**Содержание**

[Введение 4](#_Toc10097833)

[1 Постановка задачи 5](#_Toc10097834)

[2 Аналитический обзор аналогов 6](#_Toc10097835)

[3 Анализ требований к программному средству и разработка функциональных требований 8](#_Toc10097836)

[4 Разработка функциональных требований 9](#_Toc10097837)

[5 Анализ и проектирование архитектуры приложения 10](#_Toc10097838)

[5.1 Анализ платформы 10](#_Toc10097839)

[5.2 Диаграмма классов 10](#_Toc10097840)

[5.3 Диаграмма вариантов использования 11](#_Toc10097841)

[5.3 Диаграмма деятельности 11](#_Toc10097842)

[5.4 Разработка архитектуры приложения 13](#_Toc10097843)

[6 Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса 14](#_Toc10097844)

[7 Тестирование приложения 16](#_Toc10097845)

[8 Руководство по установке и использованию 17](#_Toc10097846)

[Заключение 18](#_Toc10097847)

[Список литературы 19](#_Toc10097848)

[Приложение А 20](#_Toc10097849)

[Приложение Б 21](#_Toc10097850)

[Приложение В 23](#_Toc10097851)

# Введение

Социологический опрос — метод социологического исследования, заключающийся в сборе и получении первичных эмпирических сведений об определённых мнениях, знаниях и социальных фактах, составляющих предмет исследования, путём устного или письменного взаимодействия исследователя (интервьюера) и заданной совокупности опрашиваемых (интервьюируемые, респонденты).

Если исследование проводится, например, по проблемам здоровья и здорового образа жизни, то выявляется, каким образом работает система здравоохранения в стране, ее достоинства и недостатки. Позволяет это определить и то, как люди сами понимают, что такое здоровый образ жизни, как они ведут себя, столкнувшись с болезнью, обращаются ли к врачу. Поэтому все это нужно в первую очередь для того, чтобы сделать более комфортной нашу жизнь. По результатам таких опросов государственные органы и общественные организации могут принимать определенные решения, чтобы улучшать нашу жизнь. А как это можно сделать, если нет информации о том, что происходит в обществе и что его волнует? Социологические исследования как раз-таки позволяют получать такие сведения. Результаты исследований социологов очень важны и для людей, и для самой социологической науки.

# 1 Постановка задачи

Целью проекта является разработка программного средства для сбора и обработки результатов социологических опросов. Система предназначена сократить время и ресурсы на сбор и обработку данных анкетирования, уменьшить объем бумажной документации, а также избежать ошибок при обработке данных, вызванных человеческим фактором.

Задачи:

1.Создание базы данных для хранения данных опросов;

2.Разработка формы опроса для респондентов;

3. Разработка формы для создания опроса;

3.Разработка методов для отображения результатов анкетирования;

4.Тестирование системы;

5.Разработка руководства пользователя;

# 2 Аналитический обзор аналогов

Веб-сайт «mk.ru», представлен на рисунке 2.1. Данный сервис имеет минимальный интерфейс и выполняет всего несколько функций: отображения списка тестов, примитивный анализ результатов. При этом сам тест состоит всего из одного вопроса, и пройти его можно только в течении 2-ух дней, и посмотреть результаты только после окончания теста.

Плюсы приложения: простой интерфейс

Минусы приложения: непрактичная система, отвлекающий контент

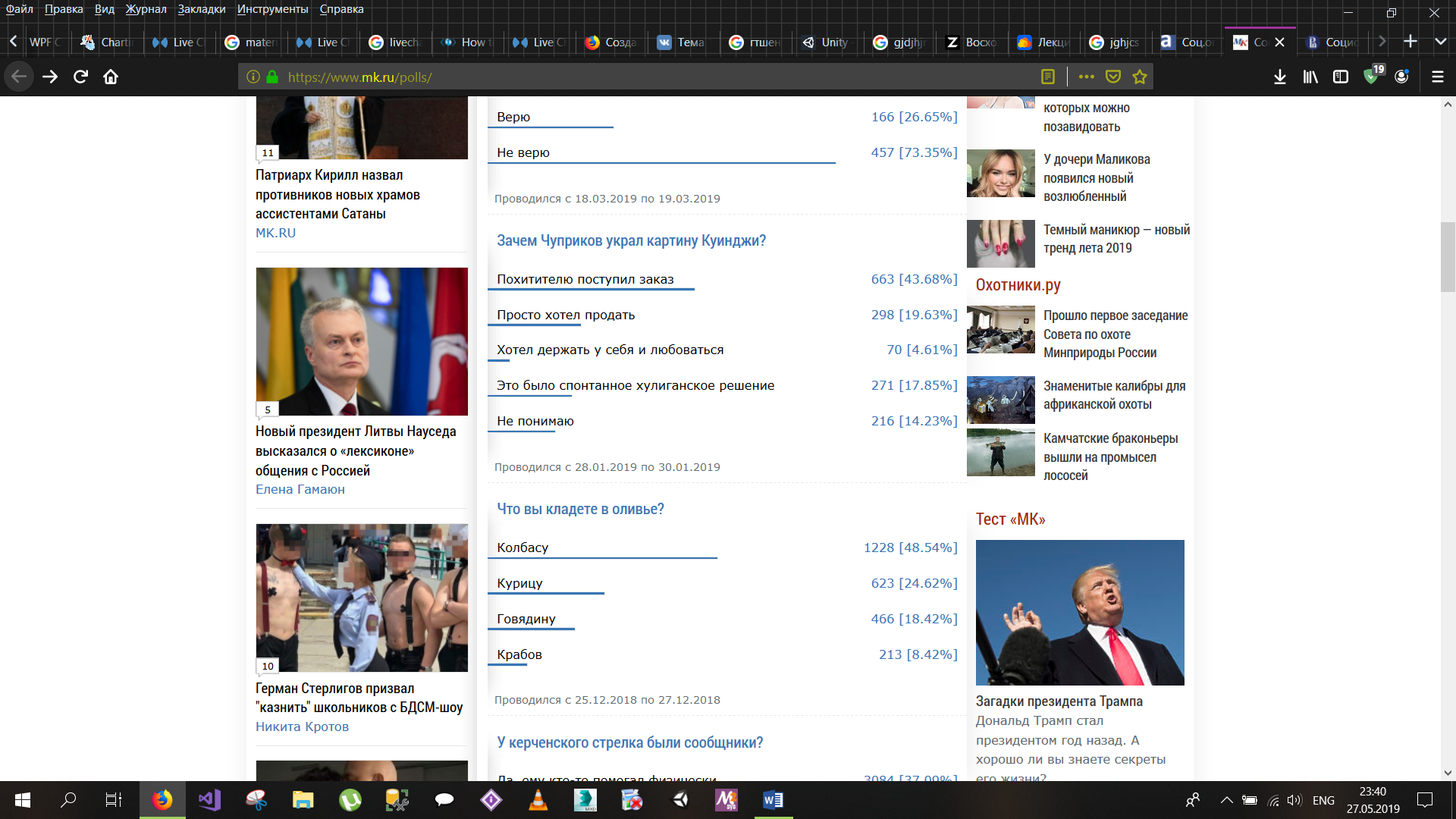


Рисунок 2.1 - Веб-сайт «mk.ru»

Веб-сайт «Анкетки.ру», представлен на рисунке 2.2. Данный сервис носит в себе скорее развлекательный характер. Здесь любой пользователь может создать собственный вопрос на любую тематику. Опросы можно найти несколькими способами: по тегу, по разделу, выбрать один из опросов, представленный на главной странице, выбрать случайный тест или одну из популярных тем. Сервис отображает результаты в виде столбчатых графиков и процентов, которые набрал тот или иной ответ. Сайт имеет примитивный спокойный дизайн, и большое количество текста на одной странице.

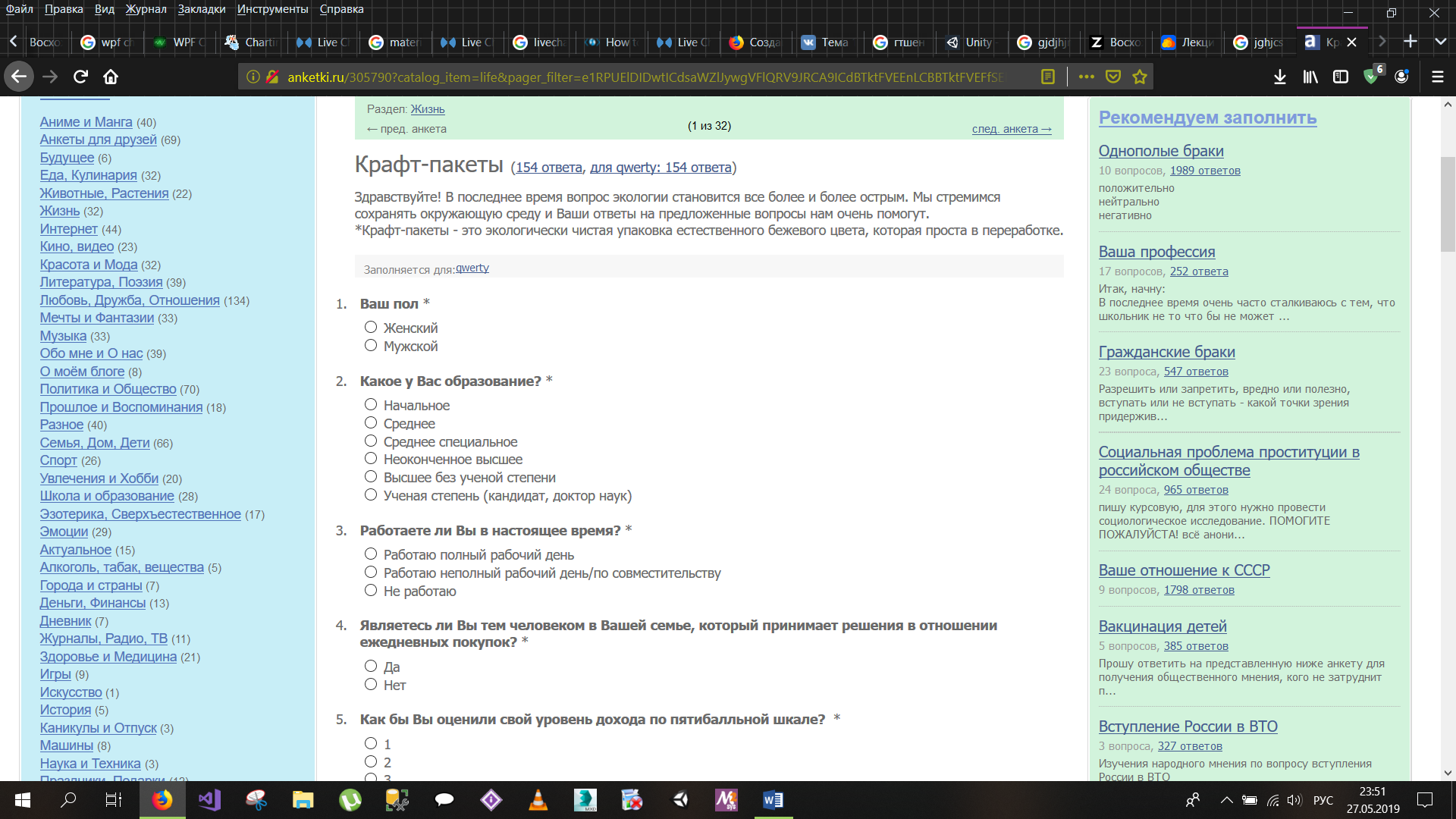


Рисунок 2.2 - Веб-сайт «Анкетки.ру»

Приложение «Опросы, тесты, мнения», представлен на рисунке 2.3. Приложение представляет собой список тестов на различные повседневные темы. Здесь присутствует система поощрений в виде баллов, которые пользователи могут потратить на небольшие бонусы или прохождение других тестов. Так же есть отдельная вкладка с профилем пользователя, где он может вводить дополнительные данные о себе. Опросы не разбиты на категории и выводятся в порядке даты добавления на главной странице.

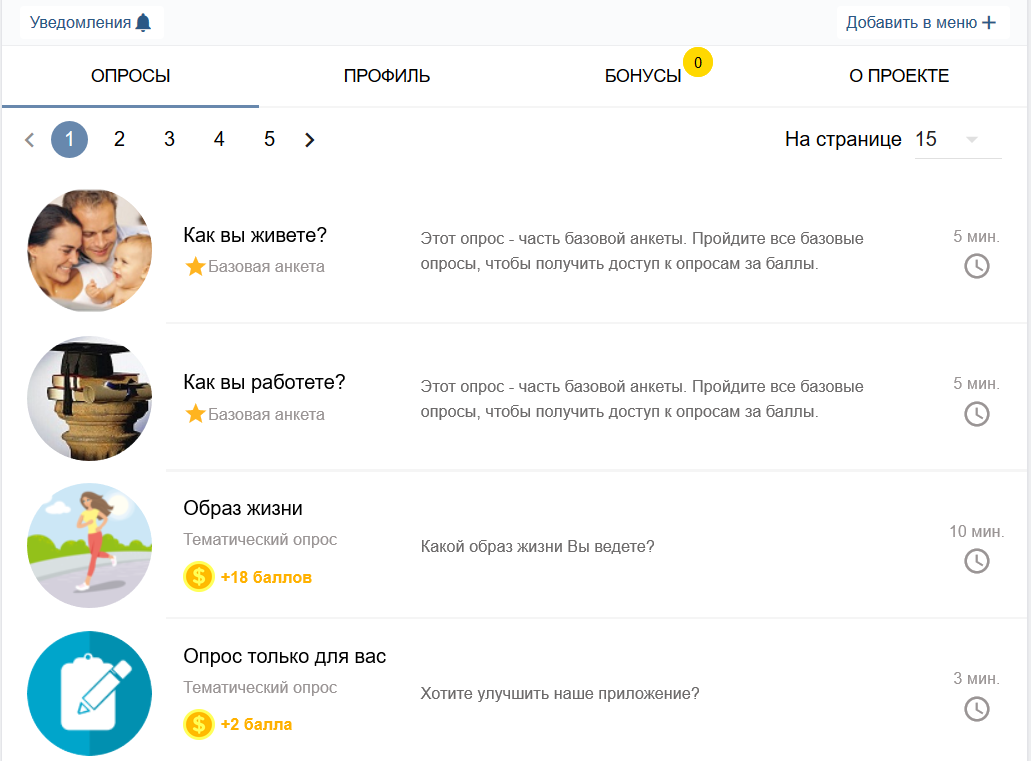


Рисунок 2.3 - Приложение «Опросы, тесты, мнения»

# 3 Анализ требований к программному средству и разработка функциональных требований

Целью проекта является разработка программного средства для сбора и обработки результатов социологических опросов. Система предназначена сократить время и ресурсы на сбор и обработку данных анкетирования, уменьшить объем бумажной документации, а также избежать ошибок при обработке данных, вызванных человеческим фактором.

Требования:

1. Должна быть разработана система авторизации и аутентификации с разделением на роли «Пользователь» и «Администратор»;
2. Необходима система тестирования – ответов на вопросы с запоминанием в базе ответов на эти вопросы;
3. Также, предоставить пользователям с ролью «Администратор» конструктор тестов;
4. Предоставить пользователям возможность смены цветовой схемы приложения, с запоминанием их выбора;
5. Реализовать возможность просмотра статистики по пройденным тестам в графическом виде.

# 4 Разработка функциональных требований

Приложение будет выполнено на платформе «Классическое приложение Windows» (WPF – Windows Presentation Foundation) на языке программирования C#, так как эта платформа является наиболее освоенной и имеет большую базу сторонних ресурсов.

В качестве базы данных будет использована система управления реляционными базами данными (РСУБД) Microsoft SQL Server. Данную РСУБД программно свяжем с объектно-ориентированной технологией доступа к данным object-relational mapping (ORM) ADO.NET Entity Framework, с помощью которой по подходу Code First построим базу данных.

Для оформления интерфейса используем библиотеку Material Designer.

# 5 Анализ и проектирование архитектуры приложения

## 5.1 Анализ платформы

Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

WPF предустановлена в Windows Vista (.NET Framework 3.0), Windows 7 (.NET Framework 3.5 SP1), Windows 8 (.NET Framework 4.0 и 4.5), Windows 8.1 (.NET Framework 4.5.1). С помощью WPF можно создавать широкий спектр как автономных, так и запускаемых в браузере приложений.

XAML представляет собой язык декларативного описания интерфейса, основанный на XML. Также реализована модель разделения кода и дизайна, позволяющая кооперироваться программисту и дизайнеру. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на элементы управления второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента.

## 5.2 Диаграмма классов

Диаграмма классов (англ. Static Structure diagram) — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. На рисунке 3.3 представлена диаграмма классов нашего проекта.

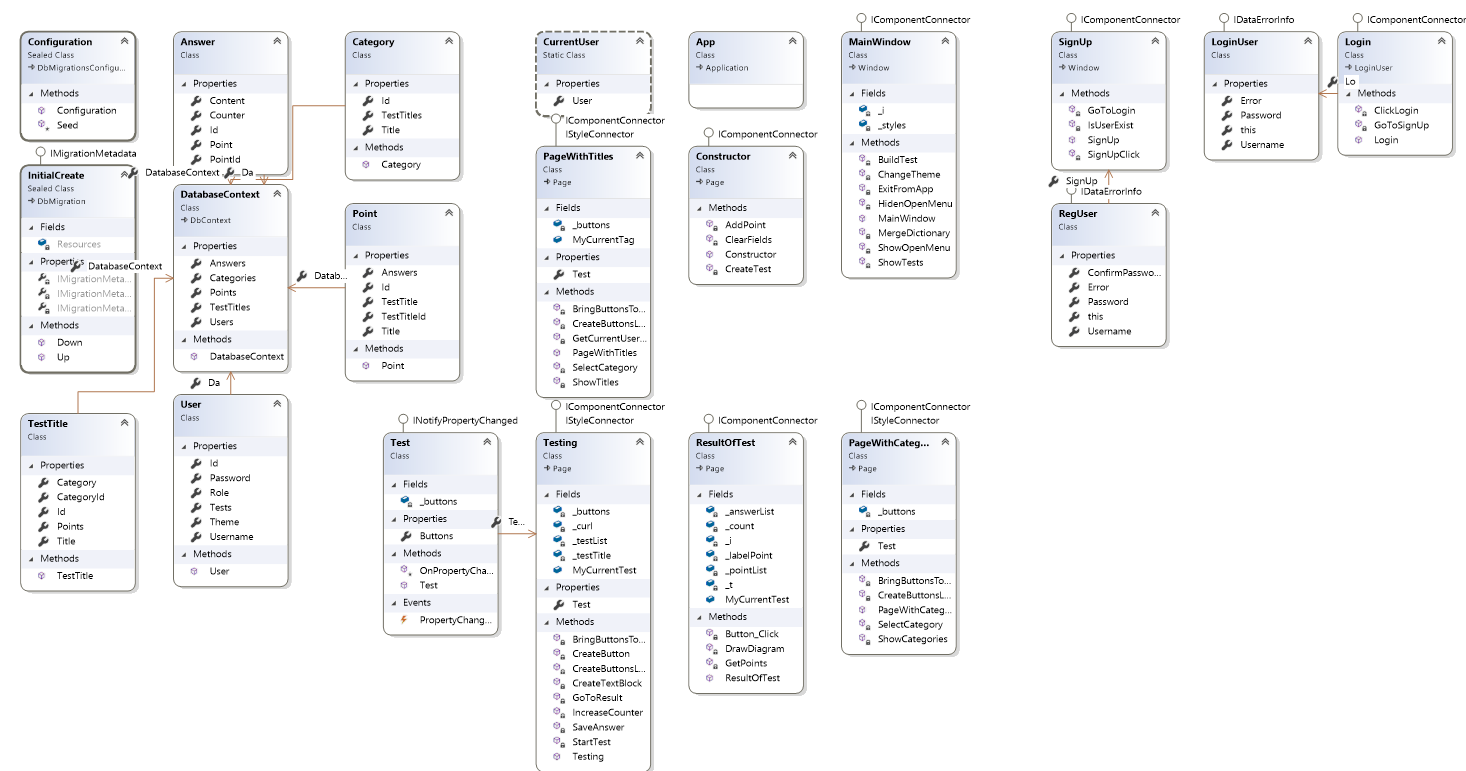


Рисунок 5.1 – Диаграмма классов

## 5.3 Диаграмма вариантов использования

Диаграмма вариантов использования (англ. use case diagram) в UML — диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Диаграмма вариантов использования представлена на рисунке 5.2.

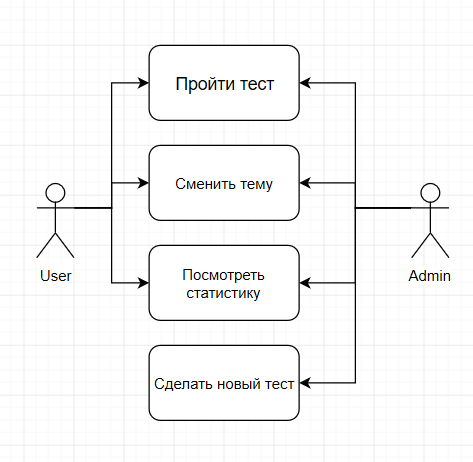


Рисунок 5.2– Диаграмма вариантов использования

## 5.3 Диаграмма деятельности

Диаграмма деятельности (англ. activity diagram) — UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий англ. action, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграмма деятельности приложения представлена на рисунке 5.3.

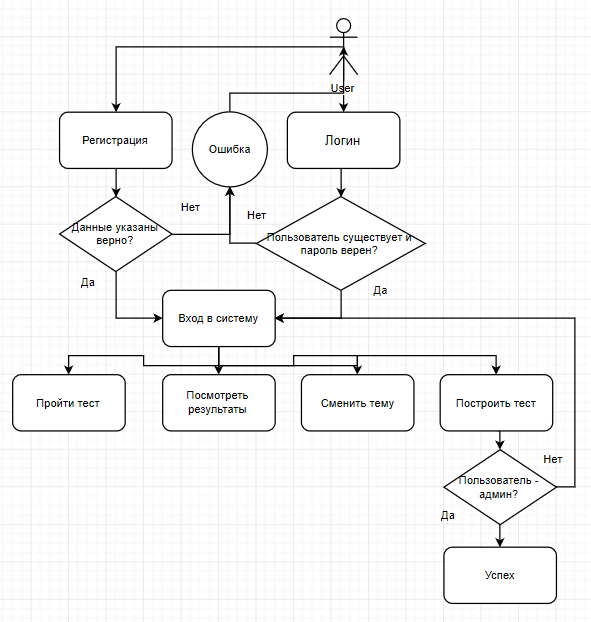


Рисунок 5.3 – Диаграмма деятельности

## 5.4 Разработка архитектуры приложения

Основное окно приложения будет реализовано с помощью класса Window, так же, как и формы для логина и регистрации. Основное окно будет служить основой для использования пользователем основного функционала приложения. Весь функционал будет заложен в отдельные страницы класса Page и помещаться в «кадр» (Frame) основного окна.

Для перехода на страницу с нужным функционалом использовано меню ListView с использованием библиотеки Material Designer для придания красочного стиля.

На форме регистрации использована система валидации на основе интерфейса IDataErrorInfo. Пример подобной реализации на основе окна Login представлен в приложении А.

Для графической части приложения (графики) использована библиотека LiveCharts.Wpf. Она предоставляет набор современных, стильных и информативных графиков и диаграмм. Использование данной библиотеки представлено в приложении Б.

Естественно, для нашего приложения необходима сущность пользователя. Так как мы ведем построение базы данных с помощью Entity Framework по подходу Code First, то нам достаточно создать модель с необходимыми свойствами. Модель пользователя представлена в приложении В. Пользователь обладает именем, паролем, предпочитаемой темой приложения и связан с тестами – отмечаются те, которые он уже прошел.

# 6 Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса

В качестве базы данных будем использовать Microsoft SQL Server. Данная РСУБД прекрасно масштабируется, обладает высокой производительностью, доступна, имеет интеграцию с другими продуктами Microsoft.

Саму базу данных будем создавать с помощью ORM Entity Framework (EF) по подходу Code First. Для этого создадим контекст базы данных и все необходимые модели. Во время запуска приложения EF будет искать имеющуюся базу данных. Если она будет отсутствовать, то он её создаст в соответствии с моделями данных. Диаграмма связей базы данных представлена на рисунке 6.1.

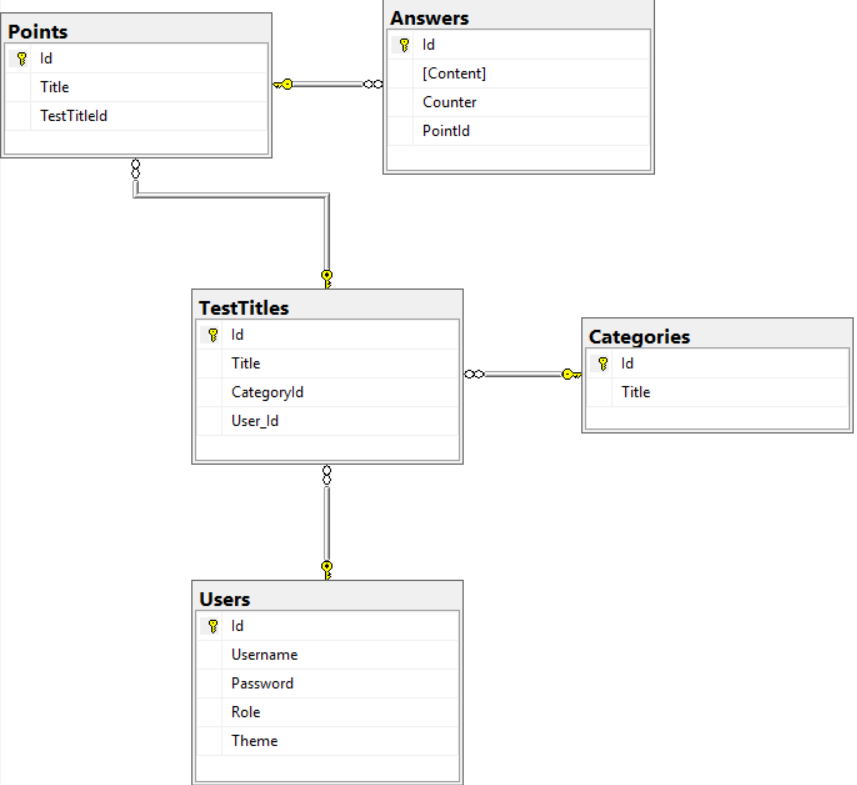


Рисунок 6.1 – Диаграмма связей базы данных

При разработке интерфейса приложения будем использовать библиотеку Material Designer. Эта библиотека имеет огромное количество красивых, изящных и функциональных элементов управления.

Главная страница приложения представлена на рисунке 6.2.



Рисунок 6.2 – Главная страница приложения

# 7 Тестирование приложения

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

На каждом этапе разработки проводили регрессионное тестирование. Это такой вид тестирования, при котором проверяется не внесли ли новые изменения и новые ошибки не только в самом новом коде, но и в старом.

Был привлечен полупрофессиональный QA-тестер, который составил тестовые сценарии, благодаря которым были выявлены как критические ошибки, так и совсем мелкие опечатки.

Затем были привлечены обычные пользователи для тестирования по принципу «черного ящика»: пользователи не имели доступа к коду приложения, а только к его интерфейсам. В целом, пользователи остались довольны.

Была протестирована система авторизации и аутентификации. Пользователь будет пропущен в систему только если введены правильные логин и пароль. Проверка логина представлена на рисунке 7.1. При регистрации проводиться проверка – существует ли пользователь с данным логином. Данная проверка представлена на рисунке 7.2.

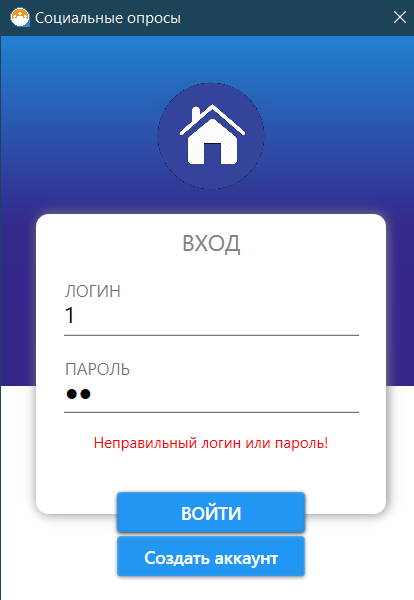


Рисунок 7.1 – Валидация логина

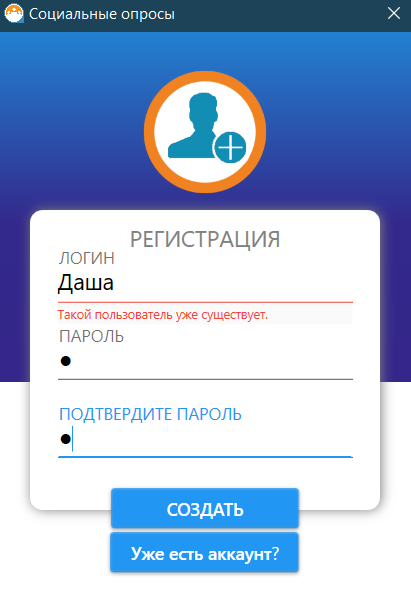


Рисунок 7.2 – Валидация регистрации

Также, пройденные тесты должны визуально отличаться от не пройденных. Результат представлен на рисунке 7.3.



Рисунок 7.3 – Пройденный тест стал серого цвета

В конструкторе тестов все поля являются обязательными для заполнения. Необходимо было протестировать соответствует ли это. Результат представлен на рисунке 7.4.



Рисунок 7.4 – Все поля обязательны в конструкторе

# 8 Руководство по установке и использованию

Приложения на основе WPF могут быть установлены тремя способами:

* с помощью консоли XCopy. Данный метод подойдет, если у нас достаточно легкое приложение и не требует интеграции в меню «Пуск»;
* с помощью установщика Windows. Это наиболее универсальный способ, так как он легок в освоении и встроен в операционные системы семейства Windows. Но данный способ не поддерживается Visual Studio с версии 16.0;
* ClickOnce. Позволяет установить приложение на удаленные станции. Также является удобным и простым способом доставки, но приложение, установленное таким образом, будет иметь ограниченный набор прав.

В нашем случае будет достаточно собрать решение типа Release.

Также, для запуска приложения необходимо иметь на компьютере актуальную версию .NET Framework.

# Заключение

В ходе выполнения работы были достигнуты следующие результаты:

Изучены основы объектно-ориентированного программирования и применены в практической разработке программного средства.

Исследованы все этапы проектирования и разработки приложения: проектирование, разработка, тестирование, реализация.

При работе над приложением была изучена платформа WPF и выделены по ней достоинства и недостатки. WPF обладает веб-подобной моделью компоновки, богатым инструментарием для стилизации элементов, системой привязок, а также большим ассортиментом сторонних плагинов и библиотек.

Однако, не обошлось и без недостатков: большое потребление оперативной памяти, язык XAML также является и недостатком – повышает уровень вхождения и более высокий уровень «ручного» программирования.

# Список литературы

1. Библиотека Microsoft MSDN - URL: http://msdn.microsoft.com/ru-ru/library
2. Шилдт Г. - C# 4.0 полное руководство - Москва: Вильямс, 2011.

3. Натан, А. WPF 4. Подробное руководство. -- СПб. : Символ-Плюс, 2011.

4. Ч. Петцольд Microsoft Windows Presentation Foundation - М. : Русская Редакция; СПб. : Питер, 2008.

# Приложение А

public class LoginUser : IDataErrorInfo

{

public string Username { get; set; }

public string Password { get; set; }

public string this[string columnName]

{

get

{

var error = string.Empty;

switch (columnName)

{

case "Username":

User user;

using (var db = new DatabaseContext())

{

user = db.Users.FirstOrDefault(u => u.Username.Equals(Username));

}

try

{

if (!string.IsNullOrEmpty(Username) &&

(user == null || !Username.Equals(user.Username) || !Password.Equals(user.Password)))

{

error = "Неправильный логин или пароль";

}

}

catch { }

break;

}

return error;

}

}

public string Error { get; }

}

}

# Приложение Б

using kurs.Models;

using LiveCharts;

using LiveCharts.Wpf;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using Point = kurs.Models.Point;

namespace kurs

{

public partial class ResultOfTest

{

public static int MyCurrentTest;

private List<Answer> \_answerList;

private List<Point> \_pointList;

private int \_count;

private int \_i;

private readonly SeriesCollection \_t = new SeriesCollection();

private readonly Func<ChartPoint, string> \_labelPoint = chartPoint =>

$"{chartPoint.Y} ({chartPoint.Participation:P})";

public ResultOfTest()

{

InitializeComponent();

GetPoints();

ButtonLeft.IsEnabled = false;

DrawDiagram(\_pointList[0]);

}

private void GetPoints()

{

using (var db = new DatabaseContext())

{

try

{

\_pointList = db.Points.Where(p => p.TestTitleId == MyCurrentTest).ToList();

\_count = \_pointList.Count;

}

catch { }

}

}

private void DrawDiagram(Point cur)

{

PieChart.Series?.Clear();

if (\_i < 1)

{

ButtonLeft.IsEnabled = false;

}

if (\_i == \_count - 1)

{

ButtonRight.IsEnabled = false;

}

if (\_i > 0)

{

ButtonLeft.IsEnabled = true;

}

if (\_i < \_count - 1)

{

ButtonRight.IsEnabled = true;

}

using (var db = new DatabaseContext())

{

\_answerList = db.Answers.Where(a => a.PointId == cur.Id).ToList();

}

foreach (var answer in \_answerList)

{

var temp = new PieSeries

{

Title = answer.Content,

Values = new ChartValues<int> { answer.Counter },

LabelPoint = \_labelPoint

};

\_t.Add(temp);

}

PieChart.Series = \_t;

TitleofTest.Text = cur.Title;

}

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

{

var temp = \_pointList[0];

var b = (Button)sender;

switch (b.Name)

{

case "ButtonLeft":

\_i--;

temp = \_pointList[\_i];

break;

case "ButtonRight":

\_i++;

temp = \_pointList[\_i];

break;

}

DrawDiagram(temp);

}

}

}

# Приложение В

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace kurs.Models

{

public class User

{

[Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string Role { get; set; }

public int Theme { get; set; }

public virtual ICollection<TestTitle> Tests { get; set; }

public User()

{

Tests = new List<TestTitle>();

}

}

}