами без модификаций;
* имеется большое количество фреймворков, упрощающих создание веб-приложений, а также мощные ORM и инструменты для тестирования и построения API;
* сообщество языка достаточно велико, что позволяет быстро находить решения типовых и нетиповых задач, а также использовать множество готовых примеров.

При разработке фронтенда на Python доступны различные фреймворки. В таблице 3 приведено сравнение двух самых популярных — Flask и Django.

1. сравнение фреймворков Flask и Django

|  |  |  |
| --- | --- | --- |
|  | Flask | Django |
| Простота освоения | очень простой и понятный api | более сложный. требует знания шаблонов django |
| Гибкость | высокая - можно использовать только необходимые компоненты | менее гибкий - предлагает стандартную структуру проекта |
| ORM | нет встроенной orm. легко подключается sqlalchemy | встроенная мощная orm |
| Шаблонизатор | встроенный | встроенный |
| Производительность | высокая производительность на малых проектах | оптимизирован для больших систем |
| Подходит для небольших проектов | да | слишком избыточен |
| Легкость настройки | просто настраивается | требует глубокого понимания архитектуры |
| Интеграция с базой данных | поддержка любой бд через sqlalchemy | postgresql / mysql / sqlite |
| Расширяемость | позволяет постепенно добавлять функционал | функционал встроен |
|  |  |  |

На основании проведённого анализа был выбран Flask. Его ключевым преимуществом оказалось то, что он позволяет строить лёгкие и быстрые веб-приложения без избыточного функционала, который не требуется реализовывать в рамках текущих задач. Это особенно актуально, если нужна гибкая архитектура без лишних зависимостей.

Кроме того, использование Python в связке с Flask и MySQL обеспечивает достаточную производительность и потенциальную возможность интеграции с корпоративными системами на последующих этапах внедрения.

### Выбор технологий для фронтенда

Пользовательский интерфейс был реализован с применением следующих технологий:

* HTML - используется для структурирования веб-страниц;
* CSS - отвечает за стили и внешний вид элементов;
* JavaScript - обеспечивает обработку действий пользователя на стороне клиента;
* Bootstrap 5 - один из наиболее популярных CSS-фреймворков, применяемый для создания современного и адаптивного дизайна;
* Jinja2 - шаблонизатор для генерации HTML-страниц на сервере.

## Вывод по главе 1

Был проведён анализ предметной области, исследованы современные подходы к управлению задачами в проекте, включая Agile-методологию и Kanban-доски как один из её ключевых инструментов, проанализированы существующие системы управления задачами. Проведённый обзор показал, что существующие решения предоставляют широкий функционал, но зачастую требуют подключения к внешним сервисам, что может быть неприемлемо для организаций, заинтересованных в локальном размещении данных и полном контроле над информацией.

На основе анализа были сформулированы функциональные и нефункциональные требования к будущему решению.

Было принято решение о реализации системы на базе Python + Flask, с использованием SQLAlchemy для работы с данными и Bootstrap для создания адаптивного пользовательского интерфейса. Выбранный стек технологий обеспечивает гибкость, простоту внедрения и возможность масштабирования решения.

# ПРОЕКТНАЯ ЧАСТЬ

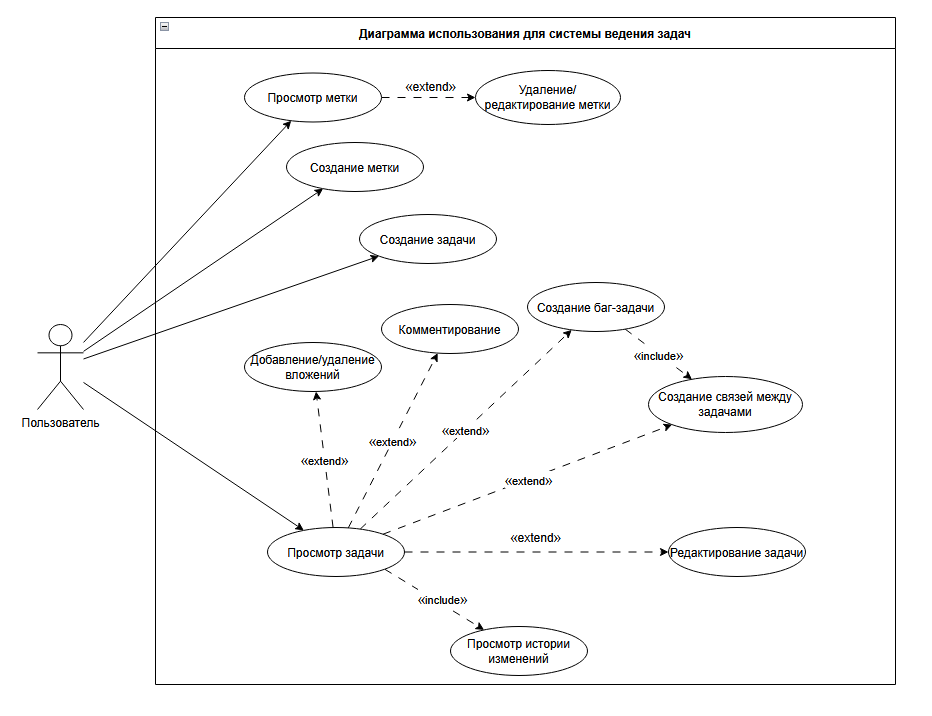
## Архитектура системы

Разрабатываемый модуль представляет собой веб-приложение, ориентированный на управление задачами с использованием принципов Kanban. Для обеспечения масштабируемости и удобства поддержки была выбрана архитектура MVC, адаптированная под специфику Flask-фреймворка.

Для работы с базой данных используется SQLAlchemy — ORM, которая предоставляет удобный способ взаимодействия с реляционными данными, сохраняя независимость от конкретной СУБД.

## Диаграмма вариантов использование

Диаграмма вариантов использования была построена для определения основных функций системы и ролей, в которых будут действовать пользователи. Она позволяет формулировать требования к системе и выделить ключевые модули, реализованные в рамках проекта. Диаграмма представлена на рисунке 1.



1. Диаграмма вариантов использования

Из участников выделен пользователь – любой, кто взаимодействует с системой: разработчики, тестировщики, аналитик и так далее. Все пользователи имеют одинаковый уровень доступа, что дает возможность не воспроизводить разделение на роли на данном этапе проектирования.

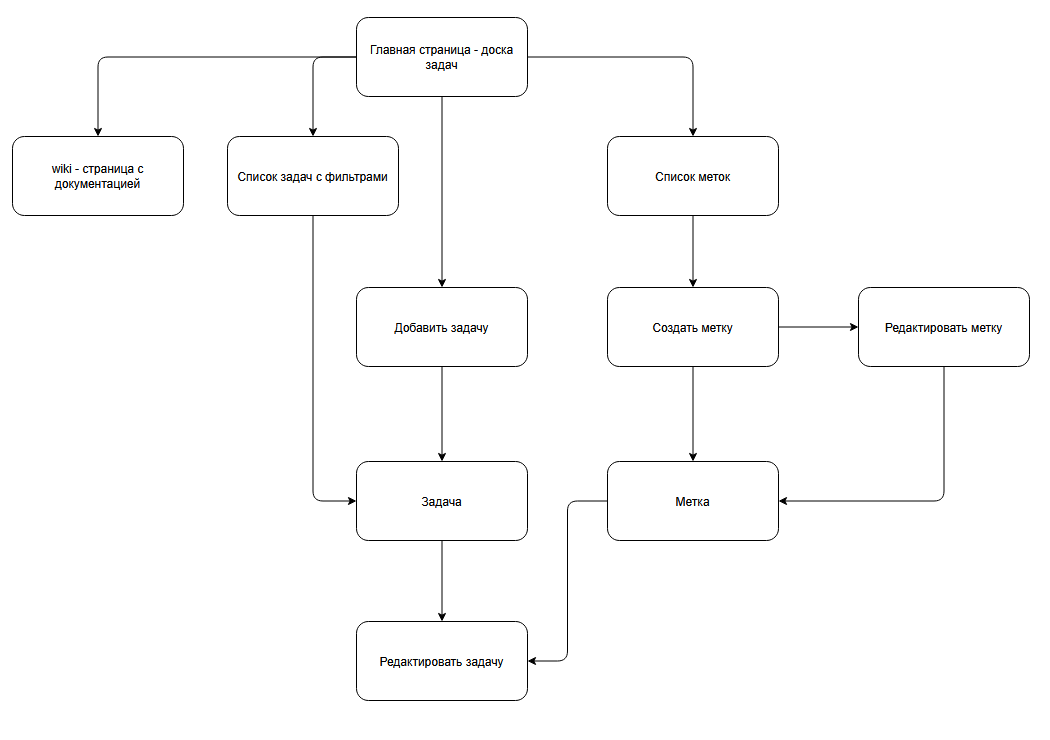
Основываясь на диаграмме прецедентов, представленной выше, выделим основные варианты использования:

* просмотр доски задач;
* создание новой задачи;
* просмотр деталей задачи;
* редактирование задачи;
* добавление и удаление вложений;
* комментирование задачи;
* создание и удаление зависимостей между задачами;
* фильтрация и сортировка задач;
* созданием меток;
* создание и удаление зависимостей между задачами и метками.

## Структура автоматизированной системы

Следующим шагом было спроектирована структура веб-приложения автоматизированной системы управления задачами, рисунок 2.

Структура сайта – это схема размещения его основных разделов и страниц относительно друг друга. Это план-схема, которая показывает, каким образом строится сайт, логическая связка его страниц.



1. Структура веб-приложения

На структуре можно выделить следующие страницы

1. Главная страница – доска задач

Это центральный узел системы. На главной странице отображается Kanban-доска, где задачи группируются по статусам. Пользователь может:

* перетаскивать задачи между колонками для изменения их статуса;
* добавлять новые задачи через кнопку: «добавить задачу»;
* переходить к детальной информации о задаче через клик на заголовок задачи.

1. Список задач с фильтрами

Страница «Список задач» предоставляет расширенные возможности поиска и фильтрации задач:

* возможность сортировки задач по различным параметрам;
* фильтр по дате создания, дате завершения и другим критериям;
* визуальное представление задач в виде таблицы с возможностью перехода к детальной информации о каждой задаче.

1. Кнопка «Добавить задачу»

На главной странице есть кнопка "Добавить задачу", которая открывает форму для создания новой задачи. После сохранения задача автоматически добавляется на Kanban-доску в соответствующую колонку.

1. Задача

Каждая задача имеет свою детальную страницу, доступную через клик на заголовок на карточке задачи на доске или через двойной клик по задаче в списке задач. На странице задачи пользователь может:

* просматривать подробную информацию о задаче - заголовок, описание, статус, метки, дата выполнения и так далее.;
* открыть форму редактирования задачи через специальную кнопку;
* добавлять комментарии;
* загружать и просматривать вложения;
* создавать связанные задачи, в том числе через создание новой баг-задачи на основе текущей;
* просматривать историю изменений (активность).

1. Страница «Список меток»

Страница меток позволяет:

* просматривать все метки с их названиями, цветами и описанием;
* создавать новые метки через кнопку «Новая метка»;
* искать метки по их названию в специальном поле.

1. Страница «Wiki»

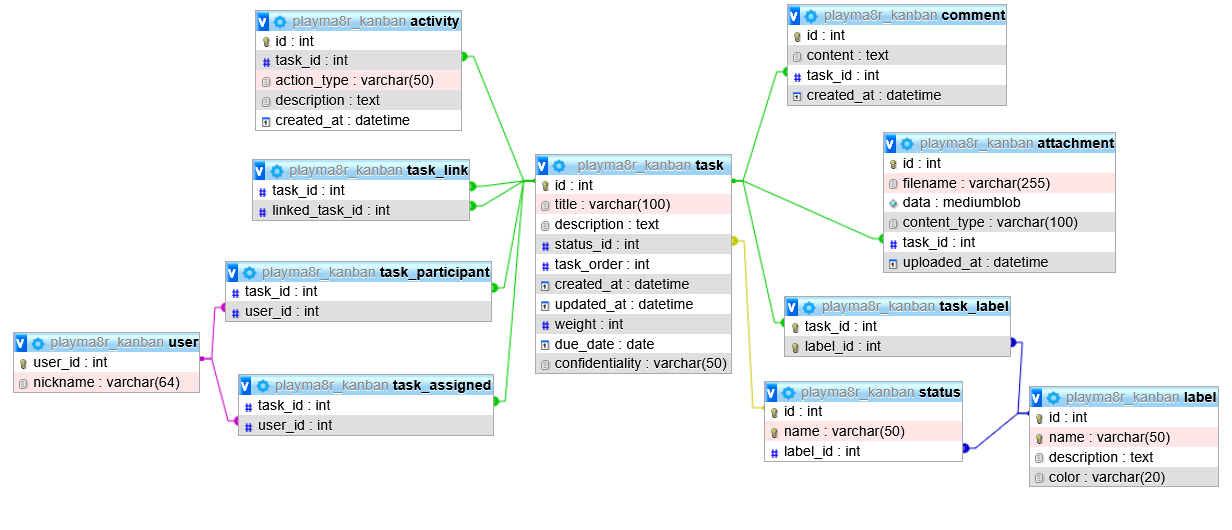
В верхней части диаграммы расположена ссылка на страницу wiki, которая содержит справочную информацию о системе, такую как описание функционала или инструкции по ее использованию.

## Проектирование физической модели базы данных

Структура базы данных спроектирована таким образом, чтобы обеспечить гибкость управления задачами, их связями и метками. Основные таблицы:

* task - хранение информации о задаче: заголовок, описание, статус, вес, дата выполнения, уровень конфиденциальности;
* status - определяет текущее состояние задачи; связан с метками через внешний ключ;
* label - пользовательские метки, применяемые к задачам;
* user - информация о участниках проекта;
* task\_label - таблица связи многие-ко-многим между задачами и метками;
* task\_link - реализует связи между задачами, в виде таблицы связи многие-ко-многим;
* task\_participant - реализует связи участников и задач, в виде таблицы отношение многие-ко-многим;
* task\_assigned – таблица связи многие-ко-многим между задачами и исполнителями;
* comment, attachment, activity - дополнительные сущности, обеспечивающие функциональность комментариев, вложений и истории действий над задачами.

На рисунке 3 представлена физическая модель базы данных.



1. Физическая модель базы данных

Для взаимодействия с базой данных используется ORM SQLAlchemy. Это позволило абстрагироваться от SQL-запросов и работать с данными в объектно-ориентированном стиле. Примеры моделей:

## Реализация модулей

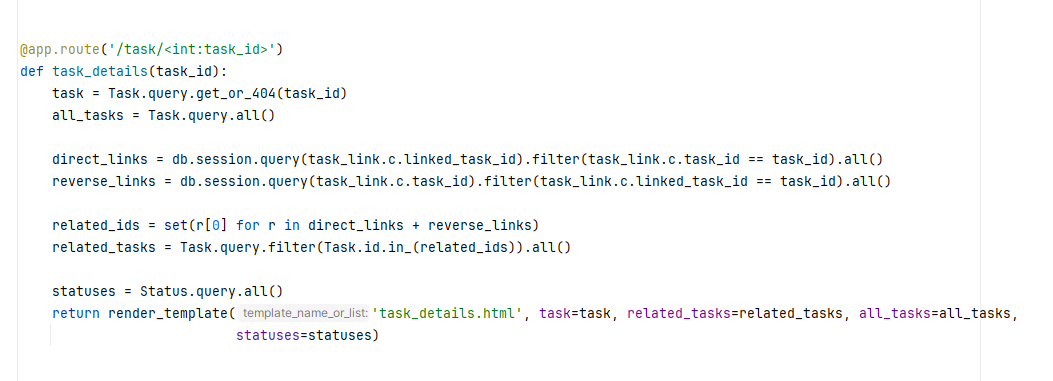
### Управление задачами (CRUD)

Задачи представляют собой центральный элемент системы. Были реализованы все операции CRUD:

* создание - задача создаётся через форму, где указываются заголовок, описание, статус по умолчанию, вес, дата завершения, метки;
* чтение - задача отображается как на доске, так и в списке, или как отдельная карточка;
* обновление - доступно полное редактирование задачи, включая изменение статуса, веса, даты, меток;
* удаление - не предусмотрено в явном виде, но возможно через закрытие задачи или удаление всех связей.

При любом изменении задачи автоматически записывается событие в таблицу «activity». Это позволяет отслеживать историю изменений.

Пример реализации операции чтения представлен на рисунке 4.



1. реализация функции «чтение»

Этот маршрут реализует операцию «read» из модели CRUD и предоставляет пользователю полную информацию о задаче, а также возможность взаимодействия с дополнительными элементами: комментарии, вложения, связанные задачи, история изменений.

### Статусы и метки

Метки используются как универсальный инструмент классификации задач. Они могут быть произвольными, с возможностью выбора цвета и описания. Некоторые метки привязаны к статусам, что позволяет определять цвет и видимость задачи визуально.

Изменение статуса задачи осуществляется через API-запрос.

На рисунке 5 представлен пример метки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

1. Пример метки «doing»

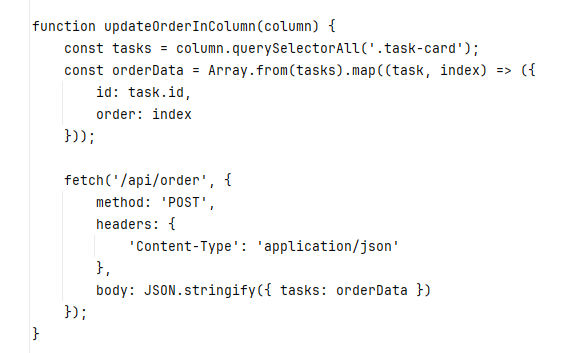
В данном случае название метки – «doing». По приведенному описанию можно понять, что метка предназначается для идентификации задач, которые находятся в процессе выполнения разработчиком.

### Доска задач и drag-and-drop

Одним из ключевых компонентов является Kanban-доска, реализованная в виде набора колонок, соответствующих статусам. В каждой колонке отображаются задачи, которые можно перетаскивать как внутри одной колонки, так и между статусами.

Перетаскивание задач реализовано с визуальной подсветкой зоны сброса и корректной сортировкой внутри колонки. Функционал переноса задач между столбцами реализован с помощью JavaScript - обновление статуса в базе данных происходит через put-запрос, а сохранение порядка задач внутри колонки через post-запрос. Функция для изменения порядка внутри колонок представлена на рисунке 6.

* PUT-запрос в основном используется для изменения всех полей объекта или создание объекта с заранее известным идентификатором;
* POST-запрос же зачастую используется при создании нового объекта при неизвестном идентификаторе или для вызова какой-то процедуры.



1. Функция для изменения порядка задач в колонке

### Логирование действий

Любые действия над задачами записываются в таблицу «activity». При этом регистрируются:

* создание задачи;
* изменение заголовка, описания, веса, статуса, меток;
* добавление и удаление вложений;
* комментирование;
* связывание задач;
* назначение и удаление назначенных на задачу пользователей.

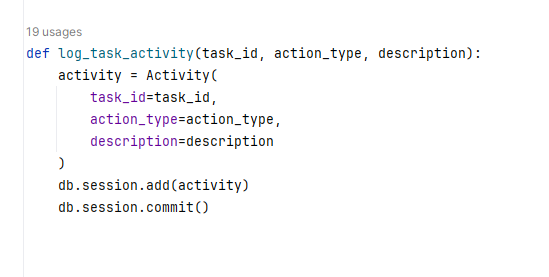
Эта информация доступна на странице просмотра задачи в разделе «Активность» - смотреть рисунок 7. Использование разных иконок и цветов помогает быстро находить нужные события.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

1. Раздел «Активность»

Реализация функции логирования представлена на рисунке 8.



1. Функция логирования событий

Функция «log\_task\_activity» записывает в таблицу «activity» данные по задаче, над которой совершилось действие, тип активности и описание события, если необходимы дополнительные данные.



1. Вызов функции «log\_task\_activity»

Функция используется при любом значимом изменении задачи, например, при изменении заголовка, описания, добавлении или удалении вложений. Пример вызов представлен на рисунке 9, на строке 125.

### Комментарии и вложения

Каждая задача может иметь комментарии, которые добавляются через простую форму. Комментарии хранятся в таблице «comment».

Кроме того, реализована возможность прикреплять файлы к задаче. Все вложения сохраняются в виде BLOB в таблице «attachment». Предусмотрена загрузка, просмотр изображений и удаление вложений. Реализация функции для загрузки вложений представлена на рисунке 10.

BLOB – тип данных, предназначенный, для хранения бинарных данных, в первую очередь, для изображений, а также компилированного программного кода.



1. Функция загрузки вложений

### Связь между задачами

Модуль связи между задачами реализован через таблицу «task\_link». Она поддерживает двусторонние связи: если задача А связана с задачей Б, то и Б будет связана с А. Связь создаётся через выпадающий список, где выбирается другая задача. После этого появляется возможность перехода к связанной задаче, а также удаления связи.

Изображение выглядит как текст, линия, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

1. Раздел «Связанные задачи»

Эта функциональность особенно полезна при работе с тестированием, когда баги создаются на основе существующих задач. Список связанных задач представлен в разделе «Связанные задачи» на рисунке 11.

### Создание задач на основе других задач

Система позволяет создавать новую задачу, например баг, на основе уже открытой задачи. При этом создаётся ссылка между задачами, и баг получает статус «Открытый» и метку «Bug».

Эта особенность делает процесс тестирования более автоматизированным и упрощает работу с ошибками, найденными при проверке задач. Окно создание баг-задачи (рисунок 12) открывается как модальное окно, что позволяет после заполнения необходимых данных оперативно вернуться к проверке задачи, в которой были найдены ошибки.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

1. Создание баг-задачи

## Пользовательский интерфейс

### Базовый шаблон

Все страницы наследуются от общего шаблона base.html, который содержит меню навигации, подключение стилей и скриптов, а также общие java-script функции, например открытие карточки задачи по клику.

Меню с вкладками для навигации по системе расположено слева и остаётся фиксированным при прокрутке контента.

### Доска задач

Главная страница содержит доску задач. Она состоит из нескольких колонок, каждая из которых соответствует определённому статусу. В правом верхнем углу отображается количество задач в каждом статусе.

Интерфейс главной страницы с доской представлен на рисунке 13. На карточке каждой задачи отображаются подвязанные к задаче метки.

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

1. Доска задач

### Просмотр, добавление и редактирование карточки задачи

Карточка задачи — это подробное представление задачи с возможностью:

* редактирования;
* добавления вложений и комментариев;
* установки связей с другими задачами;
* просмотра истории изменений.

Интерфейс разделён на две части:

* основная информация, история изменений, комментарии и вложения;
* сайдбар с метками, весом, исполнителем и участниками.

В зависимости от действия, производимого с карточкой задачи, интерфейс варьируется. Так, например, на странице создания задачи отсутствуют разделы «Активность», «Комментарии», «Связанные задачи» и «Вложения», но присутствуют самые необходимые поля при создании задачи.

На рисунке 14 показан внешний вид страницы для просмотра содержимого задачи.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

1. Просмотр содержимого задачи

В рамках данного дипломного проекта некоторые функции реализованы как «заглушки», так как отсутствует необходимость их создания, исходя из целей проекта. Примером такой «заглушки» является установка степени конфиденциальности задачи.

### Список задач

Страница «tasks.html» представляет собой таблицу со списком всех задач. Её интерфейс изображён на рисунке 15. Реализованы:

* поиск по любому столбцу;
* сортировка по столбцам;
* двойной клик открывает карточку задачи;
* отображение меток в сокращённом виде.

Таблица сделана адаптивной, а данные обновляются в реальном времени при изменении в другой вкладке, после перезагрузки страницы. Так же, при большом количестве меток, предусмотрена подсказка – добавление специального элемента, который скрывает часть меток, если их количество больше двух. При наведении на данный элемент всплывает перечень всех скрытых меток.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

1. Список задач

### Страница документации

Страница «wiki.html» содержит краткое описание системы, её назначение и возможности. Может использоваться как справочный материал для новых пользователей.

## Выводы по главе 2

Была спроектирована и реализована архитектура системы. Реализация выполнена в соответствии с моделью MVC, что позволило разделить логику представления, бизнес-логику и работу с данными. Это сделало код более читаемым, поддерживаемым и пригодным к дальнейшему расширению.

Спроектирована физическая модель базы данных, включающая таблицы для хранения задач, их статусов и меток, истории изменений, комментариев, вложений и связей между задачами. Использование ORM SQLAlchemy позволило упростить взаимодействие с базой данных.

Реализованы основные функции модуля:

* crud-операции над задачами;
* управление метками и статусами;
* логирование действий пользователя;
* добавление и удаление вложений;
* создание и удаление связей между задачами.

Пользовательский интерфейс был реализован с применением адаптивного дизайна и включает следующие страницы:

* главная страница — kanban-доска с задачами, разделёнными по статусам.
* список задач — таблица со списком всех задач с возможностью фильтрации и сортировки.
* карточка задачи — детальная информация о задаче, история изменений, комментарии, вложения и связи.
* управление метками — создание, редактирование и удаление меток.
* страница документации — справочная информация о системе и её возможностях.

В результате выполнения первой и второй глав были достигнуты следующие результаты:

* проведён анализ требований и выбраны технологии реализации;
* спроектирована и реализована физическая модель базы данных;
* создан прототип системы с базовым функционалом

Эти результаты легли в основу третьей главы, посвящённой экономической целесообразности разработки и внедрения программного продукта. Разработанный модуль полностью соответствует поставленным целям.

# ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И ВНЕДРЕНИЯ ПРОГРАММНОГО ПРОДУКТА

## Определение факторов эффективности разработки и внедрения программного продукта

Разработанный программный продукт представляет собой систему автоматизации управления задачами проектами – Канбан-доску, предназначенную для использования в компании по разработке программного обеспечения ООО НПЦ «БизнесАвтоматика».

Система ориентирована на всех участников проектной группы – руководителей проекта, разработчиков, тестировщиков и аналитиков.

Продукт включает в себя следующие основные функции:

* автоматизированное создание задач с заданными входными данными по единому шаблону;
* просмотр и редактирование созданных задач;
* присвоение задачам специальных меток для упрощенной идентификации задач как по их особенностям, так и по этапу жизненного цикла;
* комментирование задач;
* создание связей между задачами для ориентирования в проекте.

С внедрением автоматизированной системы работники получат:

* систему с современным подходом к ведению проекта;
* сокращенное время на ориентацию в проекте и нахождение проблем в разрабатываемом продукте;
* упрощение рутинных операций при оформлении задач;
* средство для прямого взаимодействия по задачам проекта;
* повышение контроля за выполнением задач.

Таким образом, внедрение разработанной системы позволит участникам проекта эффективно обмениваться информацией по проекту за счет наличия функции комментирования и упростит процесс разработки, аналитики и тестирования проектов внутри компании, а также за счет понятного интерфейса уменьшает временные затраты на ознакомление с проектами для новых сотрудников.

## Определение трудоёмкости разработки отдельных стадий программного продукта и затрат на оплату труда разработчиков

Трудоемкость относится к ключевым экономическим показателям и позволяет дать [оценку эффективности](http://fb.ru/article/44998/otsenka-effektivnosti-reklamyi) использования рабочего времени при производстве товаров или услуг, а также при выполнении каких-либо работ. Этот коэффициент говорит о том, сколько труда (часов, дней или месяцев) нужно затратить для изготовления одной единицы продукции.

Трудоемкость разработки программного продукта определяется на основании времени реального времени, затраченного на его разработку.

Результаты длительности работ сведены в таблицу 4:

1. Длительность работ при разработке программного продукта

|  |  |  |
| --- | --- | --- |
| Наименование работ | Длительность работ, дней | Длительность работ, мес. |
| Анализ требований | 4 | 0,2 |
| Определение спецификации | 4 | 0,2 |
| Проектирование | 7 | 0,3 |
| Кодирование | 22 | 1,0 |
| Тестирование | 14 | 0,6 |
| Сдача темы | 5 | 0,2 |
| Итого: | 56 | 2,5 |

Длительность работ при разработке программного продукта в месяцах рассчитывается путем деления длительности работ в днях (Дрд) на количество рабочих дней в месяце (Кд = 22 дня):

Дрм = Дрд / Кд, [мес.] (1)

* анализ требований: дрм = 4 / 22 = 0,2;
* определение спецификации: дрм = 4 / 22 = 0,2;
* проектирование: дрм = 7 / 22 = 0,3;
* кодирование: дрм = 22 / 22 = 1,0;
* тестирование: дрм = 14 / 22 = 0,6;
* сдача темы: дрм = 5 / 22 = 0,2.

Распределим выполняемые виды работ по исполнителям и определим их заработную плату за выполняемый объем работ, результаты расчетов сведем в таблицу 5.

1. Расчет основной заработной платы разработчиков программного продукта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование работ | Исполнитель | Длительность работ, мес. | Оклад специалиста, руб. | Затраты на ЗП, руб. |
| Анализ требований | специалист по информационному обеспечению | 0,2 | 156 000 | 31 200 |
| Определение спецификации | специалист по информационному обеспечению | 0,2 | 156 000 | 31 200 |
| Проектирование | программист | 0,3 | 120 000 | 36 000 |
| Кодирование | программист | 1,0 | 120 000 | 120 000 |
| Тестирование | специалист по тестированию | 0,6 | 84 000 | 50 400 |
| Сдача темы | программист | 0,2 | 120 000 | 24 000 |
| Итого (ЗПобщ): | | | | 292 800 |

Оклад специалиста (Ок) зависит от должности исполнителя. Величина оклада программиста соответствует среднему окладу, сложившемуся на рынке труда. Соотношение месячного оклада программиста к месячному окладу специалиста по информационному обеспечению определяется соотношением 1 : 1,3, а к месячному окладу специалиста по тестированию - как 1 : 0,7, таким образом:

* оклад программиста (Оп) = 120 000 руб.;
* оклад специалиста по информационному обеспечению (Осио = Оп × 1,3) = 156 000 руб.;
* специалиста по тестированию (Ост = Оп × 0,7) = 84 000 руб.

Затраты на заработную плату по видам работ определяются путем умножения длительности работы в месяцах (Дрм) на оклад (Ок).

ЗП = Дрм × Ок, [руб.] (2)

* анализ требований: зп = 0,2 \* 156 000 = 31 200 руб.;
* определение спецификации: зп = 0,2 \* 156 000 = 31 200 руб.;
* проектирование: зп = 0,3 \* 120 000 = 36 000 руб.;
* кодирование: зп = 1,0 \* 120 000 = 120 000 руб.;
* тестирование: зп = 0,6 \* 84 000 = 50 400 руб.;
* сдача темы: зп = 0,2 \* 120 000 = 24 000 руб.

Результаты проведенных расчетов сведены в таблицу 5.

Отчисления от общего фонда оплаты труда рассчитываются по формуле:

Озп = ЗПобщ х 0,30, [руб.] (3)

где ЗП общ – общие затраты на заработную плату за выполненный объем работ, руб.

ЗП общ = 31200 + 31200 + 36000 + 120000 + 50400 + 24000 = 292 800 руб.

Озп = 292 800 x 0,30 = 87 840 руб.

Фонд оплаты труда работников за разработку программного продукта определяется по формуле:

ФОТ = ЗПобщ + Озп, [руб.] (4)

где ЗП общ – общие затраты на заработную плату за выполненный объем работ, руб.

Озп – сумма отчислений от общего фонда заработной платы, руб.

ФОТ = 292 800 + 87 840 = 380 640 руб.

## Расчет амортизационных отчислений от стоимости основных средств и нематериальных активов, участвующих в процессе разработки программного продукта

Для разработки программного продукта используются основные фонды и нематериальные активы.

Основные фонды - средства труда, которые участвуют в производственном процессе многократно, сохраняя при этом свою натуральную форму, но переносят часть стоимости на производимую продукцию по мере износа. К основным фондам относятся здания, сооружения, машины и оборудование, измерительные и регулирующие приборы и устройства, вычислительная техника, транспортные средства, инструмент и прочие основные средства.

Перечень основных средств, используемых при разработке и внедрении программного продукта, и их стоимость приведены в таблице 6.

1. Перечень основных средств, используемых при разработке и внедрении программного продукта

|  |  |  |
| --- | --- | --- |
| Наименование используемых основных средств | Модель основных  средств | Стоимость основных средств (руб.) |
| Персональный компьютер | dell optiplex 7010 plus | 120 000 |
| Монитор | xiaomi 2k monitor g27qi | 15 000 |
| Итого (Сос): | | 135 000 |

Стоимость основных средств берется в соответствии со средней сложившейся на рынке ценой.

Расчет амортизационных отчислений от стоимости основных средств рассчитывается по формуле:

Аос = (Сос х На.ос.) / 100, [руб.] (5)

где Сос – стоимость основных средств, руб.

На.ос. – норма амортизационных отчислений, % (На = 6 %)

Аос = (135 000 \* 6) / 100 = 8 100 руб.

Нематериальные активы – это ценности, имеющие стоимостное выражение и не являющиеся физическими объектами, стоимость которых также, как и основных средств, включается в затраты частично, в соответствие с начисленным износом. Перечень нематериальных активов, используемых при разработке и внедрении программного продукта, и их стоимость приведены в Таблице 7.

1. Перечень нематериальных активов, используемых при разработке и внедрении программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование используемых нематериальных активов | Характеристика нематериальных активов | Количество | Стоимость нематериальных активов (руб.) |
| Windows 11 | операционная система | 1 | 5000 |
| IDE PyCharm Community Edition | среда разработки | 1 | 0 |
| MySQL | системы управления базами данных | 1 | 0 |
| Open Server Panel | локальный веб-сервер | 1 | 0 |
| Итого (Сна): | | | 5000 |

Наиболее часто к нематериальным активам относят используемое для разработки программное обеспечение, стоимость которого определяется на основе сложившейся на рынке цены.

Расчет амортизационных отчислений от стоимости нематериальных активов рассчитывается по формуле по формуле:

Ана = (Сна х На.на,) / 100, [руб.] (6)

где Сос – стоимость нематериальных активов, руб.

На.на. – норма амортизационных отчислений, % (На = 3 %)

Ана = 5 000 \* 3 / 100 = 150 руб.

Сумма амортизационных отчислений, приходящаяся на программный продукт равна:

Ао = ((Аос + Ана) / 12) х Дрм, [руб.] (7)

где Аос - амортизационные отчисления от стоимости основных средств, руб.

Ана - амортизационные отчисления от стоимости нематериальных активов, руб.

Дрм - длительности работы в месяцах;

Ао = ((8 100 + 150) / 12) \* 2,5 = 1 719 руб.

## Расчёт стоимости оборотных средств на проектирование и внедрение программного продукта

Для разработки и внедрения программного продукта на предприятии помимо основных средств используются еще оборотные средства, к которым относятся бумага, картриджи, магнитные носители информации, флэш-карты и т. д. Стоимость используемых оборотных средств сведена в таблицу 8.

1. Стоимость используемых в процессе разработки программного продукта оборотных средств

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование материала | Профиль, марка, сорт, размер | Единица измерения | Норма расхода / количество | Цена за единицу, руб. | Сумма, руб. |
| Бумага | a4, 80 г/м² | пачка 500 листов | 1 | 500 | 500 |
| Картридж | лазерный  картридж  cactus | штука | 2 | 1300 | 2600 |
| Флеш - карта | usb digma drive2 64гб | штука | 2 | 400 | 800 |
| Итого (Соб.ср): | | | | | 3 900 |

## Расчёт текущих затрат на проектирование и внедрение программного продукта

Перечень текущих затрат зависит от расходов, связанных с разработкой программного обеспечения и области его применения. Наиболее часто встречаются затраты на:

* затраты на электроэнергию, потребляемую персональным компьютером и монитором:

Компьютер: в среднем потребляет в час 0,4 кВт/час электроэнергии, соответственно за восемь часов работы потребление составит 3,2 кВт/часа. Количество электроэнергии, потребленное компьютером в процессе разработки:

Сэл. = (3,2 × Дрд) × Цэл, [руб.] (8)

где Дрд – длительность работы компьютера в днях;

Цэл. – стоимость 1 кВт/часа электроэнергии, руб.

Сэл. = (3,2 × 56) × 6,15 = 1 102,08 руб.

Монитор: в среднем потребляет в час 0,025 кВт/час электроэнергии, соответственно за восемь часов работы потребление составит 0,2 кВт/часа. Количество электроэнергии, потребленное монитором в процессе разработки:

Сэл. = (3,2 × Дрд) × Цэл, [руб.] (8)

где Дрд – длительность работы компьютера в днях;

Цэл. – стоимость 1 кВт/часа электроэнергии, руб.

Сэл. = (0,2 × 56) × 6,15 = 68,88 руб.

* коммунальные услуги, которые рассчитываются по формуле:

Ску = Дрм × Кп, [руб.] (9)

где Дрм - длительности работы в месяцах;

Кп – стоимость коммунальных платежей за месяц, руб. (Кп. = 9500 руб. /мес.)

Ску = 2,5 × 9500 = 23 750 руб.

* оплата интернет-соединения, которое необходимо для разработки продукта, рассчитывается по формуле:

Сис = Цис × Дрм, [руб.] (10)

Дрм — длительность разработки в месяцах;

где Цис. — стоимость Интернет-услуг в месяц, руб.

Сис = 900 × 2,5 = 2 250 руб.

Текущие затраты определяются за период времени, затраченный на разработку программного продукта и сведены в таблицу 9.

1. Текущие затраты на разработку и внедрение программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Вид текущих  затрат | Единица  измерения | Тариф,  руб. | Стоимость за период разработки, руб. |
| Затраты на электроэнергию, потребляемую компьютером | кВт/час | 6,15 | 1 102,08 |
| Затраты на электроэнергию, потребляемую монитором | кВт/час | 6,15 | 68,88 |
| Затраты на коммунальные услуги | руб/мес | 9 500 | 23 750 |
| Затраты на оплату интернет-соединения | руб/мес | 900 | 2 250 |
| Итого (Тз): | | | 27 170,96 |

## Расчет прочих затрат на проектирование и внедрение программного продукта

Прочие затраты принимаются в размере 5 – 10 % от суммы общей заработной платы.

Пр.з = ЗПобщ × %Пр.з, [руб.] (11)

где ЗПобщ – сумма общей заработной платы, руб.

Пр.з = 292 800 × 0,1 = 29 280 руб.

## Расчет накладных затрат (коммерческие и управленческие расходы) на проектирование и внедрение программного продукта:

Накладные расходы принимаются в размере 3 – 5 % от суммы общей заработной платы.

Нр = ЗПобщ × %Нр, [руб.] (12)

где ЗПобщ – сумма общей заработной платы, руб.

Нр = 292 800 × 0,04 = 11 712 руб.

## Определение сметы затрат на разработку программного продукта

Себестоимость продукции - общая величина затрат на производство и реали­зацию продукции в денежном выражении. Фактические затраты, понесенные предприятием в связи с производством и реализацией продукции, группируются в разрезе калькуляционных статей.

Калькуляционная статья – определенный вид затрат, образующих себестоимость продукции в целом или отдельного её вида.

Калькуляция полной себестоимости разработки программного продукта представлена в форме таблицы 10.

1. Калькуляция себестоимости разработки программного продукта.

|  |  |  |
| --- | --- | --- |
| Номер  п/п | Наименование статей расходов | Сумма (руб.) |
| 1. | Фонд оплаты труда (ФОТ) | 380 640 |
| 2. | Сумма амортизационных отчислений (Ао) | 1 719 |
| 3. | Стоимость оборотных средств (Соб.ср) | 3 900 |
| Цеховая себестоимость (Сц) | | 386 259 |
| 4. | Текущие затраты (Тз): | 27 170,96 |
| 5. | Прочие затраты (Пр.з) | 29 280 |
| Производственная себестоимость (Спр) | | 442 709,96 |
| 6. | Накладные расходы (Нр) | 11 712 |
| Полная себестоимость (Сп) | | 454 421,96 |

## Расчет экономической эффективности разработки программного продукта

Экономия от замены ручной обработки информации на автоматизированную:

* затраты на ручную обработку информации:

Зр = Ои × Сч × Гд / Нв, [руб.] (13)

где Ои – объём информации, обрабатываемой вручную, Мбайт;

Гд – коэффициент, учитывающий дополнительные затраты времени на логические операции при ручной обработке информации (Гд = 3,4 (установлен экспериментально));

Нв – норма выработки, Мбайт/час. (Нв = 0,05 проектов/час)

Сч – стоимость одного часа работы, руб./час.

Разрабатываемый продукт предназначен для автоматизации процесса управления задачами проекта, в частности, для проверки корректности их выполнения.

В рамках одной задачи обрабатывается:

* 1 изображение (≈ 0,2 МБ);
* текстовое описание задачи (≈ 0,004 МБ);
* комментарии с пояснениями (≈ 0,001 МБ).

Исходя из этих параметров можно рассчитать объём информации, обрабатываемой в год:

Ои = Оз × Кз × Кд.г, [Мбайт] (14)

Где Оз – средний объём одной задачи, Мбайт;

Кз – среднее количество подготовленных к тесту задач в день, штук (Кз = 4);

Кд.г – количество рабочих дней в году, дни (Кд.г = 247 дней)

Ои = (0,2 + 0,001 + 0,004) \* 4 \* 247 ≈ 203 Мбайт

Стоимость одного часа работы рассчитывается по формуле:

Сч = ФОТ / (Дрм× Кд.м. × Др), [руб.] (15)

где ФОТ – фонд оплаты труда, руб.

Дрм - длительности работы в месяцах;

Кд.м – количество рабочих дней в месяце, дни (Кд.м = 22 дня)

Др – длительность рабочего дня, час. (Др = 8 часов)

Сч = 380 640 / (2,5 × 22 × 8) = 380 640 / 440 = 856,1 руб/час

Зр = 203 × 856,1 × 3,4 / 0,05 = 11 817 604 руб.

* затраты на автоматизированную обработку информации:

За = (ta × См) + tо × (См + Сч), [руб.] (16)

где ta – время автоматической обработки информации, ч. (≈ 3 минут = 0.05 ч.);

Сч – стоимость одного часа работы оператора, руб./час.

tо – время работы оператора, ч., которое рассчитывается по формуле:

tо = Дрм × Кд.м. × Др, [час.] (17)

tо = 2,5 × 22 × 8 = 440 часов.

См – стоимость одного часа машинного времени, руб./час., которое рассчитывается по формуле:

См = (Аос / ФРВ) + (Мк × Цэл) × Ν, [руб.] (18)

Где ФОТ – фонд оплаты труда, руб.

Аос - сумма амортизационных отчислений от стоимости основных средств, руб.

ФРВ – годовой фонд рабочего времени в часах. При 40-часовой рабочей неделе ФРВ = 1972 часа (данные производственного календаря на 2025 год).

Мк – мощность компьютера или другого потребителя, кВт/час.

Цэл. – стоимость 1 кВт/часа электроэнергии, руб.

Ν – количество потребителей электроэнергии, шт.

См = (8 100 / 1972) + (0,4 × 6,15) × 2 = 13,12 руб./час

За = (0,05 × 13,12) + 440 × (13,12 + 856,1) ≈ 382 457 руб.

* экономия от замены ручной обработки информации на автоматизированную:

Эу = Зр - За, [руб.] (19)

где Зр – затраты на ручную обработку информации, руб.;

За – затраты на автоматизированную обработку информации, руб.

Эу = 11 817 604 – 382 457 = 11 435 147 руб.

Экономический эффект (экономия) от использования программного продукта за год:

Эг = Эу – 0.135 × Сп, [руб.] (20)

где Сп – полная себестоимость программного продукта, руб.

Эг = 11 435 147 – (0.135 × 454 421,96) ≈ 11 373 800

Срок окупаемости программного продукта, который рассчитывается по формуле:

Т = Сп / Эг, год. (мес.) (21)

где Сп – полная себестоимость программного продукта, руб.

Эг - экономический эффект (экономия) от использования программного продукта, руб.

Т = 454 421,96 / 11 373 800 = 0,04 года

## Экономические показатели разработки программного продукта

1. Экономические показатели разработки программного продукта

|  |  |  |
| --- | --- | --- |
| Наименование | Единица измерения | Значения показателя |
| Длительность работ при разработке программного продукта | дни | 56 |
| Полная себестоимость программного продукта | руб. | 454 421,96 |
| Экономия от замены ручной обработки информации на автоматизированную | руб. | 11 435 147 |
| Экономический эффект (экономия) от использования программного продукта | руб. | 11 373 800 |
| Срок окупаемости программного продукта | год | 0,04 |

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была разработана система управления задачами, ориентированная на внутреннее использование в организации ООО «НПЦ «БизнесАвтоматика». Система предоставляет пользователю функциональный интерфейс на основе Kanban-доски, что делает её интуитивно понятной и простой в освоении.

На этапе анализа были изучены несколько самых популярных существующих решений, в результате чего были выявлены ключевые требования к системе, которые легли в основу разрабатываемого решения.

Было спроектировано и реализовано базовое ядро системы:

* реализован CRUD-функционал для задач;
* добавлена возможность работы с метками и статусами;
* поддержана история действий с задачами;
* внедрена возможность прикреплять файлы к задачам;
* созданы комментарии к задачам и связь между ними;
* разработан интерфейс с поддержкой drag-and-drop, позволяющий легко управлять задачами на доске.

Архитектура системы построена по принципам MVC, что обеспечивает модульность, гибкость и возможность дальнейшего расширения. База данных спроектирована таким образом, чтобы обеспечить масштабируемость и поддержку новых типов данных и связей.

В рамках проекта были решены следующие задачи:

* проведён анализ существующих решений и определены ключевые требования к внутренней системе управления задачами;
* выбраны технологии реализации и определена архитектура модулей;
* разработан прототип системы с базовым функционалом: создание, редактирование и просмотр задач; управление метками; комментирование; добавление вложений; связывание задач;
* реализованы механизмы логирования действий и истории изменений;
* протестирована работоспособность системы и проведено технико-экономическое обоснование.

Результаты экономического расчёта показали экономическую целесообразность внедрения разработанного программного продукта. Полная себестоимость составила 454 422 руб., а срок окупаемости — менее одного месяца, что свидетельствует о высокой эффективности предложенного решения. При этом система остаётся открытой для расширения и доработки.

К перспективам дальнейшей разработки можно отнести:

* расширение прав доступа и ролевой модели;
* интеграция с внутренними корпоративными системами учёта сотрудников;
* добавление мобильной версии или отдельного клиента;
* улучшение фильтрации и поиска задач;
* добавление уведомлений и напоминаний;
* поддержка мультиязычности.

Таким образом, дипломный проект представляет собой полноценное веб-приложение, решающее актуальную задачу автоматизации управления задачами внутри проекта. Все поставленные цели достигнуты и задачи решены.