МИНИСТЕРСТВО ЭНЕРГЕТИКИ, ПРОМЫШЛЕННОСТИ И СВЯЗИ СТАВРОПОЛЬСКОГО КРАЯ

Государственное бюджетное профессиональное образовательное учреждение «Ставропольский колледж связи

имени Героя Советского Союза В.А. Петрова»

Цикловая комиссия вычислительной техники

**КУРСОВАЯ РАБОТА**

**по модулю** **ПМ.02** Осуществление интеграции программных модулей

на тему:

Автоматизация работы менеджера строительной организацией

Студента Кравченко Кирилла Александровича

Специальность 09.02.07 Информационные

системы и программирование

Курс 4 группа ИП213

Руководитель Еременко М. П.

Оценка выполнения курсовой работы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ / Еременко М.П.

Ставрополь, 2025 г.

Содержание

[Введение 3](#_Toc191333242)

[1 Теоретические основы разработки программного обеспечения 4](#_Toc191333243)

[1.1 Этапы жизненного цикла программных систем 4](#_Toc191333244)

[1.2 Функциональное моделирование программного продукта 5](#_Toc191333245)

[1.3 Объектно-ориентированный подход к разработке проекта 5](#_Toc191333246)

[2 Разработка программного продукта 6](#_Toc191333247)

[2.1 Техническое задание на разработку 6](#_Toc191333248)

[2.2 Функциональные диаграммы информационной системы 6](#_Toc191333249)

[2.3 Объектно-ориентированное проектирование системы 9](#_Toc191333250)

[2.4 Разработка интерфейса программного продукта 11](#_Toc191333251)

[2.5 Техническая документация программной системы 14](#_Toc191333252)

[Заключение 15](#_Toc191333253)

[Приложение 1. Техническое задание на разработку программного продукта 16](#_Toc191333254)

[Приложение 2. Исходный программный код информационной системы 18](#_Toc191333255)

[Приложение 3. Руководство пользователя 28](#_Toc191333256)

[Список использованных источников 30](#_Toc191333257)

Введение

Современные строительные компании сталкиваются с необходимостью автоматизации управленческих процессов для повышения эффективности работы менеджеров. В условиях увеличивающегося количества проектов, подрядчиков и клиентов, ручное ведение данных становится неэффективным и приводит к высоким затратам времени, а также возможным ошибкам при обработке информации.

Целью данного проекта является разработка программного обеспечения для автоматизации работы менеджера строительной компании. Приложение должно обеспечивать удобное управление проектами, клиентами и подрядчиками, а также предоставлять функционал для формирования отчетности по текущему состоянию работ.

Разрабатываемая система должна удовлетворять следующим требованиям:

* Обеспечение аутентификации пользователей для защиты данных;
* Хранение и управление информацией о строительных проектах (название, статус, сроки выполнения и т. д.);
* Ведение базы данных клиентов и подрядчиков;
* Формирование отчетов о текущих проектах;
* Удобный графический интерфейс с использованием Windows Forms;
* Работа с базой данных (например, SQLite или SQL Server) для хранения информации.

Для реализации программного обеспечения используются следующие технологии и методы:

* Язык программирования **C#** и технология **Windows Forms** для создания пользовательского интерфейса;
* **SQLite** или **SQL Server** в качестве базы данных для хранения информации;
* Объектно-ориентированный подход к проектированию архитектуры приложения;

Таким образом, реализация данного ПО позволит значительно упростить работу менеджеров строительных компаний, повысив точность и эффективность управления проектами и клиентской базой.

# 1 Теоретические основы разработки программного обеспечения

## 1.1 Этапы жизненного цикла программных систем

Разработка программного обеспечения проходит через несколько ключевых этапов, составляющих его жизненный цикл. Эти этапы включают:

1. **Анализ требований** – определение потребностей пользователей и формулирование технического задания. На этом этапе проводится исследование области, в которой будет использоваться система, и разрабатываются основные функциональные требования.
2. **Проектирование** – создание архитектуры системы, определение ее структуры, интерфейсов и базы данных. Используются диаграммы, схемы и модели для визуализации будущего ПО.
3. **Реализация (кодирование)** – непосредственная разработка программного кода на основе проектной документации. Используются языки программирования и инструменты разработки.
4. **Тестирование** – проверка работоспособности системы, выявление и исправление ошибок. Включает модульное, интеграционное и системное тестирование.
5. **Внедрение и эксплуатация** – установка ПО у заказчика, обучение пользователей и запуск системы в реальную эксплуатацию. На этом этапе также могут выявляться недочеты, требующие исправления.
6. **Сопровождение и обновление** – поддержка работоспособности системы, устранение возникающих ошибок и добавление нового функционала в соответствии с изменяющимися требованиями.

## 1.2 Функциональное моделирование программного продукта

Функциональное моделирование является важным этапом при разработке программных систем. Оно позволяет наглядно представить работу системы, ее функциональные возможности и взаимодействие с пользователями. Одним из наиболее распространенных методов является использование **диаграмм потоков данных (DFD), IDEF0** и **моделей бизнес-процессов (BPMN)**. Эти модели помогают разработчикам понять, какие операции выполняются в системе, какие данные обрабатываются и как происходит их передача между компонентами.

Функциональное моделирование также способствует выявлению потенциальных узких мест системы и позволяет заранее предусмотреть возможные оптимизации. Благодаря этому процессу обеспечивается соответствие разрабатываемого ПО требованиям заказчика и конечных пользователей.

## 1.3 Объектно-ориентированный подход к разработке проекта

Объектно-ориентированный подход (ООП) является одним из наиболее популярных методов разработки программного обеспечения. Он основан на концепции объектов, которые инкапсулируют данные и методы для работы с ними. Основные принципы ООП включают:

* **Инкапсуляция** – скрытие внутренней реализации объектов и предоставление доступа только через определенные интерфейсы.
* **Наследование** – возможность создания новых классов на основе уже существующих, что уменьшает дублирование кода.
* **Полиморфизм** – способность объектов обрабатывать данные разными способами в зависимости от их типа.

Применение ООП при разработке программного обеспечения повышает его гибкость, модульность и удобство поддержки. В данном проекте объектно-ориентированное программирование используется для создания удобной структуры классов, представления сущностей (например, **Проект**, **Клиент**, **Подрядчик**) и упрощения взаимодействия между компонентами системы.

Для представления будут использованы **Диаграмма вариантов использования, диаграмма последовательностей** и **диаграмма коммуникации.**

# 2 Разработка программного продукта

## 2.1 Техническое задание на разработку

Техническое задание (ТЗ) является основным документом, определяющим требования к разрабатываемой системе. Оно включает в себя описание функциональности, интерфейса, ограничений и других аспектов системы. Наличие качественного ТЗ позволяет минимизировать риски при разработке и обеспечить соответствие системы требованиям заказчика. Полное техническое задание приведено в **Приложении 1**.

## 2.2 Функциональные диаграммы информационной системы

Ниже представлены функционально-ориентированные диаграммы для проекта:

* IDEF0 диаграмма: контекстная диаграмма информационной системы A0 и A1 (рисунок 1 и рисунок 2 соответсвенно).

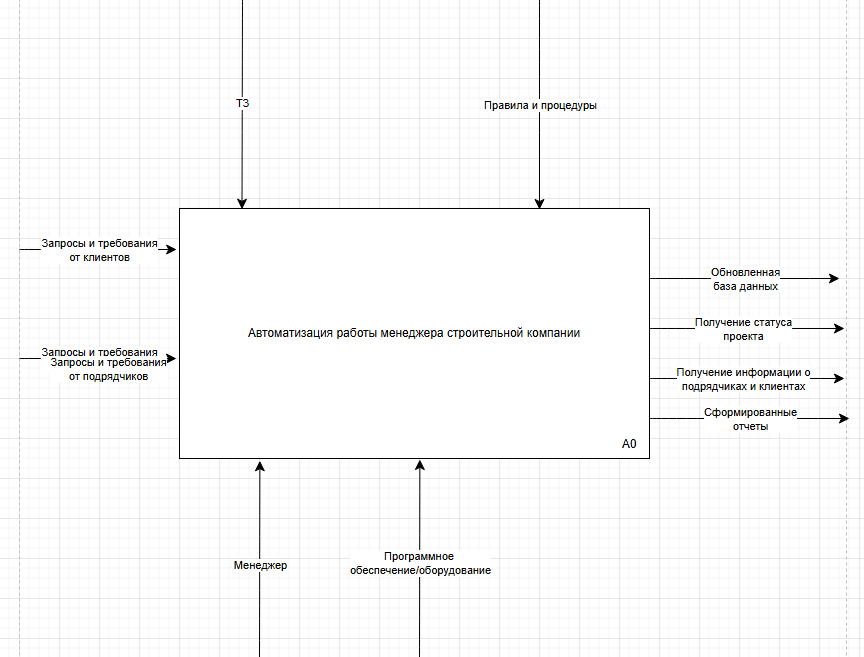


Рис. 1. Контекстная диаграмма информационной системы А0

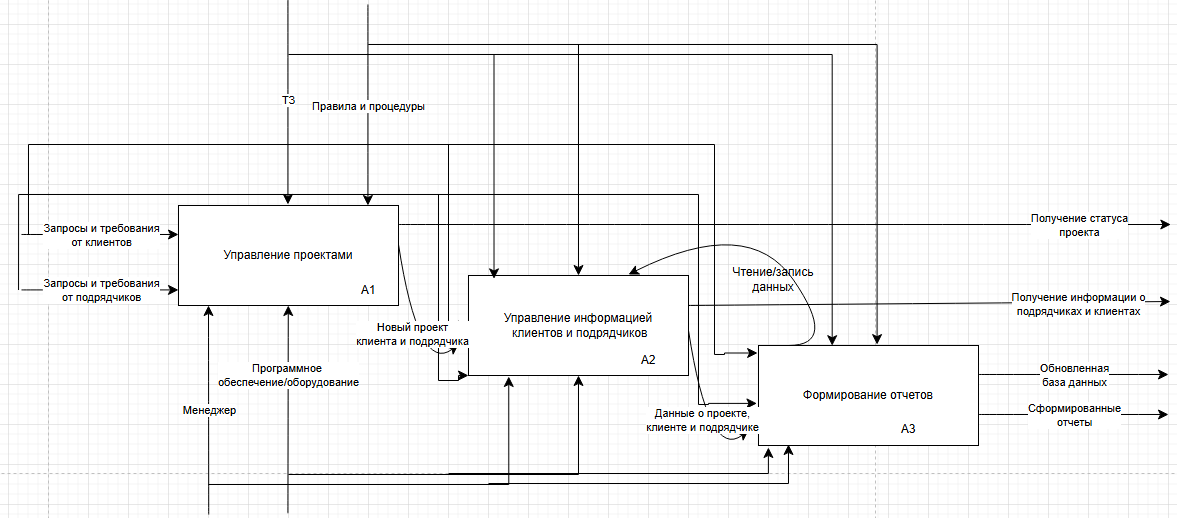


Рис. 2. Контекстная диаграмма информационной системы А1

* Диаграмма потоков данных DFD (рисунок 3).

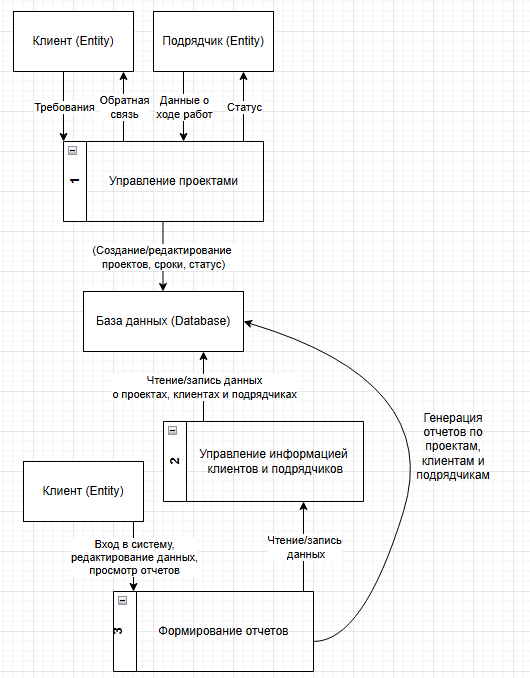


Рис. 3. Диаграмма потоков данных DFD

* Модель бизнес-процессов BPMN (рисунок 4)

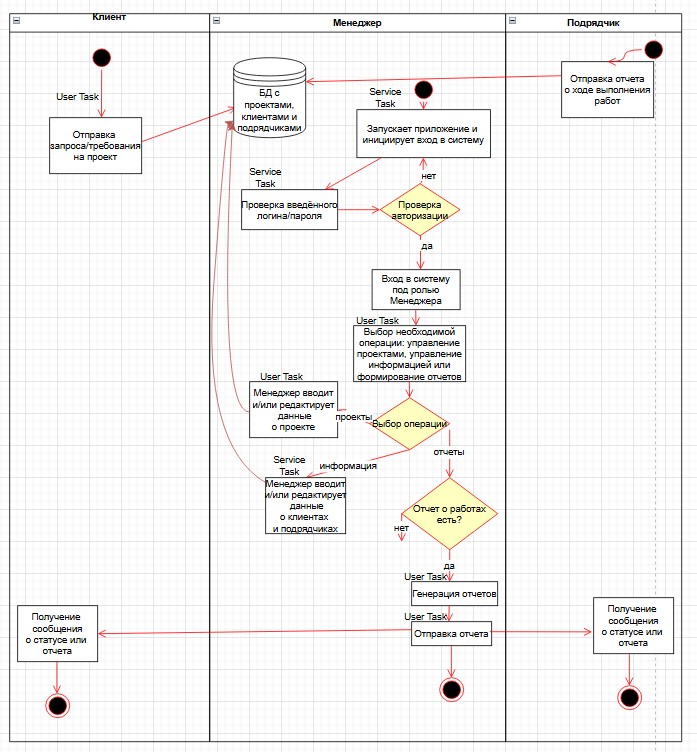


Рис. 4. Модель бизнес-процессов BPMN

## 2.3 Объектно-ориентированное проектирование системы

Ниже представлены объектно-ориентированные диаграммы для проекта:

* Диаграмма вариантов использования (рисунок 5)

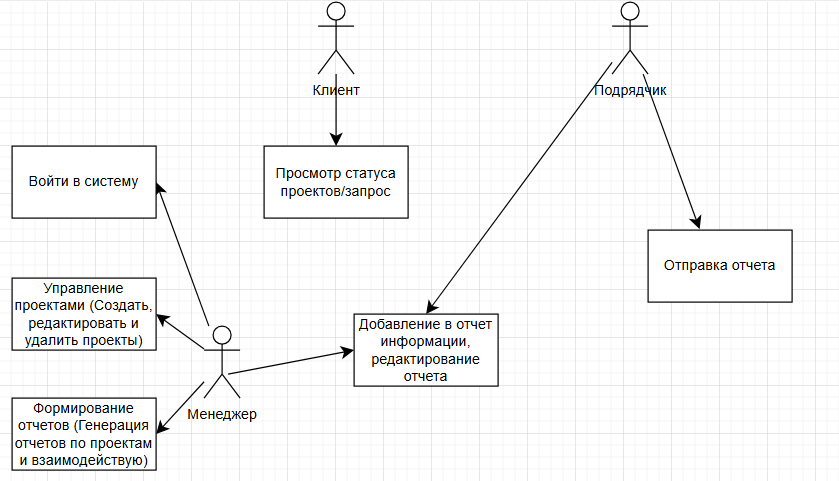


Рис. 5. Диаграмма вариантов использования

* Диаграмма последовательности (рисунок 6)

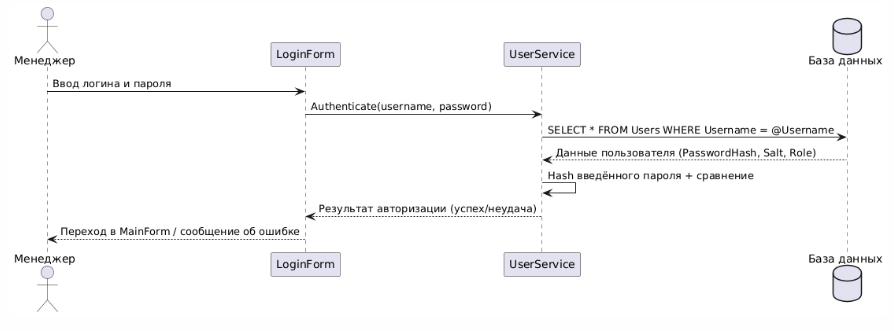


Рис. 6. Диаграмма последовательности

* Диаграмма коммуникаций (рисунок 7)

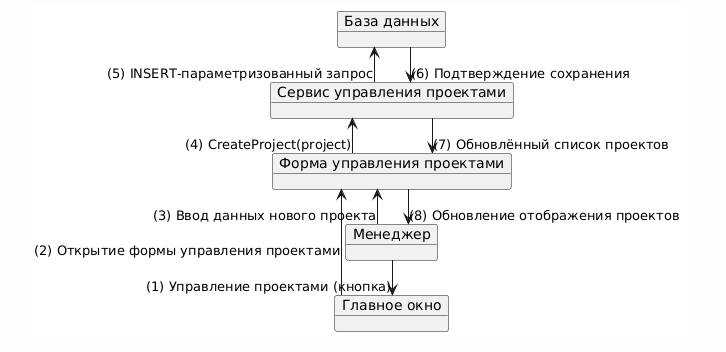


Рис. 7. Диаграмма коммуникаций

* Диаграмма деятельности (рисунок 8)

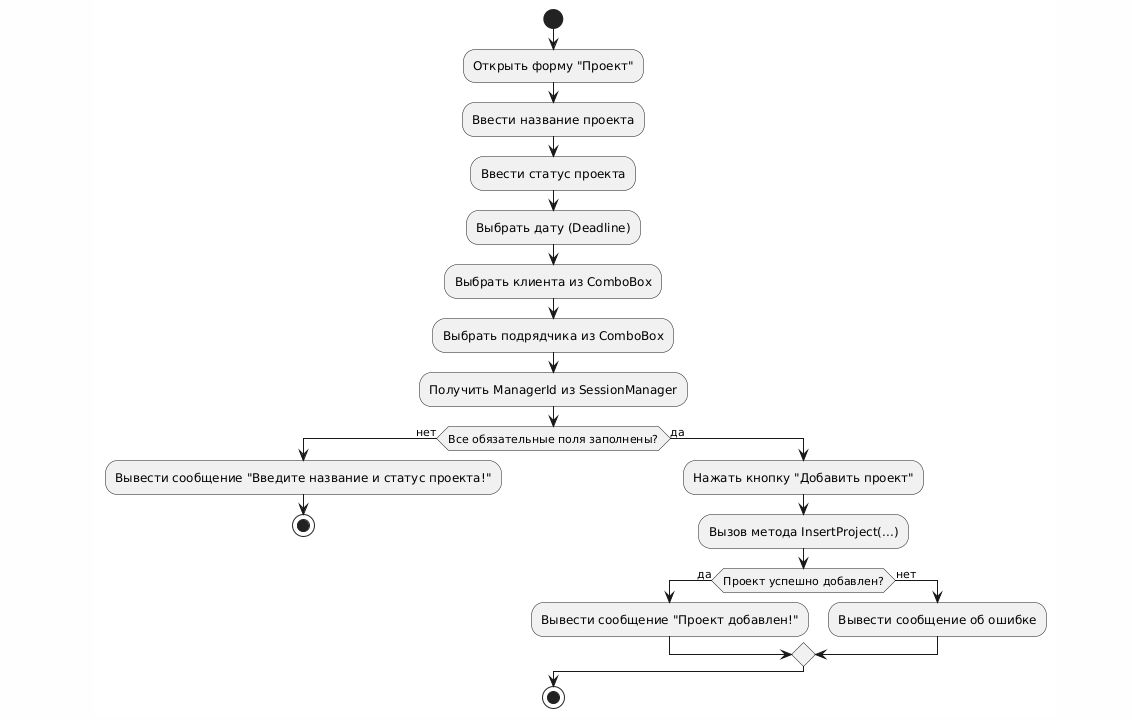


Рис. 8. Диаграмма деятельности

## 2.4 Разработка интерфейса программного продукта

Приложение использует 5 форм: авторизации (рисунок 9), меню выбора (рисунок 10), проекты (рисунок 11), клиенты и подрядчики (рисунок 12) и отчеты (рисунок 13).

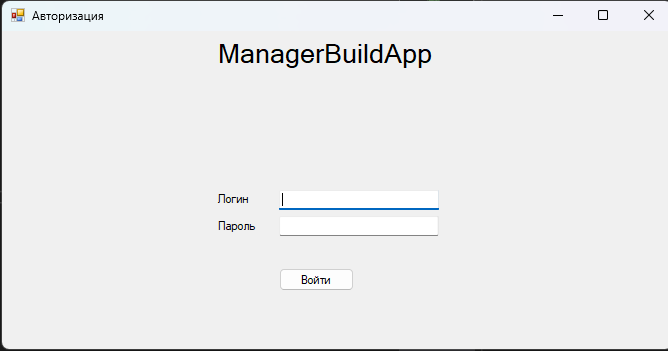


Рис. 9. Авторизация

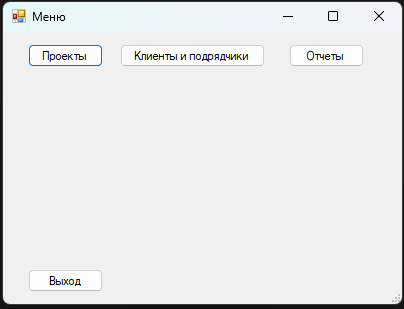


Рис. 10. Меню выбора форм

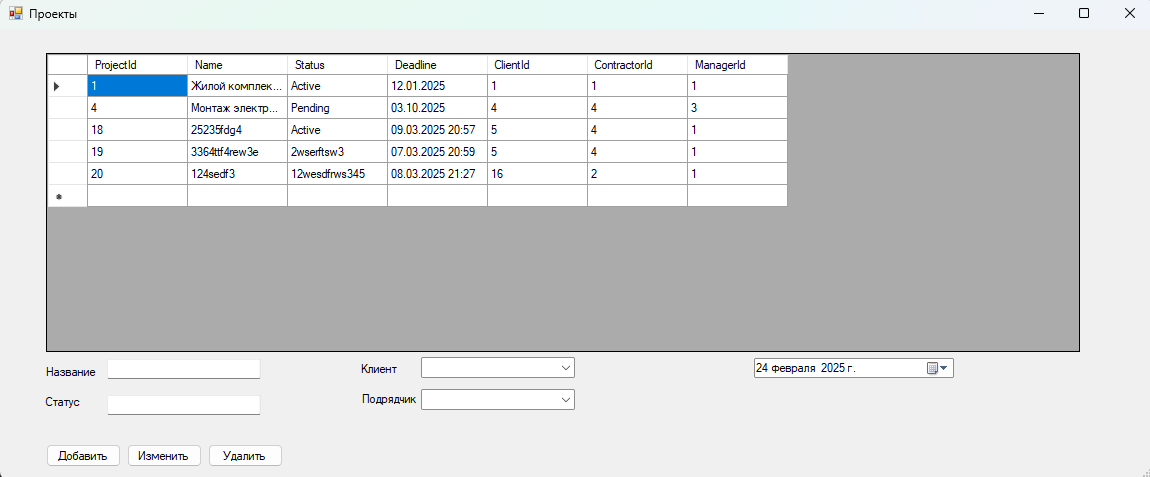


Рис. 11. Проекты

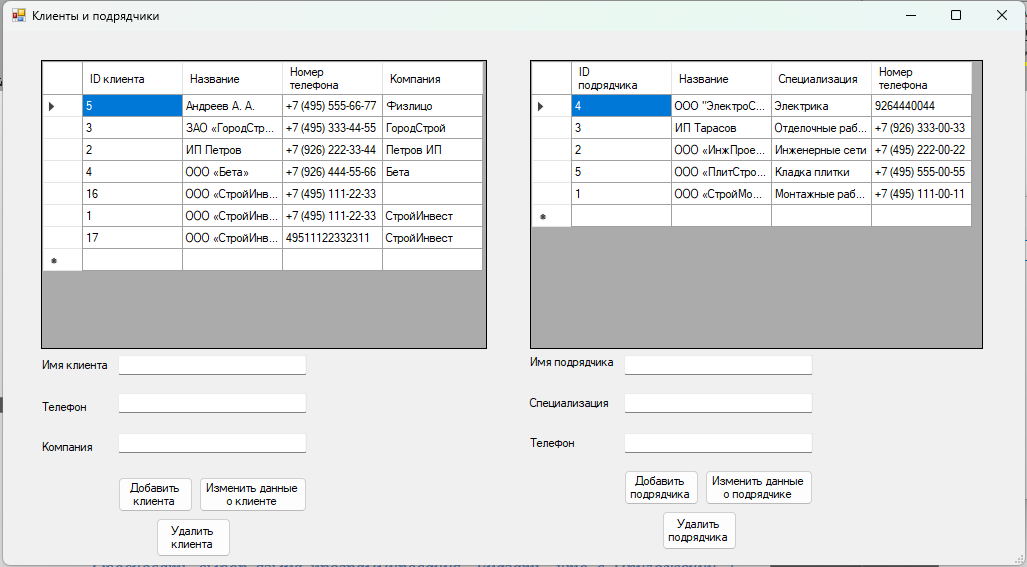


Рис. 12. Клиенты и подрядчики

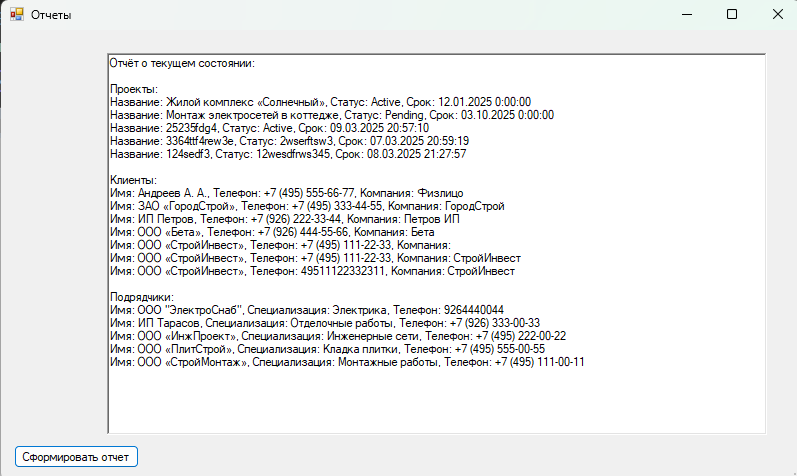


Рис. 13. Отчеты

Пользователь, авторизовавшись, выбирает нужную ему форму, добавляет/редактирует/удаляет клиентов/подрядчиков/проекты или формирует по ним отчет. Язык C# прост и удобен в освоении, Windows Forms без труда позволяют создать простое переносимое приложение для менеджера строительной компании. В **Приложении 2** приведен программный код программного продукта.

## 2.5 Техническая документация программной системы

Руководство пользователя - важной частью технической документации, поскольку предназначено для конечных пользователей системы и позволяет быстро освоить базовые функции программного продукта. Руководство содержит инструкции по установке, первичной настройке и использованию основных модулей системы, таких как авторизация, управление проектами, клиентами и подрядчиками, а также формирование отчётов.

Было разработано Руководство пользователя, представлено в **Приложении 3.**

# Заключение

В результате выполнения курсовой работы была разработана информационная система для автоматизации работы менеджера строительной организации, позволяющая организовать удобное и эффективное управление строительными проектами, клиентами и подрядчиками, а также формирование отчётов по текущему состоянию работ. Данная информационная система предназначена для повышения качества работы менеджеров за счёт сокращения времени обслуживания клиентов, ускорения процессов обработки информации и уменьшения трудоемкости рутинных операций.

В ходе реализации проекта были выявлены следующие недоработки:

* **Однопользовательский режим.** Система рассчитана на работу с одним пользователем, что ограничивает возможности многопользовательского взаимодействия. В дальнейшем планируется внедрение механизмов конкурентного доступа и распределённого управления данными.
* **Неполный функционал отчётности.** На текущем этапе отчёты формируются в виде простого текстового файла. Для повышения информативности и удобства анализа планируется реализовать графическую отчётность и возможность экспорта данных в Excel.
* **Отсутствие интеграции с внешними системами.** В будущем возможно расширение функционала за счёт интеграции с ERP-системами и CRM-платформами, что позволит автоматизировать обмен данными между различными подразделениями организации.

Предложенные пути совершенствования включают разработку многопользовательского режима, расширение возможностей генерации отчётов, а также интеграцию с внешними системами для повышения общей эффективности работы строительной организации.

# Приложение 1. Техническое задание на разработку программного продукта

**1. Введение**

**Наименование системы:** Автоматизация работы менеджера строительной организацией.

**Основание для разработки:** Необходимость повышения эффективности работы менеджеров строительных компаний за счет автоматизации процессов управления проектами, клиентами и подрядчиками.

**Цель разработки:** Создание программного обеспечения, позволяющего вести учет строительных проектов, управлять клиентской базой, формировать отчеты и отслеживать статус выполнения задач.

**2. Требования к системе**

**2.1 Функциональные требования**

Система должна обеспечивать:

* Регистрацию и аутентификацию пользователей.
* Ведение базы данных строительных проектов (название, статус, сроки, ответственные лица).
* Управление клиентами и подрядчиками (добавление, редактирование, удаление данных).
* Формирование отчетов по текущим проектам.
* Просмотр и редактирование информации через графический интерфейс Windows Forms.

**2.2 Нефункциональные требования**

* Приложение должно быть разработано с использованием языка **C#** и **Windows Forms**.
* База данных должна быть реализована с использованием **SQLite** или **SQL Server**.
* Система должна обеспечивать защиту данных пользователей.
* Интерфейс должен быть интуитивно понятным и удобным.

**3. Архитектура и технологии**

* Клиентская часть: **Windows Forms, C#**.
* Серверная часть: **SQL Server / SQLite**.
* Структура данных: **реляционная база данных**.
* Программная архитектура: **Объектно-ориентированное проектирование (ООП)**.

**4. Этапы разработки**

1. Анализ требований.
2. Проектирование архитектуры системы и базы данных.
3. Реализация функционала.
4. Тестирование.
5. Внедрение и документирование.

**5. Ограничения и допущения**

* Приложение разрабатывается для использования на ОС **Windows**.
* Поддержка работы в сети не является приоритетной задачей на данном этапе.
* Доступ к системе осуществляется через учетные записи пользователей.

**6. Требования к интерфейсу**

* Главное окно с доступом к разделам "Проекты", "Клиенты", "Подрядчики", "Отчеты".
* Форма добавления и редактирования информации о проектах.
* Отдельные окна для управления клиентами и подрядчиками.

**7. Требования к безопасности**

* Аутентификация пользователей по логину и паролю.
* Ограничение доступа к функциям системы в зависимости от уровня пользователя.
* Защита базы данных от несанкционированного доступа.

**8. Развертывание и эксплуатация**

* Программа устанавливается на локальные компьютеры пользователей.
* Данные хранятся в локальной базе SQLite или на сервере SQL Server.
* Обучение пользователей осуществляется с помощью документации и инструкций.

**9. Ожидаемые результаты**

Внедрение разработанного ПО позволит:

* Уменьшить затраты времени на ведение документации.
* Исключить вероятность ошибок, связанных с человеческим фактором.
* Повысить эффективность управления строительными проектами и клиентской базой.

# Приложение 2. Исходный программный код информационной системы

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Web;

using System.Windows.Forms;

using ManagerBuildApp.BLL;

namespace ManagerBuildApp

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

}

private void btnLogin\_Click(object sender, EventArgs e)

{

string username = txtUsername.Text.Trim();

string password = txtPassword.Text.Trim();

string checkQuery = "SELECT COUNT(\*) FROM [Users] WHERE [Username] = @Username AND [Password] = @Password";

try

{

// Проверяем, существует ли пользователь

int userCount = ExecuteScalarQuery(checkQuery, username, password);

if (userCount == 1)

{

(int userId, string currentUser, string userRole) = GetUserData(username, password);

SessionManager.CurrentUserId = userId;

SessionManager.CurrentUserName = currentUser;

SessionManager.CurrentUserRole = userRole;

// Или передать в главную форму

MainForm mainForm = new MainForm();

// mainForm.SetUserData(userId, currentUser); // если нужно

this.Hide();

mainForm.ShowDialog();

this.Show();

}

else

{

MessageBox.Show("Неверный логин или пароль!", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при авторизации: " + ex.Message,

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

private int ExecuteScalarQuery(string query, string username, string password)

{

int result = 0;

string connectionString = ConfigurationManager.ConnectionStrings["ManagerBuildApp"].ConnectionString;

using (SqlConnection conn = new SqlConnection(connectionString))

{

using (SqlCommand cmd = new SqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("@Username", username);

cmd.Parameters.AddWithValue("@Password", password);

conn.Open();

object obj = cmd.ExecuteScalar();

if (obj != null)

{

result = Convert.ToInt32(obj);

}

}

}

return result;

}

private (int userId, string currentUser, string userRole) GetUserData(string username, string password)

{

int userId = 0;

string currentUser = string.Empty;

string userRole = string.Empty;

string query = @"

SELECT [UserId], [Username], [Role]

FROM [Users]

WHERE [Username] = @Username AND [Password] = @Password";

string connectionString = ConfigurationManager.ConnectionStrings["ManagerBuildApp"].ConnectionString;

using (SqlConnection conn = new SqlConnection(connectionString))

{

using (SqlCommand cmd = new SqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("@Username", username);

cmd.Parameters.AddWithValue("@Password", password);

conn.Open();

using (SqlDataReader reader = cmd.ExecuteReader())

{

if (reader.Read())

{

userId = reader.GetInt32(0);

currentUser = reader.GetString(1);

userRole = reader.GetString(2);

}

}

}

}

return (userId, currentUser, userRole);

}

}

}

using System;

using System.Windows.Forms;

namespace ManagerBuildApp

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

private void btnProjects\_Click\_1(object sender, EventArgs e)

{

ProjectForm form = new ProjectForm();

form.ShowDialog();

}

private void btnClientsContractors\_Click\_1(object sender, EventArgs e)

{

ClientsAndContractorsForm form = new ClientsAndContractorsForm();

form.ShowDialog();

}

private void btnReports\_Click\_1(object sender, EventArgs e)

{

ReportsForm form = new ReportsForm();

form.ShowDialog();

}

private void btnExit\_Click\_1(object sender, EventArgs e)

{

this.Close();

}

}

}

using System;

using System.Data;

using System.Text.RegularExpressions;

using System.Windows.Forms;

using ManagerBuildApp;

using ManagerBuildApp.BLL;

namespace ManagerBuildApp

{

public partial class ClientsAndContractorsForm : Form

{

public ClientsAndContractorsForm()

{

InitializeComponent();

}

private void ClientsAndContractorsForm\_Load(object sender, EventArgs e)

{

LoadClients();

LoadContractors();

}

private void LoadClients()

{

string query = "SELECT [ClientId], [Name], [Phone], [Company] FROM [Clients]";

DataTable dt = DatabaseHelper.ExecuteQuery(query);

dataGridViewClients.DataSource = dt;

}

private void LoadContractors()

{

string query = "SELECT [ContractorId], [Name], [Specialty], [Phone] FROM [Contractors]";

DataTable dt = DatabaseHelper.ExecuteQuery(query);

dataGridViewContractors.DataSource = dt;

}

// Методы для работы с клиентами:

// Методы для работы с подрядчиками:

private void btnAddContractor\_Click(object sender, EventArgs e)

{

string name = txtContractorName.Text.Trim();

string specialty = txtContractorSpecialty.Text.Trim();

string phone = txtContractorPhone.Text.Trim();

if (!InputValidator.ValidatePhone(phone, out string PhoneErr))

{

MessageBox.Show(PhoneErr, "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!InputValidator.ValidateName(name, out string NameErr))

{

MessageBox.Show(NameErr, "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (ClientOrContractorExist.ContractorExists(name, specialty, phone))

{

MessageBox.Show("Пользователь уже существует");

return;

}

string query = $"INSERT INTO [Contractors]([Name],[Specialty],[Phone]) VALUES('{name}','{specialty}','{phone}')";

DatabaseHelper.ExecuteNonQuery(query);

LoadContractors();

}

private void btnUpdateContractor\_Click(object sender, EventArgs e)

{

if (dataGridViewContractors.SelectedRows.Count > 0)

{

int id = Convert.ToInt32(dataGridViewContractors.SelectedRows[0].Cells["ContractorId"].Value);

string name = txtContractorName.Text.Trim();

string specialty = txtContractorSpecialty.Text.Trim();

string phone = txtContractorPhone.Text.Trim();

if (!InputValidator.ValidatePhone(txtContractorPhone.Text.Trim(), out string PhoneErr))

{

MessageBox.Show(PhoneErr, "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!InputValidator.ValidateName(txtContractorName.Text.Trim(), out string NameErr))

{

MessageBox.Show(NameErr, "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (ClientOrContractorExist.ContractorExists(name, specialty, phone))

{

MessageBox.Show("Пользователь уже существует");

return;

}

string query = $"UPDATE [Contractors] SET [Name] = '{name}', [Specialty] = '{specialty}', [Phone] = '{phone}' WHERE [ContractorId] = {id}";

DatabaseHelper.ExecuteNonQuery(query);

LoadContractors();

}

}

private void btnDeleteContractor\_Click(object sender, EventArgs e)

{

if (dataGridViewContractors.SelectedRows.Count > 0)

{

int id = Convert.ToInt32(dataGridViewContractors.SelectedRows[0].Cells["ContractorId"].Value);

string query = $"DELETE FROM [Contractors] WHERE [ContractorId] = {id}";

DatabaseHelper.ExecuteNonQuery(query);

LoadContractors();

}

}

private void dataGridViewContractors\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridViewContractors.SelectedRows.Count > 0)

{

txtContractorName.Text = dataGridViewContractors.SelectedRows[0].Cells["ContractorName"].Value.ToString();

txtContractorSpecialty.Text = dataGridViewContractors.SelectedRows[0].Cells["Specialty"].Value.ToString();

txtContractorPhone.Text = dataGridViewContractors.SelectedRows[0].Cells["ContractorPhone"].Value.ToString();

}

}

private void btnAddClient\_Click(object sender, EventArgs e)

{

string name = txtClientName.Text.Trim();

string phone = txtClientPhone.Text.Trim();

string company = txtClientCompany.Text.Trim();

if (!InputValidator.ValidatePhone(txtClientPhone.Text.Trim(), out string PhoneErr))

{

MessageBox.Show(PhoneErr, "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!InputValidator.ValidateName(txtClientName.Text.Trim(), out string NameErr))

{

MessageBox.Show(NameErr, "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (ClientOrContractorExist.ClientExists(name, phone, company))

{

MessageBox.Show("Пользователь уже существует");

return;

}

// Если все проверки пройдены, выполняем запрос

string query = $"INSERT INTO [Clients]([Name],[Phone], [Company]) " +

$"VALUES('{name}','{phone}','{company}')";

DatabaseHelper.ExecuteNonQuery(query);

LoadClients();

}

private void dataGridViewClients\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridViewClients\_SelectionChanged\_1(object sender, EventArgs e)

{

if (dataGridViewClients.SelectedRows.Count > 0)

{

txtClientName.Text = dataGridViewClients.SelectedRows[0].Cells["ClientName"].Value.ToString();

txtClientPhone.Text = dataGridViewClients.SelectedRows[0].Cells["ClientPhone"].Value.ToString();

txtClientCompany.Text = dataGridViewClients.SelectedRows[0].Cells["Company"].Value.ToString();

}

}

private void ClientsAndContractorsForm\_Load\_1(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "managerBuildAppDataSet1.Contractors". При необходимости она может быть перемещена или удалена.

this.contractorsTableAdapter.Fill(this.managerBuildAppDataSet1.Contractors);

// TODO: данная строка кода позволяет загрузить данные в таблицу "managerBuildAppDataSet.Clients". При необходимости она может быть перемещена или удалена.

this.clientsTableAdapter.Fill(this.managerBuildAppDataSet.Clients);

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void btnUpdateClient\_Click(object sender, EventArgs e)

{

if (dataGridViewClients.SelectedRows.Count > 0)

{

int id = Convert.ToInt32(dataGridViewClients.SelectedRows[0].Cells["ClientId"].Value);

string name = txtClientName.Text.Trim();

string phone = txtClientPhone.Text.Trim();

string company = txtClientCompany.Text.Trim();

string query = $"UPDATE [Clients] SET [Name] = '{name}', [Phone] = '{phone}', [Company] = '{company}' WHERE [ClientId] = {id}";

if (!InputValidator.ValidatePhone(txtClientPhone.Text.Trim(), out string PhoneErr))

{

MessageBox.Show(PhoneErr, "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!InputValidator.ValidateName(txtClientName.Text.Trim(), out string NameErr))

{

MessageBox.Show(NameErr, "Ошибка ввода", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (ClientOrContractorExist.ClientExists(name, phone, company))

{

MessageBox.Show("Пользователь уже существует");

return;

}

DatabaseHelper.ExecuteNonQuery(query);

LoadClients();

}

}

private void btnDeleteClient\_Click(object sender, EventArgs e)

{

if (dataGridViewClients.SelectedRows.Count > 0)

{

int id = Convert.ToInt32(dataGridViewClients.SelectedRows[0].Cells["ClientId"].Value);

string query = $"DELETE FROM [Clients] WHERE [ClientId] = {id}";

DatabaseHelper.ExecuteNonQuery(query);

LoadClients();

}

}

}

}

# Приложение 3. Руководство пользователя

**1.Введение**  
Данное руководство предназначено для конечных пользователей информационной системы «Автоматизация работы менеджера строительной организацией». Система разработана для упрощения управления строительными проектами, клиентской базой и подрядчиками, а также для формирования отчётов по текущему состоянию работ.

**2. Установка и запуск**

1. Распакуйте архив с программным обеспечением в выбранную папку.
2. Запустите программу, дважды кликнув по файлу **ManagerBuildApp.exe**.

**3.Авторизация**  
При запуске открывается окно авторизации. Введите логин и пароль, полученные от администратора. При успешном входе открывается главное окно системы, в котором отображается информация о пользователе (менеджер или администратор).

**4. Основные возможности системы**

* **Управление проектами:**  
  Пользователь может добавлять, редактировать и удалять проекты. Для добавления нового проекта необходимо заполнить поля «Название», «Статус», «Срок завершения» и выбрать клиента и подрядчика из выпадающих списков. В системе ManagerId автоматически подставляется идентификатор вошедшего пользователя.
* **Управление клиентами и подрядчиками:**  
  Доступны функции добавления, редактирования и удаления записей, что позволяет вести актуальную базу клиентов и подрядчиков.
* **Формирование отчётов:**  
  Система генерирует отчёты по проектам, позволяя оценить статус работ и оперативно принимать решения.

**5. Рекомендации по эксплуатации**

* Перед использованием системы рекомендуется ознакомиться с инструкциями по установке и настройке, а также проверить корректность заполнения обязательных полей.
* При возникновении ошибок или вопросов обращайтесь к администратору системы для получения консультаций и технической поддержки.

# Список использованных источников

1. **Microsoft Corporation.**  
   *C# Programming Guide.* Microsoft Docs, 2021–2023.  
   Доступно по адресу: <https://docs.microsoft.com/en-us/dotnet/csharp/>  
   (Дата обращения: октябрь 2023).
2. **Microsoft Corporation.**  
   *Entity Framework Core Documentation.* Microsoft Docs, 2020–2023.  
   Доступно по адресу: <https://docs.microsoft.com/en-us/ef/>  
   (Дата обращения: октябрь 2023).
3. **Microsoft Corporation.**  
   *SQL Server 2019 Documentation.* Microsoft Docs, 2019–2023.  
   Доступно по адресу: <https://docs.microsoft.com/en-us/sql/sql-server/>  
   (Дата обращения: октябрь 2023).
4. **Microsoft Corporation.**  
   *Windows Forms Overview.* Microsoft Docs, 2021–2023.  
   Доступно по адресу: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/>  
   (Дата обращения: октябрь 2023).
5. **Skeet, J.**  
   *C# in Depth (4th Edition).* Manning Publications, 2019.  
   ISBN: 9781617294532.
6. **Troelsen, A., & Japikse, P.**  
   *Pro C# 7: With .NET and .NET Core.* Apress, 2017.  
   ISBN: 9781484230176.