**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**

В современном мире сети связи играют важную роль во многих областях, от бизнеса до личного пользования. В связи с этим, обеспечение надежности и эффективности сетей связи является одним из главных приоритетов для технических специалистов. Для достижения этой цели необходимо постоянно осуществлять мониторинг и анализировать работу сетей связи. Для эффективного мониторинга и анализа используются различные системы мониторинга, которые позволяют оперативно обнаруживать и устранять проблемы в работе сетей связи.

Рынок мониторинговых систем для сетей связи постоянно растет, появляются все новые и новые продукты. Так, на 2022 год рынок систем мониторинга США оценивается в 2,2 миллиарда долларов, а по прогнозам аналитического агентства Markets and Markets к 2027 году эта сумма достигнет 3 миллиардов долларов [1]. Тем не менее, при разработке новых продуктов внимание чаще уделяется расширению функционала и гибким настройкам мониторинга, позволяющим осуществлять полномасштабное наблюдение за большими и сложными сетями. Однако, в России в последние годы наблюдается тенденция к развитию малого и среднего бизнеса. Так, по данным Министерства экономики Российской Федерации доля малого бизнеса в ВВП выросла больше, чем на 10 процентов с 2017 года. Предприятия, работающие на государственной поддержке, такие, как школы и поликлиники, а также небольшие компании очень часто не имеют возможностей нанять квалифицированного системного администратора, который будет способен разобраться в сложных настройках системы мониторинга. Таким образом, актуальность разработки простой в использовании системы для мониторинга, которая не будет требовать первичной настройки для полноценной работы, сложно переоценить. Введение такой системы в эксплуатацию позволит сократить затраты на содержание отдела системного администрирования.

Целью данной работы является моделирование и разработка клиентского приложения для системного администратора, являющегося частью программного комплекса удаленного мониторинга и технической поддержки «InBetween».

Для достижения этой цели необходимо решить следующие задачи:

1. Изучить техническую литературу по теме исследования, и систематизировать полученную информацию
2. Проанализировать доступные на рынке системы мониторинга
3. Составить техническое задание, определить основной функционал программного обеспечения
4. Разработать логическую модель системы
5. Изучить техническую документацию Python и PyQT6
6. Реализовать сетевое взаимодействие всех компонентов системы мониторинга «InBetween»
7. Реализовать API-клиент для взаимодействия программы с базой данных
8. Разработать функции для визуализации изменения параметров компьютеров во времени
9. Разработать графический интерфейс системы
10. Провести ручное тестирование итоговой программы и внести необходимые изменения

**1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

* 1. **История развития систем мониторинга**

Первая компьютерная сеть, способная объединять в себе устройства, находящиеся в относительной удаленности друг от друга была создана в 1969 году Агентством Министерства обороны США. Она получила название ARPANET и стала прототипом сети Интернет, использующейся в настоящее время. Системы мониторинга сетевой инфраструктуры начали появляться практически сразу же, как эта технология начала получать распространение.

Первые инструменты для контроля сети появились в 1980-тых годах. В то время системы мониторинга сетей были примитивными и управлялись через командную строку. Они предоставляли возможности для проверки доступности устройств и сервисов в сети. Наиболее популярными инструментами мониторинга сетей в те времена были SNMP (Simple Network Management Protocol) и ICMP (Internet Control Message Protocol). ICMP был разработан в 1981 году и предназначен для проверки доступности устройств в сети. Он позволял отправлять пакеты данных на устройства в сети и получать ответы от этих устройств для проверки доступности. SNMP был создан в 1988 году и стал первым протоколом управления сетями для мониторинга состояния устройств и сервисов в сети. Он позволял сетевым администраторам получать информацию о состоянии сетевых устройств, таких как маршрутизаторы, коммутаторы, серверы и другие устройства, и анализировать эту информацию для обнаружения проблем.

Эти инструменты мониторинга сетей в 1980-х годах были довольно ограниченными, но они создали основу для развития более продвинутых систем мониторинга в последующие десятилетия.

В 1990-х годах системы мониторинга сетей продолжали развиваться, и появились новые инструменты и технологии, которые расширили возможности мониторинга и управления сетью. Одним из ключевых инструментов мониторинга сетей в 1990-х годах был протокол RMON (Remote Network Monitoring). RMON был разработан как расширение протокола SNMP и позволял собирать дополнительную информацию о состоянии сети, такую как данные о трафике, использовании пропускной способности и другие параметры. RMON был очень популярен в 1990-х годах и до сих пор используется в системах мониторинга сетей. В 1990-х годах также появились первые системы мониторинга сетей с графическим интерфейсом. Эти системы предоставляли удобный способ отображения и анализа данных о состоянии сети, что значительно упрощало работу с ними. Кроме того, в это время стали появляться специализированные системы мониторинга сетей, которые были нацелены на определенные сферы применения. Например, системы мониторинга сетей для мониторинга трафика в сетях передачи данных (WAN), системы мониторинга сетей для мониторинга производительности виртуальных частных сетей (VPN) и т.д.

В 2000-х годах произошел значительный рост в области сетевых технологий, что привело к увеличению сложности и размеров сетей. Было введено множество новых протоколов, в том числе IPv6, MPLS, VoIP и другие. Это привело к необходимости создания более сложных систем мониторинга, которые бы могли отслеживать эти новые протоколы и технологии. Одним из ключевых трендов этого десятилетия было появление систем мониторинга сетей с функцией автоматического обнаружения устройств и топологии сети. Это позволило администраторам сети автоматически находить новые устройства, отслеживать их и связи между ними, а также уведомлять об изменениях в топологии. Другой важной тенденцией стало увеличение возможностей мониторинга производительности приложений. Разработчики систем мониторинга начали предоставлять более точные и подробные отчеты о производительности приложений, которые могут помочь выявить проблемы и улучшить качество работы приложений. В это время также начали активно использоваться облачные технологии, что привело к появлению новых требований к системам мониторинга. Были созданы системы мониторинга, способные отслеживать производительность облачных ресурсов и обнаруживать проблемы в облачных средах. Кроме того, в этот период стали появляться более сложные системы мониторинга, которые могут анализировать данные многих источников, включая логи, метрики и трассировку сетевых пакетов. Это помогло снизить время реакции на проблемы в сети и улучшить производительность и безопасность сети.

В 2010-х годах системы мониторинга сетей стали еще более распространенными и продвинутыми. Вместе с развитием облачных технологий, многие системы мониторинга начали переходить на облачные платформы. Кроме того, расширение возможностей сетевого оборудования и протоколов также повлияло на развитие систем мониторинга сетей. Одной из самых популярных систем мониторинга сетей в 2010-х годах стал Nagios, которая по-прежнему является одной из наиболее используемых систем мониторинга в настоящее время. Nagios была выпущена в 1999 году, но в 2010-х годах получила новый толчок развития и стала более гибкой и функциональной. Кроме Nagios, в 2010-х годах появилось множество других систем мониторинга сетей, таких как Zabbix, Cacti, Zenoss и другие. Они предоставляют различные функции, от простого мониторинга до более продвинутых возможностей, таких как анализ данных и прогнозирование. Также в 2010-х годах появилось множество облачных систем мониторинга сетей, таких как Amazon CloudWatch, Microsoft Azure Monitor и Google Cloud Monitoring. Эти системы предоставляют возможности мониторинга облачных сервисов и инфраструктуры, а также могут использоваться для мониторинга локальных сетей и приложений.

Сегодня системы мониторинга сетей имеют широкий спектр функций и возможностей, включая отслеживание доступности и производительности устройств и сервисов, анализ потока данных, управление событиями, оповещение об авариях и многое другое. Эти инструменты являются критически важными для поддержания надежности и производительности сетей в современном мире информационных технологий.

* 1. **Принципы работы систем мониторинга**

Мониторинг компьютерной сети — это процесс постоянного отслеживания компьютерной сети на наличие медленных или неисправных компонентов, проверка состояния метрик, в том числе метрик качества предоставления сервиса [2]. Соответственно, система мониторинга – это программа или программный комплекс, который позволяет осуществлять данное отслеживание. Основными возможностями, которые предоставляют системы мониторинга сетей связи, являются [3]:

* Сбор метрик сетевых устройств – важные параметры, значение которых может значительно повлиять на работоспособность отдельного узла или всей сети, такие как загрузка CPU, температура аппаратных компонентов, доступность узла сети, количество потребляемой оперативной памяти и т.д. Метрики необходимо регулярно отслеживать.
* Агрегирование и анализ метрик. Современные системы мониторинга сетевой инфраструктуры не только собирают метрики со всех узлов в сети (либо с тех, на которых установлен агент системы), но и имеют возможность объединять метрики и анализировать их, представляя затем результаты анализа в виде графических элементов (графиков, гистограмм, диаграмм и т.д.).
* Система оповещений. При достижении одной или нескольких метрик критических, недопустимых значений, системы мониторинга отправляют сигнал системному администратору через электронную почту, СМС, мессенджер или уведомление на телефон. Наиболее продвинутые системы также обладают механизмами самовосстановления, что в некоторых ситуациях позволяет восстановить узел сети без вмешательства человека.

Несмотря на то, что мониторинг сети может осуществляться и в примитивном виде (например, периодические проверки сетевой доступности устройств с помощью утилиты ping), в настоящее время системы мониторинга чаще всего представляют собой сложную информационную систему, состоящую из нескольких компонентов (см. Рисунок 1). На удаленные узлы в сети устанавливаются специальные программы: агенты. Именно агенты позволяют собирать большое количество различных метрик с серверов, сетевого оборудования и стационарных компьютеров. Агенты обмениваются информацией с сервером, на котором хранится база данных и обрабатываются сигналы в случае, если метрики достигли недопустимых значений. Информация из базы данных, в свою очередь, отображается на интерфейсе в виде графиков и сообщений. Интерфейс может быть представлен в виде веб-приложения, программы или скрипта для командной строки.

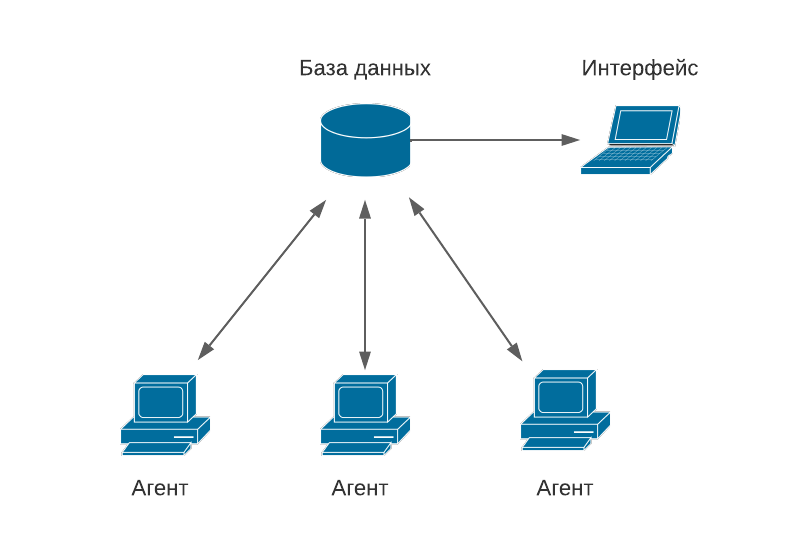


Рисунок 1 – Структура системы мониторинга

**2 ПРАКТИЧЕСКАЯ ЧАСТЬ**

**2.1 Анализ существующих систем мониторинга**

Целью данного этапа было изучить наиболее популярные в русскоязычном сегменте системы мониторинга, сравнить их между собой, выделить недостатки и преимущества, чтобы затем использовать эти сведения при разработке функциональных требований к собственной системе мониторинга.

В сравнительном анализе принимались во внимание только наиболее популярные на рынке стран СНГ продукты. Это связано с тем, что при выборе инструмента для контроля сетевой инфраструктуры компании обычно не рассматривают малоизвестные программные комплексы, предпочитая сразу же ориентироваться на лидеров рынка. Принимая во внимание этот факт, можно считать включение в работу непопулярных систем мониторинга нецелесообразным.

Для того чтобы отобрать объекты для сравнительного анализа было изучено содержание 50 статей о выборе системы мониторинга в русскоязычном сегменте сети Интернет. Всего в статьях встречались упоминания о 30 различных программных комплексах. Наиболее часто встречались упоминания о таких системах, как Zabbix, Nagios, Icinga, SolarWinds, Observium, PRTG Network Monitor, Prometheus и Cacti.

Следующим этапом при выборе систем мониторинга для анализа была сортировка отмеченных выше систем по популярности, опираясь на количество запросов в поисковой системе Яндекс. Выбор поисковой системы обусловлен следующими факторами:

* Нацеленность на русскоязычную аудиторию. Так как данная работа рассматривает системы мониторинга, наиболее популярные в русскоязычном сегменте сети Интернет, то анализ количества запросов следует проводить в поисковом сервисе, который нацелен на пользователей из России и СНГ.
* Наличие удобного инструмента. Яндекс предоставляет бесплатный инструмент Вордстат для анализа запросов в своих сервисах, которым можно пользоваться без регистрации и ограничений.

После получения результатов в сервисе Вордстат, они были отсортированы по убыванию количества запросов и представлены в виде таблицы (см. Таблицу 1)

Таблица 1 – Количество запросов о системах мониторинга в Яндексе

|  |  |
| --- | --- |
| Система мониторинга | Количество запросов за последний месяц |
| Zabbix | 103 380 |
| Cacti | 72 311 |
| Prometheus | 29 167 |
| Nagios | 2714 |
| SolarWinds | 2110 |
| PRTG Network Monitor | 586 |
| Icinga | 531 |
| Observium | 431 |

Как видно из полученных результатов, наибольшей популярностью в России и СНГ пользуются такие системы мониторинга, как Zabbix, Cacti, Prometheus и Nagios. Именно они будут выступать в качестве объектов анализа.

Так как решающую роль при выборе системы мониторинга предприятия и обычные пользователи отдают именно функционалу, подходящему под нужды бизнеса, в критерии не были включены технические характеристики продуктов, такие как язык разработки, используемая база данных и прочее. Тем не менее, если какая-либо особенность архитектуры оказывает влияние на работоспособность программы, это будет учтено при выставлении оценки.

Одним из наиболее важных критериев при выборе системы мониторинга можно считать качество визуализации, так как именно графики и гистограммы дают более наглядное и интуитивное представление о состоянии системы. Наличие качественных и подробных визуализаций позволит значительно сократить время на анализ ситуации в сети и поиск проблемы. Еще один критерий, который также может сильно повлиять на скорость обнаружения и устранения неполадок – это возможность системы мониторинга предоставлять наглядные и детальные отчёты о происшествиях. Наиболее ценной считается система, которая способна формировать отчеты для пользователей, которые не обладают высокой квалификацией в сфере IT-технологий. Также не менее важной является безопасная передача данных системой. При работе с собираемой информацией система мониторинга должна хранить, передавать и обрабатывать ее с минимальным риском к потере, повреждению или утечке. Эти три функции необходимы при работе на предприятии любого профиля и размера, без них система мониторинга не может полноценно выполнять свои функции, поэтому в процентом соотношении всех параметров им уделено наибольшее значение (см. Таблицу 2).

Далее следует упомянуть критерии, которые не играют решающей роли для отдельного пользователя, но могут проявиться при установке системы мониторинга на предприятия среднего и крупного бизнеса. К ним относятся поддержка различных типов устройств и приложений, способность собирать большое количество разнообразных метрик и готовность системы к работе с большими, высоконагруженными сетями. Эти критерии оценки хоть и не являются экстренно необходимыми всем компаниям, оказывают значительное влияние на выбор продукта.

К последней категории критериев относятся параметры, которые чаще всего не влияют на общую работоспособность системы, но при этом являются приятным дополнением, которое ускоряет и упрощает работу отдела системных администраторов. К таким критериям относятся системы оповещения пользователей, способность к прогнозированию трендов для метрик и простота первичной настройки. Эти критерии не оказывали большого влияния на конечную оценку продукта при сравнительном анализе, но все же имели значение при почти одинаковых результатах остальных параметров (см. Таблицу 2).

Итоговая шкала оценивания приведена в Таблице 2.

Таблица 2 – Шкала оценивания для сравнительного анализа

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий оценки | Минимальное значение | Максимальное значение | Долевое отношение к итоговой оценке |
| Качество визуализации | 0 | 10 | 0,20 |
| Качество отчетности | 0 | 10 | 0,15 |
| Безопасность системы | 0 | 10 | 0,20 |
| Поддержка устройств разного типа | 0 | 10 | 0,1 |
| Количество метрик | 0 | 10 | 0,1 |
| Работа в высоконагруженных сетях | 0 | 10 | 0,1 |
| Система оповещения | 0 | 10 | 0,05 |
| Способность к прогнозированию | 0 | 10 | 0,05 |
| Простота первичной настройки | 0 | 10 | 0,05 |
| Общая оценка | 0 | 100 | 1 |

Далее был проведен сравнительный анализ выбранных систем с использованием определенных ранее критериев. Далее будут пояснены выставленные оценки. Итоговый результат анализа представлен в Таблице 3.

*Качество визуализации*

Наиболее разнообразными инструментами визуализации среди рассматриваемых систем обладает Zabbix, предоставляя возможность не только строить графики по результатам сбора метрик, но и создавать пользовательские дашборды, размещая на них только необходимую информацию. Cacti тоже обладает сильным инструментом визуализации, позиционируя это, как свою основную функцию, однако разнообразие графиков не так велико, как в Zabbix. Также Cacti не предоставляет возможностей для визуального сравнения данных из разных источников.

В Prometheus и Nagios визуализация данных не предусмотрена совсем, и может быть получена либо с установкой дополнительного плагина NagVis, в случае с Nagios, либо установкой стороннего программного обеспечения Graphana.

*Качество отчетности*

Prometheus и Zabbix являются лидерами в категории составления отчётности. Функции Prometheus позволяют не только составлять отчёты по техническим метрикам, но и собирать данные об успешности бизнес-процессов с помощью настройки специальных скриптов. Недостатком отчётов Zabbix является то, что они предоставляются в достаточно вольном формате и не являются документом, который можно предоставить коллеге или пользователю. Nagios в свою очередь предоставляет очень ограниченные отчеты, а Cacti и вовсе не имеет такой функции.

*Безопасность системы*

Во время анализа данных в сети Интернет не было обнаружено информации о каких-либо серьёзных уязвимостях в системах мониторинга Zabbix и Prometheus. Имеется информация о незначительных ошибках, но они исправляются достаточно быстро. Более того, официальная документация систем содержит информацию о способах повышения безопасности их работы. Cacti является относительно стабильной системой, однако часть используемых системой серверов устарела и является желанной целью для атак злоумышленников. Система Nagios не считается безопасной и активно критикуется сообществом IT-специалистов. Существуют открытые инструкции о том, как обойти защиту Nagios, а скрипты для взлома находятся в публичном доступе.

*Поддержка устройств*

В плане поддержки устройств все системы находятся примерно на одном уровне, так как имеют очень широкий спектр устройств и сервисов, мониторинг которых можно осуществлять. Все, что невозможно мониторить по умолчанию, может быть добавлено с плагинами.

*Количество метрик*

Системы мониторинга Nagios и Zabbix умеют оперировать любыми необходимыми форматами данных – числовые, строковые, JSON-объекты, файлы и так далее. Prometheus, в свою очередь, не позиционирует себя, как система для хранения логов, а потому, за редким исключением, собирает только численные и строковые метрики без возможности собрать файлы логов или JSON-объекты. Это связано с тем, что для хранения данных Prometheus использует временные ряды и словари значений. Cacti же не имеет варианта со словарями, а потому по умолчанию позволяет сохранять только числовые значения.

*Работа в высоконагруженных сетях и масштабируемость*

При работе в крупных корпорациях часто приходится иметь дело с наблюдением за несколькими тысячами машин. Zabbix прекрасно справляется с работой в больших сетях и регулярно становится темой для обсуждения на конференции о высоконагруженных сетях Highload. Prometheus тоже хорошо работает даже при резком увеличении размеров сети, но для этого требуется установка дополнительных плагинов. В системе мониторинга Nagios показатели гораздо хуже. Несмотря на имеющиеся примеры поддержки сетей из 15 тысяч узлов, Nagios является очень статичной системой и любые изменения требуют их внесения в конфигурационные файлы вручную. Хуже всего справляется Cacti, так как даже на официальном сайте не дает информации о поддержке более, чем нескольких сотен машин.

*Система оповещения*

Система оповещений присутствует по всех рассматриваемых программах для мониторинга, что уже делает администрирование сети проще и мобильнее. Cacti и Nagios могут отправлять уведомления по смс и электронной почте, причем в Cacti для этого требуется отдельный плагин. Zabbix и Prometheus также имеют интеграцию с популярными мессенджерами, такими как Slack и Telegram.

*Способность к прогнозированию*

Прогнозирование появления аномалий на основе анализа предыдущих значений метрик имеется во всех системах, кроме Cacti, по молчанию. Cacti не имеет данной функции, однако ее можно дополнительно установить с помощью плагинов.

*Простота первичной настройки*

Nagios славится в сообществе сложностью своей настройки, требующей от пользователя глубоких знаний системного администрирования и программирования. Почти вся настройка осуществляется через конфигурационные файлы, а возможности интерфейса ограничены. Cacti также в основном настраивается без использования графического интерфейса, но за счет большей стабильности система является более простой в настройке. Zabbix и Prometheus предоставляют графический интерфейс практически для любой настройки, однако требуют усилий и времени на первичную настройку всех триггеров и дашбордов со стороны системного администратора.

Таблица 3 – Результаты сравнительного анализа

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | Zabbix | Cacti | Prometheus | Nagios |
| Качество визуализации | 9 | 7 | 2 | 3 |
| Качество отчетности | 8 | 0 | 10 | 4 |
| Безопасность системы | 10 | 6 | 10 | 2 |
| Поддержка устройств разного типа | 8 | 8 | 8 | 8 |
| Количество метрик | 10 | 4 | 7 | 10 |
| Работа в высоконагруженных сетях | 10 | 2 | 8 | 5 |
| Система оповещения | 10 | 6 | 10 | 8 |
| Способность к прогнозированию | 10 | 5 | 10 | 10 |
| Простота первичной настройки | 7 | 4 | 7 | 0 |
| Общая оценка после применения весов | 9,15 | 4,75 | 7,55 | 4,8 |

По результатам сравнительного анализа лучшей системой оказался программный комплекс Zabbix, набрав 9,15 баллов из 10 возможных. Zabbix считается наиболее популярной на текущий момент времени системой для мониторинга сети, и является универсальным инструментом. Zabbix позволяет осуществлять наблюдение не только за работой серверов, стационарных компьютеров и сетевого оборудования, но и контролировать IoT-устройства, сервисы, веб-сайты и собственные приложения. Zabbix регулярно обновляется и имеет большое сообщество разработчиков и пользователей. Prometheus может послужить альтернативой для Zabbix, так как единственным выявленным недостатком этой системы стало отсутствие полноценной визуализации метрик, что легко решается синхронизацией с Graphana. Тем не менее, оба этих инструмента, как и все рассмотренные выше системы мониторинга, требуют времени и усилий для настройки и использования. Согласно результатам сравнительного анализа на рынке систем для мониторинга сети не фигурирует программное обеспечение, позволяющее осуществлять мониторинг сразу же после установки программы на компьютер и практически не требующее от пользователя знаний системного администрирования. С учетом указанной информации в данной работе при разработке системы мониторинга упор был сделан на простоту наблюдения за сетевой инфраструктурой организации и управления ею, а также отсутствие необходимости в первичной настройке.

**2.2 Логическое проектирование приложения**

2.2.1 Формирование требований к приложению

Анализ преимуществ и недостатков различных систем мониторинга позволил выработать требования к функциональным возможностям приложения, к которым относятся:

* Аутентификация пользователя с помощью ключа
* Ограничение доступа к приложению при отсутствии ключа
* Просмотр всех устройств, на которых установлено клиентское приложение
* Просмотр информации об устройстве, включая статус в сети, сетевое имя, идентификатор материской платы, характеристики аппаратного обеспечения, операционную систему
* Просмотр логов за определенный промежуток времени
* Сортировка логов по дате получения и типу
* Наблюдение за изменением нагрузки на процессор и оперативную память устройства во времени
* Связь с пользователями устройств посредством чата
* Отключение устройства от системы мониторинга
* Создание нового ключа для пользователя системы мониторинга
* Просмотр информации о ключе, включая имя пользователя, для которого он был создан, уникальный идентификатор, настройки доступа для данного ключа и его назначение
* Преобразование ключа в QR-код для дальнейшего использования в мобильном приложении для системного администратора
* Изменение настроек доступа для существующего ключа

2.2.2 Создание блок-схемы работы приложения

С опорой на составленный ранее список функциональных требований, была создана блок-схема работы приложения (см. Рисунок 2). При открытии приложения первым делом проверяется доступность сервера, на котором хранятся и обрабатываются все данные. Если по какой-либо причине подключение к серверу невозможно осуществить, то приложение генерирует сообщение об ошибке и выводит его на экран с возможностью повторить попытку подключение. В том случае, если ответ от сервера был получен, на экран выводится сообщение об успешном подключении. Затем проверяется факт наличия предыдущего подключения, и если оно не обнаружено, перед полным доступом к функциям предложения необходимо будет ввести уникальный ключ-идентификатор, случайно генерируемый для каждого пользователя системы. Ключ проверяется на наличие в базе данных и сохраняется на устройстве. Больше ввод ключа не потребуется. После проверки доступности сервера и аутентификации пользователя будет получен доступ ко всем экранам приложения. Экраны не зависят друг от друга и могут быть открыты в любом порядке. При открытии экрана с подробно информацией о пользователе будет открыто соединение с сервером, через которое осуществляется чат. Также этот экран подразумевает возможность посмотреть логи событий и визуализацию метрик.

Схема является предварительной, принцип работы приложения может быть изменен на дальнейших этапах проектирования и разработки.

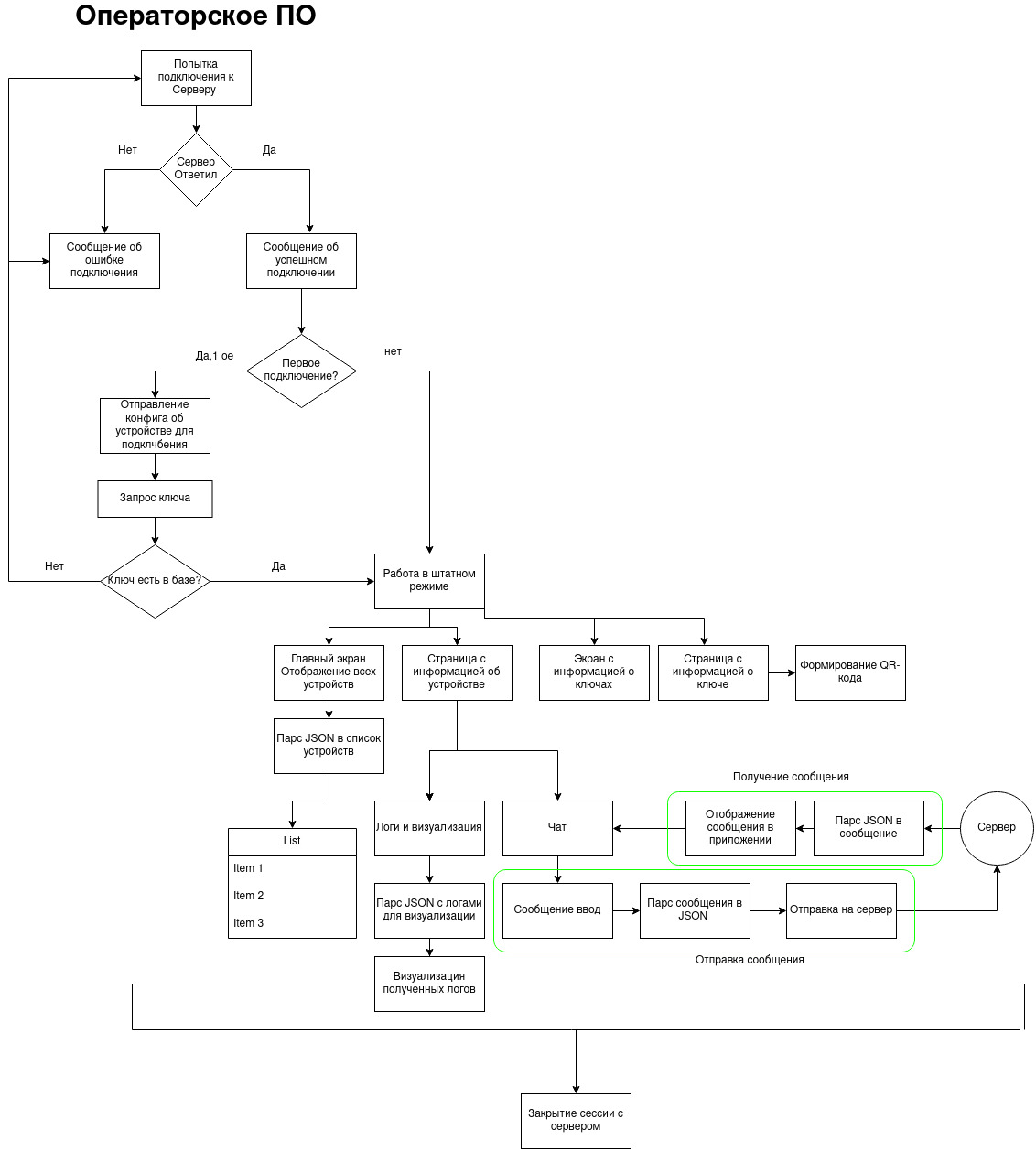


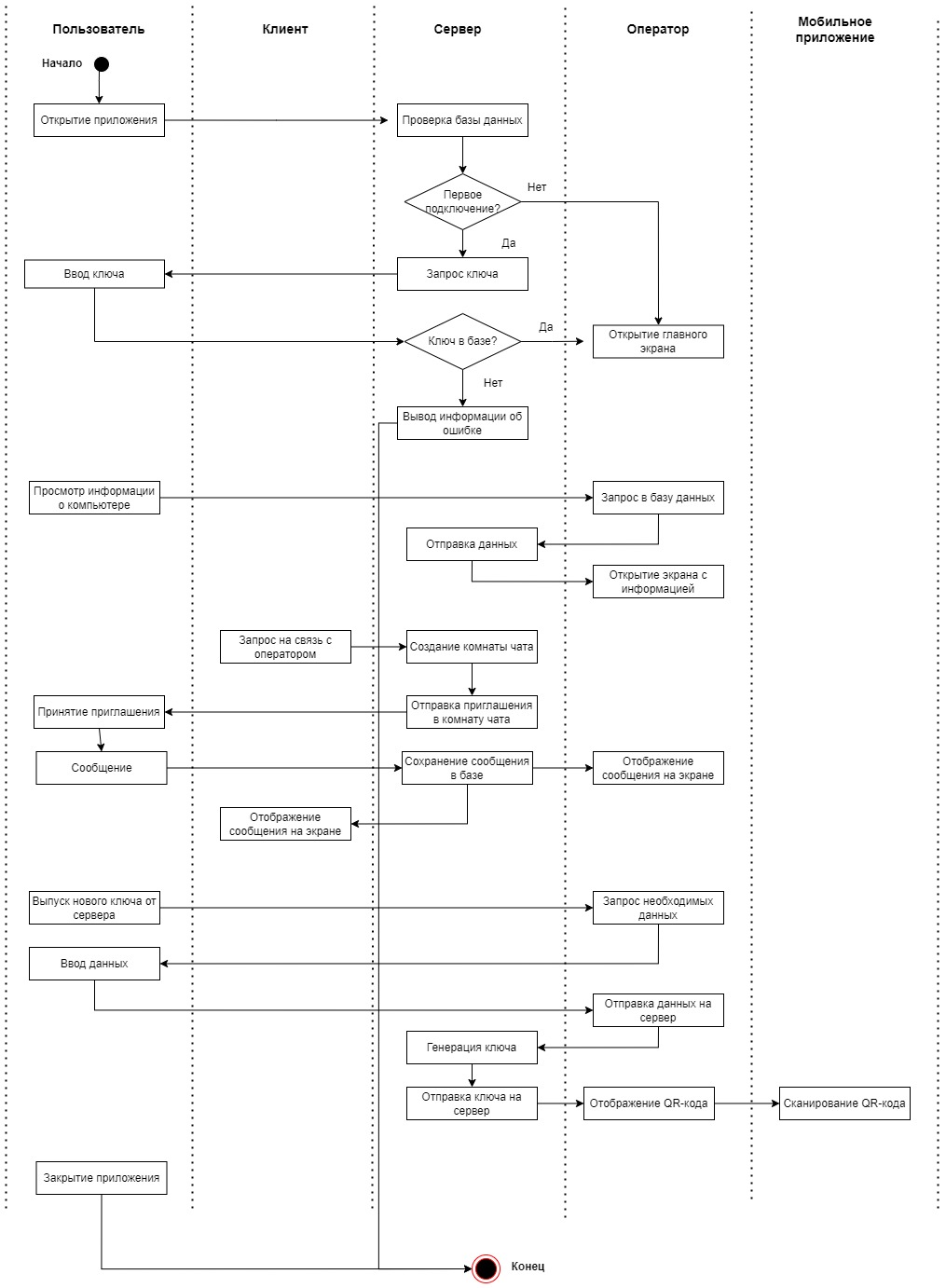
Рисунок 2 — Блок-схема работы приложения

2.2.3 Разработка диаграммы активностей

Затем была разработана диаграмма активности, демонстрирующая принцип взаимодействия всех компонентов системы мониторинга «InBetween» (см. Рисунок 3). При рассмотрении диаграммы важно учесть, что взаимодействие рассматривается со стороны системного администратора, работающего с операторским программным обеспечением (на диаграмме оно обозначено как «Оператор»). Как видно, подавляющее большинство сетевых взаимодействий производится с участием сервера. Это связано с тем, что сервер является связующим звеном между элементами системы, контролирующим исходящие и входящие данные. Даже при обмене информацией с клиентом в чате все сообщения сначала поступают на сервер, сохраняются в базу данных и только затем отсылаются сервером адресату для последующего отображения. Подобный принцип, запрещающий прямое взаимодействие двух конечных узлов, позволяет синхронизировать информацию на всех устройствах и увеличить уровень безопасности системы.

Также на сервере расположена база данных, в которой хранится вся имеющаяся в системе информация. Это позволяет уменьшить объем памяти, которую приложение занимает в системе, так как информация не сохраняется локально на компьютере пользователя.

Единственными действиями пользователя, в которых не задействован сервер, являются отображение на экране QR-кода с зашифрованным ключом от базы данных для дальнейшего сканирования мобильным устройством и закрытие приложения. Это связано с тем, что при выполнении этих действий никакая новая информация в систему не сохраняется: операторское программное обеспечение не отправляет на сервер данные о том, работает оно или нет, а QR-код генерируется на основе уже ранее запрошенной приложением информации и сохраняется только на время открытия диалогового окна.

Рисунок 3 — Диаграмма активностей

2.2.4 Разработка диаграммы классов

Диаграмма классов — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними [4]. Диаграмма классов является очень важным этапом проектирования, так как позволяет распланировать не только набор классов для создания, но также их содержимое и то, как они связаны между собой. Подобный этап планирования позволяет сократить время на разработку и количество правок, которые вносятся в код.

В ходе создания диаграммы (см. Рисунок 4) был определен 21 класс. К ним относятся 4 статических класса, которые предназначены только для использования в других частях программы и не предполагают создания отдельных экземпляров; 5 классов-виджетов, которые представляют собой элемент графического интерфейса программы, а их методы обычно предназначены для вывода на экран данных, передаваемых из других классов; 3 классов, предназначенных исключительно для хранения данных и не предполагающих создание методов для них. Остальные классы составляют основную структуру приложения и, предположительно, будут представлены в виде окон, которые обрабатывают и выводят на экран необходимую информацию. Некоторые такие классы не содержат полей, так как работают с информацией, получаемой из других элементов программы.

На диаграмме отображены 3 вида отношений между классами. Прямой линией обозначено отношение ассоциации, когда элементы существуют независимо друг от друга, но взаимодействуют в ходе работы приложения. Для таких отношений указана кратность. Белым ромбом обозначены отношения ассоциации: класс, в сторону которого направлен ромб, включает в себя второй класс, являющийся его частью, но при этом второй класс может также существовать независимо от первого. Черным ромбом обозначены отношения композиции или строгой ассоциации. В этом случае зависимый объект не может существовать без основного.

Основным классом, с которого начинается работа программы, является MainWindow, все остальные окна и виджеты, как и связанные с ними классы, открываются в ходе работы методов класса MainWindow.

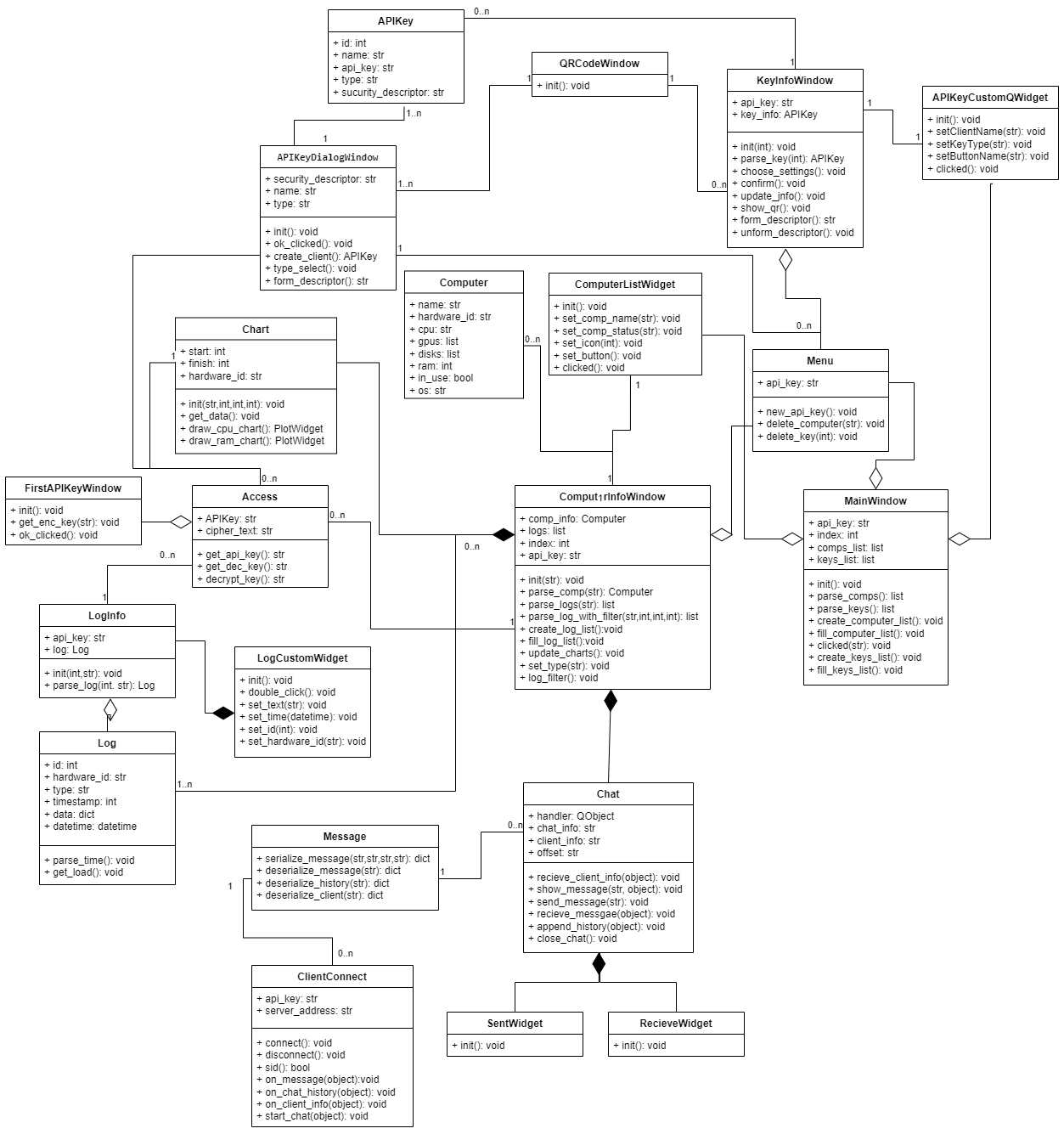


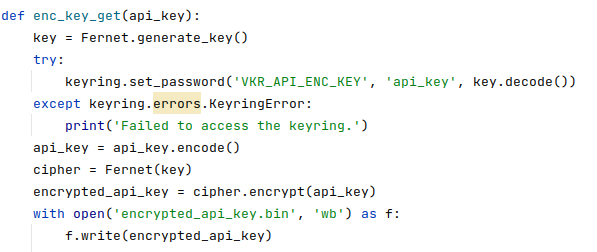
Рисунок 4 — Диаграмма классов

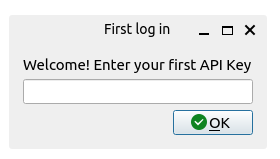
**2.3 Разработка приложения**

Разработка приложения велась на языке Python с использованием библиотеки для приложений с графическим интерфейсом PyQt6. Выбор языка обоснован следующими факторами:

* Синтаксис языка Python значительно проще и короче, чем остальных языков, которые рассматривались для разработки. Также этот язык менее чувствителен к ошибкам. В совокупности это позволяет сократить время на разработку продукта
* Для языка Python написано большое количество библиотек и фреймворков для визуализации данных, которая является основной функцией системы мониторинга
* Python хорошо подходит для анализа данных и построения математических моделей, что позволит в дальнейшем реализовать функцию прогнозирования метрик
* Графическая библиотека PyQT распространяется бесплатно с 1998 года, регулярно обновляется и дополняется, имеет большое сообщество разработчиков, как в русскоязычном, так и в англоязычном сегментах интернета

Так как система мониторинга непрерывно работает с сервером, хранящим большое количество информации о сети, в целях безопасности доступ к приложению для системного администратора предоставляется по специальному ключу. Этот же ключ позволяет получить доступ к базе данных. Когда пользователь впервые запускает приложение, открывается диалоговое окно, требующее ввести ключ доступа (см. Рисунок 6). Когда пользователь вводит известный ему ключ, начинается процесс шифрования и занесение зашифрованного ключа в базу паролей keyring (см. Рисунок 5). Эта технология позволяет безопасно сохранять пароли на устройстве, и повторное введение ключа не потребуется.

Рисунок 5 — Шифрование и сохранение ключа

Рисунок 6 — Диалоговое окно ввода ключа

Работа программы начинается с главного экрана, представленного классом MainWindow. Этот класс наследует свойства от встроенного в библиотеку PyQT6 класса QMainWindow. Главный экран представляет собой список, содержимое которого зависит от нажатия одной из трёх кнопок в верхней части экрана (см. Рисунок 7). Все три кнопки обрабатываются с помощью функции clicked, поведение которой зависит от переданного в качестве параметра сигнала (см. Рисунки 8-9) На экран выводятся список компьютеров в сети, оповещение о скором появлении новой функции (синхронизация с Active Directory через LDAP) и список API-ключей, которые есть в базе данных соответственно. Также предусмотрено предупреждение об ошибке в случае сбоя в системе и передачи в функцию несуществующего сигнала.



Рисунок 7 – Кнопки изменения содержимого экрана

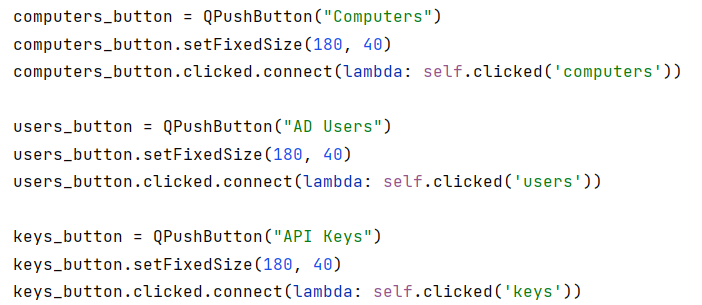


Рисунок 8 – Создание кнопок и передача сигнала

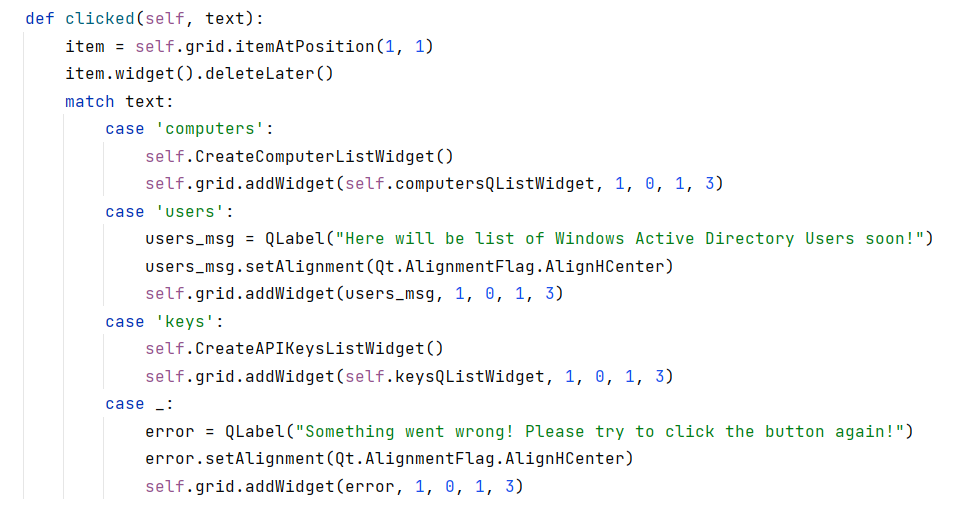
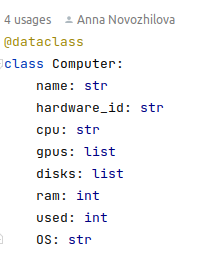
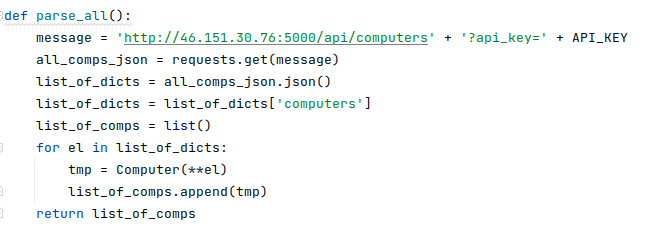


Рисунок 9 – Обработка сигналов в функции

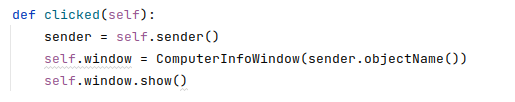
По умолчанию и при нажатии кнопки Computers на главный экран выводится список компьютеров, на которые было установлено клиентское приложение и которые, соответственно, были сохранены в базе данных. Запрос к базе отправляется каждый раз, когда необходимо создать список. Это позволяет отображать актуальную информацию сразу же после обновления базы.

Для получения информации с сервера была написана функция, отправляющая JSON-запрос на сервер (см. Рисунок 11). Для доступа к базе потребуется ключ, ранее сохраненный в keyring. После получения ответа от сервера json-объект преобразуется в список словарей, которые, в свою очередь, передаются в качестве параметра конструктора датакласса Computer (см. Рисунок 10). Датакласс — декоратор для классов в языке Python, предоставляющий функции для автоматического добавления сгенерированных специальных методов, таких как \_\_init\_\_() и \_\_repr\_\_() , в определяемые пользователем классы. Декоратор позволяет указать только поля и тип находящихся в них данных при создании класса. Эта конструкция подходит для работы с классами, предназначенными для хранения данных.

Рисунок 10 — Датакласс Computer

Рисунок 11 — Создание списка компьютеров

Для отображения полученной информации был создан пользовательский виджет. Виджет позволяет выводить на экран информацию о сетевом имени устройства, статусе в сети, изображение, соответствующее типу устройства (на данный момент система поддерживает только пользовательские компьютеры, однако в будущем планируется добавить мониторинг маршрутизаторов и коммутаторов). Также в макет виджета включена кнопка, открывающая окно с более подробной информацией о компьютере и передающая в него, как параметр, идентификатор материнской платы компьютера (см. Рисунок 12).

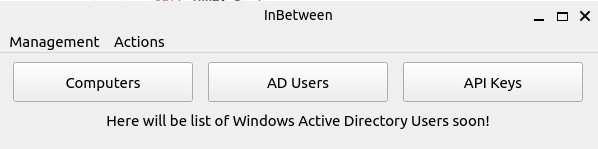
Рисунок 12 — Открытие окна с информацией о компьютере

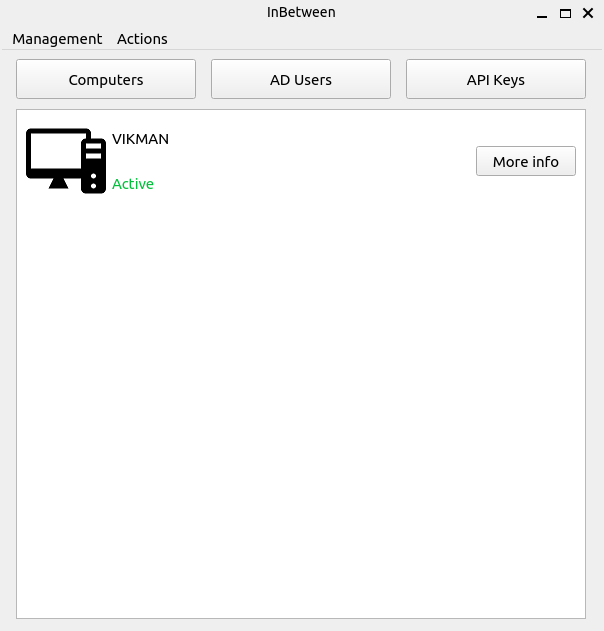
Список устройств заполняется с помощью функций CreateComputerListWidget (см. Рисунок 13), которая очищает предыдущий список, заново получает информацию из базы данных и запускает процесс создания виджетов, и FillComputerListWidget (см. Рисунок 13), которая создает виджеты, заполняет их нужной информацией и добавляет их в список.

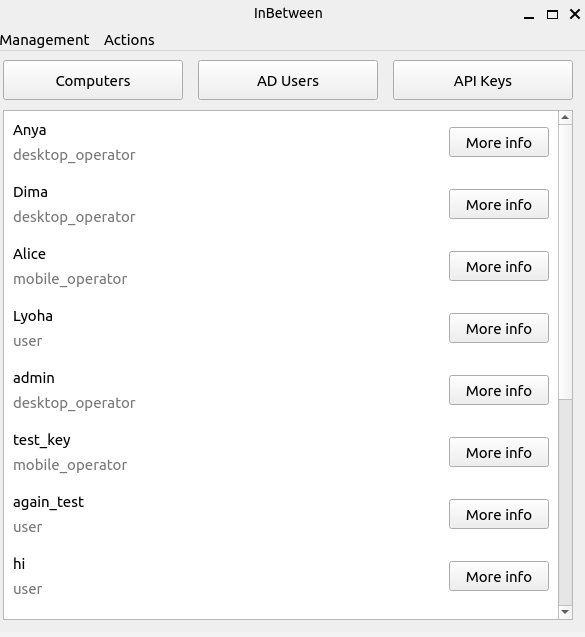
Аналогичным образом создается и заполняется список ключей, которые есть в базе данных. Отличием является только содержимое виджета: для ключа на экране отображается имя владельца, его роль в системе (рядовой пользователь, администратор, работающий с компьютера или мобильный администратор) и кнопка открытия окна с более подробной информацией.

Рисунок 13 — Создание списка компьютеров

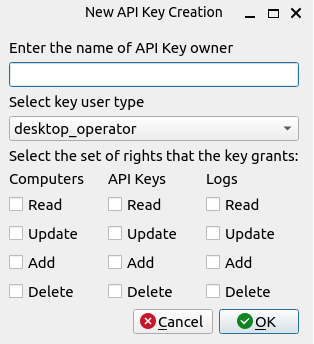
В результате работы представленного выше кода на экране отображаются списки компьютеров и ключей при нажатии на соответствующие кнопки на экране (см. Рисунки 15-16). Кнопка AD Users на момент написания работы выводит на экран сообщение с сообщением о скором появлении новой функции (см. Рисунок 14).

Рисунок 14 — Сообщение об отсутствии функции

Рисунок 15 — Список устройств, подключенных к системе

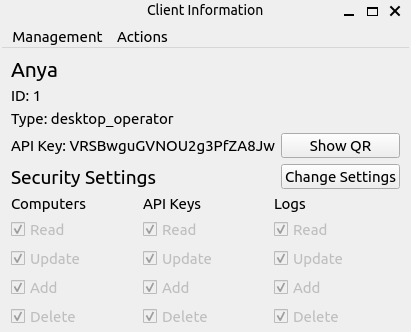
Рисунок 16 — Список API-ключей

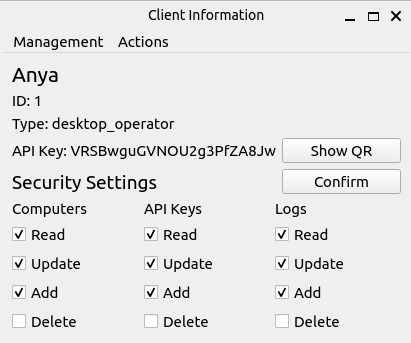
Возможности системы мониторинга не предполагают создание новых устройств вручную, так как они автоматически добавляются в базу при установке на компьютер клиентского приложения «InBetween». Однако, через верхнее меню есть возможность добавить в базу данных новый ключ от базы данных. Этот функционал реализован с помощью диалогового окна — специального класса, созданного для получения информации от пользователя.

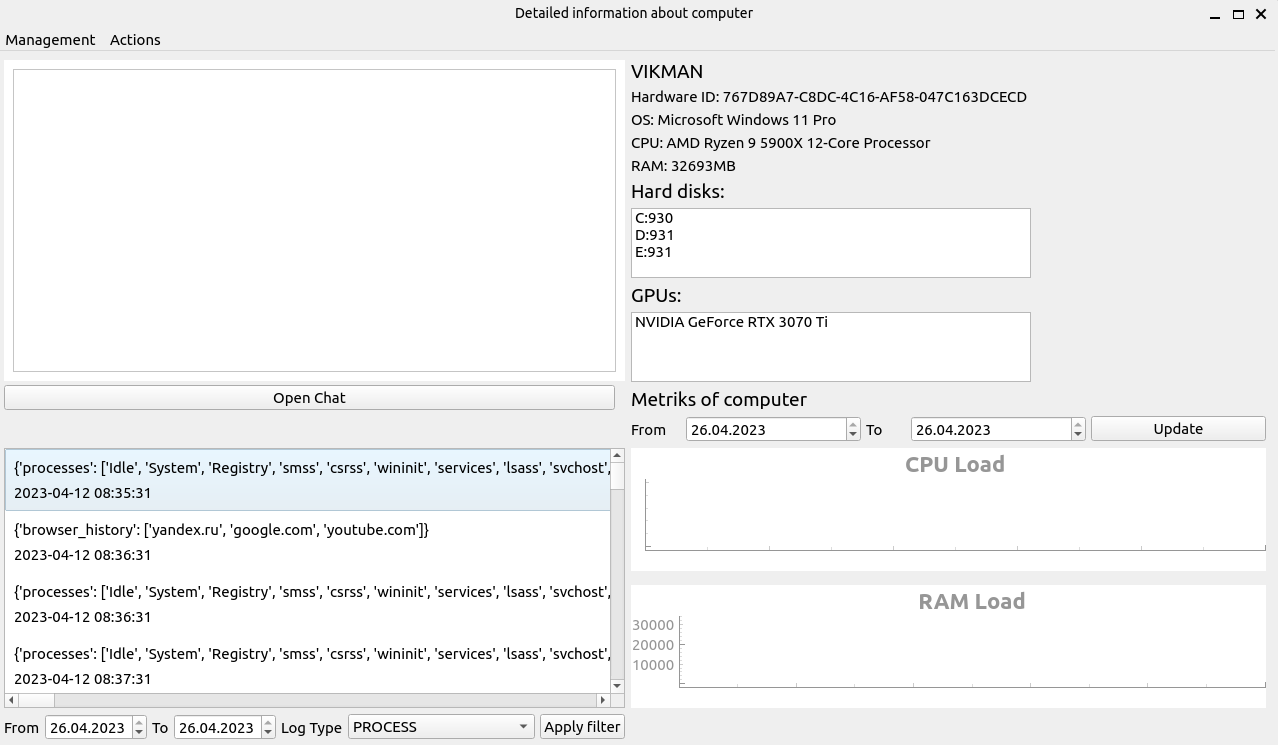
Кюар код

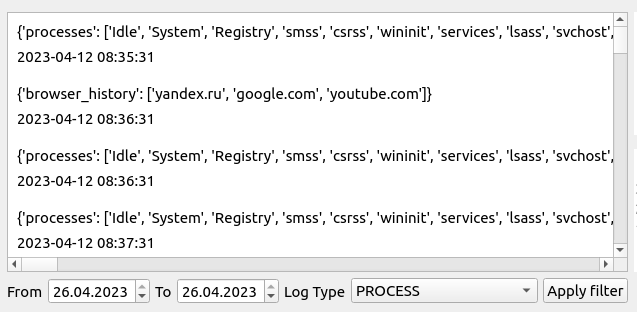


Окно информации о ключе

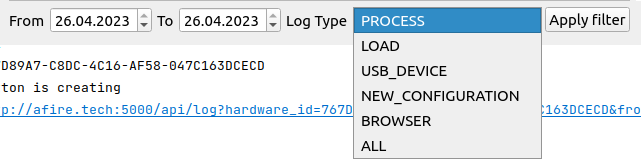
Разблокировка настроек безопасности

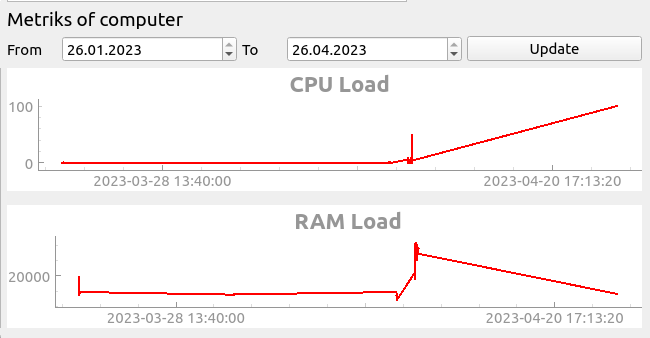
Окно с инфой о компе

Логи

Отдельный лог

Фильтр логов

Графики



**2.4 Тестирование итогового приложения**

2.4.1 Разработка сценариев использования

2.4.2 Ручное тестирование

**ЗАКЛЮЧЕНИЕ**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

**СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ**

**ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**