РЕФЕРАТ

РПЗ 30 с., 3 ч.,12 рис., 10 табл., 9 источников, 3 приложений

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР, МОНИТОРИНГ, СОСТОЯНИЕ, ГРАФИК, КЛИЕНТ, СЕРВЕР

Объектом разработки является система удаленного мониторинга состояния персонального компьютера (ПК).

Цель работы – проектирование и реализация программы, с помощью которой можно отслеживать и сохранять данные с сенсоров ПК, что позволяет оценивать энергопотребление системы в разные периоды времени, помогает в диагностике проблем с устройством, а так-же позволяет удаленно отслеживать эти показатели.

В результате разработки было спроектировано клиентские, серверное и веб-приложение, которое обеспечивает отображение данных, включающую нагрузку ядер процессора, температуру компонентов и загруженность памяти ПК, а так-же хранение этих данных.

Пользователями системы могут быть системные администраторы офисов, а так-же пользователи, нагружающие ПК сложными и длительными задачами (графика, рендер, обучение нейросетевых моделей).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В условиях интенсивного развития информационных технологий и усложнения аппаратного обеспечения критически важным становится вопрос поддержания стабильности и производительности вычислительных систем. Постоянный рост вычислительных мощностей, необходимых для рендеринга графики, обучения нейросетевых моделей и выполнения сложных инженерных расчетов, приводит к повышению нагрузки на компоненты ПК, увеличению энергопотребления и рискам перегрева оборудования.

Традиционные локальные средства мониторинга часто не обеспечивают возможности длительного сбора статистики и удаленного контроля состояния устройств. Отсутствие накопленных данных о работе системы в разные периоды времени затрудняет глубокую диагностику аппаратных проблем и оптимизацию энергозатрат.

Целесообразно спроектировать и разработать систему удаленного мониторинга состояния персонального компьютера, которая позволяет в режиме реального времени отслеживать нагрузку на ядра процессора, температуру компонентов и использование оперативной памяти. Реализация системы в виде комплекса из клиентского, серверного и веб-приложения обеспечит пользователям возможность оперативного доступа к диагностическим данным и истории работы системы из любой точки сети.

1 Анализ требований и уточнение спецификаций

1.1 Обоснование подхода к проектированию и разработке

При проектировании и разработке системы был выбран объектно-ориентированный подход [1], в основе которого лежит методология объектно-ориентированного программирования (ООП).

Структурный подход к проектированию систем ориентирован на разделение программы на отдельные модули и компоненты, которые обрабатывают данные и выполняют функции. В этом методе основное внимание уделяется алгоритмам и процессам, а не отдельным объектам.

Объектно-ориентированный подход основывается на концепции объектов, которые инкапсулируют как данные, так и поведение. Системы, построенные с использованием объектно-ориентированного подхода, легче расширять и модифицировать, поскольку изменение одного объекта не обязательно затрагивает остальные. Это приводит к более высокому качеству кода и его большей устойчивости во времени. Важными принципами ООП являются инкапсуляция, наследование и полиморфизм, что способствует повышению гибкости, масштабируемости и повторному использованию кода. Несмотря на то, что структурный подход обладает своими преимуществами, ООП является более подходящим для нашей системы. ООП основан на концепции объектов и классов, что способствует лучшей структурированности кода и упрощает его понимание.

В связи с возможными изменениями, требуемыми для дальнейшего улучшения пользовательского опыта, объектно-ориентированный подход к разработке программного обеспечения выделяется как оптимальное решение.

1.2 Выбор модели жизненного цикла программы

Были рассмотрены следующие модели жизненного цикла программного обеспечения: каскадная, V-образная, спиральная [2].

Каскадная модель представляет собой линейный и последовательный подход к разработке программного обеспечения. Эта модель проста в реализации и хорошо подходит для проектов с четко определенными

требованиями. Однако ее основные недостатки включают высокие риски возникновения проблем, если на начальных этапах были допущены ошибки. Изменения на поздних этапах могут вызвать значительные затраты и задержки.

V-образная модель расширяет каскадную, добавляя этапы верификации и валидации, которые соответствуют каждому этапу разработки. Она подчеркивает важность тестирования и проверки на каждом уровне. Хотя V-образная модель улучшает управление качеством за счет интеграции тестирования, она все равно сохраняет некоторые риски каскадного подхода, связанные с жестким порядком выполнения этапов и сложностью внесения изменений.

Спиральная модель – это гибкая и адаптивная модель, которая сочетает в себе элементы каскадной и V-образной моделей. Этот подход включает в себя повторные циклы разработки, тестирования и оценки, что позволяет быстро реагировать на изменения и потребности пользователя.

Для разработки системы, которая представляет собой развивающуюся систему с поддержкой множества устройств, лучше всего подойдет спиральная модель жизненного цикла программного обеспечения.

1.3 Выбор языка и среды программирования

При принятии решения о языке программирования для нашей системы, важно рассмотреть различные аспекты, включая анализ и визуализацию данных, производительность, расширяемость и поддержку языка. Давайте рассмотрим сравнительные характеристики следующих ЯП:

– golang – компилируемый язык с открытым исходным кодом, ориентированный на высокую производительность и удобство параллельных операций;

– php – интерпретируемый язык, широко используемый для веб-разработки, но имеет мало библиотек для анализа и визуализации данных;

– ruby – интерпретируемый язык, с популярным фреймворком Ruby on Rails для веб-разработки, который имеет богатые библиотеки для анализа и визуализации данных, такие как Ruby-Chart и Ruby-Graph;

– C# [3] – компилируемый язык программирования с упором на объектно-ориентированность, обеспечивающий высокую производительность и надежность приложений, имеет мощный фреймворк .NET[4] для создания распределенный систем. Помимо этого .NET включает среду EF Core[5] для работы с базами данных, а возможность интеграции Vue.js[6] позволяет создавать веб интерфейсы для отображения данных.

C# в сочетании с фреймворком .NET предпочтителен для разработки системы мониторинга благодаря балансу между скоростью разработки и производительностью. В отличие от интерпретируемых языков, таких как ruby или PHP, C# предоставляет более строгий контроль над ресурсами и типизацией, а использование развивающихся open-source библиотек, таких как LibreHardwareMonitorLib[7], позволяет расширять функционал системы под современные аппаратные решения без необходимости низкоуровневого программирования драйверов.

1.4 Объектная декомпозиция предметной области

Для начала разработки надо выполнить объектную декомпозицию предметной области, которая лежит в основе объектного подхода. Это поможет нам представить разрабатываемую программу в виде совокупности объектов и функций, выполняющихся посредством передачи сообщений между ними.

После изучения доступных сенсоров на нескольких ПК были выявлены интересующие наборы данных. К показателям нагрузки были отнесены следующие параметры: нагрузка ядер процессора и общая нагрузка процессора. В температурные данные вошли показатели температуры на разных участках материнских плат: оперативная память, процессор, цепь

питания и разъемы (psi-e и M.2). Показатели памяти включают занятость жестких дисков, SSD и оперативной памяти.

В результате анализа технического задания и возможных показателей, были выделены основные объекты и определены их взаимодействия друг с другом. Объекты и сообщения между ними представлены на рисунке 1.

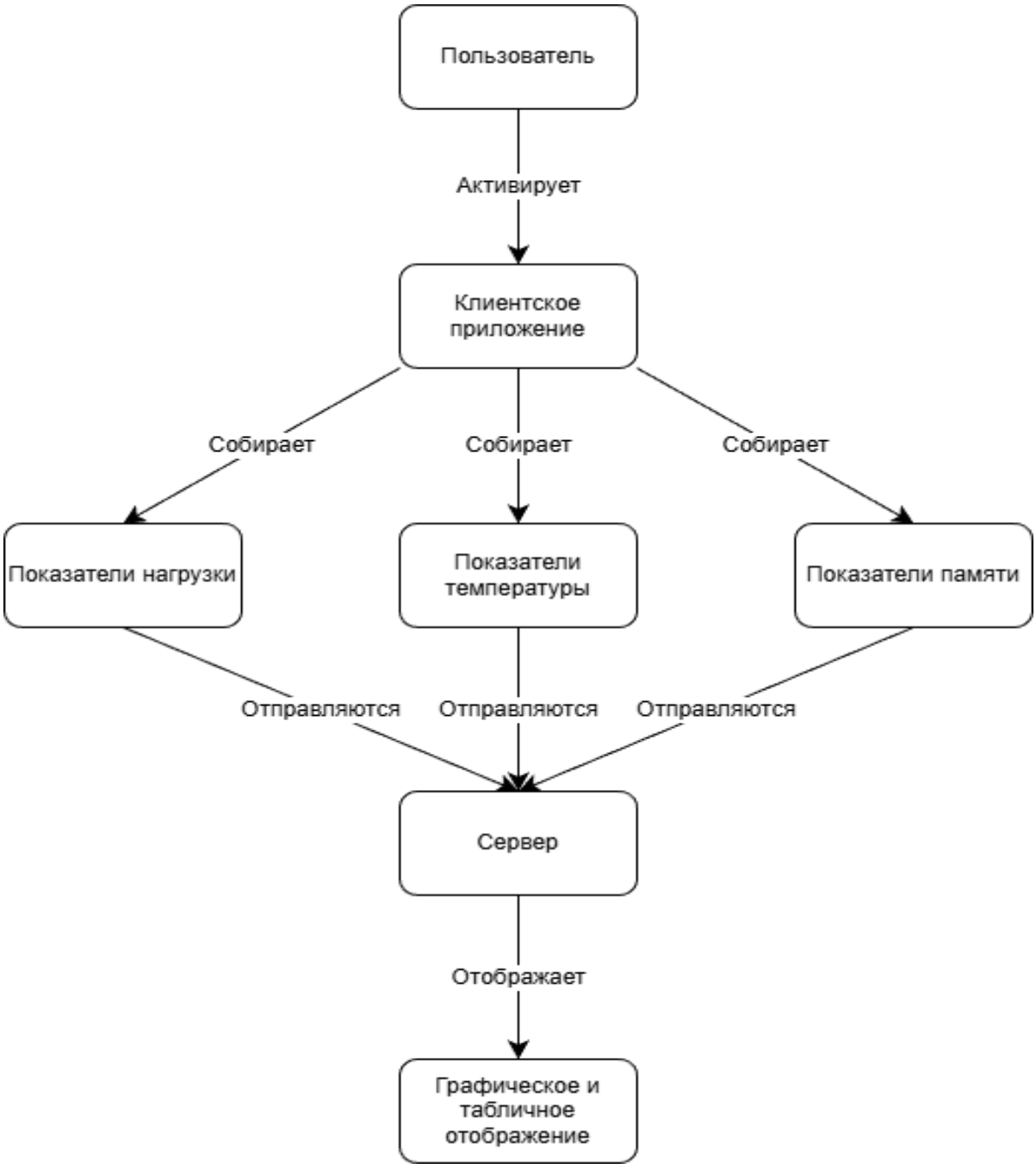

Рисунок 1 – Объектная декомпозиция СУМСПК

В результате объектной декомпозиции были выделены следующие объекты: пользователь, клиент, показатели нагрузки, показатели температуры, показатели памяти, сервер, графическое и табличное изображение.

Система работает таким образом, что пользователь запускает клиент, затем данные с клиента отправляются на сервер. Сервер обрабатывает получает данные и выводит их на веб сервис.

1.5 Определение вариантов использования

После объектной декомпозиции нужно определить пользователей системы. Для этого была разработана диаграмма вариантов использования [8]. В результате разработки диаграммы выяснилось, что с программой осуществлять взаимодействие будет только 1 группа – пользователь.

Выделены следующие варианты взаимодействия: посмотреть данные мониторинга удаленно, посмотреть данные мониторинга с ПК, посмотреть архивные данные.

В таблицах 1-6 представлены описания вышеперечисленных вариантов использования.

Таблица 1 – Описание варианта «Посмотреть данные мониторинга удаленно»

Название варианта	Посмотреть данные мониторинга удаленно
Цель	Получить и посмотреть данные мониторинга не находясь за ПК
Действующие лица	Пользователь
Краткое описание	Посмотреть данные онлайн
Тип варианта	Основной

Таблица 2 – Вариант использования «Посмотреть данные мониторинга удаленно»

Действия пользователя	Отклик системы
1. Пользователь запускает клиенское приложение заранее	
	2. Сервер получает данные с ПК, обрабатывает их и визуализирует.
	3. Система переводит пользователя на страницу с графиком или таблицей.
4. Пользователь выбирает устройство, или конкретную деталь.	
	5. Система меняет график или таблицу на другой.

Таблица 3 – Описание вариант «Посмотреть данные мониторинга с ПК»

Название варианта	Посмотреть данные мониторинга с ПК
Цель	Получить и посмотреть самые актуальные данные с ПК
Действующие лица	Пользователь
Краткое описание	Посмотреть тончые данные с ПК
Тип варианта	Дополнительный

Таблица 4 – Вариант использования «Посмотреть данные мониторинга с ПК»

Действия пользователя	Отклик системы
1. Запустить программу	
	2. Клиент собирает данные с ПК
	3. Клиент выводит данные
	4. Система меняет данные при их изменении.

Таблица 5 – Описание вариант «Посмотреть архивные данные»

Название варианта	Посмотреть архивные данные
Цель	Получить и посмотреть архивные данные мониторинга
Действующие лица	Пользователь
Краткое описание	Посмотреть визуализированные данные за определенный период
Тип варианта	Основной

Таблица 6 – Вариант использования «Посмотреть архивные данные»

Действия пользователя	Отклик системы
1. Нажатие кнопки «Архивные данные».	
	2. Система получает данные из базы данных, обрабатывает их и визуализирует.
	3. Система переводит пользователя на страницу с графиками.
4. Пользователь выбирает устройство и время.	
	5. Система меняет график на другой из карусели графиков и редактирует выбирает данные в зависимости от времени.

В результате анализа вышеописанных вариантов использования программы была разработана диаграмма вариантов использования. Пользователь может выбрать один из типов данных, на основе которых система построит графики и покажет пользователю, переведя его на новую страницу. Кроме того, диаграмма вариантов использования будет использоваться для оценки эффективности программы и выявления потенциальных проблем, что позволит принимать обоснованные решения о дальнейшем развитии и совершенствовании системы. Разработанная диаграмма представлена на рисунке 2.

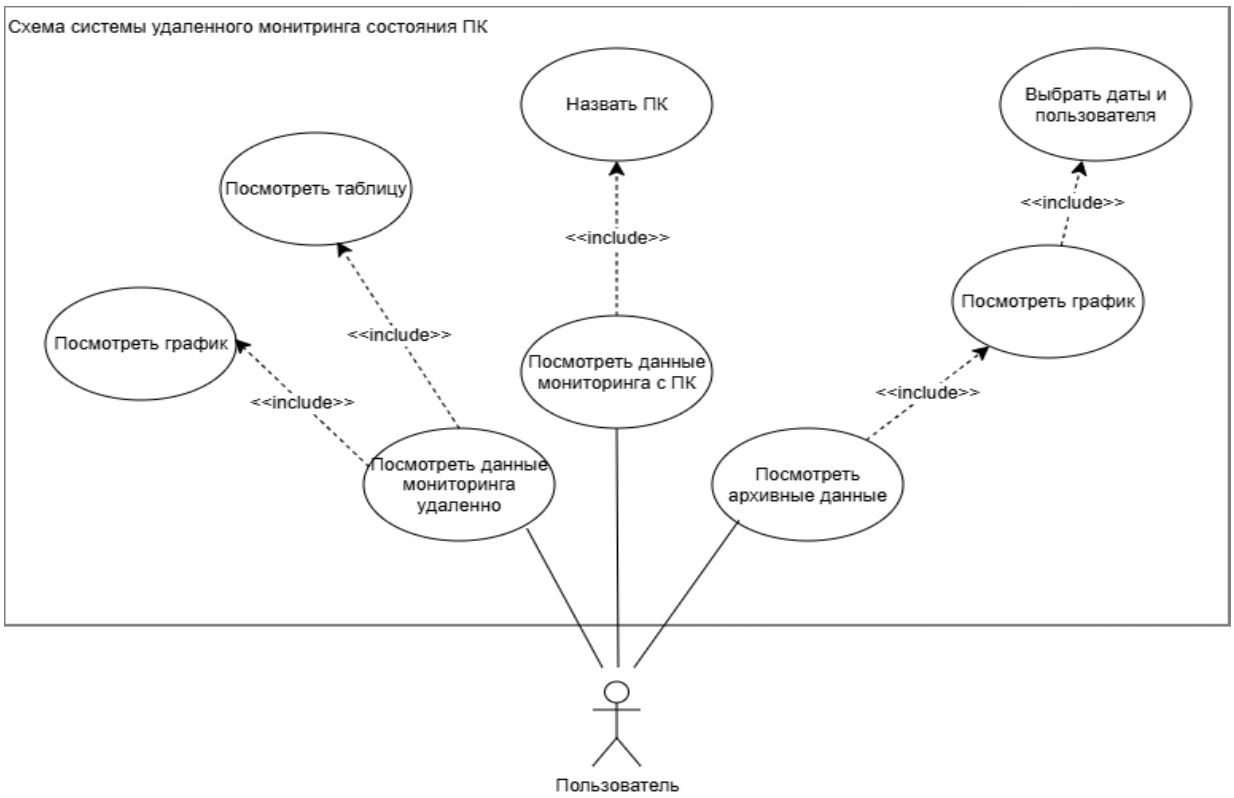

Рисунок 2 – Диаграмма вариантов использования

Таким образом, в диаграмме вариантов использования были определены функции программы, которые может использовать пользователь.

1.6 Разработка концептуальной модели предметной области

На основе выявленных объектов и вариантов использования программы, необходимо проанализировать взаимосвязи между ними, определить обмен данными и сообщениями, необходимые для удовлетворения функциональных требований и спецификаций программы, а также обеспечения простоты реализации вышеописанных функций.

Понятия, характерные для предметной области, соответствуют объектам, выделенным на этапе объектной декомпозиции программы:

- пользователь;
- клиент;
- сервер;
- актуальные данные мониторинга;
- данные о загрузке процессоров;
- данные о температуре;

– данные о загрузке дисков;

– таблица по данным мониторинга;

– графики по данным мониторинга.

Концептуальная модель предметной области представлена на рисунке 3.

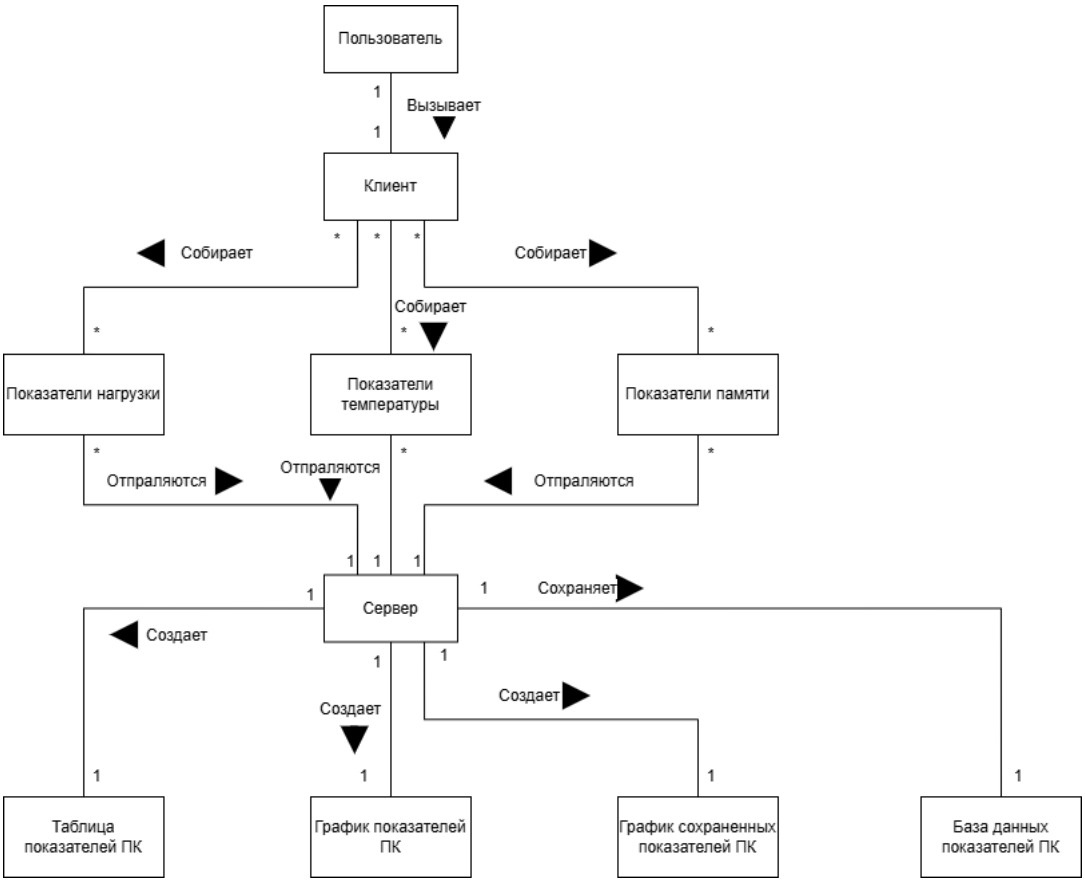

Рисунок 3 – Концептуальная модель предметной области

Разработанная концептуальная модель иллюстрирует связи между объектами и обмен сообщениями между ними. Она определяет смысловую структуру предметной области, а также причинно-следственные связи между объектами.

1.7 Разработка даталогической модели базы данных

Для обеспечения нормального функционирования нашей системы необходимо наличие базы данных. В данной системе используется полноценная базы данных SQLite [9]. Эта база данных сейчас активно используются в собирает данные с нескольких ПК. В этом контексте

рассмотрим подробности базы данных hardware_monitor, ответственной за информацию о ПК пользователей, их комплектующих и данные мониторинга.

Основные сущности базы данных:

– Computers – таблица с информацией о каждом компьютере пользователя, где записаны его данные;

– Systems – таблица с данными о комплектующих ПК;

– Sensors – таблица с показателями сенсоров;

– _EFMigrationsHistory – таблица миграций базы данных (нужна для контроля версий).

Даталогическая модель базы данных представлена на рисунке 4.

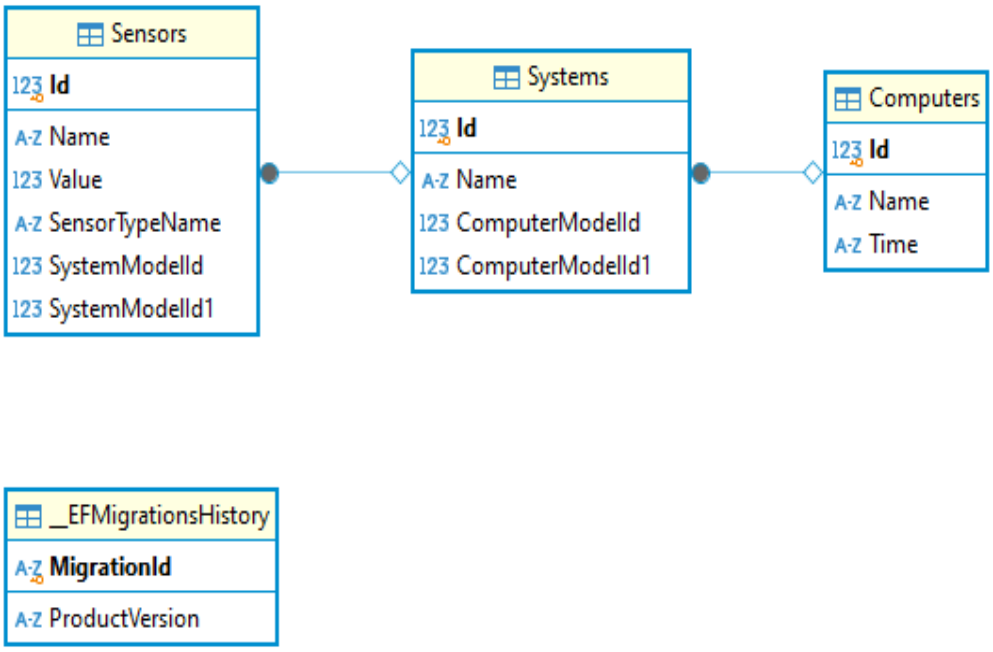

Рисунок 4 – Даталогическая модель базы данных mgerm

2 Проектирование структуры и компонентов программного продукта

2.1 Разработка интерфейса пользователя

Проектирование системы мониторинга состояния ПК началось с разработки интерфейса. В данном контексте выбор табличной формы диалога является наиболее уместным.

Табличная форма обеспечивает структурированное представление данных, которое может быть интуитивно понятным для пользователей. В данной системе, где пользователь сам выбирает, на какую страницу перейти и как отображать данные, таблица с кнопками или ссылками может быть эффективным способом представления этих выборов.

Также, использование табличной формы диалога в данной системе может снизить вероятность ошибок и упростить процесс принятия решений пользователем, так как ясно представляет доступные варианты выбора и позволяет провести сравнение между ними на основе представленных данных.

Построим граф состояний интерфейса для визуализации потока действий пользователя и наглядного представления возможных состояний, к которым может привести его взаимодействие с системой. Для построения графа состояний интерфейса СУМСПК выделим сначала основные состояния:

– главная страница – состояние интерфейса сразу после запуска программы, когда пользователь еще не нажал на кнопки для перехода на новую страницу;

– страница выбора таблиц актуальных данных – состояние интерфейса, когда пользователь нажал кнопку «Актуальные данные мониторинга»;

– страница отображения таблицы актуальных данных – состояние интерфейса, когда пользователь выбрал интересующее время и клиентский ПК;

– страница выбора графиков состояния системы – состояние интерфейса, когда пользователь нажал кнопку «Графики состояния системы»;

– страница отображения графиков актуальных данных – состояние интерфейса, когда пользователь выбрал интересующее время и клиентский ПК;

– страница выбора графиков архивных данных – состояние интерфейса, когда пользователь нажал кнопку «Архивные данные»;

– страница отображения графиков архивных данных – состояние интерфейса, когда пользователь выбрал интересующее время и клиентский ПК.

После определения состояний интерфейса необходимо установить связи между ними. Состояние "Главная страница" дает пользователю возможность выбрать одну из трех страниц. На каждой странице пользователь может выбрать интересующий клиентский ПК по имени, а так-же временной интервал, за который нужно показать данные. После этого отображается график или таблица исходя из выбранных фильтров. При отображении графика пользователь так-же выбирает и то, какой конкретно компонент желает просматривать. В случае просмотра таблицы — в ней выводятся данные всех комплектующих и их сенсоров. Кроме того, с любой страницы пользователь может вернуться на главную страницу. Построенный граф состояний интерфейса представлен на рисунке 5.

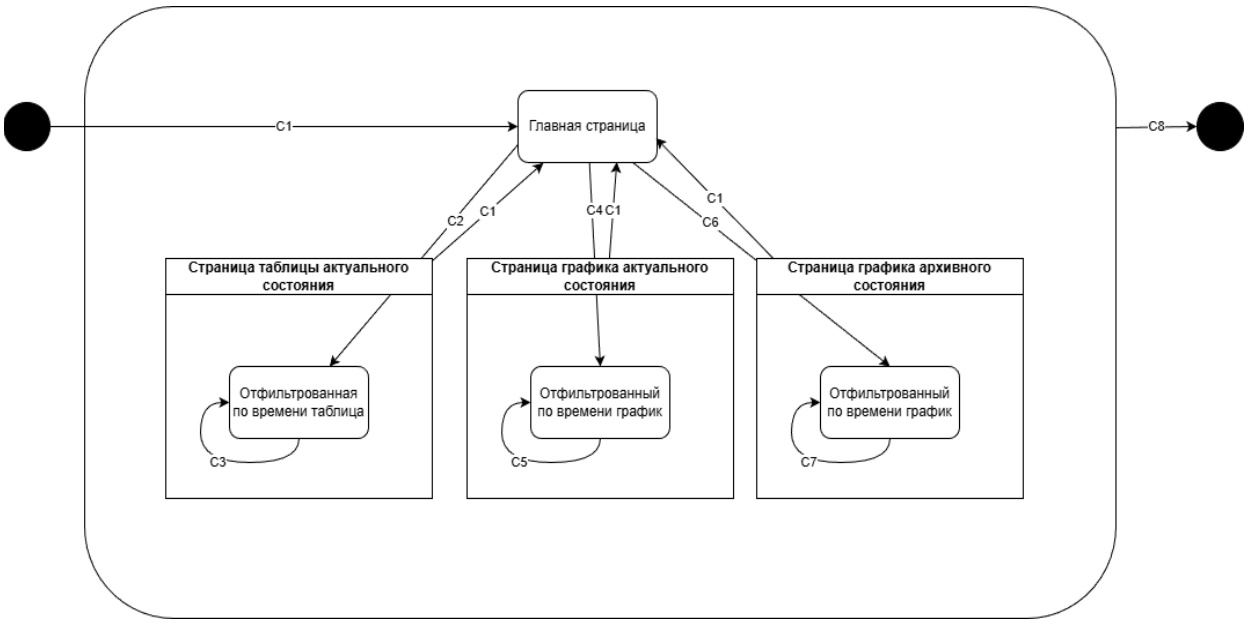

Рисунок 5 – Граф состояний интерфейса

На графе состояний интерфейса изображены следующие события:

– C1 – переход на главную страницу;

– C2 – переход на страницу табличного отображения актуального состояния при нажатии соответствующей кнопки;

– C3 – применение новых фильтров при нажатии на кнопку;

– C4 – переход на страницу графического отображения актуального состояния при нажатии соответствующей кнопки;

– C5 – применение новых фильтров при нажатии на кнопку;

– C6 – переход на страницу графического отображения архивных данных при нажатии соответствующей кнопки;

– C7 – применение новых фильтров при нажатии на кнопку;

– C8 – завершение работы программы.

2.2 Разработка форм интерфейса

На основе графа состояний были разработаны формы интерфейса, обеспечивающие взаимодействие пользователя с программой для просмотра данных.

Так как формы с данными имеют разный функционал и логику — была выбрана главная страница с ссылками на другие, она представлена на рисунке 6.

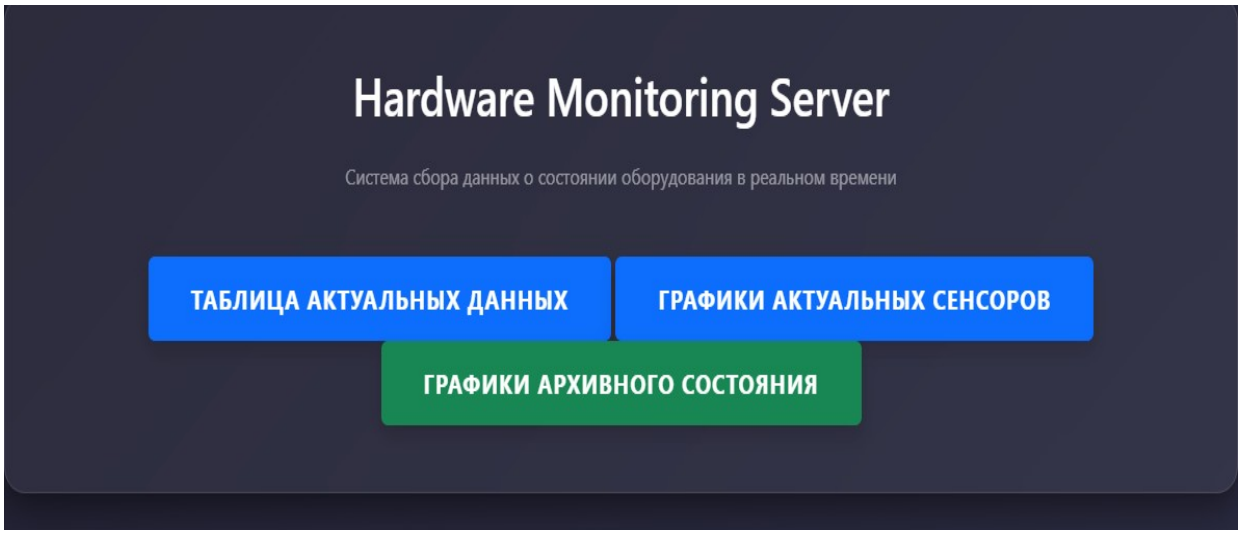

Рисунок 6 – Главная страница

При переходе на страницу «Актуальные данные мониторинга» пользователь обнаружит кнопку загрузки данных и выбор периода даты и времени. При нажатии на эту кнопку — появится таблица с данными. Если не выбирать дату — будут отображены все актуальные данные, хранимые на сервере. Данная таблица представлена на рисунке 7.

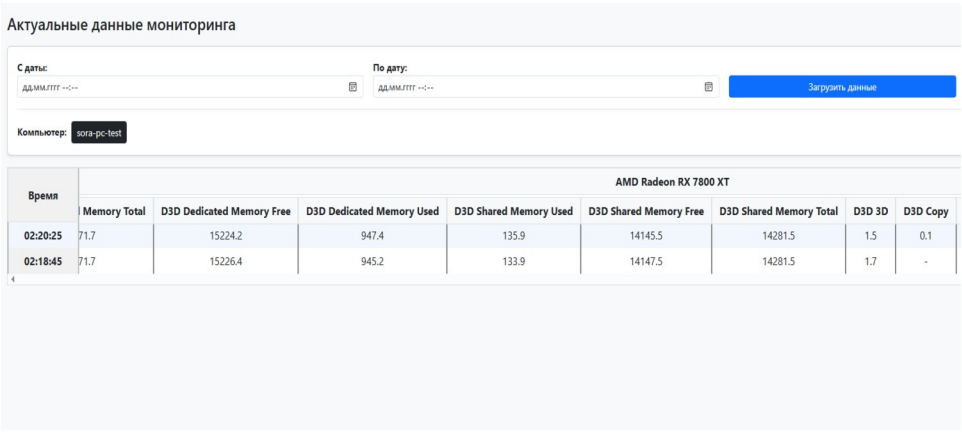

Рисунок 7 – Актуальные данные мониторинга

На этой и всех остальных страницах — присутствует кнопка перехода на главную страницу, при нажатии на которую пользователь возвращается на главную страницу. На рисунке 8 показана эта кнопка.

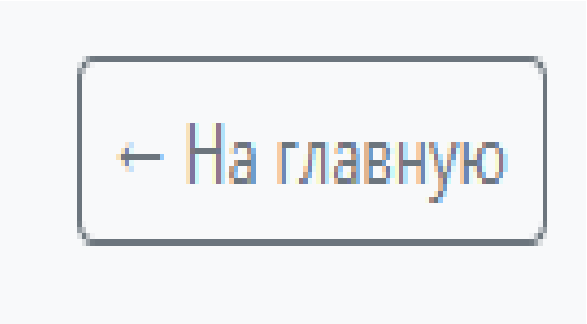

Рисунок 8 – Кнопка «На главную»

На странице графиков актуальных данных — при нажатии на кнопку обновить данные пользователю дают выбрать компонент, который он хочет отследить на графике. При ее выборе — создается график, он показан на рисунке 9.

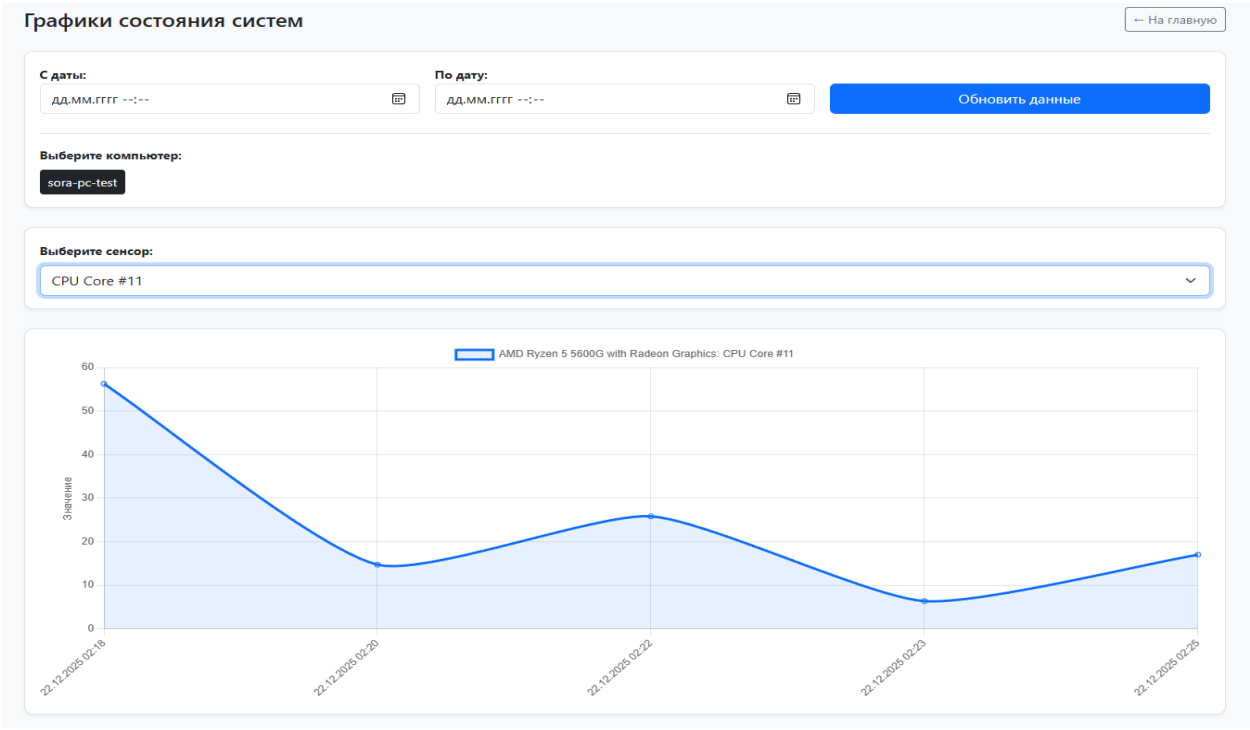

Рисунок 9 – График состояния системы

Для просмотра усредненых данных за все время отслеживания — была создана страница «Архивные данные». На рисунке 10 можно увидеть график, построенный по этим данным.

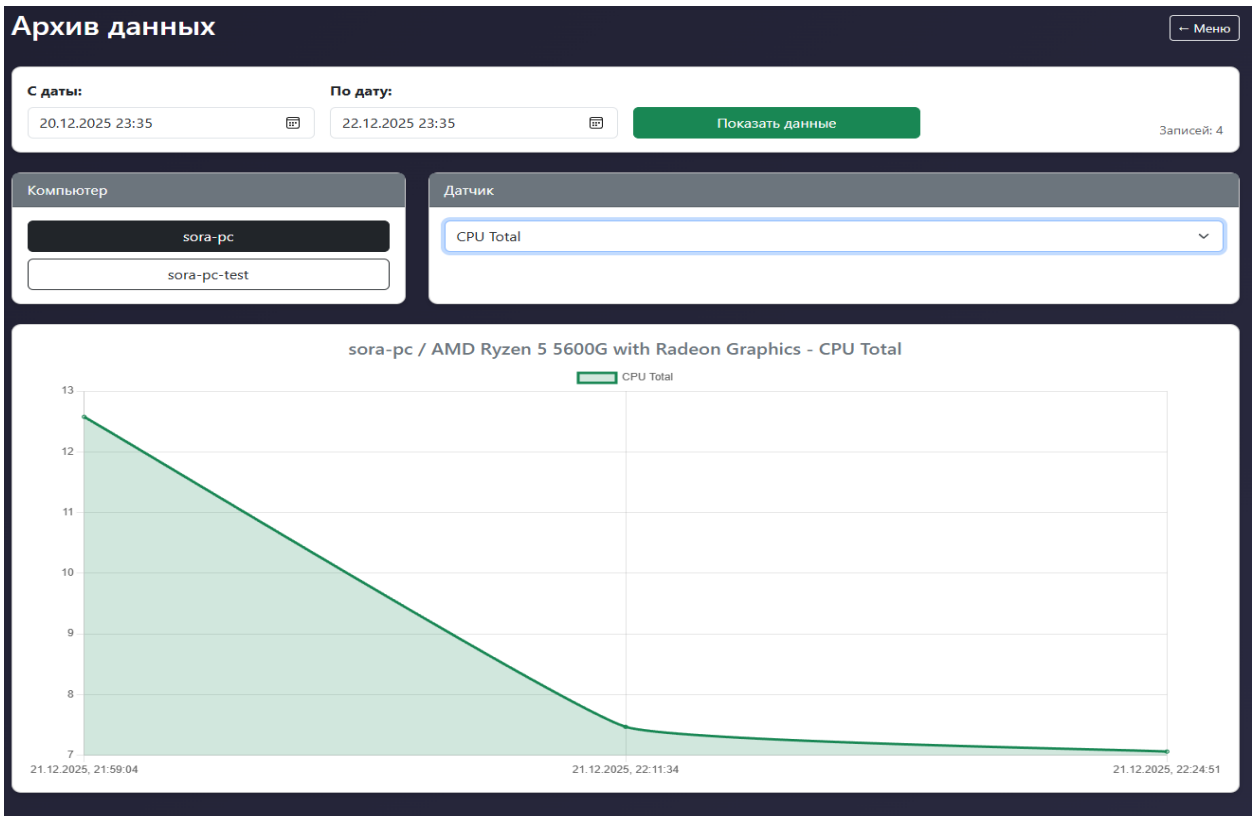

Рисунок 10 – Форма с отображенной ошибкой

2.3 Разработка классов предметной области и функции отправки данных

Сначала было необходимо изучить предметную область и в результате были выделены ключевые сущности. В контексте предметной области нужна сущность для хранения данных о ПК пользователей, компонентах ПК и о сенсохар, относящихся к компонентам. Так как у каждого ПК может быть несколько компонентов, и каждый компонент может иметь несколько сенсоров, эти сущности должны быть связаны. После выделения ключевых сущностей можно приступить к разработке структуры классов (моделей) предметной области, которые должны полностью отражать выделенные ранее сущности и их связи, что подробно описано далее.

– ComputerModel – модель для ПК пользователя, включающая имя, список компонентов и время получения данных, эта модель сопоставляет пользователя и список компонентов;

– SystemModel – модель для системного компонента, включающая имя и список сенсоров, эта модель нужна для группировки сенсоров, относящихся к одному компоненту;

– SensorModel – модел для сенсора, включающая его имя, тип собираемых данных сенсором и значение, которое он получает, эта модель нужна для хранения и сбора данных.

Такие модели позволят сгруппировать данные для удобного хранения, фильтрации и отображения в пользовательском интерфейсе.

Исходя из этого была разработана диаграмма классов предметной области, показанная на рисунке 11.

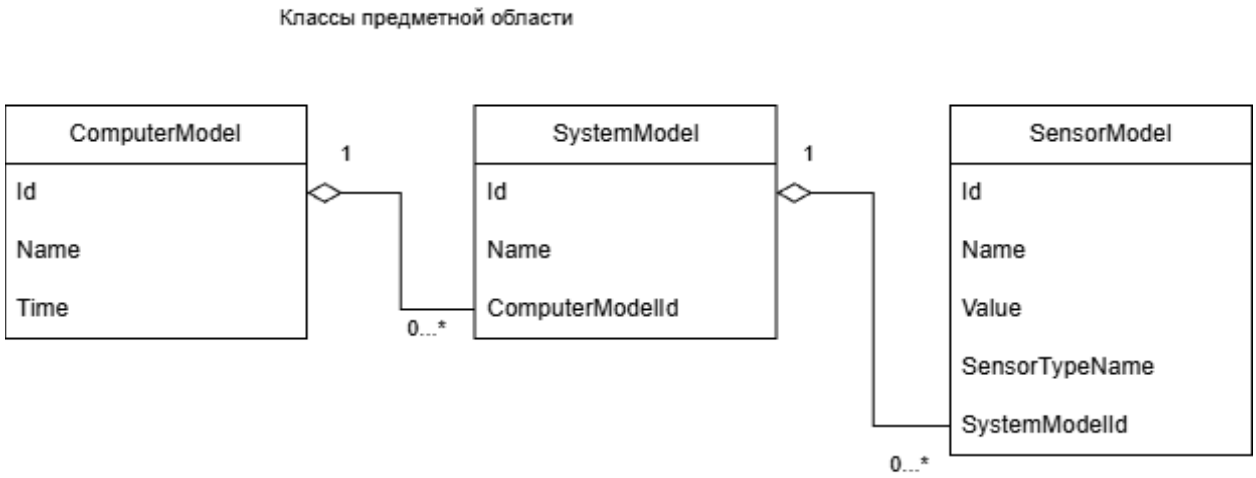

Рисунок 11 – Диаграмма классов предметной области программного обеспечения

Для успешной работы классов предметной области важно было обеспечить корректный сбор данных с компьютера и отправку их. Таким образом для сбора и передачи информации были созданны интерфейсы и классы, реализующие этот функционал:

– IHardwareDisplay – интерфейс с функциями отображения и отправки данных;

– HardwareDisplay — класс, реализующий интерфейс IHardwareDisplay, используя интерфейс IHardwareManager;

– IHardwareManager — интерфейс для отображения и обновления данных, а так-же закрытие доступа к сенсорам;

– IDataCreator – интерфейс, расширяющий IHardwareManager и добавляющий функцию создания коллекции системных компонентов;

– HardwareManager – класс, реализующий интерфейс IHardwareManager;

– DataCreator – класс, реализующий интерфейс IDataCreator;

– DataSender – класс, реализующий функцию отправки данных на сервер через http запрос.

Исходя из этого была разработана диаграмма классов клиентской части программного обеспечения, показанная на рисунке 12.

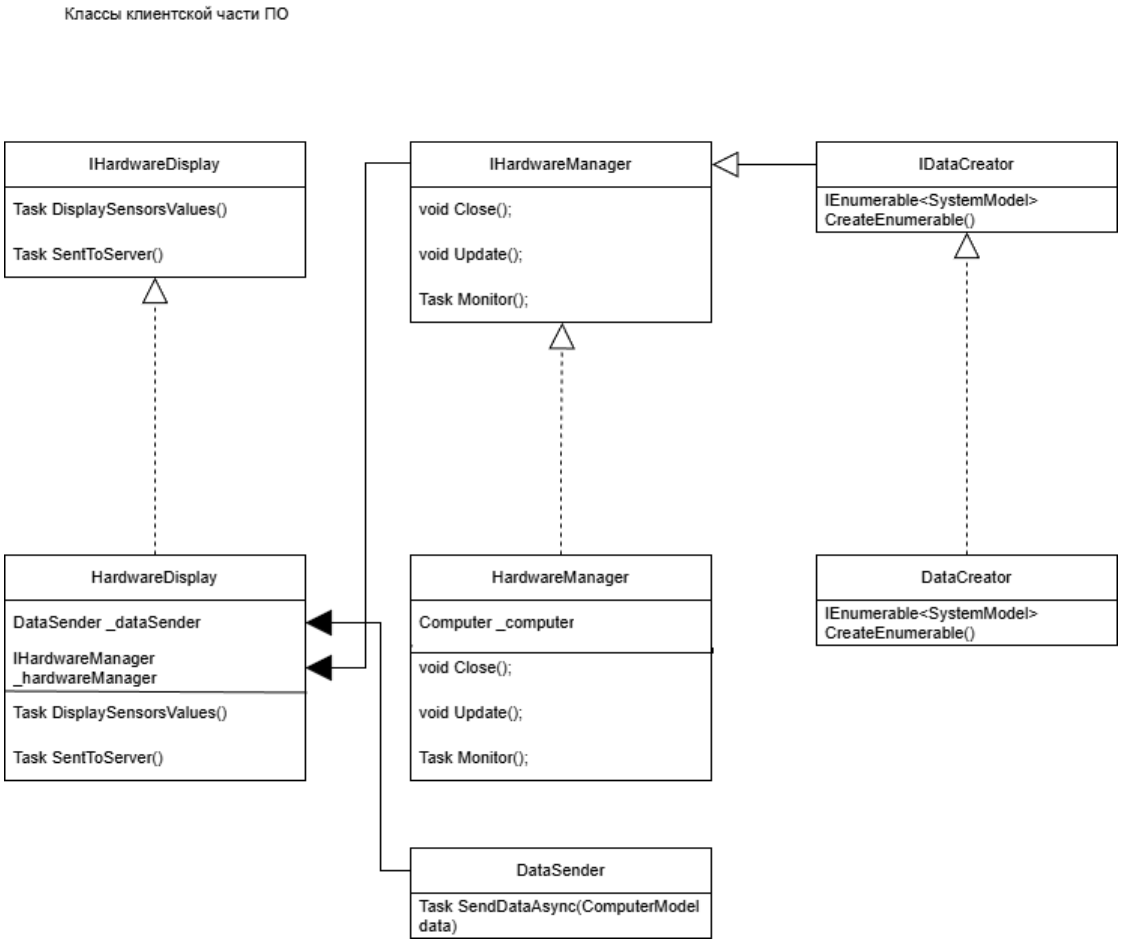

Рисунок 12 – Диаграмма классов клиентской части программного обеспечения

3 Выбор стратегии тестирования и разработка тестов

3.1 Функциональное тестирование

Для формирования наборов тестов был привлечен функциональный подход. В таком подходе программа рассматривается как "черный ящик", где основной упор делается на обнаружение расхождений между поведением программы и ее спецификацией.

Выбор функционального подхода обусловлен возможностью сконцентрироваться на высокоуровневой функциональности программы, обходя подробности внутренней реализации. Главная задача состоит в обеспечении корректной работы программы в контексте пользователя, что наилучшим образом отражается функциональными тестами.

На странице Архив данных присутствуют два поля ввода: с даты и по дату, поэтому будем использовать метод эквивалентного разбиения. Определим классы и подклассы для параметров «С даты» и «По дату».

Неправильные классы для параметра «С даты»:

– некорректный формат даты – любой формат, отличный от ДД.ММ.ГГГГ ЧЧ:мм;

– несуществующая дата – нереальные даты;

– дата позже или равна дате окончания – дата начала совпадает с датой окончания или позже нее;

– месяц вне диапазона 1-12 – значение месяца менее 1 или более 12.

Правильные классы:

– корректный формат даты – дата в формате ДД.ММ.ГГГГ ЧЧ:мм;

– существующая дата – реальная дата учитывая количество дней в месяцах;

– дата начала раньше даты окончания – дата начала предшествует дате окончания;

– месяц в диапазоне 1-12 – значение месяца не менее 1 или не более 12.

Для параметра «По дату» классы будут идентичны.

Тестирование методом эквивалентного разбиения представлено в таблице 7.

Таблица 7 – Тестирование методом эквивалентного разбиения

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат	Реакция программы	Вывод
1	Проверка корректности формата даты ДД.ММ.ГГГГ ЧЧ:мм	Дата в формате ГГГГ.ММ.ДД мм:ЧЧ	Невозможно ввести дату в таком формате	Невозможно ввести дату в таком формате	Программа работает корректно
2	Проверка, существует ли дата	Дата начала = 33.11.3000	Вывод сообщения об ошибке	Вывод сообщения об ошибке	Программа работает корректно
3	Проверка логики дат	Дата начала = 20.01.2021 Дата окончания = 10.01.2019	Вывод сообщения об ошибке	Вывод сообщения об отсутствии данных в БД	Программа работает некорректно
4	Проверка диапазона месяцев	Дата начала = 22.13.2022	Автоматическое изменение поля вне диапазона	Автоматическое изменение поля вне диапазона	Программа работает корректно

Метод эквивалентного разбиения позволил нам выявить ошибку, связанную с вводом неправильных данных.

Рассмотрим метод причинно-следственных связей. Основной целью применения метода причинно-следственных связей является выявление взаимосвязей между действиями пользователей и результатами их взаимодействия с системой. На главное странице у пользователя есть три

кнопки, каждая из которых переносит его на соответствующую страницу при нажатии. Это создает три возможных сценария взаимодействия пользователя с системой, каждый из которых может привести к различным результатам. В таблице 8 представлен метод причинно-следственных связей.

Таблица 8 – Метод причинно-следственных связей

Номер теста	Назначение теста	Значения исходных данных	Ожидаемый результат	Реакция программы	Вывод
1	Проверка корректности работы кнопки «Таблица актуальных данных»	Нажатие на кнопку «Таблица актуальных данных»	Ожидается открытие страницы таблиц актуальных данных	Открытие страницы таблиц актуальных данных	Программа работает верно
2	Проверка корректности работы кнопки «Графики актуальных сенсоров»	Нажатие на кнопку «Графики актуальных сенсоров»	Ожидается открытие страницы графиков актуальных сенсоров	Открытие страницы графиков актуальных сенсоров	Программа работает верно
3	Проверка корректности работы кнопки «Графики архивного состояния»	Нажатие на кнопку «Графики архивного состояния»	Ожидается открытие страницы графиков архивного состояния	Открытие страницы графиков архивного состояния	Программа работает верно

Данный метод показывает, что программа работает правильно.

3.2 Тестирование структурным контролем

Одним из видов тестирования был выбран структурный метод, так как он основан на продолжительном опыте работы программистов. Он подходит

для тестирования клиента, так как фокусируется на внутренней структуре кода и логике работы приложения, а также выявляет возможные ошибки и уязвимости на уровне логики работы приложения. Разработчики и тестировщики на протяжении долгого времени пополняли перечень вопросов, которые могут выявлять ошибки, что часто встречаются, но в коде все равно могут быть не видны. Результаты тестирования продемонстрированы на таблице 9.

Таблица 9 – Тестирование структурным контролем

Вопрос	Результат проверки	Вывод
Обращения к данным		
Все ли переменные инициализированы?	Да, выбранная среда разработки подсвечивала бы ошибку, если бы были не инициализированные переменные.	Все переменные инициализированы.
Не превышены ли максимальные (или реальные) размеры массивов и строк?	Размеры коллекций проверены, не превышены.	Размеры массивов и строк не превышены.
Не перепутаны ли строки со столбцами при работе с матрицами?	Матрицы в проекте не используются.	Ошибок не найдено.
Присутствуют ли переменные со сходными именами?	Обнаружены несколько переменных с похожими именами, но разница между ними очевидна по смыслу.	Ошибок не найдено.
Используются ли файлы?	Файлы не используются.	Ошибок не найдено.
Использованы ли нетипизированные переменные, открытые массивы, динамическая память?	Используются коллекции с динамической памятью, но не происходит обращение по индексу.	Нарушений работы с памятью и массивами не обнаружено.

Продолжение таблицы 9

Вопрос	Результат проверки	Вывод
Вычисления		
Правильно ли записаны выражения (порядок следования операторов)?	Порядок следования операторов проверен, ошибок не выявлено.	Выражения записаны правильно.
Корректно ли производятся вычисления неарифметических переменных?	Логические операции и сравнения строк проверены, ошибок нет.	Работа с неарифметическими переменными происходит корректно.
Корректно ли выполнены вычисления с переменными различных типов (в том числе с использованием целочисленной арифметики)?	Если было бы смешение типов, среда разработки подчеркнула бы это как ошибку. Вычисления с переменными различных типов, в том числе при приведении типов, проверены.	Все вычисления проходят корректно.
Возможно ли переполнение разрядной сетки или ситуация машинного нуля?	Переполнение разрядной сетки и деление на ноль исключены в проекте.	Переполнение разрядной сетки невозможно.
Соответствуют ли вычисления заданным требованиям точности?	Расчеты дат и временных значений проверены, точность соблюдается.	Вычисления соответствуют требованиям точности.
Присутствуют ли сравнения переменных различных типов?	Все сравнения корректны, переменные приведены к соответствующим типам перед сравнением.	Ошибок в сравнении переменных разных типов нет.
Передача управления		
Будут ли корректно завершены циклы?	Циклы завершаются корректно	Ошибок в завершении циклов нет.

Продолжение таблицы 9

Вопрос	Результат проверки	Вывод
Будет ли завершена программа?	Программа завершает выполнение корректно, финализирующие методы вызываются правильно.	Завершение программы корректное.
Существуют ли циклы, которые не будут выполняться из-за нарушения условия входа?	Все условия входа проверены, циклы выполняются корректно.	Ошибок в выполнении циклов нет.
Существуют ли поисковые циклы? Корректно ли отрабатываются ситуации «элемент найден» и «элемент не найден»?	Поисковых циклов нет.	Ошибок не найдено.
Интерфейс		
Соответствуют ли списки параметров и аргументов по порядку, типу, единицам измерения?	Списки параметров и аргументов проверены, соответствуют требованиям.	Параметры и аргументы соответствуют спецификации.
Не изменяет ли подпрограмма аргументов, которые не должны изменяться?	Все неизменяемые данные помечены как readonly, среда разработки подчеркнула бы их изменения как ошибку.	Ошибок не найдено.
Не происходит ли нарушения области действия глобальных и локальных переменных с одинаковыми именами?	Все переменные имеют разный кейс в зависимости от их доступа.	Ошибок не найдено.

3.3 Оценочное тестирование

После функционального тестирования программу необходимо было протестировать с помощью реальных пользователей. Цель тестирования: тестирование программы на соответствие основным требованиям.

В процессе оценочного тестирования участвовали 5 человек. Участники оценивали программу по двум критериям: удобство использования и удобство эксплуатации. Выбор этих критериев основан на том, что удовлетворение этих аспектов существенно влияет на общую эффективность и приемлемость программы для конечного пользователя. Удобство использования отражает, насколько легко пользователям освоить и использовать программу, тогда как удобство эксплуатации оценивает уровень удовлетворенности пользователя в процессе реального использования. Результаты тестирования представлены в таблице 10.

Таблица 10 – Результаты оценочного тестирования

№ пользователя	Удобство использования	Удобство эксплуатации
1	10	8
2	6	7
3	9	9
4	7	9
5	7	8
Средняя оценка:	7,8	8,2

Тестирование показало, что программа получила достаточно высокие средние баллы по обоим критериям. Это говорит о том, что программа удобна в использовании и эксплуатации.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была спроектирована и разработана система удаленного мониторинга состояния персонального компьютера.

Разработанная программа полностью реализует функционал, указанный в техническом задании, а именно: .

Разработанная система включает в себя модульную структуру, позволяющую легко добавлять или удалять модули в зависимости от потребностей пользователей. Визуализация данных была выполнена с помощью графиков и таблиц, что позволяет быстро и эффективно анализировать, и интерпретировать статистические данные.

Результаты работы показали, что система удаленного мониторинга состояния персонального компьютера может быть эффективным инструментом для поддержания стабильности и производительности вычислительных систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Иванова Г.С. Технология программирования. - М: КноРус, 2022. – 333 с.

2 Обоснованный выбор модели жизненного цикла как фактор повышения качества разработки информационных систем – URL: https://www.gpntb.ru/win/inter-events/crimea2009/eng/disk/138.pdf [Электронный ресурс] (дата обращения 10.09.2025).

3 Документация по C# [Электронный ресурс] - URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата обращения: 15.09.2025).

4 .NET документация фреймворка [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/dotnet/ (дата обращения: 18.09.2025).

5 EF Core документация фреймворка [Электронный ресурс] - URL: https://learn.microsoft.com/ru-ru/ef/ (дата обращения: 18.09.2025).

6 Vue 3 документация фреймворка [Электронный ресурс] - URL: https://ru.vuejs.org/guide/introduction.html (дата обращения: 19.09.2025).

7 LibreHardwareMonitorLib документация библиотеки [Электронный ресурс] - URL: https://github.com/LibreHardwareMonitor/LibreHardwareMonitor/ (дата обращения: 21.09.2025).

8 Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс] - URL: https://habr.com/ru/articles/566218/ (дата обращения: 21.09.2025).

9 SQLite документация [Электронный ресурс] - URL: https://www.sqlite.org/docs.html (дата обращения: 02.10.2025).

ПРИЛОЖЕНИЕ А

Техническое задание

Листов 8

1 Введение

Настоящее техническое задание распространяется на разработку программного продукта “Система удаленного мониторинга состояния персонального компьютера”, предназначенного для обеспечения эффективного и точного сбора, анализа и хранения данных о показателях нагрузки, температуры и памяти ПК.

Постоянный рост вычислительных мощностей, необходимых для рендеринга графики, обучения нейросетевых моделей и выполнения сложных инженерных расчетов, приводит к повышению нагрузки на компоненты ПК, увеличению энергопотребления и рискам перегрева оборудования. Отсутствие накопленных данных о работе системы в разные периоды времени затрудняет глубокую диагностику аппаратных проблем и оптимизацию энергозатрат.

Поэтому было принято разработать систему удаленного мониторинга состояния персонального компьютера, которая будет включать в себя функции общей статистики по состоянию множества ПК в сети.

2 Основания для разработки

Основанием для разработки программного продукта является учебный план кафедры ИУ6 «Компьютерные системы и сети» факультета ИУ «Информатика и системы управления» МГТУ им. Н.Э. Баумана на 5-й семестр, утвержденный ученым советом МГТУ им Н.Э. Баумана.

3 Назначение разработки

Основное назначение СУМСПК заключается в сборе и хранении статистических данных мониторинга состояния ПК, что позволяет замечать проблемы в работе техники, оптимизировать нагрузку на системы из нескольких компьютеров и отслеживать состояние вычислительной техники.

4 Требования к программному изделию

4.1 Требования к функциональным характеристикам

4.1.1 Выполняемые функции

Для администратора сети:

– выбор варианта отображения данных;

– определение временного промежутка отчета;

– выбор ПК и комплектующих;

– построение графика работы;

– просмотр архивных данных о работе ПК.

4.1.2 Исходные данные:

Для врачей:

– вариант отображения данных;

– ПК и комплектующие для просмотра данных;

– промежуток времени для расчёта и визуализации данных.

4.1.3 Результаты:

Для врачей:

– таблица по общей статистики: значение всех доступных показателей для каждого ПК;

– графики для актуального состояния конкретных комплектующих в конкретном ПК;

– графики для архивного состояния конкретных комплектующих в конкретном ПК.

4.2 Требования к надежности

4.2.1 Предусмотреть контроль вводимой информации.

4.2.2 Обеспечить целостность информации в базе данных.

4.3 Условия эксплуатации.

4.4 Требования к составу и параметрам технических средств

4.4.1 Программное обеспечение должно функционировать на IBM-совместимых персональных компьютерах.

4.4.2 Минимальная конфигурация клиента:

4.4.2.1 Тип процессора ..... Intel Pentium 4 67

4.4.2.2 Объем ОЗУ..... 2 Гб.

4.4.2.3 Последняя версия браузеров Google Chrome, Firefox или Opera.

4.4.2.4 Одна из следующих операционных систем семейств Windows, Linux.

4.4.3 Минимальная конфигурация сервера:

4.4.3.1 Тип процессора ..... Intel Xeon E3-1220

4.4.3.2 Объем ОЗУ..... 8 Гб.

4.4.3.3 Одна из следующих операционных систем: Windows Server, Linux, Unix.

4.4.3.4 Объём физической памяти сервера..... 128 Гб.

4.5 Требования к информационной и программной совместимости

4.5.1 Программное обеспечение должно работать под управлением операционных систем семейств Windows, Linux.

4.5.2 Входные данные представлены в следующем формате: текстовый, числовой.

4.5.3 Результаты должны быть представлены в следующем формате: графический, текстовый.

5 Требования к программной документации

5.1 Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

5.2 Разрабатываемое программное обеспечение должно включать справочную систему, включающую указания по настройке и поддержке систем.

5.3 В состав сопровождающей документации должны входить:

5.3.1 Расчетно-пояснительная записка на 25-30 листах формата А4 (без приложений 5.3.2, 5.3.3 и 5.3.4).

5.3.2 Техническое задание (Приложение А).

5.3.3 Руководство пользователя (Приложение Б).

5.3.4 Фрагмент исходного кода программы (Приложение В).

5.4 Графическая часть должна быть включена в расчетно-пояснительную записку в качестве иллюстраций:

5.4.1 Схемы взаимодействия объектов, объектная декомпозиция.

5.4.2 Диаграмма вариантов использования.

5.4.3 Концептуальная модель предметной области.

5.4.4 Граф состояний интерфейса.

5.4.5 Даталогическая модель базы данных.

5.4.6 Диаграммы компоновки программных компонентов.

5.4.7 Таблицы тестов.

6 Стадии и этапы разработки

Этап	Содержание этапа	Сроки и объем	Представляемые результаты	
			Спецификации и программный продукт	**Документы**
1.	Выбор темы, составление задания, решение организационных вопросов	1..2 недели (10 %)	-	**Заполненный бланк задания на курсовую работу – вывешивается на сайт кафедры для получения утверждающей подписи заведующего кафедрой**
2.	Анализ предметной области, разработка ТЗ. Выбор архитектуры системы: Клиент-серверная архитектура с веб-интерфейсом для доступа пользователей. Использование реляционной базы данных для хранения данных о пациентах и их движении.	3..4 недели	Результаты декомпозиции предметной области. Эскизный проект: интерфейс, схемы, возможно, часть программы (выбранные готовые решения).	Фрагмент расчетно-пояснительной записки с обоснованием выбора средств и подходов к разработке
3.	**Сдача ТЗ**	**4 неделя (25 %)**	**-**	**Техническое задание – утверждается руководителем**
4.	Проектирование и реализация основных компонентов – ядра программы	5..7 недели	Технический проект основной части: структура программы, алгоритмы программ, описания структур данных, диаграмма классов – в зависимости от выбранной технологии разработки. Программный продукт, реализующий основные функции (демонстрируется руководителю)	Фрагмент расчетно-пояснительной записки с обоснованием разработанных спецификаций Тексты части программного продукта, реализующего основные функции.
5.	**Сдача прототипа программного продукта**	**7 неделя (50 %)**	**Прототип программного продукта – демонстрируется руководителю**	

Этап	Содержание этапа	Сроки и объем	Представляемые результаты	
			Спецификации и программный продукт	**Документы**
6.	Разработка компонентов, обеспечивающих функциональную полноту	8..10	Рабочий проект программы. Готовая программа	Черновик расчетно-пояснительной записки. Тексты программного продукта.
7.	**Сдача программного продукта**	**11 неделя (75 %)**	**Готовая программа – оценивается руководителем в баллах**	**-**
8.	Тестирование программы и подготовка документации	12..14	Тесты и результаты тестирования.	РПЗ и Руководство пользователя.
9.	**Оформление и сдача документации**	**14 неделя (90 %)**	**–**	**Расчетно-пояснительная записка и Руководство пользователя – проверяются и подписываются руководителем**
10.	Защита курсовой работы	15..16 недели (100%)	–	Доклад (3-5 минут). Защита курсовой работы. Подписанная документация – вывешивается на сайт кафедры

7 Порядок контроля и приемки

7.1 Порядок контроля

Контроль выполнения осуществляется руководителем еженедельно.

7.2 Порядок защиты

Защита осуществляется комиссии преподавателей кафедры.

7.3 Срок защиты

Срок защиты: 15-16 недели.

8 Примечание

В процессе выполнения работы возможно уточнение отдельных требований технического задания по взаимному согласованию руководителя и исполнителя.

ПРИЛОЖЕНИЕ Б

Руководство пользователя

Листов 4

1 Общие сведения о программном продукте

Система удаленного мониторинга состояния персонального компьютера предназначена для сбора и визуализации данных. Она позволяет пользователя просматривать графики и таблицы, представляющие данные мониторинга персональных компьютеров, подключенных к сети.

Этот программный продукт является важным инструментом для системного администрирования, который помогает контролировать нагрузку и состояние компонентов компьютеров удаленно и в процессе работы за устройством.

Работа с описываемым программным продуктом осуществляется с помощью персонального компьютера.

2 Описание запуска и инструкция по работе

Чтобы начать работу с программой, нужно иметь исполняемые файлы, который можно получить в репозитории гит.

Перед запуском программ рекомендуется проверить файлы конфигурации appsettings.json в каталоге с исполняемым файлом. В файлых конфигурации стоит указать Url сервера, можно изменить интервалы сбора и отправки данных и изменить базу данных для сохранения данных.

После запуска на клиенте появится консоль, с помощью которой можно ввести имя ПК. После ввода имени в консоль будут выводится актуальные данные о состоянии ПК.

Консоль ввода имени показана на рисунке Б.1.

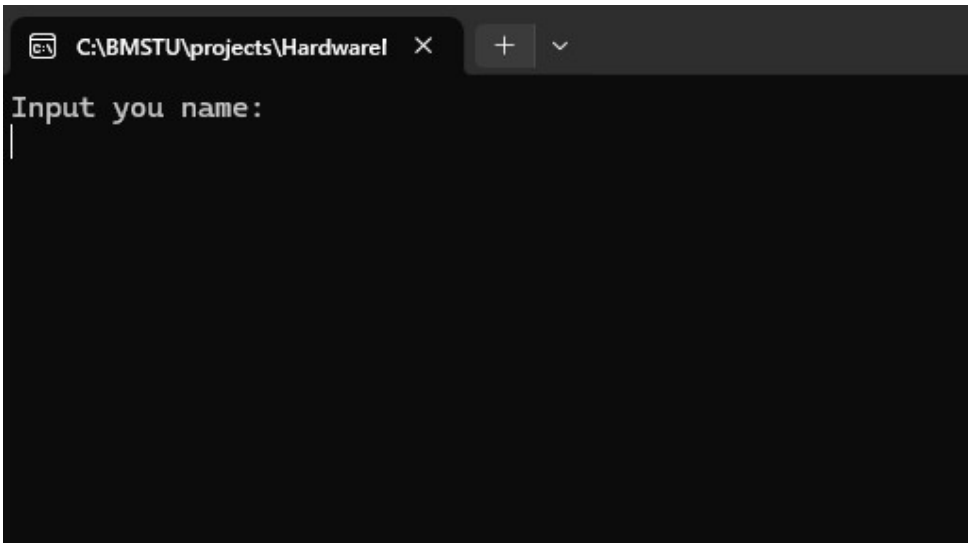

Рисунок Б.1 – Главная страница

После ввода имени в консоль будут выводится актуальные данные о состоянии ПК. Пример отображаемых данных показан на рисунке Б.2.

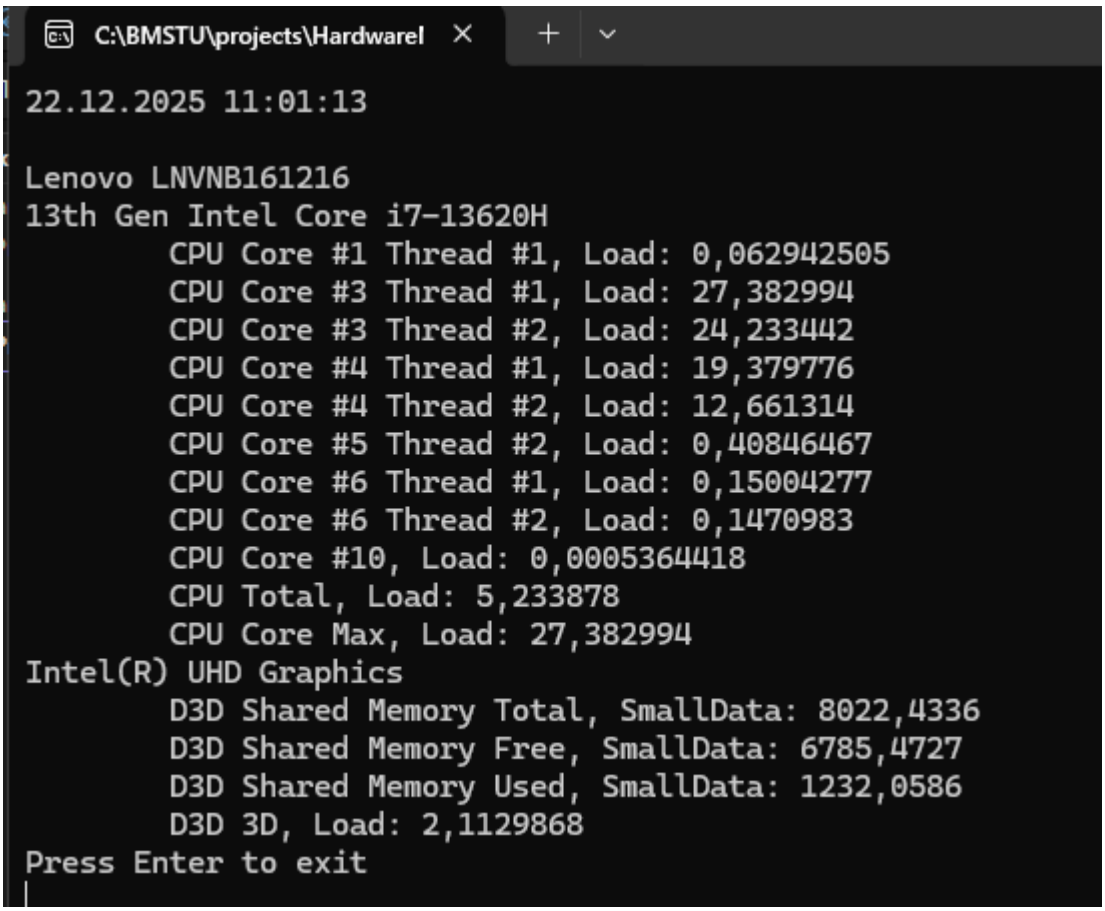

Рисунок Б.2 – Данные с сенсоров ПК

После запуска сервера пользователь может зайти по ссылке, которая указана в appsettings.json файле в поле Url и попасть на главную страницу, которая показана на рисунке Б.3.

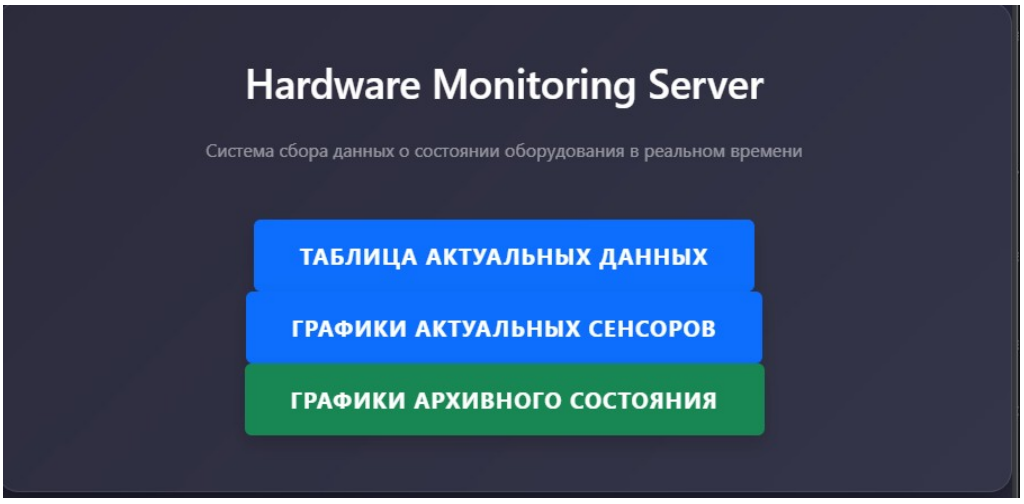

Рисунок Б.3 – Главная страница

После нажатия на кнопки в главной странице откроется соответствующая страница. На рисунке Б.4 изображена страница на примере графика состояния систем. В полях «С даты» и «По дату» вводится время

выборки. Это помогает отфильтровать данные, если требуется конкретный период времени. В страницах с графиками есть поле «Выберите сенсор», которого нет на странице таблиц. При нажатии на него отображаются доступные сенсоры. Так-же на всех страницах присутствует поле «Выберите компьютер», после которого отображается список всех доступных ПК, для которых можно отобразить данные.

После нажатия на кнопку «Обновить данные» происходит обновление данных с применением указанных фильтров и строится соответсвующий график или таблица.

Для возвращения на главную страницу достаточно нажать на кнопку «На главную» в правом верхнем углу.

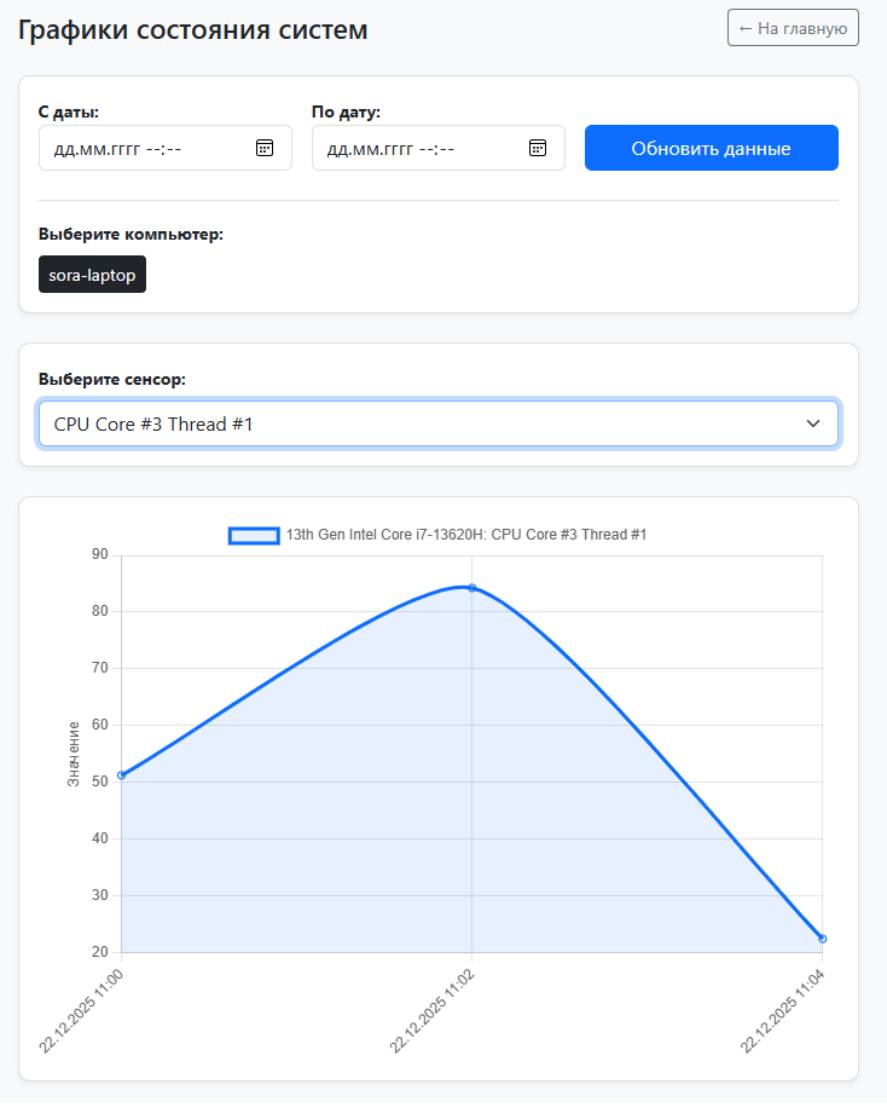

Рисунок Б.4 – Страница графики состояния систем

Для завершения работы приложений достаточно нажать ctrl+c в соответствующем приложении, либо закрыть его через крестик.

ПРИЛОЖЕНИЕ В

Фрагмент исходного кода программы

Листов 4

В качестве фрагмента исходного кода приведен код файла Program.cs, который содержит в себе подключения контроллеров, контекста базы данных и других сервисов. Модели в EF описывают схемы баз данных. Исходный код файла представлен в листинге В.1.

Листинг В.1 – Код файла Program.cs

```
using HardwareMonitoringServer.DbSetting;
using HardwareMonitoringServer.Monitor;
using Microsoft.EntityFrameworkCore;

namespace HardwareMonitoringServer
{
    class Program
    {
        private static Timer _clearTimer;

        static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            var configuration = builder.Configuration;
                                        var clearInterval =
long.Parse(configuration["TimerSettings:ClearInterval"]
                                                ?? throw new
InvalidOperationException("ClearInterval not found."));
                    var baseUrl = configuration["Base:Url"] ??
"http://localhost:5050";
                                        var dbConnection =
configuration["ConnectionStrings:DefaultConnection"] ?? "Data
Source=hardware_monitor.db";

            builder.Services.AddDbContext<AppDbContext>(options
=>
                options.UseSqlite(dbConnection));
```

Продолжение листинга В.1

```
            builder.Services.AddSingleton<DataMonitoring>();

             builder.Services.AddControllers();

             builder.WebHost.UseUrls(baseUrl);

             builder.Services.AddRazorPages();

             var app = builder.Build();

             app.UseDefaultFiles();
             app.UseStaticFiles();

             app.MapControllers();

                                  var   dataMonitoringService   =
app.Services.GetRequiredService<DataMonitoring>();

             TimerCallback timerDelegate = _ =>
             {
                 using (var scope = app.Services.CreateScope())
                 {
                                                       var   db   =
scope.ServiceProvider.GetRequiredService<AppDbContext>();
                                                    var   monitor   =
scope.ServiceProvider.GetRequiredService<DataMonitoring>();

                                                 var   averagedData   =
monitor.GetAveragedData();
                     if (averagedData.Any())
                     {
                         db.Computers.AddRange(averagedData);
```

Продолжение листинга В.1

```
                        db.SaveChanges();
                        monitor.ClearSys();
                                    Console.WriteLine($"Saved
{averagedData.Count} records to DB.");
                    }
                }
            };

            _clearTimer = new Timer(timerDelegate,
                                    null,
                                    0,
                                    clearInterval);

                    Console.WriteLine($"Server started with
Controllers...");
            Console.WriteLine($"Data clearing timer set to fire
every {clearInterval} ms.");

            app.Run();
        }
    }
}
```

Исходный код программы расположен в открытом репозитории: https://github.com/SoraEien/HardwareMonitoring