«БУ ВО ХМАО-ЮГРЫ «СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ

УНИВЕРСИТЕТ»

Реферат по дисциплине

«Методы и средства проектирования информационных систем»

по теме ВКР

«Информационная система для диагностики

аппаратной части персонального компьютера»

Работу выполнил студент группы 607-01:

Веревкин А.И.

Проверил:

Столбов Д.А.

Сургут, 2023

Оглавление

[1. Видение 4](#_Toc153035873)

[Введение 4](#_Toc153035874)

[Возможности 4](#_Toc153035875)

[Рынок 4](#_Toc153035876)

[SWOT - анализ 4](#_Toc153035877)

[Экономические предпосылки 5](#_Toc153035878)

[Тенденции 5](#_Toc153035879)

[Заинтересованные лица 5](#_Toc153035880)

[Основные задачи высокого уровня 6](#_Toc153035881)

[Позиционирование 7](#_Toc153035882)

[Продвижение 7](#_Toc153035883)

[Расчет рынка 7](#_Toc153035884)

[Описание бизнес-модели 7](#_Toc153035885)

[Объем рынка 7](#_Toc153035886)

[Ценообразование 7](#_Toc153035887)

[Конкуренты 9](#_Toc153035888)

[Перспективы решения 11](#_Toc153035889)

[2. Словарь терминов 12](#_Toc153035890)

[3. Дополнительная спецификация 14](#_Toc153035891)

[Функциональность 14](#_Toc153035892)

[Регистрация событий и обработка ошибок 14](#_Toc153035893)

[Безопасность 14](#_Toc153035894)

[Удобство использования 14](#_Toc153035895)

[Человеческие факторы 14](#_Toc153035896)

[Надежность 15](#_Toc153035897)

[Возможность восстановления информации 15](#_Toc153035898)

[Производительность 15](#_Toc153035899)

[Возможности поддержки 15](#_Toc153035900)

[Адаптация системы 15](#_Toc153035901)

[Конфигурирование 15](#_Toc153035902)

[Ограничения 16](#_Toc153035903)

[Приобретаемые компоненты 16](#_Toc153035904)

[Бесплатные компоненты на основе открытого кода 16](#_Toc153035905)

[Интерфейсы 16](#_Toc153035906)

[Важные интерфейсы и аппаратные средства 16](#_Toc153035907)

[Программные интерфейсы 16](#_Toc153035908)

[Бизнес-правила 17](#_Toc153035909)

[Вопросы законодательства 17](#_Toc153035910)

[Информация из предметной области 17](#_Toc153035911)

[Ценовая политика 17](#_Toc153035912)

[Бизнес модель 17](#_Toc153035913)

[Успешность продукта 18](#_Toc153035914)

[Налогообложение 18](#_Toc153035915)

[4. Концептуальная модель 19](#_Toc153035916)

[5. Use Case диаграмма 20](#_Toc153035917)

[6. Анализ прецедентов 21](#_Toc153035918)

[Общие параметры 21](#_Toc153035919)

[Прецедент 1 - получение информации с датчиков ПК для ознакомления с состоянием устройства. 21](#_Toc153035920)

[Основной успешный сценарий. 21](#_Toc153035921)

[Альтернативный неуспешный сценарий. 21](#_Toc153035922)

[Прецедент 2 - получение информации с датчиков ПК для устранения неполадки. 21](#_Toc153035923)

[Основной успешный сценарий. 21](#_Toc153035924)

[Альтернативный неуспешный сценарий. 22](#_Toc153035925)

[Сценарий с недостоверной информацией. 22](#_Toc153035926)

[7. Диаграмма компонентов 23](#_Toc153035927)

[1. Определение предметной области 23](#_Toc153035928)

[2. Выделение компонентов 23](#_Toc153035929)

[3. Определение интерфейсов 24](#_Toc153035930)

[8. Диаграмма классов 26](#_Toc153035931)

[9. Прототип UI 29](#_Toc153035932)

[Приложение 37](#_Toc153035933)

[Приложение 1. Реализация интерфейса. 37](#_Toc153035934)

# 1. Видение

Краткое наименование: AInPC

Полное наименование: Программа для диагностики аппаратной части ПК «Aleksey’s Information of PC»

## Введение

Проект представляет собой разработку и внедрение специализированной программы для упрощения и оптимизации процесса обслуживания и ремонта компьютерной техники.

## Возможности

1. Получение общей (неизменяющейся) информации о комплектующих ПК и информации со всех доступных датчиков этих комплектующих - температура, частота, загруженность, вольтаж, напряжение.
2. Визуализация всей числовой информации в виде графиков.
3. Наличие информации о всех устройствах системы, т.е. отсутствие специализации на одном их комплектующих.
4. Возможность выгрузить в отчет основную информацию о системе и информацию, предоставляемую программой.
5. Возможность быстрого доступа в Windows PowerShell и Командную строку Windows.

## Рынок

### SWOT - анализ

Таблица 1. SWOT - анализ

|  | Возможности | Угрозы |
| --- | --- | --- |
| Внешние | 1. ИС может быть полезна как для специалистов в обслуживании ПК, так и для рядовых пользователей.  2. В результате реализации работы ее можно будет монетизировать.  3. Так как пользователь покупает подписку на продукт, его возможно вовремя обновлять.  4. Отсутствие прямых конкурентов на Российском рынке | 1. Существуют конкуренты в виде пиратских программ иностранных производителей.  2. Ограничение в виде малого пула поддерживаемых аппаратных платформ.  3. Если пользователи не будут устанавливать обновления, то они могут не получить некоторые "фичи". |
| Внутренние | 1. Активное импортозамещение иностранных продуктов в Российской экономике.  2. Повышенный спрос на подобные продукты в связи с цифрофизацией.  3. Сотрудничество с компаниями, заинтересованными в приватности своих данных. | 1. Иностранные компании могут вернуться на Российский рынок.  2. Отсутствие опыта в разработке масштабного коммерческого продукта.  3. Малый размер команды разработчиков. |

### Экономические предпосылки

Официальный доступ к иностранным продуктам и их новейшим версиям закрыт, поэтому работа специлистов по обслуживанию компьютерной техники становится труднее. И в связи с этим, а также в условиях стремления к импортозамещению и снижению зависимости от импортных компонентов и технологий, создание и внедрение специализированной программы, оптимизирующей работу специалиста по обслуживанию компьютеров, становится особенно актуальным.  
В результате реализации проекта программа позволит упростить и оптимизировать процесс обслуживания техники, а также позволит получать обновления до новейших версий, содержащих актуальную информацию о компьютерном оборудовании, максимально оперативно.

### Тенденции

В первом квартале 2023 г. в России было продано более 1,8 млн персональных компьютеров, что на 0,4% больше, чем за аналогичный период 2022-го года. Даже учитывая уход с Российского рынка множества иностранных компанией-производителей электронной техники, наблюдается положительная динамика в количестве продаваемых компьютеров. Так, уже в 2020 году на 1000 россиян приходилось 125 персональных компьютеров.

### Заинтересованные лица

В качестве возможных потребителей рассматриваются как юридические лица - малый и средний бизнес (компьютерные сервисы и мастерские, специализирующиеся на диагностике персональных компьютеров), а также физические лица - пользователи, которые могут быть заинтересованы в получении информации с датчиков своей системы. Исходя из этого, стартап-проект ориентируется больше на рынок B2B, но не ограничивается им. Географическое расположение клиентов значение не имеет, так как распространение продукта будет происходить с использованием сети Интернет.

### Основные задачи высокого уровня

Таблица 2. Основные задачи высокого уровня

| Заинтересованное лицо | Цель высокого уровня | Проблемы, возможности и замечания | Текущие решения |
| --- | --- | --- | --- |
| Работники компьютерных сервисов | 1. Чтобы найти проблему в компьютере.  2. Чтобы устранить неполадку.  3. Чтобы заработать деньги. | Отсутствие альтернативы для коммерческого использования, которую можно получить законным образом. | Пиратские копии иностранного ПО. |
| Технические специалисты в компаниях | 1. Чтобы найти проблему в компьютере.  2. Чтобы устранить неполадку.  3. Чтобы выполнить должностные обязанности.  4. Чтобы сохранить работу.  5. Чтобы заработать деньги. | Отсутствие альтернативы для коммерческого использования, которую можно получить законным образом. | Пиратские копии иностранного ПО. |
| Компьютерные энтузиасты | 1. Чтобы оптимизировать и поддерживать работу ПК.  2. Чтобы ПК исправно работал.  3. Чтобы без проблем пользоваться им.  4. Чтобы получать моральное удовольствие. | Отстуствие возможности получить полную версию иностранного ПО, которое показывало бы всю информацию об устройстве. | Пиратские копии иностранного ПО. |
| Специалисты по кибербезопасности в крупных компаниях | 1. Чтобы при обслуживании устройств информация была в безопасности.  2. Чтобы конфинденциальная информация компании не попала в руки преступников.  3. Чтобы предприятие не несло убытки в связи с утечкой информации. | Отсутствие возможности альтернативы для коммерческого использования, которую можно получить законным образом и отсутствие каких-либо гарантий безопасности со стороны разработчиков. | Пиратские копии иностранного ПО, используемые на устройстве без подключения к Интернету, а также на системах, запускаемых с отдельного носителя информации |

### Позиционирование

Конечный продукт позиционируется как аналог иностранным продуктам, у которых на текущий момент возможно получить исключительно пиратские копии.

### Продвижение

В виде посредников будут выступать: социальные сети, где будут созданы страницы компании и продукта, которые будут держать клиентов в курсе последних событий и обновлений; блогеры и инфлюенсеры в области работы с персональными компьютерами, которые могли бы своим зрителям продемонстировать функционал и преимущества программного обеспечения.

### Расчет рынка

#### Описание бизнес-модели

Модель распространения продукта - ASP. Так, конечному пользователю будет предоставляться инсталлятор программного обеспечения и ключ активации к нему. Оплата будет происходить единожды за определенный период времени - 1 месяц/6 месяцев/1 год. В лицензиях для коммерческого использования количество активаций для одного ключа может различаться в зависимости от нужд компании, а в лицензиях для личного использования количество активаций будет ограничено заранее.

#### Объем рынка

Согласно данным на 2020 год на 1000 человек в Российской Федерации приходится в среднем 125 компьютеров. При этом, в каждом районе Москвы находится более 5000 компьютерных мастерских.

#### Ценообразование

Предположительная стоимость продукта: 2000 рублей - стоимость лицензии для личного использования, 12000 рублей - стоимость лицензии для мастерской.  
Потенциальные расходы для реализации проекта:

1. Разработка - зарплата сотрудников, приобретение лицензий на программное обеспечение и инструменты разработки, закупка оборудования, для которого делается программа.
2. Тестирование и отладка - тестирование программы на разных конфигурациях компьютеров и операционных системах, исправление ошибок и отладка программы.
3. Документация - создание руководств пользователя, технической документации и описание программы.
4. Маркетинг - создание веб-сайта, рекламные материалы и продвижение в социальных сетях.
5. Поддержка и обновление - поддержка пользователей, ответы на их вопросы, решение проблем, разработка и выпуск обновлений программы.
6. Лицензирование и сертификация.

Себестоимость продукта складывается из перечисленных выше расходов. Но на текущий момент тяжело определить его себестоимость, так как из года в год она будет меняться из-за изменчивости рынка компьютерных технологий, однако, если выдвигать предварительную себестоимость с учетом перечисленных выше цифр, то в первый год она будет составлять 1220000 рублей/год, во второй - 970000 рублей/год, в третий - 1610000 рублей/год. Данная сумма показывает не стоимость единицы, а стоимость поддержки продукта.

### Конкуренты

Таблица 3. Конкуренты

| Конкурент | Минусы | Конкретные преимущества проекта | Общие преимущества продукта |
| --- | --- | --- | --- |
| AIDA64 | * отсутствие возможности тестирования видеокарты и ограниченный набор тестов для тестирования жестких дисков и SSD (Solid State Drive); * малый набор информации о видеокарте, который включает в себя лишь поверхностную информацию, порой недостаточную для качественной диагностики; * отсутствие быстрого доступа в командную строку Windows или в Windows PowerShell, который иногда бывает необходим для получения дополнительной информации об устройстве. | * большое количество информации о видеоустройстве - частота видеоядра, частота памяти, температура видеокарты, объем памяти и т.д. | Наличие возможности получить быстрый доступ к Windows PowerShell и к командной строке Windows, а также наличие "быстрых" команд - кнопок с заранее записанными командами для данных программ. |
| HWInfo | * отсутствие каких – либо возможностей для тестирования компьютера; * отсутствие быстрого доступа в командную строку Windows или в Windows PowerShell, который иногда бывает необходим для получения дополнительной информации об устройстве; * отсутствие визуального сопровождения к данным, представленным в таблицах. | * наличие графиков, иллюстрирующих письменные данные. |
| OCCT | * отсутствие возможности тестирования жестких дисков и SSD (Solid State Drive); * отсутствие информации о каких – либо датчиках компьютера, кроме процессора, видеокарты, материнской платы и оперативной памяти; * отсутствие быстрого доступа в командную строку Windows или в Windows PowerShell, который иногда бывает необходим для получения дополнительной информации о персональном компьютере. | - |

1. AIDA64  
   AIDA64, страна разработки - Венгрия, утилита для тестирования и идентификации компонентов персонального компьютера под управлением операционных систем Windows, предоставляющая детальные сведения об аппаратном и программном обеспечении.
2. HWInfo  
   HWInfo, страна разработки - Словакия, утилита, с помощью которой можно получить детальную информацию об аппаратных компонентах персонального компьютера (например, температуре жесткого диска и данные S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology), показания датчиков и т.д.).
3. OCCT  
   OCCT, страна разработки - Франция, программа предназначена для комплексной проверки комплектующих компьютера на стабильность и помимо наборов тестов, программа отображает общую информацию о компонентах и позволяет мониторить температуру, энергопотребление и многое другое.

## Перспективы решения

Планируемый срок окупаемости проекта составляет 2 года. По итогам второго года существования проекта чистая прибыль составит - 134 000 рублей. По итогам третьего года существования проекта прибыль составит - 1 806 000 рублей.   
Данная информация рассчитана с учетом следующих параметров: 0.8 рублей - просмотр одной рекламной интеграции, 2000 рублей - стоимость лицензии для личного использования, 12000 рублей - стоимость лицензии для мастерской.

# 2. Словарь терминов

Таблица 4. Словарь терминов

|  |  |  |
| --- | --- | --- |
| Термин | Определение | Синонимы |
| Сокет | Специальное место на материнской плате компьютера, в которое вставляется процессор (центральный процессор). Каждый тип процессора обычно использует свой собственный сокет, поэтому сокет определяет, какой процессор можно установить на материнскую плату. | Разъем, гнездо, контакт. |
| Кэш процессора | Это маленькая и очень быстрая память, которая находится прямо внутри центрального процессора компьютера. Её задача - хранить часто используемые данные и команды, чтобы процессор мог получить к ним быстрый доступ без необходимости обращения к более медленной оперативной памяти или жесткому диску. | Кэш CPU, кэш микропроцессора, внутренний кэш, локальный кэш. |
| Драйвер | Это специальная программа или файл, который позволяет операционной системе понимать и взаимодействовать с конкретным аппаратным устройством. | Управляющая программа, подсистема управления, интерфейсное ПО, управляющий код. |
| Шина процессора | Служит для передачи информации между процессором и остальными компонентами устройства, чтобы компьютер мог выполнять задачи, обрабатывать данные и отображать графику. | Шина данных, внутренняя шина, центральная шина. |
| Фрейм-буфер видеокарты | Часть графического ускорителя хранящая текущий отображаемый кадр, и её нагрузка может увеличиваться при работе с приложениями, где происходит частое обновление экрана, такими как видеоигры или 3D-моделирование. | Видео-буфер, буфер кадра, видеопамять |
| BIOS | Basic Input/Output System - это набор программных инструкций, находящихся в специальном микрочипе на материнской плате компьютера. Эти инструкции являются частью системного программного обеспечения компьютера и выполняют ряд важных функций. | Загрузочная программа, загрузчик, прошивка системы |
| Файл подкачки | Это специальный файл на жестком диске компьютера, который используется операционной системой для временного хранения данных, которые не помещаются в оперативной памяти компьютера. | Виртуальная память, файл свопа, файл обмена |
| Фреймворк | Это набор готовых инструментов, правил и библиотек программного обеспечения, который помогает разработчикам создавать приложения более быстро и эффективно. | Платформа разработки, библиотека инструментов, рабочая база |
| CPU | Central Processing Unit – центральный процессор | Вычислительный узел, управляющий блок, микропроцессор |
| GPU | Graphics Processing Unit – видеокарта или графический ускоритель | Графический адаптер, видеоадаптер, видеоконтроллер, графическое устройство |
| RAM | Random Accessing Memory – оперативная память | Системная память, физическая память, временное хранилище данных |

# 3. Дополнительная спецификация

В данном документе описываются требования к ПО для диагностике аппаратного обеспечения ПК AInPC, не вошедшие в описание прецендентов.

## Функциональность

Фукнциональность, которая имеет отношение ко многим прецендентам.

### Регистрация событий и обработка ошибок

Регистрация событий главным образом происходит в момент обращения программы к функции опроса .dll библиотеки, там проверяется от какого именно из классов поступил запрос и, в зависимости от этого, данные возвращаются в некоторой форме.

Обработка ошибок реализована как на уровне запуска функции "main" - проверка на то, что пользователь, например, не отказался запускать программу от имени администратора, так и в глубине кода на уровне интерфейсов - проверка на возможность получить доступ к базу данных.

### Безопасность

Программа не собирает никаких личных данных пользователя, однако, в любом случае, гарантируется, что данные об устройстве клиента не хранятся и не передаются, как третьим лицам, так и компании-разработчику.

## Удобство использования

### Человеческие факторы

Было проанализировано 5 прямых аналогов продукта (часть из которых см. в Видение.md). В результате чего был выявлен оптимальный вариант дизайна графического интерфейса продукта, который включает в себя наилучшие решения конкурентов.

Помимо этого предполагается, что пользователь будет работать на стандартном мониторе, поэтому размер выводимой информации нужно оптимизировать под рекомендуемые 40 сантиметров расстояния до монитора.

При этом важно использовать комфортную для глаз пользователя цветовую палитру, не содержащую "ядовитых" или броских цветов, за исключением моментов, когда необходимо сфокусировать внимание пользователя на чем-либо.

## Надежность

### Возможность восстановления информации

Программа не хранит никаких данных, которые нужно было бы восстанавливать, поэтому нет необходимости обеспечивать данный функционал. В случае неожиданного завершения работы программы, простой перезапуск ПО или, в крайнем случае, перезагрузка устройства помогут возобновить работу программы.

## Производительность

Обеспечение производительности главным образом сводится к обеспечению плавной работы графического интерфейса, данная особенность реализуется при помощи многопоточности. В отдельный поток будут вынесены опрос библиотеки и подготовка данных к выводу в нужном формате.

## Возможности поддержки

### Адаптация системы

Система не будет адаптироваться под ОС клиента, однако ее вывод будет различным в зависимости от некоторых входных данных.

1. Входные данные, влияющие на вывод системы:
2. Тип устройства клиента - компьютер или ноутбук.
3. Наличие дискретной и\или встроенной видеокарты.
4. "Официальность" оперативной памяти"
5. Зарегистрирован процессор или нет (возможно, пользователь использует фабричный образец) и др.

### Конфигурирование

В зависимости от наличия или отсутствия каких-либо аппаратных компонентов в системе, программа будет показывать разные данные.

Так, например, если в устройстве пользователя используется оперативная память производителя, который не определяется даже системой, а просто показывается как набор чисел, программа уведомит пользователя о том, что данный производитель незарегистрирован. В случае, если пользователь использует ноутбук, программа скроет вкладки "Вольтаж" и "Вентиляторы", так как данный тип устройств не собирает эту информацию. В случае, если на борту системы используются дискретная и встроенная видеокарты, то раздел "Видеокарта" будет показывать информацию об обоих устройствах.

## Ограничения

К ограничениям можно отнести то, что необходима версия Windows не ниже 7 и рекомендуемый объем оперативной памяти составляет не менее 2 Гб, что, в целом, является стандартом для современного ПО.

## Приобретаемые компоненты

Для работы ПО на устройстве клиента не требуется приобретать какое-то специальное оборудование, за исключением того, что описывается в ограничениях.

## Бесплатные компоненты на основе открытого кода

К данному пункту можно отнести все модули для ЯП Python, которые используется в разработке ПО, например, PyQt6, PythonNet, os и др.

## Интерфейсы

### Важные интерфейсы и аппаратные средства

Если говорить про важные интерфейсы, реализуемые вне ПО, то к таким интерфейсам можно отнести все, реализуемые самой ОС, поэтому крайне важно, чтобы пользователь использовал официальную или же хотя бы не "репак" версию ОС, где эти интерфейсы могут отсутствовать.

### Программные интерфейсы

К программным интерфейсам относятся те же интерфейсы, что были приведены в диаграмме компонентов.

**Интерфейс для "Базы данных":**  
**Название интерфейса:** IDB  
**Описание:** интерфейс обеспечивает доступ к данным в базе данных  
**Операции:** SELECT FROM (table\_name) \* WHERE (hardware\_name = OS Output)

**Интерфейс для "dll\_fetch":**  
**Название интерфейса:** Ifetch  
**Описание:** интерфейс обеспечивает нужный формат выходных данных для каждого инициирующего компонента  
**Операции:** initialize\_lib(), fetch\_stats(), parse\_sensor(), fetch\_stats\_for\_graphs(), parse\_sensor\_for\_graphs()

**Интерфейс для "Lib.dll":**  
**Название интерфейса:** Idll  
**Описание:** интерфейс обеспечивает доступ к данным с датчиков устройства  
**Операции:** Hardware.Computer.open()

## Бизнес-правила

Таблица 5. Бизнес-правила

| Имя | Правило | Возможность изменения | Источник |
| --- | --- | --- | --- |
| ПРАВ1 | Стоимость лицензии для персонального использования - 2000 рублей, для коммерческого использования - 12000 рублей. | Крайне высокая возможность изменения. В зависимости от итогового спроса на продукт цена будет меняться. | Политика торговых организаций. |
| ПРАВ2 |  |  |  |
| ПРАВ3 |  |  |  |

## Вопросы законодательства

Рекомендуется использовать бесплатные компоненты на основе открытого кода, если их разрешено использовать в коммерческих программных продуктах.

В ходе работы возможно использование нейронных сетей, как для компенсации слабых сторон разработчиков, так и, например, для генерации изображений. Поэтому крайне важно мониторить и соблюдать законы, которые касаются ИИ.

Крайне важно на постоянной основе выплачивать все налоги, а также постоянно мониторить статус налогоплательщика во избежание проблем.

## Информация из предметной области

### Ценовая политика

Как было сказано выше, у лицензии товара будет 2 варианта покупки - для персонального и коммерческого использования. В зависимости от спроса клиентов на продукт, цена будет регулироваться, чтобы соответствовать трендам рынка.

### Бизнес модель

Модель распространения продукта - ASP. Так, конечному пользователю будет предоставляться инсталлятор программного обеспечения и ключ активации к нему. Оплата будет происходить единожды за определенный период времени - 1 месяц/6 месяцев/1 год. В лицензиях для коммерческого использования количество активаций для одного ключа может различаться в зависимости от нужд компании, а в лицензиях для личного использования количество активаций будет ограничено заранее.

### Успешность продукта

Для измерения успешности проекта будет использоваться показатель количества мастерских, использующих продукт, а также выручка, получаемая с продаж.

### Налогообложение

Вопросы налогообложения компании будут решаться при помощи сторонней компании и третьих лиц, в случае, если компания разрастется, то можно нанять специалиста, который будет состоять в штате сотрудников и соответственно получать заработную плату.

# 4. Концептуальная модель

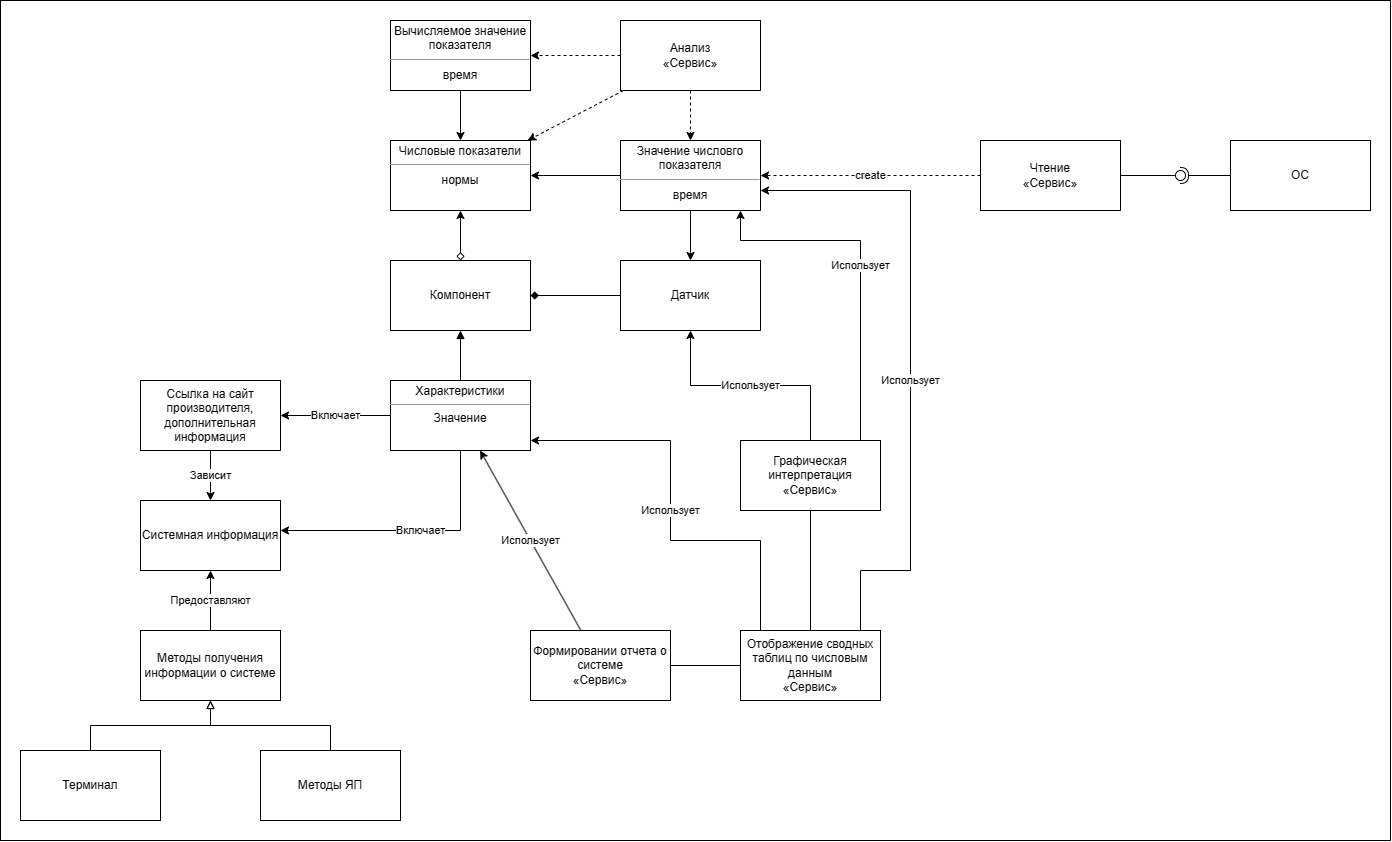


Рис. 1. Концептуальная модель

# 5. Use Case диаграмма

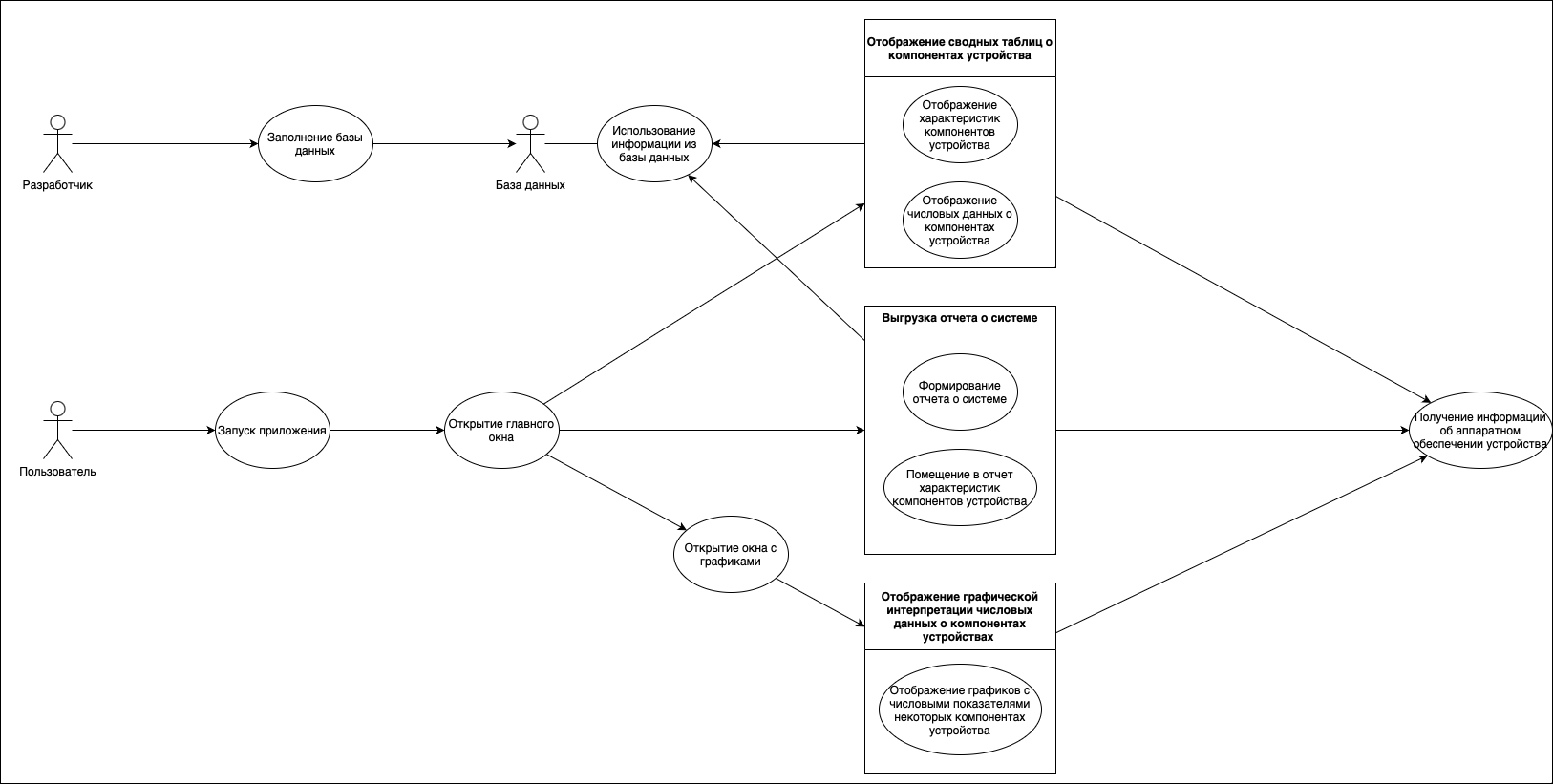


Рис. 2. Use case диаграмма

# 6. Анализ прецедентов

## Общие параметры

Основной исполнитель - пользователь

Заинтересованные лица и их требования:

1. Пользователь хочет быстро получить информацию с датчиков своего устройства, чтобы сделать выводы о состоянии устройства. Если ПК заведомо неисправен, то пользователь хочет узнать в чем может быть причина.
2. Разработчик (АлИнфоТех) имеет желание стать рентабельной компанией, которую будут узнавать. А также окупить все расходы на разработку и оптимизацию ПО.

Предусловия:

У пользователя должен быть ПК. Программа инициализировалась.

Постусловия:

Пользователь получил нужную ему информацию о своем устройстве.

## Прецедент 1 - получение информации с датчиков ПК для ознакомления с состоянием устройства.

### Основной успешный сценарий.

1. Пользователь запустил программу.
2. Пользователь выбрал часть ПК, информацию о которой хочет получить. (Возможно повторение)
3. Пользователь удостоверился, что с его ПК все в порядке.
4. Пользователь закрывает программу.

### Альтернативный неуспешный сценарий.

1. Пользователь запустил программу.
2. Пользователь выбрал часть ПК, информацию о которой хочет получить.
3. Пользователь не нашел датчика или информацию, которая его интересовала.
4. Пользователь закрывает программу.

## Прецедент 2 - получение информации с датчиков ПК для устранения неполадки.

Дополнительное предусловие: у пользователя возникли проблемы с ПК.

### Основной успешный сценарий.

1. У пользователя возникли проблемы с ПК.
2. Пользователь открывает программу.
3. Пользователь просмотрел всю доступную информацию о своем ПК.
4. Пользователь обнаружил значение датчика устройства, несоответствующее оптимальному.
5. Пользователь закрывает программу.
6. Пользователь устраняет проблему неисправной части ПК.
7. Пользователь запускает программу.
8. Пользователь проверяет датчик, значение которого было «плохим».
9. Пользователь удостоверяется, что проблема устранена.
10. Пользователь закрывает программу.

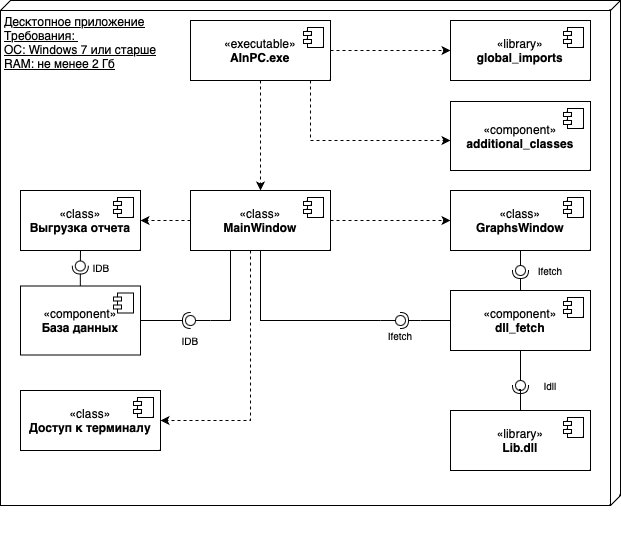
### Альтернативный неуспешный сценарий.

1. У пользователя возникли проблемы с ПК.
2. Пользователь открывает программу.
3. Пользователь просмотрел всю доступную информацию о своем ПК.
4. Пользователь не смог диагностировать проблему
5. Пользователь закрывает программу.

### Сценарий с недостоверной информацией.

1. Пользователь запустил программу.
2. Пользователь обнаружил датчик, значение которого не соответствует оптимальному.
3. Пользователь выключает программу.
4. Пользователь пытается устранить проблему.
5. Пользователь запускает программу.
6. Пользователь обнаруживает, что значение датчика все еще не соответствует оптимальному.
7. Пользователь считает, что информация является недостоверной.
8. Пользователь закрывает программу.

# 7. Диаграмма компонентов

Рис. 3. Диаграмма компонентов

## 1. Определение предметной области

Предметная область: ИС для диагностики аппаратного обеспечения компьютера.

Цель диаграммы: определение основных модулей приложения и выявление зависимостей между компонентами

Ограничения и объем: включить только основные компоненты, т.е. то, что будет на устройстве пользователя, исключая тех. поддержку

## 2. Выделение компонентов

AInPC.exe - основной исполняемый файл, который будет запускать главное окно - MainWindow.

MainWindow - главное окно программы, в котором будет выводиться как неизменяемая информация об устройстве, которая получается посредством ОС и базы данных, а также информация с датчиков оборудования. Данное окно инициирует доступ к остальным компонентам программы: графическая интерпретация данных, выгрузка отчета, запуск терминала.

GraphsWindow - окно программы, которое содержит в себе графическую интерпретацию данных с датчиков устройств.

База данных - этот компонент отвечает за хранение информации об устройствах ПК, которую нельзя получить посредством запроса в систему: ссылки на производителя, официальный драйвер и т.д.

Выгрузка отчета - данный компонент отвечает за формирование и выгрузку основной информации о системе в отдельный файл.

Доступ к терминалу - данный компонент обеспечивает пользователю доступ к терминалу, где тот сможет вводить необходимые ему дополнительные команды.

dll\_fetch - компонент, содержащий в себе набор функций, которые позволяют программе получать доступ к данным хранящимся в Lib.dll и представлять их в виде, удобном для работы на ЯП

Lib.dll - библиотека, получающая данные напрямую с датчиков устройства

global\_import - набор модулей и библиотек, используемых в рамках всего ПО

additional\_classes - компонент, содержащий классы, используемые в большой мере для дополнительных функций ПО, например, добавление гиперссылок

## 3. Определение интерфейсов

**Интерфейс для "Базы данных":**

**Название интерфейса:** IDB

**Описание:** интерфейс обеспечивает доступ к данным в базе данных

**Операции:** *SELECT FROM {table\_name} \* WHERE {hardware\_name = OS Output}*

**Интерфейс для "dll\_fetch":**

**Название интерфейса:** Ifetch

**Описание:** интерфейс обеспечивает нужный формат выходных данных для каждого инициирующего компонента

**Операции:** *initialize\_lib()*, *fetch\_stats()*, *parse\_sensor()*, *fetch\_stats\_for\_graphs()*, *parse\_sensor\_for\_graphs()*

**Интерфейс для "Lib.dll":**

**Название интерфейса:** Idll

**Описание:** интерфейс обеспечивает доступ к данным с датчиков устройства

**Операции:** *Hardware.Computer.open()*

# 8. Диаграмма классов

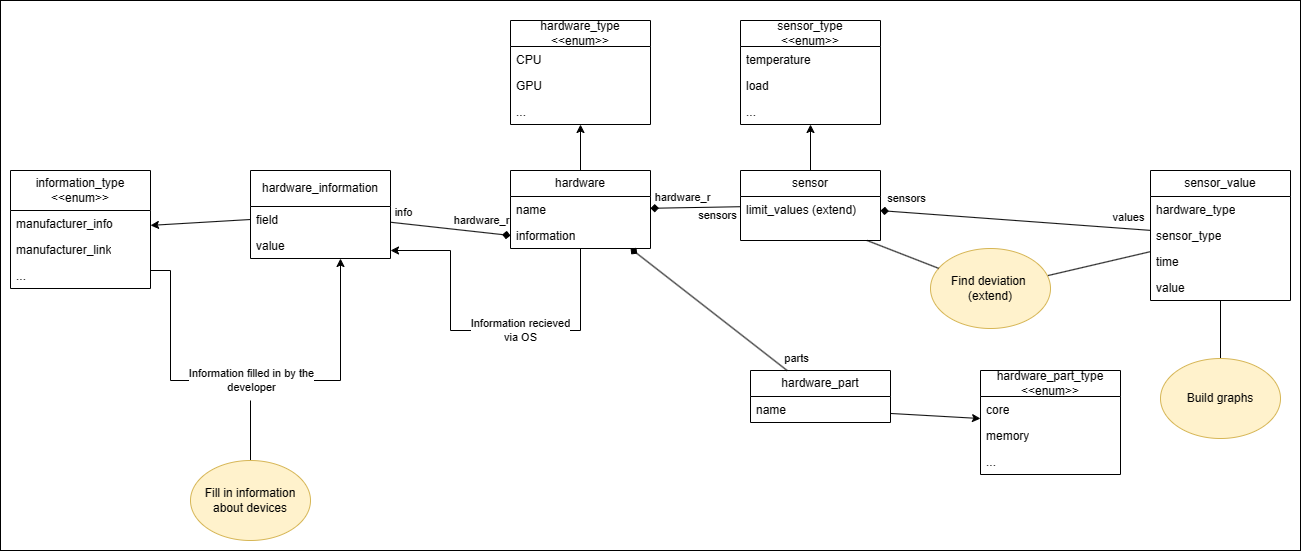


Рис. 5. Диаграмма классов

Таблица 6. «Сущность-наименование»

| Сущность | Наименование |
| --- | --- |
| hardware | Компонент |
| hardware\_information | Характеристики компонента |
| sensor | Сенсор |
| hardware\_part | Часть компонента |
| sensor\_value | Значение сенсора |
| information\_type | Тип информации |
| hardware\_type | Тип компонента |
| sensor\_type | Тип сенсора |
| hardware\_part\_type | Тип части компонента |

Таблица 7. «Сущность-ключ-…»

| **Сущность** | **Ключ** | **Атрибут** | **Обязательность** |
| --- | --- | --- | --- |
| hardware | + | name | + |
| hardware |  | information | + |
| hardware |  | info | + |
| hardware |  | sensors | + |
| hardware |  | parts | + |
| hardware\_information | + | field | + |
| hardware\_information |  | value | + |
| hardware\_information |  | hardware\_r | + |
| sensor |  | limit\_values(extend) |  |
| sensor |  | values | + |
| sensor |  | hardware\_r | + |
| hardware\_part | + | name | + |
| hardware\_part |  | hardware\_r | + |
| sensor\_value | + | hardware\_type | + |
| sensor\_value | + | sensor\_type | + |
| sensor\_value |  | time | + |
| sensor\_value |  | value | + |
| sensor\_value |  | Sensors | + |

| **Сущность** | **Тип** | **Отношение** | **Описание** |
| --- | --- | --- | --- |
| hardware | String |  | Наименование каждого компонента системы |
| hardware | String |  | Информация о каждом компоненте системы |
| hardware | hardware\_information[ ] | 1..\* |  |
| hardware | sensor[ ] | 1..\* |  |
| hardware | hardware\_parts[ ] | 1..\* |  |
| hardware\_information | String |  | Тип информации о компоненте |
| hardware\_information | String |  | Информация о компоненте из базы данных и ОС |
| hardware\_information | hardware |  |  |
| sensor | Float |  | Предельное значение для сенсора |
| sensor | sensor\_value[ ] | 1..\* |  |
| sensor | hardware |  |  |
| hardware\_part | String |  | Название части компонента |
| hardware\_part | hardware |  |  |
| sensor\_value | String |  | Тип информации о компоненте, содержащем сенсор |
| sensor\_value | String |  | Тип сенсора |
| sensor\_value | Float |  | Время, за которое идет опрос |
| sensor\_value | Float |  | Значение, передаваемое сенсором |
| sensor\_value | sensor |  |  |

# 9. Прототип UI

Дизайн интерфейса программы был разработан и визуализирован при помощи дизайнера форм Windows из Базового пакета среды разработки Visual Studio 2022.

Дизайн главного окна программы – открывающегося по умолчанию при запуске ПО – приведен на Рис. 6.

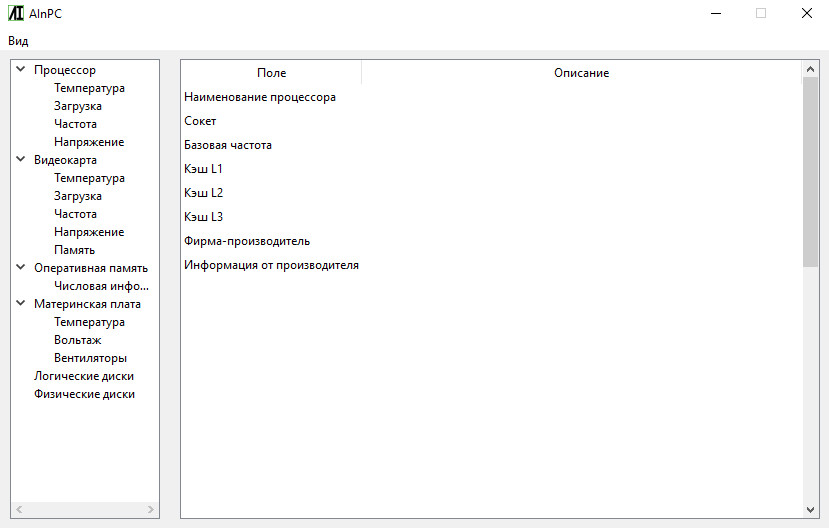


Рис. 6. Дизайн главного окна программы.

Разделы первого уровня (разделы - Процессор, Видеокарта, Оперативная память, Материнская плата) в основном подобны друг другу и содержат неизменяющуюся информацию. Они приведены на Рисунках 6, 7, 8 и 9, соответственно.

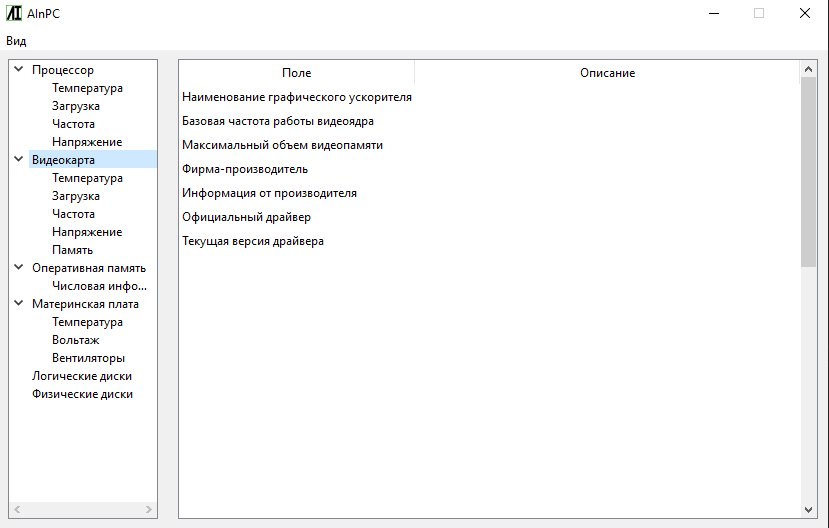


Рис. 7. Раздел «Видеокарта»

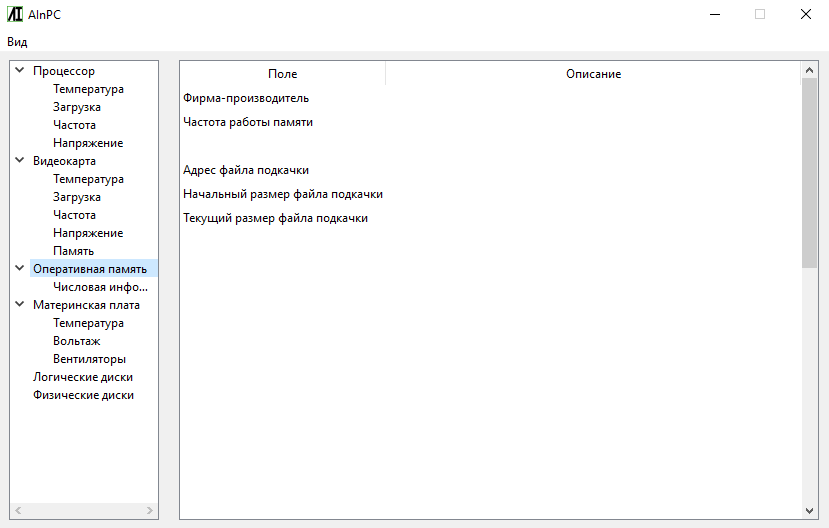


Рис. 8. Раздел «Оперативная память»

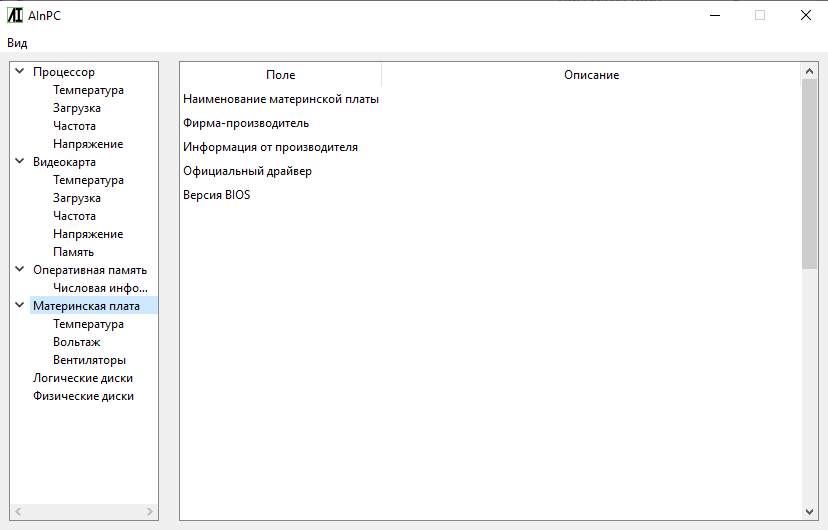


Рис. 9. Раздел «Материнская плата»

Разделы "Логические диски" и "Физические диски" также являются разделами первого уровня, однако содержат отличия от других разделов аналогичного уровня. Их дизайн требует адаптации таблицы в соответствии с функциональными требованиями. Дизайны разделов "Логические диски" и "Физические диски" приведены на Рисунках 10 и 11, соответственно.

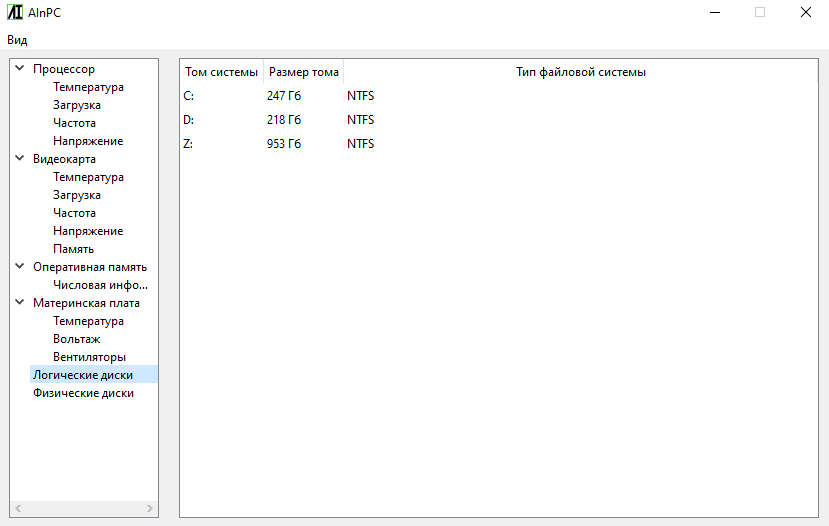


Рис. 10. Раздел «Логические диски»

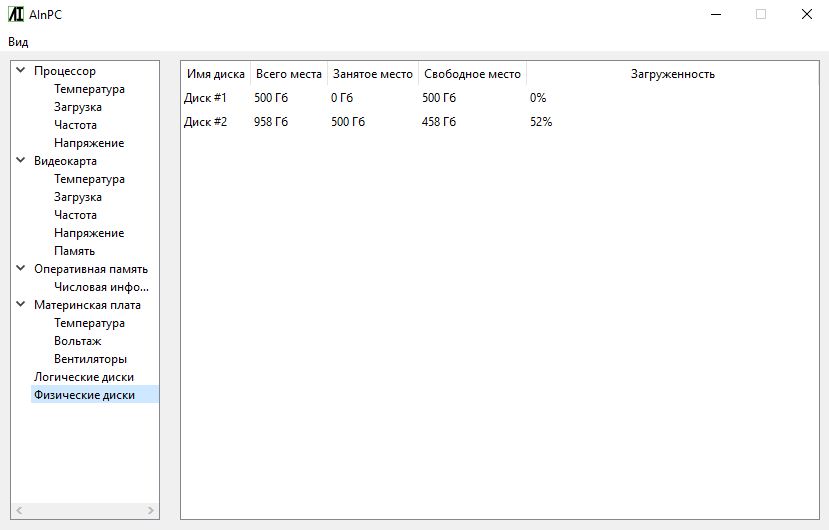


Рис. 11. Раздел «Физические диски»

Разделы второго уровня содержат числовые показатели, изменяющиеся во времени. К таким показателям относятся температура, загрузка, частота, напряжение, скорость вращения вентиляторов, использование памяти. Дизайн каждого раздела будет изменяться лишь от того, информацию о каком устройстве он на текущий момент отображает. Для примера на Рисунках 12, 13, 14, 15 приведена информация о температуре, загрузке, частоте работы каждого ядра процессора и напряжении, подаваемом на процессор, соответственно.

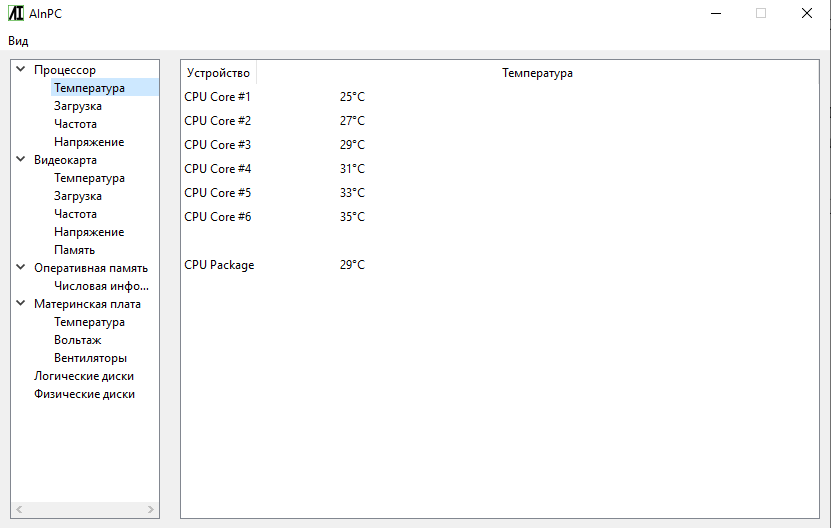


Рис. 13. Раздел «Процессор – Температура»

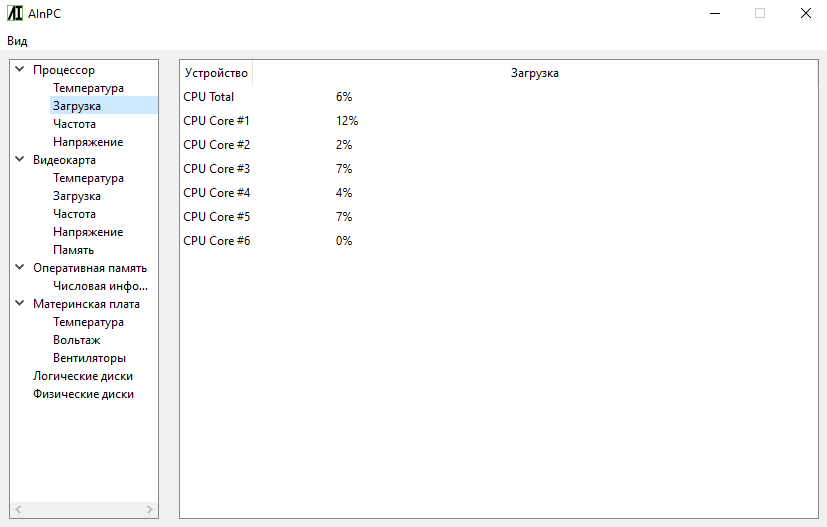


Рис. 14. Раздел «Процессор – Загрузка»

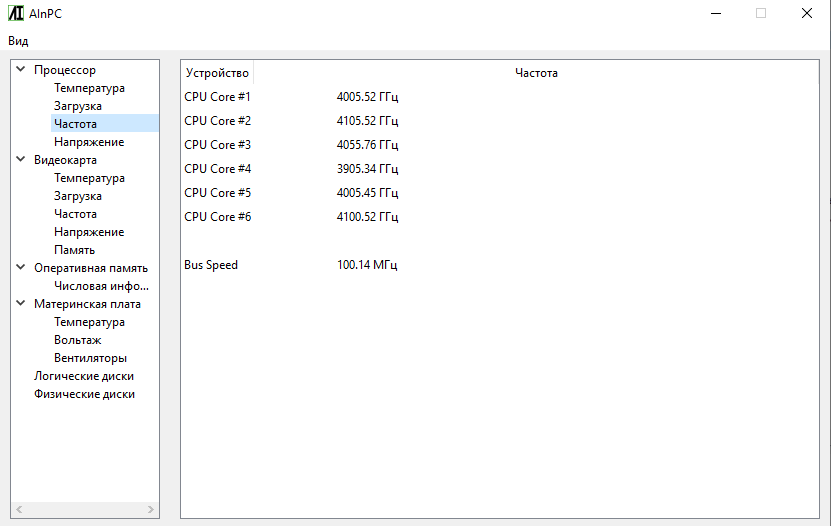


Рис. 15. Раздел «Процессор – Частота»

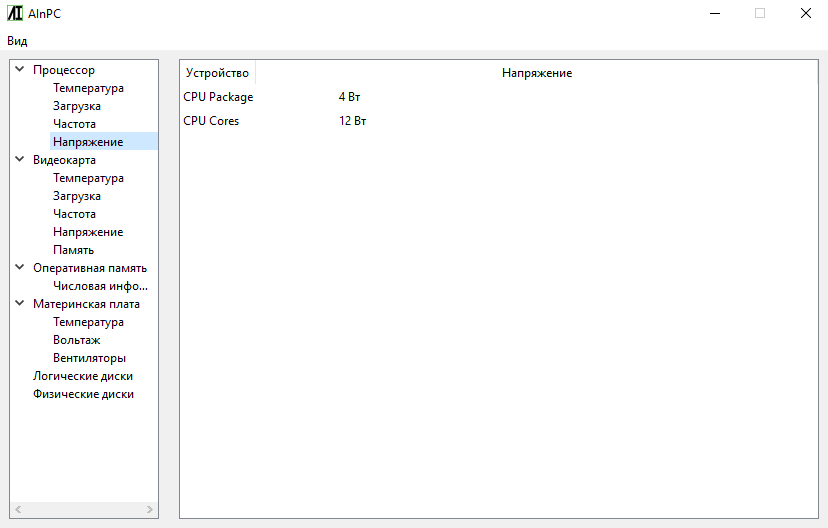


Рис. 16. Раздел «Процессор – Напряжение»

Примечание – на Рис. 6 – 16 в столбцах, идущих после первого, приведены данные, которые носят иллюстрирующий характер. В них указана информация, которая поможет получить максимально полное понимание об интерфейсе программы.

Помимо главного рабочего окна у программы существует второе окно (см. Рисунок 17), доступ к которому осуществляется путем нажатия кнопки "Вид" в верхнем меню (см. Рисунок 18), а затем клавиши "Графики".

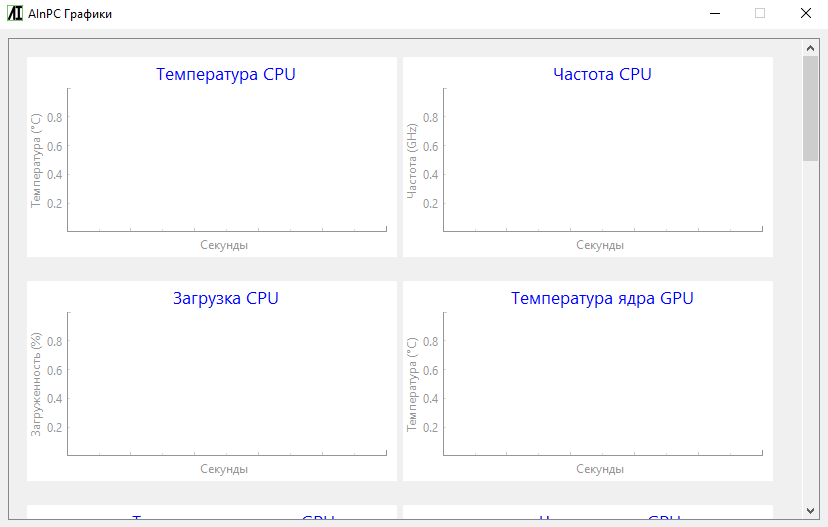


Рис. 17. Окно «Графики»

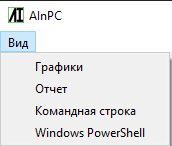


Рис. 18. Меню выбора действия

На Рис. 18. помимо клавиши "Графики" есть клавиша "Отчет", данная кнопка отвечает за выгрузку отчета, согласно функциональным требованиям.

Также, на Рисунке 18 есть клавиши "Командная строка" и "Windows PowerShell", которые выполняют свой функционал согласно тексту.

После оформления дизайна интерфейса он был реализован с использованием языка программирование Python и фреймворка PyQt6.

Полный код приведен в Приложении 1. Там же приведены комментарии ко всем частям программного кода.

# Приложение

## Приложение 1. Реализация интерфейса.

**#Импортируем необходимые для работы библиотеки**

import sys, pyqtgraph

from PyQt6.QtCore import Qt, QSize

from PyQt6.QtWidgets import QScrollArea, QAbstractItemView, QHeaderView, QTreeWidgetItem, QTreeWidget, QHBoxLayout, QTableWidgetItem, QTableWidget,QApplication, QMainWindow, QWidget, QVBoxLayout

from PyQt6.QtGui import QPalette, QIcon, QAction, QColor

**#Создаем главное рабочее окно**

class MainWindow(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

#**Настраиваем вид главного окна**

        self.setWindowTitle("AInPC")

        self.setFixedSize(QSize(830, 500))

        self.setWindowIcon(QIcon("main\_icon.png"))

        palette = QPalette()

        palette.setColor(QPalette.ColorRole.Window, QColor(192, 192, 192))

**#Создаем и настраиваем верхнее меню**

        menubar = self.menuBar()

        file\_menu = menubar.addMenu("Вид")

        menubar\_height = self.menuBar().height()

        open\_graphs = QAction("Графики", self)

        open\_report = QAction("Отчет", self)

        open\_cmd = QAction("Командная строка", self)

        open\_powershell = QAction("Windows PowerShell", self)

        file\_menu.addAction(open\_graphs)

        file\_menu.addAction(open\_report)

        file\_menu.addAction(open\_cmd)

        file\_menu.addAction(open\_powershell)

**#«Подключаем» реакцию программы на нажатие клавиши открытия окна с графиками**

        open\_graphs.triggered.connect(self.open\_graphs\_window)

**#Создаем и настраиваем левую часть окна - дерево, в котором содержится меню выбора части ПК**

        self.tree\_widget = QTreeWidget(self)

        self.tree\_widget.setGeometry(10, menubar\_height, 150, self.height() - menubar\_height - 10)

        self.tree\_widget.setHeaderHidden(True)

self.tree\_widget.setHorizontalScrollBarPolicy(Qt.ScrollBarPolicy.ScrollBarAlwaysOn)

**#Заполняем дерево «ветвями», которые показывают, какую информацию пользователь может**

#**получить**

        self.setup\_tree()

**#«Подключаем» реакцию программы на выбор пользователем другой «ветви»**

self.tree\_widget.itemSelectionChanged.connect(self.on\_item\_selected)

**#Создаем и настраиваем правую часть окна – таблицу, где отображается информация о выбранном**

**#комплектующем**

        self.table\_widget = QTableWidget(self)

self.table\_widget.setGeometry(self.tree\_widget.width() + 20 + 10, menubar\_height, self.width() –

self.tree\_widget.width() - 20 - 10 - 10, self.height() - menubar\_height - 10)

        self.table\_widget.verticalHeader().setVisible(False)

        self.table\_widget.setHorizontalScrollBarPolicy(Qt.ScrollBarPolicy.ScrollBarAlwaysOff)

        self.table\_widget.verticalHeader().setVisible(False)

        self.table\_widget.setShowGrid(False)

        self.table\_widget.horizontalHeader().setStretchLastSection(True)

        self.table\_widget.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeMode.ResizeToContents)

        self.table\_widget.verticalHeader().setDefaultSectionSize(1)

        self.table\_widget.setSelectionMode(QAbstractItemView.SelectionMode.SingleSelection)

**#Заполняем правую часть окна данными, которые должны появиться самыми первыми**

        self.initialize\_table(None, "Процессор")

**#Отображаем главное окно**

        self.show()

**#Функция, служащая для выравнивания всех ячеек в таблице справа**

    def fix\_table(self):

        for row in range(self.table\_widget.rowCount()):

            for col in range(self.table\_widget.columnCount()):

                if (self.table\_widget.rowSpan(row, col) > 1) or (self.table\_widget.columnSpan(row, col) > 1):

                    self.table\_widget.setSpan(row, col, 1, 1)

        self.table\_widget.clearContents()

**#Функция, заполняющая дерево в левой части экрана всеми «ветвями»**

    def setup\_tree(self):

        root\_item\_cpu = QTreeWidgetItem(self.tree\_widget, ["Процессор"])

        child\_item1\_cpu = QTreeWidgetItem(root\_item\_cpu, ["Температура"])

        child\_item2\_cpu = QTreeWidgetItem(root\_item\_cpu, ["Загрузка"])

        сhild\_item3\_cpu = QTreeWidgetItem(root\_item\_cpu, ["Частота"])

        child\_item4\_cpu = QTreeWidgetItem(root\_item\_cpu, ["Напряжение"])

        root\_item\_gpu = QTreeWidgetItem(self.tree\_widget, ["Видеокарта"])

        child\_item1\_gpu = QTreeWidgetItem(root\_item\_gpu, ["Температура"])

        child\_item2\_gpu = QTreeWidgetItem(root\_item\_gpu, ["Загрузка"])

        сhild\_item3\_gpu = QTreeWidgetItem(root\_item\_gpu, ["Частота"])

        child\_item4\_gpu = QTreeWidgetItem(root\_item\_gpu, ["Напряжение"])

        child\_item5\_gpu = QTreeWidgetItem(root\_item\_gpu, ["Память"])

        root\_item\_ram = QTreeWidgetItem(self.tree\_widget, ["Оперативная память"])

        child\_item1\_ram = QTreeWidgetItem(root\_item\_ram, ["Числовая информация"])

        root\_item\_mb = QTreeWidgetItem(self.tree\_widget, ["Материнская плата"])

        child\_item1\_mb = QTreeWidgetItem(root\_item\_mb, ["Температура"])

        child\_item2\_mb = QTreeWidgetItem(root\_item\_mb, ["Вольтаж"])

        child\_item3\_mb = QTreeWidgetItem(root\_item\_mb, ["Вентиляторы"])

        root\_item\_logical\_disks = QTreeWidgetItem(self.tree\_widget, ["Логические диски"])

        root\_item\_physical\_disks = QTreeWidgetItem(self.tree\_widget, ["Физические диски"])

**#Открывает все элементы дерева**

        self.tree\_widget.expandAll()

**#Функция, считывающая какую, именно «ветвь» выбрал пользователь**

    def on\_item\_selected(self):

        selected\_item = self.tree\_widget.selectedItems()[0]

        selected\_text = selected\_item.text(0)

        root\_item = selected\_item.parent()

        if root\_item is not None:

            root\_text = root\_item.text(0)

        else:

            root\_text = None

**#Информация о выбранной «ветви» передается в функцию, инициализирующую новую таблицу**

        if selected\_item:

            self.initialize\_table(root\_text, selected\_text)

**#Функция, инициализирующая новую таблицу в зависимости от того, какую «ветвь» выбрал**

**#пользователь**

    def initialize\_table(self, root, text):

        self.fix\_table()

**#Если пользователь выбрал «ветвь» первого уровня, тогда нам необходимо получить базовую –**

**#неизменяющуюся – информацию, поэтому мы можем сразу заполнить таблицу**

        if not root:

            if text == "Процессор":

                column\_names = ["Поле", "Описание"]

                self.table\_widget.setRowCount(40)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("Наименование процессора")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("Сокет")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("Базовая частота")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("Кэш L1")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("Кэш L2")))

                self.table\_widget.setItem(5, 0, QTableWidgetItem(str("Кэш L3")))

                self.table\_widget.setItem(6, 0, QTableWidgetItem(str("Фирма-производитель")))

                self.table\_widget.setItem(7, 0, QTableWidgetItem(str("Информация от производителя")))

            if text == "Видеокарта":

                column\_names = ["Поле", "Описание"]

                self.table\_widget.setRowCount(40)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("Наименование графического ускорителя")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("Базовая частота работы видеоядра")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("Максимальный объем видеопамяти")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("Фирма-производитель")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("Информация от производителя")))

                self.table\_widget.setItem(5, 0, QTableWidgetItem(str("Официальный драйвер")))

                self.table\_widget.setItem(6, 0, QTableWidgetItem(str("Текущая версия драйвера")))

            if text == "Оперативная память":

                column\_names = ["Поле", "Описание"]

                self.table\_widget.setRowCount(40)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("Фирма-производитель")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("Частота работы памяти")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("Адрес файла подкачки")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("Начальный размер файла подкачки")))

                self.table\_widget.setItem(5, 0, QTableWidgetItem(str("Текущий размер файла подкачки")))

            if text == "Материнская плата":

                column\_names = ["Поле", "Описание"]

                self.table\_widget.setRowCount(40)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("Наименование материнской платы")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("Фирма-производитель")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("Информация от производителя")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("Официальный драйвер")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("Версия BIOS")))

            if text == "Логические диски":

                column\_names = ["Том системы", "Размер тома", "Тип файловой системы"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(3)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("C:")))

                self.table\_widget.setItem(0, 1, QTableWidgetItem(str("247 Гб")))

                self.table\_widget.setItem(0, 2, QTableWidgetItem(str("NTFS")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("D:")))

                self.table\_widget.setItem(1, 1, QTableWidgetItem(str("218 Гб")))

                self.table\_widget.setItem(1, 2, QTableWidgetItem(str("NTFS")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("Z:")))

                self.table\_widget.setItem(2, 1, QTableWidgetItem(str("953 Гб")))

                self.table\_widget.setItem(2, 2, QTableWidgetItem(str("NTFS")))

            if text == "Физические диски":

                column\_names = ["Имя диска", "Всего места", "Занятое место", "Свободное место", "Загруженность"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(5)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("Диск #1")))

                self.table\_widget.setItem(0, 1, QTableWidgetItem(str("500 Гб")))

                self.table\_widget.setItem(0, 2, QTableWidgetItem(str("0 Гб")))

                self.table\_widget.setItem(0, 3, QTableWidgetItem(str("500 Гб")))

                self.table\_widget.setItem(0, 4, QTableWidgetItem(str("0%")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("Диск #2")))

                self.table\_widget.setItem(1, 1, QTableWidgetItem(str("958 Гб")))

                self.table\_widget.setItem(1, 2, QTableWidgetItem(str("500 Гб")))

                self.table\_widget.setItem(1, 3, QTableWidgetItem(str("458 Гб")))

                self.table\_widget.setItem(1, 4, QTableWidgetItem(str("52%")))

**#Если пользователь выбрал «ветвь» второго уровня, тогда мы сначала смотрим на то, какое значение**

**#имеет ветвь первого уровня, а затем смотрим значение «ветви» второго уровня, и создаем нужную**

**#таблицу**

        if root == "Процессор":

            if text == "Температура":

                column\_names = ["Устройство", "Температура"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("CPU Core #1")))

                self.table\_widget.setItem(0, 1, QTableWidgetItem(str("\t25\u00B0C")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("CPU Core #2")))

                self.table\_widget.setItem(1, 1, QTableWidgetItem(str("\t27\u00B0C")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("CPU Core #3")))

                self.table\_widget.setItem(2, 1, QTableWidgetItem(str("\t29\u00B0C")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("CPU Core #4")))

                self.table\_widget.setItem(3, 1, QTableWidgetItem(str("\t31\u00B0C")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("CPU Core #5")))

                self.table\_widget.setItem(4, 1, QTableWidgetItem(str("\t33\u00B0C")))

                self.table\_widget.setItem(5, 0, QTableWidgetItem(str("CPU Core #6")))

                self.table\_widget.setItem(5, 1, QTableWidgetItem(str("\t35\u00B0C")))

                self.table\_widget.setItem(6, 0, QTableWidgetItem(str("")))

                self.table\_widget.setItem(7, 0, QTableWidgetItem(str("CPU Package")))

                self.table\_widget.setItem(7, 1, QTableWidgetItem(str("\t29\u00B0C")))

            elif text == "Загрузка":

                column\_names = ["Устройство", "Загрузка"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("CPU Total")))

                self.table\_widget.setItem(0, 1, QTableWidgetItem(str("\t6%")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("CPU Core #1")))

                self.table\_widget.setItem(1, 1, QTableWidgetItem(str("\t12%")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("CPU Core #2")))

                self.table\_widget.setItem(2, 1, QTableWidgetItem(str("\t2%")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("CPU Core #3")))

                self.table\_widget.setItem(3, 1, QTableWidgetItem(str("\t7%")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("CPU Core #4")))

                self.table\_widget.setItem(4, 1, QTableWidgetItem(str("\t4%")))

                self.table\_widget.setItem(5, 0, QTableWidgetItem(str("CPU Core #5")))

                self.table\_widget.setItem(5, 1, QTableWidgetItem(str("\t7%")))

                self.table\_widget.setItem(6, 0, QTableWidgetItem(str("CPU Core #6")))

                self.table\_widget.setItem(6, 1, QTableWidgetItem(str("\t0%")))

            elif text == "Частота":

                column\_names = ["Устройство", "Частота"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("CPU Core #1")))

                self.table\_widget.setItem(0, 1, QTableWidgetItem(str("\t4005.52 ГГц")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("CPU Core #2")))

                self.table\_widget.setItem(1, 1, QTableWidgetItem(str("\t4105.52 ГГц")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("CPU Core #3")))

                self.table\_widget.setItem(2, 1, QTableWidgetItem(str("\t4055.76 ГГц")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("CPU Core #4")))

                self.table\_widget.setItem(3, 1, QTableWidgetItem(str("\t3905.34 ГГц")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("CPU Core #5")))

                self.table\_widget.setItem(4, 1, QTableWidgetItem(str("\t4005.45 ГГц")))

                self.table\_widget.setItem(5, 0, QTableWidgetItem(str("CPU Core #6")))

                self.table\_widget.setItem(5, 1, QTableWidgetItem(str("\t4100.52 ГГц")))

                self.table\_widget.setItem(6, 0, QTableWidgetItem(str("")))

                self.table\_widget.setItem(7, 0, QTableWidgetItem(str("Bus Speed")))

                self.table\_widget.setItem(7, 1, QTableWidgetItem(str("\t100.14 МГц")))

            elif text == "Напряжение":

                column\_names = ["Устройство", "Напряжение"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("CPU Package")))

                self.table\_widget.setItem(0, 1, QTableWidgetItem(str("\t4 Вт")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("CPU Cores")))

                self.table\_widget.setItem(1, 1, QTableWidgetItem(str("\t12 Вт")))

        elif root == "Видеокарта":

            if text == "Температура":

                column\_names = ["Устройство", "Температура"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("GPU Core")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("GPU Memory")))

                not\_item = QTableWidgetItem(str("Отсутствует датчик температуры на видеокарте"))

                not\_item.setTextAlignment(Qt.AlignmentFlag.AlignHCenter)

                self.table\_widget.setItem(1, 1, not\_item)

            elif text == "Загрузка":

                column\_names = ["Устройство", "Загрузка"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("GPU Core")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("GPU Frame Buffer")))

            elif text == "Частота":

                column\_names = ["Устройство", "Частота"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("GPU Core")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("GPU Memory")))

            elif text == "Напряжение":

                column\_names = ["Устройство", "Напряжение"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("GPU Power")))

            elif text == "Память":

                column\_names = ["Устройство", "Загруженность"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("GPU Memory Total")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("GPU Memory Used")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("GPU Memory Free")))

        elif root == "Оперативная память":

            if text == "Числовая информация":

                column\_names = ["Устройство", "Информация"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("RAM Load")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("RAM Used")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("RAM Free")))

        elif root == "Материнская плата":

            if text == "Температура":

                column\_names = ["Устройство", "Температура"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("Сенсор #1")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("Сенсор #2")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("Сенсор #3")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("Сенсор #4")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("Сенсор #5")))

                self.table\_widget.setItem(5, 0, QTableWidgetItem(str("Сенсор #6")))

            elif text == "Вольтаж":

                column\_names = ["Устройство", "Вольтаж"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("CPU VCore")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("Сенсор #2")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("AVCC")))

                self.table\_widget.setItem(3, 0, QTableWidgetItem(str("3VCC")))

                self.table\_widget.setItem(4, 0, QTableWidgetItem(str("Сенсор #5")))

                self.table\_widget.setItem(5, 0, QTableWidgetItem(str("Сенсор #6")))

                self.table\_widget.setItem(6, 0, QTableWidgetItem(str("Сенсор #7")))

                self.table\_widget.setItem(7, 0, QTableWidgetItem(str("3VSB")))

                self.table\_widget.setItem(8, 0, QTableWidgetItem(str("VBAT")))

                self.table\_widget.setItem(9, 0, QTableWidgetItem(str("VTT")))

                self.table\_widget.setItem(10, 0, QTableWidgetItem(str("Сенсор #11")))

                self.table\_widget.setItem(11, 0, QTableWidgetItem(str("Сенсор #12")))

                self.table\_widget.setItem(12, 0, QTableWidgetItem(str("Сенсор #13")))

                self.table\_widget.setItem(13, 0, QTableWidgetItem(str("Сенсор #14")))

                self.table\_widget.setItem(14, 0, QTableWidgetItem(str("Сенсор #15")))

            elif text == "Вентиляторы":

                column\_names = ["Устройство", "Скорость вращения"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(2)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("Вентилятор #1")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("Вентилятор #2")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("Вентилятор #3")))

        elif root == "Логические диски":

            if text == "Логические":

                column\_names = ["Том системы", "Размер тома", "Тип файловой системы"]

                self.table\_widget.setRowCount(15)

                self.table\_widget.setColumnCount(3)

                self.table\_widget.setHorizontalHeaderLabels(column\_names)

                self.table\_widget.setItem(0, 0, QTableWidgetItem(str("C:")))

                self.table\_widget.setItem(0, 1, QTableWidgetItem(str("247 Гб")))

                self.table\_widget.setItem(0, 2, QTableWidgetItem(str("NTFS")))

                self.table\_widget.setItem(1, 0, QTableWidgetItem(str("D:")))

                self.table\_widget.setItem(1, 1, QTableWidgetItem(str("218 Гб")))

                self.table\_widget.setItem(1, 2, QTableWidgetItem(str("NTFS")))

                self.table\_widget.setItem(2, 0, QTableWidgetItem(str("Z:")))

                self.table\_widget.setItem(2, 1, QTableWidgetItem(str("953 Гб")))

                self.table\_widget.setItem(2, 2, QTableWidgetItem(str("NTFS")))

**#Функция, открывающая окно с графиками**

    def open\_graphs\_window(self):

        self.graphs\_window = GraphsWindow()

        self.graphs\_window.show()

**#Создаем окно с графиками**

class GraphsWindow(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

**#Настраиваем окно с графиками**

        self.setWindowTitle("AInPC Графики")

        self.setWindowIcon(QIcon("main\_icon.png"))

        self.setFixedSize(QSize(830, 500))

        palette = QPalette()

        palette.setColor(QPalette.ColorRole.Window, QColor(192, 192, 192))

        central\_widget = QWidget()

        layout = QVBoxLayout()

**#Заполняем данные о том, какие графики нам необходимо построить**

        plots\_names\_list = ["Температура CPU", "Частота CPU", "Загрузка CPU", "Температура ядра GPU", "Температура памяти GPU", "Частота ядра GPU", "Частота памяти GPU", "Использование памяти GPU", "Использование RAM", "Температура сенсора #1 на МП", "Температура сенсора #2 на МП", "Температура сенсора #3 на МП", "Температура сенсора #4 на МП", "Температура сенсора #5 на МП", "Температура сенсора #6 на МП", "Скорость вращения вентилятора #1", "Скорость вращения вентилятора #2", "Скорость вращения вентилятора #3"]

**#Создаем область, в которой будет ползунок для прокрутки экрана**

        scroll\_area = QScrollArea(self)

        scroll\_widget = QWidget(scroll\_area)

        scroll\_layout = QVBoxLayout(scroll\_widget)

        rows\_of\_graphs = int(len(plots\_names\_list)/2) + len(plots\_names\_list)%2

        self.graph\_widgets = []

        num = 0

**#В каждой строке создаем по 2 графика из списка**

        for i in range(0, rows\_of\_graphs):

            graph\_row\_widget = QWidget()

            row\_layout = QHBoxLayout()

            for j in range(2):

**#Задаем настройки каждого графика – размер, цвет, подписи на осях и т.д.**

                graph\_widget = pyqtgraph.PlotWidget()

                graph\_widget.setMouseEnabled(x = False, y = False)

                graph\_widget.setBackground("w")

                graph\_widget.setFixedWidth(370)

                graph\_widget.plotItem.setFixedWidth(360)

                graph\_widget.setFixedHeight(200)

                graph\_widget.plotItem.setFixedHeight(195)

                graph\_widget.plotItem.setTitle(plots\_names\_list[num], color = "b", size = "13pt")

                if "Температура" in plots\_names\_list[num]:

                    graph\_widget.plotItem.setLabel("left", "Температура (°C)")

                elif "Загрузка" in plots\_names\_list[num]:

                    graph\_widget.plotItem.setLabel("left", "Загруженность (%)")

                elif "Частота CPU" in plots\_names\_list[num]:

                    graph\_widget.plotItem.setLabel("left", "Частота (GHz)")

                elif "Частота" in plots\_names\_list[num] and "GPU" in plots\_names\_list[num]:

                    graph\_widget.plotItem.setLabel("left", "Частота (MHz)")

                elif "Использование памяти GPU" in plots\_names\_list[num] or "Использование RAM" in plots\_names\_list[num]:

                    graph\_widget.plotItem.setLabel("left", "Память (Gb)")

                elif "Скорость вращения вентилятора" in plots\_names\_list[num]:

                    graph\_widget.plotItem.setLabel("left", "Обороты в минуту (ОБМ)")

                graph\_widget.plotItem.setLabel("bottom", "Секунды")

                x\_axis = graph\_widget.getAxis("bottom")

                x\_axis.setStyle(showValues = False)

                pen = pyqtgraph.mkPen(color = (255, 0, 0))

                plot = graph\_widget.plot([], [], pen = pen)

                self.graph\_widgets.append((graph\_widget, plot))

                row\_layout.addWidget(graph\_widget)

                num += 1

            graph\_row\_widget.setLayout(row\_layout)

            scroll\_layout.addWidget(graph\_row\_widget)

**#Добавляем график на рабочую область**

        scroll\_widget.setLayout(scroll\_layout)

        scroll\_area.setWidget(scroll\_widget)

        layout.addWidget(scroll\_area)

**#Отображаем рабочую область в окно**

        central\_widget.setLayout(layout)

        self.setCentralWidget(central\_widget)

**#Функция, создающая главное окно и запускающая его**

def main():

    app = QApplication(sys.argv)

    window = MainWindow()

    sys.exit(app.exec())

**#Вызываем функцию, которая создает главное окно**

if \_\_name\_\_ == "\_\_main\_\_":

    main()