БУ ВО

«СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ХАНТЫ-МАНСИЙСКОГО АВТОНОМНОГО ОКРУГА - ЮГРЫ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра информатики и вычислительной техники

УТВЕРЖДАЮ:

Заведующий кафедрой ИВТ

Канд. физ.-мат. наук, доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_С.А.Лысенкова

" \_\_ " \_\_\_\_\_\_\_\_\_\_\_ 2024г.

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «**Методы и средства проектирования**

**информационных систем**»

бакалавра по направлению

09.03.02 - Информационные системы и технологии

на тему ***Информационная система для диагностики аппаратного***

***обеспечения компьютера***

Выполнил: бакалавр группы: 607-01

Веревкин Алексей Ильич

(фамилия, имя, отчество)

по специальности 09.03.02 - Информационные системы и технологии

Руководитель:

преподаватель Столбов Д.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Сургут – 2024

**Реферат**

Курсовой проект содержит 60 страниц, 44 рисунка, 3 таблицы, 6 источников, 3 приложения.

**Информационная система для диагностики аппаратной части компьютера.**

Ключевые слова: предметная область, Use Case диаграмма, диаграмма классов, диаграмма компонентов, BPMN, UML.

Цель работы: проектирование информационной системы, служащей для упрощения процесса диагностики персонального компьютера.

Основные результаты:

* проведен анализ предметной области и составлено ее описание;
* сделан подробный обзор аналогов, включающий в себя выделение положительных и отрицательных сторон иностранных продуктов-конкурентов;
* выдвинуты требования к итоговой системе, которые включают в себя, как функциональные требования, так и требования к различным видам обеспечения;
* спроектирована система с использованием различного вида диаграмм, составленных с помощью языка моделирования UML.
* разработана информационная система с учетом всех требований и согласно проектному видению.

Оглавление

[Перечень сокращений 5](#_Toc159464329)

[Введение 7](#_Toc159464330)

[Основная часть 8](#_Toc159464331)

[1. Постановка задачи 8](#_Toc159464332)

[2. Описание предметной области 9](#_Toc159464333)

[2.1. Заинтересованные лица 9](#_Toc159464334)

[2.2. Модель предметной области 9](#_Toc159464335)

[3. Обзор аналогов 14](#_Toc159464336)

[3.1. AIDA64 14](#_Toc159464337)

[3.2. HWInfo 16](#_Toc159464338)

[3.3. OCCT 18](#_Toc159464339)

[3.4. CPU-Z/GPU-Z 19](#_Toc159464340)

[3.5. SiSoftware Sandra 22](#_Toc159464341)

[4. Требования к системе 25](#_Toc159464342)

[4.1. Функциональные требования 25](#_Toc159464343)

[4.2. Требования по видам обеспечения 27](#_Toc159464344)

[4.2.1. Требования к математическому обеспечению 27](#_Toc159464345)

[4.2.2. Требования к алгоритмическому обеспечению 27](#_Toc159464346)

[4.2.3. Требования к информационному обеспечению 29](#_Toc159464347)

[4.2.4. Требования к информационной безопасности 29](#_Toc159464348)

[4.2.5. Требования к лингвистическому обеспечению 29](#_Toc159464349)

[4.2.6. Требования к программному обеспечению 29](#_Toc159464350)

[4.2.7. Требования к техническому обеспечению 30](#_Toc159464351)

[4.2.8. Требования к организационному обеспечению 30](#_Toc159464352)

[4.2.9. Требования к методическому обеспечению 30](#_Toc159464353)

[5. Проектирование системы 31](#_Toc159464354)

[5.1. SWOT – анализ 31](#_Toc159464355)

[5.2. Диаграмма вариантов использования 31](#_Toc159464356)

[5.3. Анализ прецедентов 32](#_Toc159464357)

[5.4. Диаграммы процессов BPMN 34](#_Toc159464358)

[5.5. Диаграмма классов 36](#_Toc159464359)

[5.6. Информационная модель данных 37](#_Toc159464360)

[5.7. Диаграмма компонентов 37](#_Toc159464361)

[5.8. Прототип UI 39](#_Toc159464362)

[6. Разработка информационной системы 47](#_Toc159464363)

[6.1. Backlog 47](#_Toc159464364)

[6.2. Функциональные блоки программы 47](#_Toc159464365)

[6.3. Скриншоты UI 56](#_Toc159464366)

[6.4. Настройка системы 63](#_Toc159464367)

[6.5. Руководство пользователя 63](#_Toc159464368)

[Заключение 64](#_Toc159464369)

[Список источников 65](#_Toc159464370)

[Приложения 67](#_Toc159464371)

[Приложение 1. Концептуальная модель информационной системы 67](#_Toc159464372)

[Приложение 2. Диаграмма вариантов использования 68](#_Toc159464373)

[Приложение 3. Диаграмма классов 69](#_Toc159464374)

# Перечень сокращений

Таблица 1. Перечень сокращений

|  |  |  |
| --- | --- | --- |
| 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 |

# Введение

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 [1]. 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 [2]. При этом, как и любая другая электронная техника, эти устройства имеют свойство ломаться, а значит для предотвращения этих поломок необходимо вовремя обслуживать сам прибор, а в случае, если неполадка уже возникла, то нужно её устранить.

Уход иностранных компаний с Российского рынка, обозначенный ранее, оказал существенное влияние на сферу ремонта и обслуживания компьютерной техники. Помимо проблем с количеством комплектующих для ПК существуют проблемы с программным обеспечением для компьютерных мастерских и сервисов. Некоторые компании, например, отказываются предоставлять лицензии для коммерческого использования.

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 с использованием отечественных технологий, становится особенно актуальным.

# Основная часть

## 1. Постановка задачи

Цель ВКР: разработка информационной системы для диагностики аппаратного обеспечения компьютера.

Задачи ВКР:

1. Проанализировать рынок иностранных аналогов с аналогичным функционалом, чтобы выявить их преимущества и недостатки, с учетом которых будет вестись разработка ПО.
2. Сформулировать функциональные требования к ПО, которые бы соответствовали тенденциям и реалиям рыночной ниши компьютерных мастерских.
3. Реализовать программный продукт, который бы соответствовал сформулированным функциональным требованиям в пт. 2.

Цель информационной системы: обеспечение заинтересованных лиц информацией о компьютере, которая включает в себя справочную информацию о каждом компоненте, некоторую информацию с сенсоров компонентов в текстовой форме и форме графиков.

Задачи информационной системы:

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 терминал.

## 2. Описание предметной области

### 2.1. Заинтересованные лица

Таблица 2. Заинтересованные лица

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

### 2.2. Модель предметной области

Данная предметная область представляет собой обширное поле исследований и разработок, ориентированных на создание программного инструмента для анализа и предоставления информации о состоянии аппаратной части компьютера.

Главным образом эта область связана с основными компонентами компьютера, такими как процессор, оперативная память, жесткие диски, видеокарты и другие устройства. Для правильной диагностики состояния этих компонентов необходимо иметь глубокое понимание их работы, взаимодействия друг с другом, а также особенностей их функционирования в различных условиях использования.

Следующий аспект, связанный с разработкой программного обеспечения, включает в себя создание удобного пользовательского интерфейса для взаимодействия с системой. Этот интерфейс должен быть интуитивно понятным и предоставлять пользователю доступ к информации о состоянии компонентов компьютера в форме, которая позволит понять текущее состояние системы. Разработчики такой системы должны учитывать различные уровни технической осведомленности пользователей, чтобы обеспечить удобство и понятность интерфейса для всех категорий пользователей.

Также данная предметная область включает в себя исследование и разработку алгоритмов, способных эффективно считывать данные с различных датчиков и сенсоров компьютера. Эти алгоритмы должны быть оптимизированы для работы с разнообразными типами компонентов, учитывая их особенности и специфику работы. Они также должны обеспечивать корректную интерпретацию данных и формирование надёжных отчётов о состоянии каждого элемента аппаратной части компьютера.

Вопросы безопасности и конфиденциальности играет ключевую роль в разработке подобной системы. Поскольку она имеет доступ к информации о состоянии компьютера, необходимо обеспечить высокий уровень защиты данных от несанкционированного доступа и утечек. Это включает в себя применение современных методов шифрования, контроль доступа к информации и меры обеспечения конфиденциальности данных пользователей.

И если углубиться в область предсказания и предотвращения неполадок в работе ПК, то важную роль сыграет исследование и разработка методов обучения системы на основе собираемых данных. Это позволит системе стать более точной и эффективной в предоставлении информации о состоянии компонентов компьютера. Методы машинного обучения и анализа данных могут помочь улучшить систему, позволяя ей более точно адаптироваться к различным условиям работы и предоставлять более точные и полезные сведения о состоянии аппаратного обеспечения.

Перед началом процесса разработки необходимо провести структурный анализ предметной области. Можно использовать пояснительную модель. Данный вид моделей главным образом используется для объяснения и понимания терминов и процессов внутри темы. Она поможет в дальнейшем разработать саму программу, а также помочь в понимании сложных структур и предсказании их поведения [3] [4]. Пояснительная модель для диагностики аппаратной части компьютера в режиме реалтайм приведена на Рис. 1.1 – Рис. 1.3.

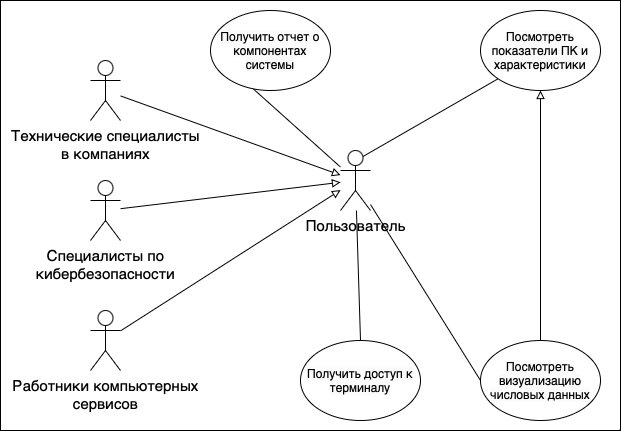


Рис. 1.1. Пояснительная модель. Функциональные области.

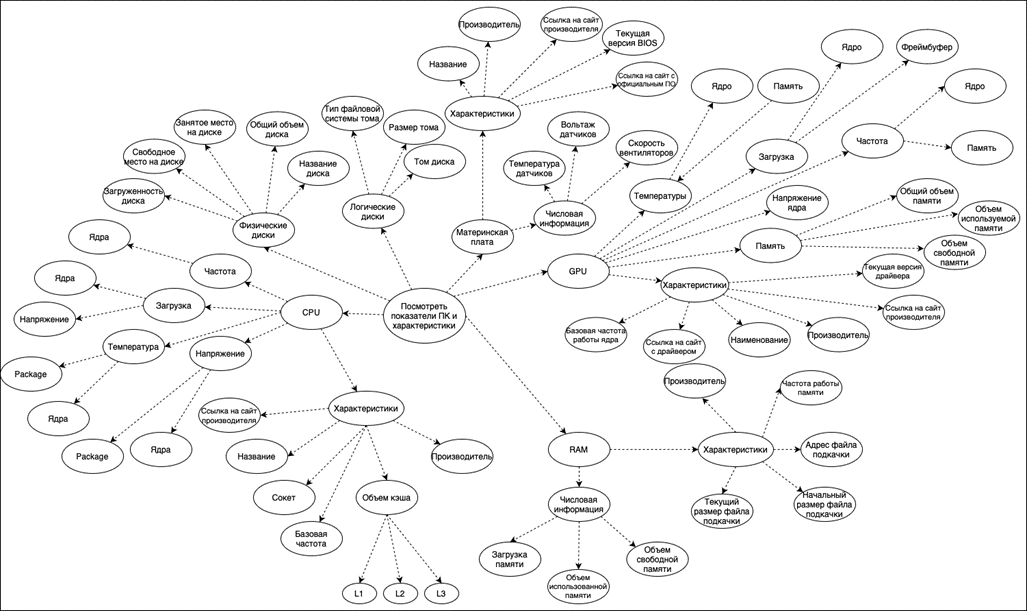


Рис. 1.2. Пояснительная модель. Вывод показателей ПК и характеристик.

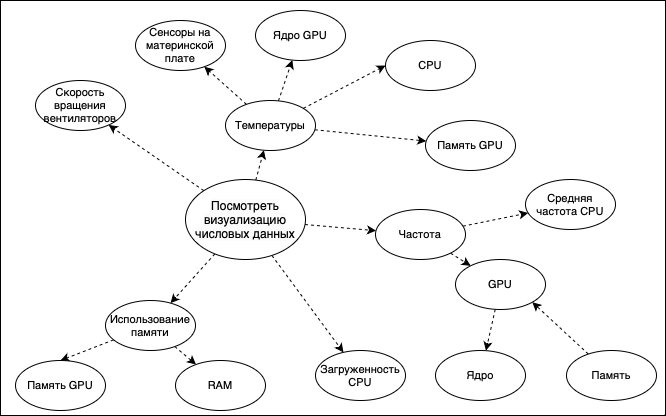


Рис. 1.3. Пояснительная модель. Вывод графиков.

На Рис. 1.1 показаны главные функциональные области программы - отображение сводных таблиц, включающих в себя как числовую информацию с датчиков компонентов устройства, так и характеристики данных компонентов, визуализация числовых данных в виде графиков, получение отчета о компонентах системы, а также получение доступа к терминалу. Помимо этого, на модели отображены основные предполагаемые пользователи:

* технические специалисты в компаниях;
* специалисты по кибербезопасности;
* работники компьютерных сервисов.

Данные группы людей столкнулись с трудностями, соответствующими их профессии, в диагностике своих ПК, поэтому они попадают под целевую аудиторию итогового продукта [5][6].

На Рис. 1.2 более конкретно обозначены числовые данные, которые будут использоваться программой, а также список характеристик каждого компонента.

На Рис. 1.3 приведен перечень данных, которые будут визуализированы в рамках графической интерпретации числовых данных.

Концептуальная модель представляет собой абстракцию предметной области, которая описывает основные понятия, их связи и основные атрибуты без уточнения технических деталей реализации. Она служит инструментом для понимания структуры и взаимодействия важных элементов в системе или предметной области. В качестве опорных пунктов для составления концептуальной модели выделяют:

1. Абстракция от технических деталей.
2. Понимание предметной области.
3. Взаимодействие с заказчиками и стейкхолдерами.
4. Основа для проектирования и разработки.
5. Избежание недопонимания и ошибок.

Модель для информационной системы приведена в Приложении 1. Она более четко отображает взаимодействия разных компонентов системы, описанных в рамках пояснительной модели.

## 3. Обзор аналогов

В качестве ориентира на примерное качество и функционал будут рассмотрены иностранные программные продукты, которые активно использовались и используются во многих сервисах и компьютерных мастерских. В результате анализа рынка было выделено следующее ПО:

1. AIDA64;
2. HWInfo;
3. OCCT;
4. CPU-Z/GPU-Z;
5. SiSoftware Sandra.

Далее будут подробно рассмотрены преимущества и недостатки данных продуктов для того, чтобы определить, какой функционал должен быть у итогового продукта, а также для того, чтобы выделить пункты, которые необходимо модернизировать в данном ПО.

### 3.1. AIDA64

Первые аналогом в представленном выше списке является AIDA64, данный программный продукт распространяется на бесплатном основании для персонального использования, но для того, чтобы использовать его в рамках бизнеса, необходимо приобретать лицензию. Интерфейс данного ПО представлен на Рис. 2.

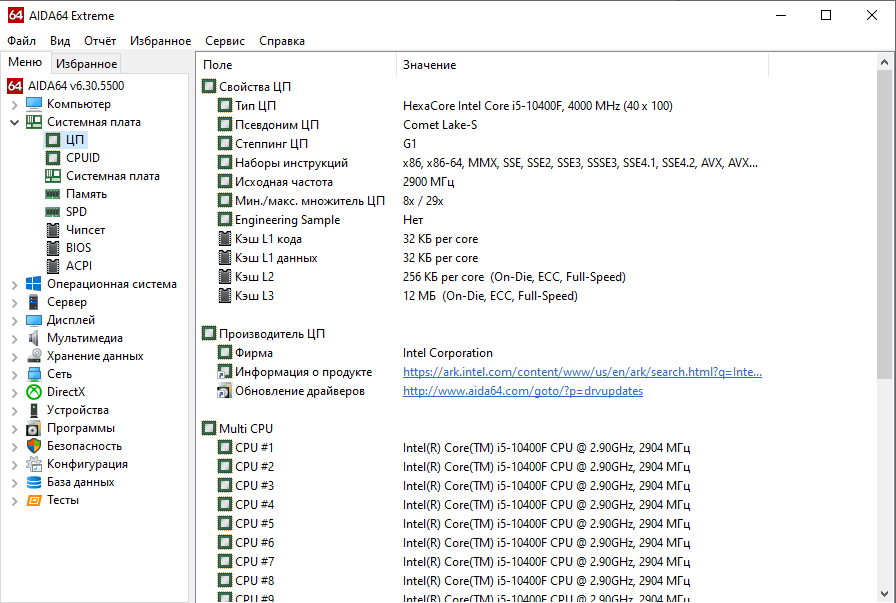


Рис. 2. Интерфейс программы AIDA64.

К достоинствам данной программы можно отнести следующие пункты:

ПО выдает подробную информацию об:

* установленных в системе устройствах, об их характеристиках и поддерживаемых ими наборах команд;
* установленном ПО, драйверах и других программах;
* имеющихся лицензиях.

Помимо информации о конфигурации ПЭВМ, программа способна проводить тесты, некоторые из них приведены ниже:

* тестирование оперативной памяти (чтение из памяти, запись в память, копирование в памяти, задержка памяти);
* тестирование процессора (CPU AES, CPU ZLib, FPU Julia и др.).

У данной программы также имеются недостатки, к которым можно отнести:

* отсутствие возможности тестирования видеокарты и ограниченный набор тестов для тестирования жестких дисков и SSD;
* 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 о ПЭВМ.

### 3.2. HWInfo

Следующим рассматриваемым аналогом является программа HWInfo, интерфейс которой представлен на Рис. 3., она является менее распространённой, нежели первый аналог, но в некотором кругу людей, занимающихся разгоном ПК (разгон - изменение некоторых штатных параметров работы компонентов ПЭВМ, вследствие чего увеличивается их производительность), данная программа получила большую известность. Данное явление связано со следующими достоинствами данного ПО:

* большое количество информации о всевозможных датчиках ПК, в том числе SSD, датчики на материнской плате и др.;
* возможность автоматического сканирования, обновления и возвращения к более старым версиям драйверов;
* наличие некоторой информации об Ethernet порте;
* возможность установить удаленное подключение с другим устройством, чтобы получать данные с него.

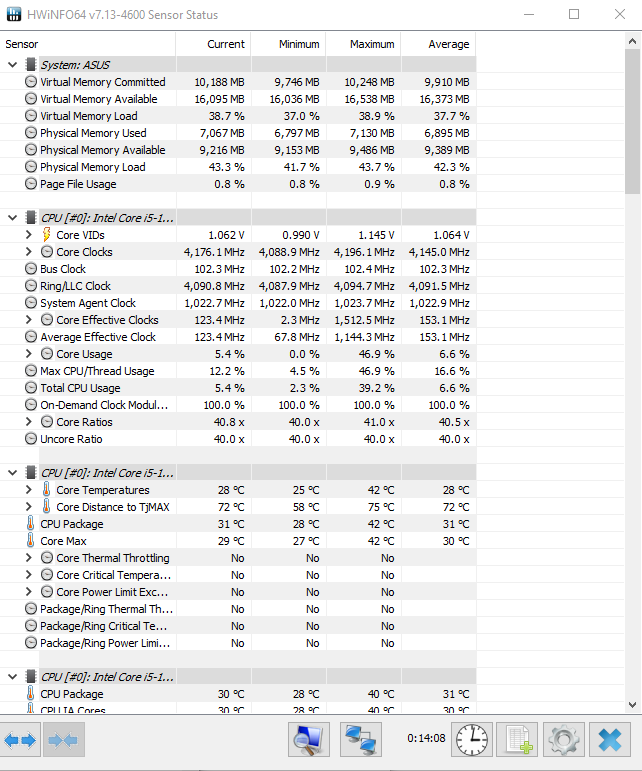


Рис. 3. Интерфейс программы HWInfo.

Однако, ввиду того, что данная программа не сыскала большой популярности среди компьютерных мастерских и сервисов, логично предположить, что она имеет и ряд недостатков. Так, некоторые из них приведены ниже:

* отсутствие каких – либо возможностей для тестирования ПЭВМ;
* отсутствие быстрого доступа в командную строку Windows или в Windows PowerShell, который иногда бывает необходим для получения дополнительной информации о ПЭВМ;
* отсутствие визуального сопровождения к данным, представленным в таблицах.

Данное ПО аналогично предыдущему распространяется на бесплатном основании для персонального использования, но требует приобретения лицензии для использования в бизнес сегменте, помимо этого для корпоративной версии предусмотрена студенческая скидка.

### 3.3. OCCT

Третий рассматриваемый в рамках работы аналог – OCCT, её интерфейс приведен на Рис. 4.



Рис. 4. Интерфейс программы OCCT.

Аналогично первому аналогу для персонального использования этот программный продукт распространяется на бесплатном основании, а для коммерческого использования необходимо приобретать лицензию. Данная программа широко распространена в сервисных центрах и мастерских, а также воспользоваться ею сможет даже не сильно опытный пользователь, что связано, на мой взгляд, с одним из главных достоинств данного ПО – понятным GUI. Что касается остальных достоинств, то они приведены ниже:

* большое количество информации о процессоре (загруженность и частота работы каждого ядра, температура), материнской плате (температуры на каждом датчике, напряжение на каждом ее элементе, а также информация об установленном BIOS), видеокарте (температура, напряжение на элементах устройства, скорость вращения вентиляторов на системе охлаждения и др.);
* большое количество тестов, направленных как на отдельные компоненты ПЭВМ (процессор, видеокарта, оперативная память), так и на ПК в целом (тест для максимальной нагрузки процессора и видеокарты, дополнительно тестирующий блок питания и цепи питания на материнской плате);
* большое количество различных графиков, сводных таблиц и рисунков, визуализирующих данные в таблицах, а также результаты тестов и общую тенденцию работы ПЭВМ.

Несмотря на все вышеперечисленные достоинства, программа имеет также и ряд недостатков, которые приведены ниже:

* отсутствие возможности тестирования жестких дисков и SSD;
* отсутствие информации о каких – либо датчиках ПЭВМ, кроме тех, что указаны выше;
* отсутствие быстрого доступа в командную строку Windows или в Windows PowerShell, который иногда бывает необходим для получения дополнительной информации о ПЭВМ.

### 3.4. CPU-Z/GPU-Z

Следующее ПО, указанное в списке – CPU-Z и GPU-Z, данные программные продукты объединены по причине того, что у них один разработчик и максимально схожий в своих областях функционал. Интерфейсы данных программ приведены на Рис. 5, а также на Рис. 6.1 и Рис. 6.2 соответственно.

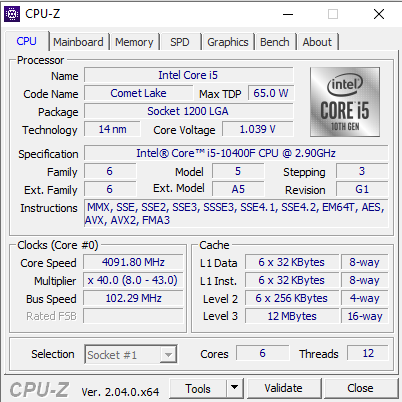


Рис. 5. Интерфейс программы CPU-Z.

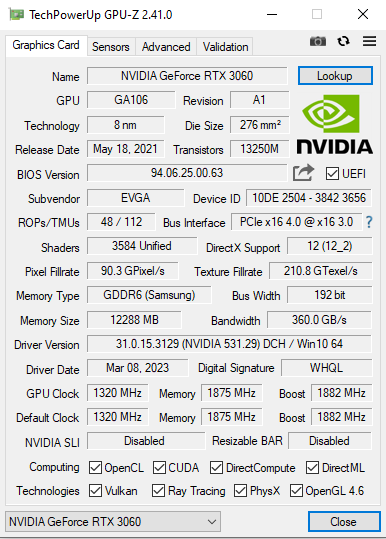


Рис. 6.1. Интерфейс программы GPU-Z.

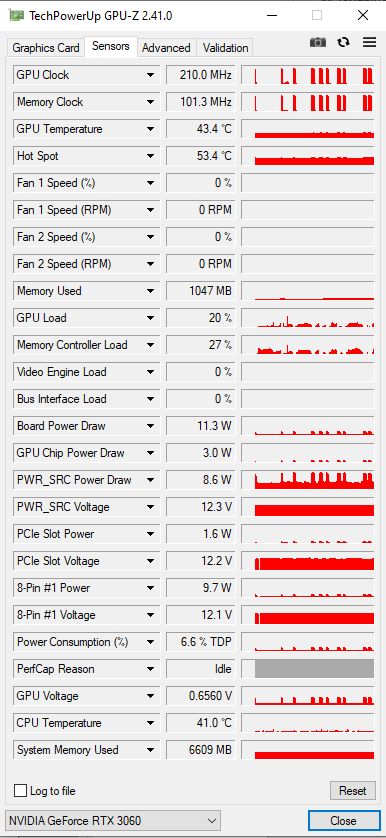


Рис.6.2. Информация с датчиков видеокарты в программе GPU-Z.

Данное ПО используется в основном когда необходимо получить максимально точную информацию о процессоре или видеокарте. Данные продукты имеют следующие достоинства:

* большое количество информации о процессоре (частота каждого ядра, температуры) и видеокарте (точные характеристики устройства, напряжение на каждом элементе видеокарты, а также графики, которые визуализируют данную информацию);
* возможность выгрузить указанную выше информацию в обычный текстовый документ;
* наличие небольшого количества информации о материнской плате и оперативной памяти, несмотря на то, что данные области не являются целевыми для данного ПО;
* полностью бесплатная модель распространения.

У данных программ имеется ряд недостатков:

* отсутствие возможности какого - либо тестирования как процессора, так и видеокарты;
* отсутствие быстрого доступа в командную строку Windows или в Windows PowerShell, который иногда бывает необходим для получения дополнительной информации о ПЭВМ;
* отсутствие какой – либо визуализации к данным о процессоре в программе CPU-Z.

### 3.5. SiSoftware Sandra

Последним рассматриваемым аналогом является программа SiSoftware Sandra, интерфейс которой приведен на Рис. 7.1. и Рис 7.2. Ее Lite – версия распространяется бесплатно, но в ней есть существенные ограничения, если говорить о версиях Engineer или Business, в них эти ограничения пропадают, но данные версии программы необходимо приобретать.

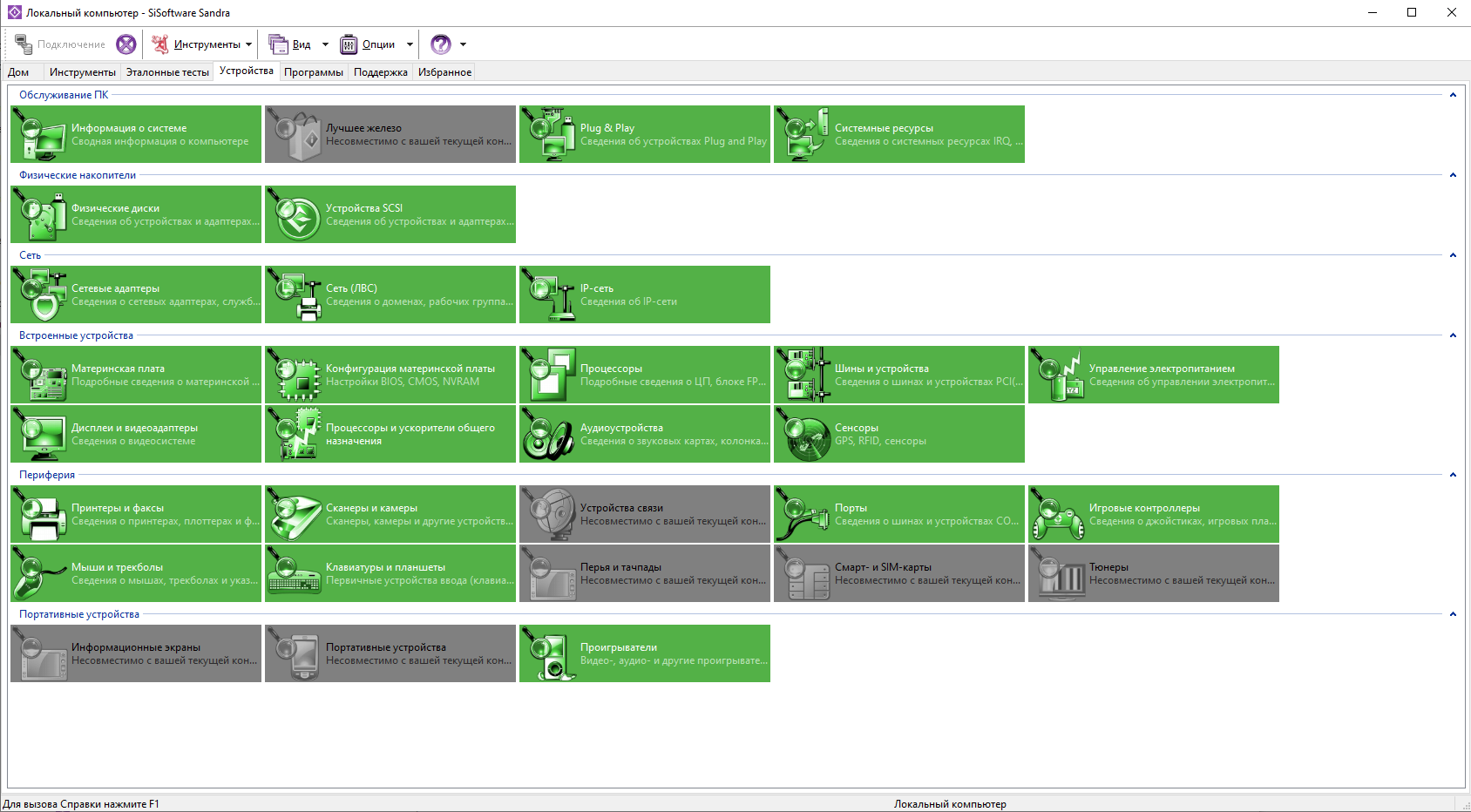


Рис. 7.1. Интерфейс программы SiSoftwate Sandra.

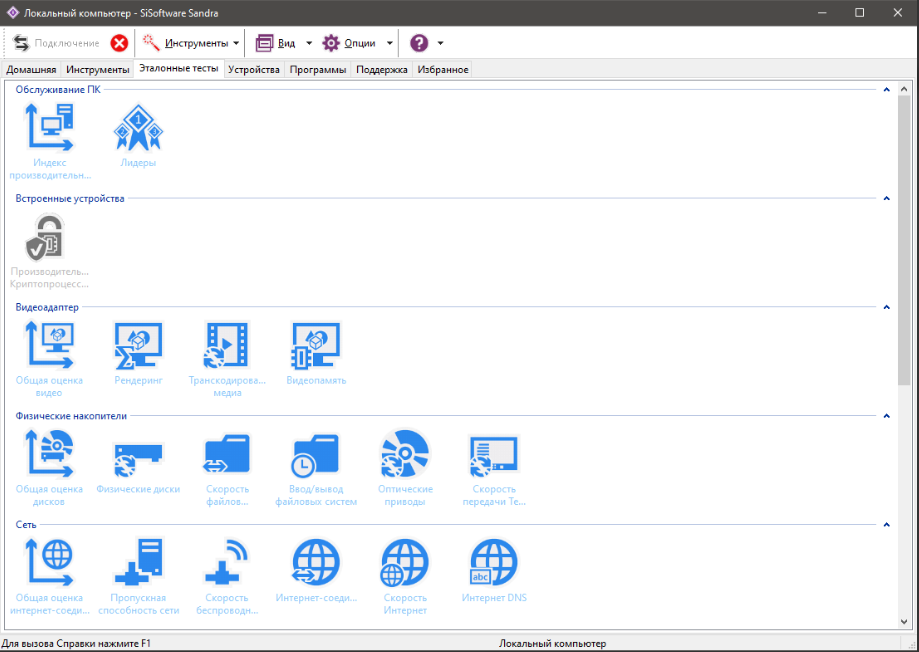


Рис. 7.2. Набор тестов, предлагаемых SiSoftware Sandra.

К достоинствам данного ПО можно отнести:

* крайне большое количество информации о всех устройствах ПЭВМ, в том числе и периферийных;
* множество встроенных тестов производительности для разного оборудования, а также формирование отчетов.

В качестве недостатков SiSoftware Sandra обычно выделяют следующие пункты:

* сложная система стресс – тестирования оборудования, которая требует ручной настройки;
* отсутствие быстрого доступа в командную строку Windows или в Windows PowerShell, который иногда бывает необходим для получения дополнительной информации о ПЭВМ.

## 4. Требования к системе

### 4.1. Функциональные требования

Программа должна показывать информацию о CPU, GPU, RAM, материнской плате и дисковых устройствах.

* Информация о CPU, которую должна предоставлять программа:
* Наименование процессора;
* Сокет процессора;
* Базовая частота процессора;
* Размер кэша процессора;
* Фирма-производитель процессора;
* Информация от производителя – ссылка на официальный сайт производителя, где приводится информация об устройстве;
* *Температура каждого ядра* (в градусах Цельсия)\*;
* *Загруженность каждого ядра* (в процентах)\*;
* *Частота работы каждого ядра* (в гигагерцах) *и шины процессора* (в мегагерцах)\*;
* *Мощность, используемая процессором: пакетом CPU, ядрами CPU* (в Ваттах).

Информация о GPU, которую должна предоставлять программа:

Наименование графического ускорителя;

* Базовая частота работы видеоядра;
* Максимальный доступный объем видеопамяти;
* Фирма-производитель видеокарты;
* Информация от производителя – ссылка на официальный сайт производителя, где приводится информация об устройстве;
* Официальный драйвер – ссылка на официальный сайт производителя, где он публикует последнюю версию драйвера для устройства;
* Текущая версия драйвера видеокарты;
* *Температура графического ядра* (в градусах Цельсия)\*;
* *Температура видеопамяти*, в случае, если такого датчика нет на устройстве, вывод информации об этом\* ;
* *Частота работы ядра и памяти видеокарты* (в мегагерцах)\*;
* *Загруженность ядра, памяти и фрейм-буфера видеокарты* (в процентах);
* Использование памяти видеокарты: максимальный объем памяти, *используемый объем памяти и доступный объем памяти* (в гигабайтах);
* *Мощность, используемая графическим ускорителем* (в ваттах).

Информация о RAM, которую должна предоставлять программа:

* Фирма-производитель оперативной памяти, в случае, если таковых несколько, перечисление всех;
* Место расположения файла подкачки в системе;
* Начальный и текущий размеры файла подкачки (в гигабайтах);
* Частота работы оперативной памяти (в мегагерцах);
* Максимальный объем памяти, *используемый объем памяти и доступный объем памяти* (в гигабайтах);
* *Использование памяти* (в процентах)\*.

Информация о материнской плате, которую должна предоставлять программа:

* Наименование материнской платы;
* Фирма-производитель материнской платы;
* Информация от производителя – ссылка на официальный сайт производителя, где приводится информация об устройстве;
* Официальный драйвер – ссылка на официальный сайт производителя, где он публикует последнюю версию драйверов и BIOS для устройства;
* Версия установленного BIOS;
* *Загруженность каждого подключенного вентилятора охлаждения* (в процентах);
* *Напряжение всех доступных датчиков* (в вольтах);
* *Температурные показатели всех доступных датчиков* (в градусах Цельсия)\*;
* *Скорость оборотов в минуту каждого вентилятора, поддерживающего данный функционал* (об/мин)\*.

Информация о дисковых устройствах, которую должна предоставлять программа:

* Название каждого тома системы и его размер (в гигабайтах);
* Файловая система каждого тома системы;
* Название каждого физического диска в системе и его загруженность (в процентах);

Примечание – показатели, выделенные курсивом, должны обновляться каждую секунду.

Каждому числовому показателю, выделенному символом «\*», должен быть поставлен в соответствие график, отображающий изменение данного показателя во времени.

Должна существовать возможность быстрого вызова Командной строки Windows и Windows PowerShell из интерфейса программы.

Должна существовать возможность выгрузки базовой (неизменяющейся) информации об устройстве в отдельный файл – отчет.

### 4.2. Требования по видам обеспечения

#### 4.2.1. Требования к математическому обеспечению

К данной системе не предъявляются требования данного вида, так как в системе не идет никаких расчетов.

#### 4.2.2. Требования к алгоритмическому обеспечению

Процесс работы пользователя с системой представлен на Рис. 8.

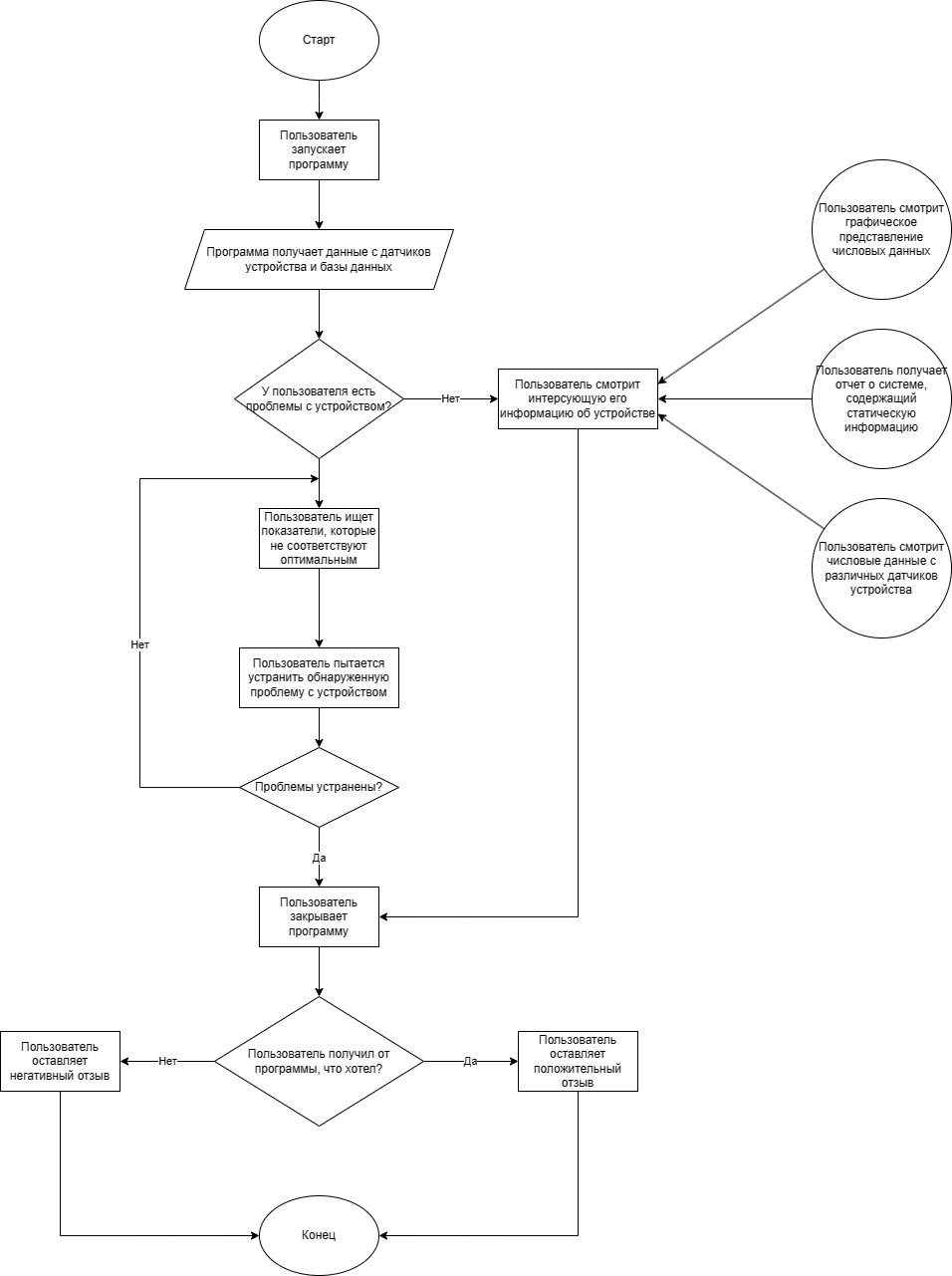


Рис. 8. Алгоритм работы пользователя с разрабатываемой информационной системой

#### 4.2.3. Требования к информационному обеспечению

В состав информационного обеспечения входит база данных, заполняемая разработчиком (внутримашинное обеспечение), а также отчеты об ОС, сгенерированные системой (выходная документация).

#### 4.2.4. Требования к информационной безопасности

Программа должна обеспечивать безопасность пользовательских данных, поэтому несмотря на упрощенную разработку и более широкий функционал сетевых фреймворков, в программе используется исключительно такие методы, получения информации о системе, которые не предполагают какого-либо взаимодействия с Интернетом.

Помимо этого программа должна соответствовать требованиям законодательства в вопросе защиты персональных данных.

#### 4.2.5. Требования к лингвистическому обеспечению

Разработка программы велась с использованием языка Python для разработки главной части программы, языка C++ - для написания исполняемого файла для работы с датчиками на macOS, фреймворка PyQt6 для построения окон, pyqtgraph – для построения графиков, Pythonnet – для работы с .dll файлами на ОС Windows.

Для работы с базой данных, в которой хранится информация о компонентах системы используется SQLite и соответствующий модуль в языке Python.

Для описания предметной области использовались языки моделирования UML, BPMN, а также диаграммы компонентов и Use Case диаграммы.

#### 4.2.6. Требования к программному обеспечению

В качестве операционных систем, для которых предназначена программа, согласно анализу рынка, выступают только macOS версии 13 и выше, а также ОС Windows 10, 11.

#### 4.2.7. Требования к техническому обеспечению

Общие требования: не менее 4 Гб оперативной памяти, 10 Мб на дисковом хранилище.

ОС Windows: стационарный компьютер.

macOS: процессор архитектуры ARM.

#### 4.2.8. Требования к организационному обеспечению

Программой может пользоваться любой желающий, однако разработка велась в соответствии с предпочтениями экспертов в области ремонта и обслуживания компьютерной техники, поэтому пользователю, не входящему в данную категорию, возможно, будет не хватать какой-то информации.

#### 4.2.9. Требования к методическому обеспечению

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.

## 5. Проектирование системы

### 5.1. SWOT – анализ

Таблица 3. 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. |

### 5.2. Диаграмма вариантов использования

Диаграмма вариантов использования (Use Case Diagram) является одним из видов диаграмм UML (Unified Modeling Language) и представляет собой визуальное средство для моделирования того, как система взаимодействует с внешними сущностями (акторами) в различных сценариях использования. Она помогает описать функциональные требования к системе, фокусируясь на том, как различные акторы взаимодействуют с системой в различных ситуациях.

Основные элементы диаграммы вариантов использования:

1. Акторы (Actors).
2. Варианты использования (Use Cases).
3. Отношения между акторами и вариантами использования.

Use Case диаграмма для рассматриваемой предметной области приведена в Приложении 2.

### 5.3. 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.

**Прецендент 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.

**Прецендент 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.

### 5.4. Диаграммы процессов BPMN

BPMN (Business Process Model and Notation) представляет собой стандартную нотацию для моделирования бизнес-процессов. Эта нотация была разработана с целью обеспечения единообразного и понятного способа представления бизнес-процессов, чтобы различные участники проекта могли лучше понимать, анализировать и оптимизировать бизнес-процессы.

Основные элементы BPMN:

1. Активности (Activities).
2. Шлюзы (Gateways).
3. События (Events).
4. Потоки (Flows).
5. Пулы и дорожки (Pools and Lanes).
6. Артефакты (Artifacts).

BPMN широко применяется в области бизнес-анализа, моделирования и оптимизации бизнес-процессов, а также при проектировании информационных систем.

BPMN диаграммы для информационной системы были составлены с учетом Use Case диаграмм, они представлены на Рис. 9.1 – Рис. 9.4.

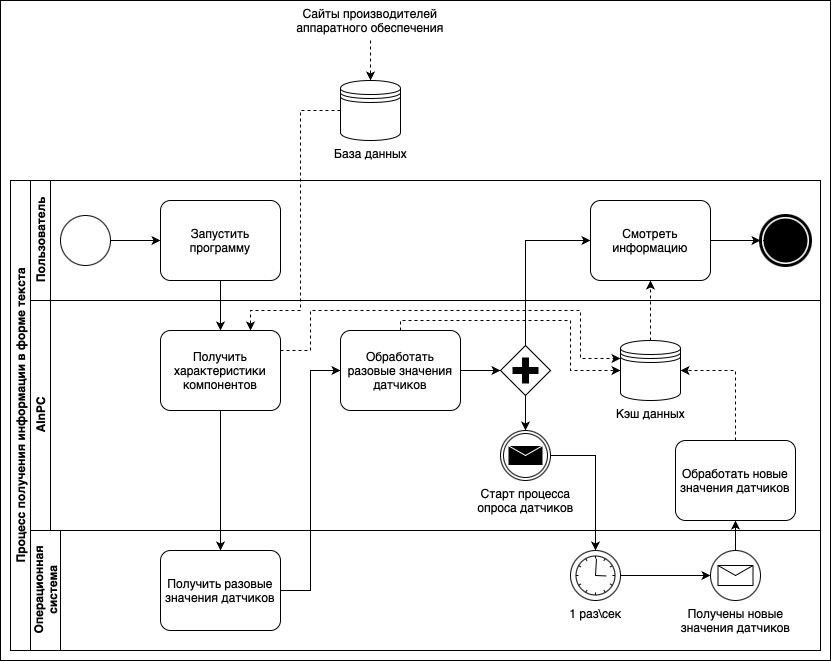


Рис. 9.1. Процесс получения информации в текстовой форме

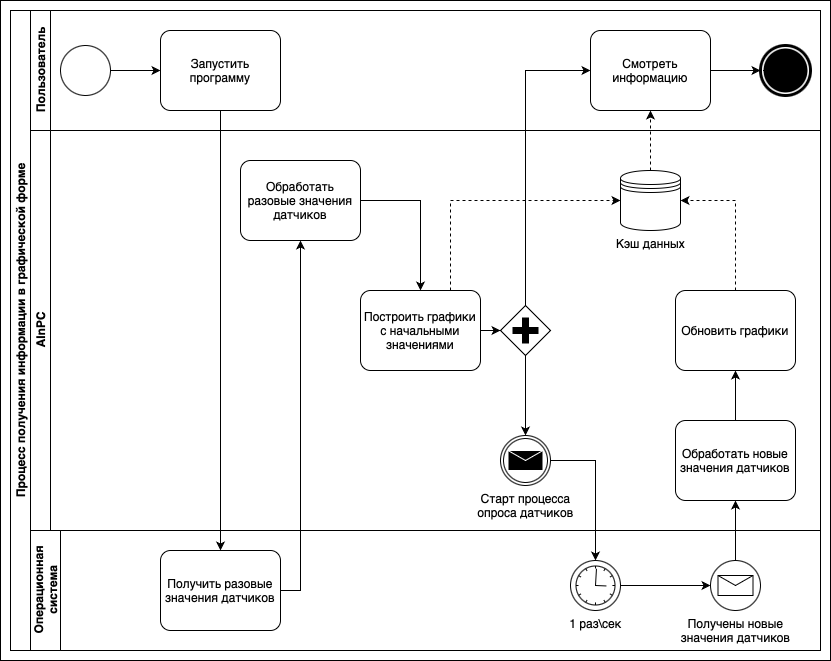


Рис. 9.2. Процесс получения информации в графической форме

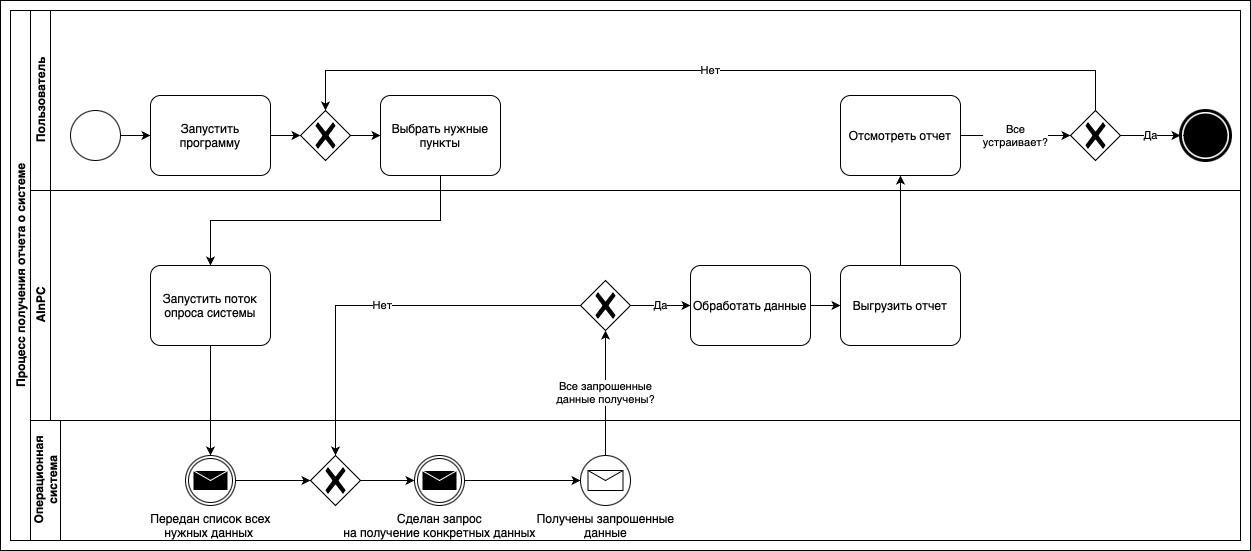


Рис. 9.3. Процесс получения отчета о системе

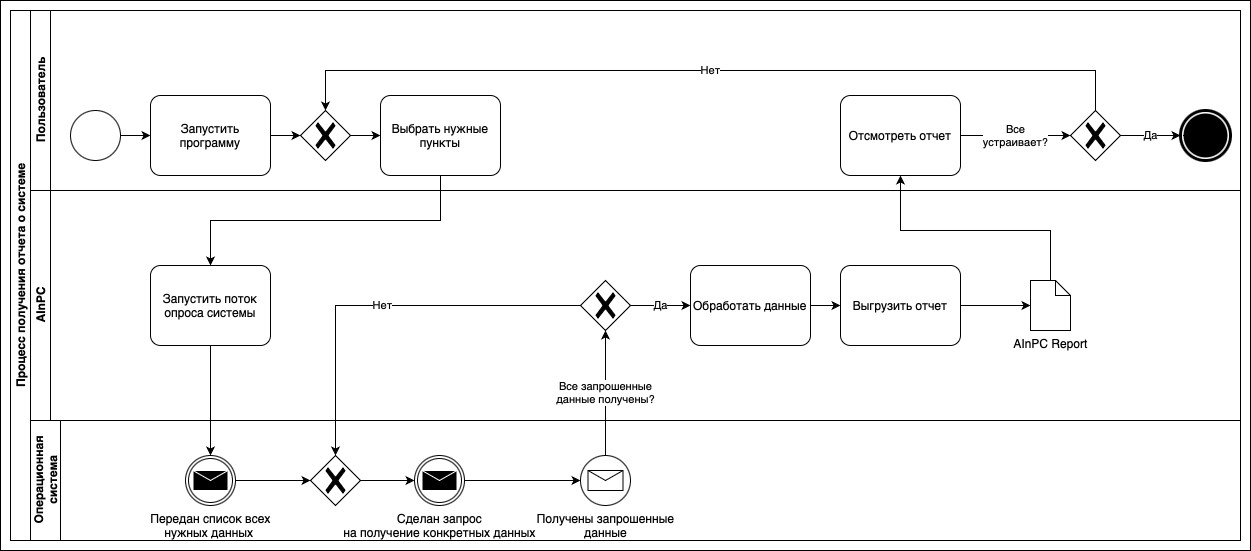


Рис. 9.4. Процесс получения отчета о системе

Примечание – на Рис. 9.3, Рис.9.4 приведены модели, отражающие один процесс, их отличие состоит в том, что на Рис. 9.4. присутствует знак выгружаемого документа, передающегося пользователю, на Рис. 9.3. – этого знака нет – подразумевается, что он сразу передается пользователю.

### 5.5. Диаграмма классов

В Приложении 3 приведена диаграмма классов для предметной области, желтыми овалами выделены дополнительные точки взаимодействия системы с другими элементами. Также с припиской “extend” указана одна из потенциальных точек расширения системы. На некоторых из стрелочек указаны пояснения к взаимодействию классов между собой.

### 5.6. Информационная модель данных

В Приложении 4 приведена таблица, объединяющая в себе таблицу сущностей и таблицу атрибутов и ассоциаций сущностей.

### 5.7. Диаграмма компонентов

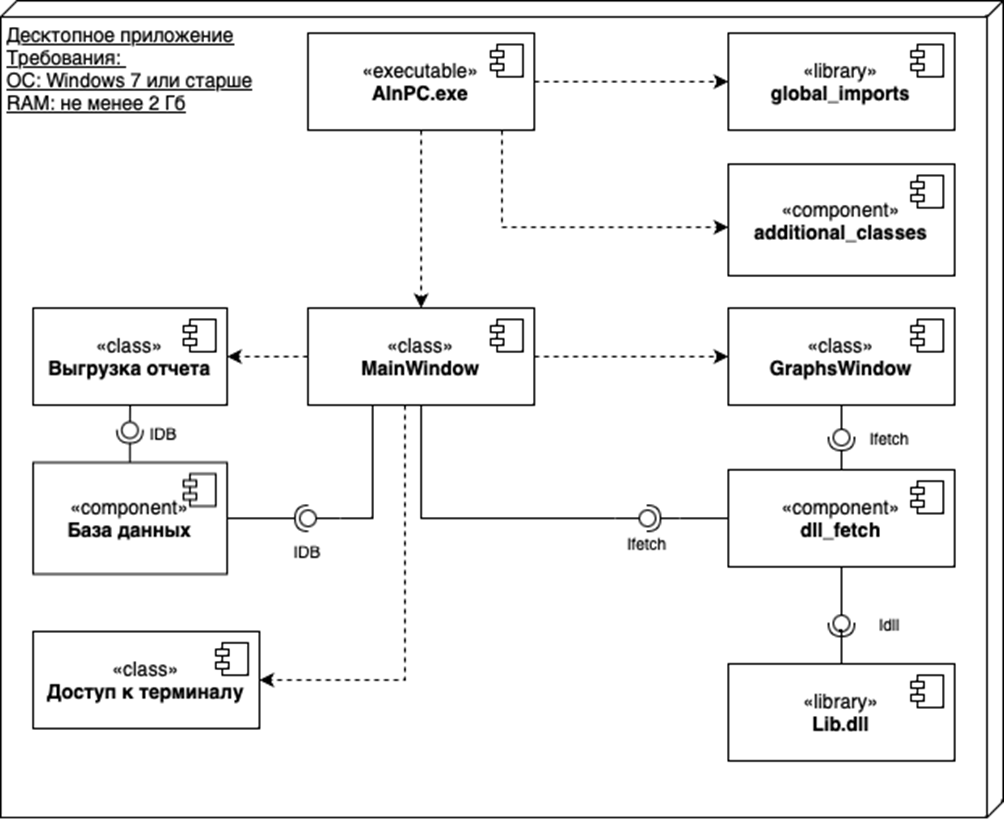


Рис. 10. Диаграмма компонентов

**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()*

### 5.8. 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. 11.



Рис. 11. 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 Рис. 11 – Рис. 15, соответственно.

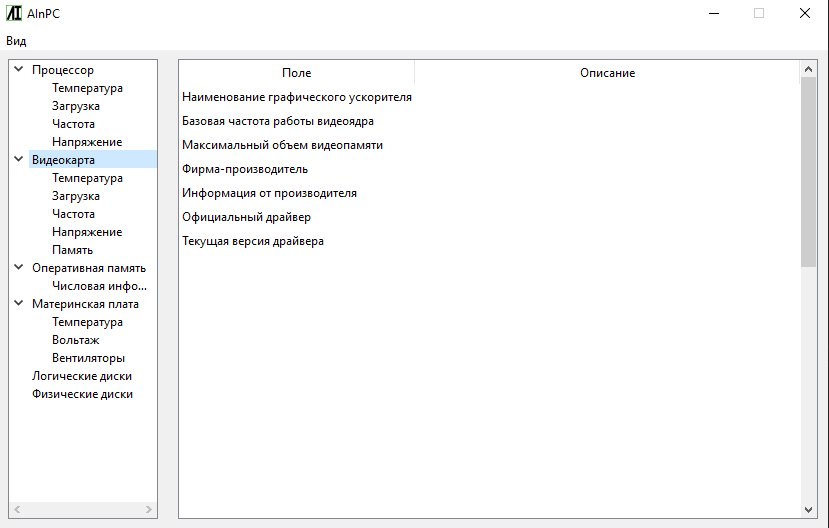


Рис. 12. Дизайн раздела первого уровня «Видеокарта»

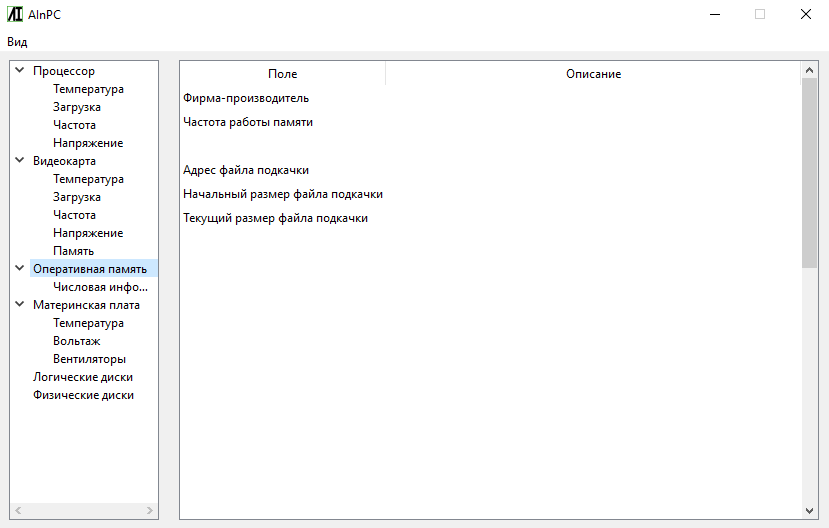


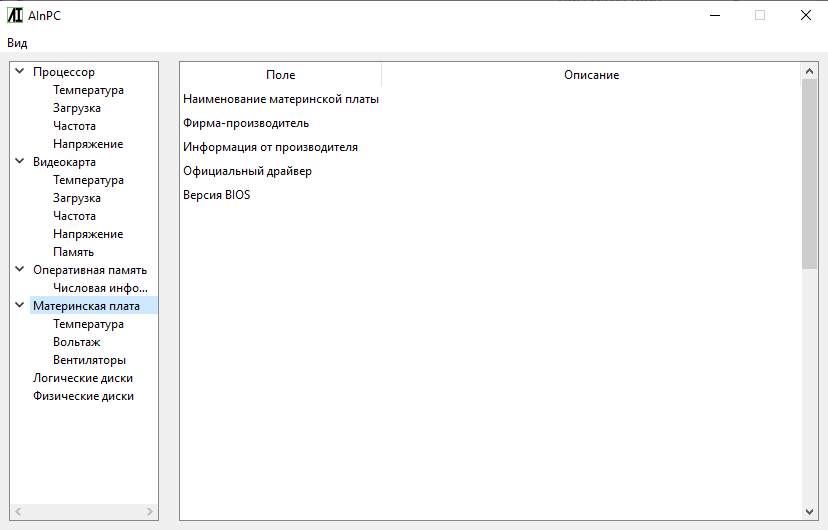
Рис. 13. Дизайн раздела первого уровня «Оперативная память»

Рис. 14. Дизайн раздела первого уровня «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 Рис. 14 и Рис. 15, соответственно.

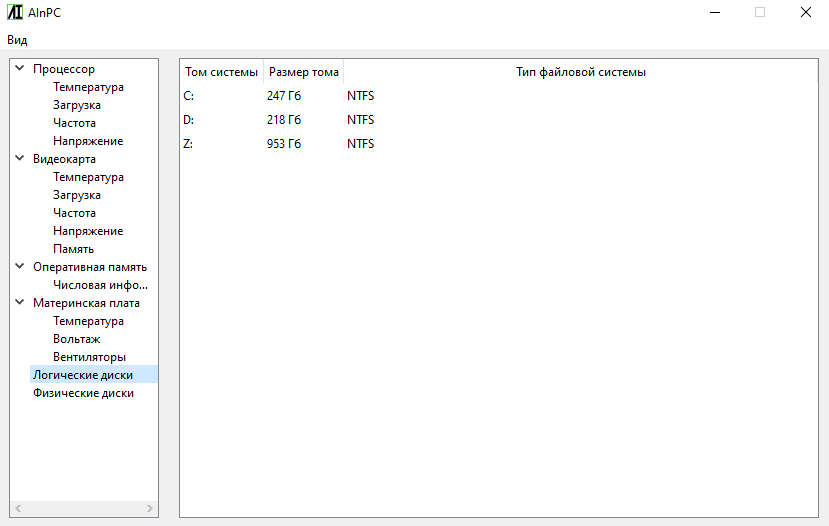


Рис. 15. Дизайн раздела первого уровня «Логические диски»

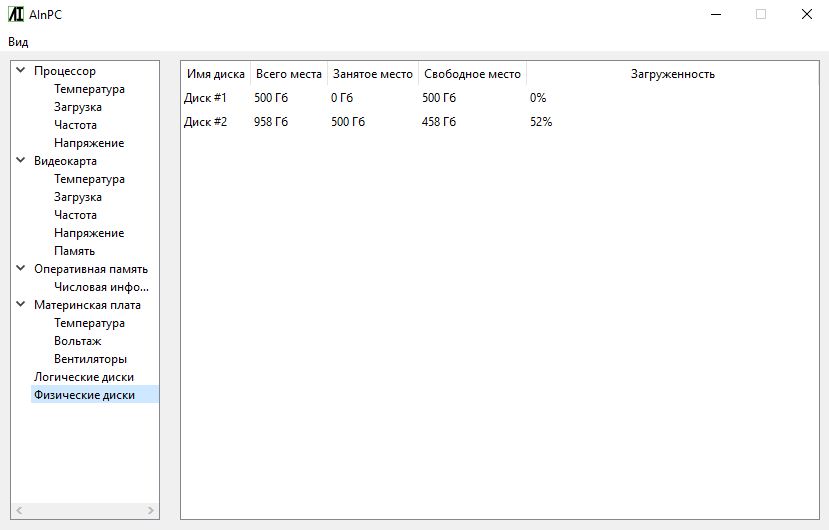


Рис. 16. Дизайн раздела первого уровня «Физические диски»

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 Рис. 17 – Рис. 20 приведена информация о температуре, загрузке, частоте работы каждого ядра процессора и напряжении, подаваемом на процессор, соответственно.

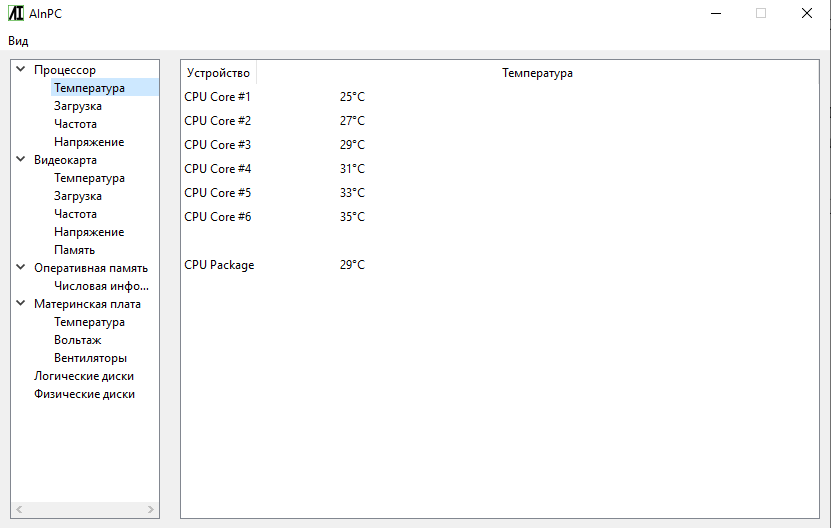


Рис 17. Дизайн раздела второго уровня «Процессор – Температура»

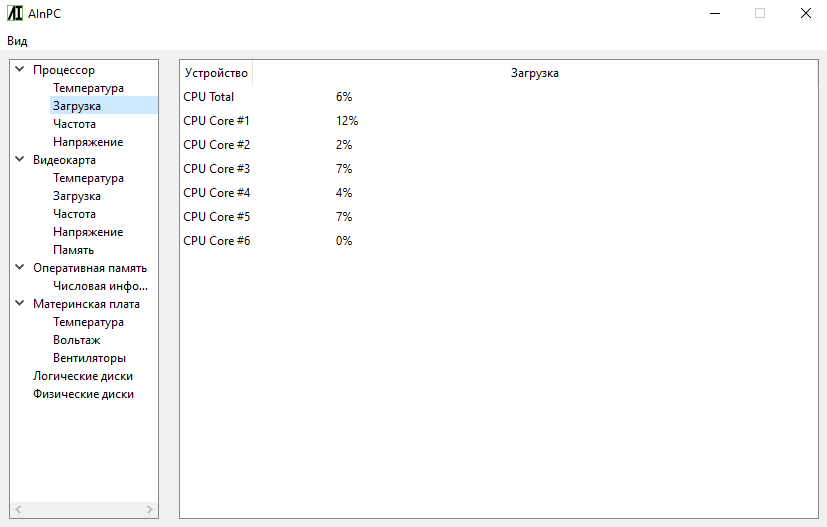


Рис 18. Дизайн раздела второго уровня «Процессор – Загрузка»

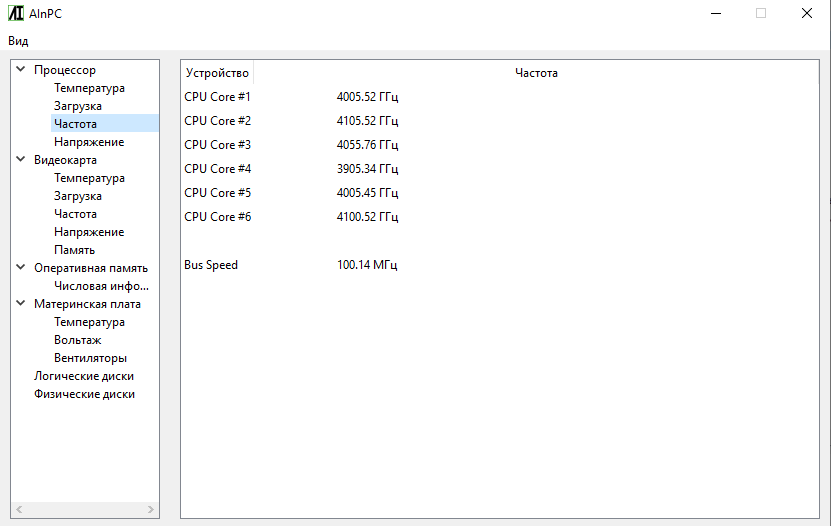
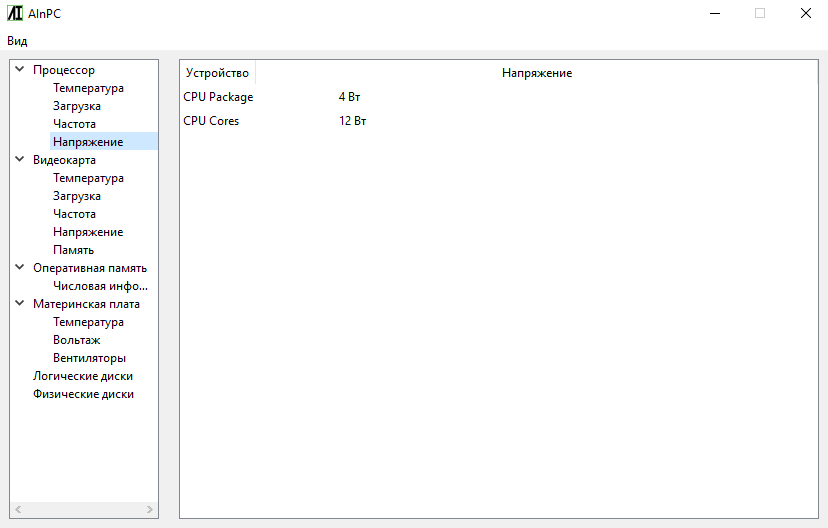


Рис 19. Дизайн раздела второго уровня «Процессор – Частота»



Рис/ 20. Дизайн раздела второго уровня «Процессор – Напряжение»

Примечание – на Рис. 11 – Рис. 20 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. Рис. 21), доступ к которому осуществляется путем нажатия кнопки "Вид" в верхнем меню (см. Рис. 22), а затем клавиши "Графики".

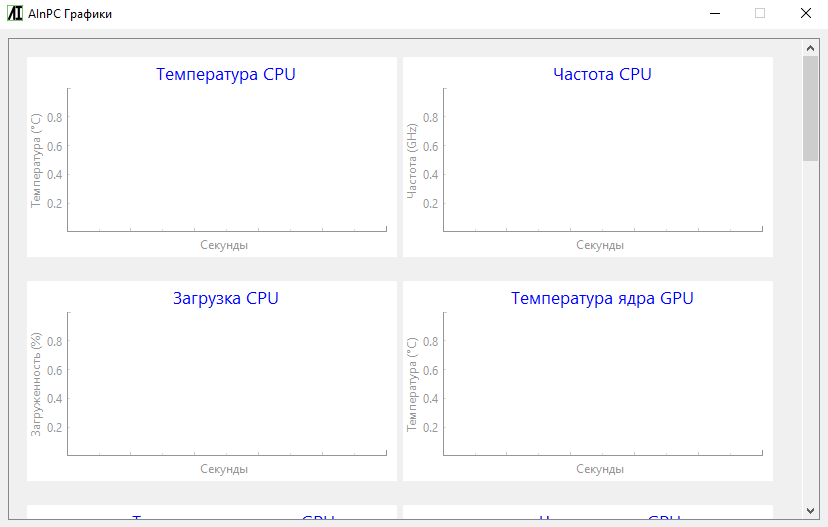


Рис. 21. Дизайн окна «Графики»

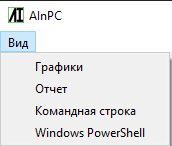


Рис. 22. Дизайн меню выбора окна

На Рис. 22 помимо клавиши "Графики" есть клавиша "Отчет", данная кнопка отвечает за выгрузку отчета, согласно функциональным требованиям.

Также, на Рис. 23 есть клавиши "Командная строка" и "Windows PowerShell", которые выполняют свой функционал согласно требованиям.

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

## 6. Разработка информационной системы

### 6.1. Backlog

Список задач по разработке информационной системы может выглядеть следующим образом:

1. Разработка функциональных требований к программе.
2. Разработка дизайна и реализация интерфейса.
3. Реализация функционала программы под ОС Windows.
4. Адаптация программы под macOS.

### 6.2. Функциональные блоки программы

Обратимся к диаграмме компонентов, которая явно показывает весь функционал кода и все взаимозависимости в нем. Обратимся к Рис. 10.

Библиотека *global\_import.py* хранит в себе все импорты дополнительных модулей, функции, которые используются в рамках всего кода, а также местоположение базы данных, и уже из нее в весь остальной код импортируется все, что нужно.

*Global\_import.py:*

#Импортируем все необходимые модули из библиотеки для работы с графическим интерфейсом

from PyQt6.QtCore import QSize, Qt, QTimer, QThread, pyqtSignal, QRegularExpression

from PyQt6.QtWidgets import QCheckBox, QGridLayout, QLineEdit, QMessageBox, QSplashScreen, QMainWindow, QApplication, QTreeWidget, QTableWidget, QAbstractItemView, QHeaderView, QTreeWidgetItem, QTableWidgetItem, QLabel, QPushButton, QDialog, QTextEdit, QVBoxLayout, QWidget, QScrollArea, QHBoxLayout

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

#Импортируем библиотеки для работы системой, базами данных, построения графиков

import sys, sqlite3, pyqtgraph

#Импортируем модули для работы с операционной системой

from os import getcwd

#Импортируем модули для работы с системным терминалом

from subprocess import \*

#Импортируем модули для получения данных с оперативной памяти

from psutil import virtual\_memory, disk\_partitions, disk\_usage

#Импортируем модуль для получения текущего времени системы

from datetime import datetime

#Импортируем модуль для работы с системными путями файлов

from pathlib import Path

#Функция, вычисляющая среднее значение массива и округляющая его значения

def get\_average(massive):

return round(sum(massive) / len (massive), 2)

#Записываем местоположение базы данных в отдельную переменную, чтобы доступ к ней

#можно было получить из всех частей кода

db = getcwd() + "/aipcdb\_mac.db"

Компонент *additional\_classes.py* хранит в себе классы, которые отвечают за дополнительный функционал программы: создание гиперссылок в ячейках или же описание работы потоков, получающих информацию о числовых показателях датчиках системы.

Additional\_classes.py:

#Класс для отображения и корректной работы гиперссылок на сайте производителя

class HyperlinkLabel(QLabel):

def \_\_init\_\_(self, text, link):

super().\_\_init\_\_()

self.setText(f'<a href="{link}">{text}</a>')

self.setOpenExternalLinks(True)

<…>

#Класс, описывающий работу потока, опрашивающего систему на предмет загрузки, частоты и напряжения компонентов пока не будет отключен

class StatsThread(QThread):

stats\_signal = pyqtSignal(list)

def \_\_init\_\_(self, password):

QThread.\_\_init\_\_(self)

self.running = False

self.type\_of\_info = None

self.password = password

#Функция, изменающая тип собираемых данных

def set\_type(self, root, text):

self.root = root

self.text = text

#Функция, запускающаяся при старте потока, собирает данные в зависимости от указанного типа. После того, как данные готовы, компанует и передает их

def run(self):

self.running = True

while self.running:

filtered\_lines = []

if self.root == "CPU":

command = "sudo -S powermetrics -n1 --samplers cpu\_power"

process = Popen(command, shell = True, stdin = PIPE, stdout = PIPE, stderr = PIPE, universal\_newlines = True)

stdout, stderr = process.communicate(self.password + '\n')

lines = stdout.split('\n')

if self.text == "Load":

filtered\_lines = [line[line.find(":") + 1 : line.find("(")].strip() for line in lines if "CPU" in line and "active residency" in line]

container = [filtered\_lines, self.root, self.text]

self.stats\_signal.emit(container)

elif self.text == "Clock":

filtered\_lines = [line[line.find(":") + 1:].strip() for line in lines if "CPU" in line and "frequency" in line]

container = [filtered\_lines, self.root, self.text]

self.stats\_signal.emit(container)

elif self.text == "Power":

filtered\_lines = [line[line.find(":") + 1:].strip() for line in lines if "CPU Power" in line]

container = [filtered\_lines, self.root, self.text]

self.stats\_signal.emit(container)

elif self.root == "GPU":

command = "sudo -S powermetrics -n1 --samplers gpu\_power"

process = Popen(command, shell = True, stdin = PIPE, stdout = PIPE, stderr = PIPE, universal\_newlines = True)

stdout, stderr = process.communicate(self.password + '\n')

lines = stdout.split('\n')

if self.text == "Load":

filtered\_lines = [line[line.find(":") + 1 : line.find("(")].strip() for line in lines if "GPU" in line and "active residency" in line]

container = [filtered\_lines, self.root, self.text]

self.stats\_signal.emit(container)

elif self.text == "Clock":

filtered\_lines = [line[line.find(":") + 1:].strip() for line in lines if "GPU" in line and "frequency" in line]

container = [filtered\_lines, self.root, self.text]

self.stats\_signal.emit(container)

elif self.text == "Power":

filtered\_lines = [line[line.find(":") + 1:].strip() for line in lines if "GPU Power" in line]

container = [filtered\_lines, self.root, self.text]

self.stats\_signal.emit(container)

#Функция, принудительно останавливающая поток

def stop(self):

self.terminate()

В классе *report\_window.py* описывается окно, которое узнает у пользователя, какую именно информацию о системе он хочет получить, а затем передает ее в поток, откуда получает эту информацию в виде текста, а затем сохраняет ее в документ.

Report\_window.py:

class ReportWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

#Создаем список для интерпретации ввода пользователя в символы терминала

self.checkbox\_translations = {

"Общая информация об устройстве": "SPHardwareDataType",

"ATA": "SPParallelATADataType",

"Apple Pay": "SPSecureElementDataType",

<…>

"Устаревшее ПО": "SPLegacySoftwareDataType",

"Язык и регион": "SPInternationalDataType",

}

<…>

#Функция, запускающая процесс обработки введеных пользователем пунктов

def showSelectedCheckboxes(self):

selected\_checkboxes = [self.checkbox\_translations[checkbox.text()] for checkbox in self.checkboxes if checkbox.isChecked()]

if selected\_checkboxes:

self.accept\_button.setEnabled(False)

self.fill\_button.setEnabled(False)

self.remove\_button.setEnabled(False)

#Создаем и настраиваем поток получения данных о системе, подключаем "реакцию" на получение данных из потока

self.report\_thread = ReportThread(selected\_checkboxes, self.checkbox\_translations)

self.report\_thread.report\_thread\_signal.connect(self.create\_report)

self.report\_thread.start()

else:

QMessageBox.warning(self, "Предупреждение", "Выберите хотя бы один пункт для формирования отчета.")

<…>

#Функция, создающая отчет о системе, используя данные, полученные из потока

def create\_report(self, data):

current\_time = datetime.now().strftime("%d.%m.%Y %H:%M:%S")

file\_name = f"AInPC Report {current\_time}.txt"

with open(file\_name, "w") as file:

file.write(data)

info\_message = QMessageBox(self)

info\_message.setWindowTitle('Информация')

info\_message.setText(f'Файл сохранен в рабочую папку с названием: {file\_name}')

timer = QTimer(self)

timer.setSingleShot(True)

timer.timeout.connect(info\_message.close)

timer.start(3000)

info\_message.exec()

self.accept\_button.setEnabled(True)

self.fill\_button.setEnabled(True)

self.remove\_button.setEnabled(True)

В классе *main\_window.py* описывается процесс создания главного окна программы, включение потоков, отображение таблиц, их обновление, и т.д.

#Создаем главное рабочее окно

class MainWindow(QMainWindow):

def \_\_init\_\_(self, password\_for\_work):

super().\_\_init\_\_()

#Создаем переменные и зависимости, используемые в рамках главного окна

self.password\_for\_work = password\_for\_work

self.root\_for\_timer = None

self.text\_for\_timer = None

self.cpu\_cores = None

<…>

self.cpu\_info\_text = initialize\_cpu()

#Настраиваем окно, отображающееся во время инициализации программы

splash = self.show\_splash\_screen()

splash.showMessage("Идет инициализация...", Qt.AlignmentFlag.AlignBottom | Qt.AlignmentFlag.AlignRight, Qt.GlobalColor.white)

#Настраиваем и запускаем поток, инициализирующий первичные показатели датчиков системы

self.initializing\_thread = InitializingThread(splash, password\_for\_work)

self.initializing\_thread.start()

self.initializing\_thread.init\_signal.connect(self.on\_change\_init, Qt.ConnectionType.QueuedConnection)

#Настраиваем вид главного окна

self.setWindowTitle("AInPC")

self.setFixedSize(QSize(830, 500))

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()

graphs\_item = QAction("Графики", self)

report\_item = QAction("Сформировать отчет", self)

terminal\_item = QAction("Терминал", self)

file\_menu.addAction(graphs\_item)

file\_menu.addAction(terminal\_item)

file\_menu.addAction(report\_item)

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

terminal\_item.triggered.connect(self.open\_terminal)

report\_item.triggered.connect(self.open\_report\_window)

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

self.tree\_widget = QTreeWidget(self)

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

<…>

#Создаем и настраиваем правую часть окна - таблицу, где отображается информация о выбранном компоненте системы

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)

<…>

#Создаем поток опроса датчков компонентов системы и "подключаем" его реакцию на готовность обновления

self.stats\_thread = StatsThread(password\_for\_work)

self.stats\_thread.stats\_signal.connect(self.on\_change, Qt.ConnectionType.QueuedConnection)

#Создаем таймер обновления и "подключаем" действие, по истечению времени таймера

self.timer = QTimer(self)

self.timer.timeout.connect(self.update\_table\_with\_timer)

<…>

#Функция, работающая по сообщению от потока, она обновляет все массивы, в которых хранятся значения датчиков

def on\_change(self, data):

if data[1] == "CPU":

if data[2] == "Load":

self.massive\_cpu\_load = data[0]

elif data[2] == "Clock":

self.massive\_cpu\_clock = data[0]

elif data[2] == "Power":

self.massive\_cpu\_power = data[0]

elif data[1] == "GPU":

if data[2] == "Load":

self.massive\_gpu\_load = data[0]

elif data[2] == "Clock":

self.massive\_gpu\_clock = data[0]

elif data[2] == "Power":

self.massive\_gpu\_power = data[0]

<…>

#Функция, работающая по сообщению от таймера, запускает функцию в зависимости от того, какая "ветвь" сейчас активна

def update\_table\_with\_timer(self):

if self.root\_for\_timer == "CPU":

if self.text\_for\_timer == "Temperature":

self.fill\_table\_cpu\_temp()

elif self.text\_for\_timer == "Load":

if not self.massive\_cpu\_load:

self.closeEvent()

if not self.stats\_thread.isRunning():

self.stats\_thread.set\_type(self.root\_for\_timer, self.text\_for\_timer)

self.stats\_thread.start()

self.fill\_table\_cpu(self.massive\_cpu\_load)

elif self.text\_for\_timer == "Clock":

if not self.massive\_cpu\_clock:

self.closeEvent()

if not self.stats\_thread.isRunning():

self.stats\_thread.set\_type(self.root\_for\_timer, self.text\_for\_timer)

self.stats\_thread.start()

<…>

elif self.root\_for\_timer == "PhysicalDisk":

self.update\_nand\_temp()

<…>

#Функция, запускающая появление окна после процессора иницилизации

def create\_main\_window(self, splash):

splash.close()

self.initialize\_table(None, "Материнская плата")

self.show()

#Функция, отображающая окно инициализации

def show\_splash\_screen(self):

splash\_pix = QPixmap(getcwd() + "/recources//main\_icon.png").scaled(QSize(500, 500), Qt.AspectRatioMode.KeepAspectRatio, Qt.TransformationMode.SmoothTransformation)

splash = QSplashScreen(splash\_pix, Qt.WindowType.WindowStaysOnTopHint)

splash.setFixedSize(500, 500)

splash.setWindowFlags(Qt.WindowType.FramelessWindowHint)

splash.setEnabled(False)

splash.show()

return splash

#Функция, открывающая терминал

def open\_terminal(self):

run(["open", "-a", "Terminal"])

#Функция, открывающая окно с графиками

def open\_graphs\_window(self):

self.graphs\_window\_code = GraphsWindow(self.password\_for\_work)

#Функция, открывающая окно формирования отчета о системе

def open\_report\_window(self):

self.report\_window\_code = ReportWindow()

#Функция, принудительно останавливающая поток при закрытии программы

def stop\_thread(self):

self.stats\_thread.stop()

#Функция, отвечающая за безопасное выключение программы

def closeEvent(self, event):

self.hide()

self.stats\_thread.running = False

self.timer.stop()

self.stop\_thread()

super().closeEvent(event)

Класс *graphs\_window.py* служит для отображения окна с графическим представлением числовых данных с датчиков системы.

*Graphs\_window.py:*

#Создаем окно с графиками

class GraphsWindow(QMainWindow):

def \_\_init\_\_(self, password\_for\_work):

super().\_\_init\_\_()

#Создаем переменные и зависимости, используемые в рамках главного окна

self.password\_for\_work = password\_for\_work

self.values\_list = []

self.graphs\_list = []

#Настраиваем окно, отображающееся во время инициализации программы

splash\_for\_graphs = self.show\_graphs\_splash\_screen()

#Настраиваем и запускаем поток, инициализирующий первичные показатели датчиков системы

self.graphs\_thread\_initialize = InitializingGraphsThread(splash\_for\_graphs, self.password\_for\_work)

self.graphs\_thread\_initialize.start()

self.graphs\_thread\_initialize.initializing\_graphs\_signal.connect(self.fill\_values\_list\_on\_init, Qt.ConnectionType.QueuedConnection)

#Создаем поток опроса датчков компонентов системы и "подключаем" его реакцию на готовность обновления

self.graphs\_thread = GraphsThread(self.password\_for\_work)

self.graphs\_thread.graphs\_signal.connect(self.fill\_values\_list\_on\_loop)

<…>

#Функция, обновляющая графики

def update\_graphs(self):

if not self.graphs\_thread.isRunning():

self.graphs\_thread.start()

self.sort\_values()

j = 0

for plot\_widget, plot in self.plot\_widgets:

x\_ax = [i for i in range(len(self.graphs\_list[j]))]

plot.setData(x\_ax, self.graphs\_list[j])

j += 1

<…>

#Разбираем значения из листа для корректного отображения

def unparse\_list(self, input\_list):

temp\_list = []

for item in input\_list:

for i in range(len(item)):

temp\_list.append(item[i])

return temp\_list

#Заполняем массив хранящий графики, всеми элементами

def create\_graphs\_list(self):

temp\_list = self.unparse\_list(self.values\_list)

for item in temp\_list:

self.graphs\_list.append([item])

#Сортируем значения, чтобы отображались последние 60 значений

def sort\_values(self):

temp\_list = self.unparse\_list(self.values\_list)

for i in range(len(self.graphs\_list)):

self.graphs\_list[i].append(temp\_list[i])

if len(self.graphs\_list[i]) > 60:

del self.graphs\_list[i][0]

Интерфейс iDB реализуется в файле *fetch.py* фукнцией:

#Функция, получающая из базы данных информацию о процессоре

def initialize\_cpu():

cpu\_info = run('system\_profiler SPHardwareDataType | grep -E "Chip|Total Number of Cores"', shell=True, capture\_output=True, text=True).stdout.split("\n")

cpu\_info = cpu\_info[:-1]

cpu\_info = [item.strip() for item in cpu\_info]

name = cpu\_info[0][cpu\_info[0].index(":") + 2:]

count\_of\_p\_cores = cpu\_info[1][cpu\_info[1].index("(") + 1:cpu\_info[1].index("performance") - 1].strip()

count\_of\_e\_cores = cpu\_info[1][cpu\_info[1].index("and ") + 3:cpu\_info[1].index("efficiency")].strip()

try:

conn = sqlite3.connect(db)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM cpu\_info WHERE cpu\_name = ? and cpu\_count\_of\_p\_cores = ? and cpu\_count\_of\_e\_cores = ?", (name, count\_of\_p\_cores, count\_of\_e\_cores))

return cursor.fetchone()

except sqlite3.Error as error:

pass

finally:

if conn:

conn.close()

В компоненте *fetch.py* описывается процесс получения информации о системе посредством взаимодействия с терминалом системы.

*Fetch.py:*

#Функция, получающая из системы данные о физических дисках и переводящая эту информацию на русский язык

def initialize\_disks():

disks = run(["system\_profiler", "SPNVMeDataType"], capture\_output=True, text=True).stdout

disks = disks.replace("\n\n", "\n").replace("Capacity", "Емкость").replace("TRIM Support", "Поддержка TRIM").replace("GB", "Гб").replace("bytes", "бит")

disks = disks.replace("Model", "Модель").replace("Serial Number", "Серийный номер").replace("Yes", "Да").replace("No", "Нет").replace("Revision", "Ревизия")

disks = disks.replace("Detachable Drive", "Внешний накопитель").replace("BSD Name", "Название BSD").replace("Partition Map Type", "Тип схемы разделов")

disks = disks.replace("Removable Media", "Съемный носитель").replace("S.M.A.R.T. status", "Статус S.M.A.R.T.").replace("GPT (GUID Partition Table)", "GPT (Таблица разделов GUID)")

disks = disks.replace("Volumes", "Тома").replace("Content", "Содержимое").replace("Verified", "Проверен")

disks = disks.split("\n")

disks = disks[:-1]

return disks

#Функция, получающая из системы данные об оперативной и переводящая эту информацию на русский язык

def initialize\_ram():

ram = run(["system\_profiler", "SPMemoryDataType"], capture\_output=True, text=True).stdout

ram = ram.replace("\n\n", "\n").replace("Memory", "Объем памяти").replace("Type", "Тип").replace("GB", "Гб").replace("Manufacturer", "Производитель")

ram = ram.split("\n")

ram = ram[1:-1]

return ram

Доступ к терминалу осуществляется посредством функции, описанной в *main\_window.py*:

#Функция, открывающая терминал

def open\_terminal(self):

run(["open", "-a", "Terminal"])

### 6.3. Скриншоты UI

В ходе реализации проекта были предприняты некоторые решения, в связи с которыми были внесены изменения в итоговой вариант дизайна пользовательского интерфейса.

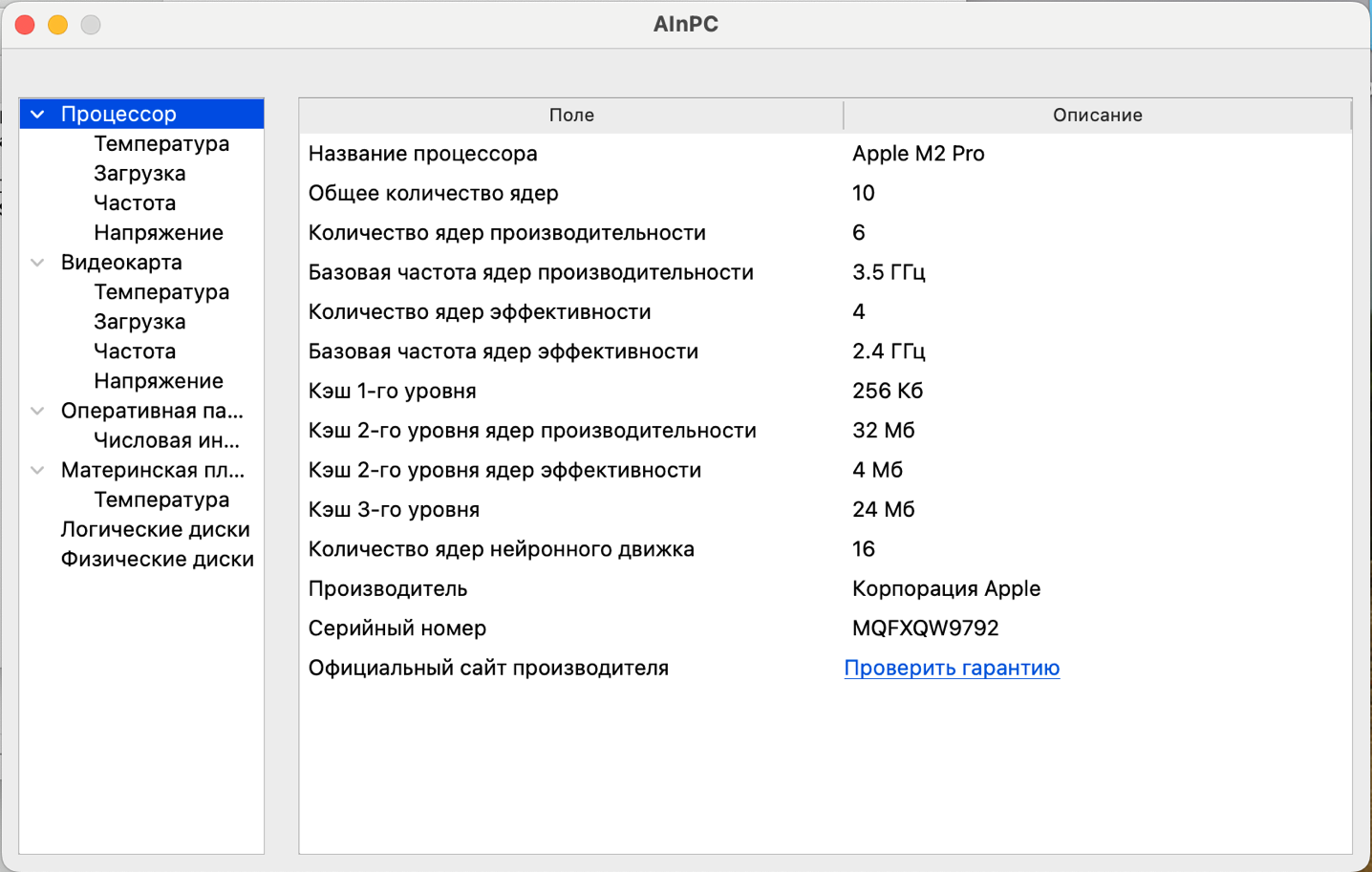


Рис. 23. Скриншот окна первого уровня «Процессор»

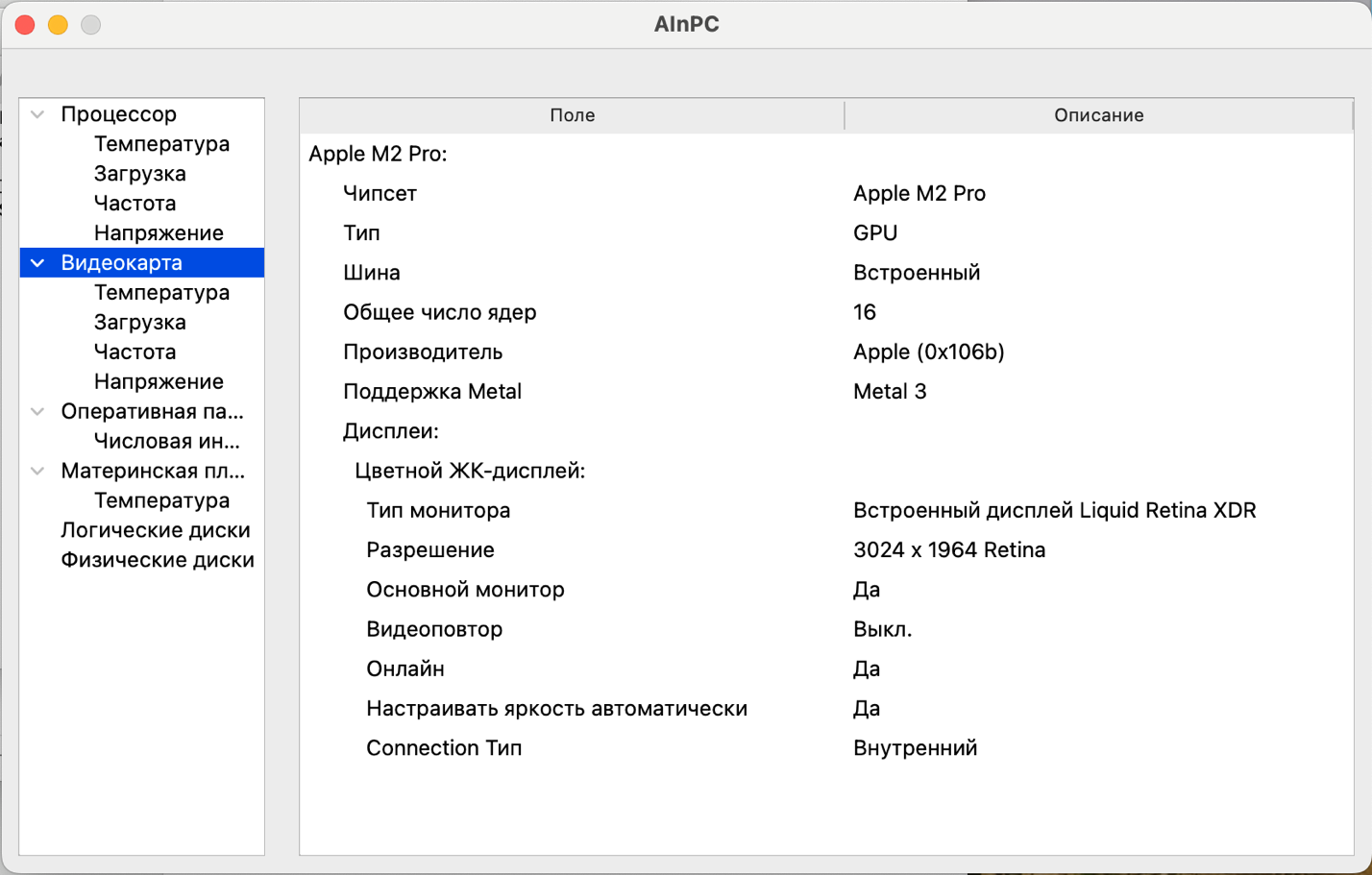


Рис. 24. Скриншот окна первого уровня «Видеокарта»

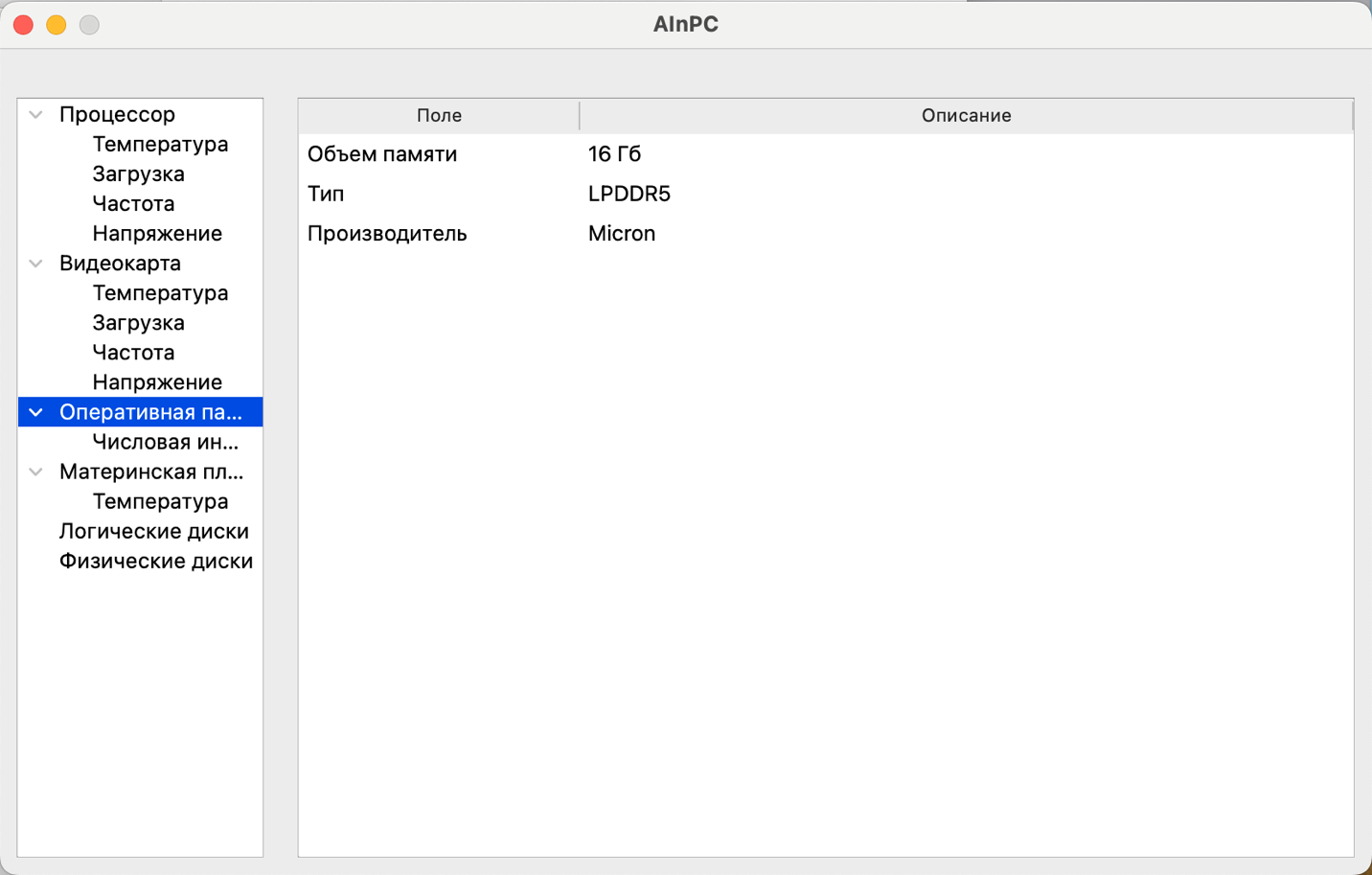


Рис. 25. Скриншот окна первого уровня «Оперативная память»

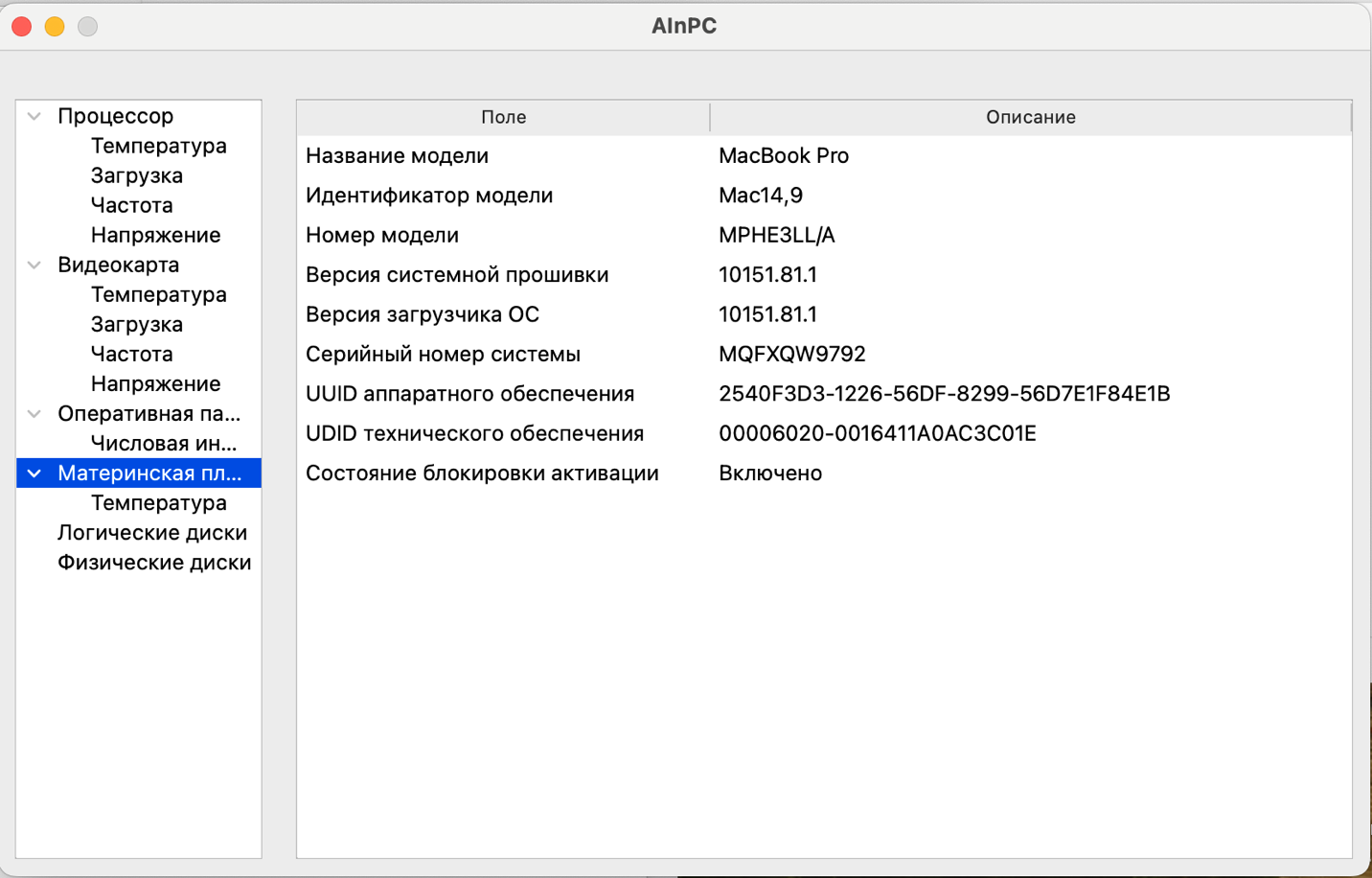


Рис. 26. Скриншот окна первого уровня «Материнская плата»

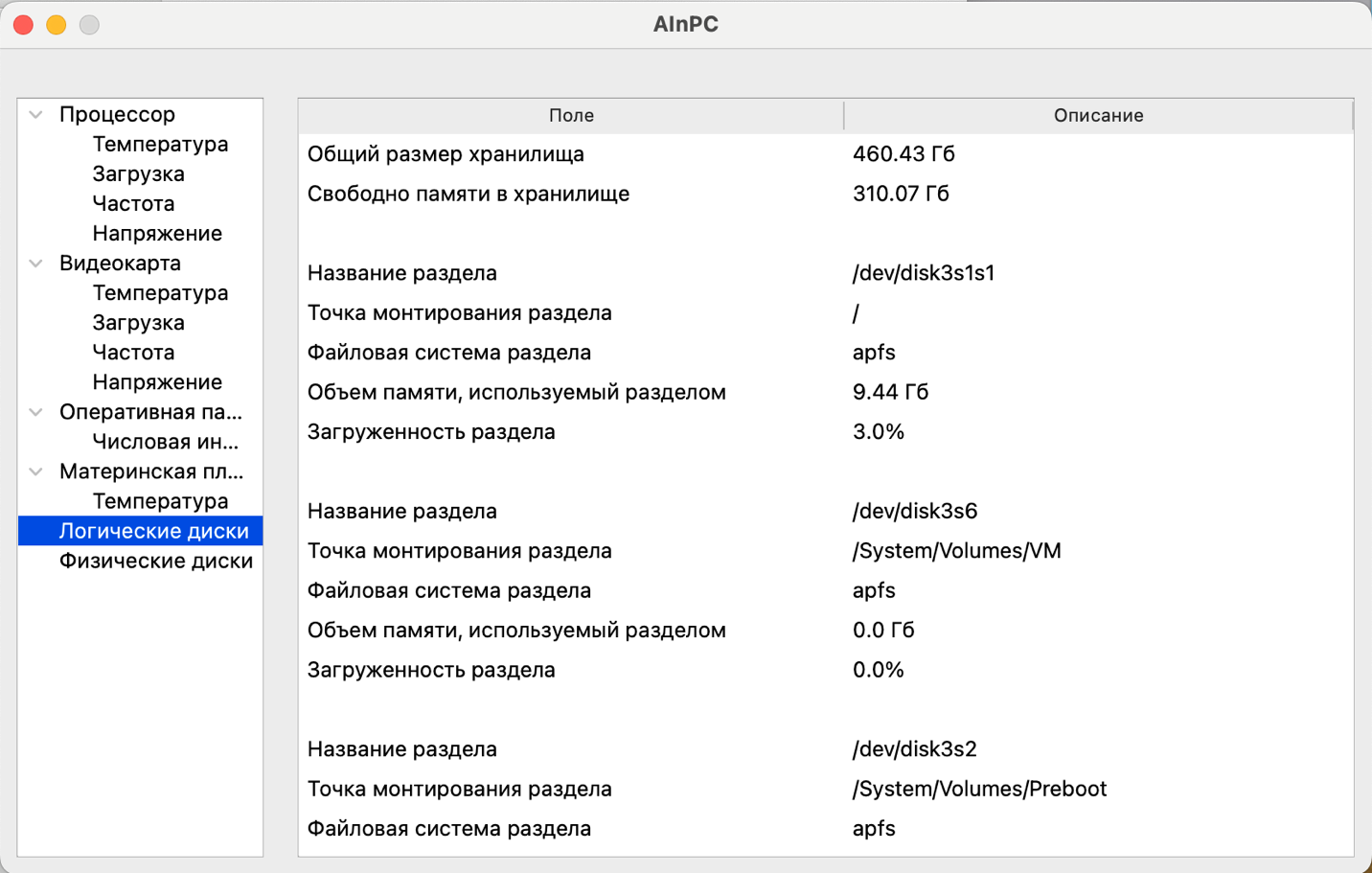


Рис. 27. Скриншот окна первого уровня «Логические диски»

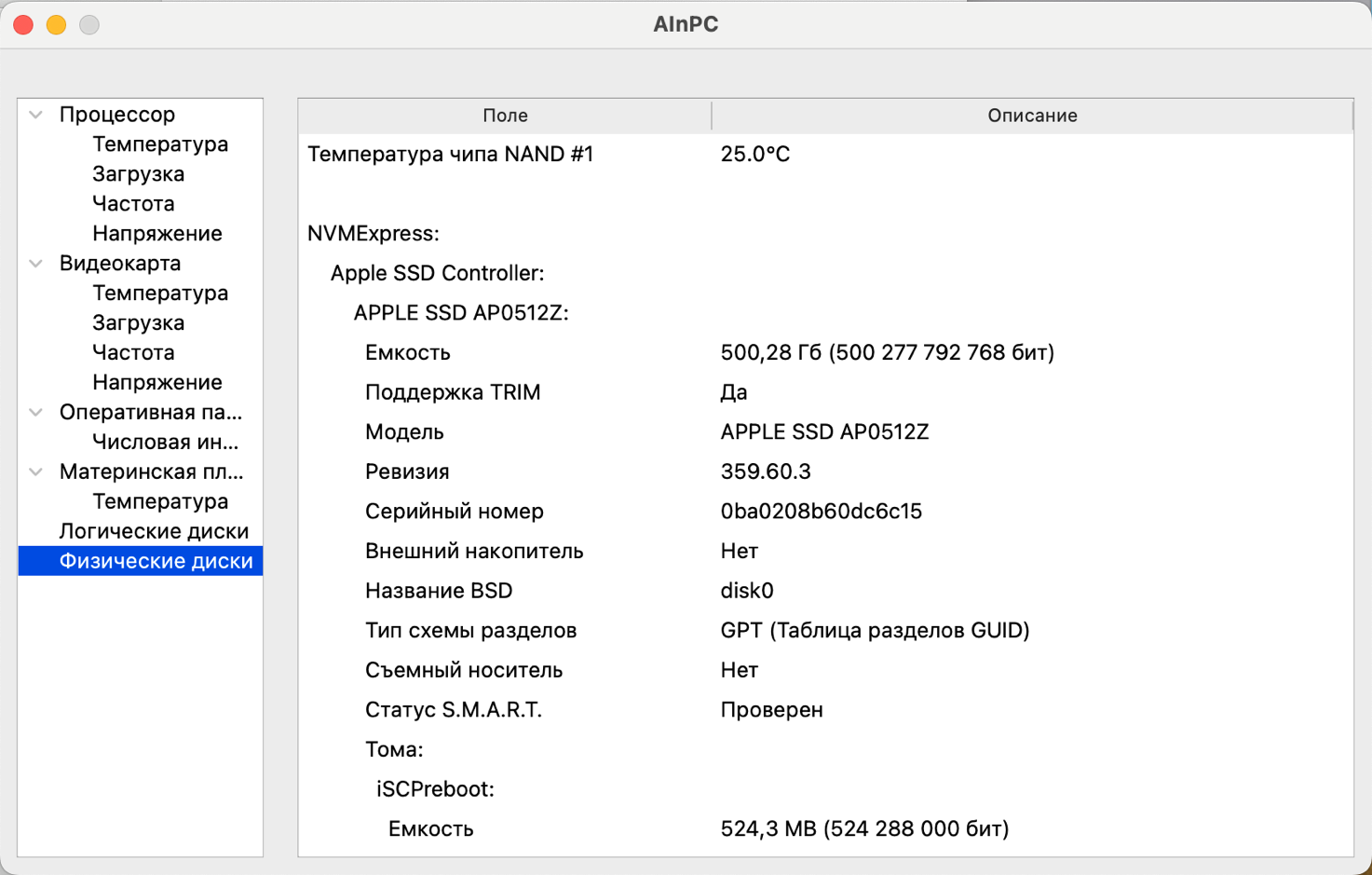


Рис. 28. Скриншот окна первого уровня «Физические диски»

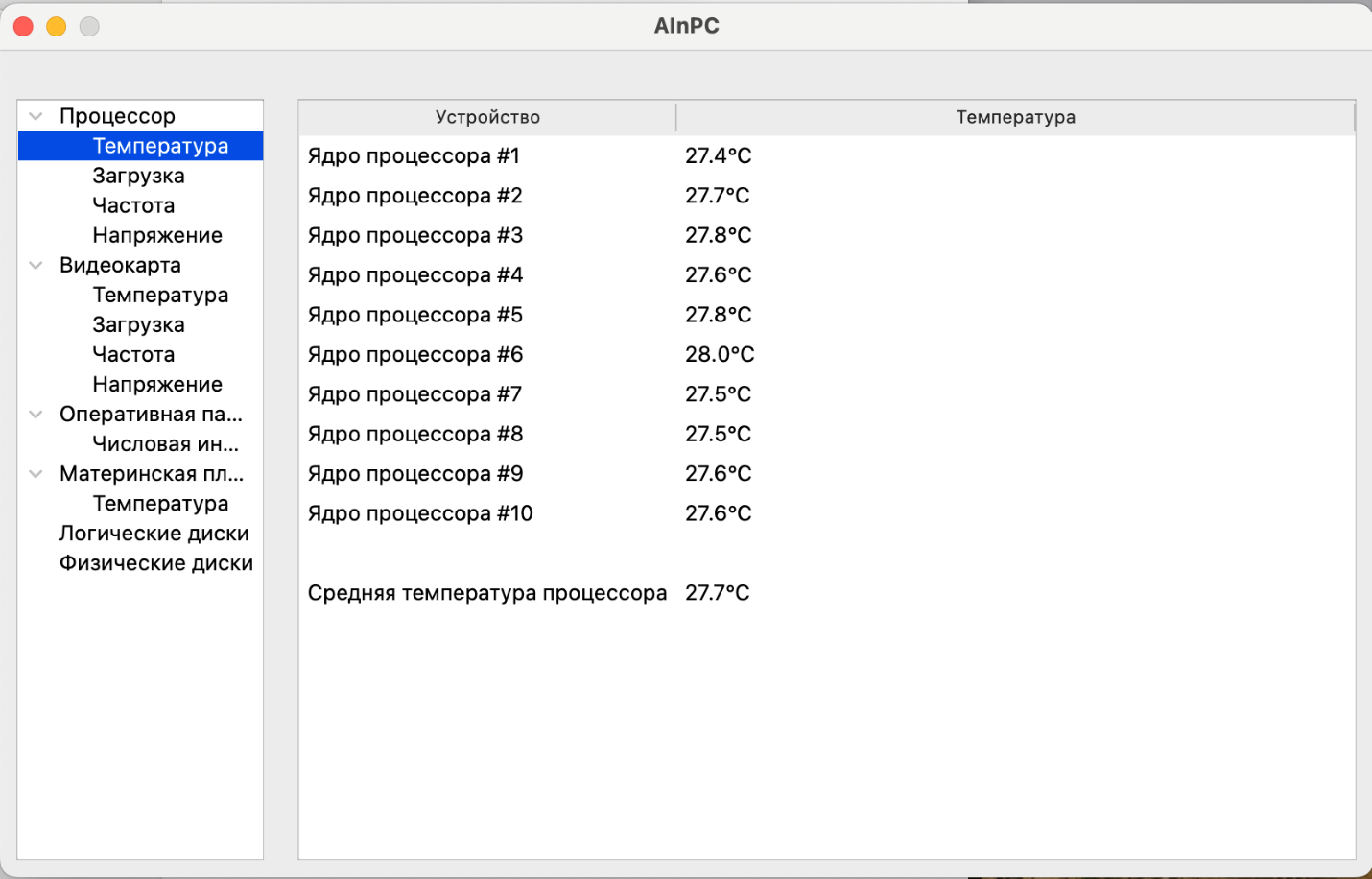


Рис. 29. Скриншот окна второго уровня «Процессор – Температура»

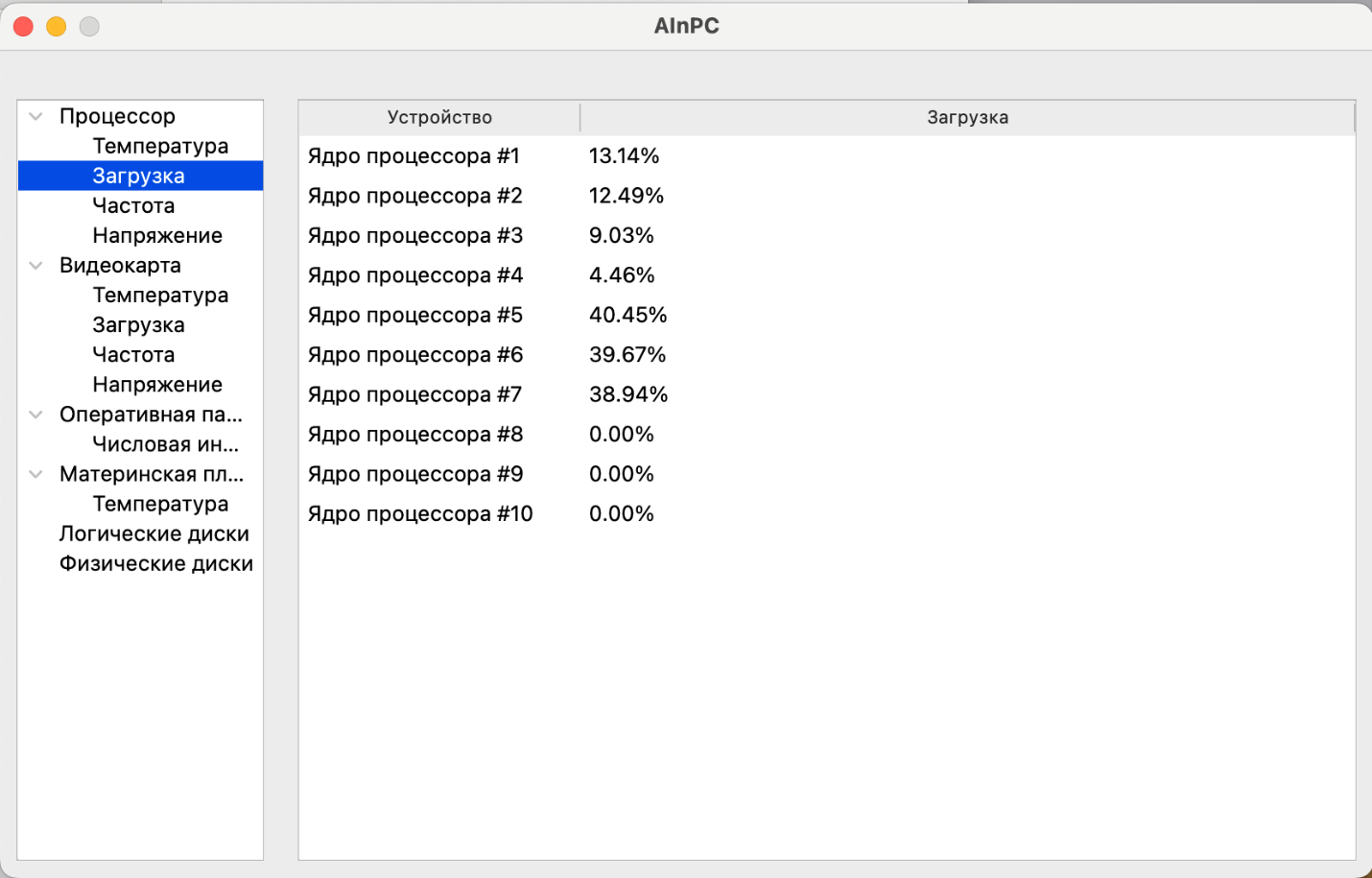


Рис. 30. Скриншот окна второго уровня «Процессор – Загрузка»

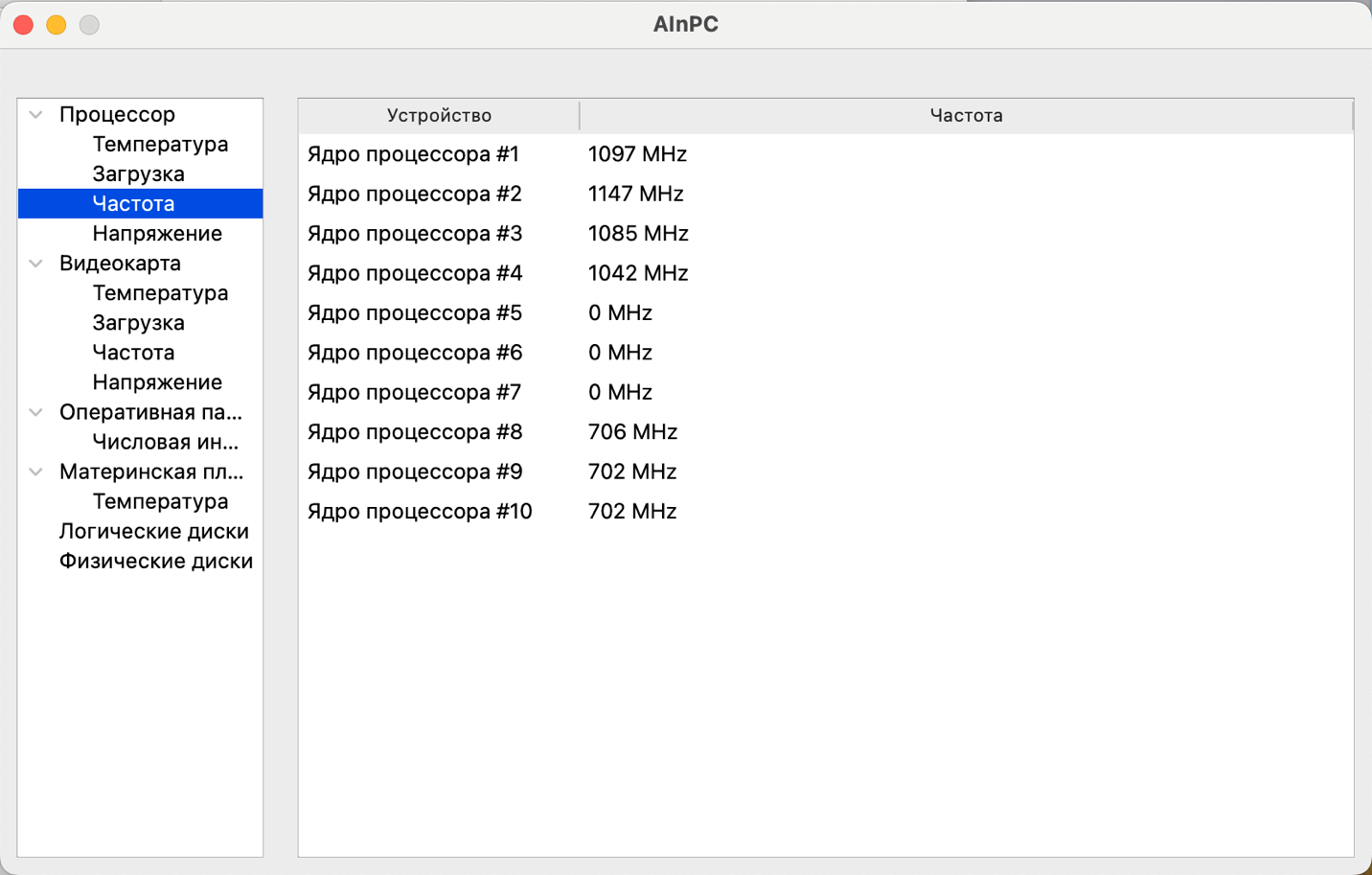


Рис. 31. Скриншот окна второго уровня «Процессор – Частота»

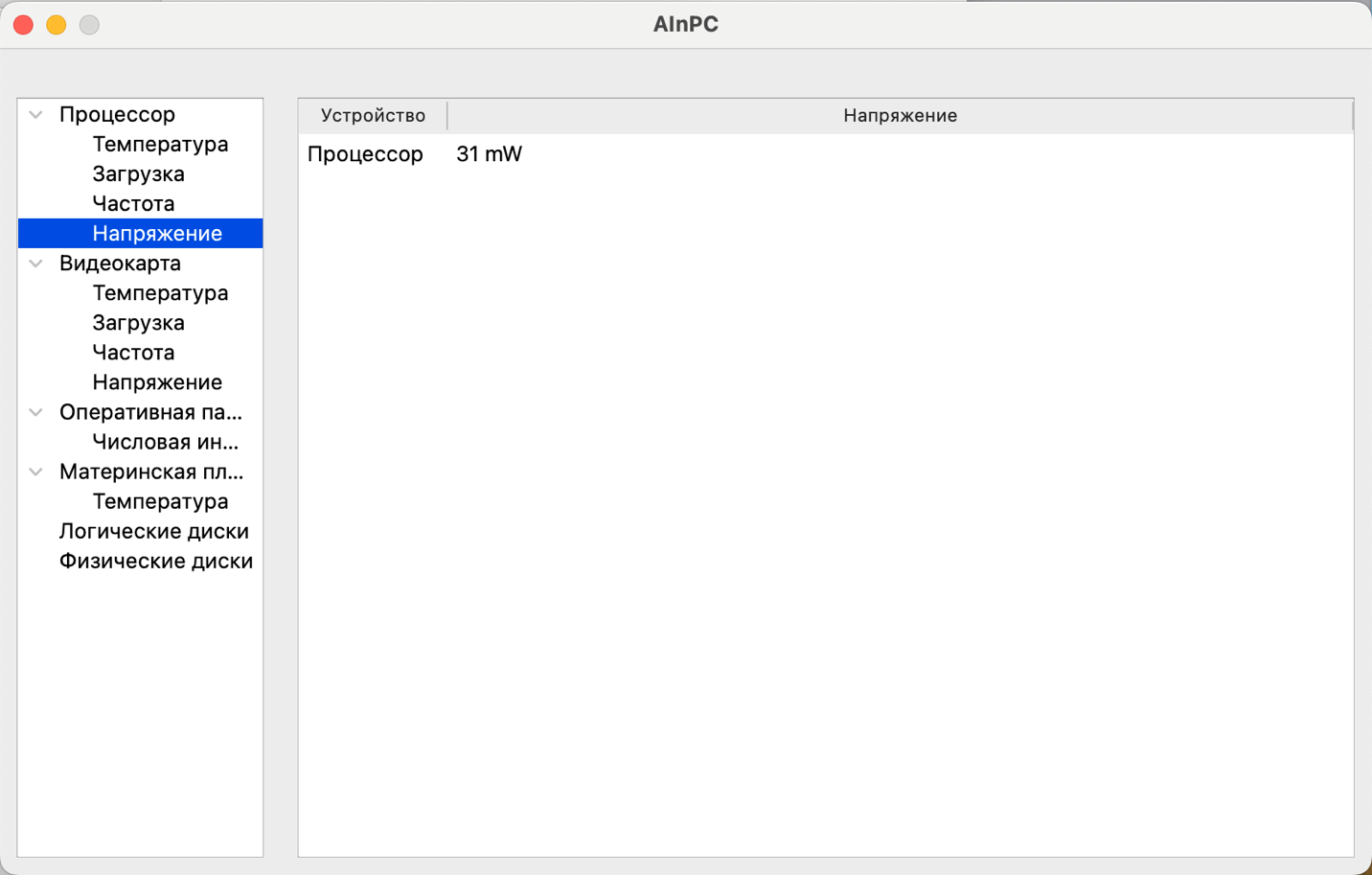


Рис. 32. Скриншот окна второго уровня «Процессор – Напряжение»

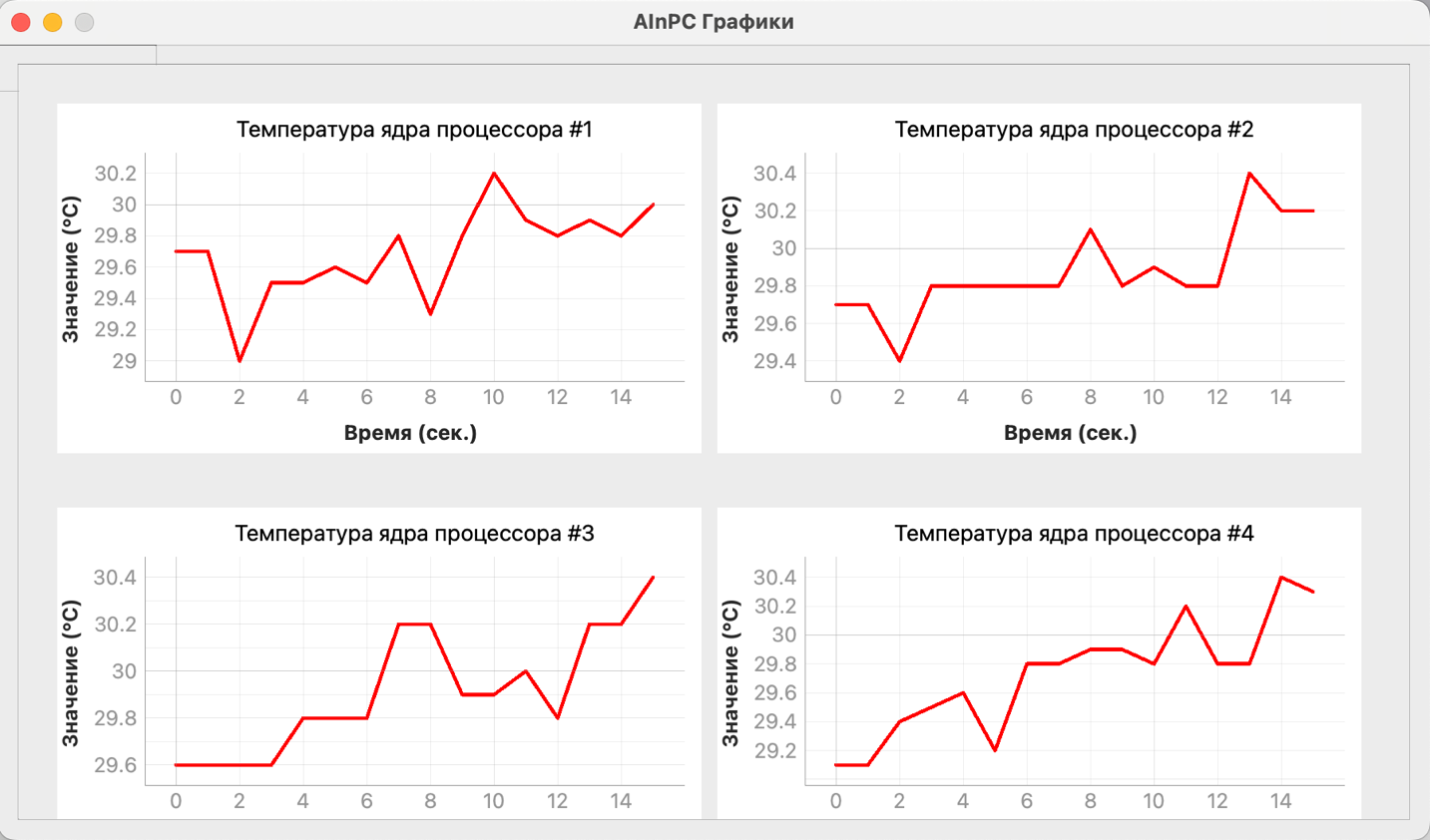


Рис. 33. Скриншот окна «Графики»

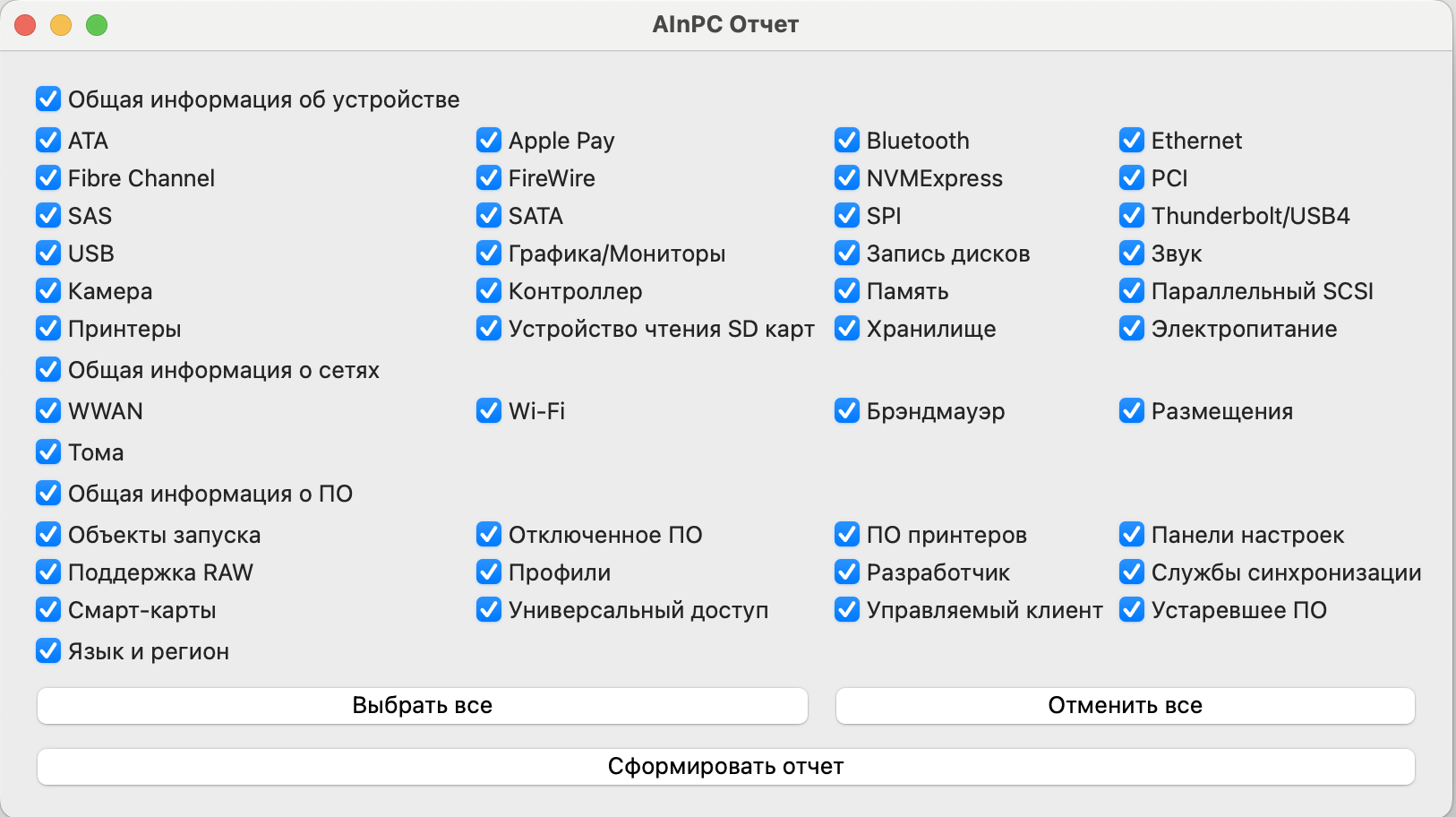


Рис. 34. Скриншот окна «Отчет»

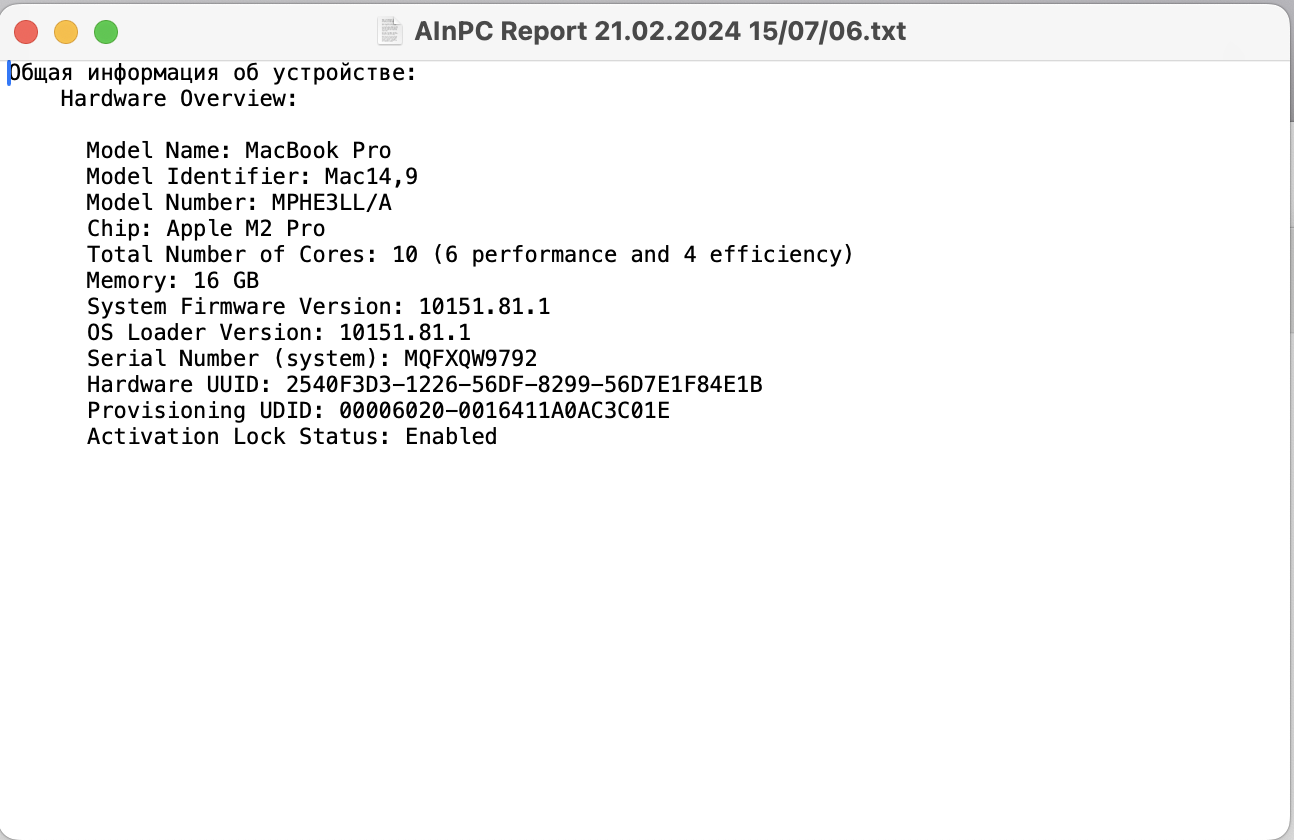


Рис. 35. Скриншот сформированного отчета

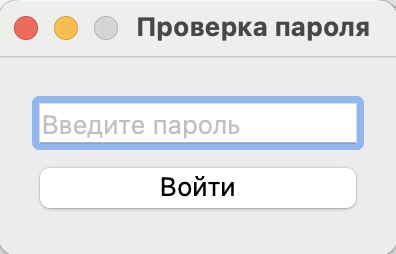


Рис. 36. Скриншот окна ввода пароля

### 6.4. Настройка системы

Информационная система не предполагает никаких настроек.

Единственный вариант, при котором пользователь может вносить изменения – это окно формирования отчета, оно подразумевает выбор пользователем компонентов, информация о которых его интересует.

### 6.5. Руководство пользователя

Информационная система не предполагает наличия инструкции, так как большой упор в ней сделан на интуитивно-понятный интерфейс.

# Заключение

Таким образом, в ходе выполнения курсового проекта в течение двух семестров по дисциплине «Методы и средства проектирования информационных систем» была создана проектная модель, которая содержит в себе все необходимые модели, которые отражают все стороны проекта, которые могут быть использованы в ходе всего периода разработки ПО.

За время, на протяжении которого выполнялась работа, были приобретены навыки работы с редакторами диаграмм и блок-схем, с учебниками и методическими материалами, описывающими процессы разработки и реализации модели.

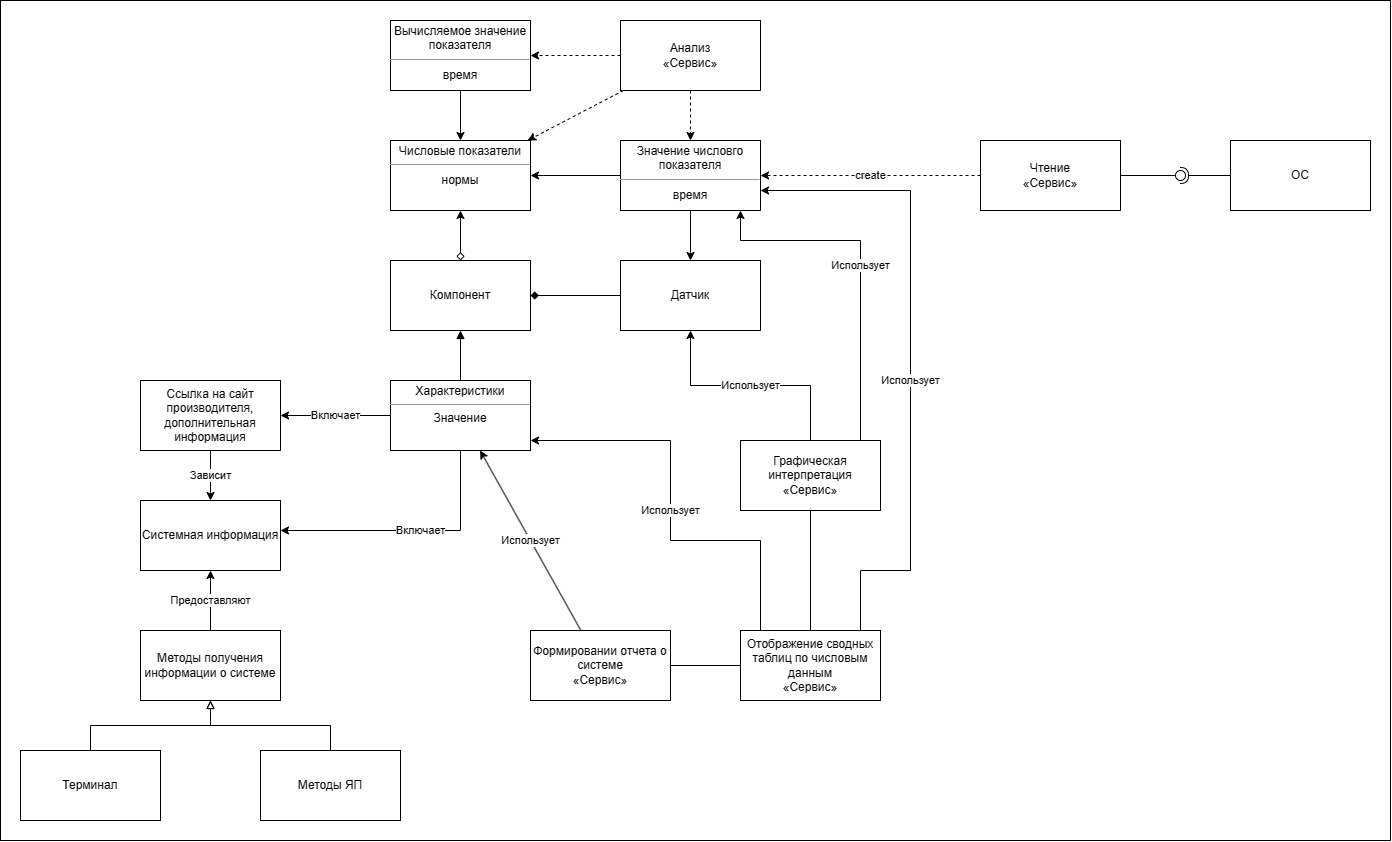
Также, за время работы были проработаны некоторые теоретические аспекты работы над проектами, которые могут помочь в дальнейшей работе в команде или над собственными разработками.

# Список источников

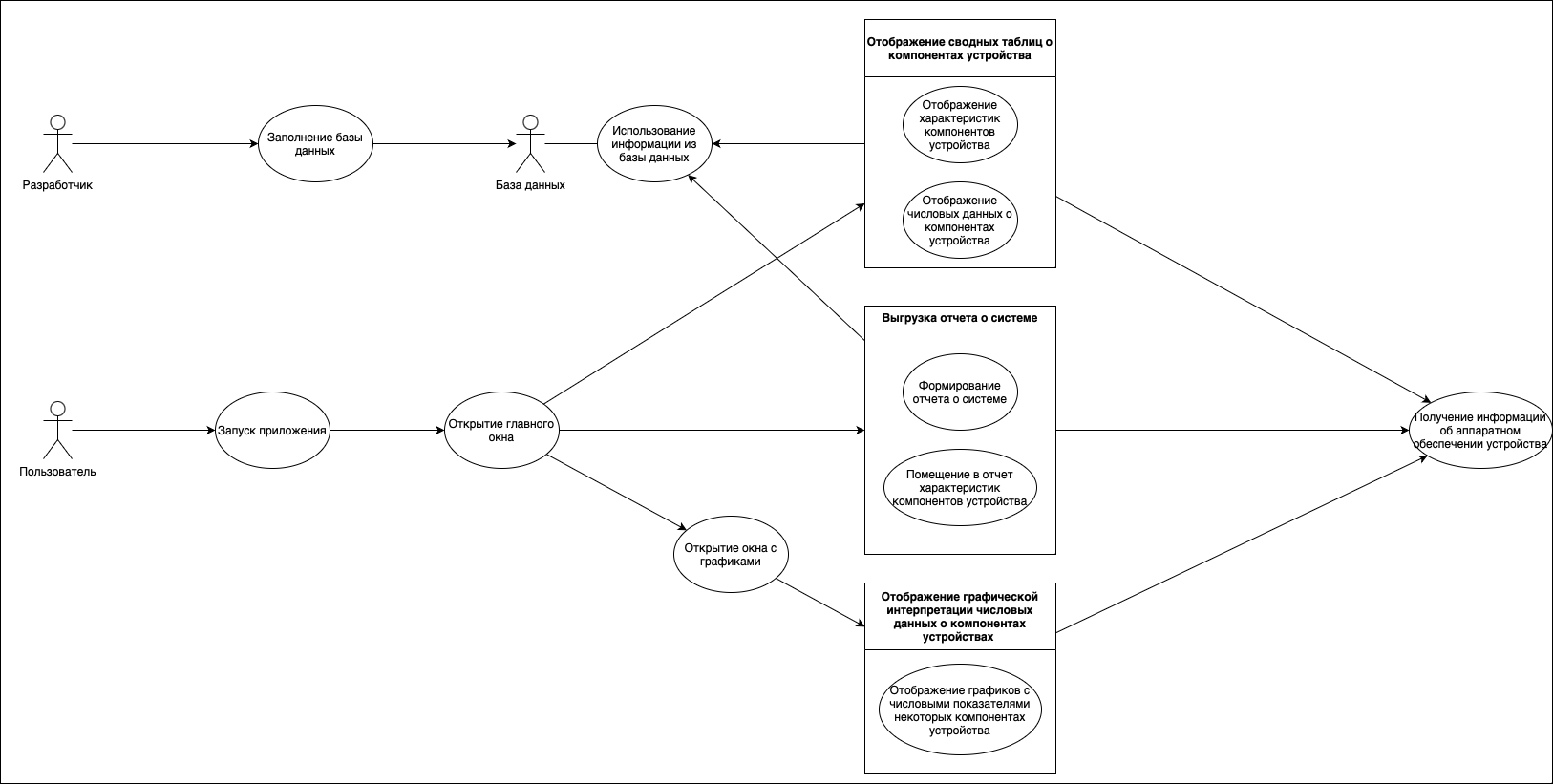
1. Рост продаж в первом квартале на 33% до 1,8 млн ПК / [Электронный ресурс] // TADVISER. Государство. Бизнес. Технологии — URL: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D1%8B\_(%D1%80%D1%8B%D0%BD%D0%BE%D0%BA\_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8)#.D0.A0.D0.BE.D1.81.D1.82\_.D0.BF.D1.80.D0.BE.D0.B4.D0.B0.D0.B6\_.D0.B2\_.D0.BF.D0.B5.D1.80.D0.B2.D0.BE.D0.BC\_.D0.BA.D0.B2.D0.B0.D1.80.D1.82.D0.B0.D0.BB.D0.B5\_.D0.BD.D0.B0\_33.25\_.D0.B4.D0.BE\_1.2C8\_.D0.BC.D0.BB.D0.BD\_.D0.9F.D0.9A (дата обращения: 10.10.2023).
2. В России количество компьютеров на четверть превысило число жителей / [Электронный ресурс] // Прайм. Агентство экономической информации — URL: https://1prime.ru/telecommunications\_and\_technologies/20201020/832186875.html (дата обращения: 10.10.2023).
3. Ларман К. Применение UML 2.0 и шаблонов проектирования. – 2007. – С. 340, 829.
4. Ревунков Г. И., Маслеников К. Ю. Построение модели предметной области автоматизированных систем //Информационно-измерительные и управляющие системы. – 2016. – Т. 14. – №. 12. – С. 51-53.
5. Бурсак А. Как бизнес учится обходиться без иностранных IT-решений / Бурсак А. [Электронный ресурс] // Ведомости. Технологии — URL: https://www.vedomosti.ru/technologies/innovation\_policy/articles/2023/10/12/1000157-kak-biznes-uchitsya-obhoditsya-bez-inostrannih-it-reshenii (дата обращения: 27.11.2023).
6. Ошанина О. Как продвигается импортозамещение в сфере IT / Ошанина О. [Электронный ресурс] // Ведомости&. — URL: https://www.vedomosti.ru/partner/articles/2023/06/27/982631-kak-prodvigaetsya-importozameschenie-v-it (дата обращения: 27.11.2023).

# Приложения

## Приложение 1. Концептуальная модель информационной системы



## Приложение 2. Диаграмма вариантов использования



## Приложение 3. Диаграмма классов

