федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

09.03.01 Информатика и вычислительная техника

код и наименование направления подготовки

**ОТЧЕТ**

по преддипломной практике

на кафедре прикладной математики и кибернетики

Выполнил:

студент гр. ПБ-15 \_\_\_\_\_\_\_\_\_\_\_\_\_ /Иванов И.И./

«12» апреля 2023 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_

оценка

Руководитель практики от университета

к.т.н., доцент кафедры ПМиК \_\_\_\_\_\_\_\_\_\_\_\_\_ /Ситняковская Е.И./

«12» апреля 2023 г.

Новосибирск 2023

**План-график проведения производственной практики**

Тип практики: преддипломная практика

Способ проведения практики: стационарная

Форма проведения практики: дискретно по видам проведения практик

Выдано обучающемуся Иванову Ивану Ивановичу

Фамилия Имя Отчество студента

Направление: 09.03.01 – Информатика и вычислительная техника

Код – Наименование направления

Профиль: Программное обеспечение средств вычислительной техники и автоматизированных систем

Группа ПБ-15

Тема ВКР: **Разработка приложения для автоматической настройки ИТ-инфраструктуры предприятия**

Содержание практики

|  |  |
| --- | --- |
| Наименование видов деятельности | Дата  (начало – окончание) |
| Постановка задачи на практику, определение конкретной индивидуальной темы, формирование плана работ | 11.03.2023 |
| Работа с библиотечными фондами, сбор и анализ материалов по теме практики | 12.03.2023-15.03.2023 |
| Выполнение работ в соответствии с составленным планом:  1. Общая характеристика и специфика рода деятельности предприятия  2. Основные технологии, определения и понятия разработки программного обеспечения и программирования  3. Анализ существующих решений  4. Описание средств разработки, применяемые для создания программного обеспечения  5. Общее описание работы приложения. Разработка структуры приложения | 15.03.2023-18.03.2023  18.03.2023-21.03.2023  21.03.2023-24.03.2023  25.03.2023-28.03.2023  29.03.2023-07.04.2023 |
| Анализ полученных результатов и произведенной работы, составление отчета по практике | 08.04.2023 – 12.04.2023 |

Согласовано:

Руководитель практики от СибГУТИ \_\_\_\_\_\_\_\_\_\_\_\_ /Ситняковская Е.И./

Содержание

[Задание на преддипломную практику 4](#_Toc183707545)

[Введение 5](#_Toc183707546)

[1. Общая характеристика предприятия 7](#_Toc183707547)

[2. Основные технологии, определения и понятия разработки программного обеспечения и программирования 9](#_Toc183707548)

[3. Анализ существующих решений 12](#_Toc183707549)

[4. Описание средств разработки, применяемые для создания программного обеспечения 14](#_Toc183707550)

[5. Общее описание работы приложения. Разработка структуры приложения. 17](#_Toc183707551)

[Заключение 28](#_Toc183707552)

[Список использованных источников 29](#_Toc183707553)

[Приложение 1 30](#_Toc183707554)

[Приложение 2 31](#_Toc183707555)

## **Задание на преддипломную практику**

Целью преддипломной практики являлось изучение области деятельности предприятия, нормативно-правовой информации, регулирующей деятельность предприятия, с последующей выработкой программного решения, для автоматизации технологических процессов настройки ИТ инфраструктуры предприятия заказчика в разрезе требований информационной безопасности.

Основными заказчиками предприятия выступают государственные учреждения. В связи с родом их деятельности, у данных учреждений существуют обязательства по соблюдению требований информационной безопасности.

Основными нормативно-правовыми актами регулирующими требования информационной безопасности являются Приказ ФСТЭК РФ от 11.02.2013 № 17 «Об утверждении требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах», Приказ ФСТЭК РФ от 14.03.2014 №31 "Об утверждении Требований к обеспечению защиты информации в автоматизированных системах управления производственными и технологическими процессами на критически важных объектах, потенциально опасных объектах, а также объектах, представляющих повышенную опасность для жизни и здоровья людей и для окружающей природной среды", Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации», Приказ ФАПСИ РФ от 13 июня 2001 г. № 152 «Об утверждении Инструкции об организации и обеспечении безопасности хранения, обработки и передачи по каналам связи с использованием средств криптографической защиты информации с ограниченным доступом, не содержащей сведений, составляющих государственную тайну», Федеральный закон от 26 июля 2017 г. N 187-ФЗ "О безопасности критической информационной инфраструктуры Российской Федерации".

В рамках преддипломной практики выполнялись следующие задачи:

1. Общая характеристика и специфика рода деятельности предприятия.

2. Основные технологии, определения и понятия разработки программного обеспечения и программирования

3. Анализ существующих решений

4. Описание средств разработки, применяемые для создания программного обеспечения

## **Введение**

На сегодняшний день, с развитием компьютерной техники широкое развитие получают информационные технологии, разрабатываемые для автоматизации прикладных задач. Использование информационных технологий позволяет значительно повысить эффективность не только работы специалистов, но и бизнеса в целом, а также сократить временные затраты на выполнение поставленных задач.

Преддипломная практика является составной частью учебного процесса подготовки квалифицированных специалистов.

Актуальность прохождения студентом практики заключается в возможности закрепить теоретические знания и приобрести умения и навыки практической работы по специальности.

Наиболее общими задачами являются:

* изучение структуры предприятия, организации, основных функций производственных, экономических и управленческих;
* обзор задач, решаемых в подразделении прохождения практики, связанных с проблематикой направления "информатика и вычислительная техника";
* разработка приложения для автоматизации процесса настройки ИТ инфраструктуры предприятия заказчика.

Автоматизация ИТ-процессов — это использование ПО и систем для автоматизированного выполнения процессов и задач вместо длительного выполнения работы ИТ-специалистами вручную.

Автоматизация ИТ-процессов ускоряет предоставление ИТ-инфраструктуры и приложений благодаря автоматизации действий, которые раньше выполнялись вручную. С помощью ПО настраиваются повторяющиеся инструкции, процессы или политики, которые помогают экономить время ИТ-специалистов и уделять больше внимания важным стратегическим задачам.

Программное обеспечение для автоматизации ИТ-процессов может выполнять широкий спектр простых и сложных ИТ-задач. Например, ПО может инициировать действие (или последовательность действий), с помощью автоматизации можно создавать шаблоны и схемы сетей и систем безопасности, а также настраивать приложения и инициализировать готовую к работе инфраструктуру.

Цель данной работы является определение технологического процесса, требующего автоматизации, и разработка соответствующего программного обеспечения.

Объектом исследования является предприятие ООО «МАСКОМ-Техлайн».

Предметом исследования является рабочий процесс настройки инфраструктуры заказчика согласно требованиям информационной безопасности, регламентируемые законодательными актами.

Структура отчета состоит из введения, основной части, заключения и списка использованных источников. В работе описываются теоретические основы разработки программного обеспечения, основная характеристика деятельности организации, разработана концепция приложения, определен состав программного обеспечения и выбраны пути решения поставленной задачи.

## **1. Общая характеристика предприятия**

Преддипломная практика проходила на базе предприятия ООО «МАСКОМ-Техлайн», входящего в холдинг МАСКОМ-Восток.

Холдинг МАСКОМ Восток - эксперт в области защиты информации, системной интеграции, интеллектуальных систем, технической защиты объектов, проектирования и строительства защищенных зданий и сооружений и систем оповещения населения.

Компании в составе Холдинга:

ООО «ДСБЦИ «МАСКОМ» - центр компетенций в области защиты государственной тайны, проектирования и строительства систем технической защиты объектов, проектирования и строительства защищенных зданий и сооружений. Производитель специализированного оборудования.

ООО «МАСКОМ-Техлайн» - центр компетенций в области защиты конфиденциальной информации, системной интеграции, создания защищенных информационно-телекоммуникационных сетей.

ООО «МАСКОМ-Инстрой» - центр компетенций в области проектирования и строительства систем оповещения населения о чрезвычайных ситуациях. Разработчик и производитель программно-аппаратного комплекса «ГАУУС-М»

ООО «МАСКОМ-АМУР» - многопрофильная компания, обладающая основными компетенциями Холдинга МАСКОМ Восток, осуществляющая деятельность в Благовещенске и Амурской области

На базе ООО «МАСКОМ-Техлайн» действует центр мониторинга и управления информационной безопасностью MSOC – SOC, вся технологическая база которого развернута в г. Хабаровске.

ООО «МАСКОМ-Техлайн», является центром компетенций в области защиты информации, системной интеграции, создания защищенных информационно-телекоммуникационных систем.

Компании холдинга «МАСКОМ Восток» предоставляют следующие услуги и решения:

1. Аттестация объектов информатизации
2. Защита информационных систем
3. Ведение секретного делопроизводства и экспертиз
4. Поставка специального оборудования
5. Аттестация информационных систем
6. Защита персональных данных
7. Защита критической информационной инфраструктуры
8. Проектирование и создание информационных систем
9. Системная и сетевая интеграция
10. Проектирование комплексных систем безопасности

Компания «МАСКОМ-Техлайн» является лицензиатом ФСБ России и так же партнёром ведущих производителей ИТ-оборудования, средств защиты информации, системного и прикладного программного обеспечения, таких как Huawei, Лаборатория Касперского, ООО «Код Безопасности», Инфотекс, АО «НПО РусБИТех» (разработчик системного ПО «Astra Linux Special Edition») ООО «РЕД СОФТ» (разработчик системного ПО «RedOS») и др.

Организационная структура в ООО «МАСКОМ-Техлайн» делится на четыре департамента: Департамент информационных технологий, Департамент мониторинга информационной безопасности, Департамент проектирования и аттестации информационных систем, Департамент по работе с клиентами.

Непосредственно департамент информационных технологий, специализируется на внедрении комплексных информационных систем, также на строительных и инженерных решениях для объектов крупных государственных заказчиков.

## **2. Основные технологии, определения и понятия разработки программного обеспечения и программирования**

Как в любом виде деятельности в программировании имеется своя технология: это знания, правила, навыки и инструменты, позволяющие получать гарантированный качественный результат. Но само по себе соблюдение ряда правил не дает гарантию качества результата. Это объясняется спецификой программирования. Программирование — это не наука, где знание формулы позволяет однозначно решить задачу, подставив в нее исходные данные и получив результат, программирование больше творческий процесс, пути решения одной и той же задачи может отличатся при реализации решения разными специалистами. Это подтверждает и существование понятия как «стиль программирования», который более или менее чётко прослеживается в работах разных программистов и носит субъективный характер. Исходя из вышесказанного, правила программирования необходимо соблюдать не столько на бумаге, сколько в голове. То есть технология программирования - это скорее способ организации процесса обдумывания программы, нежели ее записи.

Рассмотрим наиболее известные из технологий:

* метод "северо-западного" угла (имеется в виду лист бумаги или экран дисплея). Программа пишется сразу от начала до конца, без использования каких-либо общих принципов;
* технология структурного программирования, в ней предполагается придерживаться принципов модульности, нисходящего и пошагового проектирования программ, одновременного проектирования программ и структур данных.
* технология объектного программирования: связана с использованием при проектировании программы понятий объектов и их классов.

В традиционной технологии программирования взаимоотношения процедуры – данные имеют более-менее свободный характер, причем процедуры (функции) являются ведущими в этой связке: как правило, функция вызывает функцию, передавая данные друг другу по цепочке. Соответственно, технология структурного проектирования программ, прежде всего, уделяет внимание разработке алгоритма.

В технологии ООП взаимоотношения данных и алгоритма имеют более регулярный характер: во-первых, класс (базовое понятие этой технологии) объединяет в себе данные (структурированная переменная) и методы (функции). Во-вторых, схема взаимодействия функций и данных принципиально иная. Метод (функция), вызываемый для одного объекта, как правило, не вызывает другую функцию непосредственно. Для начала он должен иметь доступ к другому объекту (создать, получить указатель, использовать внутренний объект в текущем и т.д.), после чего он уже может вызвать для него один из известных методов. Таким образом, структура программы определяется взаимодействием объектов различных классов между собой. Как правило, имеет место иерархия классов, а технология ООП иначе может быть названа как программирование "от класса к классу".

Модульное программирование. Принцип модульности формулируется как требование разработки программы в виде совокупности модулей (функций). При этом разделение на модули должно носить не механический характер, а исходить из логики программы:

* размер модуля должен быть ограничен;
* модуль должен выполнять логически целостное и завершенное действие;
* модуль должен быть универсальным, то есть по возможности параметризованным: все изменяемые характеристики выполняемого действия должны передаваться через параметры;
* входные параметры и результат модуля желательно передавать не через глобальные переменные, а через формальные параметры и результат функции.

Еще одной, но уже физической единицей программы является текстовый файл, содержащий некоторое количество функций и определений типов данных и переменных. Модульное программирование на уровне файлов – это возможность разделить полный текст программы на несколько файлов, транслировать их независимо друг от друга.

Принцип модульности распространяется не только на программы, но и на данные: любой набор параметров, характеризующих логический или физический объект, должен быть представлен в программе в виде единой структуры данных (структурированной переменной).

Нисходящее программирование. Нисходящее проектирование программы заключается в том, что разработка идет от общей неформальной формулировки некоторого действия программы на естественном языке, "от общего к частному", к замене ее одной из трех формальных конструкций языка программирования:

* простой последовательности действий;
* конструкции выбора или оператора if;
* конструкции повторения или цикла.

В записи алгоритма это соответствует движению от внешней (объемлющей) конструкции к внутренней (вложенной). Эти конструкции также могут содержать в своих частях неформальное описание действий, то есть нисходящее проектирование по своей природе является пошаговым. Отметим основные свойства такого подхода:

* первоначально программа формулируется в виде некоторого неформального действия на естественном языке;
* первоначально определяются входные параметры и результат действия;
* очередной шаг детализации не меняет структуру программы, полученную на предыдущих шагах;
* если в процессе проектирования получаются идентичные действия в различных ветвях, то это означает необходимость оформления этого действия отдельной функцией;
* необходимые структуры данных проектируются одновременно с детализацией программы.

Пошаговое программирование. Нисходящее проектирование по своей природе является пошаговым, ибо предполагает каждый раз замену одной словесной формулировки на единственную конструкцию языка. Но в процессе разработки программы могут быть и другие шаги, связанные с детализацией самой словесной формулировки в более подробную.

То, что этот принцип выделен отдельно, говорит о необходимости предотвратить соблазн детализации программы сразу от начала до конца и развивать умение выделять и сосредоточивать внимание на главных, а не второстепенных деталях алгоритма.

Структурное программирование. При нисходящей пошаговой детализации программы необходимые для работы структуры данных и переменные появляются по мере перехода от неформальных определений к конструкциям языка, то есть процессы детализации алгоритма и данных идут параллельно. Однако это касается, прежде всего, отдельных локальных переменных и внутренних параметров. С самой же общей точки зрения предмет (в нашем случае - данные) всегда первичен по отношению к выполняемым с ним действиям (в нашем случае - алгоритм). Поэтому на самом деле способ организации данных в программе более существенно влияет на ее структуру алгоритма, чем что-либо другое, и процесс проектирования структур данных должен опережать процесс проектирования алгоритма их обработки.

Структурное программирование – это модульное нисходящее пошаговое проектирование алгоритма и структур данных.

## **3. Анализ существующих решений**

В связи с процессами, протекающими на отечественном ИТ рынке, в сферах программного и технического обеспечения, нововведения в законодательной безе, особенно актуальным является вопрос об импортозамещения проприаритарного иностранного программного обеспечения. На отечественном рынке, наиболее большими из представленных производителей системного программного обеспечения является АО «НПО РусБИТех» и ООО «РЕД СОФТ». Ниже рассмотрены решения, данных компаний, направленных на централизацию и автоматизацию настройки ИТ инфраструктуры предприятий.

**Программное обеспечение ALD Pro.**

Программный комплекс «ALD Pro», предназначен для централизованного администрирования ресурсов домена в компьютерных сетях под управлением «Операционной системы специального назначения «Astra Linux Special Edition посредством графического интерфейса. Средой функционирования «ALD Pro» является ОС Astra Linux. «ALD Pro» интегрировано с комплексом средств защиты информации ОС Astra Linux, является прикладным программным обеспечением и не реализует самостоятельно функции защиты информации. Областью применения «ALD Pro» является автоматизация деятельности системных администраторов в рамках эксплуатации ИТ-инфраструктуры на базе ОС Astra Linux.

«ALD Pro» предназначен для централизованного управления ресурсами под управлением ОС Astra Linux и может использоваться в организациях различного масштаба. «ALD Pro» предоставляет графический интерфейс для реализации следующих возможностей ОС Astra Linux: - управление учетными записями пользователей и групп пользователей (создание, удаление, изменение параметров, изменение состава пользователей в группах); - управление компьютерами и группами компьютеров (включение в домен, исключение из домена, управление параметрами; создание, удаление, изменение состава компьютеров в группах); - управление организационной структурой подразделений (создание и удаление подразделений, выстраивание иерархической структуры); - управление групповыми политиками (создание, удаление, изменение параметров, назначение на организационные подразделения); - управление и настройка программного обеспечения

**Программное обеспечение РЕД АДМ**

Система централизованного управления ИТ-инфраструктурой «РЕД АДМ» является программным продуктом, полностью разработанным компанией «РЕД СОФТ». РЕД АДМ имеет модульную структуру и агрегирует в себе множество модулей администрирования различного назначения.

РЕД АДМ в качестве механизма централизованного управления использует Ansible, что позволяет выполнять, в частности, такие функции как:

* централизованное распространение клиентского приложения, VNC, SSH-ключей;
* проверка статусов подключенных узлов;
* ручное выполнение конфигураций.

Представленные выше решения являются высокотехнологичными комплексными продуктами, требующие обучения и наличие специализированного опыта работы с данными продуктами. Существенным недостатком для компании, является высокая дороговизна дынных продуктов и привязка уже настроенного решения под определённую информационную структуру и организацию.

## **4. Описание средств разработки, применяемые для создания программного обеспечения**

Bash (от англ. Bourne again shell, каламбур «Born again» shell — «возрождённый» shell) — усовершенствованная и модернизированная вариация командной оболочки Bourne shell. Одна из наиболее популярных современных разновидностей командной оболочки UNIX. Особенно популярна в среде Linux, где она часто используется в качестве предустановленной командной оболочки.

Представляет собой командный процессор, работающий, как правило, в интерактивном режиме в текстовом окне. Bash также может читать команды из файла, который называется скриптом (или сценарием). Как и все Unix-оболочки, он поддерживает автодополнение имён файлов и каталогов, подстановку вывода результата команд, переменные, контроль над порядком выполнения, операторы ветвления и цикла. Ключевые слова, синтаксис и другие основные особенности языка были заимствованы из sh. Другие функции, например, история, были скопированы из csh и ksh. Bash в основном соответствует стандарту POSIX, но с рядом расширений.

Название «bash» является акронимом от англ. Bourne-again-shell («ещё-одна-командная-оболочка-Борна») и представляет собой игру слов: Bourne-shell — одна из популярных разновидностей командной оболочки для UNIX (sh), автором которой является Стивен Борн (1978), усовершенствована в 1987 году Брайаном Фоксом. Фамилия Bourne (Борн) перекликается с английским словом born, означающим «родившийся», отсюда: рождённая-вновь-командная оболочка.

BASH является не только командной оболочкой, это еще и скриптовый язык программирования. Такие языки:

1. Интерпретируются, а не компилируются;

2. Оперируют готовыми программами или высокоуровневыми командами;

3. Интегрированы в командную оболочку или ОС.

Языки программирования вообще и сценарные языки в частности могут быть классифицированы множеством различных способов.

В плане быстродействия скриптовые языки можно разделить на языки динамического разбора (sh, COMMAND.COM) и предварительно компилируемые (Perl). Языки динамического разбора считывают инструкции из файла программы минимально требующимися блоками, и исполняют эти блоки, не читая дальнейший код. Предкомпилируемые языки транслируют всю программу в байт-код и затем исполняют его. Некоторые скриптовые языки имеют возможность компиляции программы «на лету» в машинный код (т. н. JIT-компиляция).

По применению языки можно разделить на три типа:

* командно-сценарные;
* прикладные сценарные;
* универсальные сценарные.

**Командно-сценарные языки**

Появились ещё в 1960-х годах для управления заданиями в операционных системах. Из языков того времени наиболее известен JCL для OS/360. В этот класс входят языки пакетной обработки и языки командных оболочек, например, bash, sh, csh для Unix. Эти языки чаще всего используются в пакетном режиме обработки.

**Встроенные (прикладные сценарные) языки**

Сценарные языки этого типа начали появляться в 1980-е годы, когда на промышленных персональных компьютерах стало возможным интерактивное общение с ОС. В клиент-серверной архитектуре такие языки работали в клиентской части программного обеспечения. В этот класс входят, например, AutoLISP, ECMAScript и его диалекты (JScript, JavaScript, ActionScript), VBA.

**Языки общего назначения**

Этот тип сценарных языков наиболее известен (особенно в применении к веб-программированию). Языки этого типа стали возникать с 1990-х годов В этот класс входят, например, Perl, PHP, Python, Ruby.

Как и большинство языков программирования bash поддерживает:

* переменные;
* типы данных
* условия и циклы
* операторы ветвление
* функции

Zenity — утилита, которая позволяет выводить на экран диалоговые окна GTK+ из командной строки и скриптов командной оболочки.

Рассмотрим функциональные возможности данной программы. Имеется десять видов диалогов.

1. Calendar – календарь (date-picker).
2. Entry – однострочное текстовое поле ввода.
3. Text-info – диалог отображения многострочной текстовой информации, который может применяться и как поле ввода. Хотя, по всей видимости, это не является его основным назначением.
4. Error – сообщение об ошибке.
5. Info – сообщение общего характера.
6. Warning – предупреждение.
7. Question – вопросительное сообщение с возможностью ввода утвердительного или отрицательного ответа.
8. File-selection – выбор файла.
9. List – список с возможностью выбора и редактирования его элементов.
10. Progress – Progress bar dialog. Отображает статус выполнения текущей операции.

Zenity не только показывает окно, но и выводит в стандартные вывод то, что пользователь в окне сделал. Данный вывод можно перехватить, и передать в запрограммированные функции, присвоить переменным или передать в циклы на выполнение

Посредством запуска программы с соответствующими параметрами, можно реализовать требуемый интерфейс.

При завершении выполнения будут выведены результирующие данные в stdout или же код будет просто возвращён в stderr.

## **5. Общее описание работы приложения. Разработка структуры приложения.**

Файловая структура приложения представлена на рисунках 5.1, 5.2, 5.3

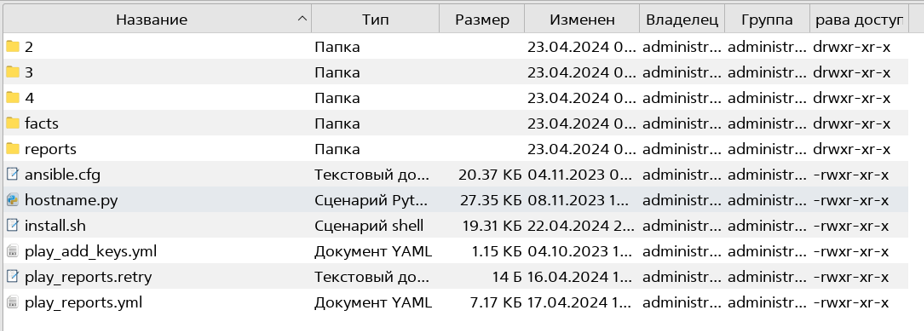


Рисунок 5.1 – Файловая структура приложения

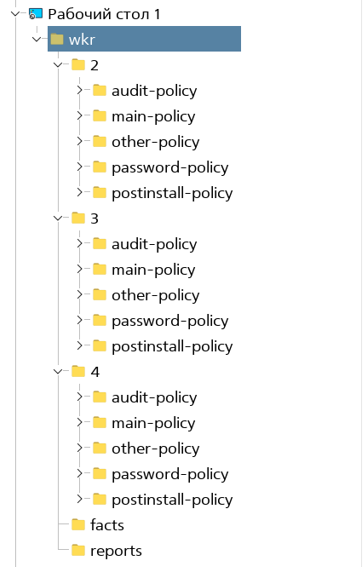


Рисунок 5.2 – Файловая структура приложения

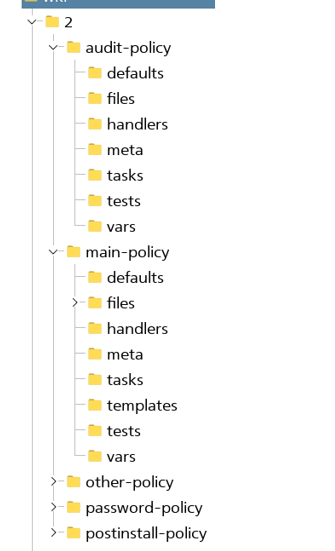


Рисунок 5.3 – Файловая структура приложения

Всего в приложении 9 диалоговых окон:

* Стартовое окно запуска приложения (рисунок 5.4)
* Диалоговое окно выбора значимости информации, обрабатываемой в ИТ инфраструктуре предприятия (рисунок 5.5)
* Диалоговое окно ввода диапазона IP адресов или адресации сети (рисунок 5.6)
* Диалоговое окно вывода результатов сканирования (рисунок 5.7)
* Диалоговое окно ввода данных для аутентификации пользователя (рисунок 5.8)
* Диалоговое окно выбора настроек ИТ инфраструктуры (рисунок 5.9)
* Диалоговое окно выбора места нахождения репозитория (рисунок 5.10)
* Диалоговое окно ввода IP адреса нахождения репозитория (сетевого или локального) (рисунок 5.11)
* Диалоговое окно подтверждения запуска настройки ИТ инфраструктуры (рисунок 5.12)

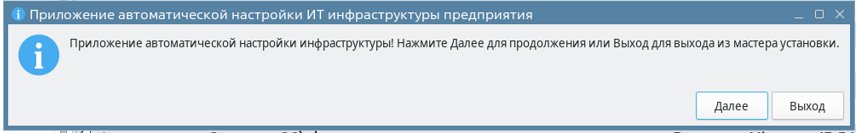


Рисунок 5.4 – Стартовое окно запуска приложения

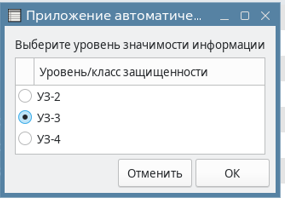


Рисунок 5.5 – Диалоговое окно выбора значимости информации

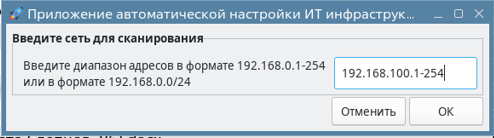


Рисунок 5.6 – Диалоговое окно ввода диапазона IP адресов или адресации сети

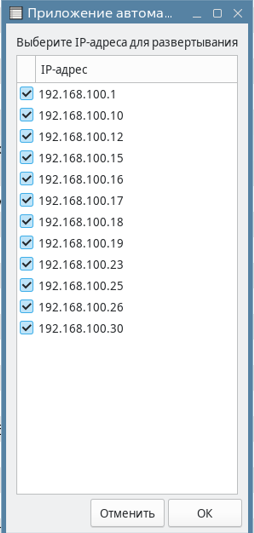


Рисунок 5.7 – Диалоговое окно вывода результатов сканирования

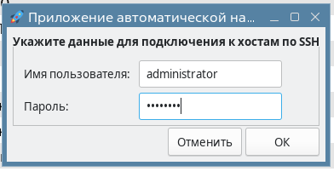


Рисунок 5.8 – Диалоговое окно ввода данных для аутентификации пользователя

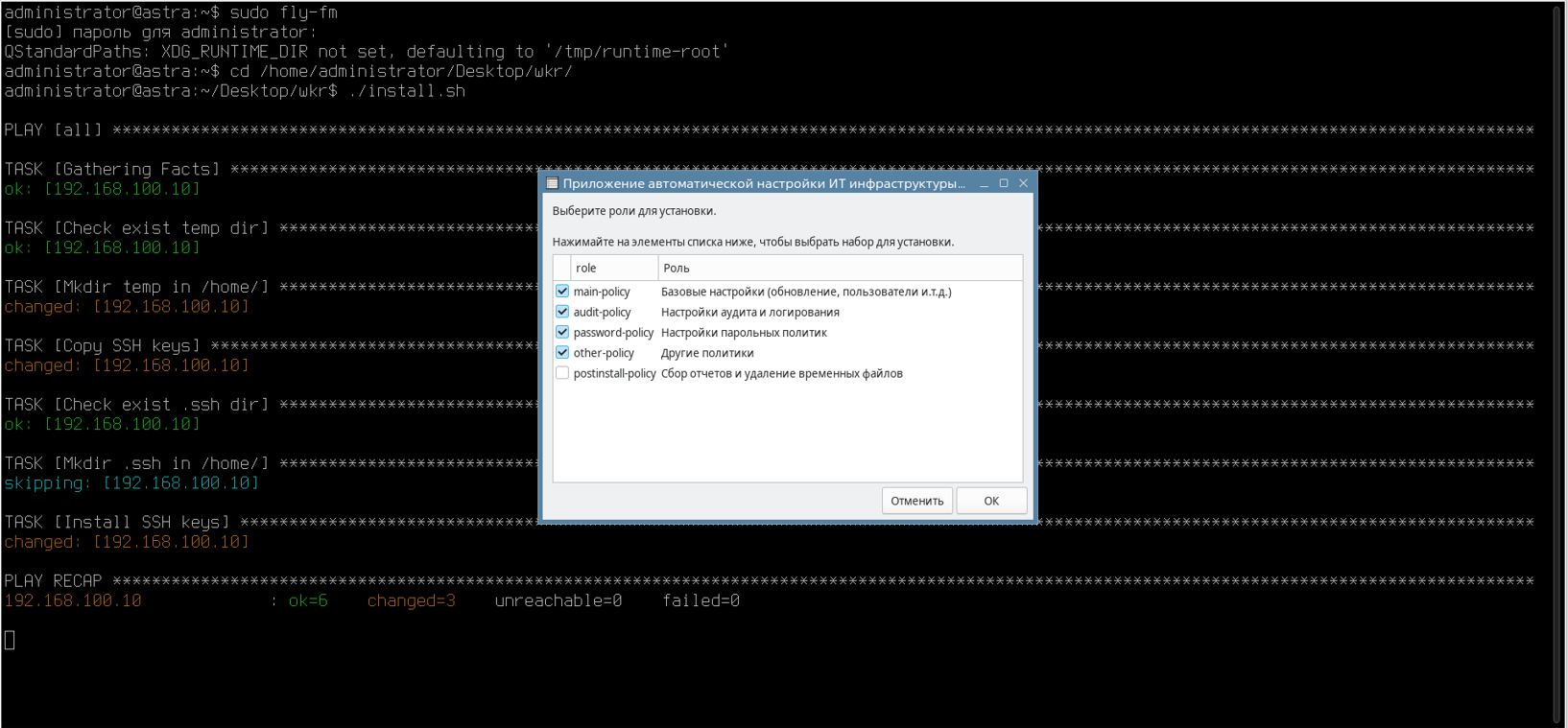


Рисунок 5.9 – Диалоговое окно выбора настроек ИТ инфраструктур

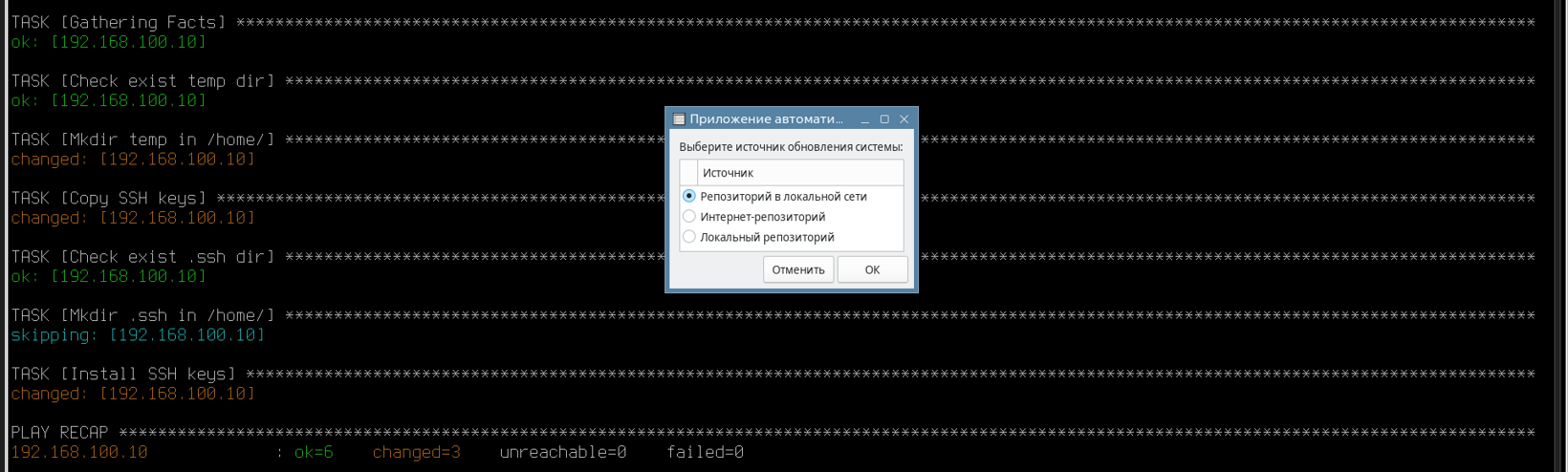


Рисунок 5.10 – Диалоговое окно выбора места нахождения репозитория

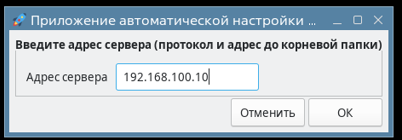


Рисунок 5.11 – Диалоговое окно ввода IP адреса нахождения репозитория

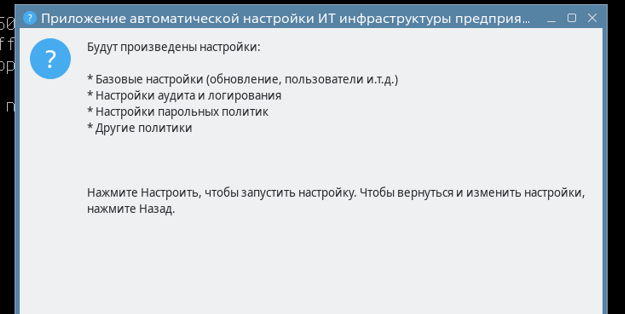


Рисунок 5.12 – Диалоговое окно подтверждения запуска настройки ИТ инфраструктуры

Работа приложения начинается с запуска скрипта install.sh. В ходе запуска инициализируются переменные, выполняется поиск директории хранения временных файлов, используемых во время работы приложения, если данная директория существует, то она удаляется со всем её содержимым и создаётся заново, выполняется проверка установленных, обязательных для работы приложения, пакетов в операционной системе. Данные действия выполняем фрагмент кода (листинг 5.1).

Листинг 5.1 – Подготовка к выполнению приложения

#!/bin/sh

WIDTH=78

HEIGHT=20

# Exit codes.

SUCCESS=0

FAILURE=1

#Buttons text

bt\_next="Далее"

bt\_exit="Выход"

bt\_yes="Да"

bt\_no="Нет"

bt\_back="Назад"

bt\_cancel="Отмена"

bt\_settings="Настроить"

bt\_select="Выбрать"

bt\_add="Добавить"

bt\_again="Заново"

#Path

TMP=tmp

OU=$1

if ls $TMP > /dev/null 2>&1 ; then

rm -rf $TMP/

fi

#Policy

BASIC="Базовые настройки (обновление, пользователи и.т.д.)"

AUDIT="Настройки аудита и логирования"

PASSWORD="Настройки парольных политик"

POSTINSTALL="Сбор отчетов и удаление временных файлов"

OTHER="Другие политики"

#Policy keys

BASIC\_role="main-policy"

AUDIT\_role="audit-policy"

PASSWORD\_role="password-policy"

POSTINSTALL\_role="postinstall-policy"

OTHER\_role="other-policy"

TITLE="Приложение автоматической настройки ИТ инфраструктуры предприятия"

TEXT\_main\_menu="Приложение автоматической настройки инфраструктуры! Нажмите $bt\_next для продолжения или $bt\_exit для выхода из мастера установки."

TEXT\_uz\_select="Выберите уровень значимости информации"

TEXT\_install\_confirmation\_head="Будут произведены настройки:

"

TEXT\_install\_confirmation\_tail="Нажмите $bt\_settings, чтобы запустить настройку. Чтобы вернуться и изменить настройки, нажмите $bt\_back."

TEXT\_select\_roles="Выберите роли для установки.

Нажимайте на элементы списка ниже, чтобы выбрать набор для установки."

TEXT\_network\_scanning="Введите сеть для сканирования"

TEXT\_wrong\_roles="Должна быть выбрана хотя бы одна из этих ролей:

\* $BASIC

\* $AUDIT

\* $PASSWORD

"

TEXT\_repo\_location="Выберите источник обновления системы:"

main() {

cd "$(dirname "$0")" || exit "$?"

test\_zenity\_and\_scripts

main\_menu

}

test\_zenity\_and\_scripts() {

if ! command -v zenity &> /dev/null ; then

echo "Ошибка: утилита zenity не найдена!" >&2

if [ -f /etc/astra\_version ] ; then

echo "Пакет zenity не установлен! Выполните 'sudo apt-get install zenity -y' для установки" >&2

elif [ -f /etc/redos-release ] ; then

echo "Пакет zenity не установлен! Выполните 'sudo yum install zenity -y' для установки" >&2

fi

exit "${FAILURE}"

fi

if ! ls ./play\_reports.yml> /dev/null 2>&1 ; then

echo "Ошибка! Playbook play\_reports.yml отсутствует!"

exit "${FAILURE}"

fi

if ! command -v nmap &> /dev/null ; then

echo "Ошибка: утилита nmap не найдена!" >&2

if [ -f /etc/astra\_version ] ; then

echo "Пакет nmap не установлен! Выполните 'sudo apt-get install nmap -y' для установки" >&2

elif [ -f /etc/redos-release ] ; then

echo "Пакет nmap не установлен! Выполните 'sudo yum install nmap -y' для установки" >&2

fi

exit "${FAILURE}"

fi

if ! command -v ansible-playbook &> /dev/null ; then

echo "Ошибка: ansible не установлен!" >&2

if [ -f /etc/astra\_version ] ; then

echo "Пакет ansible не установлен! Выполните 'sudo apt-get install ansible -y' для установки" >&2

elif [ -f /etc/redos-release ] ; then

echo "Пакет ansible не установлен! Выполните 'sudo yum install ansible -y' для установки" >&2

fi

exit "${FAILURE}"

fi

if ! command -v sudo realm &> /dev/null ; then

echo "Ошибка: realmd не установлен!" >&2

if [ -f /etc/astra\_version ] ; then

echo "Пакет realmd не установлен! Выполните 'sudo apt-get install realmd -y' для установки" >&2

elif [ -f /etc/redos-release ] ; then

echo "Пакет realmd не установлен! Выполните 'sudo yum install realmd -y' для установки" >&2

fi

exit "${FAILURE}"

else

if [ ! -h /usr/bin/realm ] ; then

sudo ln -s /usr/sbin/realm /usr/bin/realm

fi

fi

if ! command -v sudo adcli &> /dev/null ; then

echo "Ошибка: adcli не установлен!" >&2

if [ -f /etc/astra\_version ] ; then

echo "Пакет adcli не установлен! Выполните 'sudo apt-get install adcli -y' для установки" >&2

elif [ -f /etc/redos-release ] ; then

echo "Пакет adcli не установлен! Выполните 'sudo yum install adcli -y' для установки" >&2

fi

exit "${FAILURE}"

else

if [ ! -h /usr/bin/adcli ] ; then

sudo ln -s /usr/sbin/adcli /usr/bin/adcli

fi

fi

sudo cp ansible.cfg /etc/ansible/ansible.cfg

sudo cp hostname.py /usr/lib/python3/dist-packages/ansible/modules/system/hostname.py

mkdir $TMP

}

Основная функция main вызывает утилиту отрисовки стартового диалогового окна запуска приложения и обрабатывает нажатие пользователем кнопок «Далее» и «Выход». При выборе пользователем продолжить работу, функция main передаёт управление в следующую функцию (листинг 5.2).

Листинг 5.2 – Функция **main**

main\_menu() {

zenity --title "$TITLE" --info --text="$TEXT\_main\_menu" --no-wrap --ok-label "$bt\_next" --extra-button "$bt\_exit" > /dev/null 2>&1

if [ "$?" -ne "${SUCCESS}" ] ; then

exit "${SUCCESS}"

fi

select\_security\_level

}

Функция select\_security\_level вызывает утилиту отрисовки окна выбора значимости информации, обрабатывает нажатие пользователем кнопок «Отменить» и «ОК», обрабатывает положение переключателя, выбранного пользователем, записывает сделанный пользователем выбор в переменную и передаёт управление в следующую функцию (листинг 5.3).

Листинг 5.3 – Функция **select\_security\_level**

select\_security\_level() {

UZ=$( zenity --title="$TITLE" --text="$TEXT\_uz\_select" --list --radiolist --column="" --column "Уровень/класс защищенности" FALSE "УЗ-2" TRUE "УЗ-3" FALSE "УЗ-4" 2>/dev/null)

if [ "$?" -ne "${SUCCESS}" ] ; then

exit "${FAILURE}"

fi

if [ "$UZ" == "УЗ-4" ] ; then

cd 4/

input\_network

select\_roles

#UZ=4

elif [ "$UZ" == "УЗ-3" ] ; then

cd 3/

input\_network

select\_roles

#UZ=3

elif [ "$UZ" == "УЗ-2" ] ; then

cd 2/

input\_network

select\_roles

#UZ=2

fi

}

Функция **input\_network** (листинг 5.4) вызывает утилиту отрисовки диалогового окна ввода диапазона IP адресов или адреса сети и обрабатывает нажатие пользователем кнопок «Отменить» и «ОК». После ввода пользователем диапазона или адреса сети и нажатия кнопки «ОК», функция передаёт введённые параметры в функцию **check\_right\_address** (листинг 5.5), которая проверяет корректность введённых пользователем параметров сканирования сети, если параметры заданы верно, функция присваивает значение переменной и передаёт переменную в функцию **nmap\_scan** (листинг 5.6), иначе сообщает пользователю о неверных параметрах и предлагает пользователю ввести параметры снова.

Листинг 5.4 - Функция **input\_network**

input\_network() {

network=$( zenity --title="$TITLE" --text="$TEXT\_network\_scanning" --forms --add-entry="Введите диапазон адресов в формате 192.168.0.1-254

или в формате 192.168.0.0/24" 2>/dev/null)

if [ "$?" -ne "${SUCCESS}" ] ; then

exit "${SUCCESS}"

fi

check\_right\_address

}

Листинг 5.5 - Функция **check\_right\_address**

check\_right\_address() {

mask=$(echo "$network" | awk -F "/" '{print $2}' )

if [[ $mask -lt 32 && "$mask" -ne 0 ]] ; then

network=$(echo "$network" | grep -o -E '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\/[[:digit:]]{1,2}' 2>/dev/null)

if [ -n "$network" ] ; then

nmap\_scan

else

zenity --warning --text="Неверный адрес сети! Повторите ввод!" --no-wrap --ok-label "$bt\_next" 2>/dev/null

input\_network

fi

elif [[ $mask -gt 32 && "$mask" -ne 0 ]] ; then

zenity --warning --text="Маска сети больше, чем 32! Повторите ввод." --no-wrap --ok-label "$bt\_next" 2>/dev/null

input\_network

elif [ -z "$mask" ] ; then

network=$(echo "$network" | grep -o -E '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)-(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)' 2>/dev/null)

if [ -n "$network" ] ; then

nmap\_scan

else

zenity --warning --text="Неверный адрес сети! Повторите ввод!" --no-wrap --ok-label "$bt\_next" 2>/dev/null

input\_network

fi

fi

}

Листинг 5.6 - Функция **nmap\_scan**

nmap\_scan() {

echo -e "[arms]" > ../$TMP/inventory.ssh

scan= $(nmap -p 22 $network > ../$TMP/ansible.ssh)

net=$(echo "$network" | awk -F. '{print $1"."$2"."$3}')

scan=$(echo $scan | grep -o -E '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)' ../$TMP/ansible.ssh > ../$TMP/nmap\_find)

ip\_list=$(cat ../$TMP/nmap\_find)

for i in $ip\_list; do

ip="$ip TRUE $i\n"

done

echo -e "$ip" > ../$TMP/nmap\_list

#zero\_list=$(cat ../$TMP/nmap\_list | wc -l)

#echo $zero\_list

#if [ -z $zero\_list ] ; then

# zenity --warning --text="Нет доступных адресов! Проверьте настройки сети или доступность АРМ!" --no-wrap --ok-label "$bt\_next" 2>/dev/null

# input\_network

#fi

ip\_list=$(zenity --title="$TITLE" --list --checklist --text="Выберите IP-адреса для развертывания" --width="200" --height="400" --column="" --column "IP-адрес" $(cat ../$TMP/nmap\_list) 2>/dev/null)

ip\_list=$(echo ${ip\_list} | tr -s 'TRUE' ' '|tr -s '|' '\n')

echo -e "$ip\_list" >> ../$TMP/inventory.ssh

ansible\_ping

}

## **Заключение**

Во время прохождения практики были выполнены такие задачи как:

* Ознакомления с родом деятельности предприятия,
* Ознакомление с законодательной базой, регулирующей деятельность предприятия и регламентирующей действия организаций в сфере информационной безопасности,
* Определены инструменты, необходимые для решения задачи автоматизации настройки ИТ инфраструктуры предприятия-заказчика в разрезе информационной безопасности,
* Произведён анализ представленных на отечественном рынке решений, направленных на централизованную настройку ИТ инфраструктуры.

В ходе анализа представленных на рынке решений, рассмотрения сильных и слабых сторон данных решений, а также изучения предметной области деятельности организации, принято решение о разработке собственного узкоспециализированного программного обеспечения, не требующее высоких навыков и специализированного обучения от специалистов.

Таким образом поставленное на практику задание выполнено.

Собранный и проанализированный в ходе практики материал позволяет перейти к разработки программного обеспечения, предназначенного для автоматизации настройки ИТ инфраструктуры предприятия-заказчика.

## **Список использованных источников**

1. Колисниченко Д. Н. Командная строка Linux и автоматизация рутинных задач. М.: БХВ-Петербур, 2014. 368 с.
2. Бабаш А. В., Баранова Е. К. Информационная безопасность и защита. Учебное пособие. М.: РИОР, 2017. 322 с.
3. Родичев Ю. А. Нормативная база и стандарты в области информационной безопасности. Учебное пособие. М.: Санкт-Петербург: Питер, 2021. 256 с.
4. Уорд Б. Внутреннее устройство Linux. М.: Санкт-Петербург: Питер, 2022. 384 с.
5. Unix и Linux: руководство системного администратора. М.: Вильямс, 2020. 1168 с.
6. Regular expressions in grep ( regex ) with examples [Электронный ресурс]. URL: https://www.cyberciti.biz/faq/grep-regular-expressions/ (дата обращения: 25.03.2023).
7. Руководство Zenity [Электронный ресурс]. URL: https://help.gnome.org/users/zenity/stable/index.html.ru (дата обращения: 25.03.2023).

## **Приложение 1**

## **Приложение 2**