

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**«ПРОГРАММА УПРАВЛЕНИЯ
МАЙНИНГ-УСТРОЙСТВАМИ ПО
ЛОКАЛЬНОЙ СЕТИ»**

Студент М.Пяткин 20.05.26 Пяткин Максим Эдуардович
(подпись, дата) (ФИО полностью)
Руководитель Т.Ю. Леонова 20.05.26 Леонова Т.Ю.
(подпись, дата) (фамилия, инициалы)
Нормоконтролёр А.С. Бычков 25.05.26 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой М.А. Митрохин М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от
Июнь 2026г., № 25)

Секретарь ГЭК Е.И. Гурин Е.И. Гурин
(подпись) (фамилия, инициалы)
Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«07» апреля 2026 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Пяткин Максим Эдуардович гр. 22ВВП1 факультета ФВТ

2. Тема работы программа управления майнинг-устройствами локальной сети

Тема утверждена приказом ПГУ № 347/о от " 07 " апреля 2026 г.

3. Руководитель бакалаврской работы Леонова Т.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
создание программного инструмента для централизованного управления и мониторинга оборудования майнинг-центра, позволяющего автоматизировать рутинные процессы подготовки отчетности и настройки устройств. Исходные данные включают наличие сети с ASIC-майнерами, поддерживающими стандартные веб-интерфейсы управления, а также необходимость сбора технических параметров (хэшрейт, температура, время работы) и выполнения групповых операций (перезагрузка, изменение конфигурации пулов). Разрабатываемая программа решает задачу ускорения подготовки отчетности за счет автоматического сканирования сети, централизованного отображения данных и возможности массового управления оборудованием, что сокращает временные затраты персонала, исключает ошибки ручного ввода и позволяет оперативно отслеживать ключевые показатели работы устройств для повышения их энергоэффективности.

5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметных областей: изучить существующие информационные системы, выделить их достоинства и недостатки. Выявить потребности пользователей. Выбрать язык программирования, среду разработки, изучить вопрос электротехники и схемотехники. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал Visual Studio для разработки программы для управления майнинг-устройствами в локальной сети.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	07.04.2026 – 10.04.2026	lead
2	Анализ требований	32	11.04.2026 – 14.04.2026	lead
3	Создание макета графического интерфейса	30	15.04.2026 – 18.04.2026	lead
4	Выбор технологий, применяемых при проектировании и разработке	40	19.04.2026 – 24.04.2026	lead
5	Реализация графического интерфейса	66	25.04.2026 – 03.05.2026	lead
6	Разработка основной части программы	70	04.05.2026 – 13.05.2026	lead
7	Оформление пояснительной записки	54	14.05.2026 – 21.05.2026	lead

Дата выдачи задания "07" апреля 2026 г.

Руководитель бакалаврской работы lead 07.04.2026 Леонова Т.Ю.
(подпись, дата)

Задание к исполнению принял студент М.Пяткин 07.04.2026 Пяткин М.Э.
(подпись, дата)

Работу к защите допустить
Декан факультета Л.Р.Фионова 05.06.2026 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 107 листов, 47 рисунков, 1 таблицу, 6 источников, 3 приложения.

УПРАВЛЕНИЕ, ПРОГРАММИРОВАНИЕ, UML, МАЙНИНГ, МОНИТОРИНГ, WEB, СЕТЬ, ПОЛЬЗОВАТЕЛЬ, ПРОГРАММА, МИНИМИЗАЦИЯ, ЭНЕРГОЭФФЕКТИВНОСТЬ

Цель работы – разработать программу для управления майнинг-устройствами в локальной сети.

Одним из путей достижения поставленной задачи является разработка программы, обеспечивающая выполнение следующих функций: возможность работы с большим количеством устройств, простота в эксплуатации, стабильность работы системы, возможность изменения характеристик нескольких устройств, возможность дополнения системы новыми функциями.

В конечном итоге разработана программа для управления майнинг-устройствами в локальной сети, которая позволяет эффективно снизить нагрузку с человека, автоматизировать процессы добычи криптовалюты и увеличить энергоэффективность майнинг-центра. Данная программа полностью соответствует цели данной выпускной квалификационной работы. В ходе достижения результата, были использованы навыки электроники, электротехники и схемотехники. Данная программа протестирована и уже эксплуатируется.

					ПГУ1.090301.21.001 ПЗ				
Эм.	Лист	№ докум.	Подпись	Дата	Программа управления майнинг-устройствами в локальной сети Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Пяткин М.Э.		20.05.21			2	107	
Провер.		Леонова Т.Ю.		20.05.21					
В. Контр.		Бычков А.С.		25.05.21					
Утверд.		Митрохин М.А.		25.05.21					
						ФВТ, каф. ВТ гр. 22ВВП1			

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Программа управления майнинг-устройствами по локальной сети.....	8
1.1 Анализ предметной области	8
1.2 Обзор аналогов	9
1.3 Анализ требований к разработке	12
1.4 Выбор архитектуры системы	14
1.5 Выбор языка программирования	17
1.6 Выбор среды разработки	18
1.7 Разработка управляющей программы	27
1.8 Графическая часть разрабатываемой программы.....	35
1.8.1 Сценарий «Поиска устройства».....	37
1.8.2 Сценарий «Перезагрузка устройства»	38
1.8.3 Сценарий «Установка пулов»	40
1.8.4 Сценарий «Загрузка пулов»	42
1.8.5 Сценарий «Экспорт в Excel».....	43
1.8.6 Дополнительные полезные функции программы	44
2 Требования безопасности по охране труда при работе с ПЭВМ	48
2.1 Организация рабочего места.....	49
2.2 Освещение рабочего места.....	50
2.3 Электробезопасность	51
2.4 Пожарная безопасность	51
2.5 Режим труда и отдыха	52
Заключение	54
Список использованных источников	55

Приложение А — UML диаграмма и схемы	57
Приложение Б — Листинг.....	63
Приложение В — Презентация.....	107

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

Перечень сокращений и обозначений

API (Application Programming Interface) — программный интерфейс приложения набор правил и инструментов для взаимодействия одной программы с другой

ASIC (Application-Specific Integrated Circuit) — интегральная схема специального назначения; специализированное устройство для решения конкретной задачи (в контексте работы — для майнинга криптовалют)

CGI (Common Gateway Interface) — общий шлюзовой интерфейс; технология взаимодействия веб-сервера с исполняемыми программами для обработки HTTP-запросов, используемая в веб-интерфейсе майнеров

CSV (Comma-Separated Values) — текстовый формат файла, в котором значения данных разделяются запятой или точкой с запятой

HTTP (HyperText Transfer Protocol) — протокол передачи гипертекста, используемый для обмена данными между клиентом и сервером

JSON (JavaScript Object Notation) — текстовый формат обмена данными на основе синтаксиса JavaScript, применяемый в работе для обработки ответов от майнеров.

MD5 (Message Digest 5) — 128-битный криптографический алгоритм хеширования, используемый в механизме аутентификации доступа

UML (Unified Modeling Language) — унифицированный язык моделирования, применяемый для визуализации, проектирования и документирования архитектуры программных систем

URL (Uniform Resource Locator) — унифицированный указатель ресурса; адрес конкретного веб-ресурса или файла в сети.

UTF-8 (Unicode Transformation Format, 8-bit) — один из основных стандартов кодирования текстовых символов

ZIP — популярный формат сжатия и архивирования файлов без потерь

Введение

Необходимость автоматизации процессов управления и сбора отчетности имеет место быть практически в любой сфере деятельности человека, особенно в высокотехнологичных отраслях, где количество обслуживаемых устройств исчисляется сотнями. В зависимости от масштабов производства требуется значительное время на выполнение рутинных задач, которые можно оптимизировать путем реализации программных решений. Тема криптовалют и майнинга становится все более популярной в наше время, причем она набирает обороты не на уровне обывателя, а в больших промышленных масштабах. В связи с этим появляются технические проблемы, будь то ремонт оборудования или сбои в работе системы, а главное — необходимость оперативного управления большим парком устройств. Ведь одной из важнейших характеристик майнинг-системы является ее стабильная и эффективная работа, что напрямую зависит от качества управления каждым отдельным устройством.

Чтобы избежать данных проблем, одним из ключевых решений является создание системы централизованного управления, которая позволит не только формировать отчетность по работе оборудования, но и оперативно воздействовать на него, снижая нагрузку на персонал и обеспечивая контроль за функционированием всей системы. Человек во время управления техническим устройством должен рассматриваться как единая система с комплексом оборудования. Основная задача управления заключается не только в получении информации о состоянии системы, но и в возможности оперативного преобразования команд оператора в соответствующие воздействия на устройства.

Эффективность управления напрямую зависит от скорости и удобства выполнения операций над оборудованием. В условиях промышленного майнинга, где информация о хешрейте, температуре, времени работы и конфигурации пулов каждого отдельного устройства доступна через веб-интерфейсы, возникает задача не только агрегации данных, но и возможности массового управления группой устройств. Существующая практика поочередного подключения к каждому

					ПГУ1.090301.21.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

устройству для настройки пулов, перезагрузки или идентификации является критически узким местом, снижающим общую эффективность системы управления и увеличивающим время реакции на возникающие проблемы.

Целью данной работы является разработка инструмента централизованного управления, автоматизирующего процессы обнаружения, мониторинга и контроля оборудования майнинг-центра. Создаваемая программа реализует сканирование сети для автоматического поиска ASIC-майнеров, сбор телеметрии с их веб-интерфейсов, а главное — возможность выполнения групповых операций: массовой настройки пулов, удаленной перезагрузки, включения режима идентификации устройств, а также экспорт собранных данных для формирования отчетности. Такой подход позволяет унифицировать процесс управления парком оборудования, исключить влияние человеческого фактора на этапе конфигурации устройств, сократить время выполнения рутинных операций с часов до минут, а также предоставить оператору консолидированную информацию и возможность оперативного воздействия на всю систему из единого интерфейса.

					<i>ПГУ1.090301.21.001 ПЗ</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

1 Программа управления майнинг-устройствами по локальной сети

1.1 Анализ предметной области

В современной промышленности существует обширная область, связанная с эксплуатацией распределённых сетей оборудования, требующего постоянного мониторинга и централизованного управления. К таким системам относятся производственные линии, телекоммуникационное оборудование, серверные фермы, а также устройства, работающие в режиме 24/7 с высокой нагрузкой. Особенностью данной области является то, что количество обслуживаемых устройств может достигать сотен и даже тысяч единиц, а время простоя каждого элемента напрямую влияет на экономическую эффективность всего производства.

Специфика эксплуатации таких систем заключается в необходимости постоянного контроля технического состояния оборудования: температуры, производительности, времени непрерывной работы и других параметров. При этом администратор сталкивается с проблемой информационной перегрузки — ручное подключение к каждому устройству для проверки его состояния или изменения настроек становится физически невозможным при больших объёмах парка. Возникает острая потребность в автоматизации процессов обнаружения, мониторинга и группового управления оборудованием.

Дополнительной особенностью является разрозненность устройств в локальной сети, отсутствие единого централизованного интерфейса для управления всем парком. Каждое устройство, как правило, имеет собственный веб-интерфейс, доступ к которому требует отдельного подключения, авторизации и выполнения операций вручную. Это создаёт высокую нагрузку на персонал и увеличивает риск ошибок, связанных с человеческим фактором.

В условиях нарастающей тенденции к индустриальной автоматизации и цифровизации производства, выбор эффективных инструментов управления

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

распределённым оборудованием становится особенно актуальным. Современные предприятия стремятся минимизировать участие человека в рутинных операциях, перераспределяя ресурсы на решение более сложных и творческих задач.

Существующие на рынке решения либо требуют значительных финансовых вложений на приобретение лицензий и внедрение, либо обладают ограниченным функционалом, не позволяющим эффективно управлять большими парками оборудования. Многие из них требуют установки дополнительного программного обеспечения непосредственно на управляемые устройства, что сопряжено с рисками и временными затратами.

В связи с этим, разработка собственной управляющей системы, адаптированной под конкретные потребности и масштабы производства, является экономически оправданной и технически целесообразной. Возможность массового выполнения операций, автоматического обнаружения новых устройств и формирования отчётности без постоянного контроля человека позволяет значительно сократить время обслуживания и повысить общую эффективность производственного процесса.

1.2 Обзор аналогов

В рамках выполнения выпускной квалификационной работы был проведён анализ существующих программных решений, функционально пересекающихся с разрабатываемым проектом.

Были рассмотрены несколько аналогичных программ с различным функционалом.

«Antminer Tools» – данная программа представлена на рисунке 1. Antminer Toolkit – это набор инструментов для управления и мониторинга майнинг ферм Antminer, которое также включает в себя такие функции как: обновление прошивки, перезапуск майнеров, сканирование IP-адресов

майнеров в локальной сети, добавление антивирусной функции майнера и функция разгона майнера.

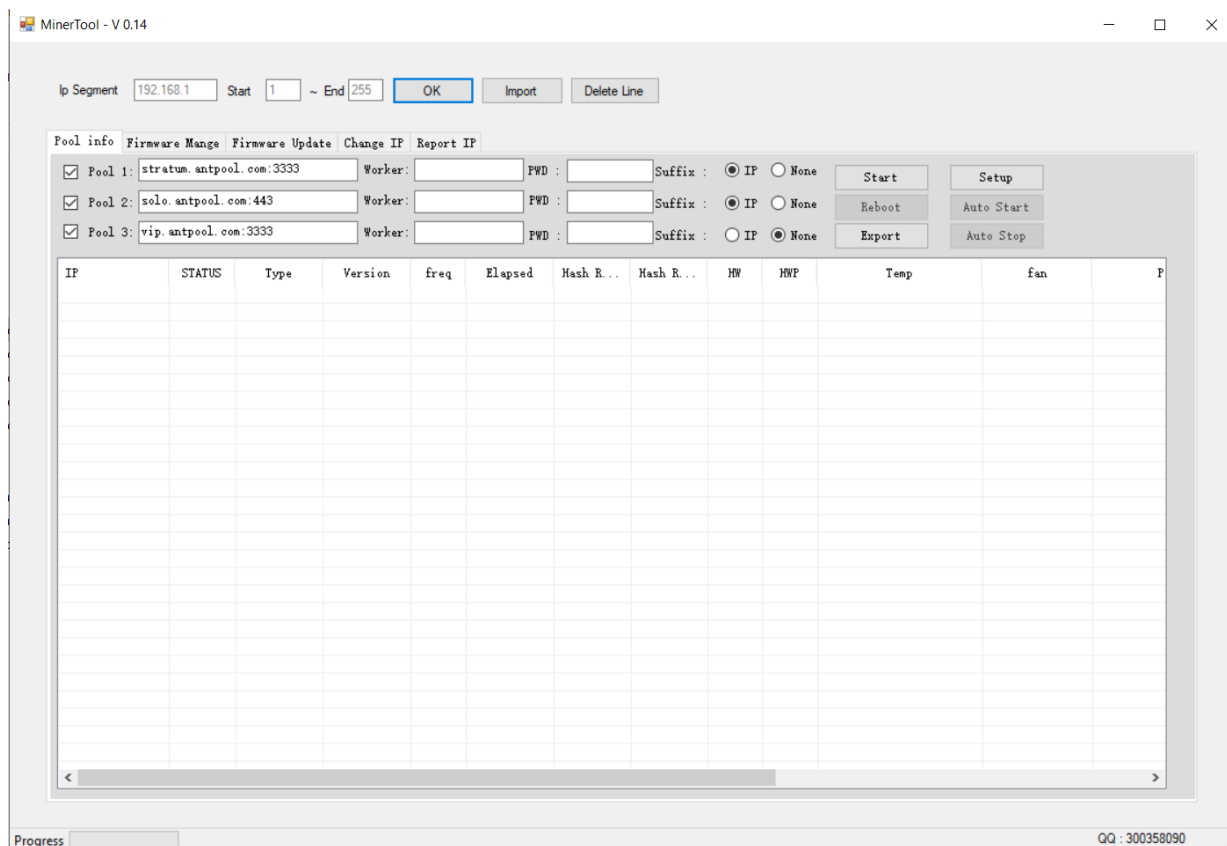


Рисунок 1 — «Antminer Tools»

PITBIT OS — это интеллектуальная система управления ASIC-майнерами, которая позволяет пользователям управлять и отслеживать работу оборудования с помощью компьютера или смартфона. Она предлагает функции, такие как автоматическая оптимизация, удаленный доступ, мониторинг в реальном времени и анализ производительности. PITBIT OS помогает увеличить хешрейт и прибыль, минимизируя издержки и риски в майнинге. Приложение представлено на рисунке 2.

BTC Tools — это программа, которая помогает сканировать, контролировать и обновлять майнеры Antminer, Avalon ASIC и другие, сортировать их, а также выполнять пакетную настройку и перезагрузку оборудования.

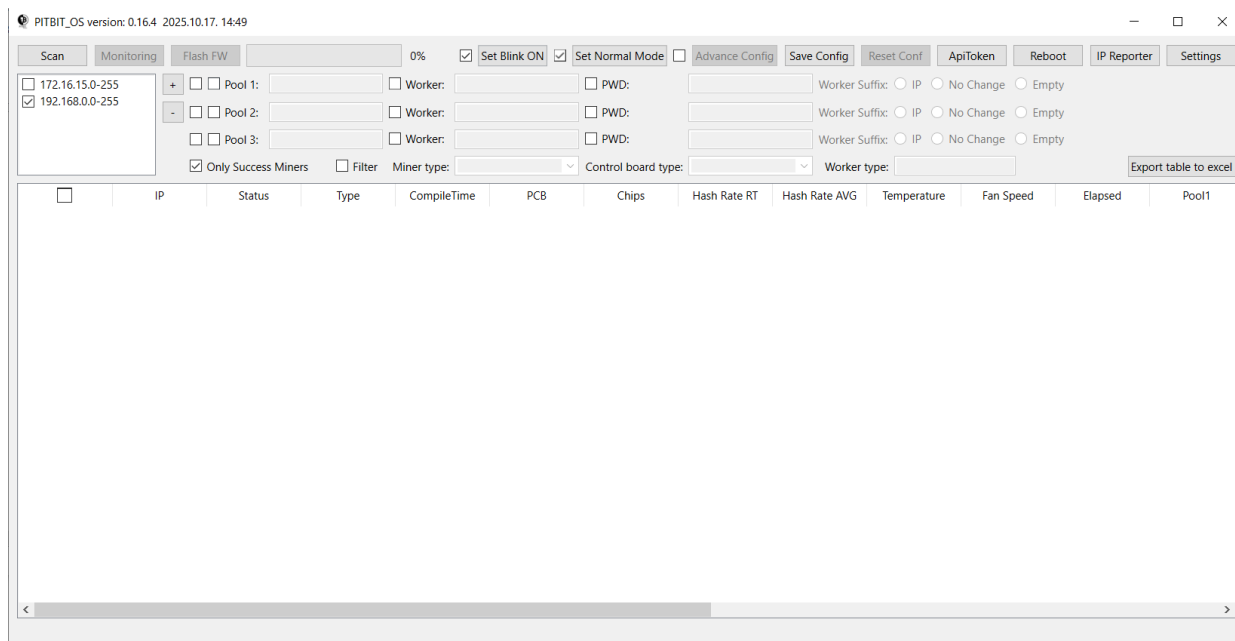


Рисунок 2 – PITBIT OS

Утилита отображает сведения о майнерах: хешрейт, температуру, скорость вращения вентиляторов, время безотказной работы, конфигурации активных пулов. Эту информацию можно сортировать, чтобы помочь выявить проблемы, влияющие на майнинг, а также автоматически выделять аномальные температуры или низкий хешрейт. Отличается простотой в использовании и подходит для опытных пользователей и новичков. BTC Tools можно скачать для Windows и Linux. Сервис представлен на рисунке 3.

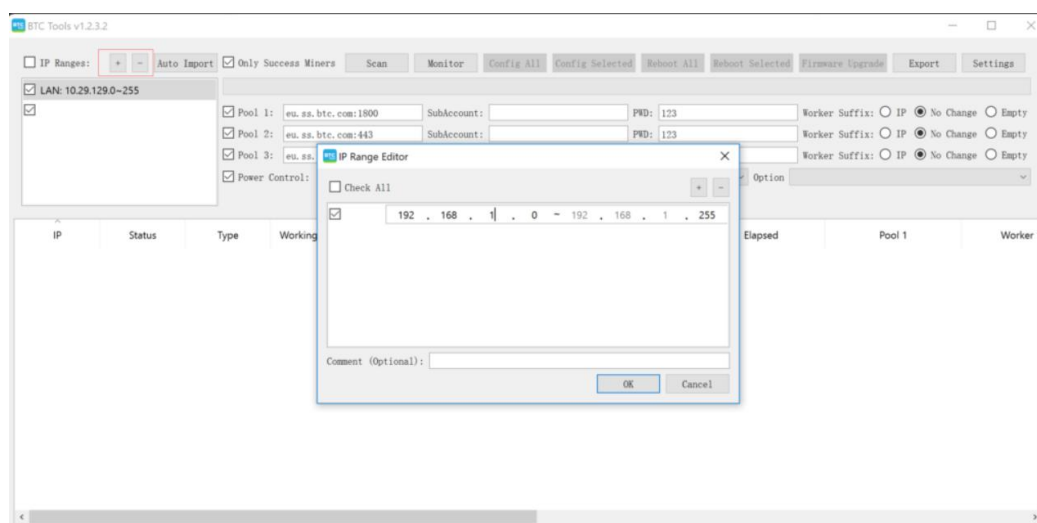


Рисунок 3 – BTC Tools

Сравнения преимуществ и недостатков представлены в таблице 1.

Таблица 1 — Сравнительная таблица аналогов

Производитель	Преимущества	Недостатки
Antminer Tools (https://shop.bitmain.com/support/download)	Бесплатный программный майнер с поддержкой нескольких алгоритмов и встроенной системой восстановления после сбоев	Требователен к аппаратному обеспечению, плохо взаимодействует с новыми устройствами
PITBIT OS (https://pitbitecosystem.com/)	Обеспечивает значительный прирост хешрейта (до 50%) за счет AI-оптимизации и автопилота	Требует рискованной замены штатной прошивки и распространяется на платной основе
BTC Tools (https://github.com/Sanberstav/BTCtools/)	Бесплатная и простая в использовании утилита для пакетной настройки и мониторинга майнеров без замены прошивки	Страдает от замедления при масштабировании и отсутствия автоматической оптимизации

На основании проведенного анализа был сделан вывод, что на рынке программ для управления майнинг-устройствами имеется потенциал для разработки новых решений.

1.3 Анализ требований к разработке

В рамках проекта разрабатывается программа для централизованного управления, мониторинга и анализа данных с ASIC-майнеров Antminer, работающих в локальной сети майнинг-центра. Программа предназначена для автоматизации процессов обнаружения оборудования, сбора технических показателей, выполнения групповых операций управления и последующего экспорта данных в структурированные отчеты, что критически важно для эффективного управления парком оборудования в майнинг-центрах.

Требуется следующий функционал:

- сканирование сети — автоматический поиск ASIC-майнеров в заданных диапазонах IP-адресов с возможностью сохранения и загрузки списков диапазонов;
- аутентификация на оборудовании — поддержка Digest-аутентификации для доступа к веб-интерфейсам майнеров с учетными данными по умолчанию (root/root);
- автоматический сбор данных— получение технических показателей (хэшрейт реальный и идеальный, температура, время работы, MAC-адрес, модель устройства) через CGI-интерфейсы майнеров с последующим извлечением данных из JSON-ответов;
- отображение собранных данных в табличном виде — визуализация информации о найденных устройствах в DataGridView с возможностью сортировки и множественного выбора строк;
- управление группой устройств — возможность одновременного выполнения операций над выбранными майнерами: настройка пулов (URL, Worker, Password), перезагрузка, включение/выключение режима идентификации Blink;
- загрузка и установка конфигурации пулов — возможность загрузить текущие настройки пулов с выбранного устройства в поля интерфейса и применить их к группе других устройств;
- экспорт данных в табличный формат — выгрузка собранных данных о всех обнаруженных майнерах в CSV-файл для последующего анализа в Excel;
- интуитивно понятный интерфейс — удобная навигация, разделение на логические зоны (таблица устройств, панель управления, настройка пулов, зона сканирования), цветовое выделение элементов управления;

– визуальная индикация состояния устройств — отображение статуса Blink в виде анимированного мигающего индикатора для быстрой идентификации устройства в стойке.

Помимо этого, для реализации и корректной работы программы были задействованы следующие программные компоненты и библиотеки:

– .NET WinForms, для создания графического интерфейса пользователя с использованием контейнеров SplitContainer, элементов управления DataGridView, CheckedListBox, ProgressBar и кастомной отрисовки ячеек;

– System.Net.HttpWebRequest, для выполнения HTTP-запросов к CGI-интерфейсам майнеров с поддержкой Digest-аутентификации и обработкой cookies;

– System.Security.Cryptography, для реализации MD5-хеширования, необходимого для вычисления параметров Digest-аутентификации (HA1, HA2, response);

– System.Text.Json, для парсинга JSON-ответов от майнеров (системная информация, статистика, конфигурация пулов, статус Blink) и сериализации новых конфигураций.

1.4 Выбор архитектуры системы

Для обеспечения стабильной и удобной работы программы выбрана событийно-ориентированная архитектура с асинхронной обработкой операций. Такой подход позволяет программе оставаться отзывчивой даже при выполнении длительных задач, таких как сканирование сети или настройка большого количества устройств. Весь процесс работы системы можно описать следующим образом:

– пользователь взаимодействует с интерфейсом. Оператор нажимает кнопку (например, «Сканировать», «Установить пулы» или «Перезагрузка»), программа фиксирует это событие и передает управление соответствующему обработчику;

– запуск асинхронной операции. Чтобы интерфейс не «зависал» на время выполнения задачи, обработчик запускает операцию в фоновом режиме с использованием механизма `async/await`. Пользователь может продолжать работать с программой или наблюдать за ходом выполнения через индикаторы прогресса;

– сетевое взаимодействие с устройствами. В зависимости от выполняемой задачи программа либо сканирует сеть для поиска майнеров, либо обращается к уже найденным устройствам для получения данных или отправки команд. Для этого используются стандартные HTTP-запросы, которые направляются на CGI-интерфейсы майнеров;

– аутентификация на майнере. При первом обращении к устройству программа проходит процедуру Digest-аутентификации, которая требуется для доступа к данным и управления. После успешной авторизации все последующие запросы выполняются с использованием полученных параметров авторизации;

– сбор и обработка данных. Полученные от майнера данные (системная информация, статистика работы, конфигурация пулов) представлены в формате JSON. Программа извлекает из них необходимые показатели: хэшрейт, температуру, время работы, MAC-адрес, модель устройства и настройки пулов;

– отображение информации. Извлеченные данные преобразуются в удобный для отображения формат и добавляются в таблицу. Для каждого найденного майнера создается отдельная строка, где отображаются все его технические характеристики. При работе с большим количеством устройств таблица обновляется динамически по мере обнаружения новых майнеров;

– выполнение команд управления. При выборе нескольких устройств и нажатии кнопки управления программа последовательно отправляет соответствующие команды каждому выбранному майнеру. Команды могут быть разными: изменение настроек пулов (URL, Worker, пароль), перезагрузка

устройства, включение или выключение режима идентификации Blink (мигание светодиодами для визуального поиска майнера в стойке);

– сохранение результатов. После завершения операций программа обновляет отображение данных в таблице, выводит сообщение о результате в статусную строку и при необходимости сохраняет информацию в файл. Для экспорта отчетности используется формат CSV, который легко открывается в Excel. Списки IP-диапазонов автоматически сохраняются в текстовый файл и загружаются при следующем запуске;

– возврат в режим ожидания. После выполнения всех действий программа возвращается в состояние готовности к следующим командам пользователя. Оператор может продолжить работу — выбрать другие устройства, изменить настройки или запустить новое сканирование.

Для более детальной и наглядной демонстрации была разработана диаграмма разворачивания, которая представлена на рисунке 4.

И для полного понимания работы программы была описана диаграмма последовательностей. Данная диаграмма показана на рисунке 5.

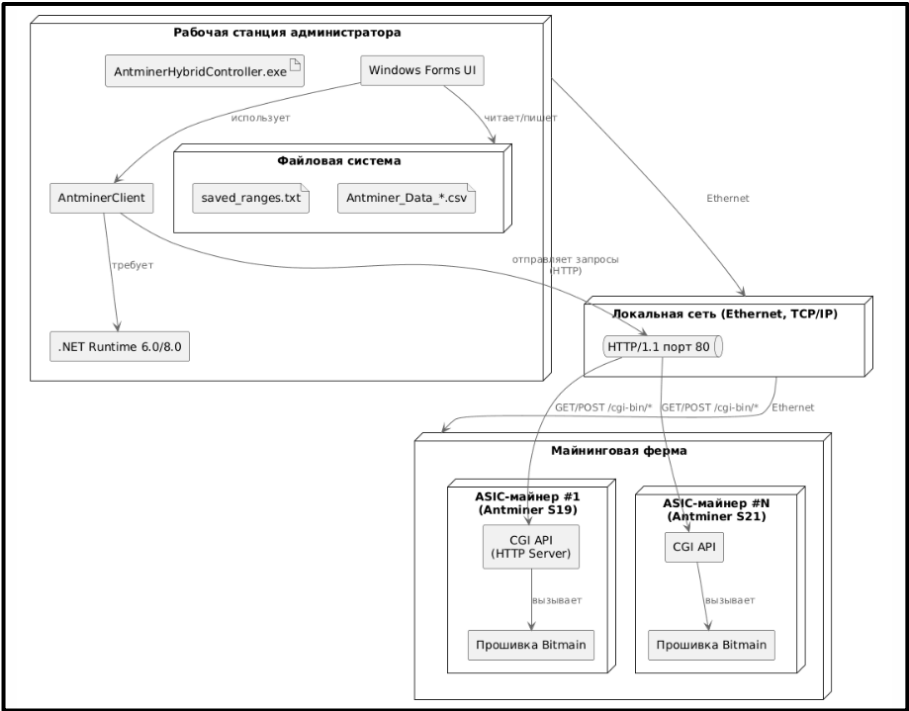


Рисунок 4 — Диаграмма разворачивания

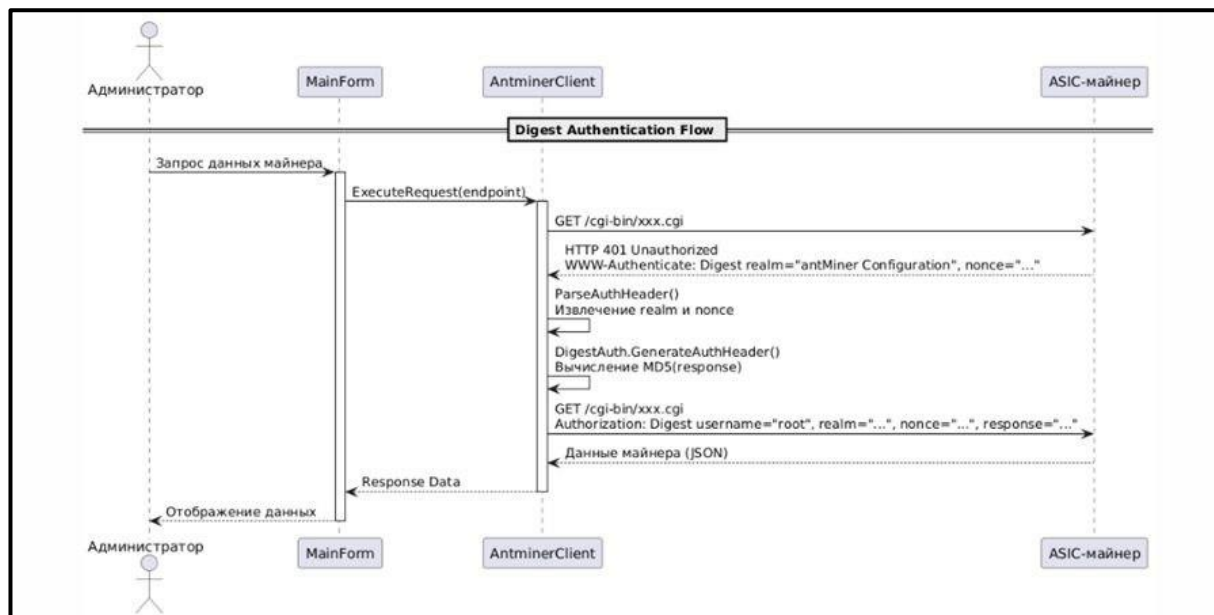


Рисунок 5 — Диаграмма последовательностей

Исходя из требуемого функционала, планируемой архитектуры проекта и выбранной целевой платформы, был определен следующий набор инструментов разработки: язык программирования C#, платформа .NET Framework / .NET Core, среда разработки Microsoft Visual Studio 2022, операционная система Microsoft Windows 10/11.

1.5 Выбор языка программирования

C# — это кроссплатформенный язык общего назначения, который делает разработчиков продуктивным при написании высокопроизводительного кода. C# имеет широкую поддержку в экосистеме и в различных нагрузках. Основанный на объектно-ориентированных принципах, он включает множество функций из других парадигм, включая функциональное программирование. Низкоуровневые функции поддерживают сценарии высокой эффективности без написания небезопасного кода. Большинство сред выполнения и библиотек написаны на C#, и достижения в C# часто используются для всех разработчиков .NET [1].

В большинстве программ C# используется dotnet build команда для компиляции группы исходных файлов в двоичный пакет. Затем вы используете dotnet run команду для запуска программы. (Этот процесс можно упростить, так как dotnet run компилирует программу перед его выполнением при необходимости.) Эти средства поддерживают широкий язык параметров конфигурации и коммутаторов командной строки. Интерфейс dotnet командной строки (CLI), включенный в пакет SDK для .NET, предоставляет множество средств для создания и изменения файлов C# [1].

Приложения C# получают преимущества от автоматического управления памятью среды выполнения. Приложения C# также используют обширные библиотеки среды выполнения, предоставляемые пакетом SDK для .NET. Некоторые компоненты являются независимыми от платформы, такими как библиотеки файловой системы, коллекции данных и математические библиотеки. Другие относятся к одной рабочей нагрузке, например к веб-библиотекам ASP.NET Core или библиотеке пользовательского интерфейса .NET MAUI. Расширенная экосистема с открытым исходным кодом в NuGet расширяет библиотеки, которые являются частью среды выполнения. Эти библиотеки предоставляют еще больше компонентов, которые можно использовать.

1.6 Выбор среды разработки

В качестве среды разработки, исходя из целевой платформы проекта, была взята Microsoft Visual Studio 2022. Для создания графического интерфейса используется технология Windows Forms.

Visual Studio — это мощная интегрированная среда разработки (IDE) для Windows, где можно разрабатывать, создавать, отлаживать, тестировать и развертывать приложения в одном месте [2].

Visual Studio 2022 предоставляет широкий набор инструментов для разработки:

- поддержка нескольких языков. Код в C#, C++, Python, JavaScript, TypeScript и многое другое;
- кроссплатформенная разработка. Создание мобильных и классических приложений с помощью .NET MAUI, веб-приложений с помощью ASP.NET или служб Node.js;
- разработка с помощью ИИ. Быстрое написание, отладка и тестирование кода с помощью GitHub Copilot;
- установщик, основанный на рабочей нагрузке. Установите только необходимые компоненты;
- интегрированная система управления версиями. Клонирование, ветвление, коммит и проверка пулл-реквестов с использованием встроенных средств Git [2];
- расширяемая платформа. Настройте интегрированную среду разработки с помощью расширений из Visual Studio Marketplace или создайте собственную среду разработки.

Пример фрагмента программы в среде Visual Studio 2022 представлен на рисунке 6:

```
// someWords is a string array.
string[] someWords = {
    "the",
    "quick",
    "brown",
    "fox",
    "jumps"
};

string[] moreWords = {
    "over",
    "the",
    "lazy",
    "dog"
};

// Alphabetically sort the words.
IEnumerable<string> query = from word in someWords
                           orderby word
                           select word;
```

Рисунок 6 — пример фрагмента алгоритма

Microsoft Visual Studio 2022 отличается удобством освоения, особенно для начинающих разработчиков, благодаря понятному интерфейсу и наличию подробной документации. Использование инструментов визуального проектирования позволяет быстро создавать прототипы приложений, а встроенные средства отладки значительно упрощают поиск и устранение ошибок [2].

Среда разработки наглядно демонстрирует взаимосвязь между элементами пользовательского интерфейса и программным кодом, который обрабатывает возникающие события. При грамотном разделении логики приложения и его представления процесс разработки становится более организованным и эффективным по сравнению с использованием обычных текстовых редакторов.

Кроме того, Visual Studio 2022 доступна для использования на бесплатной основе, что делает её доступным инструментом для широкого круга разработчиков. Сайт Visual Studio представлен на рисунке 7.

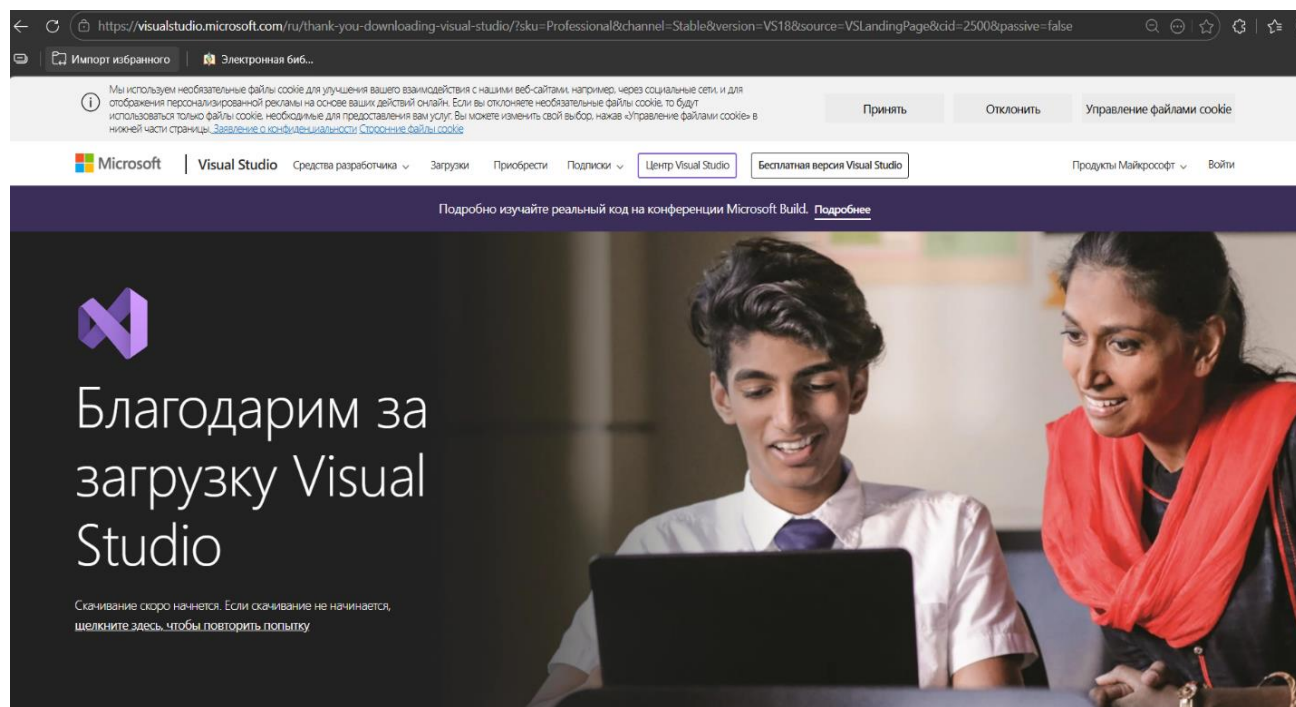


Рисунок 7 — Сайт Visual Studio

Страница загрузки Visual Studio включает:

- подробное описание функциональных возможностей среды;
- технические требования к системе;
- другие возможные версии;
- форму для обратной связи;
- кнопки загрузки установщика.

Для того чтобы скачать Visual Studio необходимо нажать на кнопку загрузки, что можно увидеть на рисунке 8.

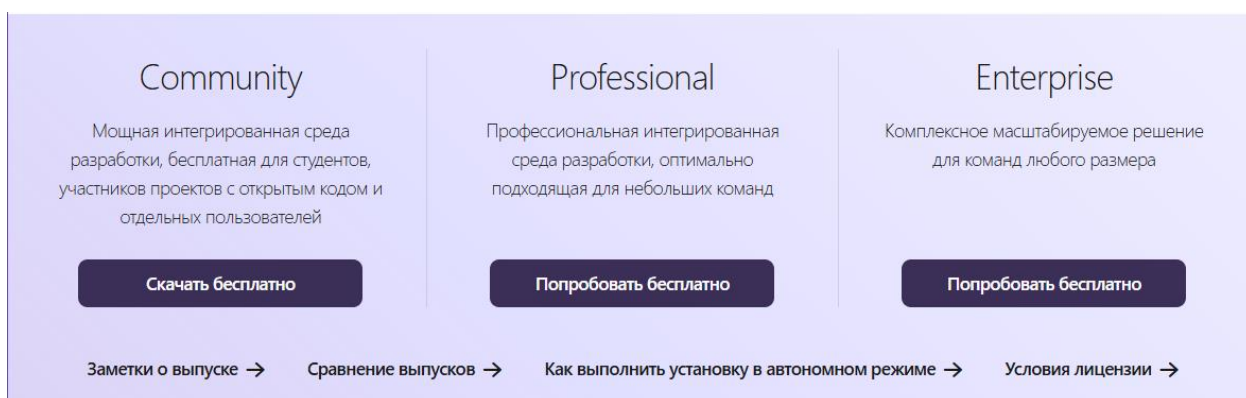


Рисунок 8 — Область загрузки среды программирования на сайте

После мы переходим на другую страницу для где происходит скачивание установщика Visual Studio, и после скачивания открываем (данное окно показано на рисунке 9).

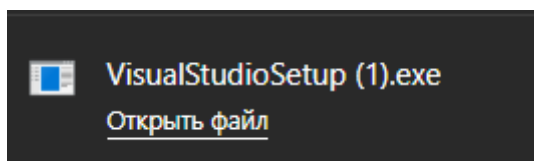


Рисунок 9 — Загрузка установочного файла среды программирования

После открытия файла запускается процесс загрузки установки среды для выбора версии разработки. Ход загрузки отображается в интерфейсе браузера и проиллюстрирован на рисунке 10.

Visual Studio Installer

Подготовка Visual Studio Installer.



Рисунок 10 — Окно с установщика

После установки открывается окно с выбором нужных компонентов для версии Visual Studio 2022. Запуск файла инициирует работу Мастера установки – для продолжения процесса нужно нажать кнопку «Установить» в открывшемся окне (рисунок 11). Далее потребуется ознакомиться с лицензионным соглашением и подтвердить его принятие.

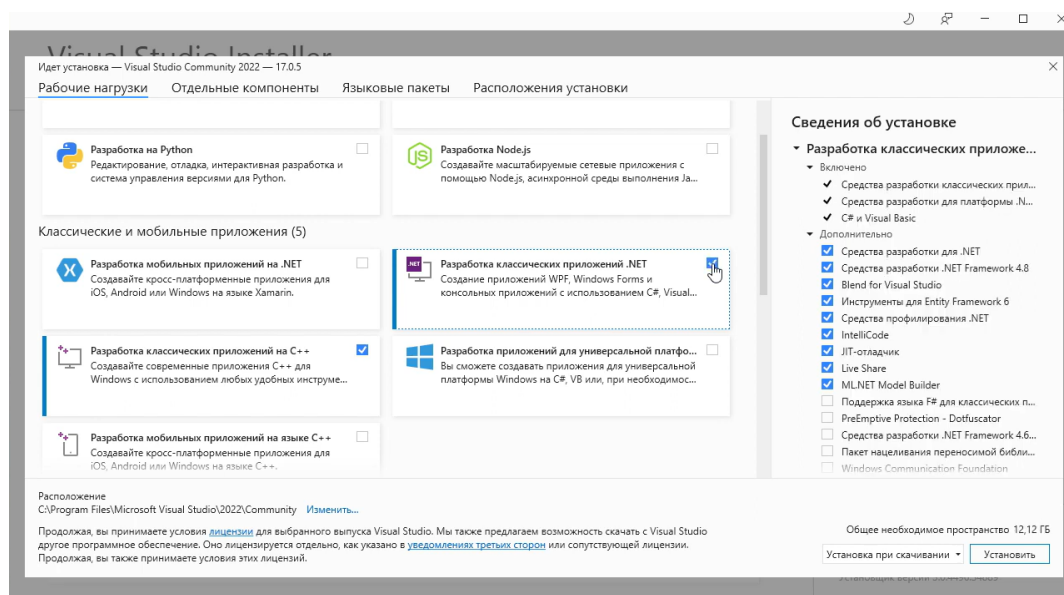


Рисунок 11 — Мастер установки Visual Studio

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

22

После успешной установки мы видим кнопку «Запустить», которая представлена на рисунке 12.

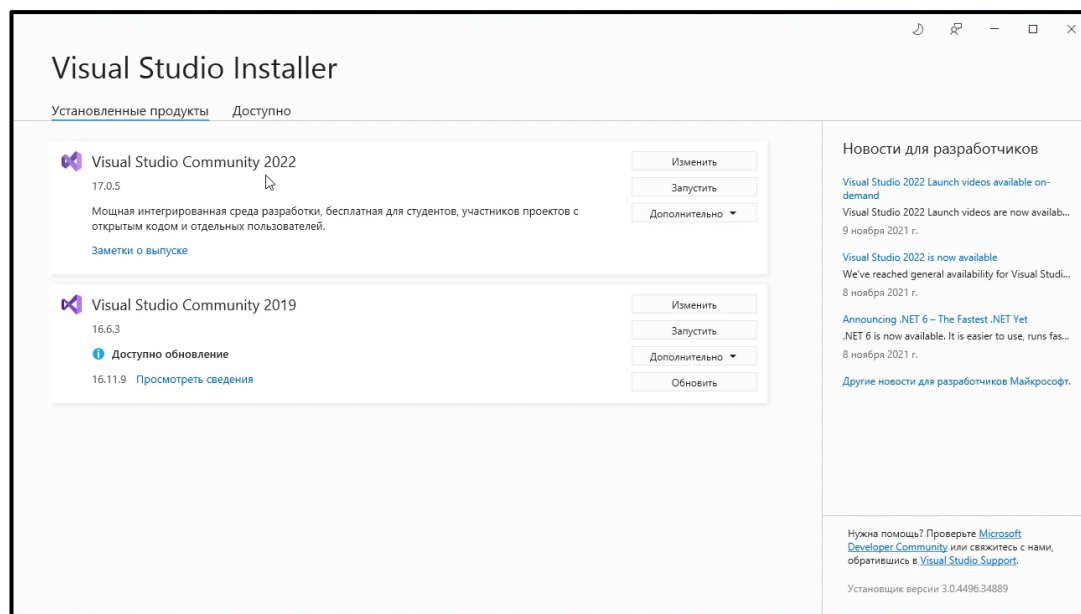


Рисунок 12 — Запуск среды программирования

Далее после запуска мы видим главное окно среды разработки (рисунок 13).

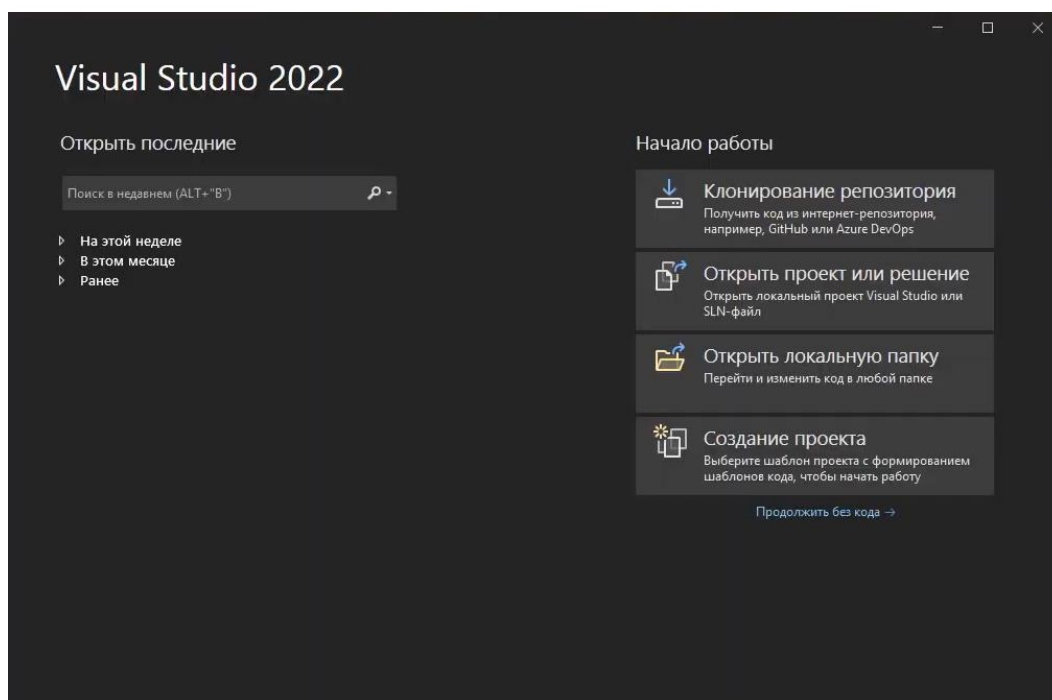


Рисунок 13 — Главное окно среды программирования

После нажимаем на кнопку создание проекта (рисунок 14).

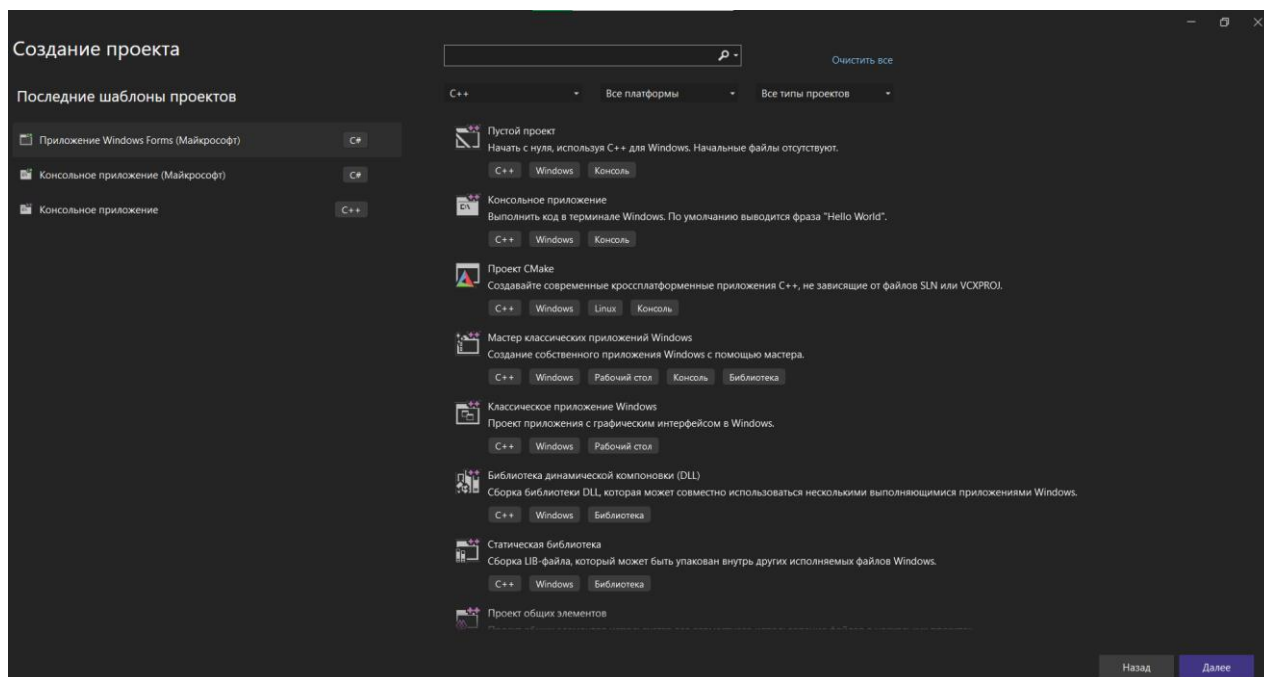


Рисунок 14 — Создание нового проекта

Затем выбираем «Приложение Windows Forms», представленное на рисунке 15.

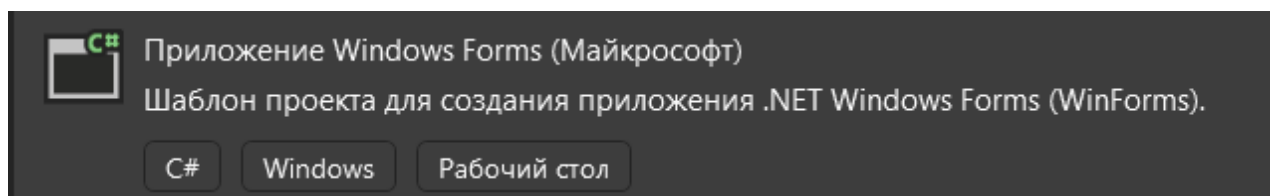


Рисунок 15 — Приложение Windows Forms

Для создания проекта необходимо выбрать название, имя расширения и папку, в которую будут сохраняться изменения по проекту, как представлено на рисунке 16.

Имя проекта

Diplom

Расположение

C:\Users\GIGI\source\repos

Имя решения ⓘ

Diplom

☐ Поместить решение и проект в одном каталоге

Проект будет создан в "C:\Users\GIGI\source\repos\Diplom\Diplom\"

Рисунок 16 — Начало создания нового проекта в среде Visual Studio 2022

Затем нажимаем на кнопку «Далее» и на рисунке 17 показан платформы, на которой будет основан проект.

Платформа ⓘ

.NET 8.0 (долгосрочная поддержка)

Рисунок 17 — Платформа проекта

Нажимаем на кнопку создать и перед пользователем появляется окно разработки приложения, как это показано на рисунке 18. Можно заметить панель управления проектом, главный файл проекта и дополнительные настройки среды разработки

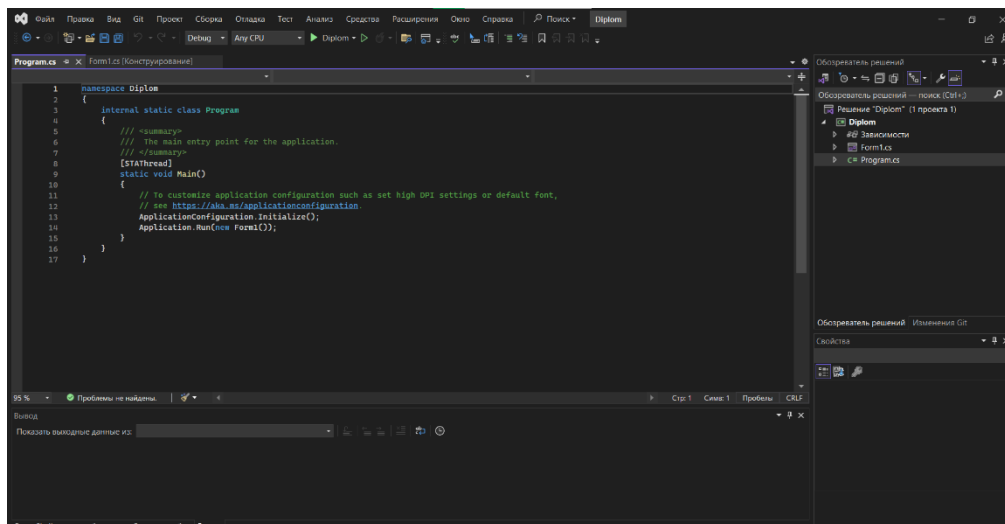


Рисунок 18 — Панель инструментов

Для дальнейшей успешной разработки программы потребуется установить дополнительные библиотеки в диспетчере пакетов Nuget. Для этого выбираем вкладку «Проект» и переходим во вкладку диспетчера пакетов Nuget (рисунок 19).

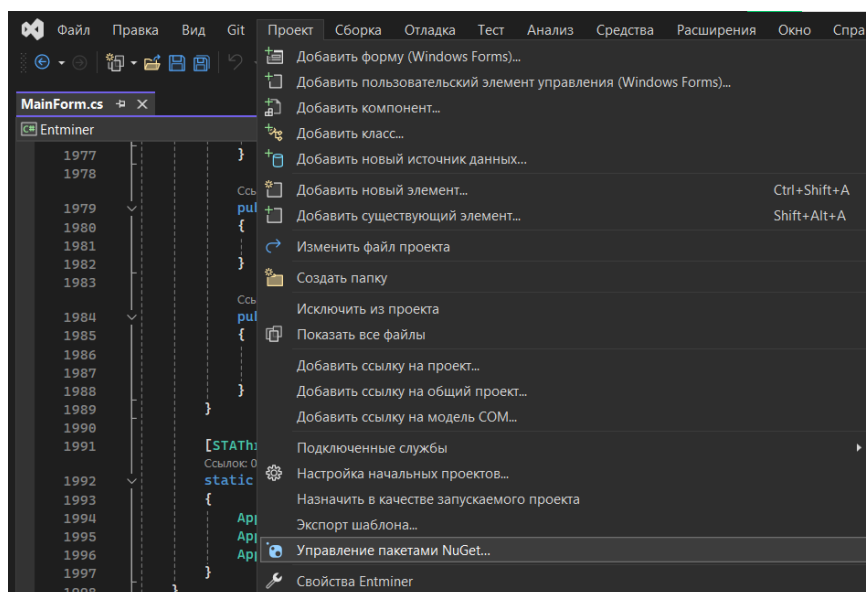


Рисунок 19 — Диспетчер пакетов Nuget

Далее выбираем необходимые для разработки сторонние библиотеки (рисунок 20). Функциональность данных библиотек позволяет комфортно и быстро ускорить процесс разработки приложения. Все параметры и настройки

будут применяться автоматически, что позволит заметно снизить количество строк кода и обеспечить сохранение правила ООП (рисунок 21).

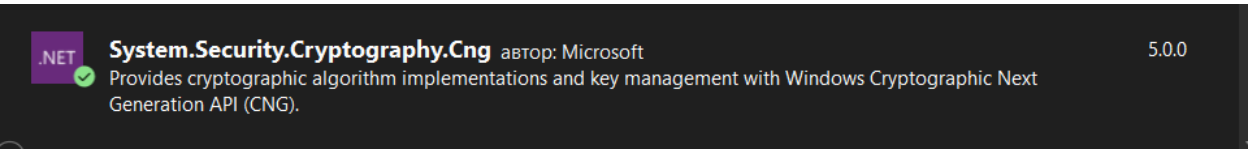


Рисунок 20 — Дополнительные библиотеки

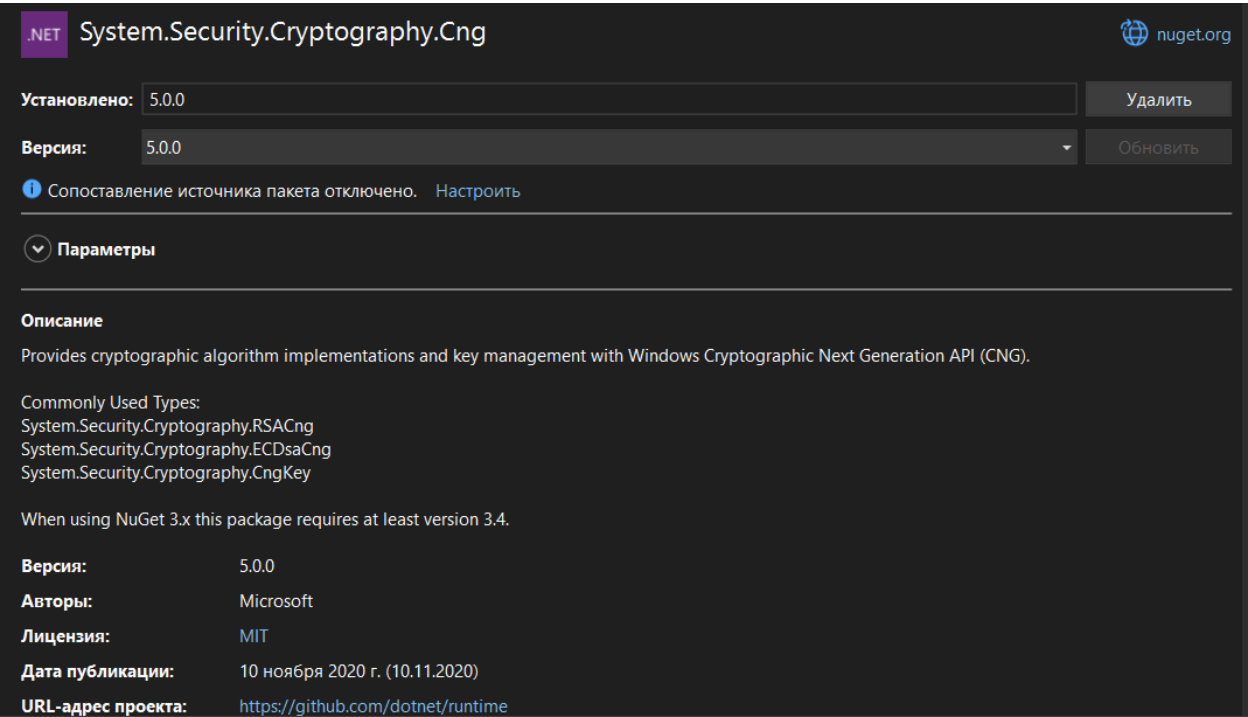


Рисунок 21 — Свойства рабочего холста проекта

1.7 Разработка управляющей программы

После завершения процесса настройки среды программирования был начат этап разработки управляющей программы. Исходя из цели проекта была построена диаграмма прецедентов пользователя, представленная на рисунке 22.



Рисунок 22 — Диаграмма прецедентов пользователя

Также для понимания алгоритма работы была разработана функциональная схема выполнения программы, показанная на рисунке 23.

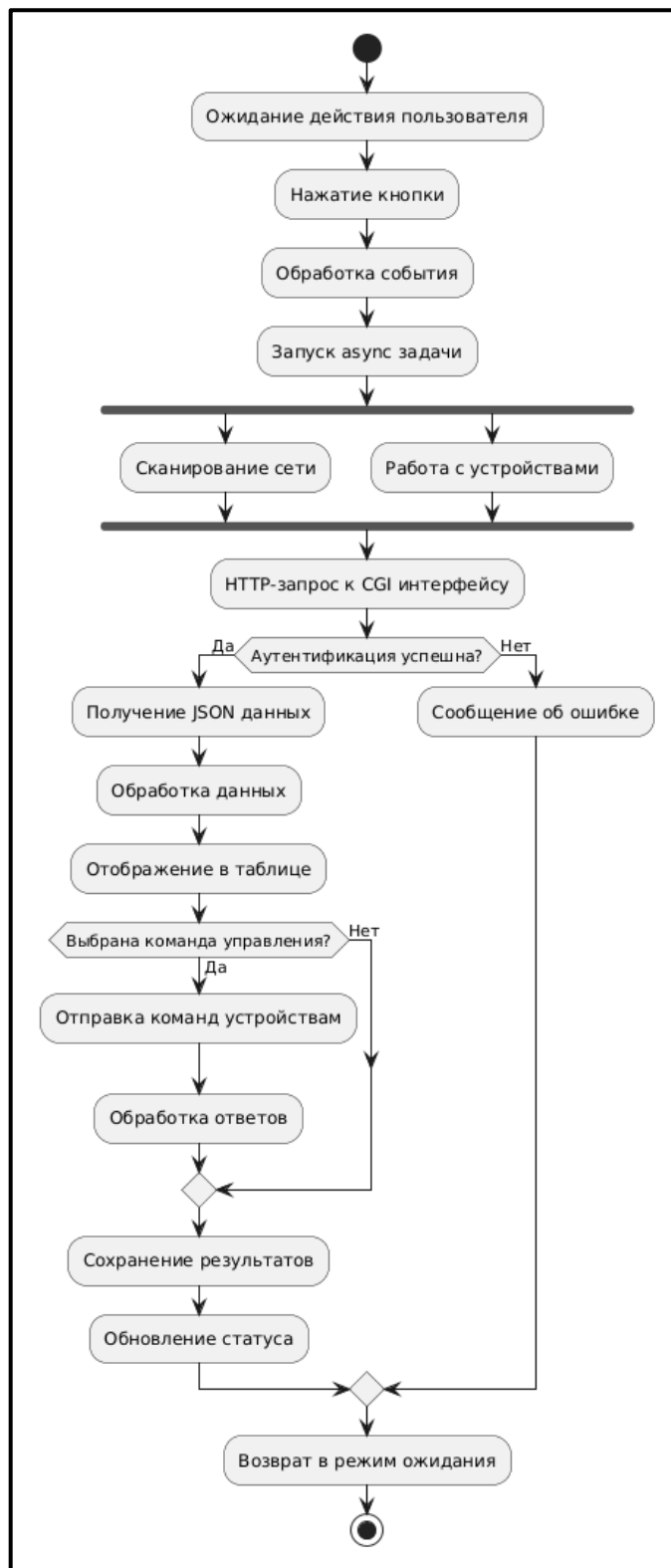


Рисунок 23 — Функциональная схема работы проекта

Перед тем как перейти к описанию разделов на рисунке 24 будет описана диаграмма классов реализуемой программы.

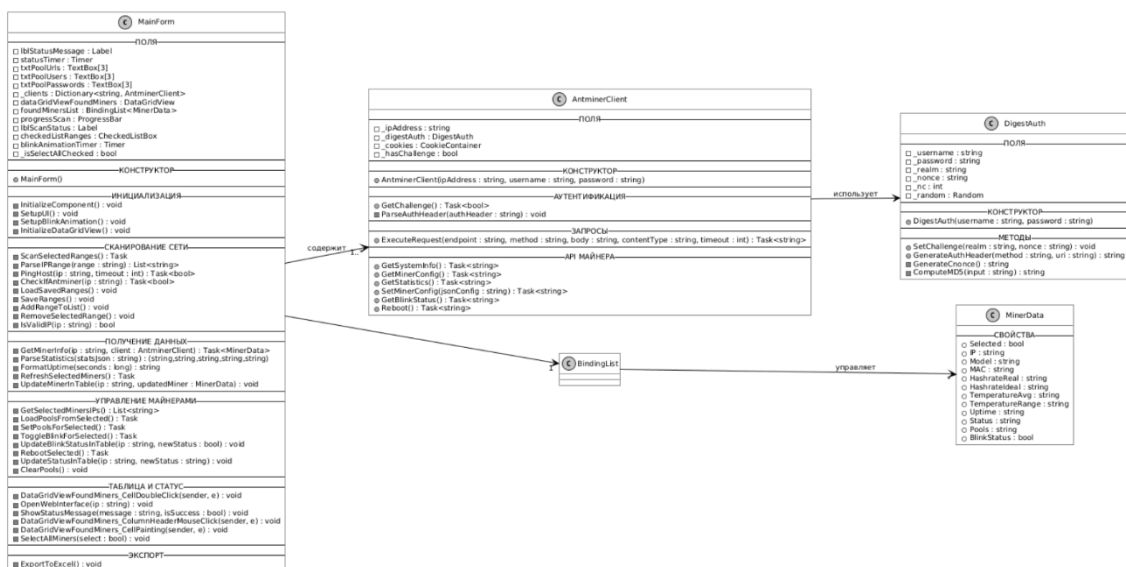


Рисунок 24 — Диаграмма классов

Далее перечислены разделы, которые отвечают за определённый функционал данного приложения. Описание главного класса представлено ниже.

Ниже описаны конструктор и методы инициализации:

- `InitializeComponent()` — метод, созданный конструктором Windows Forms. Устанавливает размер формы, заголовок «Antminer Hybrid Controller v2.0» и цвет фона;

- `SetupUI()` — метод, создающий весь пользовательский интерфейс: кнопки, панели, поля ввода, таблицу, таймеры и контейнеры `SplitContainer`;

- `InitializeDataGridView()` — метод, настраивающий таблицу `dataGridViewFoundMiners`. Создаёт колонки, привязывает `BindingList<MinerData>`, подписывает обработчики событий `CellPainting` и `ColumnHeaderMouseClick`;

- `SetupBlinkAnimation()` — метод, запускающий таймер `blinkAnimationTimer` с интервалом 500 мс для анимации мигания в колонке `Blink`.

Затем указаны функции сканирования сети:

- `ScanSelectedRanges()` — асинхронный метод. Запускает сканирование

выбранных диапазонов, параллельно проверяет IP до 30 потоков, отображает прогресс;

- ParseIPRange(string range) — метод. Расположен сразу после ScanSelectedRanges(). Преобразует строку диапазона в список IP-адресов;

- PingHost(string ip, int timeout) — асинхронный метод. Расположен после ParseIPRange(). Отправляет ping-запрос на указанный IP;

- CheckIfAntminer(string ip) — асинхронный метод. Расположен после PingHost(). Проверяет, является ли устройство майнером Antminer по ответу 401 Unauthorized;

- LoadSavedRanges() — метод. Расположен в блоке управления диапазонами. Загружает сохранённые диапазоны из файла saved_ranges.txt;

- SaveRanges() — метод. Расположен после LoadSavedRanges(). Сохраняет текущий список диапазонов в файл;

- AddRangeToList() — метод. Расположен после SaveRanges(). Добавляет новый диапазон из полей ввода в список;

- RemoveSelectedRange() — метод. Расположен после AddRangeToList(). Удаляет выбранный диапазон из списка;

- IsValidIP(string ip) — метод. Расположен после RemoveSelectedRange(). Проверяет корректность формата IP-адреса.

Описание функций получения данных с майнинг-устройства видны ниже в работе:

- GetMinerInfo(string ip, AntminerClient client) — асинхронный метод. Расположен после CheckIfAntminer(). Собирает полную информацию о майнере: модель, MAC, хешрейт, температура, пулы, Blink;

- ParseStatistics(string statsJson) — метод. Расположен после GetMinerInfo(). Парсит JSON статистики, извлекает хешрейт, температуры и время работы;

- FormatUptime(long seconds) — метод. Расположен после ParseStatistics(). Преобразует секунды в читаемый формат;

– RefreshSelectedMiners() — асинхронный метод. Расположен в блоке управления таблицей (после SelectAllMiners()). Обновляет данные для выбранных майнеров;

– UpdateMinerInTable(string ip, MinerData updatedMiner) — метод. Расположен после RefreshSelectedMiners(). Обновляет строку конкретного майнера в таблице.

Функции управления майнерами указаны ниже:

– ToggleBlinkForSelected() — асинхронный метод. Переключает состояние Blink для выбранных майнеров;

– UpdateBlinkStatusInTable(string ip, bool newStatus) — метод. Расположен после ToggleBlinkForSelected(). Обновляет отображение Blink в таблице;

– RebootSelected() — асинхронный метод. Расположен после UpdateBlinkStatusInTable(). Перезагружает выбранные майнеры;

– UpdateStatusInTable(string ip, string newStatus) — метод. Расположен после RebootSelected(). Обновляет текстовый статус майнера в таблице;

– SetPoolsForSelected() — асинхронный метод. Расположен после LoadPoolsFromSelected(). Устанавливает новые пулы для выбранных майнеров;

– LoadPoolsFromSelected() — асинхронный метод. Расположен в блоке методов управления (перед SetPoolsForSelected()). Загружает пулы с первого выбранного майнера в поля ввода;

– ClearPools() — метод. Расположен после SetPoolsForSelected(). Очищает все поля ввода пулов.

Функции работы с таблицей и выделением данных представлены далее в тексте:

– GetSelectedMinersIPs() — метод. Возвращает список IP выбранных майнеров;

– SelectAllMiners(bool select) — метод. Расположен после

`DataGridViewFoundMiners_CellPainting()`. Устанавливает или снимает выделение всех майнеров;

- `DataGridViewFoundMiners_CellDoubleClick()` — обработчик события.

Открывает веб-интерфейс майнера при двойном клике;

- `OpenWebInterface(string ip)` — метод. Расположен после обработчика двойного клика. Запускает браузер с веб-интерфейсом майнера;

- `DataGridViewFoundMiners_ColumnHeaderMouseClick()` — обработчик события. Расположен в `InitializeDataGridView()` (подписан) и реализован после `InitializeDataGridView()`. Обрабатывает клик по заголовку первой колонки;

- `DataGridViewFoundMiners_CellPainting()` — обработчик события. Расположен после `DataGridViewFoundMiners_ColumnHeaderMouseClick()`.

Отрисовывает чекбокс в заголовке и анимированный квадрат `Blink`.

Объединение функций экспорта данных, функций отображения статуса, модель данных:

- `ExportToExcel()` — метод. Сохраняет данные всех майнеров в CSV-файл;

- `ShowStatusMessage(string message, bool isSuccess)` — метод. Расположен после `OpenWebInterface()`. Показывает временное сообщение в строке состояния;

- `statusTimer` — поле класса. Инициализируется в `SetupUI()`. Автоматически очищает строку состояния через 3 секунды.

Затем идет описание класса `MinerData`:

- `Selected` — свойство. Хранит флаг выбора майнера в таблице;

- `IP` — свойство. Хранит IP-адрес майнера;

- `Model` — свойство. Хранит модель майнера;

- `MAC` — свойство. Хранит MAC-адрес майнера;

- `HashrateReal` — свойство. Хранит реальный хешрейт майнера;

- `HashrateIdeal` — свойство. Хранит идеальный хешрейт майнера;

- `TemperatureAvg` — свойство. Хранит среднюю температуру чипов;

- TemperatureRange — свойство. Хранит диапазон температур (мин-макс);
- Uptime — свойство. Хранит время непрерывной работы;
- Status — свойство. Хранит текстовый статус (✓);
- Pools — свойство. Хранит строку с адресами пулов;
- BlinkStatus — свойство. Хранит состояние Blink-индикации (true/false).

Также описание класса DigestAuth можно увидеть ниже:

- DigestAuth(string username, string password) — конструктор. Инициализирует объект для Digest-аутентификации;
- SetChallenge(string realm, string nonce) — метод. Сохраняет полученные от майнера realm и nonce;
- GenerateAuthHeader(string method, string uri) — метод. Генерирует заголовок авторизации для HTTP-запроса;
- GenerateCnonce() — метод. Создаёт случайную строку для защиты от повтора атак;
- ComputeMD5(string input) — метод. Вычисляет MD5-хеш строки.

И последний класс, который будет описан – AntminerClient. А также расписаны функции HTTP- Клиента:

- AntminerClient(string ipAddress, string username, string password) — конструктор. Создаёт клиент для взаимодействия с конкретным майнером;
- GetChallenge() — асинхронный метод. Получает от майнера challenge (realm и nonce);
- ParseAuthHeader(string authHeader) — метод. Разбирает заголовок WWW-Authenticate, извлекает realm и nonce;
- ExecuteRequest() — асинхронный метод. Универсальный метод отправки HTTP-запросов с авторизацией;
- GetSystemInfo() — асинхронный метод. Запрашивает системную информацию майнера;

- GetMinerConfig() — асинхронный метод. Запрашивает текущую конфигурацию майнера;
- GetStatistics() — асинхронный метод. Запрашивает статистику работы майнера;
- SetMinerConfig(string jsonConfig) — асинхронный метод. Отправляет новую конфигурацию на майнер;
- GetBlinkStatus() — асинхронный метод. Запрашивает состояние Blink-индикации;
- Reboot() — асинхронный метод. Отправляет команду перезагрузки майнера.

1.8 Графическая часть разрабатываемой программы

Для общего понимания функциональности разработанной программы, рассмотрим главный экран приложения на рисунке 25.

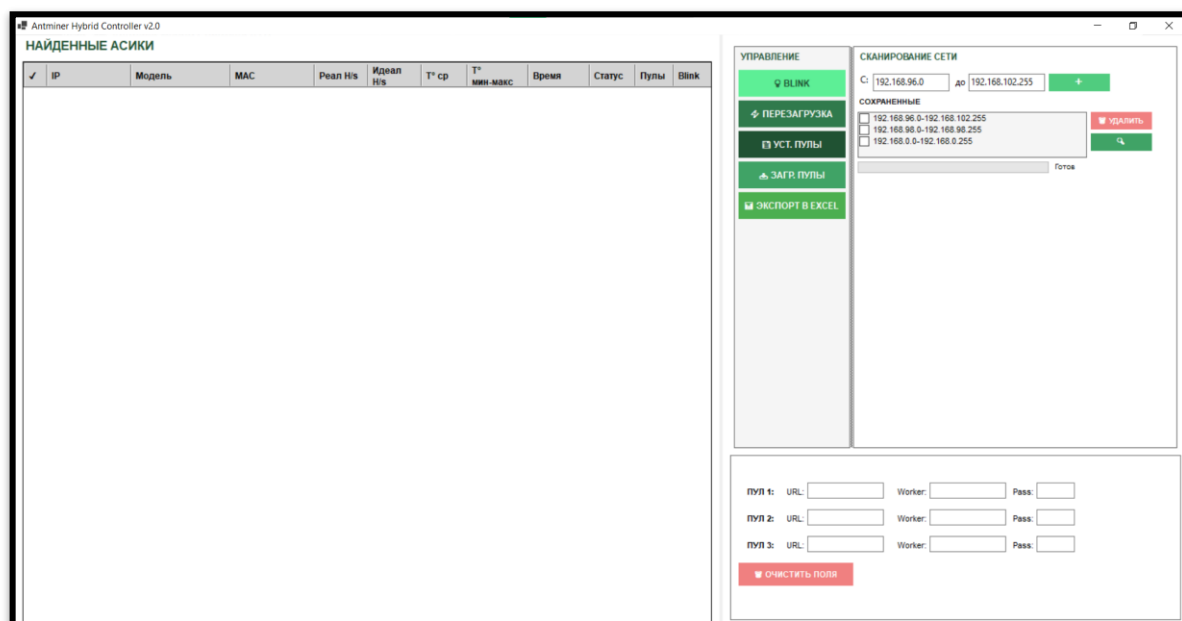


Рисунок 25 — Главный экран программы

Сразу после запуска пользователю доступны все возможности приложения, для удобства работы экран разделен на несколько вкладок

таблица «Найденные асики», «Управление», «Сканирование сети», а также нижняя вкладка с изменением полей для пулов устройства.

Для взаимодействия с устройствами пользователю необходимо выбрать диапазон или нужную подсеть IP-адресов и нажать на кнопку «Сканировать», как представлено на рисунке 26.

Рисунок 26 — Этап сканирования сети

После чего происходит процесс сканирования при помощи функции `ScanSelectedRanges`, который выполняется при нажатии пользователем кнопки «СКАНИРОВАТЬ». Алгоритм начинается с получения списка выбранных пользователем IP-диапазонов из элемента `checkedListRanges`, после чего каждый диапазон преобразуется в полный перечень IP-адресов с помощью вспомогательной функции `ParseIPRange`. Далее формируется пул параллельных задач, ограниченный 30 одновременными потоками для предотвращения перегрузки сети. Для каждого IP-адреса асинхронно выполняется проверка: сначала отправляется ICMP-запрос методом `PingHostc` таймаутом 300 мс, затем при успешном ответе вызывается метод `CheckIfAntminer`, который отправляет HEAD-запрос к эндпоинту `/cgi-bin/get_system_info.cgi` и проверяет получение статуса 401 Unauthorized (характерный признак ASIC-майнера Bitmain). При успешной идентификации создаётся экземпляр класса `AntminerClient`, сохраняется в словарь `_clients` и запрашивается полная информация через метод `GetMinerInfo`. Все

обнаруженные устройства добавляются в BindingList, что автоматически обновляет интерфейсную таблицу. Ход сканирования визуализируется с помощью прогресс-бара и текстовой метки, отображающей количество обработанных адресов и найденных устройств. Работа программы представлена на рисунке 27.

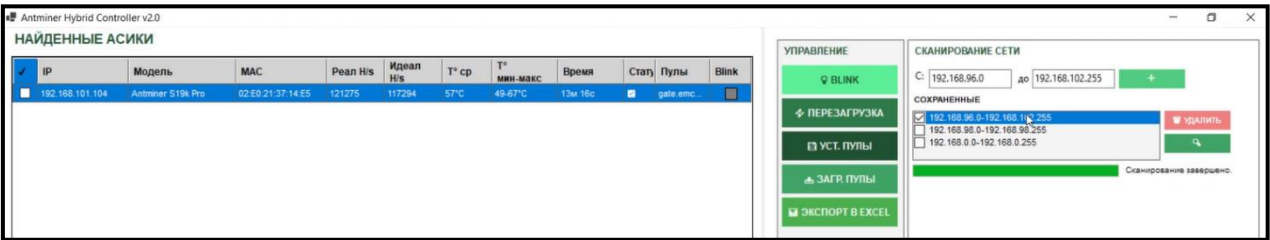


Рисунок 27 — Сканирование сети

Затем пользователь в зависимости от выбранной установки, какое именно действие требуется произвести с устройством выбирает нужный сценарий.

1.8.1 Сценарий «Поиска устройства»

Для того чтобы выполнить физический поиск необходимых устройств в разработанном проекте есть функция «Blink», которая позволяет включать светодиодную индикацию на самом Asic-устройстве. Процесс управления Blink-индикацией реализован асинхронным методом ToggleBlinkForSelected, который вызывается при нажатии пользователем кнопки «BLINK» после выбора одного или нескольких майнеров в таблице. Алгоритм начинается с получения списка IP-адресов выбранных устройств через вспомогательный метод GetSelectedMinersIPs, после чего последовательно с задержкой 100 мс между операциями для каждого майнера выполняется переключение состояния светодиодного индикатора. Для каждого IP-адреса сначала отправляется GET-запрос к эндпоинту /cgi-bin/get_blink_status.cgi через метод GetBlinkStatus клиента AntminerClient, полученный JSON-ответ парсится с помощью JsonDocument для извлечения текущего значения поля blink. Затем формируется тело POST-запроса: если текущее состояние равно true,

отправляется обратное значение false, иначе — true. Запрос направляется на эндпоинт /cgi-bin/blink.cgi с типом содержимого text/plain. После успешного выполнения запроса вызывается метод UpdateBlinkStatusInTable, который обновляет соответствующую строку в BindingList. Визуальное отображение состояния Blink реализовано через отдельную колонку таблицы, где при активном мигании отображается анимированный квадрат, меняющий цвет с красного на зелёный с периодом 500 мс управляется таймером blinkAnimationTimer. Процесс работы продемонстрирован на рисунке 28.

✓	IP	Модель	MAC	Реал H/s	Идеал H/s	T° ср	T° МИН-МАКС	Время	Статус	Пулы	Blink
☒	192.168.101.43	Antminer S19k Pro	02:E6:84:1E:FF:B7	119296	114971	54°C	47-65°C	6м 30с	☒		

Рисунок 28 — «Blink» устройства

1.8.2 Сценарий «Перезагрузка устройства»

В случае если майнинг на устройстве остановился или были технические неполадки с устройством в разработанной программе присутствует функция «Перезагрузка». Процесс перезагрузки ASIC-майнеров реализован асинхронным методом RebootSelected, который вызывается при нажатии пользователем кнопки «ПЕРЕЗАГРУЗКА» после выбора одного или нескольких устройств в таблице. Алгоритм начинается с получения списка IP-адресов выбранных майнеров через метод GetSelectedMinersIPs, после чего отображается диалоговое окно подтверждения с предупреждением, так как перезагрузка временно останавливает процесс майнинга на несколько минут. При положительном ответе пользователя программа последовательно (с задержкой 100 мс между операциями для предотвращения одновременной перезагрузки большого количества устройств) для каждого выбранного IP-адреса вызывает метод Reboot клиента AntminerClient. Данный метод формирует POST-запрос с телом {"action": "reboot"} и отправляет его на

эндпоинт /cgi-bin/reboot.cgi с типом содержимого text/plain. После успешной отправки команды (без ожидания фактического завершения перезагрузки, так как майнер теряет связь) вызывается метод UpdateStatusInTable, который информирует пользователя о том, что команда отправлена. Особенностью реализации является то, что метод не дожидается ответа от майнера после перезагрузки, так как устройство разрывает соединение.

Фрагмент работы ASICa (рисунок 29), отправленную команду (рисунок 30) и устройство после перезагрузки (рисунок 31) можно увидеть на рисунках ниже.

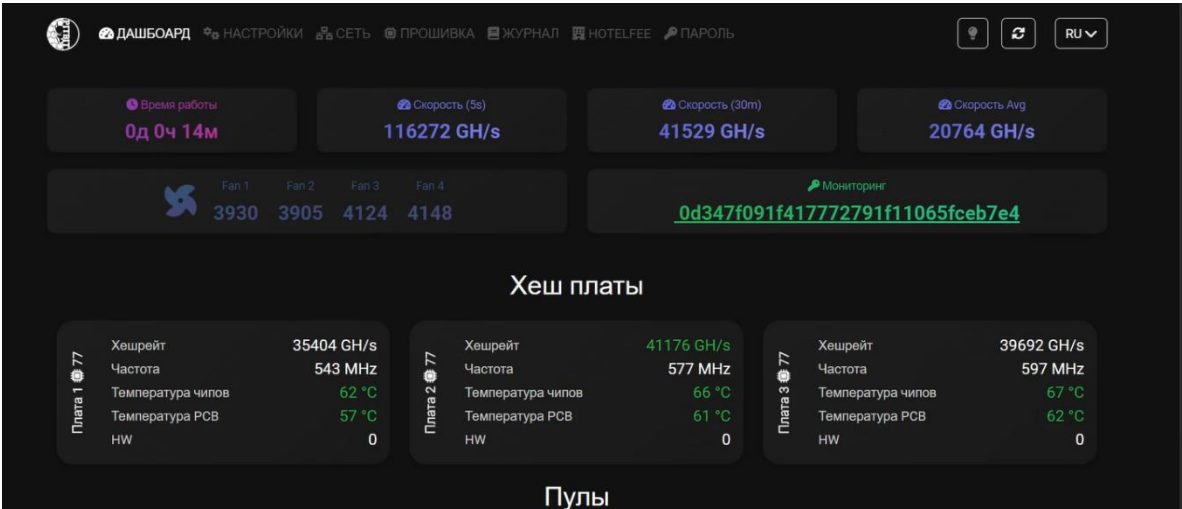


Рисунок 29 — Работа ASICa

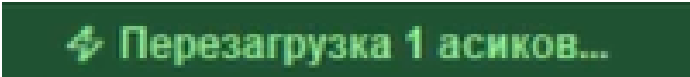


Рисунок 30 — Команда «Перезагрузка»

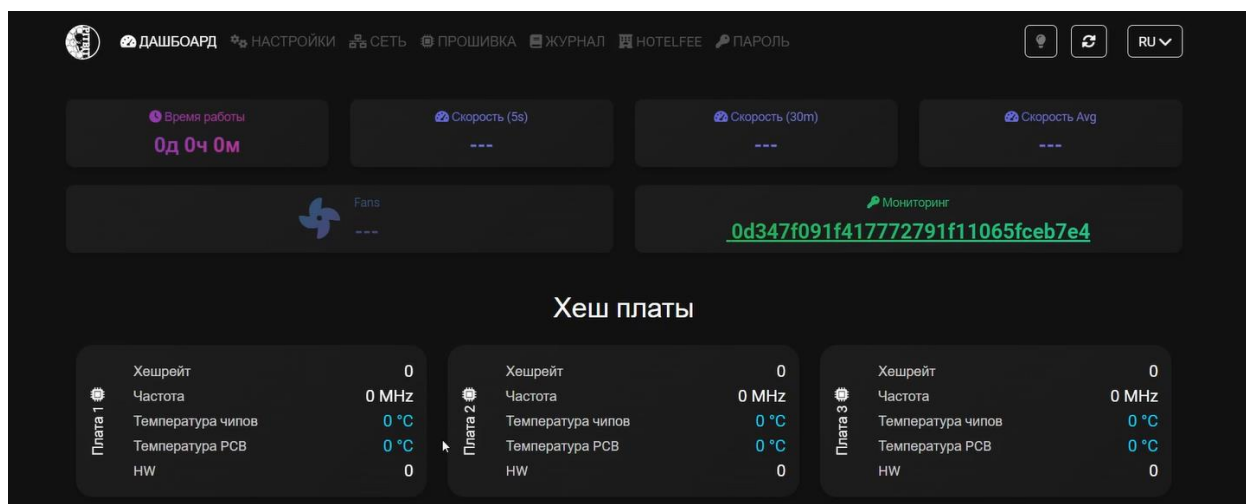


Рисунок 31 — Устройство после перезагрузки

1.8.3 Сценарий «Установка пулов»

Для того чтобы рабочий аппарат начал приносить прибыль нужно настроить пулы для майнинга. В нижнем правом углу на главном экране можно установить 3 необходимых пула для работы устройства. Процесс установки пулов в разработанной программе реализован асинхронным методом `SetPoolsForSelected()`, который вызывается при нажатии пользователем кнопки "УСТ. ПУЛЫ" после заполнения полей URL, Worker и Password для трёх пулов. Алгоритм начинается с получения списка IP-адресов выбранных майнеров через метод `GetSelectedMinersIPs()`, после чего из текстовых полей формы формируется коллекция `newPools` типа `List<Dictionary<string, string>>`, содержащая только непустые пулы. При отсутствии хотя бы одного заполненного пула пользователю выводится сообщение об ошибке. Для каждого выбранного майнера последовательно (с задержкой 500 мс между операциями для стабильности работы API) выполняется: получение текущей конфигурации майнера через метод `GetMinerConfig()`, парсинг полученного JSON с сохранением всех существующих параметров (частота, режим работы и т.д.), замена массива `pools` на новый сформированный список, сериализация обновлённой конфигурации обратно в JSON и отправка POST-запроса на эндпоинт `/cgi-`

bin/set_miner_conf.cgi через метод SetMinerConfig(). Особенностью реализации является обработка таймаутов: если сервер майнера не отвечает в течение 20 секунд (что характерно для некоторых моделей Antminer), исключение перехватывается, и операция считается успешной, так как конфигурация всё равно применяется. После обработки всех устройств пользователю выводится отчёт с количеством успешно настроенных майнеров.

Процесс конфигурирования ASIC-устройства продемонстрирован на рисунках 32-34.

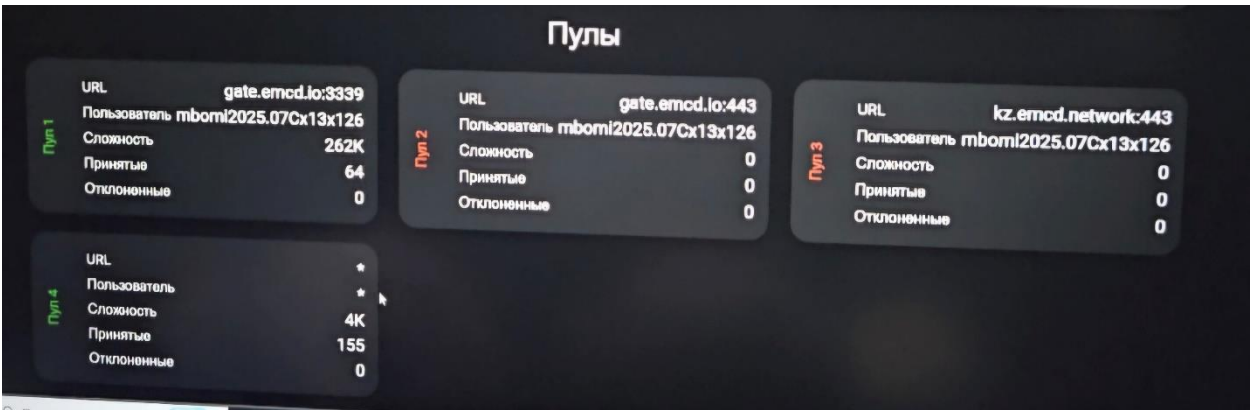


Рисунок 32 — До смены пулов на устройстве

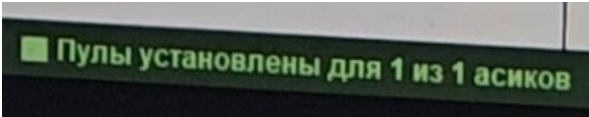


Рисунок 33 — Применение команды

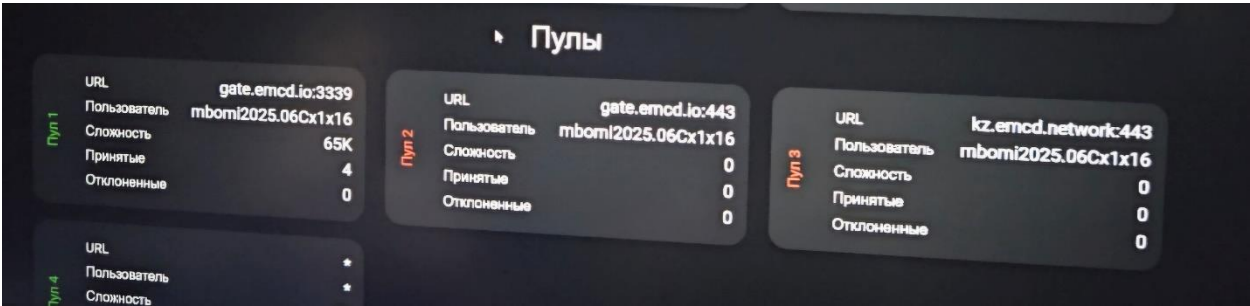


Рисунок 34 — Устройство после смены пулов

1.8.4 Сценарий «Загрузка пулов»

Во время конфигурации майнинг-стойки появляется необходимость создания одинаковых наименований устройств для удобства работы с отдельными майнинг-стойками. В случае установки новых устройств на уже сконфигурированную стойку есть реализованная кнопка «Загрузка пулов» для указания принадлежности к конкретному месту. Процесс загрузки пулов в разработанной программе реализован асинхронным методом LoadPoolsFromSelected, который вызывается при нажатии пользователем кнопки "ЗАГРУЗИТЬ ПУЛЫ" после выбора одного или нескольких майнеров в таблице. Алгоритм начинается с получения списка IP-адресов выбранных устройств через метод GetSelectedMinersIPs, после чего проверяется, что, хотя бы один майнер выбран. Для загрузки конфигурации используется только первый майнер из списка выбранных, так как предполагается, что все майнеры в ферме имеют одинаковые настройки пулов.

Метод отправляет GET-запрос к эндпоинту /cgi-bin/get_miner_conf.cgi через вызов _clients[firstIP].GetMinerConfig, после чего полученный JSON-ответ парсится с помощью JsonDocument. Из корневого элемента извлекается массив pools, и для каждого из первых трёх пулов или меньшего количества, считываются значения полей url, user и pass, которые затем подставляются в соответствующие текстовые поля интерфейса: txtPoolUrls, txtPoolUsers и txtPoolPasswords. Поля паролей настроены на отображение символов-заменителей через свойство UseSystemPasswordChar = true, что обеспечивает безопасность при визуальном отображении. В случае успешной загрузки пользователю выводится сообщение с IP-адресом майнера-источника, при возникновении ошибки соответствующее сообщение об ошибке. Работа данной функции продемонстрирована на рисунках 35, 36.

ПУЛ 1: URL: Worker: Pass:

ПУЛ 2: URL: Worker: Pass:

ПУЛ 3: URL: Worker: Pass:

ОЧИСТИТЬ ПОЛЯ

Рисунок 35 — Поля изменения пулов пустые

ПУЛ 1: URL: Worker: Pass:

ПУЛ 2: URL: Worker: Pass:

ПУЛ 3: URL: Worker: Pass:

ОЧИСТИТЬ ПОЛЯ

Рисунок 36 — Поля загружены

1.8.5 Сценарий «Экспорт в Excel»

Для необходимости сохранить данные об текущем состоянии майнинг-центра на главном экране присутствует кнопка «Экспорт в Excel». Процесс экспорта данных в разработанной программе реализован методом `ExportToExcel()`, который вызывается при нажатии пользователем кнопки "ЭКСПОРТ В EXCEL". Алгоритм начинается с проверки наличия данных в коллекции `foundMinersList`: если список пуст, пользователю выводится сообщение об ошибке. При наличии данных открывается диалоговое окно сохранения файла `SaveFileDialog` с фильтром, предлагающим форматы CSV (значения, разделённые точкой с запятой) и XLSX, причём имя файла автоматически формируется с текущей датой и временем в формате

Antminer_Data_ГГГГММДД_ЧЧММСС.csv. После выбора пользователем пути сохранения создаётся объект StringBuilder, в который сначала записывается строка-заголовок с названиями колонок: "IP; Модель; MAC; Хешрейт реал; Хешрейт идеал; Температура средняя; Температура мин-макс; Время работы; Статус; Пулы; Blink ". Затем в цикле foreach по всем объектам MinerData из foundMinersList формируются строки данных, где каждое свойство объекта преобразуется в строку, а булево поле BlinkStatus конвертируется в русские значения "Да"/"Нет". Все строки записываются в файл с кодировкой UTF-8 через File.WriteAllText(). После успешного экспорта пользователю выводится сообщение с именем сохранённого файла и отображается диалоговое окно с вопросом об открытии файла; при положительном ответе файл открывается в программе, ассоциированной с расширением CSV (обычно Microsoft Excel). Результат работы можно увидеть на рисунке 37.



Рисунок 37 — Сохраненный CSV-файл

1.8.6 Дополнительные полезные функции программы

Функция «Web-интерфейс» нужна, в случае если есть необходимость зайти в Web-интерфейс самого устройства, двойным нажатием по строке в таблице найденных ASICов и перед пользователем открывается браузер с открытой страницей характеристик аппарата. Данная функция позволяет оператору быстро перейти к штатному веб-интерфейсу майнера для детального просмотра логов, изменения расширенных настроек или диагностики. Функция открытия веб-интерфейса майнера реализована через обработчика события CellDoubleClick элемента dataGridViewFoundMiners. При двойном клике пользователя по любой ячейке строки таблицы метод DataGridViewFoundMiners_CellDoubleClick извлекает IP-адрес из колонки "IP"

соответствующей строки. Далее вызывается метод `OpenWebInterface`, который формирует URL вида `http://{ip}` и запускает системный браузер по умолчанию с помощью `Process.Start`, используя `UseShellExecute = true` для корректного открытия ссылки в ассоциированном приложении. После открытия страницы в строке состояния отображается сообщение об успешном действии. В случае ошибки выводится сообщение с описанием проблемы. Реализация продемонстрирована на рисунке 38 и 39.

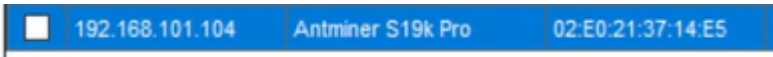


Рисунок 38 — Найденный майнер в программе

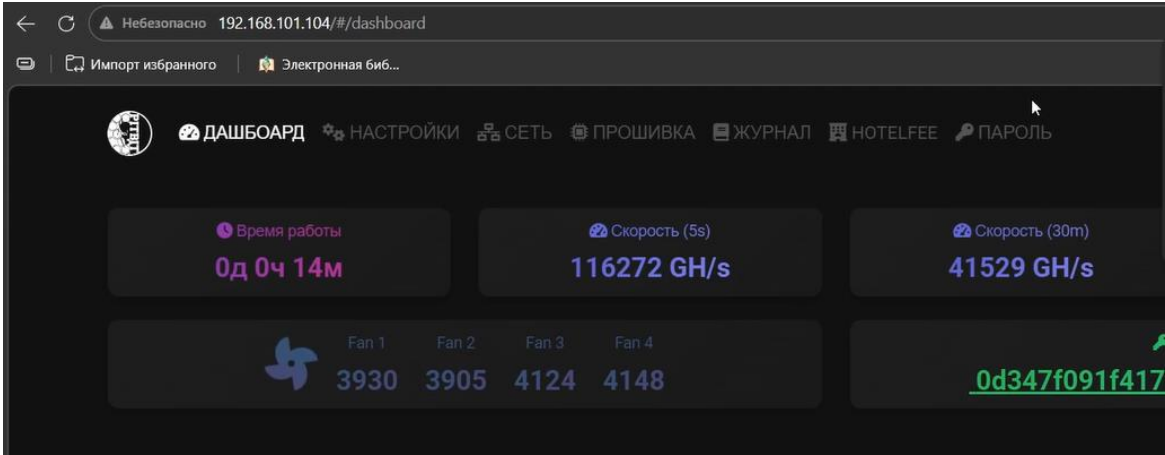


Рисунок 39 — После двойного нажатия в программе

Функция «Сохранение IP-диапазонов» обеспечивает сохранение пользовательских настроек между сеансами работы программы, исключая необходимость повторного ввода IP-диапазонов при каждом запуске. При запуске программы автоматически вызывается метод `LoadSavedRanges`, который проверяет наличие файла `saved_ranges.txt` в директории исполняемого файла. Если файл существует, все строки из него загружаются в список доступных диапазонов `checkedListRanges`. При отсутствии файла или его пустоте в список загружаются диапазоны по умолчанию: 192.168.96.0-192.168.102.255, 192.168.99.0-192.168.99.255, 192.168.100.0-192.168.100.255, 192.168.101.0-192.168.101.255. Каждый раз при добавлении нового диапазона

через кнопку «ДОБАВИТЬ» или удалении существующего через кнопку «УДАЛИТЬ» вызывается метод SaveRanges, который перезаписывает файл актуальным списком. Демонстрацию функции можно увидеть на рисунках 40–42.

Диапазон 192.168.100.0-192.168.101.255 добавлен

Рисунок 40 — Добавление нового диапазона

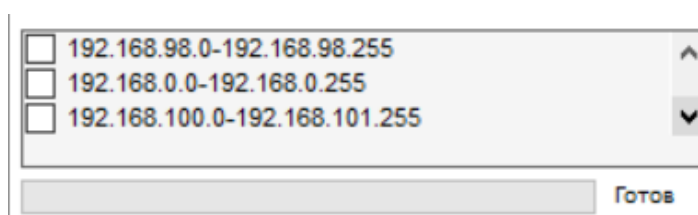


Рисунок 41 — Сохранение нового диапазона после выхода из программы

Диапазон 192.168.100.0-192.168.101.255 удален

Рисунок 42 — Удаление диапазона

Ниже будет описана функция «Выбрать все». Данная функция позволяет оператору быстро выделять все обнаруженные майнеры без необходимости поочерёдной установки флагов для выполнения массовых операций: перезагрузка, установка пулов, переключение Blink.

В элементе DataGridView реализована функция массового выбора всех майнеров через нажатие по заголовку первой колонки (колонка с чекбоксами). При инициализации таблицы в методе InitializeDataGridView() подписываются обработчики событий ColumnHeaderMouseClick и CellPainting. Когда пользователь кликает по заголовку первой колонки, срабатывает метод DataGridViewFoundMiners_ColumnHeaderMouseClick, который инвертирует значение переменной _isSelectedAllChecked и вызывает метод SelectAllMiners(), устанавливающий свойство Selected для всех объектов MinerData в

foundMinersList в соответствии с новым состоянием. Одновременно через событие CellPainting в заголовке колонки отрисовывается визуальное состояние чекбокса. Пример выполнения можно увидеть на рисунке 43.

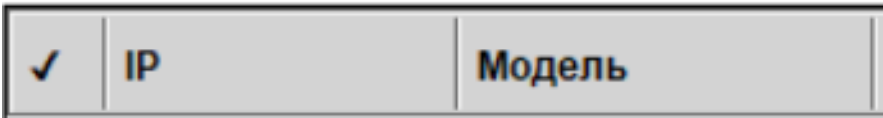


Рисунок 43 — Элемент «Выбрать все»

Функция «Очистка поля пулов» позволяет оператору быстро сбросить ранее введённые данные перед установкой новых пулов, исключая необходимость ручного удаления содержимого каждого поля по отдельности. В программе реализована скрытая функция очистки полей ввода пулов через метод ClearPools, который вызывается при нажатии кнопки "ОЧИСТИТЬ ПОЛЯ". Данный метод в цикле от 0 до 2 обнуляет свойство Text для каждого из трёх массивов текстовых полей: txtPoolUrls[i], txtPoolUsers[i], txtPoolPasswords[i]. После очистки всех полей в строке состояния отображается сообщение об успешном выполнении. Реализация кнопки продемонстрирована на рисунках 44, 45.



Рисунок 44 — Заполненные поля пулов

Рисунок 45 — После применения кнопки «Очистить поля»

Данные функции корректно работают и максимально помогают оператору снизить риск ошибок в работе.

2 Требования безопасности по охране труда при работе с ПЭВМ

В данном разделе описаны требования по обеспечению безопасности жизнедеятельности и охраны труда при работе с персональными электронно-вычислительными машинами (ПЭВМ), а также методы снижения негативного воздействия факторов трудового процесса на здоровье работника.

С 1 января 2021 года в связи с изданием Постановления Правительства РФ от 08.10.2020 № 1631 утратили силу ранее действовавшие СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»[3]. В настоящее время санитарно-эпидемиологические требования к условиям труда при работе с ПЭВМ регламентируются Санитарными правилами СП 2.2.3670-20 (раздел XXII «Требования к организации работ с персональными электронными вычислительными машинами и копировально-множительной техникой») и Методическими рекомендациями МР 2.2.0244-21 (раздел XX «Рекомендации к организации работ с персональными электронными вычислительными машинами и копировально-множительной техникой»). Работодатель обязан разрабатывать инструкции по охране труда для работников, занятых на

ПЭВМ, с учетом оценки профессиональных рисков и специфики деятельности (ст. 214 ТК РФ, Приказ Минтруда № 772н).

2.1 Организация рабочего места

Рабочее место пользователя ПЭВМ должно соответствовать эргономическим требованиям и обеспечивать оптимальную позу для работы. Конструкция рабочего стола и стула (кресла) должна позволять регулировать параметры в соответствии с антропометрическими особенностями работника [4].

Рекомендуемые параметры рабочего места:

- расстояние от глаз до экрана видеодисплейного терминала (ВДТ) — от 60 до 70 см [4];
- центр экрана должен располагаться ниже уровня глаз на 15–20 градусов;
- высота рабочей поверхности стола — в пределах 68–80 см (при отсутствии регулировки — 72,5 см) [4];
- клавиатура размещается на расстоянии 10–30 см от края стола на отдельной регулируемой подставке;
- рабочее кресло должно иметь подлокотники и регулировку высоты сиденья, обеспечивая угол сгиба ног в коленях 90–100 градусов.

На рисунке 46 представлен пример эргономичной организации рабочего места, способствующей снижению утомляемости и профилактике профессиональных заболеваний.

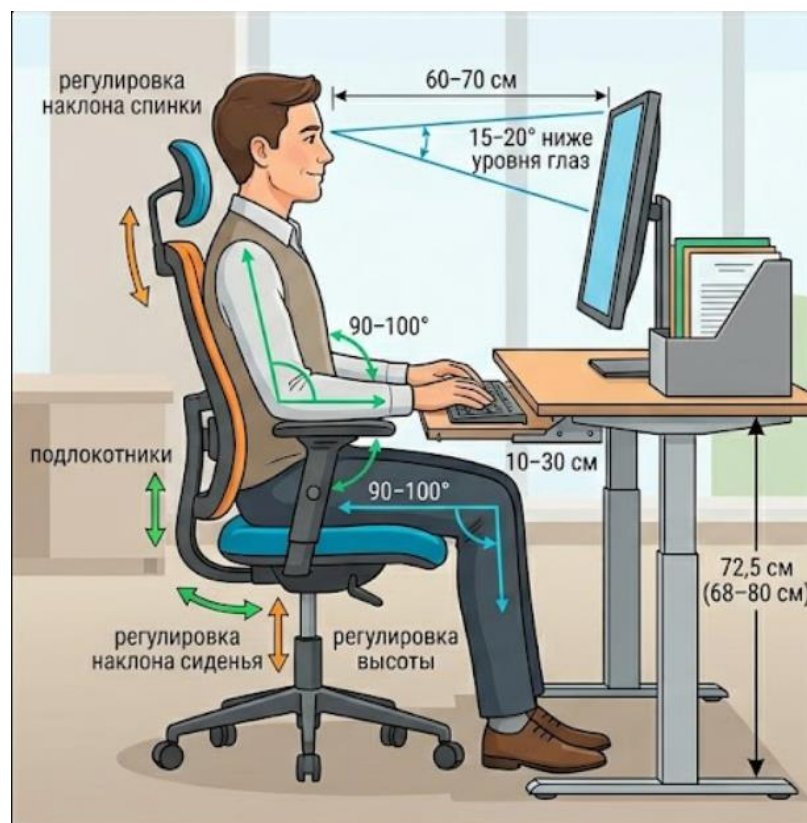


Рисунок 46 — Организация рабочего места

2.2 Освещение рабочего места

Правильно организованное освещение рабочего места является важнейшим фактором сохранения зрительной функции работника. Параметры световой среды должны соответствовать СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания».

Требования к освещению:

- уровень освещенности на рабочем столе в зоне размещения документов — 300–500 люкс;
- окна в помещении должны располагаться преимущественно с северной и северо-восточной сторон;
- на рабочих местах с ПЭВМ следует предусматривать систему общего равномерного освещения и, при необходимости, местного (для работы с документами);

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

50

- искусственные источники света не должны создавать бликов на экране монитора;
- коэффициент естественного освещения (КЕО) должен составлять не менее 1,5 %.

2.3 Электробезопасность

Электробезопасность при эксплуатации ПЭВМ регламентируется Правилами технической эксплуатации электроустановок потребителей и Правилами по охране труда при эксплуатации электроустановок. ПЭВМ являются электроприемниками, работающими от сети переменного тока напряжением 220 В, что представляет потенциальную опасность поражения электрическим током.

Основные меры электробезопасности:

- подключение ПЭВМ и периферийных устройств к электросети должно выполняться через исправные розетки с защитным заземлением (контактом «земля»);
- запрещается использование переносных удлинителей (удлинителей) для подключения нескольких мощных потребителей без устройства защитного отключения (УЗО);
- работнику запрещается самостоятельно вскрывать системный блок, монитор и другие устройства, находящиеся под напряжением;
- при обнаружении неисправности (запах гари, искрение, нагрев вилки) оборудование должно быть немедленно отключено.

2.4 Пожарная безопасность

Помещения для эксплуатации ПЭВМ относятся к категории пожарной опасности (по горючей нагрузке). Обеспечение пожарной безопасности осуществляется в соответствии с Правилами противопожарного режима в Российской Федерации.

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		51

Меры пожарной безопасности при работе на ПЭВМ:

- не допускается размещение электропроводов и кабелей в беспорядочном состоянии (необходимо использование кабельных каналов или коробов) [5];
- запрещается накрывать вентиляционные отверстия системного блока и монитора бумагой или посторонними предметами (во избежание перегрева);
- не разрешается оставлять без присмотра включенное электрооборудование, за исключением случаев, когда это предусмотрено инструкцией по эксплуатации [5];
- курить в помещениях с ПЭВМ разрешается только в специально оборудованных местах, оснащенных системой вентиляции;
- помещение должно быть оснащено первичными средствами пожаротушения (углекислотным огнетушителем) из расчета не менее одного огнетушителя на 50 кв. м площади.

2.5 Режим труда и отдыха

Режим труда и отдыха работников, занятых на ПЭВМ, устанавливается правилами внутреннего трудового распорядка, но с учетом рекомендаций по организации труда. В связи с отменой СанПиН 2.2.2/2.4.1340-03 на сегодняшний день отсутствуют жестко установленные на федеральном уровне нормативы продолжительности непрерывной работы за компьютером. Однако на работодателя возложена обязанность обеспечения режима труда и отдыха работников (ст. 214 ТК РФ), что подразумевает внедрение рациональных режимов с учетом интенсивности зрительной нагрузки [6].

На рисунке 47 представлены рекомендуемые позы и комплексы производственной гимнастики для работников, занятых на ПЭВМ.



Рисунок 47 — Памятка с рекомендациями

Рекомендуемый порядок организации перерывов:

– при работе с высокой интенсивностью зрительного и нервно-эмоционального напряжения рекомендуется устанавливать регламентированные перерывы по 10–15 минут через каждые 45–60 минут работы;

– во время перерывов целесообразно выполнять комплексы упражнений для глаз (вращение, фокусировка на дальних и ближних объектах) и упражнения для шейного отдела позвоночника (наклоны, повороты головы);

– при 8-часовой рабочей смене рекомендуется устанавливать два регламентированных перерыва (один — через 2 часа после начала смены, второй — через 2 часа после обеденного перерыва) и один обеденный перерыв продолжительностью 30–60 минут.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

53

Заключение

В процессе выполнения задания были получены навыки программирования в среде Microsoft Visual Studio 2022 на языке C#, поиска и анализа существующих методов разработки управляющих программ, анализа документации, сравнения аналогичных систем. За период выполнения работы были освоены базовые понятия и технологии разработки программ систем управления, изучены основные инструменты создания логики программ для работы с майнинг-устройствами. Во время выполнения данной работы были разработаны: программа управления майнинг-устройствами в локальной сети, настроен пользовательский интерфейс пользователя. В дальнейшем проект может быть расширен путем добавления нового функционала и новой логики взаимодействия со сторонними устройствами.

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

Список использованных источников

1. Liberty J. Programming C#: building. NET applications with C. – "O'Reilly Media, Inc.", 2005 [Электронный ресурс] / Режим доступа: https://books.google.com/books?hl=ru&lr=&id=Ns4e0ko-S7QC&oi=fnd&pg=PR5&dq=C%23&ots=dvsXS4Z3d7&sig=KrAbB2NJvhs_a76ZDyvCH1ApOR4 (дата обращения 12.05.2026)
2. Документация по Visual Studio. [Электронный ресурс]
Режим доступа: <https://learn.microsoft.com/ru-ru/visualstudio/windows/?view=visualstudio> свободный (дата обращения: 15.05.2026).
3. Блинова В. ОЦЕНКА БЕЗОПАСНОСТИ ЧЕЛОВЕКА ПРИ РАБОТЕ С ПЕРСОНАЛЬНЫМИ ЭЛЕКТРОННО-ВЫЧИСЛИТЕЛЬНЫМИ МАШИНАМИ //Международная научно-техническая конференция молодых ученых БГТУ им. ВГ Шухова. – 2015. – С. 441-444. [Электронный ресурс] / Режим доступа: <https://elibrary.ru/item.asp?id=24611771> свободный (дата обращения: 01.05.2026).
4. Васильев А. В. и др. Особенности проектирования рабочего места оператора ЭВМ с учетом эргономических требований. – 2015. – С. 99-102. [Электронный ресурс].URL: <https://elibrary.ru/item.asp?id=24887296> (дата обращения 25.05.2026)
5. Типовая инструкция о мерах пожарной безопасности в компьютерном классе [Электронный ресурс]. URL: https://osu.ru/docs/official/fire_safety/instr_comp_class_2021.pdf (дата обращения 25.05.2026)
6. Шумилин В. К., Елин А. М., Легкий Н. М. РЕКОМЕНДУЕМЫЙ ПОРЯДОК УПРАВЛЕНИЯ ПРОФЕССИОНАЛЬНЫМИ РИСКАМИ НА РАБОЧИХ МЕСТАХ С ПЭВМ С УЧЕТОМ ИЗМЕНЕНИЙ НОРМАТИВНЫХ ПРАВОВЫХ АКТОВ //Труды Ростовского государственного университета

путей сообщения. – 2022. – №. 2 (59). – С. 183-197. [Электронный ресурс] Режим доступа: <https://cyberleninka.ru/article/n/rekomenduemyy-poryadok-upravleniya-professionalnymi-riskami-na-rabochih-mestah-s-privyichnymi-s-uchetom-izmeneniy-normativnyh-pravovyh-svobodnyy> свободный. (дата обращения: 25.05.2026).

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		56

Приложение А

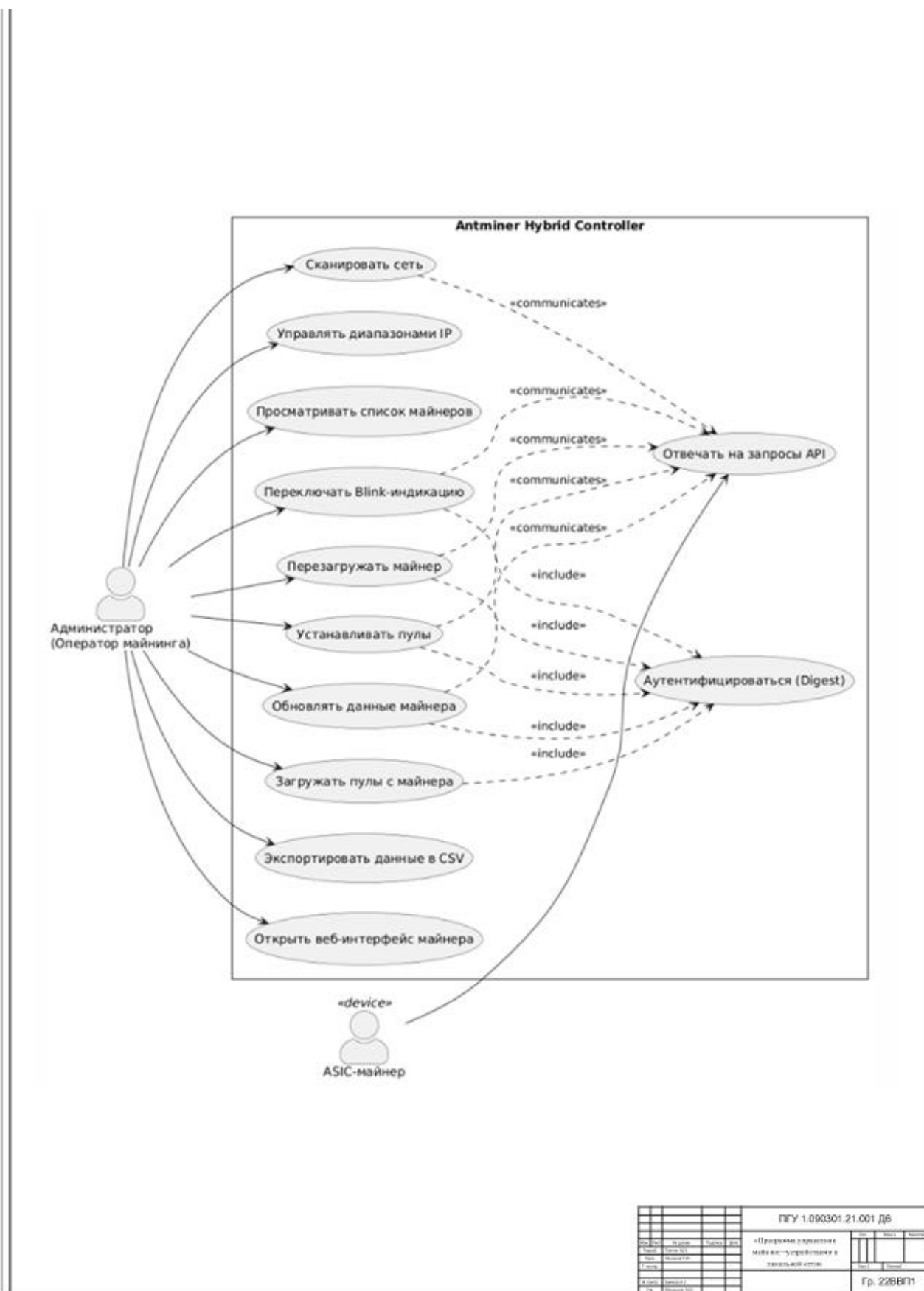
(обязательное)

«Программа управления майнинг–устройствами в локальной сети»

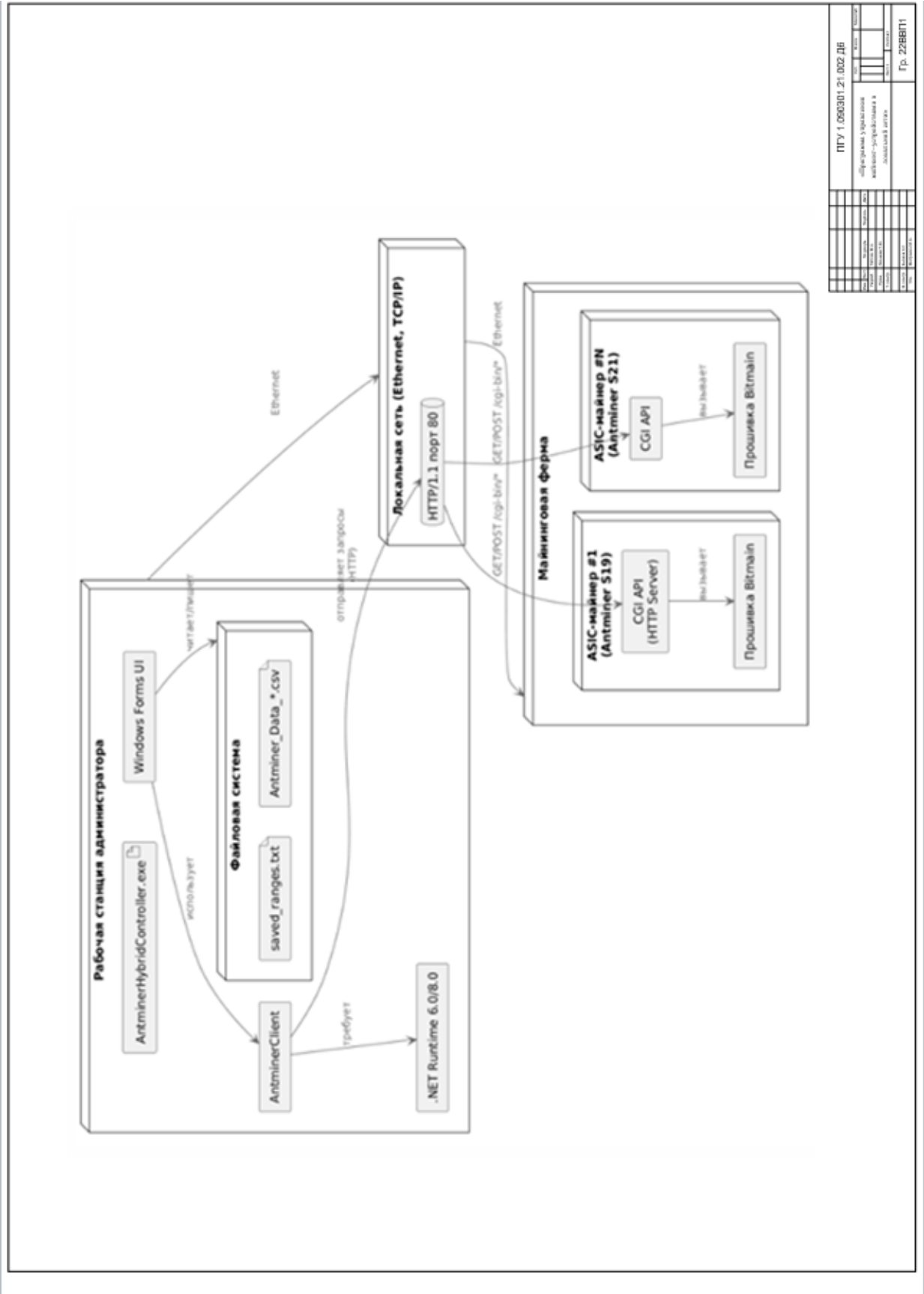
UML диаграммы и схемы

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		57

Приложение А.1 – Диаграмма прецедентов пользователя

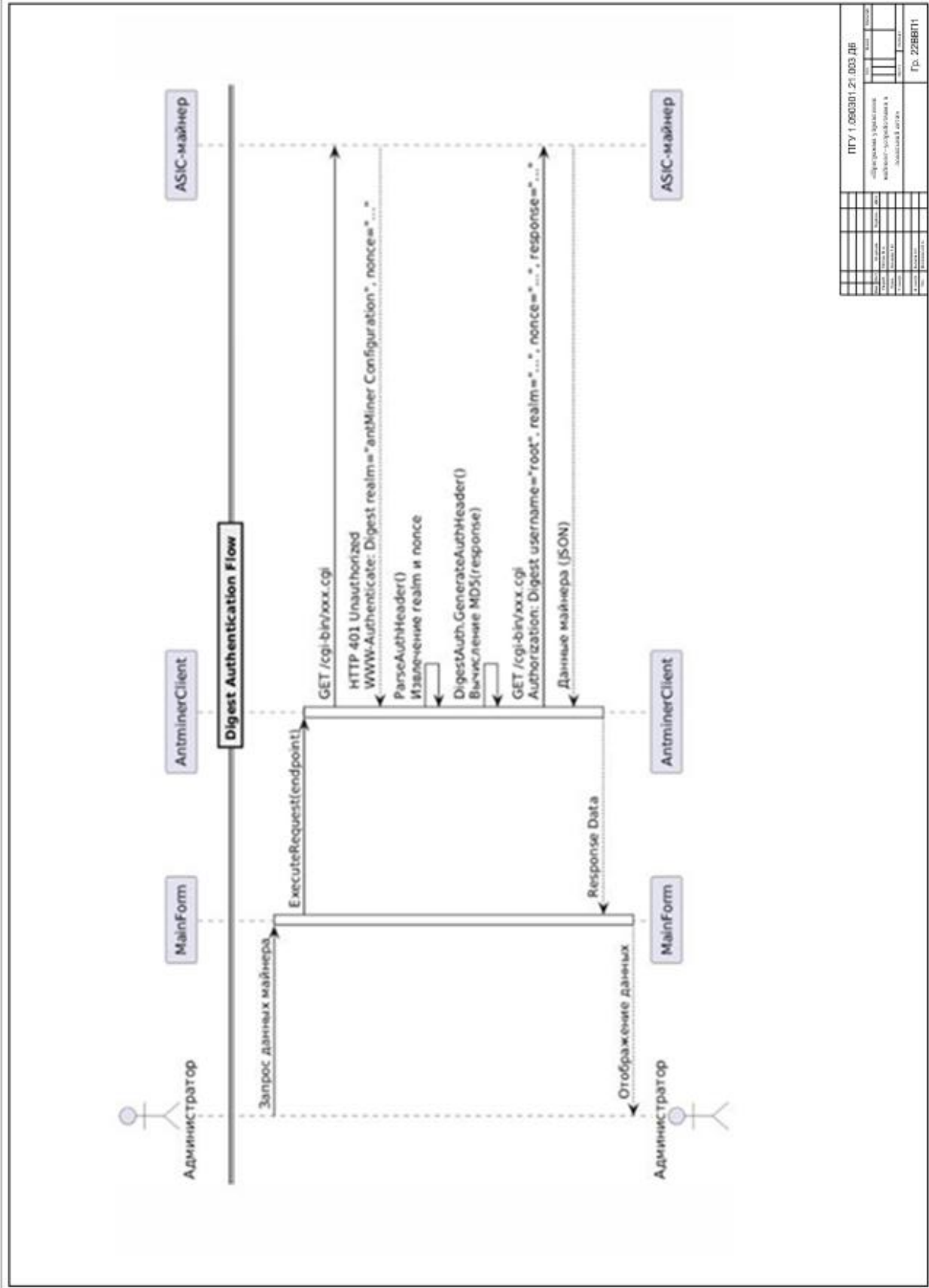


Приложение А.2 – Диаграмма разворачивания

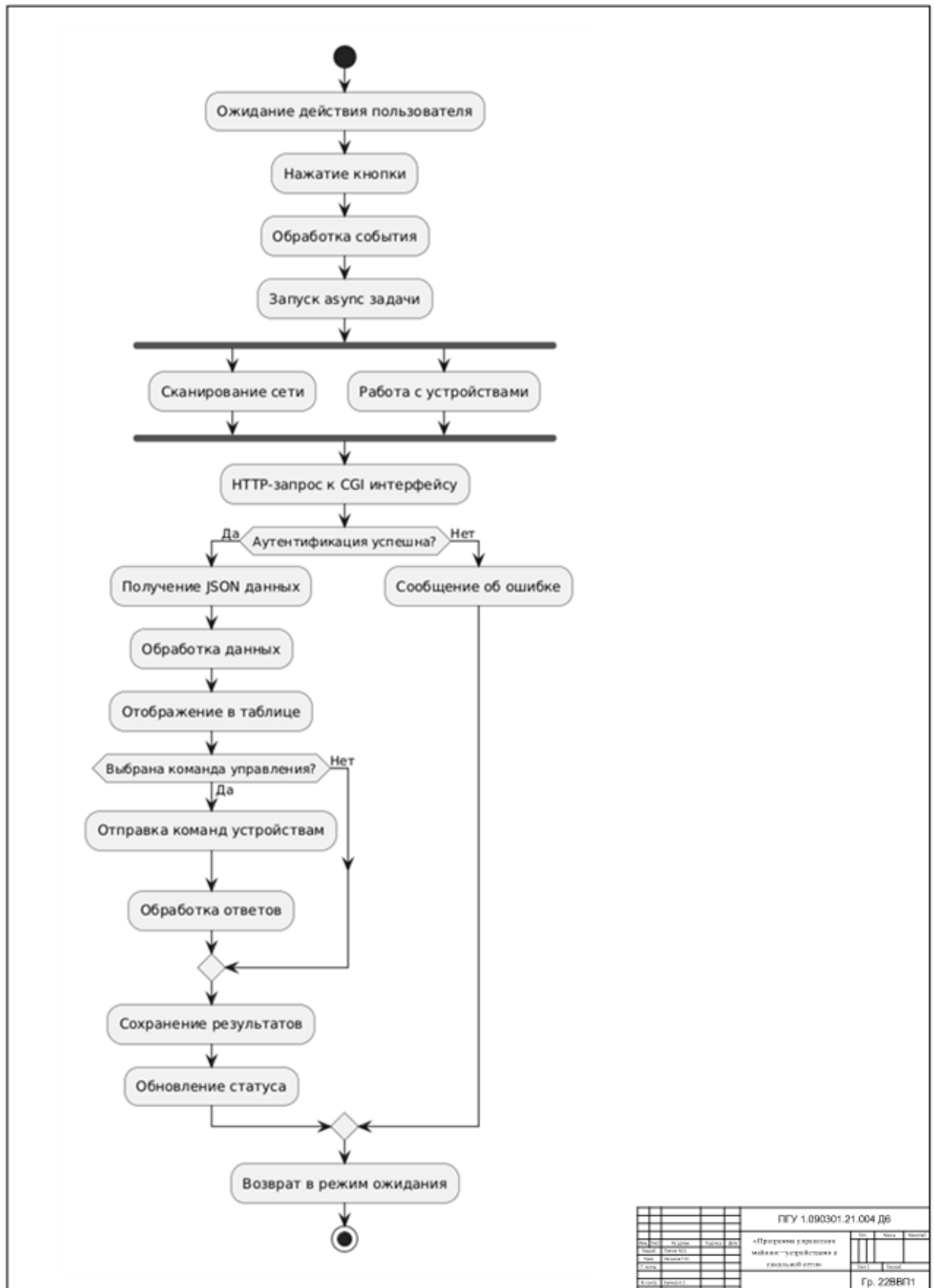


ПГУ 1.090301.21.002 ДВ		Изм.		Лист	№ докум.	Подпись	Дата
Исходный текст		1	1	1			
Исходный текст		2	2	2			
Исходный текст		3	3	3			
Исходный текст		4	4	4			
Исходный текст		5	5	5			
Исходный текст		6	6	6			
Исходный текст		7	7	7			
Исходный текст		8	8	8			
Исходный текст		9	9	9			
Исходный текст		10	10	10			
Исходный текст		11	11	11			
Исходный текст		12	12	12			
Исходный текст		13	13	13			
Исходный текст		14	14	14			
Исходный текст		15	15	15			
Исходный текст		16	16	16			
Исходный текст		17	17	17			
Исходный текст		18	18	18			
Исходный текст		19	19	19			
Исходный текст		20	20	20			
Исходный текст		21	21	21			
Исходный текст		22	22	22			
Исходный текст		23	23	23			
Исходный текст		24	24	24			
Исходный текст		25	25	25			
Исходный текст		26	26	26			
Исходный текст		27	27	27			
Исходный текст		28	28	28			
Исходный текст		29	29	29			
Исходный текст		30	30	30			
Исходный текст		31	31	31			
Исходный текст		32	32	32			
Исходный текст		33	33	33			
Исходный текст		34	34	34			
Исходный текст		35	35	35			
Исходный текст		36	36	36			
Исходный текст		37	37	37			
Исходный текст		38	38	38			
Исходный текст		39	39	39			
Исходный текст		40	40	40			
Исходный текст		41	41	41			
Исходный текст		42	42	42			
Исходный текст		43	43	43			
Исходный текст		44	44	44			
Исходный текст		45	45	45			
Исходный текст		46	46	46			
Исходный текст		47	47	47			
Исходный текст		48	48	48			
Исходный текст		49	49	49			
Исходный текст		50	50	50			
Исходный текст		51	51	51			
Исходный текст		52	52	52			
Исходный текст		53	53	53			
Исходный текст		54	54	54			
Исходный текст		55	55	55			
Исходный текст		56	56	56			
Исходный текст		57	57	57			
Исходный текст		58	58	58			
Исходный текст		59	59	59			
Исходный текст		60	60	60			
Исходный текст		61	61	61			
Исходный текст		62	62	62			
Исходный текст		63	63	63			
Исходный текст		64	64	64			
Исходный текст		65	65	65			
Исходный текст		66	66	66			
Исходный текст		67	67	67			
Исходный текст		68	68	68			
Исходный текст		69	69	69			
Исходный текст		70	70	70			
Исходный текст		71	71	71			
Исходный текст		72	72	72			
Исходный текст		73	73	73			
Исходный текст		74	74	74			
Исходный текст		75	75	75			
Исходный текст		76	76	76			
Исходный текст		77	77	77			
Исходный текст		78	78	78			
Исходный текст		79	79	79			
Исходный текст		80	80	80			
Исходный текст		81	81	81			
Исходный текст		82	82	82			
Исходный текст		83	83	83			
Исходный текст		84	84	84			
Исходный текст		85	85	85			
Исходный текст		86	86	86			
Исходный текст		87	87	87			
Исходный текст		88	88	88			
Исходный текст		89	89	89			
Исходный текст		90	90	90			
Исходный текст		91	91	91			
Исходный текст		92	92	92			
Исходный текст		93	93	93			
Исходный текст		94	94	94			
Исходный текст		95	95	95			
Исходный текст		96	96	96			
Исходный текст		97	97	97			
Исходный текст		98	98	98			
Исходный текст		99	99	99			
Исходный текст		100	100	100			
Исходный текст		101	101	101			
Исходный текст		102	102	102			
Исходный текст		103	103	103			
Исходный текст		104	104	104			
Исходный текст		105	105	105			
Исходный текст		106	106	106			
Исходный текст		107	107	107			
Исходный текст		108	108	108			
Исходный текст		109	109	109			
Исходный текст		110	110	110			
Исходный текст		111	111	111			
Исходный текст		112	112	112			
Исходный текст		113	113	113			
Исходный текст		114	114	114			
Исходный текст		115	115	115			
Исходный текст		116	116	116			
Исходный текст		117	117	117			
Исходный текст		118	118	118			
Исходный текст		119	119	119			
Исходный текст		120	120	120			
Исходный текст		121	121	121			
Исходный текст		122	122	122			
Исходный текст		123	123	123			
Исходный текст		124	124	124			
Исходный текст		125	125	125			
Исходный текст		126	126	126			
Исходный текст		127	127	127			
Исходный текст		128	128	128			
Исходный текст		129	129	129			
Исходный текст		130	130	130			
Исходный текст		131	131	131			
Исходный текст		132	132	132			
Исходный текст		133	133	133			
Исходный текст		134	134	134			
Исходный текст		135	135	135			
Исходный текст		136	136	136			
Исходный текст		137	137	137			
Исходный текст		138	138	138			
Исходный текст		139	139	139			
Исходный текст		140	140	140			
Исходный текст		141	141	141			
Исходный текст		142	142	142			
Исходный текст		143	143	143			
Исходный текст		144	144	144			
Исходный текст		145	145	145			
Исходный текст		146	146	146			
Исходный текст		147	147	147			
Исходный текст		148	148	148			
Исходный текст		149	149	149			
Исходный текст		150	150	150			
Исходный текст		151	151	151			
Исходный текст		152	152	152			
Исходный текст		153	153	153			
Исходный текст		154	154	154			
Исходный текст		155	155	155			
Исходный текст		156	156	156			
Исходный текст		157	157	157			
Исходный текст		158	158	158			
Исходный текст		159	159	159			
Исходный текст		160	160	160			
Исходный текст		161	161	161			
Исходный текст		162	162	162			
Исходный текст		163	163	163			
Исходный текст		164	164	164			
Исходный текст		165	165	165			
Исходный текст		166	166	166			
Исходный текст		167	167	167			
Исходный текст		168	168	168			
Исходный текст		169	169	169			
Исходный текст		170	170	170			
Исходный текст		171	171	171			
Исходный текст		172	172	172			
Исходный текст		173	173	173			
Исходный текст		174	174	174			
Исходный текст		175	175	175			
Исходный текст		176	176	176			
Исходный текст		177	177	177			
Исходный текст		178	178	178			
Исходный текст		179	179	179			
Исходный текст		180	180	180			
Исходный текст		181	181	181			
Исходный текст		182	182	182			
Исходный текст		183	183	183			
Исходный текст		184	184	184			
Исходный текст		185	185	185			
Исходный текст		186	186	186			
Исходный текст		187	187	187			
Исходный текст		188	188	188			
Исходный текст		189	189	189			
Исходный текст		190	190	190			
Исходный текст		191	191	191			
Исходный текст		192	192	192			
Исходный текст		193	193	193			
Исходный текст		194	194	194			
Исходный текст		195	195	195			
Исходный текст		196	196	196			
Исходный текст		197	197	197			
Исходный текст		198	198	198			
Исходный текст		199	199	199			
Исходный текст		200	200	200			
Исходный текст		201	201	201			
Исходный текст		202	202	202			
Исходный текст		203	203	203			
Исходный текст		204	204	204			
Исходный текст		205	205	205			
Исходный текст		206	206	206			
Исходный текст		207	207	207			
Исходный текст		208	208	208			
Исходный текст		209	209	209			
Исходный текст		210	210	210			
Исходный текст		211	211	211			
Исходный текст		212	212	212			
Исходный текст		213	213	213			
Исходный текст		214	214	214			
Исходный текст		215	215	215			
Исходный текст		216	216	216			
Исходный текст		217	217	217			
Исходный текст		218	218	218			
Исходный текст		219	219	219			
Исходный текст		220	220	220			
Исходный текст		221	221	221			
Исходный текст		222	222	222			
Исходный текст		223	223	223			
Исходный текст		224	224	224			
Исходный текст		225	225	225			
Исходный текст		226	226	226			
Исходный текст		227	227	227			
Исходный текст		228	228	228			
Исходный текст		229	229	229			
Исходный текст		230	230	230			
Исходный текст		231	231	231			
Исходный текст		232	232	232			
Исходный текст		233	233	233			
Исходный текст		234	234	234			
Исходный текст		235	235	235			
Исходный текст		236	236	236			
Исходный текст		237	237	237			
Исходный текст		238	238	238			
Исходный текст		239					

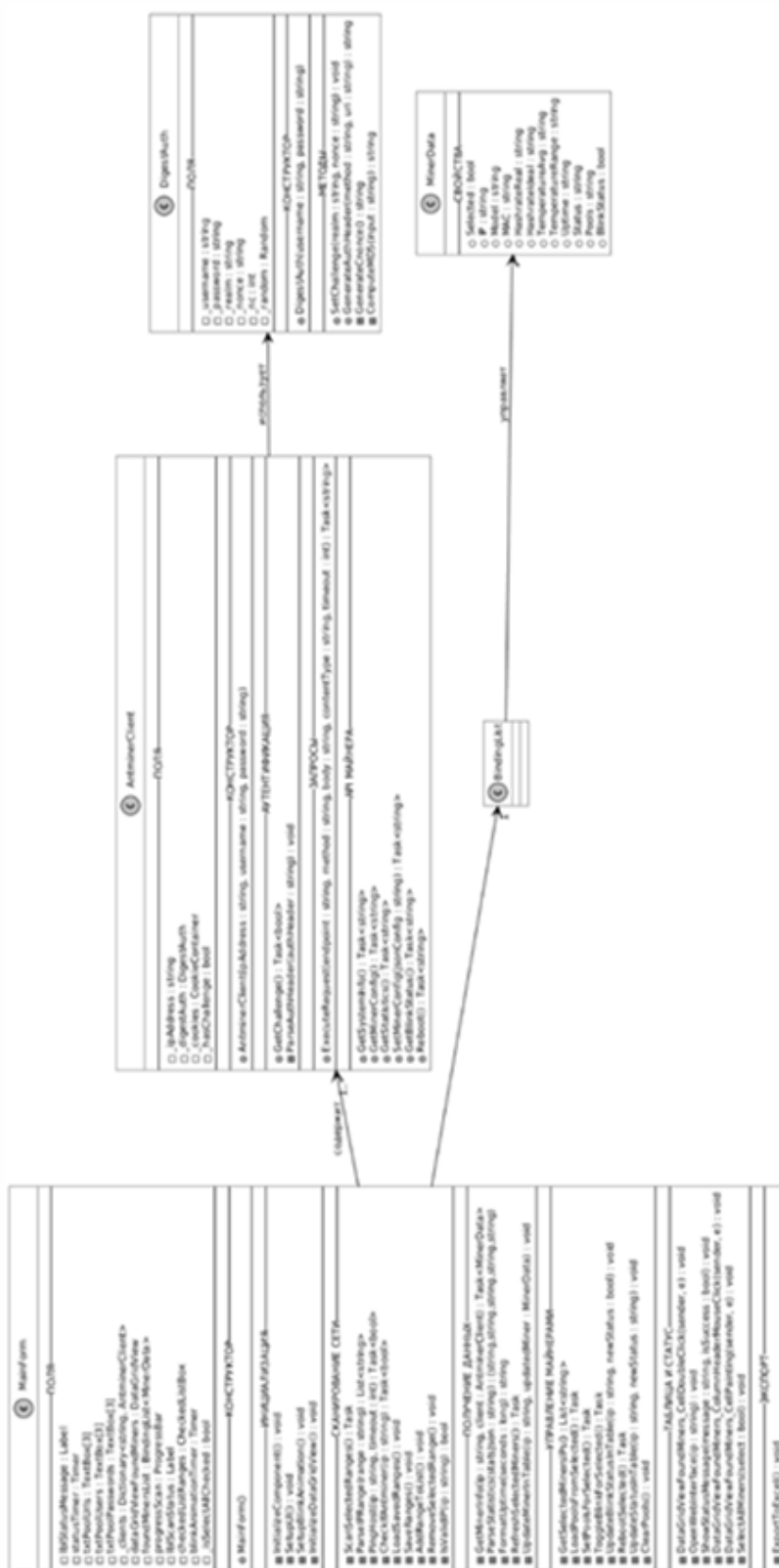
Приложение А.3 – Диаграмма последовательностей



Приложение А.4 – Функциональная схема работы проекта



Приложение А.5 – Диаграмма классов

[illegible]

Приложение Б

(обязательное)

«Программа для управления майнинг–устройствами в локальной сети»

Листинг программы

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		63

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.NetworkInformation;
using System.Net.Sockets;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;

namespace AntminerHybridController
{
    public partial class MainForm : Form
    {
        // Элементы UI
        private Label lblStatusMessage;
        private System.Windows.Forms.Timer statusTimer;

        // Поля для пулов
        private TextBox[] txtPoolUrls = new TextBox[3];
        private TextBox[] txtPoolUsers = new TextBox[3];
        private TextBox[] txtPoolPasswords = new TextBox[3];

        // Клиенты для найденных асиков
        private Dictionary<string, AntminerClient> _clients = new
Dictionary<string, AntminerClient>();

        // DataGridView для отображения найденных асиков
        private DataGridView dataGridViewFoundMiners;
        private BindingList<MinerData> foundMinersList;

        // Элементы для сканирования сети
        private ProgressBar progressScan;
        private Label lblScanStatus;
        private CheckedListBox checkedListRanges;
        private Button btnAddRange;
        private Button btnRemoveRange;
        private Button btnScanSelectedRanges;
        private TextBox txtStartIP;
        private TextBox txtEndIP;
        private Label lblRangeSeparator;

        // Таймер для анимации Blink
        private System.Windows.Forms.Timer blinkAnimationTimer;
    }
}

```

```

// Переменная для отслеживания состояния "Выбрать все"
private bool _isSelectedAllChecked = false;

// Класс для хранения данных асика
public class MinerData
{
    public bool Selected { get; set; }
    public string IP { get; set; }
    public string Model { get; set; }
    public string MAC { get; set; }
    public string HashrateReal { get; set; }
    public string HashrateIdeal { get; set; }
    public string TemperatureAvg { get; set; }
    public string TemperatureRange { get; set; }
    public string Uptime { get; set; }
    public string Status { get; set; }
    public string Pools { get; set; }
    public bool BlinkStatus { get; set; }
}

public MainForm()
{
    InitializeComponent();
    SetupUI();
    InitializeDataGridView();
    LoadSavedRanges();
    SetupBlinkAnimation();
}

private void InitializeComponent()
{
    this.SuspendLayout();
    this.ClientSize = new System.Drawing.Size(1400, 800);
    this.Name = "MainForm";
    this.Text = "Antminer Hybrid Controller v2.0";
    this.ResumeLayout(false);
}

private void SetupBlinkAnimation()
{
    blinkAnimationTimer = new
System.Windows.Forms.Timer();
    blinkAnimationTimer.Interval = 500;
    blinkAnimationTimer.Tick += (s, e) =>
    {
        if (dataGridViewFoundMiners != null)
        {
            dataGridViewFoundMiners.Invalidate();
        }
    };
    blinkAnimationTimer.Start();
}

```

```

private void SetupUI()
{
    this.Text = "Antminer Hybrid Controller v2.0";
    this.MinimumSize = new Size(1300, 700);
    this.StartPosition = FormStartPosition.CenterScreen;
    this.BackColor = Color.FromArgb(240, 240, 240);

    statusTimer = new System.Windows.Forms.Timer();
    statusTimer.Interval = 3000;
    statusTimer.Tick += (s, e) =>
    {
        lblStatusMessage.Text = "";
        statusTimer.Stop();
    };

    // Основной контейнер - вертикальный SplitContainer
    SplitContainer mainSplit = new SplitContainer
    {
        Dock = DockStyle.Fill,
        Orientation = Orientation.Vertical,
        SplitterDistance = 450,
        IsSplitterFixed = false
    };

    Panel topPanel = new Panel
    {
        Dock = DockStyle.Fill,
        BackColor = Color.White,
        Padding = new Padding(10)
    };

    Label lblMinersTitle = new Label
    {
        Text = "НАЙДЕННЫЕ АСИКИ",
        Location = new Point(10, 5),
        Size = new Size(300, 25),
        Font = new Font("Arial", 12, FontStyle.Bold),
        ForeColor = Color.FromArgb(32, 82, 51)
    };

    dataGridViewFoundMiners = new DataGridView
    {
        Location = new Point(10, 35),
        Size = new Size(topPanel.Width - 20, 330),
        Anchor = AnchorStyles.Left | AnchorStyles.Top |
AnchorStyles.Right | AnchorStyles.Bottom,
        BackgroundColor = Color.White,
        BorderStyle = BorderStyle.FixedSingle,
        AllowUserToAddRows = false,
        AllowUserToDeleteRows = false,
        RowHeadersVisible = false,
    };
}

```

```

        SelectionMode                                     =
DataGridViewSelectionMode.FullRowSelect,
        MultiSelect = true,
        ReadOnly = false
    };

    // Добавляем обработчик двойного клика для открытия
    веб-интерфейса
    dataGridViewFoundMiners.CellDoubleClick              +=
DataGridViewFoundMiners_CellDoubleClick;

    Panel bottomPanel = new Panel
    {
        Dock = DockStyle.Fill,
        BackColor = Color.White,
        Padding = new Padding(10)
    };

    // Создаем горизонтальный SplitContainer для
    разделения на левую и правую части
    SplitContainer bottomSplit = new SplitContainer
    {
        Dock = DockStyle.Fill,
        Orientation = Orientation.Horizontal,
        SplitterDistance = 180,
        IsSplitterFixed = false
    };

    Panel topBottomPanel = new Panel
    {
        Dock = DockStyle.Fill,
        BackColor = Color.White,
        Padding = new Padding(5)
    };

    // Создаем горизонтальный SplitContainer для кнопок и
    сканирования
    SplitContainer controlSplit = new SplitContainer
    {
        Dock = DockStyle.Fill,
        Orientation = Orientation.Vertical,
        SplitterDistance = 450,
        IsSplitterFixed = false
    };

    // ЛЕВАЯ ЧАСТЬ - КНОПКИ УПРАВЛЕНИЯ АСИКАМИ
    Panel buttonsPanel = new Panel
    {
        Dock = DockStyle.Fill,
        BackColor = Color.WhiteSmoke,
        BorderStyle = BorderStyle.FixedSingle,
        Padding = new Padding(5)
    }

```

```

};

Label lblControlTitle = new Label
{
    Text = "УПРАВЛЕНИЕ",
    Location = new Point(5, 5),
    Size = new Size(200, 20),
    Font = new Font("Arial", 9, FontStyle.Bold),
    ForeColor = Color.FromArgb(32, 82, 51)
};

// Цвета кнопок
Color color1 = Color.FromArgb(80, 204, 127);
Color color2 = Color.FromArgb(64, 163, 102);
Color color3 = Color.FromArgb(93, 239, 150);
Color color4 = Color.FromArgb(48, 122, 76);
Color color5 = Color.FromArgb(32, 82, 51);

// Кнопки управления - все видны
Button btnToggleBlink = new Button
{
    Text = "💡 BLINK",
    Location = new Point(5, 30),
    Size = new Size(140, 35),
    BackColor = color3,
    ForeColor = color5,
    FlatStyle = FlatStyle.Flat,
    Font = new Font("Arial", 9, FontStyle.Bold)
};
btnToggleBlink.FlatAppearance.BorderSize = 0;
btnToggleBlink.Click += async (s, e) => await
ToggleBlinkForSelected();

Button btnReboot = new Button
{
    Text = "⚡ ПЕРЕЗАГРУЗКА",
    Location = new Point(5, 70),
    Size = new Size(140, 35),
    BackColor = color4,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Font = new Font("Arial", 9, FontStyle.Bold)
};
btnReboot.FlatAppearance.BorderSize = 0;
btnReboot.Click += async (s, e) => await
RebootSelected();

Button btnSetPools = new Button
{
    Text = "📁 УСТ. ПУЛЫ",
    Location = new Point(5, 110),

```

```

        Size = new Size(140, 35),
        BackColor = color5,
        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat,
        Font = new Font("Arial", 9, FontStyle.Bold)
    };
    btnSetPools.FlatAppearance.BorderSize = 0;
    btnSetPools.Click += async (s, e) => await
SetPoolsForSelected();

    Button btnLoadPools = new Button
    {
        Text = "🔧 ЗАГР. ПУЛЫ",
        Location = new Point(5, 150),
        Size = new Size(140, 35),
        BackColor = color2,
        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat,
        Font = new Font("Arial", 9, FontStyle.Bold)
    };
    btnLoadPools.FlatAppearance.BorderSize = 0;
    btnLoadPools.Click += async (s, e) => await
LoadPoolsFromSelected();

    // Кнопка экспорта в Excel
    Button btnExportToExcel = new Button
    {
        Text = "📊 ЭКСПОРТ В EXCEL",
        Location = new Point(5, 190),
        Size = new Size(140, 35),
        BackColor = Color.FromArgb(76, 175, 80),
        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat,
        Font = new Font("Arial", 9, FontStyle.Bold)
    };
    btnExportToExcel.FlatAppearance.BorderSize = 0;
    btnExportToExcel.Click += (s, e) => ExportToExcel();

    buttonsPanel.Controls.AddRange(new Control[] {
        lblControlTitle,    btnToggleBlink,    btnReboot,
        btnSetPools, btnLoadPools,
        btnExportToExcel
    });

    // ПРАВАЯ ЧАСТЬ - СКАНИРОВАНИЕ СЕТИ
    Panel scanRightPanel = new Panel
    {
        Dock = DockStyle.Fill,
        BackColor = Color.White,
        BorderStyle = BorderStyle.FixedSingle,
        Padding = new Padding(5)
    }

```

```

};

Label lblScanTitle = new Label
{
    Text = "СКАНИРОВАНИЕ СЕТИ",
    Location = new Point(5, 5),
    Size = new Size(200, 20),
    Font = new Font("Arial", 9, FontStyle.Bold),
    ForeColor = color5
};

// Панель ввода диапазона
Label lblStartIP = new Label
{
    Text = "С:",
    Location = new Point(5, 35),
    Size = new Size(20, 20)
};

txtStartIP = new TextBox
{
    Location = new Point(25, 35),
    Size = new Size(100, 20),
    Text = "192.168.96.0"
};

lblRangeSeparator = new Label
{
    Text = "до",
    Location = new Point(130, 35),
    Size = new Size(20, 20),
    TextAlign = ContentAlignment.MiddleCenter
};

txtEndIP = new TextBox
{
    Location = new Point(150, 35),
    Size = new Size(100, 20),
    Text = "192.168.102.255"
};

btnAddRange = new Button
{
    Text = "✚ ДОБАВИТЬ",
    Location = new Point(255, 35),
    Size = new Size(80, 23),
    BackColor = color1,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Font = new Font("Arial", 7, FontStyle.Bold)
};

```

```

btnAddRange.FlatAppearance.BorderSize = 0;
btnAddRange.Click += (s, e) => AddRangeToList();

// Список диапазонов
Label lblSavedRanges = new Label
{
    Text = "СОХРАНЕННЫЕ ДИАПАЗОНЫ:",
    Location = new Point(5, 65),
    Size = new Size(150, 15),
    Font = new Font("Arial", 7, FontStyle.Bold)
};

Panel rangesListPanel = new Panel
{
    Location = new Point(5, 85),
    Size = new Size(300, 60),
    BorderStyle = BorderStyle.FixedSingle,
    BackColor = Color.WhiteSmoke
};

checkedListRanges = new CheckedListBox
{
    Dock = DockStyle.Fill,
    CheckOnClick = true,
    BorderStyle = BorderStyle.None,
    BackColor = Color.WhiteSmoke,
    Font = new Font("Arial", 8)
};

// Панель кнопок управления диапазонами
Panel rangesButtonsPanel = new Panel
{
    Location = new Point(310, 85),
    Size = new Size(100, 60)
};

btnRemoveRange = new Button
{
    Text = "🗑 УДАЛИТЬ",
    Location = new Point(0, 0),
    Size = new Size(100, 25),
    BackColor = Color.LightCoral,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Font = new Font("Arial", 6, FontStyle.Bold)
};
btnRemoveRange.FlatAppearance.BorderSize = 0;
btnRemoveRange.Click += (s, e) =>
RemoveSelectedRange();

btnScanSelectedRanges = new Button

```

```

        {
            Text = "🔍 СКАНИРОВАТЬ",
            Location = new Point(0, 28),
            Size = new Size(100, 25),
            BackColor = color2,
            ForeColor = Color.White,
            FlatStyle = FlatStyle.Flat,
            Font = new Font("Arial", 6, FontStyle.Bold)
        };
        btnScanSelectedRanges.FlatAppearance.BorderSize = 0;
        btnScanSelectedRanges.Click += async (s, e) => await
ScanSelectedRanges();

        rangesButtonsPanel.Controls.AddRange(new Control[] {
            btnRemoveRange, btnScanSelectedRanges
        });

        rangesListPanel.Controls.Add(checkedListRanges);

        // Прогресс и статус
        progressScan = new ProgressBar
        {
            Location = new Point(5, 150),
            Size = new Size(250, 15),
            Style = ProgressBarStyle.Continuous
        };

        lblScanStatus = new Label
        {
            Location = new Point(260, 150),
            Size = new Size(150, 15),
            Text = "Готов",
            Font = new Font("Arial", 7)
        };

        scanRightPanel.Controls.AddRange(new Control[] {
            lblScanTitle,          lblStartIP,          txtStartIP,
            lblRangeSeparator, txtEndIP, btnAddRange,
            lblSavedRanges,          rangesListPanel,
            rangesButtonsPanel, progressScan, lblScanStatus
        });

        controlSplit.Panel1.Controls.Add(buttonsPanel);
        controlSplit.Panel2.Controls.Add(scanRightPanel);
        topBottomPanel.Controls.Add(controlSplit);

        Panel poolsMainPanel = new Panel
        {
            Dock = DockStyle.Fill,
            BackColor = Color.White,
            BorderStyle = BorderStyle.FixedSingle,

```

```

        Padding = new Padding(10)
    };

    Label lblPools = new Label
    {
        Text = "НАСТРОЙКА ПУЛОВ",
        Location = new Point(10, 5),
        Size = new Size(200, 25),
        Font = new Font("Arial", 10, FontStyle.Bold),
        ForeColor = color5
    };

    // Создаем таблицу для пулов с нормальными размерами
    int poolY = 35;
    for (int i = 0; i < 3; i++)
    {
        // Заголовок пула
        Label lblPool = new Label
        {
            Text = $"ПУЛ {i + 1}:",
            Location = new Point(10, poolY + (i * 35)),
            Size = new Size(50, 25),
            Font = new Font("Arial", 8, FontStyle.Bold),
            TextAlign = ContentAlignment.MiddleRight
        };

        // URL
        Label lblUrl = new Label
        {
            Text = "URL:",
            Location = new Point(70, poolY + (i * 35)),
            Size = new Size(30, 25),
            Font = new Font("Arial", 8),
            TextAlign = ContentAlignment.MiddleRight
        };

        txtPoolUrls[i] = new TextBox
        {
            Location = new Point(100, poolY + (i * 35)),
            Size = new Size(100, 25),
            Font = new Font("Arial", 9),
            Text = ""
        };

        // Worker
        Label lblWorker = new Label
        {
            Text = "Worker:",
            Location = new Point(210, poolY + (i * 35)),
            Size = new Size(50, 25),
            Font = new Font("Arial", 8),
            TextAlign = ContentAlignment.MiddleRight
        };
    }

```

```

};

txtPoolUsers[i] = new TextBox
{
    Location = new Point(260, poolY + (i * 35)),
    Size = new Size(100, 25),
    Font = new Font("Arial", 9),
    Text = ""
};

// Password
Label lblPass = new Label
{
    Text = "Pass:",
    Location = new Point(360, poolY + (i * 35)),
    Size = new Size(40, 25),
    Font = new Font("Arial", 8),
    TextAlign = ContentAlignment.MiddleRight
};

txtPoolPasswords[i] = new TextBox
{
    Location = new Point(400, poolY + (i * 35)),
    Size = new Size(50, 25),
    Font = new Font("Arial", 9),
    UseSystemPasswordChar = true,
    Text = ""
};

poolsMainPanel.Controls.AddRange(new Control[] {
    lblPool, lblUrl, txtPoolUrls[i],
    lblWorker, txtPoolUsers[i],
    lblPass, txtPoolPasswords[i]
});
}

// Кнопка очистки
Button btnClearPools = new Button
{
    Text = "🗑 ОЧИСТИТЬ ПОЛЯ",
    Location = new Point(10, 140),
    Size = new Size(150, 30),
    BackColor = Color.LightCoral,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Font = new Font("Arial", 8, FontStyle.Bold)
};
btnClearPools.FlatAppearance.BorderSize = 0;
btnClearPools.Click += (s, e) => ClearPools();

poolsMainPanel.Controls.Add(btnClearPools);

```

```

        // Собираем нижнюю часть
        bottomSplit.Panel1.Controls.Add(topBottomPanel);
        bottomSplit.Panel2.Controls.Add(poolMainPanel);
        bottomPanel.Controls.Add(bottomSplit);

        Panel tableControlPanel = new Panel
        {
            Location = new Point(10, 370),
            Size = new Size(topPanel.Width - 20, 40),
            Anchor = AnchorStyles.Left | AnchorStyles.Bottom
| AnchorStyles.Right,
            BackColor = Color.Transparent
        };

        topPanel.Controls.Add(lblMinersTitle);
        topPanel.Controls.Add(dataGridViewFoundMiners);
        topPanel.Controls.Add(tableControlPanel);

        // Собираем все вместе
        mainSplit.Panel1.Controls.Add(topPanel);
        mainSplit.Panel2.Controls.Add(bottomPanel);
        this.Controls.Add(mainSplit);

        // Добавляем статусную строку внизу формы
        Panel statusPanel = new Panel
        {
            Dock = DockStyle.Bottom,
            Height = 25,
            BackColor = color5
        };

        lblStatusMessage = new Label
        {
            Dock = DockStyle.Fill,
            Text = "",
            ForeColor = Color.White,
            Font = new Font("Arial", 9, FontStyle.Bold),
            TextAlign = ContentAlignment.MiddleCenter
        };

        statusPanel.Controls.Add(lblStatusMessage);
        this.Controls.Add(statusPanel);
    }

    private void
    DataGridViewFoundMiners_CellDoubleClick(object sender,
    DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0)
        {

```

```

var row =
dataGridViewFoundMiners.Rows[e.RowIndex];
    string ip = row.Cells["IP"]?.Value?.ToString();

    if (!string.IsNullOrEmpty(ip))
    {
        OpenWebInterface(ip);
    }
}

private void OpenWebInterface(string ip)
{
    try
    {
        string url = $"http://{ip}";
        Process.Start(new ProcessStartInfo
        {
            FileName = url,
            UseShellExecute = true
        });
        ShowStatusMessage($"🌐 Открыт веб-интерфейс:
{ip}");
    }
    catch (Exception ex)
    {
        ShowStatusMessage($"❌ Ошибка открытия:
{ex.Message}", false);
    }
}

private void ShowStatusMessage(string message, bool
isSuccess = true)
{
    if (lblStatusMessage.InvokeRequired)
    {
        lblStatusMessage.Invoke(new Action(() =>
ShowStatusMessage(message, isSuccess)));
        return;
    }

    lblStatusMessage.Text = message;
    lblStatusMessage.ForeColor = isSuccess ?
Color.LightGreen : Color.LightCoral;
    statusTimer.Start();
}

private void InitializeDataGridView()
{
    foundMinersList = new BindingList<MinerData>();
}

```

```

        dataGridViewFoundMiners.DataSource = foundMinersList;
        dataGridViewFoundMiners.AutoGenerateColumns = false;
        dataGridViewFoundMiners.Columns.Clear();

        string[] columns = { "√", "IP", "Модель", "MAC", "Реал
Н/с", "Идеал Н/с",
                                "Т° ср", "Т° мин-макс", "Время",
        "Статус", "Пулы", "Blink" };
        string[] properties = { "Selected", "IP", "Model",
        "MAC", "HashrateReal", "HashrateIdeal",
                                "TemperatureAvg",
        "TemperatureRange", "Uptime", "Status", "Pools", "BlinkStatus" };
        int[] widths = { 30, 110, 130, 110, 70, 70, 60, 80,
        80, 60, 150, 50 };

        // Колонка выбора
        DataGridViewCheckBoxColumn selectColumn = new
DataGridViewCheckBoxColumn
        {
            HeaderText = columns[0],
            Name = properties[0],
            DataPropertyName = properties[0],
            Width = widths[0]
        };
        dataGridViewFoundMiners.Columns.Add(selectColumn);

        for (int i = 1; i < columns.Length; i++)
        {
            if (properties[i] == "BlinkStatus")
            {
                DataGridViewTextBoxColumn column = new
DataGridViewTextBoxColumn
                {
                    HeaderText = columns[i],
                    Name = properties[i],
                    DataPropertyName = properties[i],
                    Width = widths[i],
                    ReadOnly = true
                };
                dataGridViewFoundMiners.Columns.Add(column);
            }
            else
            {
                DataGridViewTextBoxColumn column = new
DataGridViewTextBoxColumn
                {
                    HeaderText = columns[i],
                    Name = properties[i],
                    DataPropertyName = properties[i],
                    Width = widths[i]
                };
            }
        }

```

```

        dataGridViewFoundMiners.Columns.Add(column);
    }
}

// Добавляем чекбокс "Выбрать все" в заголовок колонки
dataGridViewFoundMiners.CellPainting +=
DataGridViewFoundMiners_CellPainting;
dataGridViewFoundMiners.ColumnHeadersHeightSizeMode =
DataGridViewColumnHeadersHeightSizeMode.EnableResizing;
dataGridViewFoundMiners.ColumnHeadersHeight += 10;

// Обработчик клика по заголовку для выбора всех
dataGridViewFoundMiners.ColumnHeaderMouseClick +=
DataGridViewFoundMiners_ColumnHeaderMouseClick;

dataGridViewFoundMiners.DefaultCellStyle.Font = new
Font("Arial", 8);

dataGridViewFoundMiners.ColumnHeadersDefaultCellStyle.Font = new
Font("Arial", 9, FontStyle.Bold);

dataGridViewFoundMiners.ColumnHeadersDefaultCellStyle.BackColor =
Color.LightGray;
dataGridViewFoundMiners.EnableHeadersVisualStyles =
false;
dataGridViewFoundMiners.RowTemplate.Height = 22;
}

private void
DataGridViewFoundMiners_ColumnHeaderMouseClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == 0) // Первая колонка с чекбоксами
    {
        _isSelectedAllChecked = !_isSelectedAllChecked;
        SelectAllMiners(_isSelectedAllChecked);
        dataGridViewFoundMiners.InvalidateColumn(0); //
Перерисовываем колонку
    }
}

private void DataGridViewFoundMiners_CellPainting(object
sender, DataGridViewCellPaintingEventArgs e)
{
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
    {
        var columnName =
dataGridViewFoundMiners.Columns[e.ColumnIndex].Name;

        // Специальная отрисовка для колонки BlinkStatus
        if (columnName == "BlinkStatus")
        {

```

```

        e.Paint(e.CellBounds,
DataGridViewPaintParts.All
~DataGridViewPaintParts.ContentForeground);

        if (e.RowIndex < foundMinersList.Count)
        {
            var miner = foundMinersList[e.RowIndex];
            bool blinkStatus = miner.BlinkStatus;

            if (blinkStatus)
            {
                bool blinkPhase =
(DateTime.Now.Millisecond / 500) % 2 == 0;
                Color blinkColor = blinkPhase ?
Color.Red : Color.Lime;

                using (SolidBrush brush = new
SolidBrush(blinkColor))
                {
                    int size =
Math.Min(e.CellBounds.Width, e.CellBounds.Height) - 8;
                    Rectangle squareRect = new
Rectangle(
                        e.CellBounds.X
                        +
(e.CellBounds.Width - size) / 2,
                        e.CellBounds.Y
                        +
(e.CellBounds.Height - size) / 2,
                        size, size);

                    e.Graphics.FillRectangle(brush,
squareRect);

                    e.Graphics.DrawRectangle(Pens.Black, squareRect);
                }
            }
            else
            {
                using (SolidBrush brush = new
SolidBrush(Color.Gray))
                {
                    int size =
Math.Min(e.CellBounds.Width, e.CellBounds.Height) - 8;
                    Rectangle squareRect = new
Rectangle(
                        e.CellBounds.X
                        +
(e.CellBounds.Width - size) / 2,
                        e.CellBounds.Y
                        +
(e.CellBounds.Height - size) / 2,
                        size, size);

                    e.Graphics.FillRectangle(brush,
squareRect);

```

```

e.Graphics.DrawRectangle(Pens.Black, squareRect);
    }
}

e.Handled = true;
}
// Отрисовка чекбокса "Выбрать все" в заголовке
else if (e.RowIndex == -1 && e.ColumnIndex == 0)
{
    e.Paint(e.CellBounds,
DataGridViewPaintParts.All                                     &
~DataGridViewPaintParts.ContentForeground);

    // Рисуем чекбокс в заголовке
    CheckBoxRenderer.DrawCheckBox(e.Graphics,
        new Point(e.CellBounds.X
(e.CellBounds.Width - 15) / 2,
        e.CellBounds.Y
(e.CellBounds.Height - 15) / 2),
        _isSelectedAllChecked
System.Windows.Forms.VisualStyles.CheckBoxState.CheckedNormal
:
System.Windows.Forms.VisualStyles.CheckBoxState.UncheckedNormal)
;

    e.Handled = true;
}
}

private void SelectAllMiners(bool select)
{
    foreach (var miner in foundMinersList)
    {
        miner.Selected = select;
    }
    dataGridViewFoundMiners.Refresh();
}

private void UpdateMinerInTable(string ip, MinerData
updatedMiner)
{
    if (dataGridViewFoundMiners.InvokeRequired)
    {
        dataGridViewFoundMiners.Invoke(new Action(() =>
UpdateMinerInTable(ip, updatedMiner)));
        return;
    }
}

```

```

        for (int i = 0; i < foundMinersList.Count; i++)
        {
            if (foundMinersList[i].IP == ip)
            {
                foundMinersList[i] = updatedMiner;
                break;
            }
        }
        dataGridViewFoundMiners.Refresh();
    }

    private void ExportToExcel()
    {
        try
        {
            if (foundMinersList.Count == 0)
            {
                ShowStatusMessage("Нет данных для экспорта",
false);
                return;
            }

            SaveFileDialog saveDialog = new SaveFileDialog
            {
                Filter = "CSV файлы (*.csv)|*.csv|Excel файлы
(*.xlsx)|*.xlsx|Все файлы (*.*)|*.*",
                Title = "Сохранить данные асиков",
                FileName
=
$"Antminer_Data_{DateTime.Now:yyyyMMdd_HH:mm:ss}.csv"
            };

            if (saveDialog.ShowDialog() == DialogResult.OK)
            {
                StringBuilder csv = new StringBuilder();

                // Заголовки
                csv.AppendLine("IP;Модель;MAC;Хешрейт
реал;Хешрейт идеал;Температура средняя;Температура мин-макс;Время
работы;Статус;Пулы;Blink");

                // Данные
                foreach (var miner in foundMinersList)
                {

                    csv.AppendLine($"{{miner.IP}};{{miner.Model}};{{miner.MAC}};{{miner.Has
hrateReal}};{{miner.HashrateIdeal}};{{miner.TemperatureAvg}};{{miner.T
emperatureRange}};{{miner.Uptime}};{{miner.Status}};{{miner.Pools}};{{m
iner.BlinkStatus ? "Да" : "Нет"}}");
                }
            }
        }
    }

```

```

        File.WriteAllText(saveDialog.FileName,
csv.ToString(), Encoding.UTF8);

        ShowStatusMessage($" Данные экспортированы
в {Path.GetFileName(saveDialog.FileName)}");

        // Спрашиваем, открыть ли файл
        if (MessageBox.Show("Открыть файл?", "Экспорт
завершен",
            MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
        {
            Process.Start(new ProcessStartInfo
            {
                FileName = saveDialog.FileName,
                UseShellExecute = true
            });
        }
    }
    catch (Exception ex)
    {
        ShowStatusMessage($" Ошибка экспорта:
{ex.Message}", false);
        MessageBox.Show($"Ошибка при экспорте:
{ex.Message}", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void LoadSavedRanges()
{
    try
    {
        string filePath = "saved_ranges.txt";
        if (File.Exists(filePath))
        {
            var ranges = File.ReadAllLines(filePath);
            foreach (var range in ranges)
            {
                if (!string.IsNullOrWhiteSpace(range))
checkedListRanges.Items.Add(range.Trim());
            }
        }

        if (checkedListRanges.Items.Count == 0)
        {
            string[] defaultRanges = {
                "192.168.96.0-192.168.102.255",
                "192.168.99.0-192.168.99.255",
            }
        }
    }
}

```

```

        "192.168.100.0-192.168.100.255",
        "192.168.101.0-192.168.101.255"
    };
    foreach (var range in defaultRanges)
    {
        checkedListRanges.Items.Add(range);
    }
}
catch (Exception ex)
{
    ShowStatusMessage($"Ошибка загрузки диапазонов:
{ex.Message}", false);
}

private void SaveRanges()
{
    try
    {
        var ranges = new List<string>();
        foreach (var item in checkedListRanges.Items)
        {
            ranges.Add(item.ToString());
        }
        File.WriteAllLines("saved_ranges.txt", ranges);
    }
    catch (Exception ex)
    {
        ShowStatusMessage($"Ошибка сохранения диапазонов:
{ex.Message}", false);
    }
}

private void AddRangeToList()
{
    string startIP = txtStartIP.Text.Trim();
    string endIP = txtEndIP.Text.Trim();

    if (string.IsNullOrEmpty(startIP) ||
string.IsNullOrEmpty(endIP))
    {
        ShowStatusMessage("Введите начальный и конечный
IP адреса", false);
        return;
    }

    if (!IsValidIP(startIP) || !IsValidIP(endIP))
    {
        ShowStatusMessage("Неверный формат IP адреса",
false);
        return;
    }
}

```

```

    }

    string range = $"{startIP}-{endIP}";
    if (!checkedListRanges.Items.Contains(range))
    {
        checkedListRanges.Items.Add(range);
        SaveRanges();
        ShowStatusMessage($"Диапазон {range} добавлен");
        txtStartIP.Text = "";
        txtEndIP.Text = "";
    }
    else
    {
        ShowStatusMessage("Такой диапазон уже
существует", false);
    }
}

private bool IsValidIP(string ip)
{
    string[] parts = ip.Split('.');
    if (parts.Length != 4) return false;

    foreach (string part in parts)
    {
        if (!int.TryParse(part, out int num)) return
false;
        if (num < 0 || num > 255) return false;
    }
    return true;
}

private void RemoveSelectedRange()
{
    if (checkedListRanges.SelectedItem != null)
    {
        var range =
checkedListRanges.SelectedItem.ToString();

checkedListRanges.Items.Remove(checkedListRanges.SelectedItem);
        SaveRanges();
        ShowStatusMessage($"Диапазон {range} удален");
    }
}

private async Task ScanSelectedRanges()
{
    var checkedRanges =
checkedListRanges.CheckedItems.Cast<string>().ToList();
    if (checkedRanges.Count == 0)
    {

```

```

        ShowStatusMessage("Выберите диапазоны для сканирования", false);
        return;
    }

```

```

        ShowStatusMessage($"🔍 Запуск сканирования {checkedRanges.Count} диапазонов...");
        btnScanSelectedRanges.Enabled = false;
        progressScan.Value = 0;
        foundMinersList.Clear();
        _clients.Clear();

```

```

try
{
    var allIPs = new List<string>();
    foreach (var range in checkedRanges)
    {
        var ips = ParseIPRange(range);
        allIPs.AddRange(ips);
    }

    int totalIPs = allIPs.Count;
    int scannedCount = 0;
    int foundCount = 0;

    progressScan.Maximum = totalIPs;

    var tasks = new List<Task>();

    foreach (var ip in allIPs)
    {
        tasks.Add(Task.Run(async () =>
        {
            try
            {
                if (await PingHost(ip, 300))
                {
                    if (await CheckIfAntminer(ip))
                    {
                        try
                        {
                            var client = new
                                AntminerClient(ip, "root", "root");

                            lock (_clients)
                            {
                                _clients[ip] =
                                    client;

                                var minerInfo = await
                                    GetMinerInfo(ip, client);

```

```

=>
        if (minerInfo != null)
        {
            Invoke(new Action(()
            {
                foundMinersList.Add(minerInfo);
                foundCount++;
            }));
        }
        catch (Exception ex) { }
    }
}
catch { }
finally
{
    scannedCount++;
    Invoke(new Action(() =>
    {
        progressScan.Value =
scannedCount;
        lblScanStatus.Text = $"Скан:
{scannedCount}/{totalIPs}, Найдено: {foundCount}";
    }));
}
}
if (tasks.Count >= 30)
{
    await Task.WhenAll(tasks);
    tasks.Clear();
}

await Task.WhenAll(tasks);

ShowStatusMessage($"✅ Сканирование завершено.
Найдено: {foundCount}");
lblScanStatus.Text = $"Сканирование завершено.
Найдено: {foundCount}";
}
catch (Exception ex)
{
    ShowStatusMessage($"❌ Ошибка сканирования:
{ex.Message}", false);
    lblScanStatus.Text = "Ошибка сканирования";
}
finally
{

```

```

        btnScanSelectedRanges.Enabled = true;
    }
}

private List<string> ParseIPRange(string range)
{
    var result = new List<string>();

    try
    {
        string[] parts = range.Split('-');
        if (parts.Length == 2)
        {
            string startIP = parts[0].Trim();
            string endIP = parts[1].Trim();

            string[] startParts = startIP.Split('.');
            if (startParts.Length == 4)
            {
                string baseStartIP =
                $"{startParts[0]}.{startParts[1]}.{startParts[2]}.0";
                int start = int.Parse(startParts[3]);

                string[] endParts = endIP.Split('.');
                if (endParts.Length == 4)
                {
                    string baseEndIP =
                    $"{endParts[0]}.{endParts[1]}.{endParts[2]}.0";
                    int end = int.Parse(endParts[3]);

                    if (baseStartIP != baseEndIP)
                    {
                        for (int i = start; i <= 255; i++)
                        {
                            result.Add($"{baseStartIP}{i}");
                        }
                    }

                    string[] startBaseParts =
                    baseStartIP.Split('.');
                    string[] endBaseParts =
                    baseEndIP.Split('.');

                    int startThird =
                    int.Parse(startBaseParts[2]);
                    int endThird =
                    int.Parse(endBaseParts[2]);

                    for (int third = startThird + 1;
                    third < endThird; third++)
                    {
                        for (int i = 0; i <= 255; i++)

```

```

        {
result.Add($"{startBaseParts[0]}.{startBaseParts[1]}.{third}.{i}");
        }
    }

    for (int i = 0; i <= end; i++)
    {
result.Add($"{baseEndIP}{i}");
    }
    }
    else
    {
        for (int i = start; i <= end; i++)
        {
result.Add($"{baseStartIP}{i}");
        }
    }
    }
    }
    }
    }
    catch (Exception ex)
    {
        ShowStatusMessage($"Ошибка парсинга диапазона: {ex.Message}", false);
        return new List<string>();
    }

    return result;
}

private async Task<bool> PingHost(string ip, int timeout)
{
    try
    {
        using (var ping = new Ping())
        {
            var reply = await ping.SendPingAsync(ip, timeout);

            return reply.Status == IPStatus.Success;
        }
    }
    catch
    {
        return false;
    }
}

```

```

private async Task<bool> CheckIfAntminer(string ip)
{
    try
    {
        var request =
        (HttpWebRequest)WebRequest.Create($"http://{ip}/cgi-
        bin/get_system_info.cgi");
        request.Timeout = 1000;
        request.Method = "HEAD";

        using (var response = await
        request.GetResponseAsync() as HttpWebResponse)
        {
            return response != null && response.StatusCode
            == HttpStatusCode.Unauthorized;
        }
    }
    catch (WebException ex) when (ex.Response is
    HttpWebResponse response)
    {
        return response.StatusCode ==
        HttpStatusCode.Unauthorized;
    }
    catch
    {
        return false;
    }
}

private async Task<MinerData> GetMinerInfo(string ip,
AntminerClient client)
{
    try
    {
        var systemInfo = await client.GetSystemInfo();
        var stats = await client.GetStatistics();

        string hashrateReal = "N/A";
        string hashrateIdeal = "N/A";
        string temperatureAvg = "N/A";
        string temperatureRange = "N/A";
        string uptime = "N/A";
        string model = "Unknown";
        string mac = "Unknown";

        using JsonDocument doc =
        JsonDocument.Parse(systemInfo);
        var root = doc.RootElement;

        model = root.TryGetProperty("minertype", out var
        modelProp) ? modelProp.GetString() : "Unknown";
    }
}

```

```

        if (root.TryGetProperty("macaddr", out var
macProp))
        {
            mac = macProp.GetString();
        }

        (hashrateReal, hashrateIdeal, temperatureAvg,
temperatureRange, uptime) = ParseStatistics(stats);

        string poolsInfo = "";
        bool blinkStatus = false;

        try
        {
            var config = await client.GetMinerConfig();
            using JsonDocument configDoc =
JsonDocument.Parse(config);
            if
            (configDoc.RootElement.TryGetProperty("pools", out var pools) &&
pools.ValueKind == JsonValueKind.Array)
            {
                var poolList = new List<string>();
                for (int i = 0; i < pools.GetArrayLength()
&& i < 3; i++)
                {
                    var pool = pools[i];
                    if (pool.TryGetProperty("url", out
var url))
                    {
                        string urlStr = url.GetString();
                        if (urlStr.Contains("://"))
                        {
                            urlStr =
urlStr.Substring(urlStr.IndexOf("://") + 3);
                            poolList.Add(urlStr);
                        }
                    }
                }
                poolsInfo = string.Join(", ", poolList);
            }
        }
        catch { }

        try
        {
            var blinkResult = await
client.GetBlinkStatus();
            using JsonDocument blinkDoc =
JsonDocument.Parse(blinkResult);
            if
            (blinkDoc.RootElement.TryGetProperty("blink", out var blink))
            {

```

```

        blinkStatus = blink.GetBoolean();
    }
}
catch { }

return new MinerData
{
    IP = ip,
    Model = model,
    MAC = mac,
    HashrateReal = hashrateReal,
    HashrateIdeal = hashrateIdeal,
    TemperatureAvg = temperatureAvg,
    TemperatureRange = temperatureRange,
    Uptime = uptime,
    Status = "✅",
    Pools = poolsInfo,
    BlinkStatus = blinkStatus
};
}
catch (Exception ex)
{
    return new MinerData
    {
        IP = ip,
        Model = "Antminer",
        MAC = "Unknown",
        HashrateReal = "N/A",
        HashrateIdeal = "N/A",
        TemperatureAvg = "N/A",
        TemperatureRange = "N/A",
        Uptime = "N/A",
        Status = "✅",
        Pools = "",
        BlinkStatus = false
    };
}

private (string real, string ideal, string avgTemp, string
rangeTemp, string uptime) ParseStatistics(string statsJson)
{
    try
    {
        using JsonDocument doc =
JsonDocument.Parse(statsJson);
        var root = doc.RootElement;

        double totalRateReal = 0;
        double totalRateIdeal = 0;
        var allTemperatures = new List<int>();

```

```

        long elapsed = 0;

        if (root.TryGetProperty("STATS", out var statsArray) && statsArray.ValueKind == JsonValueKind.Array)
        {
            foreach (var statsItem in statsArray.EnumerateArray())
            {
                if (statsItem.TryGetProperty("elapsed", out var elapsedProp))
                {
                    elapsed = elapsedProp.GetInt64();
                }

                if (statsItem.TryGetProperty("chain", out var chainArray) && chainArray.ValueKind == JsonValueKind.Array)
                {
                    foreach (var chain in chainArray.EnumerateArray())
                    {
                        if (chain.TryGetProperty("rate_real", out var rateReal))
                        {
                            totalRateReal += rateReal.GetDouble();
                        }
                        if (chain.TryGetProperty("rate_ideal", out var rateIdeal))
                        {
                            totalRateIdeal += rateIdeal.GetDouble();
                        }

                        if (chain.TryGetProperty("temp_chip", out var tempChip) && tempChip.ValueKind == JsonValueKind.Array)
                        {
                            foreach (var temp in tempChip.EnumerateArray())
                            {
                                allTemperatures.Add(temp.GetInt32());
                            }
                        }
                    }
                }
            }

            string hashrateRealStr = totalRateReal > 0 ?
            $"{totalRateReal:F0}" : "N/A";

```

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		92

```

        string hashrateIdealStr = totalRateIdeal > 0 ?
        $"{totalRateIdeal:F0}" : "N/A";

        string temperatureAvgStr = "N/A";
        string temperatureRangeStr = "N/A";

        if (allTemperatures.Count > 0)
        {
            int minTemp = allTemperatures.Min();
            int maxTemp = allTemperatures.Max();
            int avgTemp = (int)allTemperatures.Average();
            temperatureAvgStr = $"{avgTemp}°C";
            temperatureRangeStr = $"{minTemp}-{
{maxTemp}°C";
        }

        string uptimeStr = FormatUptime(elapsed);

        return (hashrateRealStr, hashrateIdealStr,
temperatureAvgStr, temperatureRangeStr, uptimeStr);
    }
    catch
    {
        return ("N/A", "N/A", "N/A", "N/A", "N/A");
    }
}

private string FormatUptime(long seconds)
{
    if (seconds <= 0) return "N/A";

    TimeSpan time = TimeSpan.FromSeconds(seconds);

    if (time.TotalDays >= 1)
        return $"{time.Days}д {time.Hours}ч";
    else if (time.TotalHours >= 1)
        return $"{time.Hours}ч {time.Minutes}м";
    else if (time.TotalMinutes >= 1)
        return $"{time.Minutes}м {time.Seconds}с";
    else
        return $"{time.Seconds}с";
}

private List<string> GetSelectedMinersIPs()
{
    var selectedIPs = new List<string>();
    foreach (DataGridViewRow row in
dataGridViewFoundMiners.Rows)
    {
        if (row.Cells["Selected"]?.Value is bool
isSelected && isSelected)
        {

```

```

        var ip = row.Cells["IP"]?.Value?.ToString();
        if (!string.IsNullOrEmpty(ip) &&
        _clients.ContainsKey(ip))
        {
            selectedIPs.Add(ip);
        }
    }
    return selectedIPs;
}

private async Task LoadPoolsFromSelected()
{
    var selectedIPs = GetSelectedMinersIPs();
    if (selectedIPs.Count == 0)
    {
        ShowStatusMessage("Выберите асики из таблицы",
false);
        return;
    }

    var firstIP = selectedIPs[0];

    try
    {
        var result = await
        _clients[firstIP].GetMinerConfig();

        if (!string.IsNullOrEmpty(result))
        {
            using JsonDocument doc =
JsonDocument.Parse(result);
            var root = doc.RootElement;

            if (root.TryGetProperty("pools", out var
pools) && pools.ValueKind == JsonValueKind.Array)
            {
                for (int i = 0; i < pools.GetArrayLength()
&& i < 3; i++)
                {
                    var pool = pools[i];

                    if (pool.TryGetProperty("url", out var url))
                        txtPoolUrls[i].Text =
url.GetString();

                    if (pool.TryGetProperty("user", out
var poolUser))
                        txtPoolUsers[i].Text =
poolUser.GetString();

```

```

        if (pool.TryGetProperty("pass", out
var poolPass))
        txtPoolPasswords[i].Text =
poolPass.GetString();
    }

    ShowStatusMessage($"✅ Пулы загружены с
асика {firstIP}");
    }
}
catch (Exception ex)
{
    ShowStatusMessage($"❌ Ошибка загрузки пулов:
{ex.Message}", false);
}

private async Task SetPoolsForSelected()
{
    var selectedIPs = GetSelectedMinersIPs();
    if (selectedIPs.Count == 0)
    {
        ShowStatusMessage("Выберите асики из таблицы",
false);
        return;
    }

    var newPools = new List<Dictionary<string, string>>();
    for (int i = 0; i < 3; i++)
    {
        string poolUrl = txtPoolUrls[i].Text.Trim();
        string poolUser = txtPoolUsers[i].Text.Trim();
        string poolPass =
txtPoolPasswords[i].Text.Trim();

        if (!string.IsNullOrEmpty(poolUrl))
        {
            var pool = new Dictionary<string, string>
            {
                ["url"] = poolUrl,
                ["user"] = poolUser,
                ["pass"] = poolPass
            };
            newPools.Add(pool);
        }
    }

    if (newPools.Count == 0)
    {

```

```

        ShowStatusMessage("✗ Не указаны пулы для
настройки", false);
        return;
    }

    ShowStatusMessage($"💾 Установка пулов для
{selectedIPs.Count} асиков...");

    int successCount = 0;
    foreach (var ip in selectedIPs)
    {
        try
        {
            var currentConfig = await
_clients[ip].GetMinerConfig();
            using JsonDocument doc =
JsonDocument.Parse(currentConfig);
            var configDict = new Dictionary<string,
object>();

            foreach (var prop in
doc.RootElement.EnumerateObject())
            {
                switch (prop.Value.ValueKind)
                {
                    case JsonValueKind.String:
                        configDict[prop.Name] =
prop.Value.GetString();
                        break;
                    case JsonValueKind.Number:
                        configDict[prop.Name] =
prop.Value.GetDouble();
                        break;
                    case JsonValueKind.True:
                    case JsonValueKind.False:
                        configDict[prop.Name] =
prop.Value.GetBoolean();
                        break;
                    case JsonValueKind.Array:
                        if (prop.Name == "pools")
                        {
                            configDict["pools"] =
newPools;
                        }
                        else
                        {
                            configDict[prop.Name] =
JsonSerializer.Deserialize<object[]>(prop.Value.GetRawText());
                        }
                        break;
                    case JsonValueKind.Object:

```

```

                                configDict[prop.Name]           =
JsonSerializer.Deserialize<Dictionary<string,
object>>(prop.Value.GetRawText());
                                break;
                                default:
                                configDict[prop.Name]           =
prop.Value.ToString();
                                break;
                                }
                                }

                                string                newConfigJson           =
JsonSerializer.Serialize(configDict, new JsonSerializerOptions
{
                                WriteIndented = false
                                });

                                await
_clients[ip].SetMinerConfig(newConfigJson);
                                successCount++;
                                }
                                catch (Exception ex)
                                {
                                if (ex.Message.Contains("timeout") ||
ex.Message.Contains("timed out"))
                                {
                                successCount++;
                                }
                                }

                                await Task.Delay(500);
                                }

                                ShowStatusMessage($"✅ Пулы установлены для
{successCount} из {selectedIPs.Count} асиков");
                                }

                                private async Task ToggleBlinkForSelected()
                                {
                                var selectedIPs = GetSelectedMinersIPs();
                                if (selectedIPs.Count == 0)
                                {
                                ShowStatusMessage("Выберите асики из таблицы",
false);
                                return;
                                }

                                ShowStatusMessage($"💡 Переключение blink для
{selectedIPs.Count} асиков...");

                                foreach (var ip in selectedIPs)

```

```

        {
            try
            {
                bool currentBlinkStatus = false;
                try
                {
                    var blinkStatus = await
_clients[ip].GetBlinkStatus();
                    using JsonDocument doc =
JsonDocument.Parse(blinkStatus);
                    if
(doc.RootElement.TryGetProperty("blink", out var blink))
                    {
                        currentBlinkStatus =
blink.GetBoolean();
                    }
                }
                catch { }

                string body = currentBlinkStatus ?
"{\"blink\":false}" : "{\"blink\":true}";
                await _clients[ip].ExecuteRequest("/cgi-
bin/blink.cgi", "POST", body, "text/plain");

                UpdateBlinkStatusInTable(ip,
!currentBlinkStatus);
            }
            catch (Exception ex) { }

            await Task.Delay(100);
        }

        ShowStatusMessage($" Blink переключен для
{selectedIPs.Count} асиков");
    }

    private void UpdateBlinkStatusInTable(string ip, bool
newStatus)
    {
        if (dataGridViewFoundMiners.InvokeRequired)
        {
            dataGridViewFoundMiners.Invoke(new Action(() =>
UpdateBlinkStatusInTable(ip, newStatus)));
            return;
        }

        for (int i = 0; i < foundMinersList.Count; i++)
        {
            if (foundMinersList[i].IP == ip)
            {
                foundMinersList[i].BlinkStatus = newStatus;
            }
        }
    }

```

```

        break;
    }
}
dataGridViewFoundMiners.Refresh();
}

private async Task RebootSelected()
{
    var selectedIPs = GetSelectedMinersIPs();
    if (selectedIPs.Count == 0)
    {
        ShowStatusMessage("Выберите асики из таблицы",
false);
        return;
    }

    if (MessageBox.Show($"Перезагрузить
{selectedIPs.Count} асик(ов)?", "Подтверждение",
        MessageBoxButtons.YesNo, MessageBoxIcon.Warning)
!= DialogResult.Yes)
    {
        return;
    }

    ShowStatusMessage($"⚡                                     Перезагрузка
{selectedIPs.Count} асиков...");

    foreach (var ip in selectedIPs)
    {
        try
        {
            await _clients[ip].Reboot();
            UpdateStatusInTable(ip, "🔄");
        }
        catch (Exception ex) { }

        await Task.Delay(100);
    }

    ShowStatusMessage($"✅ Команды перезагрузки
отправлены для {selectedIPs.Count} асиков");
}

private void UpdateStatusInTable(string ip, string
newStatus)
{
    if (dataGridViewFoundMiners.InvokeRequired)
    {
        dataGridViewFoundMiners.Invoke(new Action(() =>
UpdateStatusInTable(ip, newStatus)));
        return;
    }
}

```

```

    }

    for (int i = 0; i < foundMinersList.Count; i++)
    {
        if (foundMinersList[i].IP == ip)
        {
            foundMinersList[i].Status = newStatus;
            break;
        }
    }
    dataGridViewFoundMiners.Refresh();
}

private void ClearPools()
{
    for (int i = 0; i < 3; i++)
    {
        txtPoolUrls[i].Text = "";
        txtPoolUsers[i].Text = "";
        txtPoolPasswords[i].Text = "";
    }
    ShowStatusMessage("✅ Поля пулов очищены");
}

public class DigestAuth
{
    private readonly string _username;
    private readonly string _password;
    private string _realm;
    private string _nonce;
    private int _nc;
    private readonly Random _random;

    public DigestAuth(string username, string password)
    {
        _username = username;
        _password = password;
        _nc = 1;
        _random = new Random();
    }

    public void SetChallenge(string realm, string nonce)
    {
        _realm = realm;
        _nonce = nonce;
    }

    public string GenerateAuthHeader(string method,
string uri)
    {
        string cnonce = GenerateCnonce();

```

```

        string ha1 =
ComputeMD5($"{_username}:{_realm}:{_password}");
        string ha2 = ComputeMD5($"{method}:{uri}");
        string response =
ComputeMD5($"{ha1}:{_nonce}:{_nc:X8}:{cnonce}:auth:{ha2}");

        string authHeader = $"Digest
username=\"{_username}\", " +
        $"realm=\"{_realm}\", " +
        $"nonce=\"{_nonce}\", " +
        $"uri=\"{uri}\", " +
        $"response=\"{response}\", " +
        $"qop=auth, " +
        $"nc={_nc:X8}, " +
        $"cnonce=\"{cnonce}\"";

        _nc++;
        return authHeader;
    }

    private string GenerateCnonce()
    {
        byte[] buffer = new byte[8];
        _random.NextBytes(buffer);
        return BitConverter.ToString(buffer).Replace("-",
"").ToLower();
    }

    private string ComputeMD5(string input)
    {
        using (MD5 md5 = MD5.Create())
        {
            byte[] inputBytes =
Encoding.UTF8.GetBytes(input);
            byte[] hashBytes =
md5.ComputeHash(inputBytes);

            StringBuilder sb = new StringBuilder();
            foreach (byte b in hashBytes)
            {
                sb.Append(b.ToString("x2"));
            }
            return sb.ToString();
        }
    }

    public class AntminerClient
    {
        private readonly string _ipAddress;
        private readonly DigestAuth _digestAuth;
        private CookieContainer _cookies;

```

```

        private bool _hasChallenge = false;

        public AntminerClient(string ipAddress, string
username, string password)
        {
            _ipAddress = ipAddress;
            _digestAuth = new DigestAuth(username, password);
            _cookies = new CookieContainer();
        }

        public async Task<bool> GetChallenge()
        {
            try
            {
                string url = $"http://{_ipAddress}/cgi-
bin/get_system_info.cgi";
                var request =
(HttpWebRequest)WebRequest.Create(url);
                request.Method = "GET";
                request.Timeout = 3000;
                request.Credentials = null;

                try
                {
                    using (var response =
(HttpWebResponse)await request.GetResponseAsync())
                    {
                        return true;
                    }
                }
                catch (WebException ex) when (ex.Response is
HttpWebResponse response)
                {
                    if (response.StatusCode ==
HttpStatusCode.Unauthorized)
                    {
                        string authHeader =
response.Headers["WWW-Authenticate"];
                        if (!string.IsNullOrEmpty(authHeader)
&& authHeader.StartsWith("Digest"))
                        {
                            ParseAuthHeader(authHeader);
                            _hasChallenge = true;
                            return true;
                        }
                    }
                    return false;
                }
            }
            catch (Exception)
            {
                return false;
            }
        }

```

					ПГУ1.090301.21.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		102

```

    }
}

private void ParseAuthHeader(string authHeader)
{
    authHeader = authHeader.Substring(7);
    var parameters = new Dictionary<string, string>();
    var parts = authHeader.Split(',');

    foreach (var part in parts)
    {
        var trimmed = part.Trim();
        var keyValue = trimmed.Split(new[] { '=' },
2);

        if (keyValue.Length == 2)
        {
            string key = keyValue[0].Trim();
            string value =
keyValue[1].Trim().Trim('"');
            parameters[key] = value;
        }
    }

    if (parameters.TryGetValue("realm", out var
realm) &&
parameters.TryGetValue("nonce", out var
nonce))
    {
        _digestAuth.SetChallenge(realm, nonce);
    }
}

public async Task<string> ExecuteRequest(string
endpoint, string method = "GET",
string body = null, string contentType =
"application/json", int timeout = 10000)
{
    if (!_hasChallenge)
    {
        bool gotChallenge = await GetChallenge();
        if (!gotChallenge)
        {
            throw new Exception("Не удалось получить
challenge с сервера");
        }
    }

    string url = $"http://{_ipAddress}{endpoint}";
    var request =
(HttpWebRequest)WebRequest.Create(url);
    request.Method = method;
    request.Timeout = timeout;

```

```

        request.CookieContainer = _cookies;
        request.UserAgent = "Mozilla/5.0 (Antminer-
Controller/1.0)";
        request.Accept = "application/json,
text/javascript, */*; q=0.01";
        request.Headers["X-Requested-With"] =
"XMLHttpRequest";
        request.Referer = $"http://{_ipAddress}/";

        string authHeader =
_digestAuth.GenerateAuthHeader(method, endpoint);
        request.Headers["Authorization"] = authHeader;

        if (method == "POST" &&
!string.IsNullOrEmpty(body))
        {
            request.ContentType = contentType;
            byte[] data = Encoding.UTF8.GetBytes(body);
            request.ContentLength = data.Length;

            using (var stream = await
request.GetRequestStreamAsync())
            {
                await stream.WriteAsync(data, 0,
data.Length);
            }
        }

        try
        {
            using (var response = (HttpWebResponse)await
request.GetResponseAsync())
            using (var stream =
response.GetResponseStream())
            using (var reader = new StreamReader(stream))
            {
                if (response.Cookies != null)
                {
                    foreach (Cookie cookie in
response.Cookies)
                    {
                        _cookies.Add(cookie);
                    }
                }

                return await reader.ReadToEndAsync();
            }
        }
        catch (WebException ex) when (ex.Response is
HttpWebResponse response)
        {

```

```

        using (var stream =
response.GetResponseStream())
        using (var reader = new StreamReader(stream))
        {
            string error = await
reader.ReadToEndAsync();
            throw new Exception($"HTTP
{response.StatusCode}: {error}");
        }
    }
    catch (WebException ex) when (ex.Status ==
WebExceptionStatus.Timeout)
    {
        throw new Exception($"Timeout при выполнении
запроса: {ex.Message}");
    }
}

public async Task<string> GetSystemInfo()
{
    return await ExecuteRequest("/cgi-
bin/get_system_info.cgi");
}

public async Task<string> GetMinerConfig()
{
    return await ExecuteRequest("/cgi-
bin/get_miner_conf.cgi");
}

public async Task<string> GetStatistics()
{
    return await ExecuteRequest("/cgi-
bin/stats.cgi");
}

public async Task<string> SetMinerConfig(string
jsonConfig)
{
    try
    {
        return await ExecuteRequest("/cgi-
bin/set_miner_conf.cgi", "POST", jsonConfig, "application/json",
20000);
    }
    catch (Exception ex)
    {
        if (ex.Message.Contains("timeout",
StringComparison.OrdinalIgnoreCase) ||
ex.Message.Contains("timed out",
StringComparison.OrdinalIgnoreCase))
        {

```

```

        return "success";
    }
    throw;
}

public async Task<string> GetBlinkStatus()
{
    return await ExecuteRequest("/cgi-
bin/get_blink_status.cgi");
}

public async Task<string> Reboot()
{
    string body = "{\"action\":\"reboot\"}";
    return await ExecuteRequest("/cgi-
bin/reboot.cgi", "POST", body, "text/plain");
}

[STAThread]
static void Main()
{
    Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new MainForm());
}
}

```

Приложение В

«Программа для управления майнинг-устройствами в локальной сети»

Презентация

					ПГУ1.090301.21.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ПРОГРАММА УПРАВЛЕНИЯ МАЙНИНГ-УСТРОЙСТВАМИ В ЛОКАЛЬНОЙ СЕТИ

АВТОР РАБОТЫ: СТУДЕНТ ГР.22ВВП1 ПЯТКИН М.Э.

НАУЧНЫЙ РУКОВОДИТЕЛЬ: ДОЦЕНТ, К.П.Н. ЛЕОНОВА Т.Ю.

О программе

Проблема

Создание программного инструмента для централизованного управления и мониторинга оборудования майнинг-центра, позволяющего автоматизировать рутинные процессы подготовки отчетности и настройки устройств.

Решение

Одним из путей достижения поставленной задачи является разработка программы, обеспечивающая выполнение следующих функций: возможность работы с большим количеством устройств, простота в эксплуатации, стабильность работы системы, возможность изменения характеристик нескольких устройств, возможность дополнения системы новыми функциями.

Цели и задачи

Цель работы – разработать программу для управления майнинг-устройствами в локальной сети.

Задачи:

- сканирование сети;
- аутентификация на оборудовании;
- автоматический сбор данных;
- отображение собранных данных в табличном виде ;
- управление группой устройств;
- загрузка и установка конфигурации пулов;
- экспорт данных в табличный формат;
- интуитивно понятный интерфейс.

Технологический стек разработки

Язык разработки:

C#&.NET



Среда разработки:

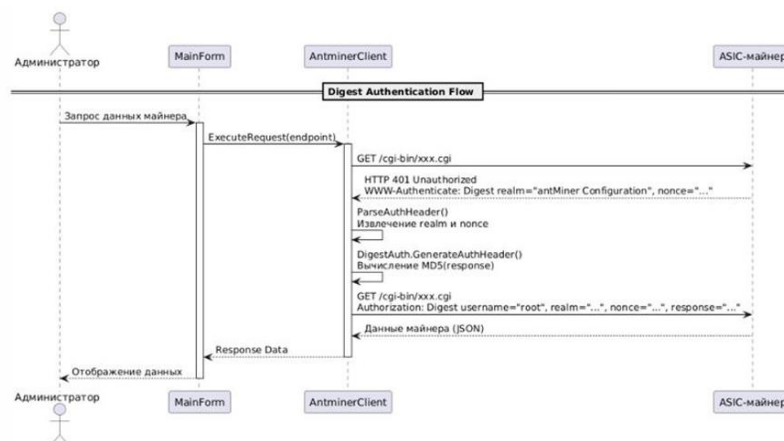
Visual Studio 2022



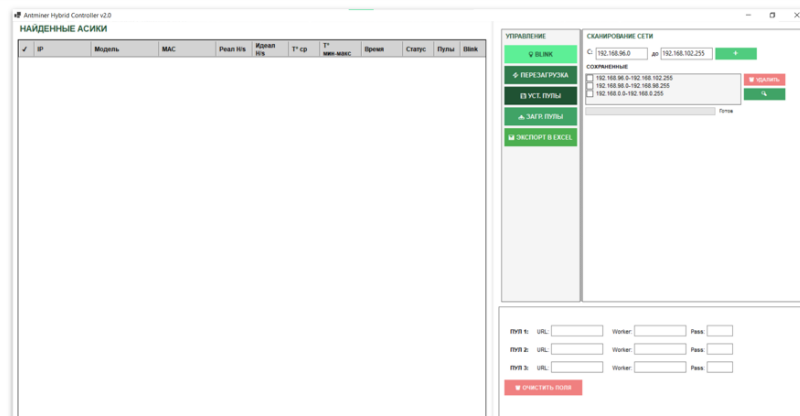
Варианты использования



Диаграмма последовательностей



Интерфейс приложения



Заключение

Во время выполнения данной работы были разработаны: программа управления майнинг-устройствами в локальной сети, настроен пользовательский интерфейс. В дальнейшем проект может быть расширен путем добавления нового функционала и новой логики взаимодействия со сторонними устройствами.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.21.001 ПЗ

Лист

111