1 АНАЛИЗ СОСТОЯНИЯ ПРОБЛЕМЫ

1.1 Использование клавиатурного почерка для идентификации пользователей

Клавиатурный почерк входит в состав большого класса биометрических параметров человека, известного как поведенческая биометрия. Общеизвестно, что поведенческая биометрия не обеспечивает необходимый уровень надёжности, в отличие от физической биометрии, используемой для аутентификации, например, с помощью, отпечатков пальцев, сканирования сетчатки глаза или ДНК, но может успешно применяться вкупе с другими средствами идентификации.

1.1.1 Понятие клавиатурный почерк. Параметры клавиатурного почерка

Клавиатурный почерк – это подробная динамическая информация, которая точно описывает особенности набора текста на клавиатуре конкретным человеком.

Клавиатурный почерк представляет собой биометрический шаблон, используемый для идентификации человека, основанный на схеме ввода, ритме и скорости ввода на клавиатуре [15]. Например, установлено, что время, необходимое для нажатия клавиши, время поиска клавиши, и время удержания клавиши может быть очень характерным для человека, независимо от того, насколько быстро он печатает. У большинства людей есть конкретные буквы, поиск и нажатие на которые занимает больше времени. Правши статистически быстрее в наборе букв, которые они нажимают пальцами правой руки, чем они же пальцами левой руки.

В настоящее время исследователи заинтересованы в использовании динамической информации о нажатии клавиш, которая обычно не используется

для проверки и тем более, для определения личности пользователя персонального компьютера, работающего за клавиатурой. Методы идентификации человека, используемые в поведенческой биометрии, широко варьируются по сложности, от простых статистических методов до использования искусственного интеллекта.

Использование клавиатурного почерка для идентификации пользователя имеет много плюсов:

– алгоритмическая реализация алгоритмов определения компьютерного почерка достаточно проста и может быть реализована как в виде программы, так и отдельного устройства;

– дешевизна реализации, для работы алгоритма достаточно обычной клавиатуры, которая по умолчанию входит в комплект любого компьютера;

– алгоритм можно использовать неявно, пользователь может и не подозревать что используется дополнительно средство идентификации;

– достаточно высокая эффективность.

Однако, данный метод имеет недостатки:

– требуется этап обучения и получения эталонных значений для сравнения;

– при смене клавиатуры, с высокой вероятность понадобится заново собирать статистику;

– так как поведенческая биометрия напрямую зависит от состояния человека, при его изменении (болезнь, усталость, волнение), система может не опознать пользователя.

Динамика нажатия клавиш обладает широким набором информации для сбора, в том числе [15]:

1) интервал между последовательными нажатиями клавиш;

2) время удержания клавиш;

3) время «полёта»;

4) время ожидания;

5) общая скорость набора (количество символов в единицу времени);

6) частота ошибок набора;

7) анализ использования характерных клавиш управления и клавиш управления курсором.

На рисунке 15 на временной шкале изображены основные параметры для сбора информации о динамике нажатий клавиш пользователем.

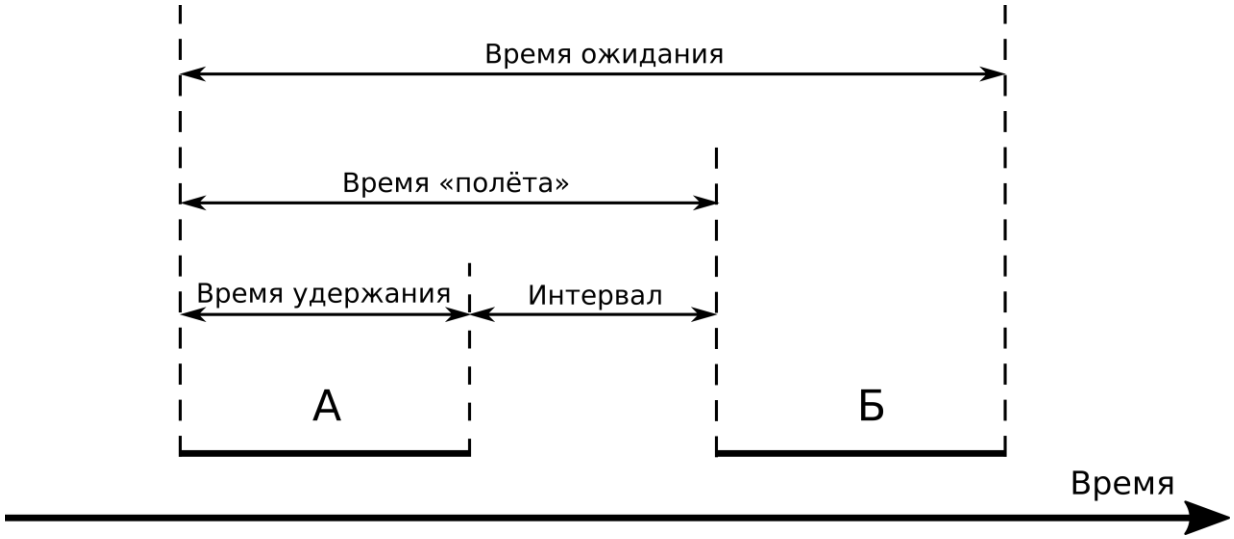


Рисунок 1 – Параметры динамики нажатий клавиш

На рисунке 15 выделены следующие параметры динамики нажатий клавиш:

– время удержания – время, в течение которого пользователь удерживает клавишу перед тем как отпустить её;

– интервал – время между отпусканием первой клавиши и нажатием на следующую;

– время «полёта» – время между последовательными попарными нажатиями на клавиши;

– время ожидания – время между нажатием на первую клавишу и отпусканием последней, применимо для *n*-грамм фиксированной длины.

При реализации алгоритма, для наибольшей достоверности оценки показателей пользователя необходимо учитывать все вышеуказанные параметры.

1.2 Обзор алгоритмов определения клавиатурного почерка

Задача распознавания пользователя решается либо путем перебора информации из базы данных, либо с помощью сравнения с неким шаблоном поведения. Существуют несколько подходов по определению клавиатурного почерка.

1. Наиболее простой метод определения компьютерного почерка состоит в сборе информации о динамических характеристиках наиболее часто встречающихся для данного языка диграмм, тригамм или в более общем случае *n*-грамм из всего текста [15]. В этом случае параметры собираются на основе ввода фиксированных текстов (статически) или переменных текстов (динамически). В случае последнего метода статистику можно собирать непрерывно, чтобы постоянно идентифицировать пользователя. Собранные данные усредняются и на их основе создаётся эталон для сравнения. Верификация заключается в вычислении разницы сравнения тестовой подписи (полученной в момент сбора) и эталоном. Положительное решение принимается в случае, если разница не превышает принятое пороговое значение.

2. Метод, разработанный Риком Джойсом и Гопалом Гупта, который заключается в представлении динамических данных в виде вектора *M,* вычисляемого по формуле [15]:

$$M = \{M_{\text{имяпользователя}}, M_{\text{пароль}}, M_{\text{имя}}, M_{\text{фамилия}}\} \quad (1)$$

Пусть $M = (m_1, m_2, \dots, m_n)$ и $T = (t_1, t_2, \dots, t_n)$, где *n* – общее количество задержек в подписи. Верификация заключается в сравнении тестовой подписи *T* (полученной при входе в систему) с *M*, вычисляя норму L_1 по формуле:

$$L_1 = \|M - T\|_1 \quad (2)$$

полученной из

$$\sum_{i=1}^{n} |m_i - t_i|. \quad (3)$$

Положительное решение принимается в случае, если величина L_1 не превышает принятое пороговое значение.

3. Метод разработанный Фабианом Монроузом, базирующийся на исследования Джойса и Гупта и использовании нейронных сетей [15, 15]. Данные о пользователях представлялись в виде *N*-мерных векторов характеристик, которые обрабатываются методом Джойса и Гупта. Полученные данные разделялись на наборы для обучения и проверки. Для идентификации пользователя используются следующие классификаторы:

– Евклидово расстояние между двумя *N*-мерными векторами *U* и *R*. Пусть $U = [u_1, u_2, \dots, u_n]$, $R = [r_1, r_2, \dots, r_n]$ тогда Евклидово расстояние между ними определяется как:

$$D(R, U) = [\sum_{i=1}^{n}(r_i - u_i)^2]^{1/2} \quad (4)$$

Для неизвестного *U* из тестового набора $D(R_i, U), i = 1,2, \dots, n$, где *n* – количество проверочных наборов. В результате выбирается набор с минимальным расстоянием *U*.

– невзвешенная вероятность: пусть *U* и *R* – *N*-мерные вектора, определенные выше. Каждая компонента векторов состоит из четырёх компонент и включает в себя набор $\langle \mu_i, \sigma_i, o_i, X_i \rangle$, состоящий из средней величины μ_i, стандартного отклонения σ_i, частоты события o_i и значения *X* для *i*-ой характеристики. Предполагая, что каждая компонента подчиняется нормальному распределению, оценка между опорным профилем *R* и неизвестным профилем *U* вычисляется как:

$$Score(R, U) = \sum_{i=1}^{n} S_{ui}, \quad (5)$$

где

$$S_{ui} = \frac{1}{o_{ui}}\left[\sum_{j=1}^{o_{ui}} Prob\left(\frac{x_{ij}^{(u)} - \mu_{ri}}{\sigma_{ri}}\right)\right], \quad (6)$$

где $x_{ij}^{(u)}$ – *j*-ое появление *i*-ой характеристики в *U*.

Другими словами, баллы для каждого u_i зависят от вероятности наблюдения величины u_{ij} в профиле *R* при средней (μ_{ri}) и стандартном отклонении (σ_{ri}) для данной характеристики в *R*. Большую вероятность имеют

значения u_i, которые находятся ближе к μ_{ri}, меньшую – те, что дальше. Вектор *U* ассоциируется с ближайшим соседом в базе данных, т. е. с оператором, для которого вектор характеристик будет максимальным;

– взвешенная вероятность: некоторые параметры являются более надежными, чем другие, потому что они представлены в большем количестве профилей или имеют более высокий процент в письменном языке; например, английские *er*, *th*, *re* имеют больший вес чем *qu* или *ts*. Исходя из этого вводится понятие веса, тогда оценка между профилями *R* и *U* вычисляется как

$$Score(R, U) = \sum_{i=1}^{n}(S_{ui} * w_{ui}), \quad (7)$$

где вес характеристики u_i определяется как соотношение частоты её появления к частоте появления всех характеристик в *U*. Характеристики с большим весом рассматриваются как более надежные. Предположим, что каждая характеристика пользователя подвержена нормальному распределению, тогда баллы в пользу профиля *R* при введенном *U* определяются максимальными баллами и ближайшим вектором характеристик.

Использование данных классификаторов позволяет распознать человека в 87,18 % случаев. При использовании Байесовых классификаторов процент распознания вырастает до 92,14 %, ещё 5 % можно получить подбирая весовые коэффициенты.

1.3 Определение функциональных и технических требований к программному средству

Так как программа должно собирать клавиатурный ввод пользователя напрямую из операционной системы в различных приложениях, в реальном времени, разрабатываемое приложение должно представлять собой приложение для персонального компьютера.

Все требования к разрабатываемому приложению можно разбить на технические требования и требования по функциям.

Техническими требованиями к разрабатываемому приложению являются:

1) операционная система Windows XP и выше;

2) виртуальная машина Java версии 8v131 и выше;

3) HID-совместимая клавиатура.

Разрабатываемое программное приложение должно обладать следующими функциями:

1) сбор всего пользовательского ввода с клавиатуры, у учётом «белого» списка приложений;

2) накопление полученной информации в маршализованных данных;

3) создание и сохранение шаблона пользовательского поведения;

4) вычисление текущего пользователя находящегося за клавиатурой.

1.4 Постановка цели и задач выпускной квалификационной работы

Программа для определения параметров клавиатурного почерка может использоваться в различных учреждения и организациях для контроля учётных записей, как простое средство идентификации или для оценки физиологического состояния человека.

Таким образом, сформулирована цель выпускной квалификационной работы: разработка и реализация программного средства для определения параметров клавиатурного почерка пользователя персонального компьютера.

Для достижения указанной цели необходимо выполнить следующие задачи:

1) изучить особенности и способы определения компьютерного почерка пользователей;

2) провести анализ и выбрать средства для реализации программного средства;

3) разработать формат хранения пользовательских данных и алгоритм работы программы;

4) реализовать программу для определения параметров клавиатурного почерка пользователя.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1 Выбор инструментов для реализации программного средства определения параметров клавиатурного почерка пользователя персонального компьютера

В настоящее время существует огромное множество различных языков программирования, библиотек и вспомогательных средств для написания настольных приложений, которые могут использоваться для программной реализации средства определения клавиатурного почерка пользователя персонального компьютера.

2.1.1 C++ и фреймворк Qt

C++ – компилируемый, статически типизированный язык программирования общего назначения. Поддерживает процедурное, объектно-ориентированное и обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включающую часто используемые структуры данных и алгоритмы, ввод-вывод, регулярные выражения, многопоточность и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. C++ 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 развлекательных приложений (игр).

Вкупе с фреймворком Qt позволяет создавать крупные высокопроизводительные проекты со сложной графикой, такие как графическая оболочка KDE или математический пакет Wolfram Mathematica.

Хотя данный язык и является крайне производительным, он не лишён недостатков [15]:

1) использование заголовочных файлов, вместо системы модулей, что приводит к увеличению времени компиляции, многократным повторам имён функций и классов, отсутствию пакетных менеджеров и удобных систем автоматизации сборки;

2) сложный синтаксис и сложная специфика языка;

3) написание кроссплатформенного кода требует очень высокого уровня знаний и опыта.

Фреймворк Qt, представляющий огромные возможности по созданию высокопроизводительных кроссплатформенных приложений с сложной графикой, также не лишён недостатков [15, 15]:

1) на практике весь функционал данного фреймворка требуется очень редко, что делает его избыточным для большинства проектов;

2) высокий порог вхождения;

3) высокая стоимость лицензии;

4) сложная система лицензирования, весь инструментарий распространяется под тремя лицензиями одновременно, две из которых не позволяют распространять приложение без открытия исходных кодов.

2.1.2 JavaScript и Electron API

JavaScript – мультипарадигменный язык программирования со слабой динамической типизацией. Поддерживает объектно-ориентированный, императивный и функциональный стили программирования. Наибольшее распространение получил в браузерах как язык сценариев, но с современным развитием компиляторов, благодаря разработке компании Google – V8 JavaScript Engine, стало возможно использование JavaScript в составе платформы Node.js как языка разработки настольных приложений, таких как веб-серверы.

Electron API представляет собой связку из V8 JavaScript Engine и части браузера Chromium, что позволяет без особых усилий создавать кроссплатформенные настольные приложения без значительных усилий с использованием веб-технологий и практик. На базе Electron API создано большое количество известных приложений, таких как: популярнейшее приложение для публичного общения – Discord, новые версии Skype и широко распространённая в кругах разработчиков универсальная среда разработки — Visual Studio Code от компании Microsoft.

Приложения написанные с помощью Electron API не требуют больших затрат денежных средств, так как большинство различных плагинов, языков и прочих инструментов для платформы Node, как и сам Electron API распространяются под свободными лицензиями по типу BSD, MIT или Apache 2.0. К сожалению данная платформа не лишена недостатков [15, 15]:

1) высокое потребление оперативной памяти в связи с необходимостью держать в оперативной памяти целый браузер и виртуальную машину;

2) недостаточная производительность сложных вычислений;

3) из-за особенностей архитектуры V8 – невозможность писать полноценный многопоточный код.

2.1.3 Java и JavaFX

Java – сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует виртуальная Java-машина. Наибольшее распространение язык Java получил в корпоративном сегменте и мобильной разработке за счёт одной из самых высоких скоростей исполнения среди языков, работающих на виртуальных машинах, независимостью от программной и аппаратной платформы, простотой разработки и поддержки

больших проектов, а также очень широкого инструментария для разработки. В среде настольных приложений Java обычно используется в качестве вспомогательного языка для создания отказоустойчивых и не слишком критичных к производительности частей, но может использоваться и обособленно. К известным приложениям написанным с помощью Java относятся: большинство приложений для ОС Android, средство для визуального проектирования и моделирования бизнес-процессов Visual Paradigm, среды разработки компании JetBrains (Web Storm, Android Studio, CLion и т. д.), система плагинов и несколько вспомогательных механизмов свободного офисного пакета LibreOffice.

Из минусов языка можно выделить высокое, по сравнению с системными языками, потребление оперативной памяти [15].

JavaFX – графический инструментарий для разработки графических приложений на Java [15], может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений, входит в стандартную поставку Java вместе с компилятором и виртуальной машиной.

Данный набор технологий (Java и JavaFX), вместе с бесплатной средой разработки IntelliJ IDEA от компании JetBrains был выбран для выполнения выпускной квалификационной работы. Причина указанного выбора – наличие большого количества удобного инструментария и библиотек, а также опыта разработки с их использованием.

2.2 Разработка структуры программного средства определения клавиатурного почерка пользователей персонального компьютер

Структура – совокупность устойчивых связей объекта, обеспечивающих его целостность и тождественность самому себе при различных внешних и внутренних изменениях. В общем случае под структурой понимается совокупность составных частей некоторого объекта.

На рисунке 15 представлена структура программы для определения клавиатурного почерка пользователей.

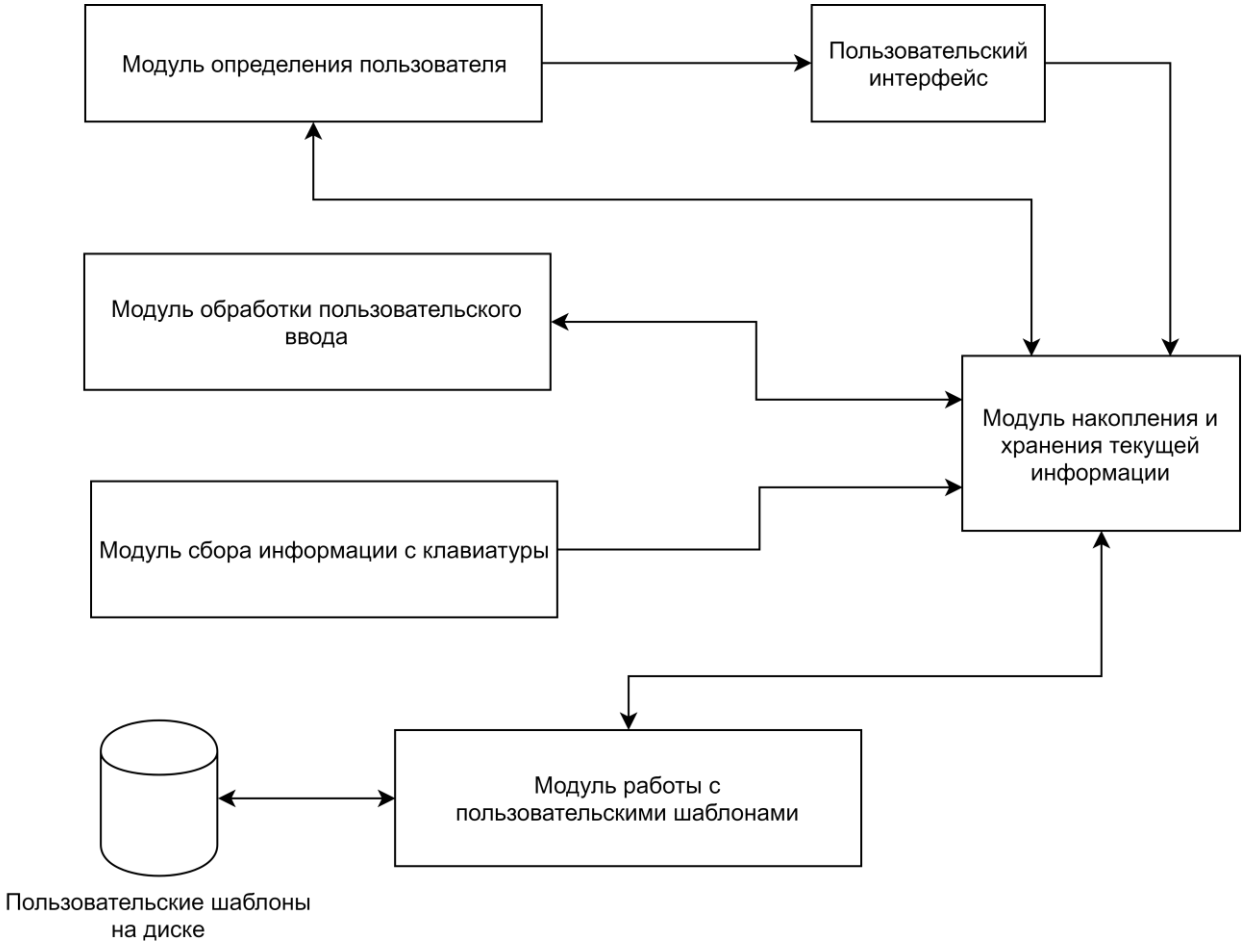


Рисунок 2 – Структура программы

Для удобства разработки и поддержки, программа была разделена на несколько частей. Все модули запускаются в параллельных потоках.

2.2.1 Модуль сбора информации с клавиатуры

Модуль предназначен для непосредственного сбора информации о клавиатурном вводе пользователя, вся полученная информация обрабатывается в соответствии с «белым» списком приложений (Notepad++, Microsoft Word, LibreOffice Writer) и накапливается в модуле накопления и хранения текущей

информации. Данный модуль базируется на библиотеках ядра операционной системы Windows:

– библиотека kernel32.dll для установки системного клавиатурного хука и получения идентификаторов активных процессов;

– библиотека user32.dll для получения заголовков окон активных приложений, получения текущей раскладки клавиатуры и параметров нажатых клавиш;

– библиотека psapi.dll для получения имен процессов и путей до их исполняемых файлов.

2.2.2 Модуль работы с пользовательскими шаблонами

Модуль работы с пользовательскими шаблонами содержит в себе два подмодуля:

– модуль маршализации и записи статистики на диск. Данный модуль занимается представлением информации о текущем пользователе в виде маршализованной информации и последующей её записи на диск;

– модуль чтения и демаршализации информации с диска. Данный модуль занимается чтением пользовательской информации с диска и последующим её сохранением в модуле накопления.

Для создания маршализованной информации использовалась стандартная библиотека Java – JAXB (Java Architecture for XML Binding) [15, 15]. Данная библиотека позволяет осуществлять простую конверсию Java-объектов в маршализованные данные и наоборот используя специальные аннотации, пример исходного кода использующего JAXB представлен на рисунке 15.

```java
@XmlRootElement
public class UserPattern {
    private String userId;
    private String userName;

    private Map<String, DigramInfo> userDigramStat = new HashMap<>();

    public UserPattern() {
    }

    public UserPattern(String userId, String userName, Map<String, DigramInfo> userPattern) {…
    }

    public String getUserName() {…
    }

    @XmlElement
    public void setUserName(String userName) {…
    }

    public String getUserId() {…
    }

    @XmlAttribute
    public void setUserId(String userId) {…
    }

```

Рисунок 3 – Исходный код класса, использующего механизм JAXB аннотаций

На рисунке 15 представлена функция реализующая маршализацию и запись пользовательского шаблона на диск.

```java
private void marshalAndWrite(UserPattern pattern) throws IOException, JAXBException {
    StringWriter writer = new StringWriter();


    JAXBContext jaxbContext = JAXBContext.newInstance(UserPattern.class);
    Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
    // output pretty printed
    jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
    jaxbMarshaller.marshal(pattern, writer);
    Path dir = Paths.get(
            CURRENT_PATH,
            "patterns");

    if (!Files.exists(dir))
        Files.createDirectory(dir);

    Path p = Paths.get(
            CURRENT_PATH,
            PATTERNS_DIR_NAME,
            pattern.getUserName().replaceAll("\\s+", "-") + "-" + pattern.getUserId() +
                    ".pattern");

    Files.write(
            p,
            writer.toString().getBytes(APPLICATION_CHARSET),
            StandardOpenOption.CREATE,
            StandardOpenOption.TRUNCATE_EXISTING
    );
}
```

Рисунок 4 – Функция маршализации и записи пользовательского шаблона на диск

Для осуществления файлового ввода/вывода была использована стандартная библиотека Java – Java NIO.2 предоставляющая возможности асинхронного ввода/вывода, что позволило осуществить одновременное чтение/запись пользовательских данных из разных потоков в один файл [15].

2.2.3 Модуль обработки пользовательского ввода

Модуль обработки пользовательского ввода содержит в себе два подмодуля:

– модуль усреднения пользовательской статистики, данный модуль занимается обработкой «сырого» клавиатурного ввода, разделяя его на отдельные диграммы и передавая их в модуль накопления и хранения текущей информации. На рисунке 15 представлена функция усреднения пользовательских данных

```
22    public void run() {
23        while (run) {
24            try {
25                Thread.sleep(EQUALIZER_SLEEP_TIME);
26            } catch (InterruptedException e) {
27                e.printStackTrace();
28            }
29
30            Storage.mainDigramStorage.forEach((key, val) -> {
31                List<DigramInfo> temp = new ArrayList<>();
32                while (!val.isEmpty())
33                    temp.add(val.poll());
34                if (!Storage.firstStepDigramStorage.containsKey(key))
35                    Storage.firstStepDigramStorage.put(key, new ConcurrentLinkedDeque<>());
36                if (!temp.isEmpty())
37                    Storage.firstStepDigramStorage.get(key).addFirst(new DigramInfo(temp));
38            });
39        }
40    }
```

Рисунок 5 – Функция усреднения пользовательских данных

– модуль обработки усредненных значений, данный модуль занимается финальной обработкой усреднённой информации и формирования шаблона текущего пользователя, на анализе которого происходит определение пользователя.

2.2.4 Модуль накопления и хранения текущей информации

Данный модуль представляет собой простой статический класс, содержащий несколько структур данных, необходимых для полноценной работы приложения, большинство из этих структур базируются на стандартной библиотеке Java – Concurrent API, для обеспечения возможности доступа из различных потоков приложения, листинг данного модуля представлен на рисунке 15.

```
13 public class Storage {
14     public static ConcurrentHashMap<String, ConcurrentLinkedQueue<DigramInfo>>
       mainDigramStorage = new ConcurrentHashMap<>();
15     public static ConcurrentHashMap<String, ConcurrentLinkedDeque<DigramInfo>>
       firstStepDigramStorage = new ConcurrentHashMap<>();
16
17     public static String userName = "";
18     public static String currentUserName = "";
19     public static String currentUserId = "-";
20     public static ConcurrentHashMap<String, DigramInfo> currentUserInfo = new
       ConcurrentHashMap<>();
21     public static ConcurrentHashMap<String, UserPattern> allUsers = new ConcurrentHashMap<>();
22     public static long isActive = 0;
23     public static long analyzeWaitTime = 0;
24     public static boolean userChanged = false;
25 }
```

Рисунок 6 – Листинг модуля накопления и хранения текущей информации

Все модули программы так или иначе используют данный модуль для хранения каких либо данных и взаимодействия между собой.

2.2.5 Модуль определения пользователя

Данный модуль занимается непосредственным определением пользователя на базе обработанной информации с модуля обработки пользовательских данных и работает по алгоритму, представленному на блок-схеме (приложение **Ошибка! Источник ссылки не найден.**).

Данный модуль также осуществляет взаимодействие с пользовательским интерфейсом для отображения в процесса определения пользователя.

2.3 Разработка формата хранения пользовательских шаблонов

Для хранения пользовательских шаблонов был выбран XML-формат. XML (eXtensible Markup Language) – расширяемый язык разметки, созданный консорциумом всемирной паутины (W3C). XML создавался как язык с простым синтаксисом, удобный как для создания и обработки программными средствами, так и удобный для чтения человеком. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчику необходимо её создать в соответствии с конкретной предметной областью, будучи ограниченным лишь синтаксическими правилами языка. Сочетание простого синтаксиса, удобства для человека, расширяемости, а также использование Юникод кодировок привело к широкому использованию как собственно XML, так и множества производных языков на его базе в самых разнообразных программных средствах [15]. Полученную структуру пользовательского шаблона можно представить в виде следующей древовидной модели (рисунок 15).

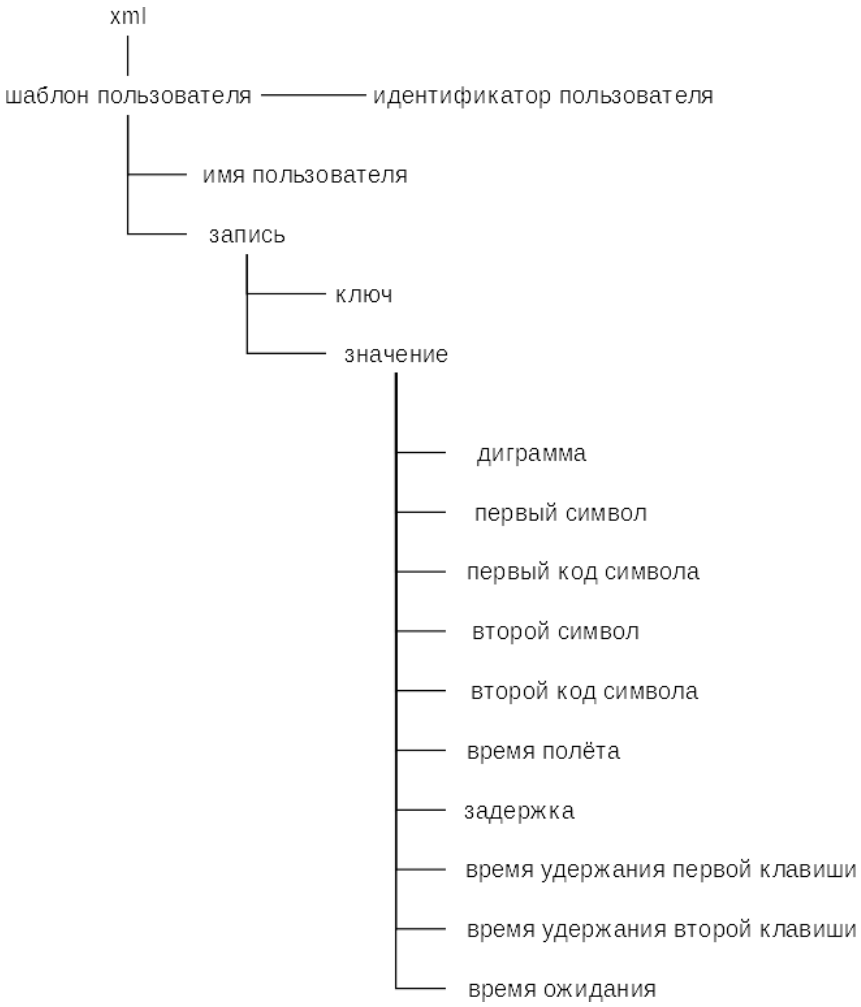


Рисунок 7 – Структура пользовательского шаблона

Пример шаблона пользователя представлен на рисунке 15.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<userPattern userId="ffff2c2e-cc8d-4d0b-95b0-67349da77cff">
    <userDigramStat>
        <entry>
            <key>аб</key>
            <value>
                <digram>аб</digram>
                <firstChar>а</firstChar>
                <firstDwellTime>80453219</firstDwellTime>
                <firstKeyCode>70</firstKeyCode>
                <flightTime>264794929</flightTime>
                <latency>184341710</latency>
                <secondChar>б</secondChar>
                <secondDwellTime>85971451</secondDwellTime>
                <secondKeyCode>188</secondKeyCode>
                <waitTime>350766380</waitTime>
            </value>
        </entry>
        </userDigramStat>
    <userName>Андрей</userName>
</userPattern>
```

Рисунок 8 – Пример шаблона пользователя

На рисунке 15 изображено конченое представление структуры пользовательского шаблона в соответствии со структурой шаблона (рисунок 15) с полным соответствием полей, например: поле диграмма соответствует полю digram в конечном представлении, аннотация идентификатор пользователя соответствует аннотации userId и так далее.

					Поле пользователя Обозначение = ВКР 09.03.02.03.000 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Поле			Практическая часть	Лит.	Лист 22 / Листов 36
Провер.		Поле					
Н. Контр.		Поле				Поле пользователя Группа = ИСТ-41	Лист
Утверд.							
Изм.	Лист	№ докум.	Подпись	Дата			

3 ПРАКТИЧЕСКАЯ ЧАСТЬ

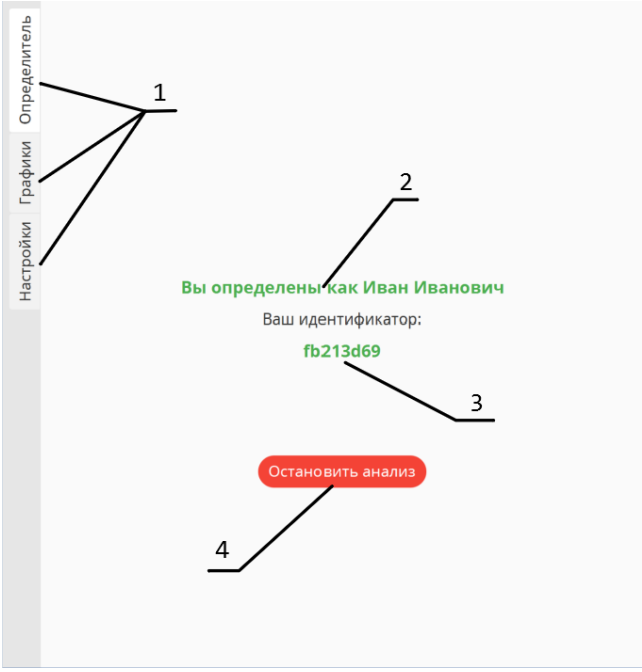
3.1 Основные элементы интерфейса программного средства для определения параметров пользователя персонального компьютера

В ходе реализации интерфейса приложения было принято разделить его на несколько вкладок:

1) вкладка *«Определитель»,* которая содержит информацию о текущем состоянии анализа клавиатурного почерка пользователя и возможностью его остановки (рисунок 15);

2) вкладка *«Графики»,* служащая для отображения разницы между отдельными категориями шаблона пользователей в графическом виде, (рисунок 15);

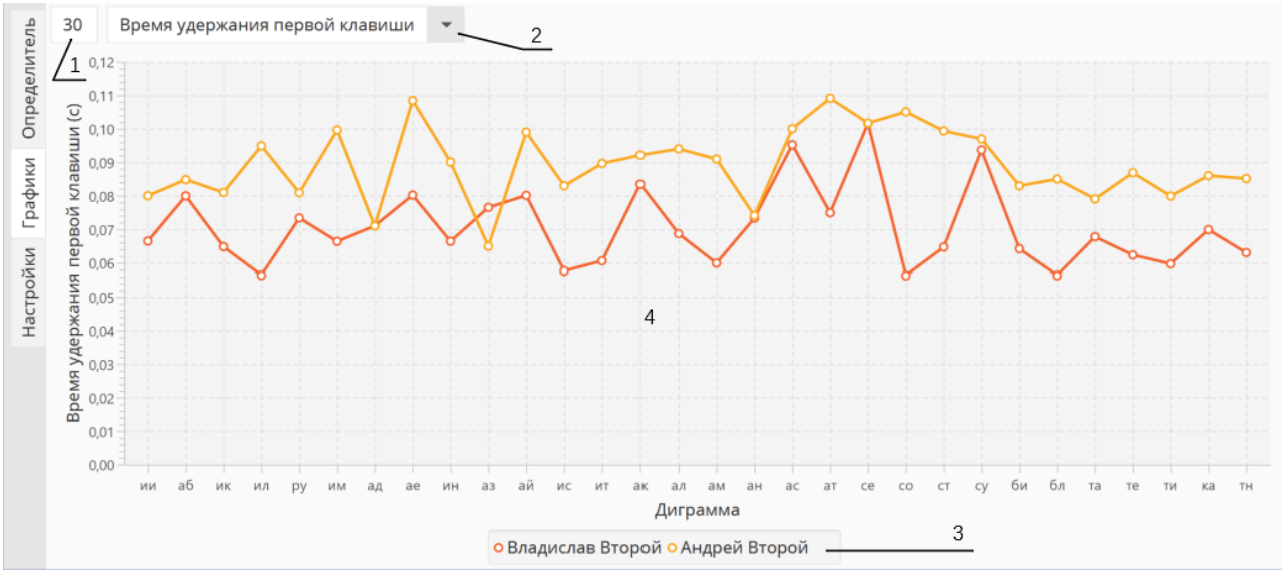
3) вкладка *«Настройки»*, позволяющая осуществлять базовую настройку программы и управлять пользователями (рисунок 15).



1 – переключатели вкладок; 2 – имя пользователя; 3 – идентификатор пользователя; 4 – кнопка остановки анализа

Рисунок 9 – Вкладка *«Определитель»*

С помощью переключателя вкладок (поз. 1 на рисунке 15) происходит смена вкладок. В области «имя пользователя» (поз. 2 на рисунке 15) отображается имя пользователя или текущее состояние анализа если пользователь ещё не определён. Кнопка остановки анализа (поз. 4 на рисунке 15) служит для остановки процесса анализа пользователя.

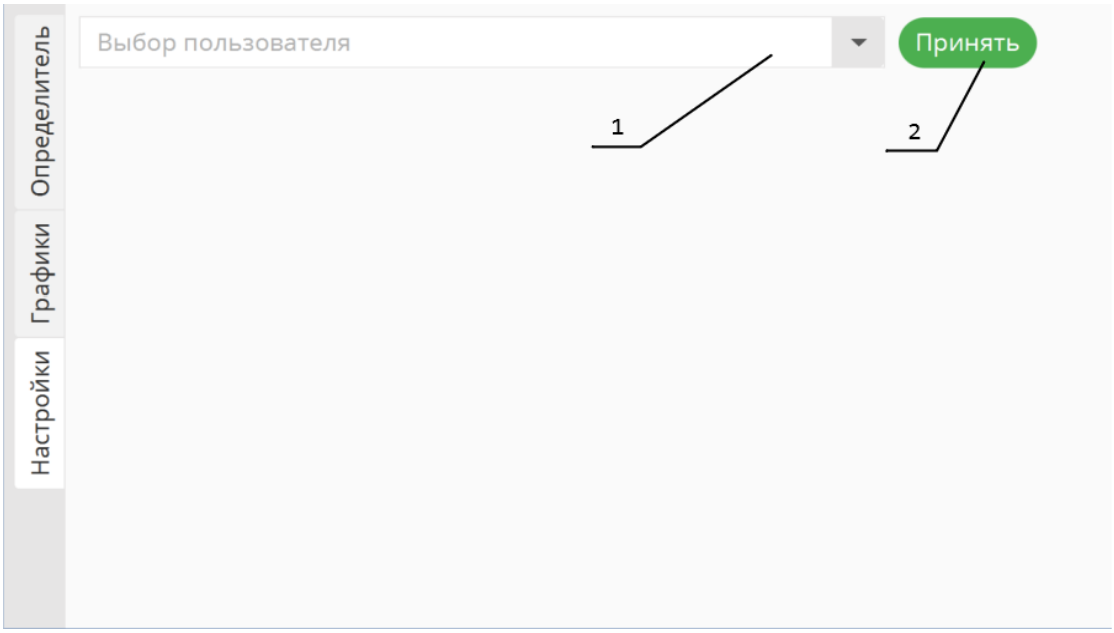


1 – числовое поле для выбора количества отображаемых значений;

2 – выпадающий список; 3 – имена пользователей; 4 – линейная диаграмма

Рисунок 10 – Вкладка *«Графики»*

С помощью текстового поля (поз. 1 на рисунке 15) производится выбор количества отображаемых точек на графике. С помощью выпадающего списка (поз. 2 на рисунке 15) производится выбор отображаемого параметра клавиатурного почерка. В области «имена пользователей» (поз. 3 на рисунке 15) отображается легенда – имена пользователей, чьи параметры клавиатурного почерка сейчас отображаются. Линейная диаграмма (поз. 4 на рисунке 15) служит для отображения графической информации о параметрах клавиатурного почерка.



1 – комбинированный список; 2 – кнопка подтверждения выбора

Рисунок 11 – Вкладка *«Настройки»*

С помощью комбинированного списка (поз. 1 на рисунке 15) производится ввод нового имени пользователя или выбор уже существующего. Кнопка подтверждения выбора (поз. 2 на рисунке 15) служит для принятия выбранного пользователя в качестве текущего.

Таким образом итоговая программа содержит одну экранную форму с несколькими вкладками, что вполне достаточно для удобства работы с программой и получения необходимых сведений о процессе анализа.

3.1.1 Запуск программы

Запуск программы, как и любого другого Windows-приложения, производится с её исполняемого файла (.exe). После включения программа сразу же начинает сбор статистики о пользователе находящимся за клавиатурой и производить параллельный его анализ на базе имеющихся сведений. Заложенный в программу алгоритм вычисляет «похожесть» текущего пользовательского ввода с эталонными пользовательскими шаблонами

хранящимися на диске, сравнивая полученный результат с принятым пороговым значением (60 процентов).

3.1.2 Определение пользователя по его клавиатурному почерку

Определение пользователя по его клавиатурному почерку происходит по алгоритму, блок-схема которого представлена в приложении **Ошибка! Источник ссылки не найден.**. Алгоритм собирает клавиатурный ввод только с приложений из «белого» списка. В данный список вошли наиболее популярные программы для набора текста: MS Word, Notepad, LibreOffice Writer. Во время анализа вычисляется схожесть пользователя со всеми шаблонами на диске, при этом на экран определения пользователя выводится наибольший процент совпадения текущего пользователя с каким-либо пользовательским шаблоном на диске, а также имя пользователя из этого шаблона (рисунок 15).

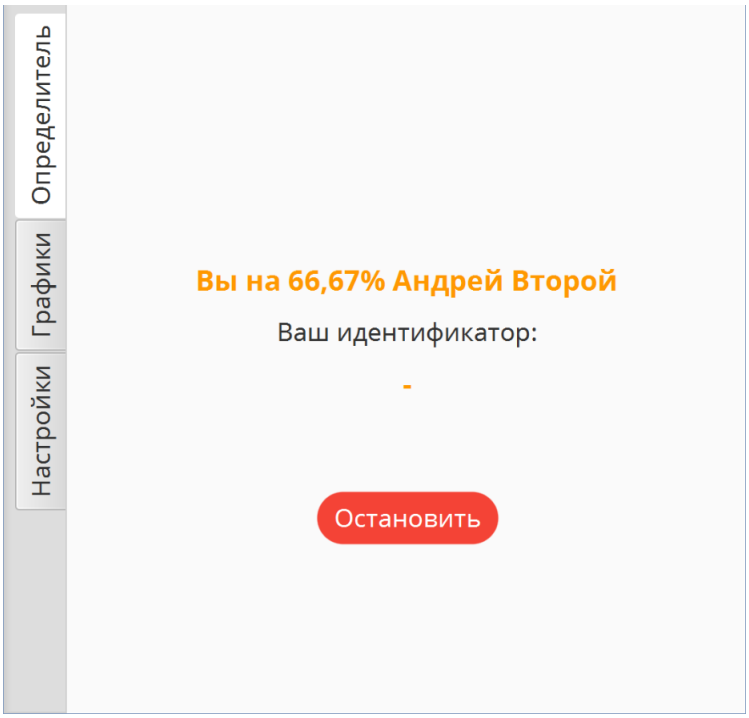


Рисунок 12 – Процесс определения пользователя

После того как система соберёт необходимое, больше 20, количество диграмм для анализа, выносится финальное решение на основе максимального

процента схожести, если процент схожести выше 60 %, считается, что пользователь определён и выносится положительное решение, рисунок 15.

Рисунок 13 – Положительное решение системы

Если же за это время процент совпадения с любым пользователем не достиг 60 %, система выносит отрицательное решение, рисунок 15.

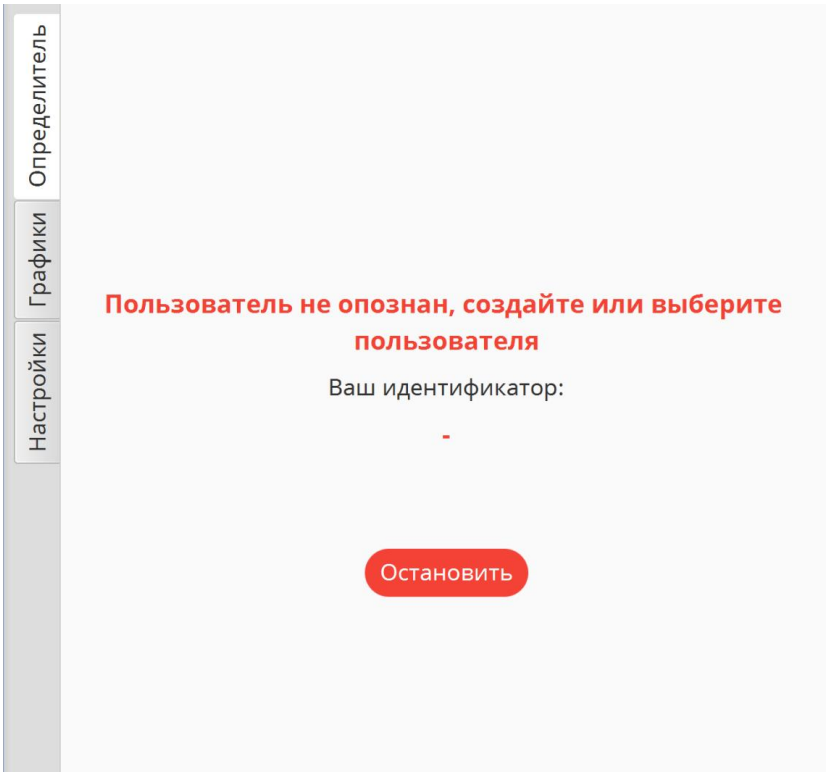


Рисунок 14 – Отрицательное решение системы

В этот момент пользователь может создать нового пользователя, или же выбрать имеющегося для дополнения уже существующего шаблона.

3.1.3 Вывод статистической информации о клавиатурном почерке пользователя

Для вывода статистической информации о клавиатурном почерке пользователя используется вкладка пользовательского интерфейса «Графики». Данная вкладка содержит линейную диаграмму для отображения основных параметров клавиатурного почерка пользователей, например время удержания первой клавиши, рисунок 15.

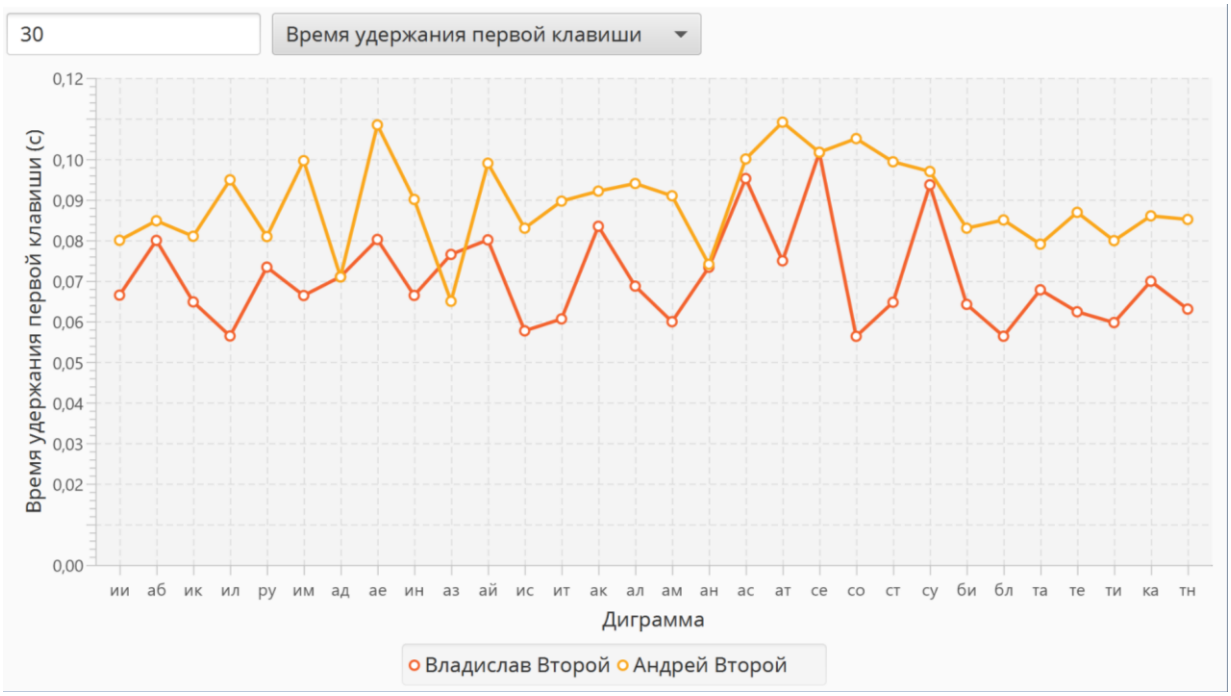


Рисунок 15 – Время удержания первой клавиши

На диаграмме отображается информация о параметрах клавиатурного почерка всех имеющихся на диске пользователей. Как видно из графика, данные различаются довольно значительно, что говорит о различиях в биометрических показателях пользователей.

На данном графике также можно отобразить остальные параметры клавиатурного почерка пользователя, такие как время удержания клавиш, интервалы между нажатиями, время «полёта» и время ожидания, выбирая их из выпадающего списка.

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы было разработано и реализовано программное средство для определения параметров клавиатурного почерка пользователя персонального компьютера.

Также были решены следующие частные задачи:

1) изучены особенности и способы определения компьютерного почерка пользователей, такие как статистический метод и методы на базе весовых коэффициентов и нейронных сетей;

2) проведён анализ и выбрано средство для реализации программы – Java и JavaFX вкупе со средой разработки IntelliJ IDEA;

3) разработан формат хранения пользовательских данных – маршализованная информация в виде XML-файлов и алгоритм работы программы;

4) реализована программа для определения параметров клавиатурного почерка пользователя, позволяющая определять основные параметры, создавать эталонные шаблоны клавиатурного почерка пользователя, а также производить его идентификацию с использованием этих шаблонов.

					Поле пользователя Обозначение = ВКР 09.03.02.03.000 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Поле			Список использованных источников	Лит.	Лист	Листов
Провер.		Поле					32	36
Н. Контр.		Поле				Поле пользователя Группа = ИСТ-41		Лист
Утверд.								
Изм.	Лист	№ докум.	Подпись	Дата				

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Identity Theft, Computers and Behavioral Biometrics // Ben Gurion University (BGU) of the Negev [Электронный ресурс]. – Режим доступа: http://www.ise.bgu.ac.il/faculty/liorr/idth.pdf – Загл. с экрана

2. Rick Joyce and Gopal Gupta – Identity Authentication Based on Keystroke Latencies // Carnegie Mellon's University [Электронный ресурс]. – Режим доступа: http://www.cs.cmu.edu/~maxion/courses/JoyceGupta90.pdf – Загл. с экрана

3. Fabian Monrose – Keystroke dynamics as a biometric for authentication // Columbia University in the city of New York [Электронный ресурс]. – Режим доступа: http://www.cs.columbia.edu/4180/hw/keystroke.pdf – Загл. с экрана

4. Яндиев И. Б. Исследование временных характеристик клавиатурного почерка для быстрой аутентификации личности // Молодой ученый. — 2017. — №14. — С. 154-158. — URL https://moluch.ru/archive/148/41543/ (дата обращения: 25.04.2018).

5. Richard L. Halterman – Fundamentals of Programming C++ // Online programming books [Электронный ресурс]. – Режим доступа: https://www.onlineprogrammingbooks.com/free-download-fundamentals-of-programming-c/ – Загл. с экрана

6. Qt Documentation // Qt.io [Электронный ресурс]. – Режим доступа: https://doc.qt.io – Загл. с экрана

7. Qt Licensing // Qt.io [Электронный ресурс]. – Режим доступа: https://www1.qt.io/licensing/ – Загл. с экрана

8. JavaScript tutorial // Tutorialspoint [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/javascript/index.htm – Загл. с экрана

9. Electron API documentation [Электронный ресурс]. – Режим доступа: https://electronjs.org/docs – Загл. с экрана

10. Java учебник [Электронный ресурс]. – Режим доступа: http://java-online.ru/java-basic.xhtml – Загл. с экрана

11. JavaFX – The Rich Client Platform // Oracle corporation [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html – Загл. с экрана

12. Extensible Markup Language (XML) 1.0 (Fifth Edition) // W3C Organization [Электронный ресурс]. – Режим доступа: https://www.w3.org/TR/REC-xml/ – Загл. с экрана

13. JSR 222: JavaTM Architecture for XML Binding (JAXB) 2.0 // Java community process [Электронный ресурс]. – Режим доступа: https://jcp.org/en/jsr/detail?id=222 – Загл. с экрана

14. Using JAXB Data Binding // Oracle documentation [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/cd/E12840_01/wls/docs103/webserv/data_types.html#wp223908 – Загл. с экрана

15. Package java.nio // Oracle documentation [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/javase/7/docs/api/java/nio/package-summary.html – Загл. с экран

					Поле пользователя Обозначение = ВКР 09.03.02.03.000 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Поле			Приложение	Лит.	Лист 35 / Листов 36
Провер.		Поле					
Н. Контр.		Поле				Поле пользователя Группа = ИСТ-41	Лист
Утверд.							
Изм.	Лист	№ докум.	Подпись	Дата			