МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»»

**Курсовая работа**

По дисциплине«Технологии программирования»

Тема: «Поддержание в актуальном состоянии схемы сети»

Выполнил:  
студент гр. БПИ-19-2  
Сенякин А. Ю.

Проверил:   
Широков А. И.

Москва, 2020 г.

Оглавление

[**1. Техническое задание** 3](#_Toc59459964)

[1.1 Введение 3](#_Toc59459965)

[*1.1.1 Наименование программы* 3](#_Toc59459966)

[*1.1.2 Характеристика области применения* 3](#_Toc59459967)

[1.2 Основание для разработки 3](#_Toc59459968)

[*1.2.1 Документ, на основании которого ведётся разработка* 3](#_Toc59459969)

[*1.2.2 Организация, утвердившая документ* 3](#_Toc59459970)

[*1.2.3 Наименование темы разработки* 3](#_Toc59459971)

[1.3 Назначение разработки 3](#_Toc59459972)

[*1.3.1 Функциональное и эксплуатационное назначение программы* 3](#_Toc59459973)

[1.4 Технические требования к программе 4](#_Toc59459974)

[*1.4.1 Требования к функциональным характеристикам* 4](#_Toc59459975)

[*1.4.2 Требования к надежности* 8](#_Toc59459976)

[*1.4.3 Условия эксплуатации* 8](#_Toc59459977)

[*1.4.4 Требования к составу и параметрам технических средств* 9](#_Toc59459978)

[*1.4.5 Требования к информационной и программной совместимости* 9](#_Toc59459979)

[1.5 Требования к программной документации 9](#_Toc59459980)

[1.6 Стадии и этапы разработки 9](#_Toc59459981)

[*1.6.1 Стадии разработки* 9](#_Toc59459982)

[*1.6.2 Этапы и содержание работы* 10](#_Toc59459983)

[*1.6.3 Сроки разработки* 10](#_Toc59459984)

[1.7 Порядок контроля и приема 10](#_Toc59459985)

[*1.7.1 Виды испытаний* 10](#_Toc59459986)

[*1.7.2 Общие требования к приёмке работы* 10](#_Toc59459987)

[**2. Эскизный проект** 11](#_Toc59459988)

[2.1 Предварительная разработка структуры входных и выходных данных 11](#_Toc59459989)

[2.2 Уточнение методов решения задачи 11](#_Toc59459990)

[2.3 Разработка общего алгоритма решения задачи 11](#_Toc59459991)

[**3. Технический проект** 12](#_Toc59459992)

[3.1 Диаграмма переходов состояний 12](#_Toc59459993)

[3.2 Уточнение структуры входных и выходных данных 13](#_Toc59459994)

[*3.2.1 Входные данные* 13](#_Toc59459995)

[*3.2.2 Выходные данные* 14](#_Toc59459996)

[3.3. Разработка алгоритма решения задачи 14](#_Toc59459997)

[*3.3.1 Общий алгоритм решения задачи* 14](#_Toc59459998)

[3.4. Определение формы представления входных и выходных данных 15](#_Toc59459999)

[*3.4.1 Входные и выходные данные* 15](#_Toc59460000)

[*3.4.2 Представление данных в базе данных* 16](#_Toc59460001)

[3.5. Разработка структуры программы 17](#_Toc59460002)

[*3.5.1 Структура кода программы* 17](#_Toc59460003)

[*3.5.2 Структура организации базы данных* 19](#_Toc59460004)

[**4. Описание программного продукта** 21](#_Toc59460005)

[4.1 Описание кода программы 21](#_Toc59460006)

[*4.1.1 Класс Main и его функции* 21](#_Toc59460007)

[*4.1.2 Класс Reference и его функции* 22](#_Toc59460008)

[*4.1.3 Класс Logbook и его функции* 23](#_Toc59460009)

[*4.1.4 Класс Update и его функции* 23](#_Toc59460010)

[*4.1.5 Класс DB и его функции* 24](#_Toc59460011)

[4.2 Описание пользовательского интерфейса 25](#_Toc59460012)

[*4.2.1 Главное окно программы* 25](#_Toc59460013)

[*4.2.2 Окно «Добавить устройство»* 26](#_Toc59460014)

[*4.2.3 Окно «Редактировать устройство»* 26](#_Toc59460015)

[*4.2.4 Окно «Журнал событий»* 27](#_Toc59460016)

[*4.2.5 Окно подтверждения удаления* 28](#_Toc59460017)

[*4.2.6 Окно справки* 28](#_Toc59460018)

[**5. Руководство пользователя** 29](#_Toc59460019)

[5.1 Первый запуск программы 29](#_Toc59460020)

[5.2 Инструкция по работе с программой 29](#_Toc59460021)

[*5.2.1 Добавление устройства* 29](#_Toc59460022)

[*5.2.2 Редактирование информации об устройстве* 29](#_Toc59460023)

[*5.2.3 Просмотр «Журнала событий»* 30](#_Toc59460024)

[*5.2.4 Удаление устройства* 31](#_Toc59460025)

[**6. Приложения** 31](#_Toc59460026)

[6.1 Код программы 31](#_Toc59460027)

[**7. Библиографический список** 39](#_Toc59460028)

# **1. Техническое задание**

## 1.1 Введение

Данное техническое задание распространяется на пользовательское приложение «Network Directory», с помощью которого пользователь сможет осуществлять учет компьютеров и сетевого оборудования, связей между устройствами и историю изменения этих связей.

### *1.1.1 Наименование программы*

Название разрабатываемой программы – “Network Directory”

### *1.1.2 Характеристика области применения*

Данная программа предназначена преимущественно для пользователей, которым необходимо иметь актуальную информацию об устройствах в сети, в том числе начинающим системным и сетевым администраторам.

## 1.2 Основание для разработки

### *1.2.1 Документ, на основании которого ведётся разработка*

Разработка данного проекта ведётся на основании задания для курсовой работы

### *1.2.2 Организация, утвердившая документ*

Организацией, утвердившей задание, является кафедра инженерной кибернетики университета НИТУ «МИСиС»

### *1.2.3 Наименование темы разработки*

Темой данной разработки является разработка программного продукта, предназначенного для поддержания в актуальном состоянии схемы сети.

## 1.3 Назначение разработки

### *1.3.1 Функциональное и эксплуатационное назначение программы*

Функциональным и эксплуатационным назначением данного продукта является использование её пользователями, которым необходимо иметь информацию о количестве, типах, настройках оборудования, находящегося в сети мелкого предприятия или организации. Продукт разрабатывается для большой аудитории пользователей, в качестве вспомогательной программы.

## 1.4 Технические требования к программе

### *1.4.1 Требования к функциональным характеристикам*

Программа должна обеспечивать возможность выполнения следующих функций:

1. Позволять пользователю вести учёт компьютеров и сетевого оборудования
2. Позволять пользователю добавлять и удалять устройства
3. Позволять пользователю добавлять, изменять и удалять информацию об устройствах
4. Позволять пользователю добавлять, изменять и удалять информацию о настройках каждого из добавленных устройств
5. Позволять пользователю просматривать журнал произведенных с устройствами операций
6. Позволять пользователю посмотреть инструкцию по работе с программой
7. Простоту и удобность эксплуатации
8. Понятность интерфейса

При запуске пользователем данной программы на экране появляется главное рабочее окно программы.

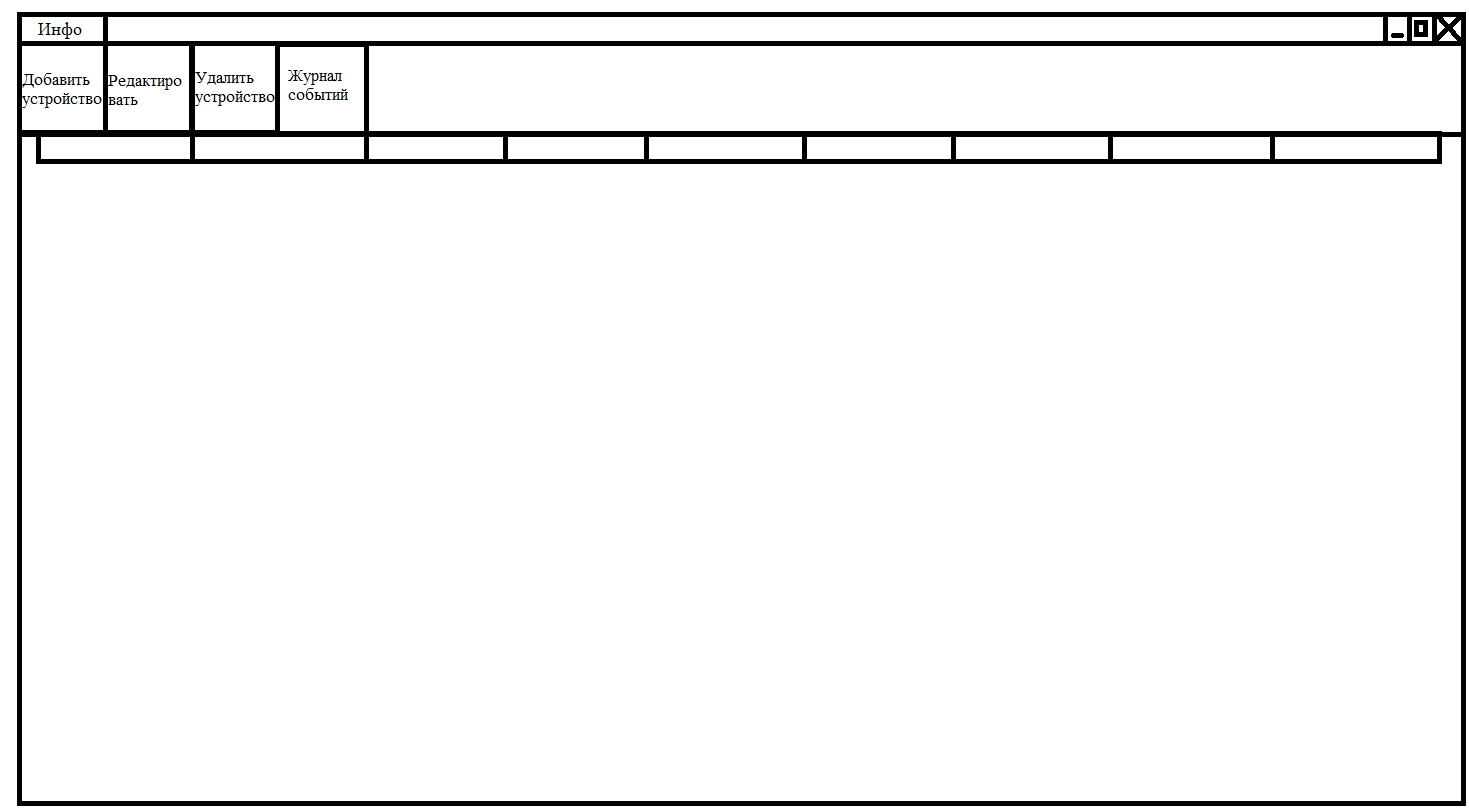


Рисунок 1 – Главное окно программы

В окне присутствуют несколько кнопок на приборной панели, а также большое поле в виде таблицы для вывода устройств в сети и информация об их настройках.

При нажатии на кнопку «Добавить устройство» в верхней панели открывается окно добавления устройства поверх основного окна. С помощью него пользователь может добавить пять видов устройств (маршрутизатор, коммутатор, сервер, компьютер, МФУ), выбрав нужное в выпадающем списке, а также заполнить все необходимые настройки конфигурации в соответствующих полях.

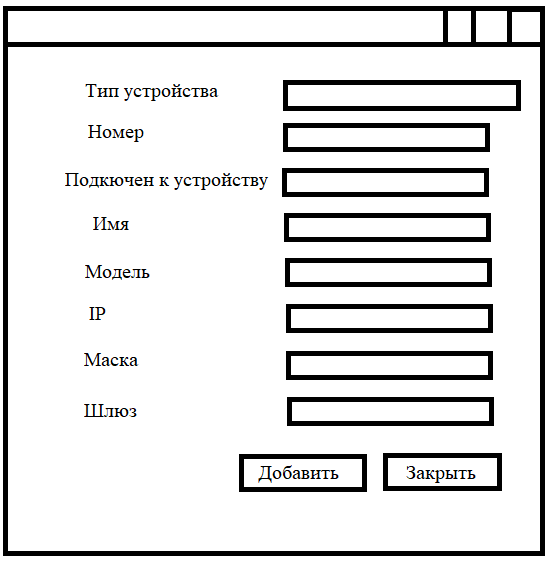


Рисунок 2 – окно добавления устройства

При нажатии на кнопку «Добавить» в диалоговом окне произойдет добавление устройства в общую таблицу.

При нажатии кнопки «Закрыть» произойдет закрытие окна добавления устройства.

При нажатии на кнопку «Редактировать» в верхней панели открывается дочернее окно, выглядящее как окно добавления, но с предварительно заполненными полями, так как информация об устройстве уже была внесена ранее. Также вместо кнопки «Добавить» располагается кнопка «Редактировать».

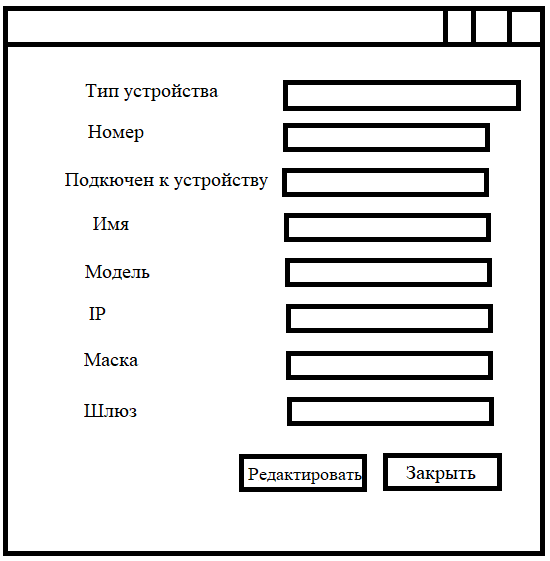


Рисунок 3 – окно редактирования информации устройства

При нажатии на кнопку «Закрыть» в окне редактирования устройства произойдет его закрытие.

При нажатии на кнопку «Удалить устройство», выбрав предварительно одно или несколько устройств для удаления (одно – щелкнув один раз левой кнопкой мыши на запись с нужным устройством, несколько – щелкнуть на каждое из устройств, которые необходимо удалить с зажатой клавишей «Ctrl» на клавиатуре), в верхней панели откроется диалоговое окно, которое запрашивает подтверждение на удаление данных.

При нажатии на кнопку «Да» произойдет удаление данных.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – окно подтверждения на удаление данных

Также будет реализована проверка на правильность вводимых данных в окнах «Добавить устройство» и «Редактировать»

При нажатии кнопки «Справка» в панели меню откроется окно с описанием доступных функций программы.

После нажатия на кнопку «Ок» это окно закроется.

При нажатии на кнопку «Журнал событий» откроется стандартное окно Windows поверх основного окна с таблицей, хранящей историю изменения соединения устройств между собой.

При нажатии на кнопку «Очистить» в окне журнала событий программа так же запросит подтверждение у пользователя.

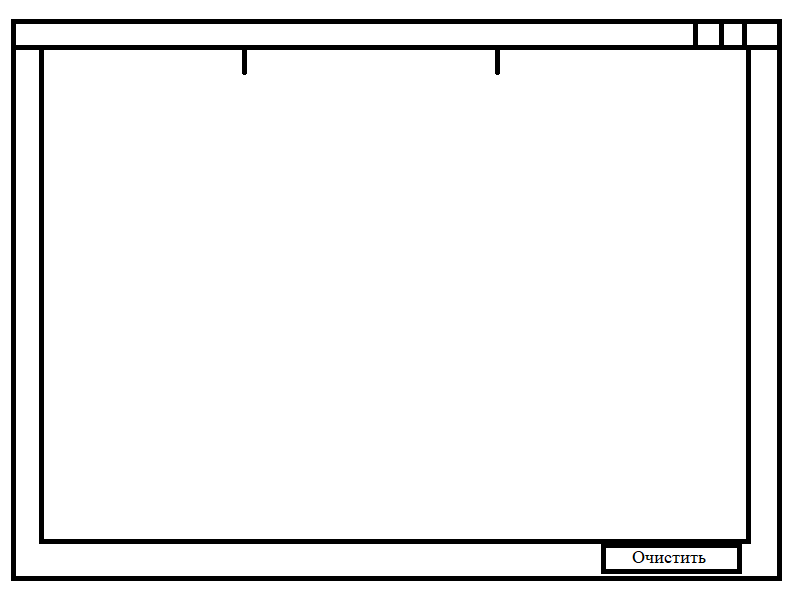


Рисунок 5 – окно «Журнала событий»

### *1.4.2 Требования к надежности*

Надежность программы будет обеспечена использованием лицензионного ПО. Также будут заранее предусмотрены и устранены возможные ошибки, которые могут возникнуть при некорректном использовании её пользователем.

### *1.4.3 Условия эксплуатации*

Разрабатываемая программа может эксплуатироваться единовременно одним пользователем, предварительно установившем её на устройство, с которого будет вестись эксплуатация.

### *1.4.4 Требования к составу и параметрам технических средств*

Для надежной работы данного программного продукта пользователю необходимо иметь персональный компьютер, содержащий:

* процессор с тактовой частотой не менее 1.2 ГГц;
* оперативную память объемом не менее 128 Mб;
* жесткий диск объемом 40 Гб, и выше;
* оптический манипулятор типа «мышь», клавиатура;

### *1.4.5 Требования к информационной и программной совместимости*

• Система должна работать под управление ОС Windows 8, Windows 8.1, Windows 10.

• Среда программирования PyCharm

• Язык программирования – Python

## 1.5 Требования к программной документации

Руководство пользователя, содержащее описание методов эксплуатации программы и руководство программиста, включающее в себя описание принципов работы программы.

## 1.6 Стадии и этапы разработки

### *1.6.1 Стадии разработки*

Разработка программного продукта проводится в соответствии со следующими стадиями: составление технического задания, составление эскизного проекта, разработка технического проекта, написание рабочего проекта, тестирования рабочего проекта, тестирование рабочего проекта, составление технической документации к данному проекту.

### *1.6.2 Этапы и содержание работы*

Разработка будет включать в себя следующий перечень программных документов:

• пояснительная записка с отчётами о проведённых испытаниях

• руководство пользователя

• листинг (код программы)

### *1.6.3 Сроки разработки*

Данный курсовой проект должен быть выполнен в соответствии со следующими сроками:

• Написание технического задания - 4.10.20-6.10.20

• Составление эскизного проекта - 7.10.20-14.10.20

• Разработка технического проекта - 15.10.18-21.11.20

• Написание рабочего проекта - 22.11.20-06.12.20

• Тестирование рабочего проекта - 07.12.20-10.12.20

• Составление технической документации – 11.12.20-21.12.20

## 1.7 Порядок контроля и приема

### *1.7.1 Виды испытаний*

Тесты на работоспособность. Тестирование проводится студентами института ИТКН и институтов других ВУЗов. Они предоставляют отчеты о проведённых тестированиях, которые будут включены в пояснительную записку.

### *1.7.2 Общие требования к приёмке работы*

После проведения тестирования в полном объеме проводится защита работы с последующей демонстрацией работы программы.

# **2. Эскизный проект**

Перейдём к разработке спецификаций, включающих в себя предварительную разработку структуры входных и выходных данных, уточнение методов решения задачи и разработку общего описания алгоритма решения задачи.

## 2.1 Предварительная разработка структуры входных и выходных данных

Для реализации поставленной нам задачи необходимо разработать структуру данных, принимаемых на вход программой, а также данных, которые будут выводиться пользователю.

Эти данные должны отражать действительные характеристики объектов, с которыми пользователь может столкнуться в реальной жизни. Для этого необходимо перечислить ряд объектов, которыми можно оперировать в программном продукте, их свойства и характеристики, а также все дополнительные условия и ограничения, связанные с этими свойствами и характеристиками, так как существует ряд нюансов, которые стоит учитывать в условиях работы с конкретной областью и данных об объектах в этой области.

## 2.2 Уточнение методов решения задачи

Основным методом в решении задачи будет являться метод пошаговой детализации: задача будет разделена на подзадачи, каждая из которых будет обладать своим набором входных и выходных данных. Разбиение задачи на составные части следует осуществлять до тех пор, пока хотя бы одну задачу нельзя будет решить известными методами.

В связи с выбранным методом рационально будет использовать объектно-ориентированный подход, так как благодаря нему будет удобно осуществлять моделирование объектов и их свойств, с которыми мы будем работать

## 2.3 Разработка общего алгоритма решения задачи

Поставленная задача подразумевает работу с большим числом объектов, над которыми необходимо будет осуществлять различные операции. Каждый из объектов имеет собственное описание, из чего следует, что мы имеем достаточно большое количество данных.

Эти данные будет удобно хранить едино в одном месте, вносить их, удалять и обращаться к ним. Для выполнения этой задачи необходимо создать базу данных, с которой будет работать программа.

При выполнении любой из основных операций программы будет происходить обращение к базе данных, а затем осуществляться вывод необходимой информации в удобном для пользователя виде.

# **3. Технический проект**

## 3.1 Диаграмма переходов состояний

Перечислим возможные состояния программы:

А) Исходное состояние

Б) Основное состояние

В) Добавить устройство

Г) Редактировать

Д) Удалить устройство

Е) Справка

Ж) Состояние завершения

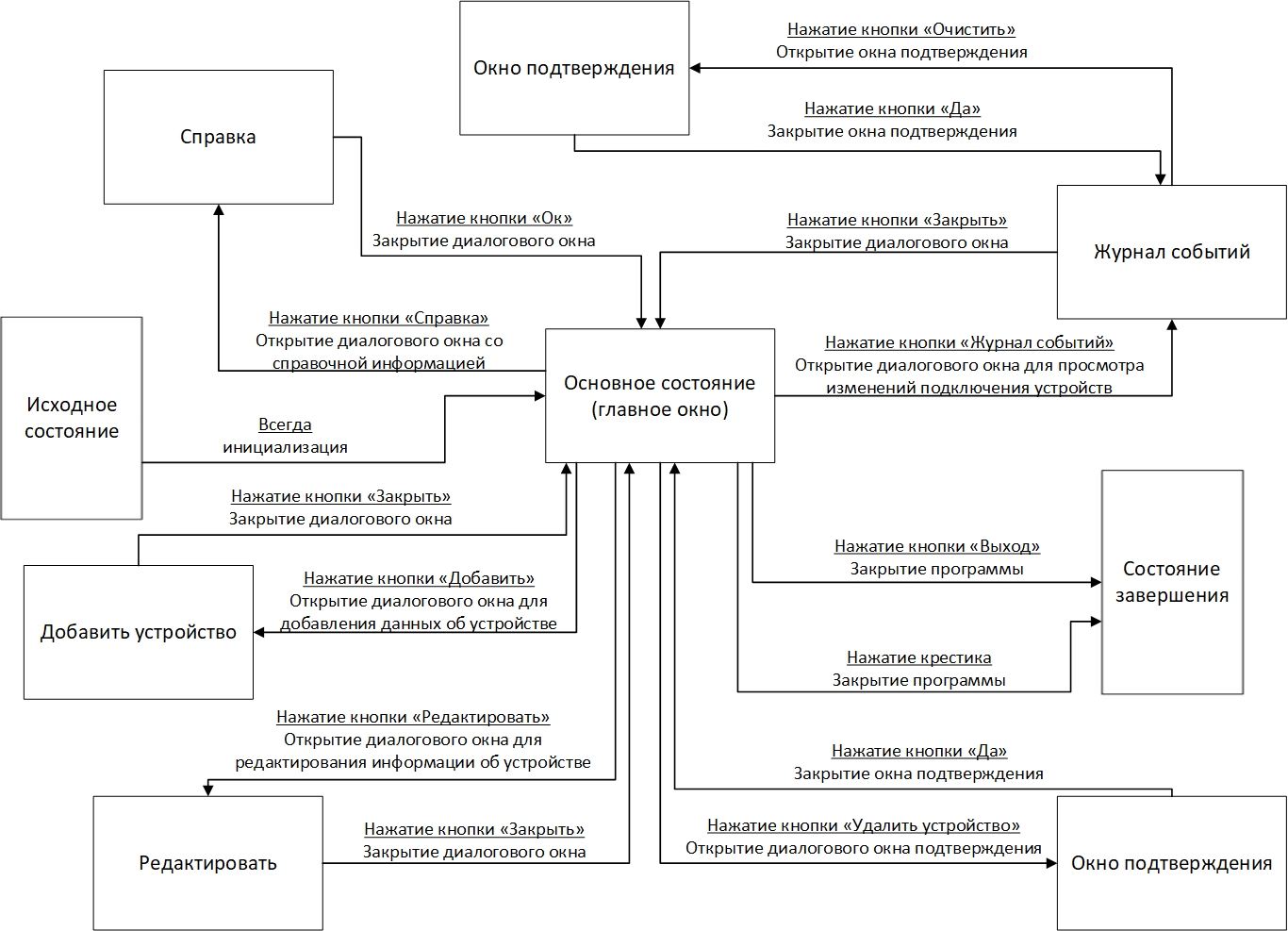


Рисунок 6 – Диаграмма переходов состояния

Представленная диаграмма описывает переходы программы из одного состояния в другое, учитывая условия перехода и действия, выполняющиеся при переходе.

## 3.2 Уточнение структуры входных и выходных данных

### *3.2.1 Входные данные*

Определим список данных, которые могут поступать на вход программы:

-Тип устройства

-Идентификационный номер устройства

-Идентификационный номер устройства родителя, то есть устройства, к которому подключено основное устройство

-Имя устройства. Его имя в сети, которое должно быть уникальным, для того чтобы не возникло неоднозначности с определением устройства.

-Модель устройства

-IP адрес устройства

-Маска подсети

-Шлюз по умолчанию

### *3.2.2 Выходные данные*

Этот же список данных будет и выходным. Вывод будет производиться в таблицу в главном окне программы.

При просмотре «Журнала событий» на выход будут предоставляться следующие данные:

- Идентификационный номер устройства

- Идентификационный номер устройства родителя

- Имя устройства

- Дата внесения изменений

- Время внесения изменений

## 3.3. Разработка алгоритма решения задачи

### *3.3.1 Общий алгоритм решения задачи*

Алгоритм решения задачи состоит из следующих этапов:

- Автоматически создается база данных, если она уже не была создана

Если необходимо внести данные:

- Нажимаем на кнопку «Добавить устройство»

- Вводим данные о новом устройстве в соответствующие поля

- Сохраняем данные в соответствующей таблице в базе данных. Создаем таблицу, если её нет в базе данных.

- Выводим в главное окно

Если необходимо изменить внесенные данные:

- Нажимаем на кнопку «Редактировать»

- Запрашиваем данные из базы данных

- Вносим изменения

Если ID устройства родителя был изменен, то вносим данные в необходимую таблицу в базе данных с информацией об изменении соединения устройств. Создаем необходимую таблицу в базе данных, если её нет в базе данных

- Сохраняем в базе данных новые данные

- Выводим в главное окно

Если необходимо удалить данные:

- Выбираем нужную строку или несколько строк

- Нажимаем на кнопку «Удалить устройство»

Если необходимо просмотреть историю изменения соединения устройств:

- Нажимаем на кнопку «Журнал событий»:

- Запрашиваем данные из соответствующей таблицы базы данных

- Выводим в окно «Журнала»

## 3.4. Определение формы представления входных и выходных данных

### *3.4.1 Входные и выходные данные*

-Тип устройства – текстовый тип, выбор из пяти вариантов

-Идентификационный номер устройства – целочисленный тип, проверка на уникальность

-Идентификационный номер устройства родителя – целочисленный тип

-Имя устройства – текстовый тип, проверка на уникальность

-Модель устройства – текстовый тип

-IP адрес устройства – текстовый тип, проверка на уникальность и правильность формата ввода

-Маска подсети – текстовый тип, проверка на правильность формата ввода

-Шлюз по умолчанию – текстовый тип, проверка на правильность формата ввода

Этот же список данных будет и выходным. Вывод будет производиться в таблицу в главном окне программы из таблицы Devices базы данных.

При просмотре «Журнала событий» на выход будут предоставляться следующие данные:

- Идентификационный номер устройства – целочисленный тип

- Идентификационный номер устройства родителя – целочисленный тип

- Имя устройства – текстовый тип

- Дата и время внесения изменений – текстовый тип

Поля даты и времени в таблице являются текстовыми так как в СУБД SQLite, которая работает с базой данных нет встроенных типов данных даты и времени.

Все перечисленные данные выводятся в окно «Журнала событий» из таблицы Logbook базы данных.

### *3.4.2 Представление данных в базе данных*

Обе таблицы хранятся в файле «Network directory.db», который представляет из себя базу данных для данной программы. База данных была разработана средствами SQLite.

## 3.5. Разработка структуры программы

### *3.5.1 Структура кода программы*

Класс Main() – класс основного окна программы

Функция \_\_init\_\_() – конструктор класса Main

Функция init\_main() – функция инициализации главного окна

Функция records() – функция, передающая аргументы для записи данных в таблицу Devices

Функция update\_records() – функция, вносящая обновленные данные в таблицу Devices

Функция view\_records() – функция, отображающая записи из таблицы Devices в главном окне

Функция delete\_records() – функция, удаляющая выбранные записи из таблицы Devices

Функция open\_delete\_dialog() – функция, запрашивающая подтверждение на удаление в диалоговом окне

Функция open\_update\_dialog() – функция, вызывающая класс Update()

Функция open\_logbook() – функция, вызывающая класс Logbook()

Функция open\_dialog() – функция, вызывающая класс Device()

Функция open\_reference() – функция, вызывающая класс Reference()

Класс Reference() – класс справки

Функция \_\_init\_\_() – конструктор класса Reference

Функция init\_reference() – функция инициализации окна справки

Класс Device() – класс добавления устройства

Функция \_\_init\_\_() – конструктор класса Device

Функция init\_device() – функция инициализации окна добавления устройства

Функция validate\_address() – функция проверки правильности ввода IP адреса и шлюза по умолчанию

Функция validate\_mask() – функция проверки правильности ввода маски подсети

Класс Logbook() – класс журнала событий

Функция \_\_init\_\_() – конструктор класса Logbook

Функция init\_logbook() – функция инициализации окна журнала событий

Функция delete\_notes() – функция, удаляющая выбранные записи из таблицы Logbook

Функция open\_delete\_dialog() – функция, запрашивающая подтверждение на удаление в диалоговом окне

Функция notes() – функция, передающая аргументы для записи данных в таблицу Logbook

Функция view\_notes() – функция, отображающая записи из таблицы Logbook базы данных

Класс Update() – класс обновления данных об устройстве

Функция \_\_init\_\_() – конструктор класса Update

Функция init\_edit() – функция инициализации окна обновления данных

Функция default\_data() – функция, заполняющая поля ввода в окне обновления данных значениями, которые уже имеет запись с устройством из таблицы Devices

Функция get\_time() – функция, возвращающая актуальную дату и время

Функция check\_difference() – функция, проверяющая изменился ли ID устройства родителя при обновлении данных

Функция init\_logbook() – функция инициализации окна журнала событий

Класс DB() – Класс, отвечающий за работу с базой данной

Функция \_\_init\_\_() – конструктор класса DB

Функция insert\_data() – функция, вносящая данные в таблицу Devices базы данных

Функция insert\_note() – функция, вносящая данные в таблицу Logbook базы данных

### *3.5.2 Структура организации базы данных*

Таблица Devices в файле с базой данных Network\_directory.db состоит из 8 полей:

* ID - идентификационный номер устройства
* Parent\_ID - идентификационный номер устройства родителя
* Type - Тип устройства
* Name - имя устройства.
* Model - модель устройства
* IP - IP адрес устройства
* Mask - маска подсети
* Gateway - шлюз по умолчанию

Таблица Logbook в файле с базой данных Network\_directory.db состоит из 5 полей:

* ID – номер строки в таблице, проставляется автоматически
* Device\_ID - идентификационный номер устройства
* Old\_Parent\_ID - идентификационный номер старого устройства родителя
* New\_Parent\_ID - идентификационный номер нового устройства родителя
* Datetime – дата и время внесения записи в таблицу

# **4. Описание программного продукта**

В этом разделе предоставляется описание кода(см. приложение 6.1) программы(классов и функций) и описание графического интерфейса программы.

## 4.1 Описание кода программы

Для работы программы сначала необходимо импортировать несколько модулей. Модуль tkinter позволяет создать графический интерфейс. Модуль sqlite3 позволяет осуществлять работу с базой данных. Модуль datetime предоставляет пользователю возможность для удобной работы с датой и временем.

Перейдем к описанию классов и функций программы.

### *4.1.1 Класс Main и его функции*

Класс Main представляет из себя класс главного окна программы, которое открывается после запуска программы. В этом классе создается само окно, виджеты, которые отображаются на этом окне, задаются параметры этого окна. Также этому классу принадлежит ряд функций.

Функция \_\_init\_\_() является конструктором класса Main. Конструктор класса – это специальный метод, который автоматически вызывается при создании класса и передает ему два аргумента: текущий экземпляр класса и корневое окно программы.

Функция init\_main() предназначена для инициализации главного окна. Она отражает на окне все виджеты, располагает их в нужных местах в правильном порядке.

Функция records() предназначена для отображения записей в таблице главного окна и для записи введенных данных в таблицу Devices при добавлении устройства.

Функция update\_records() предназначена для записи данных в таблицу Devices при обновлении информации об устройстве в процессе его редактирования.

Функция view\_records() отображает записи из таблицы Devices в главном окне программы.

Функция delete\_records() удаляет выбранные записи из таблицы Devices.

Функция open\_delete\_dialog() открывает диалоговое окно для подтверждения на удаление.

Функция open\_update () предназначена для вызова класса Update()

Функция open\_logbook() предназначена для вызова класса Logbook()

Функция open\_dialog() предназначена для вызова класса Device()

Функция open\_reference() предназначена для вызова класса Reference()

### *4.1.2 Класс Reference и его функции*

Класс Reference представляет из себя окно справки.

Функция \_\_init\_\_() является конструктором класса Reference.

Функция init\_reference() инициализирует окно справки. Размещает на нем текст справочной информации и кнопку «Ок» для закрытия окна.

Класс Device представляет из себя класс окна добавления устройства. Он предназначен для ввода различной информации об устройстве и внесения этой информации в базу данных.

Функция \_\_init\_\_() представляет собой конструктор класса Device.

Функция init\_device() инициализирует окна добавления устройства. Она отображает на нем все поля для ввода, подписи к этим полям, выпадающий список с типами устройств, а также кнопки «Добавить» и «Закрыть».

Функция validate\_address() предназначена для проверки правильности ввода IP адреса и шлюза по умолчанию при добавлении или редактировании устройства.

Функция validate\_mask() предназначена для проверки правильности ввода маски подсети при добавлении или редактировании устройства.

### *4.1.3 Класс Logbook и его функции*

Класс Logbook представляет из себя класс окна журнала событий. Он предназначен для создания этого окна, отображения на нем виджетов, создания и удаления в нем записей, которые содержат информацию об изменении связей между устройствами в сети.

Функция \_\_init\_\_() представляет собой конструктор класса Logbook.

Функция init\_logbook() инициализирует окно журнала событий размещает на окне кнопку «Очистить журнал» и таблицу с записями из таблицы Logbook из базы данных.

Функция delete\_notes() очищает таблицу Logbook в базе данных.

Функция open\_delete\_dialog() предназначена для открытия диалогового окна с подтверждением на удаление всех записей из журнала событий.

Функция notes() предназначена для записи данных об изменении связи между устройствами в таблицу Logbook в базе данных.

Функция view\_notes() отображает все записи из таблицы Logbook из базы данных в таблицу окна журнала событий.

### *4.1.4 Класс Update и его функции*

Класс Update представляет из себя класс окна обновления данных об устройстве. Данный класс наследуется от класса Device и предназначен для создания самого окна, размещения на нем виджетов и внесении в базу данных измененной информации об устройстве.

Функция \_\_init\_\_() представляет собой конструктор класса Update.

Функция init\_edit() инициализирует на окне обновления данных те же виджеты, которые имеет и класс Device, а также кнопку «Редактировать» вместо кнопки «Добавить».

Функция default\_data() предназначена для заполнения полей ввода в окне обновления данных значениями, которые уже имеет запись с устройством из таблицы Devices.

Функция get\_time() возвращает актуальную дату и время.

Функция check\_difference() проверяет изменился ли ID устройства родителя при обновлении данных. Если изменился, то вносит в таблицу Logbook в базе данных запись, которая содержит в себе самого устройства, старый ID устройства родителя, новый ID устройства родителя, а также дату и время этого изменения.

### *4.1.5 Класс DB и его функции*

Класс DB необходим для работы с базой данных. Он создает базу данных, если она не создана и делает возможным ввод данных в соответствующие таблицы в этой базе данных.

Функция \_\_init\_\_() представляет собой конструктор класса DB. Она создает базу данных и таблицы в ней, если это требуется.

Функция insert\_data() предназначена для вызова команды на языке SQL для внесения данных в таблицу Devices в базе данных. Также она предназначена для предотвращения нарушения целостности базы данных и, если возникает ситуация, в которой целостность может нарушиться, то выводит сообщение об ошибке.

Функция insert\_note() предназначена для вызова команды на языке SQL для внесения данных в таблицу Logbook в базе данных. Также она предназначена для предотвращения нарушения целостности базы данных и, если возникает ситуация, в которой целостность может нарушиться, то выводит сообщение об ошибке.

## 4.2 Описание пользовательского интерфейса

### *4.2.1 Главное окно программы*

Главное окно представляется пользователю в следующем виде: в верхней части располагается строка меню с разделом «Инфо», в котором находится подраздел «Справка», по нажатии на который пользователь может посмотреть справку к программе. Ниже находится панель инструментов, где располагаются кнопки, при нажатии на которые пользователь может всячески взаимодействовать с программой. Основное пространство окна занимает таблица с информацией об устройствах. Все записи располагаются в узлах соответствующим их типу устройств.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 7 – Главное окно программы

### *4.2.2 Окно «Добавить устройство»*

Окно добавления устройства содержит в себе выпадающий список, с различными типами устройств, поля для ввода информации об устройстве кнопка «Добавить», которая добавляет данные в базу данных и кнопка «Закрыть» для закрытия окна.

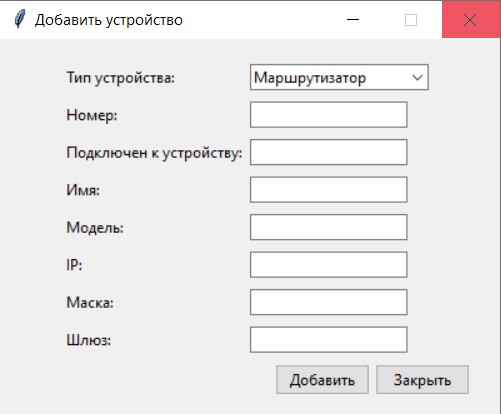


Рисунок 8 – Окно «Добавить устройство»

### *4.2.3 Окно «Редактировать устройство»*

Внешний вид окно редактирования информации об устройстве выглядит как окно «Добавить устройство» за исключением наличия кнопки «Редактировать» вместо «Добавить».

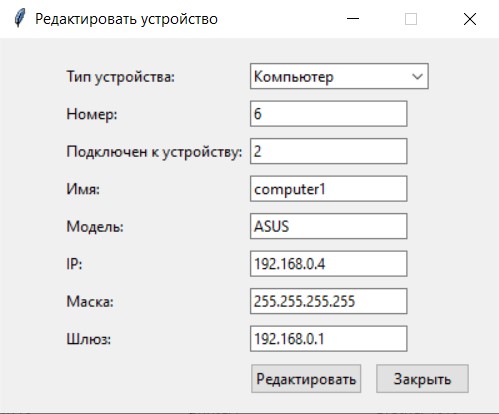


Рисунок 9 – окно «Редактировать устройство»

### *4.2.4 Окно «Журнал событий»*

Окно журнала событий выглядит следующим образом: в верху окна располагается кнопка «Очистить журнал», ниже находится таблица, в которой хранится информация об изменениях связи устройств в сети между собой.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Окно «Журнал событий»

### *4.2.5 Окно подтверждения удаления*

Данное окно всплывает после нажатие пользователем на кнопку «Удалить устройство».

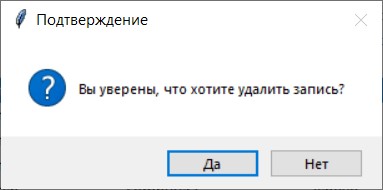


Рисунок 11 – Окно подтверждения на удаление данных

### *4.2.6 Окно справки*

Окно справки содержит небольшую инструкцию по взаимодействию с программой и кнопку «Ок», которая закрывает окно.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – Окно справки

# **5. Руководство пользователя**

## 5.1 Первый запуск программы

Для того чтобы осуществить запуск программы необходимо иметь установленный интерпретатор python версии 3 и более. Скачать его можно с официального сайта <https://www.python.org/downloads/>.

Для корректного отображения иконок на кнопках в программе необходимо поместить графические файлы в одну папку с файлом кода программы.

Код можно запустить с помощью среды разработки. Например, с помощью PyCharm, Visual Studio или Visual Сode. Скачать их можно с официальных сайтов. Далее необходимо вставить код в текстовый документ и изменить его расширение на .py. Затем его можно будет открыть в скаченной среде разработки путем нажатия по нему двойным щелчком мыши.

## 5.2 Инструкция по работе с программой

### *5.2.1 Добавление устройства*

По нажатии на кнопку «Добавить устройство» пользователю откроется окно с выпадающим списком, в котором он может указать тип добавляемого устройства и пустыми полями, в которые он вносит данные об устройстве руководствуясь правилами ввода, которые можно прочитать в справке (см. Рисунок 12). По нажатии на кнопку «Добавить» в этом окне запись об устройстве будет добавлена в базу данных и отражена на главном окне при условии отсутствия ошибок ввода. В случае наличия каких-либо ошибок появится всплывающее окно, уведомляющее об ошибке.

### *5.2.2 Редактирование информации об устройстве*

После выбора устройства нажатием по нему левой кнопкой мыши и нажатия на кнопку «Редактировать» в верхней панели главного окна пользователю откроется окно «Редактировать устройство» (см. Рисунок 9) с полями, уже заполненными информацией о выбранном устройстве. Примечание: единовременно редактировать можно только одно устройство, то есть, если выбрать несколько устройств с зажатой клавишей «Ctrl», данная функция будет работать некорректно.

При изменении содержимого поля, которое содержит номер устройства, к которому подключено редактируемое устройство, данные об этом изменении вносятся в таблицу Logbook в базе данных и будут отражены в «Журнале событий» при его открытии. На рисунке 9 можно увидеть, что ID устройства родителя – 2, после изменения его на 3 (см. Рисунок 13) в «Журнале событий» появилась запись о том, что связь между устройствами была изменена (см. Рисунок 10).

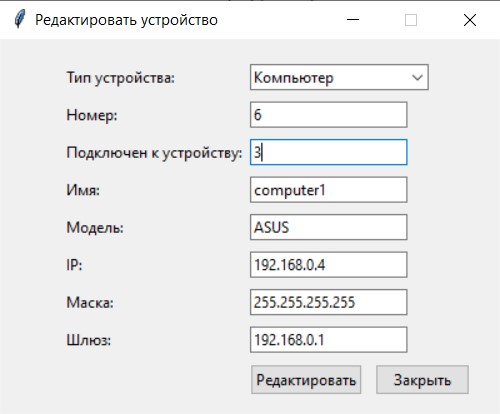


Рисунок 13 – Редактирование данных об устройстве

### *5.2.3 Просмотр «Журнала событий»*

По нажатии на кнопку «Журнал событий» в верхней панели главного окна поверх главного окна будет открыто окно «Журнал событий» содержащий историю изменения связей устройств в сети между собой (см. Рисунок 10). Для того чтобы удалить все записи из журнала, нажмите на кнопку «Очистить журнал»

### *5.2.4 Удаление устройства*

Для того чтобы удалить устройство необходимо выбрать его путем нажатия по нему левой кнопкой мыши и затем нажать на кнопку «Удалить устройство» в верхней панели главного окна. Затем в открывшемся диалоговом окне с подтверждением удаления нажмите кнопку «Да». Для того чтобы удалить несколько устройств сразу проделайте те же действия, но выбор устройств изначально осуществляйте с зажатой на клавиатуре клавишей «Ctrl».

# **6. Приложения**

## 6.1 Код программы

import tkinter as tk  
from tkinter import ttk  
from tkinter import messagebox as mb  
import sqlite3  
import datetime  
  
class Main(tk.Frame):  
 def \_\_init\_\_(self, root):  
 super().\_\_init\_\_(root)  
 self.root = root  
 self.init\_main()  
 self.db = db  
 self.view\_records()  
  
 def init\_main(self):  
 toolbar = tk.Frame(bg='#d7d8e0', bd=2)  
 toolbar.pack(side=tk.TOP, fill=tk.X)  
  
 self.menu = tk.Menu(self.root)  
 self.root.config(menu=self.menu)  
 self.fileMenu = tk.Menu(self.menu, tearoff=0)  
 self.menu.add\_cascade(label="Инфо", menu=self.fileMenu)  
 self.fileMenu.add\_command(label="Справка", command=self.open\_reference)  
  
 self.add\_img = tk.PhotoImage(file='add.gif')  
 btn\_open\_dialog = tk.Button(toolbar, text='Добавить устройство', command=self.open\_dialog, bg='#d7d8e0', bd=0,  
 compound=tk.TOP, image=self.add\_img)  
 btn\_open\_dialog.pack(side=tk.LEFT)  
  
 self.edit\_img = tk.PhotoImage(file='update.gif')  
 btn\_edit\_dialog = tk.Button(toolbar, text='Редактировать', bg='#d7d8e0', bd=0, image=self.edit\_img,  
 compound=tk.TOP, padx=10, command=self.open\_update)  
 btn\_edit\_dialog.pack(side=tk.LEFT)  
  
 self.delete\_img = tk.PhotoImage(file='delete.gif')  
 btn\_delete = tk.Button(toolbar, text='Удалить устройство', bg='#d7d8e0', bd=0, image=self.delete\_img,  
 compound=tk.TOP, padx=10, command=self.open\_delete\_dialog)  
 btn\_delete.pack(side=tk.LEFT)  
  
 self.logbook\_img = tk.PhotoImage(file='logbook.gif')  
 btn\_logbook = tk.Button(toolbar, text='Журнал событий', bg='#d7d8e0', bd=0, image=self.logbook\_img,  
 compound=tk.TOP, padx=10, command=self.open\_logbook)  
 btn\_logbook.pack(side=tk.LEFT)  
  
 self.tree = ttk.Treeview(self, columns=('ID', 'Parent\_ID', 'Type', 'Name', 'Model', 'IP', 'Mask', 'Gateway'),  
 height=30, selectmode='extended')  
  
 self.scrollbar = tk.Scrollbar(self)  
 self.tree.config(yscrollcommand=self.scrollbar.set)  
 self.scrollbar.config(command=self.tree.yview)  
 self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)  
  
 self.tree.column("#0", width=150, minwidth=150, stretch=tk.NO)  
 self.tree.column("ID", width=150, minwidth=150, stretch=tk.NO)  
 self.tree.column("Parent\_ID", width=180, minwidth=180, stretch=tk.NO)  
 self.tree.column("Type", width=150, minwidth=150, stretch=tk.NO)  
 self.tree.column("Name", width=150, minwidth=150, stretch=tk.NO)  
 self.tree.column("Model", width=150, minwidth=150, stretch=tk.NO)  
 self.tree.column("IP", width=150, minwidth=150, stretch=tk.NO)  
 self.tree.column("Mask", width=150, minwidth=150, stretch=tk.NO)  
 self.tree.column("Gateway", width=150, minwidth=150, stretch=tk.NO)  
  
 self.tree.heading("#0", text="Утройства", anchor=tk.W)  
 self.tree.heading("ID", text="ID устройства", anchor=tk.W)  
 self.tree.heading("Parent\_ID", text="ID родительского устройства", anchor=tk.W)  
 self.tree.heading("Type", text="Тип устройства", anchor=tk.W)  
 self.tree.heading("Name", text="Название", anchor=tk.W)  
 self.tree.heading("Model", text="Модель", anchor=tk.W)  
 self.tree.heading("IP", text="IP адресс", anchor=tk.W)  
 self.tree.heading("Mask", text="Маска подсети", anchor=tk.W)  
 self.tree.heading("Gateway", text="Шлюз", anchor=tk.W)  
  
 self.tree.pack(expand=True, fill=tk.BOTH)  
  
 def records(self, id, parent\_id, type, name, model, ip, mask, gateway, val\_ip, val\_m, val\_g):  
 if val\_ip != True:  
 mb.showerror('Ошибка', 'Неверный формат IP адреса')  
 elif val\_m != True:  
 mb.showerror('Ошибка', 'Неверный формат маски подсети')  
 elif val\_g != True:  
 mb.showerror('Ошибка', 'Неверный формат шлюза по умолчанию')  
 else:  
 self.db.insert\_data(id, parent\_id, type, name, model, ip, mask, gateway)  
 self.view\_records()  
  
 def update\_records(self, id, parent\_id, type, name, model, ip, mask, gateway, v\_ip, v\_m, v\_g):  
 if v\_ip != True:  
 mb.showerror('Ошибка', 'Неверный формат IP адреса')  
 elif v\_m != True:  
 mb.showerror('Ошибка', 'Неверный формат маски подсети')  
 elif v\_g != True:  
 mb.showerror('Ошибка', 'Неверный формат шлюза по умолчанию')  
 else:  
 try:  
 self.db.c.execute('''UPDATE Devices SET ID=?, Parent\_ID=?, Type=?, Name=?, Model=?, IP=?, Mask=?,   
 Gateway=? WHERE ID=?''',  
 (id, parent\_id, type, name, model, ip, mask, gateway,  
 self.tree.set(self.tree.selection()[0], '#1')))  
 self.db.conn.commit()  
 self.view\_records()  
 except:  
 mb.showerror('Ошибка', 'Проверьте корректность вводимых данных')  
  
 def view\_records(self):  
 dic = {'Маршрутизатор': 'id1', 'Коммутатор': 'id2', 'Сервер': 'id3', 'Компьютер': 'id4', 'МФУ': 'id5'}  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 for device in dic:  
 self.db.c.execute('''SELECT \* FROM Devices WHERE Type = "{0}"'''.format(device))  
 dic[device] = self.tree.insert("", "end", dic[device], text=device, open=True)  
 [self.tree.insert(dic[device], 'end', values=row) for row in self.db.c.fetchall()]  
  
 def delete\_records(self):  
 for selection\_item in self.tree.selection():  
 self.db.c.execute('''DELETE FROM Devices WHERE ID=?''', (self.tree.set(selection\_item, '#1'),))  
 self.db.conn.commit()  
 self.view\_records()  
  
 def open\_delete\_dialog(self):  
 if mb.askyesno('Подтверждение', 'Вы уверены, что хотите удалить запись?'):  
 self.delete\_records()  
  
 def open\_update(self):  
 Update()  
  
 def open\_logbook(self):  
 Logbook(t=True)  
  
 def open\_dialog(self):  
 Device()  
  
 def open\_reference(self):  
 Reference()  
  
class Reference(tk.Toplevel):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.init\_reference()  
  
 def init\_reference(self):  
 self.geometry("700x400+400+300")  
 self.resizable(False, False)  
 lbl = tk.Label(self,  
 text='Здравствуй, дорогой пользователь!'  
 '\n\nДля того чтобы добавить устройство, нажмите на кнпоку "Добавить устройство" в верхней '  
 'панели главного окна,'  
 '\nвведите необходимые данные в соответсвующие поля, предварительно выбрав тип устройства '  
 'из списка'  
 '\nзатем нажмите кнопку "Добавить".'  
 '\n\nДля того чтобы обновить информацию об устройстве, сначала нажмите на строку с '  
 'устройством, затем нажмите на'  
 '\nкнопку "Редактировать" в верхней панели главного окна, совершите все изменения и нажмите'  
 ' на кнопку'  
 '\n"Редактировать".'  
 '\n\nДля того чтобы удалить одно или несколько устройств, нажмите на строку с устройством, '  
 'затем нажмите на'  
 '\nкнопку "Удалить устройство". В окне с подтверждением нажмите "Да"'  
 '\n\nПримечание: Чтобы удалить сразу несколько устройств выберите их с зажатой на '  
 'клавиатуре клавишей "Ctrl".'  
 '\n\nВнимание!'  
 '\nПри заполнении текстовых полей в окнах "Добавить устройство" и "Редактировать '  
 'устройство" следует соблюдать'  
 '\nнекоторые правила ввода:'  
 '\n\t1) ID устройств не могут повторяться'  
 '\n\t2) Названия устройств не могут повторяться'  
 '\n\t3) IP адреса устройств не могут повторяться'  
 '\n\t4) IP адрес и шлюз устройства должны быть введены в следующем формате: х.х.х.х,'  
 '\n\tгде x - число от 0 до 255',  
 justify=tk.LEFT,  
 font=("Arial Bold", 10))  
 lbl.pack()  
 btn = ttk.Button(self, text='Ок', command=self.destroy)  
 btn.pack()  
  
class Device(tk.Toplevel):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.init\_device()  
 self.view = app  
  
 def init\_device(self):  
 self.title('Добавить устройство')  
 self.geometry('400x300+400+300')  
 self.resizable(False, False)  
  
 label\_id = tk.Label(self, text='Тип устройства:')  
 label\_id.place(x=50, y=20)  
 label\_id = tk.Label(self, text='Номер:')  
 label\_id.place(x=50, y=50)  
 label\_parent = tk.Label(self, text='Подключен к устройству:')  
 label\_parent.place(x=50, y=80)  
 label\_name = tk.Label(self, text='Имя:')  
 label\_name.place(x=50, y=110)  
 label\_model = tk.Label(self, text='Модель:')  
 label\_model.place(x=50, y=140)  
 label\_ip = tk.Label(self, text='IP:')  
 label\_ip.place(x=50, y=170)  
 label\_mask = tk.Label(self, text='Маска:')  
 label\_mask.place(x=50, y=200)  
 label\_gateway = tk.Label(self, text='Шлюз:')  
 label\_gateway.place(x=50, y=230)  
  
 self.combobox = ttk.Combobox(self, values=[u'Маршрутизатор', u'Коммутатор', u'Сервер', u'Компьютер', u'МФУ'],  
 state='readonly')  
 self.combobox.current(0)  
 self.combobox.place(x=200, y=20)  
 self.entry\_id = ttk.Entry(self)  
 self.entry\_id.place(x=200, y=50)  
 self.entry\_parent = ttk.Entry(self)  
 self.entry\_parent.place(x=200, y=80)  
 self.entry\_name = ttk.Entry(self)  
 self.entry\_name.place(x=200, y=110)  
 self.entry\_model = ttk.Entry(self)  
 self.entry\_model.place(x=200, y=140)  
 self.entry\_ip = ttk.Entry(self)  
 self.entry\_ip.place(x=200, y=170)  
 self.entry\_mask = ttk.Entry(self)  
 self.entry\_mask.place(x=200, y=200)  
 self.entry\_gateway = ttk.Entry(self)  
 self.entry\_gateway.place(x=200, y=230)  
  
 btn\_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)  
 btn\_cancel.place(x=300, y=260)  
  
 self.btn\_add = ttk.Button(self, text='Добавить')  
 self.btn\_add.place(x=220, y=260)  
  
 self.btn\_add.bind('<Button-1>', lambda event: self.view.records(self.entry\_id.get(),  
 self.entry\_parent.get(),  
 self.combobox.get(),  
 self.entry\_name.get(),  
 self.entry\_model.get(),  
 self.entry\_ip.get(),  
 self.entry\_mask.get(),  
 self.entry\_gateway.get(),  
 self.validate\_address(self.entry\_ip.get()),  
 self.validate\_mask(self.entry\_mask.get()),  
 self.validate\_address(self.entry\_gateway.get())))  
  
  
 self.grab\_set()  
 self.focus\_set()  
  
 def validate\_address(self, s):  
 a = s.split('.')  
 if len(a) != 4:  
 return False  
 for x in a:  
 if not x.isdigit():  
 return False  
 i = int(x)  
 if i < 0 or i > 255:  
 return False  
 return True  
  
 def validate\_mask(self, mask):  
 a = mask.split('.')  
 dic = {255, 254, 252, 248, 240, 224, 192, 128, 0}  
 if len(a) != 4:  
 return False  
 if int(a[0]) == 255:  
 last\_i = 255  
 b = True  
 for x in range(1, 4):  
 if a[x].isdigit() and (int(a[x]) in dic) and (int(a[x]) <= last\_i):  
 i = int(a[x])  
 if b is True and last\_i == 255 and i != 255:  
 b = False  
 elif last\_i != 255 and i != 255 and i != 0:  
 return False  
 last\_i = i  
 else:  
 return False  
 return True  
 else:  
 for i in range(1, 4):  
 if int(a[i]) != 0:  
 return False  
 else:  
 return True  
  
class Logbook(tk.Toplevel):  
 def \_\_init\_\_(self, t):  
 if t:  
 super().\_\_init\_\_()  
 self.init\_logbook()  
 self.view = app  
 self.db = DB()  
 self.view\_notes()  
 else:  
 self.db = DB()  
  
 def init\_logbook(self):  
 self.title('Журнал событий')  
 self.geometry('750x400+400+300')  
 self.resizable(False, False)  
  
 btn\_delete = ttk.Button(self, text='Очистить журнал', command=self.open\_delete\_dialog)  
 btn\_delete.pack(fill=tk.X)  
  
 self.tree1 = ttk.Treeview(self, columns=('ID', 'Old\_Parent\_ID', 'New\_Parent\_ID', 'DateTime'), height=25,  
 show='headings', selectmode="extended")  
  
 self.scrollbar = tk.Scrollbar(self)  
 self.tree1.config(yscrollcommand=self.scrollbar.set)  
 self.scrollbar.config(command=self.tree1.yview)  
 self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)  
  
 self.tree1.column("ID", width=100, stretch=tk.NO)  
 self.tree1.column("Old\_Parent\_ID", width=250, minwidth=250, stretch=tk.NO)  
 self.tree1.column("New\_Parent\_ID", width=250, minwidth=250, stretch=tk.NO)  
 self.tree1.column("DateTime", width=150, minwidth=150, stretch=tk.NO)  
  
 self.tree1.heading("ID", text="ID устройства")  
 self.tree1.heading("Old\_Parent\_ID", text="ID старого родительского устройства")  
 self.tree1.heading("New\_Parent\_ID", text="ID нового родительского устройства")  
 self.tree1.heading("DateTime", text="Время изменения")  
  
 self.tree1.pack()  
  
 self.grab\_set()  
 self.focus\_set()  
  
 def delete\_notes(self):  
 self.db.c.execute('''DELETE FROM Logbook''')  
 self.db.c.execute('''UPDATE sqlite\_sequence SET seq=0 WHERE Name="Logbook"''')  
 self.db.conn.commit()  
 self.view\_notes()  
  
 def open\_delete\_dialog(self):  
 if mb.askyesno('Подтверждение', 'Вы уверены, что хотите очистить журнал?'):  
 self.delete\_notes()  
  
 def notes(self, id, old\_parent\_id, new\_parent\_id, datetime, t1):  
 self.db.insert\_note(id, old\_parent\_id, new\_parent\_id, datetime)  
 if t1:  
 self.view\_notes()  
  
 def view\_notes(self):  
 self.db.c.execute('''SELECT Device\_ID, Old\_Parent\_ID, New\_Parent\_ID, Datetime FROM Logbook''')  
 [self.tree1.delete(i) for i in self.tree1.get\_children()]  
 [self.tree1.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
class Update(Device):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.init\_edit()  
 self.view = app  
 self.db = db  
 self.log = log  
 self.default\_data()  
  
 def init\_edit(self):  
 self.title('Редактировать устройство')  
 btn\_edit = ttk.Button(self, text='Редактировать')  
 btn\_edit.place(x=200, y=260)  
 btn\_edit.bind('<Button-1>', lambda event: self.check\_difference())  
 btn\_edit.bind('<Button-1>', lambda event: self.view.update\_records(self.entry\_id.get(),  
 self.entry\_parent.get(),  
 self.combobox.get(),  
 self.entry\_name.get(),  
 self.entry\_model.get(),  
 self.entry\_ip.get(),  
 self.entry\_mask.get(),  
 self.entry\_gateway.get(),  
 self.validate\_address(self.entry\_ip.get()),  
 self.validate\_mask(self.entry\_mask.get()),  
 self.validate\_address(self.entry\_gateway.get())),  
 add="+")  
  
 self.btn\_add.destroy()  
  
 def default\_data(self):  
 self.db.c.execute('''SELECT \* FROM Devices WHERE ID=?''',  
 (self.view.tree.set(self.view.tree.selection()[0], '#1'),))  
 row = self.db.c.fetchone()  
 dic = {0: 'Маршрутизатор', 1: 'Коммутатор', 2: 'Сервер', 3: 'Компьютер', 4: 'МФУ'}  
 for i in range(5):  
 if row[2] == dic.get(i):  
 self.combobox.current(i)  
 break  
 self.entry\_id.insert(0, row[0])  
 self.entry\_parent.insert(0, row[1])  
 self.old\_parent\_id = self.entry\_parent.get()  
 self.entry\_name.insert(0, row[3])  
 self.entry\_model.insert(0, row[4])  
 self.entry\_ip.insert(0, row[5])  
 self.entry\_mask.insert(0, row[6])  
 self.entry\_gateway.insert(0, row[7])  
  
 def get\_time(self):  
 datetime\_now = datetime.datetime.now().strftime("%d.%m.%y %H:%M:%S")  
 return datetime\_now  
  
 def check\_difference(self):  
 if self.entry\_parent.get() != self.old\_parent\_id:  
 self.log.notes(Logbook(t=False), self.entry\_id.get(), self.old\_parent\_id, self.entry\_parent.get(),  
 self.get\_time(), t1=False)  
  
  
class DB:  
 def \_\_init\_\_(self):  
 self.conn = sqlite3.connect('network\_directory.db')  
 self.c = self.conn.cursor()  
 self.c.execute(  
 '''CREATE TABLE IF NOT EXISTS Devices (ID INTEGER PRIMARY KEY, Parent\_ID INTEGER, Type TEXT,   
 Name TEXT UNIQUE, Model TEXT, IP TEXT UNIQUE, Mask TEXT, Gateway TEXT)''')  
 self.conn.commit()  
 self.c.execute(  
 '''CREATE TABLE IF NOT EXISTS Logbook (ID INTEGER PRIMARY KEY AUTOINCREMENT, Device\_ID INTEGER,   
 Old\_Parent\_ID INTEGER, New\_Parent\_ID INTEGER, Datetime TEXT)''')  
 self.conn.commit()  
  
  
 def insert\_data(self, id, parent\_id, type, name, model, ip, mask, gateway):  
 try:  
 self.c.execute('''INSERT INTO Devices(ID, Parent\_ID, Type, Name, Model, IP, Mask, Gateway)   
 VALUES (?, ?, ?, ?, ?, ?, ?, ?)''',  
 (id, parent\_id, type, name, model, ip, mask, gateway))  
 self.conn.commit()  
 except sqlite3.IntegrityError:  
 mb.showerror('Ошибка', 'Проверьте корректность вводимых данных')  
  
 def insert\_note(self, id, old\_parent\_id, new\_parent\_id, datetime):  
 try:  
 self.c.execute('''INSERT INTO Logbook(Device\_ID, Old\_Parent\_ID, New\_Parent\_ID, Datetime)   
 VALUES (?, ?, ?, ?)''',  
 (id, old\_parent\_id, new\_parent\_id, datetime))  
 self.conn.commit()  
 except sqlite3.IntegrityError:  
 mb.showerror('Ошибка', 'Проверьте корректность вводимых данных')  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 db = DB()  
 app = Main(root)  
 log = Logbook  
 app.pack()  
 root.title("Network directory")  
 root.geometry("1400x650+80+100")  
 root.mainloop()

# **7. Библиографический список**

1. <https://python-scripts.com/sqlite>
2. <https://python-scripts.com/tkinter>
3. <https://habr.com/ru/post/133337/>
4. <https://docs.python.org/3/library/tkinter.html>
5. Эрик Мэтиз Изучаем Python. Программирование игр, визуализация данных, вэб-приложения. 2-е изд.-СПБ.:Питер, 2017