Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА СИСТЕМНОГО АНАЛИЗА И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Направление: 10.03.01 — Информационная безопасность

Профиль: Безопасность компьютерных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ПРИЛОЖЕНИЯ «ЕДИНЫЙ АККАУНТ ДЛЯ ГРУППЫ СЕРВИСОВ ПОЛЬЗОВАТЕЛЯ»

Студент 4 курса
группы 09-641
"11" июня 2020 г. (Десятов А.Г.)

Научный руководитель
ст.преподаватель кафедры САИТ
"11" июня 2020 г. (Рубцова Р.Г.)

Заведующий кафедрой
д.т.н., профессор
"11" июня 2020 г. (Латыпов Р.Х.)

Казань – 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Роль мессенджеров, социальных сетей и почтовых сервисов в современном мире неизменно растет. Кроме того, растет и их количество. Пользователи создают свои аккаунты в различных сервисах для решения определённых вопросов, для обмена файлами, для простого общения. Кроме того, в период пандемии пользователи пользуются сервисами для дистанционного обучения, для удалённой работы и для удалённых совещаний. Иногда одному человеку требуется иметь сразу несколько аккаунтов в одном сервисе. Управление большим количеством своих аккаунтов в различных сервисах создает для пользователей определенные проблемы [1].

Во-первых, многие сервисы требуют от пользователя помнить логин и пароль для каждого своего аккаунта. Количество логинов и паролей у одного человека растет с каждой новой регистрацией в сервисе. Пользователь зачастую, чтобы запоминать меньше информации, использует один и тот же пароль в разных аккаунтах, тем самым нарушая правила безопасности и предоставляя злоумышленникам после взлома одного аккаунта получить доступ к другим аккаунтам.

Во-вторых, пользователи используют различные аккаунты в сервисах для различных целей. Например, определенные аккаунты они используют только для обмена информацией с коллегами, в некоторые сервисы они заходят только для общения с друзьями, а некоторые аккаунты используют только для получения рассылок.

В-третьих, пользователю приходится проверять свои аккаунты в различных сервисах поочередно на предмет непрочитанных сообщений. Отсюда появляется вероятность того, что пользователь забудет или не успеет проверить определенные сервисы, и, соответственно, не ознакомится с новыми полученными сообщениями.

Различные программы решают названные выше проблемы по отдельности. Например, определённые приложения хранят пароли.

Современный браузер не только запоминает пароли, но и позволяет быстро входить с помощью них в сервисы. Однако он не решает проблему поочередного входа в сервисы. Некоторые почтовые сервисы позволяют в один клик переключаться между своими аккаунтами, но не предоставляют пользователю возможность переключиться на аккаунт другого желаемого сервиса.

Разработанное приложение «Единый аккаунт для группы сервисов пользователя» решает сразу все перечисленные проблемы [2].

Целью выпускной квалификационной работы являлась разработка приложения, в котором будет реализован пользовательский интерфейс для работы с группой популярных сервисов и в котором будут использоваться современные методы защиты информации. Для достижения данной цели были поставлены следующие задачи:

— изучение технологий для реализации приложения,

— проектирование приложения и структуры хранения данных,

— реализация основного функционала приложения с комфортным пользовательским интерфейсом,

— исследование средств, позволяющих одновременно усложнить пароль и упростить его запоминание [3],

— проектирование и реализация системы защиты данных пользователя,

— тестирование и первоначальная эксплуатация приложения.

1. Технологии

Реализация приложения осуществлена при помощи среды разработки «Microsoft Visual Studio Community 2019» [4].

Удобный графический интерфейс реализован посредством набора библиотек интеллектуальной технологии «Windows Forms» для платформы разработки «.NET Framework» [5].

Форма Windows Forms представляет собой визуальное окно, на которой можно располагать элементы управления для ввода текста пользователем («TextBox» или «RichTextBox»), для вывода текста для него («Label»), для выбора нескольких вариантов («CheckBox»), для группировки этих вариантов («CheckedListBox»), или выбора только одного варианта («RadioButton»), для различных меню («ContextMenuStrip») и для его пунктов («ToolMenuStripItem»), для показа изображений («PictureBox»), для решения проблем, связанных с выравниванием элементов («FlowLayoutPanel», «TableLayoutPanel») [6].

В приложение встроен веб-браузер при помощи технологии Chromium Embedded Framework (CEF), библиотека реализована на языке программирования «C++». Для «.NET» доступна библиотека-обёртка «CefSharp». Во встроенном браузере имеется возможность внедрять скрипты «JavaScript» на страницу [7]. Для упрощения работы с «CefSharp». был реализован класс Browser (приложение 1. Класс Browser).

Из полученного HTML-кода страницы необходимая информация извлекается при помощи синтаксического анализатора, или парсера. Парсинг HTML-кода осуществляется при помощи библиотеки «HtmlAgilityPack» [8].

В данной работе часто упоминается понятие «Сервис», под которым подразумевается популярная социальная сеть, мессенджер или электронная почта.

Несмотря на отличительные особенности сервисов, у них имеются и сходства, которые позволяют составить общую структуру для всех сервисов. При добавлении нового сервиса в приложение сначала заносится значение

таких свойств, которые имеют все сервисы, а затем уже решаются вопросы, связанные непосредственно с отличительными особенностями.

Значения общих свойств хранятся в базе данных. Для управления базой данных используется СУБД MySQL. Модернизация базы данных осуществляется при помощи dbForge Studio for MySQL [9].

Для защиты данных пользователя используется пространство имён «System.Security.Cryptography», которое даёт возможность использовать шифрование, дешифрование и хеширование данных [10].

Для перевода языка Emoji на русский язык используется веб-служба «Яндекс.Переводчик».

2. Структура хранения данных

Данные пользователей хранятся локально на его устройстве в зашифрованном виде. У каждого пользователя есть логин, с помощью которого происходит обращение к его данным. Пароль пользователя не хранится даже в виде значения хеш-функции, однако он используется в ключе при шифровании и дешифровании данных.

После дешифрования данные десериализуются (первоначальная структура данных восстанавливается из битовой последовательности) в класс UserData (приложение 2. Класс UserData), который содержит контрольную сумму (она подтверждает успешное дешифрование), список групп, аудит всех действий пользователя, место хранения данных. Внутри класса реализованы соответствующие методы для проверки существования группы, для создания и удаления группы, для изменения названия и пароля группы, для добавления сервиса в группу и его удаления из нее, для обновления контрольной суммы, для подтверждения успешного дешифрования, для добавления строки в аудит, для сериализации (структура данных переводится в последовательность битов) данных с контрольной суммой и без неё, для подсчета текущей контрольной суммы, для записи обновленных данных в память.

Каждая группа представляет собой экземпляр класса, который содержит название, соль для пароля, список сервисов внутри группы. Внутри этого класса реализован метод вывода информации о том, в каких сервисах и под какими логинами авторизован пользователь в данной группе.

Пароль также представляет собой класс, в котором реализованы методы для получения нового пароля, для проверки текущего пароля, для хеширования пароля, для конвертации строкового типа в байты с помощью определенной кодировки, конкатенация (операция «склеивания») массивов байт для работы с солью.

Класс Service содержит общую информацию о сервисе: название, URL-адрес главной страницы, URL-адрес сообщений, URL-адрес аутентификации,

URL-адрес страницы, которая открывается после успешной аутентификации, вспомогательный JavaScript-код для открытия определенного чата, домен для очищения Cookie [11].

Класс ServiceInGroup содержит экземпляр класса Service и учётные данные для аутентификации в данном сервисе.

Все строки аудита записываются в StringBuilder в связи с тем, что этот класс позволяет изменять строку без потери производительности, в отличие от string, где любое изменение создает новую строку [12].

Класс Chat содержит всю необходимую информацию о чате/диалоге/беседе: название сервиса, порядковый номер чата в сервисе, который необходим для открытия во встроенном браузере, имя собеседника, последнее сообщение или его тема (в зависимости от сервиса), пометка о том, кто отправил последнее сообщение, пометка о наличии непрочитанных сообщений, количество непрочитанных сообщений, ссылка на чат для встроенного браузера, фото собеседника.

Обновление данных пользователя происходит после создания, изменения или удаления группы, после успешной аутентификации в сервисе, после выхода из сервиса и после любых других действий, которые необходимо запомнить. Сначала данные сериализуются, зашифровываются и затем записываются в память.

3. Система защиты данных

3.1. Шифрование данных

Для того чтобы защитить данные пользователя, их необходимо шифровать перед записью и обновлением.

Приложение осуществляет следующую последовательность действий:

— сериализация данных пользователя,

— нахождение контрольной суммы сериализованных данных с помощью хеш-функции SHA-512,

— сериализация экземпляра класса UserData, содержащего данные пользователя и контрольную сумму,

— формирование ключа с помощью пароля пользователя для алгоритма шифрования,

— шифрование с помощью современного криптостойкого алгоритма AES-128 в режиме CBC (режим цепочки блоков шифрования).

Для хеширования выбран алгоритм SHA-512, так как он является безопасным и на 64-битных архитектурах работает быстрее, чем SHA-256 [13].

Для шифрования выбран стандарт AES, который на данный момент считается криптостойким. На практике 128-битных ключей достаточно для обеспечения безопасности. При их использовании процессор нагружается слабее, чем при использовании 192-битных и 256-битных ключей, поэтому в данной работе выбран именно AES-128 [14]. Для шифрования файлов рекомендуется использовать режим CBC, потому что так увеличивается их безопасность.

Для шифрования и дешифрования реализован класс AesCryptographyService, который использует пространство имен «System.Security.Cryptography» [10]. Внутри класса можно заменить размер ключа и блока, режим шифрования и способ заполнения блока до нужной длины.

Ключ блочного шифрования доводится до необходимой длины с помощью заданных битов по умолчанию. Алгоритм AES-128 использует SP-сеть, что не позволяет выявить закономерности зашифрованной последовательности бит и входных данных.

3.2. Дешифрование данных

При входе в приложение пользователь вводит свой пароль. Данный пароль не хранится нигде даже в виде хеш-значения. С помощью данного пароля формируется ключ для дешифрования данных.

Если пароль введен верно, то после дешифрования получается сериализованный экземпляр класса UserData, затем осуществляется его десериализация, данные пользователя и контрольная сумма этих данных становятся доступными.

Затем осуществляется сериализация данных пользователя. Для сериализованных данных пользователя находится значение хеш-функции SHA-512, и результат сравнивается с контрольной суммой. Если они совпадают, то пользователь получает доступ к своим данным.

У такого способа шифрования данных есть преимущества:

— разработанное приложение не хранит пароли пользователей в виде хеш-значений, как это делают некоторые информационные системы, которые могут авторизовать злоумышленника, если он введёт любой из других прообразов данного хеша (например, когда данная хеш-функция перестанет отвечать условиям криптостойкости);

— отсутствие алгоритма нахождения ключей шифрования, так как ключ генерируется с помощью пароля, который придумает сам пользователь.

3.3. Безопасность группы

Для более тщательной защиты данных функционал приложения подразумевает возможность установить пароль на каждую группу пользователя.

Для соблюдения конфиденциальности информации о группе входить в нее можно только после ввода пароля.

В целях сохранения целостности переименование и удаление группы можно осуществить также только после подтверждения паролем.

Если бы пароли хранились в виде значений хеш-функции от пароля, то злоумышленник, получив все хеш-значения, знал бы, у каких групп стоят одинаковые пароли. Для того чтобы у одинаковых паролей отличались хеш-значения, необходимо использовать соль.

Соль приписывается к паролю справа. Затем полученная последовательность бит хешируются хеш-функцией SHA-512. Если бы соль приписывалась к паролю слева, то злоумышленник мог бы один раз вычислить некоторые операции хеш-функции от соли, а затем проводить перебор всех паролей с той скоростью, с которой перебирал бы их без соли [15].

3.4. Очищение Cookie

Внутри приложения встроен браузер, который собирает Cookie-файлы сервисов, их необходимо чистить в целях безопасности после определенных действий пользователя [11].

Приложение с помощью учётных записей само авторизует пользователя в сервисе, поэтому очищение Cookie-файлов, связанных с аутентификацией, никак не усложняет работу пользователя.

Очистка Cookie-файлов всех сервисов группы происходит при переходе пользователя в другую группу и при закрытии приложения.

При выходе пользователя из сервиса или при удалении сервиса из группы очищаются Cookie-файлы только данного сервиса.

Кроме того, с каждым запуском приложения во встроенном браузере отсутствуют все Cookie-файлы, связанные с аутентификацией. Поэтому при нештатной ситуации, например, когда пропадет питание устройства, и приложение будет закрыто принудительно, и перед закрытием не будет

осуществлена очистка Cookie-файлов, воспользоваться ими для доступа к аккаунтам пользователя будет нельзя, пока он не введет пароль для входа.

3.5. Использование Emoji в пароле

Приложение позволяет пользователю усложнить пароль, и при этом упростить его запоминание [3].

Пользователь приложения может использовать в своем пароле не только символы, но и Emoji [16]. Для достижения такой возможности были найдены и исследованы текстовые поля, которые поддерживают графический язык, и способы ввода картинок. Сочетание символов и смайликов из Emoji в пароле, во-первых, значительно усложняет полный перебор, а, во-вторых, позволяет пользователю при запоминании пароля использовать зрительную память.

Для перебора всех возможных паролей актуальна информация о том, сколько символов содержит алфавит. Язык Emoji содержит 1094 простых различных смайликов и 175 смайликов, которые могут быть применены с шестью различными оттенками цвета кожи. То есть с использованием Emoji в алфавит добавляется 2144 знака. В дополнение, сенсорная клавиатура Windows предлагает 75 смайликов, составленных из знаков препинания. Несмотря на большое количество картинок, их удобно искать. Определённый смайлик можно найти в поиске по тексту. Кроме того, все смайлики классифицированы по тематическим разделам.

Также были исследованы средства, использование которых помогают задействовать пользователю ассоциативную память при запоминании пароля. В форме регистрации и аутентификации встроен браузер, в котором открыт «Яндекс.Переводчик», который переводит сочетание смайликов в текстовый вид [17]. Из небольшого количества смайликов пользователь получает целое, логически связанное предложение, с помощью которого проще запомнить пароль.

После перевода Emoji на русский язык некоторые предложения получаются нелепыми, но при этом они легко ассоциативно запоминаются.

Использование таких текстовых предложений также делают пароль более надёжным.

Если пользователь предпочитает запоминать не сочетание картинок, а смайлики по отдельности, то он может навести мышь на любой из них, и в этом случае отобразится его текстовое значение.

3.6. Аудит

Важной составляющей безопасности данных пользователя является журнал его действий (или аудит).

После успешной аутентификации в приложении пользователь может посмотреть все свои действия. Если какие-либо действия производил не он, а злоумышленник, то с помощью аудита пользователь сможет это обнаружить.

Также предусмотрена ситуация, в которой злоумышленник перед использованием аккаунта жертвы переведёт дату и время назад, чтобы его действия затерялись в журнале среди действий жертвы. Все действия записываются последовательно. Кроме того, умышленное изменение времени сразу будет заметно в журнале.

Аудит содержит информацию о входе пользователя в свой аккаунт, о создании новой группы, о изменении ее названия и пароля, о её открытии и удалении, о добавлении определенного сервиса в группу, о сборе чатов в сервисах.

Кроме того, записываются сведения об ошибке переименования группы, об ошибке изменения её пароля, о неправильном введённом текущем пароле, об ошибке добавления сервиса, о неуспешной аутентификации пользователя в сервисе.

В журнал для каждого действия заносится дата и время, когда оно осуществилось [18].

4. Встроенный браузер и парсер

Из каждого сервиса, добавленного в группу, приложение извлекает диалоги (в зависимости от сервиса могут называться также сообщениями, беседами или чатами). Для этого приложению необходимо получить HTML-код страницы с сообщениями. А чтобы его получить, нужно, чтобы приложение, эмулируя действия пользователя, авторизовало его в сервисе с помощью учётных записей, которые он предоставит только один раз, и затем нужно, чтобы приложение дожидалось результатов выполнения скриптов.

Если использовать только get-запрос, то полученный в результате HTML-код не будет содержать необходимой информации, потому что он заполняется с помощью скриптов, а их выполнение не учитывается при получении результата от get-запроса [19]. Поэтому необходимо использовать средство, эмулирующее обычные действия пользователя и дожидающееся выполнения скриптов, заполняющих HTML-код, который приложение сможет использовать в дальнейшем.

С такой задачей справляется технология «Chromium Embedded Framework». В приложении используется её библиотека-обертка «CefSharp» [7]. С помощью нее реализован веб-браузер, встроенный в приложение.

Аутентификация в любом сервисе начинается с открытия URL-адреса входа во встроенном браузере. Затем приложение заносит учетные записи пользователя в текстовые поля и эмулирует нажатие кнопки «Войти» (в некоторых сервисах может называться по-другому).

В связи с тем, что каждый сервис имеет свою структуру, для каждого из них была реализована своя автоматизированная процедура аутентификации. Занесение данных в текстовые поля осуществляется по-разному в зависимости, от сервиса.

Например, для сервиса «Вконтакте» [20] был использован скрипт, написанный на языке «JavaScript», а для сервиса «Инстаграм» [21]

реализована эмуляция нажатия клавиш для ввода логина, для переключения на следующее текстовое поле и для ввода пароля.

Для «Инстаграма» такой способ выбран, потому что на странице входа текстовое поле заменяется на другое сразу же, как только пользователь начинает вводить учетные данные нажатием клавиш. После нажатия на кнопку входа данные берутся именно из полей, заменивших изначальные. Но замена полей не происходит, если вводить учетные записи с помощью скрипта, а не нажатием клавиш.

После успешной авторизации происходит переход на страницу с сообщениями (чатами, диалогами, беседами).

Затем со страниц с сообщениями из каждого сервиса берется HTML-код, который используется для парсинга. Из сервисов извлекается вся необходимая информация для структуры каждого чата: имя собеседника, текст последнего сообщения (или заголовок письма), фото собеседника, если оно есть, от кого отправлено последнее сообщение (имеется не во всех сервисах), пометка о наличии непрочитанных сообщений, количество непрочитанных сообщений (в некоторых сервисах имеется только пометка наличии непрочитанных сообщений, но нет их количества), ссылка и скрипт для перехода на диалог во встроенном веб-браузере.

У каждого сервиса своя HTML-разметка, поэтому при добавлении новых сервисов в приложение разрабатывается новый алгоритм парсинга. Кроме того, алгоритм парсинга иногда приходится менять после обновления добавленного сервиса.

Приложение объединяет и сортирует полученные чаты из разных сервисов. В начале списка располагаются чаты, содержащие непрочитанные сообщения. Затем приложение предоставляет пользователю все чаты в удобном виде.

Окна (или формы) для обмена сообщений у разных сервисов различаются и имеют свои особенности, например, определённые смайлики и

стикеры. Чтобы не жертвовать функционалом определенного сервиса, с помощью встроенного браузера можно попасть в данное окно (или форму).

Когда пользователь нажимает на один из чатов, во встроенном браузере открывается страница данного чата (окно переписки с собеседником), где уже можно обмениваться сообщениями. Но в некоторых сервисах, (например, в «Инстаграме») чтобы автоматически попасть на страницу с чатом, недостаточно открыть определённый URL. В них после открытия определённой страницы необходимо выполнить определённые скрипты «JavaScript», которые эмулируют клик по чату.

5. Пользовательский интерфейс

5.1. Форма регистрации и аутентификации

При запуске приложения открывается форма регистрации и аутентификации, она изображена на рисунке 5.1.1.

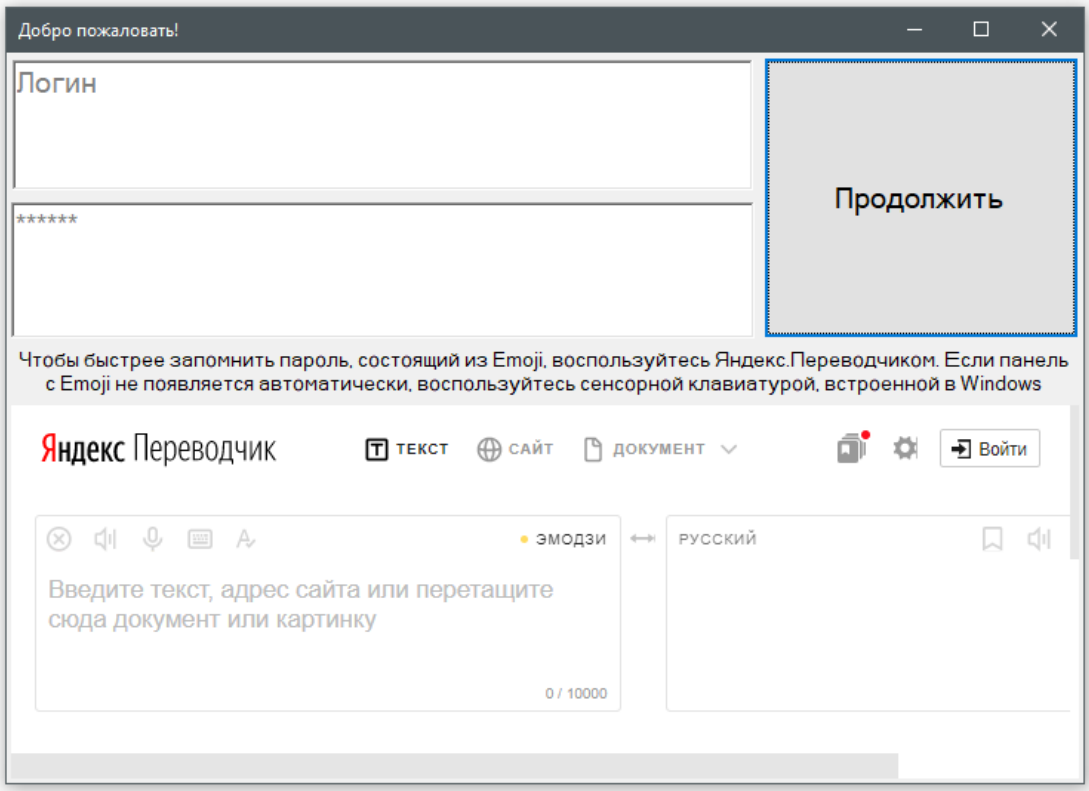

Рисунок 5.1.1 – Форма регистрации и аутентификации

Текстовые поля для логина и пароля имеют свойство подсказки «Hint» (серые надписи внутри поля, когда оно пустое). По умолчанию данного свойства нет в «.NET Framework». Данное свойство реализовано с помощью вспомогательного класса «RichTextBoxWithHint».

Когда пользователь начинает вводить пароль, автоматически выходит вспомогательное окно с Emoji [22], оно изображено на рисунке 5.1.2. Выход окна реализован с помощью эмуляции нажатия сочетания клавиш «Windows» + «Точка с запятой» [23]. Перед этим раскладка клавиатуры переключается на английский язык с помощью свойства «InputLanguage» в пространстве имён «System.Windows.Forms».

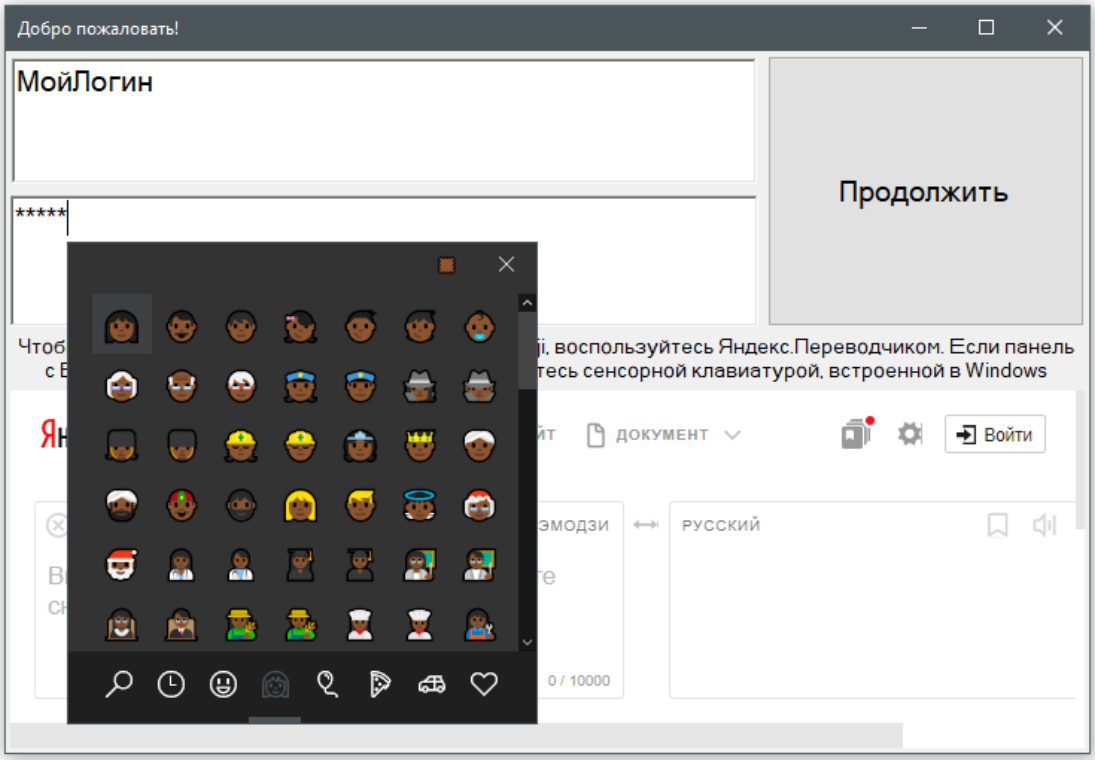

Рисунок 5.1.2 – Окно с Emoji

При вводе пароля текст скрывается с помощью символа «*». В «TextBox» для этого есть свойство «CharPassword». Однако «.NET Framework» не позволяет по умолчанию использовать одновременно окно Emoji и «CharPassword».

Поэтому был реализован класс «RichTextBoxWithCharPassword», являющийся наследником «RichTextBox». Реализованный класс использует и окно с Emoji, и «CharPassword».

Внизу формы находится встроенный браузер, в котором открыта веб-служба «Яндекс.Переводчик», переводящая сочетание Emoji на логически связанные предложения русского языка.

С её помощью можно упростить запоминание пароля, состоящего из смайликов, используя ассоциативную (текстовые предложения) и зрительную (изображения смайликов) память. Кроме того, пароль также будет надёжным, если в нём использовать полученные предложения. Пример использования веб-службы изображён на рисунке 5.1.3.

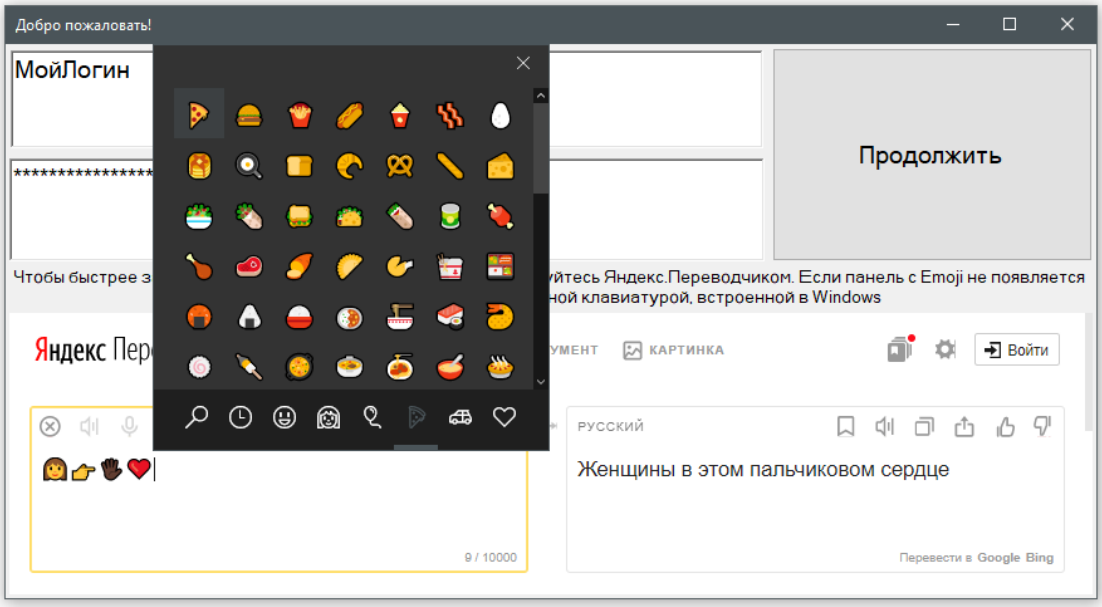

Рисунок 5.1.3 – Использование веб-службы «Яндекс.Переводчик»

Когда пользователь нажимает на кнопку «Продолжить», приложение проверяет, имеется ли пользователь с таким логином. Если такой пользователь уже есть, то проверяется введённый пароль способом, который описан в третьей главе. Если ещё не существует пользователя с данным логином, то либо пользователь ошибся, либо пользователь хочет зарегистрироваться. Поэтому приложение с помощью диалогового окна, изображённого на рисунке 5.1.4, уточняет, действительно ли пользователь хочет зарегистрироваться.

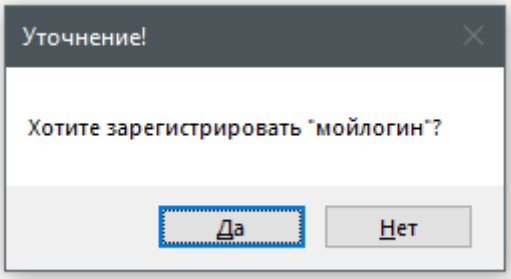

Рисунок 5.1.4 – Уточнение о регистрации в диалоговом окне

Если пароль или логин будут введены некорректно, то приложение не разрешит регистрацию. Если при аутентификации пользователь введёт неправильный пароль, приложение уведомит его об этом.

Если аутентификация или регистрация проходит успешно, то открывается основная рабочая форма.

5.2. Основная рабочая форма

Основная рабочая форма при первом запуске является незаполненной, она изображена на рисунке 5.2.1.

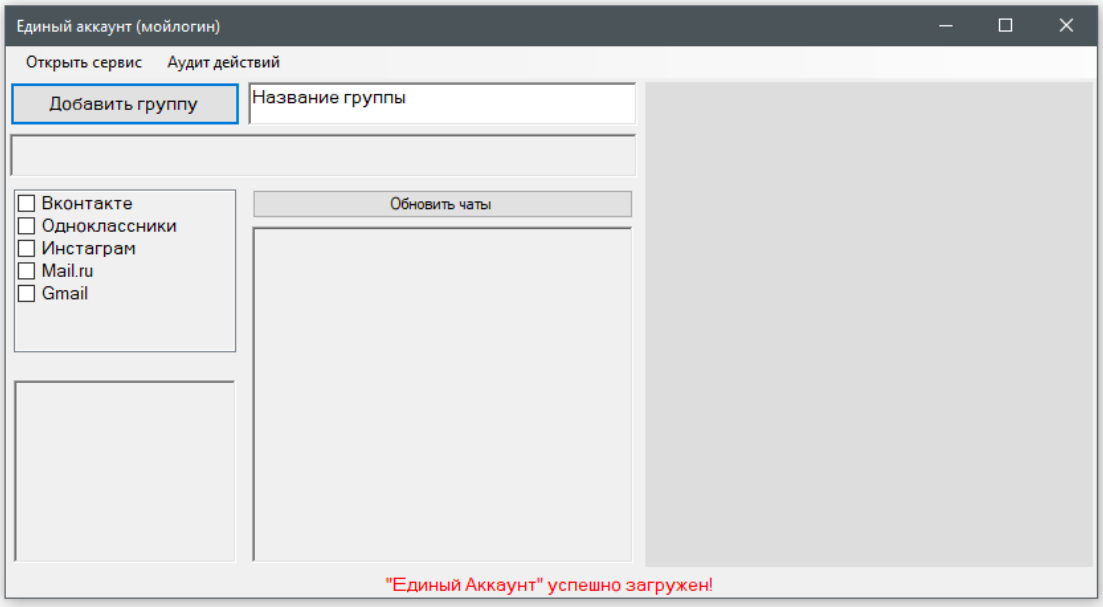

Рисунок 5.2.1 – Основная рабочая форма при первом запуске

Для начинающего пользователя приложения предусмотрено «Label» с подсказками, которое располагается внизу формы. Например, если пользователь перед тем, как создать первую группу, нажмёт на один из доступных сервисов, то в этом «Label» появится сообщение, подсказывающее пользователю верные действия. Данное сообщение в «Label» с подсказками изображено на рисунке 5.2.2.

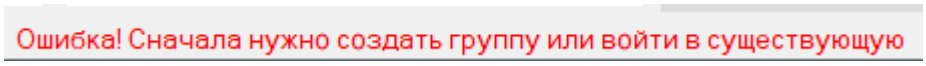

Рисунок 5.2.2 – Поле подсказок

Иногда в данном поле появляются сообщения с просьбой пользователя подождать, если какая-либо операция длится долго, например, если скорость интернета низкая.

Чтобы создать новую группу, пользователю необходимо придумать её название, записать его в «TextBox» вверху формы и нажать на кнопку

«Добавить группу» или на клавишу «Enter». Под данной кнопкой и данным «TextBox» находится «FlowLayoutPanel», внутри которого отображаются группы пользователя, внутри пользовательского интерфейса каждая из них представляет собой «RadioButton». Если групп будет много, и они не будут помещаться в доступной области, то появится полоса прокрутки, с помощью которой можно будет найти любую группу и обратиться к ней, как изображено на рисунке 5.2.3.

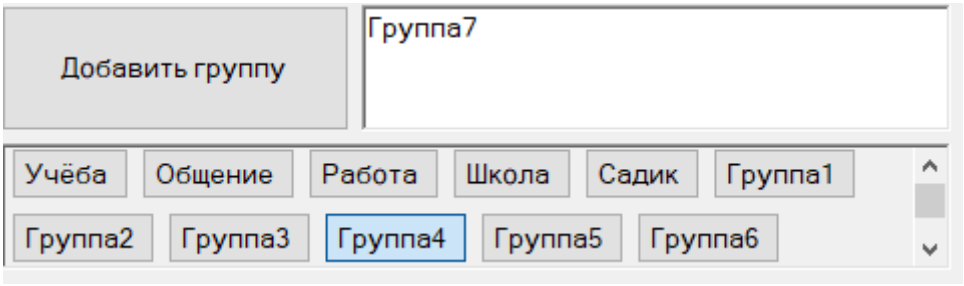

Рисунок 5.2.3 – Группы пользователя

Переключение между группами происходит в один клик, если на ту группу, на которую переключается пользователь, не установлен пароль. Если он установлен, то она откроется сразу после ввода верного пароля в диалоговом окне.

Если пользователь нажмёт правой кнопкой мыши на одну из групп, то появится контекстное меню «ContextMenuStrip», которое изображено на рисунке 5.2.4.

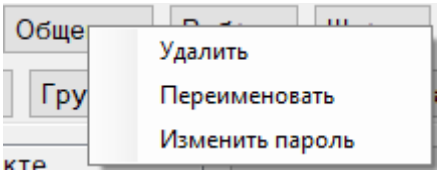

Рисунок 5.2.4 – Контекстное меню группы

В данном контекстном меню можно выбрать одно из действий с группой: «Удалить», «Переименовать» и «Изменить пароль».

Если у выбранной группы ещё нет пароля, то удалить её можно одним кликом, переименовать группу или поставить пароль можно с помощью диалогового окна, которое появится, если нажать на «Переименовать» или на «Изменить пароль», соответственно.

Если на группу уже стоит пароль, то удаление, переименование и изменение пароля осуществляется после ввода текущего пароля с помощью диалогового окна.

Открыв одну из групп, пользователь может приступать к добавлению своих аккаунтов из различных сервисов в группу. Для этого ему нужно выбрать элементы «CheckBox» с интересующими его сервисами в «CheckedListBox», изображённого на рисунке 5.2.5.

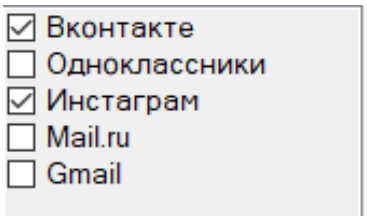

Рисунок 5.2.5 – «CheckListBox» с сервисами

После нажатия на каждый элемент появляется диалоговое окно, которое запрашивает логин для данного сервиса, а затем выходит другое диалоговое окно, которое запрашивает пароль для данного логина (в некоторых сервисах учётные записи могут отличаться, но, в любом случае, приложение получает информацию от пользователя таким образом).

В основной рабочей форме под списком сервисов также имеется текстовое поле для чтения, в котором записаны логины, с которыми пользователь авторизован в сервисах активной группы. Оно изображено на рисунке 5.2.6. Данное текстовое поле поможет не запутаться, если, например, у пользователя в одном сервисе несколько аккаунтов.

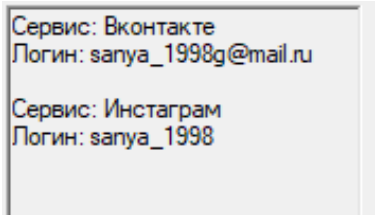

Рисунок 5.2.6 – Текстовое поле с активными аккаунтами

В рабочей форме имеется «FlowLayoutPanel» для объединения чатов из всех сервисов активной группы. Данная панель изображена на рисунке 5.2.7.

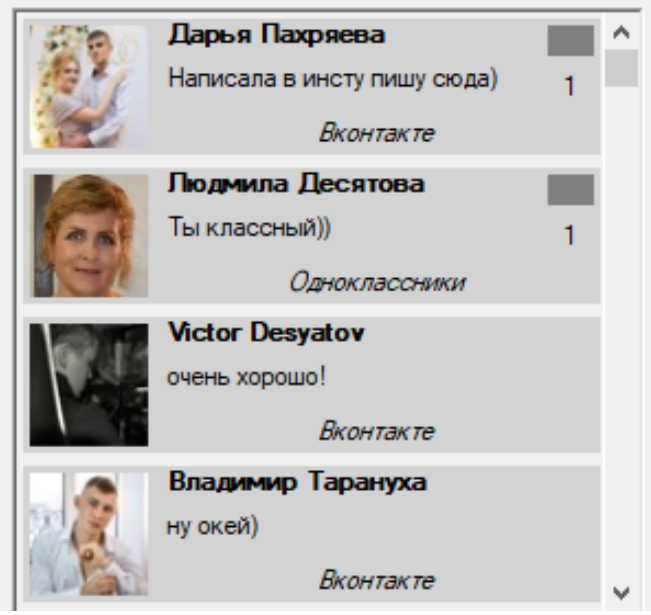

Рисунок 5.2.7 – «FlowLayoutPanel», объединяющая чаты

Каждый чат представляет собой «TableLayoutPanel». В первом столбике располагается фотография собеседника, растянутая на все строки. Во втором столбике в первой строке находится имя собеседника или название беседы, чуть ниже последнее сообщение в чате или название сообщения (в зависимости от сервиса), ещё ниже название сервиса, так как чаты собраны из разных сервисов. В третьем столбце находится пометка о наличии непрочитанных сообщений и их количество.

Список чатов формируется таким образом, что сверху располагаются те из них, которые содержат непрочитанные сообщения.

Над чатами располагается кнопка «Обновить чаты», нажатие на которую начинает поиск новых сообщений в сервисах активной группы. Кнопка изображена на рисунке 5.2.8.

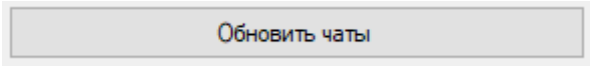

Рисунок 5.2.8 – Кнопка «Обновить чаты»

Если пользователь нажимает на один из чатов, то во встроенном браузере, находящемся в правой части рабочей формы, открывается страница диалога с выбранным собеседником на сайте сервиса, что позволяет

пользоваться всеми возможностями сервиса в диалоге. Открытый диалог одного из сервиса представлен на рисунке 5.2.9.

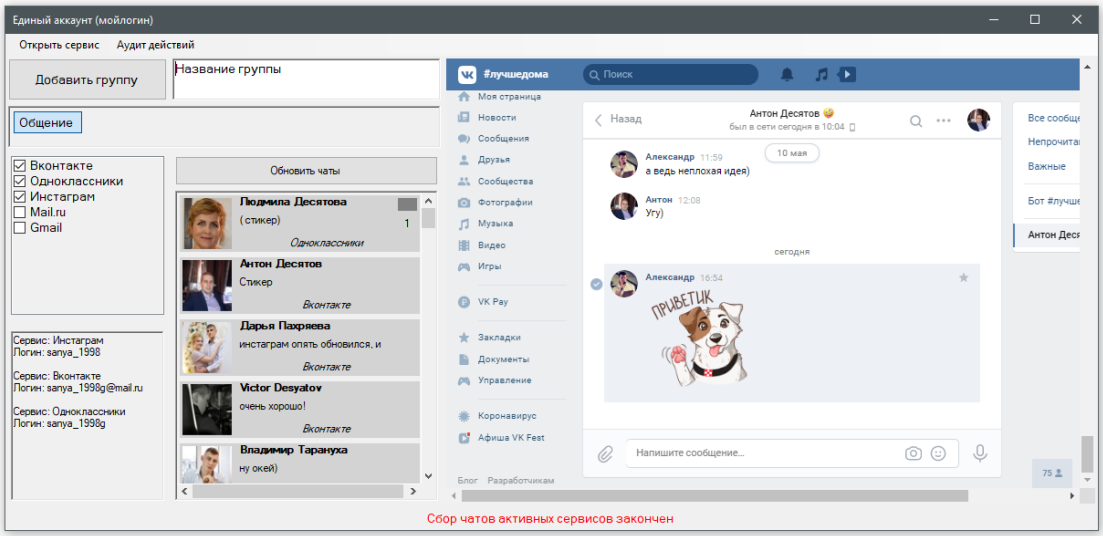

Рисунок 5.2.9 – Открытый диалог во встроенном браузере

В верхней части основной рабочей формы имеются элементы «ToolMenuStripItem», изображённые на рисунке 5.2.10, позволяющие открыть любой доступный сервис или форму аудита действий.

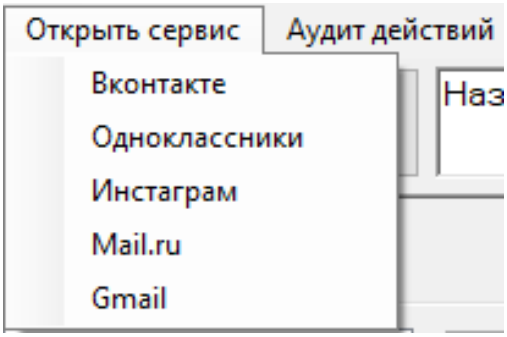

Рисунок 5.2.10 – Элементы «ToolMenuStripItem» в верхней части формы

Сервисы открываются во встроенном браузере, что позволяет использовать преимущества каждого сервиса. Например, пользователь может не только находиться в списке чатов, но и зайти на свою страницу (в свой личный кабинет) выбранного им сервиса, перейти к списку сообществ (групп), посмотреть новости друзей и так далее.

Количество различных сервисов будет увеличиваться с развитием приложения.

Код основной рабочей формы представлен в приложении 3.

5.3. Диалоговое окно

Для общения с пользователем предусмотрено диалоговое окно. Его шаблон изображён на рисунке 5.3.1.

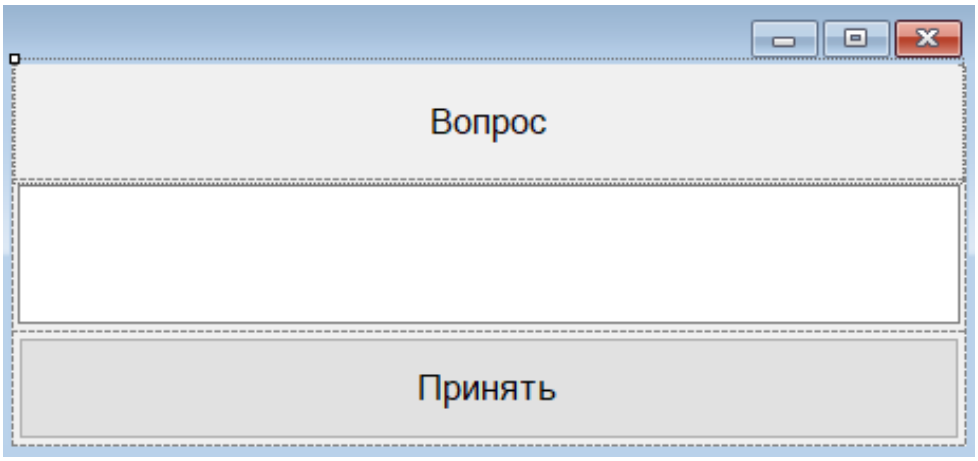

Рисунок 5.3.1 – Шаблон диалогового окна

С помощью диалогового окна у пользователя принимается новое название группы, ее текущий и новый пароль, логины для сервисов, пароли для логина, введённого ранее.

Когда подразумевается, что пользователь будет вводить в текстовое поле пароль, то весь текст заменяется символами «*».

Для удобной и более быстрой работы пользователю после ввода текста можно нажать клавишу «Enter», и его ответ будет принят.

5.4. Форма аудита действий

Пользователь может посмотреть журнал своих действий. Когда он нажмёт на кнопку "Аудит действий", которая соответствует элементу «ToolMenuStripItem», в верху основной рабочей формы, то откроется форма аудита действий. Пример такой формы изображён на рисунке 5.4.1.

В журнале рядом с каждым действием записывается дата и точное время, когда оно совершалось.

Кроме действий пользователя в журнал заносится информация о выполнении операций приложением. Также в него записываются ошибки, которые могут быть связаны с нештатными ситуациями, например, если пропадёт выход в интернет.

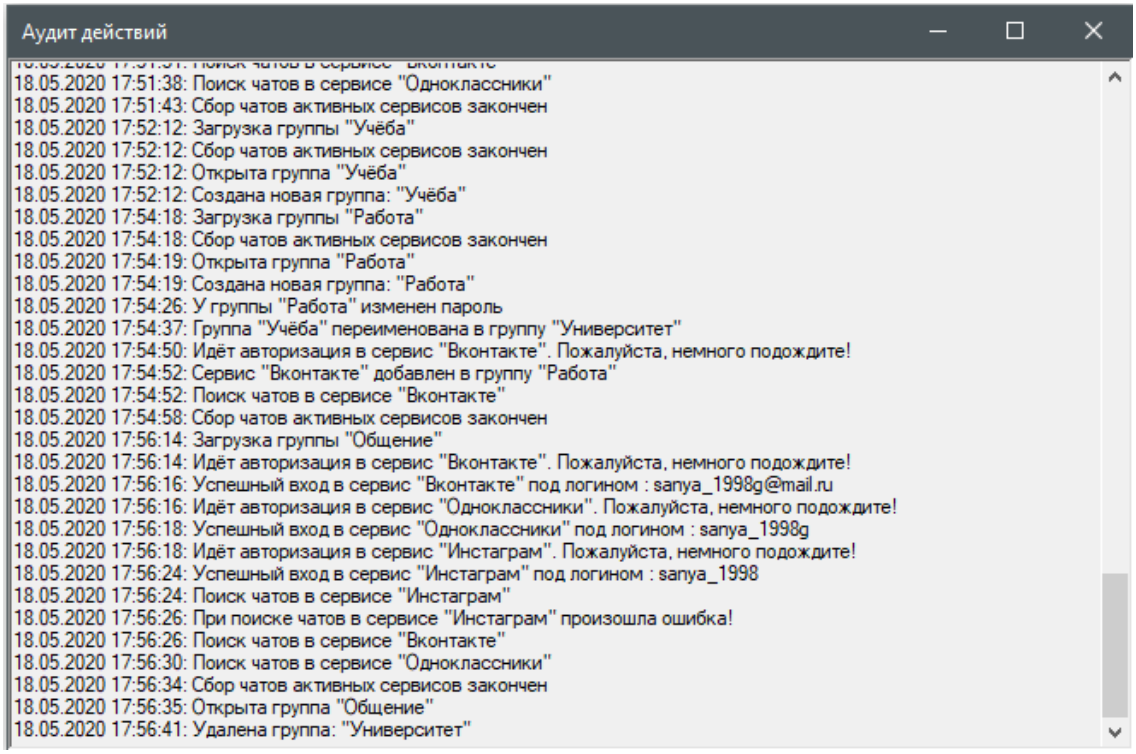

Рисунок 5.4.1 – Пример формы аудита

Во всех описанных выше формах используется «TableLayoutPanel», для того чтобы при любых изменениях размеров формы её функционал не уменьшался, а её внешний вид не ухудшался.

6. Тестирование и эксплуатация

В первую очередь, у разработанного приложения необходимо было протестировать модули отдельно, а затем их связи.

Каждый чат состоит из совокупности элементов. Так как у собеседника может отсутствовать фотография, при тестировании проверялось, чтобы при клике по тому месту, где должна быть фотография, во встроенном браузере открывался диалог. Кроме того, проверялось, чтобы чат отображался корректно всегда, чтобы внутри чата сокращались слишком длинные имя собеседника и последнее сообщение.

При тестировании были выявлены ошибки, которые затем исправлялись в программном обеспечении. Например, до исправления можно было создавать группу с пустым названием, и в интерфейсе пользователя элемент «RadioButton» для такой группы был маленького размера. Теперь на названия групп появились ограничения.

Тестировалось еще и то, чтобы при неудачной аутентификации пользователя в сервисе, данный аккаунт не добавлялся в группу.

Анализировалось также поведение приложения при нештатных ситуациях: уменьшение скорости интернета, его полное отключение, отключение питания устройства. Во всех перечисленных случаях приоритетным является безопасность учетных данных пользователя, и она не нарушается. Сохраняется и целостность, и конфиденциальность данных.

Для поиска недочётов тесты приложения проводили также пользователи, не знающие исходный код [24].

После окончания разработки приложения было произведено максимально возможное понижение целевой рабочей среды без потери функционала с «.NET Framework 4.7.2» до «.NET Framework 4.5.2» для увеличения целевой аудитории.

Разработанное приложение было установлено на устройства с различными версиями «.NET Framework» для проверки работоспособности, а затем для активного пользования.

Приложение также было установлено на устройства с разными разрешениями экранов. Для сохранения функционала на устройствах с маленьким разрешением было произведено редактирование пользовательского интерфейса [3].

ЗАКЛЮЧЕНИЕ

Разработанное приложение «Единый аккаунт для группы сервисов пользователя» запоминает учётные данные разных аккаунтов пользователя из разных сервисов и безопасно их хранит.

Все данные пользователя шифруются современным криптостойким алгоритмом AES-128. Ключ шифра формируются с использованием пароля пользователя для данного приложения. А сам пароль не хранится нигде даже в виде хеш-значения.

Пользователь приложения может использовать в своем пароле не только символы, но и язык Emoji [16]. Сочетание символов и смайликов из Emoji, во-первых, значительно усложняет полный перебор паролей (в алфавит добавляется 2144 знака), а, во-вторых, позволяет пользователю при запоминании пароля использовать зрительную память.

Веб-служба «Яндекс.Переводчик», открытая во встроенном веб-браузере и переводящая сочетание смайликов Emoji на логически связанные предложения русского языка, упрощает запоминание пароля, позволяя пользователю использовать ассоциативную память.

С помощью приложения пользователь может группировать свои аккаунты, которые он использует для определённых целей. Например, он может распределить все свои аккаунты на следующие группы: «Работа», «Учёба», «Общение» [2]. Переход между группами осуществляется в один клик. Кроме того, по желанию пользователя на каждую группу можно установить свой пароль.

Когда пользователь переходит в одну из таких групп, ему предоставляется список чатов всех сервисов, которые он ранее добавил в данную группу. А сверху этого списка располагаются чаты, содержащие непрочитанные сообщения.

Внутри приложения встроен веб-браузер, используя его, пользователь, к примеру, может перейти в свой личный кабинет любого из доступных сервисов, посмотреть на список подписчиков, ознакомиться с новыми

фотографиями друзей и их новыми видео, также он может зайти в свои сообщества или послушать музыку.

Таким образом, было разработано и введено в первоначальную эксплуатацию приложение «Единый аккаунт для группы сервисов пользователя», в котором реализован пользовательский интерфейс для комфортной работы с группой популярных сервисов и в котором используются современные методы защиты информации.

СПИСОК ЛИТЕРАТУРЫ

1. Десятов А.Г. Отчет по практике по получению первичных профессиональных умений и навыков, 2020. – 8 с.
2. Десятов А.Г. Отчет по проектно-технологической практике, 2020. – 8 с.
3. Десятов А.Г. Отчет по эксплуатационной практике, 2020. – 8 с.
4. Visual Studio. Лучшие в своем классе средства для разработчиков [Электронный ресурс]. – 2020. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения 19.03.2020).
5. Документация по .NET Framework [Электронный ресурс]. – 2020. – URL: https://docs.microsoft.com/ru-ru/dotnet/framework/ (дата обращения 19.09.2019).
6. Введение в Windows Forms – 2015. – URL: https://metanit.com/sharp/windowsforms/1.1.php (дата обращения 29.09.2019).
7. Первые шаги с Chromium Embedded Framework и .NET [Электронный ресурс]. – 2012. – URL: https://habr.com/ru/post/152637/ (дата обращения 20.03.2020).
8. Html Agility Pack (HAP) [Электронный ресурс]. – 2020. – URL: https://html-agility-pack.net/ (дата обращения 22.03.2020).
9. Самый умный инструмент для работы с MySQL dbForge Studio for MySQL [Электронный ресурс]. – 2020. – URL: https://www.devart.com/ru/dbforge/mysql/studio/ (дата обращения 24.03.2020).
10. System.Security.Cryptography Пространство имен [Электронный ресурс]. – 2020. – URL: https://docs.microsoft.com/ru-ru/dotnet/api/system.security.cryptography?view=dotnet-plat-ext-3.1 (дата обращения 24.04.2020).
11. Что такое файлы cookies и зачем они нужны [Электронный ресурс]. – 2019. – URL: https://ssl.com.ua/blog/what-are-cookies/ (дата обращения 10.04.2020).

12. StringBuilder прошлое и настоящее [Электронный ресурс]. – 2012. – URL: https://habr.com/ru/post/172689/ (дата обращения 20.10.2019).

13. SHA-2 [Электронный ресурс]. – 2020. – URL: https://ru.wikipedia.org/wiki/SHA-2 (дата обращения 20.04.2020).

14. What are the practical differences between 256-bit, 192-bit, and 128-bit AES encryption? [Электронный ресурс]. – 2020. – URL: https://crypto.stackexchange.com/questions/20/what-are-the-practical-differences-between-256-bit-192-bit-and-128-bit-aes-enc (дата обращения 22.04.2020).

15. Как надо хешировать пароли и как не надо [Электронный ресурс]. – 2014. – URL: https://habr.com/ru/post/210760/ (дата обращения 10.03.2020).

16. Почему смайлы Emoji в качестве пароля для смартфона — это сильно?! [Электронный ресурс]. – 2017. – URL: https://wylsa.com/emoji-like-a-password/ (дата обращения 26.03.2020).

17. Яндекс.Переводчик [Электронный ресурс]. – 2020. – URL: https://translate.yandex.ru/?lang=emj-ru (дата обращения 07.05.2020).

18. Что такое аудит и зачем он нужен? [Электронный ресурс]. – 2020. – URL: https://mashaudit.ru/info/chto_takoe_audit_i_zachem_on_nuz/ (дата обращения 07.02.2020).

19. HTTP-запрос методом GET. [Электронный ресурс]. – 2020. – URL: https://webkyrs.info/post/http-zapros-metodom-get (дата обращения 17.03.2020).

20. Вконтакте [Электронный ресурс]. – 2006. – URL: https://vk.com/ (дата обращения 20.04.2020).

21. Инстаграм [Электронный ресурс]. – 2010. – URL: https://www.instagram.com/ (дата обращения 21.04.2020).

22. Как получить доступ и использовать Emoji в операционной системе Windows [Электронный ресурс]. – 2018. – URL: https://emojifaces.org/ru/windows.html (дата обращения 06.05.2020).

23. Emoji (эмодзи) в Windows комбинацией клавиш [Электронный ресурс]. – 2019. – URL: https://lifeservice.me/jemodzi-v-windows-10-kombinaciej-klavish/ (дата обращения 06.05.2020).

24. Лихицкий, А.С. Исследование стратегий тестирования программного обеспечения [Текст] / А.С. Лихицкий // Молодой ученый. — 2016. — № 9 (113). — С. 71-74.

ПРИЛОЖЕНИЯ

Приложение 1. Класс Browser

```
using System;
using System.Threading;
using CefSharp;
using CefSharp.WinForms;

namespace Единый_аккаунт
{
  internal class Browser
  {
    /// <summary>
    /// Страница браузера
    /// </summary>
    internal ChromiumWebBrowser Page;
    /// <summary>
    /// Фрэйм загружен
    /// </summary>
    bool isFrameLoad = false;

    string htmlEmpty = String.Format(@"<html><body
bgcolor='DDDDDD'></body></html>");

    // По умолчанию не управляет тайм-аутами, поэтому его нужно
реализовать
    private ManualResetEvent manualResetEvent = new
ManualResetEvent(false);

    public Browser(string url)
    {
      // Автоотключение при закрытии
      CefSharpSettings.ShutdownOnExit = true;

      CefSettings settings = new CefSettings();
      Cef.Initialize(settings);

      // Страница браузера - элемент формы
      Page = new ChromiumWebBrowser(url);


      Page.FrameLoadEnd += (sender, args) =>
      {
        isFrameLoad = true;
        // Уменьшить масштаб, чтоб все поместилось
        Page.SetZoomLevel(-1);
      };
      Page.AddressChanged += (send, args) => isFrameLoad = false;
    }
    /// <summary>
    /// Открыть данный URL
    /// </summary>
```

```
    /// <param name="url">The url</param>
    /// <returns></returns>
    public void OpenUrlAndWait(string url)
    {
      try
      {
        Page.LoadingStateChanged += PageLoadingStateChanged;
        if (Page.IsBrowserInitialized)
        {
          Page.Load(url);

          // создать тайм-аут 60 секунд
          bool isSignalled =
manualResetEvent.WaitOne(TimeSpan.FromSeconds(60));
          manualResetEvent.Reset();

          // Получен ответ на запрос
          if (isSignalled)
          {
            // Дождаться понлой загрузки всех скриптов
            WaitFullLoad(Page.Address);
          }
          // Если запрос не получил ответ, принудительно
остановиться по истечении времени ожидания
          else
          {
            Page.Stop();
          }
        }
      }
      catch (ObjectDisposedException)
      {
        // Происходит на manualResetEvent.Reset (); когда токен
отмены отменил контекст
      }
      Page.LoadingStateChanged -= PageLoadingStateChanged;
    }

    /// <summary>
    /// Ожидание полной загрузки
    /// </summary>
    /// <param name="url"></param>
    /// <param name="ms">мс между попытками</param>
    /// <param name="attempts">количество попыток</param>
    internal bool WaitFullLoad(string url, int ms = 200, int
attempts = 30)
    {
      //ждать(загрузку нужной страницы)
      for (int i = 0; i < attempts; i++)
      {
        if (!isFrameLoad || url != Page.Address)
          Thread.Sleep(ms);
```

```
        else
          return true;
      }
      return false;
    }

    /// <summary>
    /// Управление параметром IsLoading
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void PageLoadingStateChanged(object sender,
LoadingStateChangedEventArgs e)
    {
      // Проверка, завершена ли загрузка - это событие вызывается
дважды:
      // сначала, когда начинается загрузка,
      // потом, когда он закончил
      if (!e.IsLoading)
      {
        manualResetEvent.Set();
      }
    }
    /// <summary>
    /// Открыть пустую страницу
    /// </summary>
    internal void OpenEmptyPage()
    {
      Page.LoadHtml(htmlEmpty, @"http://ochenunicalniyurl.ru//");
    }

  }
}
```

Приложение 2. Класс UserData

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Security.Cryptography;
using System.Text;

namespace Единый_аккаунт
{
  [Serializable]
  internal class UserData
  {
    internal string Name;

    internal string Password;

    // Группы пользователя
    internal List<Group> Groups = new List<Group>();

    // Аудит действий
    internal StringBuilder Audit = new StringBuilder();

    // Хеш данных пользователя
    private byte[] Hash;

    // Путь до файла пользователя
    string UserFilePath;
    // Путь до папки с файлами всех пользователей
    string UsersFilesPath;

    public UserData(string name, string password, string
usersFilesPath)
    {
      Name = name;
      Password = password;
      UsersFilesPath = usersFilesPath;
      UserFilePath = UsersFilesPath + name;

      // Создание первого хеша (в виде обновления хеша и БД)
      UpdateHashAndDB();
    }

    // Имеется ли группа с таким именем
    private bool GroupExists(string groupName)
    {
      return Groups.Exists((x) => x.Name == groupName);
    }
    // Добавление группы по имени. true - если успешно добавлено
    internal bool AddGroup(string groupName)
    {
      if (groupName != "" && !GroupExists(groupName))
```

```
      {
        // Добавление группы
        Groups.Add(new Group(groupName));

        // Обновление хеша и БД
        UpdateHashAndDB();

        return true;
      }
      else
      {
        return false;
      }
    }
    // Удаление группы
    internal void DeleteGroup(string name)
    {
      Groups.RemoveAll((x) => x.Name == name);

      // Обновление хеша и БД
      UpdateHashAndDB();
    }
    // Изменение имени группы
    internal void ChangeGroupName(string grOldName, string
grNewName)
    {
      Groups.Find(x => x.Name ==
grOldName).SetNewName(grNewName);
      // Обновить хеш и БД
      UpdateHashAndDB();
    }
    // Изменение пароля группы
    internal void ChangeGroupPassword(string grName, string
password)
    {
      Groups.Find(x => x.Name ==
grName).password.SetNew(password);
      // Обновить хеш и БД
      UpdateHashAndDB();
    }


    // Обновление хеша и БД
    internal void UpdateHashAndDB()
    {
      // Обновление хеша
      UpdateHash();
      // Сериализовать и записать в БД
      UpdateInfoInDB();
    }
```

```
// Обновление хеша
private void UpdateHash()
{
  Hash = CalcCurrentHash();
}
// Проверка хеша
internal bool CheckHash()
{

  // Подсчет текущего хеша
  var current_hash = CalcCurrentHash();

  if (current_hash == null && Hash == null)
    return true;
  if (current_hash.Length != Hash.Length)
    return false;

  for(int i=0; i < Hash.Length; i++)
  {
    // Сравнение текущего с Hash
    if (current_hash[i] != Hash[i])
      return false;
  }
  return true;
}
// Посчитать текущий хеш
private byte[] CalcCurrentHash()
{
  // Сериализовать
  byte[] UserDataForHash = SerializationWithoutHash();

  // Хешировать данные
  using (SHA512 mySHA512 = SHA512.Create())
  {
    return mySHA512.ComputeHash(UserDataForHash);
  }

}

// Сериализовать без хеша
private byte[] SerializationWithoutHash()
{
  MemoryStream ms_gr = new MemoryStream();
  BinaryFormatter bf_gr = new BinaryFormatter();
  bf_gr.Serialize(ms_gr, Groups);
  MemoryStream ms_au = new MemoryStream();
  BinaryFormatter bf_au = new BinaryFormatter();
  bf_au.Serialize(ms_au, Audit);
  MemoryStream ms_na = new MemoryStream();
  BinaryFormatter bf_na = new BinaryFormatter();
  bf_na.Serialize(ms_na, Name);
```

```
      var list_bytes = new List<byte>();

      list_bytes.AddRange(ms_au.ToArray());
      list_bytes.AddRange(ms_na.ToArray());
      list_bytes.AddRange(ms_gr.ToArray());

      return list_bytes.ToArray();
    }

    // Обновить бинарную информацию о пользователе
    private void UpdateInfoInDB()
    {
      // Сериализовать
      byte[] UserDataForWrite = Serialization();

      // Здесь Шифрование данных
      AesCryptographyService myAes = new
AesCryptographyService(Encoding.Default.GetBytes(Password));
      UserDataForWrite = myAes.Encrypt(UserDataForWrite);

      // Запись в файл
      // Создать папку, если ее еще нет
      UsersFilesPath = @"Users\"; // Полный путь не принимает
      if (!Directory.Exists(UsersFilesPath))
        Directory.CreateDirectory(UsersFilesPath);

      // Запись бинарных данных пользователя в память
      using (BinaryWriter bw = new
BinaryWriter(File.Open(UsersFilesPath + Name, FileMode.Create)))
      {
        bw.Write(UserDataForWrite);
      }
    }

    // Сериализация UserData в байты
    internal byte[] Serialization()
    {
      MemoryStream ms = new MemoryStream();
      BinaryFormatter formatter = new BinaryFormatter();
      formatter.Serialize(ms, this);
      return ms.ToArray();
    }

    // Удаление сервиса из группы
    internal bool DeleteServiceInGroup(string groupName, Service
service)
    {
      // Поиск группы по имени
      Group g = Groups.Find((x) => x.Name == groupName);
      // Если такой сервис в этой группе есть
      if (g.ServicesInGroup.Exists((x) => x.service.Name ==
service.Name))
```

```
      {
        // Найти этот сервис
        var sig = g.ServicesInGroup.Find((x) => x.service.Name ==
service.Name);
        // Удалить сервис из группы
        g.ServicesInGroup.Remove(sig);
        // Обновить хеш и БД
        UpdateHashAndDB();
        return true;
      }
      else
      {
        return false;
      }
    }

    // Добавление сервиса в группу
    internal bool AddServiceInGroup(string groupName,
ServiceInGroup sig)
    {
      // Поиск группы по имени
      Group g = Groups.Find((x) => x.Name == groupName);

      // Если такого сервиса в этой группе еще нет
      if (!g.ServicesInGroup.Exists((x) => x.service ==
sig.service))
      {
        // Добавить сервис в группу
        g.ServicesInGroup.Add(sig);
        // Обновить хеш и БД
        UpdateHashAndDB();
        return true;
      }
      else
      {
        return false;
      }
    }
    // Добавление строки в аудит
    internal void AddInAudit(string text)
    {
      Audit.Append(text + "\n");
      // Обновить хеш и БД
      UpdateHashAndDB();
    }
  }
}
```

Приложение 3. Код основной рабочей формы

```
using System;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Net;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using CefSharp;
using HtmlAgilityPack;
using MySql.Data.MySqlClient;

namespace Единый_аккаунт
{
  internal partial class FormWork : Form
  {
    #region Запуск
    UserData userData;
    List<Service> AvailableServices;
    // Константы с названиями сервисов нужны,
    // потому что для каждого сервиса свой код парсинга и
авторизации
    const string VKONTAKTE = "Вконтакте";
    const string ODNOKLASSNIKI = "Одноклассники";
    const string INSTAGRAM = "Инстаграм";
    const string MAILRU = "Mail.ru";
    const string GMAIL = "Gmail";

    // Конструктор.
    public FormWork(UserData ud, Browser browser_c)
    {
      InitializeComponent();

      // Инициализация браузера
      InitializeBrowser(browser_c);

      // Пользовательские данные
      userData = ud;

      Text = Text + " (" + userData.Name + ")";

      NewHint("Успешный вход в \"Единый Аккаунт\" пользователя
\"" + userData.Name + "\"");
    }
    // Метод при показе формы
    private void FormBasic_Shown(object sender, EventArgs e)
    {
      // Загрузка из бд доступных сервисов
      AvailableServices = LoadServicesFromDB();

      // AvailableServices вывести в CheckedListBox
```

```
      foreach (var serv in AvailableServices)
      {
        cLB_Services.Items.Add(serv.Name);

        // В меню сверху добавитьсервис
        tSMI_openService.DropDownItems.Add(serv.Name, null,
(send, args) => browser.OpenUrlAndWait(serv.UrlMain));
      }

      // Показать группы данного пользователя
      UpdateGroupsOnForm();

      NewHint("\"Единый Аккаунт\" успешно загружен!");
    }
    #endregion

    #region Браузер
    // Браузер
    Browser browser;

    // Инициализация
    private void InitializeBrowser(Browser browser_c)
    {

      browser = browser_c;

      // Перейти на пустую страницу
      browser.OpenEmptyPage();

      // Добавление браузера в tLP_Main (3 столбец, 1 строка)
      tableLP_Main.Controls.Add(browser.Page, 2, 0);

      // На 3 строчки расстянуть (в 1 столбце)
      tableLP_Main.SetRowSpan(browser.Page, 3);
      tableLP_Main.SetColumnSpan(browser.Page, 1);
    }

    // Авторизация в определенном сервисе
    private bool Authorization(ServiceInGroup sig)
    {
      try
      {
        // Запретить действия до конца авторизации
        NewHint("Идёт авторизация в сервис \"" + sig.service.Name
+ "\". Пожалуйста, немного подождите!");
        SetActionPermission(false);

        // Открыть страницу входа в сервис
        browser.OpenUrlAndWait(sig.service.UrlAuth);

        // Выбор сервиса для авторизации
        switch (sig.service.Name)
```

```
        {
          case VKONTAKTE:
          {

  //browser.Page.ExecuteScriptAsyncWhenPageLoaded(@"document.get
ElementById('email').value = '" + sig.Login + "';" +

  browser.Page.GetMainFrame().ExecuteJavaScriptAsync(@"document.
getElementById('email').value = '" + sig.Login + "';" +

  @"document.getElementById('pass').value = '" + sig.Password +
"';" +

  @"document.getElementById('login_button').click();");

            break;
          }
          case ODNOKLASSNIKI:
          {

  browser.Page.GetMainFrame().ExecuteJavaScriptAsync(@"document.
getElementById('field_email').value = '" + sig.Login + "';" +

  @"document.getElementById('field_password').value = '" +
sig.Password + "';" +

  @"document.getElementsByClassName('button-pro
__wide')[0].click();");
            break;
          }
          case INSTAGRAM:
          {
            // Время для загрузки элементов ввода страницы
            Thread.Sleep(ms_wait);

            // Выбрать элемент, чтобы эмуляция нажатия tab
происходила в нужном месте
            browser.Page.Select();
            // В этом задании эмуляция ввода логина и пароля
            var auth = Task.Factory.StartNew(() =>
            {

              // Эмуляция ввода логина и пароля
              SendKeys.SendWait("\t");
              SendKeys.SendWait(sig.Login);
              SendKeys.SendWait("\t");
              SendKeys.SendWait(sig.Password);
              SendKeys.SendWait("{ENTER}");
            });
            // Ожидание выполнения задания
            auth.Wait();
```

```
            break;
          }
          case MAILRU:
          {
            // Время для загрузки элементов ввода страницы
            Thread.Sleep(ms_wait);
            // Выбрать элемент, чтобы эмуляция нажатия tab
происходила в нужном месте
            browser.Page.Select();
            // В этом задании эмуляция ввода логина и пароля
            var auth = Task.Factory.StartNew(() =>
            {
              // Эмуляция ввода логина и пароля
              SendKeys.SendWait(sig.Login);
              SendKeys.SendWait("{ENTER}");
              Thread.Sleep(300); // Можно попробовать меньше
              SendKeys.SendWait(sig.Password);
              SendKeys.SendWait("{ENTER}");
            });
            // Ожидание выполнения задания
            auth.Wait();
            break;
          }
          case GMAIL:
          {
            // Время для загрузки элементов ввода страницы
            Thread.Sleep(ms_wait);
            // Выбрать элемент, чтобы эмуляция нажатия tab
происходила в нужном месте
            browser.Page.Select();
            // В этом задании эмуляция ввода логина и пароля
            var auth = Task.Factory.StartNew(() =>
            {
              // Эмуляция ввода логина и пароля
              SendKeys.SendWait(sig.Login);
              SendKeys.SendWait("{ENTER}");
              Thread.Sleep(300); // Можно попробовать меньше
              SendKeys.SendWait(sig.Password);
              SendKeys.SendWait("{ENTER}");
            });
            // Ожидание выполнения задания
            auth.Wait();
            break;
          }
          default:
            break;
        }


        // В случае двухфакторной авторизации (или авторизации
при помощи QR-кода) в этом месте необходимы дополнительные
действия пользователя
```

```
        //...

        // Ожидание загрузки страницы с сообщениями
        if (!browser.WaitFullLoad(sig.service.UrlAfterAuth))
        {
          // Пока что true, т к, возможно, произошло изменение в
сервисе, и он отправляет на другую страницу
          // browser.Page.Address;
          return true;

          // Если не хватило попыток, чтобы дождаться
          // return false;
        }
      }
      catch (Exception)
      {
        // Исключительная ситуация при авторизации
        return false;
      }
      finally
      {
        // Разрешить действия
        SetActionPermission(true);
      }

      return true;
    }

    // Установить разрешение на действия с элементами формы,
запретить при работе браузера
    private void SetActionPermission(bool v)
    {
      btnAddGroup.Enabled = v;
      btn_UpdateChats.Enabled = v;
      cLB_Services.Enabled = v;
      fLP_groups.Enabled = v;
    }

    #endregion

    #region Парсинг сервисов
    // Парсинг чатов Вконтакте
    private List<Chat> ParcingVkontakte(Service service,
HtmlAgilityPack.HtmlDocument htmlDoc)
    {
      List<Chat> newChats = new List<Chat>();

      int IndChat = -1;
      var chats =
htmlDoc.DocumentNode.SelectNodes("//*[@id=\"im_dialogs\"]/li");
      foreach (var li in chats)
      {
```

```
        IndChat++;
        // С кем диалог
        var OtKogo = li.SelectSingleNode(li.XPath +
"/div/div/div/span/span[@class=\"_im_dialog_link\"]").InnerText;

        // Последнее сообщение от пользователя
        bool isMyLastMessage = false;
        // Сообщения
        string LastMessage = "";

        // Если последнее сообщение от собеседника
        var LastMessage_node = li.SelectSingleNode(li.XPath +
"/div/div/div/span[@class=\"nim-dialog--preview
_dialog_body\"]");
        if (LastMessage_node != null)
        {
          isMyLastMessage = false;
          LastMessage = LastMessage_node.InnerText;
        }

        // Если сообщение от меня или в беседе.
        LastMessage_node = li.SelectSingleNode(li.XPath +
"/div/div/div/span/span[@class=\"nim-dialog--inner-text\"]");
        if (LastMessage_node != null)
        {
          isMyLastMessage = true;
          LastMessage = LastMessage_node.InnerText;
        }

        //Количество непрочитанных
        var count_unread_node = li.SelectSingleNode(li.XPath +
"/div/div/div/div[@class=\"nim-dialog--unread
_im_dialog_unread_ct\"]");
        var count_unread = count_unread_node.InnerText;
        // Наличие непрочитанных
        bool isHaveUnread = false;
        if (count_unread != "")
        {
          isHaveUnread = true;
        }

        //Url
        var url_dialog = @"https://vk.com/im?sel=" +
li.GetAttributeValue("data-peer", "-1");

        // Фото
        Bitmap photo = null;
        var photo_node = li.SelectSingleNode(li.XPath +
"/div/div/div/div/a/div[@class=\"im_grid\"]/img");
        if (photo_node != null)
        {
```

```
          string photo_src = photo_node.GetAttributeValue("src",
"-1");

          // если нет фото... А для бесед -1, если колаж из
участников
          if (photo_src.Equals("/images/camera_50.png?ava=1"))
          {
            // нет фото
            photo = null;
          }
          else if (photo_src.Equals("-1"))
          {
            //беседа без фото
            photo = null;
          }
          else if
(photo_src.Equals("/images/deactivated_50.png?ava=1"))
          {
            //удаленная страница
            photo = null;
          }
          else
          {
            photo = new Bitmap((new
WebClient()).OpenRead(photo_src));
          }
        }
        Chat chat = new Chat(service, IndChat.ToString(), OtKogo,
LastMessage, isMyLastMessage, isHaveUnread, count_unread,
url_dialog, photo);
        newChats.Add(chat);
      }

      return newChats;
    }

    // Парсинг чатов Одноклассников
    private List<Chat> ParcingOdnoklassniki(Service service,
HtmlAgilityPack.HtmlDocument htmlDoc)
    {
      List<Chat> newChats = new List<Chat>();

      // Индексация чатов у одного сервиса
      int IndChat = -1;
      var chats =
htmlDoc.DocumentNode.SelectNodes("//*[@tsid=\"conversation_item\
"]");
      foreach (var li in chats)
      {
        IndChat++;
        // С кем диалог
```

```
        var OtKogo = li.SelectSingleNode(li.XPath +
"/div/div[@class=\"chats_i_h ellip\"]").InnerText;

        // Последнее от пользователя
        bool isMyLastMessage = false;
        // Сообщение
        string LastMessage = "";
        var node_mess_1 = li.SelectSingleNode(li.XPath +
"/div/div/div[@class=\"chats_i_tx ellip __has-ava\"]");
        var node_mess_2 = li.SelectSingleNode(li.XPath +
"/div/div/div[@class=\"chats_i_tx ellip\"]");
        // Если сообщение от меня
        if (node_mess_1 != null)
        {
          LastMessage = node_mess_1.InnerText.Replace(" ",
"");
          isMyLastMessage = true;
        }
        // Если сообщение мне
        else if (node_mess_2 != null)
        {
          LastMessage = node_mess_2.InnerText.Replace(" ",
"");
          isMyLastMessage = false;
        }
        //Наличие непрочитанных
        bool isHaveUnread;
        //Количество непрочитанных
        string count_unread;
        var count_unread_node = li.SelectSingleNode(li.XPath +
"/div/div[@class=\"counterText\"]");
        // Если есть непрочитанные
        if (count_unread_node != null)
        {
          isHaveUnread = true;
          count_unread = count_unread_node.InnerText;
        }
        else
        {
          isHaveUnread = false;
          count_unread = "";
        }

        //Url
        var url_dialog = @"https://ok.ru" +
li.SelectSingleNode(li.XPath +
"/a[@class=\"chats_i_ovr\"]").GetAttributeValue("href", "-1");

        // Фото
        Bitmap photo;
        var phot_node = li.SelectSingleNode(li.XPath +
"/div/img");
```

```
        if (phot_node == null)
        {
          // нет фото
          photo = null;
        }
        else
        {
          string photo_src = @"https:" +
phot_node.GetAttributeValue("src", "-1").Replace("amp;", "");
          photo = new Bitmap((new
WebClient()).OpenRead(photo_src));
        }

        Chat chat = new Chat(service, IndChat.ToString(), OtKogo,
LastMessage, isMyLastMessage, isHaveUnread, count_unread,
url_dialog, photo);
        newChats.Add(chat);
      }

      return newChats;
    }

    // Парсинг чатов Инстаграма
    private List<Chat> ParcingInstagram(Service service,
HtmlAgilityPack.HtmlDocument htmlDoc)
    {
      List<Chat> newChats = new List<Chat>();

      // Индексация чатов у одного сервиса
      int IndChat = -1;
      //var chats =
htmlDoc.DocumentNode.SelectNodes("html/body/div/section/div/div/
div/div/div[1]/div/div/div/div/div/div[@class=\"
Igw0E   rBNOH        eGOV_     ybXk5    _4EzTm
\"]");
      var chats = htmlDoc.DocumentNode.SelectNodes("//*[@class=\"
DPiy6            Igw0E     IwRSH      eGOV_         _4EzTm
\"]");

      foreach (var li in chats)
      {
        IndChat++;
        // С кем диалог
        string OtKogo = "";
        // Если нет непрочитанных
        var OtKogo_node = li.SelectSingleNode(li.XPath +
"/a/div/div/div/div/div/div/div[@class=\"_7UhW9   xLCgt
MMzan  KV-D4             fDxYl     \"]");
        if(OtKogo_node != null)
        {
          OtKogo = OtKogo_node.InnerText;
```

```
        }
        // Если есть непрочитанные
        OtKogo_node = li.SelectSingleNode(li.XPath +
"/a/div/div/div/div/div/div/div[@class=\"_7UhW9   xLCgt
qyrsm KV-D4             fDxYl     \"]");
        if (OtKogo_node != null)
        {
          OtKogo = OtKogo_node.InnerText;
        }

        // Последнее сообщение от пользователя?
        bool isMyLastMessage = false;

        // Наличие непрочитанных
        bool isHaveUnread;
        //Количество непрочитанных
        var count_unread = "";
        var count_unread_node = li.SelectSingleNode(li.XPath +
"/a/div/div/div[@class=\" _41V_T  Sapc9                 Igw0E
IwRSH      eGOV_          _4EzTm
\"]");
        // Если есть этот узел, значит есть непрочитанные
        isHaveUnread = count_unread_node != null;

        // (Непрочитанное и прочитанное сообщение по другому
адресу)
        HtmlNode node_mess = null;
        if (isHaveUnread)
          node_mess = li.SelectSingleNode(li.XPath +
"/a/div/div/div/div/div/span/span[@class=\"_7UhW9   xLCgt
qyrsm KV-D4          se6yk        \"]");
        else
          node_mess = li.SelectSingleNode(li.XPath +
"/a/div/div/div/div/div/span/span[@class=\"_7UhW9   xLCgt
MMzan   _0PwGv         se6yk        \"]");

          // Последнее сообщение
        string LastMessage = "";
        // Если есть текст сообщения
        if (node_mess != null)
        {
          LastMessage = node_mess.InnerText.Replace("nbsp;","");
        }
        //Url
        // переход к чату document.getElementsByClassName("-qQT3
rOtsg")[0].click();\
        var url_dialog = @"https://www.instagram.com" +
li.SelectSingleNode(li.XPath + "/a[@class=\"-qQT3
rOtsg\"]").GetAttributeValue("href", "-1");

        // Фотка
        Bitmap photo;
```

```
        var phot_node = li.SelectSingleNode(li.XPath +
"/a/div/div/div/span/img");

        // Если нет фотографии
        if (phot_node == null)
        {
          photo = null;
        }
        else
        {
          string photo_src = phot_node.GetAttributeValue("src",
"-1").Replace("amp;", "");
          photo = new Bitmap((new
WebClient()).OpenRead(photo_src));
        }
        Chat chat = new Chat(service, IndChat.ToString(), OtKogo,
LastMessage, isMyLastMessage, isHaveUnread, count_unread,
url_dialog, photo);
        //Chat chat = new Chat(service, IndChat.ToString(),
OtKogo, "", false, false, "", "", null);
        newChats.Add(chat);
      }

      return newChats;
    }

    // Парсинг сообщений Mail.ru
    private List<Chat> ParcingMailru(Service service,
HtmlAgilityPack.HtmlDocument htmlDoc)
    {
      List<Chat> newChats = new List<Chat>();
      #region Несгруппированные сообщения
      int IndChat = -1;
      var chats =
htmlDoc.DocumentNode.SelectNodes("//*[@class=\"llc js-tooltip-
direction_letter-bottom js-letter-list-item llc_normal\"]");
      foreach (var li in chats)
      {
        IndChat++;

        // С кем диалог
        string OtKogo = "";
        // От кого
        var OtKogo_node = li.SelectSingleNode(li.XPath +
"/div/div/div/span[@class=\"ll-crpt\"]");
        if (OtKogo_node != null)
        {
          OtKogo = OtKogo_node.InnerText.Trim();
        }

        // Последнее сообщение от пользователя (здесь всегда от
собеседника)
```

```
        bool isMyLastMessage = false;
        //Количество непрочитанных (числа нет)
        var count_unread = "";

        // Наличие непрочитанных
        var isUnread_node = li.SelectSingleNode(li.XPath +
"/div/span[@class=\"ll-rs ll-rs_is-active\"]"); ;
        bool isHaveUnread = isUnread_node != null;

        // Название сообщения
        HtmlNode node_mess = li.SelectSingleNode(li.XPath +
"/div/div/div/span/span[@class=\"ll-sj__normal\"]");

        // Последнее сообщение
        string LastMessage = "";
        // Если есть текст сообщения
        if (node_mess != null)
        {
          LastMessage =
node_mess.InnerText.Trim().Replace("gt;",">").Replace("lt;","<")
;
        }

        //Url
        // переход к чату
        var url_dialog = @"https://e.mail.ru" +
li.GetAttributeValue("href", "-1");


        // Фото
        Bitmap photo;
        var phot_node = li.SelectSingleNode(li.XPath +
"/div/button/div/span[@class=\"ll-av__img\"]");

        // Если нет фотографии
        if (phot_node == null)
        {
          photo = null;
        }
        else
        {
          string photo_src = phot_node.GetAttributeValue("style",
"-1").Replace("background-image: url(&quot;",
"").Replace("&quot;);", "").Replace("amp;","");
          photo = new Bitmap((new
WebClient()).OpenRead(photo_src));
        }
        Chat chat = new Chat(service, IndChat.ToString(), OtKogo,
LastMessage, isMyLastMessage, isHaveUnread, count_unread,
url_dialog, photo);
        newChats.Add(chat);
      }
```

```
      #endregion
      #region Сгруппированные сообщения
      var chats_gr =
htmlDoc.DocumentNode.SelectNodes("//*[@class=\"metathread
metathread_collapsed\"]");
      foreach (var li in chats_gr)
      {
        IndChat++;

        // С кем диалог
        string OtKogo = "";
        // От рассылок
        var OtKogo_node = li.SelectSingleNode(li.XPath +
"/div/a/div/div/span[@class=\"mt-t mt-t_newsletters\"]");
        if (OtKogo_node != null)
        {
          OtKogo = OtKogo_node.InnerText.Trim();
        }
        // От соц сетей
        OtKogo_node = li.SelectSingleNode(li.XPath +
"/div/a/div/div/span[@class=\"mt-t mt-t_social\"]");
        if (OtKogo_node != null)
        {
          OtKogo = OtKogo_node.InnerText.Trim();
        }

        // Последнее сообщение от пользователя (здесь всегда от
собеседника)
        bool isMyLastMessage = false;
        //Количество непрочитанных (числа нет)
        var count_unread = "";

        // Наличие непрочитанных
        var isUnread_node = li.SelectSingleNode(li.XPath +
"/div/a/div/span[@class=\"ll-rs ll-rs_is-active\"]");
        bool isHaveUnread = isUnread_node != null;


        // Название сообщения
        HtmlNode node_mess = li.SelectSingleNode(li.XPath +
"/div/a/div/div/div[@class=\"mt-snt\"]");

        // Последнее сообщение
        string LastMessage = "";
        // Если есть текст сообщения
        if (node_mess != null)
        {
          LastMessage = node_mess.InnerText.Trim();
        }

        //Url
```

```
        // переход к чату. можно просто раскрыть список
document.getElementsByClassName("")[].click();
        var url_dialog = @"https://e.mail.ru/";


        // Фото (SVG не парсится)
        Bitmap photo = null;
        Chat chat = new Chat(service, IndChat.ToString(), OtKogo,
LastMessage, isMyLastMessage, isHaveUnread, count_unread,
url_dialog, photo);
        newChats.Add(chat);
      }
      #endregion
      return newChats;
    }
    #endregion

    #region Работа с чатами
    // Получить обновленные чаты.
    // При модернизации исправить: Сбор чатов сделать Offscreen,
Заменить Thread.Sleep()
    private List<Chat> GetUpdatedChats()
    {
      List<Chat> Chats = new List<Chat>();

      // Активная группа
      Group g = userData.Groups.Find((x) => x.Name ==
ActiveGroup());
      // Цикл по актвным сервисам
      foreach (var sig in g.ServicesInGroup)
      {
        try
        {
          NewHint("Поиск чатов в сервисе \"" + sig.service.Name +
"\"");
          // Парсинг, поиск чатов
          browser.OpenUrlAndWait(sig.service.UrlMessages);

          Thread.Sleep(ms_wait);

          // Получить html код
          var t = browser.Page.GetSourceAsync();
          t.Wait();
          string html_code = t.Result;

          var htmlDoc = new HtmlAgilityPack.HtmlDocument();
          htmlDoc.LoadHtml(html_code);

          switch (sig.service.Name)
          {
            case INSTAGRAM:
            {
```

```
              Chats.AddRange(ParcingInstagram(sig.service,
htmlDoc));
              break;
            }
            case ODNOKLASSNIKI:
            {
              Chats.AddRange(ParcingOdnoklassniki(sig.service,
htmlDoc));
              break;
            }
            case VKONTAKTE:
            {
              Chats.AddRange(ParcingVkontakte(sig.service,
htmlDoc));
              break;
            }
            case MAILRU:
            {
              Chats.AddRange(ParcingMailru(sig.service,
htmlDoc));
              break;
            }
            default:
            {
              break;
            }
          }
        }
        catch (Exception)
        {
          NewHint("При поиске чатов в сервисе \"" +
sig.service.Name + "\" произошла ошибка!");
          // Скорее всего, страница с чатами не успела полностью
загрузиться, можно снова попробовать
        }
      }

      NewHint("Сбор чатов активных сервисов закончен");

      return Chats;
    }

    // Обновить чаты активных сервисов
    // При модернизации скрывать страницу во время сбора чатов
    private void UpdateChats()
    {
      // Запретить действия на некоторые элементы в форме на
время обнвовления
      SetActionPermission(false);

      // Скрыть браузер
      browser.Page.Visible = false;
```

```
      // Убрать все предыдущие чаты
      fLP_chats.Controls.Clear();

      // Получить обновленные чаты
      List<Chat> Chats = GetUpdatedChats();

      // Вывести чаты с непрочитанными сообщениями
      OutputChats(Chats.FindAll(x => x.isHaveUnread));
      // Вывести чаты с прочитанными сообщениями
      OutputChats(Chats.FindAll(x => !x.isHaveUnread));

      // Показать браузер
      browser.OpenEmptyPage();

      // Разрешить действия на некоторые элементы в форме на
время обнвовления
      SetActionPermission(true);

      // Сделать браузер видимым
      browser.Page.Visible = true;
    }

    // Вывести чаты
    private void OutputChats(List<Chat> chats)
    {
      foreach (var chat in chats)
      {
        // Таблица одного чата
        var tLP_chat_one = new TableLayoutPanel
        {
          BackColor = Color.LightGray,
          ColumnCount = 3,
          RowCount = 3,
          Size = new Size(fLP_chats.Width - 30, 60),
          // Привязка чата
          Tag = chat
        };
        tLP_chat_one.Click += Chat_Click;

        tLP_chat_one.ColumnStyles.Add(new
ColumnStyle(SizeType.Percent, 23F));
        tLP_chat_one.ColumnStyles.Add(new
ColumnStyle(SizeType.Percent, 67F));
        tLP_chat_one.ColumnStyles.Add(new
ColumnStyle(SizeType.Percent, 10F));
        tLP_chat_one.RowStyles.Add(new RowStyle(SizeType.Percent,
33.33333F));
        tLP_chat_one.RowStyles.Add(new RowStyle(SizeType.Percent,
33.33333F));
        tLP_chat_one.RowStyles.Add(new RowStyle(SizeType.Percent,
33.33333F));
```

```
        // Добавление фото
        PictureBox pb_photo;
        pb_photo = new PictureBox
        {
          Dock = DockStyle.Fill,
          SizeMode = PictureBoxSizeMode.StretchImage
        };
        if (chat.Photo != null)
          pb_photo.Image = new Bitmap(chat.Photo);

        pb_photo.Click += Chat_Click;

        tLP_chat_one.Controls.Add(pb_photo, 0, 0);
        tLP_chat_one.SetRowSpan(pb_photo, 3);

        // Добавление имени собеседника
        Label labelInterlocutor = new Label
        {
          Dock = DockStyle.Fill,
          Text = chat.Interlocutor,
          Font = new Font("Microsoft Sans Serif", 8, Font.Style |
FontStyle.Bold),
          AutoSize = true
        };
        labelInterlocutor.Click += Chat_Click;
        tLP_chat_one.Controls.Add(labelInterlocutor, 1, 0);

        // Добавление последнего сообщения
        Label labelLastMessage = new Label
        {
          Dock = DockStyle.Fill,
          Text = chat.LastMessage,
          AutoSize = true
        };
        labelLastMessage.Click += Chat_Click;
        tLP_chat_one.Controls.Add(labelLastMessage, 1, 1);

        // Добавление числа непрочитанных сообщений
        Label labelUnreadCount = new Label
        {
          Dock = DockStyle.Fill,
          Text = chat.UnreadCount,
          TextAlign = ContentAlignment.MiddleCenter,
          AutoSize = true
        };
        labelUnreadCount.Click += Chat_Click;
        tLP_chat_one.Controls.Add(labelUnreadCount, 2, 1);

        // Добавление пометки о непрочитанных сообщения
        PictureBox pb_isHaveUnread = new PictureBox
        {
```

```
          Dock = DockStyle.Fill,
          SizeMode = PictureBoxSizeMode.StretchImage,
        };
        if (chat.isHaveUnread)
        {
          pb_isHaveUnread.BackColor = Color.Gray;
        }
        pb_isHaveUnread.Click += Chat_Click;
        tLP_chat_one.Controls.Add(pb_isHaveUnread, 2, 0);

        // Добавление сервиса
        Label labelService = new Label
        {
          Dock = DockStyle.Fill,
          Text = chat.service.Name,
          TextAlign = ContentAlignment.MiddleCenter,
          Font = new Font("Microsoft Sans Serif", 8, Font.Style |
FontStyle.Italic),
        };
        labelService.Click += Chat_Click;
        tLP_chat_one.Controls.Add(labelService, 1, 2);
        tLP_chat_one.SetColumnSpan(labelService, 2);

        fLP_chats.Controls.Add(tLP_chat_one);
      }
    }

    // Клик по чату
    private void Chat_Click(object sender, EventArgs e)
    {
      TableLayoutPanel tlp =
(TableLayoutPanel)((Control)sender).Parent;
      Chat chat = (Chat)tlp.Tag;

      // Открыть чат
      browser.OpenUrlAndWait(chat.Url);

      // Для некоторых сервисов (инстаграм) необходимо надо еще
скрипт выполнить
      // Редактор строки JS для данного чата
      string js = chat.service.ScriptToChat.Replace("{ind_chat}",
chat.IndChat);

      // Если скрипт не пуст, то его необходимо выполнить
      if (js != "")
      {
        // Для полной загрузки фрэймов
        Thread.Sleep(ms_wait);
        browser.Page.GetMainFrame().ExecuteJavaScriptAsync(js);
        //browser.Page.ExecuteScriptAsync(js);
      }
    }
```

```
    // Клик по кнопке "Обновить чаты"
    private void btn_UpdateChats_Click(object sender, EventArgs
e)
    {
      UpdateChats();
    }
    #endregion

    #region Работа с сервисами
    // Изменение Checked у одного из элементов cLB_Services
    private void cLB_Services_ItemCheck(object sender,
ItemCheckEventArgs e)
    {
      var cLB = (CheckedListBox)sender;

      // Убрать выделение
      if (cLB.SelectedIndex != -1)
        cLB.SetSelected(cLB.SelectedIndex, false);

      // Если нужно запрашивать пароль от пользователя. (Если
пользователь сам нажал на галочку)
      if (with_authorization_and_deleting)
      {
        // Определить, какая группа сейчас активна
        string groupName = ActiveGroup();

        // Если нет групп или нет активной
        if (groupName == null)
        {
          e.NewValue = CheckState.Unchecked;
          NewHint("Ошибка! Сначала нужно создать группу или войти
в существующую");
          return;
        }
        // Если группы есть
        else
        {
          // Название сервиса
          string serviceName =
cLB_Services.Items[e.Index].ToString();
          // Поиск выбранного сервиса в списке доступных
          Service service = AvailableServices.Find((x) => x.Name
== serviceName);

          // Если поставлена галочка (добалвен сервис)
          if (e.NewValue == CheckState.Checked)
          {
            // Если не удалось запросить у пользователя логин и
пароль для выбранного сервиса
            if (!RequestLoginPassword(service, out ServiceInGroup
sig))
```

```
            {
              NewHint("Ошибка при добавлении сервиса \"" +
serviceName + "\" в группу \"" + groupName + "\". Попробуйте ещё
раз!");
              e.NewValue = CheckState.Unchecked;
              return;
            }
            //  Процесс авторизации
            if (Authorization(sig))
            {
              // Добавление сервиса в группу пользователя
              if (userData.AddServiceInGroup(groupName, sig))
              {
                // Добавление текста в подсказку
                NewHint("Сервис \"" + serviceName + "\" добавлен
в группу \"" + groupName + "\"");
              }
              else
              {
                NewHint("Ошибка при добавлении сервиса \"" +
serviceName + "\" в группу \"" + groupName + "\". Попробуйте ещё
раз!");
                e.NewValue = CheckState.Unchecked;
              }
            }
            else{
              NewHint("Ошибка при авторизации в сервис \"" +
serviceName + "\". Попробуйте ещё раз!");
              e.NewValue = CheckState.Unchecked;
            }
          }
          // Если галочка убрана (убран сервис)
          else if (e.NewValue == CheckState.Unchecked)
          {
            // Удаление сервиса из группы пользователя
            if (userData.DeleteServiceInGroup(groupName,
service))
            {
              // Добавление текста в подсказку
              NewHint("Сервис \"" + serviceName + "\" удалён из
группы \"" + groupName + "\"");
              // Удалить куки данного сервиса

  Cef.GetGlobalCookieManager().DeleteCookies(service.DomenForCoo
kies);
            }
            else
            {
              NewHint("Ошибка при удалении сервиса \"" +
serviceName + "\" из группы\"" + groupName + "\". Попробуйте ещё
раз!");
              e.NewValue = CheckState.Checked;
```

```
            }
          }

          // Обновить чаты
          UpdateChats();
        }
      }

      // Обновить строку со списком учтных записей активной
группы
      UpdateActiveGroupServicesInfoString();
    }

    // Снять все Checked у сервисов
    private void UncheckedAllServices()
    {
      // Цикл индексов, которым соответсвуют отмеченные элементы
      foreach (int ind in cLB_Services.CheckedIndices)
        cLB_Services.SetItemChecked(ind, false);
    }

    // Запросить у пользователя логин и пароль для выбранного
сервиса.
    private bool RequestLoginPassword(Service service, out
ServiceInGroup sig)
    {
      // В различных сервисах могут быть различные учетные
записи.
      string login_service = "";
      string password_service = "";

      switch (service.Name)
      {
        case VKONTAKTE:
        case ODNOKLASSNIKI:
        case INSTAGRAM:
        case MAILRU:
        case GMAIL:
          login_service = ShowDialogBox("Введите логин для
сервиса \"" + service.Name + "\"");
          // Если логин введен
          if(login_service != null && login_service != "")
            password_service = ShowDialogBox("Ваш логин: " +
login_service + "\nВведите пароль для сервиса \"" + service.Name
+ "\"", true);
          break;
        default:
          break;
      }

      sig = new ServiceInGroup(service, login_service,
password_service);
```

```
      // нужны именно логин и пароль
      if (login_service == null || login_service =="" ||
password_service == null || password_service == "")
        return false;

      return true;
    }

    #endregion

    #region Работа с группами
    // Нажатие кнопок в текстовом поле с новым именем группы
    private void rTB_NewGroupName_KeyPress(object sender,
KeyPressEventArgs e)
    {
      // Если нажат энтер
      if (e.KeyChar == Convert.ToChar(Keys.Enter))
      {
        // Эмуляция нажатия кнопки добавления группы
        btn_add_group_Click(btnAddGroup, new EventArgs());
      }
    }
    // Клик по кнопке "Добавление группы"
    private void btn_add_group_Click(object sender, EventArgs e)
    {
      // Убрать в названии энтеры
      rTB_NewGroupName.Text =
rTB_NewGroupName.Text.Replace("\n","");

      // Создание группы. true, если успешно создана
      if (CreateGroup(rTB_NewGroupName.Text))
      {
        // Обновление групп на форме
        UpdateGroupsOnForm();

        // Сделать активной последнюю группу

  ((RadioButton)fLP_groups.Controls[fLP_groups.Controls.Count -
1]).Checked = true;

        NewHint("Создана новая группа: \"" +
rTB_NewGroupName.Text + "\"");
      }
    }
    // Создание группы (Возвращает true, если успешно создана)
    private bool CreateGroup(string groupName)
    {
      // Добавление группы к объекту класса. true, если успешно
добавлено
      if (userData.AddGroup(groupName))
      {
```

```
        return true;
      }
      else
      {
        NewHint("Выберите другое имя!");
        return false;
      }
    }
    // Удаление группы (при нажатии удалить в контекстном меню
radioButton)
    private void DeleteGroup(object sender, EventArgs e)
    {
      ToolStripMenuItem tsmi = (ToolStripMenuItem) sender;
      ContextMenuStrip cms = (ContextMenuStrip)
tsmi.GetCurrentParent();
      RadioButton rb_group = (RadioButton) cms.SourceControl;
      string grName = rb_group.Text;
      Group group = userData.Groups.Find(x => x.Name == grName);

      // Если пароль не пустой, то нужно его запросить
      if (!group.password.Check())
      {
        string pass = ShowDialogBox("Сначала введите пароль
группы \"" + grName + "\"", true);
        // Если пароль введен неверно
        if (pass == null || !group.password.Check(pass))
        {
          NewHint("Ошибка! Не удалось удалить группу \"" + grName
+ "\", так как введён неверный пароль");
          return;
        }
      }

      // Удаление группы из объекта
      userData.DeleteGroup(grName);

      // Обновление групп на форме
      UpdateGroupsOnForm();

      NewHint("Удалена группа: \"" + grName + "\"");
    }
    // Изменние пароля группы
    private void ChangeGroupPassword(object sender, EventArgs e)
    {
      // У какой группы нужно изменить пароль
      ToolStripMenuItem tsmi = ((ToolStripMenuItem)sender);
      ContextMenuStrip cms =
(ContextMenuStrip)tsmi.GetCurrentParent();
      RadioButton rb_group = (RadioButton)cms.SourceControl;
      string grName = rb_group.Text;
      Group group = userData.Groups.Find(x => x.Name == grName);
```

```
      // Если пароль не пустой, то нужно его запросить
      if (!group.password.Check())
      {
        string pass = ShowDialogBox("Сначала введите старый
пароль группы \"" + grName + "\"", true);
        // Если пароль введен неверно
        if (pass == null || !group.password.Check(pass))
        {
          NewHint("Ошибка! Не удалось изменить пароль группы \""
+ grName + "\", так как введён неверный старый пароль");
          return;
        }
      }
      // Запрос нового пароля
      string answer = ShowDialogBox("Введите новый пароль группы
\"" + grName + "\"", true);

      if (answer != null)
      {
        // Замена старого пароля на новый
        userData.ChangeGroupPassword(grName, answer);

        NewHint("У группы \"" + grName + "\" изменен пароль");
      }
    }
    // Переименование группы
    private void RenameGroup(object sender, EventArgs e)
    {
      // У какой группы нужно изменить имя
      ToolStripMenuItem tsmi = (ToolStripMenuItem)sender;
      ContextMenuStrip cms =
(ContextMenuStrip)tsmi.GetCurrentParent();
      RadioButton rb_group = (RadioButton)cms.SourceControl;
      string grOldName = rb_group.Text;
      Group group = userData.Groups.Find(x => x.Name ==
grOldName);

      // Если пароль не пустой, то нужно его запросить
      if (!group.password.Check())
      {
        string pass = ShowDialogBox("Сначала введите пароль
группы \"" + grOldName + "\"", true);
        // Если пароль введен неверно
        if (pass == null || !group.password.Check(pass))
        {
          NewHint("Ошибка! Не удалось переименовать группу \"" +
grOldName + "\", так как введён неверный пароль");
          return;
        }
      }

      // Запрос нового имени
```

```
      string answer = ShowDialogBox("Введите новое название
группы \"" + grOldName + "\"");

      // Если нажат крестик в диалоговом окне или введена пустая
строка
      if (answer == null || (answer == ""))
      {
        NewHint("Ошибка! Не удалось переименовать группу \"" +
grOldName + "\" в пустую строку");
        return;
      }

      // Если группы с таким именем еще нет
      if (userData.Groups.Find(x=>x.Name == answer) == null)
      {
        // Замена старого имени группы на новое
        userData.ChangeGroupName(grOldName, answer);
        rb_group.Name = answer;
        rb_group.Text = answer;

        NewHint("Группа \"" + grOldName + "\" переименована в
группу \"" + answer + "\"");
      }
      else
      {
        NewHint("Ошибка! Не удалось переименовать группу \"" +
grOldName + "\". Такая группа уже существует");
      }
    }


    // Какая группа активна сейчас
    private string ActiveGroup()
    {
      for (int i = 0; i < fLP_groups.Controls.Count; i++)
      {
        if (((RadioButton)fLP_groups.Controls[i]).Checked)
        {
          return ((RadioButton)fLP_groups.Controls[i]).Name;
        }
      }
      return null;
    }

    // Обновление групп на форме
    private void UpdateGroupsOnForm()
    {
      // Удаление тех групп, которые есть на форме, но которых
больше нет в userData
      for (int i = fLP_groups.Controls.Count - 1; i > -1; i--)
      {
        var control = (RadioButton)fLP_groups.Controls[i];
```

```
        // Если у объекта нет группы, а на форме есть
        if (userData.Groups.FindAll((x) => x.Name ==
control.Name).Count == 0)
        {
          fLP_groups.Controls.RemoveAt(i);
        }
      }

      // Добавление на форму тех групп, которых еще нет на форме,
но есть в userData
      foreach (var group in userData.Groups)
      {
        // Если на форме еще нет такого элемента, то добавить
        if (fLP_groups.Controls.Find(group.Name, true).Length ==
0)
        {
          // Генерация radioButton

          // Строки в контекстном меню
          ToolStripMenuItem tsmi_del = new
ToolStripMenuItem("Удалить");
          ToolStripMenuItem tsmi_rename = new
ToolStripMenuItem("Переименовать");
          ToolStripMenuItem tsmi_changePass = new
ToolStripMenuItem("Изменить пароль");
          // Событие по нажатию на строку
          tsmi_del.Click += DeleteGroup;
          tsmi_rename.Click += RenameGroup;
          tsmi_changePass.Click += ChangeGroupPassword;

          // Создание контекстного меню для будущего radioButton
          ContextMenuStrip cms = new ContextMenuStrip();
          cms.Items.AddRange(new[] { tsmi_del, tsmi_rename,
tsmi_changePass });

          // Добавление RadioButton для этой группы
          RadioButton rb = new RadioButton
          {
              Name = group.Name,
              Text = group.Name,
              Appearance = Appearance.Button,
              AutoSize = true,
              ContextMenuStrip = cms
          };
          // Добавить обработчик нажатия
          rb.CheckedChanged += radioButton_CheckedChanged;

          // Добавление radioButton на форму
          fLP_groups.Controls.Add(rb);
        }
      }
```

```
      // Если групп нет или нет активной
      if (fLP_groups.Controls.Count == 0 || ActiveGroup() ==
null)
      {
        // Убрать все отметки возле сервисов
        UncheckedAllServices();

        // Обновить строку со списком учтных записей активной
группы
        UpdateActiveGroupServicesInfoString();
      }
    }

    // Метод при изменении активного radioButton
    private void radioButton_CheckedChanged(object sender,
EventArgs e)
    {
      var rB_group = (RadioButton)sender;
      string grName = rB_group.Text;
      Group group = userData.Groups.Find(x => x.Name == grName);


      // Если этот radioButton стал активным
      if (rB_group.Checked)
      {
        // Запрос пароля
        // Если пароль не пустой, то нужно его запросить
        if (!group.password.Check())
        {
          string pass = ShowDialogBox("Сначала введите пароль
группы \"" + grName + "\"", true);
          // Если пароль введен неверно
          if (pass == null || !group.password.Check(pass))
          {
            NewHint("Ошибка! Не удалось войти в группу \"" +
grName + "\", так как введён неверный пароль");
            rB_group.Checked = false;
            fLP_groups.Select();
            return;
          }
        }

        // Очистить окно со списком учтных записей предыдущей
группы
        ClearActiveGroupServicesInfoString();

        // Подсказка о переключении группы
        NewHint("Загрузка группы \"" + ActiveGroup() + "\"");

        // Очистить куки, чтобы выйти из сервисов предыдущей
группы
        Cef.GetGlobalCookieManager().DeleteCookies("", "");
```

```
            // Выделить активную группу
            rB_group.Select();

            // Обновить выделение сервисов
            UpdateServicesThisGroup();

            // Обновить строку со списком учтных записей активной
группы
            UpdateActiveGroupServicesInfoString();

            // Обновить чаты
            UpdateChats();

            // Подсказка о переключении группы
            NewHint("Открыта группа \"" + ActiveGroup() + "\"");
          }
        }

        // Очистить информацию активных сервисов активной группы
пользователя в окне
        private void ClearActiveGroupServicesInfoString()
        {
          rTB_ActiveGroupServicesInfo.Text = "";
        }
        #endregion
      }
}
```

Реализованы следующие компетенции по направлению 10.03.01 «Информационная безопасность»

ОК-1	способность использовать основы философских знаний для формирования мировоззренческой позиции
ОК-2	способность использовать основы экономических знаний в различных сферах деятельности
ОК-3	способность анализировать основные этапы и закономерности исторического развития России, ее место и роль в современном мире для формирования гражданской позиции и развития патриотизма
ОК-4	способность использовать основы правовых знаний в различных сферах деятельности
ОК-5	способность понимать социальную значимость своей будущей профессии, обладать высокой мотивацией к выполнению профессиональной деятельности в области обеспечения информационной безопасности и защиты интересов личности, общества и государства, соблюдать нормы профессиональной этики
ОК-6	способность работать в коллективе, толерантно воспринимая социальные, культурные и иные различия
ОК-7	способность к коммуникации в устной и письменной формах на русском и иностранном языках для решения задач межличностного и межкультурного взаимодействия, в том числе в сфере профессиональной деятельности
ОК-8	способность к самоорганизации и саморазвитию
ОК-9	способность использовать методы и средства физической культуры для обеспечения полноценной социальной и профессиональной деятельности

ОПК-1	способность анализировать физические явления и процессы для решения профессиональных задач
ОПК-2	способность применять соответствующий математический аппарат для решения профессиональных задач
ОПК-3	способность применять положения электротехники, электроники и схемотехники для решения профессиональных задач
ОПК-4	способность понимать значение информации в развитии современного общества, применять информационные технологии для поиска и обработки информации
ОПК-5	способность использовать нормативные правовые акты в профессиональной деятельности
ОПК-6	способность применять приемы оказания первой помощи, методы и средства защиты персонала предприятия и населения в условиях чрезвычайных ситуаций, организовать мероприятия по охране труда и технике безопасности
ОПК-7	способность определять информационные ресурсы, подлежащие защите, угрозы безопасности информации и возможные пути их реализации на основе анализа структуры и содержания информационных процессов и особенностей функционирования объекта защиты
ПК-1	способность выполнять работы по установке, настройке и обслуживанию программных, программно-аппаратных (в том числе криптографических) и технических средств защиты информации
ПК-2	способность применять программные средства системного, прикладного и специального назначения, инструментальные средства, языки и системы

	программирования для решения профессиональных задач
ПК-3	способность администрировать подсистемы информационной безопасности объекта защиты
ПК-4	способность участвовать в работах по реализации политики информационной безопасности, применять комплексный подход к обеспечению информационной безопасности объекта защиты
ПК-5	способность принимать участие в организации и сопровождении аттестации объекта информатизации по требованиям безопасности информации
ПК-6	способность принимать участие в организации и проведении контрольных проверок работоспособности и эффективности применяемых программных, программно-аппаратных и технических средств защиты информации
ПК-7	способность проводить анализ исходных данных для проектирования подсистем и средств обеспечения информационной безопасности и участвовать в проведении технико-экономического обоснования соответствующих проектных решений
ПК-8	способность оформлять рабочую техническую документацию с учетом действующих нормативных и методических документов
ПК-9	способность осуществлять подбор, изучение и обобщение научно-технической литературы, нормативных и методических материалов, составлять обзор по вопросам обеспечения информационной безопасности по профилю своей профессиональной деятельности
ПК-10	способность проводить анализ информационной безопасности объектов и систем на соответствие

	требованиям стандартов в области информационной безопасности
ПК-11	способность проводить эксперименты по заданной методике, обработку, оценку погрешности и достоверности их результатов
ПК-12	способность принимать участие в проведении экспериментальных исследований системы защиты информации
ПК-13	способность принимать участие в формировании, организовывать и поддерживать выполнение комплекса мер по обеспечению информационной безопасности, управлять процессом их реализации
ПК-14	способность организовывать работу малого коллектива исполнителей в профессиональной деятельности
ПК-15	способность организовать технологический процесс защиты информации ограниченного доступа в соответствии с нормативными правовыми актами и нормативными методическими документами Федеральной службы безопасности Российской Федерации, Федеральной службы по техническому и экспортному контролю