

Joseph Haddad

Nuclear

PR: <https://github.com/nukeop/nuclear>

Issue: <https://github.com/nukeop/nuclear/issues/841#event-4635696119>

4/23/2021

General Project:

I am working on Nuclear, a music warehouse that gathers music from free sources and puts them at the tips of your fingers. You can stream from any free source online, it comes with YouTube and SoundCloud initially built in with an easy plugin system to add more. Nuclear is developed on GitHub so that anyone can contribute and can see the code and have the confidence in the platform's safety.

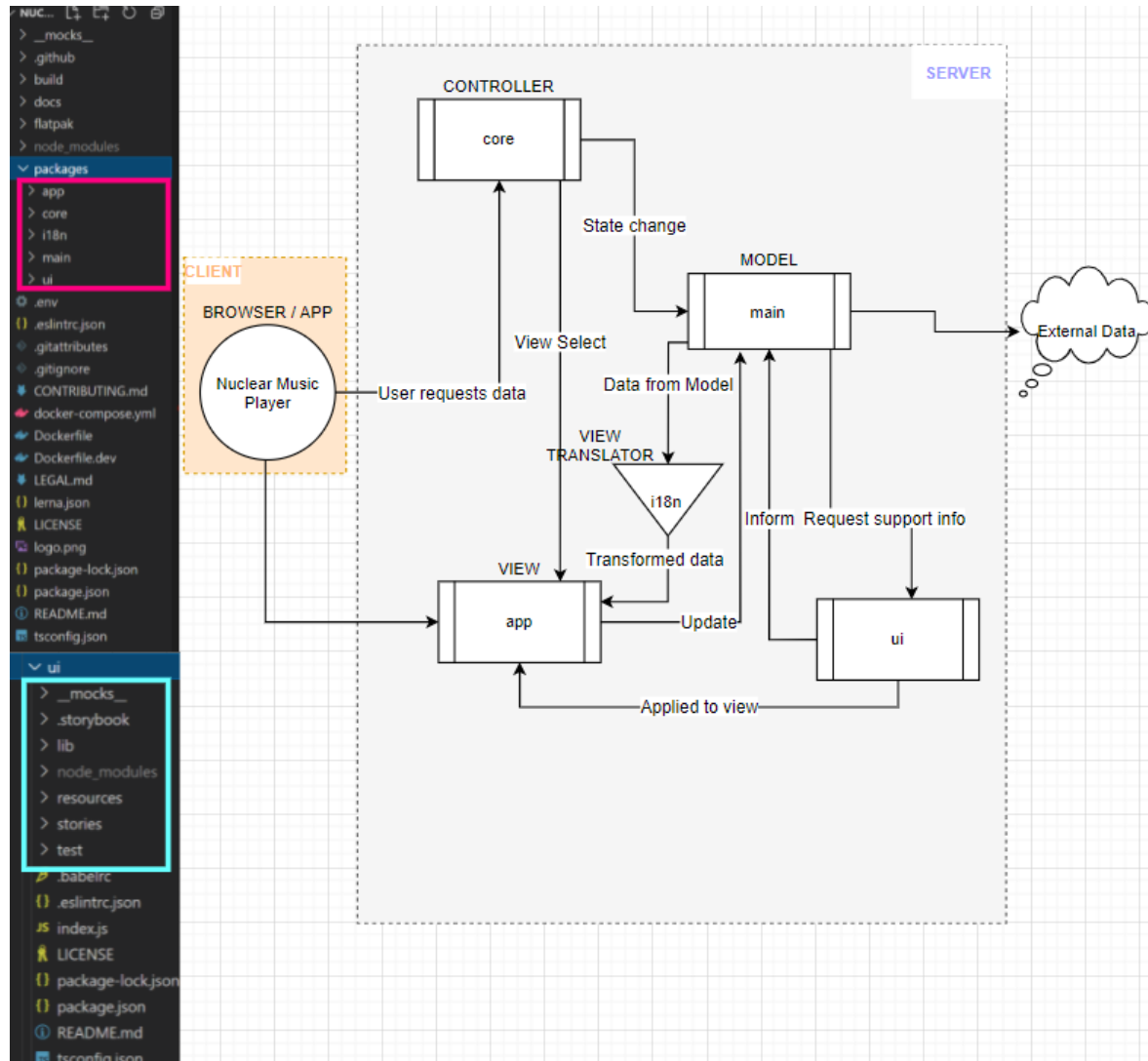
Project Architecture:

The project Nuclear runs on a very similar model to the View Controller Architecture model. I was able to make the following correlation and diagram.

- Controller
 - - **core** - a library containing functionality common to other packages
- Model
 - - **main** - package containing the electron server side code
 - - **ui** - package containing non-functional supportive ui components
- View
 - - **app** - package containing the main webapp displayed inside the electron window
 - - **i18n** - translations for presenting in different languages
- Browser / App
 - - **Nuclear Music Player**- app used by user

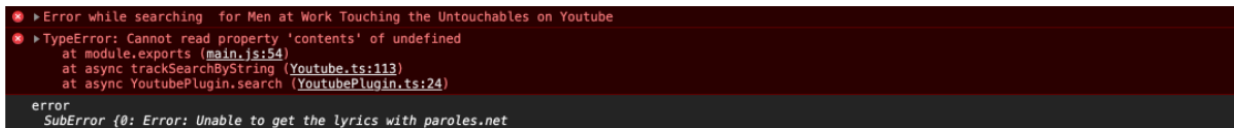
References:

The project files descriptions have been discussed with the creator of Nuclear and verified to be correct.



Issue:

I will be working on a bug fix on the Autoradio feature. Autoradio is a feature that will automatically select a song to play after your queued songs have completed. It has a scale of 1 to 10 that you can adjust which will affect how similar the next song will be to the current song. This feature can be found in `/app/app/containers/SoundContainer`. The bug is “On Autoradio mode, the playing queue should not stop when it fails to load for the next track.” a few notes and suggestions were also made stating.



```

Error while searching for Men at Work Touching the Untouchables on Youtube
TypeError: Cannot read property 'contents' of undefined
    at module.exports (main.js:54)
    at async trackSearchByString (Youtube.ts:113)
    at async YoutubePlugin.search (YoutubePlugin.ts:24)
error
SubError {0: Error: Unable to get the lyrics with paroles.net}

```

Sometime ytsr fails and then the playing queue stops. In UI, track's name still display with 0:00 duration.

Suggestion: when ytsr try re-do again or load another track.

In-Depth Architecture:

I do not expect any changes to the architecture to be made while performing this task. The feature has a model very similar to an object-oriented architecture. This is because it is always referring back to props which holds all values within it and passes this throughout the functions. While it can also be similar to Data Flow Architecture i decided it wasn't due to the multiple instances of saving and editing data locally as well as to props.

The following is an in-depth look at the architecture that makes up Autoradio. It is contained in 1 folder with 2 separate files. `index.js` will call the function `handleLoaded` which then calls `handleAutoRadio`. When autoradio is enabled and the song playing is the last song in que the function `addAutoradioTrackToQue` will be called in file `autoradio.js`. This file will then call `getSimilarTracksToQueue`. `getSimilarTracksToQueue` will then call `getSimilarTracks` 10 times within a loop and a list of similar tracks will be pushed to tracks. Then the tracks will be checked to verify at least 1 track is not in que via function `isTrackinQueue`. If true it will then score the tracks and return the best fit for given user parameters. If return is null, function `getNewTrack` will be called. If getter

equals track it will go through `getSimilarTracks` which will return a new track else `getTracksFromSimilarArtist` will be called and will the call `getSimilarArtists` passing `artistJson` which will return a `similarArtist`. This will then be passed along with `ArtistDeviation` to `getRandomElement` which will have `getArraySlice` nested within and return an array slice value which `getRandomElement` will take in and return a similar artist. Then `getArstistTopTrack` will be called and provided the previous variables and returns the `toptrack` by the artist. This will be verified not to be in que by `getTrackNotInQueue` and when true will add to the queue via `addToQueue`.

