**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение Московской области «Люберецкий техникум имени Героя Советского Союза, летчика-космонавта Ю.А.Гагарина»**

**ОТЧЕТ**

**УЧЕБНОЙ ПРАКТИКИ**

\_\_\_\_\_\_\_\_\_Буланина Анастасия Игоревна\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество студента)

по профессиональному модулю

**\_\_\_\_\_\_\_\_***ПМ.01 «Разработка модулей программного обеспечения***\_\_\_\_\_\_\_\_**

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ для компьютерных систем»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Специальность \_*09.02.07 "Информационные системы и программирование"*\_

Код, название

Курс \_3\_ Группа № \_195\_

Период практики с «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. по «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г.

Руководитель практики

от техникума Ревнивцев Максим Валерьевич

Жирнова Юлия Витальевна

Люберцы 2021

Содержание

ВВЕДЕНИЕ 3

ИЗУЧЕНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5

ИЗУЧЕНИЕ ПРОГРАММНЫХ СРЕДСТВ 6

OPENCV 9

GIT. БАЗОВЫЙ КУРС 13

ПРИЛОЖЕНИЕ WPF 15

ВЫВОДЫ 27

СПИСОК ЛИТЕРАТУРЫ 28

**Введение**

Проф. модуль Разработка модулей программного обеспечения для компьютерных систем является важным профессиональным модулем для специальности 09.02.07 «Информационные системы и программирование», обуславливающим вид для профессиональной деятельности выпускника.

Учебная практика представляет собой важнейшую составную часть

учебного процесса по подготовке специалистов, способствует повышению

общего уровня профессиональной подготовки, закреплению и углублению

полученных теоретических знаний по дисциплинам.

Целью учебной практики является реализация компьютерного зрения

с помощью библиотеки OpenCV, Разработка приложения WPF и изучение

Git. Базовый курс.

Для достижения данной цели требуется решить следующие задачи:

1. Знакомство с литературой

2. Изучить библиотеку OpenCV

3. Реализовать приложения на WPF

4. Пройти курс «Git. Базовый курс

В ходе учебной практики планируется освоение следующего вида деятельности: «Разработка модулей программного обеспечения для компьютерных систем» (ПК):

ПК 1.1 Формировать алгоритмы разработки программных модулей в

соответствии с техническим заданием.

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3 Выполнять отладку программных модулей с использованием

специализированных программных средств

ПК 1.4 Выполнять тестирование программных модулей

ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ.

4

ПК 2.1. Разрабатывать требования к программным модулям на основе

анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием

специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного

обеспечения на предмет соответствия стандартам кодирования.

**Изучение предметной области**

**Предметная область** - множество всех предметов, свойства которых и

отношения между которыми рассматриваются в научной теории.

**OpenCV** - библиотека алгоритмов компьютерного зрения, обработки

изображений и численных алгоритмов общего назначения с открытым кодом.

Реализована на C/C++, также разрабатывается для Python, Java, Ruby,

Matlab, Lua и других языков. Может свободно использоваться в академических и коммерческих целях — распространяется в условиях лицензии

BSD.

**Windows Presentation Foundation (WPF)** –это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность.

**Язык программирования С++** - Широко используется для разработки

программного обеспечения, являясь одним из самых популярных языков

программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр.

**Изучение программных средств**

1. **Microsoft SQL Server Management Studio 18** - Утилита из

Microsoft SQL Server 2005 и более поздних версий для конфигурирования,

управления и администрирования всех компонентов Microsoft SQL Server.

Утилита включает скриптовый редактор и графическую программу, которая

работает с объектами и настройками сервера.

Достоинства:

**Полная Web ориентированность**. Осуществление запросов, анализ и

управление через Web. Использование языка XML для обмена информацией

между удаленными системами.

**Масштабируемость и надежность**. SQL Server 2018 обеспечивает

практически неограниченный рост объемов хранения за счет увеличения

надежности и масштабируемости системы, используя все преимущества

мультипроцессорной обработки файлов. Это безопасная, надежная, масштабируемая платформа, защищающая информацию в приложениях и повышающая её доступность.

**Скорость создания решений**. SQL Server 2018 в сочетании с .NET

Framework уменьшает время разработки, внедрения и выхода на рынок современных приложений, ускоряет процесс поиска информации, упрощает

управление, позволяет использовать создаваемые пользователем функции в

других приложениях, предоставляет широкие возможности для создания

Web-приложений.

**Возможность обработки вычислений в оперативной памяти**

(in-memory OLTP)

**Возможность взаимодействия с публичным облаком Windows Azure.**

Облачное резервирование локальных систем и аварийное восстановление из

облака повышает отказоустойчивость критичных для бизнеса приложений.

**Microsoft Visual Studio** — линейка продуктов компании Microsoft,

включающих интегрированную среду разработки программного обеспечения

и ряд других инструментов

**Встроенный Web-сервер.** Для обслуживания Web-приложения

ASP.NET необходим Web-сервер, который будет ожидать Web-запросы и обрабатывать соответствующие страницы. Наличие в Visual Studio интегрированного Web-сервера позволяет запускать Web-сайт прямо из среды проектирования, а также повышает безопасность, исключая вероятность получения доступа к тестовому Web-сайту с какого-нибудь внешнего компьютера, поскольку тестовый сервер может принимать соединения только с локального компьютера.

**Поддержка множества языков при разработке**. Visual Studio позволяет писать код на своем языке или любых других предпочитаемых языках, используя все время один и тот же интерфейс (IDE). Более того, Visual Studio

также еще позволяет создавать Web-страницы на разных языках, но помещать их все в одно и то же Web-приложение.

**Интуитивный стиль кодирования.** По умолчанию Visual Studio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения

и менее подверженным ошибкам.

**Более высокая скорость разработки.** Многие из функциональных возможностей Visual Studio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции

IntelliSense (которая умеет перехватывать ошибки и предлагать правильные

варианты), функции поиска и замены (которая позволяет отыскивать ключевые слова как в одном файле, так и во всем проекте) и функции автоматического добавления и удаления комментариев (которая может временно скрывать блоки кода), позволяют разработчику работать быстро и эффективно.

**PyCharm** — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на

Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

Достоинства:

**Простая организация проектов.** Очень просто создавать проекты и открывать уже существующие: «пайчарм» буквально в два клика позволяет

приступить к редактированию кода. Не нужно импортировать проект в

workspace или делать что-то подобное.

**Удобный автокомпилит.** Автокомплит работает мгновенно, не нужно

вызывать его «хоткеем». Работает не только в привычных местах, но и в

шаблонах Джанго, для подстановки путей к файлам, к тому же очень шустрый.

**OpenCV**

**OpenCV** - Библиотека компьютерного зрения и машинного обучения с

открытым исходным кодом. В неё входят более 2500 алгоритмов, в которых

есть как классические, так и современные алгоритмы для компьютерного

зрения и машинного обучения. Эта библиотека имеет интерфейсы на различных языках, среди которых есть Python, Java, C++ и Matlab.

Для работы c OpenCV был создан отдельный репозиторий на GitHub [**educational-practice**](https://github.com/BulaninaNastya/educational-practice). Весь код, его описание и результаты расположены в папке OpenCV

**Описание кода.**

В самом начале нам понадобится создать проект в Microsoft Visual Studio.

Открыв приложение, видим меню, нажимаем кнопку «Создание проекта». Далее будет предложено создать проект нужного нам расширения, вбиваем в поиск “C++”, выбираем «Пустое приложение», нажимаем далее.   
Вводим запрашиваемую приложением информацию (путь, название проекта), нажимаем «создать».   
Проект создан. На данном этапе требуется подключить библиотеку Opencv.  
Для этого выполняем следующие действия:   
Изображение выглядит как текст

Автоматически созданное описание

Для трансляции видео в VS была использована веб-камера.

**Работа с кодом:**

Для загрузки изображения мы используем функцию cv2.imread(), где первым аргументом указывается путь к изображению, а вторым аргументом, который является необязательным, мы указываем, в каком цветовом пространстве мы хотим считать наше изображение. Чтобы считать изображение в RGB — cv2.IMREAD\_COLOR, в оттенках серого — cv2.IMREAD\_GRAYSCALE. По умолчанию данный аргумент принимает значение cv2.IMREAD\_COLOR. Данная функция возвращает 2D (для изображения в оттенках серого) либо 3D (для цветного изображения) массив NumPy. Форма массива для цветного изображения: высота x ширина x 3, где 3 — это байты, по одному байту на каждую из компонент. В изображениях в оттенках серого всё немного проще: высота x ширина.

**def** **loading\_displaying\_saving**():

img = cv2.imread('girl.jpg', cv2.IMREAD\_GRAYSCALE)

cv2.imshow('girl', img)

cv2.waitKey(0)

cv2.imwrite('graygirl.jpg', img)

Для того, чтобы узнать высоту, ширину и количество каналов у изображения можно использовать атрибут shape:

print("Высота:"+str(img.shape[0]))

print("Ширина:" + str(img.shape[1]))

print("Количество каналов:" + str(img.shape[2]))

Чтобы получить доступ к значению пикселя, нам просто нужно указать координаты x и y пикселя, который нас интересует.

(b, g, r) = img[0, 0]

print("Красный: {}, Зелёный: {}, Синий: {}".format(r, g, b))

Сначала мы берём пиксель, который расположен в точке (0,0). Данный пиксель, да и любой другой пиксель, представлены в виде кортежа. Заметьте, что название переменных расположены в порядке b, g и r. В следующей строке выводим значение каждого канала на экран. Как можно увидеть, доступ к значениям пикселей довольно прост, также просто можно и манипулировать значениями пикселей:

img[0, 0] = (255, 0, 0)

(b, g, r) = img[0, 0]

print("Красный: {}, Зелёный: {}, Синий: {}".format(r, g, b))

**Распознавание лиц**

import cv2

image\_path = "./путь/к/фото.расширение"

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

image = cv2.imread(image\_path)

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(

gray,

scaleFactor= 1.1,

minNeighbors= 5,

minSize=(10, 10)

)

faces\_detected = "Лиц обнаружено: " + format(len(faces))

print(faces\_detected)

# Рисуем квадраты вокруг лиц

for (x, y, w, h) in faces:

cv2.rectangle(image, (x, y), (x+w, y+h), (255, 255, 0), 2)

viewImage(image,faces\_detected)

**detectMultiScale** — общая функция для распознавания как лиц, так и объектов. Чтобы функция искала именно лица, мы передаём ей соответствующий каскад.

Функция detectMultiScale принимает 4 параметра:

1. Обрабатываемое изображение в градации серого.
2. Параметр scaleFactor. Некоторые лица могут быть больше других, поскольку находятся ближе, чем остальные. Этот параметр компенсирует перспективу.
3. Алгоритм распознавания использует скользящее окно во время распознавания объектов. Параметр minNeighbors определяет количество объектов вокруг лица. То есть чем больше значение этого параметра, тем больше аналогичных объектов необходимо алгоритму, чтобы он определил текущий объект, как лицо. Слишком маленькое значение увеличит количество ложных срабатываний, а слишком большое сделает алгоритм более требовательным.
4. minSize — непосредственно размер этих областей.

**Contours** — распознавание объектов

Распознавание объектов производится с помощью цветовой сегментации изображения.   
Для этого есть две функции: cv2.findContours и cv2.drawContours.

**Разметка дороги**

Цель данного проекта — построить простую линейную модель для покадрового распознавания полос движения.

Процесс построения детектора состоит из трех основных шагов:

* Предобработка данных, фильтрация от шума и векторизация изображения.
* Обновление состояния линий дорожной разметки по данным из первого шага.
* Рисование обновленных линий и других объектов на исходном изображении.

Сначала на вход функции image\_pipeline подается 3-канальное изображение формата RGB, которое затем фильтруется, преобразовывается, а внутри функции обновляются объекты Line и Lane. Затем поверх самого изображения рисуются все необходимые элементы

Применим два преобразования:

1. Детектор границ Кэнни: алгоритм оптимального определения границ, который рассчитывает градиенты интенсивности изображения, а затем с помощью двух порогов удаляет слабые границы, оставляя искомые (мы используем (280, 360)) как пороговые значения в функции canny.
2. Преобразование Хафа: получив границы с помощью алгоритма Кэнни, мы можем соединить их с помощью линий. Я не хочу вдаваться в математику алгоритма — она достойна отдельного поста — эта ссылка или ссылка выше поможет вам, если вас заинтересовал метод. Главное, что, применив это преобразование, мы получаем набор линий, каждая из которых, после небольшой дополнительной обработки и фильтрации, становится экземпляром класса Line с известным углом наклона и свободным членом.

**Оператор Собеля**

**Оператор Собеля** — это дискретный дифференциальный оператор, вычисляющий приближение градиента яркости изображения.  
Оператор вычисляет градиент яркости изображения в каждой точке. Так находится направление наибольшего увеличения яркости и величина её изменения в этом направлении. Результат показывает, насколько «резко» или «плавно» меняется яркость изображения в каждой точке, а значит, вероятность нахождения точки на грани, а также ориентацию границы.

В OpenCV оператор Собеля реализуется функцией cvSobel()  
Вычисление производной изображения (градиента), используя оператор Собеля (aperture\_size = 1,3,5,7) или Щарра (aperture\_size = -1):  
CVAPI(void) cvSobel( const CvArr\* src, CvArr\* dst,

int xorder, int yorder,

int aperture\_size CV\_DEFAULT(3));

src — исходное изображение  
dst — изображение для сохранения результа  
xorder — порядок производной по x (0,1 или 2)  
yorder — порядок производной по y (одновременно нулевой может быть только либо xorder, либо yorder)  
aperture\_size — размер ядра оператора Собеля (1,3,5,7)

**Центр масс**

**Преобразование Хафа (Hough Transform) —** это метод для поиска линий, кругов и других простых форм на изображении.  
Преобразование Хафа основывается на представлении искомого объекта в виде параметрического уравнения. Параметры этого уравнения представляют фазовое пространство (т.н. аккумуляторный массив/пространство, пространство Хафа).  
Затем, берётся двоичное изображение (например, результат работы детектора границ Кенни). Перебираются все точки границ и делается предположение, что точка принадлежит линии искомого объекта — т.о. для каждой точки изображения рассчитывается нужное уравнение и получаются необходимые параметры, которые сохраняются в пространстве Хафа.

**CV\_HOUGH\_STANDARD** — классический вариант трансформации Хафа. Каждая линия представляется двумя числами типа float (rho, theta), где rho — дистанция между точкой (0,0) и линией, а theta — угол между осью x и нормалью к линии (т.е. матрица должна иметь тип of CV\_32FC2)  
**CV\_HOUGH\_PROBABILISTIC** — вероятностный метод трансформации Хафа (более эффективен с случае изображений с несколькими длинными линейными сегментами). Возвращает сегменты линии, которые представляются точками начала и конца (т.е. матрица должна иметь тип of CV\_32SC4)  
**CV\_HOUGH\_MULTI\_SCALE** — масштабный вариант классической трансформации Хафа. Линии представлены так же, как и в CV\_HOUGH\_STANDARD

vector<Moments>mu(contours.size()); //создается вектор моментов, куда передается количество контуров, которое зависит от изображения

drawContours(drawing, contours, i, color, 2, 8, hierarchy, 0, Point()); //передается картинка, hierarchy - возвращает иерархию внешних контуров и отверстий.

circle(drawing, mc[i], 4, color, -1, 5, 0); //выбирается изображение, центр масс, 4 - радиус, цвета, толщина

**Git. Базовый курс**

Основы работы с Git

13 роликов научат вас основам работы с системы контроля версий Git.  
Вы узнаете, почему она важна в современной разработке. Изучите базовые понятия: репозиторий, коммиты, ветки и тэги. Освоите ключевые операции: создание и клонирование репозитория, слияние веток, запрос истории изменений и многое другое.

После обучения вы будете готовы использовать Git для подготовки домашних заданий на курсах по программированию на любом языке. Кроме того, вы приблизитесь к прохождению собеседования, где несколько вопросов обычно посвящены системе контроля версий.

Чему Вы научитесь

* Хранить свой код в интернете, и делать его доступным по сети всем участникам разработки;
* Писать код командно;
* Разрабатывать сразу несколько версий программы;
* Откатываться к предыдущим версиям вашей программы;
* Контролировать процесс выпуска различных версий вашей программы

Программа курса

Урок 1. Введение в Git

Урок 2. Установка и настройка Git

Урок 3. Основные команды терминала

Урок 4. Работа с репозиториями в Git

Урок 5. Управление файлами репозитория

Урок 6. История изменений

Урок 7. Работа с ветками репозитория

Урок 8. Публикация репозитория

Урок 9. Слияния веток

Урок 10. Управление версиями

Урок 11. Создание pull-request

Урок 12. Сложные операции

Урок 13. Работа с Fork-репозиториями

С тем, что получилось можно ознакомиться в презентации “Git. Базовый курс.pptx” [здесь](https://github.com/BulaninaNastya/BulaninaNastya_195/tree/main/Courses)

**WPF**

**Windows Presentation Foundation (WPF)** - аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык XAML.

В основе WPF лежит независимый от разрешения векторный модуль визуализации, использующий возможности современного графического оборудования. Возможности этого модуля расширяются с помощью комплексного набора функций разработки приложений, которые включают в себя язык XAML, элементы управления, привязку к данным, макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF является частью .NET, поэтому вы можете создавать приложения, включающие другие элементы .NET API.

Чтобы начать работу нам потребуется IDE Visual Studio 2022. Переходим на официальный сайт Microsoft и скачиваем [Visual Studio 2022 Community Edition](https://visualstudio.microsoft.com/ru/vs/community/) (рисунок 1).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.

После стандартной установки переходим к созданию проекта. Создаём проект и выбираем «Приложение WPF (NET.Framework)» (рисунок 2).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.

Называем наш проект. В окне MainWindow.xaml с помощью кода прописываем дизайн нашего калькулятора (рисунок 3).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3.

В файле MainWindow.xaml.cs напишем функции для калькулятора (рисунок 4).

Изображение выглядит как текст, монитор, снимок экрана, экран

Автоматически созданное описание

Рисунок 4.

После нам понадобится создать новый проект. Добавим в него библиотеку MaterialDesignThemes c помощью «Управления пакетами NuGet» (рисунок 5).

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Рисунок 5.

Чтобы подключить библиотеку к проекту нужно перейти на официальный сайт этой библиотеки, и, скопировав нужный код, вставить его в App.xaml (рисунок 6).

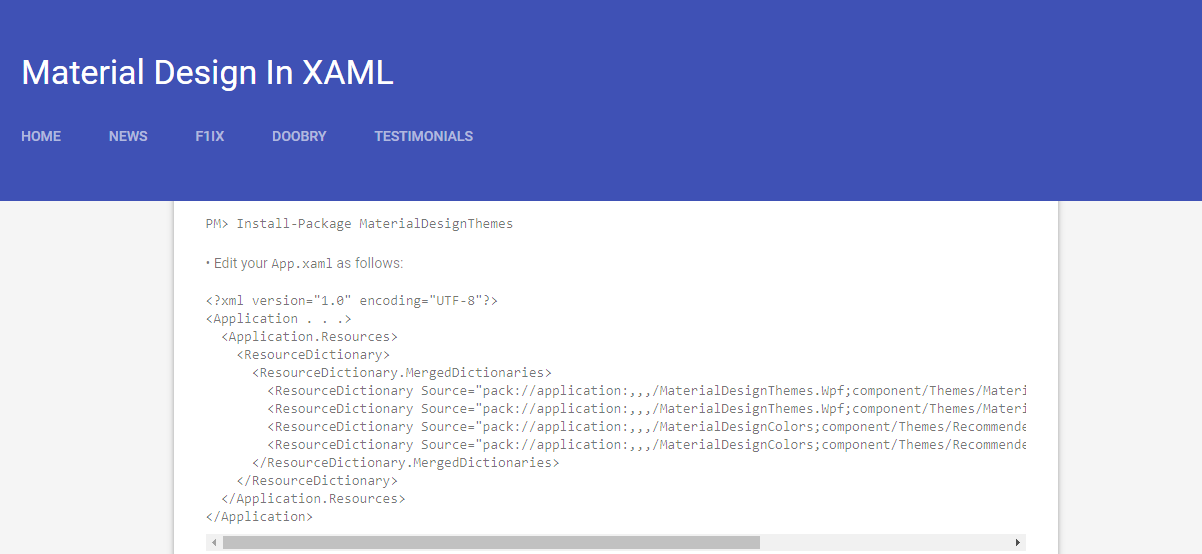


Рисунок 6.

В файле MainWindow.xaml мы можем прописать дизайн будущей программы (рисунок 7).

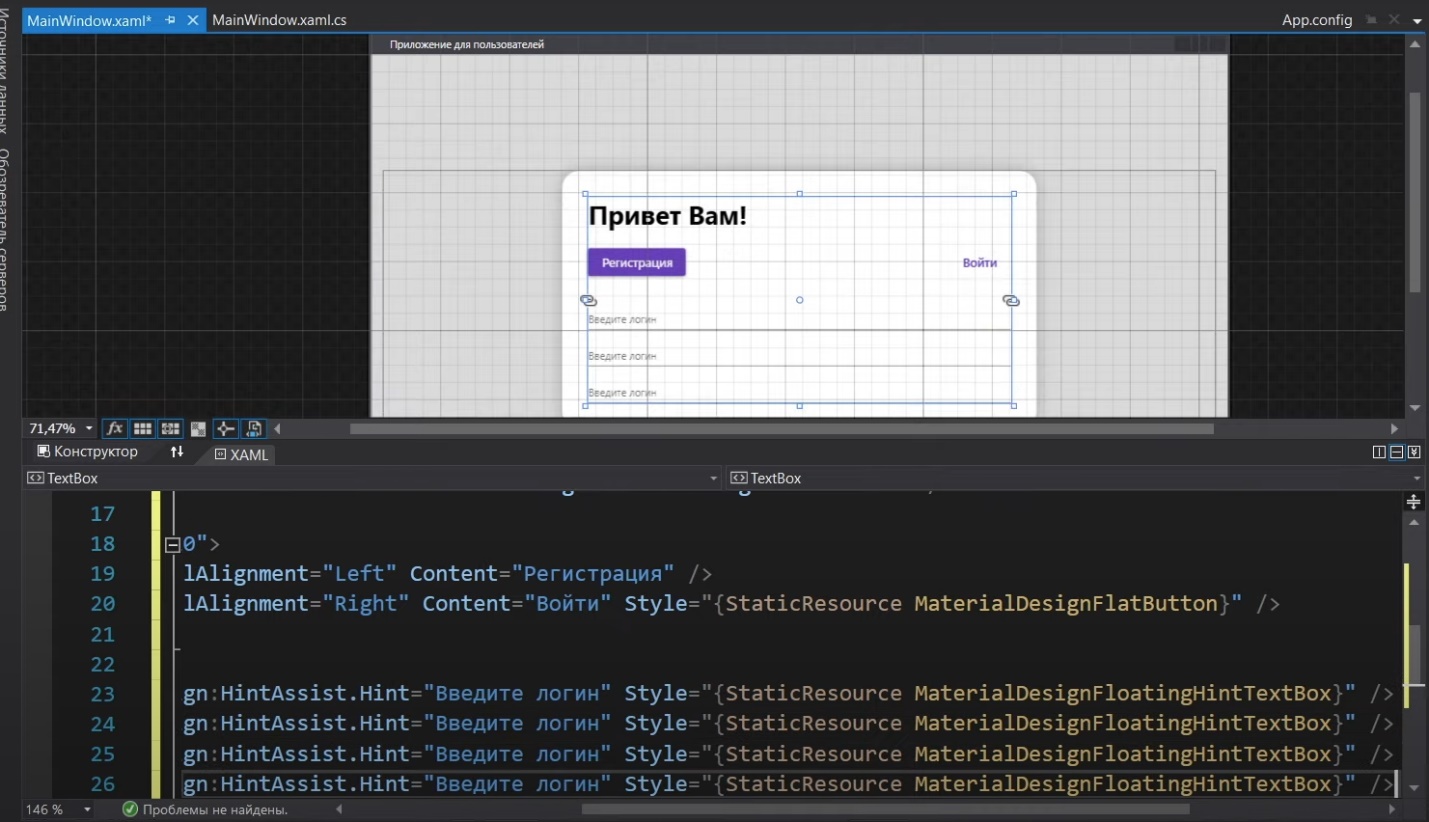


Рисунок 7.

Теперь приступим к созданию и подключению базы данных. Заранее скачанная с официального сайта программа Microsoft SQL Server Management Studio поможет нам в этом (рисунок 8).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8.

После установки вводим запрашиваемые программой данные и создаем базу данных.

В созданной базе данных нужно создать таблицу «Users» (рисунок 9).

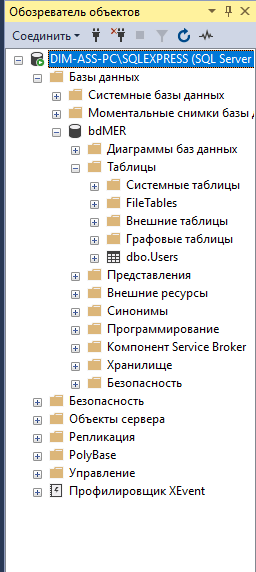


Рисунок 9.

Чтобы подключить базу к проекту нажимаем правой кнопкой мыши по файлу нашего проекта и нажимаем кнопку «Добавить», «Создать элемент». В окне «Добавление нового элемента» в поле «Поиск» вводим «edm» и добавляем Модель ADO.NET.EDM (рисунок 10.)

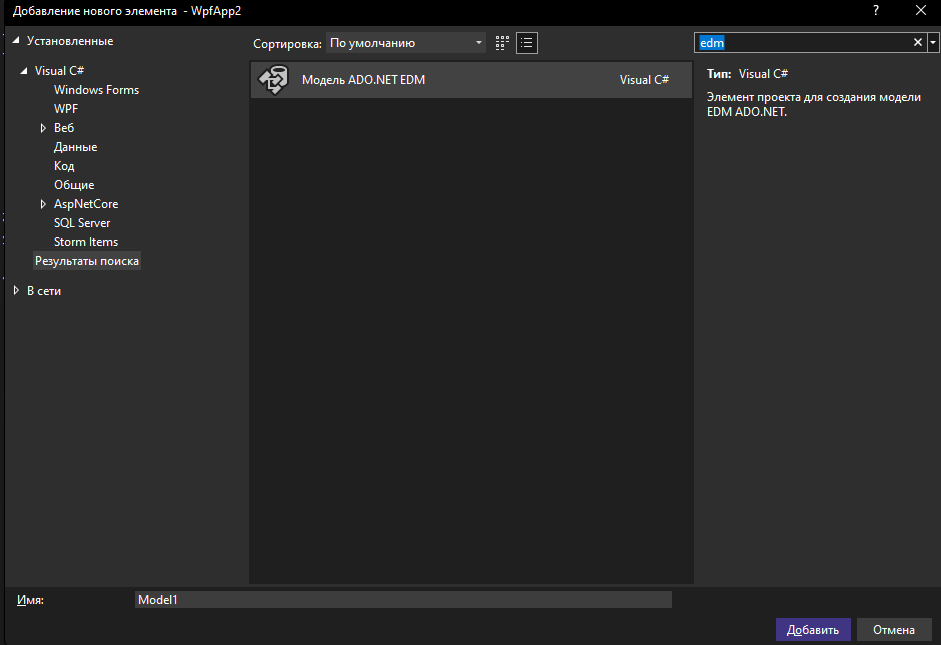


Рисунок 10.

Нам требуется ввести имя нашего сервера, выбрать нашу базу данных, далее выбрать таблицу «Users», после чего произойдет подключение (рисунок 11).

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Рисунок 11.

Создаём класс-модель для работы с таблицей (рисунок 12).

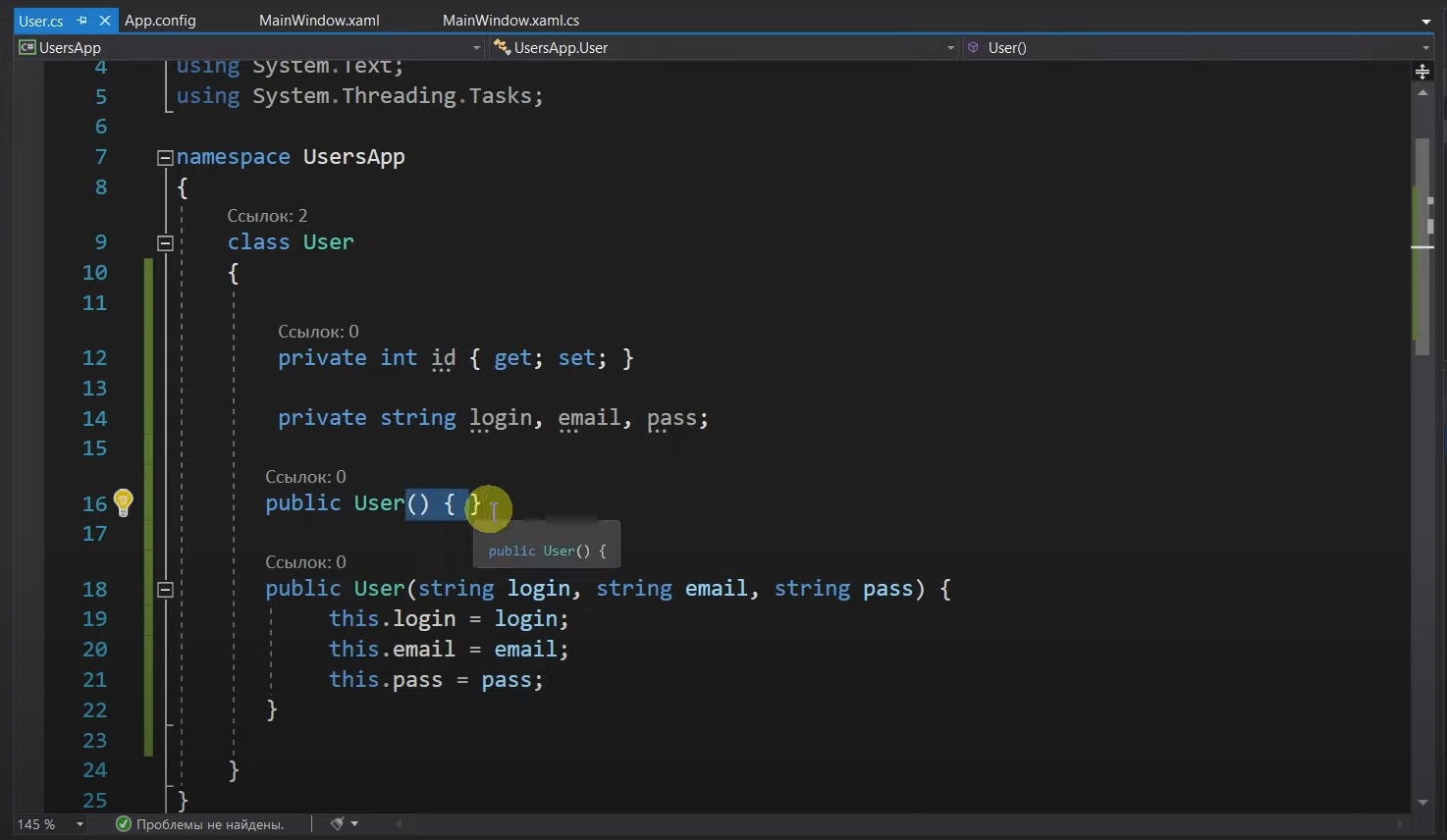


Рисунок 12.

Создаём класс для работы с базой данных (рисунок 13).

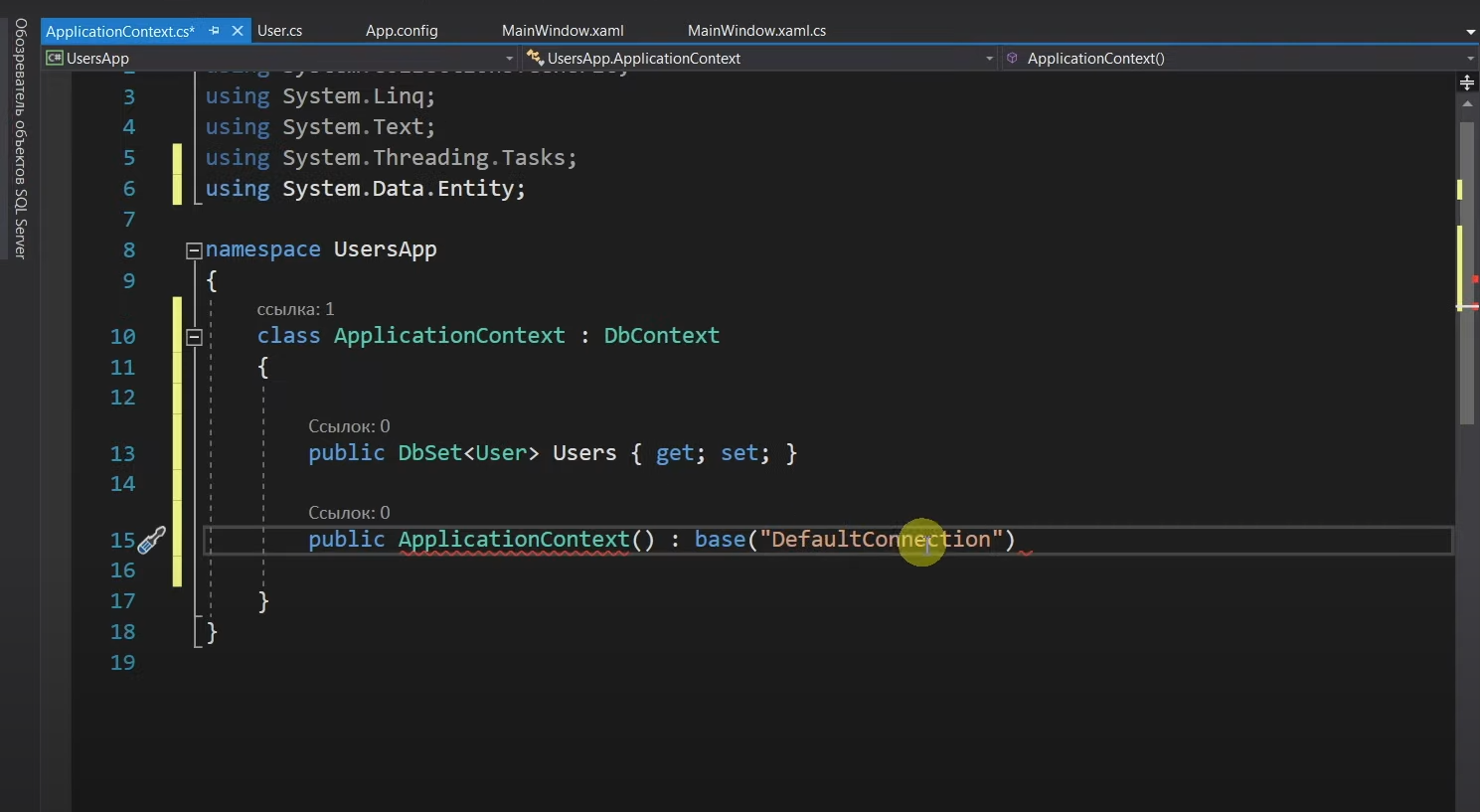


Рисунок 13.

Теперь добавим объект в БД (рисунок 14).

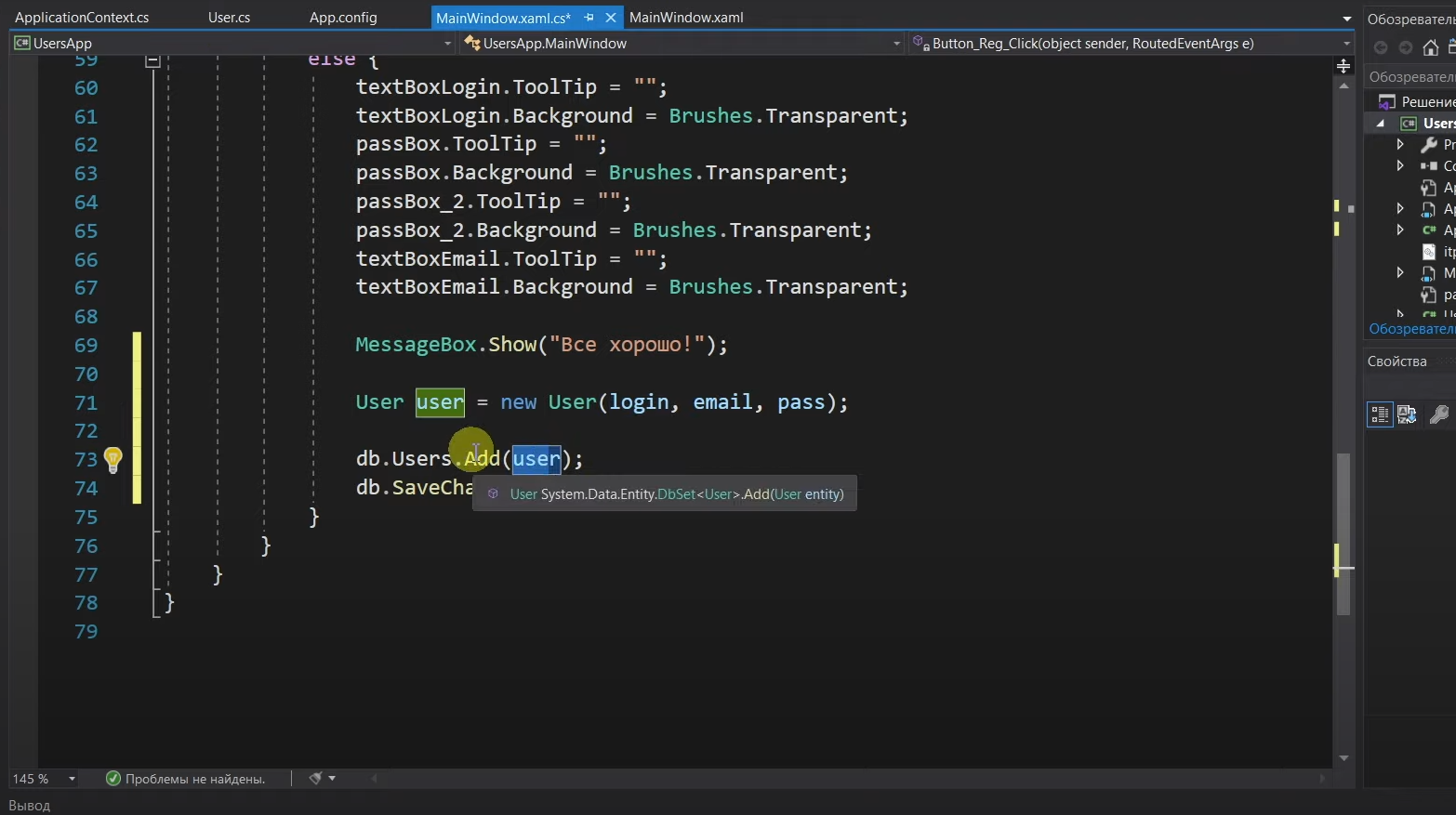


Рисунок 14.

Данные, которые мы добавили в таблицу нужно отобразить (рисунок 15).

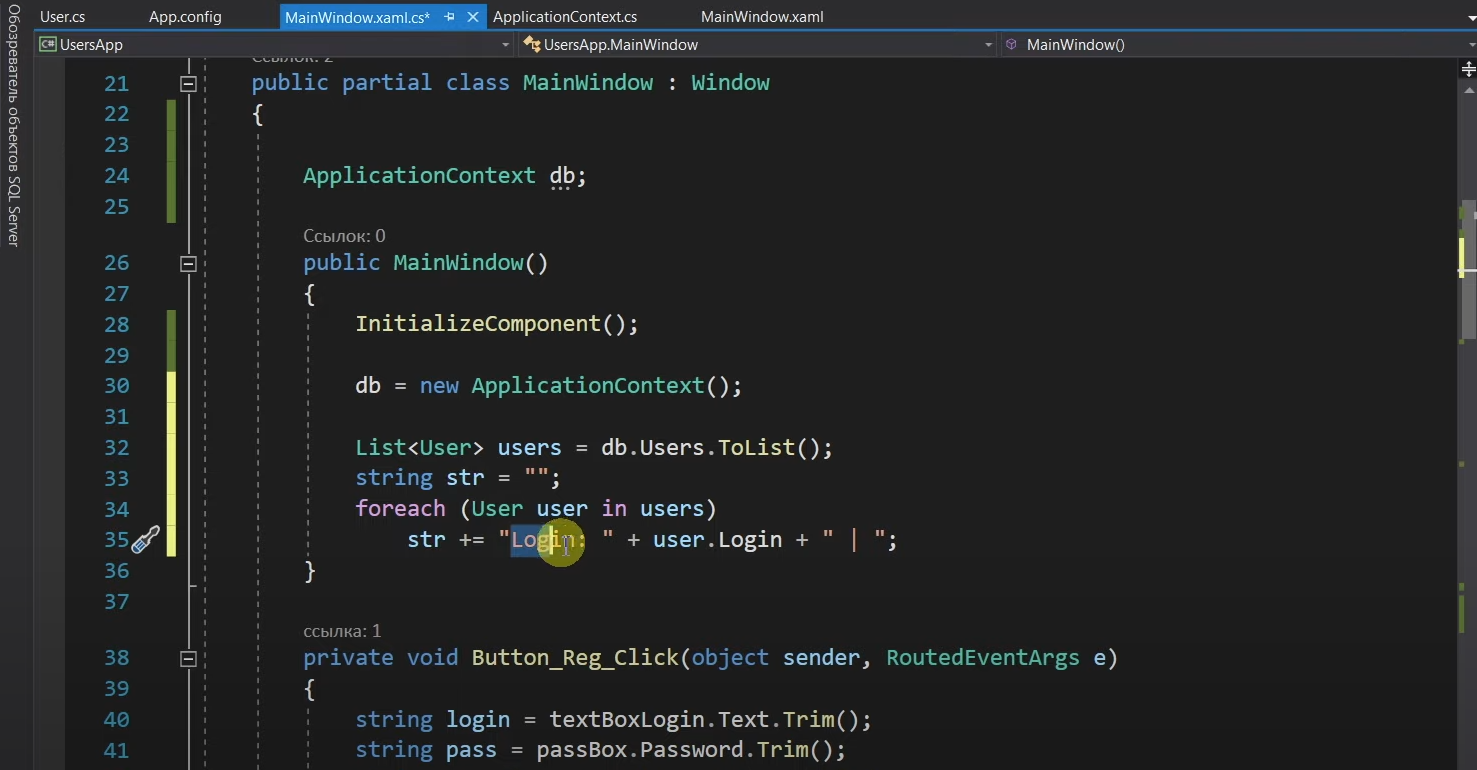


Рисунок 15.

Проверяем нашу программу (рисунок 16).

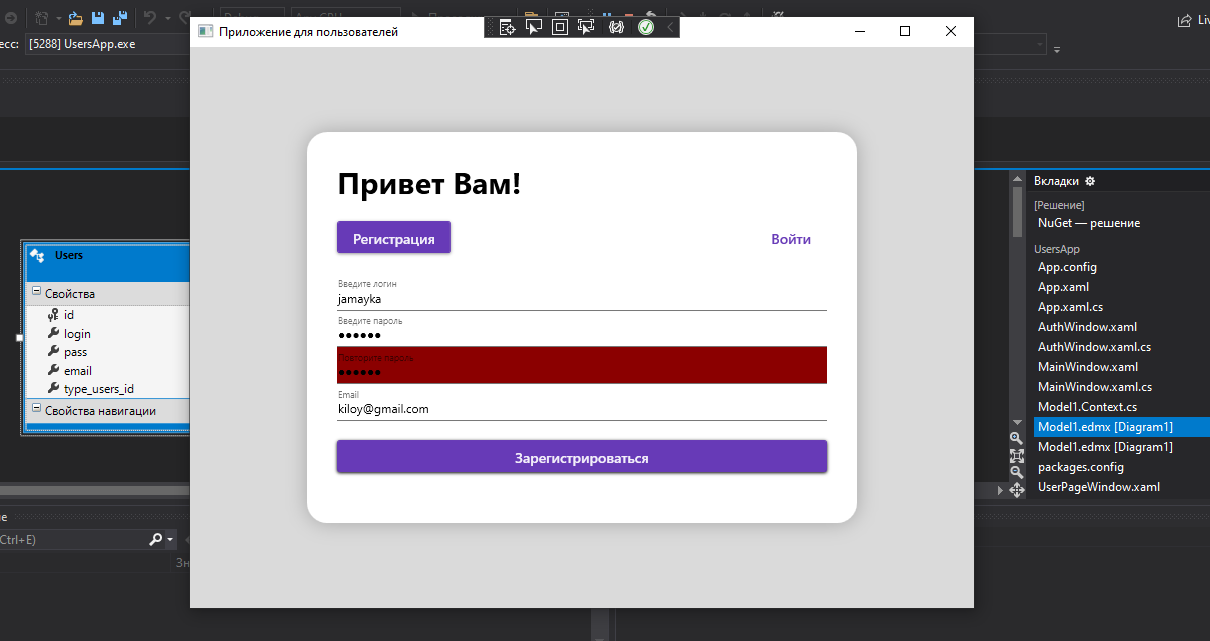


Рисунок 16.

После проверки нашей программы, переходим к окну с авторизацией. Создаём новое окно AuthWindow.xaml, в файле AuthWindow.xaml.cs прописываем код стиля, после чего выводится окно (рисунок 17).

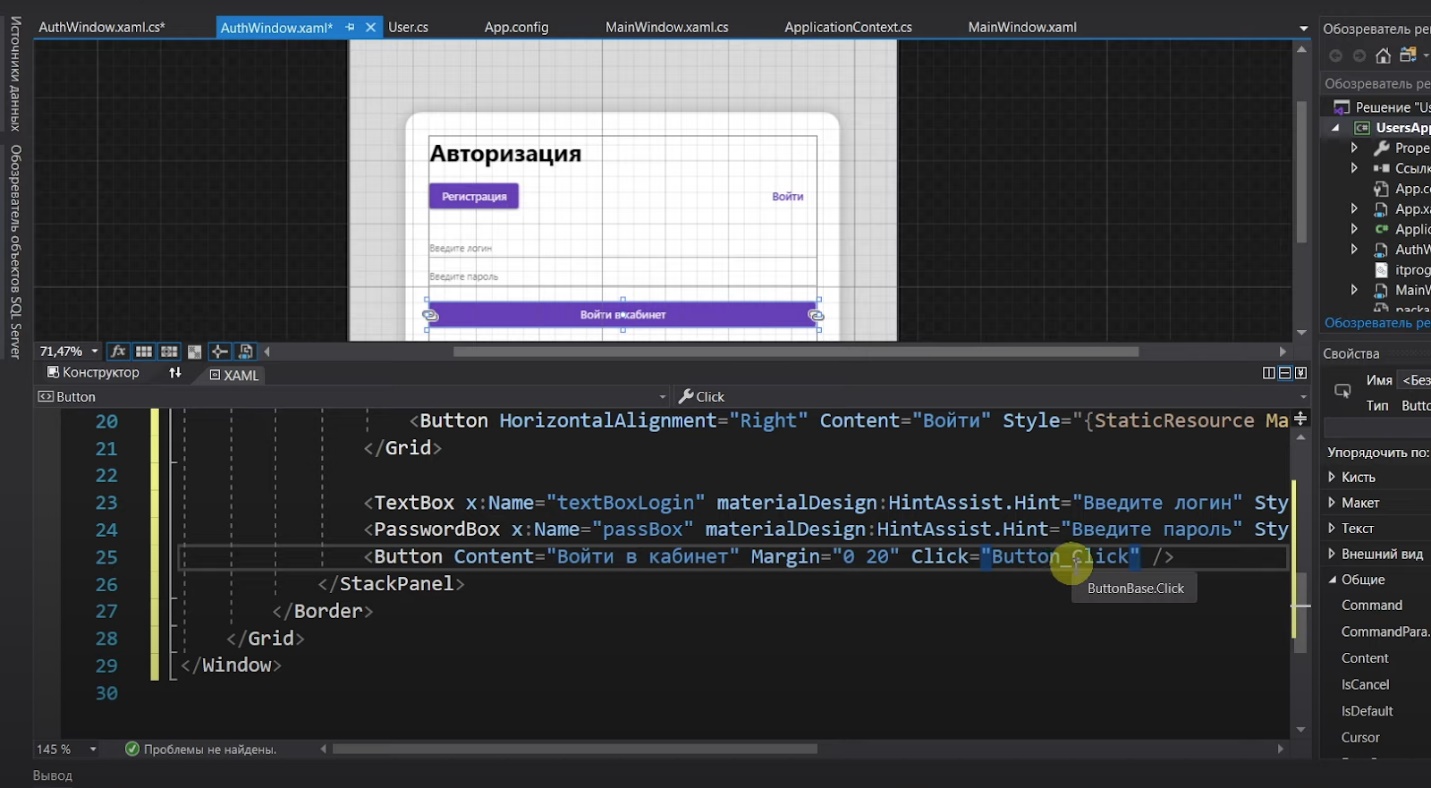


Рисунок 17.

Добавим проверку на введенные пользователем данные и данные из нашей Базы данных (рисунок 18).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 18.

Снова проверяем работу нашей программы (рисунок 19).

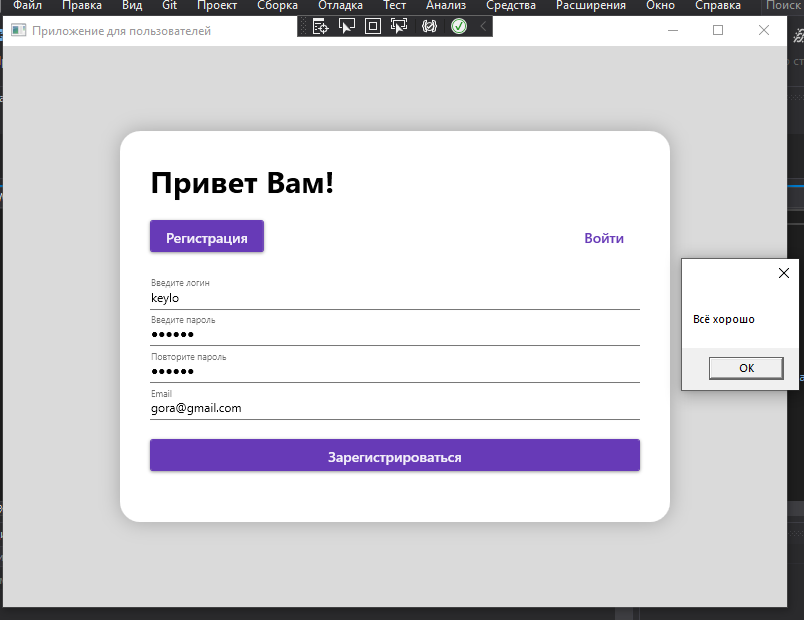


Рисунок 19.

На данном этапе переходим к созданию «Кабинета пользователя». Для этого нам понадобится настроить переход между страницами. Создаём само окно «Кабинет пользователя». Прописываем код вывода всех зарегистрированных пользователей и проверяем работу программы (рисунок 20).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20.

Создание нашего WPF проекта закончено. Весь процесс был изначально затеян для получения навыков работы с базами данных, создания переходов между окнами, добавления новых пользователей в базу данных и тому подобное.

**Выводы**

По окончании учебной практики были решены следующие задачи:

1. Знакомство с литературой
2. Изучить библиотеку OpenCV
3. Реализация приложения на WPF
4. Прохождение курса «Git. Базовый курс»

В ходе этой практики были сформированы следующие профессиональные компетенции:

ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств

ПК 1.4 Выполнять тестирование программных модулей

ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ.

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент.

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение.

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

**Список литературы**

* 1. AcademiaThemes. Портал информатики для гиков. – URL: <https://tinyurl.com/mzz9k3ys>
  2. Мюллер Д. П. C# для чайников / Мюллер Д. П. – Москва; Москва, 2019, – 608 с
  3. Семакин И. Г. Основы алгоритмизации и программирования : учебник для студ. учреждений сред. проф. образования / И. Г. Се- макин, А. П. Шестаков. — М. : Издательский центр «Академия», 2017. — 304 с
  4. Джепикс Филипп, Троелсен Эндрю. Язык программирования C# 7 и платформы .NET и .NET Core / Джепикс Филипп, Троелсен Эндрю – М.; Вильямс, 2018. – 1328. «Наука», 2017 – 87с
  5. <https://docs.google.com/presentation/d/17YdUQJrXn2dPckjSeDarp3v6mYDyaHhsLKEFe0GpqEk/edit#slide=id.g101651fe5da_0_13>
  6. <https://github.com/Sokolovafm?tab=repositories>
  7. <https://github.com/Alena1380>
  8. <https://github.com/Da1monMasa>
  9. <https://gb.ru/courses/1117>
  10. <https://github.com/Julia-Zhirnova/OpenCVexe>
  11. Орлов С.А. Программная инженерия. Технологии разработки программного обеспечения: учебник / С. А. Орлов. – 5-е изд., обновл. и доп. – Москва ; Санкт- Петербург:Питер, 2016
  12. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2017
  13. Семакин И. Г. Основы алгоритмизации и программирования : учебник для студ. учреждений сред. проф. образования / И. Г. Се- макин, А. П.
  14. Шестаков. — М. : Издательский центр «Академия», 2017
  15. Хабибулин И.Ш. Программирование на языке высокого уровня. C/C++. – СПб.: БХВ– Петербург, 2017
  16. Прайс Марк. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов / Прайс Марк – Москва; Вильямс, 2018
  17. Скит Джон. C# для профессионалов. Тонкости рограммирования /Скит Джон – Москва; Вильямс, 2019
  18. Мюллер Д. П. C# для чайников / Мюллер Д. П. – Москва; Москва, 2019