МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ «ВИТЕБСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ П. М. МАШЕРОВА»

Факультет математики и информационных технологий

Кафедра прикладного и системного программирования

КУРСОВОЙ ПРОЕКТ

по дисциплине «Дополнительные главы информатики»

СОЗДАНИЕ СЛУЖБЫ WINDOWS НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C#

.

Царенко Антон Дмитриевич,

4 курс, 44 группа

Научный руководитель:

Новый Вадим Владимирович,

Старший преподаватель кафедры прикладного и системного программирования

Витебск, 2023

**Реферат**

Курсовая работа 18 стр., 4 рис., 4 листингов, 5 источников.

WINDOWS, СЛУЖБА WINDOWS, ЖУРНАЛ СОБЫТИЙ WINDOWS, C#.

**Объект исследования** – технологии для разработки службы Windows.

**Предмет исследования** – алгоритмы проектирования и разработки служб Windows.

**Цель работы** – разработка службы Windows, отслеживающей вход и выход пользователей из учетных записей.

**Задачи**:

1. Выполнить анализ открытых источников информации, литературы в сфере базовых принципов функционирования операционных систем, а также служб Windows.
2. Ознакомиться с особенностями реализации службы Windows на языке программирования C#.
3. Подготовить план разработки приложения.
4. Разработать приложение.

**Методы исследования** – изучение документации, материалов научных и периодических изданий по теме исследования, существующих систем, общенаучные методы исследования (описание, анализ и т.д.).

**Теоретическая значимость и элементы новизны**: совершенствование знаний в области операционных систем, разработка службы Windows.

**Практическая значимость**: разработанная служба позволяет отслеживать вход и выход пользователей из учетных записей.

Содержание

[Введение 4](#_Toc153793987)

[1 Обоснование выбора средства реализации и его краткая характеристика 6](#_Toc153793988)

[2 Общая характеристика и функционал разрабатываемого приложения. Этапы разработки 8](#_Toc153793989)

[2.1 Общая характеристика и функционал разрабатываемого приложения 8](#_Toc153793990)

[2.2 Описание этапов разработки 8](#_Toc153793991)

[Заключение 17](#_Toc153793992)

[Список использованных источников 18](#_Toc153793993)

Введение

Службы Windows (Windows Services) – это программные компоненты операционной системы Microsoft Windows, которые выполняются в фоновом режиме без прямого участия пользователя. Они представляют собой автономные процессы, которые могут работать независимо от текущего пользователя или его взаимодействия с системой.

Журналы событий Windows (Windows Event Logs) представляют собой системные журналы, в которых регистрируются различные события и сообщения, связанные с работой операционной системы Windows и установленных приложений. Журналы событий предоставляют ценную информацию для отслеживания и анализа проблем, мониторинга системы и выполнения диагностики.

Журналы событий Windows представляют структурированную и централизованную систему журналирования, которая помогает в анализе и мониторинге работы операционной системы и приложений. Они имеют несколько категорий, таких как "Application" (приложения), "System" (системные), "Security" (безопасность) и другие, каждая из которых содержит соответствующие записи событий.

**Цель работы** – разработка службы, позволяющей отслеживать вход и выход пользователей из учетные записи.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Выполнить анализ открытых источников информации, литературы в сфере функционирования ОС Windows, а именно служб Windows.
2. Ознакомиться с особенностями разработки приложения с использованием языка C#.
3. Составить план разработки приложения и подготовить необходимые шаблоны и библиотеки.

**Методы исследования** – изучение документации, материалов научных и периодических изданий по теме исследования, существующих систем разработки редакторов, общенаучные методы исследования (описание, анализ, классификация, сравнение, аналогия).

Работа включает введение, 2 главы, заключение, список использованных источников. В первой главе предоставлено обоснование выбора средства и платформы для разработки, а также языка программирования. Во второй главе приводится общая характеристика, функционал приложения и описание этапов разработки.

1 Обоснование выбора средства реализации и его краткая характеристика

C# – это гибкий, современный и объектно-ориентированный язык программирования, разработанный компанией Microsoft в рамках платформы .NET. Он был представлен в начале 2000-х годов и получил значительную популярность среди разработчиков для создания различных типов приложений, включая настольные, веб, мобильные и игровые приложения.

Вот некоторые ключевые особенности и характеристики C#:

* Объектно-ориентированное программирование: C# является объектно-ориентированным языком, что позволяет разработчикам организовывать свой код в виде объектов, имеющих свойства и методы. ООП позволяет создавать более модульные, гибкие и легко поддерживаемые приложения.
* Платформа .NET: C# разрабатывался для использования вместе с платформой .NET, которая предоставляет богатый набор инструментов и библиотек для разработки приложений. .NET обеспечивает среду выполнения, управление памятью, безопасность и другие важные функции для работы с C#.
* Многофункциональность: C# поддерживает широкий спектр возможностей, включая многопоточность, обработку исключений, делегаты, события, LINQ (Language Integrated Query) и многое другое. Это позволяет разработчикам создавать мощные и эффективные приложения.
* Переносимость кода: Благодаря платформе .NET, код, написанный на C#, может быть перенесен и запущен на разных платформах, таких как Windows, macOS и Linux. Это позволяет разработчикам создавать многоплатформенные приложения с минимальными изменениями.
* Безопасность и типизация: C# обладает строгой статической типизацией, что позволяет обнаруживать ошибки на стадии компиляции и предотвращать множество ошибок времени выполнения. Кроме того, C# обеспечивает встроенные механизмы безопасности, такие как контроль доступа и обработка исключений.

C# является одним из наиболее популярных языков программирования, особенно в экосистеме разработки Microsoft. Он активно используется для создания широкого спектра приложений, включая Windows-приложения, веб-сайты, серверные приложения, приложения для мобильных устройств и игры.

Язык C# содержит шаблоны и классы для создания служб Windows, а также для работы с журналом событий Windows, что является приоритетным для данной работы.

2 Общая характеристика и функционал разрабатываемого приложения. Этапы разработки

2.1 Общая характеристика и функционал разрабатываемого приложения

Целью разрабатываемого приложения является отслеживание входа и выхода пользователей из учетных записей.

Базовый функционал приложения позволяет просматривать время входа и выхода пользователей из учетных записей.

2.2 Описание этапов разработки

Этапы проекта в соответствии с каскадной моделью:

1. Подготовка и проектирование.

Первым шагом в разработке является подготовка, в процессе которой изучается документация, выдвигаются технические требования, процесс разделяется на ряд мелких задач, ставятся сроки их выполнения.

Техническое задание определяет название средства, описание, основное его назначение, требования к выполнению и результатам работы. В ходе проектирования были выдвинуты следующие технические требования:

1. Возможность запуска службы операционной системы Windows;
2. Возможность просматривать информацию о входе и выходе пользователей из учетных записей Windows.

Следующим этапом после подготовки идёт проектирование, в ходе которого разрабатываются системы и алгоритмы взаимодействия данных, а также структурные компоненты.

1. Реализация приложения.

При создании службы можно использовать шаблон проекта Visual Studio, который называется **Служба Windows**. Этот шаблон автоматически выполняет основную часть работы, ссылаясь на необходимые классы и пространства имен, устанавливая наследование от базового класса для служб.

Следующий шаг – задание определенных свойств для службы:

* Имя службы – AuthLogService;
* Свойство CanStop – false, для предотвращения останова службы (например, из диспетчера служб);
* CanShutdown – true, чтобы указать, что служба хочет принимать уведомления о выключении компьютера, на котором она работает, позволяя ему вызывать процедуру [OnShutdown](https://learn.microsoft.com/ru-ru/dotnet/api/system.serviceprocess.servicebase.onshutdown);
* CanPauseAndContinue – false, для предотвращения приостановки и возобновления работы службы;
* CanHandlePowerEvent – false, чтобы не сообщать службе об изменениях состояния питания компьютера;
* AutoLog – true, для внесения информационных записей в журнал событий приложения, когда служба выполняет некоторое действие;

Затем следует переопределить методы OnStart (код, который выполняется при запуске операционной системы) и OnStop (код, который при передаче службе команды останова диспетчером служб).

Для записи информации о входе и выходе пользователей из учетных записей будем использовать журнал создаваемой службы:

Листинг 1 – создание журнала службы.

eventLog1 = new System.Diagnostics.EventLog();

if(!System.Diagnostics.EventLog.SourceExists("AuthLogSource"))

{

System.Diagnostics.EventLog.CreateEventSource("AuthLogSource", "AuthLogLog");

}

eventLog1.Source = "AuthLogSource";

eventLog1.Log = "AuthLogLog";

Отслеживать вход и выход пользователей из учетных записей можно с помощью Журнала событий Windows, а именно Журнала безопасности. Идентифицировать событие можно по коду:

* 4624 – пользователь успешно вошел на компьютер;
* 4625 – сбой входа. Была предпринята попытка входа с неизвестным именем пользователя или известным именем пользователя с неправильным паролем;
* 4634 – процесс выхода был завершен для пользователя;
* 4647 – пользователь инициировал процесс выхода из системы;

В программе будем использовать код 4624 для отслеживания входа пользователя и 4647 для выхода. Однако событие с кодом 4624 генерируется не только при входе пользователей в систему, но и при других обстоятельствах, которые можно различать по типу входа:

* 0 – System (используется только системной учетной записью, например при запуске системы);
* 2 – Interactive (пользователь успешно вошел в систему на данном компьютере);
* 3 – Network (пользователь или компьютер вошли в систему на данном компьютере через сеть);
* 4 – Batch (пакетный тип входа используется пакетными серверами, исполнение процессов на которых производится по поручению пользователя, но без его прямого вмешательства);
* 5 – Service (служба была запущена диспетчером служб);
* 7 – Unlock (эта рабочая станция была разблокирована);
* 8 – NetworkClearText (пользователь вошел в систему на данном компьютере через сеть, пароль пользователя передан в пакет проверки подлинности в нехешированной форме);
* 9 – NewCredentials (инициатор вызова клонировал свой текущий маркер и указал новые учетные данные для исходящих соединений);
* 10 – RemoteInteractive (пользователь выполнил вход в систему на этом компьютере через службы терминалов или удаленного рабочего стола);
* 11 – CachedInteractive (пользователь выполнил вход в систему на этом компьютере с сетевыми учетными данными, которые хранились локально на компьютере);

Мы будем отслеживать события входа с кодом 2. События с таким кодом также генерируют виртуальные учетные записи (учетная запись, используемая какой-либо службой), поэтому такие записи следует игнорировать.

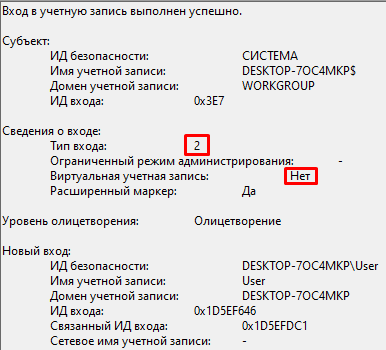


Рисунок 1 – события входа пользователя, код 4624.

Для идентификации события выхода используем код 4647, так как такие события порождает только обычные пользователи (что упрощает работу с такими событиями, так как отсутствует необходимость отслеживания типа входа и виртуальных учетных записей, как в событиях с кодом 4634) и согласно описанию этого кода, данное событие можно рассматривать как событие выхода.

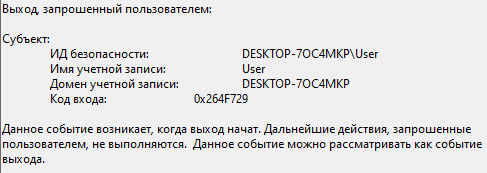


Рисунок 2 – события выхода пользователя, код 4647.

При занесении записи в журнал генерируется событие EntryWritten. Нам нужно получит журнал безопасности Windows и повесить на него обработчик событий.

Листинг 2 – открытие журнала и добавление обработчика событий.

//получаем журнал безопасности

EventLog securityLog = new EventLog("Security");

//добавляем обработчик событий записи

securityLog.EntryWritten += new EntryWrittenEventHandler(MyOnEntryWritten);

Здесь MyOnEntryWritten – функция обработчик события. В ней мы получаем добавленную запись, а затем в зависимости от кода события выполняем определенные действия.

Листинг 3 – обработка события входа.

case 4624:

//получаем тип входа, нам нужен 2

if (GetType(eventLogEntry.Message) == 2)

{

//проверка на виртуальную учетную запись

if (IsVirtual(eventLogEntry.Message))

{

//прверяем время последней записи

                  if (eventLogEntry.TimeGenerated.CompareTo(lastEventTime) > 0)

                  {

                  lastEventTime= eventLogEntry.TimeGenerated;

                        this.machineName = eventLogEntry.MachineName;

                        StringBuilder message2 = new StringBuilder();

                      message2.Append("logon;").Append(eventLogEntry.TimeGenerated)

.Append(";").Append(macAddress).Append(";").Append(ipAddress)

.Append(";").Append(machineName).Append(";")

.Append(GetUserName(eventLogEntry.Message))

.Append(";").Append(GetEnterId(eventLogEntry.Message));

                      eventLog1.WriteEntry(message2.ToString());

            }

    }

  }

      break;

Информацию (время события (т.е. входа пользователя), mac-адрес сетевой карты, ip-адрес компьютера, имя компьютера, имя вошедшего пользователя и код сеанса) записываем в журнал событий создаваемой службы.

Листинг 4 – обработка событий выхода пользователей из учетных записей.

case 4647:

if (eventLogEntry.TimeGenerated.CompareTo(lastEventTime) > 0)

    {

      lastEventTime = eventLogEntry.TimeGenerated;

          StringBuilder message1 = new StringBuilder();

          message1.Append("logoff;").Append(eventLogEntry.TimeGenerated)

.Append(";").Append(macAddress).Append(";").Append(ipAddress).Append(";")

.Append(machineName).Append(";").Append(GetUserName(eventLogEntry.Message))

.Append(";").Append(GetEnterId(eventLogEntry.Message));

          eventLog1.WriteEntry(message1.ToString());

    }

      break;

Получаемая информация аналогична получаемой при входе пользователя.

Следующий шаг – установка службы.

Для установки службы можно использовать команду installutil:



Рисунок 2 – установка службы.

Запустить службу можно из приложения Windows Службы:

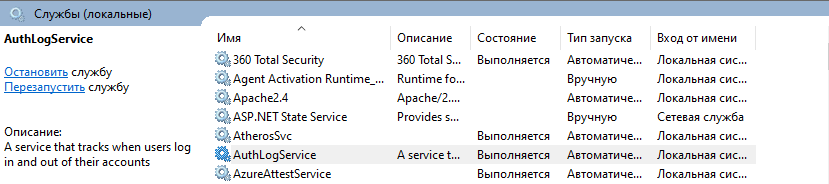


Рисунок 3 – запуск службы.

Теперь при входе и выходе пользователей из системы данные будут заносится в журнал службы AuthLogService.

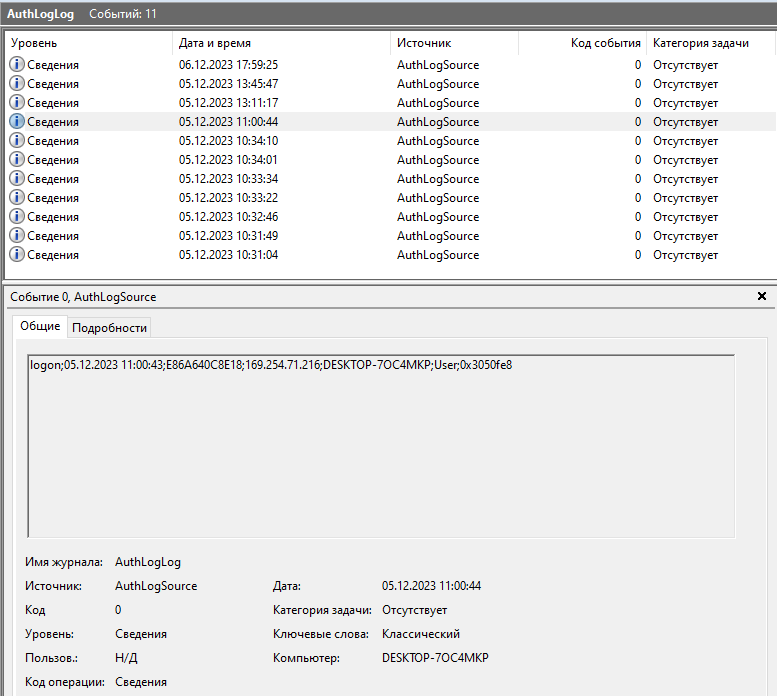


Рисунок 4 – пример просмотра информации о входе пользователя.

Тестирование.

Этап тестирования подразумевает детальную проверку каждого элемента программы с целью нахождения ошибок или неточностей в работе разрабатываемого средства.

В ходе разработки проводилось модульное, интеграционное и системное тестирование.

Модульное тестирование – процесс тестирования, позволяющий проверить работу отдельных модулей исходного кода программы на корректность. На этом этапе проверялась каждая отдельная функция (например, функция создания параметра в реестре).

Интеграционное тестирование – это тестирование, при котором программные модули объединяются логически и тестируются как группа (например, часть приложения для работы с разделами).

Системное тестирование – это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям, как функциональным, так и не функциональным.

1. *Написание документации.*

Завершающим этапом разработки редактора является документирование результатов работы. В случае курсовой работы документированием является детальное изложение каждого этапа разработки, описание технического задания.

Заключение

В данной курсовой работе выполнена разработка службы, записывающей сведения о входе и выходе пользователей из учетных записей.

В результате проделанной работы были решены следующие задачи:

1. Проанализированы открытые источники информации, литература по теме курсовой работы.
2. Изучены особенности создания служб на языке программирования C#.
3. Подготовлен план разработки службы Windows.
4. Реализовано приложение.

Созданное приложение с заданным набором функций свидетельствует о выполнении поставленных задач и о том, что цель работы достигнута.

Список использованных источников

1. Учебник. Создание приложения службы Windows – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/framework/windows-services/walkthrough-creating-a-windows-service-application-in-the-component-designer> – Дата доступа: 17.12.2023
2. Аудит безопасности – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/security/threat-protection/auditing/security-auditing-overview> – Дата доступа: 17.12.2023
3. Программная архитектура приложений служб – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/framework/windows-services/service-application-programming-architecture> – Дата доступа: 17.12.2023
4. Практическое руководство. Создание служб Windows – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/framework/windows-services/how-to-create-windows-services> – Дата доступа: 17.12.2023
5. Протоколирование событий – Режим доступа: <https://professorweb.ru/my/csharp/base_net/level1/1_3.php> – Дата доступа: 17.12.2023