Personal Information

Name: Ashutosh Aswal

IRC nick: yellowhatpro

Email: yellowhatpro3119@gmail.com

Github: yellowhatpro

Time Zone: UTC+05:30

RustBrainzDB: Rust bindings for the MusicBrainz database

Project Overview

MusicBrainz has a large database of different entities and relationships. There are various applications that can be built on top of this huge database. With Rust gaining popularity, we need a way for rustaceans to access the musicbrainz entities, so that they can create cool applications on top of musicbrainz database.
The project aims to initiate rust development in the brainz family by generating rust mappings for the musicbrainz schema, and create an application to make use of it.
The application will be a standalone rust program that polls musicbrainz database, queries URL relationships from `edit_note` and `edit_data` tables, and saves them using Wayback Machine API.

My contributions up till now:

- Fixes for sql-gen

Up till now, my main focus was to make the existing sql-gen project compatible with the musicbrainz schema. With the project working, we are able to build a binary, which can generate all the rust types and important queries to work with the database.
I have made the generated entities available as a github repository, currently mb-rs, so that they can be used with the rust application we

are coming up with.
At the moment, they have successfully compiled, but there might be runtime issues, so we can fix them as we proceed with the application development.

Implementation

The idea is to test the generated rust entities with a fully functional application. Therefore, the objective of the proposal will be developing a rust app, while continuously maintaining the rust entities. By the end of the project, we will have a working set of entities for musicbrainz schema, ready to be published, and an app running on top of it.

Here are 4 broad aspects of the implementation:

1. Making the sql-gen library working

The sql-gen library is a great tool to generate rust types for postgres databases schemas. It uses `sqlx` crate under the hood, which is a really nice tool for writing SQL queries. The problem with sql-gen is, it does not very well map with postgres types, which caused a lot of errors in generating the rust entities. To deal with this, new mappings were added:
`"_int8" => "i64", "_int4" => "i32", "_int2" => "i16", "_text" => "String", "time" => "chrono::NaiveTime", "jsonb" => "serde_json::Value", "bool" => "bool", "bpchar" => "String", "char" => "String", "character" => "String"`
sql-gen also panicked on custom musicbrainz types, like `EditNoteStatus`, `Fluency`, `CoverArtPresence`, etc. As a fix, respective rust enums were added.
All these fixes and more can be seen here.

2. Generating rust types and functions for the database using sql-gen and publishing the crate

Once sql-gen is able to generate all the musicbrainz types and necessary database functions, the task is to make the generated types accessible for everyone. While in development, we can go with the github version of the library, currently mb-rs, where the generated rust types are available. As we proceed to the end of the program, we will publish the crate as well.

However, what good are the mappings if we have no app using it?

Now, we can start working on our rust app. The rust app has 2 main functions:

1. Interacting with the database, and querying URLs, (and this is where our rust bindings will shine)
2. Save the queried URLs using Wayback Machine APIs

3. Interacting with the database

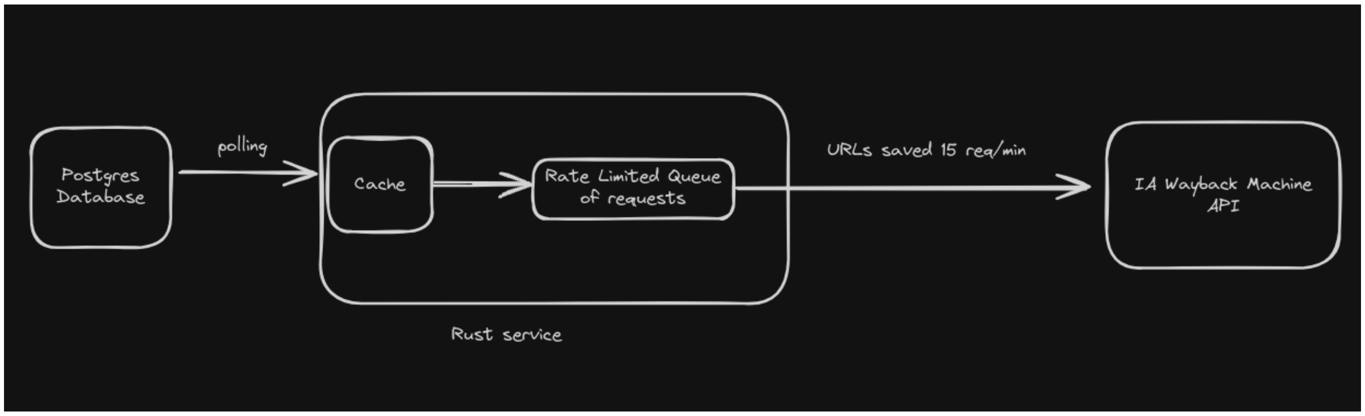
Interaction with the database is the first challenge. Having multiple ways to query the database, the preferred and easier solution, which I found on discussion in IRC is polling. Additionally, a cache layers will be sitting between the app and the database, ensuring we don't query same item twice in a day.

4. Saving the URLs using the Wayback Machine APIs

The other interface of the app interacts with the Wayback Machine, saving the queried URLs.
This is the API we will be interacting with:

```
curl -X POST -H "Accept: application/json" -H "Authorization: LOW myaccesskey:mysecret"
-d'url=http://brewster.kahle.org/' https://web.archive.org/save
```

Since there is rate limiting associated, we need to self rate limit our requests.
However, this particular section can be decoupled from the poll results, as we can extend its use case for more tasks, such as re-archiving links which can be changed later by artists, like spotify, itunes, which also need to be rate-limited.



This is just a high level overview of the core architecture. Expect more engineering tasks to be delivered :)

Timeline

Following is the timeline for the 350 hours of GSoC coding period I'll be devoted to:

Community Bonding Period :

- Discuss various engineering aspects with the mentors, including caching, polling, rate limiting.

Week 1

- Start with setting up of environment for development, which includes writing container scripts, setting up required dependencies needed to run the project.
- Setup a basic rust project to meet the requirements
- This will involve a db connection, and a `reqwest` method to save the URLs.

Week 2

- Discuss and implement database queries and constraints, e.g. skipping edits of editors marked spammers.
- The objective of this week will be to get desired results from database.

Week 3

- Working on the cache layer.
- This will include setting up and integrating redis with our running app.
- Devise cache strategies to prevent duplicate URLs to be saved.

Week 4

- Work on database polling mechanism.
- This will require a postgres function that uses a cursor to fetch data from db, which can be called from the rust app using `sqlx`.

Week 5

- Work on the self-rate limiting, so that we do not exceed rate limits imposed by Internet Archive.
- A rate limited queue implementation will be required to do this task.
- As per Internet Archive, the current limit is 15 saves per minute, therefore we need to create a queue implementation, that delays the next request for some time. This seems a workable idea as of now, but we can have more ideas on discussion ahead.

Week 6/7

- Work on the corner cases, which arise when editor edits or deletes their edits
- This might require notify mechanism of the database, but again, a little chat with mentors might be required.
- There are still pending edits stored in the tables which we can save using Wayback Machine.
  - This will be a detached project from current container, and run independently from the current app.

Week 8

- Work on analytics and testing of the app.
- This will include integrating analytics tools with the container, and piping the logs to necessary location.

Week 9/10

- This week will involve deploying the container to appropriate registry, and automating the process of deploying the container app.
- Fix any post-deployment bugs.

Week 11/12

- Finalizing work and documenting it.
- Publishing the rustbrainz-db crate.
- Start working on other ideas of IE integration.

Stretch goals

- Honestly, the main project can end sooner than expected, given we don't run into unexpected errors. But if we get lucky, my task in hand will be to work on 2 more IA integrations:
  1. Re-archiving links which can be changed later by artists, like spotify, itunes.
  2. Musicbrainz DB Dumps automation.

Detailed Information About Yourself

My name is Ashutosh Aswal. I am a senior pursuing Bachelors in CSE from Punjab Engineering College, Chandigarh, India. Having contributed to Android Projects before, now I want to explore new domains of tech and gain hands on experience.

Tell us about the computer(s) you have available for working on your SoC project!

I have my HP Pavilion ec0101ax running Arch Linux, with Ryzen 5 3550H and 24 gigs of RAM.

When did you first start programming?

I began my programming journey with C++ in 11th grade.

What type of music do you listen to? (Please list a series of MBIDs as examples.)

I mostly listen to Indie music artists like The Local Train, Nikhil D'Souza, Anuj Jain etc.

What aspects of the project you're applying for interest you the most?

I like exploring new tools and languages in my free time. In recent times, I started exploring rust, and wanted to build some stuff with it. Other than that, I like how this particular project is introducing different engineering concepts which I have only read about, and I'd get a change to implement them.

Have you ever used MusicBrainz to tag your files?

Yes, I have.

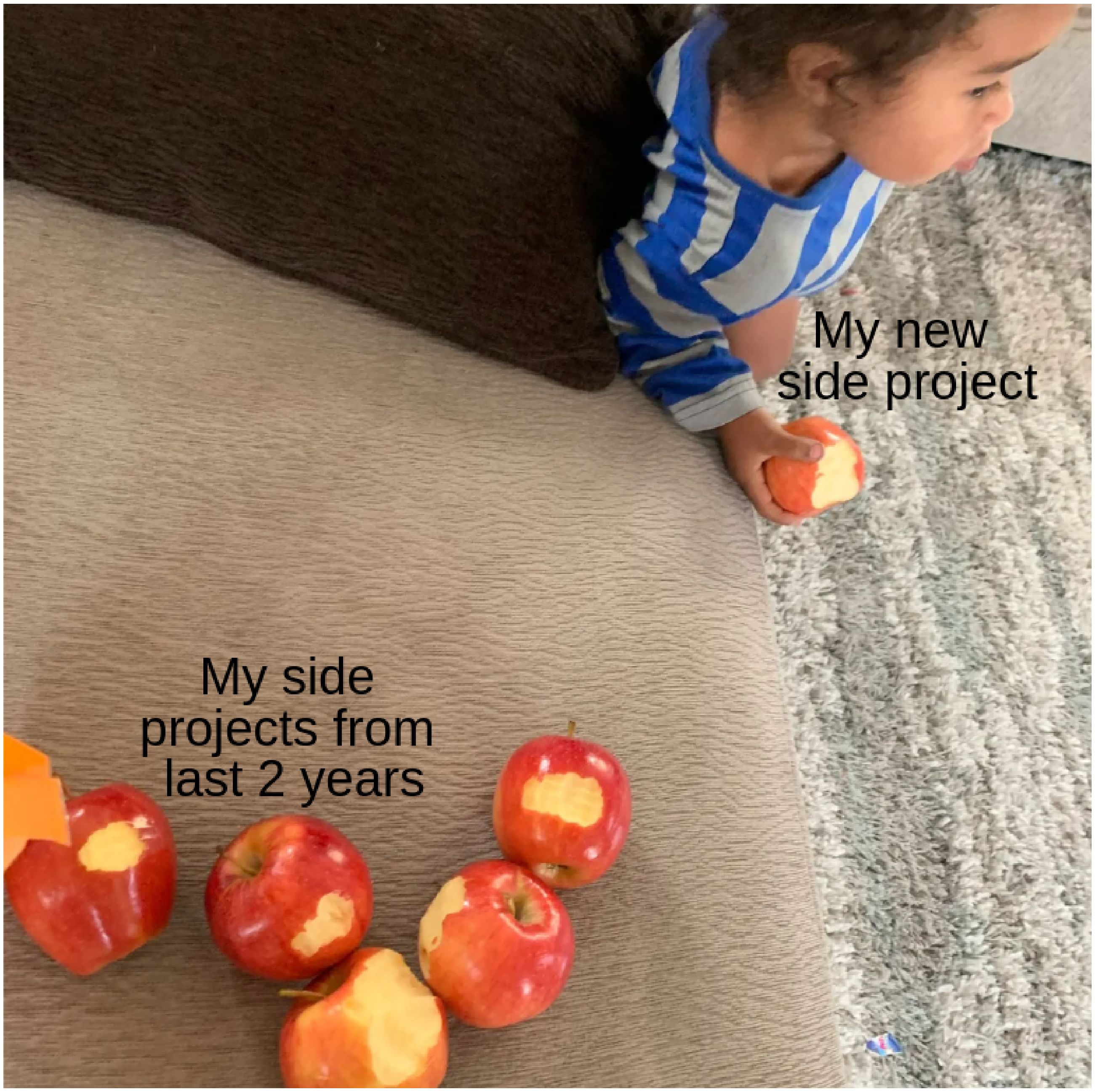
Have you contributed to other Open Source projects? If so, which projects and can we see some of your code?

I have earlier contributed to the Metabrainz android projects, musicbrainz-android, and listenbrainz-android.

What sorts of programming projects have you done on your own time?

I have prior experience in android development, and in my free time, I love to explore different other tech stacks and frameworks.

- i-remember: A cli tool in rust to save shell commands, batch them, and reuse them. (WIP)
- codes-practice : A react app to show my code solutions to various algorithmic problems, and competitive programming problems.
- Apart from them, I have many undeployed, half-baked, good-for-nothing projects I cook in free time.



How much time do you have available, and how would you plan to use it?

I will be working ~30 hours per week on the project.