ФИЛИАЛ ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРИБОРОСТРОЕНИЯ

И ИНФОРМАТИКИ»

в г. Ставрополе

Кафедра «Автоматизированные системы управления»

«Утверждаю»

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_ В.В. Бережной

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_ 2013 г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

«Базы данных»

на тему: Разработка базы данных гостиницы

Автор курсового проекта \_\_\_\_\_\_\_\_\_\_\_Караваев Вадим Юрьевич

Обозначение курсового проекта КП-МГУПИ-230100.62-ИВТ-111-01

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проект защищен: \_\_\_\_\_\_\_\_\_\_\_Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка, подпись руководителя

Ставрополь, 2013

ФИЛИАЛ ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРИБОРОСТРОЕНИЯ

И ИНФОРМАТИКИ»

в г. Ставрополе

Кафедра «Автоматизированные системы управления»

«Утверждаю»

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_ В.В. Бережной

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_ 2013 г.

**ЗАДАНИЕ**

**ПО КУРСОВОМУ ПРОЕКТИРОВАНИЮ**

Студент Караваев В.Ю Группа ИВТ-111

1. Тема: Разработка базы данных предприятия.

2.Исходные данные к проекту**:**

Реализовать базу данных гостиницы на основе MS SQL сервер

3. Содержание пояснительной записки:

1) теоретический раздел;

2) разработка базы данных.

4. Дата выдачи задания «15» сентября 2013 г.

5. Срок сдачи готового проекта «15» декабря 2013 г.

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А.Авакян

Задание принял к исполнению « » сентября 2013 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оглавление

[ВВЕДЕНИЕ 4](#_Toc375017911)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc375017912)

[1.1 Описание предметной области 5](#_Toc375017913)

[1.2 Словесное описание задачи 6](#_Toc375017914)

[2 КОНЦЕПТУАЛЬНОЕ ПОСТРОЕНИЕ 7](#_Toc375017915)

[2.1 Описание сущностей 7](#_Toc375017916)

[2.2 Описание связей 8](#_Toc375017917)

[2.3 Информационно логическая модель 8](#_Toc375017918)

[3 ИСПОЛЬЗУЕМЫЕ ИНСТРУМЕНТЫ РАЗРАБОТКИ 9](#_Toc375017919)

[3.1 Выбор инструментов разработки 9](#_Toc375017920)

[3.2 Среда разработки – Visual Studio 2012 Express 9](#_Toc375017921)

[3.3 Язык программирования – С# 10](#_Toc375017922)

[3.4 Язык интегрированных запросов – LINQ 11](#_Toc375017923)

[3.5 Графический интерфейс – Windows Presentation Foundation 11](#_Toc375017924)

[3.6 База данных – MS SQL сервер 12](#_Toc375017925)

[3.7 Система контроля версий – Git 13](#_Toc375017926)

[4 ФИЗИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 15](#_Toc375017927)

[4.1 Интерфейс 15](#_Toc375017928)

[4.2 Класс Tables 15](#_Toc375017929)

[4.3 Классы-наследники класса Tables 16](#_Toc375017930)

[4.4 Реализация LINQ запросов 16](#_Toc375017931)

[4.5 Обработка исключительных ситуаций 17](#_Toc375017932)

[4.6 Подключение к локальной БД 18](#_Toc375017933)

[4.7 Класс Msg 18](#_Toc375017934)

[4.8 Построение графа зависимостей 18](#_Toc375017935)

[СПИСОК ЛИТЕРАТУРЫ 19](#_Toc375017936)

[ПРИЛОЖЕНИЕ 1 XAML РАЗМЕТКА ИНТЕРФЕЙСА 20](#_Toc375017937)

[ПРИЛОЖЕНИЕ 2 КЛАСС TABLES 21](#_Toc375017938)

[ПРИЛОЖЕНИЕ 2 НАСЛЕДНИКИ КЛАССА TABLES 23](#_Toc375017939)

[ПРИЛОЖЕНИЕ 3 КЛАСС MSG 29](#_Toc375017941)

[ПРИЛОЖЕНИЕ 4 КЛАСС ОБРАБОТЧИКА ФОРМЫ 30](#_Toc375017942)

# ВВЕДЕНИЕ

На дворе 21 век, миром правят изменчивые IT технологии. Они окружают нас повсюду, мы пользуемся ими на работе, в учебе, в повседневной жизни. И конечно же они не могли обойти стороной сектор предприятий. Замена монотонной рутиной работы нескольких человек на всего один программный комплекс может оказаться очень выгодным вложением. В рамках данного курсового проекта мы разработаем полноценное приложение с помощью которого можно вести учет небольшой гостиницы или отеля. Преимущества внедрения приложения такого рода очевидны. Это снижение количества требуемых бумажных носителей информации (журналы, документы), практически мгновенный доступ практически к любой информации касающейся предприятия, сведение к минимуму различных ошибок учета, облегчение труда персонала.

# 1 ПОСТАНОВКА ЗАДАЧИ

## 1.1 Описание предметной области

Первоначальный (и пожалуй самый ответственный) этап при проектировании абсолютно любого приложение – изучение предметной области. Каким бы ни был совершенным исходный код, какие бы паттерны программирования не использовались – без знания предметной области невозможно создать приложение удовлетворяющее конечного заказчика. Известный программист, автор книг по проектированию и разработке программного обеспечения Стив Макконнелл приводил простой пример наглядно иллюстрирующий данную ситуацию «Если необходимо что бы приложение выполняло операцию А за 1 сек., и были разработаны 2 приложения, одно из которых выполняет операцию B за 0.1 сек., а другое операцию A за 10 сек. то приоритет стоит отдать второму приложению»

Для того что бы разработать программное обеспечение для гостиницы необходимо иметь представление о функционировании, обычном режиме работы среднестатистического отеля/гостиницы. Является хорошим тоном требовать у заказчика подробное ТЗ (Техническое задание) в котором будет наиболее полно описаны требования, желаемый функционал, сроки разработки, условия оплаты. Обычно в ТЗ также заранее обговариваются предписания на случай если заказчик решит изменить свои требования по поводу разработки и функционала.

Так как в нашем случае заказчик как таковой отсутствует, изучение предметной области и составление требований, сроков разработки ложится непосредственно на плечи и совесть разработчика.

## 1.2 Словесное описание задачи

Словесное описание задачи является неотъемлемой частью ТЗ. Заказчик не зная ни одного языка программирования, не имея никаких познаний в разработке приложений вполне может очень ясно и понятно сформулировать желаемый функционал приложения. Заказчик на обычном естественном языке формулирует «ЧТО» он желает получить, разработчик в свою очередь занимается реализацией. Словесное нестрогое описание задачи данного курсового проекта выглядит следующим образом:

«Разработать приложения для управления гостиницей. Обязательным условием является использование базы данных MS SQL. Основная цель приложения – автоматизация учета гостиницы.»

# 2 КОНЦЕПТУАЛЬНОЕ ПОСТРОЕНИЕ

## 2.1 Описание сущностей

В проекте «Администратор гостиничного комплекса» в соответствии с предметной областью были определены следующие сущности:

• Таблица "Номера" (поля "Идентификатор (этаж + номер, пример 412 –12 номер на 4 этаже, первичный ключ)", "Класс", "Количество мест", "Статус (до какого числа предположительно занято, если свободно то NULL)", «Стоимость номера»)

• Таблица "Сессии" поля «ID заселения» (первичный ключ, «Дата по которую оплачено», «Дата вселения» «Стоимость текущего заселения» (стоимость номера + стоимость всех услуг, «Занимаемый номер» (внешний ключ), «ФИО» (обязательное поле) «Пол» (по умолчанию – 'м' так как мужчины чаще бывают в командировках и разъездах)

• Таблица «Персонал» (поля «ID сотрудника» (первичный ключ), «ФИО», «Пол», «Дата рождения», «занимаемая должность», «ID Категории услуг» (внешний ключ))

• Таблица «Категория услуг» («ID категории», «Название категории»)

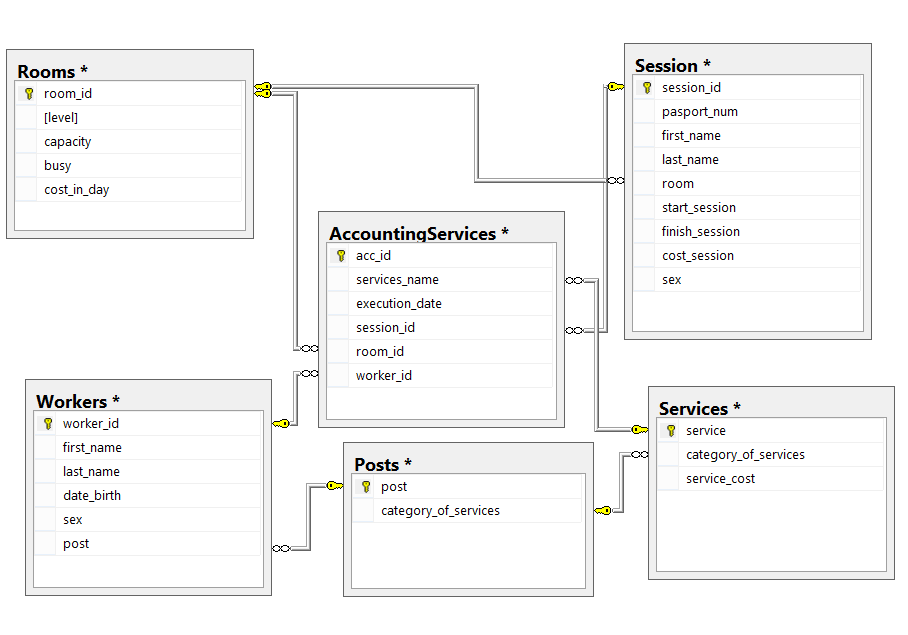
• Таблица «Услуги» (поля «название услуги» (первичный ключ), «Категория услуг», «Стоимость услуги»)

• Таблица «Учет услуг» (поля «ID услуги» (имеется в виду конкретная услуга примененная в момент времени к конкретному номеру/клиенту, т.е не «Завтрак», а “Завтрак который заказал Виталий Петрович 30 февраля”) «Название услуги», «Дата выполнения услуги», «Клиент», «Номер», « ID Сотрудника»)

## 2.2 Описание связей

|  |  |  |  |
| --- | --- | --- | --- |
| Родительская таблица | Дочерняя таблица | Ключи | Вид связи |
| Номер | Учет услуг | Идентификатор Номер | 1:М |
| Сессии | Номера | Занимаемый номер Идентификатор | 1:1 |
| Категория услуг | Услуги | Название категории  Категория услуг | 1:М |
| Персонал | Категория услуг | ID категории услуг ID категории | 1:1 |
| Учет услуг | Постояльцы | Клиент  Номер паспорта | 1:М |
| Учет услуг | Персонал | ID сотрудника ID сотрудника | 1:1 |

## 2.3 Информационно логическая модель



# 3 ИСПОЛЬЗУЕМЫЕ ИНСТРУМЕНТЫ РАЗРАБОТКИ

## 3.1 Выбор инструментов разработки

Выбор правильных инструментов разработки является определяющим для таких параметров приложения как: длительность и стоимость разработки, сложность и стоимость сопровождения, стоимость дальнейшей модернизации приложения. Одной из самых больших ошибок является использование непроверенной технологии, с которой разработчик мало знаком. Инструмент должен быть проверен временем, хорошо зарекомендован и знаком разработчику.

## 3.2 Среда разработки – Visual Studio 2012 Express

Microsoft Visual Studio несомненно является одним из лидеров на рынке IDE. Подсветка кода, автодополнение (Intellisence), средства рефакторинга, полная поддержка технологий .NET, средства совместной разработки (TFS), интеграция с системами контроля версий. В этой IDE есть все для создания полноценного приложения для платформы Windows (а также Windows Phone, и Android при условии использования Xamarin). Express версия является бесплатной и свободно распространяемой.

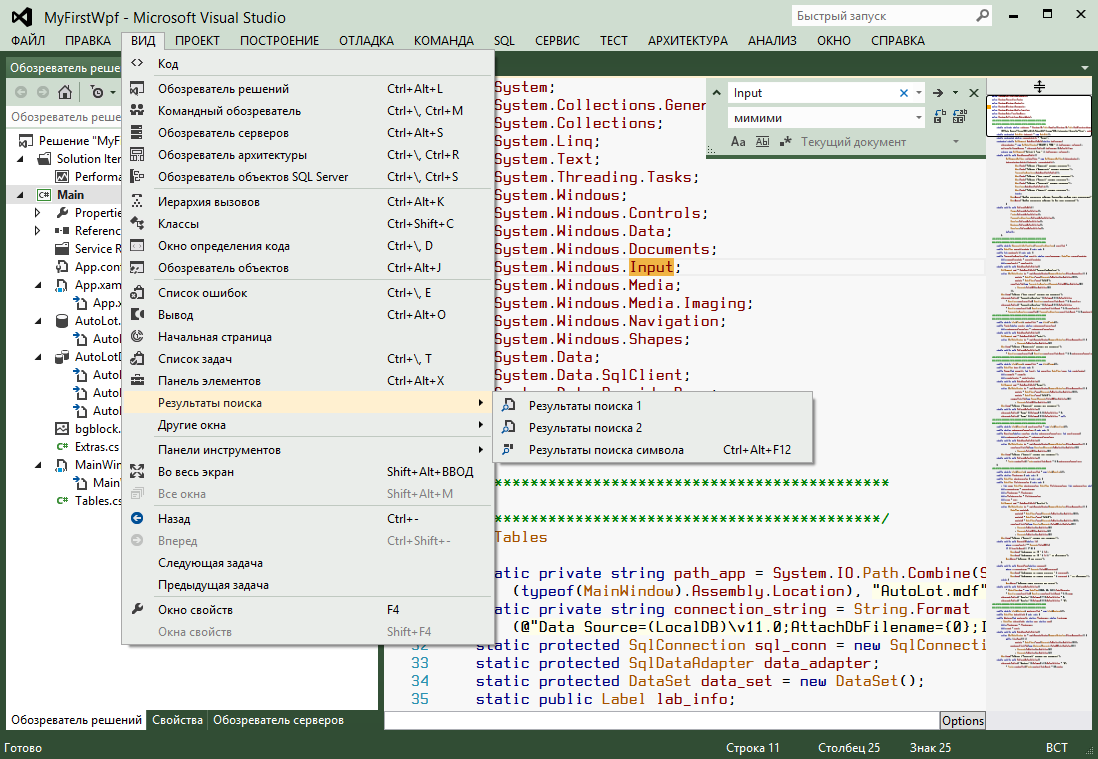


Рисунок 1 – Общий вид IDE

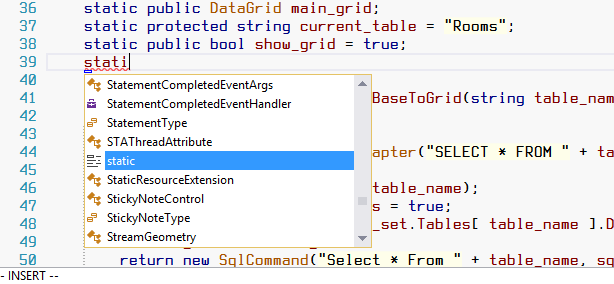


Рисунок 2 - Автодополнение Intellisence

## 3.3 Язык программирования – С#

C# является объектно-ориентированным языком, с мощным С-подобным синтаксисом и отличной документацией которую можно найти на ресурсе http://msdn.microsoft.com/ru-RU/. Разработан компанией Microsoft в 2000 году в качестве аналога языку Java (имеется в виду компиляция в байт-код, а также некоторые синтаксические особенности). Текущая версия 5.0, язык активно развивается.

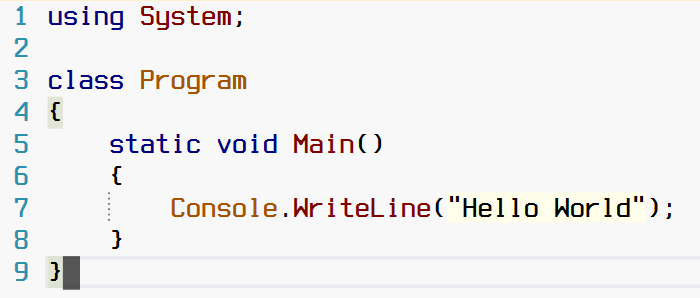


Рисунок 3 – Реализация программы “Hello World” на языке C#

## 3.4 Язык интегрированных запросов – LINQ

Language Integrated Query (LINQ) является встроенным в С# с версии 3.0 языковым средством, целью которого является осуществление запросов. Синтаксис LINQ отдаленно напоминает синтаксис SQL, что неудивительно учитывая схожее предназначение этих языков. В большинстве случаев код написанный с использованием LINQ короче, проще и понятнее, нежели при использовании чистого кода C#. На примере ниже представлен пример кода выводящий на экран числа больше нуля. Выборка результатов происходит посредством построения и дальнейшего вызова LINQ запроса.

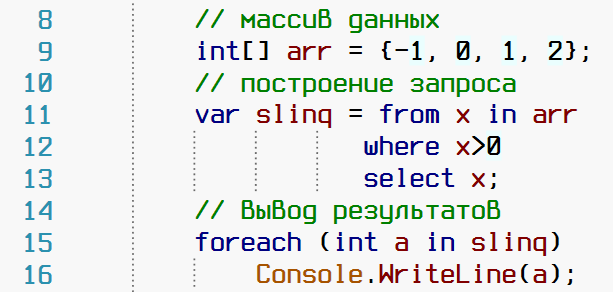


Рисунок 4 – Простейший LINQ запрос

## 3.5 Графический интерфейс – Windows Presentation Foundation

WPF пришла на смену устаревшему Windows Forms. Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Таким образом производительность WPF гораздо выше. В качестве языка разметки используется XAML. Можно также использовать и редактор WYSIWYG (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et, «что вижу, то и получаю») аналогичный тому, который использовался при построении приложений в основе графического интерфейса которых лежит Windows Forms.

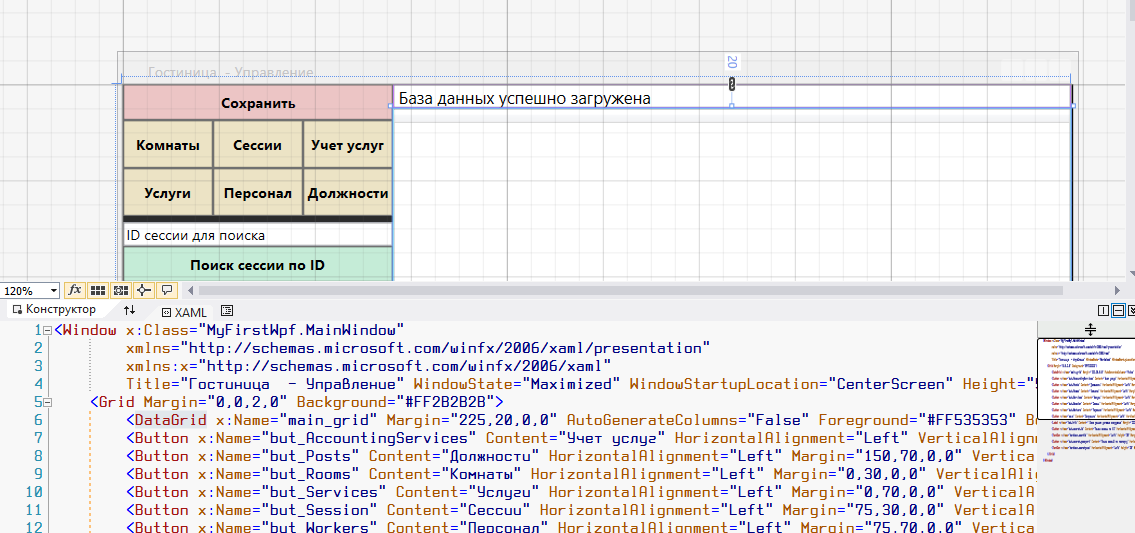


Рисунок 5 – Создание интерфейса WPF-приложения

## 3.6 База данных – MS SQL сервер

MS SQL Server – система управления реляционными базами данных. Разработана компанией Microsoft. Основной используемый язык запросов Transact-SQL. В нашем приложении для обеспечения переносимости будет использоваться локальная база данных, не требующая предустановленного MS SQL Server.

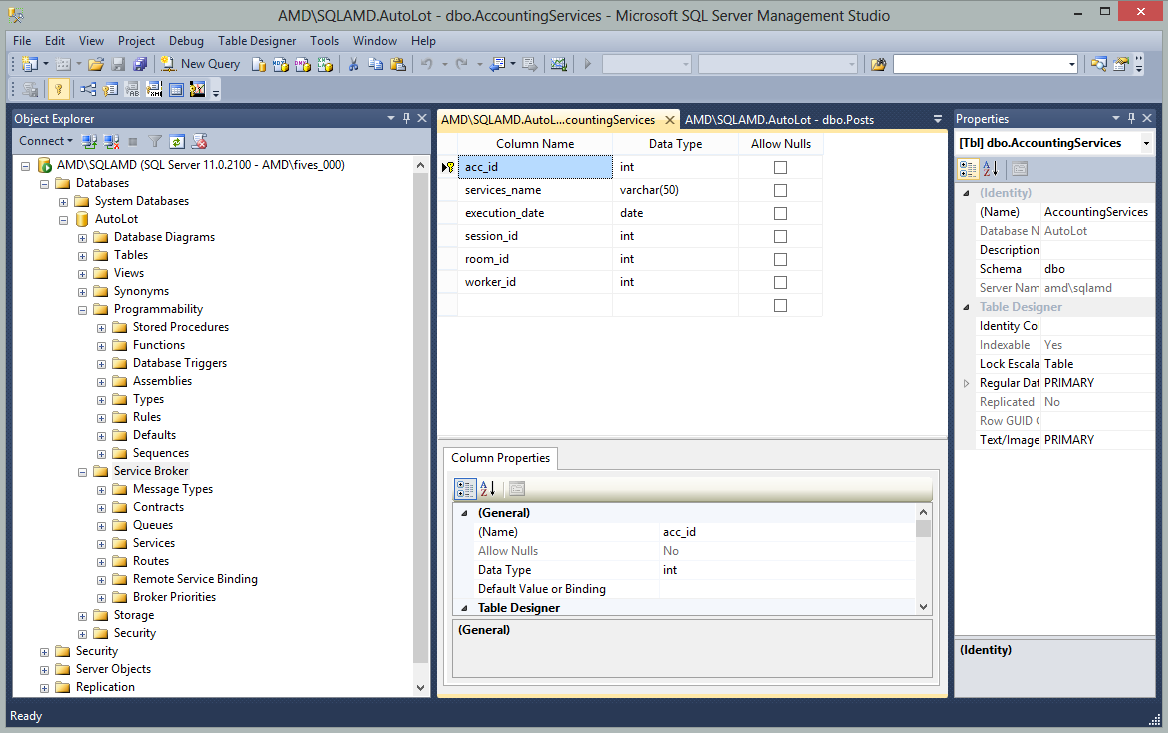


Рисунок 6 – SQL Server Management Studio

## 3.7 Система контроля версий – Git

Построение сложных программных комплексов требует мощного инструмента для автоматизации таких важных операций как: одновременная работа над проектом нескольких разработчиков и безболезненное слияние результатов их работы (merge), возможность отката на предыдущие версии проекта, возможность создавать задачи (tasks) и делать багрепорты (issues), нелишней является возможность быстрого создания ответвлений (forks). Git — распределённая система управления версиями файлов и великолепно зарекомендовавшая себя в большом количестве проектов.

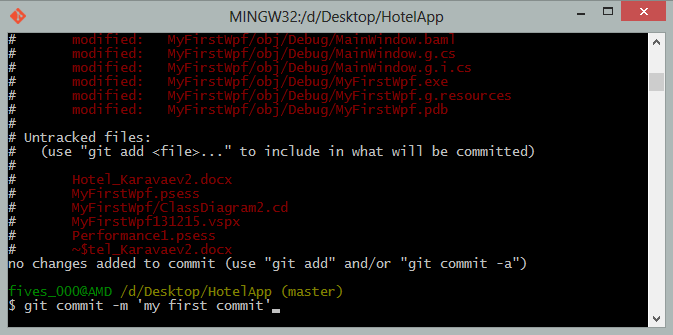


Рисунок 7 – Консоль управления Git

Для хостинга курсового проекта используется ресурс http://github.com, на данный момент являющийся самым крупным хостингом для IT-проектов (особенно open source). На текущий момент репозиторий находится по адресу https://github.com/5665tm/HotelApp. В репозитории можно посмотреть предыдущие версии проекта, сравнить изменения между версиями, посмотреть исходный код и многое другое.

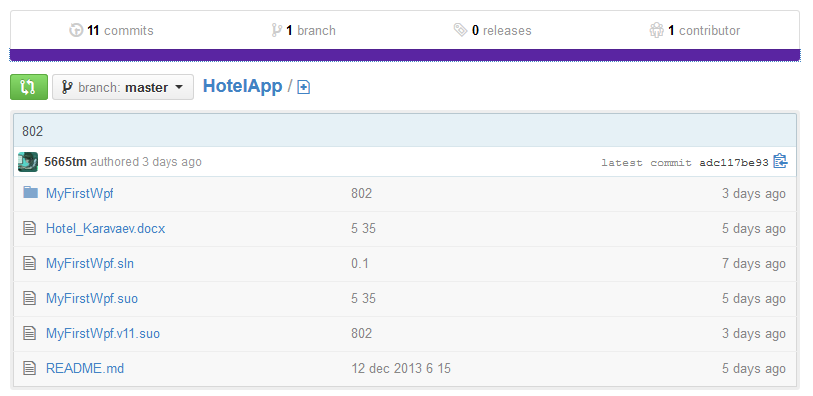


Рисунок 8 – Главная страница репозитория

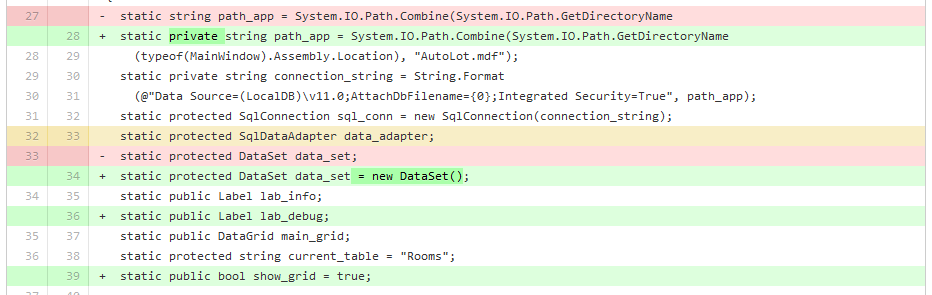


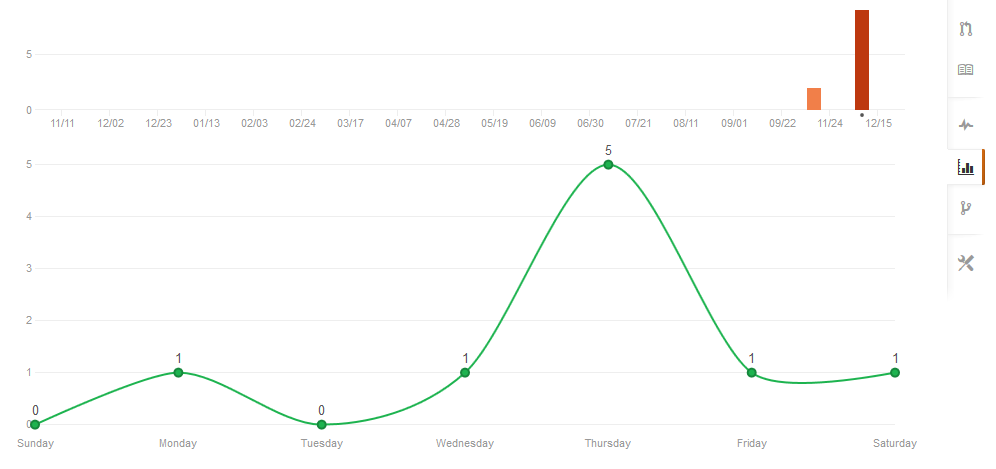
Рисунок 9 – Просмотр изменений между коммитами

Рисунок 10 – Распределение коммитов по дням

# 4 ФИЗИЧЕСКАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

Рассмотрение и полный разбор создания приложения займет несколько десятков страниц, поэтому не будем рассматривать заведомо очевидные моменты, и рассмотрим лишь ключевые и требующие особого пояснения.

## 4.1 Интерфейс

Основная задача была построить простой, лаконичный и интуитивно понятный пользователю интерфейс.

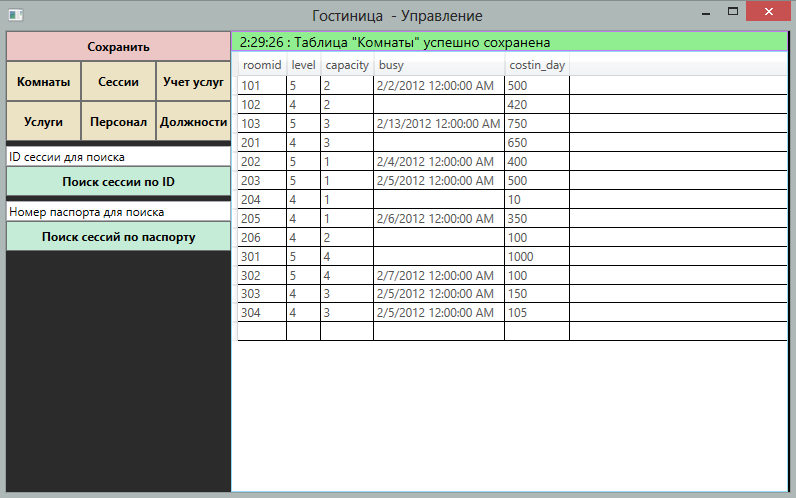


Рисунок 11 – Интерфейс пользователя

Вся работа свелась к редактированию по типу WYSIWYG, непосредственное редактирование кода XAML практически не понадобилось. Код XAML приведен в приложении 1

## 4.2 Класс Tables

Класс Tables является суперклассом (родительским) для всех классов представляющих собой таблицы. В классе определено подключение к базе данных, сохранение результатов, обновление данных в памяти. Листинг класса Tables приведен в приложении 2

## 4.3 Классы-наследники класса Tables

Классы-наследники класса Tables (а их 6, по числу таблиц – Rooms, AccountingServices, Posts, Services, Personal, Session) предопределяют собой каждую из записей в соответствующей таблице. Общим для них является наличие статитической (общей для всех объектов) коллекции типа List<T> в которой хранятся все записи конкретной таблицы. В каждом классе помимо конструктора реализованы следующие методы:

* DataBaseTableToList – выполняет запись из БД в List<T>
* RefreshDefaultValue – выполняет обновление значений по умолчанию

Также в некоторых классах реализованы LINQ запросы для выборки данных. Листинг приведен в приложении 3.

## 4.4 Реализация LINQ запросов

Как уже было сказано в разделе 3.4 LINQ запросы обычно короче чистого кода C#. Реализация поиска пользователя по паспорту без использования LINQ запросов.

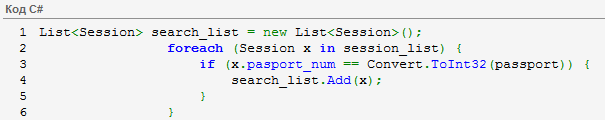


Рисунок 12 – Поиск по паспорту без LINQ

В зависимости от стиля фигурных скобок (K&R либо классический) реализация занимает 6-8 строк, не говоря о том что читабельность такого кода оставляет желать лучшего. Реализация на LINQ занимает 3 строки и является гораздо более наглядной по своей структуре:

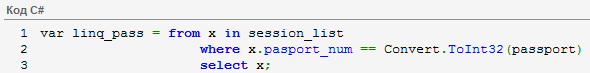


Рисунок 13 – LINQ запрос для поиска по паспорту

## 4.5 Обработка исключительных ситуаций

В результате работы приложения могут возникать так называемые «исключительные ситуации». Например причиной возникновения исключительной ситуации DivideByZeroException является деление на ноль. К счастью в языковые средства C# включен средства для обработки исключительных ситуаций при помощи кодовых блоков try и catch. К тому же возникновение и перехват исключительных ситуаций идеальный инструмент для проверки корректных действий пользователя. В приложении такой механизм встроен в нескольких местах. Ниже приведен листинг для предотвращения сохранения пользователем в БД нескольких записей с одинаковым первичным ключем, либо записей с незаполненными полями.

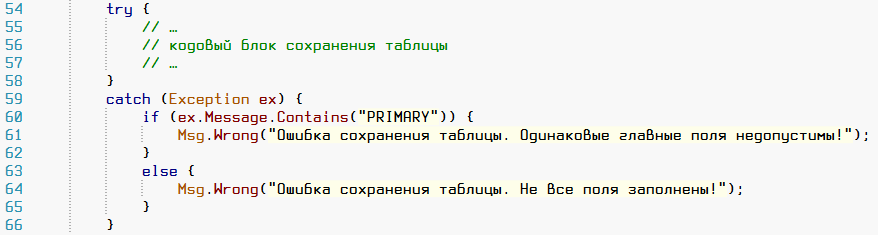


Рисунок 14 – Механизм обработки исключительных ситуаций

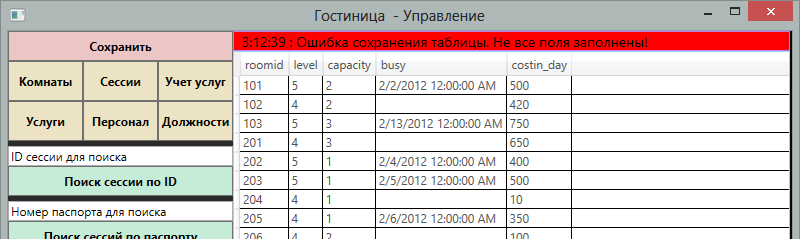


Рисунок 15 – Предупреждение пользователю

## 4.6 Подключение к локальной БД

Для обеспечения лучшей переносимости приложения было принято решение использовать локальную базу данных в формате \*.MDF. Подключение такой базы данных осуществляется следующим образом.

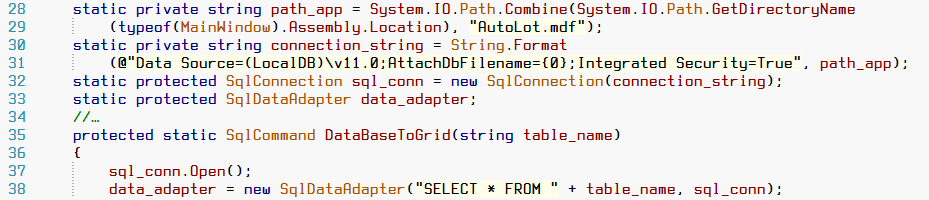


Рисунок 16 – Подключение к локальной БД AutoLot.mdf

## 4.7 Класс Msg

В классе Msg предопределены методы для вывода сообщений на экран:

* Norm – обычное сообщение
* Wrong – сообщение об ошибке
* Ready – успешное завершение операции

## 4.8 Построение графа зависимостей

Visual Studio располагает встроенными средствами для построения диаграмм зависимостей. Часто такие диаграммы помогают лучше понять структуру программы стороннему разработчику.

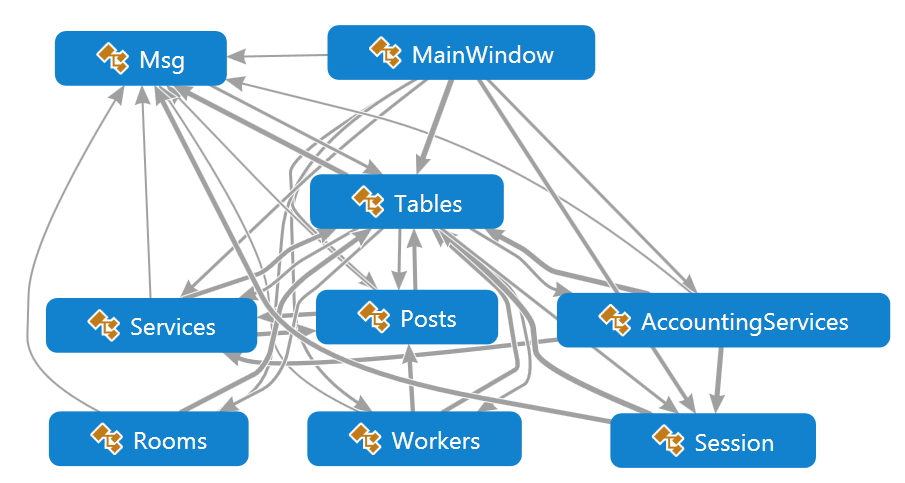


Рисунок 17 –Диаграмма зависимостей

# СПИСОК ЛИТЕРАТУРЫ

1. Ицик Бен Ган. Microsoft SQL Server. Основы T-SQL, 2009.
2. Макконнелл Стив. Совершенный код, 2010.
3. Шилдт Герберт. С# 4.0 Полный справочник, 2011.
4. Троэлсен Эндрю. Язык С# 5.0 и платформа .NET 4.5, 2013.

# ПРИЛОЖЕНИЕ 1 XAML РАЗМЕТКА ИНТЕРФЕЙСА

<Window x:Class="MyFirstWpf.MainWindow"

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

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

Title="Гостиница - Управление" WindowState="Maximized" WindowStartupLocation="CenterScreen" Height="500" MinWidth="800" MinHeight="500" Width="800" Background="#FF060606">

<Grid Margin="0,0,2,0" Background="#FF2B2B2B">

<DataGrid x:Name="main\_grid" Margin="225,20,0,0" AutoGenerateColumns="False" Foreground="#FF535353" BorderBrush="#FF93DBF5" Background="White" Cursor="" CanUserResizeRows="False" RowEditEnding="Rowing"/>

<Button x:Name="but\_AccountingServices" Content="Учет услуг" HorizontalAlignment="Left" VerticalAlignment="Top" Width="75" Click="but\_AccountingServices\_Click" Height="40" Background="#FFECE3C5" FontWeight="Bold" Margin="150,30,0,0"/>

<Button x:Name="but\_Posts" Content="Должности" HorizontalAlignment="Left" Margin="150,70,0,0" VerticalAlignment="Top" Width="75" RenderTransformOrigin="0.667,0.45" Click="but\_Posts\_Click" Height="40" Background="#FFECE3C5" FontWeight="Bold"/>

<Button x:Name="but\_Rooms" Content="Комнаты" HorizontalAlignment="Left" Margin="0,30,0,0" VerticalAlignment="Top" Width="75" Click="but\_Rooms\_Click" Height="40" Background="#FFECE3C5" FontWeight="Bold"/>

<Button x:Name="but\_Services" Content="Услуги" HorizontalAlignment="Left" Margin="0,70,0,0" VerticalAlignment="Top" Width="75" Click="but\_Services\_Click" Height="40" Background="#FFECE3C5" FontWeight="Bold"/>

<Button x:Name="but\_Session" Content="Сессии" HorizontalAlignment="Left" Margin="75,30,0,0" VerticalAlignment="Top" Width="75" Click="but\_Session\_Click" Height="40" Background="#FFECE3C5" FontWeight="Bold"/>

<Button x:Name="but\_Workers" Content="Персонал" HorizontalAlignment="Left" Margin="75,70,0,0" VerticalAlignment="Top" Width="75" Click="but\_Workers\_Click" Height="40" Background="#FFECE3C5" FontWeight="Bold"/>

<Button x:Name="save" Content="Сохранить" HorizontalAlignment="Left" VerticalAlignment="Top" Width="225" Click="save\_Click" Height="30" Background="#FFECC5C5" FontWeight="Bold"/>

<Label x:Name="lab\_info" Content=" База данных успешно загружена" Margin="225,0,0,0" FontSize="14" Padding="5,5,5,-5" Foreground="Black" HorizontalContentAlignment="Stretch" VerticalContentAlignment="Stretch" RenderTransformOrigin="0,0" BorderThickness="1" BorderBrush="#FFBE7CF5" Background="White" Height="20" VerticalAlignment="Top"/>

<Button x:Name="but\_search\_id" Content="Поиск сессии по ID" HorizontalAlignment="Left" Margin="0,135,0,0" VerticalAlignment="Top" Width="225" Height="30" FontWeight="Bold" Background="#FFC5ECD7" BorderThickness="1" BorderBrush="#FF707070" Click="but\_search\_id\_click"/>

<TextBox x:Name="textbox\_searhid" HorizontalAlignment="Left" Height="20" Margin="0,115,0,0" TextWrapping="Wrap" Text="ID сессии для поиска" VerticalAlignment="Top" Width="225" VerticalContentAlignment="Center"/>

<Button x:Name="but\_search\_passport" Content="Поиск сессий по паспорту" HorizontalAlignment="Left" Margin="0,190,0,0" VerticalAlignment="Top" Width="225" Height="30" FontWeight="Bold" Background="#FFC5ECD7" BorderThickness="1" BorderBrush="#FF707070" Click="but\_search\_pass\_click"/>

<TextBox x:Name="textbox\_searchpass" HorizontalAlignment="Left" Height="20" Margin="0,170,0,0" TextWrapping="Wrap" Text="Номер паспорта для поиска" VerticalAlignment="Top" Width="225" VerticalContentAlignment="Center"/>

</Grid>

</Window>

# ПРИЛОЖЕНИЕ 2 КЛАСС TABLES

class Tables

{

static private string path\_app = System.IO.Path.Combine(System.IO.Path.GetDirectoryName

(typeof(MainWindow).Assembly.Location), "AutoLot.mdf");

static private string connection\_string = String.Format

(@"Data Source=(LocalDB)\v11.0;AttachDbFilename={0};Integrated Security=True", path\_app);

static protected SqlConnection sql\_conn = new SqlConnection(connection\_string);

static protected SqlDataAdapter data\_adapter;

static protected DataSet data\_set = new DataSet();

static public Label lab\_info;

static public DataGrid main\_grid;

static protected string current\_table = "Rooms";

static public bool show\_grid = true;

protected static SqlCommand DataBaseToGrid(string table\_name)

{

sql\_conn.Open();

data\_adapter = new SqlDataAdapter("SELECT \* FROM " + table\_name, sql\_conn);

data\_set = new DataSet();

data\_adapter.Fill(data\_set, table\_name);

main\_grid.AutoGenerateColumns = true;

main\_grid.ItemsSource = data\_set.Tables[ table\_name ].DefaultView;

current\_table = table\_name;

return new SqlCommand("Select \* From " + table\_name, sql\_conn);

}

static public void SaveCurrentTable()

{

try {

SqlCommandBuilder sql\_builder = new SqlCommandBuilder(data\_adapter);

sql\_builder.QuotePrefix = "[";

sql\_builder.QuoteSuffix = "]";

data\_adapter.Update(data\_set, current\_table);

switch (current\_table) {

case "Rooms":

Rooms.DataBaseTableToList();

Msg.Ready("Таблица \"Комнаты\" успешно сохранена");

break;

case "Posts":

Posts.DataBaseTableToList();

Msg.Ready("Таблица \"Должности\" успешно сохранена");

break;

case "AccountingServices":

AccountingServices.DataBaseTableToList();

Msg.Ready("Таблица \"Учет услуг\" успешно сохранена");

break;

case "Session":

Session.DataBaseTableToList();

Msg.Ready("Таблица \"Сессии\" успешно сохранена");

break;

case "Workers":

Workers.DataBaseTableToList();

Msg.Ready("Таблица \"Персонал\" успешно сохранена");

break;

case "Services":

Services.DataBaseTableToList();

Msg.Ready("Таблица \"Услуги\" успешно сохранена");

break;

default:

break;

}

}

catch (Exception ex) {

if (ex.Message.Contains("PRIMARY")) {

Msg.Wrong("Ошибка сохранения таблицы. Одинаковые главные поля недопустимо!");

}

else {

Msg.Wrong("Ошибка сохранения таблицы. Не все поля заполнены!");

}

}

}

static public void RefreshToEdit()

{

switch(current\_table) {

case "Rooms":

Rooms.RefreshDefaultValue();

break;

case "Posts":

Posts.RefreshDefaultValue();

break;

case "AccountingServices":

AccountingServices.RefreshDefaultValue();

break;

case "Session":

Session.RefreshDefaultValue();

break;

case "Workers":

Workers.RefreshDefaultValue();

break;

case "Services":

Services.RefreshDefaultValue();

break;

default:

break;

}

}

}

# ПРИЛОЖЕНИЕ 2 НАСЛЕДНИКИ КЛАССА TABLES

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ACCOUNTING SERVICES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class AccountingServices : Tables

{

public static List<AccountingServices> acc\_list =

new List<AccountingServices>();

public int acc\_id { get; set; }

public string services\_name { get; set; }

public DateTime execution\_date { get; set; }

public int session\_id { get; set; }

public int room\_id { get; set; }

public int worker\_id { get; set; }

public AccountingServices(int acc\_id, string services\_name, DateTime execution\_date

, int session\_id, int room\_id, int worker\_id)

{

this.acc\_id = acc\_id;

this.services\_name = services\_name;

this.execution\_date = execution\_date;

this.session\_id = session\_id;

this.room\_id = room\_id;

this.worker\_id = worker\_id;

}

static public void DataBaseTableToList()

{

acc\_list.Clear();

SqlCommand cmd = DataBaseToGrid("AccountingServices");

using (SqlDataReader dr = cmd.ExecuteReader(CommandBehavior.CloseConnection)) {

while (dr.Read()) {

DateTime mydata;

try {

mydata = DateTime.Parse(Convert.ToString(dr.GetValue(2)));

}

catch {

mydata = DateTime.Parse("1/1/0");

}

acc\_list.Add(new AccountingServices(Convert.ToInt32(dr.GetValue(0))

, Convert.ToString(dr.GetValue(1))

, mydata

, Convert.ToInt32(dr.GetValue(3))

, Convert.ToInt32(dr.GetValue(4))

, Convert.ToInt32(dr.GetValue(5))

));

}

}

Msg.Norm("Таблица \"Учет услуг\" открыта для просмотра");

}

static public void RefreshDefaultValue()

{

data\_set.Tables[ "AccountingServices" ].Columns[ 0 ].DefaultValue

= (DateTime.Now - new DateTime(2013, 12, 12)).TotalSeconds;

data\_set.Tables[ "AccountingServices" ].Columns[ 1 ].DefaultValue

= Services.services\_list[ Services.services\_list.Count - 1 ].service;

data\_set.Tables[ "AccountingServices" ].Columns[ 2 ].DefaultValue

= DateTime.Now;

data\_set.Tables[ "AccountingServices" ].Columns[ 3 ].DefaultValue

= Session.session\_list[ Session.session\_list.Count - 1 ].session\_id;

data\_set.Tables[ "AccountingServices" ].Columns[ 4 ].DefaultValue

= Session.session\_list[ Session.session\_list.Count - 1 ].room;

data\_set.Tables[ "AccountingServices" ].Columns[ 5 ].DefaultValue

= AccountingServices.acc\_list[ AccountingServices.acc\_list.Count - 1 ].worker\_id;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

POSTS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Posts : Tables

{

public static List<Posts> posts\_list = new List<Posts>();

public string posts { get; set; }

public string category\_of\_services { get; set; }

public Posts(string posts, string category\_of\_services)

{

this.posts = posts;

this.category\_of\_services = category\_of\_services;

}

static public void DataBaseTableToList()

{

SqlCommand cmd = DataBaseToGrid("Posts");

posts\_list.Clear();

using (SqlDataReader dr = cmd.ExecuteReader(CommandBehavior.CloseConnection)) {

while (dr.Read()) {

posts\_list.Add(new Posts(Convert.ToString(dr.GetValue(0))

, Convert.ToString(dr.GetValue(1))

));

}

}

Msg.Norm("Таблица \"Должности\" открыта для просмотра");

}

static public void RefreshDefaultValue()

{

data\_set.Tables[ "Posts" ].Columns[ 1 ].DefaultValue

= Services.services\_list[ Services.services\_list.Count - 1 ].category\_of\_services;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ROOMS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Rooms : Tables

{

public static List<Rooms> rooms\_list = new List<Rooms>();

public int room\_id { get; set; }

public int level { get; set; }

public int capacity { get; set; }

public DateTime busy { get; set; }

public int cost\_in\_day { get; set; }

public Rooms(int room\_id, int level, int capacity, DateTime busy, int cost\_in\_day)

{

this.room\_id = room\_id;

this.level = level;

this.capacity = capacity;

this.busy = busy;

this.cost\_in\_day = cost\_in\_day;

}

static public void DataBaseTableToList()

{

SqlCommand cmd = DataBaseToGrid("Rooms");

rooms\_list.Clear();

using (SqlDataReader dr = cmd.ExecuteReader(CommandBehavior.CloseConnection)) {

while (dr.Read()) {

DateTime mydata;

try {

mydata = DateTime.Parse(Convert.ToString(dr.GetValue(3)));

}

catch {

mydata = DateTime.Parse("1/1/0");

}

rooms\_list.Add(new Rooms(Convert.ToInt32(dr.GetValue(0))

, Convert.ToInt32(dr.GetValue(1))

, Convert.ToInt32(dr.GetValue(2))

, mydata

, Convert.ToInt32(dr.GetValue(4))

));

}

}

Msg.Norm("Таблица \"Комнаты\" открыта для просмотра");

}

static public void RefreshDefaultValue()

{

data\_set.Tables[ "Rooms" ].Columns[ 1 ].DefaultValue = 5;

data\_set.Tables[ "Rooms" ].Columns[ 2 ].DefaultValue = 2;

data\_set.Tables[ "Rooms" ].Columns[ 3 ].DefaultValue = null;

data\_set.Tables[ "Rooms" ].Columns[ 4 ].DefaultValue = 800;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SERVICES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Services : Tables

{

public static List<Services> services\_list = new List<Services>();

public string service { get; set; }

public string category\_of\_services { get; set; }

public int service\_cost { get; set; }

public Services(string service, string category\_of\_services, int service\_cost)

{

this.service = service;

this.category\_of\_services = category\_of\_services;

this.service\_cost = service\_cost;

}

static public void DataBaseTableToList()

{

SqlCommand cmd = DataBaseToGrid("Services");

services\_list.Clear();

using (SqlDataReader dr = cmd.ExecuteReader(CommandBehavior.CloseConnection)) {

while (dr.Read()) {

services\_list.Add(new Services(Convert.ToString(dr.GetValue(0))

, Convert.ToString(dr.GetValue(1))

, Convert.ToInt32(dr.GetValue(2))

));

}

}

Msg.Norm("Таблица \"Услуги\" открыта для просмотра");

}

static public void RefreshDefaultValue()

{

data\_set.Tables[ "Services" ].Columns[ 1 ].DefaultValue

= Posts.posts\_list[ Posts.posts\_list.Count - 1 ].category\_of\_services;

data\_set.Tables[ "Services" ].Columns[ 2 ].DefaultValue = 200;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SESSION

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Session : Tables

{

public static List<Session> session\_list = new List<Session>();

public int session\_id { get; set; }

public int pasport\_num { get; set; }

public string first\_name { get; set; }

public string last\_name { get; set; }

public int room { get; set; }

public DateTime start\_session { get; set; }

public DateTime finish\_session { get; set; }

public int cost\_session { get; set; }

public string sex { get; set; }

public Session(int session\_id, int pasport\_num, string first\_name, string last\_name

, int room, DateTime start\_session, DateTime finish\_sessison, int cost\_session, string sex)

{

this.session\_id = session\_id;

this.pasport\_num = pasport\_num;

this.first\_name = first\_name;

this.last\_name = last\_name;

this.room = room;

this.start\_session = start\_session;

this.finish\_session = finish\_session;

this.cost\_session = cost\_session;

this.sex = sex;

}

static public void DataBaseTableToList()

{

SqlCommand cmd = DataBaseToGrid("Session");

session\_list.Clear();

using (SqlDataReader dr = cmd.ExecuteReader(CommandBehavior.CloseConnection)) {

while (dr.Read()) {

DateTime mydata1;

DateTime mydata2;

try {

mydata1 = DateTime.Parse(Convert.ToString(dr.GetValue(5)));

}

catch {

mydata1 = DateTime.Parse("1/1/0");

}

try {

mydata2 = DateTime.Parse(Convert.ToString(dr.GetValue(6)));

}

catch {

mydata2 = DateTime.Parse("1/1/0");

}

session\_list.Add(new Session(Convert.ToInt32(dr.GetValue(0))

, Convert.ToInt32(dr.GetValue(1))

, Convert.ToString(dr.GetValue(2))

, Convert.ToString(dr.GetValue(3))

, Convert.ToInt32(dr.GetValue(4))

, mydata1

, mydata2

, Convert.ToInt32(dr.GetValue(7))

, Convert.ToString(dr.GetValue(8))

));

}

}

Msg.Norm("Таблица \"Сессии\" открыта для просмотра");

}

static public void SearchID(string id)

{

var linq\_id = from x in session\_list

where x.session\_id == Convert.ToInt32(id)

select x;

try {

if (linq\_id.Count() != 0) {

main\_grid.ItemsSource = linq\_id;

Msg.Norm("Информация по ID " + id);

}

else {

Msg.Norm("Информации по ID " + id + " не обнаружено");

}

}

catch {

Msg.Wrong("Неверный ID для поиска");

}

}

static public void SearchPass(string passport)

{

var linq\_pass = from x in session\_list

where x.pasport\_num == Convert.ToInt32(passport)

select x;

try {

if (linq\_pass.Count() != 0) {

main\_grid.ItemsSource = linq\_pass;

Msg.Norm("Информация по номеру паспорта " + passport);

}

else {

Msg.Norm("Информации по номеру паспорта " + passport + " не обнаружено");

}

}

catch {

Msg.Wrong("Неверный номер паспорта для поиска");

}

}

static public void RefreshDefaultValue()

{

data\_set.Tables[ "Session" ].Columns[ 0 ].DefaultValue

= (DateTime.Now - new DateTime(2013, 12, 12)).TotalSeconds;

data\_set.Tables[ "Session" ].Columns[ 4 ].DefaultValue

= Session.session\_list[ Session.session\_list.Count - 1 ].room;

data\_set.Tables[ "Session" ].Columns[ 5 ].DefaultValue

= DateTime.Now;

data\_set.Tables[ "Session" ].Columns[ 6 ].DefaultValue

= DateTime.Now.AddDays(1);

data\_set.Tables[ "Session" ].Columns[ 7 ].DefaultValue = 0;

data\_set.Tables[ "Session" ].Columns[ 8 ].DefaultValue = "М";

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

WORKERS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Workers : Tables

{

public static List<Workers> workers\_list = new List<Workers>();

public int worker\_id { get; set; }

public string first\_name { get; set; }

public string last\_name { get; set; }

public DateTime date\_birth { get; set; }

public string sex { get; set; }

public string post { get; set; }

public Workers(int worker\_id, string first\_name, string last\_name

, DateTime date\_birth, string sex, string post)

{

this.worker\_id = worker\_id;

this.first\_name = first\_name;

this.last\_name = last\_name;

this.date\_birth = date\_birth;

this.sex = sex;

this.post = post;

}

static public void DataBaseTableToList()

{

SqlCommand cmd = DataBaseToGrid("Workers");

workers\_list.Clear();

using (SqlDataReader dr = cmd.ExecuteReader(CommandBehavior.CloseConnection)) {

while (dr.Read()) {

DateTime mydata;

try {

mydata = DateTime.Parse(Convert.ToString(dr.GetValue(3)));

}

catch {

mydata = DateTime.Parse("1/1/0");

}

workers\_list.Add(new Workers(Convert.ToInt32(dr.GetValue(0))

, Convert.ToString(dr.GetValue(1))

, Convert.ToString(dr.GetValue(2))

, mydata

, Convert.ToString(dr.GetValue(4))

, Convert.ToString(dr.GetValue(5))

));

}

}

Msg.Norm("Таблица \"Персонал\" открыта для просмотра");

}

static public void RefreshDefaultValue()

{

data\_set.Tables[ "Workers" ].Columns[ 4 ].DefaultValue = "M";

data\_set.Tables[ "Workers" ].Columns[ 5 ].DefaultValue

= Posts.posts\_list[Posts.posts\_list.Count - 1].posts;

}

}

# ПРИЛОЖЕНИЕ 3 КЛАСС MSG

class Msg

{

static public void Norm(string msg)

{

ShowText(System.Windows.Media.Brushes.White, msg);

}

static public void Wrong(string msg)

{

ShowText(System.Windows.Media.Brushes.Red, msg);

}

static public void Ready(string msg)

{

ShowText(System.Windows.Media.Brushes.LightGreen, msg);

}

static private void ShowText(System.Windows.Media.Brush col, string msg)

{

Tables.lab\_info.Background = col;

Tables.lab\_info.Content = " " + DateTime.Now.Hour + ":"

+ DateTime.Now.Minute + ":" + DateTime.Now.Second + " : " + msg;

}

}

# ПРИЛОЖЕНИЕ 4 КЛАСС ОБРАБОТЧИКА ФОРМЫ

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

Tables.lab\_info = lab\_info;

Tables.main\_grid = main\_grid;

Msg.Norm("Приложение запущено");

// show all tables and load data in list

AccountingServices.DataBaseTableToList();

Posts.DataBaseTableToList();

Services.DataBaseTableToList();

Session.DataBaseTableToList();

Workers.DataBaseTableToList();

Rooms.DataBaseTableToList();

Tables.RefreshToEdit();

}

private void but\_AccountingServices\_Click(object sender, RoutedEventArgs e)

{

AccountingServices.DataBaseTableToList();

Tables.RefreshToEdit();

}

private void but\_Posts\_Click(object sender, RoutedEventArgs e)

{

Posts.DataBaseTableToList();

Tables.RefreshToEdit();

}

private void but\_Rooms\_Click(object sender, RoutedEventArgs e)

{

Rooms.DataBaseTableToList();

Tables.RefreshToEdit();

}

private void but\_Services\_Click(object sender, RoutedEventArgs e)

{

Services.DataBaseTableToList();

Tables.RefreshToEdit();

}

private void but\_Session\_Click(object sender, RoutedEventArgs e)

{

Session.DataBaseTableToList();

Tables.RefreshToEdit();

}

private void but\_Workers\_Click(object sender, RoutedEventArgs e)

{

Workers.DataBaseTableToList();

Tables.RefreshToEdit();

}

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

{

Tables.SaveCurrentTable();

}

private void but\_search\_id\_click(object sender, RoutedEventArgs e)

{

Session.SearchID(textbox\_searhid.Text);

}

private void but\_search\_pass\_click(object sender, RoutedEventArgs e)

{

Session.SearchPass(textbox\_searchpass.Text);

}

private void Rowing(object sender, DataGridRowEditEndingEventArgs e)

{

Tables.RefreshToEdit();

}

}