УДК 004.42:378.14

Большаков А.О.
Студент магистратуры 2 курса
РТУ МИРЭА – Российский технологический университет,
ранее Московский институт радиотехники, электроники и автоматики
Россия, Москва

Леонтьев А.С.
Кандидат технических наук, старший научный сотрудник, доцент
РТУ МИРЭА – Российский технологический университет,
ранее Московский институт радиотехники, электроники и автоматики
Россия, Москва

АВТОМАТИЗАЦИЯ ПРОЦЕССОВ МОНИТОРИНГА И ИНВЕНТАРИЗАЦИИ ИНФОРМАЦИОННО-ТЕХНОЛОГИЧЕСКОЙ ИНФРАСТРУКТУРЫ, ПРИМЕНЯЕМОЙ В УЧЕБНОМ ПРОЦЕССЕ

Аннотация: Объектом исследования выступает система мониторинга и инвентаризации информационно-технологической инфраструктуры, применяемой в учебном процессе. Целью научно-исследовательской работы выступает анализ предметной области для дальнейшего проектирования системы мониторинга и инвентаризации информационно-технологической инфраструктуры, применяемой в учебном процессе.

Результаты проведенного исследования могут быть использованы в целях дальнейшего проектирования систем мониторинга и инвентаризации информационно-технологической инфраструктуры, применяемой в учебном процессе. Кроме того, результаты статьи могут использоваться другими разработчиками, целью которых является создание своих систем.

Ключевые слова: мониторинг учебных компьютеров, поддержка процессов администрирования, инвентаризация информационно-технологической инфраструктуры, базы данных, графический интерфейс.

Bolshakov A.O.
2nd year Master's degree student
MIREA - Russian Technological University, formerly known as the Moscow Institute of Radio Engineering, Electronics and Automation
Russia, Moscow

Leontyev A.C.
Candidate of Technical Sciences, Senior Researcher, Associate Professor
MIREA Russian Technological University (formerly Moscow Institute of Radio Engineering, Electronics, and Automation)
Russia, Moscow

AUTOMATION OF MONITORING AND INVENTORYING INFORMATION TECHNOLOGY INFRASTRUCTURE USED IN THE EDUCATIONAL PROCESS.

Abstract: The object of the research is the monitoring and inventory system of the information technology infrastructure used in the educational process. The aim of the research work is to analyze the subject area for the further design of the monitoring and inventory system of the information technology infrastructure used in the educational process.

The results of the conducted research can be utilized for the purpose of further designing monitoring and inventory systems for the information technology infrastructure used in the educational process. Additionally, the results of the article can be utilized by other developers whose goal is to create their own systems.

Keywords: monitoring of educational computers, support of administration processes, inventorying of information technology infrastructure, databases, graphical interface.

Введение

Одна из больших структур общества – это учебные заведения. Каждый день в каждой школе, институте и подобных организациях происходит формирование и передача больших объемов данных, зачастую, чтобы контролировать эти процессы, люди прибегают к использованию специальных утилит. К одним из часто используемых программных средств, которые используют крупные учебные заведения, особенно в последние годы, это СДО – системы дистанционного обеспечения. Зачастую институты и школы используют систему с открытым исходным кодом moodle, ввиду экономичности решения. Кроме того, это не единственный тип программного обеспечения, которые используют учебные заведения, одним из таких является объект исследования этой работы – система мониторинга и инвентаризации информационно-технологической инфраструктуры.

1. Выбор архитектурного решения

В первую очередь необходимо определится с архитектурой системы для того, чтобы подобрать необходимый стек технологий. Существуют несколько подходов для реализации подобных систем.

Для систем мониторинга и инвентаризации информационно-технологической инфраструктуры может быть применено множество архитектурных решений:

— Клиент-серверная архитектура - эта архитектура основывается на разделении приложения на две части: клиентскую и серверную. Клиентская часть обычно устанавливается на рабочих станциях пользователей и используется для сбора информации об инфраструктуре, в то время как серверная часть установлена на центральном сервере и используется для хранения и обработки собранной информации. Эта архитектура обеспечивает легкую масштабируемость, высокую производительность и удобную администрирование системы.

— Распределенная архитектура - в этой архитектуре приложение разделено на несколько компонентов, которые могут быть распределены по

различным серверам в сети. Компоненты могут взаимодействовать друг с другом через сетевые протоколы и API. Эта архитектура позволяет более эффективно использовать ресурсы и масштабировать приложение.

— Микросервисная архитектура - эта архитектура основывается на создании небольших сервисов, которые выполняют отдельные функции и могут взаимодействовать друг с другом. Эти сервисы могут быть разработаны и развернуты независимо друг от друга и могут быть масштабированы в зависимости от потребностей системы.

Какое архитектурное решение выбрать, зависит от масштаба системы, требований к производительности, безопасности. В нашем случаи если программа планируется использовать в институте на несколько групп, то, возможно, лучшим архитектурным решениям будет клиент-серверная архитектура.

2. Выбор способа реализации

При клиент-серверной архитектуре, как правило интерфейс пользователя представляет из-за себя веб-приложение или программное обеспечение, которое может быть установлено на компьютер (далее десктопное приложение). Подробнее рассмотрим эти варианты и выберем наилучшее решение для данной системы.

Веб-приложение позволяет пользователям работать с программой через браузер, что удобно и доступно для большинства пользователей. Оно может быть развернуто на центральном сервере, который будет доступен из любой точки сети института.

Веб-приложение также может иметь клиент-серверную архитектуру, где серверная часть будет отвечать за обработку запросов и хранение данных, а клиентская часть будет отображать данные и обеспечивать пользовательский интерфейс.

Таким образом, веб-приложение может быть хорошим архитектурным решением для программы, которая должна использоваться в институте на

несколько групп, поскольку это позволяет пользователям работать с программой из любой точки сети института, имеет простой и понятный пользовательский интерфейс и легко масштабируется.

Десктопные приложения могут обеспечивать более высокую производительность, чем веб-приложения, поскольку они работают непосредственно на компьютере пользователя и не зависят от скорости интернет-соединения или производительности сервера. Они также могут предоставлять более широкий набор функций, чем веб-приложения, и могут работать в автономном режиме, что удобно, когда доступ к интернету ограничен или нестабилен.

Однако, десктопные приложения могут быть более сложными в установке и обновлении, поскольку каждое устройство, на котором приложение должно быть установлено, должно иметь необходимое программное обеспечение и версию операционной системы.

Таким образом, выбор между веб-приложением и десктопным приложением зависит от требований и потребностей конкретного случая. Если необходима высокая производительность, широкий набор функций и возможность работы в автономном режиме, то десктопное приложение может быть предпочтительнее. Если же необходима доступность из любой точки сети, простой пользовательский интерфейс и возможность легкой масштабируемости, то веб-приложение может быть более подходящим решением.

Клиентская часть десктопного приложения может быть написана на одном из языков программирования, поддерживаемых операционной системой, таких как Java, C++, C# или Python. Она обеспечивает пользовательский интерфейс и обработку действий пользователя.

Серверная часть приложения может быть написана на любом языке программирования, который поддерживает необходимые функции, такие как хранение и обработка данных. Для работы с серверной частью могут использоваться различные технологии, например, сокеты, HTTP или RPC.

Для реализации данной системы наиболее подходящее решение – десктопное приложение. Потенциально из-за наличия большого количества пользователей необходима высокая производительность, но также преимущество веб-приложений такие как доступ из любой точки сети значительно выделит разрабатываемую систему перед её аналогами. Кроме того, некоторые функции, например, блокировка экрана пользователю, нельзя реализовать при интерфейсе веб-приложения. В таком случае, подойдем к решению задачи не стандартным способом, а именно реализуем три модуля приложения: сервер, интерфейс клиента и интерфейс администратора. Используем язык C#, который предлагает фреймворк для реализации легко масштабируемых веб-приложений APS.NET и при этом реализуем интерфейс десктопных приложений при помощи технологий на платформе .NET - Windows Forms (WinForms) или Windows Presentation Foundation (WPF).

3. Выбор способа реализации пользовательского интерфейса

При выборе между Windows Forms (WinForms) и Windows Presentation Foundation (WPF) для реализации системы мониторинга и инвентаризации информационно-технологической инфраструктуры в учебном процессе, необходимо учесть несколько факторов. Проведем сравнительный анализ технологий:

Простота и знакомство: Windows Forms является более простым и прямолинейным инструментом для создания пользовательского интерфейса по сравнению с WPF. Он предоставляет знакомую модель программирования, основанную на формах и элементах управления, которая может быть легко понята и использована разработчиками.

Производительность: Windows Forms имеет более низкие требования к аппаратному обеспечению и обладает лучшей производительностью в сравнении с WPF. Если системе требуется высокая отзывчивость и быстрая обработка данных, особенно при работе с большим объемом информации или

при использовании сложных алгоритмов, то Windows Forms может быть предпочтительным выбором.

Поддержка сторонних библиотек и интеграция: Windows Forms имеет более широкую поддержку сторонних библиотек и компонентов в сравнении с WPF. Если необходимо использовать конкретные инструменты или библиотеки, которые не полностью поддерживают WPF, то Windows Forms может быть более подходящим вариантом. Кроме того, Windows Forms лучше интегрируется с существующими Win32-приложениями и позволяет использовать существующий код.

Однако, несмотря на преимущества Windows Forms, следует отметить, что WPF предлагает более современный подход к созданию пользовательского интерфейса, обладает богатыми возможностями стилизации и анимации, а также более гибким и разделенным на уровни кодом. Если требуется более сложный и выразительный интерфейс, а также более гибкое управление представлением данных, то WPF может быть предпочтительным выбором.

Важно учитывать, что оба инструмента являются частями технологий .NET Framework и предлагают возможности для создания функциональных и эффективных пользовательских интерфейсов. Выбор между ними должен быть основан на конкретных потребностях проекта, опыте разработчиков и требованиях к производительности и функциональности системы.

В итоге, выбор между Windows Forms и WPF зависит от конкретных требований и ограничений проекта. В нашем случае важна простота, высокая производительность и поддержка сторонних библиотек, Windows Forms более подходящий выбор.

4. Формирование структурной и функциональной схемы

Такого рода сервисы, как правило, реализованы в виде десктопного приложения – наиболее подходящий способ. Преимущество такой реализации – сокрытие информации и удобство администрирования, предприятие имеет обособленную базу данных от других высших заведений, использующих

программный продукт. Для реализации обмена информации при помощи сети интернет необходимо разделить программу на несколько частей, одна для подчиненных компьютеров, другая для главного компьютера. Связи со спецификой разработки схема приложения будет выглядеть следующим образом (Рисунок 1).

Рисунок 1 – Структурная схема

Функциональная схема - отображает взаимодействие компонентов системы и информационных потоков, состав данных в потоках с указанием используемых устройств. Для формирования функциональной схемы необходимо использовать общепризнанный стандарт.

Функциональные схемы содержат больше информации о работе системы, чем структурные. Несмотря на это, блок-схемы также важны, при таком подходе тщательно прорабатываются спецификации межпрограммных интерфейсов, поскольку от качества их описания зависит количество наиболее затратных ошибок. Поскольку на этапе тестирования выявляются ошибки при комплексном тестировании, потребуется повторное тестирование уже отлаженных тестов.

Исходя из предыдущего пункта, функциональная схема будет состоять из трех частей: интерфейс управления, серверная часть, база данных.

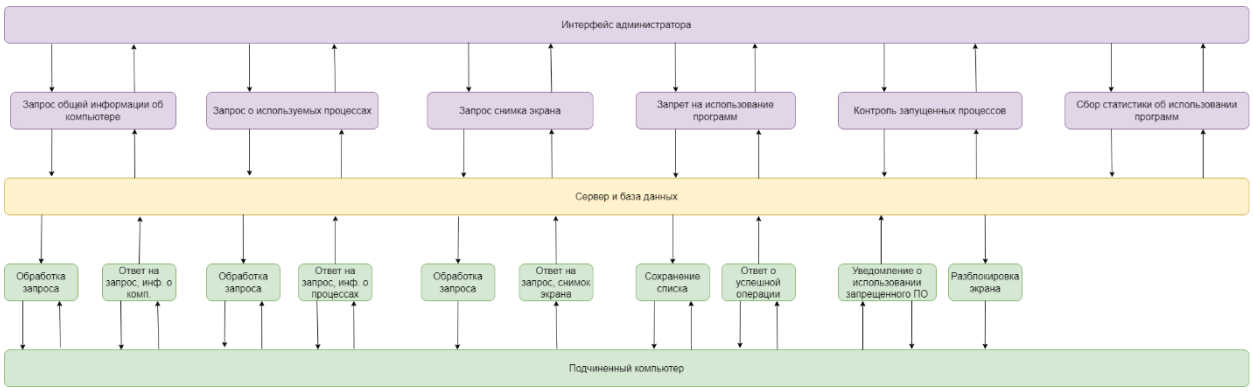

Рисунок 2 – Функциональная схема

Опишем функции, представленные в функциональной схеме:

— Обработка и разрешение на использование – авторизация

— пользователя, для определения какой машиной пользуется человек, пока не будет пройдена, пользоваться компьютером невозможно.

— Запрос общей информации об компьютере – отправляет на

— интерфейс управления информацию о операционной системе и комплектующих устройства, установленных программах.

— Запрос снимка экрана – отправляет на

— интерфейс управления информацию о текущем изображении экрана

— Запрет использования программ – создает список программ,

— запрещенных для использования на подчиненных компьютерах.

— Блокировка экрана – при выявлении использования запрещенной

— программы, блокирует возможность использования компьютера, до одобрения администратором.

Если говорить о работе системы в целом, её основой инструмент - это обмен данными через Hub и клиентов. Механизм, используемый в технологии SignalR в ASP.NET для реализации взаимодействия в режиме реального времени между сервером и клиентскими приложениями.

Взаимодействие начинается с установки соединения между клиентским приложением и сервером через SignalR. При установке соединения клиент

отправляет запрос серверу, и при успешной установке соединения сервер создает экземпляр Hub, который служит центральной точкой обмена данными.

Hub представляет собой класс, унаследованный от класса Hub в SignalR, и содержит методы, которые могут быть вызваны клиентом или сервером. Клиент может вызывать методы на Hub для отправки данных или выполнения определенных действий на сервере, а сервер может вызывать методы на Hub для отправки данных клиенту или уведомления клиента о событиях.

При обмене данными через Hub и клиентов используются два основных подхода:

Server-to-Client (S2C): Сервер вызывает методы на Hub для отправки данных клиенту. Это позволяет серверу активно уведомлять клиента о событиях или передавать ему обновленные данные. Например, сервер может вызывать метод на Hub для отправки уведомления о новом сообщении, изменении состояния или обновлении данных. Затем Hub отправляет это уведомление или данные клиенту, который может обработать их и отобразить на пользовательском интерфейсе в реальном времени.

Client-to-Server (C2S): Клиент вызывает методы на Hub для отправки данных или выполнения операций на сервере. Это позволяет клиенту взаимодействовать с сервером и отправлять обновленные данные или запросы на обработку. Например, клиент может вызывать метод на Hub для отправки сообщения на сервер или запроса на выполнение определенной операции. Затем Hub обрабатывает этот запрос и выполняет соответствующие действия или отправляет обратную связь клиенту.

Обмен данными через Hub и клиентов в технологии SignalR основан на подходе "публикация-подписка" (Publish-Subscribe). Клиенты подписываются на определенные события или методы Hub, и когда эти события происходят или методы вызываются, Hub автоматически отправляет данные или уведомления всем подписанным клиентам.

Таким образом, обмен данными через Hub и клиентов позволяет реализовать взаимодействие в режиме реального времени между сервером и

клиентами. Это обеспечивает возможность передачи данных и уведомлений в реальном времени без необходимости клиентским приложениям активно запрашивать информацию у сервера.

Когда клиент устанавливает соединение с сервером через SignalR, происходит регистрация клиента на Hub для получения уведомлений или данных. Клиенты могут подписываться на конкретные события или методы Hub, указывая, какие типы данных или события их интересуют.

Когда на сервере происходит событие или вызывается метод на Hub, SignalR автоматически оповещает всех подписанных клиентов. Данные или уведомления могут быть отправлены всем клиентам или только тем, которые подписались на соответствующие события.

При получении данных или уведомлений от сервера, клиентское приложение может обработать эти данные и отобразить их в пользовательском интерфейсе в реальном времени. Это позволяет создавать динамические и интерактивные приложения, которые мгновенно отображают изменения и обновления, даже без активных запросов клиента.

Обмен данными через Hub и клиентов в SignalR обеспечивает эффективное взаимодействие между сервером и клиентами, позволяя передавать данные и уведомления в режиме реального времени, что способствует более плавному и интерактивному пользовательскому опыту.

Список использованных источников

1. Abadie C., Lalande J., Tcherkez G. Exact mass GC- MS analysis: Protocol, database, advantages and application to plant metabolic profiling. – 2022. – Т. 45. – №. 10. – С. 3171-3183.
2. Yang W. et al. Pase: Postgresql ultra-high-dimensional approximate nearest neighbor search extension //Proceedings of the 2020 ACM SIGMOD international conference on management of data. – 2020. – С. 2241-2253.
3. Coulon F. et al. Modular and distributed IDE //Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering. – 2020. – С. 270-282.
4. Marticorena-Sánchez R. et al. UBUMonitor: an open-source desktop application for visual E-learning analysis with moodle //Electronics. – 2022. – Т. 11. – №. 6. – С. 954.
5. Pasztaleniec M., Skublewska-Paszkowska M. Comparative analysis of Windows Presentation Foundation and Windows Forms //Journal of Computer Sciences Institute. – 2020. – Т. 14. – С. 26-30.
6. DeBell T. et al. Opens hub: Real-time data logging, connecting field sensors to google sheets //Frontiers in Earth Science. – 2019. – Т. 7. – С. 137.
7. Bakonyi V. et al. SignalR based Real-Time applications in education //2023 3rd International Conference on Innovative Practices in Technology and Management (ICIPTM). – IEEE, 2023. – С. 1-6.