**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СОГЛАСОВАНО  Профессор департамента  программной инженерии  факультета компьютерных наук  канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Кириченко  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. |  | | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **С#-АНАЛИЗАТОР КЛАВИАТУРНОГО ПОЧЕРКА**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.509000-01 81 01-1-ЛУ** | | | | |
|  | |  | | |
| Исполнитель  студент группы 151 ПИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Н.А. Мариносян /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. | | |
|  | | | | |
|  | | | |  |

**2016**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.509000-01 81 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **С#-АНАЛИЗАТОР КЛАВИАТУРНОГО ПОЧЕРКА**  **Пояснительная записка**  **RU.17701729.509000-01 81 01-1**  **Листов 20** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**2016**

**АННОТАЦИЯ**

В данном программном документе приведена пояснительная записка к программе «С# Analyzer of Keystroke Dynamics.ехе» («C#-анализатор клавиатурного почерка»), предназначенной для идентификации людей по их клавиатурному почерку.

В данном программном документе, в разделе «Введение» указано наименование программы, краткое наименование программы и документы, на основании которых ведется разработка.

В разделе «Назначение и область применения» указано функциональное назначение программы, эксплуатационное назначение программы и краткая характеристика области применения программы.

В данном программном документе, в разделе «Технические характеристики» содержатся следующие подразделы:

* постановка задачи на разработку программы;
* описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами;
* описание и обоснование выбора метода организации входных и выходных данных;
* описание и обоснование выбора состава технических и программных средств.

В разделе «Ожидаемые технико-экономические показатели» указана предполагаемая потребность и экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами.

Настоящий документ разработан в соответствии с требованиями:

1. ГОСТ 19.101-77 Виды программ и программных документов [1];
2. ГОСТ 19.102-77 Стадии разработки [2];
3. ГОСТ 19.103-77 Обозначения программ и программных документов [3];
4. ГОСТ 19.104-78 Основные надписи [4];
5. ГОСТ 19.105-78 Общие требования к программным документам [5];
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7].

Изменения к данному Техническому заданию оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведенной в Приложении 1 настоящего технического задания.

СОДЕРЖАНИЕ

[**1.** **ВВЕДЕНИЕ** 4](#_Toc451260482)

[**1.1.** **Наименование программы** 4](#_Toc451260483)

[**1.2.** **Документы, на основании которых ведется разработка** 4](#_Toc451260484)

[**2.** **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ** 5](#_Toc451260485)

[**2.1.** **Назначение программы** 5](#_Toc451260486)

[**2.1.1.** **Функциональное назначение** 5](#_Toc451260487)

[**2.1.2.** **Эксплуатационное назначение** 5](#_Toc451260488)

[**2.2.** **Краткая характеристика области применения** 5](#_Toc451260489)

[**3.** **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ** 6](#_Toc451260490)

[**3.1.** **Постановка задачи на разработку программы** 6](#_Toc451260491)

[**3.2.** **Описание алгоритма и функционирования программы** 6](#_Toc451260492)

[**3.3.** **Обоснование выбора алгоритма решения задачи** 7](#_Toc451260493)

[**3.4.** **Описание и обоснование выбора метода организации входных и выходных данных** 7](#_Toc451260494)

[**3.4.1.** **Описание метода организации входных и выходных данных** 7](#_Toc451260495)

[**3.4.2.** **Обоснования выбора метода организации входных и выходных данных** 8](#_Toc451260496)

[**3.5.** **Описание и обоснование выбора состава технических и программных средств** 8](#_Toc451260497)

[**3.5.1.** **Состав технических и программных средств** 8](#_Toc451260498)

[**3.5.2.** **Обоснование выбора технических и программных средств** 8](#_Toc451260499)

[**4.** **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ** 9](#_Toc451260500)

[**4.1.** **Предполагаемая потребность** 9](#_Toc451260501)

[**4.2.** **Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами** 9](#_Toc451260502)

[**5.** **ИСТОНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ** 10](#_Toc451260503)

[**ПРИЛОЖЕНИЕ 1** 11](#_Toc451260504)

[**ТЕРМИНОЛОГИЯ** 11](#_Toc451260505)

[**ПРИЛОЖЕНИЕ 2** 12](#_Toc451260506)

[**ОПИСАНИЕ ФОРМАТА .bin** 12](#_Toc451260507)

[**ПРИЛОЖЕНИЕ 3** 13](#_Toc451260508)

[**ПРИЛОЖЕНИЕ 4** 14](#_Toc451260509)

1. **ВВЕДЕНИЕ**
   1. **Наименование программы**

Наименование программы – «С#-анализатор клавиатурного почерка».

Краткое наименование программы – «C# Analyzer of Keystroke Dynamics».

* 1. **Документы, на основании которых ведется разработка**

Разработка «C#-анализатора клавиатурного почерка» ведется на основании Приказа № 6.18.1-02/1112-19 от 11.12.2015 «Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук»

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. **Назначение программы**
      1. **Функциональное назначение**

Функциональным назначением программы является запоминание клавиатурного почерка одного или нескольких человек при наборе эталонной фразы и последующее распознавание человека, набравшего эталонную фразу повторно.

* + 1. **Эксплуатационное назначение**

Анализ клавиатурного почерка является востребованным на сегодняшний день ввиду повышенного внимания к информационной безопасности. Приложение «С#-анализатор клавиатурного почерка» может быть использовано для идентификации людей по такой биометрической характеристике, как клавиатурный почерк.

* 1. **Краткая характеристика области применения**

«С#-анализатор клавиатурного почерка» – программа, позволяющая идентифицировать людей по их клавиатурному почерку.

Данный программный продукт может быть использован в качестве биометрического средства аутентификации в целях повышения информационной безопасности какой-либо информационной системы.

1. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**

Разрабатываемая программа должна:

1. иметь функцию запоминания клавиатурного почерка человека;
2. иметь функцию удаления запомненного клавиатурного почерка;

3) распознавать человека по его клавиатурному почерку при повторном вводе эталонной фразы

* 1. **Описание алгоритма и функционирования программы**
     1. **Получение данных о клавиатурном почерке человека:**

Для хранения информации о клавиатурном почерке человека разработан специальный класс KeyStrokeRecorder, экземпляры которого с помощью таймера (Stopwatch) фиксирует время нажатия клавиш, время отпускания клавиш с помощью событий KeyDown и KeyUp в MainWindow.cs. На основе этих данных и подсчитывается время удержания каждой из клавиш и промежутки между их нажатиями, а все информация о клавиатурном почерке записывается в список, являющийся одним из полей класса KeyStrokeRecorder.

Также создан вспомогательный класс KeyStroke, который представляет собой значения нажатой клавиши и ее характеристики: время нажатия, время отпускания и продолжительность нажатия.

После завершения считывания данных о клавиатурном почерке человека создается аккаунт (класс Account), содержащий в себе имя пользователя, его пароль и его клавиатурный почерк для данного пароля.

После окончания работы программы создается экземпляр класса Database, в который записываются все созданные аккаунты (экземпляры класса Account) пользователей. Далее данный экземпляр класса Database с использованием встроенной сериализации записывается в файл «accounts.bin», из которого далее, при последующих запусках программы, и будет считываться вся информация о существующих пользователях.

* + 1. **Идентификация пользователя по клавиатурному почерку:**

Для идентификации пользователя ему предлагается ввести пароль. Программа считывает данные о его клавиатурном почерке с помощью экземпляра класса KeyStrokeRecorder. Затем в MainWindow в событии попытки входа в систему Next\_button3\_Click c помощью Евклидового расстояния проверяется сходство полученных данных о клавиатурном почерке с каждым из почерков из базы данных Database \_db, которые хранятся в списке matchedAccounts. Сравнение происходит по формуле:

Здесь xi и уi – соответсвующие характеристики почерка полеченного при входе в систему и почерка из базы данных.

List<Account> matchedAccounts = new List<Account>();

for (int i = 0; i < \_db.Accounts.Count; i++)

{

if (\_db.Accounts[i].Password == LoginPassword\_richTextBox.Text)

{

matchedAccounts.Add(\_db.Accounts[i]);

}

}

if (matchedAccounts.Count > 0)

{

Account tryLogIn = new Account("TryLogIn", LoginPassword\_richTextBox.Text, \_ksr.KeyStrokes);

double min = double.MaxValue;

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

{

if (min > matchedAccounts[i].CompareTo(tryLogIn))

{

min = matchedAccounts[i].CompareTo(tryLogIn);

bestMatch = matchedAccounts[i];

}

}

Минимальному Евклидовому расстоянию между клавиатурным почерком, полученным при попытке входа в систему, и каким-либо почерком из базы данных соответствует аккаунт искомого пользователя, т.к. его характеристики клавиатурного почерка ближе всего к почерку, полученному при попытке входа в систему.

Однако если разница между почерками слишком велика, это означает, что человек, который попытался войти в систему не является владельцем аккаунта. Поэтому программа запретит ему доступ. Для этого в том же событии проверяется, чтобы почерки отличались не более, чем на величину константы «error» (по умолчанию - эмпирически подобранная константа, которую можно изменить в настройках программы).

if (min < \_db.Error) //Right?

* + 1. **Удаление запомненного клавиатурного почерка пользователя**

Удаление пользователя осуществляется с помощью события Delete\_button\_Click в MainWindow. Информация о пользователе, нажавшем на кнопку «Удалить пользователя» полностью удаляется из базы данных \_db, а файл с базой данных перезаписывается при закрытии программы.

\_db.Accounts.Remove(\_db.Accounts.Find(x => x.Name == InfoUsername\_label.Text));

* 1. **Обоснование выбора алгоритма решения задачи**

При разработке программы был использован алгоритм сравнения клавиатурных почерков, основанный на расстоянии Евклида. Алгоритм был предложен заказчиком, т.е. научным руководителем.

* 1. **Описание и обоснование выбора метода организации входных и выходных данных**
     1. **Описание метода организации входных и выходных данных**

Входные данные программа получает с помощью класса KeyStrokeRecorder и событий KeyUp/KeyDown, когда пользователь вводит пароль на клавиатуре. Данные затем записываются в файл формата .bin с помощью встроенной функции сериализации [см. Приложение 2] (и считываются из него в дальнейшем с помощью встроенной функции десериализации).

Выходными данными можно считать идентифицированного пользователя, имя которого (выбранное имя при регистрации в системе) выводится на экран.

* + 1. **Обоснования выбора метода организации входных и выходных данных**

Механизм сериализации был выбран из-за того что сериализация позволяет:

1. сохранить объект в файле и обращаться к нему в любое время;
2. воссоздать объект в его первоначальную форму.

Сериализиция позволяет выгружать объекты класса, вместе со всеми свойствами автоматически, что существенно упрощает реализацию.

Также механизм сериализации/десериализации позволяет не потерять данные при сохранении и открытии файла. А также сократить количество программного кода.

* 1. **Описание и обоснование выбора состава технических и программных средств**
     1. **Состав технических и программных средств**

Для надёжной и бесперебойной работы программы требуется следующий состав технических средств [12]:

1) персональный компьютер, оснащенный 32-разрядным (x86) или 64-разрядным (x64) процессором с тактовой частотой 1 ГГц или выше;

2) 1 ГБ(для 32-разрядной системы) или 2 ГБ(для 64-разрядной системы) оперативной памяти или больше;

3) не менее 2 ГБ свободного места на жестком диске;

4) видеокарта и монитор, поддерживающие режим SVGA с разрешением не менее чем 850x600 точек;

5) совместимое указывающее устройство;

6) клавиатура;

7) CD-ROM привод;

8) операционная система Microsoft Windows 7 или более поздняя версия;

9) установленный Microsoft .NET Framework 4.0, требующий Windows Installer 5.0 или более поздняя версия.

* + 1. **Обоснование выбора технических и программных средств**

При реализации программы использовался язык С# 6.0 и Microsoft .NET Framework 4.0. Поэтому для надежной и бесперебойной работы указываются характеристики, соответствующие этим версиям программных средств, использованных при реализации:

Разрешение монитора не должно быть меньше размера окна приложения.

1. **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**
   1. **Предполагаемая потребность**

В настоящее время вопрос информационной безопасности является одним из самых важных, так как подавляющее большинство различных структур, в том числе государственных, используют в своей работе информационные технологии.

Запоминание и распознавание клавиатурного почерка человека востребовано в настоящее время, так как способно решить проблему информационной безопасности. Анализ клавиатурного почерка позволяет успешно идентифицировать пользователя, обладающего правами на работу с какой-либо информацией, и пресекать попытки несанкционированного получения доступа к этой информации со стороны других лиц. «С#-анализатор клавиатурного почерка» является востребованным во всех сферах человеческой деятельности, в которых важен вопрос безопасности информации, т.к. данное программное обеспечение позволяет отличать одного пользователя от других, т.е. идентифицировать пользователей.

В функционал данного программного продукта не входит непосредственная аутентификация пользователя посредством анализа его клавиатурного почерка с дальнейшей авторизацией. Предусмотрена только идентификация. То есть программа не может быть использована для предоставления пользователю доступа к какой-либо информации или отказа в доступе к ней. Однако «С#-анализатор клавиатурного почерка» может быть использован в качестве модуля для более сложной системы, предусматривающей аутентификацию пользователя и авторизацию в этой системе, с учетом особенностей организации, которая будет производить эксплуатацию продукта.

Стоит отметить, что для организации такого рода защиты информации не требуется никакого дополнительного оборудования (используется только ПК, мышь и клавиатура), поэтому использовать такой способ защиты своих данных может любой пользователь ПК, соответствующего системным требованиям данного продукта.

* 1. **Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами**

Быстрый поиск в сети Интернет на момент создания приложения не выявил аналогов данной программы.

Данное приложение:

* является бесплатным;
* может свободно распространяться;
* не требует вложения денежных средств во время использования;
* имеет неограниченный срок службы;

может быть модифицировано другими лицами в том случае, если они согласуют детали с разработчиком, который, в свою очередь, предоставит им доступ к исходному коду данного программного продукта.

1. **ИСТОНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**
2. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11. ГОСТ 15150-69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды. – М.: Изд-во стандартов, 1997.
12. Устинов В.. Надежность оптических дисков: как их правильно хранить и использовать. //Журнал «625» №7. М.: Издательство «625», 2005.
13. Системные требования ОС Windows 7. [Электронный ресурс]// URL: <http://windows.microsoft.com/systemrequirements?4bcfd458> (Дата обращения: 11.03.2016, режим доступа: свободный).
14. ГОСТ Р 7.02-2006 Консервация документов на компакт-дисках. Общие требования. – М.: ИПК Издательство стандартов, 2006.
15. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
16. Идентификация. [Электронный ресурс]// URL: <https://ru.wikipedia.org/wiki/Идентификация> (Дата обращения: 11.03.2016, режим доступа: свободный).
17. Аутентификация. [Электронный ресурс]// URL: <https://ru.wikipedia.org/wiki/Аутентификация> (Дата обращения: 11.03.2016, режим доступа: свободный).
18. Авторизация. [Электронный ресурс]// URL: <https://ru.wikipedia.org/wiki/Авторизация> (Дата обращения: 11.03.2016, режим доступа: свободный).

Описание формата .bin [Электронный ресурс]//URL: <https://ru.wikipedia.org/wiki/Двоичный_файл> (Дата обращения: 11.03.2016, режим доступа: свободный).

# ПРИЛОЖЕНИЕ 1

**ТЕРМИНОЛОГИЯ**

Ниже приведен список необходимых терминов для ознакомления:

* Клавиатурный почерк – поведенческая биометрическая характеристика человека, обуславливающаяся особенностями его работы с клавиатурой, такими как время удержания нажатой клавиши, время паузы между нажатием клавиш и частота нажатия клавиш клавиатуры.
* Идентификация – присвоение субъектам и объектам идентификатора и (или) сравнение идентификатора с перечнем присвоенных идентификаторов. В данном случае идентификатором служит клавиатурный почерк пользователя. [15]
* Аутентификация – проверка подлинности предъявленного пользователем идентификатора. В данном случае идентификатором служит клавиатурный почерк пользователя. [16]
* Авторизация – предоставление пользователю прав доступа к ресурсам, определенным для выполнения его задач. [17]

# ПРИЛОЖЕНИЕ 2

## **ОПИСАНИЕ ФОРМАТА .bin**

Двоичный (бинарный) файл — последовательность произвольных байтов. Название связано с тем, что байты состоят из бит, то есть двоичных (англ. binary) цифр.

При этом программа «C#-анализатор клавиатурного почерка» работает только с файлами данного формата, созданными ею самой.

Файл используется программой и не предназначен для открытия и/или изменения самим пользователем/другим ПО.

**ПРИЛОЖЕНИЕ 3**

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

Таблица 3.1 – Описание и функциональное назначение классов

|  |  |
| --- | --- |
| Класс | Назначение |
| MainWindow | Главное окно приложения, в котором содержатся обработчики событий. В нем также организовано взаимодействие объектов других классов. |
| KeyStrokeDynamics | Класс, осуществляющий запись клавиатурного почерка пользователя |
| KeyStroke | Класс, представляющий значения клавиш и их характеристик, используемых для определения клавиатурного почерка. |
| Account | Класс, содержащий имя пользователя, его пароль и его клавиатурный почерк для данного пароля. |
| Database | Сериализуемый класс, содержащий все аккаунты пользователей. Используется для записи в файл и чтения из него. |
| TablessControl | Класс, представляющий собой компонент формы. Позволяет реализовать элемент формы с вкладками, которые не видны пользователю. |

**ПРИЛОЖЕНИЕ 4**

**ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ МЕТОДОВ И СВОЙСТВ**

Таблица 4.1 – Описание полей методов и свойств класса MainWindow.cs

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| \_ksr | | private | | KetStrokeRecorder | | Текущий экземпляр объекта для записи клавиатурного почерка | |
| \_userInfo | | private | | Account | | Лист с информацией о вершинах. | |
| \_db | | private | | Databse | | Массив массивов, содержащий матрицы смежности. | |
| \_kdrList | | private | | List<KeyStrokeRecorder> | | Массив массивов, содержащий численные значения цветов. | |
| \_username | | private | | string | | Имя текущего пользователя | |
| \_password | | private | | string | | Пароль текущего пользователя | |  | |  |
| bestMatch | | private | | Account | | Аккаунт из базы данных с наименьшим отличием клавиатурного почерка | |
| firstKey | | private | | bool | | Переменная указывающая, была ли клавиша нажата первой | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| MainWindow | public | | – | | – | | Конструктор, задает начальные значения и выполняет подготовку к работе программы. |
| Input\_richTextBox\_KeyDown | public | | void | | object sender, KeyEventArgs e | | Метод обработки нажатия клавиши |
| Input\_richTextBox\_KeyUp | public | | void | | object sender, KeyEventArgs e | | Метод обработки отпускания клавиши |
| Next\_button1\_Click | private | | void | | object sender, EventArgs e | | Метод обработки события щелчка по кнопке «Вперед». Выполняет необходимые проверки и осуществляет переход на следующую вкладку. |
| Next\_button2\_Click | private | | void | | object sender, EventArgs e | | Метод обработки события щелчка по кнопке «Вперед». Выполняет необходимые проверки и осуществляет переход на следующую вкладку. |
| Next\_button3\_Click | private | | void | | object sender, EventArgs e | | Метод обработки события щелчка по кнопке «Вперед». Выполняет необходимые проверки, создает новый аккаунт и осуществляет переход на следующую вкладку. |
| Back\_button2\_Click | private | | void | | object sender, EventArgs e | | Возвращает пользователя на предыдущую вкладку |
| Login\_button\_Click | private | | void | | object sender, EventArgs e | | Запускает процесс идентификации пользователя |
| newUser\_button\_Click | private | | void | | object sender, EventArgs e | | Запускает процесс создания нового пользователя |
| Back\_button3\_Click | private | | void | | object sender, EventArgs e | | Возвращает пользователя на стартовую страницу |
| MainWindow\_FormClosing | private | | void | | object sender, EventArgs e | | Записывает данные о пользователях в файл при закрытии программы |
| Logut\_button\_Click | private | | void | | object sender, EventArgs e | | Выполняет выход из аккаунта |
| LoginPassword\_richTextBox\_KeyPress | private | | void | | object sender, KeyEventArgs e | | Событие обработки нажатой клавиши. Используется для проверки корректности вводимых данных при входе в систему |
| Delete\_button\_Click | private | | void | | object sender, EventArgs e | | Выполняет удаление данных о пользователе |
| LoginPassword\_richTextBox\_KeyPress | private | | void | | object sender, KeyEventArgs e | | Событие обработки нажатой клавиши. Используется для проверки корректности вводимых данных при регистрации |
| NextKDR3\_button\_Click | private | | void | | object sender, EventArgs e | | Создает нового пользователя по полученным данным |
| NextKDR2\_KeyPress | private | | void | | object sender, KeyEventArgs e | | Событие обработки нажатой клавиши. Используется для проверки корректности вводимых данных при регистрации | Событие обработки нажатой клавиши. Используется для проверки корректности вводимых данных при регистрации |
| NextKDR3\_KeyPress | private | | void | | object sender, KeyEventArgs e | | Событие обработки нажатой клавиши. Используется для проверки корректности вводимых данных при регистрации |
| оПрограммеToolStripMenuItem1\_Click | private | | void | | object sender, EventArgs e | | Открывает окно с информацией о программе |
| созданиеНовогоПользователяToolStripMenuItem\_Click | private | | void | | object sender, EventArgs e | | Открывает окно с информацией о том, как создать нового пользователя |
| входВСистемуToolStripMenuItem\_Click | private | | void | | object sender, EventArgs e | | Открывает окно с информацией о том, как выполнить вход в систему |
| Settings\_button\_Click | private | | void | | object sender, EventArgs e | | Открывает вкладку с настройками |
| PreferencesOK\_button\_Click | private | | void | | object sender, EventArgs e | | Принимает внесенные изменения и переходит в «Личный кабинет» |
| error\_trackBar\_Scroll | private | | void | | object sender, EventArgs e | | Меняет отображаемое текущее значение максимальный разница почерков, допустимой при входе в систему |

Таблица 4.2 – Описание полей методов и свойств класса KeyStrokeRecorder.cs

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| \_sw | | private readonly | | Stopwatch | | Таймер | |
| \_dict | | private readonly | | Dictionary<Keys, KeyStroke> | | Словарь, в котором каждой клавише ставится в соответствие экземпляр класса KeyStroke с ее характеристиками | |
| KeyStrokes | | public | | List<Keystroke> | | Список с данными о клавиатурном почерке | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| KeyStrokes | public | | конструктор | | – | | Конструктор. |
| IsKeyPressed | public | | bool | | Keys k | | Указывает, была ли клавиша уже нажата или нет |
| KeyPressed | public | | void | | Keys k | | Добавляет клавишу в словарь, записывает время нажатия |
| KeyReleased | public | | void | | Keys k | | Удаляет клавишу из словаря и добавляет ее характеристики в список KeyStrokes |
| Start | public | | void | | – | | Запускает/перезапускает таймер \_sw |
| Stop | public | | void | | – | | Останавливает таймер \_sw |

Таблица 4.3 – Описание полей методов и свойств класса KeyStroke.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | Назначение | |
| Key | public | Keys | Клавиша | |
| Pressed | public | TimeSpan | Время нажатия | |
| Released | public | TimeSpan | Время отпускания | |
| **Свойства** | | | | |
| Имя | Модификатор доступа | Тип | Доступ | Назначение |
| PressTime | public | TimeSpan | get | Возвращает время удержания клавиши |

Таблица 4.4 – Описание полей методов и свойств класса Account.cs

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| \_username | | private | | string | | Имя пользователя | |
| \_password | | private | | string | | Пароль пользователя | |
| \_keystrokeDynamics | | private | | List<KeyStroe> | | Список с характеристиками клавиш | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| Account | public | | конструктор | | string Username, string Password, List<KeyStroke> KeystrokeDynamics | | Конструктор. |
| CompareTo | public | | double | | Account ac | | Сравнивает клавиатурные почерки аккаунта с аккаунтов в аргументе и возвращает разницу |
| Interval | public | | double | | int index | | Возвращает интервал клавиши с указанным индексом |
| PressTime | public | | double | | int index | | Возвращает время удержания клавиши с указанным индексом |
| Key | public | | Keys | | int index | | Возвращает значение клавиши с указанным индексом |
| **Свойства** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Доступ | | Назначение |
| Name | public | | string | | get | | Возвращает имя пользователя |
| Password | public | | string | | get | | Возвращает пароль пользователя |

Таблица 4.5 – Описание полей методов и свойств класса Database.cs

|  |  |  |  |
| --- | --- | --- | --- |
| **Поля** | | | |
| Имя | Модификатор доступа | Тип | Назначение |
| Accounts | public | List<Account> | Список аккаунтов |
| Error | public | int | Значение максимального допустимого отличия в почерках |

Таблица 4.6 – Описание полей методов и свойств класса TablessControl.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| WindProc | protected (override) | void | Point, Point | Метод, «прячущий» вкладки путем блокировки TCM ADJUSTRECT message. |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |