EisenRadio

René Horn

Feb 25, 2024

CONTENTS

Designed to organize your web radios.

- Radio Database with tools to delete and update, backup and restore
- Style your app with pictures, write a comment or import a poem, song or study text to have a good time
- A full-blown SVG animation to multiply your fun and energy usage
- Create a shuffled playlist within a local audio files folder in seconds.
- *Android apk package: download to mobile, rename *WHL to *ZIP, extract with Android file manager <https://pypi.org/project/eisenradio-apk/>*

The backend (Python) serves:

- Front page as flask 'home' route
- Tools and About pages as 'utils' route
- Command line, grab radio streams and listen to one of them on the local HTTP proxy

CHAPTER

ONE

CONTENTS

1.1 Eisenradio - a Web radio expandable collection

1.1.1 Info

- Comic style animated internet radio
- Organize your web radios; delete and update, backup and restore
- Style your app with pictures, write a comment or import a poem, song or study text to have a good time
- Create a shuffled playlist within a local audio files folder in seconds
- grab content until ISP bandwidth limit hits
- includes professional aacPlus (on the fly) file repair

1.1.2 Links

newest version on GitHub dev branch, then PyPi Package, Android or Snap, Docker

- Android - download to mobile (link below .-apk), rename *WHL to *ZIP*, extract with Android file manager
- Android - https://pypi.org/project/eisenradio-apk/
- Snap - https://snapcraft.io/eisenradio
- GitHub - https://github.com/44xtc44/eisenradio

1.1.3 Eisenradio - the boring details

- REST API app on blueprints and ApplicationFactory of the Flask microframework with a SQLite database
- First Internet Radio App that can run a Spectrum Analyser in a Web browser (Feb,2022)
- A local Python Flask Web Server connects to the radio server in behalf of you. Your browser connects to Flask * Backend (server) opens the connection, buffers the radio stream and presents it to localhost IP: 127.0.0.1 * Frontend (browser) controls the backend, plays internet and local audio playlists * Browser audio element connects *http://localhost:5050/sound/classic* that streams *http://37.251.146.169:8000/streamHD* * Closing the browser does not disconnect the server listen (buffer discarded) nor streaming connections
- Plays and repairs aac plus files; play (1.3), repairs since version (1.4);
- Backup and restore are easy work with the help of an optional ex/imported human-readable *ini* file

- Blacklist feature for recorded files (titles); delete only once * lists can be ex/imported via a json dictionary file to other devices
- playing local audio uses the web server multiple file upload feature
- Multithreading allows you an unlimited number of radio connections at the same time, until the ISP Bandwidth limit hits
- Android APK Package uses Python Kivy for multi-touch and promotes the app to "foreground service" (to not get killed)

sketch:

```
         |B |                |S | Flask web server, Header[Werkzeug/2.0.2 Python/3.10.1]
         |r |listen          |e |-------> starRadio
         |o |------->   <--  |r |
         |w |GhettoRecorder  |v |-------> planetRadio
         |s |--->    <-----  |e |
         |e |                |r |-------> satteliteRadio
         |r |                |  |
         net: localhost     net: internet
         CORS: accept       CORS: deny
         audioNode: 1,-1    audioNode: 0, 0
         JavaScript,CSS     Python,SQL

    Cross-Origin Resource Sharing mechanism (CORS)
    i.a. prevents a Browser from analysing audio from internet


     net: localhost     net: internet
     CORS: accept       CORS: deny
     audioNode: 1,-1    audioNode: 0, 0
     JavaScript,CSS     Python,SQL

Cross-Origin Resource Sharing mechanism (CORS)
i.a. prevents a Browser from analysing audio from internet
```

1.1.4 command line

Start browser from command line

$ eisenradio

Call the command line app with one of two commands.

$ eisenradio-cmd

EisenRadio sits on top of 'GhettoRecorder' package https://pypi.org/project/GhettoRecorder/

$ ghettorecorder

You can export your *settings.ini* and *blacklist.json* via "Tools/Export/Names and URLs" menu from your database. An updated *blacklist.json* can be imported into your database.

- The default save path is in the package folder. You should change it via the menu options

  /home/osboxes/.local/lib/python3.6/site-packages/ghettorecorder/radios

1.1.5 pip install

$ pip3 install eisenradio # Tux > pip install eisenradio # M$

1.1.6 Pytest

Shows how to init a flask instance and perform some tests on it. More hints in the test comments.

> ~ … /eisenradio $ pytest -s # -s print to console

Is now part of the testautomation with flake8 and tox on GitHub.

1.1.7 Uninstall

Python user:

- find the module location
- uninstall and then remove remnants

>$ pip3 show eisenradio

>$ pip3 uninstall eisenradio

Location: … /python310/site-packages

1.2 Technical Overview

Project is the successor of GhettoRecorder and shall provide a database and HTML frontEnd.

1.2.1 How is this wired?

Flask server is divided by two routes, home and utils. To separate start page from "Tools" and "About".

1.2.2 Home

indexStructure.txt:
Project overview for raw div structure of start page to jump fast in and around.

canvas.js:
animation of the canvas itself to detach the canvas window and make a small show on reconnect

index.js:
init stuff and rec and listen traffic of browser to server, dark mode, toggle …

playlist.js:
Functions to run the local playlist feature with for-/backward controls

radioStyles.js:
full html css decoration for full animation and low for base, some decoration for recording

svgAnimation.js:
inkscape inline svg images animated in comic style (zeppelin, checkered balloon, floe by night)

1.2.3 Utils

bp_utils.js:
auto button press to hide buttons, write fake lists with parent, child divs to color log files, edit/delete blacklists, switch divs to show different tools options

The db access is accomplished with pure sql. There was an attempt to switch to ORM, but the app will be a reference for sql.

To show how to start a project with ORM, a side project Python package was created "Flask-SQLAlchemy-Project-Template 1.1".

All animations can be fully or partial disabled to run the app on small devices. Playlist option with for-/backward controls available.

Python modules for:
monitor recordings: write grabbed files into blacklists, lists can be edited

Aac file repair:
needed to fix incorrectly recorded aac files (no timer used to quit), so playlist not stuck

Export and Import:
radio settings with *ini* files blacklists with *json* files, dictionary style

delete all:
delete all radio table rows to restore from exported *ini* file

1.3 Feature Overview

- REST API app on blueprints and ApplicationFactory of the Flask microframework with a SQLite database
- First Internet Radio App that can run a Spectrum Analyser in a Web browser (Feb,2022)
- **A local Python Flask Web Server connects to the radio server in behalf of you. Your browser connects to Flask**
  - Backend (server) opens the connection, buffers the radio stream and presents it to localhost IP: 127.0.0.1
  - Frontend (browser) controls the backend, plays internet and local audio playlists
  - Browser audio element connects *http://localhost:5050/sound/classic* that streams *http://37.251.146.169:8000/streamHD*
  - Closing the browser does not disconnect the server listen (buffer discarded) nor streaming connections
- Plays and repairs aac plus files; play (1.3), repairs since version (1.4);
- Backup and restore are easy work with the help of an optional ex/imported human-readable *ini* file
- **Blacklist feature for recorded files (titles); delete only once**
  - lists can be ex/imported via a json dictionary file to other devices
- playing local audio uses the web server multiple file upload feature
- Multithreading allows you an unlimited number of radio connections at the same time, until the ISP Bandwidth limit hits
- Android APK Package uses Python Kivy for multi-touch and promotes the app to "foreground service" (to not get killed)

1.4 main page and JS load order

Div structure:

Find position and z-index for quicker restructuring and error tracking.

- Most animation SVG are <use> tags in a div container; original drawings are 100px high and width. No viewport set. All images are collected in a <symbol> tag at the end of the main page.
- **The images are drawn in two ways.**
  1. Complete <g> group within a <svg> with one <use> tag.
  2. Ordered (split) <g> group within a <svg> with multiple <use> tags.

Second option allows for automatic assignment of random colors for specific <g> group elements. Often used is hardcoded scale to fit images into the scene. Mostly by using CSS, but also some hardcoded in HTML, which is subject to change.

How <use> and tags can be used:

> Animation with pure <svg> <use id="gImage"/> </svg> and <svg> <use id="gImage"/> </svg> differs in some ways. <use> element encapsulated groups can be animated within the <svg></svg> borders only. A transform action like translate (moving) leads to a disappearing image into the border of the svg. See clouds and satellites. z-index, or HTML page order (one below the other) of multiple <use> elements is the layer order. Lower <use> elements cover upper <use> layers.
>
> elements can be moved within the whole html page. With all css style attributes and zindex applied. A encapsulated <use> element can "leave" the <svg> borders

Table 1: Front page

Hierarchy	Name	Description
N/A		loader animations, div set for bright and dark style
level_1	divStartPageFadeIn	fade in effekt for start page, broken since audio needs user interaktion
level_2	frontPic	{justify-content:center;align-items:center;} a teaser pic
level_2	progress	{position:relative;} progress bar base for timer
level_3	divProgressBar	{position: absolute;z-index: 1;} progress bar
level_2	bar_wrap_secondary_ac	wrap to position the center div with secondary menu items
level_3	wrap_secondary_action	{text-align: center} combo boxes, playlist, quick jump …
level_2	divPlayListContainer	playlist file names, a list made of divs to color 'em
h5	divCacheListFeedAnchorJump	top anchor to jump upward from console
level_2	divRadioContainer	{position:relative;} hull for radio styling
div		without name to group radio and comment at bottom
level_3	divRadioFrontPlate	{position:relative,z-index:2;max-width:1000px;justify-content: center;align-items: center;}
level_4	divMainAnimationContainer	wrapper for animated SVG
level_5	animatedBackGround	{position: absolute;z-index:0;overflow:hidden;} sky and ocean, svg 1000x1000 bottom is the rectangle tuxStageSky
level_5	divSvgBuoy	{position: absolute;z-index:10;} Edit button buoy with blinking top
level_5	divSvgScrewHeadTopRight	{position: absolute;z-index:2;} screw head rotated
level_5	divSvgGlasBreakTopRight	{position: absolute;z-index:1;} broken glas image

continues on next page

Table 1 – continued from previous page

Hierarchy	Name	Description
level_5	divSvgScrewHeadTopLeft	{position: absolute;z-index:2;}
level_5	divSvgScrewHeadBottomLeft	{position: absolute;z-index:2;}
level_5	divSvgScrewHeadBottomRight	{position: absolute;z-index:2;}
level_5	divSvgFlatSpeaker	{position: absolute;z-index:2;} speaker big for left
level_5	divSvgFlatSpeakerTopRight	{position: absolute;z-index:2;} speaker small for right
level_5	divCenterMovingSvgs	{position:relative;z-index:3;} inhabitants can move over screws and speaker
level_6	divAnimationContainer	{[position: absolute;] } spawn point for moving divs
level_7	divSvgZ1	{[position: absolute;z-index: 2;] } zeppelin
level_7	divSvgTux	{[position: absolute;z-index: 31;] } penguin
level_7	divSvgIceTux	{[position: absolute;z-index: 30;] } ice floe
level_7	divSvgPolarBear	{[position: absolute;z-index: 31;] } PolarBear
level_7	divSvglaGata	{[position: absolute;z-index: 31;] } la Gata del Diablo, Tiger
level_7	divSvgB1	{[position: absolute;] } balloon
level_7	divA1AirCraft	{[position: absolute;z-index:-1;]} aircraft
level_7	divDragRopeA1AirCraft	{[position: absolute;z-index:4;]} parachute drop, scripted in JS, children of parent divDragRopeA1AirCraft
level_7	divHorizon	calculate ocean horizon for satellites to disappear
level_4	divHeaderShadow	{position: absolute;} only a shadow for the radio headline
level_4	divMeasurementsUpper	{position: absolute;z-index:2;} redesigned, keep name, badge with speed, kb/s
level_4	divGracefulDegradation	{position: absolute;z-index:11;} CPU buttons to switch most animations off
level_4	divBtnContainer	{position:relative} listen, record buttons
level_5	divBtnAbsWrapper	{position:absolute;} wraps divPictureRow, divStationGenre, record buttons to easy relocate
level_6	divPictureRow	{position:relative;} custom pic
level_6	divButtons	{position:relative;z-index:10;} record, listen buttons
level_3	divCustomText	custom text outside the radio to get a “painting in museum with description effect”

Note: Level are hierarchy. 1 encapsulate 2, 2 encapsulate 3 and so forth.

order of loading JS:

<script type=”text/javascript” src=”…”></script>

- src=”{{ url_for(‘eisenhome_bp.static’, filename=’/js/index.js’) }}
- src=”{{ url_for(‘eisenhome_bp.static’, filename=’/js/animate.js’) }}
- src=”{{ url_for(‘eisenhome_bp.static’, filename=’/js/canvas.js’) }}”
- src=”{{ url_for(‘eisenhome_bp.static’, filename=’/js/radioStyles.js’) }}”
- src=”{{ url_for(‘eisenhome_bp.static’, filename=’/js/svgAnimation.js’) }}”

- src="{{ url_for('eisenhome_bp.static', filename='/js/playlist.js') }}"

1.5 technical sketch

sketch:

```
     |B |                |S | Flask web server, Header[Werkzeug/2.0.2 Python/3.10.1]
     |r |listen          |e |-------> starRadio
     |o |------->   <-- |r |
     |w |GhettoRecorder |v |-------> planetRadio
     |s |--->    <----- |e |
     |e |                |r |-------> satteliteRadio
     |r |                |  |
     net: localhost     net: internet
     CORS: accept       CORS: deny
     audioNode: 1,-1    audioNode: 0, 0
     JavaScript,CSS     Python,SQL

Cross-Origin Resource Sharing mechanism (CORS)
i.a. prevents a Browser from analysing audio from internet
```

1.6 pip install

pip install:

```
""" xxs Linux xxs """
$ pip3 install eisenradio
$ python3 -m eisenradio.wsgi  # watch flask

""" xxm Windows xxm """
> pip install eisenradio
> python -m eisenradio.wsgi

""" xxl big company xxl """
$$$ pip3 install eisenradio
$$$ python3 -m eisenradio.app  # serve flask
""" for the sake of completeness, a python
    production server 'waitress' is started """
```

1.7 known problems

As bigger as the JS part gets, more problems arise.

JS script loading order:

animate.js:
countUpDownInclusiveInt(min, max) feeds global 'animatedFunctionTimer' and could be replaced by svgAnimation.js 'class CountUpDown' instance to get consistent across the app. But then svgAnimation.js must be loaded before animate.js in index.html. This is not

possible, since svgAnimation.js has other dependencies. And breaks the scripts then. Only way out is to restructure and use node.js to get the load functionality. module.exports …, var tools = require('./tools');

1.8 eisenradio

1.8.1 eisenradio package

Subpackages

eisenradio.api package

Module contents

eisenradio.eisenhome package

Submodules

eisenradio.eisenhome.eishome module

eisenradio.eisenhome.routes module

eisenradio.eisenhome.watchdog module

Module contents

eisenradio.eisenutil package

Submodules

eisenradio.eisenutil.browser_stream module

eisenradio.eisenutil.config_html module

eisenradio.eisenutil.eisutil module

eisenradio.eisenutil.monitor_records module

eisenradio.eisenutil.request_info module

eisenradio.eisenutil.routes module

eisenradio.eisenutil.stopped_stations module

eisenradio.eisenutil.tools module

Module contents

eisenradio.lib package

Submodules

eisenradio.lib.eisdb module

eisenradio.lib.platform_helper module

Module contents

Submodules

eisenradio.cmd module

eisenradio.db module

eisenradio.gui module

eisenradio.wsgi module

Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search