Санкт-Петербургский государственный университет

НИКИТИНА Анастасия Максимовна

Выпускная квалификационная работа

Разработка инструмента для графического конфигурирования оборудования сети 4G

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5162.2020 «Технологии программирования»*

Научный руководитель:
профессор кафедры системного программирования, д.ф.-м.н. А.Н. Терехов

Рецензент:
программист отдела разработки департамента разработки и внедрения программного обеспечения ООО «МОБИЛ-ГРУП» И. О. Гущин

Санкт-Петербург
2024

Saint Petersburg State University

Anastasiia Nikitina

Bachelor's Thesis

Development of a tool for graphical configuration of 4G network equipment

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5162.2020 "Programming Technologies"*

Scientific supervisor:
Sc.D (Physics and Mathematics), prof. A.N. Terekhov

Reviewer:
Software Engineer, Department of SWE Development and Integration at "Mobil-group"
I.O. Gushchin

Saint Petersburg
2024

Оглавление

Введение

4G — это стандарт беспроводной связи четвертого поколения, который обеспечивает высокоскоростной доступ к интернету и передачу данных на мобильных устройствах. Сами сети 4G существуют с 2009 года, то есть уже около пятнадцати лет [10].

До недавнего времени настройкой сетей занимались люди, работающие совместно с операторами сотовой связи, имеющие при этом специальное образование. Однако в последние несколько лет наметилась тенденция внедрения технологий 4G в самые разнообразные области, туда, где они раньше не использовались — например, на промышленные предприятия или в транспортную отрасль [18]. Часто приходится разворачивать собственные локальные LTE-сети там, где отсутствует доступ к полноценным сетям. Например, на месторождениях при добыче полезных ископаемых.

Компании ищут способы внедрить новые технологии в работу, затратив при этом минимальное количество ресурсов, но далеко не на всех предприятиях имеются сотрудники, квалификация которых позволяет заниматься настройкой и администрированием сетей.

Существует два решения данной проблемы. Первое заключается в обучении сотрудников новым технологиям, однако это требует значительных затрат, как финансовых, так и временных. Вторым вариантом решения проблемы является упрощение процесса настройки сетевых систем. Если сделать управление сетями более понятным и доступным, то решением этой задачи смогут заниматься даже люди, специализация которых отлична от области сетевых технологий. Это поможет сэкономить ресурсы, которые могли быть затрачены на переквалификацию имеющихся кадров или поиск новых сотрудников.

Таким образом появилась потребность в продукте, который позволит настраивать параметры сетей 4G людям без глубоких знаний сетевых технологий.

Исходя из этого запроса российские компании ООО «Мобил-груп» и ООО «ЛИС» приняли решение разработать комплекс программ, кото-

рый позволит настраивать сети 4G графическим способом. Выбор в сторону визуального представления был сделан по причине того, что графические средства являются наиболее понятными и доступными для людей, которые имеют минимальные знания о программировании.

Планируется, что продукт будет поставляться предприятиям в виде комплекса программ, состоящего из десктопного приложения и редактора диаграмм.

1 Постановка задачи

Цель работы — разработать инструмент, позволяющий графически настраивать сети 4G. Для достижения этой цели были поставлены следующие задачи.

- Сформулировать требования к разрабатываемому инструменту.
- Выполнить обзор существующих решений для графической настройки сетей.
- Выполнить обзор технологий и инструментов, которые позволяют реализовать необходимую функциональность.
- Разработать архитектуру инструмента.
- Реализовать инструмент согласно требованиям и архитектуре.
- Провести апробацию реализованного решения.

2 Обзор

2.1 Обзор предметной области

Основная цель разрабатываемого приложения — предоставить простой доступ к конфигурированию сетей 4G. Для настройки оборудования необходимо уметь менять различные параметры в конфигурационных файлах. При этом, конфигурационных файлов для настройки одной сети может быть несколько, и в каждом из них большое количество параметров, в которых довольно легко запутаться неподготовленному пользователю. Основная идея для решения этой проблемы — вынести на диаграмму сети только необходимые параметры из конфигурационных файлов, которые пользователи смогут менять.

Стоит отметить, что в рамках данной работы рассматривается взаимодействие с оборудованием Amarisoft [3]. В связи с этим ниже представлено описание файлов именно этого производителя. Однако, конфигурационные файлы других производителей оборудования устроены похожим образом, и архитектура разрабатываемой системы предусматривает внедрение компонентов для обработки других видов конфигурационных файлов.

На рисунке 1 представлена структура конфигурационных файлов Amarisoft.

После изучения структуры конфигурационных файлов было выяснено, что пользователю достаточно уметь изменять параметры в трех файлах:

- Файл `enb.cfg` отвечает за настройку базовой станции.
- Файл `ims.cfg` содержит параметры, связанные с работой IP Multimedia Subsystem (IMS), отвечающей за предоставление услуг мультимедиа в сети LTE.
- Файл `mme.cfg` содержит параметры, связанные с работой Mobility Management Entity (MME). Эти параметры влияют на процессы аутентификации, авторизации и управления сессиями.

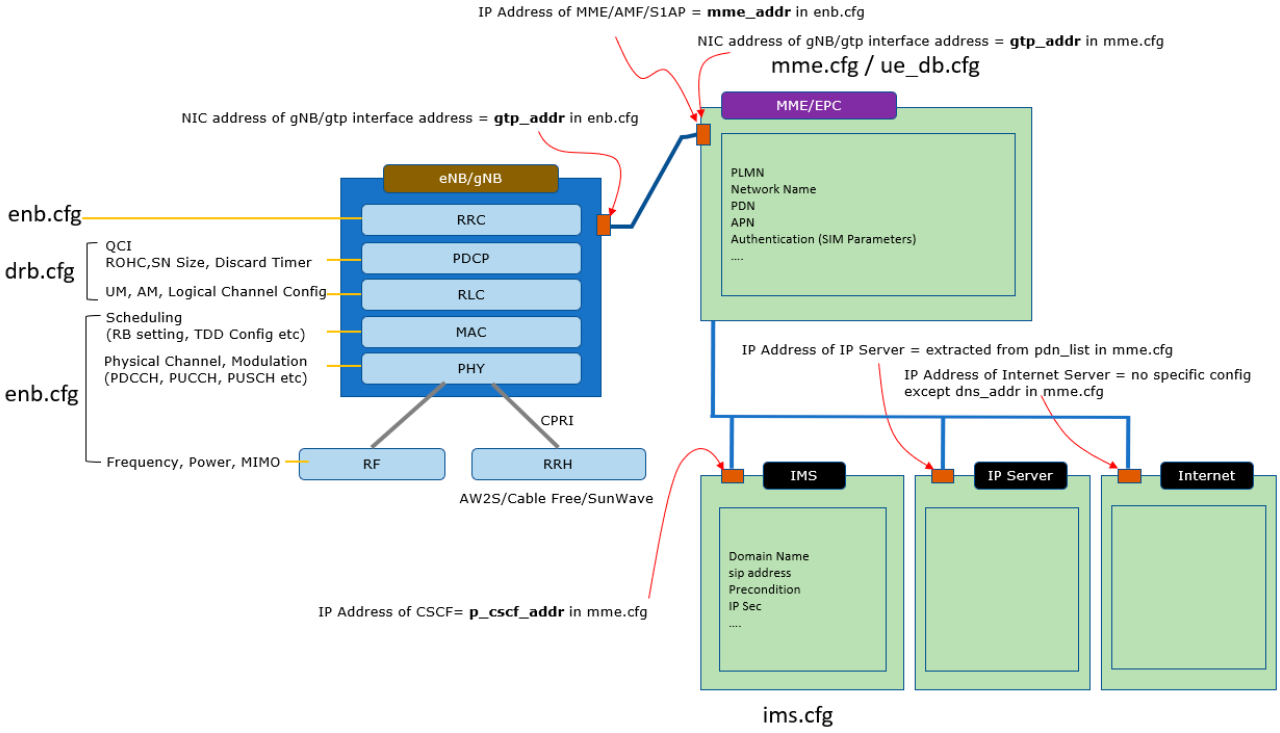


Рис. 1: Устройство конфигурационных файлов оборудования Amarisoft [3]

Помимо трех описанных выше файлов для настройки сетей существуют еще два вспомогательных файла:

- Файл `drb.cfg` относится к конфигурации Data Radio Bearer (DRB) в LTE сети. DRB - это канал передачи данных, используемый для передачи пользовательских данных, таких как веб-страницы, видео и т. д.

- Файл `ue_db.cfg` содержит конфигурацию базы данных User Equipment (UE), то есть устройств, подключающихся к сети LTE.

Однако управление параметрами в данных файлах чаще происходит автоматически, а неправильные изменения могут привести к нестабильности сети или сбоям в обслуживании абонентов. Именно поэтому ручная настройка требует высокой квалификации и применяется довольно редко.

Исходя из вышесказанного, далее на диаграмму выносятся параметры именно из трех файлов, описанных выше.

Сами конфигурационные файлы записываются формате с синтаксисом, похожим на Javascript Object Notation (JSON) [7], однако, с некоторыми отличиями:

- Кавычки для имен свойств не используются, если только имя не начинается с цифры.
- Фигурные скобки на верхнем уровне необязательны.
- Свойства в файле могут дублироваться. При этом применяется то значение, которое записано ниже по ходу чтения файла.
- Могут быть включены другие файлы с помощью ключевого слова `include`. Разрешение ситуации с одинаковыми параметрами внутри разных файлов происходит аналогично ситуации с дублированием внутри одного файла, описанной выше.

2.2 Обзор существующих решений

В данном разделе приведен обзор некоторых приложений, которые позволяют графически управлять настройками различного сетевого оборудования.

- АРГУС СИРИУС (Система Интеграции Ресурсов и Управления Сетью) [29] — программно-аппаратный комплекс, который предоставляет интерфейс для управления сетевым оборудованием. Приложение состоит из нескольких компонентов: модуль исследования сети, диагностики и конфигурации. Модуль исследования позволяет получить данные сети и автоматически построить по ним схему. Модуль конфигурации позволяет управлять сетевым оборудованием, приложениями и базами данных оператора на основе заранее составленных базовых действий. Модуль диагностики предназначен для проверки работоспособности оборудования и услуг на сети оператора. Данное решение не поддерживает работу с сетями 4G.

- 10-Strike LANState [1] — программа для администрирования и мониторинга серверов, компьютеров и прочих сетевых устройств. Данное приложение позволяет сканировать сеть, распознавать типы подключенных устройств, а также строить графическую карту сети. Помимо построения карты, программа позволяет администрировать устройства внутри сети: управлять запуском и перезагрузкой удаленных серверов и рабочих станций. При довольно обширной функциональности данный инструмент не позволяет настраивать сети 4G.

- Algorius Net Viewer [2] — это комплексное программное обеспечение для мониторинга и управления компьютерными сетями. Он предоставляет широкий набор инструментов для визуализации, контроля и анализа сетевой инфраструктуры. Программа позволяет обнаруживать и визуализировать сеть с помощью интерактивных карт, а также управлять устройствами внутри сети. Так же, как предыдущие аналоги, приложение не поддерживает работу с сетями стандарта 4G.

Описанные выше решения были выбраны для сравнения ввиду наличия функциональности, схожей с той, которая реализуется в рамках данной работы. Стоит отметить, что полного по функциональности аналога, поддерживающего работу с сетями 4G, найдено не было.

2.3 Выбор редактора диаграмм

Перед началом работы необходимо было выбрать один из двух подходов: реализовывать собственный графический интерфейс, либо взять за основу уже существующий редактор диаграмм и на основе его элементов строить диаграммы сети, после чего отправлять данные в другие компоненты приложения. Выбор был сделан в пользу второго варианта, так как данное решение позволит ускорить процесс разработки из-за отсутствия необходимости в настройке и отладке самого процесса построения схем.

К редактору, на основе которого будет реализовываться инструмент, были предъявлены следующие требования:

- Возможность экспорта диаграммы не только в графическом формате, но и в виде кода на одном из языков разметки (например, XML или HTML).
- Лицензия, позволяющая бесплатно использовать данный редактор в разработке коммерческого программного обеспечения.

На предмет соответствия установленным выше требованиям было проанализировано несколько популярных решений:

- Lucidchart [9] — веб-инструмент для визуального представления информации в виде блок-схем и диаграмм. Данное решение имеет довольно удобный и простой интерфейс, однако экспорт возможен только в графических форматах, либо в CSV, что является неудобным для дальнейшей обработки диаграмм.
- Microsoft Visio [12] — редактор диаграмм и блок-схем от Microsoft [11]. Изначально разрабатывался как десктопное приложение под Windows. Позже появилась возможность использовать Visio в браузере с хранением созданных проектов в облаке. Данный инструмент поддерживает экспорт файлов в формате .vsdx. Этот формат представляет из себя архив, в котором содержатся все данные документа Visio (в том числе XML-файл). Однако, чтобы использовать данный инструмент в коммерческом продукте, необходимо приобретать специальную лицензию у Microsoft.
- Draw.io [4] — десктопный редактор диаграмм (также существует веб-версия Diagrams.net). Данный инструмент поддерживает экспорт проекта в формате XML. Draw.io имеет открытый исходный код и лицензию Apache 2.0, которая разрешает использование ПО в том числе в коммерческих целях.

- yEd [28] — бесплатное приложение для создания и редактирования диаграмм, разработанное компанией yWorks. Оно позволяет визуализировать и анализировать различные типы диаграмм и графиков. Решение поддерживает экспорт в формате XML, однако имеет проприетарную лицензию, запрещающую использование программы в коммерческих целях.

Также в обзор был включен следующий инструмент, не являющийся редактором диаграмм, однако часто использующийся в разработке программного обеспечения для визуального моделирования.

- Eclipse Graphical Model Project [5] — ключевой фреймворк в экосистеме Eclipse, предоставляющий базовые технологии и инфраструктуру для создания инструментов и приложений, основанных на модельно-ориентированной разработке (MDD). Имеет устаревший дизайн и высокий порог вхождения, что приведет к более долгому процессу разработки.

После проведенного обзора выбор был сделан в пользу Draw.io, так как он прост в освоении, а его функциональность и лицензия удовлетворяют поставленным требованиям.

2.4 Выбор библиотеки для создания графического интерфейса

Перед тем как начинать разработку продукта, необходимо было выбрать инструмент, который позволяет создавать приложения с графическим интерфейсом.

В главе «Требования к системе» описано, что разработка десктопного приложения на Python являлась одним из требований компании ООО «Мобил-груп». Именно поэтому в дальнейшем в обзоре участвовали Python-библиотеки для создания графического интерфейса.

Помимо компонентов, подходящих под макет, одним из важных требований к библиотеке было наличие лицензии, которая позволяет

бесплатно использовать библиотеку в разработке коммерческого программного обеспечения с закрытым исходным кодом. Было рассмотрены несколько наиболее популярных фреймворков и библиотек:

- Tkinter [22];
- Kivy [8];
- PyQT [15];
- PySide [16];

Их сравнение представлено в таблице 1.

После сравнения решено было остановиться на библиотеке PySide, так как она является наиболее популярной при разработке десктопных приложений, имеет большое сообщество и предоставляет более современные компоненты интерфейса. Также лицензия LGPL позволяет бесплатно использовать данный инструмент, не предоставляя исходный код разрабатываемого программного обеспечения.

Название	Краткое описание	Лицензия
Tkinter	Кросс-платформенная библиотека на основе средств Tk. Входит в стандартную библиотеку Python. Предоставляет устаревший интерфейс.	Python-лицензия

Kivy	Фреймворк Python с открытым исходным кодом для разработки приложений с нативным пользовательским интерфейсом (NUI). Чаще используется для создания мобильных приложений, так как в библиотеку встроена поддержка жестов сенсорного интерфейса.	MIT (возможно использовать код в закрытом коммерческом ПО)
PyQT	Набор расширений графического фреймворка Qt для языка программирования Python.	Имеет двойную лицензию: GPLv3 (подразумевает, что продукты созданные на основе данной библиотеки, обязательно должны поставляться с исходным кодом), а также коммерческую (разрешает использование в коммерческом продукте, но только при покупке специальной лицензии).

PySide	Привязка языка Python к инструментарию Qt, совместимая на уровне API с PyQt	LGPL (в отличие от PyQT библиотека доступна для использования как в открытых, так и в закрытых продуктах, в том числе в коммерческих)

Таблица 1: Таблица сравнения библиотек для создания графического интерфейса

3 Требования к системе

Были выработаны следующие требования к функциональности системы.

- Загрузка конфигурационных файлов с удаленного сервера, поддерживающего связь с оборудованием.
- Отображение ошибок при подключении к удаленному серверу и загрузке директорий с конфигурационными файлами.
- Построение диаграммы сети по загруженным конфигурационным файлам.
- Редактирование параметров сети на диаграмме.
- Выделение цветом линий, соответствующих интерфейсам, в зависимости от статуса соединения.

Помимо функциональных требований, были также выдвинуты следующие нефункциональные требования к системе.

- Реализация инструмента на языке Python.
- Реализация инструмента в виде десктопного приложения.
- Кроссплатформенность.
- Покрытие кода модульными тестами.
- Интуитивно понятный интерфейс.
- Наличие пользовательской инструкции по работе с приложением.

4 Описание архитектуры

В этой главе описываются важнейшие решения по организации системы.

4.1 Общее описание работы системы

Прежде чем настраивать сеть графически, необходимо построить диаграмму сети, где будут отображаться параметры, которые пользователь сможет изменять. Диграмма строится по конфигурационным файлам сети. Данные файлы изначально хранятся на сервере, где установлено программное обеспечение, поддерживающее связь с оборудованием и передающее все изменения в конфигурационных файлах непосредственно в сеть.

Подключение к серверу осуществляется по протоколу SSH. После подключения выполняется загрузка конфигурационных файлов с сервера на локальный компьютер, где происходит парсинг и дальнейшая обработка файлов. После конвертации конфигурационных файлов в файл диаграммы отдельным процессом запускается редактор Draw.io с построенной диаграммой.

Диаграмма сети состоит из нескольких страниц. На первой странице, открывающейся перед пользователем, располагается общая структура сети, где элементы являются подсистемами сети. За каждым из этих элементов закреплена гиперссылка, с помощью которой пользователь может перейти на страницу с более подробной структурой конкретной подсистемы.

После того, как пользователь изменяет параметры, обновленные конфигурационные файлы загружаются обратно на сервер, который передает все изменения непосредственно на оборудование сети. Если новые конфигурационные файлы успешно применились, соответствующий интерфейс на диаграмме подсвечивается зеленым, иначе — красным.

4.2 Диаграмма классов

Приложение реализовано в рамках шаблона «Model-View-Controller». На рисунке 2 представлена UML-диаграмма классов системы.

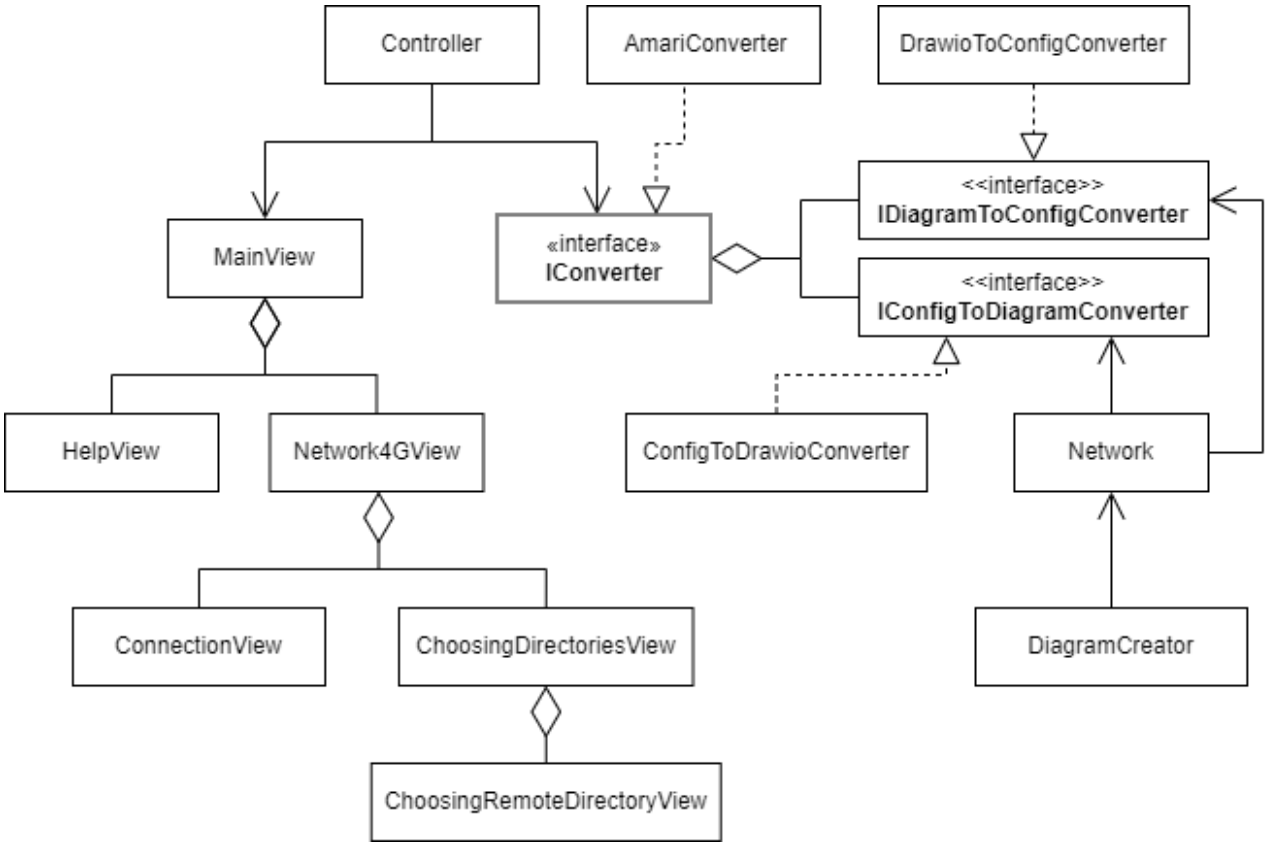


Рис. 2: Диаграмма классов системы

Основную функциональность в модуле Model реализуют наследники интерфейса IConverter. Именно там происходит конвертация конфигурационных файлов в файл диаграммы. На выходе конвертера создается объект класса Network, который содержит путь к директории, а также информацию о самой сети. По этому объекту строится диаграмма сети в классе DiagramCreator. Взаимодействие с конфигурационными файлами Amarisoft происходит в классе AmariConverter, который реализует описанный выше интерфейс.

Модуль View предназначен для отображения данных. Основной класс в этом модуле — MainView, в котором с помощью объекта класса QTabWidget библиотеки PySide происходит изменение интер-

фейса в зависимости от того, какая вкладка открыта пользователем. Класс HelpView отвечает за окно с пользовательской инструкцией, а Network4GView — за основную часть приложения, и состоит из нескольких виджетов: ConnectionView — содержит текстовые поля для ввода данных, чтоб подключиться к серверу по SSH, ChoosingDirectoriesView — обеспечивает возможность выбора директорий, откуда и куда будут загружаться конфигурационные файлы. Класс ChoosingRemoteDirectoryView предоставляет диалоговое окно со списком директорий на удаленном сервере.

За Controller-часть отвечает класс Controller, который содержит в себе объект класса Network.

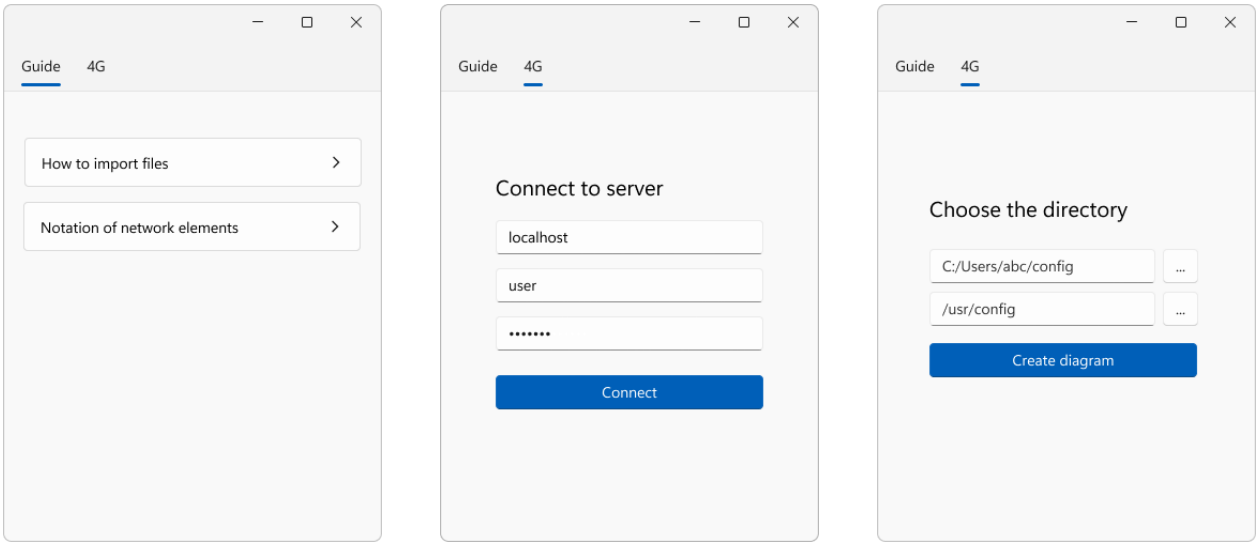
4.3 Пользовательский интерфейс

Перед тем, как реализовывать пользовательский интерфейс, были составлены его макеты в онлайн-сервисе Figma [6]. Они представлены на рисунке 3.

На рисунке 3a изображен макет окна с пользовательской инструкцией. При нажатии на кнопку “How to import files” становится доступным окно с подробным описанием процесса импорта файлов. При нажатии на кнопку “Notation of network elements” открывается окно с условными обозначениями элементов диаграммы.

На рисунке 3b изображен макет окна с подключением к удаленному серверу. Пользователю необходимо ввести имя хоста, имя пользователя и пароль.

Если подключение произошло успешно, пользователь видит интерфейс, макет которого представлен на рисунке 3c. Здесь пользователю необходимо выбрать удаленную директорию, откуда импортируются конфигурационные файлы, а также директорию на локальном компьютере, куда сохраняются эти файлы.



(a) Пользовательская инструкция

(b) Подключение к удаленному серверу

(c) Выбор удаленной и локальной директорий

Рис. 3: Макеты пользовательского интерфейса.

4.4 Используемые технологии

В качестве основного языка программирования для реализации используется Python [17]. Данное решение является требованием компании ООО «Мобил-Груп».

Для конвертации конфигурационных файлов была выбрана библиотека json-cfg [24]. Её методы позволяют обрабатывать синтаксис, который был описан в обзоре предметной области.

Для отслеживания изменений в файлах диаграмм использовалась библиотека watchdog [23].

Чтобы подключаться к удаленному серверу, используется протокол SSH. Был выбран именно он, так как по сравнению со своими аналогами (например, Telnet [21]) является более безопасным.

На основе SSH файлы с удаленного сервера, а также на него передаются через протокол SFTP [19]. Для этих целей используется Python-библиотека paramiko [14].

5 Описание реализации

В данном разделе приведены особенности реализации системы.

5.1 Описание графических обозначений элементов сети

Для построения диаграммы сети необходимо было выбрать фигуры, которые будут обозначать те или иные элементы сети. Ниже описываются обозначения, которые были выбраны из доступных в Draw.io фигур.

Для обозначения элемента eNodeB (eNB), который отвечает за настройку базовой станции, была выбрана фигура базовой станции (рис. 4a).

Для обозначения элементов, связанных с хранением и обработкой каких-либо данных, была выбрана фигура в виде цилиндра, которая обычно используется для обозначения баз данных (рис. 4b).

Чтобы обозначить элементы, которые не связаны непосредственно с данными, а отвечают за настройку каких-либо функций сети, используется фигура сервера, на котором изображена первая буква названия узла сети (рис. 4c)

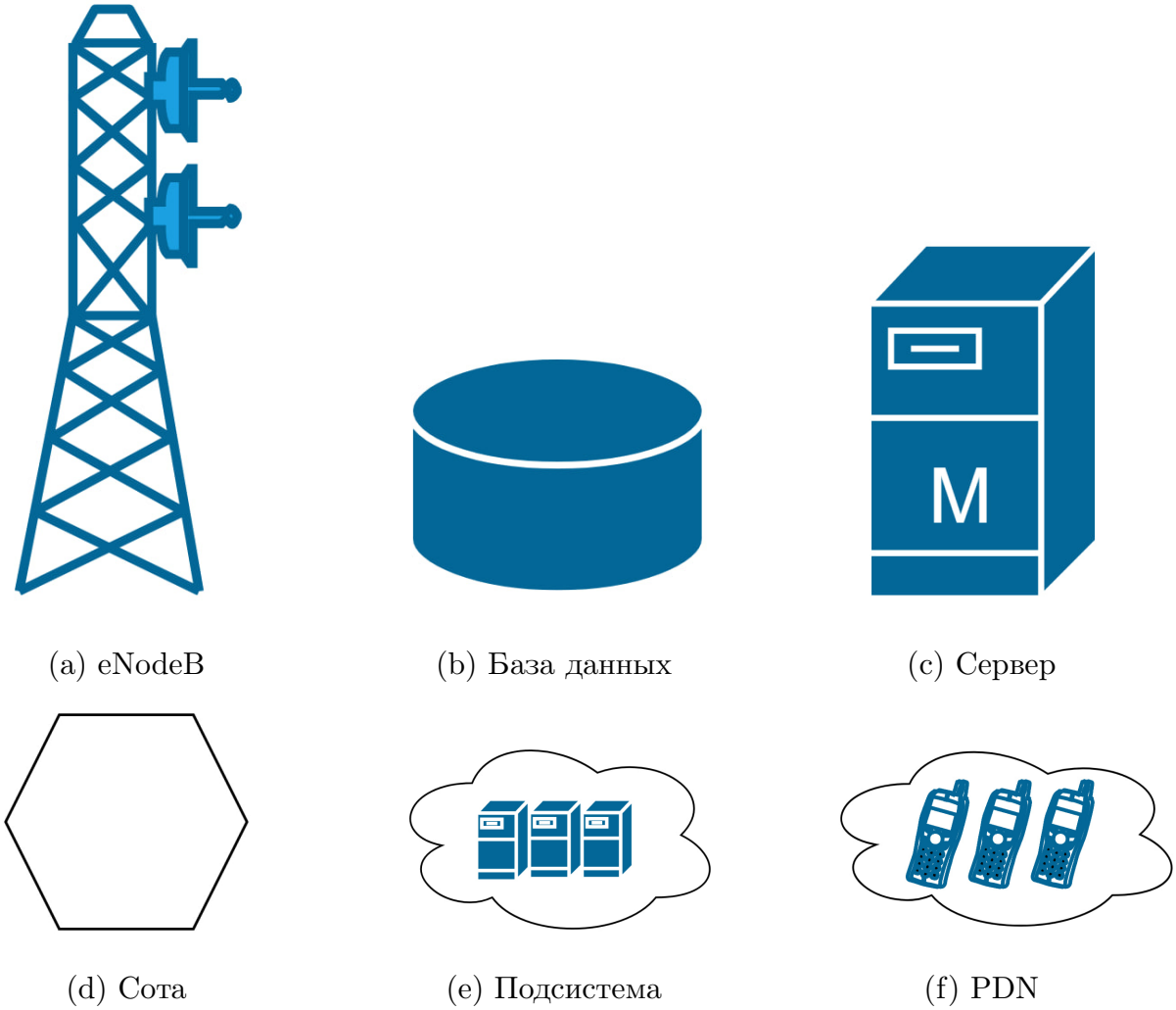
Соты, из которых состоит сеть, обозначаются правильными шестиугольниками, что является общепринятым обозначением (рис. 4d).

Если элемент сети является не узлом, а подсистемой или набором каких-либо других элементов, на схеме он обозначается в виде облака с серверами внутри (рис. 4e).

Для обозначения Public Data Networks (PDN), внутри которого содержится список доступных сетей, также используется облако. Однако в данном случае фигуры серверов заменяются фигурами телефонных трубок, которые обозначают пользовательские устройства (рис. 4f).

Если для одного из узлов необходимо отображать более двух параметров, вместо текстовых лейблов создается лист с параметрами. Изначально он находится в свернутом виде, а при раскрытии в каждой

строке записан отдельный параметр.



(a) eNodeB (b) База данных (c) Сервер

(d) Сота (e) Подсистема (f) PDN

Рис. 4: Обозначения элементов на диаграмме.

Помимо самих элементов на диаграмме необходимо обозначить связи между ними. Интерфейсы обозначаются прямыми линиями, соединяющими два компонента. Интерфейсы, которые отвечают за передачу данных, выделяются пунктиром, а те связи, по которым передаются сигналы — сплошные. К самой линии прикреплен текстовый лейбл, указывающий на название данного интерфейса. Также на диаграмме есть линии, состоящие из точек — мелкого пунктира. Они используются в том случае, когда необходимо указать, что между подсистемами существует связь, но без указания конкретного интерфейса.

По умолчанию цвет линии — черный. Если пользователь вносит изменения в параметры, затрагивающие данный интерфейс, автоматиче-

ски происходит изменение цвета в зависимости от статуса соединения (успешно — зеленый, неудачно — красный).

На рисунке 5 демонстрируется пример построенной диаграммы сети.

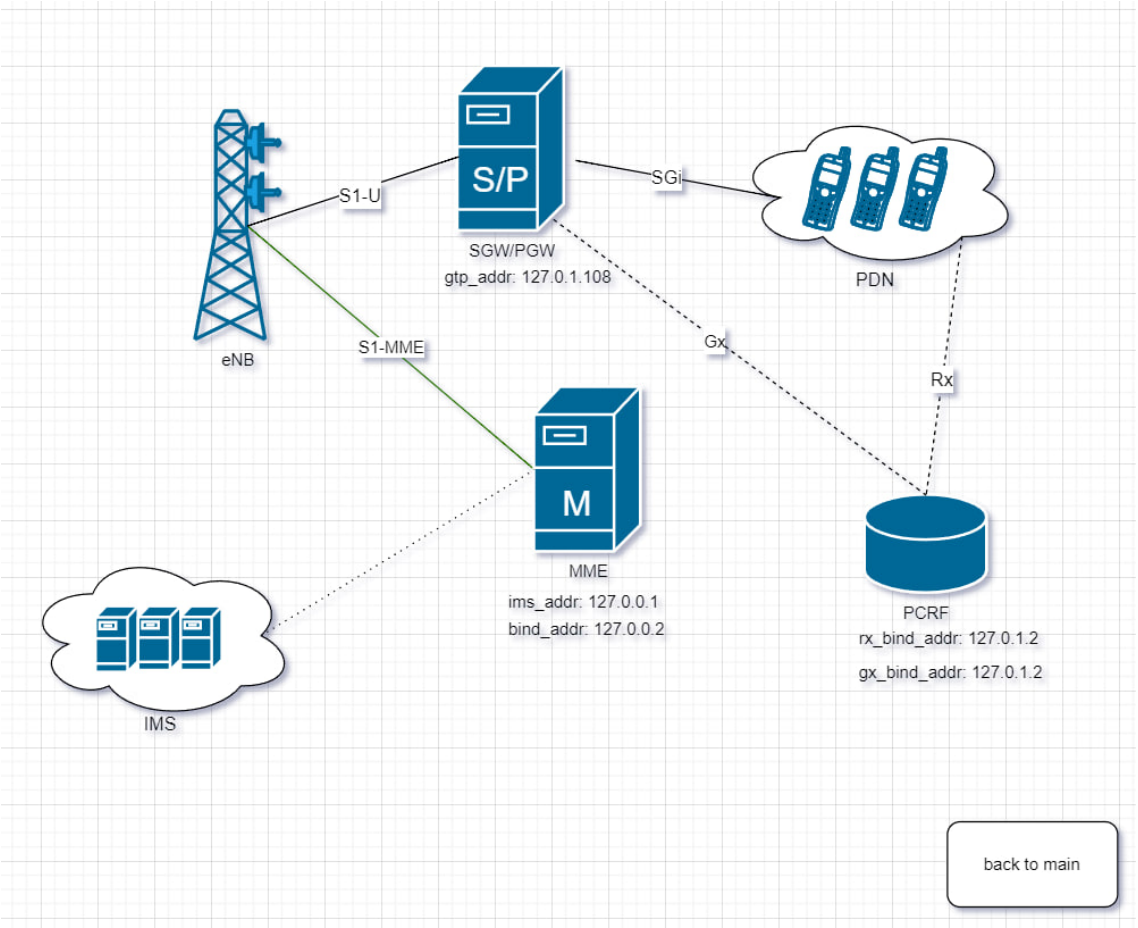


Рис. 5: Пример диаграммы в Draw.io

Кроме элементов, обозначающих части сети, на каждой странице диаграммы расположена кнопка «Back to main», к которой привязана гиперссылка на страницу с общей структурой сети. Данная кнопка не относится к устройству самой сети, однако обеспечивает пользователю удобную навигацию по частям диаграммы.

5.2 Импорт конфигурационных файлов

Прежде чем начать работать с сетью, необходимо импортировать конфигурационные файлы, которые хранятся в отдельной директории на удаленном сервере, где установлено ПО для связи с оборудованием.

Импорт происходит с помощью библиотеки paramiko [14].

После ввода пользователем данных для подключения к серверу, кнопка «Connect» становится активной, и после нажатия на нее создается объект класса SSHClient. К нему применяется метод `connect`, посредством которого устанавливается защищенное соединение с сервером.

Когда соединение установлено, в окне приложения отображается интерфейс с текстовыми полями для выбора локальной и удаленной директорий. Выбор локальной директории происходит из окна файловой системы компьютера пользователя, а удаленная директория выбирается в реализованном окне, которое является объектом класса ChoosingRemoteDirectoryView. Подгрузка списка директорий с удаленного сервера осуществляется при помощи метода `listdir_attr` библиотеки paramiko.

Далее, когда пользователь выбрал необходимые директории, кнопка «Create diagram» становится активной, и после нажатия на нее создается SFTP-сессия на основе открытого ранее SSH-соединения. Конфигурационные файлы выгружаются с сервера в выбранную директорию локального компьютера посредством метода `get`.

Аналогичным образом происходит передача файлов на сервер. В данном случае используется метод put класса SFTPClient.

5.3 Конвертация конфигурационных файлов в файлы диаграмм

Для парсинга конфигурационных файлов используется библиотека json-cfg [24].

Так как исходный JSON содержит вложенные параметры, результатом работы парсинга являются также вложенные объекты классов-наследников ConfigNode библиотеки json-cfg. Так, простой параметр представляется классом ConfigJSONScalar, параметр, у которого в значении содержится список — ConfigJSONArray, а параметр со вложенным словарем парсится в ConfigJSONObject.

При дальнейшей обработке полученных после парсинга объектов строится так называемый плоский JSON. В нем вложенные параметры записываются через точку. Так, например, следующий отрывок конфигурационного файла

```
{
    sctp_addr: ''127.0.0.1'',
    sip_addr: [
        {addr: ''10.4.0.104'', port_min: 5000, port_max: 20000},
        {addr: ''192.168.50.1'', port_min: 10000, port_max: 20000},
    ],
    trunk: {
            addr: ''10.4.0.221'',
        },
}
```

после конвертации преобразуется в словарь

```
{sctp_addr: ''127.0.0.1'',  sip_addr_1.addr: ''10.4.0.104'',
sip_addr_1.port_min: 500, sip_addr_1.port_max: 20000,
sip_addr_2.addr: ''192.168.50.1'',  sip_addr_2.port_min:
10000, sip_addr_2.port_max: 20000, trunk.addr: ''10.4.0.221''}
```

Далее по полученному словарю строится основная диаграмма сети, а также отдельные страницы с подсистемами сети.

Построение диаграммы с общей структурой сети, располагающейся на первой странице, реализовано с помощью класса `drawio_diagram` библиотеки N2G [13]. Методы данного класса позволяют создавать графы, не взаимодействуя напрямую с языком файлов Draw.io. Необходимо задать нужные имена узлов, стили, а также указать, между какими из узлов должны быть связи. Все эти данные подгружаются из описанного выше словаря. Помимо этого для каждого узла основной диаграммы создается пустая страница, которая позже будет заполнена более

подробным устройством данного компонента сети. Переход на данную страницу осуществляется по кликабельной ссылке, которая закреплена за каждым узлом основной диаграммы.

Диаграммы внутренних подсистем требуют отображения параметров элементов сети, которые представляются на диаграмме в виде текстовых лейблов. Их создание довольно проблематично с использованием описанной выше библиотеки, поэтому построение данных диаграмм реализовано с помощью библиотеки для работы с XML xml.etree.ElementTree [27]. Внесение параметров осуществляется посредством добавления новых объектов в существующий файл диаграммы. Данное действие позволяет реализовать метод `SubElement` из названной выше библиотеки.

После конвертации полученная в результате диаграмма разворачивается для пользователя в приложении Draw.io. Запуск Draw.io с диаграммой сети реализован с помощью библиотеки subprocess [25] в Python. В метод `Popen` в качестве параметров передается путь к exe-файлу редактора, а также сам XML-файл построенной ранее диаграммы.

5.4 Изменение параметров на диаграмме

Для того, чтобы параметры, которые меняет пользователь на диаграмме, записывались в конфигурационные файлы, необходимо отслеживать изменения в файле, содержащем диаграмму. Для этого используется библиотека watchdog [23].

Создается объект класса Observer — наблюдатель, у которого в параметрах путь к директории с XML-файлом диаграммы. Наблюдатель непрерывно отслеживает файл с диаграммой и в случае изменений вызывает обработчик событий — объект класса FileHandler. Данный обработчик внутри своего метода `on_modified` вызывает метод `write_to_cfg`, который парсит диаграмму, получая из нее измененные параметры в виде словаря с плоским JSON, и после чего записывает данные параметры в исходный конфигурационный файл.

5.5 Изменение цвета связей в зависимости от статуса соединения

После получения ответа необходимо поменять цвет соединений на диаграмме, чтобы подсказать пользователю, верны ли его изменения в параметрах.

Изменение цвета происходит в методе `change_color` при помощи библиотеки xml.etree.ElementTree. Данный метод парсит XML-файл, находит необходимую линию соединения, после чего меняет значение атрибута, отвечающего за цвет линии. Если после изменения параметров соединение по данному интерфейсу успешно, цвет линии меняется на зеленый, иначе — на красный.

Если в файле, который открыт в Draw.io, происходят изменения, в углу всплывает сообщение, которое говорит пользователю о том, что файл был изменен (рис. 6). При нажатии на данное сообщение происходит синхронизация, и изменения подгружаются на диаграмму.

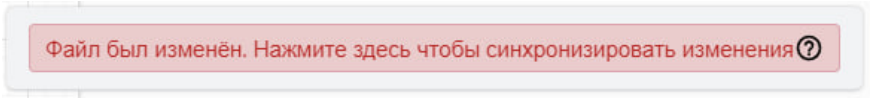


Рис. 6: Сообщение об изменениях в файле.

6 Тестирование и апробация

6.1 Модульное тестирование

Чтобы протестировать систему, были реализованы модульные тесты, покрывающие следующие функции.

- Функция конвертации конфигурационных файлов: здесь результат работы функции сравнивался с вручную созданными объектами класса Network.
- Функция, которая по диаграмме создает конфигурационные файлы. В тестах результат работы функции сравнивался с прописанными вручную конфигурационными файлами с нужными параметрами.
- Функция создания диаграммы сети. Результат работы функции сравнивался с диаграммами, построенными вручную.

Для тестирования была выбрана библиотека unittest [26], так как она по умолчанию включена в стандартную библиотеку Python, не требует установки дополнительных пакетов и подходит для случаев, когда требуются базовые возможности тестирования.

6.2 Апробация

Для того, чтобы оценить удобство использования инструмента, было принято решение провести апробацию. Участниками апробации стали восемь человек, ранее не знакомых с инструментом. Им было предложено выполнить пользовательское задание, которое составлялось таким образом, чтобы по максимуму покрыть всю реализованную функциональность приложения.

После выполнения инструкций участникам апробации было предложено дать обратную связь, посредством ответа на вопросы, основанные на шкале System Usability Scale (SUS) [20]. Данная анкета включает в

себя 10 вопросов, в каждом из которых дано утверждение, касающееся использования продукта, а также предложено 5 вариантов ответа от «Полностью согласен» до «Совершенно не согласен».

После сбора ответов был посчитан итоговый балл, который составил 70, что говорит о том, что пользоваться инструментом достаточно удобно.

В процессе анализа ответов были выявлены некоторые недостатки. Пользователями было замечено, что редактирование параметров на диаграмме недостаточно удобное, так как при двойном клике выделяется не только значение параметра, но и его имя.

Также было отмечено, что при работе с диаграммой было бы удобно иметь рядом условные обозначения (на данный момент они описаны в пользовательской инструкции), без переключения в основное окно приложения.

Первый недочет может быть исправлен в следующих версиях инструмента путем разделения текстового лейбла с параметром на два: для имени, и для значения. Для того, чтобы исправить второй, отмеченный выше недостаток, необходимо добавить на диаграмму отдельную страницу, где будут прописаны обозначения.

Заключение

В ходе выполнения работы получены следующие результаты.

- Были сформулированы основные требования к инструменту.
- Проведен обзор существующих решений.
- Проведен обзор используемых технологий и инструментов.
- Разработана архитектура системы.
- Выполнена реализация инструмента в виде Desktop-приложения согласно требованиям и архитектуре.
- Проведена апробация по шкале System Usability Scale.
- Работа была представлена на конференции «Современные технологии в теории и практике программирования» [30], а ее тезисы опубликованы в сборнике материалов к этой конференции [31].

Исходный код является собственностью ООО «Мобил-груп» и не может быть предоставлен.

Список литературы

[1] 10-Strike LANState. — URL: https://www.10-strike.ru/lanstate/ (дата обращения: 2024-04-27).

[2] Algorius Net Viewer. — URL: https://algorius.ru/ (дата обращения: 2024-04-27).

[3] Amarisoft. — URL: https://www.amarisoft.com/ (дата обращения: 2024-05-01).

[4] Draw.io. — URL: https://www.drawio.com (дата обращения: 2023-12-15).

[5] Eclipse GMP. — URL: https://eclipse.dev/modeling/gmp/ (дата обращения: 2024-04-24).

[6] Figma. — URL: https://www.figma.com (дата обращения: 2024-04-06).

[7] JSON. — URL: https://www.json.org/json-en.html (дата обращения: 2024-04-06).

[8] Kivy. — URL: https://kivy.org/ (дата обращения: 2024-04-06).

[9] Lucidchart. — URL: https://www.lucidchart.com/pages/ (дата обращения: 2023-12-15).

[10] M. Hicham N. Abghour M. Ouzzif. 4G system: Network Architecture and Performance // International Journal of Innovatice Research in Advanced Engineering. — 2015. — Vol. 2, no. 4. — P. 215–220.

[11] Microsoft. — URL: https://www.microsoft.com/ru-ru/ (дата обращения: 2024-05-15).

[12] Microsoft Visio. — URL: https://www.microsoft.com/ru-ru/microsoft-365/visio/flowchart-software (дата обращения: 2023-12-15).

[13] N2G. — URL: https://n2g.readthedocs.io/en/latest/diagram_plugins (дата обращения: 2024-04-28).

[14] Paramiko. — URL: https://www.paramiko.org/ (дата обращения: 2024-04-10).

[15] PyQT. — URL: https://pypi.org/project/PyQt6/ (дата обращения: 2024-04-06).

[16] PySide. — URL: https://pypi.org/project/PySide/ (дата обращения: 2024-04-06).

[17] Python. — URL: https://www.python.org/ (дата обращения: 2023-12-15).

[18] Raghunandan Krishnamurthy. Introduction to Wireless Communications and Networks // International Journal of Innovatice Research in Advanced Engineering. — 2017.

[19] SFTP. — URL: https://www.sftp.net/ (дата обращения: 2024-04-10).

[20] System Usability Scale. — URL: https://digital.gov/topics/usability/ (дата обращения: 2024-04-28).

[21] Telnet. — URL: https://ru.wikipedia.org/wiki/Telnet (дата обращения: 2024-04-06).

[22] Tkinter. — URL: https://docs.python.org/3/library/tkinter.htmll (дата обращения: 2024-04-06).

[23] Watchdog. — URL: https://pypi.org/project/watchdog/ (дата обращения: 2024-04-06).

[24] json-cfg. — URL: https://pypi.org/project/json-cfg/ (дата обращения: 2023-12-15).

[25] subprocess. — URL: https://docs.python.org/3/library/subprocess.html (дата обращения: 2023-12-15).

[26] unittest. — URL: https://docs.python.org/3/library/unittest.html (дата обращения: 2024-04-06).

[27] xml.etree.ElementTree. — URL: https://docs.python.org/3/library/xml.etree.elementtree.html (дата обращения: 2023-12-15).

[28] yEd. — URL: https://www.yworks.com/products/yed (дата обращения: 2024-04-24).

[29] АРГУС СИРИУС. — URL: https://argustelecom.ru/produktyi/vzaimodejstvie-s-oborudovaniem.html (дата обращения: 2023-12-15).

[30] Конференция «Современные технологии в теории и практике программирования». — URL: https://hsse.spbstu.ru/sovremennue_tehnologii/ (дата обращения: 2024-05-22).

[31] Никитина А.М. Разработка комплекса программ для графического программирования оборудования 4G // Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых. — 2024. — P. 78–79. — URL: https://elibrary.ru/item.asp?id=66237522.