**КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА**

**САНКТ- ПЕТЕРБУРГСКИОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
  
«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»**

Специальность 09.02.03 «Программирование в компьютерных системах»

**УТВЕРЖДАЮ**

**Зам. директора по УМР**

**\_\_\_\_\_\_\_\_\_\_\_\_ Венедиктова О.Н.**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.**

**ДИПЛОМНЫЙ ПРОЕКТ**

Пояснительная записка

**Тема: «Разработка приложения для предприятия ЖКХ»**

Группа 463

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нормоконтроль\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.Н. Венедиктова

Консультант по экономической части \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Львова

Руководитель дипломного проекта \_\_\_\_\_\_\_\_СОГЛАСОВАНО\_\_\_\_\_\_\_ И.П. Смирнова

Выполнил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_К .Н. Григорьев

**СОДЕРЖАНИЕ**

[1. ОБЩАЯ ЧАСТЬ 6](#_Toc42787394)

[1.1. Цель разработки 6](#_Toc42787395)

[1.2. Анализ рынка подобных программных решений 6](#_Toc42787396)

[1.2.1 1C: Учет в управляющих компаниях ЖКХ, ТСЖ И ЖСК 6](#_Toc42787397)

[1.3. Анализ среды разработки 10](#_Toc42787398)

[1.4. Интегрированная среда разработки Visual Studio 10](#_Toc42787399)

[1.4.1. SQL Server Management Studio 12](#_Toc42787400)

[1.4.2. Анализ методов разработки 13](#_Toc42787401)

[1.4.3. Обзор методов решения 14](#_Toc42787402)

[1.4.4. Выбор языка разработки 14](#_Toc42787403)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 18](#_Toc42787404)

[2.1. Техническое задание 18](#_Toc42787405)

[2.2. Назначение программного продукта 18](#_Toc42787406)

[2.3. Основание для разработки 18](#_Toc42787407)

[2.4. Требования к программе 18](#_Toc42787408)

[2.5 Описание функционально-логической структуры данных 20](#_Toc42787409)

[2.5.1 Структурная схема приложения 20](#_Toc42787410)

[2.5.2 Функциональная схема приложения 21](#_Toc42787411)

[2.5.3 Диаграммы методологии IDEF0 21](#_Toc42787412)

[2.5.4 Диаграмма вариантов использования(UML) 23](#_Toc42787413)

[2.6 Логическая модель данных 23](#_Toc42787414)

[2.6.1 База данных 23](#_Toc42787415)

[2.7 Описание алгоритма 28](#_Toc42787416)

[2.7.1 Общий алгоритм работы программы 28](#_Toc42787417)

[2.7.2 Алгоритм работы отдельных модулей 30](#_Toc42787418)

[2.8 Программная реализация 33](#_Toc42787419)

[2.8.1 Пользовательский интерфейс 33](#_Toc42787420)

[3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 39](#_Toc42787421)

[3.2 Расчет трудоемкости 39](#_Toc42787422)

[3.3 Расчет расходов на заработную плату 40](#_Toc42787423)

[3.4 Расчет затрат на материалы 41](#_Toc42787424)

[3.5 Расчет затрат на электроэнергию и амортизацию оборудования 41](#_Toc42787425)

[3.6 Расчет сметы затрат 42](#_Toc42787426)

[3.7 Расчет экономического эффекта 42](#_Toc42787427)

[ЗАКЛЮЧЕНИЕ 43](#_Toc42787428)

[ЛИТЕРАТУРА 44](#_Toc42787429)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 44](#_Toc42787430)

**ВВЕДЕНИЕ**

В современном информационном обществе производится и хранится информация в огромных количествах. В связи с тем, что хранимую информацию необходимо структурировать, начали появляться программы для работы с базой данных. Эти программы были призваны упорядочить данные таблиц, которые объединены между собой в базу данных.

Базы данных значительно изменились с момента их появления, и сейчас значительно облегчает работу с ними специализированное программное обеспечение – система управления базой данных (СУБД). В каждой компании используют свою базу данных, для упрощения своих задач. ЖКХ одна из таких компаний, которая хранит большую базу данных о своих клиентах. В ЖКХ есть проблемы с неуплатой счётов своих клиентов.

В итоге становится заметна проблема – как быстро найти нужного клиента, и отследить его оплату за всё время пользования услугами, что без возможности сортировки сделать сложно, и для этого необходимы отдельные программы.

Данный диплом подтверждает освоение следующих компетенций профессиональных модулей:

1. ПМ 01. Разработка программных модулей программного обеспечения для компьютерных систем:

* ПК 1.1. Выполнять разработку спецификаций отдельных компонент;
* ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля;
* ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств;
* ПК 1.4. Выполнять тестирование программных модулей;
* ПК 1.5. Осуществлять оптимизацию программного кода модуля;
* ПК 1.6. Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

1. ПМ 02. Разработка и администрирование баз данных:

* ПК 2.1. Разрабатывать объекты базы данных;
* ПК 2.2. Реализовывать базу данных в конкретной системе управления базами данных;
* ПК 2.3. Решать вопросы администрирования базы данных;
* ПК 2.4. Реализовывать методы и технологии защиты информации в базах данных.

1. ПМ 03. Участие в интеграции программных модулей:

* ПК 3.1. Анализировать проектную и техническую документацию на уровне взаимодействия компонент программного обеспечения;
* ПК 3.2. Выполнить интеграцию модулей в программную среду;
* ПК 3.3. Выполнить отладку программного продукта с использованием специализированных программных средств;
* ПК 3.4. Осуществлять разработку тестовых наборов и тестовых сценариев;
* ПК 3.5. Производить инспектирование компонент программного продукта на предмет соответствия стандартам кодирования;
* ПК 3.6. Разрабатывать техническую документацию.

# ОБЩАЯ ЧАСТЬ

В этом разделе описаны средства разработки, использованные при создании продукта, указаны цели разработки дипломного проекта, проанализированы различные средства разработки.

## Цель разработки

Целью разработки данного дипломного проекта является создание программного продукта, на который возлагается задача отображения и обработки информации о сумме выплат и задолженности за услуги ЖКХ пользователей.

## Анализ рынка подобных программных решений

Примерами аналогов программного продукта являются далее рассмотренные программные продукты.

# 1C: Учет в управляющих компаниях ЖКХ, ТСЖ И ЖСК

Сведения о продукте:

1С: Управляющая компания ЖКХ. Модуль для ERP и КА2 позволяет решать управленческие и учетные задачи управляющих организаций в сфере ЖКХ. Модуль предназначен для автоматизации процессов оказания услуг ЖКХ. Является дополнением (модулем) к типовым конфигурациям ERP и КА2.

Для кого:

Для управляющих компаний и организаций сферы жилищно-коммунального хозяйства.

Пользователи:

* директор;
* главный инженер;
* бухгалтеры;
* экономисты;
* администраторы;
* сотрудники аварийно-диспетчерской службы;
* сотрудники паспортного стола.

Функциональные возможности: «Подомовой учет затрат»:

* настройка способов распределения затрат;
* закрепление за объектом учета ответственных лиц/подразделений;
* распределение затрат на услуги и материалы по объектам учета;
* распределение затрат по зданиям, закрепленным за определенным мастером/подразделением.

Интеграция с Государственной информационной системой (ГИС) ЖКХ:

* настройка механизмов интеграции с ГИС ЖКХ;
* передача данных в ГИС ЖКХ;
* получение данных из ГИС ЖКХ.

Учет жилого и нежилого фонда:

* хранение различных характеристик зданий;
* прикрепление ответственных мастеров по обслуживанию зданий;
* печать энергопаспорта. Учет дополнительного оборудования зданий;
* задание дополнительных характеристик зданий;
* учет типов помещений. Хранение типов собственности;
* установка типов и размеров площадей;
* дополнительные характеристики помещений;
* хранение многочисленной информации по паркингам и парковочным местам;
* управление парковочными местами;
* привязка автомобилей к парковочным местам.

Учет коммунальной инфраструктуры:

* учет котельных, водозаборов, скважин, трубопроводов, электросетей, дорог, земельных участков и прочих объектов коммунальной инфраструктуры;
* прикрепление объекта к сотруднику или подразделению организации, ответственному за работы по объекту;
* обслуживающие мастера, привязка к основным средствам.
* дополнительные характеристики объектов, отражение взаимосвязи объекта со зданиями, сооружениями;
* печать карточки объекта.

Учёт жильцов:

* учет физических и юридических лиц;
* общие собрания собственников;
* учет сведений о жильцах;
* подготовка тем голосования;
* составление и печать реестров участников голосования;
* автоматическое формирование бюллетеней голосования;
* подведение итогов голосования;
* печать результатов голосования;
* ввод данных об проживающих.

Паспортный учет:

* регистрация граждан по месту жительства;
* прибытие граждан со старого места жительства (постановка на учет);
* выбытие граждан на новое место жительства (снятие с учета);
* регистрация изменения паспортных данных;
* замена документа, удостоверяющего личность;
* изменение ФИО граждан;
* регистрация причин изменения паспортных данных;
* формирование справок и списков паспортного стола.

Техническое обслуживание, ремонт и благоустройство:

* аварийно-диспетчерская служба;
* регистрация заявок/жалоб жильцов;
* распределение заявок между ответственными мастерами;
* контроль выполнения заявок;
* формирование наряд-заказов;
* списание материалов по видам работам;
* сотрудники паспортного стола;
* планирование работ;
* формирование отчетности.

Отчётность:

* список объектов учета;
* реестры лицевых счетов/проживающих;
* отчет об изменении численности проживающих;
* списки парковочных мест и их владельцев;
* справка о задолженности;
* справка на субсидию;
* доходы и расходы по зданиям;
* отчет по затратам на содержание и ремонт зданий.

Удобная и функциональная программа для автоматизации предприятий сферы ЖКХ: от управления объектами инфраструктуры, начисления квартплаты до бухгалтерского и налогового учета.

Стоимость данного продукта начинается от 1150 рублей на одного сотрудника, цена поднимается до 1 776 000 рублей, где клиентская лицензия на 500 сотрудников (рисунок 1).

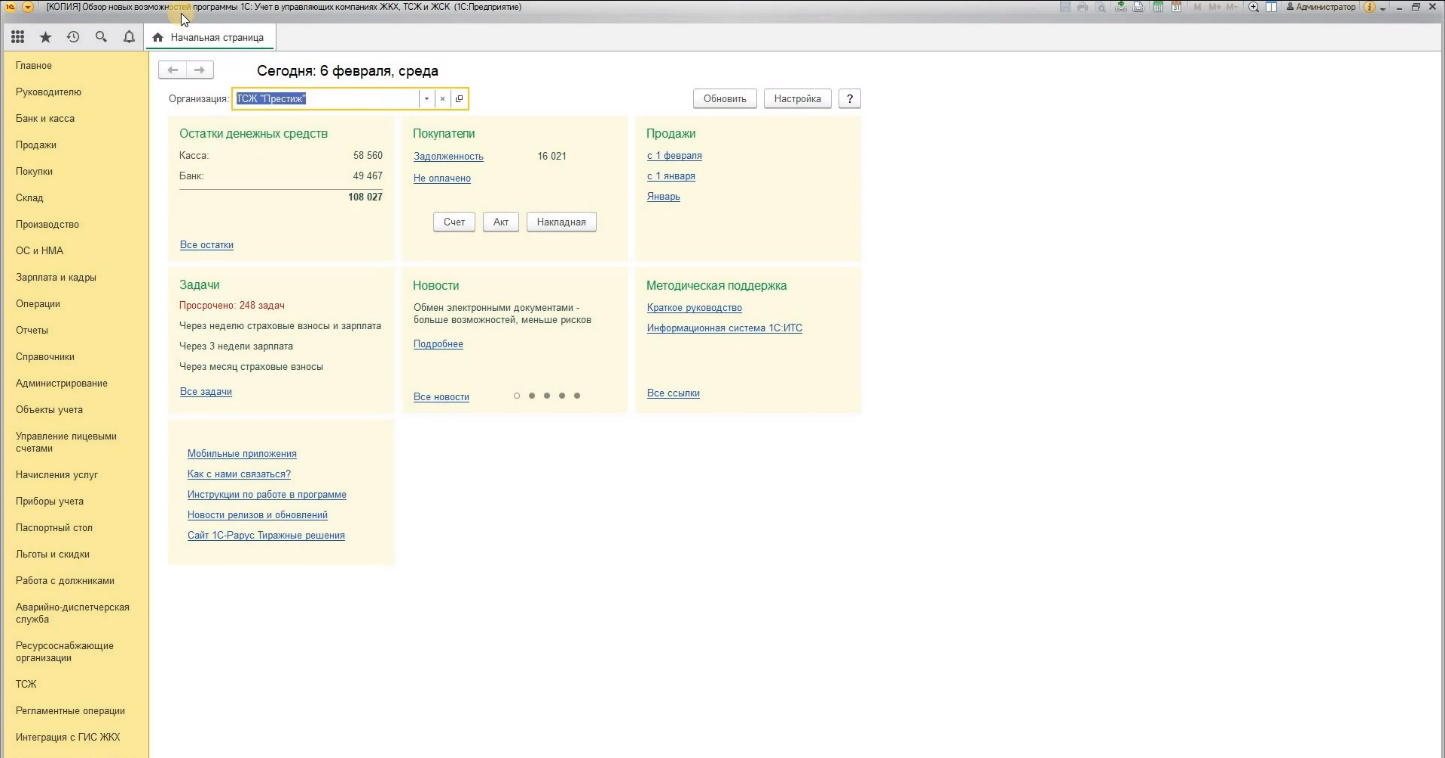


Рисунок 1 – Интерфейс 1C: Учет в управляющих компаниях ЖКХ, ТСЖ И ЖСК

Плюсы:

* большой функционал.

Минусы:

* сложность в освоении;
* высокая цена.

## Анализ среды разработки

Для создания приложения необходимо использование среды разработки с наиболее удобным инструментарием для создания кода и пользовательского интерфейса.

## Интегрированная среда разработки Visual Studio

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки (рисунок 2).



Рисунок 2 – Логотип Visual Studio

Плюсы:

* простота создания приложений с графическим интерфейсом;
* простой интерфейс;
* технология автодополнения IntelliSense;
* удобная отладка программ.

Минусы:

* поддержка ОС только семейства Windows;
* сложность в освоении.

С помощью Visual Studio можно разрабатывать:

* классические приложения для компьютера под управлением операционной системы Windows;
* мобильные приложения (Windows, iOS, Android);
* Web-приложения;
* облачные приложения;
* различные расширения для Office, SharePoint, а также создание собственных расширений для Visual Studio;
* игры;
* базы данных SQL Server и SQL Azure.

В Visual Studio можно использовать следующие технологии и языки программирования: .NET, Node.js, C, C#, C++, Python, Visual Basic, F#, JavaScript.

Системные требования Visual Studio 2019:

Среду разработки Visual Studio 2019 можно установить и работать в ней на следующих операционных системах (перечислены официально поддерживаемые версии):

* Windows 7 с Service Pack 1;
* Windows 8.1 (с обновлением 2919355);
* Windows 10 (1703 и выше).

Минимальные требования к оборудованию:

* процессор с тактовой частотой не ниже 1,8 ГГц. Рекомендуется использовать как минимум двухъядерный процессор;
* 2 Гб оперативной памяти, рекомендуется 8 Гб (если устанавливать на виртуальную машину, то минимум 2,5 Гб);
* свободного места на жестком диске от 800 Мб до 210 Гб, в зависимости от установленных компонентов;
* видеоадаптер с минимальным разрешением 1280 на 720 пикселей (для оптимальной работы Visual Studio рекомендуется разрешение 1366 на 768 пикселей и более высокое).

Дополнительные важные моменты:

* для установки Visual Studio 2019 требуются права администратора;
* для работы Visual Studio 2019 требуется платформа .NET Framework 4.7.2, она будет установлена во время установки среды;
* варианты «Основные серверные компоненты» и «Минимальный серверный интерфейс» не поддерживаются при запуске на Windows Server;
* запуск Visual Studio 2019 (Professional, Community и Enterprise) в контейнерах Windows не поддерживается;
* для интеграции Team Foundation Server 2019 с Office требуется Office 2016, Office 2013 или Office 2010;
* для установки компоненты «Разработка мобильных приложений на C++, JavaScript или .NET» в ОС Windows 7 требуется PowerShell 3.0 или более поздняя версия;
* Visual Studio Community 2019 требуется активировать в течение 30 дней после установки, для этого необходимо создать учетную запись.

### SQL Server Management Studio

Среда SQL Server Management Studio (SSMS) – это графический набор средств, для разработки сценариев на T-SQL и управления всеми компонентами SQL Server. Management Studio является основным инструментом любого разработчика или администратора MS SQL сервера (рисунок 3).



Рисунок 3 – SQL Server Management Studio

Плюсы:

* СУБД масштабируется, поэтому работать с ней можно на портативных ПК или мощной мультипроцессорной технике. Процессор может одновременно обрабатывать большой объем запросов;
* реализован поиск по фразам, тексту, словам, можно создавать ключевые индексы;
* eсть репликации через интернет, предусмотрена синхронизация;
* есть полноценный веб-ассистент для форматирования страниц;
* запросы можно формулировать на английском языке, без программирования.

Минусы:

* сложность в освоении.

### Анализ методов разработки

Для разработки приложения была выбрана среда разработки Visual Studio Community 2019 – бесплатная, полнофункциональная и расширяемая интегрированная среда разработки для создания современных приложений для UWP, Android и iOS, а также веб-приложений и облачных служб.

Возможности Visual Studio Community:

* помощь в написании кода, навигации и исправления ошибок;
* сворачивание кода;

### Обзор методов решения

Во время разработки рассматривалась возможность создания приложения для платформ: ОС Windows. Windows – данная платформа содержит наибольшее число потенциальных пользователей.

### Выбор языка разработки

В данном разделе пояснительной записки к дипломному проекту рассматриваются языки, подходящие для реализации данного дипломного проекта.

* + - 1. **Расширяемый язык разметки XAML**

XAML (eXtensible Application Markup Language) – язык разметки, используемый для инициализации объектов в технологиях на платформе .NET. Применительно к WPF (Windows Presentation Foundation), данный язык используется, прежде всего, для создания пользовательского интерфейса декларативным путем. Хотя функциональность XAML только графическими интерфейсами не ограничивается: данный язык также используется в технологиях WCF (Windows Communication Foundation) и WF (Windows Forms), где он никак не связан с графическим интерфейсом. То есть его область шире. Применительно к WPF, который позволяет создавать декларативным путем интерфейс, наподобие HTML (HyperText Markup Language) в веб-программировании.

Плюсы:

* возможность отделить графический интерфейс от логики приложения, благодаря чему над разными частями приложения могут относительно автономно работать разные специалисты: над интерфейсом – дизайнеры, над кодом логики – программисты;
* объемная документация;
* компактность, понятность, код на XAML относительно легко поддерживать.

Минус:

* ориентированность, в основном, на .NET.
  + - 1. **Язык программирования C#**

C# – объектно-ориентированный язык программирования разработанный Microsoft для платформы. NET и многое перенявший у Java и С++. C# содержит в себе полиморфизм, наследование, статическую типизацию, итераторы, перегрузку операторов и это лишь часть его возможностей. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

Плюсы:

* кроссплатформенность;
* быстрое развитие и поддержка Microsoft;
* средний порог вхождения.

Минусы:

* C# очень легко дизассемблируется.
  + - 1. **Элементы выбранного языка**

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к Java и C++.

Пример 1 – «Сравнение двух чисел » на C#:

using System;

namespace HelloApp

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите первое число: ");

int num1 = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите второе число: ");

int num2 = Convert.ToInt32(Console.ReadLine());

if (num1 > num2)

{

Console.WriteLine("Первое число больше второго");

}

else if (num1 < num2)

{

Console.WriteLine("Первое число меньше второго");

}

else

{

Console.WriteLine("Оба числа равны");

}

Console.ReadKey();

}

}

}

Пример 2 – «Сортировка пузырьком» на C#:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HelloApp

{

class Program

{

static int[] BubbleSort(int[] mas)

{

int temp;

for (int i = 0; i < mas.Length; i++)

{

for (int j = i + 1; j < mas.Length; j++)

{

if (mas[i] > mas[j])

{

temp = mas[i];

mas[i] = mas[j];

mas[j] = temp;

}

}

}

return mas;

}

static void Main(string[] args)

{

Console.WriteLine("Сколько чисел будем сортировать?");

int N = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите числа для сортировки:");

int[] mas = new int[N];

for (int i = 0; i < mas.Length; i++)

{

mas[i] = Convert.ToInt32(Console.ReadLine());

}

BubbleSort(mas);

Console.WriteLine("После сортировки:");

for (int i = 0; i < mas.Length; i++)

{

Console.WriteLine(mas[i]);

}

Console.ReadLine();

}

}

}

# СПЕЦИАЛЬНАЯ ЧАСТЬ

В этом разделе описаны все методы, алгоритмы и процессы разработки дипломного проекта, а также техническое задание и инструкция пользователя.

## Техническое задание

В текущем разделе представлено техническое задание, содержащее постановку задачи, назначение программного продукта, основание для разработки, а также требования к конечному программному продукту.

Разработать приложение для ОС Windows 7 и новее, выполняющее следующие функции:

* + возможность просмотра информации о пользователях;
  + возможность подать заявку на отключение пользователя;
  + возможность подать заявку на перерасчёт пользователя;
  + иметь простотой и интуитивно понятный интерфейс.

## Назначение программного продукта

Данный программный продукт должен предоставлять администратору удобный интерфейс для доступа к информации о задолженностях пользователей услуг ЖКХ.

## Основание для разработки

Основанием для разработки программы является задание на дипломное проектирование и задание на преддипломную практику.

## Требования к программе

В этом разделе описаны аппаратные требования, для корректного функционирования программы.

* + 1. **Требования к функциональным характеристикам**

Программа должно выполнять следующие функции:

* + возможность просмотра информации о пользователях;
  + возможность подать заявку на отключение пользователя;
  + возможность подать заявку на перерасчёт пользователя;
  + простота и интуитивность интерфейса.

Входные данные в приложении – это выбранные данные из таблиц: названия улиц, дома, квартира, год, месяц.

Выходные данные программы – это отображаемая информация на экране, текстовые данные, отправка сообщения по кнопке «Отключение», «Перерасчёт».

* + 1. **Требования к аппаратным и программным средствам**

Требования к аппаратным средствам представлены ниже в таблице 1.

Таблица 1 – Системные требования для установки программы

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п |  | Минимальные | Рекомендуемые |
| 1 | Операционная система (ОС) | Windows | Windows |
| 2 | Версия ОС | 7 | 10 |
| 3 | Память на диске | 200МБ | 200МБ |
| 4 | Оперативная память | 1ГБ | 2ГБ+ |
| 5 | Процессор | Intel Core i3 540 | Intel Core i3 8100 |

* + 1. **Требования к надежности**

Данный раздел определяет требуемый уровень надежности разрабатываемого приложения.

* + - 1. **Требования к надежному функционированию системы**

Надежное (устойчивое) бесперебойное функционирование программы должно быть обеспечено выполнением следующих организационных мер: бесперебойное питание компьютера.

Также необходимо позаботиться об ошибках, возникающих во время работы программы, обрабатывать их и выводить пользователю варианты решения вплоть до переустановки приложения.

* + - 1. **Отказы из-за некорректных действий пользователей системы**

Для того, чтобы избежать критических ошибок, необходимо проверять все данные, которые выбирает пользователь. При неправильных входных данных, необходимо оповестить пользователя и запросить ввод подходящих данных.

* + 1. **Защита и сохранность данных**

Личные данные пользователя, используемы в приложении, должны быть обеспечены надлежащим уровнем защиты информации, в соответствии с законом о защите персональной информации.

Для сохранности данных приложения необходима возможность создания резервной копии необходимых файлов с возможностью их восстановления.

## Описание функционально-логической структуры данных

В этой части пояснительной записки расположены разработанные схемы и диаграммы, описывающие алгоритмы разрабатываемого комплекса.

### Структурная схема приложения

Структурная схема – это совокупность элементарных звеньев объекта и связей между ними, один из видов графической модели. Под элементарным звеном понимают часть объекта, системы управления и т. д., которая реализует элементарную функцию.

Структурная схема приложения представлена ниже на рисунке 5.

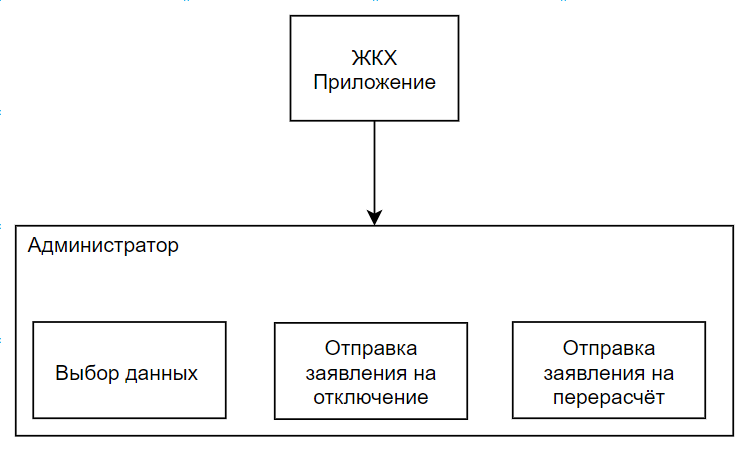


Рисунок 5 – Структурная схема приложения

### Функциональная схема приложения

Функциональная схема – документ, разъясняющий процессы, протекающие в отдельных функциональных цепях программы. Функциональная схема является поясняющим материалом отдельных видов процессов, протекающих в целостных функциональных блоках и цепях устройства.

Функциональная схема приложения представлена на рисунке 6.

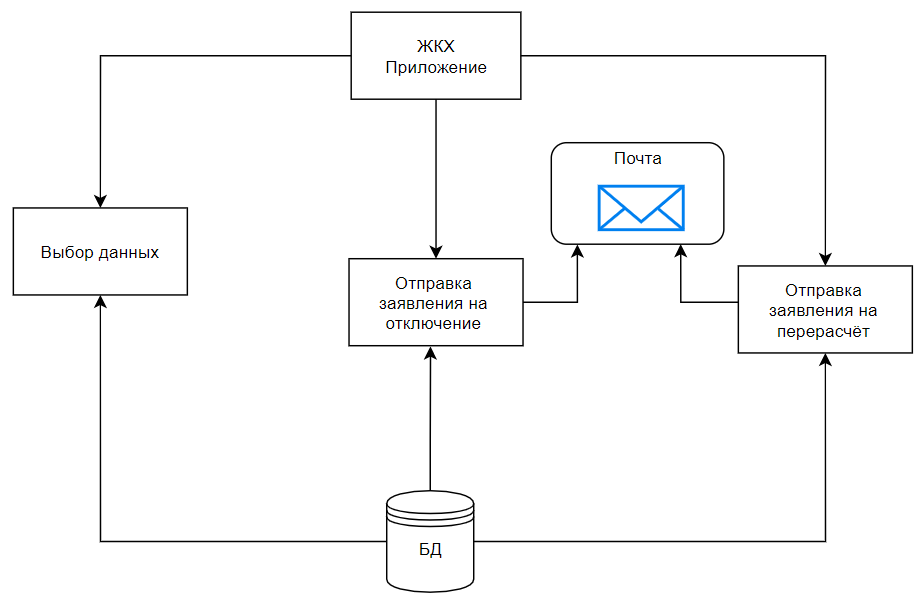


Рисунок 6 – Функциональная схема приложения

### Диаграммы методологии IDEF0

В основе IDEF0 (Integrated Definition Function Modeling) методологии лежит понятие блока, который отображает некоторую бизнес-функцию. Каждая сторона блока имеет свою роль: левая имеет значение "входа", правая - "выхода", верхняя - "управления", нижняя - "механизма".

Существует два вида таких диаграмм: контекстная и детализирующая.

Для начала посмотрим на контекстную диаграмму процесса «Прокладывание маршрута» (рисунок 7).

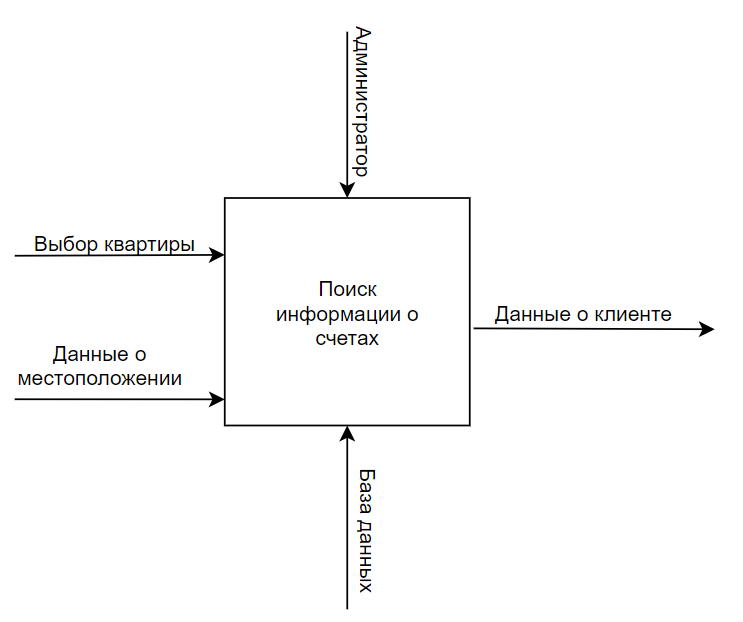


Рисунок 7 – Контекстная диаграмма приложения

Здесь представлена диаграмма детализации первого уровня этого процесса, которая отображает процессы, происходящие внутри контекстной диаграммы. Детализирующая диаграмма представлена на рисунке 8.

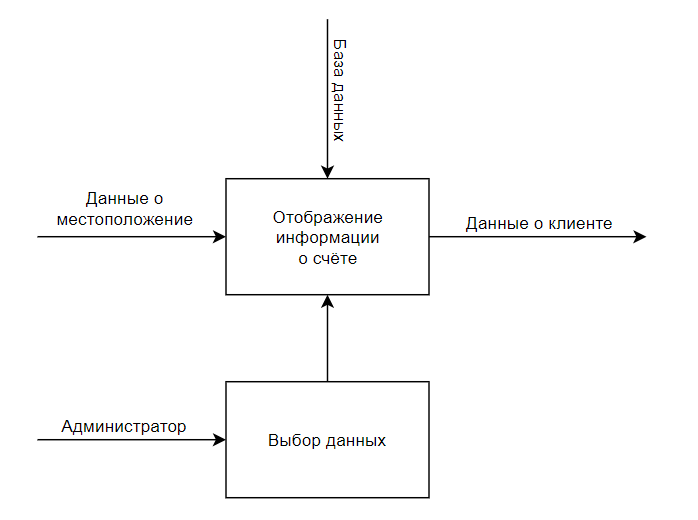
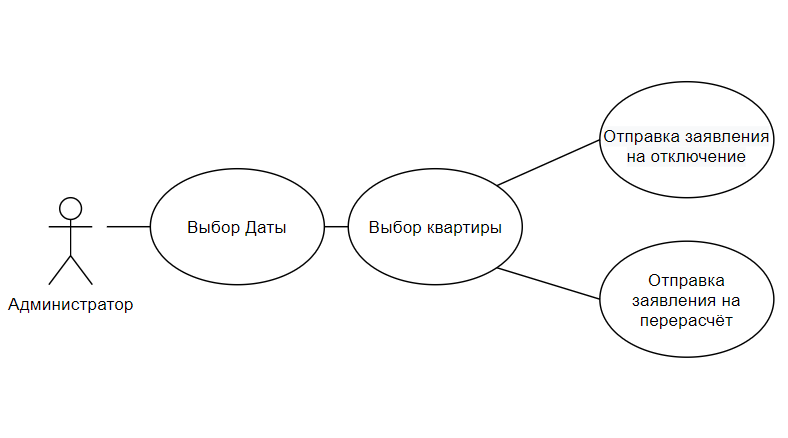


Рисунок 8 – Детализирующая диаграмма первого уровня

### Диаграмма вариантов использования(UML)

Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, все возможности приложения во время работы. Она является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки. У данной диаграммы будет только один взаимодействующий с ней актер – пользователь.

Построенная диаграмма вариантов использования приложения представлена на рисунке 9.

  
Рисунок 9 – Диаграмма вариантов использования

## Логическая модель данных

Для понимания алгоритма обработки входных и выходных данных, требуется ознакомиться со структурой этих данных в приложении. Именно в разделе «Логическая модель данных» подробно описана структура данных приложения.

### База данных

В данном подразделе будет представлена информация о базе данных, которая была создана в процессе разработки программы, а также описываются таблицы необходимые для работы приложения. Ниже приведена ER-диаграмма (рисунок 10).

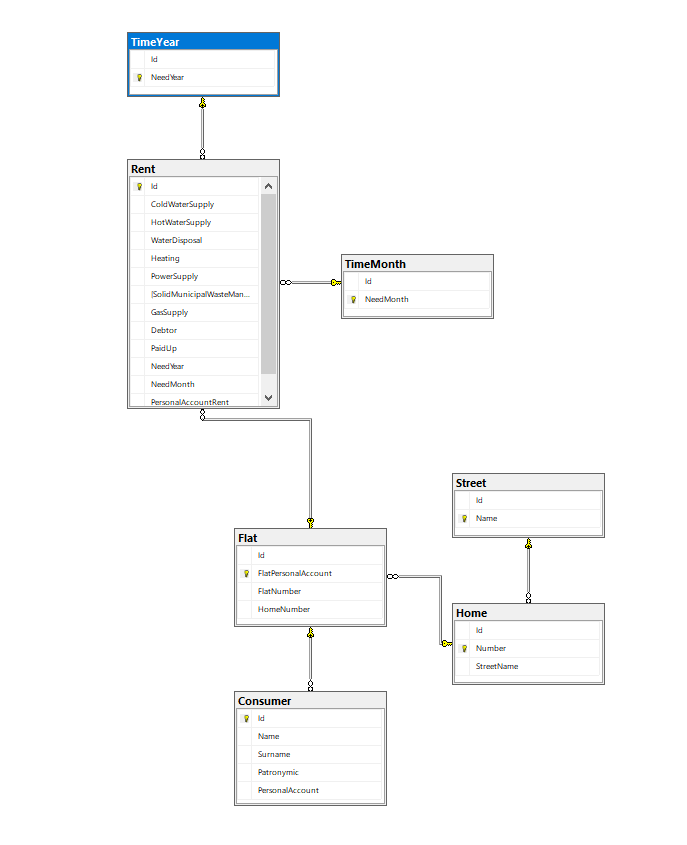


Рисунок 10 – ER-диаграмма

Для работы данного приложения достаточно одной локальной базы данных SQL, которую можно увидеть на рисунке 11.

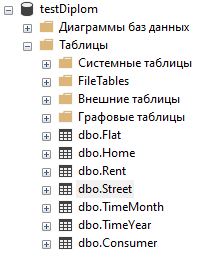


Рисунок 11 – Таблицы БД «testDiplom»

Таблица «Flat». Данная таблица требует предварительного заполнения (рисунок 12).

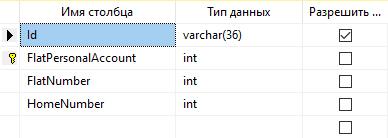


Рисунок 12 – Столбцы в таблице «Flat»

В данной таблице присутствуют столбцы:

* Id – предназначен для однозначной идентификации маркера;
* FlatPersonalAccount – предназначен для хранения номера счета квартиры;
* FlatNumber – предназначен для хранения номера квартиры;
* HomeNumber – предназначен для хранения в каком доме квартира.

Таблица «Home». Данная таблица требует предварительного заполнения (рисунок 13).

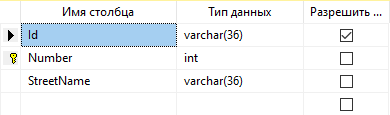


Рисунок 13 – Столбцы в таблице «Home»

В данной таблице присутствуют столбцы:

* id – предназначен для однозначной идентификации маркера;
* Number – предназначен для хранение номера дома;
* StreetName – предназначен для хранения на какой улице дом.

Таблица «Rent». Данная таблица требует предварительного заполнения (рисунок 14).

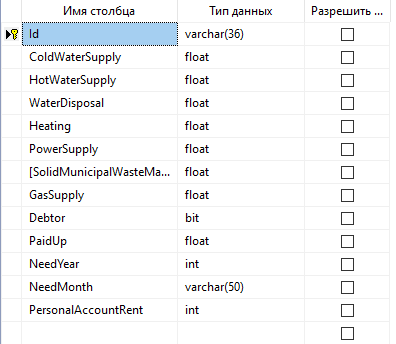


Рисунок 14 – Столбцы в таблице «Rent»

В данной таблице присутствуют столбцы:

* id – предназначен для однозначной идентификации маркера;
* ColdWaterSupply – предназначен для счёта по услуге Холодное водоснабжение;
* HotWaterSupply – предназначен для счёта по услуге Горячее водоснабжение;
* WaterDisposal – предназначен для счёта по услуге Водоотведение;
* Heating – предназначен для счёта по услуге Отопление;
* PowerSupply – предназначен для счёта по услуге Электроснабжение.
* SolidMunicipalWasteManagement(MSW) – предназначен для счёта по услуге Обращение с твердыми коммунальными отходами (ТКО);
* GasSupply – предназначен для счёта по услуге Газоснабжение;
* NeedYear – предназначен для хранения года в котором был счёт;
* NeedMonth – предназначен для хранения месяца в котором был счёт;
* PersonalAccountRent – предназначен для хранения номера счёта.

Таблица «Street». Данная таблица требует предварительного заполнения (рисунок 15).

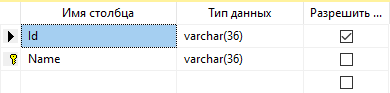


Рисунок 15 – Столбцы в таблице «Street»

В данной таблице присутствуют столбцы:

* id – предназначен для однозначной идентификации маркера;
* Name – предназначен для определения просмотрен фильм в списке или нет.

Таблица «TimeMonth». Данная таблица требует предварительного заполнения (рисунок 16).

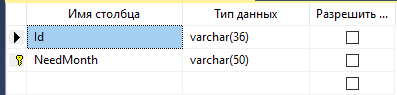


Рисунок 16 – Столбцы в таблице «TimeMonth»

В данной таблице присутствуют столбцы:

* id – предназначен для однозначной идентификации маркера;
* NeedMonth – предназначен для хранения всех месяцев.

Таблица «TimeYear». Данная таблица требует предварительного заполнения (рисунок 17).

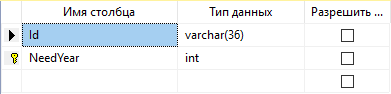


Рисунок 17 – Столбцы в таблице «TimeYear»

В данной таблице присутствуют столбцы:

* id – предназначен для однозначной идентификации маркера;
* NeedYear – предназначен для хранения чисел нужного года.

Таблица «Consumer». Данная таблица требует предварительного заполнения (рисунок 18).

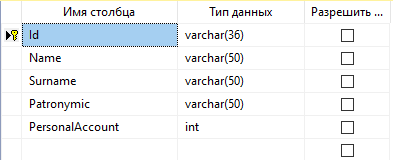


Рисунок 18 – Столбцы в таблице «Consumer»

В данной таблице присутствуют столбцы:

* id – предназначен для однозначной идентификации маркера;
* Name – предназначен для хранения Имя клиента;
* Surname – предназначен для хранения Фамилии клиента;
* Patronymic – предназначен для хранения Отчества;
* PersonalAccount – предназначен для хранения персонального счёта.

## Описание алгоритма

Этот раздел описывает алгоритмы приложения и алгоритмы для работы отдельных модулей программы.

## Общий алгоритм работы программы

При первом открытии приложения все, что увидит администратор – это три таблицы в левой части программы «Все улицы», «Год», «Месяц» и браузер в правой, поэтому первое, что ему требуется это выбрать дату.

После выбора даты администратору надо выбрать улиц, дом, квартиру. После выбора квартиры в таблице «данные о потребителе» должна появиться информация о клиенте, проживающем в этой квартире, которую администратор сможет отправить на запрос по отключению или перерасчёту.

Алгоритм работы программы описан в виде блок-схемы.

Блок-схема – распространенный тип схем, описывающий алгоритмы или процессы, изображая шаги в виде блоков различной формы, соединенных между собой стрелками (рисунок 19).

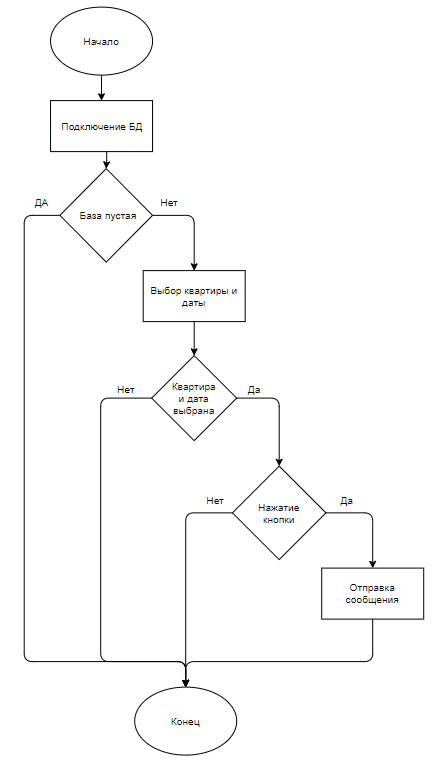


Рисунок 19 – Алгоритм работы программы

## Алгоритм работы отдельных модулей

Рассмотрим части приведённого выше алгоритма подробнее.

* + - 1. **Выбор квартиры и даты**

В модуле выбора квартиры и даты, идёт запрос в базу данных по выбранным данным. И после получения информации из базы данных записывает их в таблицу данные о потребителе (рисунок 20).

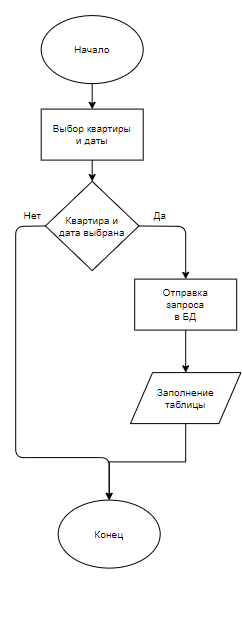


Рисунок 20 – Выбор квартиры и даты

* + - 1. **Нажатие кнопки «Запрос на отключение» или «Запрос на перерасчёт»**

В модуле нажатие кнопки считываются данные из таблицы в переменную, которая впоследствии используется в функции на отправление сообщения на почту (рисунок 21).

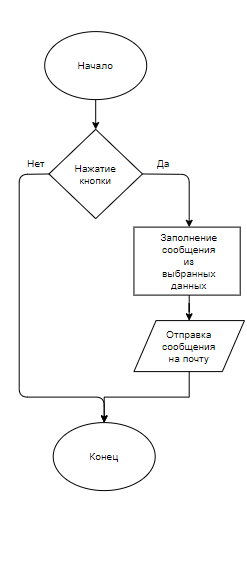


Рисунок 21 - Нажатие кнопки

## Программная реализация

В этом разделе пояснительной записки к дипломному проекту рассматривается программная реализация описанных ранее алгоритмов, а также проводится тестирование на соответствие заявленным требованиям отказоустойчивости приложения.

### Пользовательский интерфейс

Рассмотрим данное приложение с точки зрения пользователя.

При первом запуске приложения пользователя встречает 3 списка (рисунок 22):

* все улицы;
* год;
* месяц.

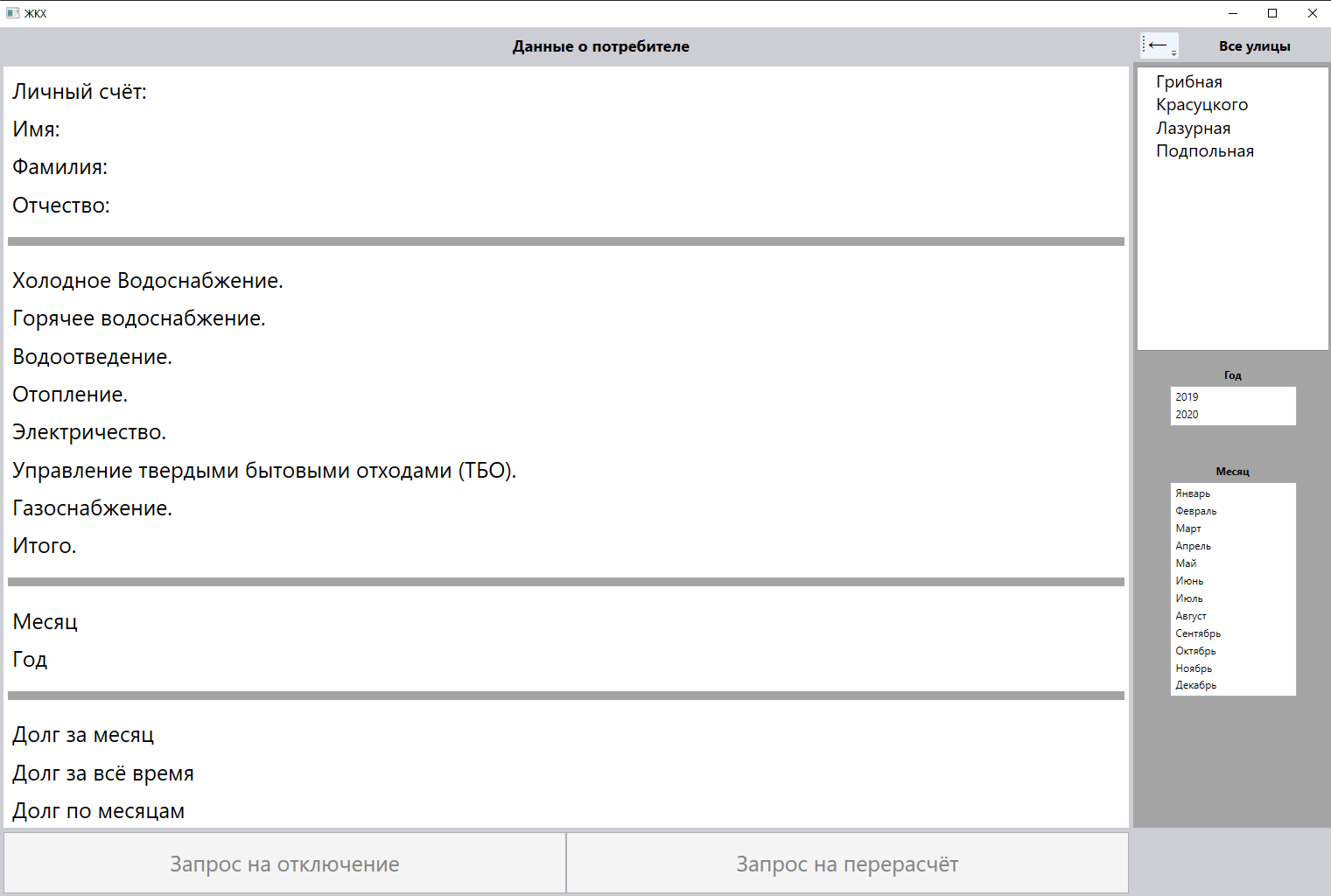


Рисунок 22 – Окно приложения

При выборе улицы, список выдаст все дома находящиеся на этой улице, для примера была выбрана улица Лазурная (рисунок 23).

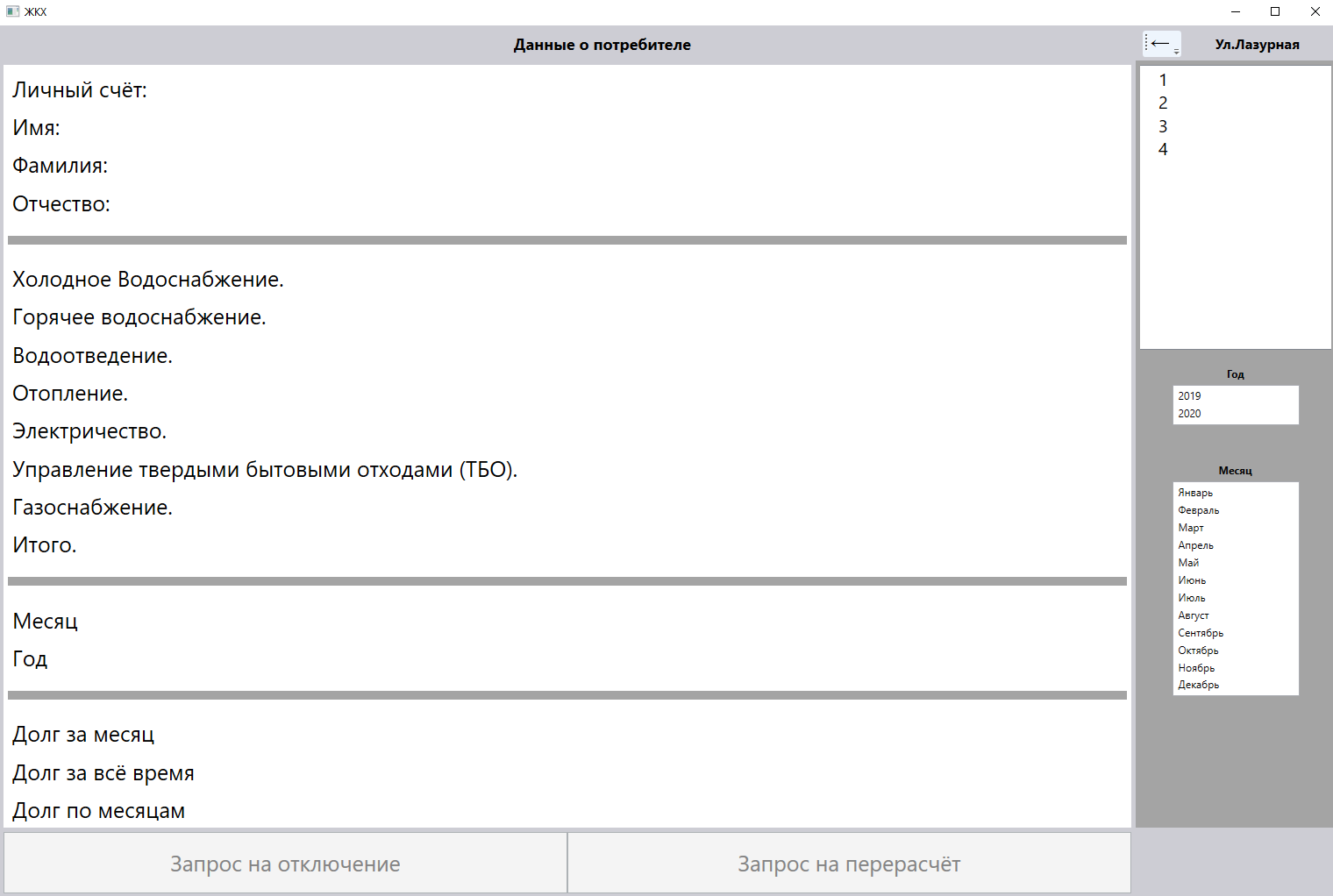


Рисунок 23 – Окно после выбора улицы

При выборе дома, список выдаст все квартиры находящиеся в этом доме, для примера был выбран дом 1 (рисунок 24).

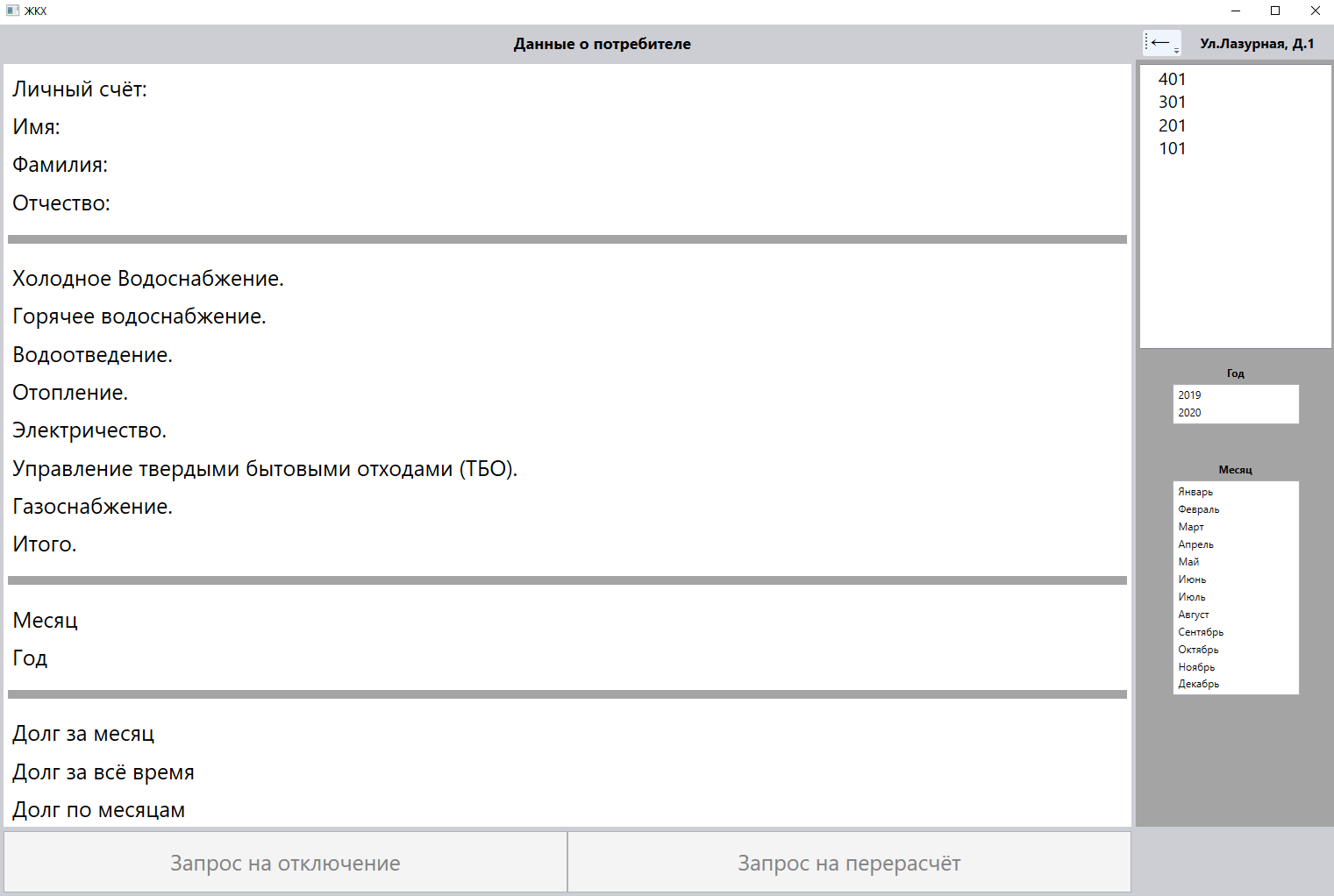


Рисунок 24 – Окно после выбора дома

При выборе квартиры приложение выдаст ошибку, сообщающую о том, что нужно выбрать год и месяц (рисунок 25).

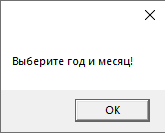


Рисунок 25 – Сообщение при выборе квартиры

При выборе года и месяца, для примера был выбран 2019 год и месяц июнь (рисунок 26).

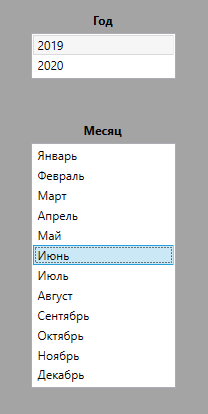


Рисунок 26 – Списки Год, Месяц

Таблица данных о потребителе, заполнится значениями из БД (рисунок 27).

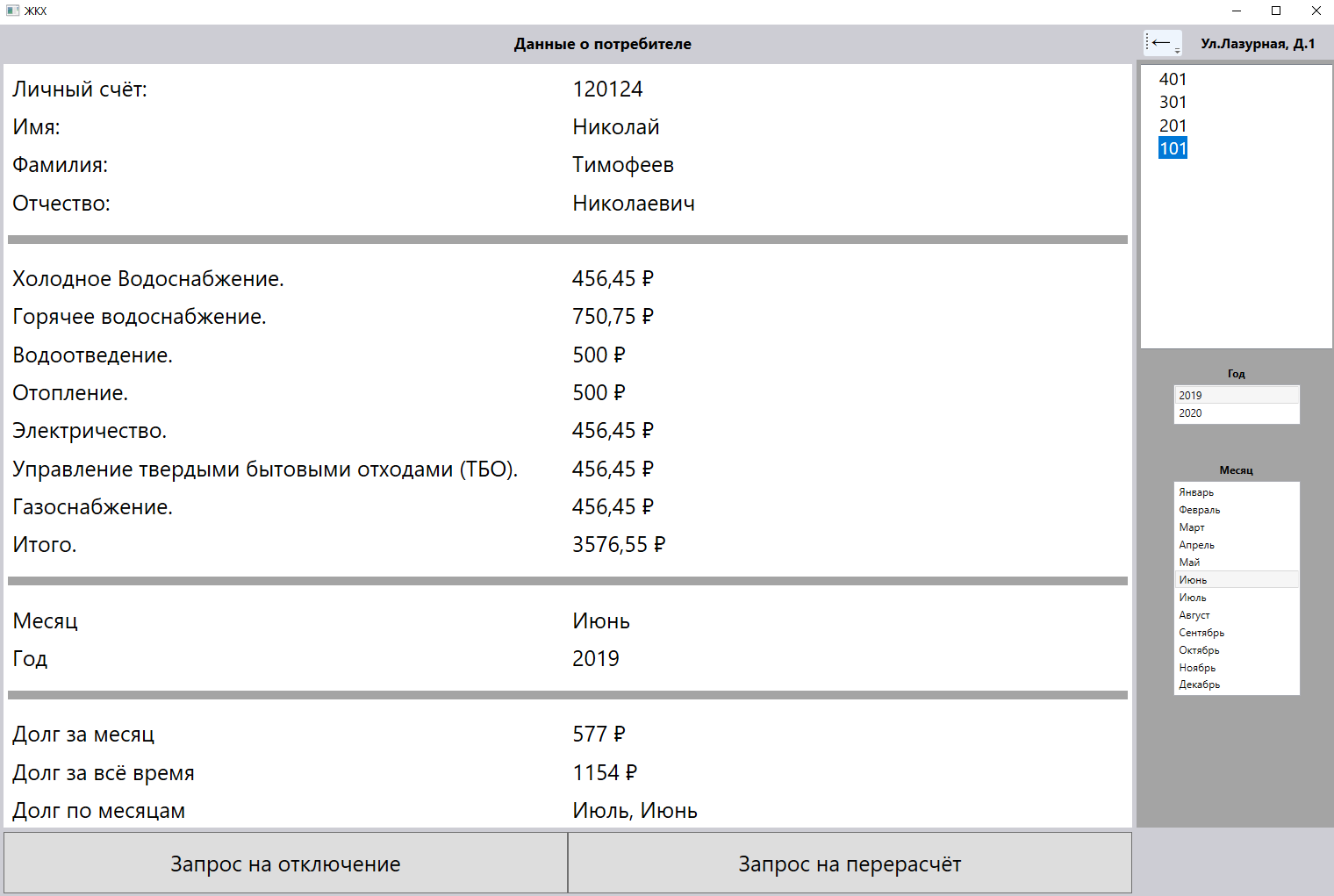


Рисунок 27 – Результат выбора квартиры

При нажатие на кнопку Запрос на отключение, на почту отправляется сообщение (рисунок 28):

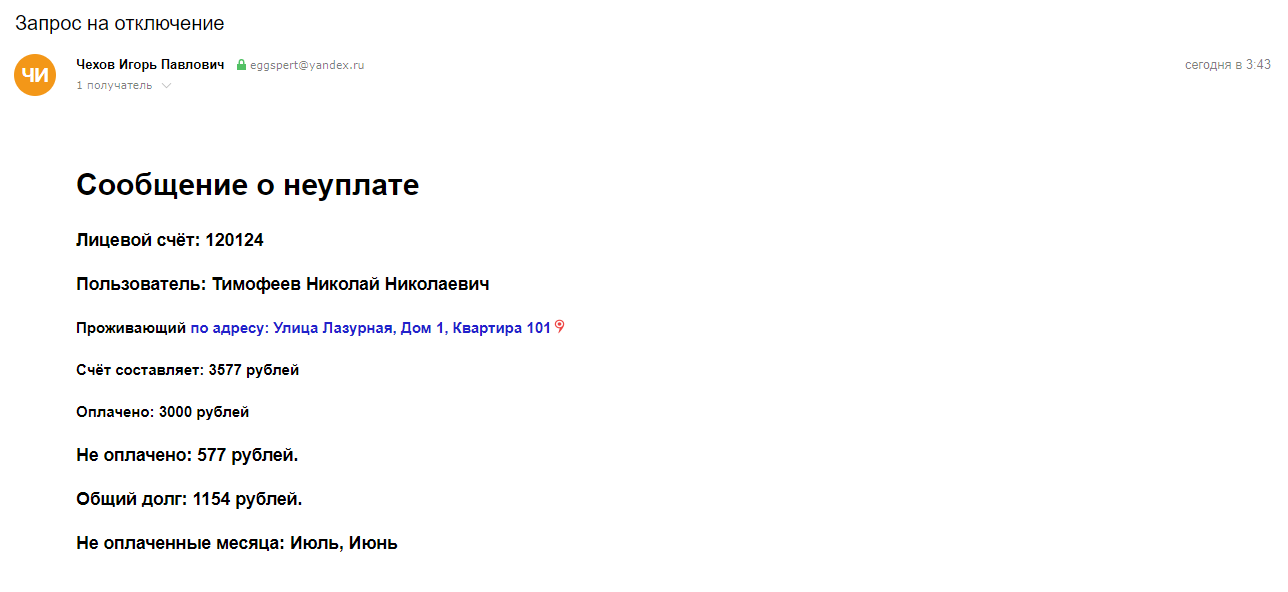


Рисунок 28 – Содержание сообщения на отключение

При нажатие на кнопку Запрос на перерасчёт. На почту отправляется сообщение (рисунок 29):

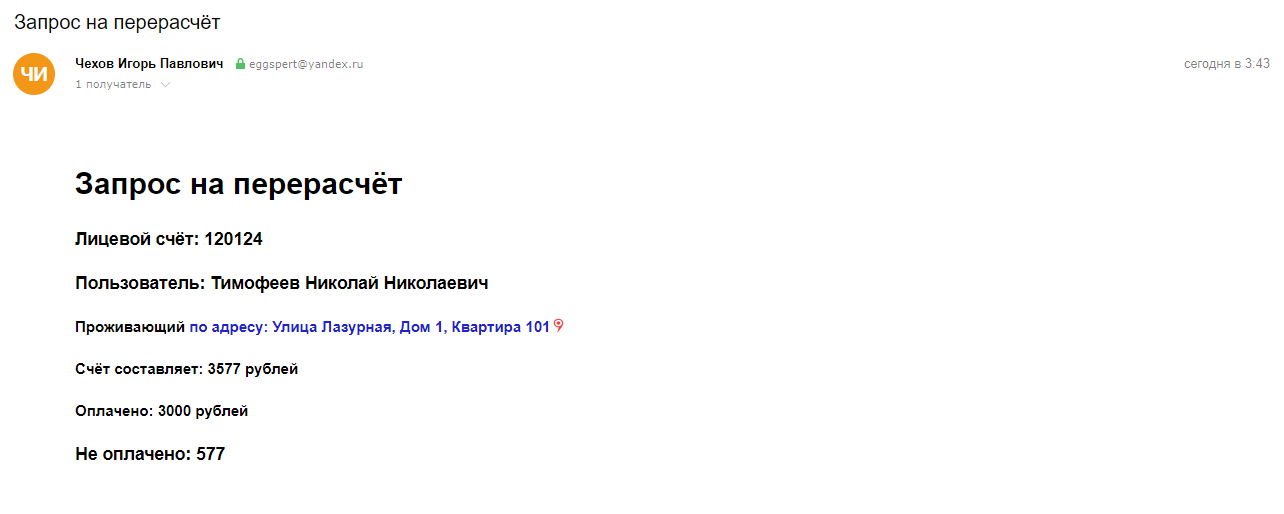


Рисунок 29 – Содержание сообщения о перерасчёте

После получения сообщения на почту, планируется, что по этому поводу будет проходить разбирательство. И через какое-то время будут внесены изменения в базу данных.

# ЭКОНОМИЧЕСКАЯ ЧАСТЬ

3.1 Расчет предполагаемой прибыли от реализации созданного программного продукта

Данный проект представляет собой программу для просмотра информации о пользователях услуг ЖКХ, с возможной последующей отправкой запроса на перерасчёт или отключение услуг.

Для этого в приложении предусмотрены несколько функций:

* + возможность просмотра информации о пользователях;
  + возможность подать заявку на отключение пользователя;
  + возможность подать заявку на перерасчёт пользователя.

Программа имеет интуитивно-понятный интерфейс, что позволяет быстро и удобно использовать его функции.

В этом разделе оценивается стоимость разработки дипломного проекта и рассчитывается предполагаемая прибыль от реализации программы.

## Расчет трудоемкости

Разработка данного программного продукта, потребовала определенных материальных, временных и трудовых затрат, а следовательно, должна окупиться.

С экономической точки зрения затраты на выполнение проекта, должны быть покрыты благодаря доходам от реализации конечного продукта.

Трудоемкость выполнения программного продукта характеризуется перечнем основных этапов и видов работ, которые должны быть выполнены.

Форма расположения работ по этапам:

1. Разработка технического задания (ТЗ): получение ТЗ.
2. Подготовительный этап:

* сбор информации;
* выбор объектного построения программы;
* разработка общей методики создания продукта;
* разработка системы взаимодействия с потребителем.

1. Основной этап:

* разработка основных алгоритмов;
* создание интерфейса;
* отладка;
* тестирование.

1. Завершающий этап:

* подготовка технической документации;
* сдача продукта.

Трудоемкость выполнения работы оценивается в человеко-часах и представлена в таблице 2.

Таблица 2 – Расчет трудоемкости

|  |  |  |
| --- | --- | --- |
| № п/п | Виды работ | Трудоёмкость |
| 1 | Получение ТЗ | 1 |
| 2 | Сбор информации и ознакомление с предметной областью | 10 |
| 3 | Выбор объектного построения программы | 12 |
| 4 | Разработка общей методики создания продукта | 15 |
| 5 | Разработка системы взаимодействия с потребителем | 10 |
| 6 | Разработка основного алгоритма | 65 |
| 7 | Создание интерфейса | 25 |
| 8 | Отладка | 13 |
| 9 | Тестирование | 10 |
| 10 | Подготовка технической документации | 15 |
| 11 | Сдача продукта | 1 |
|  | Итого | 177 |

Суммарная трудоемкость составила 177 человеко-часов.

## Расчет расходов на заработную плату

На основе данных о трудоемкости и средней заработной плате по профессии рассчитываем основную заработную плату. Предположим, что заработная плата программиста без опыта работы составляет 20 тыс. руб. в месяц (23 рабочих дня, 8 часовой рабочий день) или 108,70 руб./час.

Таким образом, расходы на заработную плату по нашему проекту составляют:

108,70\*177 = 19239,9 руб.

Начисления на заработную плату (Пенсионный фонд, Фонд социального страхования, Федеральный фонд обязательного медицинского страхования) составляют 30%. В денежном выражении составляют:

19239,9 \*0,3 = 5 771,97‬ руб.

## Расчет затрат на материалы

Материалы, затраченные на создание проекта, приведены в таблице 3.

Таблица 3 – Расчет затрат на материалы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Наименование | Единицы измерения | Количество | Цена за единицу, руб. | Сумма, руб. |
| 1 | Интернет-тариф | год. | 1 | 6000 | 6000 |
| 2 | Интернет установка | шт. | 1 | 9000 | 9000 |
| 3 | Роутер | шт. | 1 | 1000 | 1000 |
|  | Итого |  |  |  | 16000 |

## Расчет затрат на электроэнергию и амортизацию оборудования

Стоимость компьютера, на котором велась разработка, 70 тыс. руб. Используем его в течение 7 лет (84 мес.). Годовая амортизация составит в месяц 833,3 рублей. В час: 4,5 руб. Умножим на трудоёмкость:

АО= 4,5\*177 = 792,5 руб.

Рассчитаем расходы на электроэнергию.

Персональный компьютер в среднем потребляет 0,6 кВт/час:

0,6\*177=107 кВт.

Примерная стоимость 1 кВт/час около 4,32 руб.

Итого: 107 \*4,32 = 462,24 руб.

## Расчет сметы затрат

Результаты расчета затрат приведены в таблице 4.

Таблица 4 – Расчет сметы затрат

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование статей затрат | Сумма, руб. |
| 1 | Расходные материалы | 16000 |
| 2 | Основная заработная плата | 19239,9 |
| 3 | Расходы на электроэнергию | 462,24 |
| 4 | Амортизационные отчисления | 792,5 |
| 5 | Отчисления на ЗП | 5771,97 |
|  | Итого | 42266,61 |

## Расчет экономического эффекта

В данном проекте экономическим эффектом является предполагаемая прибыль, полученная в результате реализации программного продукта.

Для того чтобы окупить себя приложение должно принести прибыли больше, чем затраты на производство. Соответственно это приложение необходимо продать по цене большей, чем 42266,61 в этом случае она будет экономически выгодной. В среднем цена на аналоги составляет около 44000 руб. Тогда прибыль составит 1733,9 руб.

Главными достоинствами этого приложения являются:

1. Программа, имеющая узкую специализацию для администратора предприятия по предоставлению и учету услуг ЖКХ.
2. Интуитивно понятный интерфейс.
3. Легкость нахождения сведений о потребителе услуг.
4. Возможность оповещения пользователей о неуплате и перерасчёте по электронной почте.
5. Подробная информация по счетам конкретного потребителя.

# ЗАКЛЮЧЕНИЕ

К коммунальным услугам принято относить расходы по оплате услуг отопления, горячего и холодного водоснабжения, предоставления газа и электроэнергии, канализации (водоотведения), а также некоторые иные виды расходов, обусловленные условиями (географическими, инфраструктурными, климатическими и т.п.) осуществления деятельности организациями (например, вывоз твердых и жидких отходов, утилизация отходов и т.п.).

Тема дипломного проекта была выбрана после просмотра и анализа программных продуктов (далее – программы), реализующих деятельность предприятий в сфере ЖКХ. В этих программах была замечена недостаточно полная работа с базами данных, в которых не раскрывался весь возможный потенциал, в результате чего, было принято решение облегчить работу с базой данных в этом дипломном проекте и постараться раскрыть наибольшие возможности и простоту работы с базами данных.

Во время создания продукта учитывались достоинства и недостатки альтернативных программных продуктов.

Данная программа может послужить удобным инструментом для администратора предприятия ЖКХ, которому нужно быстро и эффективно, без дополнительного изучения работы программы, найти нужный счёт для отправления запросов и выявления добросовестных пользователей услуг ЖКХ, а также должников.

При разработке приложения были закреплены знания, такие как: работа с СУБД SQL Management studio, и IDE (Integrated Development Environment) Visual studio, построение графических приложений в WPF, а также языки программирования C#, и язык разметки XAML.

В дальнейшем, данное приложение будет развиваться в плане функционала и дизайна, будут разработаны возможности получения информации различных дополнительных услуг, такие как: содержание и ремонт лифтов, домофон, АППЗ и т.д.

# ЛИТЕРАТУРА

1. Албахари Д., Албахари Б., C# 6.0. Карманный справочник, 2015 – 453 с.
2. Браст, Э.Дж. Разработка приложений на основе Microsoft SQL Server 2008 / Э.Дж. Браст. - М.: Русская Редакция, 2010. - 751 c.
3. Вишневский, Алексей Microsoft SQL Server. Эффективная работа / Алексей Вишневский. - М.: Питер, 2009. - 143 c.
4. Голицына О., Максимов Н., Информационные системы. Учебное пособие, 2014 – 448 с.
5. Рихтер Д., CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#, 2017 – 896 с.

# 

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ГОСТ 19.101-77 Виды программ и программных документов;
2. ГОСТ 19.102-77 Стадии разработки;
3. ГОСТ 19.103-77 Обозначение программ и программных документов;
4. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом;
5. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению;
6. ГОСТ 19.202-78 Спецификация. Требования к содержанию и оформлению;
7. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению;
8. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом;
9. Stack Overflow – популярная система вопросов и ответов о программировании.
10. METANIT – форум программистов и сисадминов.

**ПРИЛОЖЕНИЕ А. Исходный код программы**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace JKH

{

class SQLPrebludi

{

public string Account { get; set; }

public SqlConnection getConnection()

{

return new SqlConnection( new SqlConnectionStringBuilder()

{

DataSource = @".\SQLExpress",

InitialCatalog = "testDiplom",

IntegratedSecurity = true

}.ConnectionString );

}

public string TakeNameStreet()

{

return "SELECT [Name] FROM [Street]";

}

public string TakeNameMonth()

{

return "SELECT [NeedMonth] FROM [TimeMonth] ORDER BY [Id]";

}

public string TakeNameYear()

{

return "SELECT [NeedYear] FROM [TimeYear]";

}

public string TakeNumberHome(string word)

{

return "SELECT Number FROM [Home] Where StreetName = " + "'" + word + "'";

}

public string TakeNumberFlat(string word)

{

return "SELECT FlatNumber FROM [Flat] Where HomeNumber = " + "'" + word + "'";

}

public string TakePersonalAccount(string flatNumber)

{

return "SELECT FlatPersonalAccount FROM [Flat] Where FlatNumber = " + "'" + flatNumber + "'";

}

public string TakeConsumerInfo(string accountNumber)

{

return "SELECT \* FROM [Сonsumer] Where PersonalAccount = " + "'" + accountNumber + "'";

}

public string TakeRentInfo(string accountNumber, int selectYear, string selectMonth)

{

return "SELECT \* FROM [Rent] Where PersonalAccountRent = " + "'" + accountNumber + "'" + " AND NeedYear = " + "'" + selectYear + "'" + " AND NeedMonth = " + "'" + selectMonth + "'";

}

public string TakeDebtorRent(string accountNumber)

{

return "SELECT \* FROM [Rent] Where Debtor = " + "'1'";

}

public DebtorVal TakeDebtor(string method)

{

DebtorVal debtor = new DebtorVal();

string query = method;

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Rent rent = new Rent();

rent.ColdWaterSupply = float.Parse(reader[1].ToString());

rent.HotWaterSupply = float.Parse(reader[2].ToString());

rent.WaterDisposal = float.Parse(reader[3].ToString());

rent.Heating = float.Parse(reader[4].ToString());

rent.PowerSupply = float.Parse(reader[5].ToString());

rent.SolidMunicipalWasteManagement = float.Parse(reader[6].ToString());

rent.GasSupply = float.Parse(reader[7].ToString());

rent.DeptorType = Convert.ToBoolean(reader[8]);

rent.PaidUp = float.Parse(reader[9].ToString());

rent.NeededYear = Convert.ToInt32(reader[10]);

rent.NeededMonth = reader[11].ToString();

rent.PersonalAccountRent = Convert.ToInt32(reader[12]);

debtor.DebtorRent.Add(rent);

}

reader.Close();

connection.Dispose();

connection.Close();

return debtor;

}

}

public NeedMonth TakeMonth(string method)

{

NeedMonth month = new NeedMonth();

string query = method;

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

month.AllMonth.Add(reader[0].ToString());

}

reader.Close();

connection.Dispose();

connection.Close();

return month;

}

}

public NeedYear TakeYear(string method)

{

NeedYear year = new NeedYear();

string query = method;

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

year.AllYear.Add(Convert.ToInt32(reader[0]));

}

reader.Close();

connection.Dispose();

connection.Close();

return year;

}

}

public Home TakeHome(string method)

{

Home home = new Home();

string query = method;

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

home.NumberHome.Add(Convert.ToInt32(reader[0]));

}

reader.Close();

connection.Dispose();

connection.Close();

return home;

}

}

public Flat TakeFlat(string method)

{

Flat flat = new Flat();

string query = method;

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

flat.NumberFlat.Add(Convert.ToInt32(reader[0]));

}

reader.Close();

connection.Dispose();

connection.Close();

return flat;

}

}

public Street TakeStreet(string method)

{

Street street = new Street();

string query = method;

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

street.nameStreet.Add((reader[0].ToString()));

}

reader.Close();

connection.Dispose();

connection.Close();

return street;

}

}

public void TakeAccount(string method)

{

string query = method;

try

{

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Account = reader[0].ToString();

}

reader.Close();

connection.Dispose();

connection.Close();

}

}

catch (Exception E)

{

MessageBox.Show(E.Message);

}

}

public Consumer TakeConsumer(string method)

{

Consumer consuk = new Consumer();

string query = method;

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

consuk.Name = reader[1].ToString();

consuk.Surname = reader[2].ToString();

consuk.Patronymic = reader[3].ToString();

consuk.PersonalAccount = Convert.ToInt32(reader[4]);

}

reader.Close();

connection.Dispose();

connection.Close();

return consuk;

}

}

public Rent TakeRent(string method)

{

Rent rent = new Rent();

string query = method;

using (SqlConnection connection = new SQLPrebludi().getConnection())

{

connection.Open();

SqlCommand command = new SqlCommand(query, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

rent.ColdWaterSupply = float.Parse(reader[1].ToString());

rent.HotWaterSupply = float.Parse(reader[2].ToString());

rent.WaterDisposal = float.Parse(reader[3].ToString());

rent.Heating = float.Parse(reader[4].ToString());

rent.PowerSupply = float.Parse(reader[5].ToString());

rent.SolidMunicipalWasteManagement = float.Parse(reader[6].ToString());

rent.GasSupply = float.Parse(reader[7].ToString());

rent.DeptorType = Convert.ToBoolean(reader[8]);

rent.PaidUp = float.Parse(reader[9].ToString());

rent.NeededYear = Convert.ToInt32(reader[10]);

rent.NeededMonth = reader[11].ToString();

rent.PersonalAccountRent = Convert.ToInt32(reader[12]);

}

reader.Close();

connection.Dispose();

connection.Close();

return rent;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.Net.Mail;

namespace JKH

{

public class MessagesForRecalculation

{

public MessagesForRecalculation(Street street, Home home, Flat flat, Consumer consumer, float allDolg, Rent rent)

{

MailAddress From = new MailAddress("EGGspert@yandex.ru", "Чехов Игорь Павлович");

MailAddress To = new MailAddress("EGGspert@yandex.ru");

MailMessage message = new MailMessage(From, To);

message.Subject = "Запрос на перерасчёт";

string mda = "<h1 align='Center'>Запрос на перерасчёт</h1><h3> Лицевой счёт: " + consumer.PersonalAccount.ToString() + "</h3><h3> Пользователь: " + consumer.Surname.ToString() + " " + consumer.Name.ToString() + " " + consumer.Patronymic.ToString() + "</h3><h4> Проживающий по адресу: Улица " + street.selectedStreetName.ToString() + ", Дом " + home.SelectedNumberHome + ", Квартира " + flat.SelectedNumberFlat + "</h4><h4> Счёт составляет: " + Math.Round(allDolg) + " рублей</h4><h4> Оплачено: " + Math.Round(rent.PaidUp) + " рублей</h4><h3> Не оплачено: " + (Math.Round(allDolg) - Math.Round(rent.PaidUp)) + "</h3>";

message.Body = mda;

message.IsBodyHtml = true;

SmtpClient smtp = new SmtpClient("smtp.yandex.ru", 587);

smtp.Credentials = new NetworkCredential("EGGspert@yandex.ru", "Tions3120771");

smtp.EnableSsl = true;

smtp.Send(message);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.Net.Mail;

namespace JKH

{

public class MessagesForShutdown

{

public MessagesForShutdown(Street street, Home home, Flat flat, Consumer consumer, float allDolg, Rent rent, string allMonth, string allMoney)

{

MailAddress From = new MailAddress("EGGspert@yandex.ru", "Чехов Игорь Павлович");

MailAddress To = new MailAddress("EGGspert@yandex.ru");

MailMessage message = new MailMessage(From, To);

message.Subject = "Запрос на октлючение";

string mda = "<h1 align='Center'>Сообщение о неуплате</h1><h3> Лицевой счёт: " + consumer.PersonalAccount.ToString() + "</h3><h3> Пользователь: " + consumer.Surname.ToString() + " " + consumer.Name.ToString() + " " + consumer.Patronymic.ToString() + "</h3><h4> Проживающий по адресу: Улица " + street.selectedStreetName.ToString() + ", Дом " + home.SelectedNumberHome + ", Квартира "+ flat.SelectedNumberFlat + "</h4><h4> Счёт составляет: " + Math.Round(allDolg) + " рублей</h4><h4> Оплачено: " + Math.Round(rent.PaidUp) + " рублей</h4><h3> Не оплачено: " + (Math.Round(allDolg) - Math.Round(rent.PaidUp)) + " рублей.</h3><h3> Общий долг: " + allMoney.ToString() + " рублей.</h3><h3> Не оплаченные месяца: " + allMonth.ToString() + "</h3>";

message.Body = mda;

message.IsBodyHtml = true;

SmtpClient smtp = new SmtpClient("smtp.yandex.ru",587);

smtp.Credentials = new NetworkCredential("EGGspert@yandex.ru","Tions3120771");

smtp.EnableSsl = true;

smtp.Send(message);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace JKH

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

SQLPrebludi sqlPr = new SQLPrebludi();

public Consumer consumer { get; set; }

public Rent RentMode { get; set; }

public Home HomeMode { get; set; }

public Flat FlatMode { get; set; }

public Street StreetMode { get; set; }

public NeedYear NeedYearMode { get; set; }

public NeedMonth NeedMonthMode { get; set; }

public DebtorVal debtorAll { get; set; }

public float DolgForMonth { get; set; }

public float DolgAll { get; set; }

public string AllMonth { get; set; }

public MainWindow()

{

InitializeComponent();

sqlPr.getConnection();

StreetMode = sqlPr.TakeStreet(sqlPr.TakeNameStreet());

foreach (string name in StreetMode.nameStreet)

{

Lez.Items.Add(name);

}

NeedYearMode = sqlPr.TakeYear(sqlPr.TakeNameYear());

foreach (int name in NeedYearMode.AllYear)

{

ListYear.Items.Add(name);

}

NeedMonthMode = sqlPr.TakeMonth(sqlPr.TakeNameMonth());

foreach (string name in NeedMonthMode.AllMonth)

{

ListMonth.Items.Add(name);

}

}

void DebtorInfo(DebtorVal debtorInfo)

{

float doljok;

float info = 0;

string month = null;

for (int i = 0; i < debtorInfo.DebtorRent.Count; i++)

{

if (month == null)

{

month = debtorInfo.DebtorRent[i].NeededMonth;

}

else

{

month = month + ", " + debtorInfo.DebtorRent[i].NeededMonth;

}

doljok = debtorInfo.DebtorRent[i].ColdWaterSupply + debtorInfo.DebtorRent[i].HotWaterSupply + debtorInfo.DebtorRent[i].WaterDisposal + debtorInfo.DebtorRent[i].Heating + debtorInfo.DebtorRent[i].PowerSupply + debtorInfo.DebtorRent[i].SolidMunicipalWasteManagement + debtorInfo.DebtorRent[i].GasSupply;

info = info + (float)(Math.Round(doljok) - Math.Round(debtorInfo.DebtorRent[i].PaidUp));

}

DebtorMonth.Content = month;

DebtorAll.Content = info + " ₽";

DolgAll = info ;

AllMonth = month;

}

private void Lez\_SelectedItemChanged(object sender, RoutedPropertyChangedEventArgs<object> e)

{

if (Lez.SelectedItem != null)

{

if (labAdress.Content.ToString().Contains("Д.") && labAdress.Content.ToString().Contains("Ул."))

{

if (ListYear.SelectedItem != null && ListMonth.SelectedItem != null)

{

sqlPr.TakeAccount(sqlPr.TakePersonalAccount(Lez.SelectedItem.ToString()));

consumer = sqlPr.TakeConsumer(sqlPr.TakeConsumerInfo(sqlPr.Account));

RentMode = sqlPr.TakeRent(sqlPr.TakeRentInfo(sqlPr.Account, NeedYearMode.SelectedYear, NeedMonthMode.SelectedMonth));

debtorAll = sqlPr.TakeDebtor(sqlPr.TakeDebtorRent(sqlPr.Account));

AccountNumber.Content = sqlPr.Account;

FlatMode.SelectedNumberFlat = Convert.ToInt32(Lez.SelectedItem);

RealName.Content = consumer.Name;

RealFam.Content = consumer.Surname;

RealPat.Content = consumer.Patronymic;

ColdWater.Content = RentMode.ColdWaterSupply + " ₽";

HotWater.Content = RentMode.HotWaterSupply + " ₽";

WaterOut.Content = RentMode.WaterDisposal + " ₽";

Otop.Content = RentMode.Heating + " ₽";

PowerLight.Content = RentMode.PowerSupply + " ₽";

Musor.Content = RentMode.SolidMunicipalWasteManagement + " ₽";

Gas.Content = RentMode.GasSupply + " ₽";

NameMonth.Content = RentMode.NeededMonth;

NameYear.Content = RentMode.NeededYear;

DolgForMonth = RentMode.ColdWaterSupply + RentMode.HotWaterSupply + RentMode.WaterDisposal + RentMode.Heating + RentMode.PowerSupply + RentMode.SolidMunicipalWasteManagement + RentMode.GasSupply;

SUD.IsEnabled = true;

Replace.IsEnabled = true;

Debtor.Content = Math.Round(DolgForMonth) - Math.Round(RentMode.PaidUp) + " ₽";

Itog.Content = DolgForMonth.ToString() + " ₽";

DebtorInfo(debtorAll);

}

else

{

MessageBox.Show("Выберите год и месяц!");

}

}

else if (labAdress.Content.ToString().Contains("Ул."))

{

labAdress.Content = labAdress.Content + ", Д." + Lez.SelectedItem.ToString();

FlatMode = sqlPr.TakeFlat(sqlPr.TakeNumberFlat(Lez.SelectedItem.ToString()));

HomeMode.SelectedNumberHome = Convert.ToInt32(Lez.SelectedItem);

Lez.Items.Clear();

foreach (int number in FlatMode.NumberFlat)

{

Lez.Items.Add(number);

}

}

else

{

labAdress.Content = "Ул." + Lez.SelectedItem.ToString();

HomeMode = sqlPr.TakeHome(sqlPr.TakeNumberHome(Lez.SelectedItem.ToString()));

StreetMode.selectedStreetName = Lez.SelectedItem.ToString();

Lez.Items.Clear();

foreach (int number in HomeMode.NumberHome)

{

Lez.Items.Add(number);

}

}

}

}

private void arrowBack\_Click(object sender, RoutedEventArgs e)

{

if (labAdress.Content.ToString().Contains("Д.") && labAdress.Content.ToString().Contains("Ул."))

{

Lez.Items.Clear();

labAdress.Content = "Ул." + StreetMode.selectedStreetName;

HomeMode = sqlPr.TakeHome(sqlPr.TakeNumberHome(StreetMode.selectedStreetName));

SUD.IsEnabled = false;

Replace.IsEnabled = false;

Lez.Items.Clear();

foreach (int number in HomeMode.NumberHome)

{

Lez.Items.Add(number);

}

}

else if (labAdress.Content.ToString().Contains("Ул."))

{

Lez.Items.Clear();

labAdress.Content = "Все улицы";

StreetMode = sqlPr.TakeStreet(sqlPr.TakeNameStreet());

SUD.IsEnabled = false;

Replace.IsEnabled = false;

foreach (string name in StreetMode.nameStreet)

{

Lez.Items.Add(name);

}

}

else

{

MessageBox.Show("Не куда больше идти!");

}

}

private void SUD\_Click(object sender, RoutedEventArgs e)

{

MessagesForShutdown hoba = new MessagesForShutdown(StreetMode, HomeMode, FlatMode, consumer, DolgForMonth,RentMode,AllMonth,DolgAll.ToString());

}

private void ListYear\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

NeedYearMode.SelectedYear = Convert.ToInt32(ListYear.SelectedItem);

}

private void ListMonth\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

NeedMonthMode.SelectedMonth = ListMonth.SelectedItem.ToString();

}

private void Replace\_Click(object sender, RoutedEventArgs e)

{

MessagesForRecalculation hoba = new MessagesForRecalculation(StreetMode, HomeMode, FlatMode, consumer, DolgForMonth, RentMode);

}

}

}

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JKH

{

public class Consumer

{

public int PersonalAccount { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public string Patronymic { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JKH

{

public class Flat

{

public List<int> NumberFlat { get; set; }

public int SelectedNumberFlat { get; set; }

public Flat()

{

NumberFlat = new List<int>();

}

}

}

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JKH

{

public class Street

{

public List<string> nameStreet { get; set; }

public string selectedStreetName { get; set; }

public Street()

{

nameStreet = new List<string>();

}

}

}

<Window x:Class="JKH.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:JKH"

mc:Ignorable="d"

Title="ЖКХ" Height="1024" Width="1540">

<Grid Background="#FFCDCDD4">

<Grid.RowDefinitions>

<RowDefinition Height="40"/>

<RowDefinition Height="339"/>

<RowDefinition Height="536"/>

<RowDefinition Height="80"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="10\*"/>

<ColumnDefinition Width="153\*"/>

<ColumnDefinition Width="228"/>

</Grid.ColumnDefinitions>

<Label x:Name="labAdress" Grid.Row="0" Grid.Column="2" HorizontalAlignment="Center" VerticalAlignment="Top" FontSize="16" Content="Все улицы" FontWeight="Bold" Height="31" Margin="50,5,0,0" />

<Label x:Name="lab" Grid.Row="0" Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="18" Content="Данные о потребителе" FontWeight="Bold" Height="32" />

<StackPanel Height="536" Grid.Column="2" Grid.Row="2" HorizontalAlignment="Center" VerticalAlignment="Top" Width="228" Background="#FFA4A4A4">

<StackPanel HorizontalAlignment="Center" Height="100" Margin="5" VerticalAlignment="Top" Width="145">

<Label Content="Год" Grid.Row="0" VerticalAlignment="Center" HorizontalAlignment="Center" FontWeight="Bold" Height="26" Width="30" />

<ListBox x:Name="ListYear" Height="46" VerticalAlignment="Top" Width="145" SelectionChanged="ListYear\_SelectionChanged"/>

</StackPanel>

<StackPanel HorizontalAlignment="Center" Height="275" Margin="5" Grid.Row="2" VerticalAlignment="Top" Width="145">

<Label Content="Месяц" Grid.Column="1" Grid.Row="1" HorizontalAlignment="Center" FontWeight="Bold" Width="49"/>

<ListBox x:Name="ListMonth" Height="245" VerticalAlignment="Top" Width="145" SelectionChanged="ListMonth\_SelectionChanged"/>

</StackPanel>

</StackPanel>

<UniformGrid Grid.Column="2" Grid.Row="1" Background="#FFA4A4A4">

<TreeView x:Name="Lez" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="5" Height="325" Width="220" SelectedItemChanged="Lez\_SelectedItemChanged" FontSize="20" />

</UniformGrid>

<UniformGrid Grid.Column="2" Grid.Row="0" Margin="5" >

<ToolBar HorizontalAlignment="Left" Height="32" VerticalAlignment="Top" Width="48" FontWeight="Bold" Margin="0,0,0,-1">

<Button x:Name="arrowBack" Content="←" FontSize="25" HorizontalAlignment="Center" VerticalAlignment="Center" Width="24" Height="31" FontFamily="Yu Gothic UI" Padding="0" Margin="0,0,0,5" Click="arrowBack\_Click"/>

</ToolBar>

</UniformGrid>

<UniformGrid Grid.Column="0" Grid.Row="3" Margin="5" Rows="1" Grid.ColumnSpan="2">

<Button x:Name="SUD" IsEnabled="False" Content="Запрос на отключение" FontSize="25" Click="SUD\_Click"/>

<Button x:Name="Replace" IsEnabled="False" FontSize="25" Content="Запрос на перерасчёт" Click="Replace\_Click"/>

</UniformGrid>

<StackPanel Height="870" VerticalAlignment="Top" Background="White" Margin="5,5,5,0" Grid.Column="0" Grid.Row="1" Grid.RowSpan="2" Grid.ColumnSpan="2" >

<UniformGrid Grid.Column="2" Grid.Row="1" Margin="5" Columns="2" >

<Label FontSize="25" Content="Личный счёт:"/>

<Label FontSize="25" x:Name="AccountNumber"/>

<Label FontSize="25" Content="Имя:"/>

<Label FontSize="25" x:Name="RealName"/>

<Label FontSize="25" Content="Фамилия:"/>

<Label FontSize="25" x:Name="RealFam"/>

<Label FontSize="25" Content="Отчество:"/>

<Label FontSize="25" x:Name="RealPat"/>

<Label FontSize="25" Height="10" Background="#FFA4A4A4"/>

<Label FontSize="25" Height="10" Background="#FFA4A4A4"/>

<Label FontSize="25" Content="Холодное Водоснабжение."/>

<Label FontSize="25" x:Name="ColdWater"/>

<Label FontSize="25" Content="Горячее водоснабжение."/>

<Label FontSize="25" x:Name="HotWater"/>

<Label FontSize="25" Content="Водоотведение."/>

<Label FontSize="25" x:Name="WaterOut"/>

<Label FontSize="25" Content="Отопление."/>

<Label FontSize="25" x:Name="Otop"/>

<Label FontSize="25" Content="Электричество."/>

<Label FontSize="25" x:Name="PowerLight"/>

<Label FontSize="25" Content="Управление твердыми бытовыми отходами (ТБО)."/>

<Label FontSize="25" x:Name="Musor"/>

<Label FontSize="25" Content="Газоснабжение."/>

<Label FontSize="25" x:Name="Gas"/>

<Label FontSize="25" Content="Итого."/>

<Label FontSize="25" x:Name="Itog"/>

<Label FontSize="25" Height="10" Background="#FFA4A4A4"/>

<Label FontSize="25" Height="10" Background="#FFA4A4A4"/>

<Label FontSize="25" Content="Месяц"/>

<Label FontSize="25" x:Name="NameMonth"/>

<Label FontSize="25" Content="Год"/>

<Label FontSize="25" x:Name="NameYear"/>

<Label Height="10" Background="#FFA4A4A4"/>

<Label Height="10" Background="#FFA4A4A4"/>

<Label FontSize="25" Content="Долг за месяц"/>

<Label FontSize="25" x:Name="Debtor"/>

<Label FontSize="25" Content="Долг за всё время"/>

<Label FontSize="25" x:Name="DebtorAll"/>

<Label FontSize="25" Content="Долг по месяцам"/>

<Label FontSize="25" x:Name="DebtorMonth"/>

</UniformGrid>

</StackPanel>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JKH

{

public class DebtorVal

{

public List<Rent> DebtorRent { get; set; }

public DebtorVal()

{

DebtorRent = new List<Rent>();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JKH

{

public class NeedMonth

{

public string SelectedMonth { get; set; }

public List<string> AllMonth { get; set; }

public NeedMonth()

{

AllMonth = new List<string>();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JKH

{

public class NeedYear

{

public int SelectedYear { get; set; }

public List<int> AllYear { get; set; }

public NeedYear()

{

AllYear = new List<int>();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JKH

{

public class Home

{

public List<int> NumberHome { get; set; }

public int SelectedNumberHome { get; set; }

public Home()

{

NumberHome = new List<int>();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace JKH

{

public class Rent

{

public float ColdWaterSupply { get; set; }

public float HotWaterSupply { get; set; }

public float WaterDisposal { get; set; }

public float Heating { get; set; }

public float PowerSupply { get; set; }

public float SolidMunicipalWasteManagement { get; set; }

public float GasSupply { get; set; }

public float PersonalAccountRent { get; set; }

public bool DeptorType { get; set; }

public float PaidUp { get; set; }

public string NeededMonth{ get; set;}

public int NeededYear{ get; set;}

}

}