ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ КОНТРОЛЯ ВЫПОЛНЕНИЯ ЗАДАЧ СОТРУДНИКАМИ ПРЕДПРИЯТИЯ

Выпускная квалификационная работа
обучающегося по направлению подготовки 09.03.02 Информационные системы и технологии
очной формы обучения, группы 07001407
Синюкова Дениса Владимировича

Научный руководитель
профессор
Щеглов И.Г.

БЕЛГОРОД 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современная экономика обладает ресурсами в большом количестве. И рациональное управление ими, определяет характер развития экономики, масштаб, темп и структуру ее развития.

Глобализация, большой прогресс в области компьютерных технологий, науки и техники в целом, изменил характер бизнес-среды. В такой ситуации и с быстро растущей сложностью бизнес-среды существует настоятельная необходимость активного участия правительства в создании изменений, а также быстрой адаптация к постоянно новым условиям. Учитывая, что творчество и приспособляемость организации в решающей степени зависят от навыков, знаний сотрудников, эти аспекты становятся наиболее важными не только в частных, но и в общественных организациях.

В сегодняшнем конкурентном мире организации осознают, что человеческие ресурсы являются важным активом, который может обеспечить устойчивое конкурентоспособное преимущество. Сотрудники по управлению персоналом должны организовывать и управлять своими ресурсами, чтобы повысить их ценность и производительность. Кадровые ресурсы, их знания и навыки могут стать основой для развития организационных конкурентных преимуществ на современном рынке. Важность наличия конкурентоспособного человеческого ресурса является синонимом успеха любого предприятия.

Эффективное внедрение инструментов и процедур управления временем в проектной команде чрезвычайно важно. Возможность точно фиксировать время, затраченное на проекты и мероприятия, имеет решающее значение, когда дело доходит до принятия бизнес-решений.

Использование инструментов управления, тесно связанных с выполнением задач, обеспечивает прозрачность и лучшие условия для управления. Естественная связь между инструментом контроля задач и проектной деятельностью облегчает последующую деятельность, а близость к членам проекта способствует сотрудничеству команды.

Цель работы – повышение эффективности процесса контроля выполнения задач на предприятии «Кроникс Микросистемс», за счет разработки автоматизации информационной системы.

Для реализации поставленной цели необходимо решить следующие задачи:

– изучить теоретические и методологические основы формирования и развития системы контроля выполнения задач на современных предприятиях;

– выбор метода и средств контроля выполнения задач сотрудниками предприятия;

– разработать систему контроля выполнения задач сотрудниками;

– провести тестирование разработанной системы.

Объект исследования: система контроля выполнения задач сотрудниками предприятия «Кроникс Микросистемс».

Предмет исследования: методы и средства контроля выполнения задач на предприятии «Кроникс Микросистемс».

Структура работы состоит из введения, трёх глав, заключения, списка использованных источников и приложений, изложена на 60 страницах компьютерного текста, содержит 7 таблиц и 31 рисунков. Список использованных источников насчитывает 31 источник, к работе приложено 4 приложения.

1 Аналитическая часть

1.1 Технико-экономическая характеристика предприятия

1.1.1 Характеристика предприятия

Открытое акционерное общество «Кроникс Микросистемс» сформировано по всем требованиям законодательства Российской Федерации и действует на основании Федерального закона «Об обществах с ограниченной ответственностью», иного законодательства Российской Федерации, а также настоящего Устава.

Юридический адрес (он же фактический адрес): 302028, Орловская область, город Орёл, улица Максима Горького, дом 17, помещение 52; ИНН 5753065278, КПП 575301001, ОГРН 1165749055075.

Предприятие имеет филиалы в Воронеже и Белгороде.

Фактический адрес: Россия, 308031, Белгородская обл., г. Белгород, Гражданский проспект, 36.

Общество является коммерческой организацией. Его деятельность направлена на удовлетворение общественных потребностей и извлечение прибыли.

Основной вид деятельности – разработка компьютерного программного обеспечения.

Дополнительные виды деятельности:

– деятельность рекламных агентств;

– деятельность, связанная с использованием вычислительной техники и информационных технологий;

– деятельность консультативная и работы в области компьютерных технологий;

– деятельность по обработке данных, предоставление услуг по размещению информации и связанная с этим деятельность;

– деятельность по созданию и использованию баз данных и информационных ресурсов;

– ремонт компьютеров и периферийного компьютерного оборудования.

Основной целью создания общества является удовлетворение общественных потребностей и извлечение прибыли.

Общество имеет круглую печать, содержащую его полное фирменное наименование на русском языке и указание на место нахождения общества. Печать общества содержит фирменное наименование общества на английском языке. Общество имеет штампы и бланки со своим фирменным наименованием, собственную эмблему, а также зарегистрированный в установленном порядке товарный знак и другие средства индивидуализации.

Организационная структура предприятия показана на рисунке 1.1.

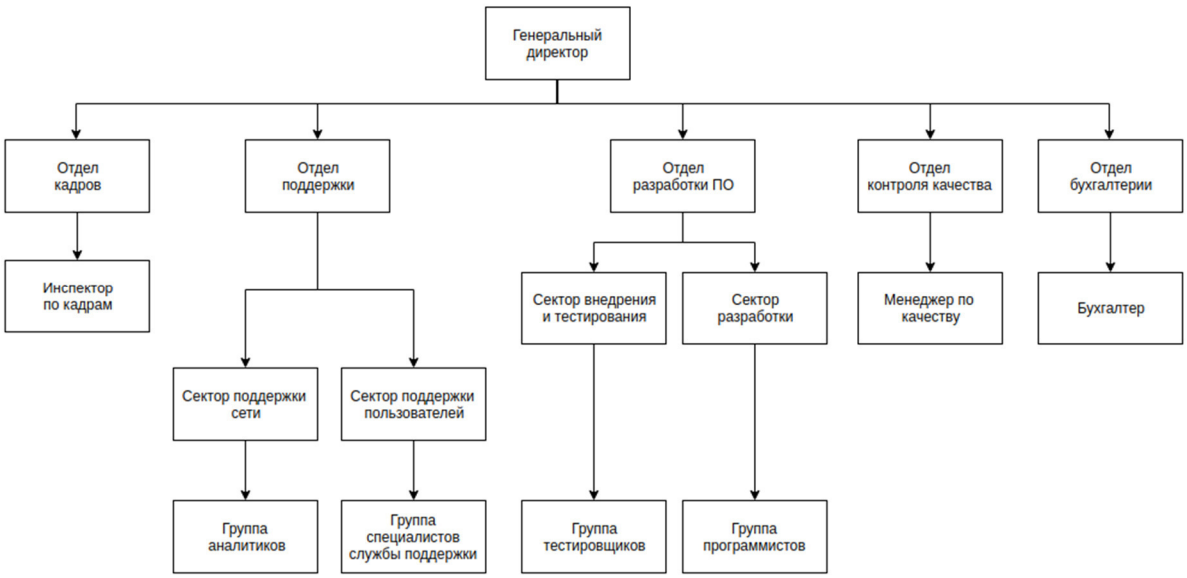


Рисунок 1.1 – Организационная структура предприятия

Для осуществления своих целей общество осуществляет следующие основные виды деятельности:

– web-разработка и web-дизайн;

– разработка мобильных приложений;

– разработка настольных приложений;

– видео, анимация и 3D-моделирование.

1.1.2 Краткая характеристика подразделения и видов его деятельности

В ООО «Кроникс Микросистемс» используется функциональная структура управления предприятием. Высшим органом является генеральный директор. Он является центром управления всеми отделами предприятия.

Организация состоит из 5 отделов, за каждым из которых прикреплен один руководитель, отвечающий за контроль выполнения всех работ, а также разрешение всех проблем, связанных с отделом.

Функциональная организационная структура представляет собой иерархическую организационную структуру, в которой люди сгруппированы в соответствии с их областью специализации. Эти люди контролируются функциональным менеджером с опытом в той же области. Этот опыт помогает ему эффективно использовать навыки сотрудников, что в конечном итоге помогает организациям в достижении своих бизнес-целей.

В отделе разработки программного обеспечения имеется два сектора, направленные на разработку и тестирование программного продукта, в каждом работают более 20 сотрудников.

Отдел разработки программного обеспечения обеспечивает высокое качество систематизированных решений для бизнеса-процессов различных подразделений.

Проекты программного обеспечения, как правило, инициируются с целью создания ценного, высококачественного программного обеспечения. Работа отдела разработки ПО основывается согласно стандартным схемам разработки продуктов:

- анализ определение целей разработки;
- написание и утверждение технического задания;
- программирование и отладка системы;
- тестирование программного продукта;
- организация описаний и инструкций для конечных пользователей.

1.2 Экономическая сущность задачи

В секторе услуг значительную долю занимают расходы на персонал. Поэтому при оценке затрат на проект или операционную деятельность требуется организация контроля выполнения задач.

Каждое динамично развивающее предприятие, в скором времени, нуждается в средствах повышения эффективности. Повышение работоспособности одного сотрудника так и всего отдела в целом. Разработка программного обеспечения дает возможность повысить производительность труда и производственных процессов. Автоматизации процессов позволяет сократить количество рабочих мест. Благодаря разработке и интеграции программного обеспечения снижается необходимость в значительном количестве сотрудников, в следствии чего происходит уменьшение затрат на них. За счет полного контроля за сотрудниками организации можно определить эффективность работы каждого из них, исключить малоэффективных сотрудников, для набора новых кадров, с целью повышения производительности всего предприятия.

Задача автоматизации контроля относится к классу задач «Анализа деятельности предприятия» и необходима для определения текущего состояния и тенденции развития выбранного предприятия.

Автоматизация системы контроля выполнения задач сотрудников предприятия позволяет решить следующие задачи:

- сбережение финансовых и временных ресурсов;
- увеличение уровня рабочей дисциплины;
- расчет заработной платы согласно отработанному времени;
- упрощение процедуры заполнения отчетности;
- эффективное планирование каждого дня/недели;
- выделение ресурсов с учетом важности задачи;
- распределение нагрузки пропорционально времени;
- мониторинг задач и хода реализации проектов.

1.3 Обоснование необходимости и цели использования вычислительной техники для решения задачи

Информационная система выполняет три этапа обработки информации. Основной целью первого этапа является сбор и предоставление сведений для последующей обработки. Итогом считается составление документа. Цель второго этапа – это перемещение сведений на машинные носители и формирование информационной базы. Заключительный этап включает в себя операции обработки, накопления, корректировки данных и предоставление результатов.

В данный момент в организации не внедрено ни одной системы ведения контроля над задачами. Все действия по ведению отчетности, о проделанной работе, отдаются либо устной, либо ручным заполнением табеля ответственным работником подразделения. Отсутствие общей картины, по каждому из сотрудников, плохо отражается на эффективности работы организации.

В ходе рабочего процесса могут часто возникают вопросы с не правильным начислением заработной платы, несправедливого взыскания со стороны работодателя, а также вопрос эффективности работы. Для исключения подобных ситуаций необходимо вести учет и анализ деятельности всех сотрудников.

Автоматизация рабочих процессов позволит сократить ручные операции, увеличить прозрачность выполнения работ, даст оценку деятельности предприятия за определенный период времени.

Внедрение информационной системы позволит хранить и осуществлять поиск информации о сотрудниках и их личных данных и структуре компании.

Начальными данными для ведения учета являются предоставленные документы и отчеты сотрудников. При ручном учете, каждый сотрудник отдела разработки программного обеспечения выполняет одни и те же действия от руки в бумажном формате, что влечет за собой потерю времени и бумажного

носителя информации. В случае утери информации тратится дополнительное время на составление нового отчета, возможно с недостоверной информацией.

1.4 Цель и назначение автоматизированного варианта решения задачи

Развитие информационных технологий является основным источником будущего экономического прогресса.

Информационные технологии используются для автоматизации простых и рутинных задач, таких как обработка текстов, планирование и логистика. Таким образом, информационные технологии позволяют предприятиям работать эффективно и прибыльно.

В основу разработки указанной системы заложены основные принципы:

- масштабируемость;
- отказоустойчивость;
- надежность;
- гибкость;
- переносимость.

При увеличении нагрузок программная часть системы не должна требовать каких-либо доработок, либо переделок. А увеличение возможностей должно достигаться расширением аппаратной части.

В системе должна быть возможность для быстрого восстановления работоспособности в случае аварий либо сбоев, так же система должна обеспечивается надежностью программных средств: средств разработки, используемых технологий и СУБД. Система должна быть построена так чтобы ее можно было доработать без изменения старой структуры. Система должна иметь возможность работать на различных платформах, с разными СУБД.

Для организации архитектуры аппаратной платформы была использована архитектура клиент-сервер [4].

Клиент-серверная архитектура, архитектура компьютерной сети, в которой много клиентов (удаленных процессоров) запрашивают получить

ресурсы с централизованного сервера (хост-компьютер). Клиентские компьютеры предоставляют интерфейс, позволяющий пользователю компьютера запрашивать службы сервера и отображать результаты, возвращаемые сервером. Серверы ждут поступления запросов от клиентов, а затем отвечают на них. Многие клиенты могут получить доступ к серверу информации одновременно, и, в то же время, клиентский компьютер может выполнять другие задачи, такие как отправка электронной почты. Потому что оба клиента и сервера компьютеры считаются интеллектуальными устройствами, клиент-серверная модель полностью отличается от старой модели «ЭВМ», в которой централизованные ЭВМ выполняли все задачи, связанные «немыми» терминалами [4].

Изменения в функциях подразделения, связанных со сбором, обработкой и выдачей информации:

– возможность обработки и сохранения данных с помощью ручного ввода в программу, а также импортом из внешних источников (Microsoft Excel, LibreOffice);

– возможность создания резервных копий;

– возможность наглядного просмотра графика работ по каждому проекту, при помощи диаграммы Гантта;

– возможность создание задач с указанием необходимой информации.

Источниками поступления информации являются сотрудники, поэтому автоматизация никак не повлияла на данный этап. Периодичность поступления информации определяется степенью загруженности сотрудника проектами и задачами. На периодичность так же влияют навыки сотрудника, их отсутствие, влечет за собой вопросы тормозящие весь процесс разработки.

Однако автоматизация существенно влияет на временной период заполнения результирующей информации. Сотрудник затрачивает дополнительное время на заполнения ежедневных или еженедельных отчетов о затраченном времени на выполнение задачи или проектов, ответов на вопросы коллег по работе в течение рабочего дня. Но существенно сокращает время

руководителя на проверку поставленных задач и предоставляют подробную статистику работы как сотрудника, так и целого отдела, с учетом всех использованных ресурсов, с которыми происходила работа. Управляющий может включить эту информацию в отчетность организации, существенно сэкономив время на ее подготовку.

Применение ЭВМ, для достижения цели проекта, позволит решить следующие задачи:

– возможность заполнение справочников пользователей из внешних источников (Microsoft Excel), так же сохранится возможность ручного ввода данным в систему;
– разграничение прав доступа к различным функциям системы;
– возможность создания резервных копий;
– возможность иллюстрации плана, графика работ по проекту.

1.5 Анализ существующих разработок и обоснование выбора технологии проектирования

Ценность использования программного обеспечения для осуществления контроля выполнения задач и проектов давно установлена. Тем не менее, все больше и больше менеджеров понимают, что контроль над выполнением задач является еще одной важной особенностью, которая имеет потенциал для улучшения процесса планирования работы, более точной оценки, и биллинга клиентов с большой точностью.

Инструменты контроля выполнения задач, в том числе и проектов, позволяют лидерам команд и менеджерам легко отслеживать абсолютно все, не отягчая членов команды дополнительными процессами. Целый ряд программного обеспечения был создан, чтобы помочь сделать работу по запуску проектов проще и эффективнее, и каждое решение предлагает ряд функций и возможностей, которые могут помочь преобразовать повседневную жизнь сотрудников.

Из-за большого количества программных решений для контроля над задачами бывает трудно решить, какой из них идеально подходит для команды. Рассмотрим несколько примеров.

«Битрикс24» облачный сервис для управления и автоматизации отношений между клиентами. Применяется разным уровням бизнеса: малому, среднему, большому. Система автоматизирует бизнес-процессы, взаимодействия персонала предприятия между собой.

Среди множества преимуществ можно выделить ряд недостатков:

- высокая стоимость лицензии;
- плохая логистика интерфейса;
- плохая работа службы поддержки;
- плохая техническая реализация.

Microsoft Project Server 2016 – это локальное приложение, созданное для эффективного управления корпоративными проектами. Система позволяет создавать бизнес-проекты любой сложности и объема. Совместная работа в режиме онлайн дает возможность пользователям получают возможность редактировать и обсуждать важные вопросы в любом месте и в любое время.

Среди преимуществ можно выделить:

- хорошая техническая поддержка;
- простой и понятный интерфейс;
- богатые возможности по настройке в стиле формул Microsoft Excel.

В системе так же присутствуют недостатки:

- высокая стоимость лицензии;
- требуется опытный консультант для постановки и контроля процесса;

Oracle Primavera – система для управления проектами и задачами, программами и портфелями проектов, разработанное корпорацией Oracle и предназначено для задач связанных с разработкой календарных планов, планированием ресурсов и финансов по задачам, контроля хода исполнения проекта и анализе объёмов работ.

Преимущества Oracle Primavera:

– быстрая реализация проектов и полная доступность информации;

– гибкость управления;

– наличие возможности управление рисками.

Недостатки Oracle Primavera:

– продукт относится к высокой ценовой категории, решение могут себе позволить средний и крупный бизнес;

– небольшая популярность системы и ориентация на строительный бизнес.

Выводы по первому разделу.

Таким образом, для достижения цели работы был выбран вариант реализации, посредством разработки автоматизированной системы контроля ведения задач сотрудниками предприятия, который в полном объеме учитывает особенности процесса обработки информации в компании ООО «Кроникс Микросистемс».

Для разработки программного обеспечения был выбран метод нисходящего программирования, основанный на прохождение всего жизненного цикла для каждого разрабатываемого модуля.

2 Характеристика комплекса задач и обоснование необходимости автоматизации

2.1 Обоснование проектных решений по техническому обеспечению

Техническое обеспечение – это совокупность технических средств, с помощью которых информационная система осуществляет манипулирование данными (сбор, обработку, преобразование, хранение, передачу).

Основным средством вычислительной техники является компьютер.

Основные элементы технического обеспечения:

- комплекс технических средств;
- организационные формы использования технических средств;
- персонал, который работает на технических средствах;
- инструктивные материалы по использованию техники.

Под комплексом технических средств подразумевают набор взаимосвязанных и (или) автономных технических средств манипулирования информацией для осуществления задач автоматизации и взаимодействия между техническими средствами.

Комплекс технических средств используется с целью автоматизации систем управления и задает уровень их эффективности.

Основные формы организации технического обеспечения:

- централизованная;
- частично или полностью децентрализованная.

Централизованное техническое обеспечение основывается на применении вычислительных центров и больших ЭВМ.

Децентрализация технических средств подразумевает разработку информационных подсистем за персональными компьютерами.

Структура технического обеспечения представляет собой набор сетевых узлов, которых связываются через общую среду передачи данных. Под узлами

понимаются устройства в сети, которые имеют отдельный IP-адрес, их так же называют автоматизированные рабочие станции [1].

На рисунке 2.1 показана общая структура технического обеспечения.

Рисунок 2.1 – Структура технического обеспечения

В организации «Кроникс Микросистемс» все компьютеры объединены в единую локальную сеть с выделенным сервером.

Работа по локальной сеть обеспечивает:

– коллективную работу над ресурсами сервера;

– обмен ресурсами между пользователями, подключенными по локальной сети;

– совместное использование программ, совместное использование периферийных устройств;

– одновременную печать на сетевых принтерах.

Однако локальные сети имеют ряд недостатков:

– затраты на обучение и поддержку оборудования;

– затраты на обслуживание дополнительного персонала.

Для начала разработки и отладки работы был выбран локальный сервер, который представлен в виде персонального компьютера.

Конфигурация локального сервера показана в таблице 2.1.

Таблица 2.1 – Конфигурация локального сервера

Видеокарта	описание: VGA compatible controller продукт: 3rd Gen Core processor Graphics Controller производитель: Intel Corporation разрядность: 64 bits частота: 33MHz
Оперативная память	4GB
Винчестер	Hitachi HTS543232A7A384 500GB
Процессор	Intel(R) Core(TM) i3–3217U CPU @ 1.80GHz
Операционная система	Linux 4.13.0–41–generic #46~16.04.1–Ubuntu x86_64 x86_64 GNU/Linux
Версия PHP	PHP 7.2.5
Версия Nginx	nginx/1.10.3 (Ubuntu)
Версия PHPUnit	PHPUnit 7.1.5

Стабильная работа и грамотная настройка сервера позволит:

– предоставление максимально быстрого и простого взаимодействия между сотрудниками;

– наличие постоянного доступа к данным;

– надежную защиту информации от потери или несанкционированного доступа.

После настройки, создания и отладки информационной системы все файлы базы данных будут храниться на сервере, а сама система продолжит выполняться в многопользовательском режиме.

Для корректной работы с системой аппаратное обеспечение пользователей должно удовлетворять следующим требованиям, представленным в таблице 2.2.

Таблица 2.2 – Минимальные аппаратные требования для конечного пользователя

Операционная система	Linux, Windows, MacOS
Оперативная память	1 Гб и более
Свободное дисковое пространство	64 Гб и более (зависит от размера хранимых в системе документов)
Необходимый процессор	Intel Pentium E6600или более мощный
Периферия	Клавиатура, мышь, принтер
Сеть	Наличие подключения к Интернету
Веб–обозреватель	Microsoft Windows Internet Explorer 8.0 или более поздняя версия Google Chrome, FireFox

К техническим средствам предъявляются требования:

- надежность;
- высокая пропускную способность;
- относительно низкая стоимость;
- высокие эргономические характеристики.

2.2 Обоснование проектных решений по информационному обеспечению

Информационное обеспечение (ИО) – это методика обеспечения информацией и управления рисками, связанными с использованием, обработки, хранения и передачи информации или данных, а также систем и процессов, используемых для этих целей.

Основными требованиями, предъявляемыми к ИО, являются: целостность, доступность, полнота, отказоустойчивость и конфиденциальность данных пользователя. Для выполнения этих задач используются физические, технические и административные средства управления. Эти средства защиты

применяются для транзитных данных, как физических, так и электронных форм, а также данных в различных видах физических и электронных хранилищ.

Создание информационной модели экономической системы, отражающей ее состояние на всем временном промежутке, является основной задаче информационного обеспечения.

Структура информационного обеспечения включает в себя два комплекса:

– компоненты внемашинного ИО (информационные потоки, классификаторы технико-экономической информации и документация);

– компоненты внутримашинного ИО (информационные файлы, базы данных, хранилища данных, базы знаний).

Структура информационного обеспечения показана на рисунке 2.2.

Рисунок 2.2 – Структура информационного обеспечения

С целью эффективной обработки на ЭВМ, поиска и передачи по каналам связи, информацию нужно представить в цифровом виде. Для получения желаемого результата информацию нужно классифицировать, а затем формализовать с использованием классификатора.

Классификация – процесс распределения множества объектов на подмножество согласно их однообразию либо отличию в согласовании с

общепринятыми способами. Классификация отмечает логические взаимосвязи между объектами классов. Под объектом понимается любой предмет, процесс, явление материального или нематериального свойства.

Система классификации позволяет выделить общие свойства для класса объектов.

В состав информационного обеспечения должны входить:

– потоки входной информации, к которым относятся сведения о задаче выполнения, количество затраченного времени, информация об исполнителе и организаторе, сроки выполнения;

– потоки выходной информации, к которым можно отнести сведения об общем количестве затраченного времени и отчеты эффективности работы.

В состав классификаторов входят следующие:

– справочник «Пользователи»;

– справочник «Проекты»;

– справочник «Задачи».

В список первичных документов входят:

– требования к проекту.

В результате должны формироваться следующие документы:

– отчет для руководителя.

Для ввода и вывода информации используются экранные формы, которые проектируются и создаются до этапа внедрения системы и написания технического задания.

2.3 Обоснование проектных решений по программному обеспечению

Под программным обеспечением (ПО) понимается пакет программ обрабатывающей вычислительной машиной.

На данный момент существует два основных типа программного обеспечения: системное (общее) и прикладное (специальное), выполняющие

разный функции. Набор программ управляющие компонентами вычислительной системы называется системным обеспечением.

Под прикладным обеспечением понимается программы, ориентированные на взаимодействие с пользователями и выполняющие их задачи.

Оба вида программного обеспечения могут быть представлены в виде диаграммы, которая показана на рисунке 2.3. Системное программное обеспечение гарантирует и осуществляет, и контролирует доступа к аппаратному обеспечиванию компьютера. Прикладное программное обеспечение осуществляет взаимодействие с аппаратными компонентами через системное. Конечные пользователи в основном работают с прикладным программным обеспечением.

Рисунок 2.3 – Диаграмма программного обеспечения

Системное ПО включает в себя:

а) операционная система;

1) распределяет ресурсы компьютера;
2) планирует использование ресурсов;
3) контролирует работу компьютера;

б) языковые трансляторы;

1) интерпретатор;

2) компилятор;

в) утилиты;

1) обслуживание компьютера и периферийных устройств.

Прикладное ПО включает в себя различные языки программирования.

Практические все информационные технологии, в этих областях, выделяют три класса программных продуктов: системное программное обеспечение, программное обеспечение и прикладное программное обеспечение, однако различие является произвольным и часто размытым. Схема классов программных продуктов показана на рисунке 2.4.

Рисунок 2.4 – Классы программных продуктов

Описание классов программных продуктов:

– системное программное обеспечение;

– прикладное программное обеспечение;

– инструментальное программное обеспечение.

Системное программное обеспечение помогает запустить компьютерное оборудование и компьютерную систему. Оно включает в себя операционные системы, драйвера устройств, серверы, оконные системы, утилиты и многое другое [5].

Цель системного программного обеспечения – максимально изолировать прикладного программиста от деталей конкретного используемого компьютерного комплекса, особенно памяти и других аппаратных возможностей, а также таких вспомогательных устройств, как принтеры, считыватели, дисплеи, клавиатуры.

Прикладное программное обеспечение предлагает инструменты для осуществления помощи программисту в написании компьютерных программ для этого используются различные языки программирования. Инструменты включают в себя текстовые редакторы, компиляторы, интерпретаторы, компоновщики, отладчики.

Инструментальное программное обеспечение позволяет конечным пользователям выполнять одну или несколько конкретных задач. Оно часто приобретается отдельно от компьютерного оборудования. Иногда приложения поставляются в комплекте с компьютером, но это не меняет того факта, что они запускают независимые приложения. Типичные применения включают промышленную автоматизацию, программное обеспечение промышленного производства, коммерческое программное обеспечение, медицинское программное обеспечение, базы данных, и компьютерные игры.

Предприятия, вероятно, являются крупнейшими пользователями прикладного программного обеспечения, но почти каждая область человеческой деятельности в настоящее время использует ту или иную форму прикладного программного обеспечения [5].

Выбор операционной системы является одним из важнейших критериев для разработки технических средств. Операционная система манипулирует всеми ресурсами компьютера с первых секунд его запуска. Каждый программный продукт использует ресурсы, предоставляемыми операционной системой.

На рабочих местах предприятия чаще всего применяются операционные системы с архитектурой x64:

– операционные системы семейства Windows, фирмы Microsoft (Windows 95/98/Me, Windows NT4.0/2000/XP/7/10);

– UNIX подобные операционные системы (Ubuntu) [5].

Для разработки автоматизированной системы выбор той или иной операционной системы не повлияет на функциональность по причине того, что при реализации алгоритмов программного приложения не требуется использования специфических функций операционной системы. Оба типа операционных систем дают возможность разрабатывать программный продукт без утраты производительность, по причине наличия программных сред для обоих видов операционных систем.

В качестве операционной среды для разработки была выбрана операционная система семейства Linux основанная на Debian GNU/Linux. Операционная система обладает удобным и простым интерфейсом и включает в себя огромное количество различного рода утилит [5]. Дистрибутив имеет все необходимые сервисы: серверы, облака, контейнеры, микросервисы.

База данных является жизненно важным элементом в компьютерной индустрии и правильный выбор системы управления баз данных (СУБД) представляет собой непростую задачу, которая является важным этапом при разработке приложений. СУБД должна полностью удовлетворять как нынешним, так и предстоящим нуждам компании.

Оценка на основе выдвинутых требований к системе является наиболее простым подходом при выборе СУБД. При этом необходимо опираться на некие критерии отбора, которым должна соответствовать разрабатываемая система:

– гибкость системы;

– настраиваемость;

– условия лицензирования.

Наиболее распространенными в практике являются реляционные базы данных. Легкость реализации, независимость данных, простота использования являются основными причинами выбора данной классификации.

Реляционные системы баз данных реализуют реляционную модель для работы с данными. Реляционная модель формирует любую информацию, которая будет храниться, определяя их как связанные сущности с атрибутами в таблицах (т. е. схемах).

Этот тип систем управления базами данных требует, чтобы были определены структуры (таблица), чтобы содержать и работать с данными. В таблицах каждый столбец (атрибут) содержит информацию другого типа (тип данных). Каждая запись в базе данных с уникальным идентификатором, с ключи, преобразуется в строку, которая относится к таблице, с каждой строки последовательности атрибутов представлены в виде столбцов таблицы – все взаимосвязано, как это определено в реляционной модели [28].

Отношения можно рассматривать как математические множества, которые содержат ряд атрибутов, которые в совокупности представляют базу данных и хранящуюся информацию.

При определении таблицы для вставки записей, каждый элемент формирования записи (т. е. атрибут) должен соответствовать определенный тип данных (например, целое число, дата и т. д.). Различные реляционные системы управления базами данных реализуют различные типы данных, которые не всегда взаимозаменяемы.

В таблице 2.3 приведена сравнительная характеристика трех распространенных СУБД, которые занимают лидирующие позиции на рынке программного обеспечения по основным показателям.

Таблица 2.3 – Сравнение СУБД

Показатели	SQLite	MySQL	PostgreSQL
Поддерживаемые операционные системы	Windows, Linux, Unix, Mac	Windows, Linux, Unix, Mac	Windows, Linux, Unix, Mac
Условии лицензирования	Бесплатная лицензия	Коммерческая лицензия и GNU GPL	Лицензия BSD Open Source.
Модель базы данных	реляционная	реляционная	реляционная
Поддержка языков программирования	Basic, C, C#, C++, Delphi, Fortran, Java, JavaScript, Lisp, MatLab, Python, PHP	Ada, C, C#, C++, Delphi,Java, JavaScript, Perl, PHP, Python, Ruby	.Net C, C++, Delphi, Java, Perl, PHP, Python
Максимальный размер базы	32 ТБ	Нет ограничений	Нет ограничений
Графический интерфейс	да	да	да

Таким образом, для разработки информационной системы, наиболее приемлемыми СУБД являются SQLite и MySQL. SQLite применяется в качестве базы для тестирования.

SQLite – это библиотека, которая встраивается в приложение, которое использует. Как автономная база данных на основе файлов, SQLite предлагает удивительный набор инструментов для обработки всех видов данных с гораздо меньшим ограничением и легкостью по сравнению с размещенными, процессными (серверными) реляционными базами данных [29].

Когда приложение использует SQLite, интеграция работает с функциональными и прямыми вызовами к файлу, содержащему данные (т. е. базу данных SQLite) вместо того, чтобы общаться через интерфейс видов (т. е. порты, сокеты). Это делает SQLite чрезвычайно быстрым и эффективным, а также мощным благодаря базовой технологии библиотеки.

Преимущества SQLite:

– файловое хранение данных. Вся база данных состоит из одного файла на диске, что делает ее чрезвычайно переносимой;

– отлично подходит для разработки и тестирования. На этапе разработки большинства приложений крайне вероятно, что потребуется решение, способное масштабироваться для параллелизма. SQLite, с его богатой базой функций, может предложить больше, чем необходимо для разработки с простотой работы с одним файлом и связанной библиотекой на основе C.

Недостатки SQLite:

– отсутствует управление пользователями. Расширенные базы данных поставляются с поддержкой пользователей, т. е. управляемых соединений с установленными привилегиями доступа к базе данных и таблицам.

MySQL является самым популярным из всех крупномасштабных серверов баз данных. Это многофункциональный продукт с открытым исходным кодом, который обеспечивает работу многих веб-сайтов и приложений в интернете. Начало работы с MySQL относительно легкое, и разработчики имеют доступ к массивной информации о базе данных в интернете.

Преимущества MySQL:

– простота использования;

– безопасность;

– масштабированность;

– высокая скорость работы.

Отказ от некоторых стандартов позволяет MySQL работать эффективно, таким образом обеспечивая прирост скорости [17].

Недостатки MySQL:

– ограниченные возможности;

– слабое развитие.

Хотя MySQL все еще является техническим продуктом с открытым исходным кодом, есть жалобы относительно процесса разработки с момента его приобретения.

Следующим этапом стоит выбор языка программирования. Автоматизированная система будет являться web-приложением, поэтому необходим язык программирования, который даст возможность работы подсистемы в многопользовательском режиме без ошибок.

Согласно статистике использования технологий, для разработки web-приложений, можно построить диаграмму наиболее использованных технологий [30], которая показана на рисунке 2.5.

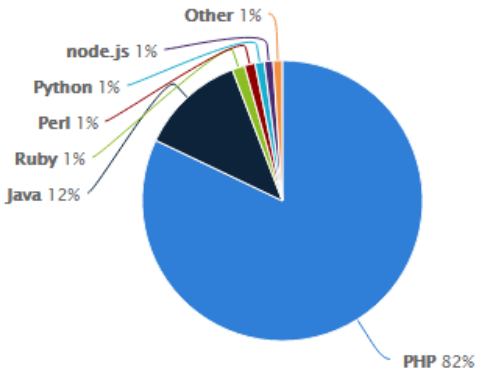


Рисунок 2.5 – Частота использования технологий для разработки приложений

PHP изначально создавался как встраиваемый язык для разработки динамических страниц. Поэтому в данном языке программирования изначально предусмотрены возможности для решения задач, связанных с web и он идеально подходит для выполнений поставленных задач.

Основные преимущества PHP:

– подключается к любым БД;

– свободный язык программирования и не требует лицензий на разработки;

– можно запускать как из командной строки, так и через планировщика cron что позволяет управлять рассылками и другими событиями;

– объектно-ориентированный язык [13];

– прост в изучении;

– свободная типизация данных;

– кроссплатформенный.

Недостатки языка программирования PHP:

– низкий порог вхождения;

– плохая защищенность при небрежной разработке.

2.4 Обоснование проектных решений по технологическому обеспечению

Данные являются информацией и могут включать в себя все, от буквенно-цифровых символов до дат, уравнений и мультимедии. Обработка данных является одним из этапов обработки информации и происходит после получения, ввода и проверки данных.

Технологическое обеспечение (EDP), также называемая автоматической обработкой данных (ADP) или обработкой информации, представляет собой автоматизированное использование компьютерных технологий для обработки данных, будь то при классификации, записи, обобщении или манипулировании ими [9].

Поскольку повреждение оборудования для обработки данных и данных может быть вызвано электрическими проблемами, изменениями температуры и влажности, а также магнитными помехами, их необходимо проводить специальный отбор.

Технологический процесс обработки дынных – это операции направленные на преобразование информации и отвечающие целям пользователя от начала возникновения информации до ее использования.

Этапы технологического процесса:

– сбор, регистрация, передача данных для дальнейшей обработки;

– перенос данных на машинные носители и первоначальное формирование информационной базы;

– накопление, сортировка, корректировки и обработка информации;

– контроль, выпуск и передача итоговой информации, размножение и хранение.

Использование массивов нормативно-справочной информации дает возможность увеличения скорости поиска, сортировки, фильтрации информации [23].

Методы построение моделей на основе диалогового режима обеспечивает гибкую связь между пользователями и ЭВМ. С этой целью было предпринято решение разработки системы с использованием интерактивного режима общения между пользователем и ЭВМ.

Диалоговый режим имеет ряд преимуществ:

- простота использования и работа с базой данных;
- обеспечение защиты при взломах;
- высокая эффективность при поиске, выдаче информации;
- быстрый переход от одной операции к другой.

2.5 Обоснование выбора программных средств

Для реализации АРМ, в разделе 2.3, был выбран объектно-ориентированный язык PHP, так как данный язык хорошо зарекомендовал себя в web-разработке, и среда программирования PhpStorm. Выбранная IDE была специально разработана под нужды работы с web-технологиями [13] и богатый функционал с большим количеством дополнительных модулей может обеспечить высокий уровень разработки.

Преимущества PhpStrom:

- возможность командной разработки;
- бесплатная лицензия для обучающихся в вузах;
- возможность подключения дополнительных плагинов;
- возможность быстрой навигации по проекту;
- авто дополнение кода;
- работа с базами данных.

Для реализации работы с данными системы, в разделе 2.3, были выбраны СУБД MySql и SQLite. SQLite хорошо подходит для тестирования подсистемы путем написания тестов к реализующему функционалу.

MySQL обеспечивает гибкость и быстродействие работы с данными. Работа с обеими СУБД обеспечит высокую производительность и качественную реализацию разрабатываемой подсистемы [20].

Для документирования и наглядного представления структуры базы данных было выбрано Case-средство Allfusion Data Modeler Erwin. Данный инструмент обладает следующими преимуществами [31]:

– поддерживается прямое (создание БД на основе модели) и обратное (генерация модели по имеющейся базе данных) проектирование для 20 типов СУБД;

– увеличивает производительность труда благодаря удобному интерфейсу и автоматизации рутинных процедур;

– поддерживает методологию структурного моделирования SADT и следующие нотации: DEF1x, IE, Dimensional (последняя – для проектирования хранилищ данных);

– поддерживает 20 различных СУБД: настольные, реляционные и специализированные СУБД, предназначенные для создания хранилищ данных;

– позволяет повторно использовать компоненты созданных ранее моделей, а также использовать наработки других разработчиков;

– возможна совместная работа группы проектировщиков с одними и теми же моделями;

– позволяет переносить структуру БД из СУБД одного типа СУБД в другой;

– позволяет документировать структуру БД.

Основной целью выбора корпоративного стандарта организационного проектирования является задание общего и обязательного к применению языка общения управленческого звена компании, разработчиков организационных и технологических процессов и исполнителей этих процессов. Частными

применениями таких стандартов является синтез требований к создаваемым системам, положений об организационных подразделениях, служебные инструкции и т.д.

AllFusion Process Modeler 7 [16], ранее Bpwin, было выбрано как основное средство для графического представления бизнес-процессов. Данный инструмент помогает, в случае необходимости, четко документировать важные аспекты [18] любых бизнес-процессов:

Данный инструмент обладает следующими преимуществами:

- поддержка нескольких нотаций;
- интуитивно-понятный графический интерфейс;
- анализ показателей затрат и производительности;
- организационные графики;
- интеграция процессов/данных.

Для проверки правильности работы модулей приложения был выбран быстро развивающий framework – PHPUnit. Вся техника для реализации приложения основана на технике TDD, основная идея, которой заключается в том, что сначала пишутся тесты и только после написания тестов пишется код приложения, который должен пройдет эти тесты.

Выводы по второму разделу.

В разделе были рассмотрены вопросы по выбору проектных решений по техническому, информационному, технологическому обеспечению, а также вопросы по выбору программных средств для разработки системы.

Также была выбрана архитектура приложения, сформирован и обоснован выбор технической базы как для серверной, так и для клиентской части, обоснован состав классификаторов, состав и содержание входных и выходных документов, состав и методы построения экранных форм.

3 Проектная часть

3.1 Информационное обеспечение задачи

3.1.1 Информационная модель и ее описание

Проектирование информационных систем тяжелая и ресурсозатратная работа, которая требует высококвалифицированных кадров.

Переход от кустарных к индустриальным способам создания ПО привел к потребности контролирования процесса разработки ПО, прогнозированию и обеспечению гарантий уменьшения стоимости разработки, срокам реализации и повешения качества конечного продукта [24]. Такой переход поспособствовал появлению совокупности инженерных методов с средств создания ПО, которые объединены общим названием «программная инженерия». Программная инженерия основывается на систематизацию и стандартизация процессов в течении жизненного цикла ПО.

На основе структурного подхода программная инженерия помогает систематизировать и стандартизировать весь процесс разработки ПО.

Суть структурного подхода к разработке информационных систем состоит в ее декомпозиции на функциональные блоки, которые в свою очередь разбиваются вниз по иерархии [27].

Последним уровнем декомпозиции будет тот уровень, который наглядно отображает работу всех процессов.

Средствами программного продукта Bpwin и методологии IDEF0 были смоделированы бизнес-процессы деятельности организации «Кроникс Микросистемс».

Для отображения назначения системы и ее взаимодействие с внешней средой построим контекстную диаграмму средствами методологии IDEF0. В каждой модели может быть только одна контекстная диаграмма [3]. После

описания основной функции выполняется функциональная декомпозиция, т. е. определяются функции, из которых состоит основная.

На рисунке 3.1 показана контекстная диаграмма «Контроль ведения задач».

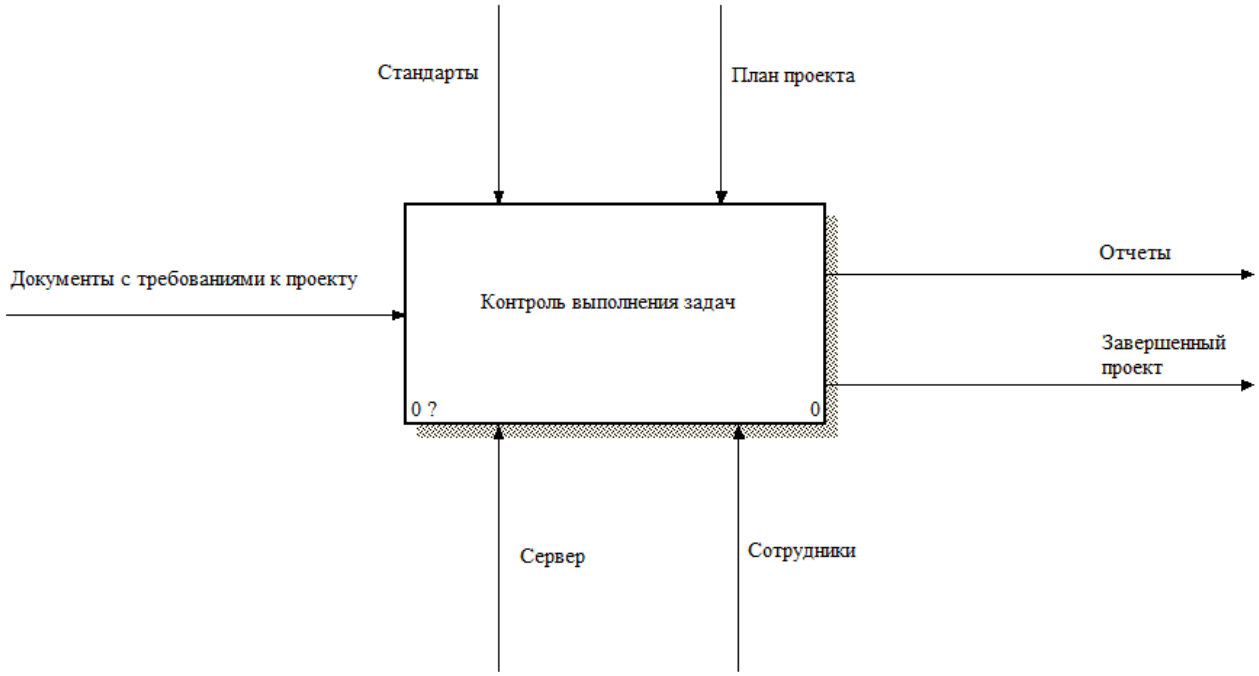


Рисунок 3.1 – Контекстная диаграмма «Контроль выполнения задач»

Контекстная диаграмма отображает обобщенный вид описания деятельности организации [8]. Связующими элементами выступают потоки, связывающие компанию с внешним миром.

Контекстная диаграмма процесса «Контроль ведения задач» описывает общий вид управления проектными задачами. На вход поступают документы, связанные с проектами. После ряда операций, выполненный согласно плану проекта и поставленным стандартам, руководитель может либо завершить проект, либо его приостановить. Сервер является главным источником обработки информации, которая поступает от сотрудников предприятия.

Проведя декомпозицию контекстной диаграммы, выделим процессы, которые участвуют в деятельности предприятия.

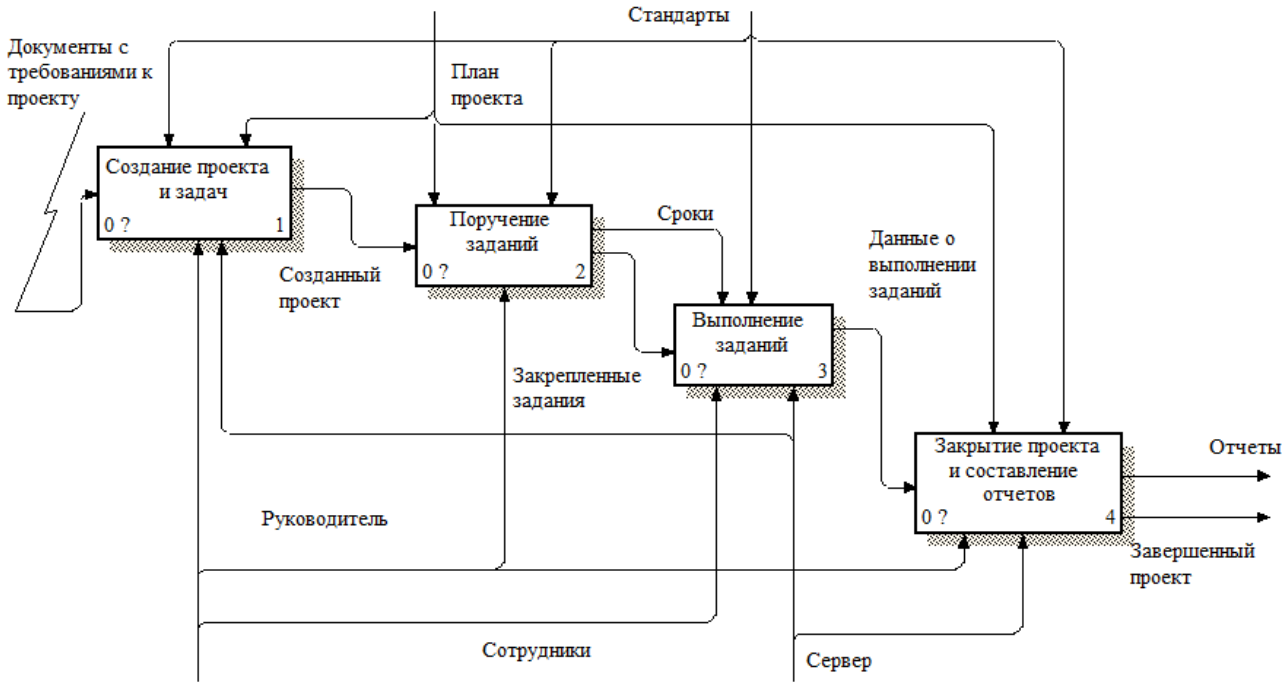


Рисунок 3.2 – Декомпозиция контекстной диаграммы

Входными документами для решения поставленной задачи являются документы с требованиями к проекту, которые формирует заказчик. Они определяют условия и порядок проведения работ проекта. Эти документы содержат информацию об основном назначение системы, описание первичных данных, целей и задач, сроков выполнения работ.

В процессе под номером 1, на рисунке 3.2, сотрудник с правами администратора, создает проект и список задач сотрудникам для выполнения.

Процесс под номером 2, на рисунке 3.2, описывает действия связанные с процессом поручения заданий. В которых сотрудникам прикрепляется ряд задач и делает прикрепленных сотрудников исполнителями.

Процесс под номером 3 на рисунке 3.2, описывает все действия, связанные поставленными задачами.

Процесс под номером 4, на рисунке 3.2, является завершающим. Заключается в составлении отчетности руководителем отдела, на основе предоставленных данных сотрудниками организации.

На рисунке 3.3 показана декомпозиция второго уровня «Создание проекта и задач».

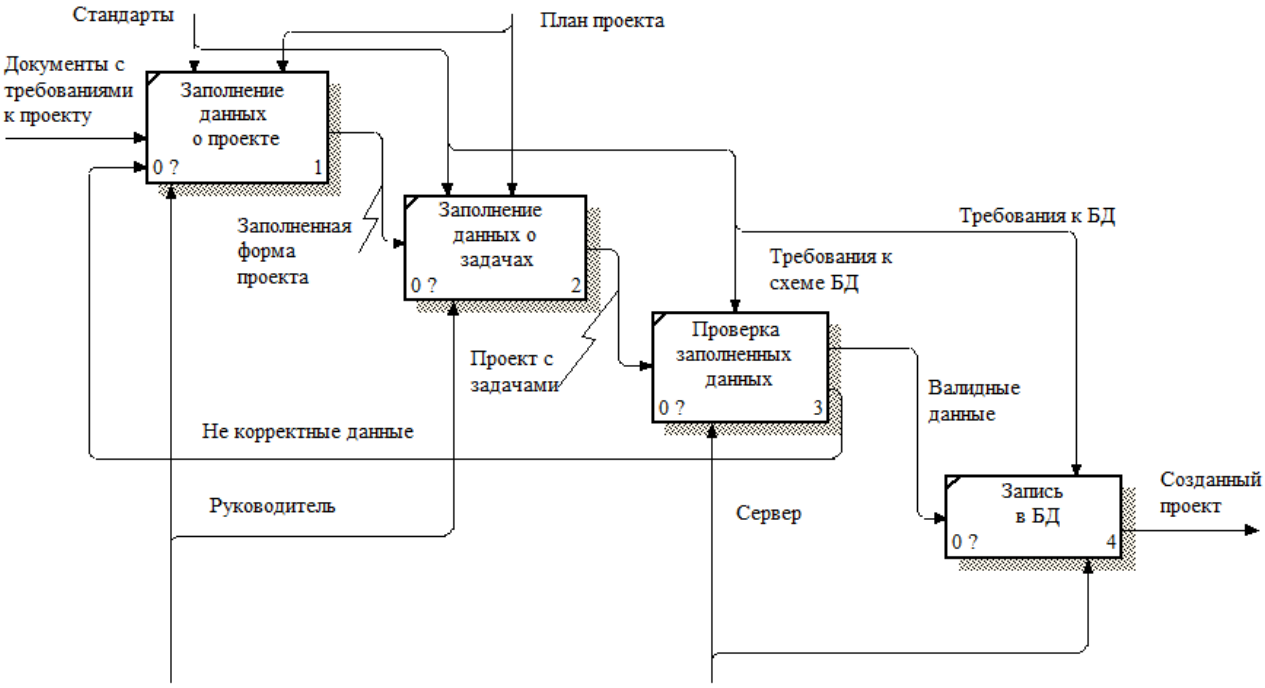


Рисунок 3.3 – Диаграмма декомпозиции второго уровня «Создание проекта и задач»

На основании входных документов формируется сам проект со всеми поставленными задачами. После заполнения всех данных о проекте и задачах, сервер, на основании алгоритмов проверки введенных данных, проверяет на корректность их заполнения.

В случае заполнения не всех данных или введение данных не по установленному стандарту, процесс заполнения переходит в начальную позицию. Если данные отвечают всем требованиям они записываются в базу данных и выходным элементов становится созданный проект, который содержит список задач.

После создания проекта необходимо назначить задачи для их выполнения. Задачи назначаются сотрудниками, у которого есть права на выполнения данных действий. Далее задачам необходимо назначить исполнителей.

Процесс поручения заданий показан на рисунке 3.4.

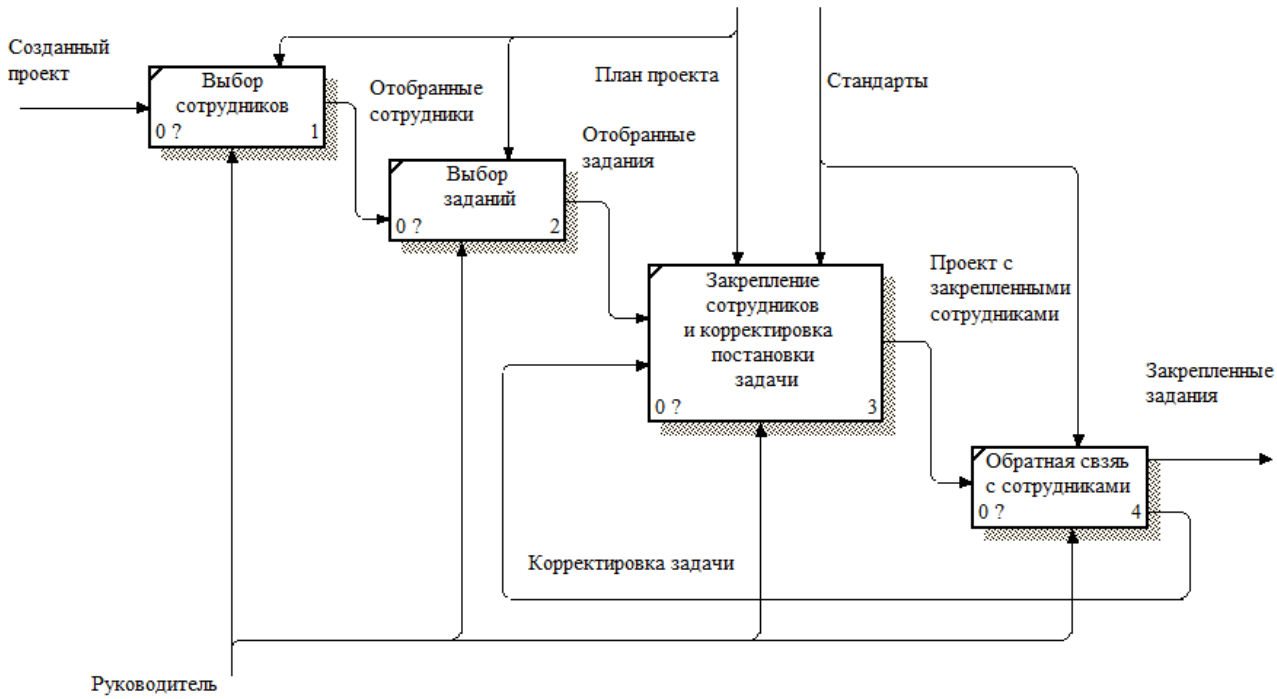


Рисунок 3.4 – Диаграмма декомпозиции второго уровня «Поручение заданий»

В начале необходимо выбрать список сотрудников, которые отвечают за реализацию задач, согласно своим навыкам и квалификации. Руководитель выбирает ряд сотрудников и задания, которые в последствии закрепляются за каждым работником отдела.

В случае неясности задачи исполнителем, руководитель разъясняет и делает корректировку задачи, в целях получения необходимого итогового результата.

После постановки задач сотрудникам и уточнение деталей, сотрудники приступают к выполнению поставленных указаний.

На рисунке 3.5 показана декомпозиция второго уровня процесса «Выполнение заданий».

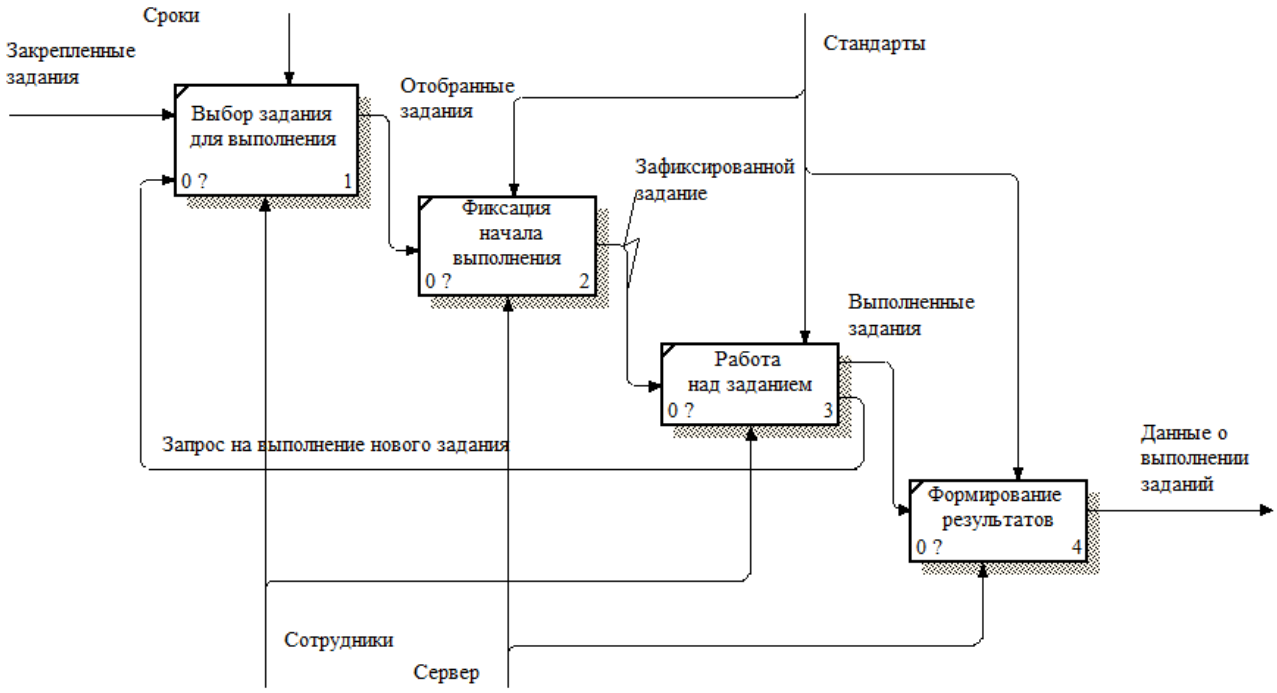


Рисунок 3.5 – Диаграмма декомпозиции второго уровня «Выполнение заданий»

После получения заданий сотруднику необходимо выбрать, из списка задач, ту задачу, которую он будет выполнять в данный момент.

Таймер начала выполнения стартует после выбора задачи и продолжается до момента его завершения. В случае приостановления, время останавливается до момента возобновления работы над задачей. После выполнения очередного задания сотрудник может переключиться на новую.

Когда все задачи выполнены, сервер, формирует данные для руководителя в электронном формате. Данные включают в себя общее время работы над задачами, временный интервалы выполнения, всю информации об исполнителе, ресурсы, связанные с задачами.

На рисунке 3.6 показана декомпозиция второго уровня процесса «Закрытие проекта и составление отчетов».

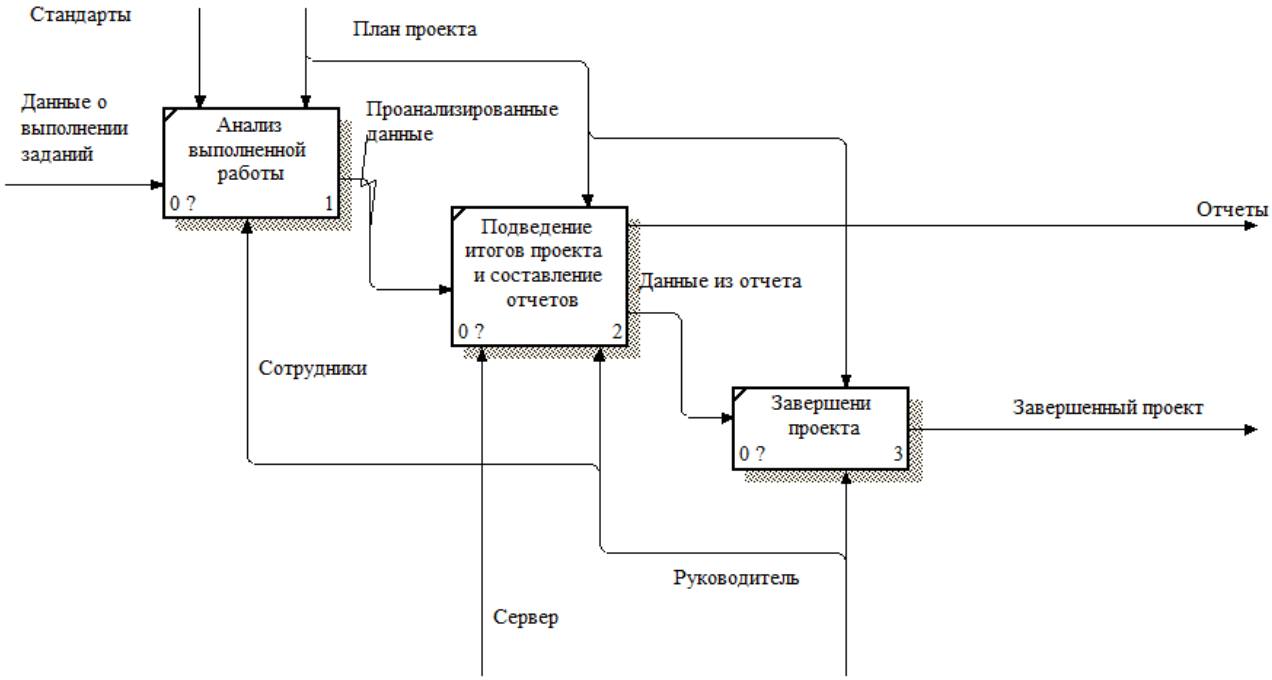


Рисунок 3.6 – Диаграмма декомпозиции второго уровня «Закрытие проекта и составление отчетов»

Финальным этапом является анализ данных о выполненных заданиях, которые пришли от сотрудника. Руководитель отдела вместе с аналитиками на основании проанализированных данных и плану проекта формирует данные для отчета и табеля рабочего времени. Данные отчета являются документами на основании, которых руководитель определяет этап завершенности проекта.

3.1.2 Используемые классификаторы и системы кодирования

Классификации предполагают расположение записей на основе некоторой иерархической структуры.

В многоаспектных системах классификации исходное множество рассматривается одновременно в разных аспектах.

Именно классификация является основой кодирования.

Кодирование – это процесс преобразования входной информации из одной системы знаков, в другую.

В разрабатываемой автоматизированной системе кодированию подлежат следующие классификаторы:

- классификатор сотрудника и групп сотрудников;
- классификатор проекта;
- классификатор задачи;
- классификатор колонок доски для задач.

В таблице 3.1 представлен перечень используемых классификаторов.

Таблица 3.1 – Перечень используемых классификаторов

Наименование кодируемого множества объектов	Рабочее наименование	Значность кода	Система кодирования	Система классификации	Вид классификатора
Ид сотрудника	Идс	4	порядковая	Иерархическая	Локальный
Ид проекта	Идп	4	порядковая	Иерархическая	Локальный
Ид задачи	Идз	6	порядковая	Иерархическая	Локальный
Ид доски	Идд	6	порядковая	Иерархическая	Локальный
Ид группы пользователей	Идг	3	порядковая	Иерархическая	Локальный

Для классификатора сотрудников используется порядковая система кодирования. Классификатор является локальным и состоит из 4 знаков. Структурная формула классификатора заказов: Ф1 = [XXXX].

Для классификатора проектов используется порядковая система кодирования. Классификатор является локальным и состоит из 4 знаков: Ф2 = [XXXX].

Для классификатора задач используется порядковая система кодирования. Классификатор является локальным и состоит из 6 знаков: Ф3 = [XXXXXX].

Для классификатора колонок доски для задач используется порядковая система кодирования. Классификатор является локальным и состоит из шести знаков: Ф4 = [XXXXXX].

Для классификатора групп пользователей используется порядковая система кодирования. Классификатор является локальным и состоит из 3 знаков. Ф4 = [XXX].

3.1.3 Характеристика первичных документов с нормативно-справочной и входной оперативной информацией

Вся информация, которая находится на различных носителях – относится к входной информации. Она может быть получена из первичных документов, машинных носителей или памяти ЭВМ.

Информация, которая не изменяется с течением времени и служит для многократно использования в течение длительного периода времени называется условно-постоянной. Целью создания такого вида информации является централизованное хранение данных, устранение дублей, повышение достоверности получаемых данных, а также уменьшение объема работ с ЭВМ.

Условно-постоянная информация представлена следующими справочниками:

– справочник «Настройки БД».

Справочник «Настройки БД» содержит начальные сведения работы с таблицами содержит следующие реквизиты:

– тип записи может иметь два варианта: таблица либо индекс;
– название таблицы или индекса;
– название таблицы;
– сортировка;
– sql скрипт для создания таблицы или индекса к таблице.

Для ввода данных используются экранные формы:

– «Новый проект»;
– «Новое задание»;
– «Новый пользователь»;
– «Новая колонка для доски»;
– «Новая категория».

3.2 Характеристика базы данных

Для обеспечения наглядности представления информации, которая будет храниться в базе данных применятся инфологическое моделирование. Инфологическую модель данных обычно строят по аналогии с естественным языком и в ее состав входят сущности и их атрибуты.

На основании анализа предметной области выделим сущности модели «сущность-связь» и изобразим их в виде диаграммы, которая показана на рисунке 3.7.

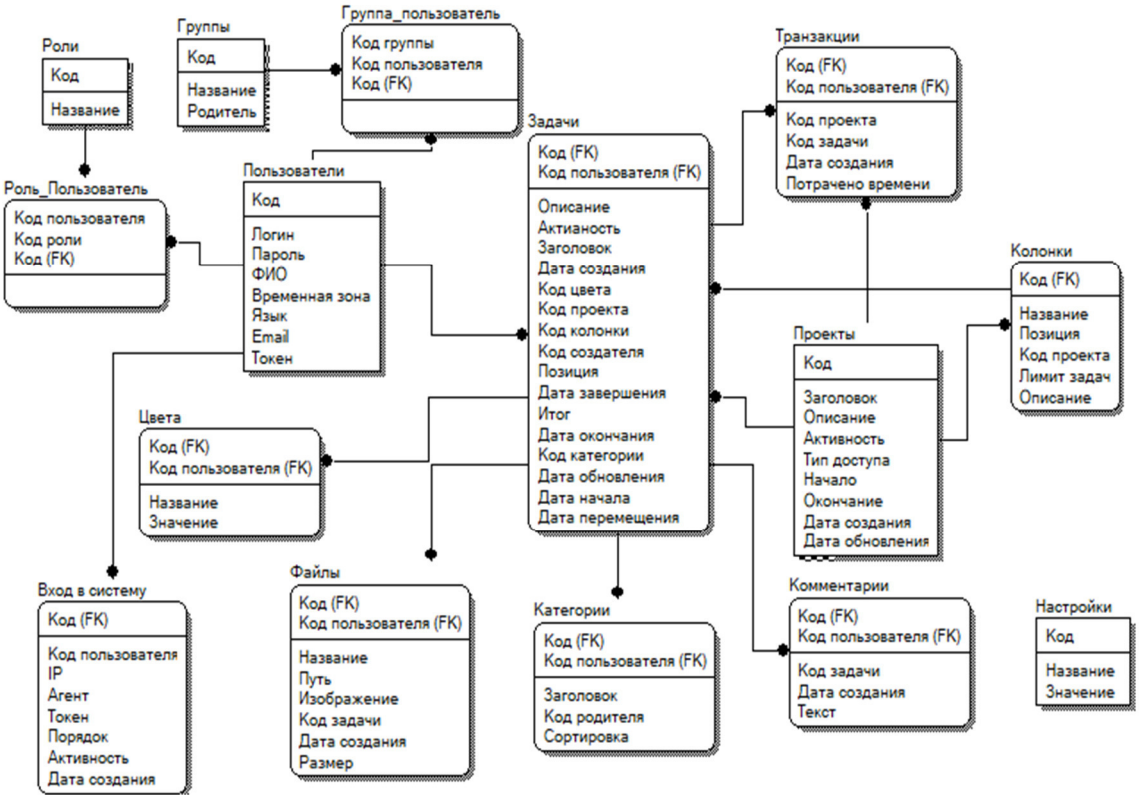


Рисунок 3.7 – Инфологическая модель базы данных

В качестве базы данных, для реализации системы, во 2 главе, была выбрана база данных MySql. В разрабатываемой системе для работы с таблицами, манипулирования записями, используется декларативный язык программирования SQL.

Даталогическая модель базы данных представлена в приложении А на рисунке А.1.

Рассмотрим основные таблицы из вышеприведенной модели.

Сущность «users», хранит в себе все данные о сотрудниках предприятия: данные для авторизации, временную зону. В таблице 3.2 показан полный список атрибутов данной сущности.

Таблица 3.2 – Сущность «users»

Название поля	Тип	Разрядность	NULL	По умолчанию	PK	FK	Ссылка
id	integer	10	Нет		Да		
username	varchar	255	Нет				
password	varchar	255	Нет				
name	varchar	255	Нет				
email	varchar	255	Нет				
timezone	varchar	255	Нет				
language	varchar	255	Нет				
token	varchar	255	Нет				

Сотрудник тесно связан с сущностью «roles», она хранит в себе название всех ролей. Список ролей для каждого сотрудника хранится в смежной сущности «user_role», которая имеет только два поя: код роли и код сотрудника.

Сущность «tasks», хранит в себе все данные о задачах в проектах, как для отдельного сотрудника, так и для групп. В таблице 3.3 показан полный список атрибутов данной сущности.

Таблица 3.3 – Сущность «tasks»

Название поля	Тип	Разрядность	NULL	По умолчанию	PK	FK	Ссылка
id	integer	10	Нет	–	Да	–	–
title	varchar	255	Нет	–	–	–	–
description	integer	1	Нет	–	–	–	–
date_creation	integer	10	Нет	–	–	–	–
color_id	integer	10	Нет	–	–	Да	colors
project_id	integer	10	Нет	–	–	Да	projects
column_id	integer	10	Нет	–	–	Да	columns
owner_id	integer	10	Нет	–	–	Да	users
position	integer	3	Да	0	–	–	–
is_active	integer	1	Да	1	–	–	–
date_completed	timestamp	–	Да	–	–	–	–
score	integer	10	Нет	–	–	–	–
date_due	timestamp	–	Да	–	–	–	–
category_id	integer	10	Нет	–	–	Да	categories
date_modification	timestamp	–	Да	–	–	–	–
date_started	timestamp	–	Нет	–	–	–	–
date_moved	timestamp	–	Нет	–	–	–	–

Сущность «settings» содержит в себе глобальные настройки для всего приложения и значения по умолчанию.

Сущность «projects», хранит в себе все данные о проектах предприятия.

В таблице 3.4 показан полный список атрибутов данной сущности.

Таблица 3.4 – Сущность «projects»

Название поля	Тип	Разрядность	NULL	По умолчанию	PK	FK	Ссылка
id	integer	10	Нет	–	Да	–	–
title	varchar	255	Нет	–	–	–	–
is_active	integer	1	Нет	–	–	–	–
type_access	integer	10	Нет	–	–	–	–
description	text	–	Нет	–	–	–	–
start	timestamp	–	Нет	–	–	–	–
end	timestamp	–	Нет	–	–	–	–
created_at	timestamp	–	Нет	–	–	–	–
updated_at	timestamp	–	Да	–	–	–	–

3.3 Описание программного интерфейса автоматизированной системы

Основным инструментом, для разработки системы, была выбрана интегрированная среда разработки PhpStorm. Она включает библиотеки и готовые шаблоны для создания стартовой страницы приложений, а также визуальный интерфейс для работы с базой данных.

Проект был построен при помощи пакетного менеджера – composer, который был разработан специально для языка PHP, для решения задач по управлению зависимостями.

Проект построен по архитектуре MVC, это значит, что модификация любого компонента может осуществляться независимо друг от друга.

Интерфейс IDE и структура проекта показана на рисунке 3.8.

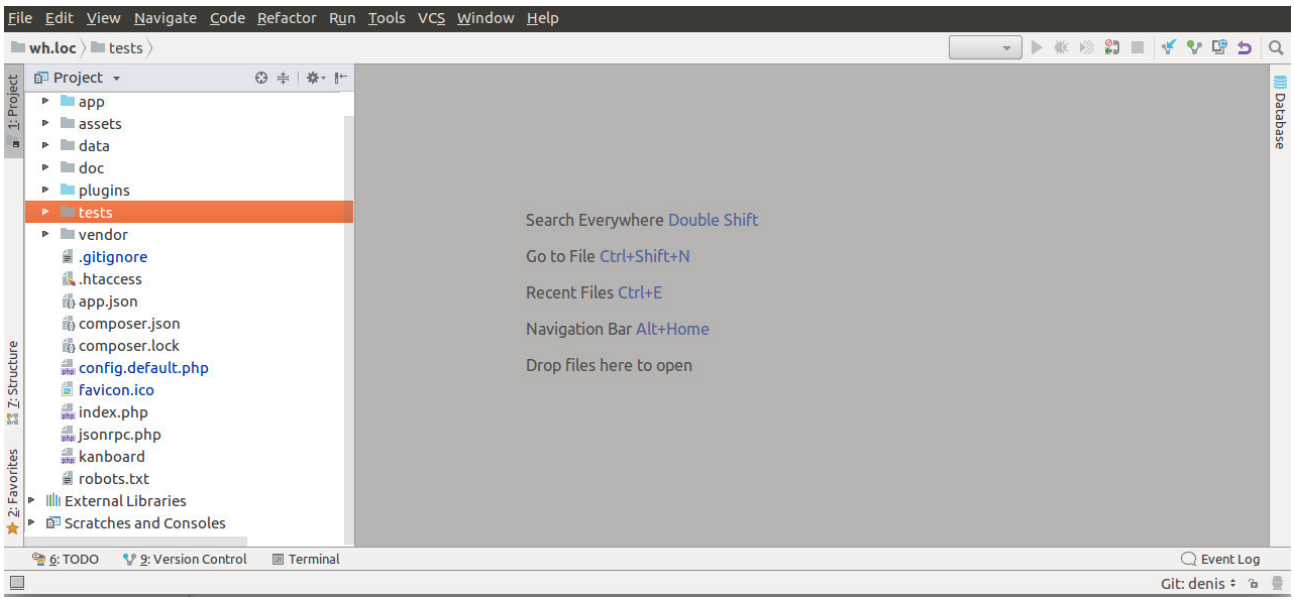


Рисунок 3.8 – Структура проекта

Директория «app» содержит классы с моделями, контроллерами и файлами для представления данных.

Директория «assets» содержит файлы стилей, шрифтов, скриптом, а также изображения для web-приложения.

В директории «data» будут хранятся файлы, который сотрудники смогут загружать для проектов или задач.

Директория «doc» содержит файлы интерактивных подсказок для работы с системой.

Директория «plugins» содержит дополнительные плагины, которые могут быть внедрены в систему.

Директория «tests» содержит написанные тесты для проверки корректной работы системы.

Директория «vendor» содержит зависимости, подключенные к проекту при помощи composer.

Файл config.default.php содержит набор констант, при помощи которых происходит подключение к БД, соединение с хостами и тд.

Для ведения учета за каждым сотрудником, необходимо чтобы каждый из них имел свой аккаунт, для этого была разработана форма авторизации, через которую, после введения учетных данных, сотрудник попадает в свой личный

кабинет. В приложении Б представлен класс-контроллер работы с авторизацией пользователей.

На рисунке 3.9 показана форма авторизации пользователей.

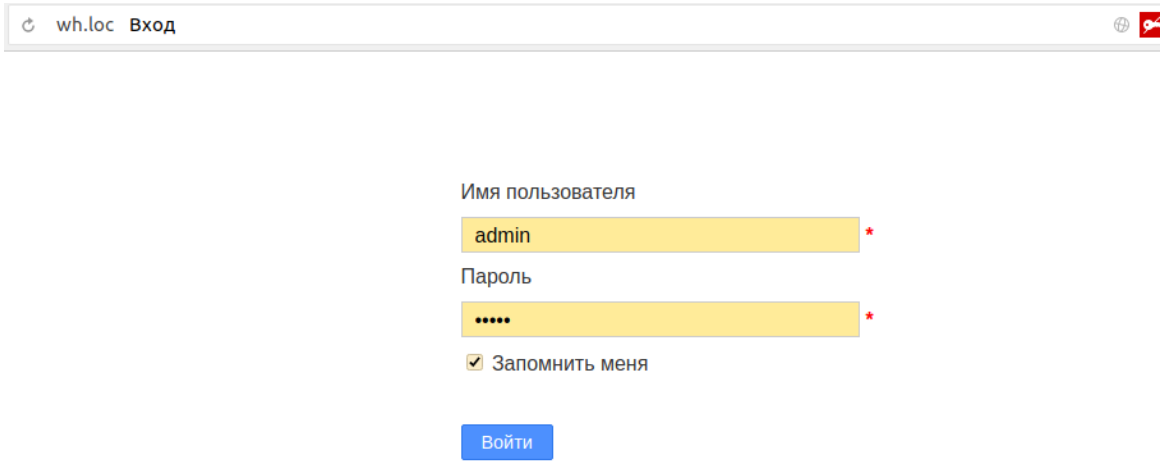


Рисунок 3.9 – Форма авторизации пользователей

Так как система является внутрикорпоративная, регистрация сотрудников, не работающих на предприятии, не предусмотрена. Для получения учетных данных, пользователю на электронную почту высылаются данные для входа в систему.

На рисунке 3.10 показана основная панель инструментов для администратора.

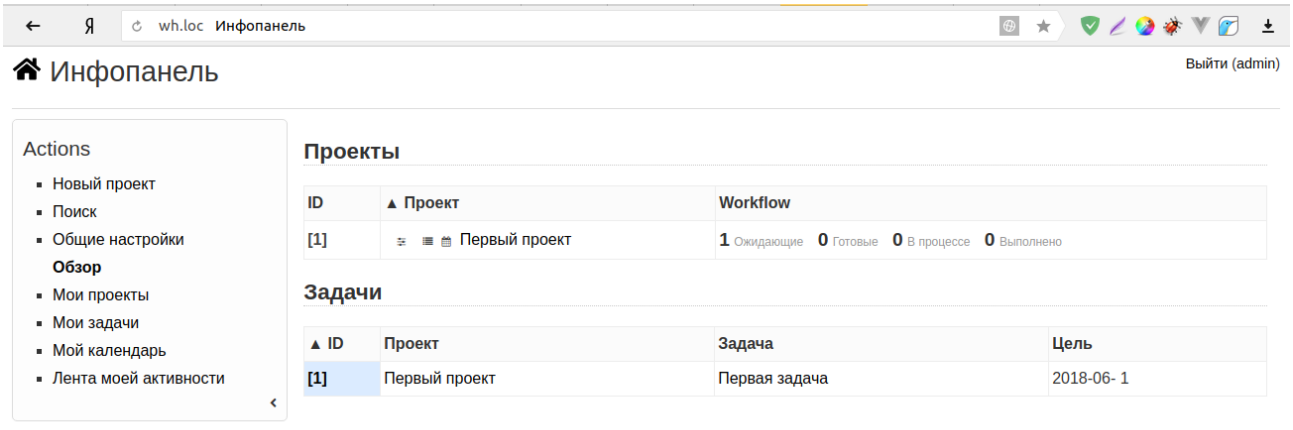


Рисунок 3.10 – Основная панель инструментов для администратора

Основным отличием интерфейса для администратора и пользователя со стандартными правами, является количество доступных инструментов для манипулирования информацией. Приложение В показывает класс-модель пользователя. На рисунке 3.11 показана основная панель для сотрудника предприятия со стандартными правами.

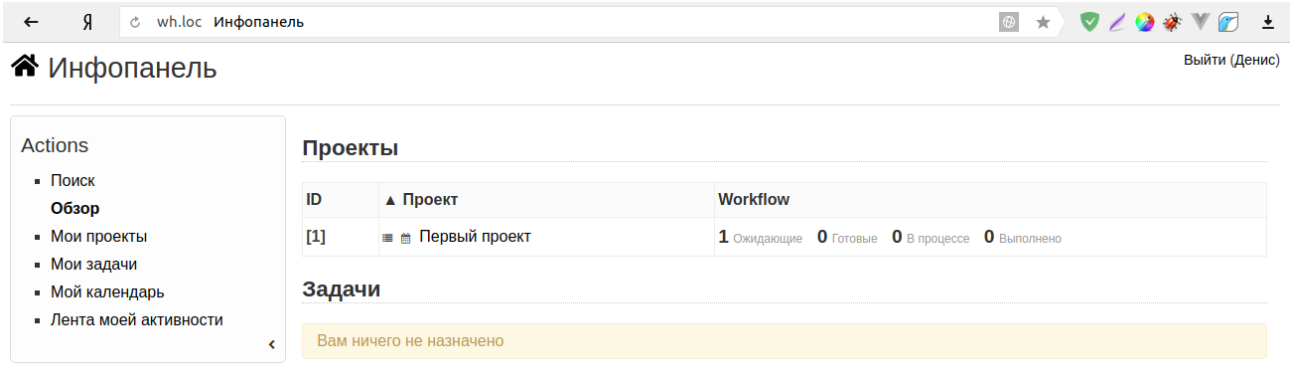


Рисунок 3.11 – Основная панель сотрудника с обычными правами

Созданием новых проектов и задач, а также прикреплением исполнителя, занимается сотрудник, наделенный правами администратора. На рисунке 3.12 показана форма создания нового проекта.

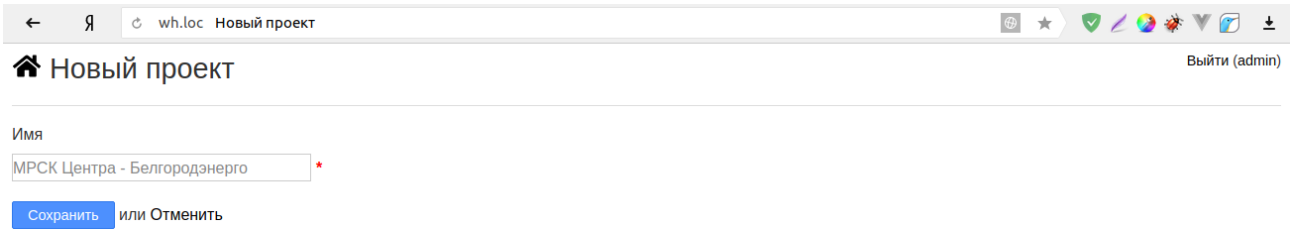


Рисунок 3.12 – Форма создание нового проекта

После создания проекта, можно настроить его под свои нужны. Настроить категорию, доступ, добавить описание, дату начала и завершения проекта и прочее. На рисунке 3.13 показана панель управления проектом.

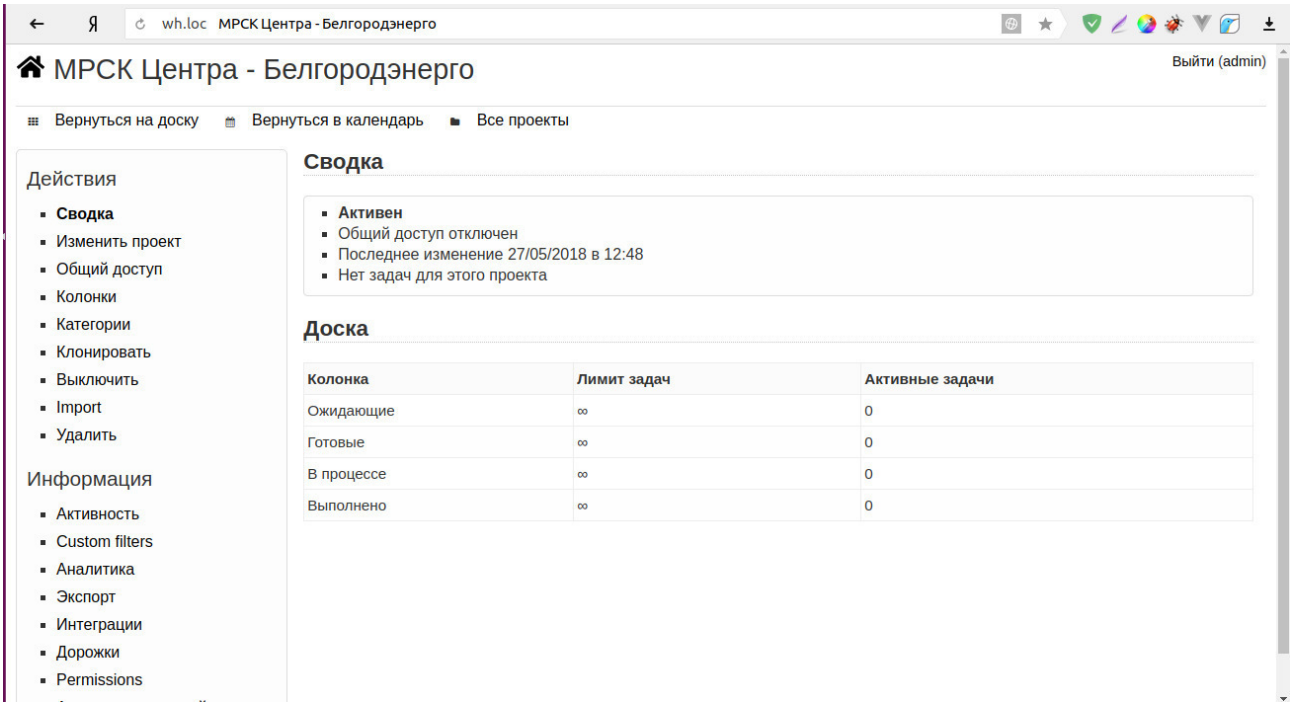


Рисунок 3.13 – Панель управления проектом

Для каждого проекта выделяется доска для задач, на которой сотрудники могут ставить задачи как для себя, так и для других работников.

На рисунке 3.14 показана доска для размещения задач проекта.

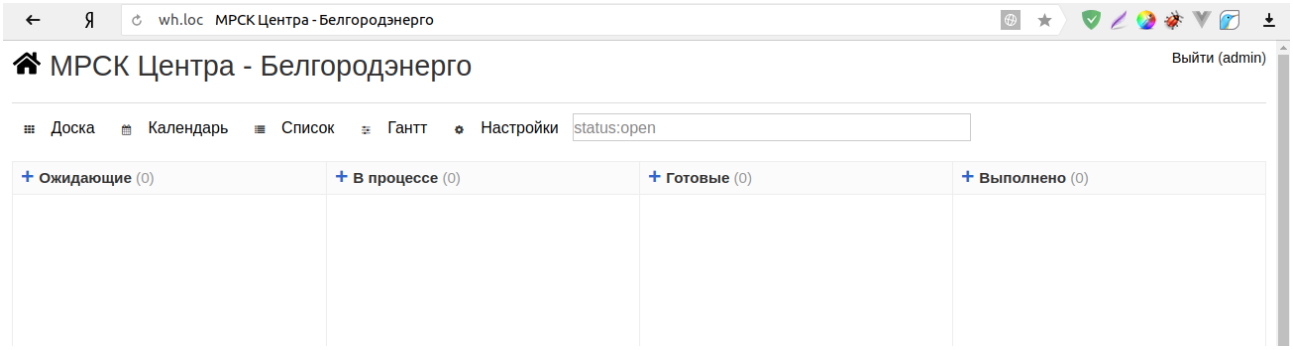


Рисунок 3.14 – Доска задач для проекта

По умолчанию доска имеет четыре колонки. В колонке «Ожидающие» находятся все доступные задачи для выполнения. Колонка «В процессе» отвечает за отображение задач, которые находятся на этапе разработки. В колонке «Готовые» отображаются задачи, которые были выполнены, но ожидают проверки руководителя. После оценки качества выполнения задача

переносится в колонку «Выполнено» после чего ее можно считать успешно завершенной.

Доска является гибким инструментом, который легко может быть настроен. При необходимости добавления нового этапа разработки, можно добавить ранее не существующую колонку, а также сменить порядок их отображения.

На рисунке 3.15 показана форма добавления новой колонки для текущего проекта.

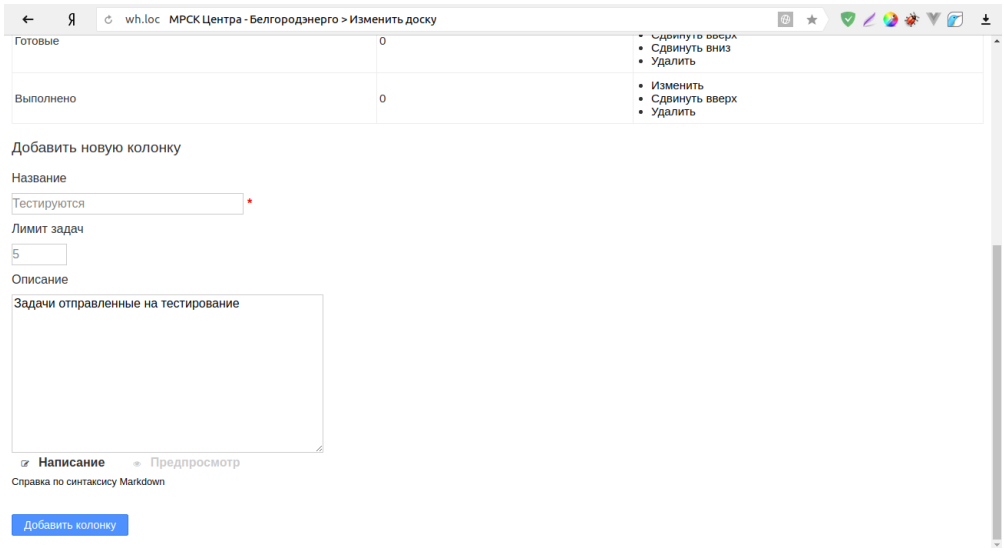


Рисунок 3.15 – Форма добавления новой колонки для проекта

Среди доступных полей, представленных на рисунке 3.15, можно выделить поле «Лимит задач». Данное поле отвечает за ограничение количества задач, находящихся в колонке, что позволяет отобразить этап разработки на котором, в потоке операций, возникает «затор», который оперативно устраняется.

Создание задач и прикрепления исполнителя, происходит при вызове модального окна на форме, которая показана на рисунке 3.16.

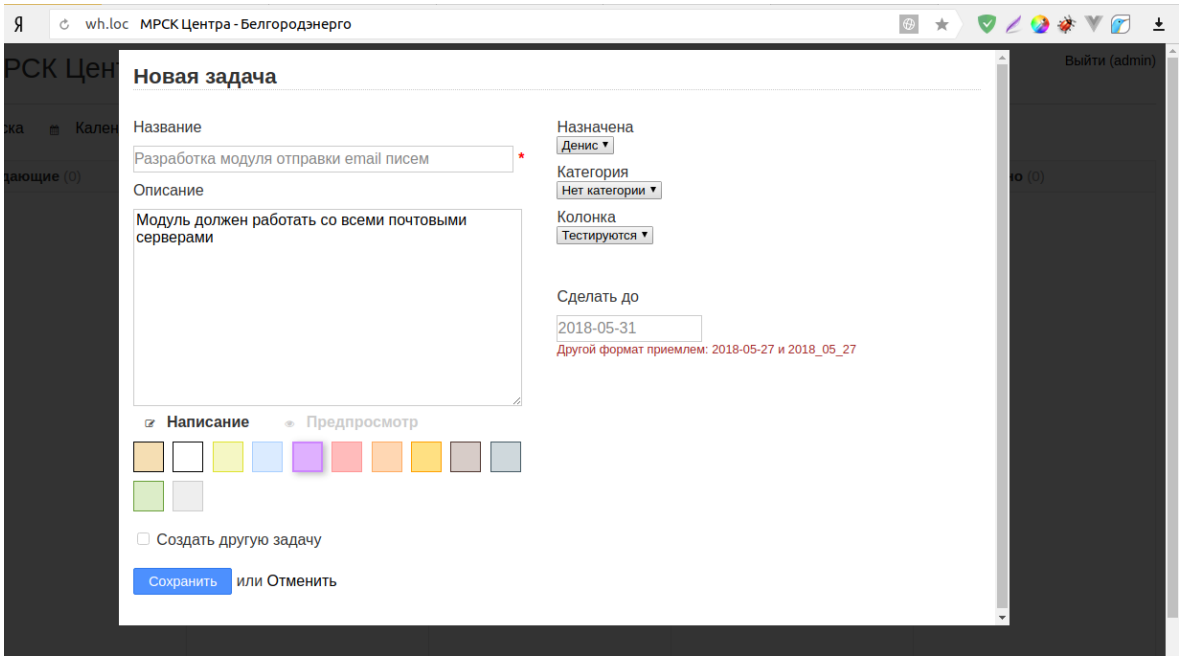


Рисунок 3.16 – Форма создания новой задачи

На рисунке 3.16 видно для кого была назначена задача и сроки ее выполнения. Выбор цвета визуально отделяет одну задачу от другой.

Вид доски после создание задачи показан на рисунке 3.17.

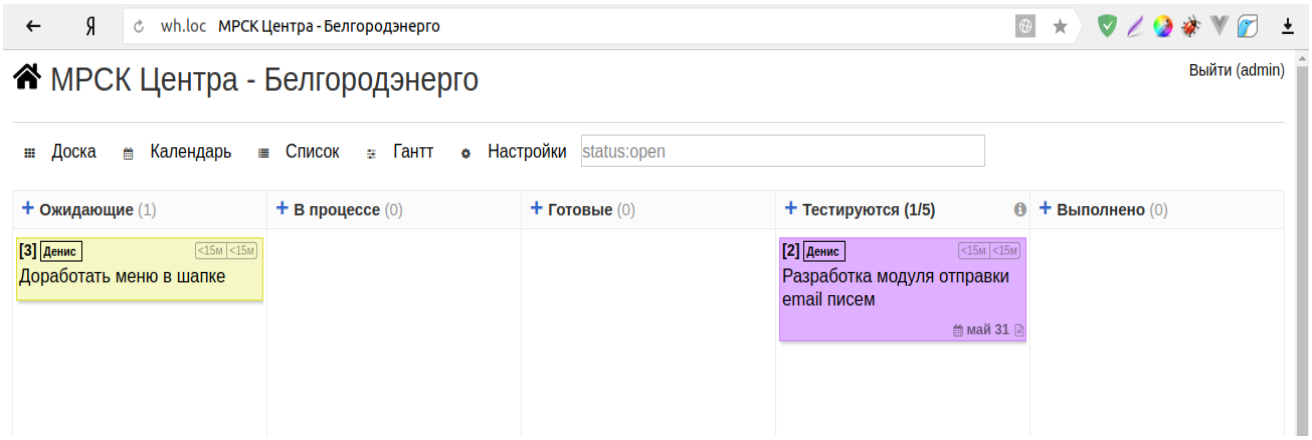


Рисунок 3.17 –Доска проекта с поставленными задачами

В режиме пользователя, сотрудник исполнитель, может добавлять комментарии к задаче, а также прикреплять файлы.

На рисунке 3.18 показана панель управления задачей.

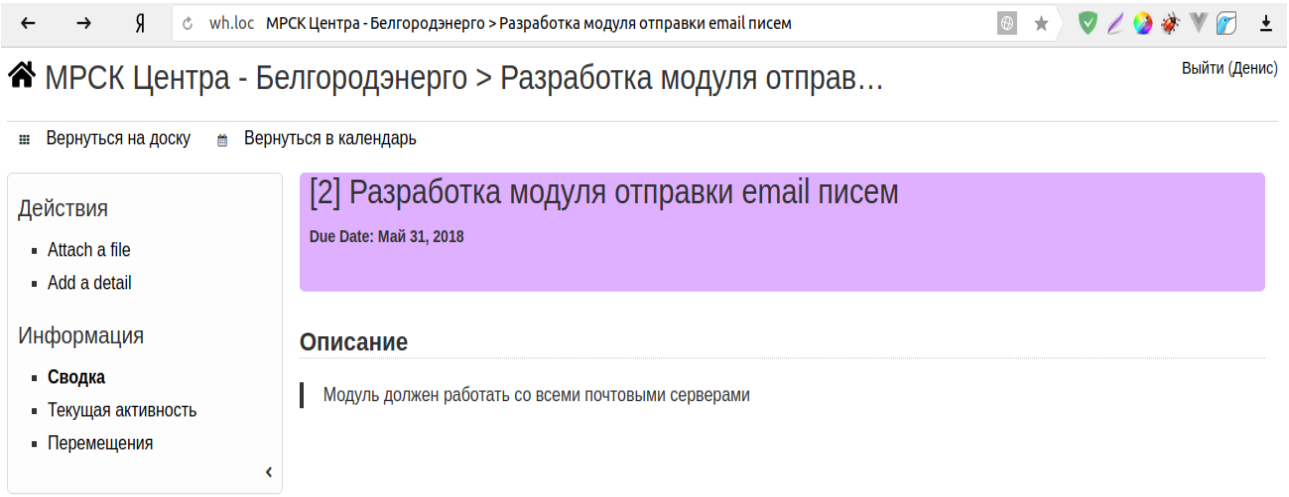


Рисунок 3.18 – Панель управления задачей

На рисунке 3.19 показан результат заполнения данных для задачи.

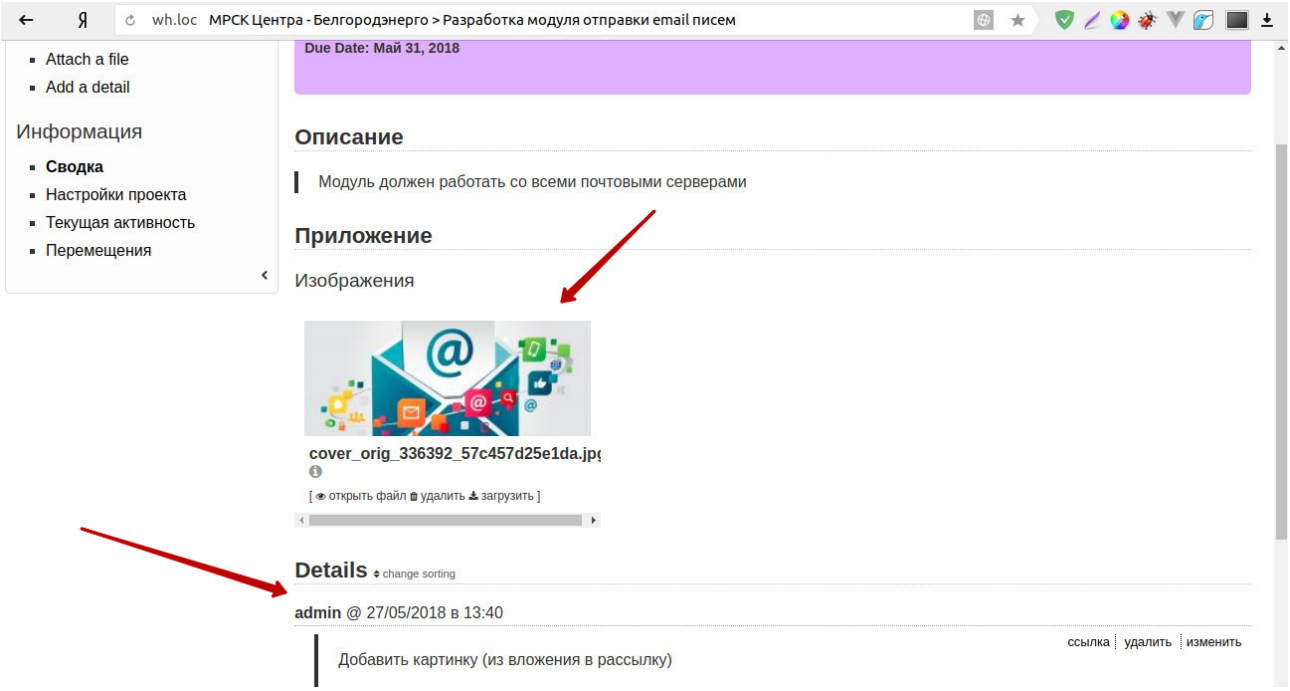


Рисунок 3.19 – Результат добавления новой информации для задачи

Так же для иллюстрации плана, графика работ по какому-либо проекту, доступна диаграмма Гантта, которая показана на рисунке 3.20.

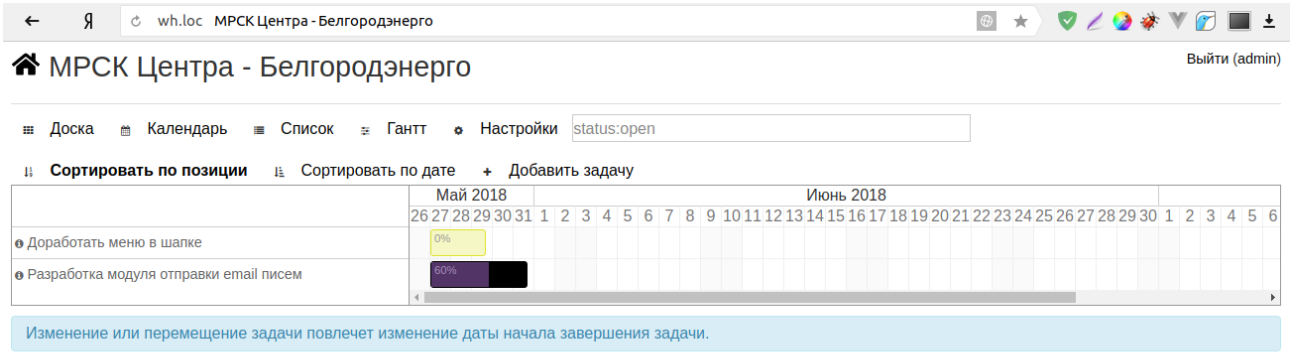
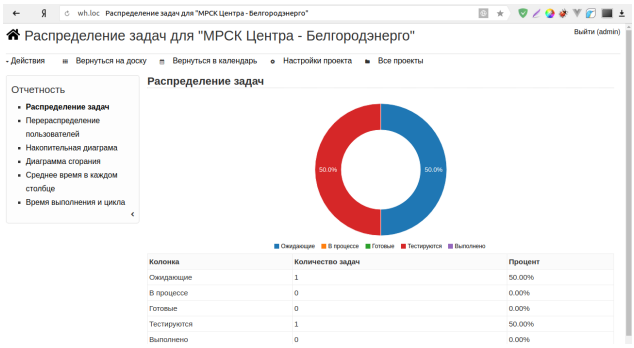
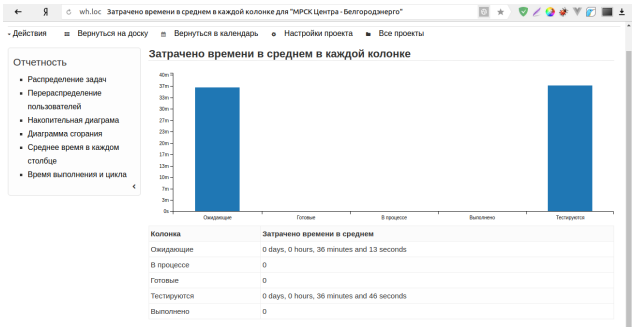


Рисунок 3.20 – Диаграмма Гантта проекта

Для анализа распределения и выполнения задач и количества затраченного времени на каждую из задач, имеется возможность построить диаграмму. На рисунке 3.21(а) показана диаграмма распределения задач, на рисунке 3.21(б) диаграмма, которая показывает среднее время, проведенное задачами в каждой колонке.



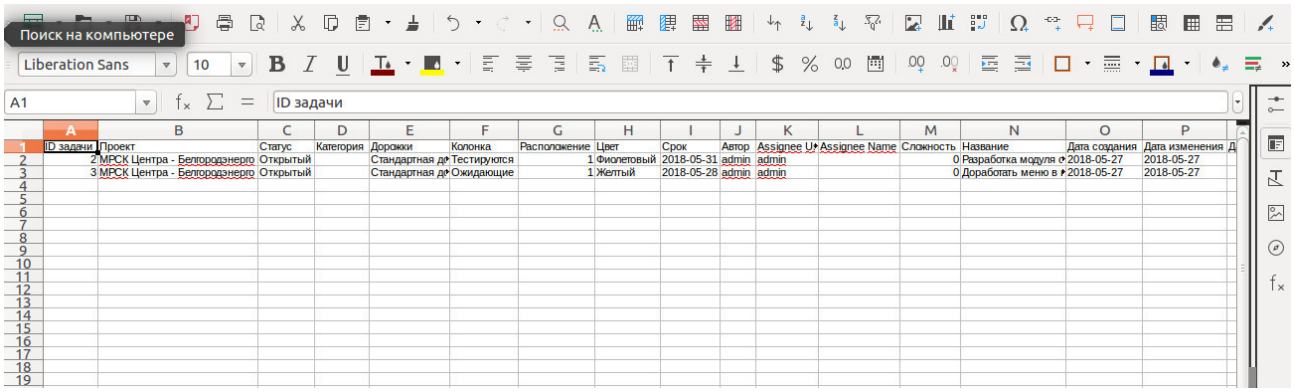


а) б)

Рисунок 3.21 – Диаграммы анализа данных: а) диаграмма распределения задач; б) диаграмма затрачено времени в среднем в каждой колонке

Для составления отчетности по проекту или задаче необходимо сделать экспорт данных. Все данные формируются в файл с расширением «.csv», которые содержат в себе транзакции осуществленные конкретным пользователем на временном интервале, который так же включен в содержимое файла. Посмотреть содержимое файла можно через Excell, на Windows, или

LibreOffice Calc, на операционной системе Linux. Содержимое файла отчетности показано на рисунке 3.22.



ID задачи	Проект	Статус	Категория	Дорожки	Колонка	Расположение	Цвет	Срок	Автор	Assignee UI	Assignee Name	Сложность	Название	Дата создания	Дата изменения
2	МРСК Центра - Белгородэнерго	Открытый		Стандартная до	Тестируются	1	Фиолетовый	2018-05-31	admin	admin		0	Разработка модуля о	2018-05-27	2018-05-27
3	МРСК Центра - Белгородэнерго	Открытый		Стандартная до	Ожидающие	1	Желтый	2018-05-28	admin	admin		0	Доработать меню в р	2018-05-27	2018-05-27

Рисунок 3.22 – Содержимое файла отчетности

3.4 Тестирование автоматизированной системы

Для проверки корректной работы функционала приложения протестируем основные модули, при помощи framework – PHPUnit. Написание тестов поможет осуществить проверку, что данный модуль работает именно так как ожидается. Написав один раз тесты, всякий раз при внесении изменения в существующий код, останется только запустить тесты, для проверки, что всё работает правильно и новый код не повлиял на работу старого.

В ходе разработки автоматизированной системы было написано 479 тестов, которые покрывают весь реализованный функционал. Тестирование системы, по времени, занимает 1 минуту и 35 секунд, а необходимый объем памяти занимает 80 Мб.

Выходными данными для каждого написанного теста является ожидаемый результат.

На рисунке 3.23 показан тест для проверки корректного входа в систему с учетными данными.

```php
public function testAuthenticate()
{
    $provider = new DatabaseAuth($this->container);

    $provider->setUsername('admin');
    $provider->setPassword('admin');
    $this->assertTrue($provider->authenticate());

    $provider->setUsername('admin');
    $provider->setPassword('test');
    $this->assertFalse($provider->authenticate());
}
```

Рисунок 3.23 – Тестирование входа в систему

Исходя из рисунка 3.23, видно, что при попытке входа с учетными данными: admin и admin, ожидается вход в систему, это вызвано утверждением «assertTrue» и наличием данных в базе. Учетные данные: admin и test отсутствуют в базе, поэтому ожидается что такой набор данных вернет false, но тест успешно пройдет так как в утверждении ожидается false.

На рисунке 3.24 показано выполнение всех тестов в режиме консоли.

```
PHPUnit 7.1.5 by Sebastian Bergmann and contributors.

.............................................................  63 / 479 ( 13%)
.............................................................  126 / 479 ( 26%)
.............................................................  189 / 479 ( 39%)
.............................................................  252 / 479 ( 52%)
.............................................................  315 / 479 ( 65%)
.............................................................  378 / 479 ( 78%)
.............................................................  441 / 479 ( 92%)
......................................                         479 / 479 (100%)

Time: 1.35 minutes, Memory: 80.00MB

OK (479 tests, 40521 assertions)
```

Рисунок 3.24 – Выполнение реализованных тестов

На рисунке 3.24 видно, что все тесты успешно пройдены и не вызвали никаких ошибок. Поэтому можно сделать вывод, что система находится в исправном состоянии и готова для использования.

Выводы по третьему разделу.

В данном разделе была создана информационная модель, а также ее описание. Построены инфологическая и даталогическая модели базы данных. Разработана и протестирована информационная системы контроля выполнения задач сотрудниками предприятия «Кроникс Микросистемс».

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы были исследованы и проанализированы назначения, состав, основные характеристики и способы создания автоматизированных систем.

Для реализации поставленной цели, был проведен анализ работы предприятия «Кроникс Микросистемс» и рассмотрены принципы ведения контроля задач сотрудниками предприятия. Проведя анализ предметной области было определено основное назначение разрабатываемой системы и набор необходимых, для реализации, функций.

Для реализации поставленной цели были решены следующие задачи:

– изучены теоретические и методологические основы формирования и развития систем контроля выполнения задач на современных предприятиях;

– выбран метод и средства контроля выполнения задач сотрудниками предприятия;

– спроектированы инфологическая и даталогическая модели БД;

– разработана система контроля выполнения задач сотрудниками предприятия «Кроникс Микросистемс»;

– произведено тестирование разработанной системы.

После анализа деятельности предприятия были сформированы требования к информационному, аппаратному, и программному обеспечению.

По результатам тестирования можно сделать вывод, что разработанная система полностью отвечает оговоренным требованиям и готова для ввода в эксплуатацию. Основными преимуществами автоматизации являются:

– увеличение производительности и пропускной способности предприятия;

– повышение качества выполнения задач;

Основными недостатками автоматизации являются:

– высокая стоимость разработки;

– затраты на поддержку и исследование новых разработок.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Айвалиотис, Д.Н Администрирование сервера NGINX: учебное пособие / Д.Н.Айвалиотис. – Москва: ДМК Пресс, 2013. – 288 с.

2. Андерсон, Д.Э. Канбан. Альтернативный путь в Agile / Д.Э.Андерсон. – Москва: Манн, 2017. – 350 с.

3. Андрейчиков, А.В. Стратегический менеджмент в инновационных организациях: системный анализ и принятие решений: учебное пособие / А.В. Андрейчиков, О.Н. Андрейчикова. – Москва: ИНФРА-М, 2013. – 394 с.

4. Артюшина, Е.А. Разработка web-приложения с использованием архитектуры «клиент-сервер» / Е.А. Артюшина, Е.И. Маркин, К.М.Рябова. // Международный студенческий научный вестник. – 2016. – №. 3-1. – С. 84-86.

5. Бербер, Э. Red Hat Linux для системных администраторов / Д. Барбер, Э. Тернер, Т. Шенк. – Москва: DiaSoft, 2011. – 672 с.

6. Блам, Р.С. Система электронной почты на основе Linux / Р.С.Блам. – Москва: Вильямс, 2017. – 448 с.

7. Веллинг, Л. Разработка веб-приложений с помощью PHP и MySQL / Л.Веллинг, Л.Томсон. – Москва: Альфа-книга, 2017. – 768 с.

8. Дубейковский, В.А. Эффективное моделирование с CA ERwin Process Modeler (BPwin; AllFusion Process Modeler) / В.А.Дубейковский. – Москва: Диалог-МИФИ, 2009. – 384 с.

9. Жданов, С.А. Экономические модели и методы в управлении / С.А.Жданов. – Москва: Дис, 2012. – 63 с.

10. Закас, Н.С. Ajax для професионалов / Н.С.Закас, Д.Ф.Мак-Пик, Д.Фосетт. – Санкт-Петербург: Символ-Плюс, 2013. – 488 с.

11. Козлов, В.Н. Системный анализ, оптимизация и принятие решений: учебное пособие / В.Н.Козлов. – Санкт-Петербург: Проспект, 2013. – 174 с.

12. Котеров, Д.В. PHP7. Наиболее полное руководство / Д.В. Котеров, И.В.Симдянов. – Санкт-Петербург: БВХ, 2016. – 1088 с.

13. Кузнецов, М.В. Объектно-ориентированное программирование на PHP / М.В.Кузнецов, И.В.Симдянов. – Москва: БХВ-Петербург, 2008. – 608 с.

14. Курзаева, Л. В. Введение в теорию систем и системный анализ: учебное пособие / Л.В.Курзаева. – Магнитогорск: МаГУ, 2015. – 185 с.

15. Лившиц, В. Н. Системный анализ рыночного реформирования нестационарной экономики России, 1992-2013 /В.Н.Лившиц. – Москва: Ленанд, 2013. – 640 с.

16. Маклаков, С.В, BPwin и Erwin. CASE-средства для разработки информационных систем / С.В.Маклаков. – Москва: Диалог-МИФИ, 2000. – 256 с.

17. Мартишин, С.А. Базы данных. Практическое применение СУБД SQL- и NoSOL-типа для применения проектирования информационных систем / С.А.Мартишин, В.Л.Симонов, М.В.Храпченко. – Москва: Форум, 2018. – 368 с.

18. Матюшечев, Л.Ю. Моделирование информационных систем: учебное пособие / Л.Ю.Матюшевич, А.В.Флегонтов – Санкт-Петербург:Лань, 2018.– 112 с.

19. Мейер, Б. Основы объектно-ориентированного программирования: учебное пособие / Б.Мейер. – Москва: НОУ "Интуит", 2016. – 970 с.

20. Мюллер, Р.Д. Базы данных и UML. Проектирование / Р.Д. Мюллер. – Москва: ЛОРИ, 2015. – 420 с.

21. Осипов, А.Л. Информационная система учета и анализа индивидуальных планов: инновационные подходы в решении проблем современного общества / А.Л.Осипов, В. П.Трушина. – Москва: ИНФРА-М, 2017. – 192 с.

22. О'Лири Д. ERP системы. Современное планирование и управление ресурсами предприятия: текст / Д.О'Лири. – Москва: ИНФРА-М, 2009. – 272 с.

23. Полянсков, Ю.В. Проектирование и разработка инструмента для автоматизации процессов планирования и управления производством и производственными ресурсами на примере авиастроительного предприятия /

Ю.В.Полянсков // Известия самарского научного центра Российской академии наук. – 2016. – Т. 18. – №. 4-3. – С. 19.

24. Поршнева, А.Г. Управление организацией / А.Г.Поршнева, З.П. Румянцева – Москва: ИНФРА-М, 2013. – 736 с.Риз, Д.MySQL и mSQL.Базы данных для небольших предприятий и Интернета / Д.Риз, Т.Кинг, Д.Р.Яргер. – Санкт-Петербург: Символ-Плюс, 2014. – 560 с.

25. Сергеев, В.И. Логистика в бизнесе / В.И.Сергеев. – Москва: Инфра-М, 2011. – 601 с.

26. Схиртладзе, А.Г. Автоматизация технологических процессов: учебное пособие / А.Г.Схиртладзе, С.В.Бочкарев, А.Н.Лыков. – Старый Оскол: ТНТ, 2013. – 524 с.

27. Фуфаев, Д.Э. Разработка и эксплуатация автоматизированных информационных систем / Д.Э.Фуфаев, Э.В.Фуфаев. – Москва: Академия, 2014. – 304 с.

28. Хаббард, Дж. Автоматизированное проектирование баз данных / Дж. Хаббард. – Москва: Мир, 2013. – 296 с.

29. Allen, G. The Definitive Guide to SQLite (Second Edition) / G.Allen, M.Owens – Москва: Книга по Требованию, 2010. – 369 с.

30. Guermazi, H.H. Online Project Management and PHP7 Application: A Real Case Study / H.H.Guermazi, A.Zorai // International Conference on Digital Economy. – Springer, Cham, 2017. – С. 116-128.

31. Russel J.B. CA ERwin Data Modeler / J.B.Russel. – Москва: Книга по Требованию, 2012. – 848 с.

ПРИЛОЖЕНИЕ А

Даталогическая модель базы данных

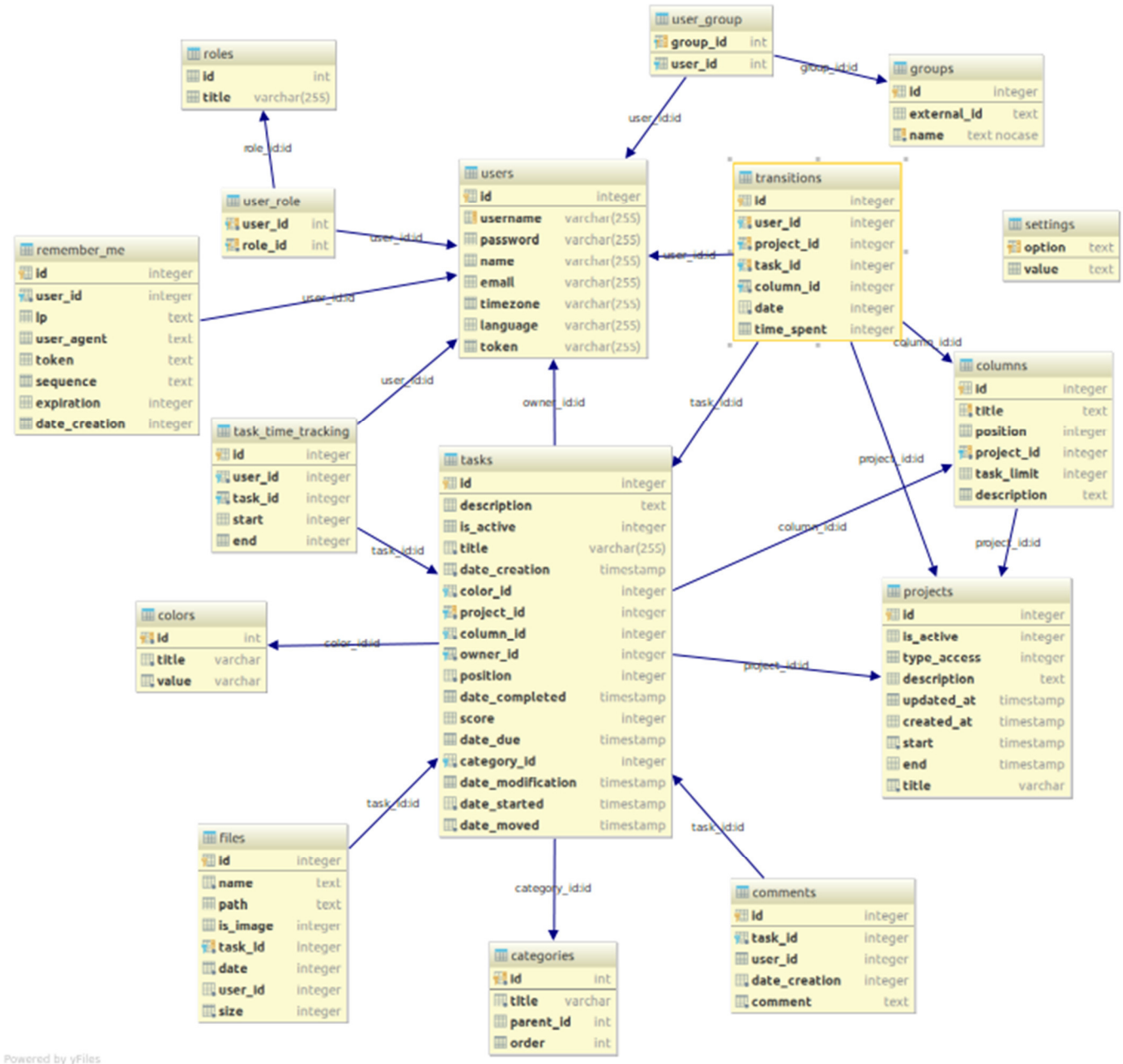


Рисунок А.1 – Даталогическая модель базы данных

ПРИЛОЖЕНИЕ Б

Класс-контроллер авторизации пользователей

```
<?php
namespace Kanboard\Auth;
use Kanboard\Core\Base;
use Kanboard\Core\Security\PasswordAuthenticationProviderInterface;
use Kanboard\Core\Security\SessionCheckProviderInterface;
use Kanboard\Model\User;
use Kanboard\User\DatabaseUserProvider;

class DatabaseAuth extends Base implements PasswordAuthenticationProviderInterface,
SessionCheckProviderInterface
{
   private $userInfo = array();
   private $username = '';
   private $password = '';
   public function getName()
   {
      return 'Database';
   }
   public function authenticate()
   {
      $user = $this->db
         ->table(User::TABLE)
         ->columns('id', 'password')
         ->eq('username', $this->username)
         ->eq('disable_login_form', 0)
         ->eq('is_ldap_user', 0)
         ->findOne();
      if (! empty($user) && password_verify($this->password, $user['password'])) {
         $this->userInfo = $user;
         return true;
      }
      return false;
   }
   public function isValidSession()
   {
      return $this->user->exists($this->userSession->getId());
   }
   public function getUser()
   {
      if (empty($this->userInfo)) {
         return null;
      }

      return new DatabaseUserProvider($this->userInfo);
   }
   public function setUsername($username)
   {
      $this->username = $username;
   }
   public function setPassword($password)
   {
      $this->password = $password;}
   }
```

ПРИЛОЖЕНИЕ В

Класс-контроллер работы с доской задач

```
<?php
namespace Kanboard\Controller;
class Board extends Base
{
    public function readonly()
    {
        $token = $this->request->getStringParam('token');
        $project = $this->project->getByToken($token);
        if (empty($project)) {
            $this->forbidden(true);
        }
        $this->response->html($this->template->layout('board/view_public', array(
            'project' => $project,
            'swimlanes' => $this->board->getBoard($project['id']),
            'title' => $project['name'],
            'description' => $project['description'],
            'no_layout' => true,
            'not_editable' => true,
            'board_public_refresh_interval' => $this->config->get('board_public_refresh_interval'),
            'board_private_refresh_interval' => $this->config->get('board_private_refresh_interval'),
            'board_highlight_period' => $this->config->get('board_highlight_period'),
        )));
    }
    public function show()
    {
        $params = $this->getProjectFilters('board', 'show');
        $this->response->html($this->template->layout('board/view_private', array(
            'categories_list' => $this->category->getList($params['project']['id'], false),
            'users_list' => $this->projectUserRole->getAssignableUsersList($params['project']['id'], false),
            'custom_filters_list' => $this->customFilter->getAll($params['project']['id'], $this->userSession->getId()),
            'swimlanes' => $this->taskFilter->search($params['filters']['search'])->getBoard($params['project']['id']),
            'description' => $params['project']['description'],
            'board_private_refresh_interval' => $this->config->get('board_private_refresh_interval'),
            'board_highlight_period' => $this->config->get('board_highlight_period'),
        ) + $params));
    }
    public function save()
    {
        $project_id = $this->request->getIntegerParam('project_id');
        if (! $project_id || ! $this->request->isAjax()) {
            return $this->response->status(403);
        }
        if (! $this->projectPermission->isUserAllowed($project_id, $this->userSession->getId())) {
            $this->response->text('Forbidden', 403);
        }
        $values = $this->request->getJson();
        $result =$this->taskPosition->movePosition(
            $project_id,
            $values['task_id'],
            $values['column_id'],
            $values['position'],
            $values['swimlane_id']
        );
        if (! $result) {
            return $this->response->status(400);
        }
        $this->response->html($this->renderBoard($project_id), 201);
```

```
    }
    public function check()
    {
        if (! $this->request->isAjax()) {
            return $this->response->status(403);
        }
        $project_id = $this->request->getIntegerParam('project_id');
        $timestamp = $this->request->getIntegerParam('timestamp');
        if (! $this->projectPermission->isUserAllowed($project_id, $this->userSession->getId())) {
            $this->response->text('Forbidden', 403);
        }
        if (! $this->project->isModifiedSince($project_id, $timestamp)) {
            return $this->response->status(304);
        }
        $this->response->html($this->renderBoard($project_id));
    }
    public function reload()
    {
        if (! $this->request->isAjax()) {
            return $this->response->status(403);
        }
        $project_id = $this->request->getIntegerParam('project_id');
        if (! $this->projectPermission->isUserAllowed($project_id, $this->userSession->getId())) {
            $this->response->text('Forbidden', 403);
        }
        $values = $this->request->getJson();
        $this->userSession->setFilters($project_id, empty($values['search']) ? '' : $values['search']);
        $this->response->html($this->renderBoard($project_id));
    }
    public function collapse()
    {
        $this->changeDisplayMode(true);
    }
    public function expand()
    {
        $this->changeDisplayMode(false);
    }
    private function changeDisplayMode($mode)
    {
        $project_id = $this->request->getIntegerParam('project_id');
        $this->userSession->setBoardDisplayMode($project_id, $mode);
        if ($this->request->isAjax()) {
            $this->response->html($this->renderBoard($project_id));
        } else {
            $this->response->redirect($this->helper->url->to('board', 'show', array('project_id' => $project_id)));
        }
    }

    private function renderBoard($project_id)
    {
        return $this->template->render('board/table_container', array(
            'project' => $this->project->getById($project_id),
            'swimlanes' => $this->taskFilter->search($this->userSession->getFilters($project_id))->getBoard($project_id),
            'board_private_refresh_interval' => $this->config->get('board_private_refresh_interval'),
            'board_highlight_period' => $this->config->get('board_highlight_period'),
        ));
    }
}
```

ПРИЛОЖЕНИЕ Г

Класс-модель «Пользователь»

```
<?php
namespace Kanboard\Model;
use PicoDb\Database;
use SimpleValidator\Validator;
use SimpleValidator\Validators;
use Kanboard\Core\Session\SessionManager;
use Kanboard\Core\Security\Token;
use Kanboard\Core\Security\Role;

class User extends Base
{

   const TABLE = 'users';
   const EVERYBODY_ID = -1;

   public function exists($user_id)
   {
      return $this->db->table(self::TABLE)->eq('id', $user_id)->exists();
   }
   public function getQuery()
   {
      return $this->db
                  ->table(self::TABLE)
                  ->columns(
                     'id',
                     'username',
                     'name',
                     'email',
                     'role',
                     'is_ldap_user',
                     'notifications_enabled',
                     'google_id',
                     'github_id',
                     'twofactor_activated'
                  );
   }
   public function getFullname(array $user)
   {
      return $user['name'] ?: $user['username'];
   }
   public function isAdmin($user_id)
   {
      return $this->userSession->isAdmin() ||  // Avoid SQL query if connected
            $this->db
               ->table(User::TABLE)
               ->eq('id', $user_id)
               ->eq('role', Role::APP_ADMIN)
               ->exists();
   }
   public function getById($user_id)
   {
      return $this->db->table(self::TABLE)->eq('id', $user_id)->findOne();
   }
   public function getByExternalId($column, $id)
   {
      if (empty($id)) {
         return false;
      }
```

```
    return $this->db->table(self::TABLE)->eq($column, $id)->findOne();
}
public function getByUsername($username)
{
    return $this->db->table(self::TABLE)->eq('username', $username)->findOne();
}
public function getIdByUsername($username)
{
    return $this->db->table(self::TABLE)->eq('username', $username)->findOneColumn('id');
}
public function getByEmail($email)
{
    if (empty($email)) {
        return false;
    }

    return $this->db->table(self::TABLE)->eq('email', $email)->findOne();
}
public function getByToken($token)
{
    if (empty($token)) {
        return false;
    }

    return $this->db->table(self::TABLE)->eq('token', $token)->findOne();
}
public function getAll()
{
    return $this->getQuery()->asc('username')->findAll();
}
public function count()
{
    return $this->db->table(self::TABLE)->count();
}
public function getList($prepend = false)
{
    $users = $this->db->table(self::TABLE)->columns('id', 'username', 'name')->findAll();
    $listing = $this->prepareList($users);

    if ($prepend) {
        return array(User::EVERYBODY_ID => t('Everybody')) + $listing;
    }

    return $listing;
}
public function prepareList(array $users)
{
    $result = array();

    foreach ($users as $user) {
        $result[$user['id']] = $this->getFullname($user);
    }
    asort($result);
    return $result;
}
public function prepare(array &$values)
{
    if (isset($values['password'])) {
        if (! empty($values['password'])) {
            $values['password'] = \password_hash($values['password'], PASSWORD_BCRYPT);
        } else {
            unset($values['password']);
```

```
        }
    }

    $this->removeFields($values, array('confirmation', 'current_password'));
    $this->resetFields($values, array('is_ldap_user', 'disable_login_form'));
    $this->convertNullFields($values, array('gitlab_id'));
    $this->convertIntegerFields($values, array('gitlab_id'));
}
public function create(array $values)
{
    $this->prepare($values);
    return $this->persist(self::TABLE, $values);
}
public function update(array $values)
{
    $this->prepare($values);
    $result = $this->db->table(self::TABLE)->eq('id', $values['id'])->update($values);

    // If the user is connected refresh his session
    if ($this->userSession->getId() == $values['id']) {
        $this->userSession->initialize($this->getById($this->userSession->getId()));
    }
    return $result;
}
public function remove($user_id)
{
    return $this->db->transaction(function (Database $db) use ($user_id) {

        // All assigned tasks are now unassigned (no foreign key)
        if (! $db->table(Task::TABLE)->eq('owner_id', $user_id)->update(array('owner_id' => 0))) {
            return false;
        }
        // All assigned subtasks are now unassigned (no foreign key)
        if (! $db->table(Subtask::TABLE)->eq('user_id', $user_id)->update(array('user_id' => 0))) {
            return false;
        }
        // All comments are not assigned anymore (no foreign key)
        if (! $db->table(Comment::TABLE)->eq('user_id', $user_id)->update(array('user_id' => 0))) {
            return false;
        }
        // All private projects are removed
        $project_ids = $db->table(Project::TABLE)
            ->eq('is_private', 1)
            ->eq(ProjectUserRole::TABLE.'.user_id', $user_id)
            ->join(ProjectUserRole::TABLE, 'project_id', 'id')
            ->findAllByColumn(Project::TABLE.'.id');
        if (! empty($project_ids)) {
            $db->table(Project::TABLE)->in('id', $project_ids)->remove();
        }
        // Finally remove the user
        if (! $db->table(User::TABLE)->eq('id', $user_id)->remove()) {
            return false;
        }
    });
}
public function enablePublicAccess($user_id)
{
    return $this->db
            ->table(self::TABLE)
            ->eq('id', $user_id)
            ->save(array('token' => Token::getToken()));
}
```

```
    public function disablePublicAccess($user_id)
    {
        return $this->db
                ->table(self::TABLE)
                ->eq('id', $user_id)
                ->save(array('token' => ''));
    }
    private function commonValidationRules()
    {
        return array(
            new Validators\MaxLength('role', t('The maximum length is %d characters', 25), 25),
            new Validators\MaxLength('username', t('The maximum length is %d characters', 50), 50),
            new Validators\Unique('username', t('The username must be unique'), $this->db->getConnection(),
self::TABLE, 'id'),
            new Validators\Email('email', t('Email address invalid')),
            new Validators\Integer('is_ldap_user', t('This value must be an integer')),
        );
    }
    private function commonPasswordValidationRules()
    {
        return array(
            new Validators\Required('password', t('The password is required')),
            new Validators\MinLength('password', t('The minimum length is %d characters', 6), 6),
            new Validators\Required('confirmation', t('The confirmation is required')),
            new Validators\Equals('password', 'confirmation', t('Passwords don\'t match')),
        );
    }
    public function validateCreation(array $values)
    {
        $rules = array(
            new Validators\Required('username', t('The username is required')),
        );

        if (isset($values['is_ldap_user']) && $values['is_ldap_user'] == 1) {
            $v = new Validator($values, array_merge($rules, $this->commonValidationRules()));
        } else {
            $v = new Validator($values, array_merge($rules, $this->commonValidationRules(), $this-
>commonPasswordValidationRules()));
        }

        return array(
            $v->execute(),
            $v->getErrors()
        );
    }
    public function validateModification(array $values)
    {
        $rules = array(
            new Validators\Required('id', t('The user id is required')),
            new Validators\Required('username', t('The username is required')),
        );

        $v = new Validator($values, array_merge($rules, $this->commonValidationRules()));

        return array(
            $v->execute(),
            $v->getErrors()
        );
    }
    public function validateApiModification(array $values)
    {
        $rules = array(
            new Validators\Required('id', t('The user id is required')),
```

```
        );

        $v = new Validator($values, array_merge($rules, $this->commonValidationRules()));

        return array(
            $v->execute(),
            $v->getErrors()
        );
    }
    public function validatePasswordModification(array $values)
    {
        $rules = array(
            new Validators\Required('id', t('The user id is required')),
            new Validators\Required('current_password', t('The current password is required')),
        );

        $v = new Validator($values, array_merge($rules, $this->commonPasswordValidationRules()));

        if ($v->execute()) {
            if ($this->authenticationManager->passwordAuthentication($this->userSession->getUsername(),
$values['current_password'], false)) {
                return array(true, array());
            } else {
                return array(false, array('current_password' => array(t('Wrong password'))));
            }
        }

        return array(false, $v->getErrors());
    }}
```