Содержание

[Введение 4](#_Toc509217476)

[1 Предварительный анализ 5](#_Toc509217477)

[2 Основные проектные решения 8](#_Toc509217478)

[2.1 Разработка интерфейса пользователя 8](#_Toc509217479)

[2.2 Разработка модуля программы 10](#_Toc509217480)

[2.3 Разработка печатной формы (отчета) 16](#_Toc509217481)

[2.4 Разработка ролей и пользователей 22](#_Toc509217482)

[3 Тестирование 24](#_Toc509217483)

[3.1 Описание контрольного примера 24](#_Toc509217484)

[3.2 Тестирование системы 25](#_Toc509217485)

[Заключение 29](#_Toc509217486)

[Список использованных источников 30](#_Toc509217487)

[Приложение А Руководство пользователя 31](#_Toc509217488)

[Приложение Б Электронный носитель с программой и пояснительной запиской 33](#_Toc509217489)

# Введение

Данный курсовой проект выполнен в соответствии с заданием на курсовое проектирование. Темой курсового проекта является «Эксплуатация и модификация ИС по расчету параметров строительства домов и сооружений».

Целью курсового проекта является закрепление полученных знаний во время обучения дисциплины МДК 01.01 Эксплуатация информационной системы.

Задачами курсового проектирования являются:

* изучение предметной области;
* разработка структуры базы данных;
* разработка подсистемы наполнения базы данных;
* разработка интерфейса пользователя;
* разграничение доступа;
* тестирование системы;
* разработка печатной формы;
* разработка технической документации

Данная информационная система призвана упростить работу, благодаря чему минимизируется время на расчет. С помощью него у пользователей появится возможность оперативно и точно просмотреть все необходимые им данные о расчетах, тем самым избежав возможных ошибок и сократив время, которое было бы затрачено при ручном выполнении данных операций.

# 1 Предварительный анализ

Строительство сооружений – комплекс технологических операций по возведению устойчивых зданий, рассчитанных на продолжительное использование без негативных последствий. После постройки здания, необходима внутренняя отделка, в частности отделка стен, потолка и пола различными материалами: краской для стен, грунтовой краской, натяжными потолками, обоями, потолочными и напольными плинтусами, линолеумом, паркетом и т.д. В результате, для точного расхода материалов чтобы не допустить перерасход или недостаток, человеку необходимо рассчитать площадь и периметр стен, пола, потолка, а также общую площадь проемов.

Например: для расчёта площади стен в обычной комнате, человеку необходимо сделать по одному расчёту используя формулы 1.1; 1.2; 1.3 (Рисунок 1.1).

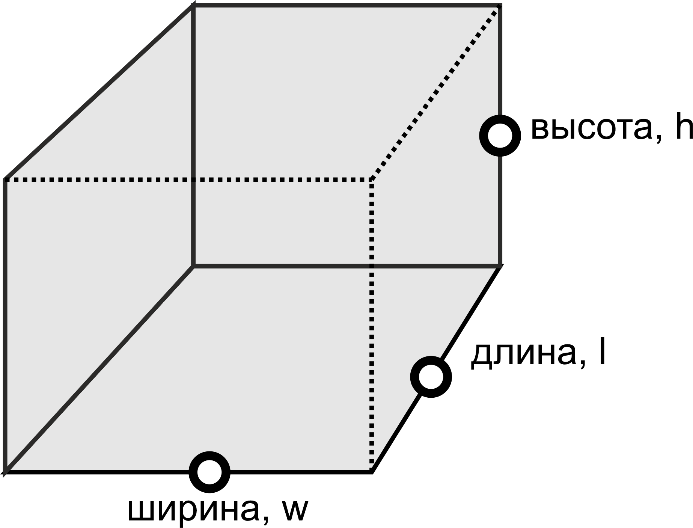


Рисунок 1.1 – Схема обычной комнаты

(1.1)

(1.2)

(1.3)

А для расчёта пола/потолка может использоваться только одна формула - (1.4).

(1.4)

Но если комната имеет хотя бы одну входную дверь или окно (рисунок 1.2).

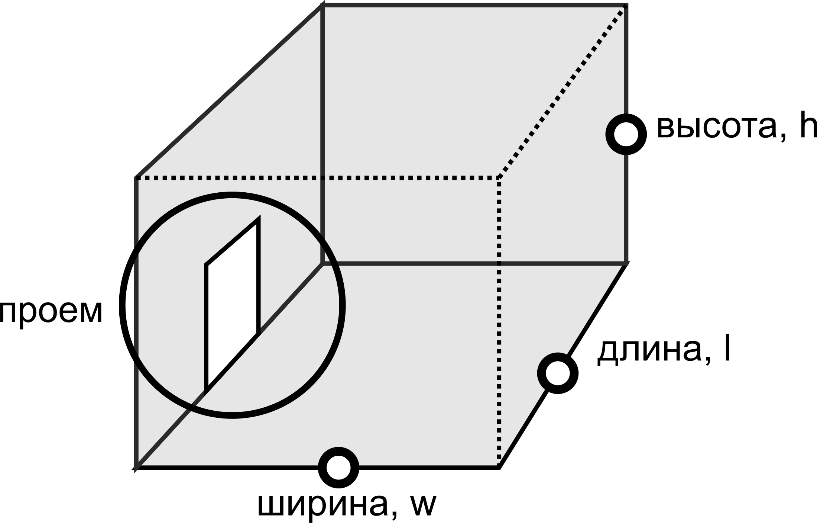


Рисунок 1.2 – Схема обычной комнаты с проемом

Тогда человеку необходимо так же сделать три вычисления используя формулы 1.1; 1.2; 1.3, но помимо этого нужно также рассчитать площадь проемов (дверь или окно) по формуле 1.5; 1.6, после использовав формулу 1.7 получить площадь стен без проемов.

(1.5)

(1.6)

(1.7)

В результате чтобы рассчитать обычную комнату человеку необходимо сделать 7 расчётов при общем количестве входных данных равных 7. Но помимо обычных комнат с параллельным полу потолком, есть комнаты со скошенным потолком (Рисунок 1.3).

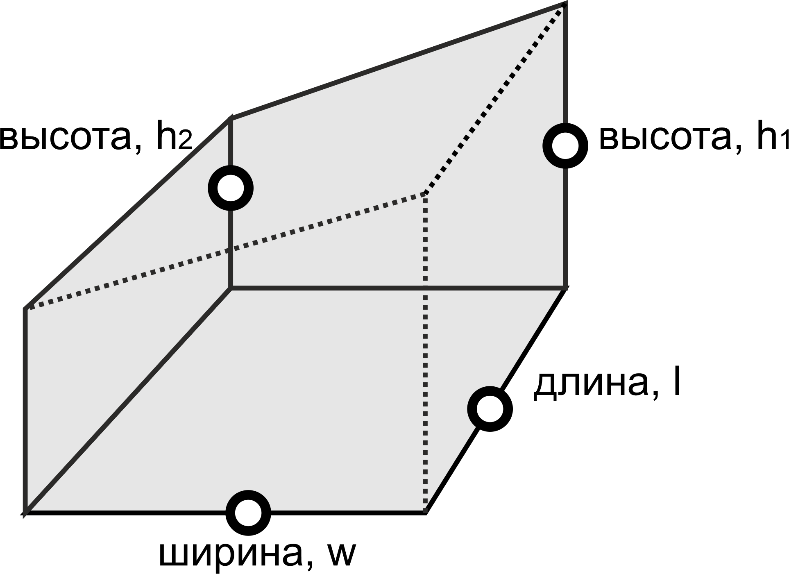


Рисунок 1.3 – Схема комнаты со скошенным потолком

Обычно такие комнаты находятся на самых верхних этажах. Рассчитывать такие комнаты сложнее так как добавляется новое входное значение – вторая высота комнаты (h2). Например, чтобы рассчитать площадь потолка, необходимо воспользоваться формулой нахождение разностей высот (1.8) а также вычисление одной из сторон потолка по формуле пифагора с последующим умножением на другую сторону потолка и нахождение его площади (1.9). В этом примере добавилось новое условие (скошенный потолок) и новое входное значение (h2), но вычисление увеличилось с одного простого расчёта с двумя входными данными по формуле 1.4, до двух расчётов с квадратным корнем и тремя входными значениями по формулам 1.8; 1.9. Так же нахождение периметра потолка обычной комнаты по формуле 1.10, будет рассчитываться по формуле 1.11.

(1.8)

(1.9)

(1.9)

(1.10)

(1.11)

# 2 Основные проектные решения

# 2.1 Разработка интерфейса пользователя

Форма авторизации состоит из полей ввода логина и пароля, а также кнопок «Войти», «Регистрация», «Тестовый режим». После ввода логина и пароля, в зависимости от роли откроется доступ к специальным вкладкам.

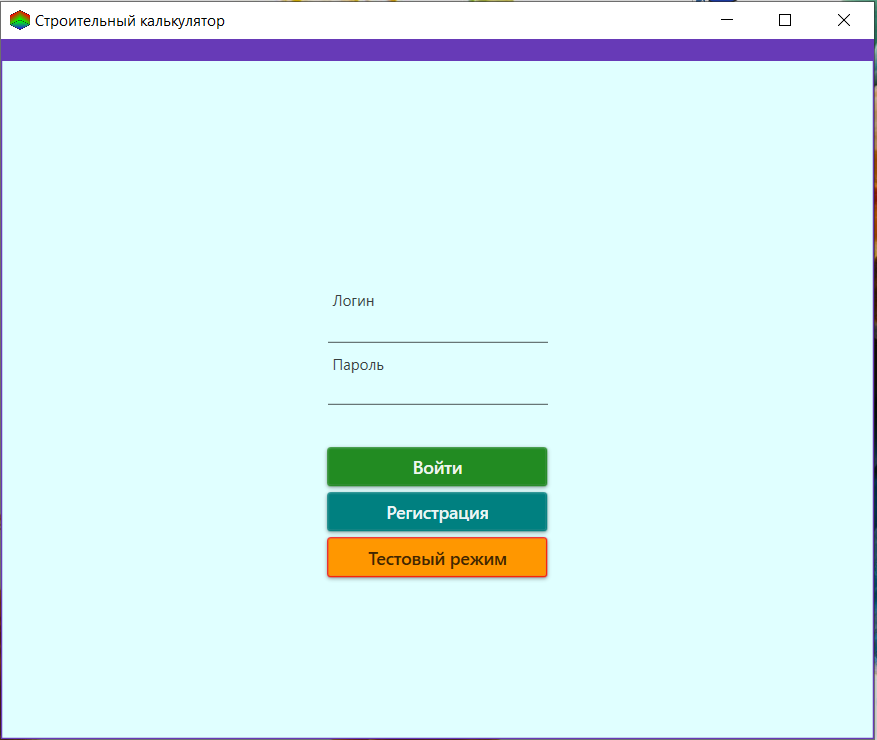


Рисунок 2.1 – Форма авторизации

Если на сайте авторизуется обычный пользователь, то в меню появится ссылка на историю расчетов.

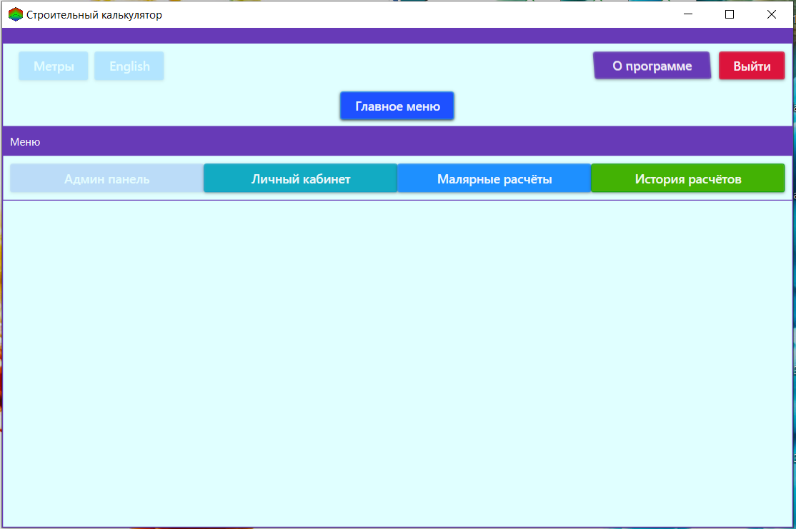


Рисунок 2.2 – Главное меню (Обычный пользователь)

Во вкладке «Личный кабинет» пользователь видит информацию о своем профиле.

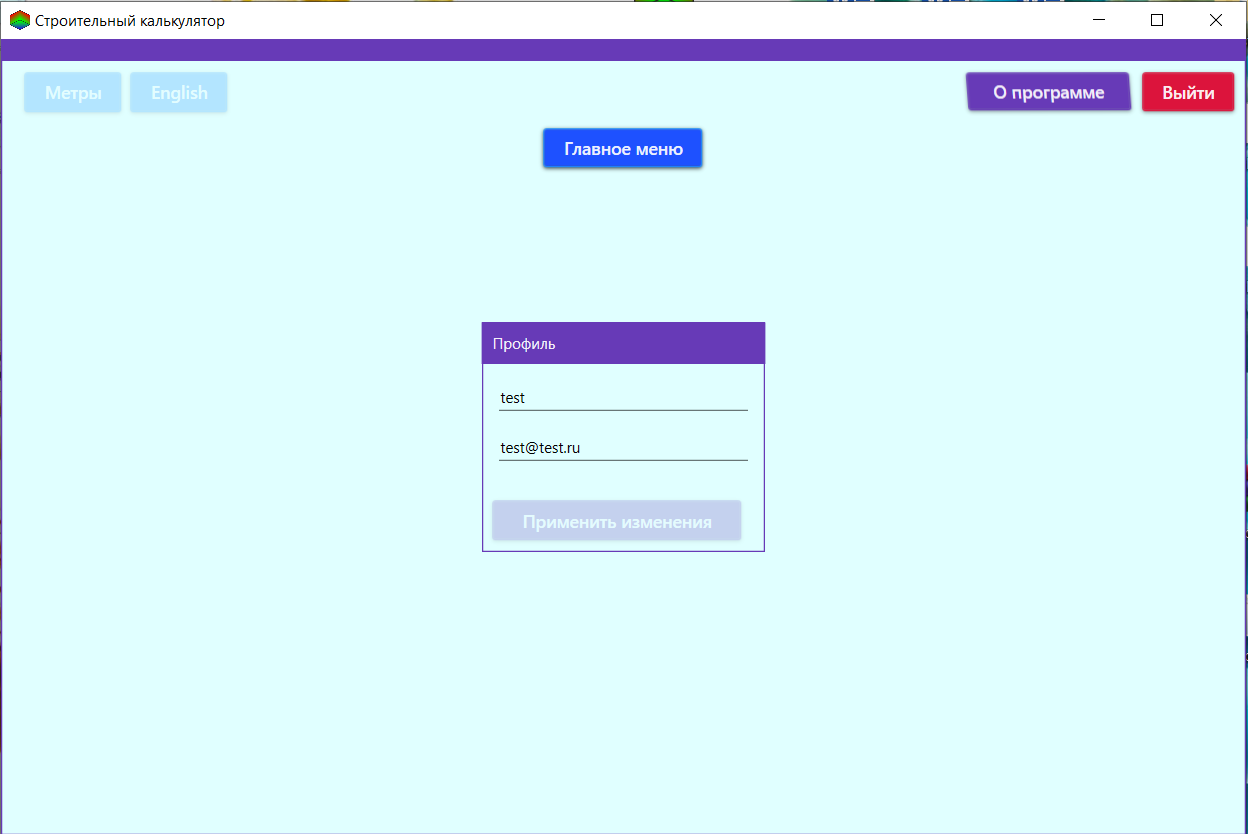


Рисунок 2.3 – Личный кабинет

Если на сайте авторизуются администратор системы, то ему открыт доступ к информации о пользователях, так же возможность их удалять.

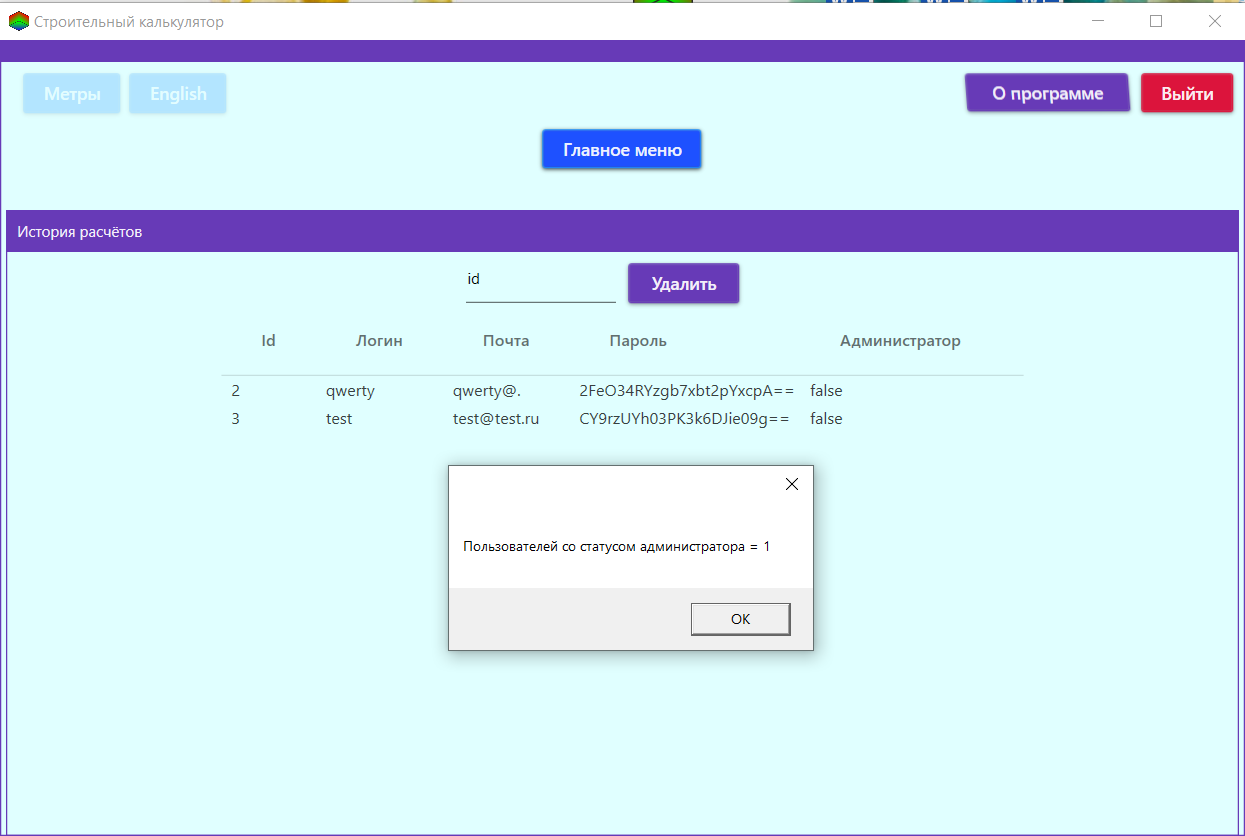


Рисунок 2.4 – Управление пользователями

# 2.2 Разработка модуля программы

Таблица 2.1 - Входные данные.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Единица измерения | Обозначение |
| Id | int | - | Ключевое поле |
| User\_id | text | - | Номер пользователя |
| Calc\_type | text | - | Тип расчёта |
| An1 | text | Градусы, 0°- 360° | Угол между стенами №1 |
| An2 | text | Градусы, 0°- 360° | Угол между стенами №2 |
| An3 | text | Градусы, 0°- 360° | Угол между стенами №3 |
| An4 | text | Градусы, 0°- 360° | Угол между стенами №4 |
| W1 | text | Метры, м | Ширина комнаты №1 |
| W2 | text | Метры, м | Ширина комнаты №2 |
| L1 | text | Метры, м | Длина комнаты №1 |
| L2 | text | Метры, м | Длина комнаты №2 |
| H1 | text | Метры, м | Высота комнаты №1 |
| H2 | text | Метры, м | Высота комнаты №2 |
| H3 | text | Метры, м | Высота комнаты №3 |
| H4 | text | Метры, м | Высота комнаты №4 |
| Space\_square | text | Квадратные метры, м^2 | Общая площадь проемов |
| Date | text | - | Время и дата расчёта |

Таблица 2.2 - Выходные данные.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Единица измерения | Обозначение |
| Id | int | - | Ключевое поле |
| User\_id | text | - | Номер пользователя |
| Wall\_square | text | Квадратные метры, м^2 | Площадь стен |
| Ceiling\_perimetr | text | Метры, м | Периметр потолка |
| Ceiling\_area | text | Квадратные метры, м^2 | Площадь потолка |
| Ground\_area | text | Квадратные метры, м^2 | Площадь пола |
| Ground\_perimetr | text | Метры, м | Периметр пола |
| Space\_square | text | Квадратные метры, м^2 | Общая площадь проемов |
| Height\_lenght | text | Метры, м | Общая длина высот |
| Date | text | - | Время и дата расчёта |

Форма регистрации пользователя:

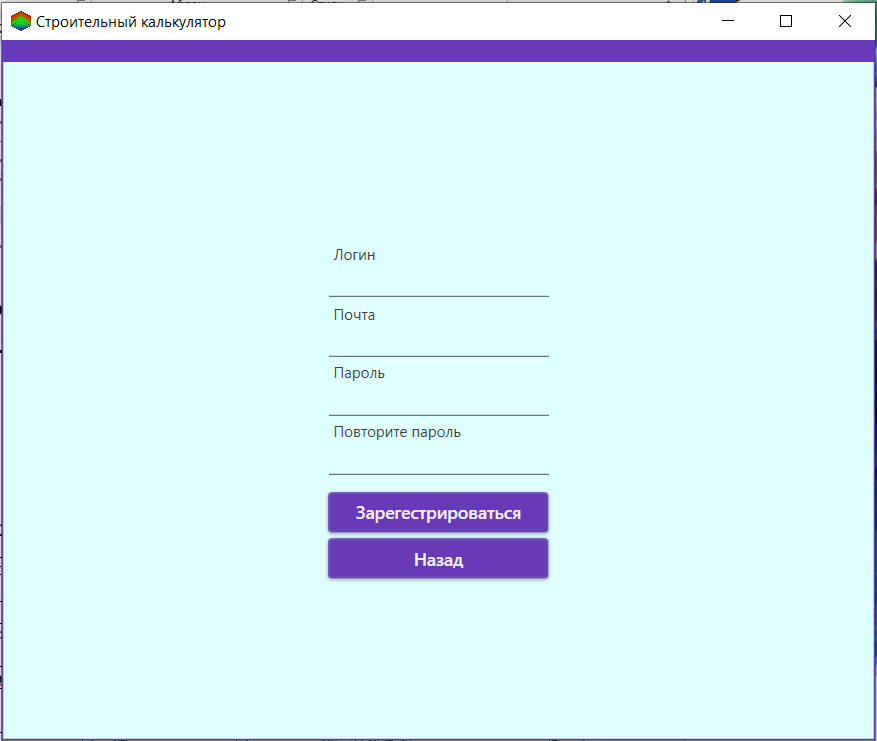


Рисунок 2.5 – Форма регистрации

Код модуля:

<GroupBox x:Name="register\_grid" HorizontalContentAlignment="Stretch" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2" Grid.RowSpan="2" d:IsHidden="True" >

<Grid HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBox x:Name="registration\_login" Height="23" Margin="0,36,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="176" Foreground="Black" Background="#00FF0000"/>

<TextBox x:Name="registration\_email" Height="23" Margin="0,84,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="176"/>

<PasswordBox x:Name="registration\_pass1" Height="23" Margin="0,131,0,0" VerticalAlignment="Top" Width="176"/>

<PasswordBox x:Name="registration\_pass2" Height="23" Margin="0,178,0,0" VerticalAlignment="Top" Width="176"/>

<Button x:Name="register\_btn" Content="Зарегестрироваться" Margin="0,215,0,0" VerticalAlignment="Top" Width="auto" Click="Register\_btn\_Click"/>

<Label x:Name="login\_label" Content="Логин" HorizontalAlignment="Left" Margin="0,12,0,0" VerticalAlignment="Top"/>

<Label x:Name="email\_label" Content="Почта" HorizontalAlignment="Left" Margin="0,60,0,0" VerticalAlignment="Top"/>

<Label x:Name="pass1\_label" Content="Пароль" HorizontalAlignment="Left" Margin="0,107,0,0" VerticalAlignment="Top"/>

<Label x:Name="pass2\_label" Content="Повторите пароль" HorizontalAlignment="Left" Margin="0,154,0,0" VerticalAlignment="Top"/>

<Button Content="Назад" Margin="0,252,0,0" VerticalAlignment="Center" Width="auto" Click="Button\_Click\_7"/>

</Grid>

</GroupBox>

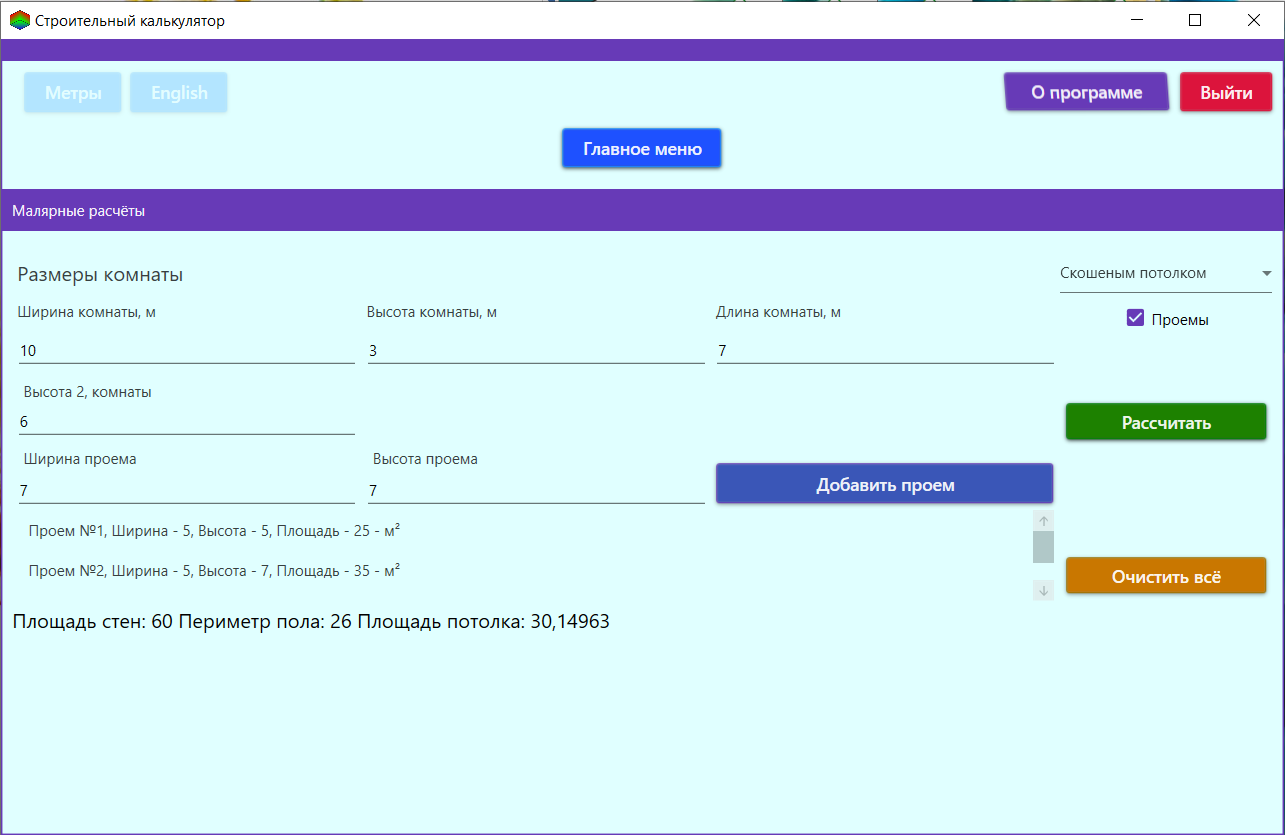


Рисунок 2.6 – Форма «Ввод данных»

Код модуля:

<TextBox x:Name="width\_textbox" Grid.Column="0" Height="23" Margin="5,74,5,0" TextWrapping="Wrap" Text="5" VerticalAlignment="Top"/>

<TextBox x:Name="length\_textbox" Grid.Column="1" Height="23" Margin="5,74,5,0" TextWrapping="Wrap" Text="5" VerticalAlignment="Top"/>

<TextBox x:Name="height\_textbox" Grid.Column="2" Height="23" Margin="5,74,5,0" TextWrapping="Wrap" Text="5" VerticalAlignment="Top"/>

<Label x:Name="width\_label" Content="Ширина комнаты, м" Grid.Column="0" Margin="0,43,0,0" VerticalAlignment="Top" Width="136" HorizontalAlignment="Left"/>

<Label x:Name="height\_label" Content="Высота комнаты, м" Grid.Column="1" Margin="0,43,0,0" VerticalAlignment="Top" Width="141" HorizontalAlignment="Left"/>

<Label x:Name="length\_label" Content="Длина комнаты, м" Grid.Column="2" Margin="0,43,0,0" VerticalAlignment="Top" Width="120" HorizontalAlignment="Left"/>

<Label Content="Размеры комнаты" HorizontalAlignment="Left" Grid.Column="0" Grid.ColumnSpan="3" Margin="0,10,0,0" VerticalAlignment="Top" FontSize="16"/>

Форма вывода истории:

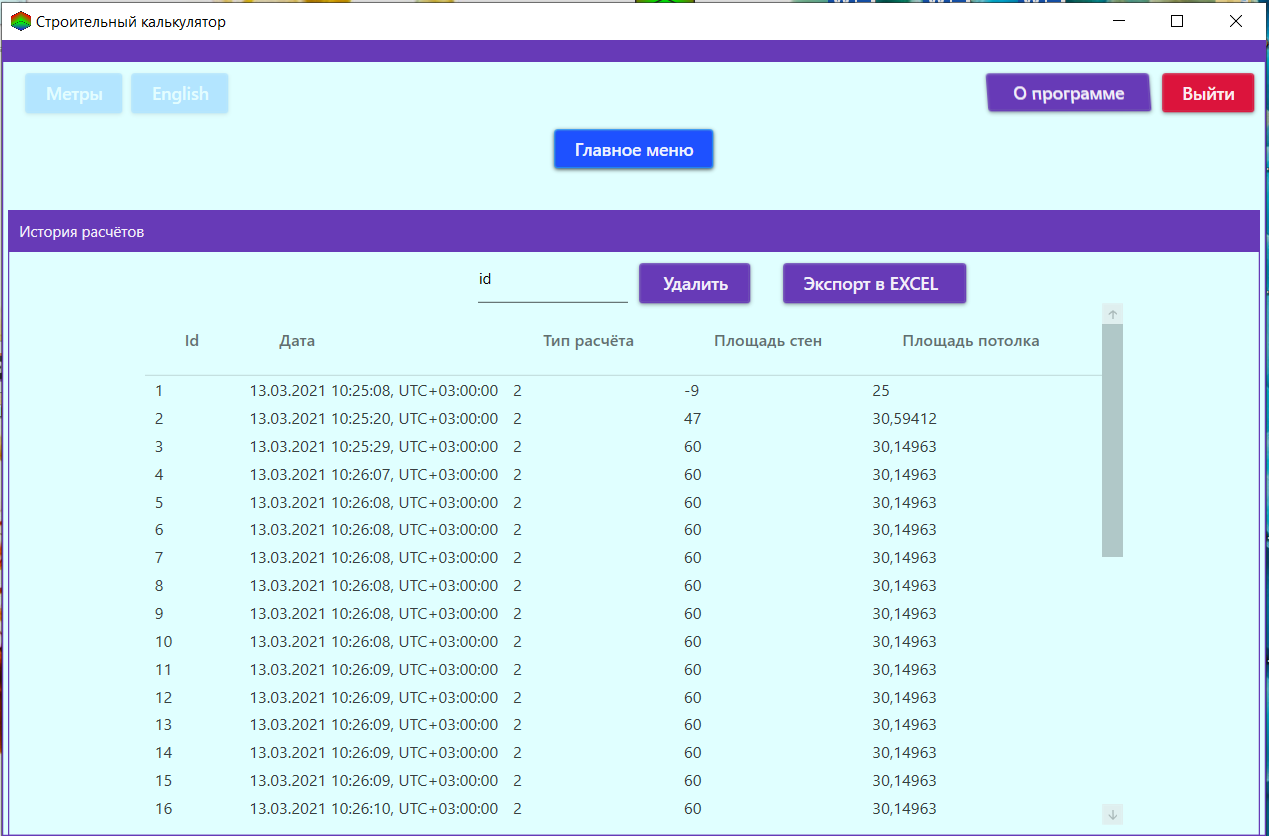


Рисунок 2.7 – Форма «История расчетов»

Код модуля:

public void calcs\_history\_update()

{

List<input\_paints> inputs;

List<output\_paints> outputs;

if (user\_type == 2)

{

inputs = db.input\_paint.ToList();

outputs = db.output\_paint.ToList();

} else

{

inputs = db.input\_paint.Where(b => b.User\_id == current\_user\_id.ToString()).ToList();

outputs = db.output\_paint.Where(b => b.User\_id == current\_user\_id.ToString()).ToList();

}

calcs\_out\_listview.Items.Clear();

foreach (input\_paints input in inputs)

{

int cur\_id = input.id;

int out\_id;

string out\_square = "0", out\_square2 = "0";

foreach (output\_paints output in outputs)

{

if (output.id == cur\_id)

{

out\_id = output.id;

out\_square = output.Wall\_square;

out\_square2 = output.Ceiling\_area;

}

}

calcs\_out\_listview.Items.Add(new MyItem { id\_calc = input.id, date\_calc = input.Date, type\_calc = input.Calc\_type, square\_calc = out\_square, square2\_calc = out\_square2 });

}}

# 2.3 Разработка печатной формы (отчета)

# Была создана кнопка для возможного экспорта всей истории расчётов в таблицу excel. После того, как пользователь произвел расчеты, при необходимости, он может сохранить их в документе формата .xlsx и после отредактировать, и распечатать.

# Рисунок 2.8 – Форма страницы экспорта

# Код модуля:

# void Create(string filePath){string calc\_id, user\_id, calc\_type, an1, an2, an3, an4, w1, w2, l1, l2, h1, h2, h3, h4, space\_square, date;string wall\_square, ceiling\_perimetr, ceiling\_area, ground\_area, ground\_perimetr, height\_length;List<input\_paints> inputs;

# List<output\_paints> outputs;

# if (user\_type == 2)

# {

# inputs = db.input\_paint.ToList();

# outputs = db.output\_paint.ToList();

# }

# else

# {

# inputs = db.input\_paint.Where(b => b.User\_id == current\_user\_id.ToString()).ToList();

# outputs = db.output\_paint.Where(b => b.User\_id == current\_user\_id.ToString()).ToList();

# }

# string a\_col = "A", b\_col = "B", c\_col = "C", d\_col = "D", e\_col = "E", f\_col = "F", g\_col = "G", h\_col = "H";

# string i\_col = "I", j\_col = "J", k\_col = "K", l\_col = "L", m\_col = "M", n\_col = "N", o\_col = "O", p\_col = "P";

# string q\_col = "Q", r\_col = "R", s\_col = "S", t\_col = "T", u\_col = "U", v\_col = "V", w\_col = "W", x\_col = "X";

# int row\_count = 2;

# var wb = new XLWorkbook();

# var ws = wb.Worksheets.Add("Расчёты");

# ws.Cell("A1").Value = "Id Расчета";

# ws.Cell("B1").Value = "Id Пользователя";

# ws.Cell("C1").Value = "Тип расчёта";

# ws.Cell("D1").Value = "Угол №1";

# ws.Cell("E1").Value = "Угол №2";

# ws.Cell("F1").Value = "Угол №3";

# ws.Cell("G1").Value = "Угол №4";

# ws.Cell("H1").Value = "Ширнина №1";

# ws.Cell("I1").Value = "Ширнина №2";

# ws.Cell("J1").Value = "Длина №1";

# ws.Cell("K1").Value = "Длина №2";

# ws.Cell("L1").Value = "Высота №1";

# ws.Cell("M1").Value = "Высота №2";

# ws.Cell("N1").Value = "Высота №3";

# ws.Cell("O1").Value = "Высота №4";

# ws.Cell("P1").Value = "Площадь стен";

# ws.Cell("Q1").Value = "Периметр потолка";

# ws.Cell("R1").Value = "Площадь потолка";

# ws.Cell("S1").Value = "Площадь пола";

# ws.Cell("T1").Value = "Периметр пола";

# ws.Cell("U1").Value = "Площадь проемов";

# ws.Cell("V1").Value = "Общая высота всех углов";

# ws.Cell("W1").Value = "Дата";

# 

# foreach (input\_paints input in inputs)

# {

# int cur\_id = input.id;

# calc\_id = cur\_id.ToString();

# user\_id = current\_user\_id.ToString();

# calc\_type = input.Calc\_type;

# an1 = input.An1;

# an2 = input.An2;

# an3 = input.An3;

# an4 = input.An4;

# w1 = input.W1;

# w2 = input.W2;

# l1 = input.L1;

# l2 = input.L2;

# h1 = input.H1;

# h2 = input.H2;

# h3 = input.H3;

# h4 = input.H4;

# space\_square = input.Space\_square;

# date = input.Date;

# int out\_id;

# string out\_square = "0", out\_square2 = "0";

# foreach (output\_paints output in outputs)

# {

# 

# if (output.id == cur\_id)

# {

# wall\_square = output.Wall\_square;

# ceiling\_perimetr = output.Ceiling\_perimetr;

# ceiling\_area = output.Ceiling\_area;

# ground\_area = output.Ground\_area;

# ground\_perimetr = output.Ground\_perimetr;

# height\_length = output.Height\_lenght;

# a\_col += row\_count.ToString();

# b\_col += row\_count.ToString();

# c\_col += row\_count.ToString();

# d\_col += row\_count.ToString();

# e\_col += row\_count.ToString();

# f\_col += row\_count.ToString();

# g\_col += row\_count.ToString();

# h\_col += row\_count.ToString();

# i\_col += row\_count.ToString();

# j\_col += row\_count.ToString();

# k\_col += row\_count.ToString();

# l\_col += row\_count.ToString();

# m\_col += row\_count.ToString();

# n\_col += row\_count.ToString();

# o\_col += row\_count.ToString();

# p\_col += row\_count.ToString();

# q\_col += row\_count.ToString();

# r\_col += row\_count.ToString();

# s\_col += row\_count.ToString();

# t\_col += row\_count.ToString();

# u\_col += row\_count.ToString();

# v\_col += row\_count.ToString();

# w\_col += row\_count.ToString();

# x\_col += row\_count.ToString();

# ws.Cell(a\_col).Value = cur\_id;

# ws.Cell(b\_col).Value = current\_user\_id;

# ws.Cell(c\_col).Value = calc\_type;

# ws.Cell(d\_col).Value = an1;

# ws.Cell(e\_col).Value = an2;

# ws.Cell(f\_col).Value = an3;

# ws.Cell(g\_col).Value = an4;

# ws.Cell(h\_col).Value = w1;

# ws.Cell(i\_col).Value = w2;

# ws.Cell(j\_col).Value = l1;

# ws.Cell(k\_col).Value = l2;

# ws.Cell(l\_col).Value = h1;

# ws.Cell(m\_col).Value = h2;

# ws.Cell(n\_col).Value = h3;

# ws.Cell(o\_col).Value = h4;

# ws.Cell(p\_col).Value = wall\_square;

# ws.Cell(q\_col).Value = ceiling\_perimetr;

# ws.Cell(r\_col).Value = ceiling\_area;

# ws.Cell(s\_col).Value = ground\_area;

# ws.Cell(t\_col).Value = ground\_perimetr;

# ws.Cell(u\_col).Value = space\_square;

# ws.Cell(v\_col).Value = height\_length;

# ws.Cell(w\_col).Value = date;

# a\_col = "A"; b\_col = "B"; c\_col = "C"; d\_col = "D"; e\_col = "E"; f\_col = "F"; g\_col = "G"; h\_col = "H";

# i\_col = "I"; j\_col = "J"; k\_col = "K"; l\_col = "L"; m\_col = "M"; n\_col = "N"; o\_col = "O"; p\_col = "P";

# q\_col = "Q"; r\_col = "R"; s\_col = "S"; t\_col = "T"; u\_col = "U"; v\_col = "V"; w\_col = "W"; x\_col = "X";

# row\_count++;

# }

# }

# wb.SaveAs(filePath);

# }

# }

# SaveFileDialog saveFileDialog = new SaveFileDialog();

# saveFileDialog.Filter = "Таблица Excel |\*.xlsx";

# if (saveFileDialog.ShowDialog() == true)

# {

# try

# {

# string path2 = saveFileDialog.FileName;

# Create(path2);

# } catch

# {

# 

# }

# }

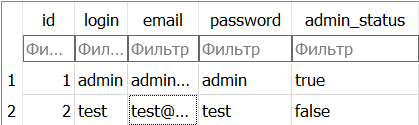
# } }

# 2.4 Разработка ролей и пользователей

Разграничение доступа - это средство защиты от несанкционированного доступа к информации.

Под разграничением доступа понимается система разрешений и запретов на доступ к различным страницам и возможностям системы для различных пользователей. Разграничение доступа построено на трех понятиях – права, роли, и пользователи.

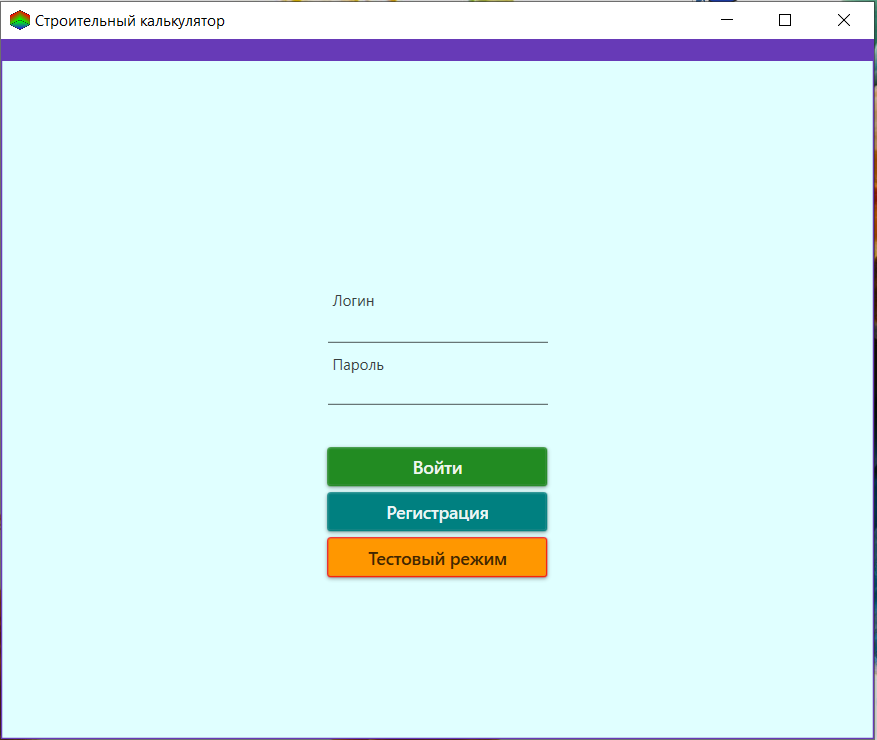
В данной системе существуют 3 роли – «Гость», «Администратор» и «Пользователь» (Рисунок 2.9).



# Рисунок 2.10 – Таблица пользователей

Гостем считается пользователь, который зашёл в «Тестовый режим».

Для получения доступа к некоторым страницам и возможностям системы, пользователям необходимо авторизоваться в специальной вкладке



# Рисунок 2.10 – Форма «Авторизация»

Роли и права для страниц распределены так, что гость имеет право сделать расчёт и посмотреть информацию о программе.

Авторизованный пользователь имеет доступ к возможности сохранять историю расчётов, своему профилю (Рисунок 2.3), просмотр истории расчётов (Рисунок 2.7).

Администратор системы помимо всего остального имеет доступ к управлению пользователями (Рисунок 2.4).

Вся информация о пользователях системы хранится в базе данных, в таблице «users» (Рисунок 2.10). Каждый пользователь защищен паролем.

# 3 Тестирование

# 3.1 Описание контрольного примера

В данном разделе описаны математические вычисления, производимые в программе с конкретными данными. Все расчеты произведены в двух вариантах ручным способом на калькуляторе и при помощи разработанной программы. Полученные результаты проанализированы и исправлены выявленные ошибки в программе.

Расчёт комнаты со скошенным потолком:

Входные данные:

Ширина комнаты(м) = 6

Длина комнаты(м) = 10

Высота комнаты(м) = 2

Высота №2 комнаты(м) = 4

Ширина проема №1(м) = 1,5

Высота проема №1(м) = 1,5

Ширина проема №2(м) = 0,8

Высота проема №2(м) = 2

Расчет:

Общая площадь стен(м2) = 92,55

Периметр потолка(м) = 32,649

Площадь потолка(м2) = 63,245

Площадь пола(м2) = 60

Периметр пола(м) = 32

Общая площадь проемов(м2) = 3,45

Общая высота углов(м) = 12

# 3.2 Тестирование системы

Широко используемыми методами тестирования являются модульное тестирование, интеграционное тестирование, приемочное тестирование, и тестирование системы. Программное обеспечение подвергается этим испытаниям в определенном порядке.

* Модульное тестирование
* Интеграционное тестирование
* Системное тестирование
* Приемочные испытания

Модульное тестирование

В первую очередь проводится модульный тест. Как подсказывает название, это метод испытания на объектном уровне. Отдельные программные компоненты тестируются на наличие ошибок. Для этого теста требуется точное знание программы и каждого установленного модуля. Таким образом, эта проверка осуществляется программистами, а не тестерами. Для этого создаются тест-коды, которые проверяют, ведет ли программное обеспечение себя так, как задумывалось.

Интеграционное тестирование

Отдельные модули, которые уже были подвергнуты модульному тестированию, интегрируются друг с другом, и проверяются на наличие неисправностей. Такой тип тестирования в первую очередь выявляет ошибки интерфейса. Интеграционное тестирование можно осуществлять с помощью подхода "сверху вниз", следуя архитектурному сооружению системы.

Приемочные испытания

Это последний тест, который проводится перед передачей программного обеспечения клиенту. Он проводится, чтобы гарантировать, что программное обеспечение, которое было разработано отвечает всем требованиям заказчика.

Есть несколько основных методов тестирования, которые формируют часть режима тестирования программного обеспечения. Эти тесты обычно считаются самодостаточными в поиске ошибок и багов во всей системе.

Тестирование методом черного ящика осуществляется без каких-либо знаний внутренней работы системы. Тестер будет стимулировать программное обеспечение для пользовательской среды, предоставляя различные входы и тестируя сгенерированные выходы. Этот тест также известен как Black-box, closed-box тестирование или функциональное тестирование.

Тестирование методом "Белого ящика", в отличие от "черного ящика", учитывает внутреннее функционирование и логику работы кода. Для выполнения этого теста, тестер должен иметь знания кода, чтобы узнать точную часть кода, имеющую ошибки.

Тестирование методом серого ящика или Graybox тестирование, это что-то среднее между WhiteBox и BlackBox тестированием, где тестер обладает лишь общими знаниями данного продукта, необходимыми для выполнения теста. Эта проверка осуществляется посредством документации и схемы информационных потоков. Тестирование проводится конечным пользователем, или пользователям, которые представляются как конечные.

Тестирование безопасности Безопасность приложения является одной из главных задач разработчика. Тестирование безопасности проверяет программное обеспечение на обеспечение конфиденциальности, целостности, аутентификации, доступности и безотказности. Индивидуальные испытания проводятся в целях предотвращения несанкционированного доступа в программный код.

Программное обеспечение проверяется на совместимость с внешними интерфейсами, такими как операционные системы, аппаратные платформы, веб-браузеры и т.д. Тест на совместимость проверяет, совместим ли продукт с любой программной платформой.

Таблица 3.1 – Общая информация о тестировании

|  |  |
| --- | --- |
| Название проекта | Настольное ПК приложение «Building calculator» |
| Номер версии | 2.0.1 |
| Имя тестера | Багманов Р.Н. |
| Даты тестирования | 07.03.2021 |

Таблица 3.2 – Проверка процедуры создания аккаунта

|  |  |
| --- | --- |
| Test Case | № 1 |
| Приоритет теста | Высокий |
| Наименование варианта тестирования | Проверка функции регистрации пользователя |
| Резюме испытания | Необходимо проверить корректность выполнения регистрации пользователя в системе. |
| Шаги тестирования | Зайти на страницу регистрации, заполнить все поля и нажать кнопку «Зарегистрироваться». |
| Тестовые данные | Email: ivan@yandex.ru  Логин: ivan1100  Пароль: ivan1100  Повторите пароль: ivan1100 |
| Ожидаемый результат | Данные пользователя должны быть внесены в таблицу «users». |
| Фактический результат | Пользователь внесен в таблицу «users» и получил роль «Пользователь». |
| Предусловия | Совпадение паролей |
| Постусловия | После выполнения теста приложение выводит сообщение об успешной регистрации. |
| Статус | Pass |

Таблица 3.3 – Проверка вывода данных в разделе «История»

|  |  |
| --- | --- |
| Test Case | N\_2 |
| Приоритет теста | Высокий |
| Наименование варианта тестирования | Проверка вывода данных в разделе «История» |
| Резюме испытания | Необходимо проверить правильность заполнения областей и корректность получения данных с помощью запросов фреймворка Entinity |
| Шаги тестирования | Зайти в раздел «История», проверить вывод всех атрибутов. |
| Тестовые данные | Вкладка «История расчетов» |
| Ожидаемый результат | Все данные и области заполняются. |
| Фактический результат | Все данные и области корректно заполняются. |
| Предусловия | Нет |

# Заключение

В результате курсового проектирования была разработана информационная система по строительным расчетам.

Система имеет удобный для пользователя дизайн. ИС разрабатывалась на основе СУБД и включает в себя базу данных и приложение.

В ходе создания данной информационной системы были выполнены следующие задачи:

* развитие практических навыков самостоятельной работы;
* разработано настольное ПК приложение;
* разработана база данных с таблицей в SQLlite;
* изучен формальный язык описания внешнего вида документа, написанного с использованием языка разметки XAML;
* написано настольное ПК приложение с помощью выбранного языка программирования C#.
* приложение собранно пакетным сборщиком Inno setup для дальнейшей установки для других персональных компьютеров.

Во время выполнения данного курсового проекта была достигнута поставленная в начале проектирования цель - систематизированы, расширены и закреплены теоретические знания по специальности. В дальнейшем будут применены полученные знания и навыки при решении конкретных задач.

# Список использованных источников

Государственные стандарты

1. ГОСТ 2.106-96 ЕСКД. Текстовые документы.
2. ГОСТ 2.104-2006 ЕСКД. Основные надписи.
3. ГОСТ 28388-89. Система обработки информации. Документы по носителям данных. Порядок выполнения и обращения.

Нормативная

1. Инструкция №2.100 по охране труда при работе в компьютерных кабинетах.

Учебная

1. Слинкин A.A, Книга Натана Адама «WPF 4. Подробное руководство»; [Символ-Плюс](https://www.ozon.ru/publisher/simvol-plyus-856498/), 2011. - 880 с.
2. Мэтью Мак-Дональд, WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов; Вильямс, 2013. – 1024с.
3. Симдянов, И.В.; Кузнецов, М.В. MySQL; БХВ-Петербург - Москва, 2015. - 747 c.

<https://www.youtube.com/user/PlurrimiTube>

<https://www.youtube.com/watch?v=4_K4MgWHzb8>

# Приложение А

# Руководство пользователя

Разработанное приложение служит для удобного расчета освоения скважин.

Приложение позволяет воспользоваться следующими видами сервисов:

Для пользователей:

* Произведение расчетов;
* Сохранение результатов;
* Экспорт результатов.

Для администратора:

* Управление историей всех пользователей;
* Управление пользователями.

Приложение выполняет функции ввода данных, различного рода расчетов и сохранение результатов в базе данных.

Режим работы приложения не регламентирован. Пользование приложением возможно в любое время суток.

Для работы приложения требуется запустить файл setup\_biulding\_calculator.exe. После установить программу и запустить её.

Откроется главное меню приложения, представленное на рисунке А.1.

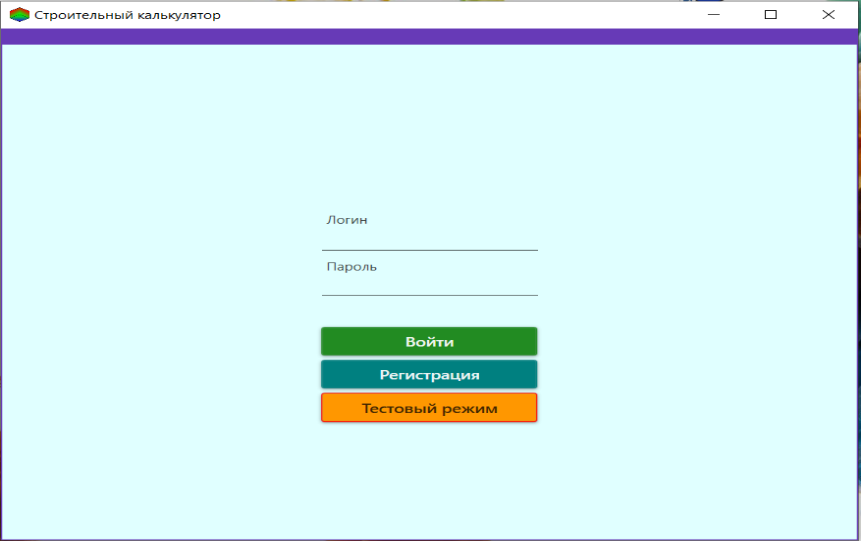


Рисунок А.1 – Главная страница приложения

Для перехода к расчетам необходимо авторизоваться.

Для того, чтобы рассчитать, перейдите в «Главное меню», а затем нажмите на кнопку «Малярные расчеты», далее введите свои данные. После нажатия кнопки «Рассчитать» появятся результаты.

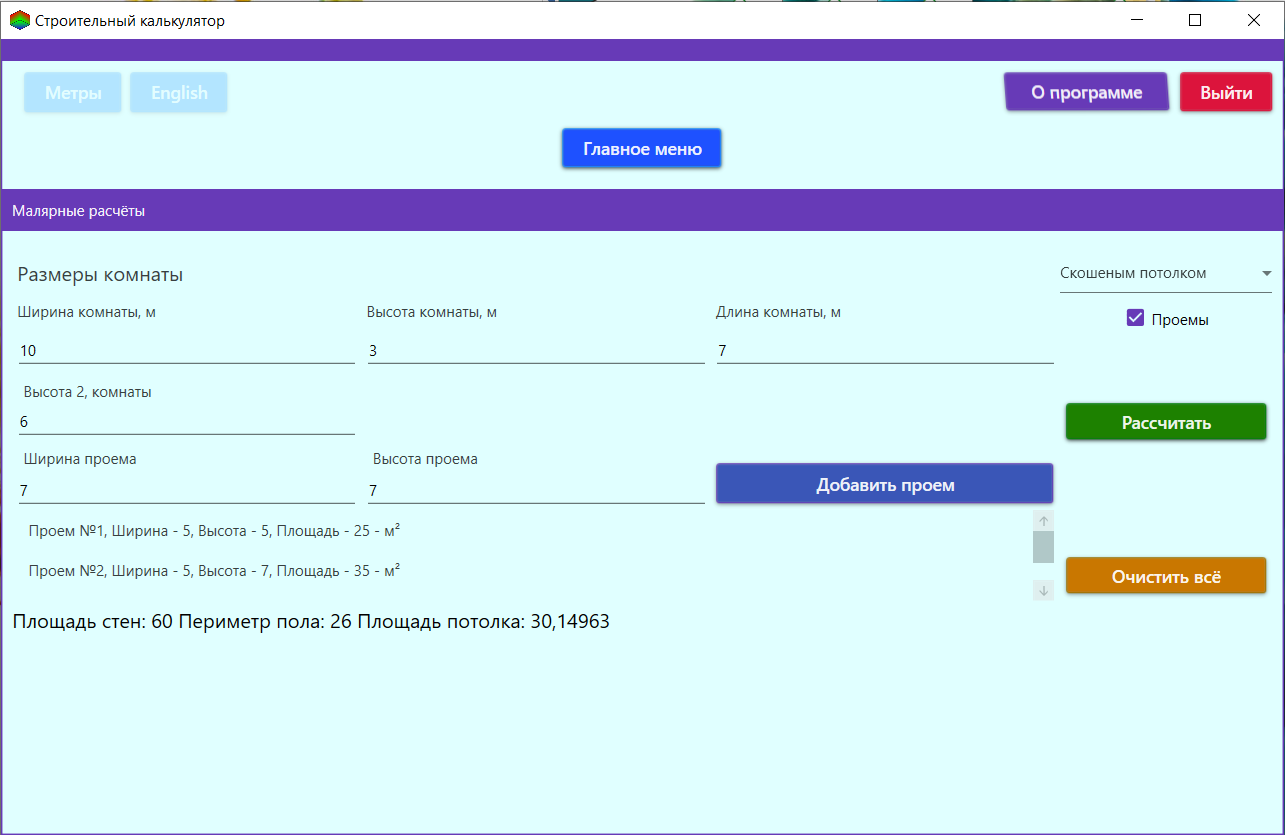


Рисунок А.2 – Форма расчетов

Все произведенные расчеты записываются в базу данных и хранятся в виде истории во вкладке «История расчетов», где их можно просмотреть, очистить, либо экспортировать.

# Приложение Б

# Электронный носитель с программой и пояснительной запиской