МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»
Кафедра «Информационных систем и кибербезопасности»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD
_____А.К. Мукашева
«____»_____2021 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка информационной системы «IT Group»

Специальность 5В070300 – «Информационные системы»

Выполнила Ниязова К.К. Группа ИС-17-2
Научный руководитель к.т.н., доцент Тусупова Б.Б.

Консультанты:
по экономической части: д.э.н., проф. Сатова Р.К.

_____«___»_____2021 г.

по безопасности жизнедеятельности: д.х.н., проф. Приходько Н.Г.

_____«____»_____2021 г.

по применению
вычислительной техники: _____

_____«____»_____2021 г.

Нормоконтролер: ст. преп. Пипия М.Т.

_____«____»_____2021 г.

Рецензент: Ph.D, доцент Айдаров К.А.

_____«____»_____2021 г.

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»

Институт информационных технологий

Кафедра «Информационных систем и кибербезопасности»

Специальность 5В070300 - Информационные системы

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Ниязовой Карине Курбанжановне

Тема проекта: Разработка информационной системы «IT Group»

Утверждена приказом по университету №____от «_____» 2021 г.

Срок сдачи законченного проекта «____»_____2021 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): данные преддипломной практики

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

а) определение основных целей и требований к ИС;
б) исследование и анализ существующих систем;
в) разработка информационного обеспечения ИС;
г) разработка интерфейса и программная реализация системы;
д) расчет экономических показателей;
е) расчет показателей по обеспечению безопасности жизнедеятельности

Перечень графического материала (с точным указанием обязательных чертежей): имеется 26 таблиц и 70 иллюстраций.

Основная рекомендуемая литература:

1 Фленов М.Е. Библия C#. – 4-е изд. перераб. и доп. – СПб.: БХВ-Петербург, 2019. – 512 с.: ил.

2 Грекул В. И. и др. Проектирование информационных систем. – М.: Интернет Университет Информационных Технологий, 2010.

3 Албахари, Джозеф, Албахари, Бен. C#7.0. Карманный справочник. :

Пер. с англ. – Пер. с англ. – СпБ. : ООО «Альфа-Книга», 2017. – 224 . : ил. – Парал. тит. англ.

4 Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2006. – 544 с: ил.

Консультация по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Сатова Р.К.	19.04.2021–10.05.2021	
Безопасности жизнедеятельности	Приходько Н.Г.	19.04.2021–14.05.2021	
Программная часть	Тусупова Б.Б.	17.05.2021–24.05.2021	
Нормоконтролер	Пипия М.Т.	31.05.2021–08.06.2021	

ГРАФИК
подготовки дипломной работы
(проекта)

Наименования разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечания
Анализ и исследование предметной области	01.02.2021 – 20.03.2021	
Проектирование приложения	01.02.2021 – 28.03.2021	
Программная реализация	01.02.2021 – 30.04.2021	

Дата выдачи задания «____»_____2021 г.

Заведующий кафедрой _____А.К. Мукашева

Научный руководитель проекта _____Б.Б. Тусупова

Задание принял к исполнению студент _____К.К. Ниязова

Аңдатпа

Берілген дипломдық жобаның мақсаты – «S&G IT Group» ЖШС-нің тапсырысы бойынша кәсіпорынның бизнес-көрстекіштерін автоматтандыру, қызметкерлерінің жұмыстарын орындалуын қадағалау және ститистика мен есеп беру ұйымдастыруын жүзеге асыратын ақпараттық жүйе құрастыру. Жүйе мақсаты – кәсіпорынның басқарылуын және жобалық жұмыстарын оңайлату.

Бұл жүйе desktop-қосымшасы түрінде жүзеге асыралды. Бағдарлама C# бағдарламалау тілінде, графикалық интерфейс .NET платформасы UI-фреймворктың көмегімен орындалды. Жобаға MS SQL Server дерекқорды басқару жүйесі қолданылды.

Жобада қолданыстағы ұқсас жүйелер, кәсіпорынның құрылымы және бизнес-процестері, дерекөорын жобалау, ақпараттық және бағдарламалық қамтамасыз етуді құрастыру қарастырылды.

Жобаның нәтижесі – кәсіпорынды басқаратын ақпараттық жүйе.

Аннотация

Данный дипломный проект посвящен разработке информационной системы для ТОО «S&G IT Group» для автоматизации контроля бизнес-показателей, отслеживания выполнения задач сотрудниками, сбора статистики и организации отчетности. Цель системы – упрощение управления организацией и работы над проектами.

Данная система реализована в виде настольного приложения. Программа была реализована на языке программирования C#, графический интерфейс создан с помощью UI фреймворка платформы .NET. Для проекта использовалась СУБД MS SQL Server.

В проекте рассматриваются существующие аналогичные системы, структура и бизнес-процессы предприятия, проектирование базы данных, разработка информационного и программного обеспечения.

Результатом проекта является информационная система управления предприятием.

Abstract

This thesis project is dedicated to the development of an information system for S&G IT Group LLP to automate the control of business indicators, track the performance of tasks by employees, collect statistics and organize reporting. The purpose of the system is to simplify the management of the organization and work on projects.

This system is implemented as a desktop application. The program was implemented in the C # programming language, the graphical interface was created using the .NET platform UI framework. The MS SQL Server DBMS was used for the project.

The project examines the existing similar systems, structure and business processes of the enterprise, database design, development of information and software.

The result of the project is an enterprise management information system.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день в работе разного рода организаций неотъемлемой частью является управление проектами, актуальное не только для реализации крупных, масштабных задач, но и для деятельности организации в целом. В быстроразвивающихся компаниях с увеличением числа проектов увеличивается и количество информации. Для повышения эффективности своей деятельности компании зачастую используют системы управления проектами.

Данные системы позволяют повысить эффективность и продуктивность сотрудников в проектной работе, так как отпадает необходимость запоминать большое количество информации. Пользователи системы управления проектами могут контролировать график выполнения работ, получать оперативную информацию по проектам. Для руководителей обеспечен контроль реализации проектов компании, поддержка планирования, анализ эффективности деятельности. Многие такие системы представлены в виде мобильных и веб-приложений. Однако реализация в виде настольных (desktop) приложений имеет ряд преимуществ:

- автономная работа без зависимости от интернет-соединения;
- более безопасны и имеют средства развертывания;
- более функциональны;
- использование настольного приложения позволяет работать, не отвлекаясь на множество вкладок в браузере.

Многие поставщики программного обеспечения предлагают решения в формате настольных приложений, в том числе и для управления проектами (Basecamp, Wrike, Notion). В этом заключается актуальность проекта.

Целью проекта является разработка информационной системы в виде desktop-приложения, предназначенной для:

- автоматизации контроля сотрудников и бизнес-показателей;
- организации оперативной и хронологической отчетности;
- сбора, обработки и актуализации статистики;
- отслеживания выполнение задач сотрудниками.

В соответствии с целью проекта определены следующие задачи:

- изучение предметной области;
- изучение аналогичных систем;
- определение основных требований к ИС (постановка задачи, техническое задание);
- разработка информационного обеспечения (проектирование функциональной структуры, выбор СУБД);
- разработка программного обеспечения (выбор средств разработки, разработка, тестирование и отладка);
- расчет экономической эффективности;
- рассмотрение вопросов безопасности жизнедеятельности.

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Обзор литературы

В ходе работы над проектом были использованы научная литература, документация языка программирования C# и статьи из Интернет. В качестве основной литературы были использованы книги «Библия C#» от автора М. Фленова, «Язык программирования C# 7 и платформы .NET и .NET Core» авторов Эндрю Троелсена и Филиппа Джепикса, а также «C# 8.0. Карманный справочник» Джозефа Албахари и Бена Албахари.

Все необходимые сведения о разработке приложений под ОС Windows изложены в книге «Библия C#». Не смотря на свое название, книга посвящена не только языку программирования, но и .NET – платформе от компании Microsoft, позволяющей решать широкий круг задач. В книге рассматриваются темы, начиная от с основ языка и разработки программ для работы в командной строке и заканчивая созданием приложений различной сложности (создание графических программ, подключение баз данных и т.д.) [1].

В книге «Язык программирования C# 7 и платформы .NET и .NET Core» рассматриваются платформы .NET и .NET Core, объектно-ориентированное программирование, многопоточное, параллельное и асинхронное программирование, а также доступ к данным (ADO.NET и Entity Framework) и ASP.NET [2].

Третий источник был больше вспомогательным, нежели основным. В нем содержатся основные сведения о языке разработки, изложенные кратко, поэтому им удобно пользоваться как справочником.

Более детальная и подробная информация, касающаяся разработки на C# и платформы .NET изложена в официальной документации по C# на сайте https://docs.microsoft.com/ru-ru/dotnet/csharp/, которая доступна на русском языке. Здесь очень подробно описываются основные понятия, дается руководство по программированию и справочник по языку, а также приводятся различные статьи о нововведениях в языке.

Так как разработка информационной системы не может обойтись без баз данных, была изучена соответствующая литература. Можно выделить книги «Технологии проектирования баз данных» Осипова Д.Л. и «Разработка и администрирование баз данных» Федоровой Г.Н. В книге «Технологии проектирования баз данных» даются теоретические сведения о базах данных и системах управления базами данных, рассматриваются вопросы технологии разработки баз данных (жизненный цикл), концептуального и логического проектирования, безопасности, а также язык структурированных запросов [3].

Во втором источнике разъясняются принципы построения моделей данных, процесс нормализации, описываются различные классы CASE-технологий. Помимо этого рассматриваются основные приемы управления

базой данных с использованием языка SQL и дается обзор наиболее популярных современных СУБД [4].

Так как в качестве СУБД был выбран MS SQL Server, то для работы была достаточна официальная документация, представленная на сайте https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15. Здесь описываются внутренние компоненты и архитектура СУБД, структура, приведены статьи о безопасности и инструментах разработки, а также справочник.

1.2 Анализ существующих систем

Деятельность современной IT-компании так или иначе связана с ведением проектов. Работа над проектом начинается с выбора методологии разработки проекта. На сегодняшний день популярными методологиями являются каскадная Waterfall и гибкая Agile, хотя все чаще большинство разработчиков делают выбор в пользу второй. Поскольку гибкая методология разработки очень распространена, множество программ и сервисов по управлению проектами поддерживают ее основные методики (Scrum и Kanban).

Системы управления проектами помогают команде сотрудников четко распределять время и задачи по проекту, а также контролировать выполнение этих задач и делать работу эффективнее. Руководитель проекта обеспечивается инструментарием, который дает ему возможность контролировать ход реализации проекта, распределять нагрузку между сотрудниками, принимать управленческие решения. Сотрудники получают конкретные задачи, у каждой из которых есть свой срок исполнения и приоритет, и необходимую информацию для выполнения задач.

Системы управления проектами в основном реализованы в виде информационной системы, представляющей собой организационно-технологический комплекс программных, технических и информационных средств и инструментов, направленный на реализацию, поддержку и повышение эффективности процессов управления проектами [5].

Далее будут рассмотрены существующие системы управления проектами.

1) MS Project – программа управления проектами, продукт компании Microsoft. MS Project можно рассматривать как инструмент менеджера проектов, который позволяет разрабатывать планы распределения ресурсов в соответствии с задачами [6]. Программа может создавать расписания критического пути проекта, который выводится в виде диаграммы Ганта. Кроме этого, применение программы обеспечивает календарное, ресурсное и бюджетное планирование и контроль проектов, а также управление рисками. Рабочая область Microsoft Project представлена на рисунке 1.1.

Работа начинается с создания проекта, который обычно представлен в виде файла с расширением «*.mpp*». Затем в проект добавляются задачи с

указанием дат начала и окончания, на основе которых вычисляется длительность.

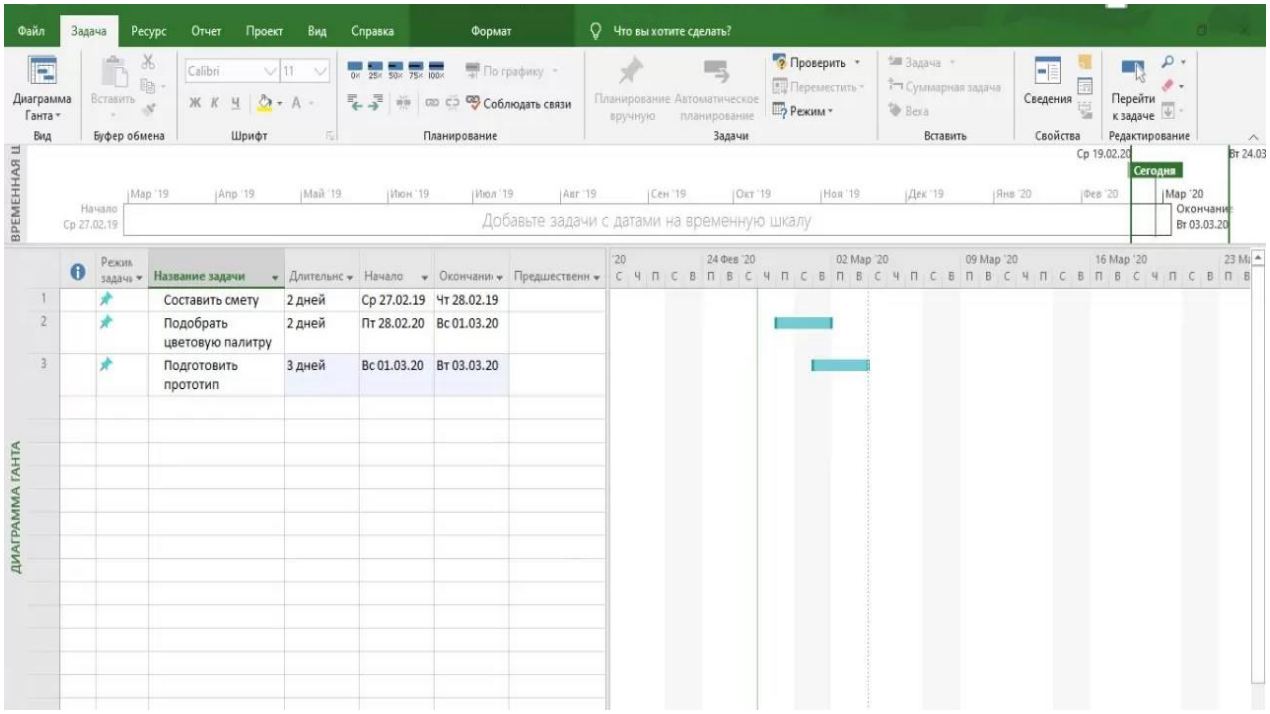


Рисунок 1.1 – Программа Microsoft Project

Каждая задача помимо дат имеет параметры: название, способ управления, процент завершения и приоритет. Между задачами можно создавать связи, которые отображаются на рабочей области во вкладке «Предшественники». Программа может создать диаграмму проекта произвести расчет критического пути, как на рисунке 1.2.

Преимущества MS Project [6]:

- продукт входит в семейство Microsoft Office, то есть возможность доработки под определенные нужды путем программирования или покупки готовых решений;
- стабильность системы;
- интерфейс приближен к Microsoft Excel;
- возможность доработки;
- простота установки.

Недостатки [6]:

- эффективно работать с данной программой можно, имея теоретические знания в области управления проектами;
- не поддерживается совместная работа;
- оценка степени готовности проекта не соответствует действительности;
- несмотря на то, что интерфейс приближен к MS Excel, он остается перегруженным
- сложно получить нужную информацию.

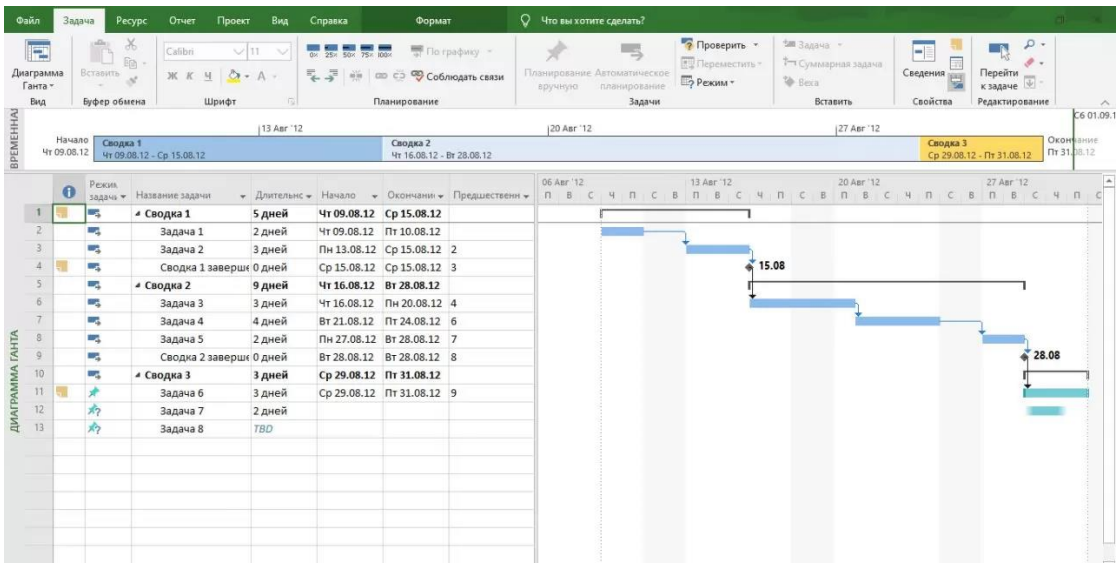


Рисунок 1.2 – Диаграмма проекта в Microsoft Project

2) Trello – платформа для управления проектами. Интерфейс Trello основан на использовании Kanban-досок, на которых располагаются задачи в виде карточек, распределяющиеся по типам: запланированные, текущие и выполненные. Есть возможность создавать несколько досок. Пользователю доступно как в виде десктоп-приложений, так и в веб-интерфейсе [7].

Доски, списки и карточки с задачами могут называться как угодно. Доски могут быть публичными, командными и командными. Пользователь может оформлять их по своему вкусу, производить поиск по доске, делиться ссылкой на нее. Список – объединение карточек с заданиями в одну группу. Их можно перемещать по доскам, сортировать карточки внутри списков. Карточка – непосредственно задача, которую необходимо выполнить. В карточке обычно указываются заголовок, описание, крайний срок исполнения задачи и исполнитель.

Рабочая область программы показана на рисунке 1.3.

К преимуществам можно отнести [7]:

- простой интерфейс;
- бесплатный доступ;
- количество языков интерфейса;
- доступность как на ПК, так и в виде веб-версии;
- различный функционал у сотрудников и администраторов.

Недостатками являются [7]:

- бесплатная версия имеет несколько ограниченный функционал;
- установка плагинов ограничено и зависит от подключенного тарифа;
- подходит в основном малым предприятиям;
- при многозадачных проектах становится сложнее ориентироваться в интерфейсе;
- нет учета времени по задачам;
- нет возможности добавлять описания проектов.

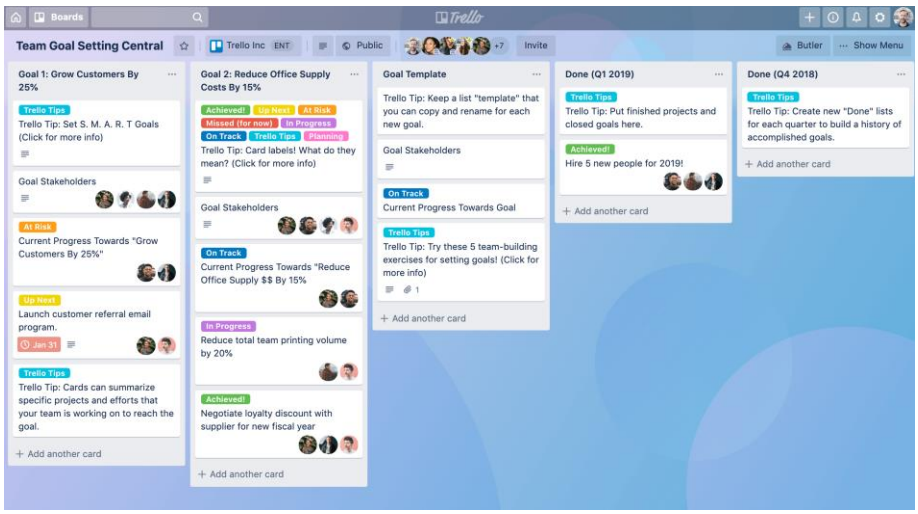


Рисунок 1.3 – Рабочая область Trello

3) Jira – распространенная система управления проектами от Atlassian, хотя изначально проектировалась как система отслеживания ошибок. Используется чаще всего в компаниях, занимающихся разработкой ПО. Имеет веб-интерфейс. Jira позволяет планировать проекты, создавать задачи и назначать их исполнителей, контролировать время работы над ними, выставлять приоритеты [8].

Рабочая область представлена на рисунке 1.4.

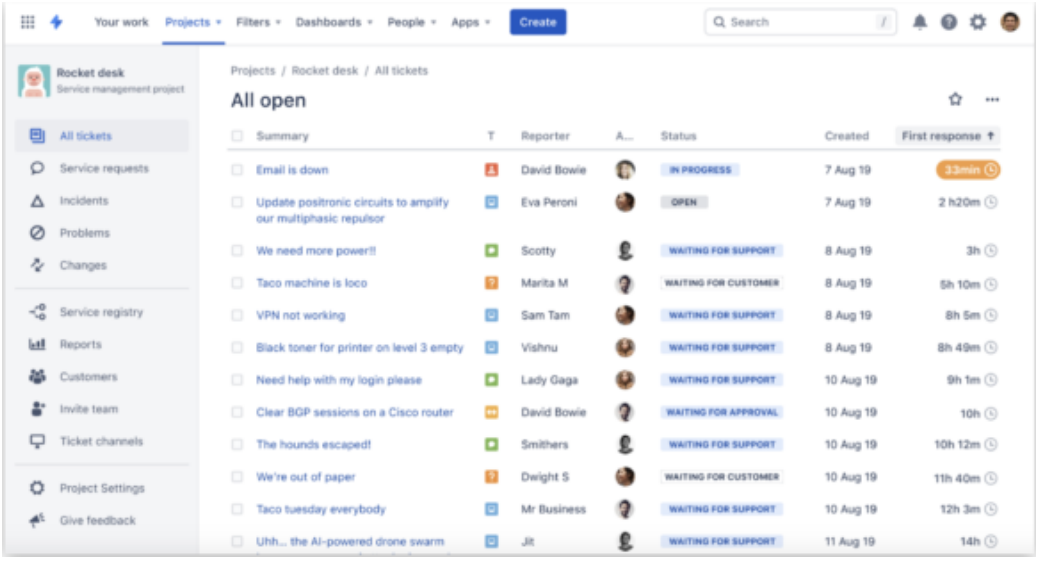


Рисунок 1.4 –Рабочая область Jira

Как и в любой системе управления проектами, для начала необходимо создать проект. Для этого нужно ввести название и ключ проекта, а также выбрать шаблон, по которому будет вестись управление. В проект добавляются задачи, которым назначаются исполнители, а также статус и тип задачи. К статусам относят состояние готовности задачи: «к выполнению», «готово», «в

работе». Выделяют базовый типы задач: «epic»; «story»; «bug». Для удобства Jira предоставляет различные фильтры для поиска задач по проектам, а также инструменты визуализации данных (графики, диаграммы), представленные на рисунке 1.5.

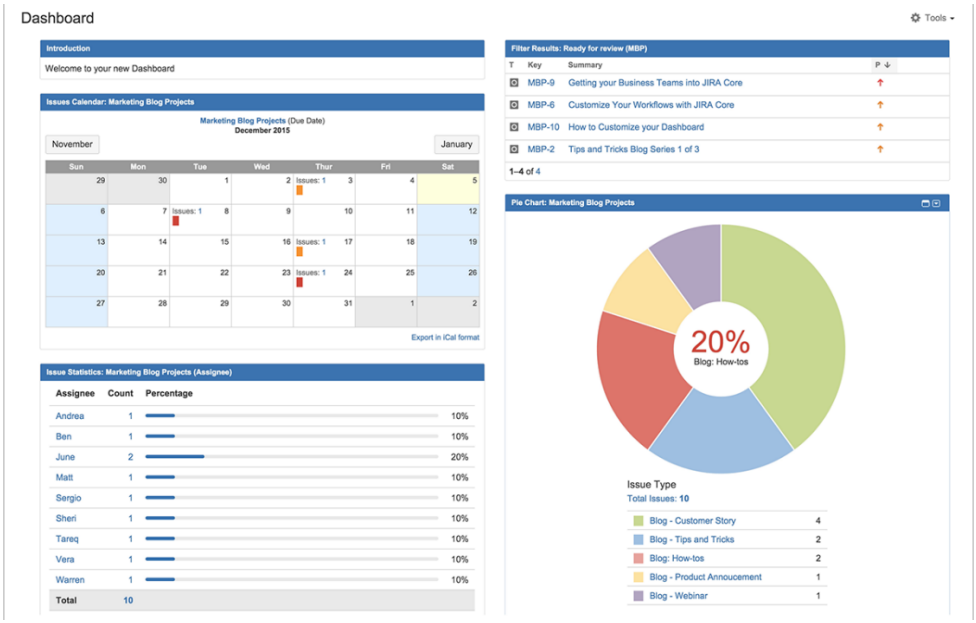


Рисунок 1.5 – Визуализация данных в Jira

Преимущества системы [8]:
- мультиязычность;
- готовые бесплатные расширения;
- гибкое управление проектами;
- имеется конструктор отчетов;
- визуализация данных.

Недостатки системы [8]:
- перегруженный интерфейс;
- сложность настройки и обслуживания;
- бесплатная версия только для небольших команд;
- чем крупнее проект, тем медленнее работа системы.

4) YouTrack – система для управления проектами разработки программного обеспечения от JetBrains. Очень удобен при использовании IDE от JetBrains. Поддерживает методики гибкой разработки, а также работу по свободной методике.

YouTrack позволяет вести контроль не только над текущими, но и над просроченными задачами, объединять задачи в связки, изменять несколько задач сразу автоматически. Помимо этого есть функции построения диаграммы Ганта, оценки и учета затрат, отслеживания прогресса в процентах и составления отчетов [9].

Рабочая область показана на рисунке 1.6.

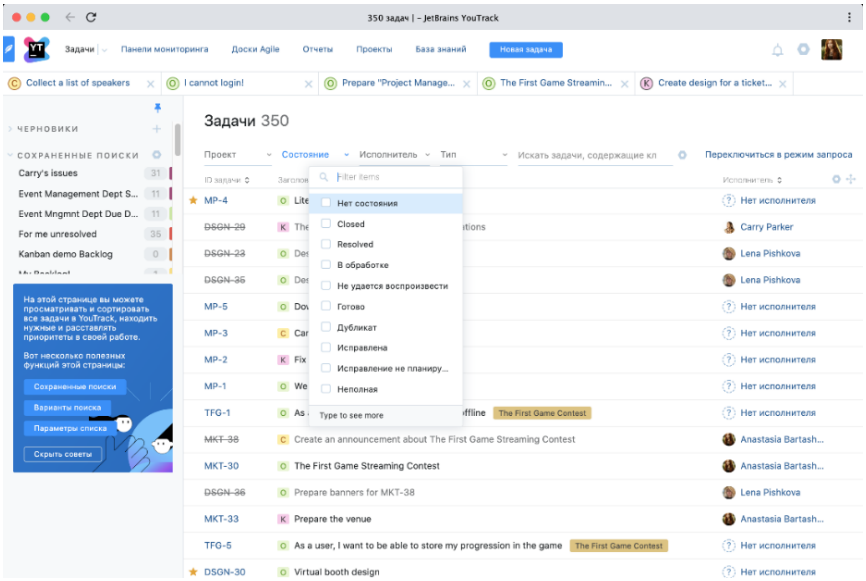


Рисунок 1.6 – Рабочая область YouTrack

Визуализация данных в аналитической панели Dashboard представлена на рисунке 1.7.

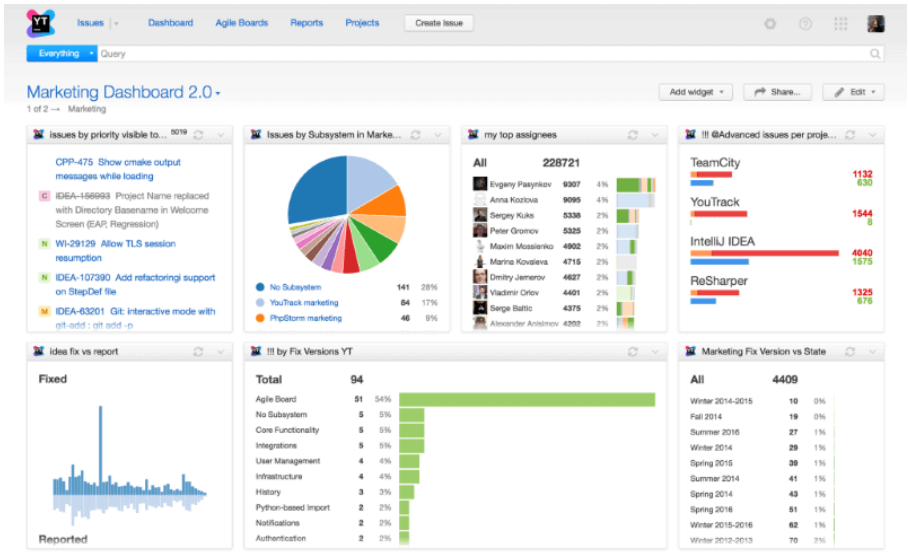


Рисунок 1.7 – YouTrack Dashboard

Преимущества [9]:

- возможность кастомизации;
- составление прогнозов;
- оценка состояния проектов;
- управление временем;

Недостатки [9]:

- подходит для небольших команд, с ростом числа пользователей осуществляется переход на платный тариф;
- интерфейс требует более глубокого изучения;

- с ростом числа проектов система работает медленнее;
- неточность сгенерированных отчетов.

Рассмотренные выше системы имеют общие черты: создание проектов, задач, сортировка задач по статусам и категориям, назначение исполнителей, контроль сотрудников и бизнес-показателей, организация отчетности, сбор статистики.

В ходе анализа существующих систем было выявлено, что основными недостатками, на которые пользователи обращают внимание, являются перегруженный интерфейс и сложность настройки, а также множество функций, которыми пользователи практически не пользуются. Среди преимуществ выделены возможность вести контроль над временем и генерировать отчеты.

Таким образом, разрабатываемая система должна обладать следующим функционалом:

- добавление проектов;
- назначение задач сотрудникам по проектам;
- возможность учета времени работы над задачами;
- деление задач по статусам и приоритетам;
- получение необходимой для работы информации;
- сбор статистики и генерация отчетов.

1.3 Постановка задачи

Для автоматизации процесса распределения задач и работ по проектам, а также сбора и анализа статистики по бизнес-показателям необходимо разработать информационную систему, которая будет представлять настольное приложение и должна решать следующие задачи:

- хранение данных о сотрудниках компании;
- возможность создания новых задач в рамках проектов;
- назначение задач определенным сотрудникам;
- определение сроков и приоритетов задач;
- формирование отчетов.

Функции приложения:

- удобный, понятный и дружественный интерфейс;
- получение информации по задачам и проектам.

Требование к информационному обеспечению:

- необходимо изучить предметную область, а именно: структуру организации; виды проектов, над которыми организация ведет работу; компании, выступающие в роли заказчиков; используемые способы мониторинга за состояниями проектов;
- на основе анализа предметной области выделить сущности и их атрибуты;
- построить логическую модель баз данных: определить первичные ключи у каждой сущности; определить связи между сущностями;

- выбрать СУБД, которая может обеспечить безопасность и целостность данных, а также масштабируема.

- построить реляционную модель баз данных.

Требования к программному обеспечению:

Язык интерфейса приложения – русский, казахский, английский.

Разработанное программное обеспечение (далее ПО) должно быть:

- совместимо с устройствами на базе ОС Windows 10.

Серверная часть должна быть написана с использованием технологии платформы .NET на языке программирования C#. Реализация серверной части должна поддерживать работу MS SQL.

Клиентское приложение должно быть разработано на платформе WPF, UWP или WinForms. Для реализации подсистемы хранения данных должна использоваться СУБД MS SQL Server.

Требования к ПК, на котором будет установлено клиентское рабочее место:

- ОС Windows 10;
- процессор не менее 2 GHz.
- жесткий диск объем не менее 50 Gb.
- оперативная память не менее 2 Gb. Желательно, 4 Gb.
- ПК должен содержать не менее 1 (одного) Ethernet- интерфейса 100 Mbps.

2 РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ

2.1 Исследование объекта автоматизации

Объектом автоматизации являются бизнес-процессы ТОО «SNG IT Group». ТОО «S&G IT GROUP» - компания, занимающаяся автоматизацией и сопровождением бизнес-процессов и предприятий. Деятельность компании осуществляется по следующим направлениям:

- автоматизация бизнес-процессов;
- разработка/поддержка /продвижение web-сайтов;
- IP-телефония;
- продажа программного обеспечения;
- бухгалтерские программы.

Информационная система должна автоматизировать составление отчетов и сбор статистики и представить их в виде, позволяющем получить полное представление о состоянии задач.

Необходимо разработать подсистемы для таких пользователей, как руководитель, менеджер, администратор и сотрудник.

В обязанности менеджера входят:

- просмотр и назначение текущих и выполненных проектов;
- просмотр, редактирование и назначение задач пользователям;
- составление отчетности по проектам;
- формирование графика работ;
- ведение клиентской базы.

В обязанности сотрудника входят:

- выполнение задач в рамках проектов;
- сбор и предоставление первичной информации по состояниям задач;
- составление отчетности по своим задачам и проектам.

В обязанности администратора входят:

- добавление новых пользователей в систему;
- назначение прав доступа пользователям;
- редактирование задач и проектов пользователей.

В обязанности руководителя входят:

- просмотр заключенных договоров с заказчиками;
- анализ отчетности;
- контроль за работой менеджеров и сотрудников;
- контроль за документооборотом.

На предприятии документация ведется вручную, так же как и учет и проверка документов. Анализ и сбор показателей по проектам и задачам не автоматизирован. Учет клиентов и графики работ так же ведутся вручную. Сотрудники получают задания от менеджера и руководителя устно или по электронной почте.

2.2 Проектирование функциональной структуры

В ходе анализа предметной области и организационной структуры предприятия были выделены сущности, которые можно представить в виде подсистем:

- подсистема «Офис» позволяет сотрудникам просматривать контакты других сотрудников в модуле «Контакты»: номер телефона, адрес электронной почты;
- подсистема «Администрирование» должна обеспечить добавление и изменение данных администраторами и менеджерами. Изменяться и добавляться могут данные о сотрудниках, отделах, задачах и проектах, а также добавляться объявления и уведомления;
- подсистема «Управление» должна обеспечить доступ к документам компании, получение оперативной информации о текущих результатах деятельности предприятия, как в целом, так и детализацией по отдельным проектам, планирование деятельности предприятия;
- подсистема «Отчеты» должна отвечать за формирование отчетов по заданным пользователем запросам за определенный период, результаты запросов должны экспортироваться в файлы с форматом *.xls* или *.doc*; отображение динамических и хронологических показателей в виде.
- подсистема «Заказчики». Данная подсистема должна обеспечить хранение данных о заказчиках: название организации-заказчика, адрес, ФИО представителя, контактные номер телефона и почта, заказы (выполненные и текущие).
- подсистема «Задачи» позволяет просматривать задачи, назначенные текущему пользователю, а также добавление новых задач и назначения исполнителе. Помимо этого, в ходе выполнения задачи возможно изменение ее стадии (статуса), приоритета, а также сроков выполнения;
- подсистема «Проекты». отображает проекты, в которых участвует текущий пользователь, позволяет создать новый проект. При создании нового проекта необходимо заполнить основную информацию по проекту, после создания можно добавлять задачи по проекту. Функциональная структура приложения приведена на рисунках 2.1 и 2.2.

2.3 Разработка бизнес модели ИС

2.3.1 Предметная область

Компания «S&G IT Group» занимается разработкой и внедрением программ согласно заказам клиентов. Заказы осуществляются в виде проектов, которыми занимаются сотрудники. Работу сотрудников и ход выполнения работ контролируют менеджеры. Руководитель заключает договоры с заказчиками, контролирует работу сотрудников и документооборот, а также проводит анализ отчетности.

Рисунок 2.1 – Функциональная структура (*а*)

Рисунок 2.2 – Функциональная структура (*б*)

2.3.2 Предпроектное обследование

Функциональная и информационная модели

Главным назначением информационной системы является автоматизация процесса распределения задач и работ по проектам, также сбора, обработки и анализа статистики по бизнес-показателям.

Пользователь создает событие (например, отправка заполненный формы), которое преобразовывается в запрос. Данный запрос отправляется на сервер баз данных, где производится его компиляция. Затем пользователю возвращается результат его запроса (например, сообщение об успешной отправке).

Функциональная модель ИС представлена на рисунке 2.3.

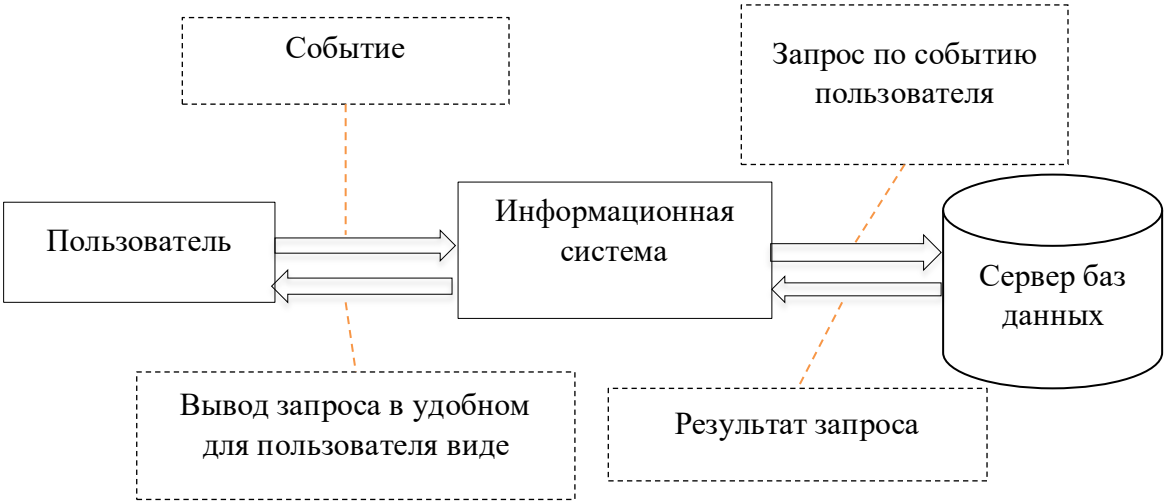


Рисунок 2.3 – Функциональная модель

Информационная модель подсистем представлена на рисунках 2.4-2.7.

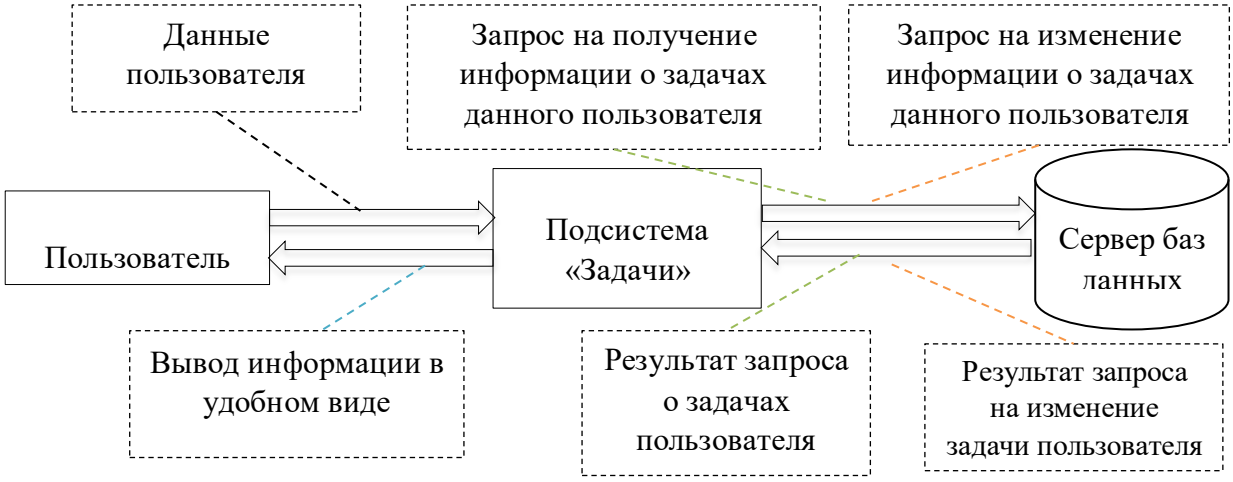


Рисунок 2.4 – Информационная модель подсистемы «Задачи»

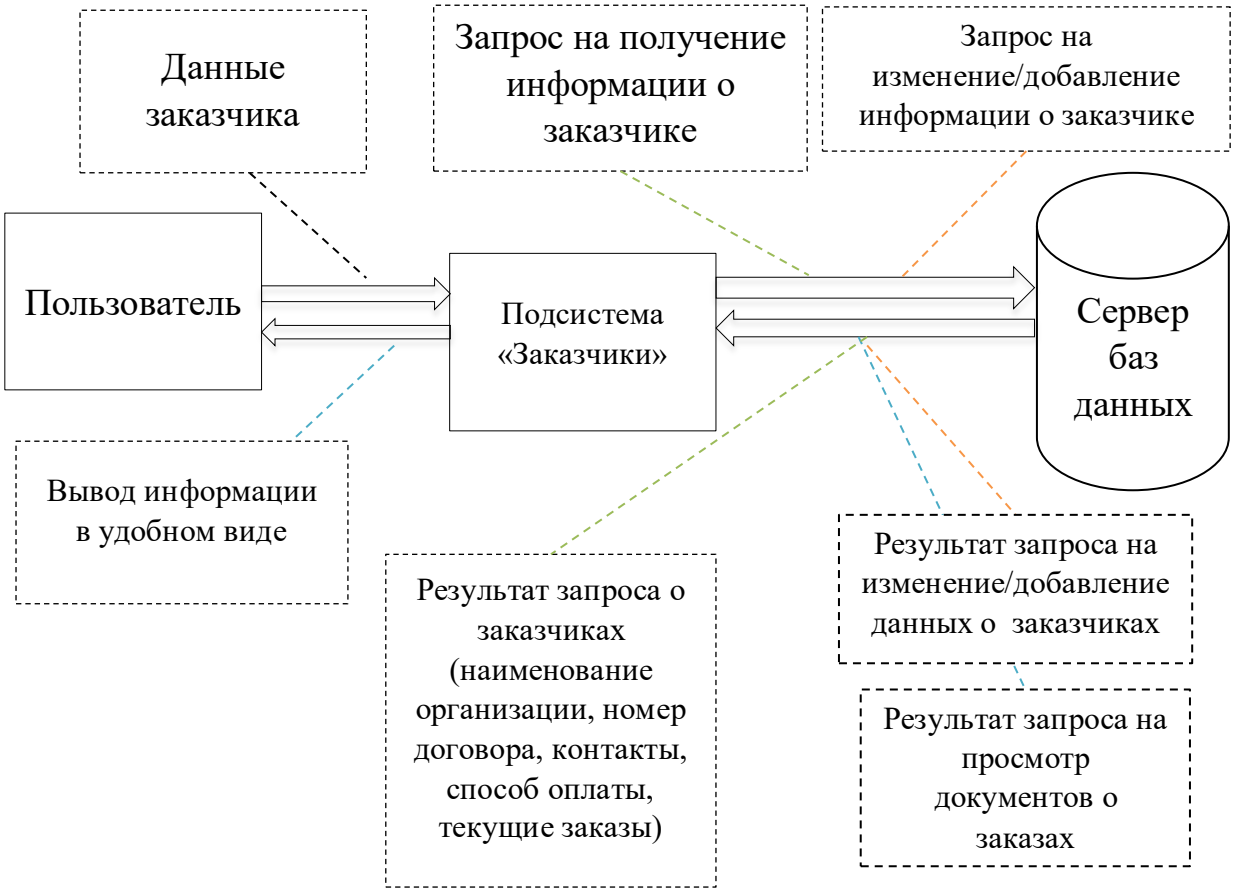


Рисунок 2.5 – Информационная модель подсистемы «Заказчики»

Рисунок 2.6 – Информационная модель подсистемы «Администрирование»

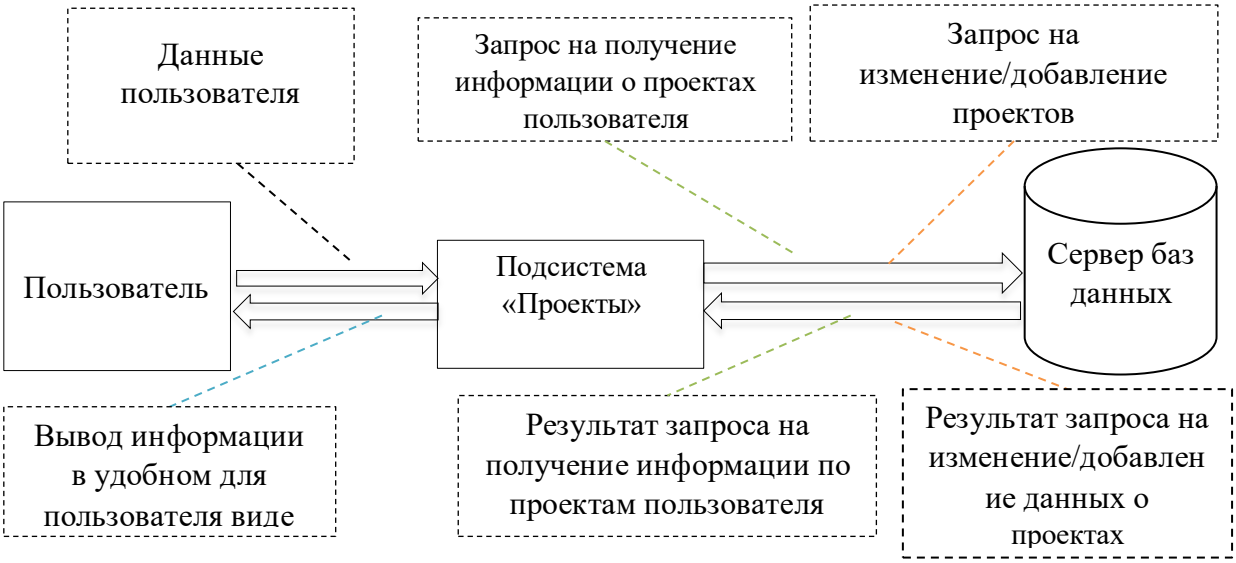


Рисунок 2.7 – Информационная модель подсистемы «Проекты»

2.3.3 Выделение бизнес-процессов

Необходимо выделить бизнес-процессы и представить их в виде схем.

Руководитель заключает с заказчиком договор, на основании которого выполняется проект. Руководитель передает информацию менеджеру. Менеджер назначает задачи по текущему проекту сотрудникам, формирует график работ и ведет отчетность. Сотрудники выполняют поставленные задачи по проекту.

Менеджер назначает задачи сотрудникам. Сотрудники фиксируют сроки и стадии работ и передают информацию менеджеру.

Таким образом, можно выделить бизнес-процессы «Назначение проекта», «Назначение задачи» и «Контроль работы над проектом», схемы которых продемонстрированы на рисунках 2.8-2.10.

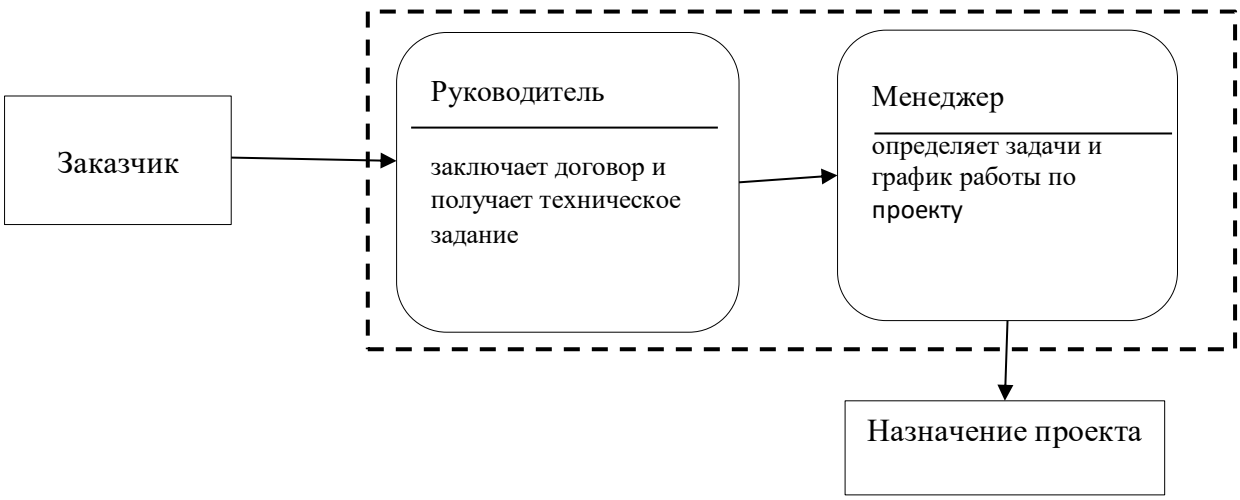


Рисунок 2.8 – Бизнес-процесс «Назначение проекта»

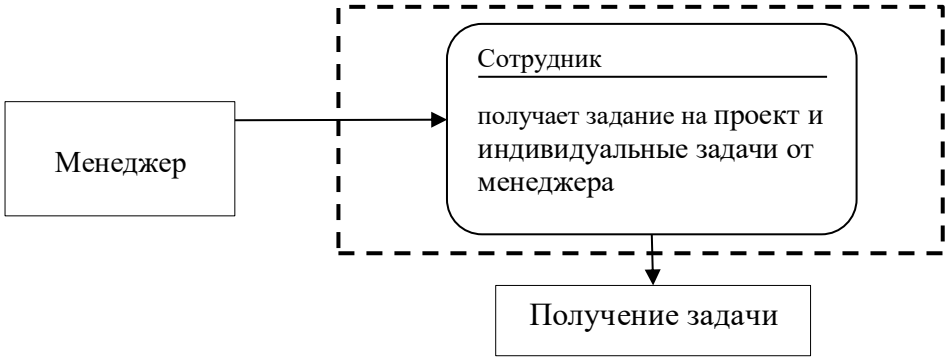


Рисунок 2.9 – Бизнес-процесс «Назначение задачи»

Рисунок 2.10 – Бизнес-процесс «Контроль работы над проектом»

2.3.4 Анализ и оптимизация бизнес-процессов

Для оптимизации бизнес-процессов необходимо создать базу данных, которая обеспечит обмен данными между управляющими и управляемыми компонентами. Оптимизированные бизнес-процессы представлены на рисунках 2.11-2.13.

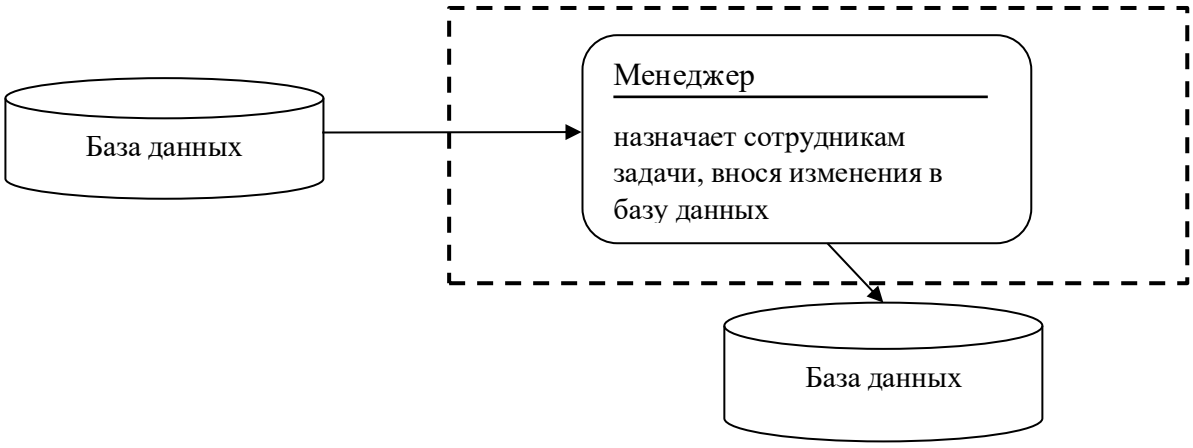


Рисунок 2.11 – Оптимизированный бизнес-процесс «Назначение проекта»

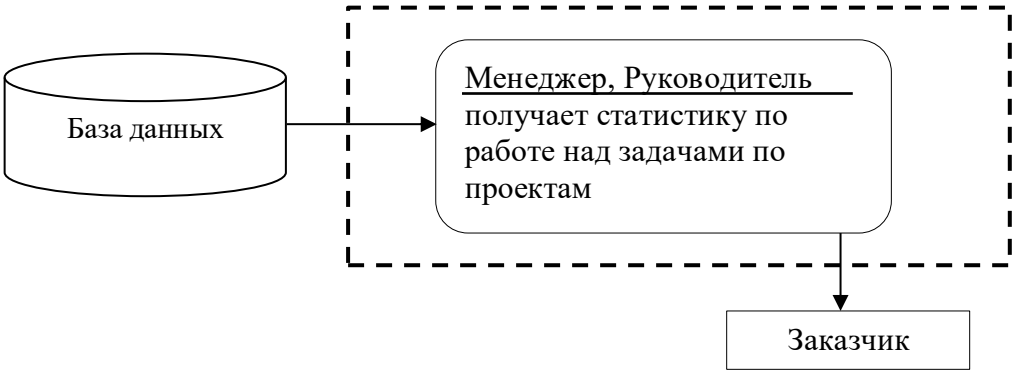


Рисунок 2.12 – Оптимизированный бизнес-процесс «Назначение задачи»

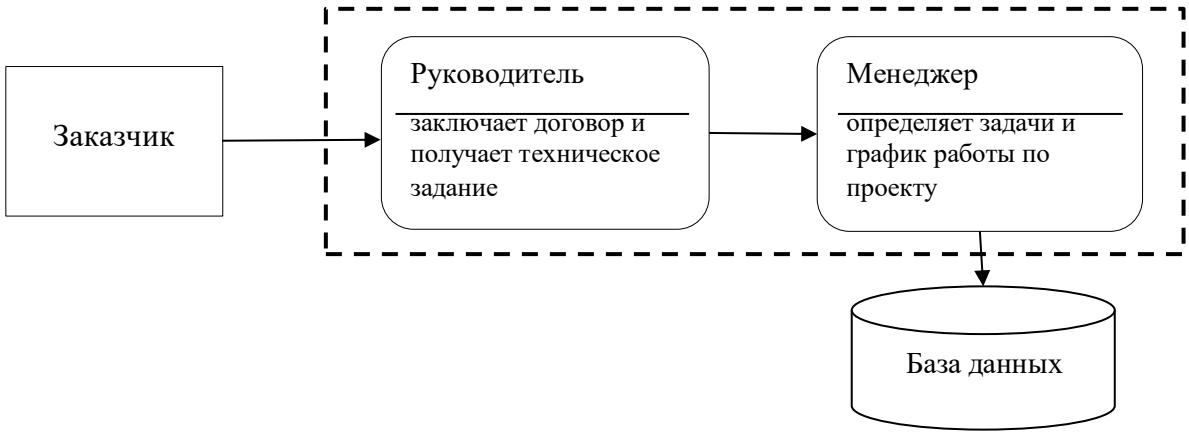


Рисунок 2.13 – Оптимизированный бизнес-процесс «Контроль работы над проектом»

2.4 Описание информационного обеспечения

2.4.1 Обоснование выбора СУБД

База данных – это совокупность данных, организованных по определенным правилам. БД позволяет структурировать, хранить и обрабатывать данные различного типа. [3,10]

Система управления базами данных (СУБД) – это совокупность языковых и программных средств, предназначенных для создания, ведения и использования БД. Эффективность функционирования системы, использующей БД, зависит как от выбора архитектуры БД, так и от выбора СУБД [3,10].

Выбор СУБД зависит от следующих факторов:

- надежность;
- стоимость;
- быстрота и стабильность работы;
- совместимость со средствами разработки.

Так как разрабатывается настольное приложение для информационной системы, то такие веб-ориентированные СУБД (например, MySQL) не будут рассмотрены в таблице 2.1.

Таблица 2.1 – Популярные системы управления данными

Название	Разработчик	Лицензия	Написана на
Oracle	Oracle Corporation	Проприетарная	Assembly, C, C++
Microsoft SQL Server	Microsoft Corporation	Проприетарная	C, C++
PostgreSQL	PostgreSQL Global Development Group	Лицензия PostgreSQL (бесплатное ПО с открытым исходным кодом)	C
MongoDB	MongoDB Inc.	Различные варианты лицензирования	C++, C, JavaScript
DB2	IBM	Проприетарная EULA	Assembly, C, C++
Microsoft Access	Microsoft Corporation	Пробное ПО	
Redis	Salvatore Sanfilippo	Лицензия BSD	ANSI C

Из перечисленных выше СУБД более детально следует рассмотреть Oracle, Microsoft SQL Server и PostgreSQL.

Oracle Database – практически самая популярная СУБД у разработчиков, так как является эталоном надежности и безопасности [11].

Достоинства этой СУБД заключаются в следующем:

- высокая надежность и безопасность;
- возможность размещения на нескольких серверах;
- возможность работы с облачным средами;
- понятная документация;

К недостаткам относятся:

- стоимость;
- требования к техническому обеспечению.

PostgreSQL – бесплатная СУБД, часто используемая для ведения баз данных веб-сайтов, однако ее можно использовать и для настольных приложений. Так как эта СУБД была разработана давно, то уже успела развиться и может использоваться на различных платформах [3, 11].

К достоинствам можно отнести:

- масштабируемость;
- возможность использования предопределенных функций;
- доступность ряда интерфейсов;

- возможность расширения функционала;
- стоимость (бесплатно).

Недостатки PostgreSQL:
- нечеткая документация;
- сложная конфигурация;
- невысокая производительность;
- нестабильность скорости работы.

Microsoft SQL Server – СУБД от компании Microsoft, которая привязана к ОС Windows. Движок может работать и на локальных, и на облачных серверах [11].

Достоинства данной СУБД:
- быстрота и стабильность работы;
- возможность использования на разных типах серверов;
- совместимость со средствами разработки;
- возможность регулирования производительности;
- простота использования;
- четкая документация.

Недостатки:
- стоимость;
- требования к техническому обеспечению;
- SQL Server может потреблять все доступные ресурсы.

На основании данных сравнительного анализа различных СУБД для разрабатываемой информационной системы была выбрана СУБД Microsoft SQL Server.

Система управления базами данных Microsoft SQL Server удобна, так как в организации уже используются продукты Microsoft.

SQL Server легко масштабируется и содержит усовершенствованный процессор запросов, который может обработать сложные запросы баз данных больших объемов. С точки зрения администрирования данная СУБД проста, так как выполнение многих рутинных задач автоматизировано: управление памятью, блокировками, настройка профилей пользователей [11].

В состав Microsoft SQL Server входит мощный язык работы с данными Transact SQL – расширение стандартного SQL, которое рассматривается как предпочтительный диалект SQL. Transact-SQL поддерживает такие объекты БД, как хранимые процедуры, триггеры, представления, поддержка целостности и т.д. [12].

С точки зрения безопасности данная СУБД обеспечивает авторизацию пользователей либо на основе собственного списка пользователей, либо на основе базы пользователей Windows, что является более эффективным вариантом.

Одним из недостатков является стоимость SQL Server, однако Microsoft предлагает бесплатную версию данной СУБД - Microsoft SQL Server Express. Ее можно бесплатно загружать, распространять и использовать. Она также подходит для небольших организаций [12].

2.4.2 Описание структур таблиц

База данных для информационной состоит из 15 таблиц: Сотрудник (Employees), Должности (Positions), Роли(Roles) Отделы (Departments), Квалификация (Qualification), Сотрудник-Квалификация (Employee-Qualification), Образование (Education), Заказчик (Clients), Тип заказчика (Clients type), Договор (Contract), Проект (Order), Тип заказа (Order type), Задача (Task), Статус задачи (Task status), Приоритет (Priority), Журнал проектов (Project_log), Журнал задач (Task_log).

Таблица «Сотрудник» (Employee) содержит уникальный номер сотрудника, его персональные данные, информацию о должности и квалификации. Структура показана в таблице 2.2

Таблица 2.2 – Сотрудник

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Employee	INTEGER		Первичный ключ. Идентификатор сотрудника
2	Login	VARCHAR	30	Хранит сведения о логине
3	Password	VARCHAR	30	Хранит сведения о пароле
4	Surname	VARCHAR	30	Хранит сведения о фамилии
5	Name	VARCHAR	30	Хранит сведения об имени
6	Patronymic	VARCHAR	30	Хранит сведения об отчестве
7	Position	INTEGER		Хранит сведения об идентификаторе занимаемой должности

Таблица «Должность», представленная в таблице 2.3, хранит в себе сведения о коде должности и ее названии.

Таблица 2.3 – Должность

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Position	INTEGER		Первичный ключ. Идентификатор должности
2	Position_name	VARCHAR	50	Хранит сведения о названии должности

Таблица «Квалификация», содержит код квалификации и название. Ее структура приведена в таблице 2.4.

Таблица 2.4 – Квалификация

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Qualification	INTEGER		Первичный ключ. Идентификатор квалификации
2	Qualification_name	VARCHAR	70	Хранит сведения о названии квалификации

Так как сотрудник может иметь несколько квалификаций, равно как и одну квалификацию могут иметь несколько сотрудников, то связь «многие-ко-многим» будет реализована в виде таблицы пересечения «Сотрудник_Квалификация», структура которой представлена в таблице 2.5.

Таблица 2.5 – Сотрудник_Квалификация

№	Наименование поля	Тип поля	Размер	Назначение поля
1	Qualification_ID	INTEGER		Идентификатор квалификации
2	Employee_ID	INTEGER		Идентификатор сотрудника

Структура таблицы «Образование» приведена в таблице 2.6.

Таблица 2.6 – Образование

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Education_type	INTEGER		Первичный ключ. Идентификатор образования
2	Education_type	VARCHAR	50	Хранит сведения о типе образования

В таблице «Заказчик» хранятся такие сведения, как БИН (код заказчика), название организации, юридический адрес организации, тип клиента, ФИО представителя, который следит за ходом работы над заказом, и контактные данные (номер телефона и электронная почта). Структура приведена в таблице 2.7.

Таблица 2.7 – Заказчик

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Client	BIGINT		Первичный ключ. Идентификатор заказчика
2	Name_Oraganization	VARCHAR	100	Хранит сведения о названии организации
3	Address	VARCHAR	100	Хранит сведения об адресе организации
4	Client_type	INTEGER		Хранит сведения об идентификаторе типа клиента
5	CP_surname	VARCHAR	70	Хранит сведения о фамилии контактного лица
6	CP_name	VARCHAR	30	Хранит сведения об имени
7	CP_patronymic	VARCHAR	30	Хранит сведения об отчестве контактного лица
8	CP_email	VARCHAR	30	Хранит сведения об электронном адресе контактного лица
9	CP_phone_number	VARCHAR	15	Хранит сведения о номере телефона представителя

В таблице «Тип заказчика» содержатся следующие сведения: код типа, название, скидка в процентах. Структура приведена в таблице 2.8.

Таблица 2.8 – Типы заказчиков

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Client_Type	INTEGER		Первичный ключ. Идентификатор типа заказчика
2	Client_Type_name	VARCHAR	30	Хранит сведения о названии типа заказчика
3	Discount_percentage	INTEGER		Хранит сведения о скидке для данного типа заказчика

В таблице «Договор» содержатся сведения о номере договора, БИН клиента и сумме. Структура представлена в таблице 2.9.

Таблица 2.9 – Договор

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Contract	INTEGER		Первичный ключ. Идентификатор договора
2	ID_Client	BIGINT		Хранит сведения об идентификаторе заказчика
3	Contract_sum	MONEY		Хранит сведения о сумме договора

Таблица «Проекты» содержит сведения о коде проекта, коде заказчика, номере договора, по которому выполняется проект, даты начала и окончания, статус проекта, дата обновления статуса. Структура приведена в таблице 2.10.

Таблица 2.10 – Проекты

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Project	INTEGER		Первичный ключ. Идентификатор проекта
2	Client_ID	BIGINT		Хранит сведения об идентификаторе заказчика
3	Contract_ID	INTEGER		Хранит сведения об идентификаторе договора
4	Project_start_date	DATE		Хранит сведения о дате начала проекта
5	Project_end_date	DATE		Хранит сведения о дате окончания проекта
6	Project_status_ID	INTEGER		Хранит сведения об идентификаторе статуса проекта
7	Status_updated_date	DATE		Хранит сведения о дате обновления статуса проекта

Сведения о задачах хранятся в таблице «Задача». Это номер задачи, код проекта, описание задачи, даты начала и окончания, код статуса и дата его обновления, приоритет задач. Структура таблицы представлена в таблице 2.11.

Таблица 2.11 – Задача

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Task	INTEGER		Первичный ключ. Идентификатор задачи
2	Project_ID	INTEGER		Хранит сведения об идентификаторе заказа
3	Task_desc	VARCHAR(200)		Хранит сведения о комментарии к задаче
4	Task_start	DATE		Хранит сведения о дате начала выполнения задачи
5	Task_end	DATE		Хранит сведения о сроке выполнения задачи
6	Task_status_ID	INTEGER		Хранит сведения об идентификаторе статуса задачи
7	Task_status_upd_date	DATE		Хранит сведения о дате обновления статуса задачи
8	Task_priority	INTEGER		Хранит сведения об идентификаторе приоритетности задачи

Таблица «Отделы» хранит номер отдела и его название. Структура представлена в таблице 2.12.

Таблица 2.12 – Отделы

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Department	INTEGER		Первичный ключ. Идентификатор отдела
2	Department_Name	VARCHAR	50	Хранит сведения о названии

Заказы можно классифицировать по различным признакам, следовательно, выделяется таблица «Тип заказа», которая должна хранить информацию об идентификаторе типа, названии, минимальной стоимости. Структура представлена в таблице 2.13.

Таблица 2.13 – Тип заказа

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_order_type	INTEGER		Первичный ключ. Идентификатор типа заказа
2	Order_type_name	VARCHAR	50	Хранит сведения о названии типа заказа
3	Min_cost	MONEY		Хранит информацию о минимальной стоимости заказа определенного типа

Сведения о статусах задачи хранятся в таблице «Статус задачи», в которой указываются идентификатор статуса, название и описание. Структура представлена в таблице 2.14.

Таблица 2.14 – Статус задачи

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_status	INTEGER		Первичный ключ. Идентификатор статуса задачи
2	Status_name	VARCHAR	15	Хранит сведения о названии статуса задачи
3	Status_desc	VARCHAR	100	Хранит сведения о пояснении статуса

Таблица «Приоритет», структура которой приведена в таблице 2.15, содержит данные об идентификаторе приоритета, его названии и описании.

Таблица 2.15 – Приоритет

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Priority	INTEGER		Хранит сведения об идентификаторе приоритетности
2	Priority_name	VARCHAR	15	Хранит сведения о названии степени важности
3	Priority_desc	VARCHAR	100	Хранит сведения о комментарии к степени важности

В таблице «Роль пользователя» хранятся сведения об идентификаторе роли, названии и описании. Структура представлена в таблице 2.16.

Таблица 2.16 – Роль пользователя

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_Role	INTEGER		Первичный ключ. Идентификатор роли пользователя
2	Role_name	VARCHAR	15	Хранит сведения о названии роли
3	Role_desc	VARCHAR	50	Хранит сведения об описании роли пользователя

В таблицу «Журнал проектов» вносятся сведения, с помощью которых можно собрать хронологические показатели, т.е. историю, проектов: номер

записи, номер проекта, идентификатор статуса проекта, дата изменения статуса. Структура представлена в таблице 2.17.

Таблица 2.17 – Журнал проектов

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_pl	INTEGER		Первичный ключ, номер записи
2	Project_ID	INTEGER		Хранит сведения об идентификаторе проекта
3	Status_ID	INTEGER		Хранит сведения о статусе проекта
4	Date_set	DATETIME		Хранит сведения о дате назначения статуса проекта

Таблица «Журнал задач» содержит вносятся сведения, с помощью которых можно собрать хронологические показатели, т.е. историю, задач: номер записи, номер проекта, номер задачи, идентификатор статуса, дата изменения статуса. Структура описана в таблице 2.18.

Таблица 2.18 – Журнал задач

№	Наименование поля	Тип поля	Размер	Назначение поля
1	ID_tl	INTEGER		Первичный ключ, номер записи
2	Project_ID	INTEGER		Хранит сведения об идентификаторе проекта
3	Task_ID	INTEGER		Хранит сведения об идентификаторе задачи
4	Status_ID	INTEGER		Хранит сведения о статусе задачи
5	Date_set	DATETIME		Хранит сведения о дате назначения статуса задачи

Логическая модель – это описание предметной области с выделением в ней сущностей и атрибутов, а так же ограничения и отношения между ними. Эта модель не привязана к конкретной СУБД [13, 14].

Логическая модель базы данных построена в программе Data Modeler. Логическая модель с выделенными сущностями и связями между ними представлена на рисунке 2.14.

Для преобразования логической модели в реляционную существует функция Engineer to Relational Model. Реляционная модель представлена на рисунке 2.15. В данной модели добавились таблицы пересечения, которые осуществляют связь «многие ко многим».

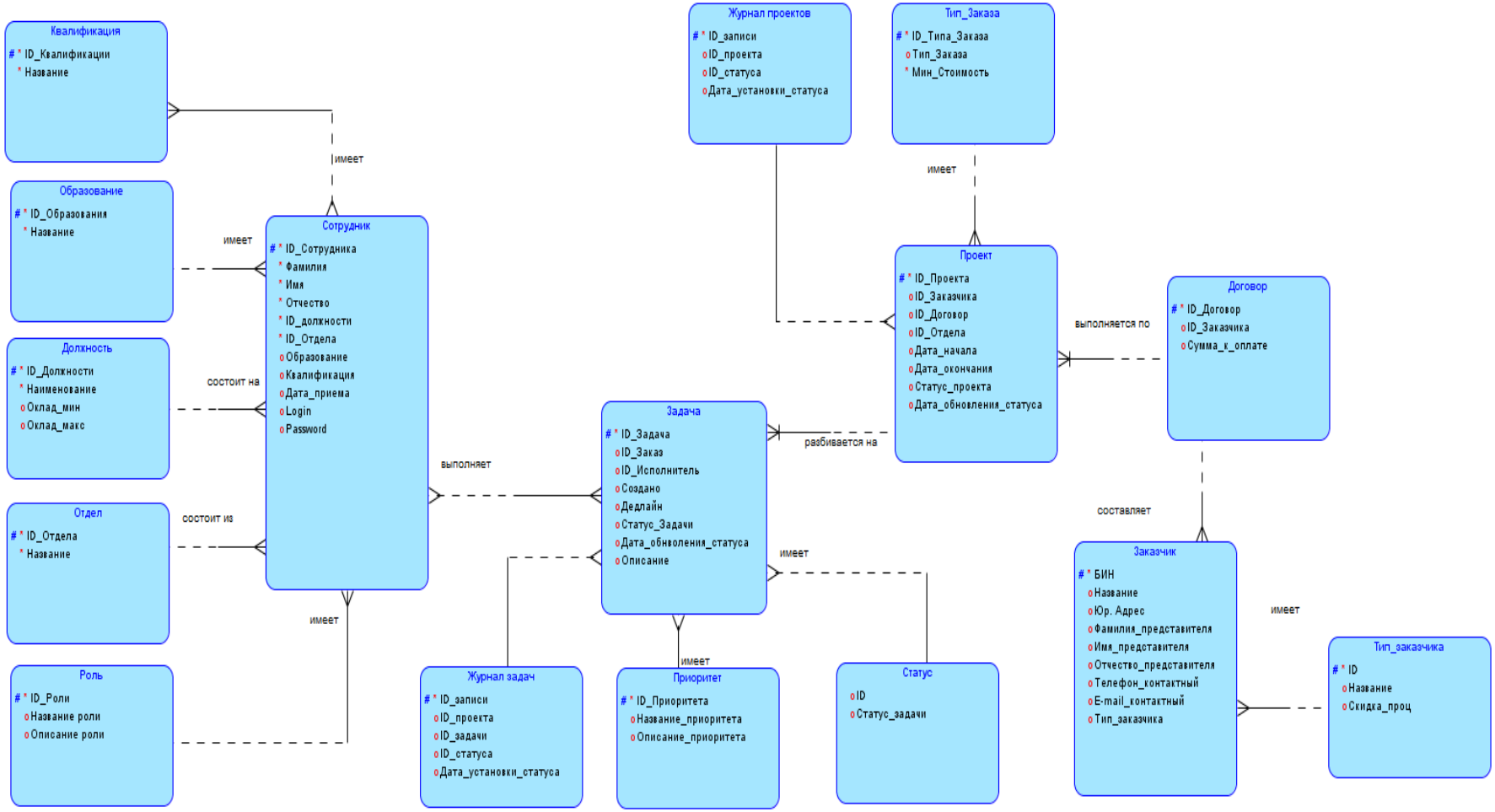


Рисунок 2.14 – Логическая модель

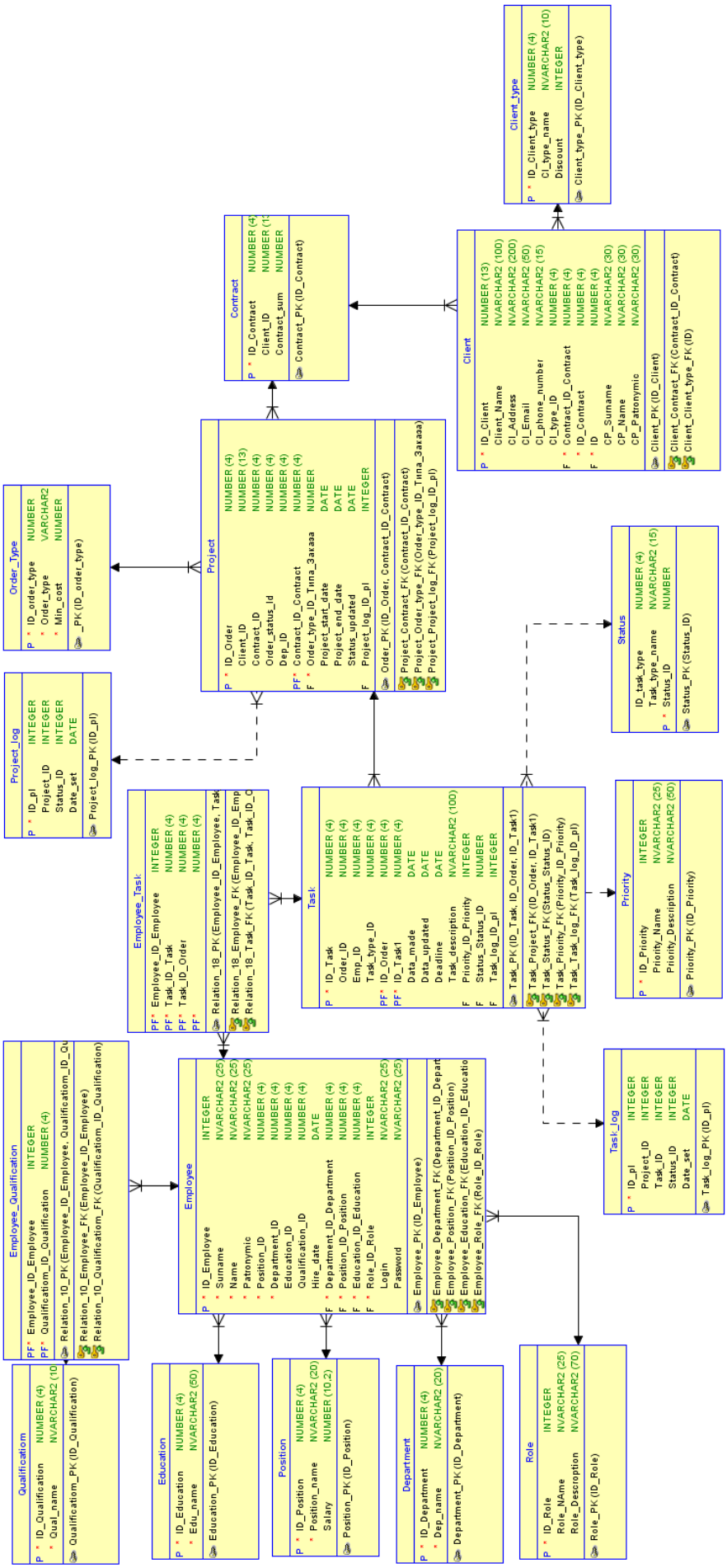


Рисунок 2.15 – Реляционная модель

2.4.3 Хранимые процедуры и триггеры

В базе данных проекта разработаны хранимые процедуры и триггеры, которые существенно упрощают выполнение часто используемых запросов. Процедуры, хранимые на сервере, вызываются из кода программы [14].

Процедура «*addClient*» принимает несколько аргументов, которые затем используются в запросе на вставку данных в таблицу «Client». Структура процедуры представлена на рисунке 2.16.

```
create procedure [dbo].[addClient]
    @bin bigint,
    @company_name varchar(70),
    @address varchar(150),
    @clientType int,
    @surname varchar(30),
    @name varchar(30),
    @patronymic varchar(30),
    @email varchar(30),
    @phone varchar(30)
AS
    begin
        insert into Client values (@bin, @company_name, @address, @surname, @name, @patronymic, @email, @phone, @clientType)
    end
```

Рисунок 2.16 – Структура процедуры «*addClient*»

Процедура «*taskTable*» принимает в качестве аргумента логин пользователя и в соответствии с ним осуществляет выборку данных о задачах, которые отображаются в виде таблицы в программе. Структура данной процедуры показана на рисунке 2.17.

```
CREATE procedure [dbo].[taskTable]
    @ulogin varchar(30)
as
begin
select Task_desc as Название
    , convert(varchar, Task_start, 4) as Начало
    , case when Task_end IS NULL  then '-' else convert(varchar, Task_end, 4) end Окончание
    , ts.Status_name Статус
    , tp.Priority_name Приоритет
    , p.Project_name Проект
from Task t
inner join
    TaskStatus ts on t.Task_status_ID=ts.ID_Status
inner join
    TaskPriority tp on t.Task_priority=tp.ID_Priority
inner join Employee_Task et on t.ID_Task=et.Task_ID
inner join Employee e on et.Employee_ID = e.ID_Employee
inner join Project p on t.Project_ID = p.ID_Project
where et.Employee_ID = (select ID_Employee from Employee where Ulogin=@ulogin);
end
```

Рисунок 2.17 - Структура процедуры «*taskTable*»

Процедура «*taskChartsByUser*» используется для заполнения данными диаграмм аналитических панелей в приложении. В качестве аргументов принимаются логин пользователя и идентификатор проекта. Затем создается временная таблица, в которую помещаются данные о принятых, текущих и выполненных задачах проекта. Структура процедуры представлена на рисунке 2.18.

```
ALTER procedure [dbo].[taskChartsByUser]
    @id_project int,
    @ulogin varchar(30)
as
begin
create table #tbl2 (    ttype varchar(20),  amount int)

insert into #tbl2 (amount, ttype)
VALUES
(  (select count(Task_ID) from Employee_Task where Project_ID=@id_project) , 'Всего' )
,
    ( (select count(Task_ID) from Employee_Task
    inner join Task on Employee_Task.Task_ID = Task.ID_Task
    inner join Employee on Employee_Task.Employee_ID = Employee.ID_Employee
    where [Task].[Task_status_ID] = 1 and Task.Project_ID=@id_project
    and Employee_Task.Employee_ID = (select Employee_ID from Employee where Ulogin = @ulogin) ), 'Принято' )
,
    ( (select count(Task_ID) from Employee_Task
    inner join Task on Employee_Task.Task_ID = Task.ID_Task
    where [Task].[Task_status_ID] = 2 and Task.Project_ID=@id_project
    and Employee_Task.Employee_ID = (select Employee_ID from Employee where Ulogin = @ulogin) ), 'В процессе')
 ,
    ( (select count(Task_ID) from Employee_Task
    inner join Task on Employee_Task.Task_ID = Task.ID_Task
    where [Task].[Task_status_ID] = 3 and Task.Project_ID=@id_project
    and Employee_Task.Employee_ID = (select Employee_ID from Employee where Ulogin = @ulogin) ), 'Выполнено' )

select * from #tbl2 where ttype <> 'Всего'
drop table #tbl2

end
```

Рисунок 2.18 – Структура процедуры «*taskChartsByUser*»

Триггер «*taskLog*» вызывается при обновлении таблицы «Task». Если таблица задач обновлена, то в таблицу журналирования «Task_log» вносятся данные об идентификаторе задачи, новом статусе и дате изменения. Таким образом обеспечивается отслеживание хронологии выполнения задач. Структура триггера представлена на рисунке 2.19.

```
create trigger [dbo].[taskLog]
on [dbo].[Task] after update
as
insert into Task_log (Task_ID, Status_ID, Date_set) values
((select i.ID_Task from inserted i join deleted d on (i.ID_Task = d.ID_Task)),
(select i.Task_status_ID from inserted i ),GETDATE())
insert into Log values (CONCAT('Запись в журнале задач обновлена ', convert(varchar(20),GETDATE(),2)))
```

Рисунок 2.19 – Структура триггера «*taskLog*»

Триггер «*projectLog*» схож с предыдущим, он используется для заполнения таблицы журналирования «Project_log». Структура показана на рисунке 2.20.

```
create trigger [dbo].[projectLog]
on [dbo].[Project] after update
as
insert into Project_Log( Project_ID, Status_ID, Date_set)
values
((select ID_Project from inserted),
(select p.Project_status_ID from inserted p join deleted d on p.ID_Project=d.ID_Project ),GETDATE() )
insert into Log values (CONCAT('Запись в журнале проектов обнволена ', convert(varchar(20),GETDATE(),2)))
```

Рисунок 2.20 – Структура триггера «*projectLog*»

2.4.4 Диаграммы UML

Диаграммы UML позволяют наглядно представить разработчику свое видение системы [15]. Существует несколько видов диаграмм UML, но в обязательном порядке используются следующие:

- диаграммы прецедентов (вариантов использования);
- диаграммы последовательностей;
- диаграммы сотрудничества (кооперативные).

Построение диаграммы прецедентов

Диаграмма прецедентов отображает взаимодействие пользователя с системой. Для построения такой диаграммы необходимо выделить блоки использования, которые представляют собой функции системы для пользователя.

Диаграмма состоит из элементов:

- акторы – собственно пользователи, взаимодействующие с системой;
- варианты использования, то есть функции системы;
- связи между акторами и вариантами использования.

На рисунках 2.21-2.24 представлены диаграммы вариантов использования для пользователей Сотрудник, Менеджер, Администратор и Руководитель.

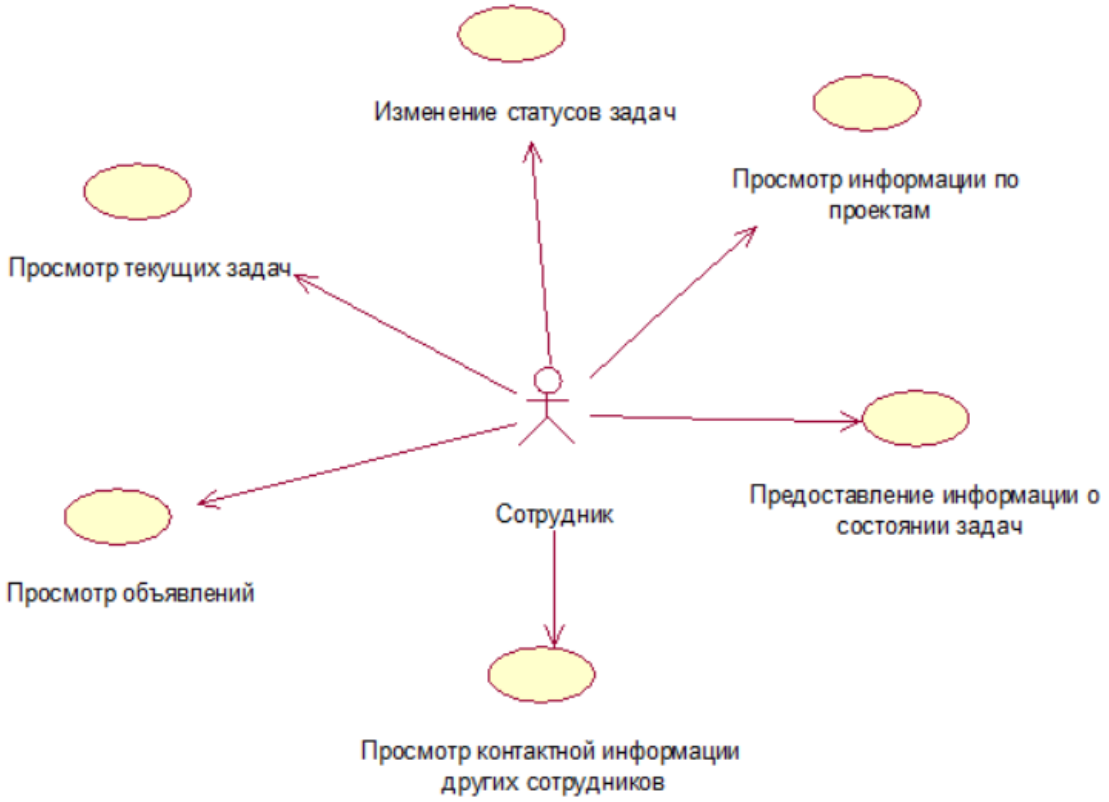


Рисунок 2.21 – Диаграмма вариантов использования актора «Сотрудник»

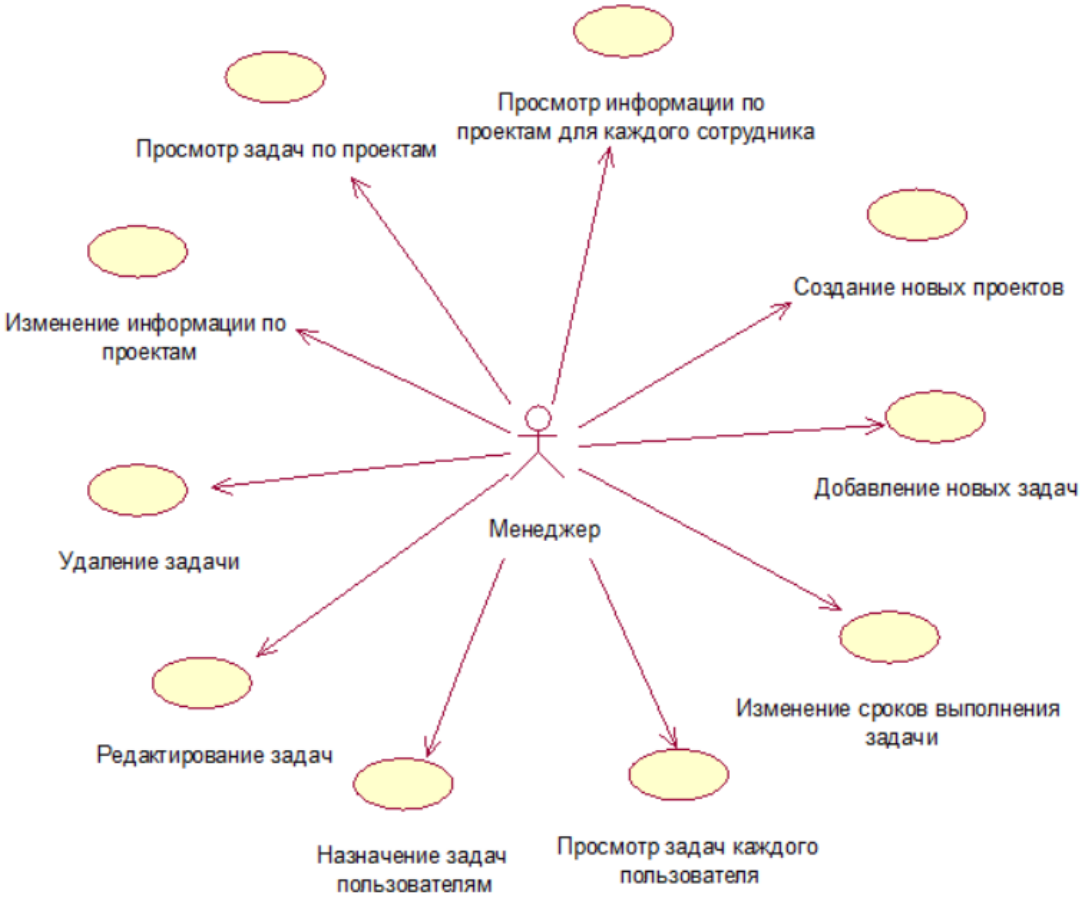


Рисунок 2.22 – Диаграмма вариантов использования актора «Менеджер»

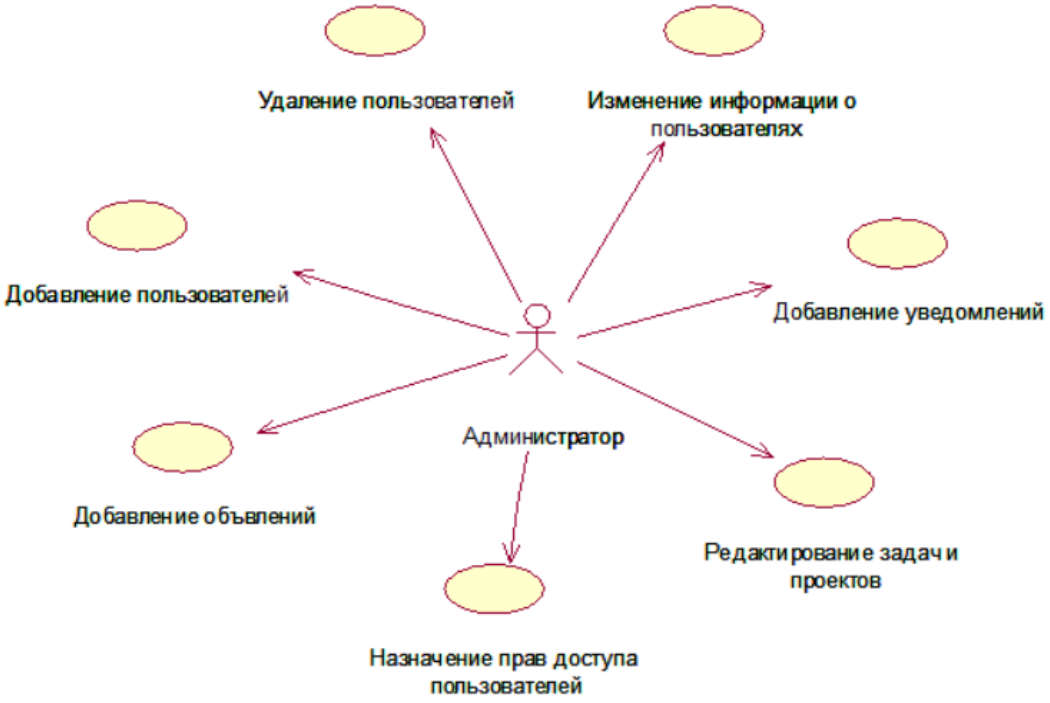


Рисунок 2.23 – Диаграмма вариантов использования актора «Администратор»

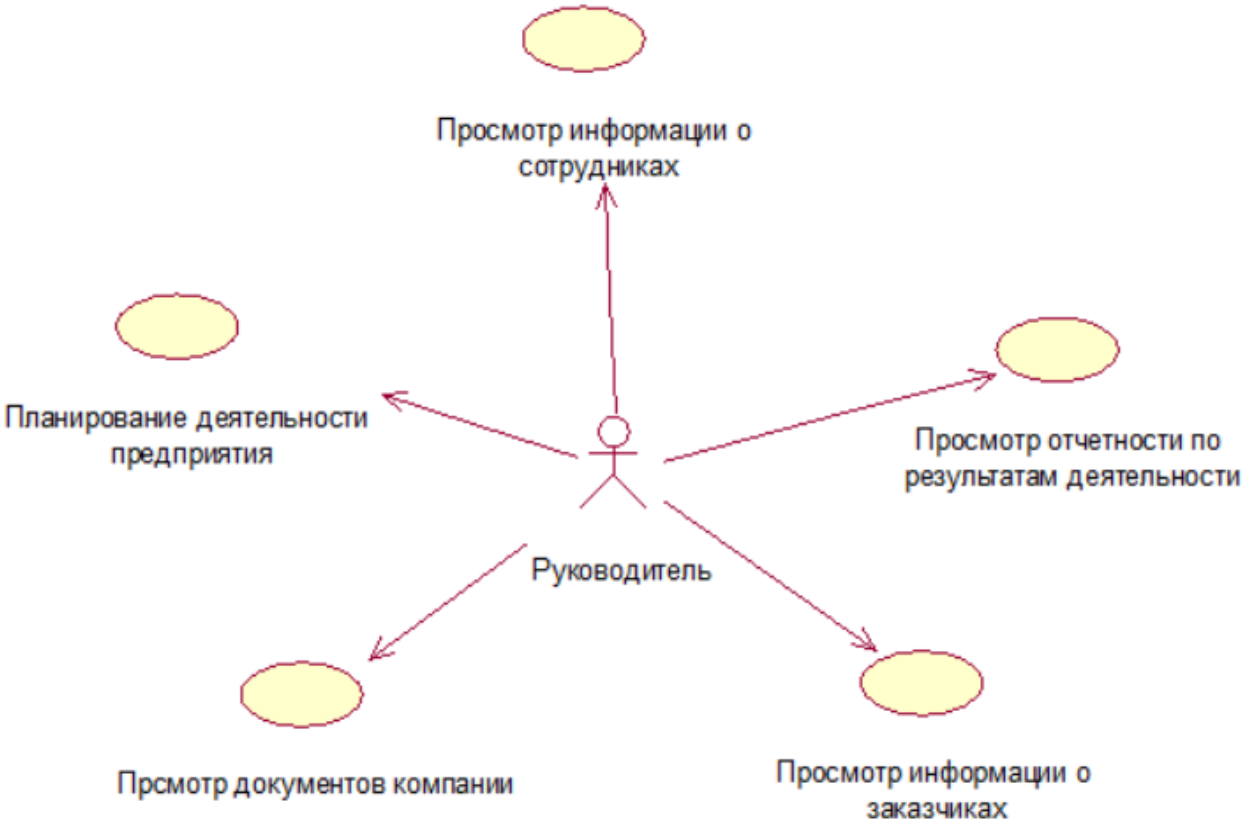


Рисунок 2.24 – Диаграмма вариантов использования актора «Руководитель»

На рисунках 2.25-2.26 представлены диаграммы вариантов использования для пользователя Менеджер в подсистемах «Проекты» и «Задачи».

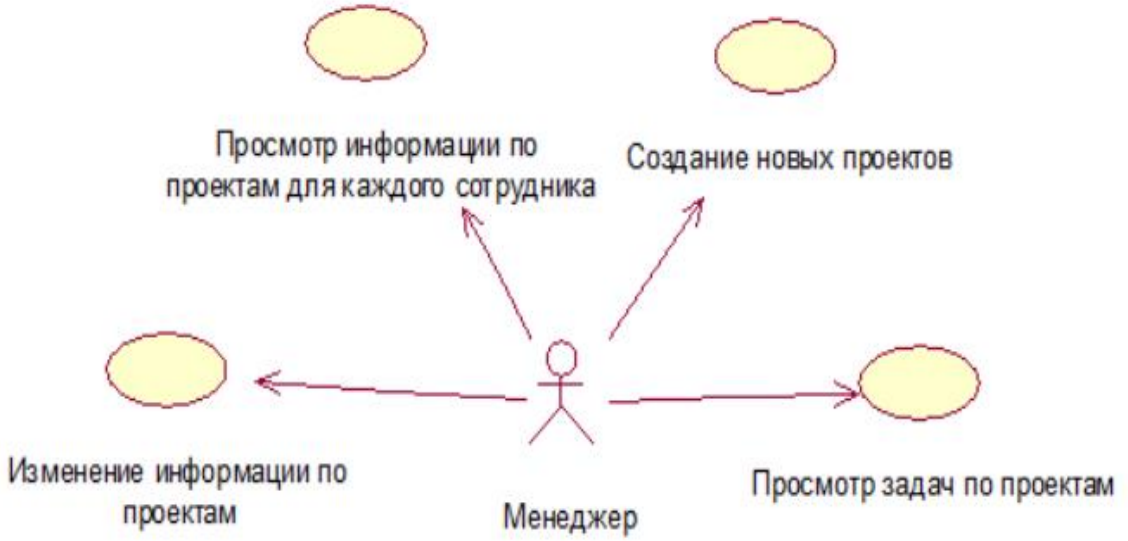


Рисунок 2.25 – Диаграмма вариантов использования для актора «Менеджер» в подсистеме «Проекты»

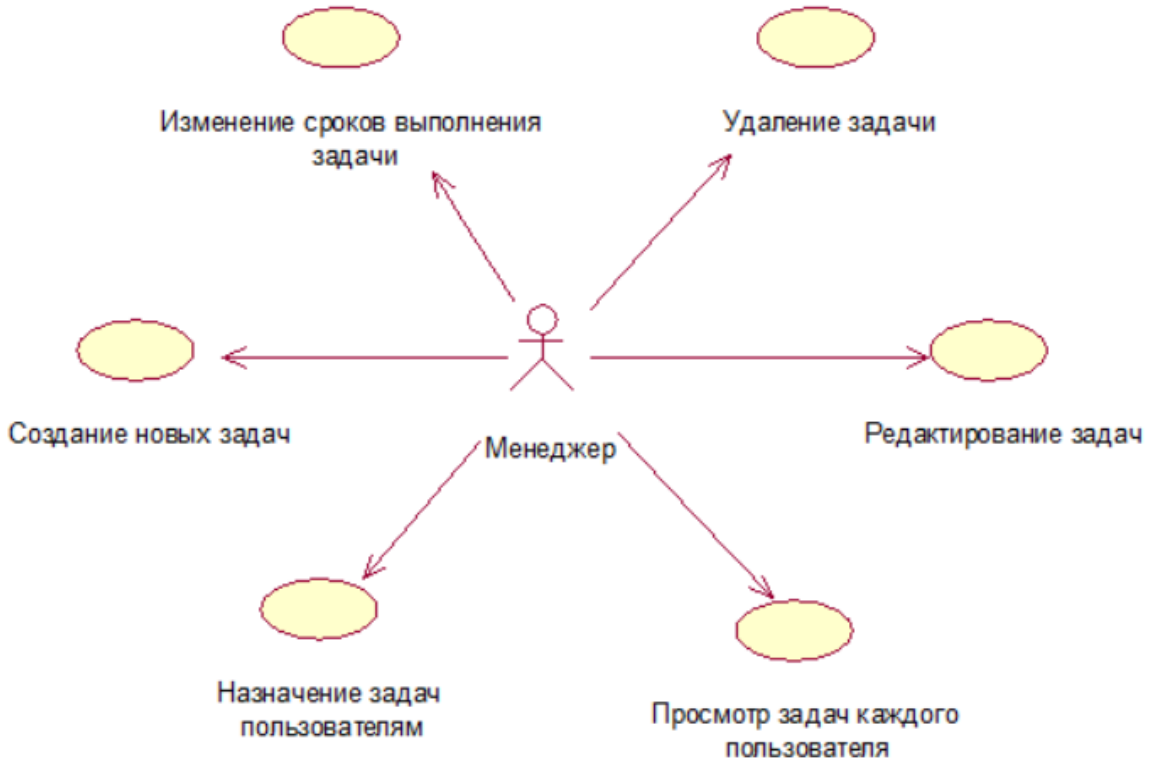


Рисунок 2.26 – Диаграмма вариантов использования для актора «Менеджер» в подсистеме «Задачи»

Построение диаграмм последовательностей

Диаграмма последовательностей отображает последовательность взаимодействия объектов в динамике, то есть во времени.

На рисунках 2.27-2.29 представлены диаграммы последовательностей для прецедентов «Просмотр информации по проектам», «Просмотр информации по задачи», «Просмотр контактов других пользователей».

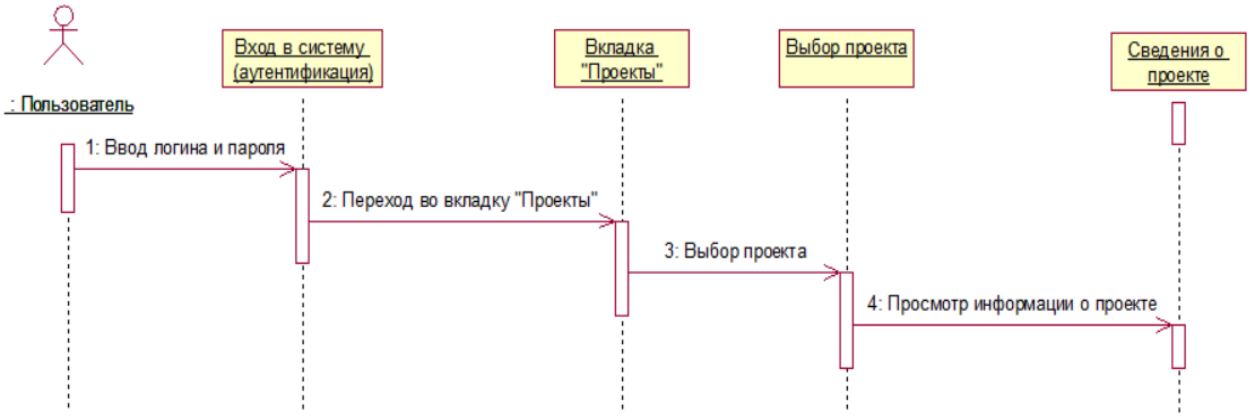


Рисунок 2.27 – Диаграмма последовательностей прецедента «Просмотр информации по проектам»

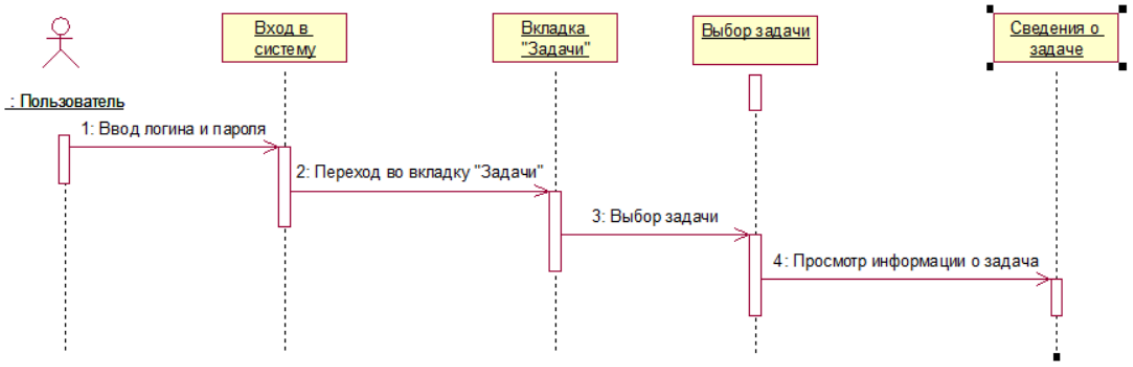


Рисунок 2.28 – Диаграмма последовательностей прецедента «Просмотр информации по задачам»

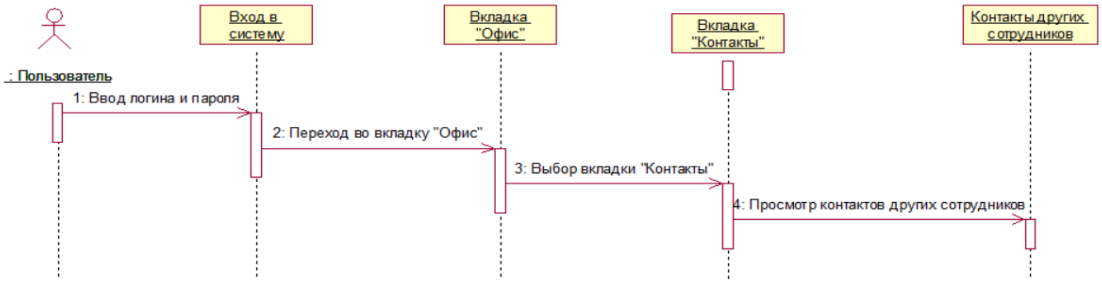


Рисунок 2.29 - Диаграмма последовательностей прецедента «Просмотр контактов сотрудников»

Построение кооперативных диаграмм

Кооперативные диаграммы, или диаграммы взаимодействия, описывает методы взаимодействия с помощью объектов и отношений между ними. На рисунках 2.30-2.32 представлены кооперативные диаграммы для прецедентов «Просмотр информации по проектам», «Просмотр информации по проектам», «Просмотр контактов других пользователей».

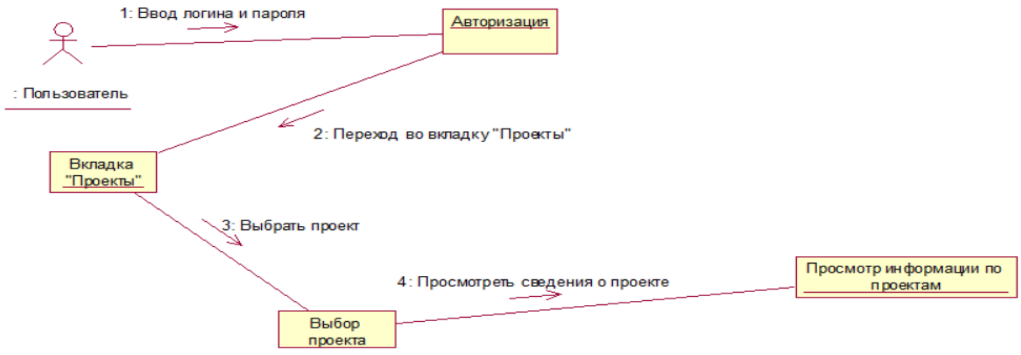


Рисунок 2.30 – Кооперативная диаграмма для прецедента «Просмотр информации по проектам»

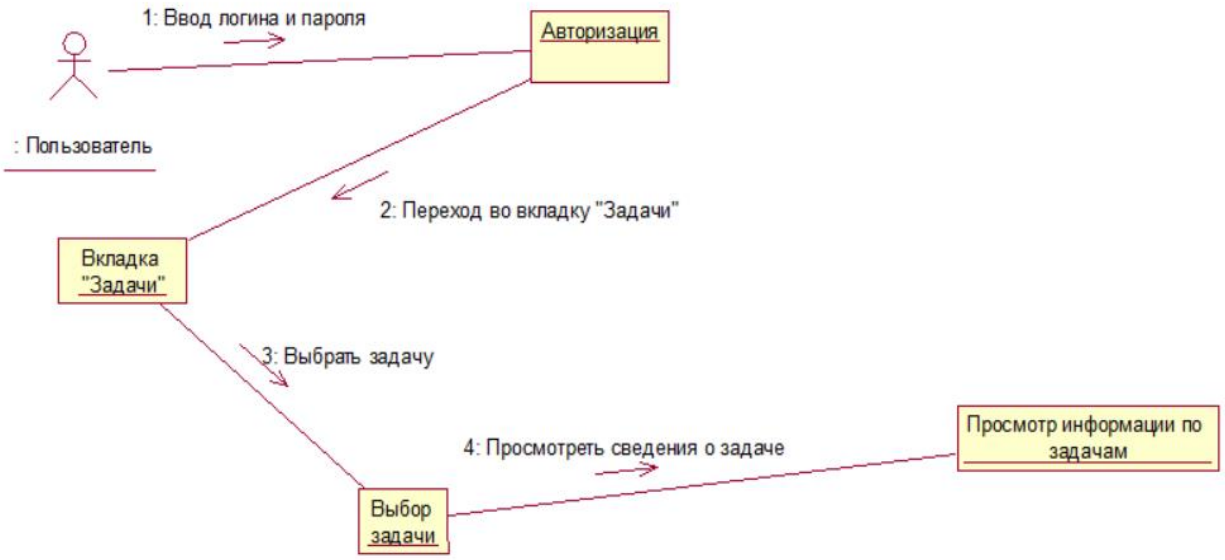


Рисунок 2.31 – Кооперативная диаграмма для прецедента «Просмотр информации по задачам»

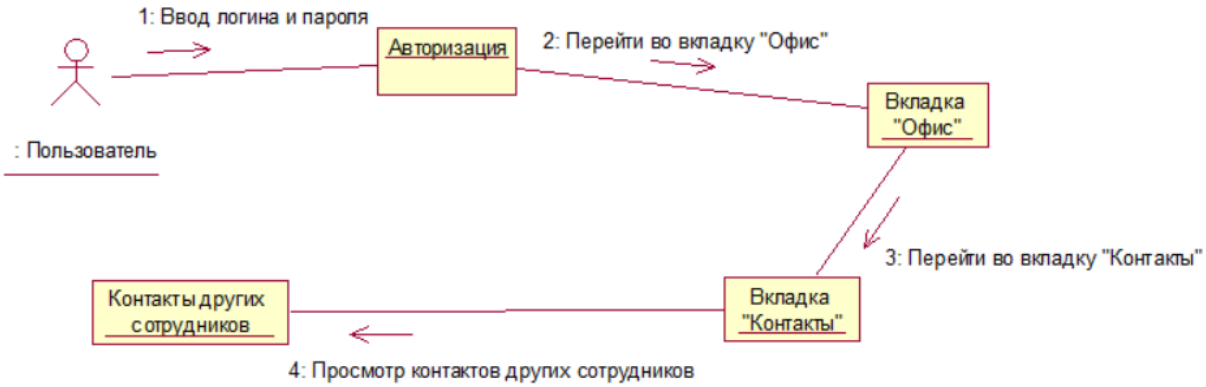


Рисунок 2.32 – Кооперативная диаграмма для прецедента «Просмотр контактов сотрудников»

3 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ

3.1 Проектирование интерфейса

3.1.1 Описание ПО для разработки интерфейса ИС

Так как программа пишется на языке C# и тип проекта – WinForms, то для разработки интерфейса установка стороннего программного обеспечения не требуется. Создание и работа над проектом ведется с помощью среды разработки MS Visual Studio 2019.

Для проектирования пользовательского интерфейса можно было бы использовать WPF, так как это более современная и гибкая система, с помощью XAML можно легко создавать и редактировать GUI, позволяя разделить работу дизайнера (XAML) и программиста.

В отличие от WPF технология WinForms лучше испытанная и предлагает множество готовых элементов управления. Это позволяет разработчику сосредоточиться на написании кода программы.

Несмотря на отсутствие такой гибкости, как у WPF, в WinForms так же можно разработать удобный графический интерфейс пользователя.

MS Visual Studio предлагает Конструктор для создания макетов форм. Вместе с ним предоставляется панель элементов, которые можно использовать при разработке.

Конструктор Windows Forms предоставляет множество средств для создания приложений Windows Forms. С его помощью можно выполнять следующие задачи [16]:

- размещать элементы управления с помощью линий привязки; - выполнять задачи конструктора с помощью смарт-тегов; - устанавливать поля и отбивки для элементов управления;

- располагать элементы управления с помощью элемента управления TableLayoutPanel;

- разделять макет элемента управления с помощью элемента управления SplitContainer;

- просматривать макет в окне «Структура документа»;

- размещать элементы управления с отображением размера и сведений о расположении;

- выполнять задачи конструктора с помощью смарт-тегов;

- задавать значения свойств в окне «Свойства».

Есть возможность использовать готовые пакеты для проектирования интерфейса, например Bunifu.UI, MaterialDesign. Данные пакеты можно установить с помощью диспетчера пакетов NuGet, однако они не бесплатные. Поэтому более выгодно воспользоваться графическим элементами по умолчанию.

3.1.2 Макет пользовательского интерфейса

Пользовательский интерфейс разработан в стиле «flat design» («плоский дизайн»). Плоский дизайн популярен за счет своей простоты и универсальности, то есть он применим для разных приложений и разных размеров экрана.

Элементы должны выглядеть просто и иметь четкие контуры. Шрифты могут быть разного типа, но так же должны быть простыми и не выбиваться из общего дизайна.

Главный экран

Данный экран, как на рисунке 3.1, представляется пользователю при запуске приложения.

В левом верхнем углу представлено название приложения и логотип.

Слева располагается панель меню, в которой находятся кнопки для навигации. Каждая кнопка имеет текст и соответствующую иконку, которые позволяют пользователю более быстро ориентироваться в приложении.

В нижней части панели меню располагается панель, отображающая имя пользователя. В этой панели также предусмотрена кнопка выхода, позволяющая завершить работу под текущим пользователем. После нажатия на нее производится выход из главного окна приложения, и открывается окно авторизации.

Переход на различные страницы осуществляется посредством вызова дочерних форм с помощью функций, приведенных в Приложении А.

Рисунок 3.1 – Главный экран

Экран домашней страницы пользователя

Экран домашней страницы пользователя отображает для пользователя следующую информацию: имя пользователя, должность, номер телефона.

Также представлены контакты коллег, представленные в виде таблицы. Помимо контактных данных, отображаются и должности сотрудников.

Для заполнения полей предусмотрено подключение к базе данных и выполнение запросов.

Каждый элемент имеет установленные свойства Anchor и Dock, таким образом реализуется адаптивный дизайн.

Макет экрана домашней страницы представлен на рисунке 3.2.

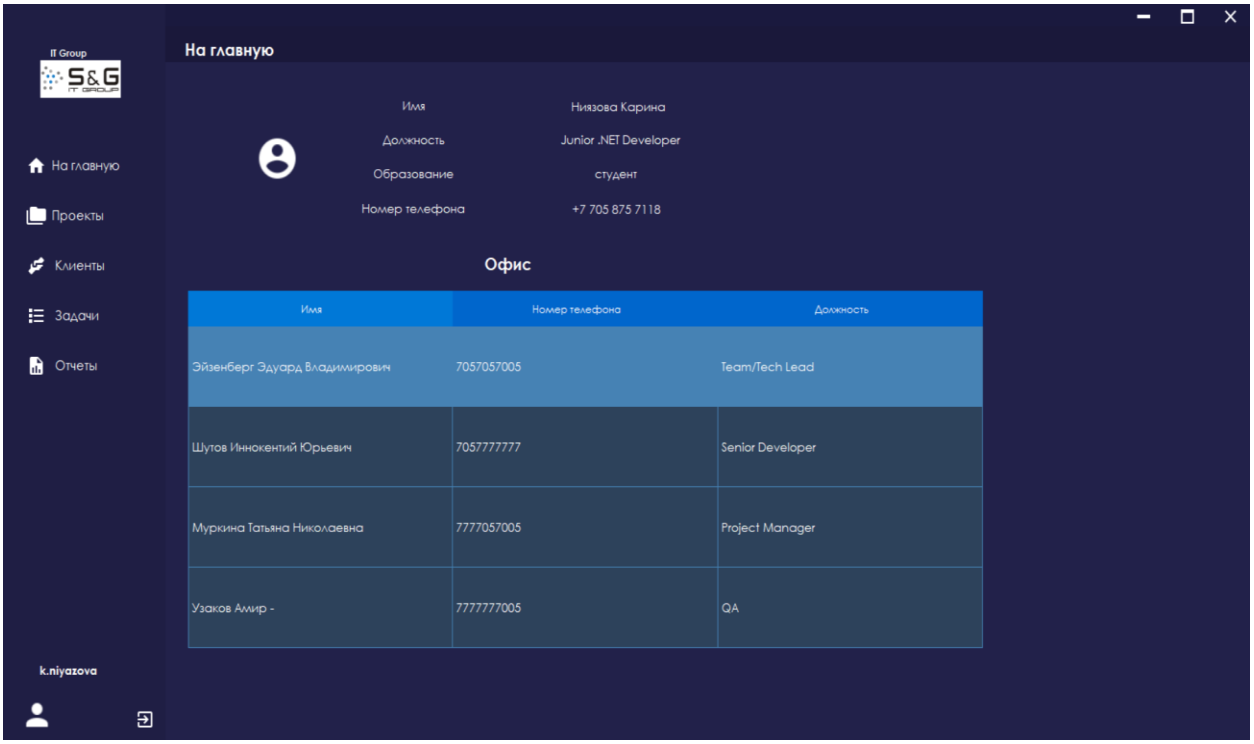


Рисунок 3.2 – Домашняя страница пользователя

Информация о клиентах

На данной странице отображается список клиентов. Список представляет собой коллекцию карточек, расположенных друг под другом и содержащих следующую информацию: полное наименование организации, контактные данные, имя представителя, и детали договора (номер, дата начала и окончания действия, сумма).

Наименование каждого пункта из карточки о клиенте выделяется полужирным шрифтом, остальной текст – обычным. Слева на карточке располагается краткое название организации.

Если список не помещается на страницу, то становится доступной полоса прокрутки.

На рисунке 3.3 представлен макет страницы с информацией о клиентах.

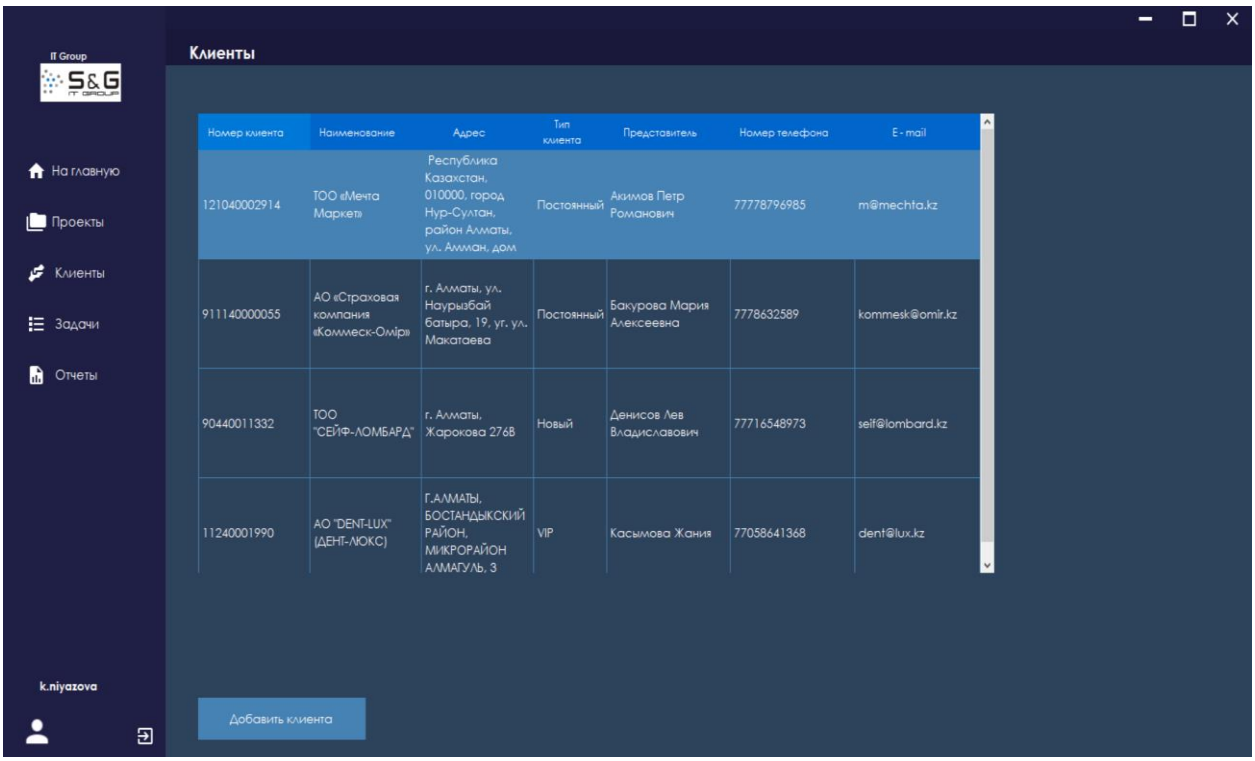


Рисунок 3.3 – Экран информации о клиентах

Экран проектов пользователя

На экране проектов пользователю представлена вся информация о проектах в виде диаграмм и карточек, как на рисунке 3.4.

Карточки отображают прогресс проекта, данные количестве задач и вопросов, разделенных по приоритетам. Каждая карточка имеет свою цветовую окраску, что позволяет пользователю проще ориентироваться на странице.

Карточка прогресса содержит название и круговую диаграмму, которая отображает процент завершенности проекта. Диаграмма поделена на три сектора, каждый из которых отображает статус задач.

Карточка задач содержит три пункта: задачи на рассмотрении, задачи в процессе и выполненные задачи. Задачи, принятые на исполнение, отмечаются красным цветом, текущие задачи – оранжевым, выполненные – зеленым. Цвета приоритетов подобраны таким же образом.

Карточка приоритетов отображает количество задач по приоритетам: высокий, средний и низкий. Задачи высокого приоритета отображаются красным цветом, среднего – оранжевым, низкого – зеленым. Так пользователь может правильно распределить очередность решаемых задач.

Слева от рабочей области рядом с панелью меню располагается панель меню данной страницы, которая содержит кнопки, отвечающие за каждый проект. При нажатии на кнопку информация на рабочей панели меняется на соответствующую выбранному проекту. Для удобства пользователя нажатая кнопка выделяется своим уникальным цветом.

Ниже отображаются название проекта (тип) и его краткое описание.

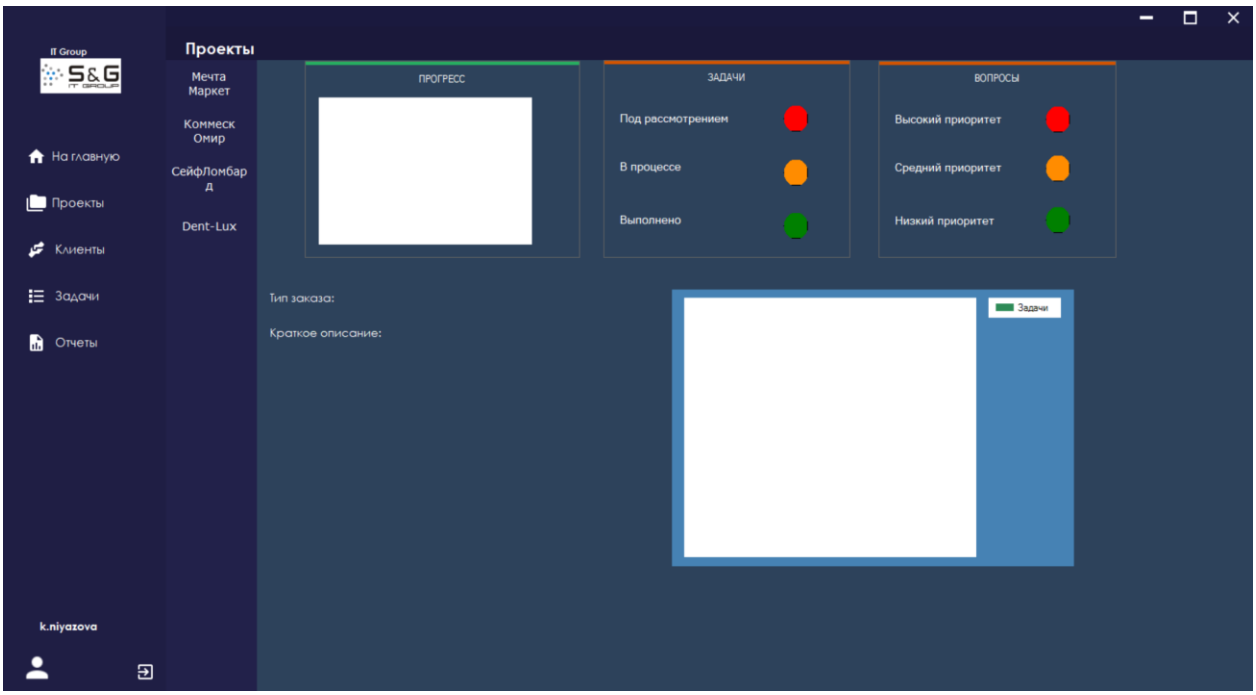


Рисунок 3.4 – Экран проектов пользователя

3.2 Обоснование выбора программного обеспечения

В качестве среды программирования была выбрана MS Visual Studio 2019. Эта среда разработки позволяет писать код и производить его сборку, откладку и публикации приложения. Также она включает компиляторы, графические конструкторы и возможность подключить различные расширения.

Visual Studio поддерживает технологию автодополнения IntelliSense, которая ускоряет разработку ПО и выводит часть соответствующей документации в виде всплывающих окон [17].

Преимущества Visual Studio [17]:

- удобный и понятный интерфейс;
- поддержка различных языков разработки;
- автоматическое установка деталей и ссылок во время написания кода;
- автоматическое форматирование кода и возможность ее настройки;
- высокая скорость разработки;
- автоматическое обнаружение ошибок в коде;
- поддержка рефакторинга кода;
- возможность отладки и диагностики.

К недостаткам можно отнести отсутствие кроссплатформенности продукта. C# – высокоуровневый, объектно-ориентированный язык программирования, применяемый для разработки различных приложений. Этот язык относится к C-подобным, поэтому синтаксис языка схож с синтаксисом C++ и Java.

C# имеет строгую статическую типизацию. Также поддерживает полиморфизм, перегрузку операторов, события и т.д. [18, 19].

Данный язык программирования может использоваться для разработки разных приложений и программ: игры, машинное обучение, мобильные приложения, веб-приложения и сайты. Для получения доступа к данным из базы данных есть возможность использования компонентов ADO.NET, которая предоставляет доступ к реляционным базам данных, другим источникам данных, файловой системе и каталогам [20, 21].

В качестве СУБД будет использована MS SQL Server. Так как данная СУБД разработана Microsoft, то ее лучше и проще связать с приложением на языке C#, то есть поддерживает работу с другими продуктами Microsoft. СУБД хорошо масштабируется и адаптируется, позволяет обрабатывать большой объем запросов, реализует эффективный поиск и позволяет создавать отчеты [22].

В ADO.NET имеются провайдеры в том числе и для MS SQL Server – пространство имен System.Data.SqlClient.

3.3 Описание программного обеспечения

3.3.1 Общие сведения

Программа была написана на языке программирования C#, средство разработки – Microsoft Visual Studio. Реализация представлена в виде решения под названием *ITGROUP*, который содержит файлы с непосредственно кодом программы, файлы конфигурации, файл-решение.

Приложение создано на языке программирования C# в среде разработки Microsoft Visual Studio 2019. C# – объектно-ориентированный язык программирования и разработки приложений для платформы Microsoft .NET Framework. Поэтому данная программа будет функционировать на платформах семейства ОС Windows. Назначение программы – упрощение контроля работы команды разработчиков над проектами [23].

Программа выполняет функции сбора и отображения показателей работы сотрудников, сбора и обработки статистики, организации хронологической отчетности. Отчеты, представленные в виде таблиц и, могут быть распечатаны в формате PDF и сохранены в таблицу Excel. Диаграммы могут быть сохранены как изображения в форматах PNG, JPEG, BMP. Для устойчивого функционирования приложения рабочая станция должна удовлетворять следующим требованиям:

- операционная система Windows 10;
- процессор с частотой не менее 2 ГГц;
- жесткий диск объем не менее 50 Гб;
- не менее 2 Гб оперативной памяти.

3.3.2 Функциональное назначение

В каталоге содержатся папки *ChildForms, Resources* и следующие файлы:

- *Program.cs.* Выполнение приложения начинается с этого класса. Для этого подключаются необходимые пространства имен и в метод Application.Run() в качестве аргумента передается экземпляр класса *Authorization.cs,* с которого должен производиться запуск;

- *Authorization.cs,* – класс, выполняющий функции авторизации;

- *MainForm.cs,* основная родительская форма, в которой открываются дочерние формы;

- *CircularButton.cs* – наследник класса Button, представляет элемент «кнопка» в круглой форме;

- *DGVPrinter.cs*, класс для печати таблиц в файл формата PDF;

- *App.config* – файл XML со множеством конфигураций графического пользовательского приложения;

В папке *ChildForms* содержатся дочерние формы:

- *Clients.cs,* выводит список существующих клиентов в виде таблицы и позволяет добавлять новых клиентов;

- *Projects.cs,* выводит данные по проектам в виде диаграмм и таблиц;

- *HomePage.cs*, выводит общую сводку по проектам и задачам;

- *Tasks.cs*, отображает данные по задачам текущего пользователя;

- *Reports.cs*, отображает таблицы и диаграммы для отчетности.

Помимо вышеперечисленных файлов, каталог содержит папки:

- *Properties*, содержит описательные свойства проекта;

- *Resources,* папка с ресурсами проекта. В данном случае это изображения различных расширений.

В состав программы также входит пользовательский интерфейс, который представлен файлами *MainForm.Designer*, *Authorization.Designer*, *Clients.Designer*, *HomePage.Designer*, *Projects.Designer*, *Reports.Designer*, *Tasks.Designer*, описывающими соответственно перечисленные выше формы.

Само приложение реализовано в виде файла с расширением *.exe*. Данное приложение имеет размер около 53 Мб. Для функционирования приложения нужно, чтобы были установлены все необходимые динамические библиотеки.

3.3.3 Описание логической структуры

Приложение построено по принципу клиент-серверной архитектуры с «толстым» клиентом. Данный тип архитектуры позволяет обеспечить локальное функционирование. Клиентская часть реализует интерфейс взаимодействия, обработку и представление данных (бизнес-логику), таким образом приложение имеет полную функциональность и независимость от сервера. Серверная часть является хранилищем данных, так как на сервере располагается база данных.

Логическая структура класса Authorization.cs программы «ITGROUP.exe» (листинг приведен в Приложении А) с привязкой к строкам кода представлена ниже.

1-12 – подключение пространства имен;

13 – объявление пространства имен проекта;

15 – объявление частичного класса *Authorization*;

17 – объявление строковых переменных;

25-37 – использование компонентов динамической библиотеки для реализации функции перетаскивания окна по рабочей области;

38-78 – проверка логина и пароля, введенных пользователем, при успешной авторизации – открытие главного окна программы.

Логическая структура класса *MainForm.cs* программы «ITGROUP.exe» с привязкой к строкам кода представлена ниже:

1-10 – подключение пространств имен;

12 – инициализация пространства имен проекта;

13-23 – инициализация конструктора класса;

24-41 – метод открытия дочерней формы;

42-56 – реализация функции перетаскивания окна по рабочей области посредством зажатия кнопкой мыши верхней панели окна;

57-82 – методы навигации окна программы (минимизация, максимизация, сворачивание);

83-88 – метод открытия формы «На главную»;

89-94 – метод открытия формы «Проекты»;

95-100 – метод открытия формы «Клиенты»;

101-106 – метод открытия формы «Задачи»;

107-112 – метод открытия формы «Отчеты»;

113-117 – загрузка данных о пользователе;

118-124 – метод выхода пользователя из системы.

Логическая структура класса *HomePage* программы «ITGROUP.exe» с привязкой к строкам кода представлена ниже:

1-12 – подключение пространств имен;

12 – инициализация пространства имен проекта;

13-21 – инициализация конструктора класса;

22-46 – получение данных о пользователе;

47-75 – получение контактов других пользователей;

76-150 – получение данных пользователя (должность, контакты);

151-159 – метод загрузки текущего окна.

Логическая структура класса *Clients* программы «ITGROUP.exe» с привязкой к строкам кода представлена ниже:

1-13 – подключение пространств имен;

14 - инициализация пространства имен проекта;

15-23 – инициализация конструктора класса;

24-60 – метод получения данных о клиентах и вывода в виде таблицы;

62-66 – метод загрузки текущей страницы;

67-74 – метод сокрытия дочерней панели для добавления клиентов;

75-83 – метод открытия дочерней панели для добавления клиентов, на которой располагаются поля ввода;

84-165 – метод ввода и проверки данных от пользователя для добавления клиентов.

Логическая структура класса *Projects* программы «ITGROUP.exe» с привязкой к строкам кода представлена ниже:

1-13 – подключение пространств имен

14 – инициализация пространства имен проекта;

15-24 – инициализация конструктора класса;

25-35 – метод для активации панели пользовательского интерфейса в меню навигации;

36-51 – метод очистки диаграмм;

52-165 – методы получения данных о проектах (название, описание, статус);

166-219 – методы получения данных для диаграмм;

220-313 – методы получения данных по каждому проекту и их отображение.

Логическая структура класса *Reports* программы «ITGROUP.exe» с привязкой к строкам кода представлена ниже:

1-18 – подключение пространств имен

19 – инициализация пространства имен проекта;

23-38 – инициализация конструктора класса;

39-96 – метод получения и отображения данных по задачам пользователя;

97-105 – метод загрузки текущей страницы;

106-133 – методы получения информации по проектам;

134-148 – метод для печати отчета в формате PDF;

149-183 – метод получения информации для диаграмм;

184-230 – метод печати таблиц в Excel;

231-255 – метод сохранения диаграммы;

256-266 – метод очистки диаграмм;

Логическая структура класса *Tasks* программы «ITGROUP.exe» с привязкой к строкам кода представлена ниже:

1-15 – подключение пространств имен

16 – инициализация пространства имен проекта;

17-26 – инициализация конструктора класса;

27-42 – метод очистки диаграмм;

43-170 – методы получения информации по задачам пользователя;

171-200 – метод вывода данных о задачах в таблицу;

201-219 – метод загрузки текущей формы;

220-234 – метод печати таблицы в формате PDF;

279 – метод печати таблицы в формате Excel.

3.3.4 Используемые технические средства

Для реализации проекта было использовано техническое средство – ноутбук с характеристиками:

- ОС – Microsoft Windows 10 Pro;
- тип системы – 64-разрядная операционная система, процессор x64. Процессор – Intel™ Core™ i5-9300H CPU @ 2.40GHz;
- ОЗУ – 8ГБ.

В качестве среды программирования был выбран пакет программ MS Visual Studio 2019.

3.3.5 Вызов и загрузка

Запуск программы можно осуществить двумя способами:

1. Запустить приложение в сборке Release – «ITGROUP.exe».
2. Открыть данный проект в Microsoft Visual Studio, нажать на кнопку «Пуск» в режиме отладки (Ctrl+F5 либо F5).

После запуска откроется окно авторизации, показанное на рисунке 3.5. После успешной авторизации открывается главное окно (рисунок 3.6).

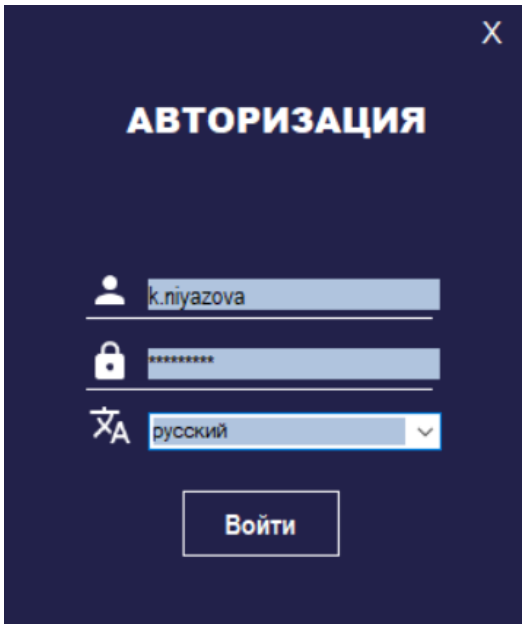


Рисунок 3.5 – Окно авторизации запущенной программы

3.3.6 Входные данные

В качестве входных данных выступают значения полей для ввода логина и пароля в окне авторизации. После нажатия на кнопку «Войти» производится проверка данных. Если данные верны, то открывается главное окно программы, представленное на рисунке 3.6.

Рисунок 3.6 – Главное окно программы

На вкладке «Клиенты» расположена форма ввода данных о новых клиентах, показанная на рисунке 3.7. Для поля «БИН» производится проверка символов (допускаются только числовые). Параметр «Тип клиента» позволяет пользователю выбрать один из существующих типов. Данные в поле «Телефон» могут быть введены только по шаблону «###-###-####».

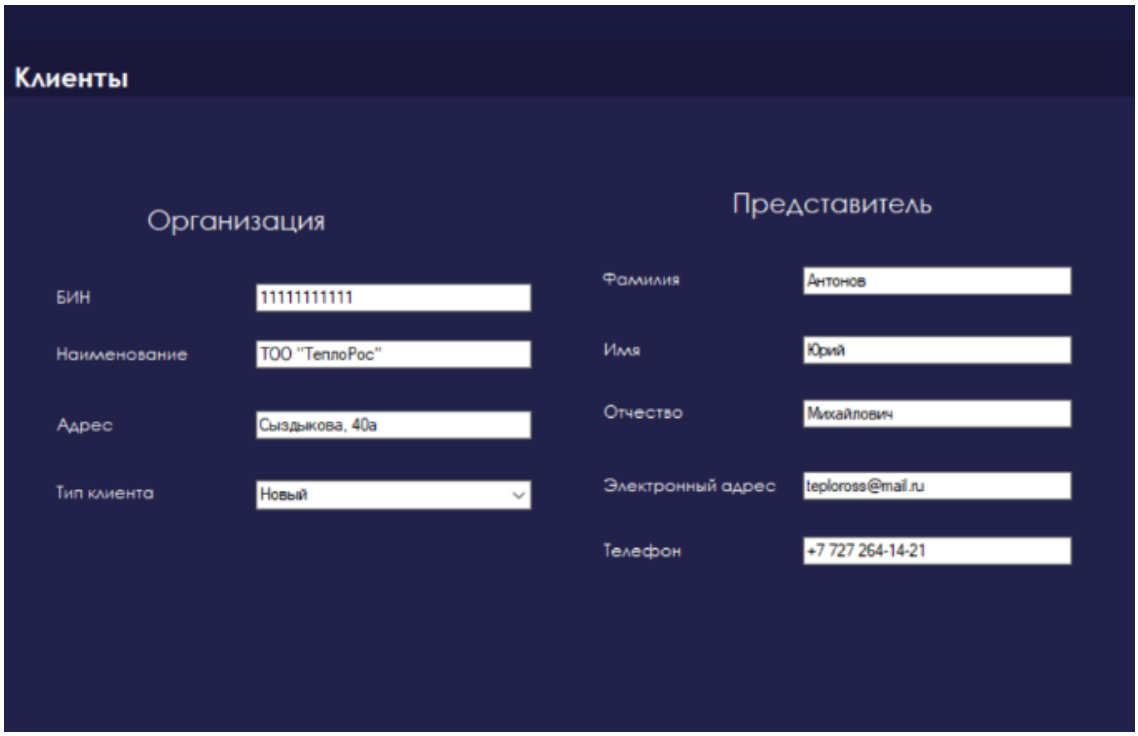


Рисунок 3.7 – Поля ввода данных

3.3.7 Выходные данные

В качестве выходных данных выступают диаграммы и таблицы. На рисунке 3.8 представлена таблица вывода списка клиентов компании в виде таблицы.

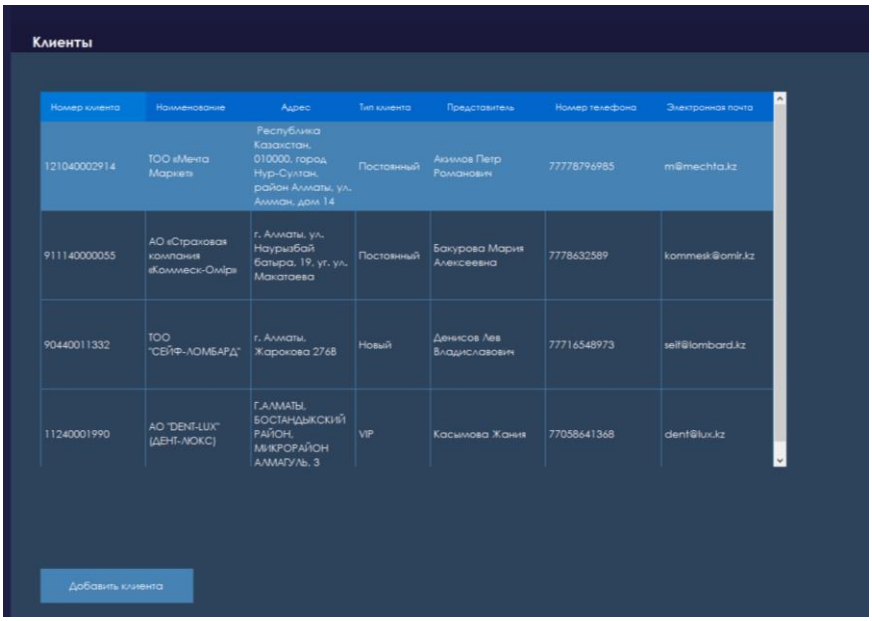


Рисунок 3.8 – Форма «Клиенты»

При переходе на форму «На главную» отображается информация о пользователе и контакты других сотрудников. Форма представлена на рисунке 3.9.

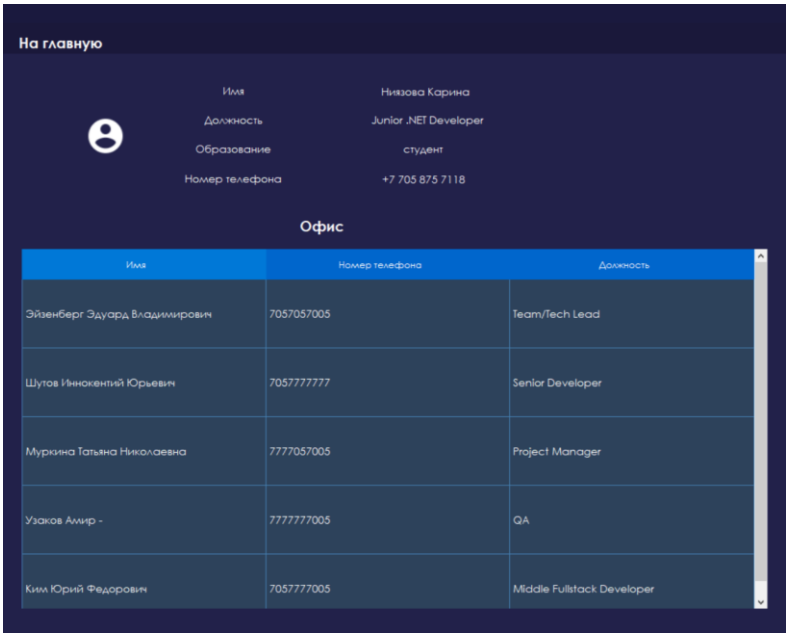


Рисунок 3.9 – Форма «На главную»

При переходе на форму «Проекты», отображается информация по задачам пользователя по конкретному проекту. Форма представлена на рисунке 3.10.

Рисунок 3.10 – Форма «Проекты»

На форме «Задачи», представленной на рисунке 3.11, помимо диаграмм и таблицы есть возможность печати таблицы в двух форматах.

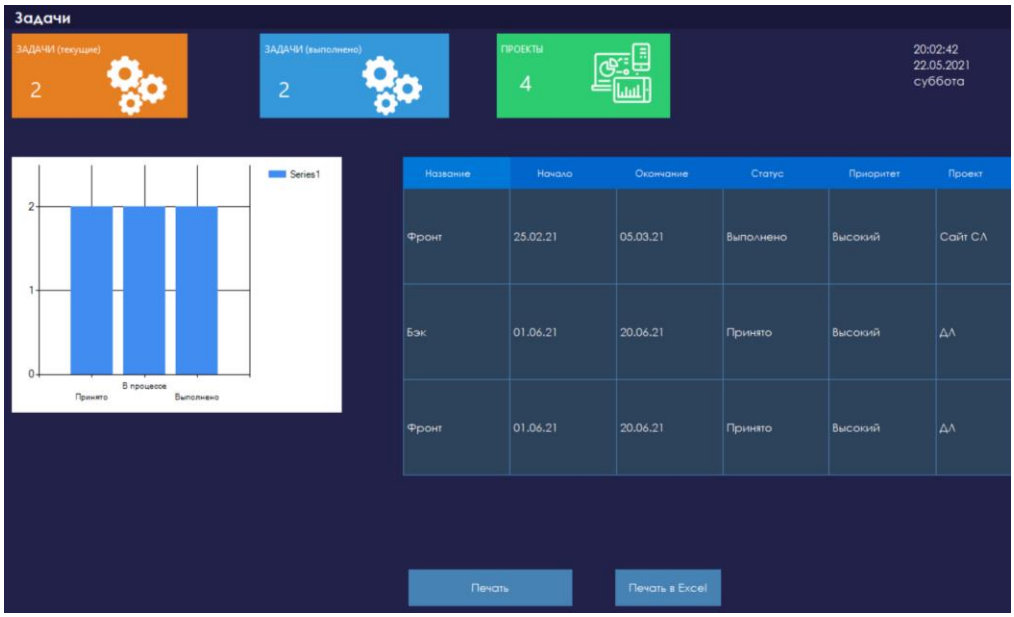


Рисунок 3.11 – Форма «Задачи»

На форме «Отчеты» отображается таблица и диаграмма, а также кнопки для сохранения данных, как на рисунке 3.12.

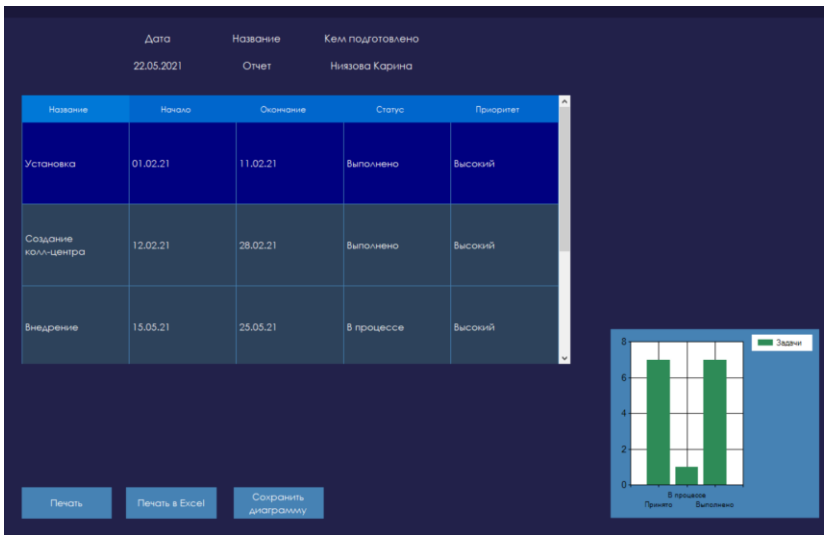


Рисунок 3.12 – Форма «Отчеты»

Если пользователь выберет «Печать», то откроется диалоговое окно, как на рисунке 3.13.

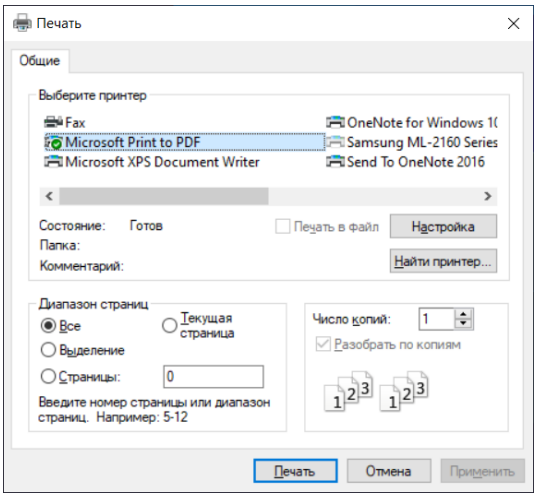


Рисунок 3.13 – Диалоговое окно «Печать»

Сохраненный документ будет иметь вид как на рисунке 3.14.

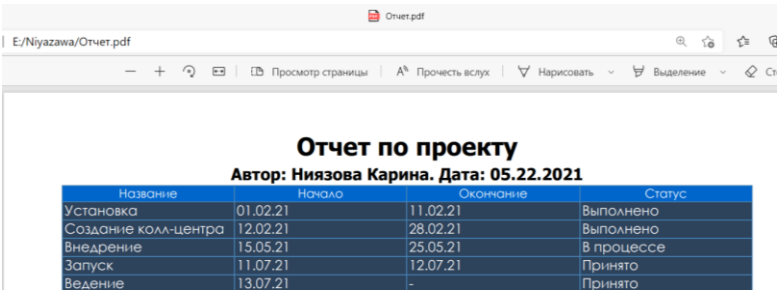


Рисунок 3.14 – Сохраненный файл

Если пользователь выберет «Печать в Excel», то откроется окно для выбора пути сохранения документа в заранее установленном формате, представленное на рисунке 3.15.

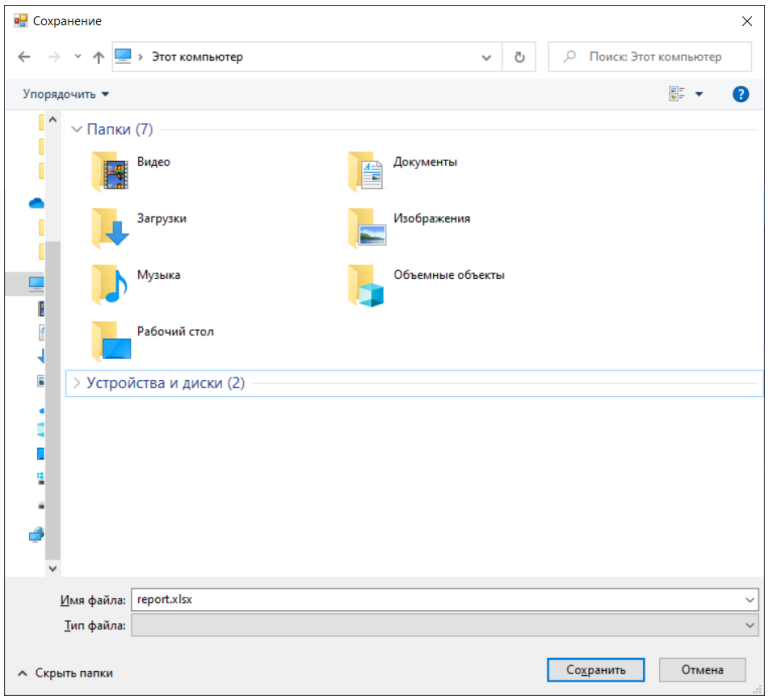
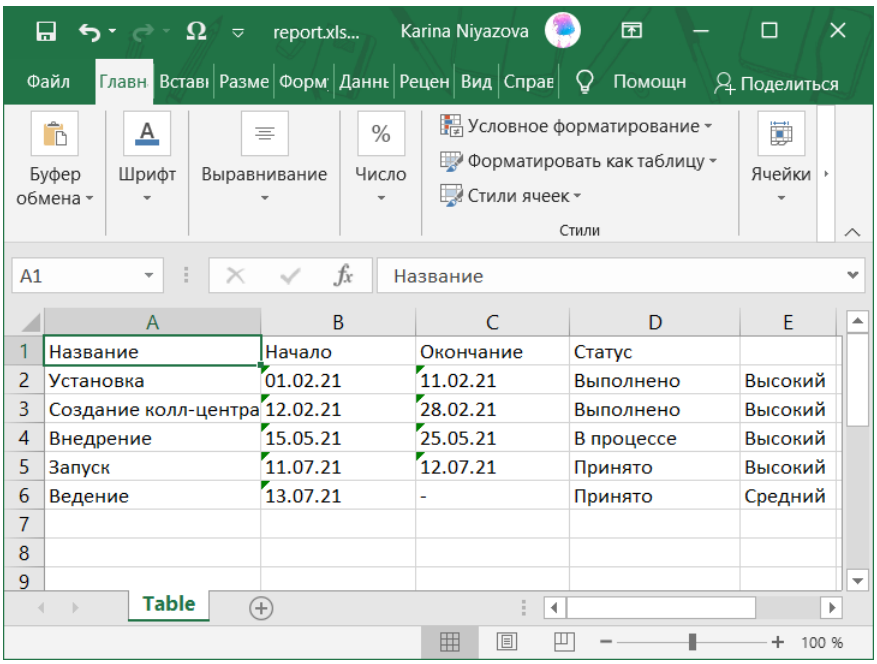


Рисунок 3.15 – Выбор пути сохранения файла

Сохраненный файл будет иметь вид, как на рисунке 3.16. Рабочая область будет иметь название «Table».



	A	B	C	D	E
1	Название	Начало	Окончание	Статус	
2	Установка	01.02.21	11.02.21	Выполнено	Высокий
3	Создание колл-центра	12.02.21	28.02.21	Выполнено	Высокий
4	Внедрение	15.05.21	25.05.21	В процессе	Высокий
5	Запуск	11.07.21	12.07.21	Принято	Высокий
6	Ведение	13.07.21	-	Принято	Средний
7					
8					
9					

Рисунок 3.16 – Сохраненный файл

Диаграмма сохраняется при нажатии кнопки «Сохранить диаграмму». При нажатии кнопки открывается диалоговое окно, где нужно выбрать путь сохранения файла и его формат. Сохраненное изображение представлено на рисунке 3.17.

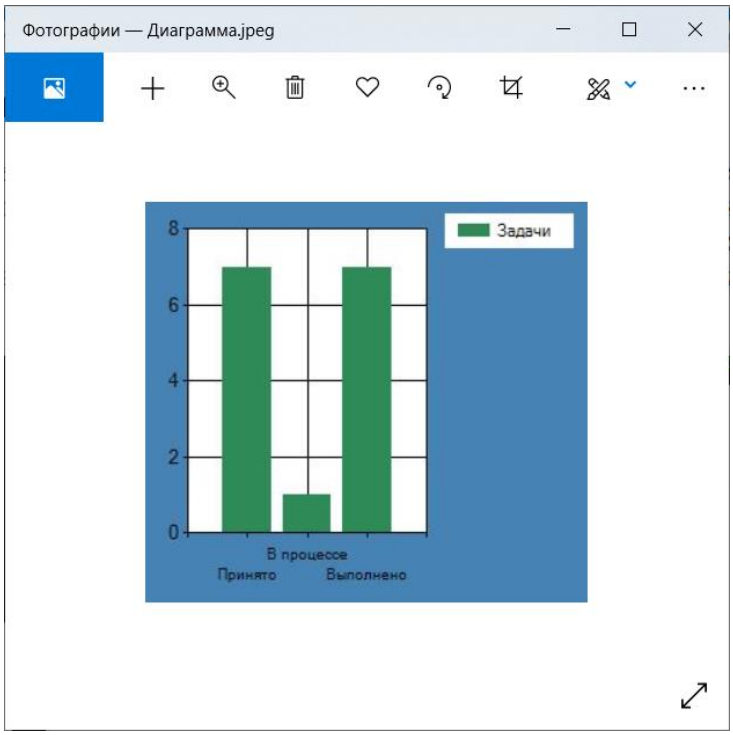


Рисунок 3.17 – Сохраненное изображение

Для приложения были подключены пакеты «NLog» и «NLog.Config» для журналирования работы программы. При неудачной попытке входа в систему, успешной авторизации или других событиях информация записывается в лог-файл, который находится в рабочем каталоге программы.

Пример файла показан на рисунке 3.18. В названии файла указывается дата, когда был создан файл.

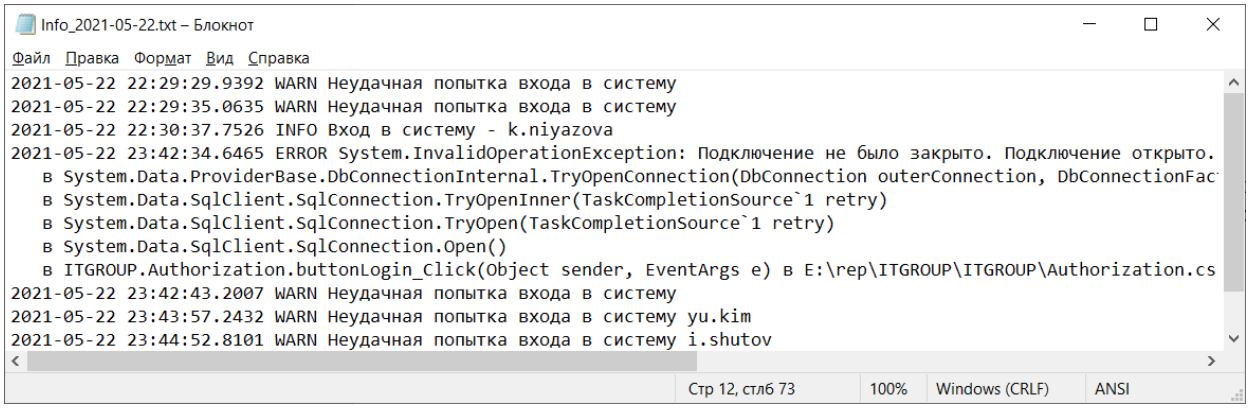


Рисунок 3.18 – Файл логирования

3.4 Контрольный пример

Для работы с программой необходимо ее установить, то есть произвести развертывание приложения на компьютере пользователя. Для этого создан инсталляционный файл на основе проекта инсталлятора Setup Project, который помещается в одном решении с основным проектом. В инсталляторе есть возможность как установки, так и удаления программы. Пошаговая установка программы показана на рисунке 3.19.

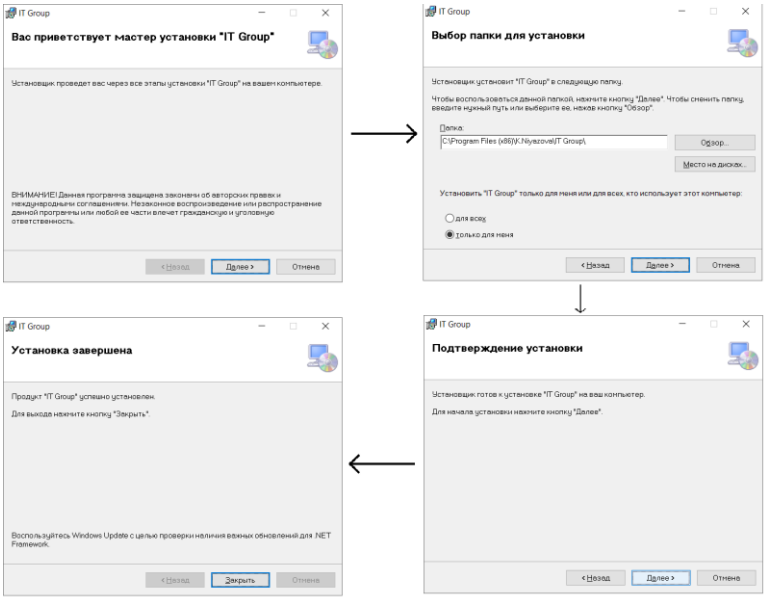


Рисунок 3.19 – Установка программы

Первое окно после запуска программы – окно авторизации. В поля ввода пользователь вводит логин и пароль, а также выбирает язык (рисунок 3.20).

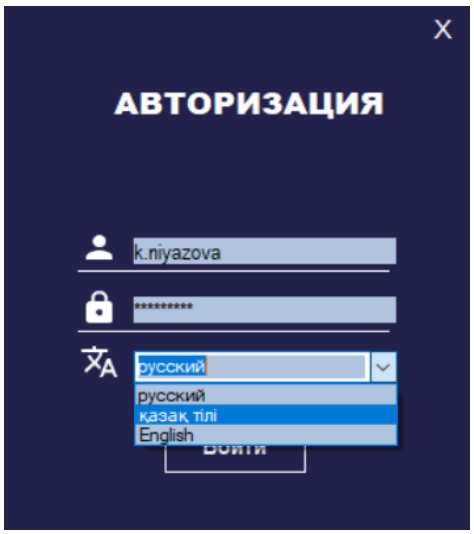


Рисунок 3.20 – Авторизация и выбор языка интерфейса

Если введены неправильные данные, то выйдет сообщение об ошибке, и данные об этом будут записаны в локальный лог-файл. Если данные поля верны, то открывается главное окно. Главное (родительское) окно программы представлено на рисунке 3.21. Слева отображается панель меню.

Рисунок 3.21 – Главное окно программы

В окне «На главную», представленном на рисунке 3.22, отображается информация о текущем пользователе и контактная информация сотрудников офиса.

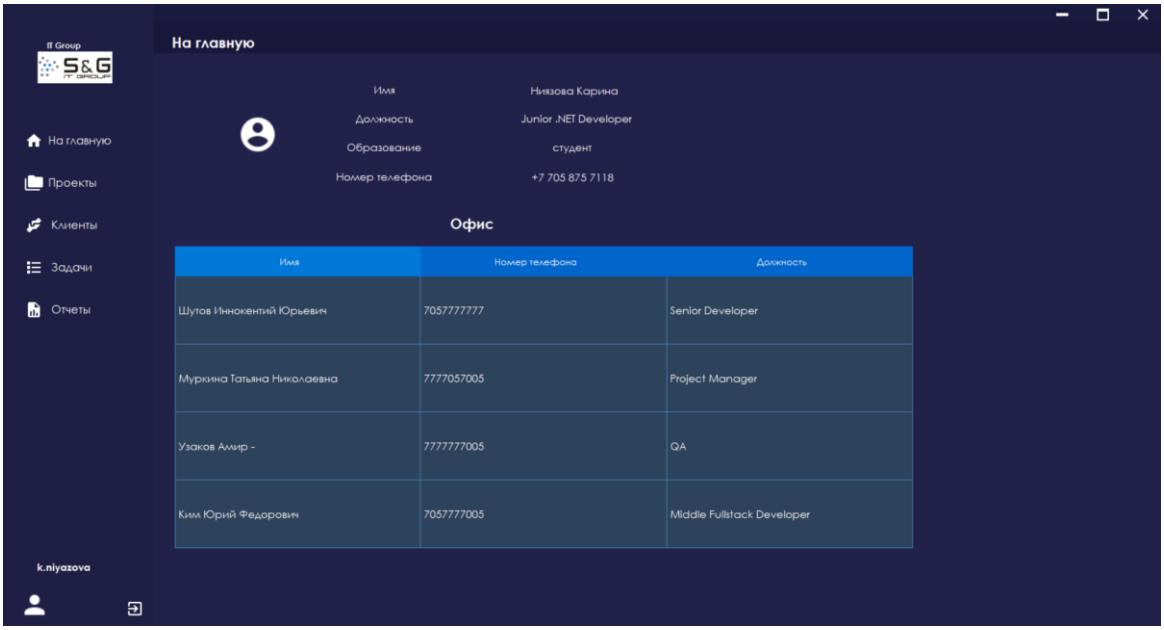


Рисунок 3.22 – Окно «На главную»

В окне «Проекты» отображается небольшой дашборд – аналитическая панель, на которой данные о проектах пользователя представлены компактно в виде диаграмм и таблиц, как на рисунке 3.23.

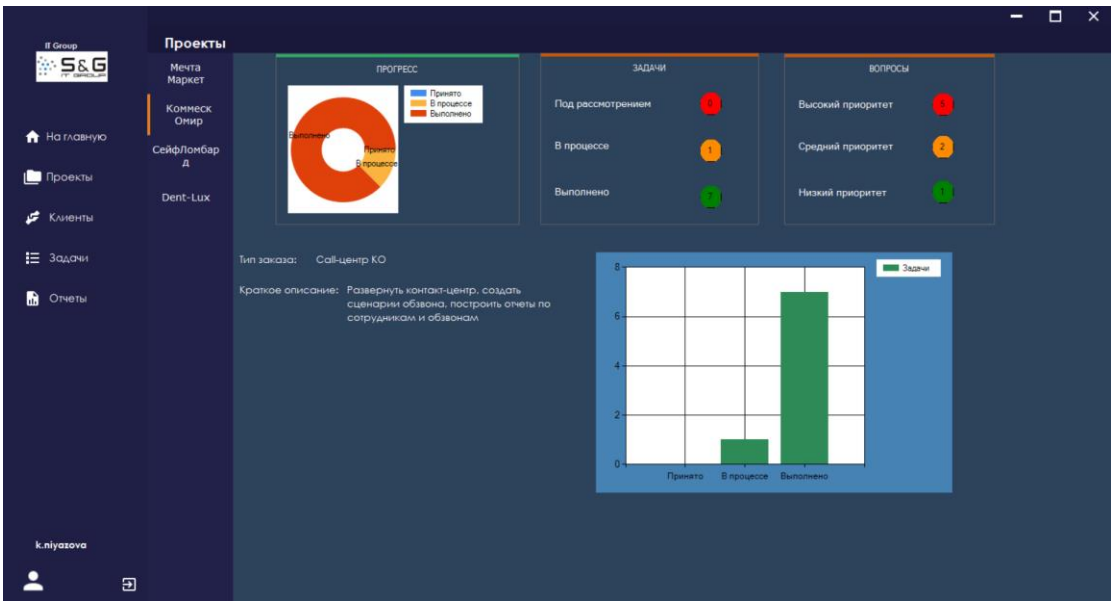


Рисунок 3.23 – Аналитическая панель окна «Проекты»

В окне клиенты отображается список клиентов, как на рисунке 3.24. Нажатие на кнопку «Добавить клиента» отображает панель для ввода данных.

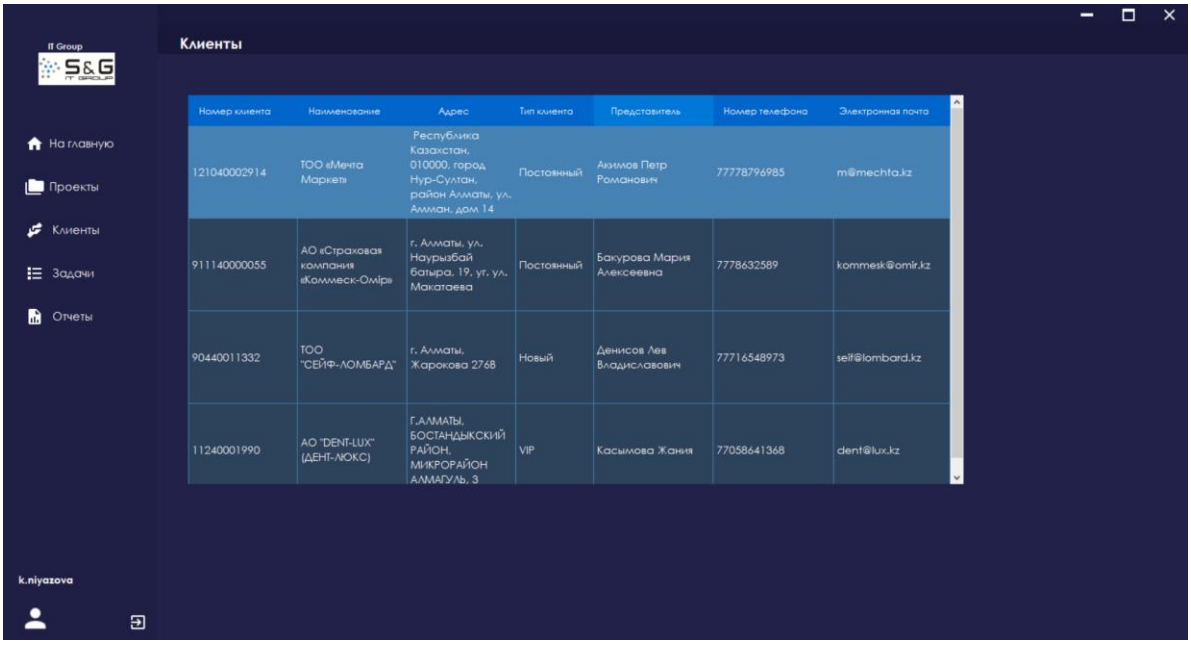


Рисунок 3.24 – Окно «Клиенты»

В окне «Задачи» также отображается аналитическая панель, в которой собрана статистика по задачам пользователя. Здесь отображается количество

текущих и выполненных задач, их список, сравнительная диаграмма. В правом верхнем углу расположена панель, отображающая дату, время и день недели, чтобы пользователь мог сориентироваться в датах начала и окончания задач, просматривая таблицу. Пользователь может менять статус задачи и оставлять комментарий к ней. Окно «Задачи» представлено на рисунке 3.25.

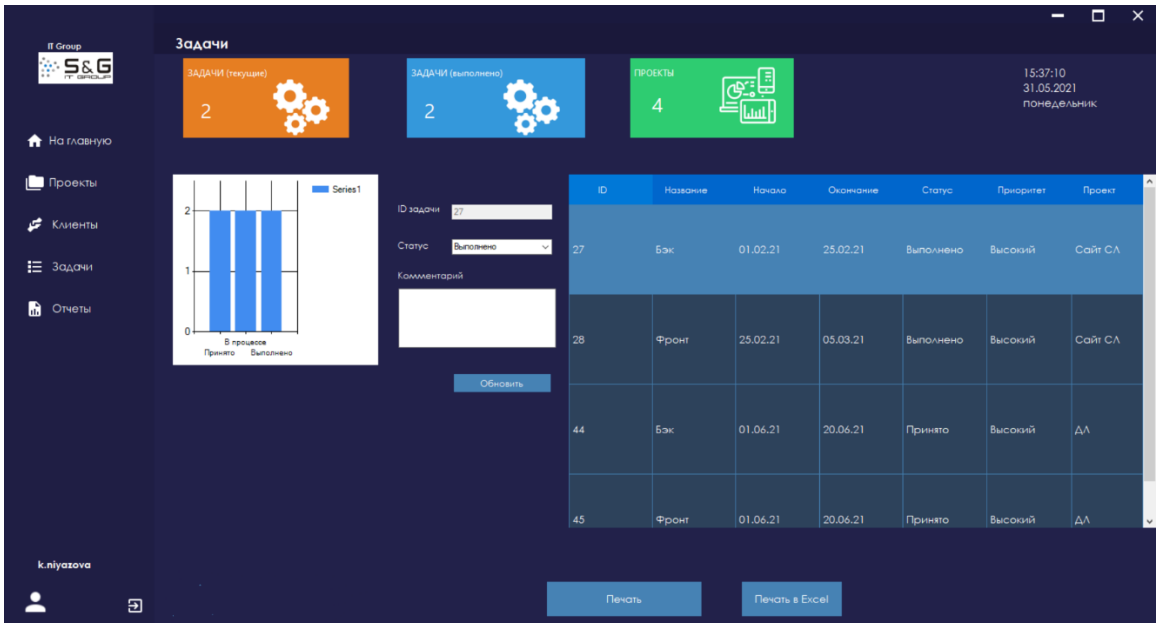


Рисунок 3.25 – Окно «Задачи»

В окне «Отчеты», показанном на рисунке 3.26, пользователь может выбрать, как сохранить статистическую информацию по своей работе.

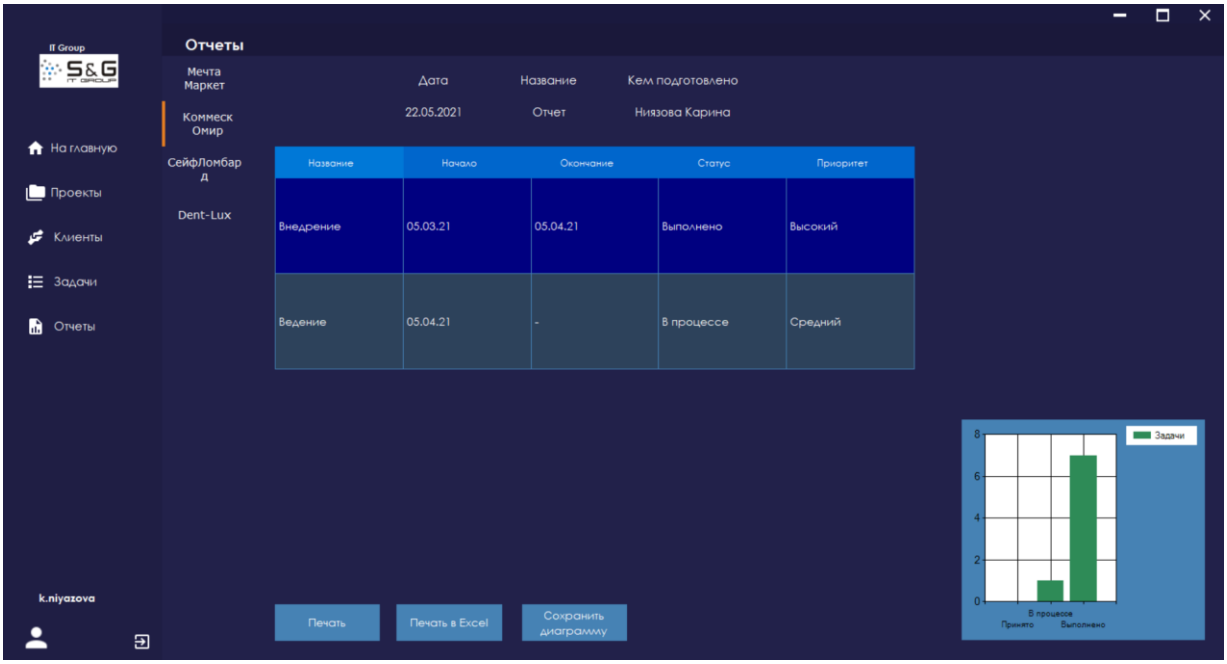


Рисунок 3.26 – Окно «Отчеты»

В окне «Пользователи», которое доступно только пользователю «Администратор», можно добавлять новых пользователей, обновлять информацию и удалять существующих пользователей. Фома редактирования представлена на рисунке 3.27

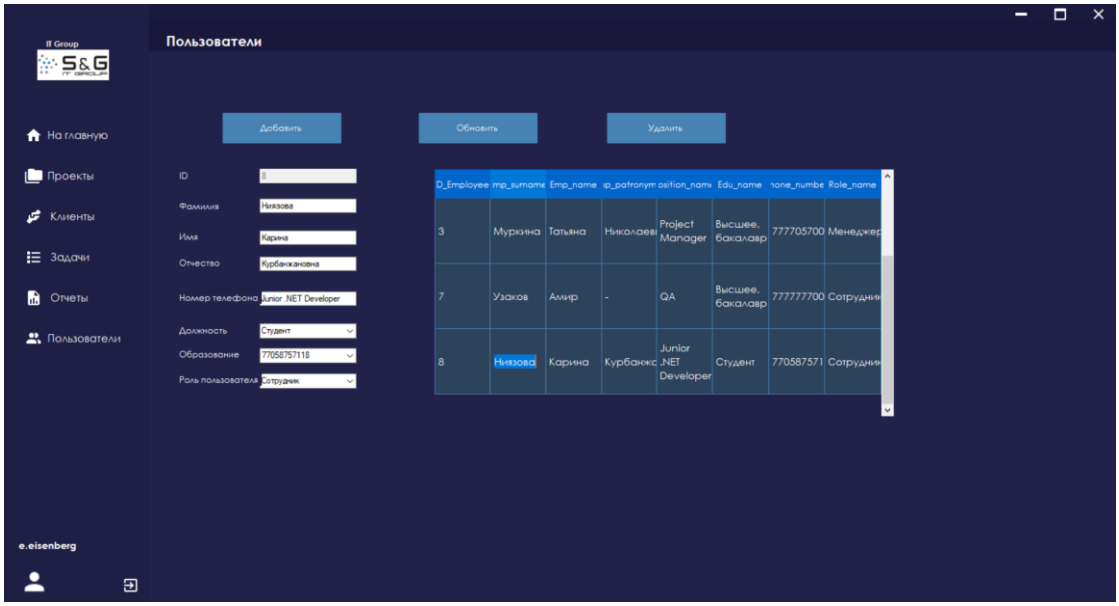


Рисунок 3.27 – Панель администратора

Пользователи «Администратор» и «Менеджер» могут добавлять новых клиентов. В окне «Клиенты» становится доступной кнопка «Добавить клиента», по нажатию которой открывается форма для заполнения данных, как на рисунке 3.28.

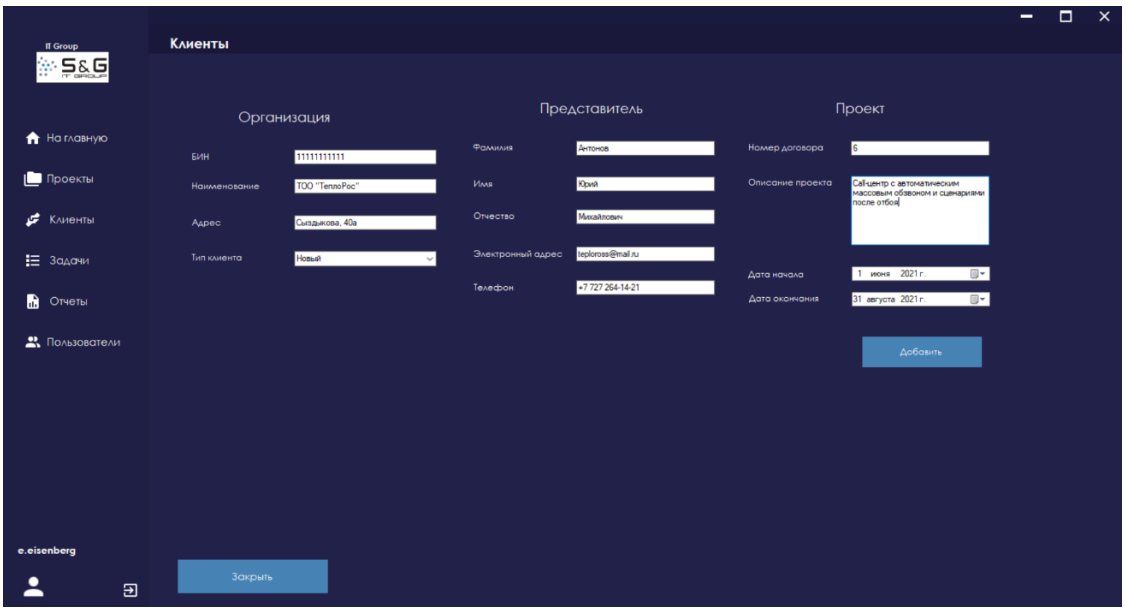


Рисунок 3.28 – Добавление клиента

Также данные пользователи могут добавлять и удалять задачи. Формы добавления и удаления задач представлены на рисунках 3.29 и 3.30.

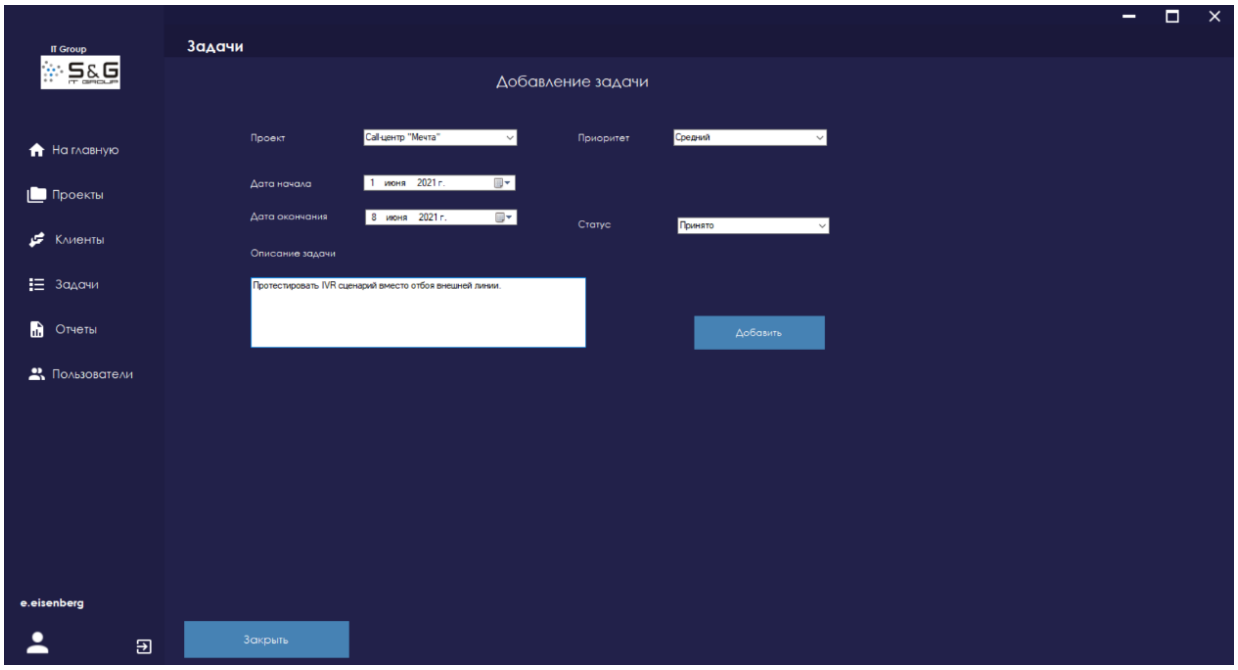


Рисунок 3.29 – Добавление задачи

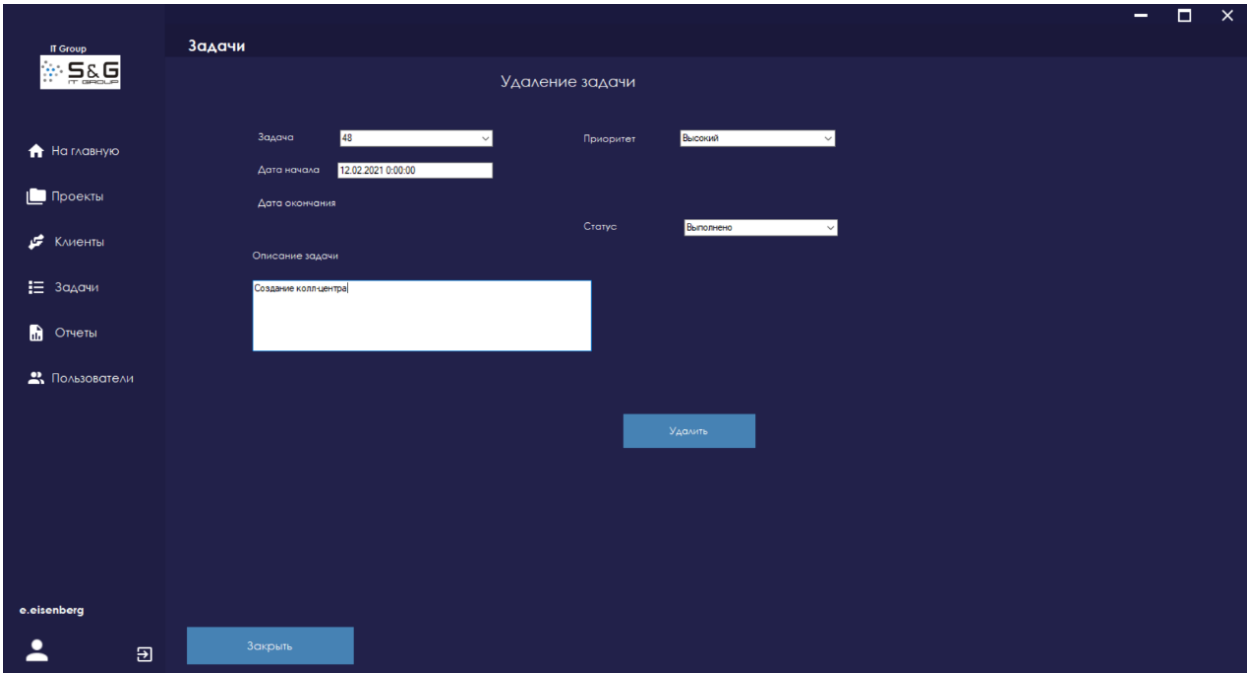


Рисунок 3.30 – Удаление задачи

4 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

4.1 Резюме

Разрабатываемый проект представляет собой информационную систему для информационно-внедренческой компании, предназначенную для управления проектами компании. Цель разработки информационной системы – отслеживание выполнения задач сотрудниками, автоматизация контроля бизнес-показателей компании, сбор и обработка статистики.

Каждый сотрудник – пользователь системы – может просматривать назначенные ему задачи, их сроки, приоритеты и персональную статистику. Это повышает эффективность команды в проектной работе, так как вся информация находится под рукой. Пользователи могут контролировать график выполнения работ и получать оперативную информацию по проектам. Руководитель может просматривать статистику выполнения задач по всем проектам, контролировать их реализацию, поддерживать планирование, анализировать эффективность деятельности.

В данном разделе необходимо провести расчет трудоемкости, себестоимости и экономической эффективности разрабатываемого проекта. На основе данных расчетов можно сделать выводы о целесообразности создания программного продукта.

4.2 Трудоемкость разработки программного продукта

Для определения трудоемкости необходимо разделить работы на этапы, определить нагрузку на каждом из них [24]. В таблице 4.1 приведено разделение работ по этапам.

Таблица 4.1 - Распределение работ и оценка их трудоемкости

№	Этап разработки	Вид работы на данном этапе	Трудоемкость разработки	
			Чел.×час	Час× день
1	Анализ предметной области	Анализ предметной области, установление целей и задач.	1×16	8×2
2	Анализ требований к продукту	Анализ требований, документирование	1×16	8×2
3	Проектирование	Эскизный и технический проект	1×40	8×5

Продолжение таблицы 4.1

4	Реализация	Построение БД, ее заполнение, реализация интерфейса, программирование и отладка	1×160	8×20
5	Тестирование	Тестирование, исправление ошибок и неполадок продукта.	1×24	8×3
6	Внедрение и поддержка	Установка программного обеспечения, исправление выявленных ошибок, ознакомление персонала с продуктом и его сопровождение.	1×16	8×2
Итого трудоемкость выполненного проекта			1×272	8×34

Трудоемкость проекта составила 272 часа, или 34 дня.

4.3 Расчет затрат на разработку программного продукта

Расчет полных затрат на разработку информационных технологий осуществляется по формуле (4.1):

$$C_{ni} = З_{фот} + З_{сзi} + М_i + Р_{сi} + П_{зi} + Р_{нi}\ , \quad (4.1)$$

где $З_{фот}$ – общий фонд оплаты труда разработчиков, тенге;

$З_{сзi}$ – отчисления по социальному налогу, тенге;

$М_i$ – затраты на материалы, тенге;

$Р_{сi}$ – затраты на программные средства, тенге;

$П_{зi}$ – прочие затраты, тенге;

$Р_{нi}$ – накладные расходы, тенге.

Размер фонда оплаты труда разработчиков рассчитывается по формуле (4.2):

$$З_{ФОТ} = З_{оi} + З_{дi}, \quad (4.2)$$

где $З_{оi}$ – основная заработная плата, тенге;

$З_{дi}$ – дополнительная заработная плата, тенге;

Оплата труда разработчика рассчитывается по формуле (4.3):

$$З_о = \sum_{i=1}^{n} ЧС_i \times T_i\ . \quad (4.3)$$

Затраты на оплату труда занесены в таблицу 4.2.

Таблица 4.2 – Затраты на оплату труда

Исполнитель	Трудоемкость, чел. х час	Часовая ставка, тенге/час	Сумма, тенге
Разработчик - программист	1 х 272	900	244 800
Итого затрат на оплату труда			244 800

Дополнительная заработная плата, составляющая в среднем 10% от основной заработной платы, рассчитывается по формуле (4.4):

$$З_{дi} = З_{oi} \times H_{д}/100\ , \qquad (4.4)$$

где $Н_д$ – коэффициент дополнительной заработной платы.

Тогда дополнительная заработная плата составляет:

$$З_{дi} = 244800 \times \frac{10}{100} = 24\ 480 \text{ тенге}$$

Фонд оплаты труда составит:

$$З_{фот} = З_{oi} + З_{дi} = 244800 + 24480 = 269\ 280 \text{ тенге}$$

Социальный налог составляет 9,5% (ст. 358 п. 1 НК РК) от дохода работника, вычисляется по формуле (4.5):

$$СН = \left(З_{фот} - ПО - ОСМС\right) \times 9{,}5\% - З_{соi}, \qquad (4.5)$$

где ПО – пенсионные отчисления, составляют 10% от ФОТ и социальным налогом не облагаются.

Пенсионные отчисления рассчитываются по формуле (4.6):

$$ПО = З_{фот} \times 10\%, \qquad (4.6)$$

Пенсионные отчисления составляют:

$$ПО = 269\ 480 \times 10\% = 26928 \text{ тенге}$$

Отчисления на взнос по обязательному социальному медицинскому страхованию (ОСМС) составляют 2% от заработной платы работника и рассчитываются по формуле (4.7):

$$\text{ОСМС} = \ З_{\text{тр}} \times 2\% \ . \quad (4.7)$$

Отчисления на ОСМС составят:

$$\text{ОСМС} = 269280 \times 2\% = 5385{,}6 \text{ тенге}$$

Социальный налог составляет:

$$\text{СН} = (269280 - 26928 - 5385) \times 9{,}5 - 8482{,}32 = 14029{,}5 \text{ тенге}$$

Социальные отчисления составляют 3,5 % от дохода работника и рассчитываются по формуле (4.8):

$$З_{\text{со}i} = \left(З_{\text{тр}} - \text{ПО}\right) \times 3{,}5\%, \quad (4.8)$$

Социальные отчисления составляют:

$$З_{\text{со}i} = (269280 - 26928) \times 3{,}5\% = 8482{,}32 \text{ тенге}$$

Сумма всех уплаченных налогов составляет:

$$З_{\text{сз}i} = \text{СН} + \text{СО} + \text{ОСМС} = 27\ 897 \text{ тенге.}$$

Затраты на материалы рассчитываются по формуле (4.9):

$$\text{М}_i = \frac{З_{\text{осн}} \times \text{Н}_{\text{мз}}}{100\%}, \quad (4.9)$$

где $\text{Н}_{\text{мз}}$ – норма расхода материалов от заработной платы (3-5%). Тогда величина затрат на материалы при норме расхода 3%:

$$\text{М}_i = \frac{244\ 800 \times 3}{100\%} = 7\ 344 \text{ тенге}$$

Расходы на приобретение технических и программных средств включаются вычисляются по формуле (4.10):

$$P_{ci} = \ \sum_{i=1}^{n} \text{Ц}_{ci}, \quad (4.10)$$

Затраты на электроэнергию рассчитываются по формуле (4.11):

$$З_{\text{э}} = \sum_{i=1}^{n} \text{М}_i \times \text{К}_i \times \text{Т}_i \times \text{Ц}_i, \quad (4.11)$$

где M_i – паспортная мощность электрооборудования;

$К_i$- коэффициент использования мощности, $К_i$ равен 0,7;

$Т_i$ – время работы оборудования;

$Ц_i$ – цена электроэнергии, тг/кВт·ч.

Для разработки понадобятся средства, программное и техническое обеспечение, представленное в таблице 4.3.

Таблица 4.3 – Спецоборудование

Наименование	Кол-во, шт.	Цена за единицу, тг	Сумма, тг
MS Visual Studio 2019	1	0	0
MS SQL Server Express	1	0	0
Мышь Logitech M280	1	8990	9000
Ноутбук Acer Nitro 5	1	300 000	300 000
Итого			309 000

Максимальная мощность потребления энергии ноутбука по паспорту составляет 135 Вт или же 0,135 кВт. Цена электроэнергии с 1 января 2021 года составляет в среднем 18,88 тенге с НДС за 1 кВт/ч.

Следовательно затраты на электроэнергию равны:

$$З_э = 0{,}135 \times 0{,}7 \times 272 \times 18{,}88 = 485.3 \text{ тенге}$$

Для вычисления общих затрат на разработку программного продукта необходимо рассчитать амортизационные отчисления за период разработки по формуле (4.12) [25]:

$$З_{ам} = \frac{Ф \times Н_а \times N}{100\% \times 12 \times t}, \quad (4.12)$$

где Ф – первоначальная стоимость основных производственных фондов;

$Н_а$ – норма амортизации;

N – время использования программного продукта, дни;

t – количество рабочих дней в месяце.

Для программного продукта срок полезного использования Т – 4 года [25]. Норма амортизации принимается равной:

$$Н_а = \frac{100}{Т} = \frac{100}{4} = 25$$

Основные производственные фонды используются только на протяжении периода разработки и тестирования продукта, что значит время использования равно 272 часам или 34 дням. Тогда амортизационные отчисления составят:

$$З_{ам} = \frac{309000 \times 25 \times 34}{100\% \times 12 \times 21} = 10\ 422 \text{ тенге}$$

Прочие затраты вычисляются по формуле (4.13):

$$П_з = З_{оi} \times \frac{Н_{пз}}{100}, \quad (4.13)$$

где $H_{пз}$ - норматив прочих затрат в целом по организации в (%), в дипломной работе нужно брать 20%. Прочие затраты равны:

$$П_з = 244800 \times \frac{20}{100} = 48960 \text{ тенге}$$

Накладные расходы рассчитываются по формуле (4.14):

$$P_{нi} = З_{оi} \times \frac{Н_{рн}}{100}, \quad (4.14)$$

где $Р_{нi}$ - накладные расходы на конкретную ПО, тенге;

$Н_{нр}$ - норматив накладных расходов в целом по организации в (%), в дипломной работе нужно брать 70%. Накладные расходы равны:

$$P_{нi} = 244800 \times \frac{70}{100} = 171360$$

Смета затрат представлена в таблице 4.4. На рисунке 4.1 представлена диаграмма затрат.

Таблица 4.4 – Смета затрат на разработку

Наименование статьи расходов	Условное обозначение	Значение	В процентах от общей суммы
Фонд оплаты труда	$З_{ФОТ}$	269280,00	33,05
Социальный налог	$З_{сзi}$	27897,40	3,42
Материалы	$М_i$	7344,00	0,90
Затраты на электроэнергию	$З_Э$	485,30	0,06
Амортизация	$З_{ам}$	10422,00	1,28
Спецоборудование	$Р_{Сi}$	309000,00	37,93
Прочие затраты	$П_{зi}$	48960,00	6,01

Накладные расходы	$Р_{нi}$	141360,00	17,35
Итого	$С_{пi}$	814750	100,00

Рисунок 4.1 – Затраты на информационные технологии

Прибыль по создаваемому ПО рассчитывается по формуле (4.15):

$$П_{oi} = C_{ni} \times \frac{У_{pni}}{100}, \quad (4.15)$$

где $У_{pni}$ – уровень рентабельности ПО (%), в дипломной работе брать 40- 60%;

$С_{пi}$ – себестоимость ПО, тенге;

Прибыль составит:

$$П_{oi} = 814750 \times \frac{40}{100} = 325\ 900 \text{ тенге}$$

Прогнозируемая цена без налогов рассчитывается по формуле (4.16):

$$Ц_{пi} = C_{ni} + П_{oi} \quad (4.16)$$

Тогда прогнозируемая цена без налогов составит:

$$Ц_{пi} = C_{ni} + П_{oi} = 814\ 750 + 325\ 900 = 1\ 140\ 650 \text{ тенге}$$

Прогнозируемая отпускная цена рассчитывается по формуле (4.17):

$$Ц_{o} = Ц_{ni} + НДС, \quad (4.17)$$

где НДС – ставка на добавленную стоимость, 12% от отпускной цены.

Прогнозируемая отпускная цена равна:

$$Ц_о = 1\,140\,650 + 1\,140\,650 \times 0{,}12 = 1\,277\,528 \text{ тенге.}$$

4.4 Расчет результатов от создания и использования ИС

Информационная система позволяет руководителю контролировать работу по проектам и проводить анализ ее эффективности. Так как компания ведет большое количество проектов, в организации имеется один project-manager с заработной платой 240 000 тенге в месяц. В год это составляет 2 880 000 тенге. В его обязанности входит ведение проектов, в том числе контроль выполнения задач сотрудниками, сбор статистики, подготовка отчетов.

Величина годового экономического эффекта от внедрения информационной системы рассчитывается по формуле (4.18):

$$Э_г = Э_{уг} - К \times Е_н, \quad (4.18)$$

где $Э_г$ – ожидаемый годовой экономический эффект, тенге;

$Э_{уг}$ – ожидаемая условно-годовая экономия, тенге;

К – капитальные вложения, тенге;

$Е_н$ – нормативный коэффициент эффективности капитальных вложений.

$Е_н$ рассчитывается по формуле (4.19):

$$Е_н = \frac{1}{Т_н}, \quad (4.19)$$

где $Т_н$ – нормативный срок окупаемости капитальных вложений, лет. Для программных продуктов срок окупаемости равен 4 годам.

Тогда годовой экономический эффект составит:

$$Э_г = 2\,880\,000 - 1\,277\,528 \times \frac{1}{4} = 2\,560\,618 \text{ тенге}$$

Расчетный коэффициент экономической эффективности капитальных вложений рассчитывается по формуле (4.20):

$$Е_р = \frac{Э_{уг}}{К} \quad (4.20)$$

Тогда коэффициент экономической эффективности капитальных вложений составит:

$$E_р = \frac{Э_{уг}}{К} = \frac{2\,880\,000}{1\,277\,528} = 2{,}2$$

Расчетный срок окупаемости капитальных вложений рассчитывается по формуле (4.21):

$$T_р = \frac{1}{E_р} \quad (4.21)$$

Расчетный срок окупаемости капитальных вложений составит:

$$T_р = \frac{1}{E_р} = \frac{1}{2{,}2} \approx 0{,}5$$

Срок окупаемости составляет 0,5 года или около 6 месяцев. Результаты расчетов внесены в таблицу 4.5, где отображены показатели сравнительной экономической эффективности от внедрения программного продукта.

Таблица 4.5 – Показатели сравнительной экономической эффективности

Наименование показателей	Значение
Условная годовая экономия затрат, тенге	2880000
Коэффициент экономической эффективности капитальных вложений	2,2
Сроки окупаемости капитальных вложений, год	0,5

4.5 Вывод по экономическому разделу

В данном разделе были рассмотрены экономические вопросы по разрабатываемому проекту – информационной системе для управления проектами IT-компании. Проект разделен на шесть этапов, по известным данным о нагрузке на которых был произведен расчет трудоемкости эмпирическим путем. Трудоемкость составила 272 часа, или 34 дня.

Был произведен расчет затрат на разработку программного продукта. В статью расходов «Спецоборудование» были включены лишь затраты на техническое обеспечение, так как программные средства для разработки приобретены бесплатно.

Так как в компании увеличилось число проектов, руководитель не успевает контролировать выполнение каждого из них, как раньше. Для этого был нанят менеджер проектов с заработной платой 240 000. Менеджер проектов занимается контролем работы и ходом выполнения задач по проектам.

После внедрения информационной системы отпадет необходимость содержать в штате менеджера проектов, так как руководитель сможет снова

самостоятельно контролировать работу команды. Таким образом, продукт может сократить затраты на содержание персонала и закуп дополнительного программного обеспечения. Условная годовая экономия затрат составит 2 880 000 тенге, а срок окупаемости проекта составит около 6 месяцев.

5 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

5.1 Анализ потенциально опасных и вредных факторов воздействия на персонал в процессе работы

Здоровый и производительный труд возможен при правильной огранизации и хорошем содержании рабочего места, так как за ним работающий человек проводит большую часть своего времени.

Рабочее место программиста является одним из безопасных рабочих мест, так как работа в офисе не связана с различными производственными рисками. Но при этом охрана труда офисных работников остается актуальной, хотя многие вредные факторы не всегда очевидны. На работников офиса в основном действуют вредные факторы техногенного характера, то есть связанные с техникой.

Вредный производственный фактор (ВПФ) – это производственный фактор, воздействие которого на человека, работающего в определенных условиях, приводит к заболеванию или снижению работоспособности Опасные и вредные производственные факторы в соответствии с ГОСТ 12.0.003–74 подразделяются по природе действия на физические, химические, биологические и психофизиологические [26].

Среди физических вредных производственных факторов, воздействующих на программиста, выделяют:

- электромагнитное излучение;
- статическое электричество;
- повышенный уровень шума;
- неправильное производственное освещение;
- запыленность воздуха рабочей зоны;
- ионизация воздуха.

При работе с персональным компьютером (ПК) человек подвергается электромагнитному излучению, при воздействии которого в организме человека могут развиться нарушения в деятельности сердечно-сосудистой системы, органов дыхания, пищеварения, возможно изменение состава крови и др. На работе сердечно-сосудистой и нервной систем сказывается также статическое электричество [27].

Шум – один из самых распространненных вредных факторов. Максимально допустимый уровень шума в офисном помещении не должен превышать 80 дБ. Если уровень шума превышает данное значение, то при продолжительном воздействии он может привести к заболеваниям нервной и кровеносной систем, снижению слуха и зрения, неспособности сосредоточиться, что ведет к снижению производительности труда. Если уровень шума превышает 80 дБ, то работнику нужно выдавать индивидуальные средства защиты слуха. Долгое влияние шума более 100 дБ может привести к

существенному снижению слуха. Нормы шума на рабочих местах регламентируются ГОСТ 12.1.003-2014 «ССБТ. Шум. Общие требования безопасности» [27].

Предельно допустимые уровни звука на рабочих местах приведены в таблице 5.1.

Таблица 5.1 - Предельные уровни звука на рабочих местах

Категория напряженности труда	Категория тяжести труда			
	I. Легкая	II.Средняя	III.Тяжелая	IV.Очень тяжелая
I. Мало напряженная	80	80	75	75
II.Умеренно напряженный	70	70	65	65
III.Напряженный	60	60	–	–
IV.Очень напряженный	50	50	–	–

Неотъемлемым элементом условий трудовой деятельности является освещение. Правильная организация производственного освещения позволяет обеспечить сохранность зрения человека и нормальное состояние его нервной системы, а также безопасность на производстве. Недостаточное же освещение, несоответствующее характеру выполняемой работы, вызывает утомление глаз, что способствует проявлению либо прогрессированию близорукости [27].

Работа за персональным компьютером относится к зрительным работам высокой точности, поэтому поле зрения должно быть освещено равномерно, то есть освещение помещения и яркость экрана должны быть примерно одинаковы. При этом свет должен иметь правильно направление, а освещение не должно вызывать ослепленности.

Нормы освещенности регламентируются в СНиП РК 2.04-05-2002 «Естественное и искусственное освещение. Нормы проектирования». Согласно этому документу в помещении должны использоваться системы комбинированного освещения, освещенность поверхности рабочего стола должна составлять не менее 500 лк, освещенность поверхности экрана должна быть не более 200 лк [28].

Неблагоприятным фактором производственной среды является пыль, которая наносит большой вред человеческому организму. Вредность воздействия пыли зависит от количества вдыхаемой пыли и ее химического состава. Пыль также может быть носителем микробов, клещей и др [29].

Воздушный комфорт в помещении также определяется уровнем положительной и отрицательной аэроинонизации. Отрицательно влияют на организм человека как избыточная, так и недостаточная ионизация воздуха. Отрицательные ионы способствуют снижению пыли в воздухе, неутрализуют некоторые газы, устраняют электростатические заряды с поверхностей. Уменьшение числа отрицательных ионов приводит к потере воздухом его

освежающих свойств, что неблагоприятно воздействует на организм человека. Оптимальный уровень ионизации воздушной среды подразумевает содержание в 1 $см^3$ 1500-3000 ионов положительной полярности и 3000-5000 ионов отрицательной полярности. Искусственная ионизация воздуха должна проводится только при достаточном воздухоснабжении, иначе это может оказать негативный эффект [29, 30].

К химическим вредным производственным факторам относят повышенное содержание в воздухе рабочей зоны двуокиси углерода, озона, аммиака, фенола, формальдегида и полихлорированных бифенилов. При работе печатного и копировального оборудования выделяется большое количество озона. Повышенное содержанием озона приводит к появлнию таких симптомов, как усталость, сонливость, утомление [30].

К биологическим факторам относят повышенное содержание в воздухе рабочей зоны микроорганизмов. Они возникают при неправильном воздухообмене, несвоевременной чистки систем вентиляции и несоблюдении правил уборки помещений [30].

Среди психофизиологических факторов выделяют:

- напряжение зрения;
- длительные статические нагрузки;
- монотонность труда;
- эмоциональные и интеллектуальные нагрузки;
- неправильная организация рабочего места.

При работе за компьютером большая нагрузка приходится на зрение. Несоблюдение зрительного режима может привести к ухудшению зрения. Длительная работа за компьютером приводит к появлению таких симптомов зрительного утомления, как сухость в глазах, жжение, покраснения, головная боль. Для уменьшения зрительного напряжения необходимо установить оптимальный цветовой режим на экране монитора. Монитор должен находиться на расстоянии не ближе 50 см. Рекомендуется делать перерывы при работе за компьютером, во время которых нужно смотреть вдаль, то есть возвращая глаз в состояние покоя, и делать зрительную гимнастику [31].

Работа программиста подразумевает постоянную сидячую работу, вызывающую напряжение и усталость мышц. Пониженная подвижность – гиподинамия – вредна для опорно-двигательного аппарата и приводит к застою крови во внутренних органах и капиллярах, мышечной слабости и изменению формы позвоночника. Для профилактики гиподинамии необходимо делать разминку во время перерывов в работе [31].

Сидячая работа также является признаком монотонности, так как рабочие действия не отличаются многообразием. Монотонная работа усиливает нервное напряжение, раздражительность, приводит к снижению двигательной активности, повышенной заболеваемости и ухудшению общего функционального состояния организма. Уменьшить монотонию можно, внедрив короткие перерывы, разминки в режим рабочего дня [32].

Интеллектуальные нагрузки – нагрузки при выполнении работы, связанной с решением сложных задач. Такие нагрузки тесно связаны с эмоциональными нагрузками, так как с возрастанием сложности решаемых задач увеличивается эмоциональное напряжение. Эмоциональные и интеллектуальные нагрузки вместе являются составляющими степени тяжести и напряженности работы. Повышение таких нагрузок приводит к ухудшению концентрации и внимания, утомляемости и стрессу [32].

Неправильная организация рабочего места, а точнее несоблюдение правил его эргономики, только усугубляет действие перечисленных выше вредных производственных факторов. На рабочем месте программиста главными элементами являются стол и кресло.

Стол желательно подбирать слегка изогнутой формы, длина столешницы должна быть 50-80 см, высота стола – 80 см. На поверхности стола не должны быть блики. Высота сидения рабочего кресла должна составлять 40-50 см, передний край кресла должен быть закруглен, а угол наклона спинки должен регулироваться. Для комфортной работы также рекомендуется оборудовать рабочее место подставкой для ног высотой до 15 см. Монитор должен быть установлен на расстоянии вытянутой руки с наклоном до 15° по отношению к пользователю. Клавиатура должна находиться близко к монитору, мышь следует расположить на одной поверхности с клавиатурой. Часто используемые предметы должны находиться в зоне досягаемости [32].

Правильная позиция при работе за компьютером приведена на рисунке 5.1.

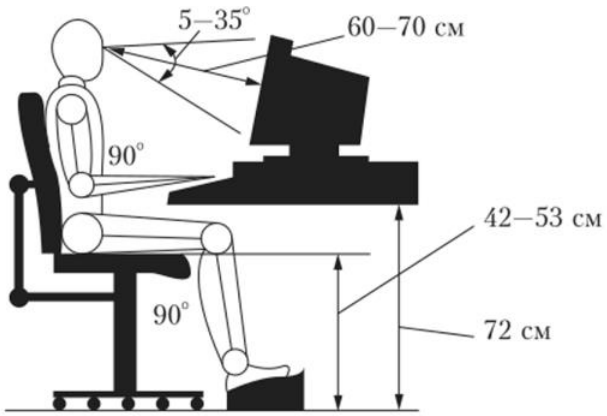


Рисунок 5.1 – Правильная позиция во время работы за компьютером

5.2 Расчетная часть

5.2.1 Акустический расчет

Шум – это нежелательные звуки, оказывающие вредное воздействие на организм человека. Длительный шум снижает остроту слуха и зрения, утомляет центральную нервную систему, из-за чего ослабляется внимание и может

возрасти количество ошибок, снижается производительность труда. Необходимость проведения мероприятий по снижению шума определяется на основании акустического расчета.

Допустимые уровни звукового давления регламентируются Межгосударственным стандартом ГОСТ 12.1.003-2014 «Система стандартов безопасности труда (ССБТ). Шум», а также СН РК 2.04-02-2011 «Защита от шума».

Основными источниками шума на рабочем месте программиста являются системы охлаждения и периферийные устройства. Уровни звукового давления источников шума, действующих на рабочем месте, представлены в таблице 5.2.

Таблица 5.2 – Уровни звукового давления

Источник шума	Уровень шума, дБ
Компьютер	30
Лазерный принтер	45
Кондиционер	45

Допустимым уровнем шума считается 50 дБ [33].

Расчет шума внутри помещений проводится по формуле (5.1):

$$L_{\text{сум}} = 10\lg\left(\sum_{i=1}^{m}\frac{\Delta_i\cdot\chi_i\cdot\Phi}{S_i} + 4\frac{\psi}{B}\sum_{i=1}^{n}\Delta_i\right), \quad (5.1)$$

где $\Delta_i = 10^{0.1L_{pi}}$, L_{pi} – октавный уровень звуковой мощности в дБ, создаваемый i-тым источником шума;

m – количество источников шума, ближайших к расчетной точке;

χ – коэффициент , учитывающий характер звукового поля в зависимости от расположения расчетной точки и максимального габаритного размера источника;

S – площадь, м^2;

B – постоянная помещения, м^2;

ψ - коэффициент, учитывающий нарушение диффузности звукового поля в помещении;

Ф – фактор направленности (Ф=1);

n – общее число источников шума в помещении.

На частоте 1000 Гц уровни громкости приняты равными уровням звукового давления. Поэтому в расчетах будут рассмотрены уровни звукового давления на частоте 1000 Гц.

В помещении размещено 5 источников шума двух типов: 3 компьютера одного типа, второй тип – принтер и кондиционер. Источники первого типа расположены от расчетной точки на расстоянии 1 м, источники второго типа – на расстоянии 2,5 м.

Наибольший габаритный размер рассматриваемых источников l_{max}=1 м, следовательно для всех источников выполняется условие 2·0,75<5, поэтому можно принять $S_i=2\pi r^2$:

$$S_1 = S_2 = S_3 = 2 \cdot 3{,}14 \cdot 1 = 6{,}28 \text{ м}^2$$
$$S_4 = S_5 = 2 \cdot 3{,}14 \cdot 2{,}5 = 39{,}27 \text{ м}^2$$

Постоянная помещения В определяется по формуле (5.2):

$$B = B_{1000} \cdot \mu, \text{ м}^2, \quad (5.2)$$

где $B_{1000}=\frac{V}{k}$ – постоянная помещения на среднегеометрической частоте 1000 Гц, м^2;

V – объем помещения, м^3;

k – коэффициент , учитывающий тип помещения.

В данном случае объем помещения составляет 48,6 м^3, а коэффициент k равен 10. Тогда B_{1000} = 4,86 м^2.

Частотный множитель μ на частоте 1000 Гц при любых объемах помещения принимается как 1.

Коэффициент ψ берется в зависимости от отношения $B/S_{огр}$. В данном случае $B/S_{огр}$=0.2, значит ψ = 0,85. Коэффициент χ для ИШ первого типа равен 2, для второго типа χ=1.

Октавный уровень для ИШ первого типа:

$$\Delta_1 = \Delta_2 = \Delta_3 = 10^{0{,}1 \cdot 30} = 1000 \text{ дБ}.$$

Октавный уровень для ИШ второго типа:

$$\Delta_4 = \Delta_5 = 10^{0{,}1 \cdot 45} = 31622{,}78 \text{ дБ}.$$

Первая часть формулы (5.1) вычисляется следующим образом:

$$\sum_{i=1}^{5} \frac{\Delta_i \cdot \chi_i \cdot \Phi_ш}{S_i} = 3 \cdot \frac{1000 \cdot 2 \cdot 1}{6{,}28} + 2 \cdot \frac{31622{,}78}{39{,}27} = 2565{,}5$$

Второе слагаемое вычисляется как:

$$\frac{4 \cdot \psi}{B} \cdot \sum_{i=1}^{n} \Delta_i = 0{,}7 \cdot (3 \cdot 1000 + 2 \cdot 31622{,}78) = 46344{,}63$$

Тогда общий уровень звукового давления будет равен:

$$L_{\text{общ}} = 10\lg(2565.5 + 46344.63) = 46.9 \approx 47 \text{ дБ}$$

Полученное значение не превышает допустимого уровня звукового давления. Также необходимо отметить, что одновременная работа всех источников шума маловероятна, поэтому фактический уровень шума будет меньше полученного значения.

5.2.2 Расчет времени эвакуации людей

Опасность для работников несут возгорания на производственных помещениях. Главным требованием противопожарной безопасности является возможность произвести эвакуацию людей в короткое время. Пожарная безопасность регламентируется ГОСТ 12.1.004 – 91 «Пожарная безопасность. Общие требования», противопожарные требования также изложены в СН РК 2.02-01-2019.

Эвакуационными выходами считают дверные проемы, ведущие наружу, или проходы, коридоры с выходом наружу. Эвакуационный пути – пути, ведущие к эвакуационным выходам и обеспечивающее безопасное движение людей в течение определенного времени.

Сокращение времени эвакуации достигается путем правильного построения плана. Чем меньше продолжительность эвакуации, тем меньше вероятность возникновения опасных последствий для людей. Необходимое время эвакуации определяют по таблицам, приведенным в СНиП II-2-80.

В данном расчете рассматривается офисное помещение, расположенное на 4 этаже бизнес-центра. Здание по взрывной, взрывопожарной и пожарной опасности относится к категории Д – производства с непожароопасными технологическими процессами, где имеются несгораемые вещества и материалы в холодном состоянии, степень огнестойкости II. Для таких зданий, согласно СНиП II-2-80, необходимое время эвакуации может длиться до 6 минут. Согласно СНиП II-2-80 ширина эвакуационных путей должна составлять не менее 1 м, минимальная ширина дверей – 0.8 м, высота прохода – не менее 2 м [34].

В данном случае офис находится рядом с лестничным маршем на расстоянии 3 метров. Количество эвакуирующихся работников равно 4. Ширина коридоров равна 2 метрам, ширина лестничного марша составляет 1,5 метра. Длина межэтажной площадки равна 1,9 метров.

Имеются как горизонтальные, так и наклонные пути эвакуации. Горизонтальные пути – участки, имеющие горизонтальный уровень пола, наклонные пути – лестницы и пандусы.

Расчетная длина лестничного марша вычисляется по формуле (5.3):

$$l = \frac{l'}{\cos\alpha}, \tag{5.3}$$

где l' - горизонтальная проекция длины наклонного участка, м;
α – угол наклона к горизонтали.

В данном случае длина составит:

$$l = \frac{2,4}{\cos 35^\circ} \approx 2,9 \text{ м.}$$

Для упрощения вычислений допускается принимать небольшие межэтажные площадки за наклонные участки. Тогда длина площадки составит:

$$l = \frac{1,9}{\cos 35^\circ} \approx 2,3 \text{ м.}$$

Общая длина пути с одного этажа на следующий составит 8,2 м.

Эвакуационный путь разделен на несколько участков. Первый участок – путь от кабинета до лестничной площадки. Плотность потока характеризует степень свободы перемещения людей на участке и определяется по формуле (5.4):

$$D = \frac{N_i \cdot f}{\delta_i \cdot l_i}, \qquad (5.4)$$

где N – количество людей на участке;

f – средняя площадь горизонтальной проекции человека, м^2 (f=0,1),

δ_i – ширина участка, м;

Плотность потока на первом участке составит:

$$D_1 = \frac{4 \cdot 0,01}{3 \cdot 2} = 0,07 \text{ м}^2/\text{м}^2.$$

Интенсивность и скорость движения определяются по СНиП II-2-80, значения которых указаны в таблице 5.3, с учетом интерполяции, и в данном случае составят соответственно $q_1 = 6.2$ м/мин, $v_1 = 92$ м/мин. Так как $q_1 < q_{max} = 16$ м/мин, то задержки происходить не будет.

Скорость движения зависит от плотности потока и условий эвакуации. Интенсивность движения характеризует кинетику движения людского потока и равна количеству людей, проходящих через отрезок пути единичной ширины в единицу времени [35].

Таблица 5.3 – Скорость движения людского потока

Плотность потока D м^2/м^2	Горизонтальный путь		Дверной проем	Лестница вниз		Лестница вверх	
	скорость v, м/мин	интенсивность q, м/мин	интенсивность q, м/мин	скорость v, м/мин	интенсивность q, м/мин	скорость v, м/мин	интенсивность q, м/мин

Продолжение таблицы 5.3

0,01	100	1	1	100	1	60	0,6
0,05	100	5	5	100	5	60	3
0,1	80	8	8,7	95	9,5	53	5,3
0,2	60	12	13,4	68	13,6	40	8
0,3	47	14,1	16,5	52	15,6	32	9,6
0,4	40	16	18,4	40	16	26	10,4
0,5	33	16,5	19,6	31	15,5	22	11
0,6	27	16,2	19	24	14,4	18	10,8
0,7	23	16,1	18,5	18	12,6	15	10,5
0,8	19	15,2	17,3	13	10,4	13	10,7
0,9 и более	15	13,5	8,5	8	7,2	11	9,9

Время движения людского потока на участке представляет собой отношение длины участка к скорости движения и рассчитывается по формуле (5.5):

$$t = \frac{l}{v}, \tag{5.5}$$

где l – длина участка пути;

v – скорость движения людского потока на данном участке.

Время движения людского потока составит:

$$t_1 = \frac{3}{92} = 0{,}033 \text{ мин.}$$

Второй участок – путь по лестничному маршу с четвертого этажа на третий. Плотность потока:

$$D_2 = \frac{4 \cdot 0{,}01}{8{,}2 \cdot 1{,}5} = 0{,}03 \text{ м}^2/\text{м}^2.$$

Интенсивность движения составит $q_2 = 2{,}9$ м/мин, а скорость равна $v_2 = 100$ м/мин. Так как $q_2 < q_{\text{пред}}$, задержки движения не будет. Время движения людского потока составит:

$$t_2 = \frac{8{,}2}{100} = 0{,}082 \text{ мин.}$$

Третий участок – путь на третьем этаже. На третьем этаже работает 20 сотрудников, для этого участка плотность потока составит:

$$D_3 = \frac{20 \cdot 0{,}01}{12 \cdot 2} = 0{,}083 \text{ м}^2/\text{м}^2.$$

Интенсивность движения составит $q_3 = 6{,}98$ м/мин, а скорость равна $v_3 = 86{,}8$ м/мин. Задержки движения не будет. Время движения людского потока составит:

$$t_3 = \frac{12}{86{,}8} = 0{,}14 \text{ мин.}$$

На четвертом участке пути произойдет слияние потоков людей с четвертого этажа с работниками с третьего этажа, так как здесь расположен лестничный марш с третьего этажа на второй. При слиянии потоков используется формула (5.6), выведенная из уравнения неразрывности людского потока:

$$q_i = \frac{\sum q_{i-1} \cdot \delta_{i-1}}{\delta_i}, \quad (5.6)$$

где q_{i-1} – интенсивность движения людского потока, на предыдущем участке м/мин;

δ_{i-1} – ширина предыдущего участка, м;

δ_i – ширина рассматриваемого участка пути, м.

Интенсивность движения людского потока на четвертом участке составит:

$$q_4 = \frac{2{,}9 \cdot 1{,}5 + 6{,}98 \cdot 2}{1{,}5} = 12{,}2 \text{ м/мин.}$$

Скорость движения на участке составит $v_4 = 77{,}22$ м/мин. Задержки не будет. Время движения составит:

$$t_4 = \frac{8{,}2}{77{,}22} = 0{,}12 \text{ мин.}$$

Пятый участок – второй этаж здания, на котором работают 30 человек. Плотность потока составит:

$$D_5 = \frac{30 \cdot 0{,}01}{12 \cdot 2} = 0{,}125 \text{ м}^2/\text{м}^2.$$

Интенсивность движения составит $q_3 = 9$ м/мин, а скорость равна $v_3 = 75$ м/мин. Время движения будет равно:

$$t_5 = \frac{12}{75} = 0{,}16 \text{ мин.}$$

Шестой участок – лестничный марш со второго на первый этаж, на котором произойдет слияние потоков.

Интенсивность определяется по формуле (5.4):

$$q_4 = \frac{12{,}2 \cdot 1{,}5 + 9 \cdot 2}{1{,}5} = 24{,}2 \text{ м/мин.}$$

Интенсивность на данном участке превышает предельную $q_{\text{пред}} = 16$ м/мин, следовательно, произойдет задержка.

Если интенсивность движения превышает предельное значение, то необходимо увеличить ширину участка пути. Если это сделать невозможно, то интенсивность и скорость движения людского потока по участку пути определяют по таблице 4.3 при значении D=0,9 и более.

Время задержки определяется по формуле Предтеченского (5.7):

$$t^з = N \cdot f \cdot \left(\frac{1}{q_{\text{пред}} \cdot \delta_i} - \frac{1}{\sum q_{i-1} \cdot \delta_{i-1}}\right), \qquad (5.7)$$

где t^3 – время задержки на i участке;

N – количество эвакуирующихся;

f – средняя площадь горизонтальной проекции человека, м^2/чел;

δ_{i-1}, δ_i – ширина предыдущего и последующего участков, м;

l_i – длина участка, на котором происходит задержка движения, м;

q_{i-1} – интенсивность движения на предыдущем участке, м/мин;

$q_{\text{пред}}$ – предельное значение интенсивности движения потока.

Далее в расчетах предельные параметры интенсивности находятся по таблице 4 приложения 1 СНиП II-2-80 при значении D=0,09 и более.

Так как рассматриваемый участок – лестница вниз, то q =7,2 м/мин, v = 8 м/мин.

На шестом участке задержка будет равна:

$$t_6^3 = 54 \cdot 0{,}1 \cdot \left(\frac{1}{7{,}2 \cdot 1{,}5} - \frac{1}{(12{,}2 \cdot 1{,}5 + 9 \cdot 2)}\right) = 0{,}351 \text{ мин.}$$

Время движения с учетом задержки будет равно:

$$t_6 = \frac{8{,}2}{8} + 0{,}351 = 1{,}025 + 0{,}351 = 1{,}376 \text{ мин}$$

Седьмой участок – первый этаж, на котором работают 30 человек. Все параметры совпадают с пятым участком. Восьмой участок – выход с первого этажа наружу. На этом участке произойдет слияние потоков работников со всех этажей. Поэтому интенсивность движения людского потока определяется по формуле (5.4):

$$q_8 = \frac{7{,}2 \cdot 1{,}5 + 9 \cdot 2}{1{,}5} = 19{,}2 \text{ м/мин.}$$

Так как интенсивность превышает предельное значение, то на этом участке также произойдет задержка. Так как рассматриваемый участок горизонтальный, то q =13,5 м/мин, v = 15 м/мин. Время задержки составит:

$$t_8^3 = 84 \cdot 0{,}1 \cdot \left(\frac{1}{13{,}5 \cdot 1{,}5} - \frac{1}{(7{,}2 \cdot 1{,}5 + 9 \cdot 2)}\right) = 0{,}123 \text{ мин.}$$

Время движения с учетом задержки будет равно:

$$t_8 = \frac{5}{15} + 0{,}123 = 0{,}33 + 0{,}123 = 0{,}456 \text{ мин}$$

Общее время эвакуации рассчитывается как сумма времени движения людского потока по отдельным участкам пути (5.8):

$$t_p = t_1 + t_2 + \cdots + t_i \ . \qquad (5.8)$$

Общее время эвакуации составит:

$$t_p = 0{,}033 + 0{,}082 + 0{,}14 + 0{,}12 + 0{,}16 + 1{,}376 + 0{,}16 + 0{,}456 = 2{,}53 \text{ мин}$$

Расчетное время эвакуации для рассматриваемого офиса не превышает допустимого значения. Этажный план эвакуации из офисного помещения представлен на рисунке 5.2.

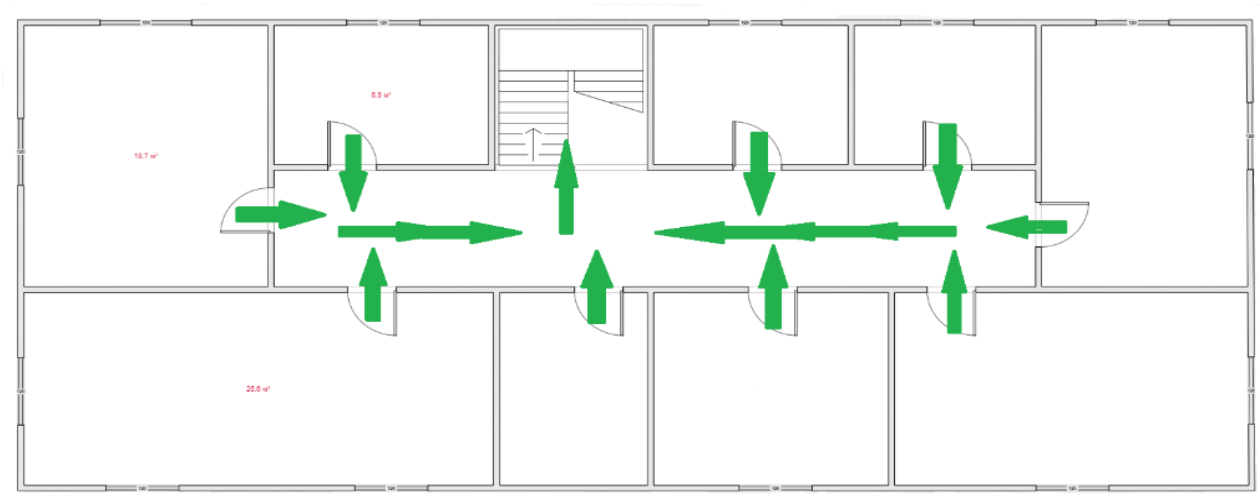


Рисунок 5.2 – План эвакуации из офисного помещения

ЗАКЛЮЧЕНИЕ

Результатом выполнения данного дипломного проекта является информационная система «IT Group» для управления проектами информационно-внедренческой компании «S&G IT Group». Система реализована в виде настольного приложения (desktop).

В ходе работы над проектом была изучена предметная область – структура компании и ее основные бизнес-процессы. Были спроектированы функциональная структура ИС, функциональная и информационные модели, информационное обеспечение (логическая и физическая модели базы данных, выбор СУБД и описание структур таблиц), UML-диаграммы (диаграммы прецедентов, диаграммы последовательностей, кооперативная диаграмма). Также были проведены анализ и оптимизация бизнес-процессов.

Разработка программного обеспечения была выполнена в среде MS Visual Studio 2019 на языке программирования C#. В роли СУБД был использован MS SQL Server.

Разработанная информационная система упрощает работу команды разработчиков, оперативно предоставляя информацию о текущих задачах и проектах. Также информационная система позволяет руководителю самостоятельно контролировать ход работы над проектами, генерируя отчеты по собранной статистике. Настольный вид приложения позволяет сотрудникам работать, не отвлекаясь на вкладки в браузере.

В экономической части был произведен расчет стоимости разработки программного продукта и срок окупаемости. По результатам расчета стоимость программного продукта составила 1 277 528 тенге. Срок окупаемости проекта – полгода.

В разделе безопасности жизнедеятельности был проведен анализ вредных факторов, воздействующих на персонал в процессе разработки информационной системы. Были произведены акустический расчет и расчет времени эвакуации людей.

СПИСОК ЛИТЕРАТУРЫ

1 Фленов М. Е. Библия C#. – 4-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2019. – 512 с.: ил.

2 Троелсен, Эндрю, Джепикс, Филипп. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ. – СПб. : ООО “Диалектика”, 2018 – 1328 с. : ил. – Парал. тит. англ.

3 Осипов Д. Л. Технологии проектирования баз данных. – М.: ДМК Пресс, 2019. – 498 с.: ил.

4 Федорова Г.Н. Разработка и администрирование баз данных : учеб. для студ. учреждений сред. проф. образования. – М. : Издательский центр «Академия», 2015. – 320 с.

5 Информационная система управления проектами (ИСУП) // MAHAMBA.COM: сервис вопросов и ответов для специалистов в области менеджмента. URL: http://mahamba.com/ru/informacionnaya-sistema-upravleniya-proektami-isup (дата обращения: 05.03.2021)

6 MS PROJECT – программа для планирования и управления проектами // УЧИСЬУЧИСЬ.РФ: федеральный образовательный портал. URL: https://xn--h1aa0abgczd7be.xn--p1ai/blog/Perspektivy-v-obrazovanii/ms-project-programma-dlya-planirovaniya/ (дата обращения: 05.03.2021)

7 Jose Maria Delos Santos. Trello Software Review for 2021 // PROJECT-MANAGEMENT.COM: best project management software reviews. URL: https://project-management.com/trello-software-review/ (дата обращения: 06.03.2021)

8 What is Jira used for? // ATLASSIAN.COM: инструменты для разработки ПО и совместной работы. URL: https://www.atlassian.com/software/jira/guides/use-cases/ what-is-jira-used-for#Jira-for-requirements-&-test-case-management (дата обращения: 09.03.2021)

9 Описание продукта JetBrains YouTrack // ITPRO.UA. URL: https://itpro.ua/product/jetbrains-youtrack/?tab=description (дата обращения: 09.03.2021)

10 ИлюшечкинВ. М. Основы использования и проектирования баз данных : учебник для СПО / В. М. Илюшечкин. – М. : Издательство Юрайт, 2019. – 213 с. – Серия : Профессиональное образование.

11 Сергей Кривошеев //COMPRICE.RU: Сравнительная характеристика СУБД MS SQL Server и Informix OnLine Dynamic Server URL: http://www.comprice.ru/articles/detail.php?ID=42530 (дата обращения 10.03.2021)

12 Митин, А. И. Работа с базами данных Microsoft SQL Server: сценарии практических занятий / А. И. Митин. – Москва ; Берлин : Директ-Медиа, 2020. – 142 с.

13 Новиков Б. А. Основы технологий баз данных: учеб. пособие / Б. А. Новиков, Е. А. Горшкова; под ред. Е. В. Рогова. – М.: ДМК Пресс, 2019. – 240 с

14 Куликов, С. С. Реляционные базы данных в примерах : практическое пособие для программистов и тестировщиков / С. С. Куликов. – Минск: Четыре четверти, 2020. – 424 с

15 Фаулер М. UML. Основы, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 192 с., ил.

16 Руководство по использованию Конструктора Windows Forms. URL: https://docs.microsoft.com/ru-ru/visualstudio/designers/walkthrough-windows-forms-designer?view=vs-2019 (дата обращения 12.03.2021)

17 Общие сведения о Visual Studio. URL: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019 (дата обращения 13.03.2021)

18 Албахари, Джозеф, Албахари, Бен. C#7.0. Карманный справочник. : Пер. с англ. – Пер. с англ. – СпБ. : ООО «Альфа-Книга», 2017. – 224 . : ил. – Парал. тит. англ.

19 Пахомов Б. И. C# для начинающих. – СПб.: БХВ-Петербург, 2014. – 432 с.: ил.

20 Евдокимов П.В. C# на примерах. – СПб.: Наука и Техника, 2016. – 304 с., ил.

21 Введение в ADO.NET. URL: https://metanit.com/sharp/adonet/1.1.php (дата обращения 13.03.2021)

22 Введение в MS SQL Server и T-SQL. URL: https://metanit.com/sql/sqlserver/1.1.php (дата обращения 15.03.2021)

23 К.К. Ниязова, Б.Б. Тусупова – О разработке информационной системы для информационно-внедренческой компании / К.К. Ниязова, Б.Б. Тусупова // материалы «Computer Science and Cyber Security - 2021»: Международная научно-практическая конференция, Алматы, Казахстан, 28 апреля 2021 года = «Computer Science and Cyber Security - 2021»: Халықаралық ғылыми-тәжірибелік конференциясының материалдары. 28 ақпан 2021 ж. Алматы, Қазақстан = «Computer Science and Cyber Security - 2021»: International Scientific and Practical Conference, Almaty, Kazakhstan, April 28, 2021. – Алматы: Қ.И. Сәтбаев атындағы ҚазҰТЗУ, 2015. – қазақша, орысша, ағылшынша.

24 Г. Боканова Методические указания по выполнению экономической части дипломных работ Алматы, АУЭС, 2020 – 35с.

25 Бекишева А.И. Методические указания к выполнению экономической части дипломной работы для бакалавров специальности 5В0703 – Информационные системы – Алматы: АУЭС; 2013. – 24 с.

26 Безопасность жизнедеятельности. Учеб. пособие для вузов / Под ред. проф. Л.А. Муравья. – 2-е изд., перераб. и доп. – М.: ЮНИТИ-ДАНА, 2010. – 431 с.

27 Назаренко О.Б. Безопасность жизнедеятельности: учебное пособие / О.Б. Назаренко, Ю.А. Амелькович; Томский политехнический университет. – 3-е изд., перераб. и доп. – Томск: Изд-во Томского политехнического университета, 2013. – 178 с.

28 СНиП РК 2.04-05-2002 «Естественное и искусственное освещение.

Нормы проектирования».

29 Зотов Б.И. Безопасность жизнедеятельности на производстве: Учебник для студентов вузов, обучающихся по специальностям 311300, 311500, 311900 / В.И. Курдюмов. – 2-издание, переработанное и дополненное. – М.: Колос, КолосС, 2003. – 432 с. : ил. – (Учебники и учебные пособия для студентов высших уч)

30 Глебова, Е. В. Производственная санитария и гигиена труда: Учебное пособие для вузов / Е.В. Глебова. – 2-е издание, переработанное и дополненное — М.: Высшая школа, 2007. – 382с: ил.

31 Дыхан, Л.Б. Безопасность труда при работе на персональной электронновычислительной машине (ПЭВМ) : учебное пособие / Дыхан Л. Б. : Южный федеральный университет. – Таганрог : Издательство Южного федерального университета, 2016. – 128 с., ил.

32 Безопасность жизнедеятельности. Безопасность технологических процессов и производств (охрана труда): Учебное пособие для вузов./ П.П. Кукин и др. - Из-во «Высшая школа», 2002. - 318 с.

33 ГОСТ 12.1.003-2014 «Система стандартов безопасности труда (ССБТ). Шум»

34 СНиП II-2-80 «Противопожарные нормы проектирования зданий и сооружений»

35 Абикенова А.А., Санатова Т.С. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Пожарная профилактика» в выпускных работах для всех специальностей. Бакалавриат – Алматы: АИЭС, 2009. – 32 с.

ПРИЛОЖЕНИЕ А

Техническое задание на разработку системы

1 Общие сведения

1.1 Наименование системы

Информационная система управления организацией информационно-внедренческой компании «IT Group»

1.2 Сроки начала и окончания работ

Дата начала: 01.02.2021
Дата окончания: 25.05.2021

2 Назначение и цели создания системы

2.1 Назначение системы

Информационная система предназначена для автоматизации контроля сотрудников и бизнес-показателей, организации оперативной и хронологической отчетности, сбора, обработки и актуализации статистики, отслеживания выполнение задач сотрудниками.

3 Характеристика объектов автоматизации

Объектом автоматизации являются бизнес-показатели ТОО «S&G IT Group». Приложение должно автоматизировать составление отчетов и сбор статистики и представить их в виде, позволяющем получить полное представление о состоянии задач. Необходимо разработать подсистемы для таких пользователей, как менеджер, администратор и сотрудник.

4 Требования к системе

4.1 Требования к системе в целом

ИС должна быть централизованной, т.е. все данные должны располагаться в центральном хранилище.

В Системе предлагается выделить следующие функциональные подсистемы:

- подсистема «Офис», в которой каждый сотрудник может работать над своими задачами и проектами, просматривать и отмечать их статусы;

- подсистема «Администрирование», которая отвечает за добавление новых сотрудников, задач и проектов;
- подсистема «Отчеты», в которой выполнятся построение отчетов и сбор статистики по задачам и проектам.

4.2 Требования к функциям, выполняемым системой

Требования к интерфейсу приложения: удобство, дружественность, понятность. Интерфейс должен содержать окна авторизации, просмотра отчетов и статистики, добавление данных о сотрудниках, добавление задач.

Подсистема «Офис» позволяет сотрудникам просматривать информацию о компании, объявления, отмечать статус заданий, выполняемых в рамках проектов. Также предоставляются контакты других сотрудников в модуле «Контакты»: номер телефона, адрес электронной почты.

Подсистема «Администрирование» должна обеспечить добавление и изменение данных администраторами и менеджерами. Изменяться и добавляться могут данные о сотрудниках, отделах, задачах и проектах, а также добавляться объявления и уведомления.

Подсистема «Отчеты» должна отвечать за формирование отчетов по заданным пользователем запросам за определенный период, результаты запросов должны экспортироваться в файлы с форматом *.xls* или *.doc*; отображение динамических и хронологических показателей в виде графиков для мониторинга и аналитики бизнес-процессов компании и их сохранения в удобной для пользователя форме.

4.3 Требования к видам обеспечения

4.3.1 Требования к ПО

Язык интерфейса приложения: русский, казахский, английский.

Разработанное программное обеспечение (далее ПО) должно быть:

- совместимо с устройствами на базе ОС Windows 10.

Серверная часть должна быть написана с использованием технологии платформы .NET на языке программирования C#.

Подключение к базе данных осуществляется по протоколу TCP/IP. Для реализации подсистемы хранения данных должна использоваться СУБД MS SQL Server.

4.3.2 Требования к техническому обеспечению

Требования к серверу:

- процессор с частотой не менее 2 GHz, количество процессоров/ядер не менее 2х (двух).
- память: 8Гб;
- Hdd: Raid массив от 1Тб;
- БД: SQL Server 2008 R2 Standard Edition или выше;

Требования к ПК, на котором будет установлено клиентское рабочее место:

- ОС Windows 10;
- процессор не менее 2 GHz;
- жесткий диск объем не менее 50 Gb;
- оперативная память не менее 2 Gb. Желательно, 4 Gb;
- ПК должен содержать не менее 1 (одного) Ethernet- интерфейса 100 Mbps.

5 Состав и содержание работ по созданию системы

Работы по созданию системы выполняются в семь этапов: анализ требований, проектирование, разработка дизайна проекта, разработка технического проекта, тестирование, внедрение и сопровождение.

6 Источник разработки

Настоящее Техническое Задание разработано на основе следующих документов и информационных материалов:

- договор от 01.02.2021 между директором ТОО «S&G IT Group» в лице Шутова И.Ю. и Ниязовой К.К.

ПРИЛОЖЕНИЕ Б

Листинг программы

Класс Authorization:

```
using System;
using System.Data;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Data.SqlClient;
using System.Configuration;
using NLog;
namespace ITGROUP
{
        public partial class Authorization : Form
    {
        public Authorization()
        {
            InitializeComponent();
        }
        public static string Ulogin = "";
        Logger logger = LogManager.GetCurrentClassLogger();
        #region Drag&Drop
        [DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]
        private extern static void ReleaseCapture();
        [DllImport("user32.DLL", EntryPoint = "SendMessage")]
        private extern static void SendMessage(System.IntPtr hWnd, int
wMsg, int wParam, int lParam);
        #endregion
        private void panel3_MouseDown(object sender, MouseEventArgs e)
        {
            ReleaseCapture();
            SendMessage(this.Handle, 0x112, 0xf012, 0);
        }
        private void buttonLogin_Click(object sender, EventArgs e)
        {           try           {
                using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Connecti
onString))   {
                    connection.Open();
                    string loginUser = textBoxUsername.Text;
                    string passUser = textBoxPassword.Text;
```

```
                    Ulogin = textBoxUsername.Text;
                    DataTable table = new DataTable();
                    SqlDataAdapter adapter = new SqlDataAdapter();
                    SqlCommand command = new SqlCommand("select * from
Employee where Ulogin = @ul and UPassword = @up", connection);
                    command.Parameters.Add("@ul",
SqlDbType.VarChar).Value = loginUser;
                    command.Parameters.Add("@up",
SqlDbType.VarChar).Value = passUser;
                    adapter.SelectCommand = command;
                    adapter.Fill(table);
                    if(table.Rows.Count > 0)
                    {
                        Form form = new MainForm();
                        form.Show();
                        this.Hide();
                        logger.Info($"Вход в систему - {Ulogin}");
                    }
                    else
                    {
                        MessageBox.Show("Попробуйте еще раз", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                        logger.Warn($"Неудачная попытка входа в систему
{Ulogin}");
                        SqlCommand c = new SqlCommand( );
                        c.Connection = connection;
                        c.CommandType = CommandType.Text;
                        c.CommandText = $"insert into Log values ('Неудачная
попытка входа в систему {Ulogin}');";
                        //c.Parameters.Add("@mess", SqlDbType.VarChar).Value
= Ulogin;
                        c.ExecuteNonQuery();
                    }
                }
            }
            catch(Exception ex)        {
                MessageBox.Show(ex.Message);
                logger.Error(ex);
            }       }       private void button2_Click(object sender, EventArgs )
        {           Application.Exit();       }
```

Класс MainForm:

```
using System;
using System.Windows.Forms;
using System.Runtime.InteropServices;
namespace ITGROUP{
        public partial class MainForm : Form{
        public MainForm(){
           InitializeComponent();
           this.Text = string.Empty;
           this.ControlBox = false;
           this.DoubleBuffered = true;
   this.MaximizedBounds = Screen.FromHandle(this.Handle).WorkingArea;}
         private Form activeForm = null;
        private void openChildForm(Form childForm)
      {
         if (activeForm != null)
            activeForm.Close();
         activeForm = childForm;
         childForm.TopLevel = false;
         childForm.FormBorderStyle = FormBorderStyle.None;
         childForm.Dock = DockStyle.Fill;
         panelChildForm.Controls.Add(childForm);
         panelChildForm.Tag = childForm;
         childForm.BringToFront();
         childForm.Show();
      }
      private void buttonHome_Click(object sender, EventArgs e){
         openChildForm(new ChildForms.HomePage());
         label3.Text = buttonHome.Text;}
private void buttonProjects_Click(object sender, EventArgs e){
         openChildForm(new ChildForms.Projects());
         label3.Text = buttonProjects.Text;}
      private void buttonClients_Click(object sender, EventArgs e){
         label3.Text = buttonClients.Text;
         openChildForm(new ChildForms.Clients());}
      private void buttonTasks_Click(object sender, EventArgs e){
         label3.Text = buttonTasks.Text;
         openChildForm(new ChildForms.Tasks());}
      private void buttonReports_Click(object sender, EventArgs e) {
         label3.Text=buttonReports.Text;
         openChildForm(new ChildForms.Reports()); }
```

```
        private void MainForm_Load(object sender, EventArgs e){
          ulogin.Text = Authorization.Ulogin;}

       private void buttonExitUser_Click(object sender, EventArgs e){
          this.Close();
          Authorization authorization = new Authorization();
          authorization.Show();}    } }
```

Класс HomePage:

```
using System;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace ITGROUP.ChildForms
{public partial class HomePage : Form
   {public HomePage()
      {InitializeComponent(); }
       public void getName()
      {try
         { using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))  {
              SqlCommand command = new SqlCommand("SELECT
CONCAT(Emp_surname, ' ', Emp_name, ' ', Emp_patronymic) from
Employee where Ulogin = @ulogin", connection);
              command.CommandType = CommandType.Text;
              command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
              SqlDataAdapter da = new SqlDataAdapter(command);
              DataTable t = new DataTable();
              da.Fill(t);
              textBox_Name.Text = "";
              textBox_Name.Text = t.Rows[0][0].ToString();}
         }  catch (Exception ex){  MessageBox.Show(ex.Message); }
      }
      public void getContacts()
      { try   {
```

```
            using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))  {
                SqlCommand command = new SqlCommand();
                command.Connection = connection;
                command.CommandText = "select CONCAT(Emp_surname, ' ',
Emp_name, ' ', Emp_patronymic) Имя, Phone_number 'Номер телефона'" +
                                      ", p.Position_name Должность from Employee e
inner join Position p on e.Position_ID=p.ID_Position"+
                                      " where Ulogin!=@ulogin";
                command.CommandType = CommandType.Text;
                command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
                DataTable dataTable = new DataTable();
                SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
                dataAdapter.Fill(dataTable);
                dataGridView_Contacts.DataSource = dataTable;        } }
          catch(Exception ex)        {   MessageBox.Show(ex.Message);        }
      }
      public void getPosition()      {
          try        {
              using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))
              {             SqlCommand command = new SqlCommand("SELECT
p.Position_name from Employee e inner join Position p on
e.Position_ID=p.ID_Position where Ulogin = @ulogin", connection);
                command.CommandType = CommandType.Text;
                command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
                SqlDataAdapter da = new SqlDataAdapter(command);
                DataTable t = new DataTable();
                da.Fill(t);
                textBox_Position.Text = "";
                label5.Text = t.Rows[0][0].ToString();              }
          }  catch (Exception ex){  MessageBox.Show(ex.Message); }
      }
      public void getEducation()
      {  try      {
```

```
        using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))  {
                SqlCommand command = new SqlCommand("SELECT
ed.Edu_name from Employee e inner join Education ed on
e.Education_ID=ed.ID_Education where Ulogin = @ulogin", connection);
                command.CommandType = CommandType.Text;
                command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
                SqlDataAdapter da = new SqlDataAdapter(command);
                DataTable t = new DataTable();
                da.Fill(t);
                textBox_Education.Text = "";
                textBox_Education.Text = t.Rows[0][0].ToString();     }
          }    catch (Exception ex)    {  MessageBox.Show(ex.Message);        }
      }

      public void getPhone()
      {    try    {
            using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))       {
                SqlCommand command = new SqlCommand("SELECT
Phone_number from  Employee where Ulogin = @ulogin", connection);
                command.CommandType = CommandType.Text;
                command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
                SqlDataAdapter da = new SqlDataAdapter(command);
                DataTable t = new DataTable();
                da.Fill(t);
                textBox_Position.Text = "";
                textBox_Position.Text = t.Rows[0][0].ToString();       }
          }        catch (Exception ex) {   MessageBox.Show(ex.Message);    }
      }

      private void HomePage_Load(object sender, EventArgs e)
      {  getName();
         getPosition();
         getEducation();
         getPhone();
         getContacts();     }  }}
```

Класс Clients:

```
using System;
using System.Data;
using NLog;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;
using System.Text.RegularExpressions;
namespace ITGROUP.ChildForms
{
   public partial class Clients : Form
   {       public Clients()
      {           InitializeComponent();       }
      Regex regex = new Regex(@"\d{3}\-\d{3}-\d{4}");
      Logger logger = LogManager.GetCurrentClassLogger();
      public void displayClientData()
      {           try           {
            using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))
            {                 SqlCommand command = new SqlCommand();
               command.Connection = connection;
               command.CommandText = "select ID_Client as 'Номер клиента'
, Client_name as 'Наименование' , Clinet_address as 'Адрес'" +
                                 ", ct.Client_type_name as 'Тип клиента',
CONCAT(CP_surname, ' ', CP_name, ' ', CP_patronymic) as 'Представитель'"
+
                                 ", CP_phone_number as 'Номер телефона' ,
CP_email as 'Электронная почта' from Client "+
                                 "left join Client_type ct on Client_type_ID =
ct.ID_Client_type";
               DataTable dataTable = new DataTable();
               SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
               dataAdapter.Fill(dataTable);
               dataGridView_Clients.DataSource = dataTable;
               dataGridView_Clients.Visible = true;
               dataGridView_Clients.Columns[3].Width = 100;
               dataGridView_Clients.Columns[6].Width = 150;           }        }
         catch(Exception ex)  {
            MessageBox.Show(ex.Message,                    "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);       }  }
```

```
//Загрузка страницы
private void Clients_Load(object sender, EventArgs e)
{          displayClientData();       }
public  void closePanel(object sender, EventArgs e)
{
   panel_AddClients.Visible = false;
   dataGridView_Clients.Visible = true;
   this.button_AddClient.Text = "Добавить клиента";
   displayClientData();
}
private void button_AddClient_Click(object sender, EventArgs e)
{
   fillComboboxClientType();
   dataGridView_Clients.Visible = false;
   panel_AddClients.Visible = true;
   this.button_AddClient.Text = "Закрыть";
   this.button_AddClient.Click += new EventHandler(this.closePanel);
}
//Заполнение комбобокса
public void fillComboboxClientType()
{   try       {
        using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))       {
            connection.Open();
            SqlCommand command = new SqlCommand("select
Client_type_name from Client_type", connection);
            SqlDataReader dataReader = command.ExecuteReader();
            while (dataReader.Read())
            {string Client_type_name =
dataReader["Client_type_name"].ToString();
              clientType.Items.Add(Client_type_name);          }       }
      }  catch(Exception ex)     {
        MessageBox.Show(ex.Message, "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);          }       }
//Добавить клиента
private void button_Add_Click(object sender, EventArgs e)
{       try      {
        using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))      {
            connection.Open();
```

```
            int a = clientType.SelectedIndex + 1;
            SqlCommand command = new SqlCommand("addClient",
connection);
            command.CommandType = CommandType.StoredProcedure;
            if ((string.IsNullOrWhiteSpace(bin.Text) == false) &&
(string.IsNullOrWhiteSpace(company_name.Text) == false) &&
        (string.IsNullOrWhiteSpace(address.Text) == false) &&
(string.IsNullOrWhiteSpace(surname.Text) == false) &&
(string.IsNullOrWhiteSpace(name.Text) == false))
            { if (regex.IsMatch(phone.Text))
               { try {
                    command.Parameters.AddWithValue("@bin",
SqlDbType.BigInt).Value = bin.Text;
                   command.Parameters.AddWithValue("@company_name",
SqlDbType.VarChar).Value = company_name.Text;
                    command.Parameters.AddWithValue("@address",
SqlDbType.VarChar).Value = address.Text;
                    command.Parameters.AddWithValue("@clientType",
SqlDbType.VarChar).Value = a;
                    command.Parameters.AddWithValue("@surname",
SqlDbType.VarChar).Value = surname.Text;
                    command.Parameters.AddWithValue("@name",
SqlDbType.VarChar).Value = name.Text;
                    command.Parameters.AddWithValue("@patronymic",
SqlDbType.Int).Value = patronymic.Text;
                    command.Parameters.AddWithValue("@phone",
SqlDbType.Int).Value = phone.Text;
                    command.Parameters.AddWithValue("@email",
SqlDbType.VarChar).Value = email.Text;
                    command.ExecuteNonQuery();
                    MessageBox.Show("Добавлено", "Информация",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                } catch (Exception ex)
                { MessageBox.Show(ex.Message);
                  logger.Error(ex); }}
              else { MessageBox.Show("Некорректный формат номера
телефона!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                } } } }
        catch(Exception ex) {
          MessageBox.Show(ex.Message, "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
          logger.Error(ex); } }
```

```
        //Нажатие цифр на поле с БИН
      private void bin_KeyPress(object sender, KeyPressEventArgs e)
      {
         if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
         {              e.Handled = true;          }        }
   }}
```

Класс Projects:

```
using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;
using System.Collections;
namespace ITGROUP.ChildForms
{    public partial class Projects : Form
   {   ArrayList ttype = new ArrayList();
      ArrayList amount = new ArrayList();
      public Projects()
      {       InitializeComponent();    }
      private Panel activepanel = null;
      private void ActivatePanel(Panel newPan)
      {  if (activepanel != null)
         {          activepanel.Visible = false;        }
         activepanel = newPan;
         activepanel.Visible = true;      }
      public                                                          void
ClearChart(System.Windows.Forms.DataVisualization.Charting.Chart chart)
      {    chart.Series[0].Points.Clear();
         foreach (var series in chart.Series)
         {              series.Points.Clear();          }       }
      public void clearArraylist(ArrayList arrayList)
      {          arrayList.Clear();       }
      public void getProjectInfo(int a)
      {          try          {              using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))          {
               connection.Open();
               SqlCommand command = new SqlCommand("tasksCharts",
connection);
               command.CommandType = CommandType.StoredProcedure;
```

```
            command.Parameters.Add("@id_project", SqlDbType.Int).Value
= a;
            SqlDataReader dataReader = command.ExecuteReader();
            while (dataReader.Read())
            {   ttype.Add(dataReader.GetString(0));
               amount.Add(dataReader.GetInt32(1));                }
            chart2.Series[0].Points.DataBindXY(ttype, amount);
            dataReader.Close();             }
        }                       catch(Exception   ex)                   {
MessageBox.Show(ex.Message);          }       }

    public void getProjectInfoByUser(int a)
    {   ArrayList ttype1 = new ArrayList();
        ArrayList amount1 = new ArrayList();
        try             {using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))           {
            connection.Open();
            SqlCommand command = new
SqlCommand("taskChartsByUser", connection);
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.Add("@id_project", SqlDbType.Int).Value
= a;                                    command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
            SqlDataReader dataReader = command.ExecuteReader();
            while (dataReader.Read())
            {  ttype1.Add(dataReader.GetString(0));
               amount1.Add(dataReader.GetInt32(1));                }
            chart1.Series[0].Points.DataBindXY(ttype1, amount1);
            dataReader.Close();             }
        } catch (Exception ex)    {  MessageBox.Show(ex.Message);   }    }
    public void getProjectName (int a)
    {         try         {
          using         (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))           {
            SqlCommand command = new SqlCommand("SELECT
Project_name from Project where ID_Project = @id", connection);
            command.CommandType = CommandType.Text;
            command.Parameters.Add("@id", SqlDbType.Int).Value = a;
            SqlDataAdapter da = new SqlDataAdapter(command);
            DataTable t = new DataTable();
```

```
                da.Fill(t);
                label8.Text = "";
                label8.Text = t.Rows[0][0].ToString();          }       }
          catch (Exception ex)      MessageBox.Show(ex.Message);    }    }
      public void getProjectDesc(int a){
          try       {           using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))       {
                SqlCommand command = new SqlCommand("SELECT
Project_desc from Project where ID_Project = @id", connection);
           command.CommandType = CommandType.Text;
                command.Parameters.Add("@id", SqlDbType.Int).Value = a;
                SqlDataAdapter da = new SqlDataAdapter(command);
                DataTable t1 = new DataTable();
                da.Fill(t1);
                textBox1.Text = "";
                textBox1.Text = t1.Rows[0][0].ToString();       }
          } catch(Exception ex){  MessageBox.Show(ex.Message);    }    }
      public void getStatus(int a, string s, CircularButton circular)
      {       try       {           using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))       {           SqlCommand command = new
SqlCommand("statusLabel", connection);
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.Add("@status",       SqlDbType.VarChar,
20).Value       =       s;       command.Parameters.Add("@id_project",
SqlDbType.Int).Value = a;
                SqlDataAdapter da = new SqlDataAdapter(command);
                DataTable t = new DataTable();
                da.Fill(t);
                circular.Text = t.Rows[0][1].ToString();        }       }
          catch(Exception ex)   {   MessageBox.Show(ex.Message);  }  }
      public void getPriority(int a, string s, CircularButton circular)
      {       try       {           using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))       {           SqlCommand command = new
SqlCommand("priorityLabel", connection);
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.Add("@priority",     SqlDbType.VarChar,
20).Value = s;
```

```
                command.Parameters.Add("@id_project", SqlDbType.Int).Value
= a;            SqlDataAdapter da = new SqlDataAdapter(command);
                DataTable t = new DataTable();
                da.Fill(t);
                circular.Text = t.Rows[0][1].ToString();      }    }
          catch (Exception ex)    { MessageBox.Show(ex.Message);     }  }
       private void button_P1_Click(object sender, EventArgs e)
       {   ActivatePanel(panel6);
          clearArraylist(ttype);
          clearArraylist(amount);
          ClearChart(chart2);
           getProjectInfo(1);
          getProjectInfoByUser(1);
          getProjectName(1);
          getProjectDesc(1);
          getStatus(1, "Принято", circularButton1);
          getStatus(1, "В процессе", circularButton2);
          getStatus(1, "Выполнено", circularButton3);
          getPriority(1, "Высокий", circularButton4);
          getPriority(1, "Средний", circularButton5);
          getPriority(1, "Низкий", circularButton6);
       }     }}
```

Класс Tasks:

```
using System;
using System.Data;
using System.Drawing;
using Excel = Microsoft.Office.Interop.Excel;
using System.Data.SqlClient;
using System.Configuration;
using System.Windows.Forms;
using System.Collections;
using DGVPrinterHelper;
namespace ITGROUP.ChildForms
{    public partial class Tasks : Form
    {       public Tasks()
       {           InitializeComponent();       }
       ArrayList ttype1 = new ArrayList();
       ArrayList amount1 = new ArrayList();
       public void clearArraylist(ArrayList arrayList)
       {           arrayList.Clear();       }
```

```
public void ClearChart(System.Windows.Forms.DataVisualization.Charting.Chart chart)
{ chart.Series[0].Points.Clear();
  foreach (var series in chart.Series)
  { series.Points.Clear(); } }
public void getCurrentTasksAmount()
{ try { using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].ConnectionString)) {
        connection.Open();
        SqlCommand command = new SqlCommand();
        command.Connection = connection;
        command.CommandType = CommandType.Text;
        command.CommandText = "select count(Task_ID) from Employee_Task et " +
                              "inner join Employee e on et.Employee_ID = e.ID_Employee " +
                              "inner join Task t on et.Task_ID = t.ID_Task " +
                              "where et.Employee_ID = (select ID_Employee from Employee where Ulogin = @ulogin ) " +
                              "and t.Task_status_ID in (1,2);";
        command.Parameters.Add("@ulogin", SqlDbType.VarChar).Value = Authorization.Ulogin;

        SqlDataAdapter da = new SqlDataAdapter(command);
        DataTable t = new DataTable();
        da.Fill(t);
        label8.Text = "";
        label8.Text = t.Rows[0][0].ToString();
      } } catch (Exception ex) {
      MessageBox.Show(ex.Message); } } public void getDoneTasksAmount()
{ try
  { using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].ConnectionString))
    { connection.Open();
      SqlCommand command = new SqlCommand();
      command.CommandType = CommandType.Text;
      command.Connection = connection;
      command.CommandText = "select count(Task_ID) from Employee_Task et " +
                            "inner join Employee e on et.Employee_ID = e.ID_Employee " +
```

```
                    "inner join Task t on et.Task_ID = t.ID_Task " +
                    "where et.Employee_ID = (select ID_Employee
from Employee where Ulogin = @ulogin ) " +
                    "and t.Task_status_ID=3;";
            command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
            SqlDataAdapter da = new SqlDataAdapter(command);
            DataTable t = new DataTable();
            da.Fill(t);
            label1.Text = "";
            label1.Text = t.Rows[0][0].ToString();
        }       }
      catch (Exception ex)
      {         MessageBox.Show(ex.Message);       }     }
    public void getProjectsAmount()
    {       try
      {                    using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.CommandType = CommandType.Text;
            command.Connection = connection;
            command.CommandText = "select count(Project_ID) from
Employee_Task et " +
                    "inner join Employee e on et.Employee_ID =
e.ID_Employee " +
                    "where et.Employee_ID = (select ID_Employee
from Employee where Ulogin = @ulogin ) ";
            command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
            SqlDataAdapter da = new SqlDataAdapter(command);
            DataTable t = new DataTable();
            da.Fill(t);
            label5.Text = "";
            label5.Text = t.Rows[0][0].ToString();
        }       }   catch (Exception ex)       {
        MessageBox.Show(ex.Message);       }     }
    public void getTasksAmountByUser()
```

```
{ try { using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString)) {
            connection.Open();
            SqlCommand command = new
SqlCommand("getTasksAmountByUser", connection);
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
            SqlDataReader dataReader = command.ExecuteReader();
            while (dataReader.Read())
            { ttype1.Add(dataReader.GetString(0));
              amount1.Add(dataReader.GetInt32(1)); }
            chart1.Series[0].Points.DataBindXY(ttype1, amount1);
            dataReader.Close(); } }
      catch (Exception ex)
      { MessageBox.Show(ex.Message); } }
    public void taskTable()
    { try {
        using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString)) {
            SqlCommand command = new SqlCommand("taskTable",
connection);
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
            DataTable dataTable = new DataTable();
            SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
            dataAdapter.Fill(dataTable);
            dataGridView_Tasks.DataSource = dataTable;
          }
          for (int i = 0; i < dataGridView_Tasks.Columns.Count; i++)
          {
            dataGridView_Tasks.Columns[i].Width = 60; }
      } catch (Exception ex) {
          MessageBox.Show(ex.Message); } }
    private void timer1_Tick(object sender, EventArgs e)
    { labelCurrentTime.Text = DateTime.Now.ToLongTimeString();
      labelDate.Text = DateTime.Now.ToShortDateString();
      labelDayofWeek.Text = DateTime.Today.ToString("dddd");
    }
```

```
private void Tasks_Load(object sender, EventArgs e)
      {  taskTable();
         clearArraylist(ttype1);
         clearArraylist(amount1);
         ClearChart(chart1);
         getCurrentTasksAmount();
         getDoneTasksAmount();
         getProjectsAmount();
         getTasksAmountByUser();
      }
      private void button_AddClient_Click(object sender, EventArgs e)
      {
         DGVPrinter printer = new DGVPrinter();
         printer.Title = "Отчет по задачам ";
         printer.SubTitle = $"Автор: {Authorization.Ulogin}.
{string.Format("Дата: {0}",
DateTime.Now.Date.ToString("MM/dd/yyyy"))}";
         printer.SubTitleFormatFlags = StringFormatFlags.LineLimit |
StringFormatFlags.NoClip;
         printer.PageNumbers = true;
         printer.PageNumberInHeader = false;
         printer.PorportionalColumns = true;
         printer.HeaderCellAlignment = StringAlignment.Near;
         printer.Footer = "ITGroup";
         printer.FooterSpacing = 15;
         printer.PrintDataGridView(dataGridView_Tasks);
      }
      private void button1_Click(object sender, EventArgs e)
      {
         try{
            if (dataGridView_Tasks.Rows.Count > 0)     {
               Excel.Application app = new Excel.Application();
               Excel.Workbook workbook =
app.Workbooks.Add(Type.Missing);
               Excel.Worksheet worksheet = null;
               worksheet = workbook.Sheets["Лист1"];
               worksheet = workbook.ActiveSheet;
               worksheet.Name = "Table";
               for (int i = 1; i < dataGridView_Tasks.Columns.Count; i++){
worksheet.Cells[1, i] = dataGridView_Tasks.Columns[i - 1].HeaderText;     }
```

```
            for (int i = 0; i < dataGridView_Tasks.Rows.Count; i++)
            {                                                    for (int j = 0; j <
dataGridView_Tasks.Columns.Count; j++)
              {                                  worksheet.Cells[i + 2, j + 1] =
dataGridView_Tasks.Rows[i].Cells[j].Value.ToString();
              }              }
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.FileName = $"report";
            saveFileDialog.DefaultExt = ".xlsx";
            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {                    workbook.SaveAs(saveFileDialog.FileName,
Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing,
Excel.XlSaveAsAccessMode.xlExclusive, Type.Missing, Type.Missing,
Type.Missing, Type.Missing, Type.Missing);              }
            app.Quit();          }          }
        catch(Exception ex)          {
          MessageBox.Show(ex.Message);          }       }    }}
```

Класс Reports:

```
using System;
using NLog;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Drawing;
using Excel = Microsoft.Office.Interop.Excel;
using System.Windows.Forms;
using DGVPrinterHelper;
using System.Collections;
using System.IO;
using System.Windows.Forms.DataVisualization.Charting;
namespace ITGROUP.ChildForms
{  public partial class Reports : Form
   {   public Reports()
       {          InitializeComponent();       }
       Logger logger = NLog.LogManager.GetCurrentClassLogger();
       private Panel activepanel = null;
       private void ActivatePanel(Panel newPan)
       {
 if (activepanel != null)
```

```
        {    activepanel.Visible = false;          }
          activepanel = newPan;
          activepanel.Visible = true;       }
      public void getUserName()
      {          try          {              using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))              {                  SqlCommand command = new
SqlCommand();
                  command.Connection = connection;
                  command.CommandType = CommandType.Text;
                  command.CommandText = "select CONCAT(Emp_surname,' ',
Emp_name, ' ', Emp_patronymic) from Employee where Ulogin = @ulogin ";
                  command.Parameters.AddWithValue("@ulogin",
Authorization.Ulogin);
                  SqlDataAdapter da = new SqlDataAdapter(command);
                  DataTable t = new DataTable();
                  da.Fill(t);
                  label_Emp.Text = "";
                  label_Emp.Text = t.Rows[0][0].ToString();              }          }
        catch(Exception e)          {          MessageBox.Show(e.Message);  } }
      public void displayProjectDataByUser(int a)       {          try
        {using          (SqlConnection          connection          =          new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))       {
                  connection.Open();
                  SqlCommand          command          =          new
SqlCommand("forUserReportByProject", connection);
                  command.CommandType = CommandType.StoredProcedure;
                  command.Parameters.Add("@id_project", SqlDbType.Int).Value
=  a;                                    command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
                  DataTable dataTable = new DataTable();
                  SqlDataAdapter dataAdapter = new SqlDataAdapter(command);
                  dataAdapter.Fill(dataTable);
                  dataGridView_Tasks.DataSource = dataTable;              }          }
        catch(Exception ex)       {     MessageBox.Show(ex.Message); }      }
      private void Reports_Load(object sender, EventArgs e)
      {       ClearChart(chart1);
        label_Date.Text = DateTime.Now.ToShortDateString();
        //getUserName();          }
       private void button_P1_Click(object sender, EventArgs e)
      {  displayProjectDataByUser(1);
```

```
            ActivatePanel(panel6);
            getDataForChart(1);       }
private void button_AddClient_Click(object sender, EventArgs e)
        {  DGVPrinter printer = new DGVPrinter();
            printer.Title = "Отчет по проекту ";
            printer.SubTitle = $"Автор: {label_Emp.Text}. {string.Format("Дата:
{0}", DateTime.Now.Date.ToString("MM/dd/yyyy"))}";
            printer.SubTitleFormatFlags = StringFormatFlags.LineLimit |
StringFormatFlags.NoClip;
            printer.PageNumbers = true;
            printer.PageNumberInHeader = false;
            printer.PorportionalColumns = true;
            printer.HeaderCellAlignment = StringAlignment.Near;
            printer.Footer = "ITGroup";
            printer.FooterSpacing = 15;
            printer.PrintDataGridView(dataGridView_Tasks);
            logger.Info("Печать таблицы");       }
           public void getDataForChart(int a)       {
           ArrayList ttype1 = new ArrayList();
           ArrayList amount1 = new ArrayList();
           try             {                 using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))             {                 connection.Open();
                SqlCommand                 command                 =                 new
SqlCommand("taskChartsByUser", connection);
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.Add("@id_project",  SqlDbType.Int).Value
=  a;                                        command.Parameters.Add("@ulogin",
SqlDbType.VarChar).Value = Authorization.Ulogin;
                SqlDataReader dataReader = command.ExecuteReader();
                while (dataReader.Read())
                {ttype1.Add(dataReader.GetString(0));
                  amount1.Add(dataReader.GetInt32(1));                 }
                chart1.Series[0].Points.DataBindXY(ttype1, amount1);
                dataReader.Close();             }         }
          catch (Exception ex)         {     MessageBox.Show(ex.Message);   }
       }       public void printExcel()
       {          try
{
 if (dataGridView_Tasks.Rows.Count > 0)
             {
Excel.Application app = new Excel.Application();
```

```
Excel.Workbook workbook = app.Workbooks.Add(Type.Missing);
                Excel.Worksheet worksheet = null;
                worksheet = workbook.Sheets["Лист1"];
                worksheet = workbook.ActiveSheet;
                worksheet.Name = "Table";
                for (int i = 1; i < dataGridView_Tasks.Columns.Count; i++)
                {                                                   worksheet.Cells[1,  i]  =
dataGridView_Tasks.Columns[i - 1].HeaderText;                }
                for (int i = 0; i < dataGridView_Tasks.Rows.Count; i++)
                {     for (int j = 0; j < dataGridView_Tasks.Columns.Count; j++)
                   {                                    worksheet.Cells[i  +  2,  j  +  1]  =
dataGridView_Tasks.Rows[i].Cells[j].Value.ToString();                                  }
}                SaveFileDialog saveFileDialog = new SaveFileDialog();
                saveFileDialog.FileName = $"report";
                saveFileDialog.DefaultExt = ".xlsx";
                if (saveFileDialog.ShowDialog() == DialogResult.OK)
                {                        workbook.SaveAs(saveFileDialog.FileName,
Type.Missing,  Type.Missing,  Type.Missing,  Type.Missing,  Type.Missing,
Excel.XlSaveAsAccessMode.xlExclusive,  Type.Missing,  Type.Missing,
Type.Missing, Type.Missing, Type.Missing);                }
                app.Quit();                }
          }          catch(Exception ex) {  MessageBox.Show(ex.Message); }      }
      public void printChart()
      {           try           {               SaveFileDialog saveFileDialog = new
SaveFileDialog();
            saveFileDialog.Filter          =          "JPEG          (*.jpeg)|*.jpeg|BMP
(*.bmp)|*.bmp|PNG (*.png)|*.png|All files (*.*)|*.*";
            saveFileDialog.RestoreDirectory = true;
            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {                                                                                  if
(Path.GetExtension(saveFileDialog.FileName).ToLower() == ".png")
                this.chart1.SaveImage(saveFileDialog.FileName,
ChartImageFormat.Png);
                else if (Path.GetExtension(saveFileDialog.FileName).ToLower()
== ".bmp")
                this.chart1.SaveImage(saveFileDialog.FileName,
ChartImageFormat.Bmp);
                else if (Path.GetExtension(saveFileDialog.FileName).ToLower()
== ".jpeg")
                this.chart1.SaveImage(saveFileDialog.FileName,
ChartImageFormat.Jpeg);
```

```
saveFileDialog.Dispose();            }        }        catch (Exception ex)
        {            MessageBox.Show(ex.Message);        }      }
      public                                                              void
ClearChart(System.Windows.Forms.DataVisualization.Charting.Chart chart)
      {       chart.Series[0].Points.Clear();
        //chart.Series[0].Points.RemoveAt(0);
        foreach (var series in chart.Series)
        {            series.Points.Clear();        }      }
      private void button1_Click(object sender, EventArgs e)
      {       printExcel();       logger.Info("Печать таблицы в Excel");
      }
      private void button2_Click(object sender, EventArgs e)
      {       printChart();       logger.Info("Печать диаграммы");      }
   }
}
```

Класс AdminPanel:

```
using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;

namespace ITGROUP.ChildForms
{
   public partial class AdminPanel : Form
   {
      SqlDataAdapter dataAdapter;
      DataTable dataTable;
      public AdminPanel()
      {       InitializeComponent();      }

      public void getData()      {
        using          (SqlConnection          conn          =          new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))
        {            dataAdapter = new SqlDataAdapter("select ID_Employee,
Emp_surname, Emp_name, Emp_patronymic " +
            ", p.Position_name, ed.Edu_name, Phone_number, u.Role_name
from Employee e " +
            " inner join Position p on e.Position_ID = p.ID_Position " +
```

```
" inner join Education ed on e.Education_ID = ed.ID_Education "
                " inner join URole u on e.URole = u.ID_Role"
                , conn);
            dataTable = new DataTable();
            dataAdapter.Fill(dataTable);
            dataGridView_Users.DataSource = dataTable;          }       }

        private void AdminPanel_Load(object sender, EventArgs e)
        {
            getData();
            fillComboboxRole();
            fillComboboxEducation();
            fillComboboxPosition();
        }

        private void button_Update_Click(object sender, EventArgs e)
        {
            try             {                   using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))
                {                   connection.Open();
                    SqlCommand      cmd     =     new     SqlCommand("updateUser",
connection);
                    cmd.CommandType = CommandType.StoredProcedure;
                    cmd.Parameters.AddWithValue("@surname", surname.Text);
                    cmd.Parameters.AddWithValue("@name", textBox1.Text);
                    cmd.Parameters.AddWithValue("@patr", textBox2.Text);
                    cmd.Parameters.AddWithValue("@ph", textBox3.Text);
                    cmd.Parameters.AddWithValue("@pos",  position.SelectedIndex
+ 1);
                    cmd.Parameters.AddWithValue("@edu",
education.SelectedIndex + 1);
                    cmd.Parameters.AddWithValue("@role",  role.SelectedIndex  +
1);                 cmd.Parameters.AddWithValue(@"id", textBox_ID.Text);
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Обновлено",                "Информация",
MessageBoxButtons.OK, MessageBoxIcon.Information);              }           }
            catch (Exception ex)
            {
           MessageBox.Show(ex.Message);          }
            getData();
}
```

```
private void button_Add_Click(object sender, EventArgs e)
{ try
  { using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString)) { connection.Open();
SqlCommand cmd = new SqlCommand("addUser", connection);
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.AddWithValue("@surname", surname.Text);
cmd.Parameters.AddWithValue("@name", textBox1.Text);
cmd.Parameters.AddWithValue("@patr", textBox2.Text);
cmd.Parameters.AddWithValue("@ph", textBox3.Text);
cmd.Parameters.AddWithValue("@pos",
position.SelectedIndex+1);
cmd.Parameters.AddWithValue("@edu",
education.SelectedIndex+1);
cmd.Parameters.AddWithValue("@role", role.SelectedIndex +
1);
cmd.ExecuteNonQuery();
MessageBox.Show("Добавлено", "Информация",
MessageBoxButtons.OK, MessageBoxIcon.Information); } }
  catch (Exception ex)
  { MessageBox.Show(ex.Message); }
  getData(); }

public void fillComboboxRole()
{ try {
  using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))
  { connection.Open();
SqlCommand command = new SqlCommand("select Role_name
from URole", connection);
SqlDataReader dataReader = command.ExecuteReader();
while (dataReader.Read())
{ string name = dataReader["Role_name"].ToString();
  role.Items.Add(name); } }
} catch (Exception ex) {
  MessageBox.Show(ex.Message, "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error); } }
public void fillComboboxEducation()
{ try
{
```

```
using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].ConnectionString))
            {
                connection.Open();
                SqlCommand command = new SqlCommand("select Edu_name from Education", connection);
                SqlDataReader dataReader = command.ExecuteReader();
                while (dataReader.Read())
                {
                    string name = dataReader["Edu_name"].ToString();
                    education.Items.Add(name);
                }            }
        } catch (Exception ex) { MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error); } }

    public void fillComboboxPosition()
    { try {
            using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].ConnectionString)) {
                connection.Open();
                SqlCommand command = new SqlCommand("select Position_name from Position", connection);
                SqlDataReader dataReader = command.ExecuteReader();
                while (dataReader.Read())
                {
                    string name = dataReader["Position_name"].ToString();
                    position.Items.Add(name);
                }            }        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error); } }
    private void dataGridView_Users_SelectionChanged(object sender, EventArgs e) {

        foreach (DataGridViewRow row in dataGridView_Users.SelectedRows) {
            surname.Text = row.Cells[1].Value.ToString();
            textBox1.Text = row.Cells[2].Value.ToString();
            textBox2.Text = row.Cells[3].Value.ToString();
```

```
            textBox3.Text = row.Cells[4].Value.ToString();
            position.Text = row.Cells[5].Value.ToString();
            education.Text = row.Cells[6].Value.ToString();
            role.Text = row.Cells[7].Value.ToString();
            textBox_ID.Text = row.Cells[0].Value.ToString();          }       }

        private void button_Delete_Click(object sender, EventArgs e)
        {           try           {               using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ITGroup"].Conne
ctionString))           {               connection.Open();
                SqlCommand cmd = new SqlCommand("delete from Employee
where ID_Employee = @id", connection);
                cmd.Parameters.AddWithValue("@id", textBox_ID.Text);
                cmd.ExecuteNonQuery();              }           }
          catch (Exception ex){ MessageBox.Show(ex.Message); }}  }}
```

ПРИЛОЖЕНИЕ В

Акт внедрения

Утверждаю
Директор
ТОО «S&G IT Group»
Шутов И.Ю.
«27» мая 2021 г.

АКТ ВНЕДРЕНИЯ

Настоящий акт подтверждает, что результат выпускной работы студента Алматинского университета энергетики и связи группы ИС-17-2 Ниязовой К. на тему «Разработка информационной системы «IT Group» был передан в эксплуатацию в ТОО «S&G IT Group» для использования в качестве программного комплекса. Использование результата выпускной работы Ниязовой К. обеспечивает автоматизацию контроля бизнес-показателей, сбор, обработку и актуализацию статистики, составление отчетности, повышение производительности сотрудников.

Директор
ТОО «S&G IT Group» Шутов И.Ю.