СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc152768210)

[1 ЦЕЛЬ И ЗАДАЧИ УЧЕБНОЙ ПРАКТИКИ 4](#_Toc152768211)

[1.1 Основные требования по технике безопасности во время прохождения учебной практики 4](#_Toc152768212)

[1.2 Нормативно-правовые документы 5](#_Toc152768213)

[2 ПРЕДМЕТНАЯ ОБЛАСТЬ. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ 7](#_Toc152768214)

[2.1 Описание предметной области 7](#_Toc152768215)

[2.2 Группы пользователей информационной системы 7](#_Toc152768216)

[2.3 Основные требования, предъявляемые к информационной системе 8](#_Toc152768217)

[3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ЭнергоУчет» 10](#_Toc152768218)

[3.1 Моделирование некоторых динамических аспектов системы 10](#_Toc152768219)

[1.1.1 Создание диаграммы вариантов использования 10](#_Toc152768220)

[1.1.2 Создание диаграммы деятельности 11](#_Toc152768221)

[3.2 Разработка функциональной структуры информационной системы. Карта навигации по информационной системе 11](#_Toc152768222)

[3.3 Разработка макета информационной системы «ЭнергоУчет» 11](#_Toc152768223)

[4 РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ «НАЗВАНИЕ» (Название взять из варианта) 12](#_Toc152768224)

[4.1 Моделирование и разработка базы данных 12](#_Toc152768225)

[4.2. Разработка программных модулей информационной системы «НАЗВАНИЕ» (Название взять из варианта) 13](#_Toc152768226)

[5 РАЗРАБОТКА ТЕСТОВЫХ НАБОРОВ И ТЕСТОВЫХ СЦЕНАРИЕВ 13](#_Toc152768227)

[5.1 Назначение эксперимента. Выбор и обоснование методики проведения тестирования 13](#_Toc152768228)

[5.2 Технология тестирования 14](#_Toc152768229)

[5.3 Результаты проведения тестирования 15](#_Toc152768230)

[ЗАКЛЮЧЕНИЕ 16](#_Toc152768231)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 17](#_Toc152768232)

[ПРИЛОЖЕНИЕ А 18](#_Toc152768233)

[ПРИЛОЖЕНИЕ Б 19](#_Toc152768234)

# ВВЕДЕНИЕ

Сфера учета и начисления платы за потребленную электроэнергию является одной из наиболее важных и ответственных областей деятельности предприятия "Хабэнерго". С развитием технологий и повышением требований к точности учета, появилась необходимость в разработке специализированного программного обеспечения, которое бы упростило и автоматизировало процесс учета и начисления платы.

Сегодня на рынке существует множество программных продуктов, предназначенных для учета электроэнергии. Они отличаются функциональностью, возможностями, ценой и т.д. Однако, все они направлены на решение одной задачи - обеспечение точного учета потребляемой электроэнергии и автоматического начисления платы.

Современные программы для учета электроэнергии позволяют не только отслеживать потребление электроэнергии, но и анализировать его, выявлять причины перерасхода, оптимизировать расходы на энергоснабжение и т.д. Кроме того, они обеспечивают максимальную точность учета, что позволяет предотвращать недополучение платежей и снижать риски возникновения конфликтов между предприятием и потребителями.

Таким образом, разработка программного обеспечения для учета и начисления платы за потребленную электроэнергию является актуальной и важной задачей, которая позволит повысить эффективность деятельности предприятия "Хабэнерго" и обеспечить максимально точный учет потребляемой электроэнергии.

# 1 ЦЕЛЬ И ЗАДАЧИ УЧЕБНОЙ ПРАКТИКИ

Разработка программных модулей для системы отслеживания данных множества счетчиков направлена на повышение эффективности и надежности. Целью является разработать программное обеспечение и проектирование информационной системы для учета платы за потребленную электроэнергию на предприятии "Хабэнерго", способствующей обеспечению точному и надежному учету потребления электроэнергии каждой квартиры.

Задачи:

1. Автоматизация учета потребляемой электроэнергии в больших объемах домов.
2. Разработать базу данных для хранения информации о потреблении электроэнергии каждой квартиры.
3. Обеспечение безопасности данных: разработка модулей для защиты конфиденциальности данных, связанных с потреблением электроэнергии каждой квартиры.

Цель и задачи в совокупности направлены на повышение эффективности работы с большими объемами данных и надежность их достоверности через современные информационные технологии.

## Основные требования по технике безопасности во время прохождения учебной практики

К работе в компьютерном классе допускаются лица, ознакомленные с данной инструкцией по технике безопасности и правилам поведения. Необходимо неукоснительно соблюдать правила по технике безопасности, т.к. нарушение этих правил может привести к поражению электрическим током, вызвать возгорание и навредить вашему здоровью.

- Работа обучающихся в компьютерном классе разрешается только в присутствии преподавателя (лаборанта).

- Во время занятий посторонние лица могут находиться в классе только с разрешения преподавателя.

- Во время перемен между уроками проводится обязательное проветривание компьютерного кабинета с обязательным выходом обучающихся из класса.

Каждый обучающийся в ответе за состояние своего рабочего места и сохранность размещенного на нем оборудования.

Обучающийся, допустивший нарушение по технике безопасности и правилам поведения в компьютерном классе, может быть привлечен к дисциплинарной ответственности. Если нарушение техники безопасности связано с причинением имущественного ущерба, обучающийся несет и материальную ответственность в установленном законом порядке.

## Нормативно-правовые документы

Оформление и содержание технического задания должно соответствовать требованиям стандарта «ГОСТ 19.201-78. ЕСПД. Общие положения.

Техническое задание оформляют в соответствии с ГОСТ 19.106-78 на листах формата 11 и 12 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставляются в верхней части листа над текстом.

Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78.

Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

Для внесения изменений или дополнений в техническое задание на последующих стадиях разработки про1раммы или программного изделия выпускают дополнение к нему.

Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

Техническое задание должно содержать следующие разделы:

- введение;

- основания для разработки;

- назначение разработки;

- требования к программе или программному изделию;

- требования к программной документации;

- технико-экономические показатели;

- стадии и этапы разработки;

- порядок контроля и приемки;

- в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

# 2 ПРЕДМЕТНАЯ ОБЛАСТЬ. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ

Предметная область информационной системы — это материальная система или система, характеризующая элементы материального мира, информация о которой хранится и обрабатывается. Предметная область рассматривается как некоторая совокупность реальных объектов и связей между ними.

## 2.1 Описание предметной области

Необходимо разработать и спроектировать информационную систему для эффективного и удобного учета потребления электроэнергии и просмотра данных счетчиков с разных квартир. Регистрацию и хранение информации о каждом квартиросъемщике.

Квартиры - объекты потребления электроэнергии, каждая из которых имеет свой уникальный номер и характеристики, такие как количество жильцов, тип счетчика и тарифный план.

Безопасность и защита данных - меры, предпринимаемые для защиты информации о потреблении электроэнергии каждой квартиры от несанкционированного доступа и утечек данных.

Информационная система "ЭнергоУчет" призвана содействовать повышению эффективности в процессах подсчёта данных электроэнергии.

## 2.2 Группы пользователей информационной системы

Сотрудники, в зависимости от своих функциональных обязанностей, могут выполнять действия, направленные на мониторинг, управление, анализ данных и обеспечение безопасности в рамках проектируемой информационной системы "ЭнегроУчет".

Роли и их Обязанности:

* + Администратор: управление всеми системами и их редактирование; оптимизация работы системы для обеспечения высокой производительности.
  + Инспектор: получение информации о счетчиках всех квартиросъемщиков и редактирование этих данных.
  + Квартиросъемщик: получение информации о показаниях своего счетчика в месяц.

## 2.3 Основные требования, предъявляемые к информационной системе

Данная информационная система должна позволять автоматически вести учет потребляемой электроэнергии.

Кроме того, должна иметь возможность просматривать каждого счетчика и квартиросъемщика.

Перечень требований пользователя к программному продукту:

* + Гибкость: система должна обладать высокой степенью редактирования, чтобы пользователи могли адаптировать её под различные потребности в рамках своих обязанностей; возможность легкого масштабирования системы при увеличении объема данных или числа пользователей.

Надежность информационной системы подразумевает ее функционирование без искажения информации, потери данных по «техническим причинам».

* + Эффективность: обеспечение высокой производительности при обработке данных, особенно в условиях одновременного использования множества пользователей; эффективное использование вычислительных ресурсов, чтобы система функционировала быстро и эффективно.
  + Безопасность: реализация надежной системы аутентификации для проверки личности пользователей и установление строгих прав доступа; и использование шифрования для защиты передаваемой и хранимой информации.

Вывод по главе: было описано, что такое предметная область и её описание; группы пользователей, которые будут пользоваться информационной системой и основные системные требования к ней

# 3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ЭнергоУчет»

Проектированием информационных систем называется многоступенчатый процесс их создания и/или модернизации путём применения упорядоченной совокупности методологий и инструментария. Проектирование (в отличие от моделирования) предполагает работу с пока несуществующим объектом и направлено на создание информационной системы.

## 3.1 Моделирование некоторых динамических аспектов системы

Диаграммы деятельности применяются для моделирования динамических аспектов поведения системы. Эти динамические аспекты могут включать деятельность на любом уровне абстракции в любом представлении системной архитектуры, включая классы, интерфейсы, компоненты и узлы.

При разработке программного обеспечения заблаговременное планирование и моделирование значительно упрощают программирование. На этапе создания концептуальной модели для описания функционального назначения используется диаграмма вариантов использования, а для описания логики процедур и бизнес-процессов используется диаграмма деятельности.

## Создание диаграммы вариантов использования

В приложении А на рисунке 1 представлен пример диаграммы вариантов использования информационной системы «ЭнергоУчет».

## Создание диаграммы деятельности

В приложении А на рисунках 2 – 4 представлена диаграмма деятельности (по ролям) информационной системы «ЭнергоУчет».

## 3.2 Разработка функциональной структуры информационной системы. Карта навигации по информационной системе

Карта переходов по информационной системе «ЭнергоУчет» представлена в приложении А на рисунке 5.

## 3.3 Разработка макета информационной системы «ЭнергоУчет»

В приложении А на рисунке 6представлено окно авторизации.

На этапе авторизации пользователь может ввести логин и пароль, чтобы продолжить работу в информационной системе.

После того, как пользователь успешно прошел авторизацию, он может продолжить работу с таблицами на разных Формах. В приложении А на рисунках 6 – 11 представлены формы, которые будут отображены, в зависимости от прав пользователя в системе. Например, Квартиросъемщик имеет доступ к форме «Tenat»

Дизайн выполнен в простом стиле для того, чтобы пользователь с любым уровнем владения компьютером мог без препятствий работать с программным обеспечением.

Вывод по главе: была спроектирована диаграмма вариантов использования, диаграмма деятельности и макеты будущей информационной системы.

# 4 РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ «ЭнергоУчет»

Прототип — это один из этапов разработки, который заключается в продумывании содержания и расположения важных элементов интерфейса. Прототип — это моделирование конечного продукта. Это интерактивный макет, который может иметь любую степень точности. Основная цель создания прототипов – проверить, насколько последователен путь пользователя, и выявить препятствия, которые могут возникнуть в процессе его взаимодействия с продуктом.

## 4.1 Моделирование и разработка базы данных

Исходя из анализа предметной области, можно выделить шесть сущностей: «Tenat», «Admin», «Inspector», «Registration», «User», «Пользователиr».

В приложении “А” на рисунке 11 представлена логическая модель данных базы данных «ЭнергоУчет».

Создание базы данных происходит в MS SQL, в примере представлено создание базы данных в MS SQL Server.

С помощью раздела «Создание» и конструктора таблиц MS SQL созданы все таблицы и поля, представленные на логической модели данных.

Названия полей и их типы данных представлены в таблицах 1-7.

Таблица 1– Поля таблицы «Apartment»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Apartment\_code | Числовой |
| Flat | Числовой |
| FlatType | Числовой |
| People | Числовой |
| EStove | Логический |
| Building number | Числовой |
| LastName | Текстовый |
| ForstName | Текстовый |
| Patronymic | Текстовый |
| Passport | Текстовый |

Таблица 2 – Поля таблицы «Building»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Building code | Числовой |
| Address | Текстовый |
| District | Текстовый |
| Material | Текстовый |
| Doorway | Числовой |
| Flats | Числовой |
| Floor | Текстовый |
| Owner | Текстовый |

Таблица 3 – Поля таблицы «Counter»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Counte code | Числовой |
| Type | Текстовый |
| Factor | Числовой |
| Flat | Числовой |

Таблица 4– Поля таблицы «Electricity\_tariff»

| Название поля | Тип данных |
| --- | --- |
| Tariff code | Числовой |
| Month | Числовой |
| Price\_per\_month | Денежный |

Таблица 5 – Поля таблицы «Indicators»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Indicator code | Числовой |
| BeginMonth | Числовой |
| EndMonth | Числовой |

Таблица 6 – Поля таблицы «Pay»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Pay code | Числовой |
| Counter | Числовой |
| Tariff | Числовой |
| Day | Числовой |
| Night | Числовой |
| hour24 | Числовой |
| Privilege | Денежный |
| Surname | Текстовый |
| Tariff\_code | Числовой |
| Indicators | Числовой |

Таблица 7 – Поля таблицы «Пользователи»

|  |  |
| --- | --- |
| Название поля | Тип данных |
| Код gользователя | Числовой |
| Логин | Текстовый |
| Пароль | Текстовый |
| Права доступа | Текстовый |
| reg | Числовой |

В приложении А на рисунке 12 представлены все созданные таблицы в обозревателе объектов базы данных.

После того, как таблицы созданы, необходимо создать между ними связи.

В приложении А на рисунке 13 представлена физическая модель данных, отражающая все таблицы и их поля, а также связи между таблицами.

## 4.2. Разработка программных модулей информационной системы «ЭнергоУчет»

Разработанные в MS Visual Studio формы продемонстрированы в приложении А на рисунках 14-18

Свойства элементов формы «User» представлены в таблице 8.

Таблица 8 – Свойство элементов формы «User»

| Объект | Свойство | Значение |
| --- | --- | --- |
| User | Text | «User» |
| StartPosition | CenterScreen |
| Size | 273; 274 |

Событийно-управляемые процедуры в форме «User» представлены в таблице 9

Таблица 9 – Событийно-управляемые процедуры в форме «User»

| Название объекта | Событийно-управляемая процедура |
| --- | --- |
| Класс User | public partial class User : Form  {  public User()  {  InitializeComponent();  }  int k = 0;  } |
| Событие buttonlogin\_Click | private void buttonlogin\_Click(object sender, EventArgs e)  {  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog =уП\_ПМ01\_ИСПП\_5\_Буйлов\_МА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter("select \* from Пользователи",sqlConnect);  DataTable dt = new DataTable();  da.Fill(dt);  Boolean flag = false;  for (int i = 0; i < dt.Rows.Count; i++)  if ((dt.Rows[i]["Логин"].ToString() == textBoxLogin.Text) &&  (dt.Rows[i]["Пароль"].ToString() == textBoxPassword.Text)) {flag = true;k = i; }    if (flag == true)  {switch (dt.Rows[k]["Права\_Доступа"].ToString())  {  case "Админ":  this.Hide();  Admin UsrFrm = new Admin();  UsrFrm.Show(); break;    case "Квартиросъемщик":  this.Hide();  Tenat form3 = new Tenat();  form3.Show();  form3.Registr(dt.Rows[k]["reg"].ToString());  break;  case "Инспектор":  this.Hide();  Inspector form4 = new Inspector();  form4.Show(); break;  }  }  else MessageBox.Show("Неверный пароль или логин!");  } |
| Событие buttonExit\_Click | private void buttonExit\_Click(object sender, EventArgs e)  {  Application.Exit();  } |
| Событие buttonRegistr\_Click | private void buttonRegistr\_Click(object sender, EventArgs e)  {  Registration check = new Registration();  check.Show();  Hide();  } |

Свойства элементов формы «Admin» представлены в таблице 10.

Таблица 10 – Свойство элементов формы «Admin»

| Объект | Свойство | Значение |
| --- | --- | --- |
| Admin | Text | «Admin» |
| StartPosition | CenterScreen |
| Size | 1135; 685 |

Событийно-управляемые процедуры в форме «Admin» представлены в таблице 11

Таблица 11 – Событийно-управляемые процедуры в форме «Admin»

| Название объекта | Событийно-управляемая процедура |
| --- | --- |
| Класс Admin | public partial class Admin : System.Windows.Forms.Form  {  public Admin()  {  InitializeComponent();  }  } |
| Событие Form2\_Load | private void Form2\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Electricity\_tariff". При необходимости она может быть перемещена или удалена.  this.electricity\_tariffTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Electricity\_tariff);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Indicators". При необходимости она может быть перемещена или удалена.  this.indicatorsTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Indicators);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Pay". При необходимости она может быть перемещена или удалена.  this.payTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Pay);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Counter". При необходимости она может быть перемещена или удалена.  this.counterTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Counter);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Building". При необходимости она может быть перемещена или удалена.  this.buildingTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Building);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments". При необходимости она может быть перемещена или удалена.  this.apartmentsTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments);  } |
| События buttonPrev\_Click | private void buttonPrev\_Click(object sender, EventArgs e)  {  int i = dataGridView1.RowCount;  int index = dataGridView1.CurrentRow.Index;  dataGridView1.Rows[index].Selected = true;  if (index != 0)  {  dataGridView1.CurrentCell = dataGridView1[0, index - 1];  }  else  {  dataGridView1.Rows[index].Selected = true;  dataGridView1.CurrentCell = dataGridView1[0, i - 1];  }  } |
| События buttonNext\_Click | private void buttonNext\_Click(object sender, EventArgs e)  {  int i = dataGridView1.RowCount;  int index = dataGridView1.CurrentRow.Index;  if ((i - 1) <= index) { dataGridView1.CurrentCell = dataGridView1[0, 0]; }  else  {  dataGridView1.Rows[index].Selected = true;  dataGridView1.CurrentCell = dataGridView1[0, index + 1];  }  } |
| События buttonPrev2\_Click | private void buttonPrev2\_Click(object sender, EventArgs e)  {  int i = dataGridView2.RowCount;  int index = dataGridView2.CurrentRow.Index;  dataGridView2.Rows[index].Selected = true;  if (index != 0)  {  dataGridView2.CurrentCell = dataGridView2[0, index - 1];  }  else  {  dataGridView2.Rows[index].Selected = true;  dataGridView2.CurrentCell = dataGridView2[0, i - 1];  }  } |
| События buttonNext2\_Click | private void buttonNext2\_Click(object sender, EventArgs e)  {  int i = dataGridView2.RowCount;  int index = dataGridView2.CurrentRow.Index;  if ((i - 1) <= index) { dataGridView2.CurrentCell = dataGridView2[0, 0]; }  else  {  dataGridView2.Rows[index].Selected = true;  dataGridView2.CurrentCell = dataGridView2[0, index + 1];  }  } |
| События buttonPrev3\_Click | private void buttonPrev3\_Click(object sender, EventArgs e)  {  int i = dataGridView3.RowCount;  int index = dataGridView3.CurrentRow.Index;  dataGridView3.Rows[index].Selected = true;  if (index != 0)  {  dataGridView3.CurrentCell = dataGridView3[0, index - 1];  }  else  {  dataGridView3.Rows[index].Selected = true;  dataGridView3.CurrentCell = dataGridView3[0, i - 1];  }  } |
| События buttonNext3\_Click | private void buttonNext3\_Click(object sender, EventArgs e)  {  int i = dataGridView3.RowCount;  int index = dataGridView3.CurrentRow.Index;  if ((i - 1) <= index) { dataGridView3.CurrentCell = dataGridView3[0, 0]; }  else  {  dataGridView3.Rows[index].Selected = true;  dataGridView3.CurrentCell = dataGridView3[0, index + 1];  }  } |
| События buttonPrev5\_Click | private void buttonPrev5\_Click(object sender, EventArgs e)  {  int i = dataGridView6.RowCount;  int index = dataGridView6.CurrentRow.Index;  dataGridView6.Rows[index].Selected = true;  if (index != 0)  {  dataGridView6.CurrentCell = dataGridView6[0, index - 1];  }  else  {  dataGridView6.Rows[index].Selected = true;  dataGridView6.CurrentCell = dataGridView6[0, i - 1];  }  } |
| События buttonNext5\_Click | private void buttonNext5\_Click(object sender, EventArgs e)  {  int i = dataGridView6.RowCount;  int index = dataGridView6.CurrentRow.Index;  if ((i - 1) <= index) { dataGridView6.CurrentCell = dataGridView6[0, 0]; }  else  {  dataGridView6.Rows[index].Selected = true;  dataGridView6.CurrentCell = dataGridView6[0, index + 1];  }  } |
| События buttonPrev6\_Click | private void buttonPrev6\_Click(object sender, EventArgs e)  {  int i = dataGridView5.RowCount;  int index = dataGridView5.CurrentRow.Index;  dataGridView5.Rows[index].Selected = true;  if (index != 0)  {  dataGridView5.CurrentCell = dataGridView5[0, index - 1];  }  else  {  dataGridView5.Rows[index].Selected = true;  dataGridView5.CurrentCell = dataGridView5[0, i - 1];  }  } |
| События buttonNext6\_Click | private void buttonNext6\_Click(object sender, EventArgs e)  {  int i = dataGridView5.RowCount;  int index = dataGridView5.CurrentRow.Index;  if ((i - 1) <= index) { dataGridView5.CurrentCell = dataGridView5[0, 0]; }  else  {  dataGridView5.Rows[index].Selected = true;  dataGridView5.CurrentCell = dataGridView5[0, index + 1];  }  } |
| События buttonPrev4\_Click | private void buttonPrev4\_Click(object sender, EventArgs e)  {  int i = dataGridView4.RowCount;  int index = dataGridView4.CurrentRow.Index;  dataGridView4.Rows[index].Selected = true;  if (index != 0)  {  dataGridView4.CurrentCell = dataGridView4[0, index - 1];  }  else  {  dataGridView4.Rows[index].Selected = true;  dataGridView4.CurrentCell = dataGridView4[0, i - 1];  }  } |
| События buttonNext4\_Click | private void buttonNext4\_Click(object sender, EventArgs e)  {  int i = dataGridView4.RowCount;  int index = dataGridView4.CurrentRow.Index;  if ((i - 1) <= index) { dataGridView4.CurrentCell = dataGridView4[0, 0]; }  else  {  dataGridView4.Rows[index].Selected = true;  dataGridView4.CurrentCell = dataGridView4[0, index + 1];  }  } |
| События buttonBack\_Click | private void buttonBack\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| События buttonBack2\_Click | private void buttonBack2\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| События buttonBack3\_Click | private void buttonBack3\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| События buttonBack4\_Click | private void buttonBack4\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| События buttonBack5\_Click | private void buttonBack5\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| События buttonBack6\_Click | private void buttonBack6\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| Событие buttonUpdate\_Click | private void buttonUpdate\_Click(object sender, EventArgs e)  {  apartmentsTableAdapter.Update(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet);  } |
| Событие buttonUpdate2\_Click | private void buttonUpdate2\_Click(object sender, EventArgs e)  {  buildingTableAdapter.Update(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet);  } |
| Событие buttonUpdate3\_Click | private void buttonUpdate3\_Click(object sender, EventArgs e)  {  counterTableAdapter.Update(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Counter);  } |
| Событие buttonUpdate4\_Click | private void buttonUpdate4\_Click(object sender, EventArgs e)  {  payTableAdapter.Update(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Pay);  } |
| Событие buttonUpdat5\_Click | private void buttonUpdate5\_Click(object sender, EventArgs e)  {  electricity\_tariffTableAdapter.Update(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Electricity\_tariff);  } |
| Событие buttonUpdate6\_Click | private void buttonUpdate6\_Click(object sender, EventArgs e)  {  indicatorsTableAdapter.Update(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Indicators);  } |
| Событие buttonAdd\_Click | private void buttonAdd\_Click(object sender, EventArgs e)  {  //Добавление нового пользователя через запрос SQl  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ01\_ИСПП\_5\_Буйлов\_МА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter($"INSERT INTO Apartments(Flat,FlatType,People,EStove,Building\_number,Lastname,ForstName,Patronymic,Passport) VALUES('{textBox1.Text}','{textBox4.Text}','{textBox2.Text}','{textBox9.Text}','{textBox17.Text}','{textBox3.Text}','{textBox11.Text}','{textBox10.Text}','{textBox8.Text}');", sqlConnect); ;  DataTable dt = new DataTable();  da.Fill(dt);  //Перерисовка таблицы в datagridview  apartmentsTableAdapter.Fill(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments);  MessageBox.Show("Успешно добавлен");  } |
| Событие buttonDelete\_Click | private void buttonDelete\_Click(object sender, EventArgs e)  {  //Удаление пользователя  DialogResult result = MessageBox.Show("Выберите один из вариантов", "Сообщение", MessageBoxButtons.OKCancel);  if (result == DialogResult.OK)  {  for (int j = 0; j < dataGridView1.RowCount; j++)  {  if (dataGridView1.Rows[j].Cells[0].Value.ToString().Contains(Kod\_apart.Text))  {  dataGridView1.Rows[j].Selected = true;  dataGridView1.CurrentCell = dataGridView1[0, j];  dataGridView1.Rows.RemoveAt(j);  break;  }  }  }    } |
| Событие buttonAdd2\_Click | private void buttonAdd2\_Click(object sender, EventArgs e)  {  //Добавление нового пользователя через запрос SQl  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ01\_ИСПП\_5\_Буйлов\_МА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter($"INSERT INTO Building (Address,District,Material,Doorway,Flats,Floor,Owner) VALUES('{textBox16.Text}','{textBox13.Text}','{textBox14.Text}','{textBox15.Text}','{textBox7.Text}','{textBox5.Text}','{textBox6.Text}');", sqlConnect); ;  DataTable dt = new DataTable();  da.Fill(dt);  //Перерисовка таблицы в datagridview  buildingTableAdapter.Fill(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Building);  MessageBox.Show("Успешно добавлен");  } |
| Событие buttonDelete2\_Click | private void buttonDelete2\_Click(object sender, EventArgs e)  {  //Удаление пользователя  DialogResult result = MessageBox.Show("Выберите один из вариантов", "Сообщение", MessageBoxButtons.OKCancel);  if (result == DialogResult.OK)  {  for (int j = 0; j < dataGridView2.RowCount; j++)  {  if (dataGridView2.Rows[j].Cells[0].Value.ToString().Contains(Kod\_build.Text))  {  dataGridView2.Rows[j].Selected = true;  dataGridView2.CurrentCell = dataGridView2[0, j];  dataGridView2.Rows.RemoveAt(j);  break;  }  }  }    } |
| Событие buttonAdd3\_Click | private void buttonAdd3\_Click(object sender, EventArgs e)  {  //Добавление нового пользователя через запрос SQl  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ01\_ИСПП\_5\_Буйлов\_МА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter($"INSERT INTO Counter (Type,Factor,Flat) VALUES('{textBox23.Text}','{textBox21.Text}','{textBox12.Text}');", sqlConnect); ;  DataTable dt = new DataTable();  da.Fill(dt);  //Перерисовка таблицы в datagridview  counterTableAdapter.Fill(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Counter);  MessageBox.Show("Успешно добавлен");  } |
| Событие buttonDelete3\_Click | private void buttonDelete3\_Click(object sender, EventArgs e)  {  //Удаление пользователя  DialogResult result = MessageBox.Show("Выберите один из вариантов", "Сообщение", MessageBoxButtons.OKCancel);  if (result == DialogResult.OK)  {  for (int j = 0; j < dataGridView3.RowCount; j++)  {  if (dataGridView3.Rows[j].Cells[0].Value.ToString().Contains(Kod\_count.Text))  {  dataGridView3.Rows[j].Selected = true;  dataGridView3.CurrentCell = dataGridView3[0, j];  dataGridView3.Rows.RemoveAt(j);  break;  }  }  }  } |
| Событие buttonAdd5\_Click | private void buttonAdd5\_Click(object sender, EventArgs e)  {  //Добавление нового пользователя через запрос SQl  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ01\_ИСПП\_5\_Буйлов\_МА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter($"INSERT INTO Electricity\_tariff (Month,Price\_per\_month) VALUES('{textBox20.Text}','{textBox19.Text}');", sqlConnect); ;  DataTable dt = new DataTable();  da.Fill(dt);  //Перерисовка таблицы в datagridview  electricity\_tariffTableAdapter.Fill(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Electricity\_tariff);  MessageBox.Show("Успешно добавлен");  } |
| Событие buttonDelete5\_Click | private void buttonDelete5\_Click(object sender, EventArgs e)  {  //Удаление пользователя  DialogResult result = MessageBox.Show("Выберите один из вариантов", "Сообщение", MessageBoxButtons.OKCancel);  if (result == DialogResult.OK)  {  for (int j = 0; j < dataGridView6.RowCount; j++)  {  if (dataGridView6.Rows[j].Cells[0].Value.ToString().Contains(Kod\_elect.Text))  {  dataGridView6.Rows[j].Selected = true;  dataGridView6.CurrentCell = dataGridView6[0, j];  dataGridView6.Rows.RemoveAt(j);  break;  }  }  }  } |
| Событие buttonAdd6\_Click | private void buttonAdd6\_Click(object sender, EventArgs e)  {  //Добавление нового пользователя через запрос SQl  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ01\_ИСПП\_5\_Буйлов\_МА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter($"INSERT INTO Indicators (BeginMonth,EndMonth) VALUES('{textBox22.Text}','{textBox18.Text}');", sqlConnect); ;  DataTable dt = new DataTable();  da.Fill(dt);  //Перерисовка таблицы в datagridview  indicatorsTableAdapter.Fill(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Indicators);  MessageBox.Show("Успешно добавлен");  } |
| Событие buttonDelete6\_Click | private void buttonDelete6\_Click(object sender, EventArgs e)  {  //Удаление пользователя  DialogResult result = MessageBox.Show("Выберите один из вариантов", "Сообщение", MessageBoxButtons.OKCancel);  if (result == DialogResult.OK)  {  for (int j = 0; j < dataGridView5.RowCount; j++)  {  if (dataGridView5.Rows[j].Cells[0].Value.ToString().Contains(Kod\_ind.Text))  {  dataGridView5.Rows[j].Selected = true;  dataGridView5.CurrentCell = dataGridView5[0, j];  dataGridView5.Rows.RemoveAt(j);  break;  }  }  }  } |
| Событие buttonAdd4\_Click | private void buttonAdd4\_Click(object sender, EventArgs e)  {  //Добавление нового пользователя через запрос SQl  SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog = уП\_ПМ01\_ИСПП\_5\_Буйлов\_МА; Integrated Security = True");  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter($"INSERT INTO Pay (Counter,Tariff,Day,Night,hour24,Privilege,Surname,Tariff\_code,Indicators) VALUES('{textBox32.Text}','{textBox30.Text}','{textBox31.Text}','{textBox27.Text}','{textBox29.Text}','{textBox25.Text}','{textBox26.Text}','{textBox28.Text}','{textBox24.Text}');", sqlConnect); ;  DataTable dt = new DataTable();  da.Fill(dt);  //Перерисовка таблицы в datagridview  payTableAdapter.Fill(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Pay);  MessageBox.Show("Успешно добавлен");  } |
| Событие buttonDelete4\_Click | private void buttonDelete4\_Click(object sender, EventArgs e)  {  //Удаление пользователя  DialogResult result = MessageBox.Show("Выберите один из вариантов", "Сообщение", MessageBoxButtons.OKCancel);  if (result == DialogResult.OK)  {  for (int j = 0; j < dataGridView5.RowCount; j++)  {  if (dataGridView4.Rows[j].Cells[0].Value.ToString().Contains(Kod\_pay.Text))  {  dataGridView4.Rows[j].Selected = true;  dataGridView4.CurrentCell = dataGridView4[0, j];  dataGridView4.Rows.RemoveAt(j);  break;  }  }  }  } |

Свойства элементов формы «Inspector» представлены в таблице 12.

Таблица 12 – Свойство элементов формы «Inspector»

| Объект | Свойство | Значение |
| --- | --- | --- |
| Inspector | Text | «Inspector» |
| StartPosition | CenterScreen |
| Size | 320; 354 |

Событийно-управляемые процедуры в форме «Inspector» представлены в таблице 13

Таблица 13– Событийно-управляемые процедуры в форме «Inspector»

| Название объекта | Событийно-управляемая процедура |
| --- | --- |
| Класс Inspector | public partial class Inspector : Form  {  public Inspector()  {  InitializeComponent();  } |
| Событие button1\_Click | private void button1\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| Событие Form4\_Load | private void Form4\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet1.Pay". При необходимости она может быть перемещена или удалена.  this.payTableAdapter1.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet1.Pay);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Pay". При необходимости она может быть перемещена или удалена.    } |
| Событие buttonNext\_Click | private void buttonNext\_Click(object sender, EventArgs e)  {  int i = dataGridView1.RowCount;  int index = dataGridView1.CurrentRow.Index;  if ((i - 1) <= index) { dataGridView1.CurrentCell = dataGridView1[0, 0]; }  else  {  dataGridView1.Rows[index].Selected = true;  dataGridView1.CurrentCell = dataGridView1[0, index + 1];  }    } |
| Событие buttonPrev\_Click | private void buttonPrev\_Click(object sender, EventArgs e)  {  int i = dataGridView1.RowCount;  int index = dataGridView1.CurrentRow.Index;  dataGridView1.Rows[index].Selected = true;  if (index != 0)  {  dataGridView1.CurrentCell = dataGridView1[0, index - 1];  }  else  {  dataGridView1.Rows[index].Selected = true;  dataGridView1.CurrentCell = dataGridView1[0, i - 1];  }  } |
| Событие buttonOpen\_Click | private void buttonOpen\_Click(object sender, EventArgs e)  {  if (k == 0)  {  dataGridView1.Visible = true;  this.Width = 900;  buttonOpen.Text = "Закрыть";  k = 1;  }  else  {  dataGridView1.Visible = false;  this.Width = 320;  buttonOpen.Text = "Открыть";  k = 0;  }  } |
| Событие buttonUpdate\_Click | private void buttonUpdate\_Click(object sender, EventArgs e)  {  payTableAdapter1.Update(уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet1.Pay);  } |

Свойства элементов формы «Registration» представлены в таблице 14.

Таблица 14– Событийно-управляемые процедуры в форме «Registration»

| Объект | Свойство | Значение |
| --- | --- | --- |
| Regisrationt | Text | «Regisrationt» |
| StartPosition | CenterScreen |
| Size | 509; 416 |

Событийно-управляемые процедуры в форме «Regisrationt» представлены в таблице 15

Таблица 15 – Событийно-управляемые процедуры в форме «Regisrationt»

| Название объекта | Событийно-управляемая процедура |
| --- | --- |
| Класс Registration | public partial class Registration : Form  {  public Registration()  {  InitializeComponent();  } |
| Событие buttonRegister\_Click | private void buttonRegister\_Click(object sender, EventArgs e)  {  using (SqlConnection sqlConnect = new SqlConnection("Data Source=sql;Initial Catalog =уП\_ПМ01\_ИСПП\_5\_Буйлов\_МА; Integrated Security = True"))  {  sqlConnect.Open();  SqlDataAdapter da = new SqlDataAdapter("select \* from Пользователи", sqlConnect);  DataTable dt = new DataTable();  da.Fill(dt);  Boolean Логин = true;  for (int i = 0; i < dt.Rows.Count; i++)  {  if (dt.Rows[i]["Логин"].ToString() == textBoxЛогин.Text)  {  Логин = false;  }  }  if (textBoxЛогин.Text== "" || textBoxПароль.Text== "")  {  MessageBox.Show("Введите все данные!");  }  else  {  if (Логин == false)  {  MessageBox.Show("Пользователь уже существует!");  }  else  {      SqlDataAdapter infol2 = new SqlDataAdapter($"INSERT INTO Apartments(Flat, FlatType,People,Building\_number,Lastname,ForstName,Patronymic,Passport,[EStove])VALUES('{textBoxКвартира.Text}','{textBoxТипКвартиры.Text}','{textBoxЛюди.Text}','{textBoxДом.Text}','{textBoxИмя.Text}','{textBoxФамилия.Text}','{textBoxОтчество.Text}','{textBoxПаспорт.Text}','{checkBox1.Checked}')", sqlConnect);  DataTable dt2 = new DataTable();  int i = dataGridView1.RowCount;  dataGridView1.Rows[index].Selected = true;  dataGridView1.CurrentCell = dataGridView1[0, i - 2];    infol2.Fill(dt2);SqlDataAdapter infol1 = new SqlDataAdapter($"INSERT INTO Пользователи(Логин, Пароль,Права\_Доступа,reg)VALUES('{textBoxЛогин.Text}','{textBoxПароль.Text}','Квартиросъемщик','{label4.Text}')", sqlConnect);  DataTable dt1 = new DataTable();  infol1.Fill(dt1);  MessageBox.Show("Пользователь создан!");  User check = new User();  check.Show();  Hide();  }  }  }  } |
| Событие buttonBack\_Click | private void buttonBack\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| Событие Registration\_Load | private void Registration\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Counter". При необходимости она может быть перемещена или удалена.  this.counterTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Counter);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments". При необходимости она может быть перемещена или удалена.  this.apartmentsTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments);  } |

Свойства элементов формы «Tenat» представлены в таблице 16.

Таблица 16– Событийно-управляемые процедуры в форме «Tenat»

| Объект | Свойство | Значение |
| --- | --- | --- |
| Tenat | Text | «Tenat» |
| StartPosition | CenterScreen |
| Size | 329; 385 |

Событийно-управляемые процедуры в форме «Tenat» представлены в таблице 17

Таблица 17 – Событийно-управляемые процедуры в форме «Tenat»

| Название объекта | Событийно-управляемая процедура |
| --- | --- |
| Класс Tenat | public partial class Tenat : Form  {  public Tenat()  {  InitializeComponent();  } |
| Событие buttonBack\_Click | private void buttonBack\_Click(object sender, EventArgs e)  {  User check = new User();  check.Show();  Hide();  } |
| Событие Registr | public void Registr(string reg)  {  regist = reg;  label3.Text = regist;  for (int j = 0; j < dataGridView1.RowCount; j++)  { if (dataGridView1.Rows[j].Cells[0].Value.ToString().Contains(label3.Text))  {  dataGridView1.Rows[j].Selected = true;  dataGridView1.CurrentCell = dataGridView1[0, j];  break;  }    }  } |
| Событие Form3\_Load | private void Form3\_Load(object sender, EventArgs e)  {  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments". При необходимости она может быть перемещена или удалена.  this.apartmentsTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Electricity\_tariff". При необходимости она может быть перемещена или удалена.  this.electricity\_tariffTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Electricity\_tariff);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Indicators". При необходимости она может быть перемещена или удалена.  this.indicatorsTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Indicators);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Pay". При необходимости она может быть перемещена или удалена.  this.payTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Pay);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Counter". При необходимости она может быть перемещена или удалена.  this.counterTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Counter);  // TODO: данная строка кода позволяет загрузить данные в таблицу "уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments". При необходимости она может быть перемещена или удалена.  this.apartmentsTableAdapter.Fill(this.уП\_ПМ01\_ИСПП\_5\_Буйлов\_МАDataSet.Apartments);  } |
| Метод searchInDataGridView | public void searchInDataGridView(string k)  {  /\*Метод для поиска данных на datagridView\*/  for (int j = 0; j < dataGridView2.RowCount; j++)  {  dataGridView2.Rows[j].Selected = false;  if (dataGridView2.Rows[j].Cells[3].Value != null)  {  if (dataGridView2.Rows[j].Cells[3].Value.ToString().Contains(k))  {  dataGridView2.Rows[j].Selected = true;  dataGridView2.CurrentCell = dataGridView2[3, j];  break;  }  }  }  for (int j = 0; j < dataGridView3.RowCount; j++)  {    if (dataGridView3.Rows[j].Cells[1].Value.ToString().Contains(counter.Text))  {  dataGridView3.Rows[j].Selected = true;  dataGridView3.CurrentCell = dataGridView3[1, j];  break;  }    }  for (int j = 0; j < dataGridView4.RowCount; j++)  {  dataGridView4.Rows[j].Selected = false;  if (dataGridView4.Rows[j].Cells[0].Value != null)  {  if (dataGridView4.Rows[j].Cells[0].Value.ToString().Contains(indi.Text))  {  dataGridView4.Rows[j].Selected = true;  dataGridView4.CurrentCell = dataGridView4[0, j];  break;  }  }  }  for (int j = 0; j < dataGridView5.RowCount; j++)  {  dataGridView5.Rows[j].Selected = false;  if (dataGridView4.Rows[j].Cells[0].Value != null)  {  if (dataGridView5.Rows[j].Cells[0].Value.ToString().Contains(Tarif.Text))  {  dataGridView5.Rows[j].Selected = true;  dataGridView5.CurrentCell = dataGridView5[0, j];  break;  }  }  }  } |

Вывод по главе: была спроектирована и разработана система базы данных, были полностью спроектированы и разработаны все программные модули информационной системы «ЭнергоУчет».

# 5 РАЗРАБОТКА ТЕСТОВЫХ НАБОРОВ И ТЕСТОВЫХ СЦЕНАРИЕВ

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Тестовый сценарий – это неавтоматизированный или автоматизированный сценарий, содержащий инструкции по реализации тестового набора. Тестовый сценарий может быть написан вручную (для выполнения человеком) либо полностью или частично автоматизирован.

## 5.1 Назначение эксперимента. Выбор и обоснование методики проведения тестирования

В данном разделе проведены два вида тестирования. Провести анализ тестирования и выбрать, как будет проходить тестирование ИС.

Модульное тестирование. Цель: проверить, что код работает именно так, как должен (при заданных входных параметрах выдает предполагаемый результат).

Анализ покрытия кода тестами. Цель: проверить, что весь наш код отрабатывает при модульном тестировании, что нет, не участвующих в тестировании участков кода.

Анализ стилистики кода. Цель: проверка кода на соблюдение стандартов разработки (отраслевых, корпоративных, принятых на проекте).

Анализ производительности. Цель: проанализировать производительность ключевых операций, потребление памяти, утечку памяти.

## 5.2 Технология тестирования

Рассмотрим, какие технологии можно применить для модульного тестирования и покрытия кода тестами.

Как в ходе выполнения проекта, так и при развитии системы, изменения в ее алгоритмы вносятся в соответствии с техническими заданиями, хотя данный документ может иметь различные названия. В технических заданиях, как правило, описываются критерии приемки: как мы и заказчик убедимся в том, что заказанный функционал работает верно.

Автоматизированное тестирование лишено данных недостатков. Конечно, возникает отдельная задача кодирования данного теста на основании критериев приемки, но это скорее дисциплинирующий фактор…

Прежде чем начинать юзабилити-тестирование необходимо собрать группу пользователей, которые будут тестировать данную информационную систему. Количество привлеченных пользователей должно быть не менее пяти человек. Пользователям будет представлен список некоторых вопросов, относящихся к графическому интерфейсу.

В таблице 18 представлены сведения о пользователях.

Таблица 18 – Информация о пользователях

|  |  |  |  |
| --- | --- | --- | --- |
| Фамилия Имя Отчество | Пол | Возраст | Образование |
| Неверов Дмитрий Сергеевич | Мужской | 18 | Общее образование |
| Шелков Илья Дмитреевич | Мужской | 19 | Общее образование |
| Макуров Владимир Алексеевич | Мужской | 18 | Общее образование |
| Швидко Иван Анатольевич | Мужской | 18 | Общее образование |
| Степурин Никита Сергеевич | Мужской | 20 | Среднее образование |

## 5.3 Результаты проведения тестирования

В ходе тестирования был применен метод нагрузочного тестирования и юзабилити-тестирования, которые проверяют работу при нагрузке системы и сделать простым и удобным в пользовании информационной системы.

Суть тестирования заключается в проверке работы всех обязательных функций.

Результат проведения нагрузочного тестирования продемонстрирован в приложении «А» на рисунке 19, на котором указывается нагрузка на процессор и оперативную память.

Исходя из результатов тестирования, можно прийти к выводу, что нагрузка, оказываемая на центральный процессор и память, не имеет критичных значений. Это значит, что показатель производительности информационной системы на приемлемом уровне, то есть сбои и длительная загрузка не должны помешать пользователю при работе с данным программным продуктом.

Таблица 19**–** Результаты проведения юзабилити-тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вопрос | Неверов Дмитрий Сергеевич | Макуров Владимир Алексеевич | Швидко Ивана Анатолевич | Шелков Илья Дмитреевич | Степурин Никита Сергеевич |
| Присутствуют ли такие дизайн-элементы, которые неприятны для зрения? | Да | Да | Нет | Да | Да |
| Присутствует ли отвлекающая информация? | Нет | Нет | Нет | Нет | Нет |
| Понятна ли структура приложения на интуитивном уровне? | Да | Да | Да | Да | Да |
| Смогли ли вы найти интересующую вас информацию без затруднений? | Да | Да | Да | Да | Да |
| Понятен ли смысл приложения? | Да | Да | Да | Да | Да |
| Общая оценка | 8/10 | 7/10 | 9/10 | 6/10 | 7/10 |

Исходя из средней оценки информационной системы, рассчитанной на основе общих оценок всех пользователей с учетом ответов на все поставленные вопросы, можно сделать вывод о том, что внешние характеристики графического интерфейса находятся на довольно высоком уровне. Также в информационной системе нет помех для пользователя в виде отвлекающей информации, так как все отображаемые данные находятся в рамках предметной области.

Вывод по главе: были проведены тесты путём проведения модульного тестирования, анализа покрытия кода тестами, анализа стилистики кода и анализом производительности. Также было проведено нагрузочное тестирование, в результате чего нагрузка на процессор и память не имела критичных значений.

# ЗАКЛЮЧЕНИЕ

В результате учебной практики был проведен анализ предметной области, спроектирована и разработана информационная система «ЭнергоУчет» серверная часть которой была реализована в MS SQL, а клиентская часть в MS Visual Studio.

В MS SQL была создана база данных для хранения информации о счетчиках, зданиях, квартир и договоров по плате за электроэнергию.

Клиентская часть, реализованная в MS Visual Studio, отражает все хранящиеся в базе данных таблицы. Также разработаны некоторые запросы, позволяющие редактировать и удалять существующие данные, добавлять новые и осуществлять поиск по таблицам.

Цели учебной практики были достигнуты путем проведения анализа предметной области, проектирования информационной системы и разработки программных модулей.

Это позволяет сделать вывод, что поставленные цели и задачи выполнены в полном объеме.

В дальнейшем данная информационная система может быть модернизирована путем добавления формы личного кабинета, создания версии для слабовидящих людей или улучшения и изменения дизайна.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОСТ 20886-85. Организация данных в системах обработки данных. Термины и определения [Электронный доступ]. — Режим доступа: [ГОСТ 20886-85. Организация данных в системах обработки данных. Термины и определения (rostest.info)](https://rostest.info/gost/001.001.040.035/gost-20886-85/#:~:text=%D0%93%D0%9E%D0%A1%D0%A2%2020886%2D85.%20%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%B2,%D0%BD%D0%B0%D1%83%D1%87%D0%BD%D0%BE%2D%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%) (Дата обращения: 22.05.2013 г.).
2. ГОСТ Р. 59793–2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания [Электронный доступ]. — Режим доступа: [ГОСТ Р. 59793-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания (internet-law.ru)](https://internet-law.ru/gosts/gost/77858/?ysclid=lifap277mg519765664) (Дата обращения: 30.04.2022 г.).
3. ГОСТ 19.201-78. ЕСПД. Общие положения. [Электронный доступ]. — Режим доступа: [ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению (swrit.ru)](https://www.swrit.ru/doc/espd/19.201-78.pdf?ysclid=lpi5u6naom706376064) (Дата обращения: 01.10.1981 г.).
4. Microsoft SQL Server [Электронный доступ]. — Режим доступа: [Как устроен язык SQL и почему он так востребован — Анализ данных (practicum.yandex.ru)](https://practicum.yandex.ru/blog/chto-takoe-sql/) (Дата обращения: 27.07.2022 г.).
5. Microsoft Visual Studio [Электронный доступ]. — Режим доступа: [Руководство. Создание приложения — Microsoft (microsoft.com)](https://learn.microsoft.com/ru-ru/visualstudio/ide/walkthrough-building-an-application?view=vs-2022) (Дата обращения: 02.11.2023 г.).

# ПРИЛОЖЕНИЕ А

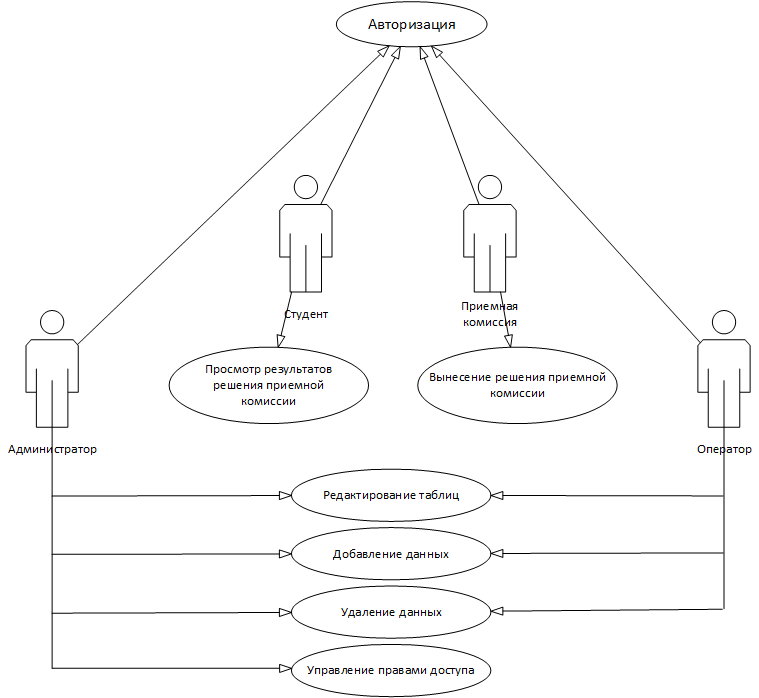


Рисунок 1 - Диаграмма вариантов использования

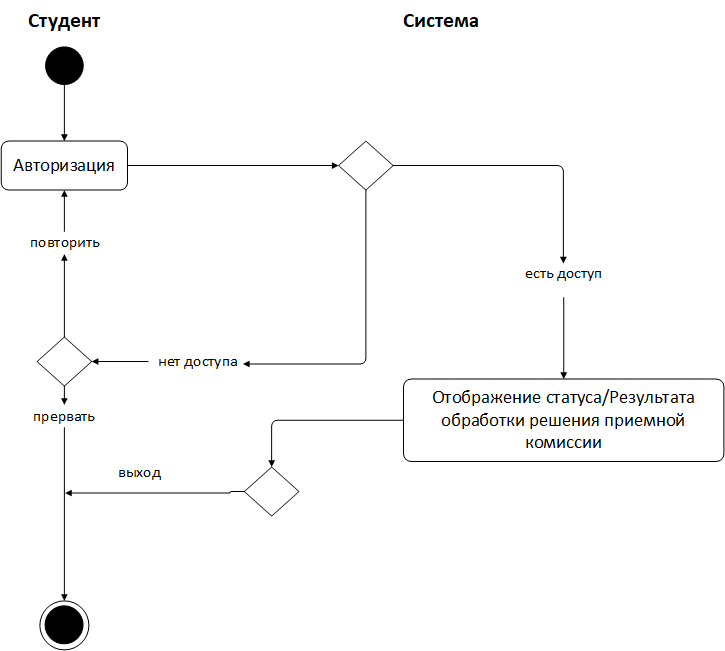


Рисунок 2 – Диаграмма деятельности «Квартиросъемщик»

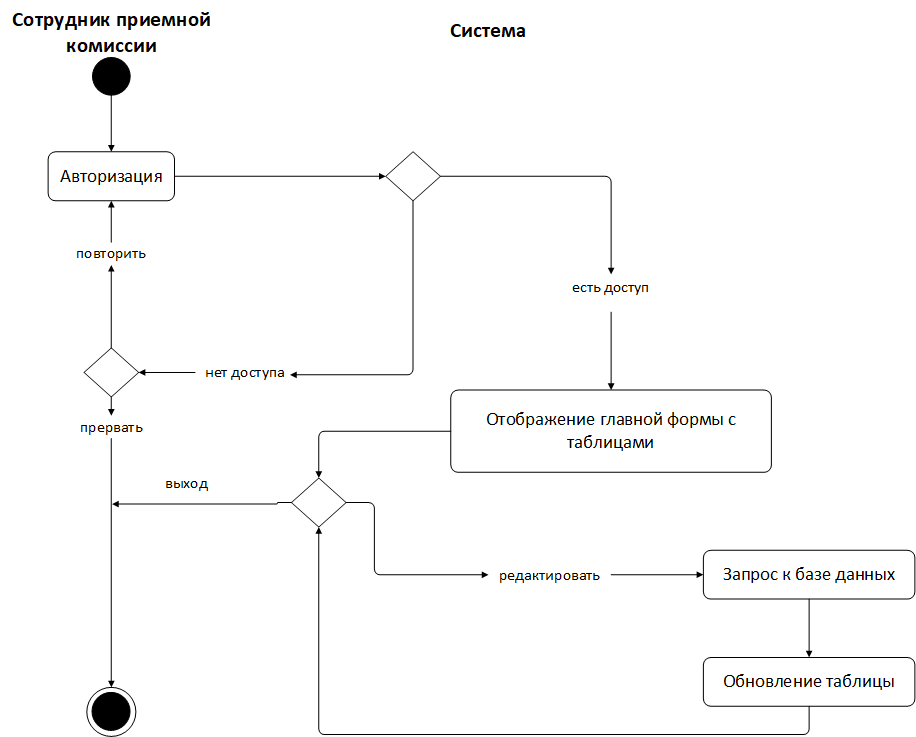


Рисунок 3 – Диаграмма деятельности «Инспектор»

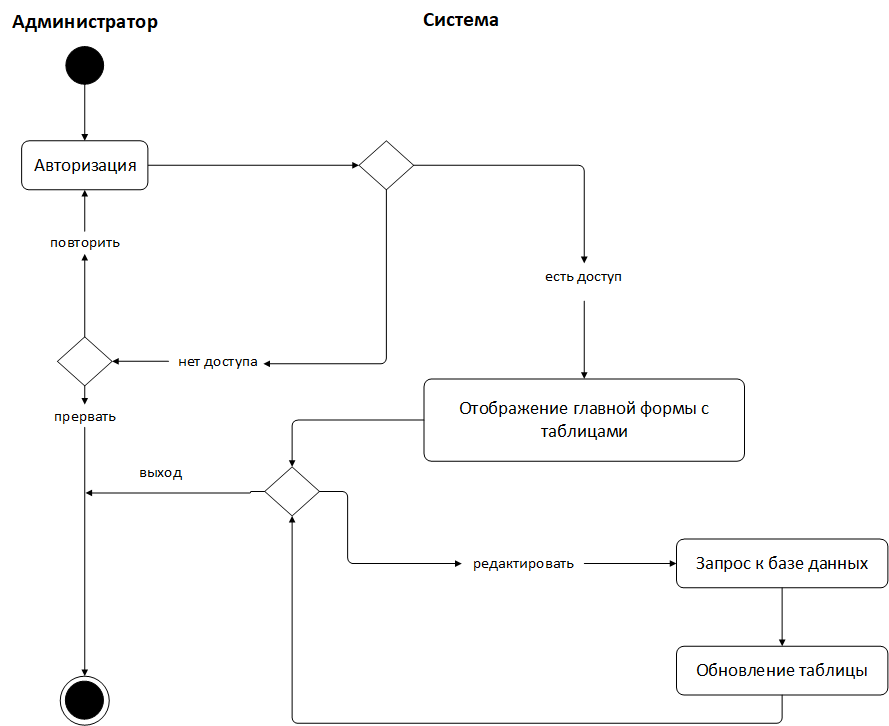


Рисунок 4– Диаграмма деятельности «Админ»

Рисунок 5 – Карта переходов

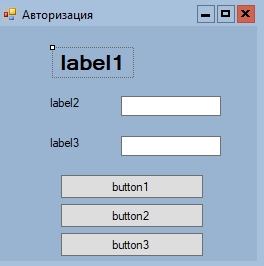


Рисунок 6 – Макет Авторизации



Рисунок 7 – Макет Регистрации

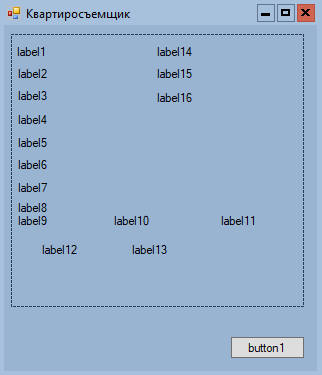


Рисунок 8 – Макет Квартиросъемщика

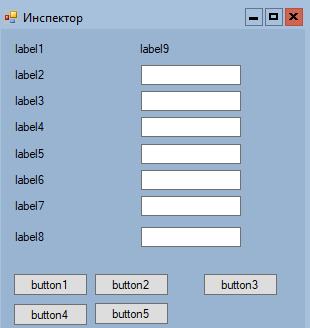


Рисунок 9 – Макет Инспектора

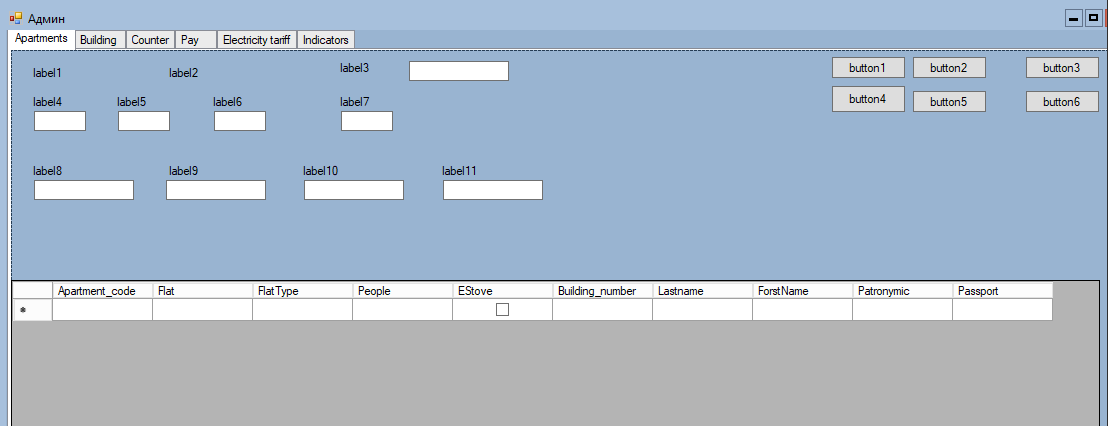


Рисунок 10 – Макет Админа



Рисунок 11 – Логическая модель данных

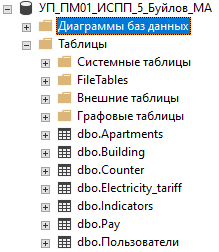


Рисунок 12 – Созданные таблицы

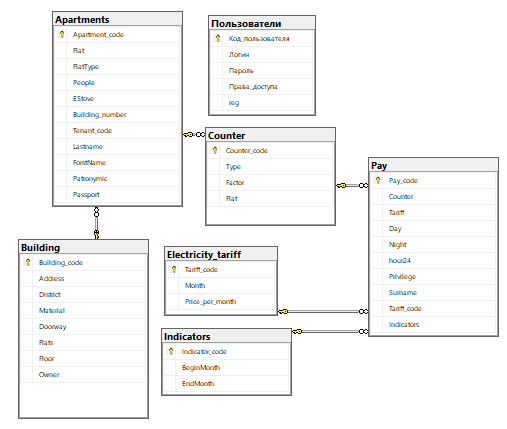


Рисунок 13 – Физическая модель базы данных

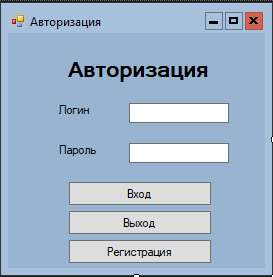


Рисунок 14 – Форма «Авторизация»

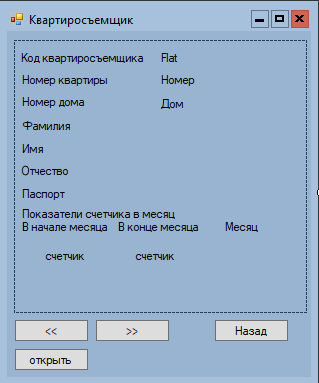


Рисунок 15 – Форма «Квартиросъемщик»

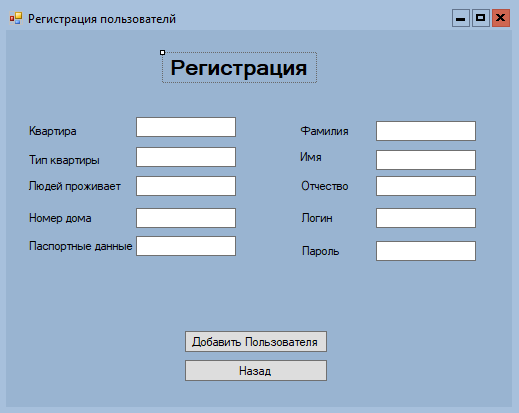


Рисунок 16 – Форма «Регистрации»

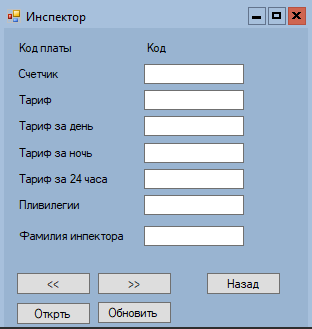


Рисунок 17 – Форма «Инспектора»

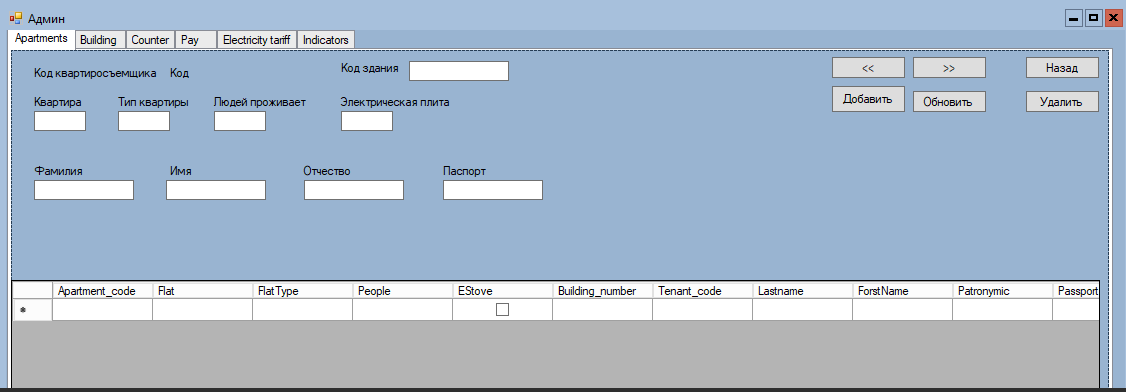


Рисунок 18 – Форма «Админа»

Рисунок 19 – Нагрузочное тестирование приложения