



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский технологический университет»
(МГУПИ)

Институт ИКБСП специальность (направление) 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту (работе) на тему:

Разработка агрегатора интернет-радио

Студент _____ Бердов С.В.
подпись, дата

Группа БСБО-02-15 шифр 15Б0642

Проект (работа) защищен(а) на оценку _____

Руководитель проекта (работы) _____ Лебедев А.С.
подпись, дата

Члены комиссии _____ Лебедев А.С.
подпись, дата

подпись, дата инициалы и фамилия

подпись, дата инициалы и фамилия

МОСКВА 2017 г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский технологический университет»
(МГУПИ)

Институт ИКБСП специальность (направление) 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

Студент Бердов С.В. шифр 15Б0642 группа БСБО-02-15

1 Тема:

Разработка агрегатора интернет-радио

2 Срок представления проекта (работы) к защите 23.12.2017 г.

3 Исходные данные для разработки

Аудио-поток, ОС Linux, интерпретатор Python

4 Содержание пояснительной записки:

Титульный лист

Содержание

Введение

Техническое задание

Заключение

Список использованных источников

Приложение

Руководитель проекта (работы) _____ Лебедев А.С.

подпись, дата

Задание принял к исполнению _____ Бердов С.В.

подпись, дата

Содержание

Введение	4
Цели	6
Задачи	7
Техническое задание	8
Основания для разработки.....	8
Назначение разработки	8
Требования к программе или программному изделию.....	8
Требования к функциональным характеристикам	8
Требования к составу и параметрам технических средств.....	14
Требования к информационной и программной совместимости	15
Стадии и этапы разработки	19
Модули приложения	20
Форматы входных и выходных данных	20
Заключение.....	22
Приложение 1.....	24
Исходный код программы:	24
Radiolib.py	24
Syles.css.....	31

Введение

В данной курсовой работе будет разработан агрегатор интернет-радио. Агрегатор интернет-радио будет использоваться на операционных системах (ОС) Linux. Предназначение - аккумулирование и воспроизведение набора интернет радиостанций.

Интернет-радио – группа технологий передачи потоковых аудиоданных через сеть Интернет для осуществления широкоэмительных передач. Также, в качестве термина интернет-радио или веб-радио может пониматься радиостанция, использующая для вещания технологию потокового вещания в глобальной сети Интернет. Также, в качестве термина интернет-радио или веб-радио может пониматься радиостанция, использующая для вещания технологию потокового вещания в глобальной сети Интернет.

В технологической основе системы лежит три элемента:

Станция — генерирует аудиопоток и направляет его серверу.

Сервер (повторитель потока) — принимает аудиопоток от станции и перенаправляет его копии всем подключённым к серверу клиентам, по сути является репликатором данных.

Клиент — принимает аудиопоток от сервера и преобразует его в аудиосигнал, который и слышит слушатель интернет-радиостанции.

Потоковое мультимедиа требует доставки данных с теми же временными характеристиками одному или нескольким получателям. Транспортировка потока обслуживается двумя тесно взаимосвязанными протоколами. Протокол Real-time Transport Protocol (RTP) делит поток на пакеты, каждый из которых содержит отметку времени, порядковый номер и информацию об отправителе. Спецификация RTP также определяет протокол RTP Control Protocol (RTCP), по которому осуществляется обратная связь с информацией о качестве передачи и идентификационных данных участников потока.

MP3 – это наиболее популярный формат хранения и передачи информации в цифровой форме, использующий компрессию сигнала. Обеспечивает высокое качество звука при сравнительно небольших размерах файла. Высокая степень сжатия в MP3 достигается за счёт достаточно сложного алгоритма кодирования. Используются как математические методы компрессии, так и особенности человеческого слуха (психоакустическая модель): эффект маскировки слабого звука одной частоты более громким звуком такой же или соседней частоты, понижение чувствительности уха к тихому звуку сразу после громкого, невосприимчивость к звукам ниже определённого уровня громкости. Поток звука при кодировании разбивается на равные участки (фреймы). Каждый из фреймов кодируется отдельно со своими параметрами и содержит заголовок, в котором эти параметры указаны.

Сжатие может быть выполнено с разным качеством и соответственно размером конечного файла. Степень сжатия характеризуется битрейтом

(bitrate) — количество передаваемой за единицу времени информации. Файлы MP3 обычно закодированы с битрейтом от 64 до 320 килобит в секунду (kbps или kb/s), а также с переменным битрейтом (VBR) — когда для каждого фрейма используется свой, оптимальный для данного участка, битрейт.

Исходный сигнал с помощью фильтров разделяется на несколько частотных диапазонов, для каждого диапазона определяется величина маскирующего эффекта от соседних диапазонов и предыдущего фрейма, несущественные сигналы игнорируются. Для оставшихся данных для каждого диапазона определяется, сколькими битами можно пожертвовать, чтобы потери были ниже величины маскирующего эффекта. На этом работа психоакустической модели завершается, а итоговый поток дополнительно сжимается по алгоритму Хаффмана (аналогично RAR архиватору).

Разрабатываемый радио-агрегатор будет использовать протокол RTP и RTCP, а также поддерживать формат хранения MP3.

Цели

Разработать агрегатор интернет-радио с возможностями воспроизводить, приостанавливать, останавливать аудиопоток, загружать и сохранять записи с аудио потоками, осуществлять добавление, удаление и изменение записей аудиопотока.

Задачи

1. Изучить библиотеки PyGobject, GTK+, Gstreamer.
2. Реализовать графический интерфейс приложения с применением GTK+.
3. Реализовать управление воспроизведением потока интернет-радио с применением Gstreamer.
4. Реализовать добавление, удаление, сохранение, изменение и загрузку аудиозаписи.

Техническое задание

Основания для разработки

Учебный план по дисциплине «Технология программирования в среде «Python», направление бакалавриата «09.03.02», МТУ, 2017.

Назначение разработки

Приложение предназначено для аккумуляирования и воспроизведения набора интернет-радиостанций, управления записями аудиопотоков.

Требования к программе или программному изделию

Требования к функциональным характеристикам

Воспроизведение аудио-потока. Наличие минимального графического интерфейса, такого как: кнопки воспроизведения для каждой радиостанции, регулятор громкости, кнопка паузы. Наличие возможности пользователю добавлять, удалять, изменять записи радиостанций, а также сохранять, в созданный пользователем файл с расширением .txt, введенные через окно «добавления записи» наборы радиостанций и загружать созданный пользователем файл с расширением .txt. Формат текстового файла представлен в пункте **«Формат сохраняемого/загружаемого текстового файла»**

Действия, инициируемые элементами управления:

- Открыть файл – открытие файла формата .txt для загрузки списка радиостанций.
- Сохранить файл – создание/изменение файла формата .txt для сохранения списка радиостанций.
- Закрытие файла – закрытие списка радиостанций без сохранения в файл.
- Выход – закрытие программы.
- Добавление записи – добавление новой записи в конец списка радиостанций.
- Изменение записи – изменение выбранной записи из списка радиостанций.
- Удаление записи – удаление выбранной записи из списка радиостанций.
- О программе – сведенье о данной программе.

При запуске программы должно отображаться главное окно. Для открытия файла используется меню «File» – «Open file». В открывшемся диалоговом окне выбирается необходимый файл с данными о радиостанциях. Данные из текстового файла будут загружены. (рис.№1, 2, 3)

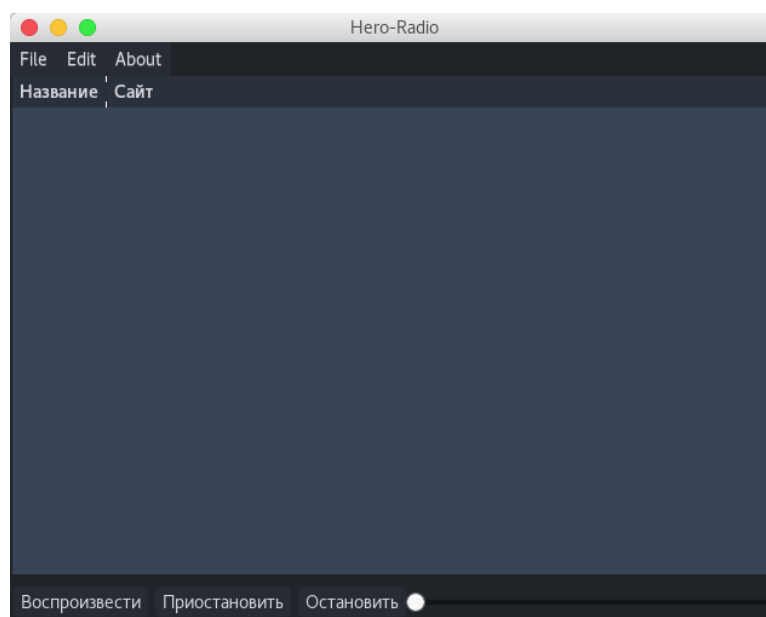


Рисунок 1 – Старт программы

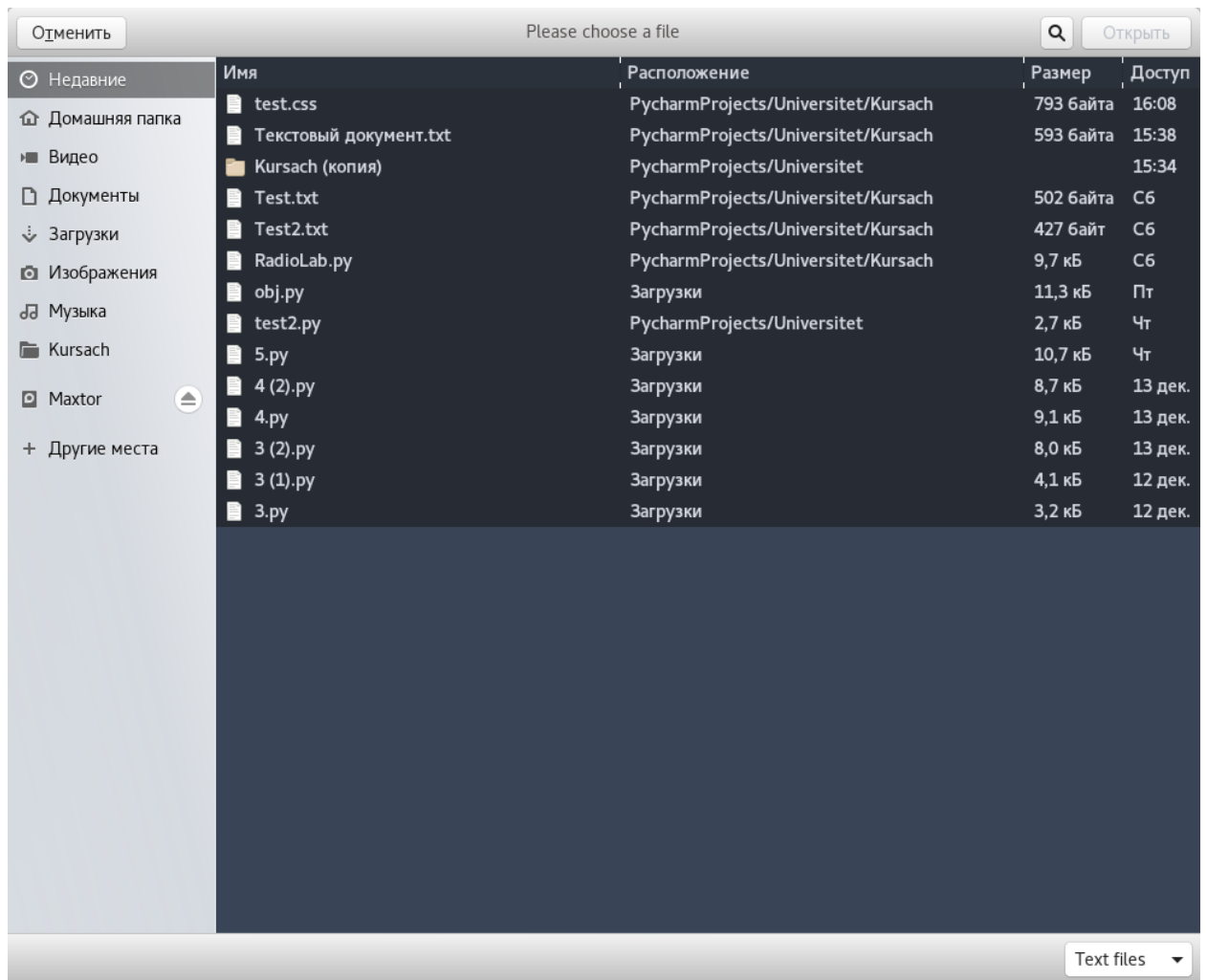


Рисунок 2 – Окно открытия файла

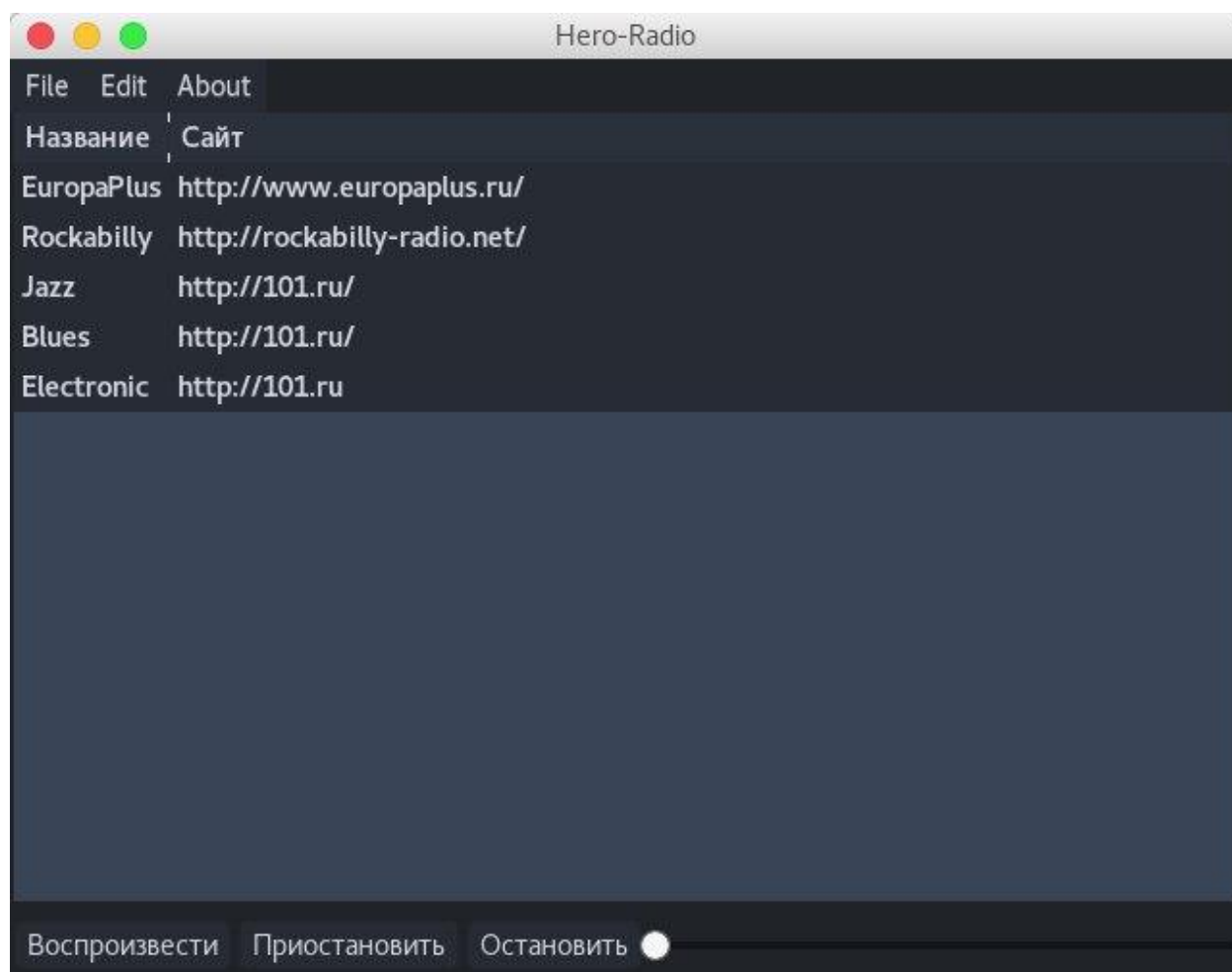


Рисунок 3 – Окно загруженных данных в программу

«Воспроизвести» – воспроизведение выбранной радиостанции.

«Приостановить» – приостановка выбранной радиостанции для дальнейшего воспроизведения с того же места, на котором радиостанция была приостановлена.

«Остановить» – остановка радиостанции.

Горизонтальный слайдер – регулировка громкости.

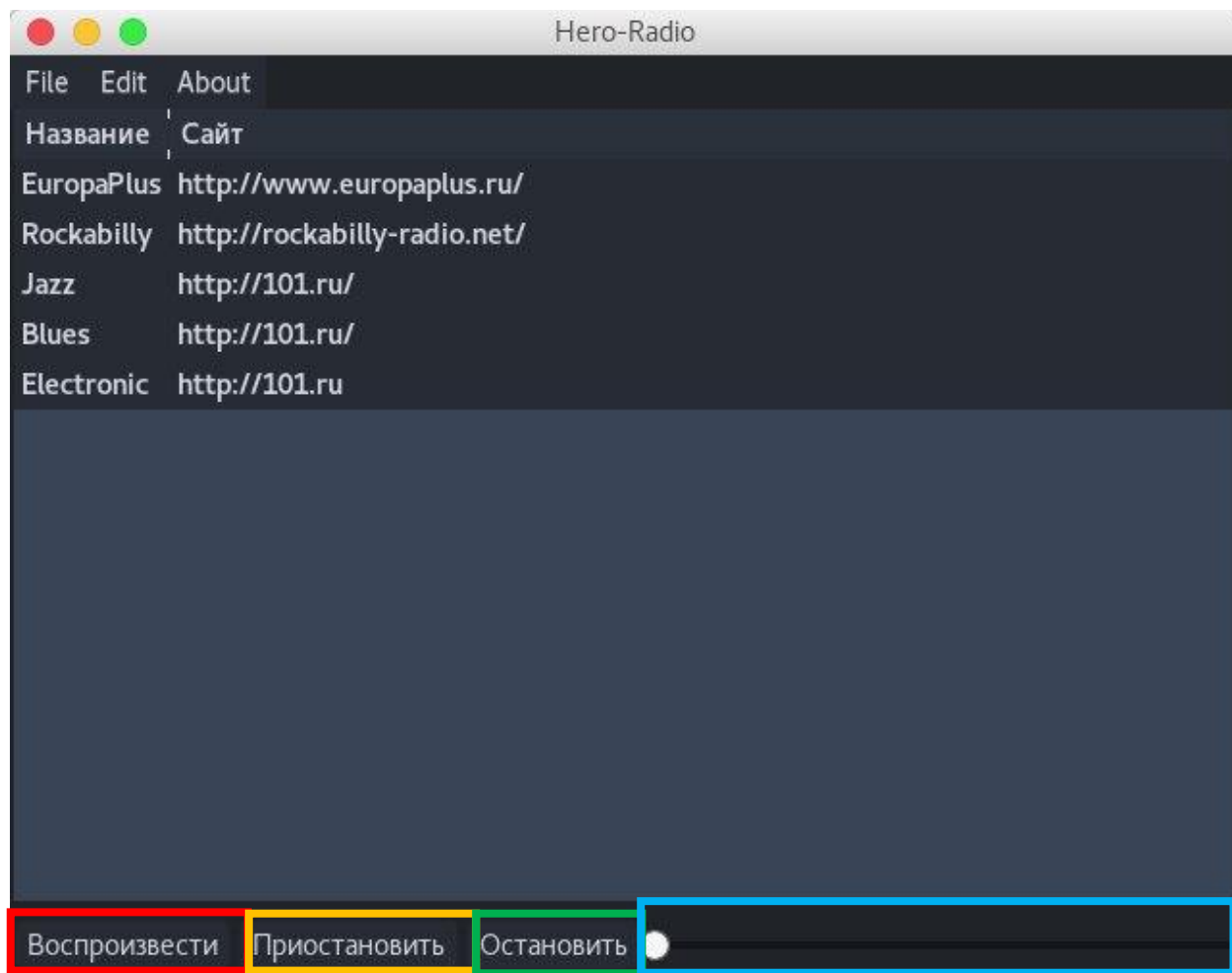


Рисунок 4 – Окно с пояснением элементов управления радиостанциями

Для добавления запись в список радиостанций используется меню «Edit» – «Add record».

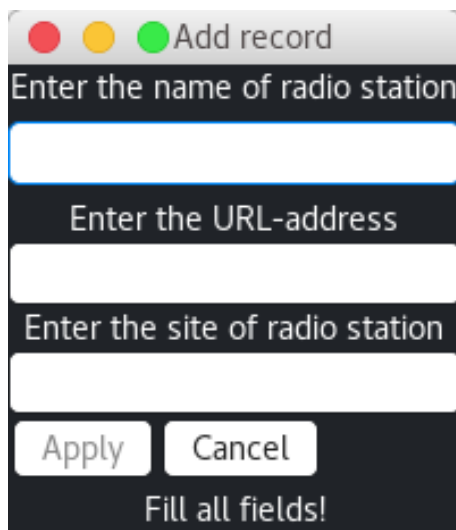


Рисунок 5 – Окно добавления записи

Все поля в данном окне являются обязательными к заполнению.

«Apply» – добавление новой записи в конец списка радиостанций.

«Cancel» – закрытие окна добавления новой записи в список радиостанций.

Для удаления выбранной записи из списка радиостанций используется меню «Edit» – «Delete record».

Для изменения выбранной записи используется меню «Edit» – «Change record».

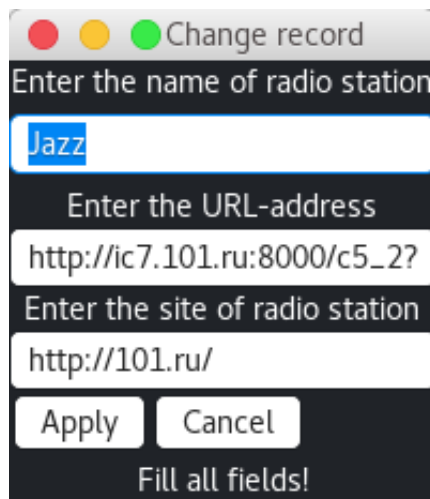


Рисунок 6 – Окно изменения записи

Поля этого окна заполняются автоматически после выбора записи в главном меню.

«Apply» – изменение выбранной записи на новую запись.

«Cancel» – закрытие окна изменения записи.

Для сохранения добавленных записей используется меню «File» – «Save file».

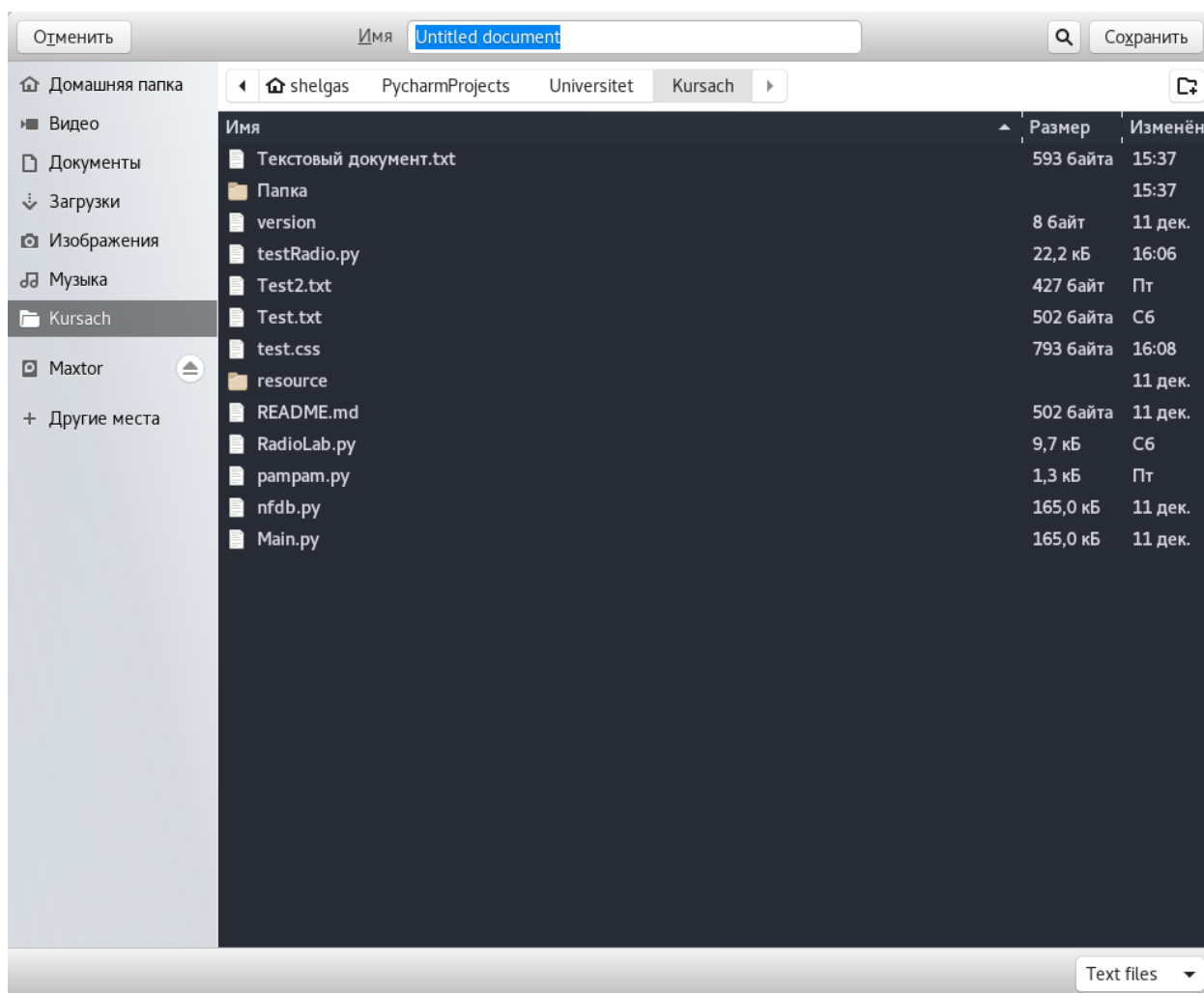


Рисунок 7 – Окно сохранения файла

Требования к составу и параметрам технических средств

- Процессор: Intel Core i3-4340 / AMD FX-6300
- Оперативная память: 1 GB ОЗУ
- Сеть: Широкополосное подключение к интернету
- Место на диске: 1 GB
- Наличие монитора

Требования к информационной и программной совместимости

- Наличие ОС Linux Fedora 25 и новее
- Наличие библиотеки Gstreamer версии 1.0 и новее
- Наличие библиотеки PyGObject версии 2.24.0 и новее
- Наличие GTK+ версии 3.0 и новее

Архитектура GStreamer

В GStreamer есть несколько основных компонентов:

- Элементы
- Pads
- Контейнеры bin и pipeline

Элементы

Практически все в GStreamer является элементом. Все, начиная от обычных источников потоков (filesrc, alsasrc, и т. п.), обработчиками потоков (демультиплексоры, декодеры, фильтры, и т. п.) и заканчивая конечными устройствами вывода (alsasink, fakesink, filesink, и т. п.).

Pads

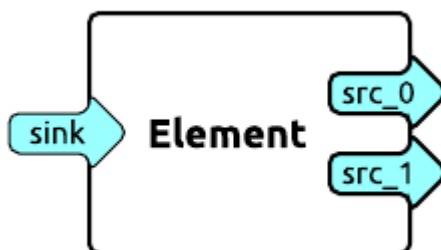


Рисунок 8 – Схема элемента

Pad — это некая точка подключения элемента к другому элементу, если более просто — это входы и выходы элемента. Обычно они именуются «sink» — вход и «src» — выход.

Элементы всегда имеют как минимум один pad. Например, filesrc — элемент для чтения данных из файловой системы — имеет только один pad с названием «src», т. к. он не имеет входа, а может только превращать поток из файловой системы в внутреннее представление с которым уже будут работать другие элементы. Так же и элемент alsasink, он имеет один pad с названием «sink», т. к. он может только принимать внутренний поток и выводить его на звуковую карту через alsa. Элементы из разряда «filters» имеют две и более точек подключения. Например, элемент volume имеет pad с именем «sink», на

который поступает поток, внутри этого элемента трансформируется (изменяется громкость), и через раd с названием «src» уже продолжает свой путь. Так же имеются элементы где может быть несколько как входов, так и выходов.

Контейнеры

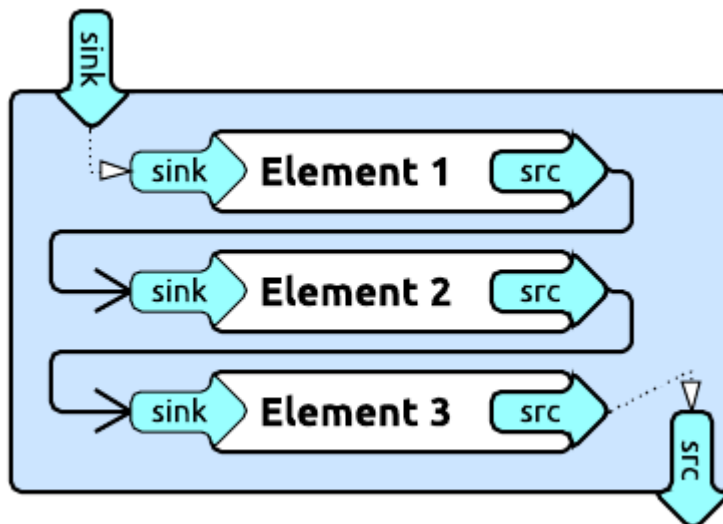


Рисунок 9 – Схема контейнера

Внутри контейнеров элементы проводят свой жизненный цикл. Контейнер управляет рассылкой сообщений от элемента к элементу, управляет статусами элементов. Контейнеры делятся на два вида:

- Bin
- Pipeline

Pipeline является контейнером верхнего уровня, он управляет синхронизацией элементов, рассылает статусы. Например, если pipeline установить статус PAUSED, этот статус будет автоматически разослан всем элементам, которые находятся внутри него. Pipeline является реализацией Bin.

Bin — простой контейнер, который управляет рассылкой сообщений от элемента к элементу которые находятся внутри него. Bin обычно используется для создания группы элементов, которые должны совершать какое-либо действие. Например, decodebin — элемент для декодирования потока, который автоматически выбирает нужные элементы для обработки потока в зависимости от типа данных (vorbisdec, theoraec, и т. п.)

Также имеются законченные самодостаточные контейнеры, такие как playbin. Playbin же, по сути является полноценным плеером, который содержит в своем составе все нужные элементы для воспроизведения аудио и видео, но он не очень гибок.

GTK+

GTK+ — это фреймворк для создания кроссплатформенного графического интерфейса пользователя (GUI). Наряду с Qt он является одной из двух наиболее популярных на сегодняшний день библиотек для X Window System.

Изначально эта библиотека была частью графического редактора GIMP, но позже стала независимой и приобрела популярность. GTK+ — это свободное ПО, распространяемое на условиях GNU LGPL и позволяющее создавать как свободное, так и проприетарное программное обеспечение.

GTK+ написан на языке Си, однако несмотря на это, является объектно-ориентированным. Также можно использовать обёртки для следующих языков: Ada, C, C++, C#, D, Erlang, Fortran, GOB, Genie, Haskell, FreeBASIC, Free Pascal, Java, JavaScript, Lua, OCaml, Perl, PHP, PureBasic, Python, R, Ruby, Smalltalk, Tcl, Vala.

Внутри GTK+ состоит из двух компонентов: GTK, который содержит набор виджетов (кнопка, метка и т.д.) и GDK, который занят выводом результата на экран.

Внешний вид приложений может меняться программистом и/или пользователем. По-умолчанию приложения выглядят нативно, т.е. так же, как и другие приложения в этой системе. Кроме того, начиная с версии 3.0, можно менять внешний вид элементов с помощью CSS.

PyGObject

PyGObject — это пакет Python'а, который обеспечивает привязки для библиотек на основе GObject, таких как GTK +, GStreamer, WebKitGTK +, GLib, GIO и многие другие.

Он поддерживает Linux, Windows и macOS и работает с Python 2.7+, а также с Python 3.4+.

С помощью PyGObject пишутся программы на Python для GNOME или приложения на Python с GUI с помощью GTK +.

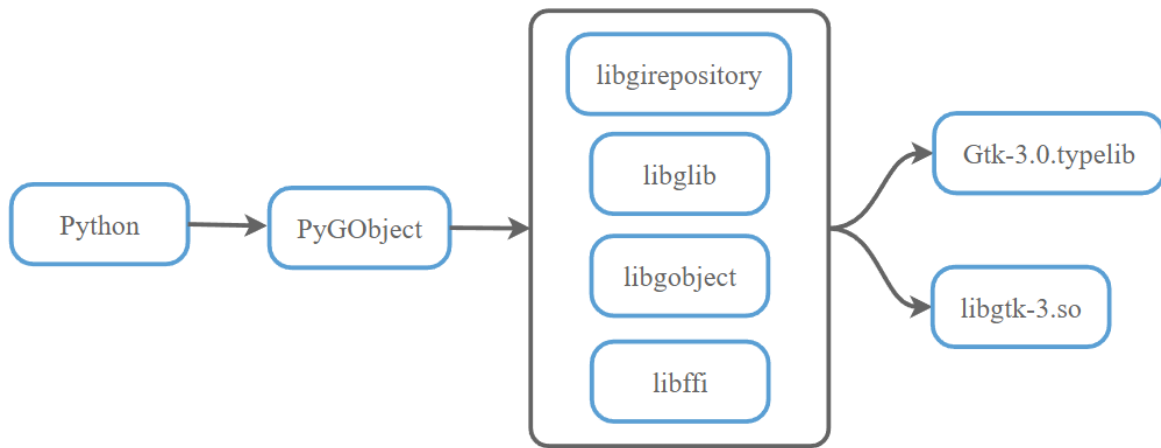


Рисунок 10 – Схема использования библиотек PyGObject

PyGObject использует glib, gobject, girepository, libffi и другие библиотеки для доступа к библиотеке C (libgtk-3.so) в сочетании с дополнительными метаданными из сопроводительного файла typelib (Gtk-3.0.typelib) и динамически предоставляет интерфейс Python на основе этой информации.

Стадии и этапы разработки

1. Анализ предметной области.
2. Разработка технического задания на разработку Программы.
3. Разработка общей архитектуры программы.
4. Разработка программного кода, реализующего функциональные требования к программе.

Разработка программного кода должна происходить на языке программирования Python версии 3.6.2 со средой выполнения PyCharm версии 2017.3.

Модули приложения

Файл Radiolib.py содержит основной код программы в который входит:

- Главное окно
- Главное меню
- Окно загрузки
- Окно сохранения
- Окно добавления записи
- Окно изменения записи

Файл Style.css содержит в себе стили для изменения внешнего вида GUI.

Добавить описание форматов входного аудиопотока, которые поддерживаются.

Форматы входных и выходных данных

Формат аудиопотока - MP3.

MP3 это наиболее популярный формат хранения и передачи информации в цифровой форме, использующий компрессию сигнала.

Формат сохраняемого/загружаемого текстового файла

Список радиостанций хранится в текстовом файле со структурой:

```
<Название>,<URL-адрес>,<Описание>\n  
<Название>,<URL-адрес>,<Описание>\n  
<Название>,<URL-адрес>,<Описание>\n
```

“,” – обязательный разделитель между элементами одной записи, “\n” – переход на следующую строку и обязательный разделитель между записями.

Пример содержимого текстового файла:

EuropaPlus,<http://ep256.streamr.ru>,<http://www.europaplus.ru/>
Rockabilly,http://lin3.ash.fast-serv.com:6026/stream_96,<http://rockabilly-radio.net/>
Jazz,http://ic7.101.ru:8000/c5_2?userid=0&setst=opt6ncajj8vu7a7bhiqk5j0up3&tok=29862838N0Tkm%2FjEeJJR0P2743Qo8g%3D%3D2&city=283094,<http://101.ru/>
Blues,http://ic7.101.ru:8000/c10_5?userid=0&setst=opt6ncajj8vu7a7bhiqk5j0up3&tok=29862838N0Tkm%2FjEeJJR0P2743Qo8g%3D%3D3&city=283094,<http://101.ru/>
Electronic,http://ic7.101.ru:8000/c15_20?userid=0&setst=opt6ncajj8vu7a7bhiqk5j0up3&tok=29862838N0Tkm%2FjEeJJR0P2743Qo8g%3D%3D6&city=283094,<http://101.ru>

Заключение

1. Изучены библиотеки PyGobject, GTK+, Gstreamer.
2. Реализован графический интерфейс приложения с применением GTK+.
3. Реализовано управление воспроизведением потока интернет-радио с применением Gstreamer.
4. Реализовано добавление, удаление, сохранение, изменение и загрузку аудиозаписи.

Список использованных источников:

1. Gstreamer Team. GStreamer Manual // gstreamer.freedesktop.org: Официальный сайт разработчика Gstreamer; URL: <https://gstreamer.freedesktop.org/documentation/index.html> (дата обращения: 11.12.2017).
2. Henstridge, J., & Dahlin, J. PyGobject Manual // pygobject.readthedocs.io : официальный сайт разработчика PyGobject; URL: <https://media.readthedocs.org/pdf/pygobject/latest/pygobject.pdf> (дата обращения: 14.12.2017).
3. PyGTK Team. PyGTK Manual // pygtk.org :Официальный сайт разработчиков PyGtk; URL: <http://www.pygtk.org/> (дата обращения: 16.12.2017).
4. Taymans, W., Baker, S., Wingo, A., Bultje, R., & Kost , S. GStreamer Application Development Manual.// benow.ca/docs: Сайт расположения файла; URL: <http://benow.ca/docs/gstreamer-manual.pdf> (дата обращения: 16.12.2017).
5. Лутц, М. Программирование на Python. Том 1, 4-е издание . Москва: Символ-Плюс 2011.
6. Эндрю Таненбаум, Дэвид Уэзеролл Компьютерные сети. 5-е издание Санкт-Петербург: Питер 2012.

Приложение 1

Исходный код программы:

Radiolib.py

```
import sys
import gi
gi.require_version('Gst', '1.0')
gi.require_version('Gtk', '3.0')
gi.require_version('GdkX11', '3.0')
gi.require_version('GstVideo', '1.0')
from gi.repository import Gst, Gtk, Gdk, GLib, GdkX11, GstVideo, Gio
class ChangeRecordWindow(Gtk.Window):
    def __init__(self, main_win):
        self.old_value1=""
        self.old_value2=""
        self.old_value3=""
        self.main_win=main_win
        self.textarea1= ""
        self.textarea2= ""
        self.textarea3= ""
        Gtk.Window.__init__(self, title="Change record")
        self.set_size_request(200, 100)
        self.main_box = Gtk.VBox.new(False, 0)
        self.controls = Gtk.HBox.new(False, 0)
        self.label1=Gtk.Label("Enter the name of radio station")
        self.label2=Gtk.Label("Enter the URL-address")
        self.label3=Gtk.Label("Enter the site of radio station")
        self.label4=Gtk.Label("Fill all fields!")
        self.button1=Gtk.Button()
        self.button1.set_label("Apply")
        self.button2=Gtk.Button()
        self.button2.set_label("Cancel")
        self.controls.pack_start(self.button1, False, False, 2)
        self.controls.pack_start(self.button2, False, False, 2)
        self.entry1 = Gtk.Entry()
        self.entry1.set_text(self.main_win.deleteable_Name)
        self.entry1.set_size_request(20,5)
        self.entry2 = Gtk.Entry()
        self.entry2.set_text(self.main_win.deleteable_URL)
        self.entry2.set_size_request(20, 5)
        self.entry3 = Gtk.Entry()
        self.entry3.set_text(self.main_win.deleteable_All)
        self.entry3.set_size_request(20, 5)
        self.old_value1=self.entry1.get_text()
        self.old_value2=self.entry2.get_text()
        self.old_value3=self.entry3.get_text()
        self.entry1.connect("changed", self.set_disabled)
        self.entry2.connect("changed", self.set_disabled)
        self.entry3.connect("changed", self.set_disabled)
        self.button1.connect("clicked", self.change_record)
        self.button2.connect("clicked", self.cancel)
        self.main_box.pack_start(self.label1, False, False, 0)
        self.main_box.pack_start(self.entry1, False, False, 5)
        self.main_box.pack_start(self.label2, False, False, 0)
        self.main_box.pack_start(self.entry2, False, False, 0)
        self.main_box.pack_start(self.label3, False, False, 0)
        self.main_box.pack_start(self.entry3, False, False, 0)
        self.main_box.pack_start(self.controls, False, False, 2)
        self.main_box.pack_start(self.label4, False, False, 2)
        self.add(self.main_box)
        print(self.entry1.get_text(),"!!!")
        print(self.entry1.get_text())
        print(self.main_win.deleteable_All ,"old list")
    def cancel(self, widget):
        self.destroy()
    def change_record(self, widget):
        self.textarea1=self.entry1.get_text()
        self.textarea2=self.entry2.get_text()
        self.textarea3=self.entry3.get_text()
self.indexData=self.main_win.data_list.index((self.old_value1,self.old_value2))
```



```

self.indexInter=self.main_win.interface_list.index((self.old_value1,self.old_value3))
self.indexRadio_dic=self.main_win.radio_dic.index((self.old_value1,self.old_value2))
self.main_win.data_list[self.indexData]=(self.textarea1,self.textarea2)
self.main_win.interface_list[self.indexInter]=(self.textarea1,self.textarea3)
self.main_win.radio_dic[self.indexRadio_dic]=(self.textarea1,self.textarea2)
self.main_win.list_radio.clear()
for software_ref in self.main_win.interface_list:
    self.main_win.list_radio.append(list(software_ref))
for i in range(len(self.main_win.data_list)):
    self.main_win.radio_dic.append(self.main_win.data_list[i])
def set_disabled(self,widget):
    if self.entry1.get_text()!=" and self.entry2.get_text()!=" and self.entry3.get_text()!=":
        self.button1.set_sensitive(True)
    else:
        self.button1.set_sensitive(False)
class AddRecordWindow(Gtk.Window):
    def __init__(self,main_win):
        self.main_win=main_win
        self.textarea1= ""
        self.textarea2= ""
        self.textarea3= ""
        Gtk.Window.__init__(self, title="Add record")
        self.set_size_request(200, 100)
        self.timeout_id = None
        self.main_box = Gtk.VBox.new(False, 0)
        self.controls = Gtk.HBox.new(False, 0)
        self.label1=Gtk.Label("Enter the name of radio station")
        self.label2=Gtk.Label("Enter the URL-address")
        self.label3=Gtk.Label("Enter the site of radio station")
        self.label4=Gtk.Label("Fill all fields!")
        self.button1=Gtk.Button()
        self.button1.set_label("Apply")
        self.button1.set_sensitive(False)
        self.button2=Gtk.Button()
        self.button2.set_label("Cancel")
        self.controls.pack_start(self.button1, False, False, 2)
        self.controls.pack_start(self.button2, False, False, 2)
        self.entry1 = Gtk.Entry()
        self.entry1.set_text("")
        self.entry1.set_size_request(20,5)
        self.entry2 = Gtk.Entry()
        self.entry2.set_text("")
        self.entry2.set_size_request(20, 5)
        self.entry3 = Gtk.Entry()
        self.entry3.set_text("")
        self.entry3.set_size_request(20, 5)
        self.entry1.connect("changed", self.set_disabled)
        self.entry2.connect("changed", self.set_disabled)
        self.entry3.connect("changed", self.set_disabled)
        self.button1.connect("clicked", self.add_record)
        self.button2.connect("clicked", self.cancel)
        self.main_box.pack_start(self.label1, False, False, 0)
        self.main_box.pack_start(self.entry1, False, False, 5)
        self.main_box.pack_start(self.label2, False, False, 0)
        self.main_box.pack_start(self.entry2, False, False, 0)
        self.main_box.pack_start(self.label3, False, False, 0)
        self.main_box.pack_start(self.entry3, False, False, 0)
        self.main_box.pack_start(self.controls, False, False, 2)
        self.main_box.pack_start(self.label4, False, False, 2)
        self.add(self.main_box)
        print(self.entry1.get_text(),"!!!")
        print(self.entry1.get_text())
    def on_error_clicked(self):
        dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
                                   Gtk.ButtonsType.OK, "ERROR")
        dialog.format_secondary_text(
            "This name of radio is already exist.")
        dialog.run()
        print("ERROR dialog closed")
        dialog.destroy()
    def cancel(self,widget):
        self.destroy()

```

```

def add_record(self, widget):
    self.textarea1=self.entry1.get_text()
    self.textarea2=self.entry2.get_text()
    self.textarea3=self.entry3.get_text()
    self.main_win.data_list.append((self.textarea1,self.textarea2))
    self.main_win.interface_list.append((self.textarea1,self.textarea3))
    self.main_win.radio_dic.append((self.textarea1,self.textarea2))
    print(self.main_win.radio_dic, "radidic")
    self.main_win.list_radio.clear()
    for software_ref in self.main_win.interface_list:
        self.main_win.list_radio.append(list(software_ref))
def set_disabled(self,widget):
    if self.entry1.get_text()!=" and self.entry2.get_text()!=" and self.entry3.get_text()!=":
        self.button1.set_sensitive(True)
    else:
        self.button1.set_sensitive(False)
class FileChooserWindowOpen(Gtk.Window):
    def on_file_clicked(self):
        dialog = Gtk.FileChooserDialog("Please choose a file", self,
            Gtk.FileChooserAction.OPEN,
            (Gtk.STOCK_CANCEL, Gtk.ResponseType.CANCEL,
             Gtk.STOCK_OPEN, Gtk.ResponseType.OK))
        self.add_filters(dialog)
        response = dialog.run()
        if response == Gtk.ResponseType.OK:
            print("Open clicked")
            print("File selected: " + dialog.get_filename())
        elif response == Gtk.ResponseType.CANCEL:
            print("Cancel clicked")
        self.filename=dialog.get_filename()
        dialog.destroy()
    def add_filters(self, dialog):
        filter_text = Gtk.FileFilter()
        filter_text.set_name("Text files")
        filter_text.add_mime_type("text/plain")
        dialog.add_filter(filter_text)
        filter_py = Gtk.FileFilter()
        filter_py.set_name("Python files")
        filter_py.add_mime_type("text/x-python")
        dialog.add_filter(filter_py)
        filter_any = Gtk.FileFilter()
        filter_any.set_name("Any files")
        filter_any.add_pattern("*")
        dialog.add_filter(filter_any)
    def return_filename(self):
        return self.filename
class FileChooserWindowSave(Gtk.Window):
    filename=""
    def add_filters(self, dialog):
        # Add text file filter
        filter_text = Gtk.FileFilter()
        filter_text.set_name("Text files")
        filter_text.add_mime_type("text/plain")
        dialog.add_filter(filter_text)
        filter_py = Gtk.FileFilter()
        filter_py.set_name("Python files")
        filter_py.add_mime_type("text/x-python")
        dialog.add_filter(filter_py)
        filter_any = Gtk.FileFilter()
        filter_any.set_name("Any files")
        filter_any.add_pattern("*")
        dialog.add_filter(filter_any)
    def button_pressed(self,list,dict):
        dialog = Gtk.FileChooserDialog("Save your text file", self,
            Gtk.FileChooserAction.SAVE,
            (Gtk.STOCK_CANCEL, Gtk.ResponseType.CANCEL,
             Gtk.STOCK_SAVE, Gtk.ResponseType.ACCEPT))
        dialog.set_default_size(800, 400)
        self.add_filters(dialog)
        Gtk.FileChooser.set_do_overwrite_confirmation(dialog, True)
        Gtk.FileChooser.set_current_name(dialog, "Untitled document")
        response = dialog.run()

```

```

if response == Gtk.ResponseType.ACCEPT:
    filename = Gtk.FileChooser.get_filename(dialog)
    print(
        "This is the filename: " + filename)
    self.save_to_file(filename,list, dict)

    dialog.destroy()
def save_to_file(self,filename,list,dict):
    save_list=[]
    str=""
    file = open(filename, "w")
    for i in range(len(list)):
        save_list.append(list[i][0]+","+dict[i][1]+","+list[i][1])
    for i in range(len(save_list)):
        str=str+save_list[i]+"\\n"
    file.write(str)
    file.close()
    print(save_list)
    pass
def return_filename(self):
    return self.filename
def __del__(self):
    print("window deleted")
class RadioConnect():
    def __init__(self, URL, StartVolume):
        Gst.init(sys.argv)
        self.state = Gst.State.NULL
        self.duration = Gst.CLOCK_TIME_NONE
        self.playbin = Gst.ElementFactory.make("playbin", "playbin")
        if not self.playbin:
            print("ERROR: Could not create playbin.")
            sys.exit(1)
        # set up URI
        self.playbin.set_property(
            "uri", URL)
        self.playbin.set_property("volume", StartVolume)
    def play(self):
        self.playbin.set_state(Gst.State.PLAYING)
    def pause(self):
        self.playbin.set_state(Gst.State.PAUSED)
    def stop(self):
        self.playbin.set_state(Gst.State.READY)
    def on_slider_changed(self, value):
        self.playbin.set_property("volume", value)
    def __del__(self):
        print("deleted")
class RadioWindow(Gtk.Window):
    data_list = []
    interface_list = []
    def __init__(self):
        main_window = Gtk.Window.new(Gtk.WindowType.TOPLEVEL)
        main_window.connect("delete-event", self.on_delete_event)
        main_window.set_title("Hero-Radio")
        main_window.set_name('pam')
        self.obj = ""
        self.radio_dic=[]
        self.button1 = Gtk.Button.new_from_stock(Gtk.STOCK_MEDIA_PLAY)
        self.button1.connect("clicked", self.start)
        self.button1.set_name("lal")
        self.button2 = Gtk.Button.new_from_stock(Gtk.STOCK_MEDIA_PAUSE)
        self.button2.set_name("lal")
        self.button2.connect("clicked", self.on_pause)
        self.button3 = Gtk.Button.new_from_stock(Gtk.STOCK_MEDIA_STOP)
        self.button3.connect("clicked", self.on_stop)
        self.button3.set_name("lal")
        self.slider = Gtk.HScale.new_with_range(0, 3, 0.01)
        self.slider.set_draw_value(False)
        self.slider.set_name("lals")
        self.slider_update_signal_id = self.slider.connect(
            "value-changed", self.on_slider_changed)
        self.list_radio = Gtk.ListStore(str, str)

```

```

for software_ref in self.interface_list:
    self.list_radio.append(list(software_ref))
self.software_TreeStore = Gtk.TreeView(self.list_radio)
for i, column_title in enumerate(["Название", "Сайт"]):
    renderer = Gtk.CellRendererText()
    column = Gtk.TreeViewColumn(column_title, renderer, text=i)
    self.software_TreeStore.append_column(column)
self.scrollable_treelist = Gtk.ScrolledWindow()
self.scrollable_treelist.set_vexpand(True)
self.scrollable_treelist.add(self.software_TreeStore)
controls = Gtk.HBox.new(False, 0)
controls.pack_start(self.button1, False, False, 2)
controls.pack_start(self.button2, False, False, 2)
controls.pack_start(self.button3, False, False, 2)
controls.pack_start(self.slider, True, True, 0)
self.main_hbox = Gtk.HBox.new(False, 0)
self.main_hbox.pack_start(self.scrollable_treelist, True, True, 2)
grid=Gtk.Grid()
mb=Gtk.MenuBar()
mb.set_name('pam')
file=Gtk.MenuItem("File")
file_open=Gtk.MenuItem("Open file")
file_save=Gtk.MenuItem("Save file")
file_close=Gtk.MenuItem("Close file")
exit = Gtk.MenuItem("Exit")
edit=Gtk.MenuItem("Edit")
add_record=Gtk.MenuItem("Add record")
add_record.set_name('pam')
delete_record=Gtk.MenuItem("Delete record")
delete_record.set_name('pam')
change_record=Gtk.MenuItem("Change record")
change_record.set_name('pam')
about = Gtk.MenuItem("About")
file_open.connect("activate", self.open_file)
file_open.set_name('pam')
file_save.connect("activate", self.window_save)
file_save.set_name('pam')
file_close.connect("activate",self.clear_ui)
file_close.set_name('pam')
exit.connect("activate", self.exit)
exit.set_name('pam')
about.connect("activate",self.call_about)
add_record.connect("activate",self.create_add_record_window)
delete_record.connect("activate",self.delete_record)
change_record.connect("activate",self.create_change_record_window)
filemenu=Gtk.Menu()
filemenu.set_name('pam')
filemenu.append(file_open)
filemenu.append(file_save)
filemenu.append(file_close)
filemenu.append(exit)
file.set_submenu(filemenu)
editmenu=Gtk.Menu()
editmenu.set_name('pam')
editmenu.append(add_record)
editmenu.append(delete_record)
editmenu.append(change_record)
edit.set_submenu(editmenu)
mb.append(file)
mb.append(edit)
mb.append(about)
grid.attach(mb, 0, 0, 1, 1)
self.main_box = Gtk.VBox.new(False, 0)
self.main_box.pack_start(grid,False,False,0)
self.main_box.pack_start(self.main_hbox, True, True, 0)
self.main_box.pack_start(controls, False, False, 0)
print(type(main_window))
main_window.add(self.main_box)
main_window.set_default_size(640, 480)
main_window.show_all()
self.selected_row = self.software_TreeStore.get_selection()
self.selected_row.connect("changed", self.item_selected)

```

```

def delete_record(self):
    print(self.list_radio)
    print(self.deleteble_URL,"!!!")
    print(self.deleteble_Name,"!!!")
    print(self.deleteble_All,"!!!")
    # print(self.data_list)
    print(self.interface_list)
    self.data_list.remove((self.deleteble_Name,self.deleteble_URL))
    self.interface_list.remove((self.deleteble_Name,self.deleteble_All))
    # print(self.data_list)
    print(self.interface_list)
    self.list_radio.clear()
    for software_ref in self.interface_list:
        self.list_radio.append(list(software_ref))
    for i in range(len(self.data_list)):
        self.radio_dic.append(self.data_list[i])
    self.URL=""
    self.deleteble_All = ""
    self.deleteble_Name = ""
    self.deleteble_URL = ""
def create_add_record_window(self,widget):
    self.record = []
    winF=AddRecordWindow(self)
    winF.set_name('pam')
    winF.show_all()

def create_change_record_window(self,widget):
    winF=ChangeRecordWindow(self)
    winF.show_all()
    winF.set_name('pam')
    pass
def window_open(self):
    self.filename=""
    winF=FileChooserWindowOpen()
    winF.on_file_clicked()
    self.filename=winF.return_filename()
    print(self.filename)
    winF.__del__()
    return self.filename
def window_save(self,widget):
    winF=FileChooserWindowSave()
    winF.button_pressed(self.interface_list,self.radio_dic)
    winF.__del__()
def open_file(self, widget):
    self.filename=self.window_open()
    if self.filename!=None:
        self.get_data_from_file(self.filename)
def clear_ui(self,widget):
    self.list_radio.clear()
    self.interface_list=[]
    self.data_list=[]
    self.radio_dic=[]
    self.URL=""
    self.deleteble_Name=""
    self.deleteble_URL=""
    self.on_stop(self.button1)
def exit(self,widget):
    Gtk.main_quit()
def get_url_from_dic(self, model):
    for i in self.radio_dic:
        if i[0]==model:
            self.result=i[1]
            #print(i)
    return self.result
def item_selected(self, selection):
    model, row = selection.get_selected()
    print(row,"it'srow")
    if row is not None:
        print(model[row][0],"ride")
        print(model[row][1],"ride")
        print(model[row][0],"ride!!!!")
        print(self.get_url_from_dic(model[row][0]), "ride URL")

```

```

        self.URL = self.get_url_from_dic(model[row][0])
        self.deleteble_URL=self.get_url_from_dic(model[row][0])
        self.deleteble_Name=model[row][0]
        self.deleteble_All=model[row][1]
def on_slider_changed(self, widget):
    value = self.slider.get_value()
    if self.obj != "":
        self.obj.on_slider_changed(value)
def on_play(self):
    self.obj.play()

def on_pause(self, button):
    if self.obj != "":
        print(button)
        self.obj.pause()
def on_stop(self, button):
    if self.obj != "":
        self.obj.stop()
        self.obj = ""
        print(self.obj, "))))))")
def start(self, widget):
    print(widget, "@4344")
    print(self.data_list)
    if self.obj == "":
        self.obj = RadioConnect(self.URL, 0.5)
        self.BufferUrl = self.URL
        self.obj.play()
    elif self.obj != "" and self.BufferUrl != self.URL:
        self.on_stop(self.button3)
        self.obj = RadioConnect(self.URL, 0.5)
        self.BufferUrl = self.URL
        self.obj.play()
        print("kek3")
    elif self.obj != "" and self.BufferUrl == self.URL:
        self.obj.play()
        print("kek4")
def on_delete_event(self, widget, event):
    self.on_stop(None)
    Gtk.main_quit()
def get_data_from_file(self, filename):
    file = open(filename)
    str = file.read()
    print(str)
    str = str.split("\n")
    self.list_radio.clear()
    self.radio_dic=[]
    self.data_list=[]
    self.interface_list=[]
    for i in range(len(str) - 1):
        str2 = str[i].split(",")
        print(str2)
        if str2!="":
            self.interface_list.append((str2[0], str2[2]))
            self.data_list.append((str2[0], str2[1]))
    print(self.data_list, "111111")
    # self.URL = self.data_list[5][1]
    for software_ref in self.interface_list:
        self.list_radio.append(list(software_ref))
    for i in range(len(self.data_list)):
        self.radio_dic.append(self.data_list[i])
def call_about(self,widget):
    self.button_callback()
def about(self):
    Gtk.MessageDialog.__init__(self, "HERO-RADIO v0.1")
    self.format_secondary_text(
        "Supe Duper Hyper Radio EVER!!!!1111")
    self.run()
    self.destroy()
def button_callback(self):
    self.dialog = Gtk.MessageDialog(None,0, Gtk.MessageType.ERROR,Gtk.ButtonsType.OK, "HERO-RADIO v0.1")
    self.dialog.set_name('pam')
    self.dialog.format_secondary_text(

```

```

        "Supe Duper Hyper Radio EVER!!!!1111")
    self.dialog.run()
    print("ERROR dialog closed")
    # dialog.set_title("MessageDialog Example")
    self.dialog.destroy()
if __name__ == '__main__':
    win = RadioWindow()
    style_provider = Gtk.CssProvider()
    style_provider.load_from_path("test.css")
    Gtk.StyleContext.add_provider_for_screen(
        Gdk.Screen.get_default(), style_provider, Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION )
    Gtk.main()

```

Styles.css

```

#pam {
    background: #202328;
    color: #ffffff;
}
#pam:focus {
    color: #d5d8e0;
    background-color: #394557;
    background: white;
}
#ram {
    background: #272b33;
    color: #d5d8e0;
}
#lac {
    background-color: #272b33;
    color: #d5d8e0;
}
treeview.view {
    color: #d5d8e0; background: #272b33; font-weight: bold; text-shadow: none; box-shadow: none; }
treeview.view header button{
    color: #d5d8e0;
    background: #2b313a;
    font-weight: bold;
    text-shadow: none;
    box-shadow: none;
}
treeview.view:focus {
    color: #d5d8e0;
    background: #394557;
}
#lal {
    margin-top: 10px;
    margin-bottom: 5px;
    background: #272b33;
    color: #d5d8e0;
}
#lals {
    margin-top: 10px;
    margin-bottom: 5px;
    background: #202328;
    color: #d5d8e0;
}

```