Our Black Gems API

CS460 Project by Roderick Bishop & Taighlor Moultrie

Introduction & Purpose

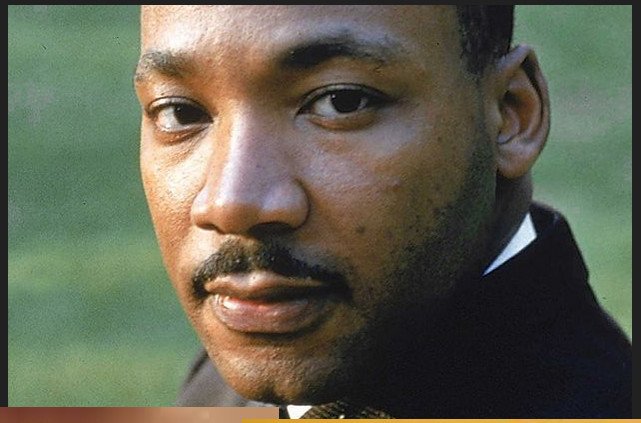
Introduction

- African Americans are woefully underserved when it comes to having our history presented in a way that is modern, truthful, and impactful.
- So much of our history has been buried, lied about, and thus forgotten.

It’s time to display that greatness in a way that works for the future. Our Black Gems API is both a celebration of Black or African American contributions to society.

Purpose

OBGAPI is an API that serves relevant data to shine a light on the greatness of Black culture and creations



Requirements & mVP

Data Requirements:

- The data to support this api is based on a FIGURE. FIGUREs consists of First & Last name, a Bio, Birth and Death dates, Birth City, a Category, a bio and a unique Id.
- An INVENTION is created by a FIGURE. Each INVENTION has database id number, Patent No./id No., name of the Invention, names of Inventors, a Category and Description
  The /fact endpoint will show a black history fact of the day. This data will be represented as a FACT entity. A FACT has a Date, a Fact, a figure Id and name, and a Link (url string to further reading)
- The /song endpoint will send a random song released by a black artist. This data is organized into SONG entities. A SONG has an Id, Artist(s), song title, Year, name of the Album, name of the recording studio, year, and Link.

Functional Requirements:

Role-Specific:

- Update and delete all SONGS, FIGURES,INVENTIONS, and FACTS entitles as Administrator

Client/End User:

- Retrieve a FIGURE (all entity data)
- Retrieve FIGUREs in specific categories
- Retrieve FIGUREs from a birth city
- Retrieve a random FACT
- Retrieve a random FACT for any day of the year
- Retrieve an INVENTION (all entity data)
- Retrieve an INVENTION BY patent/id number.
- Retrieve an INVENTION BY Category.
- Retrieve a SONG (all entity data)
- Retrieve SONGs from a Studio
- Retrieve SONGs from an Artist

The Minimally Viable Product (mVP)

Since this project is meant to be an ongoing and open source project, it was important to define an mVP for this course.

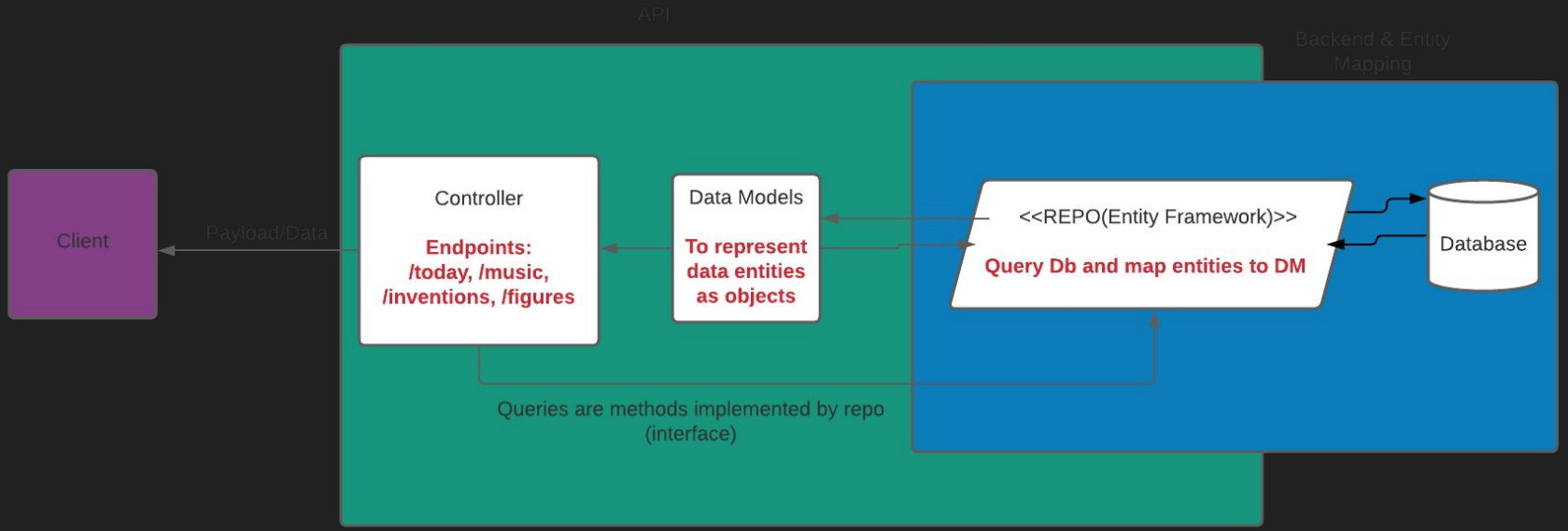
Our mVP for OBGAPI is defined as:

- Four distinct endpoints, with one that is demo-ready (facts)
- Demo can be done with hard-coded data, using endpoint testing tools (Swagger or Insomnia)
- Database schema for the backend defined and test tuples in database

A minimum viable product (MVP) is a development technique in which a new product or website is developed with sufficient features to satisfy early adopters. The final, complete set of features is only designed and developed after considering feedback from the product's initial users.

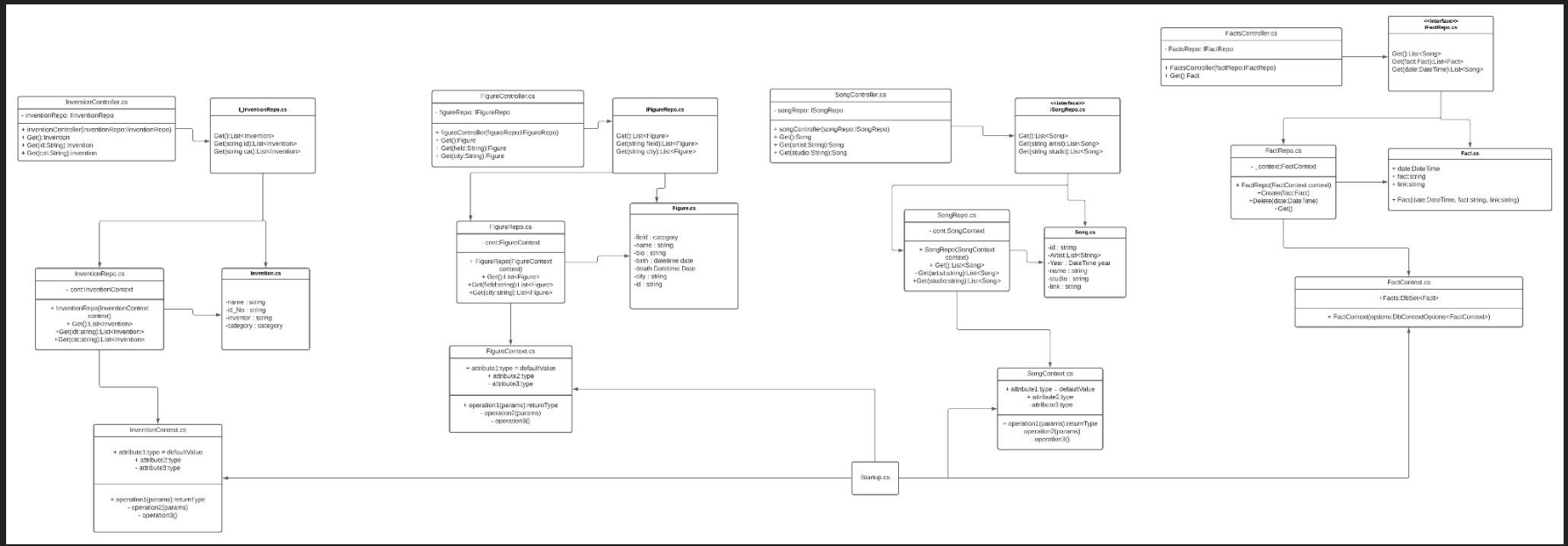
Architecture and Design

Project Architecture:



Controller, Data Models, and Entity is the heart of the API.

UMLS:



The client is served by the controller of the respective endpoint which connects in various ways to other files we've created for this project.

API Code Overview

Controllers

- Controllers for API’s handle incoming HTTP requests and send back responses.
- Instead of a large and monolithic controller, we decided to have 4 endpoints with each controller at its core.
  - Allows for modular code, easy maintenance, and the ability to use powerful OOP on the server side.
- Controllers follow the format {endpoint}controller and we have the following four:
  - FactsController, FigureController, SongController, Invention Controller

Data Models

- The data models were created as an OOP server-side “mirror” for the entities from the DB. The data represented in these models are called “Gems”. Each data field corresponds to an attribute in the relation set and has a matching domain.
- There are 4 data models defined in this early version of the API:
  - Song(int id, List<string> artists, year, name, studio, link)
  - Fact(DateTime date, string fact, string link)
  - Figure(int id, string name, int temp, DateTime birthDate, DateTime deathDate, string bio, string city)
  - Invention(int id, string patentId, string inventionName, string desc, Category category, List<Figure> inventors)

Repositories

The repositories are abstractions of the data access layer in your API. With this abstraction, we can precisely define how we want to access our data on the server side using queries.

The Service Layer

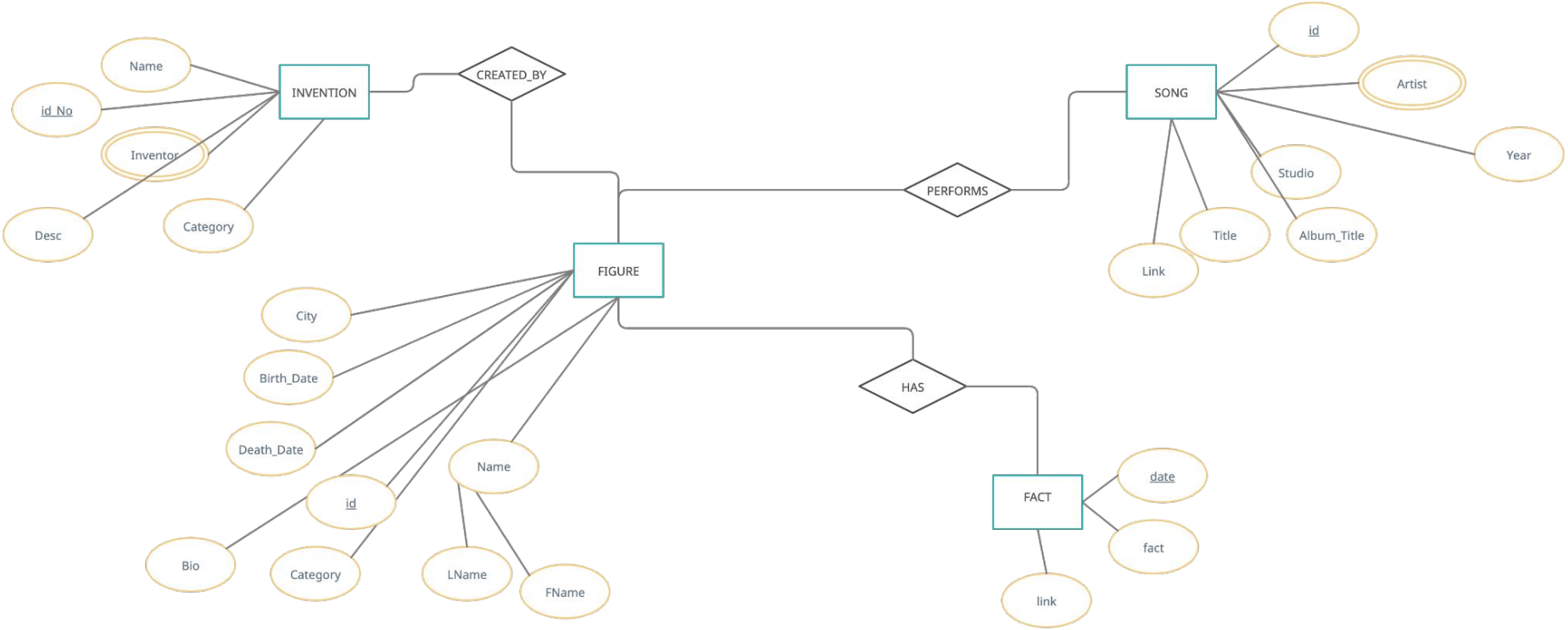
The service layer adds necessary and shared functionality for our controllers, repos, and models.

In our current version (v0.0.1), the two services are:

Selectsingle.cs- for selecting a random Song, Fact, Figure, or Invention from the list returned via query

SelectCategory.cs- for selecting one of our enumerators given an int stored in the db to represent it’s value.

Database Implementation Overview



EER Diagram of our database backend. Figure is a central component to our db's organization

Testing

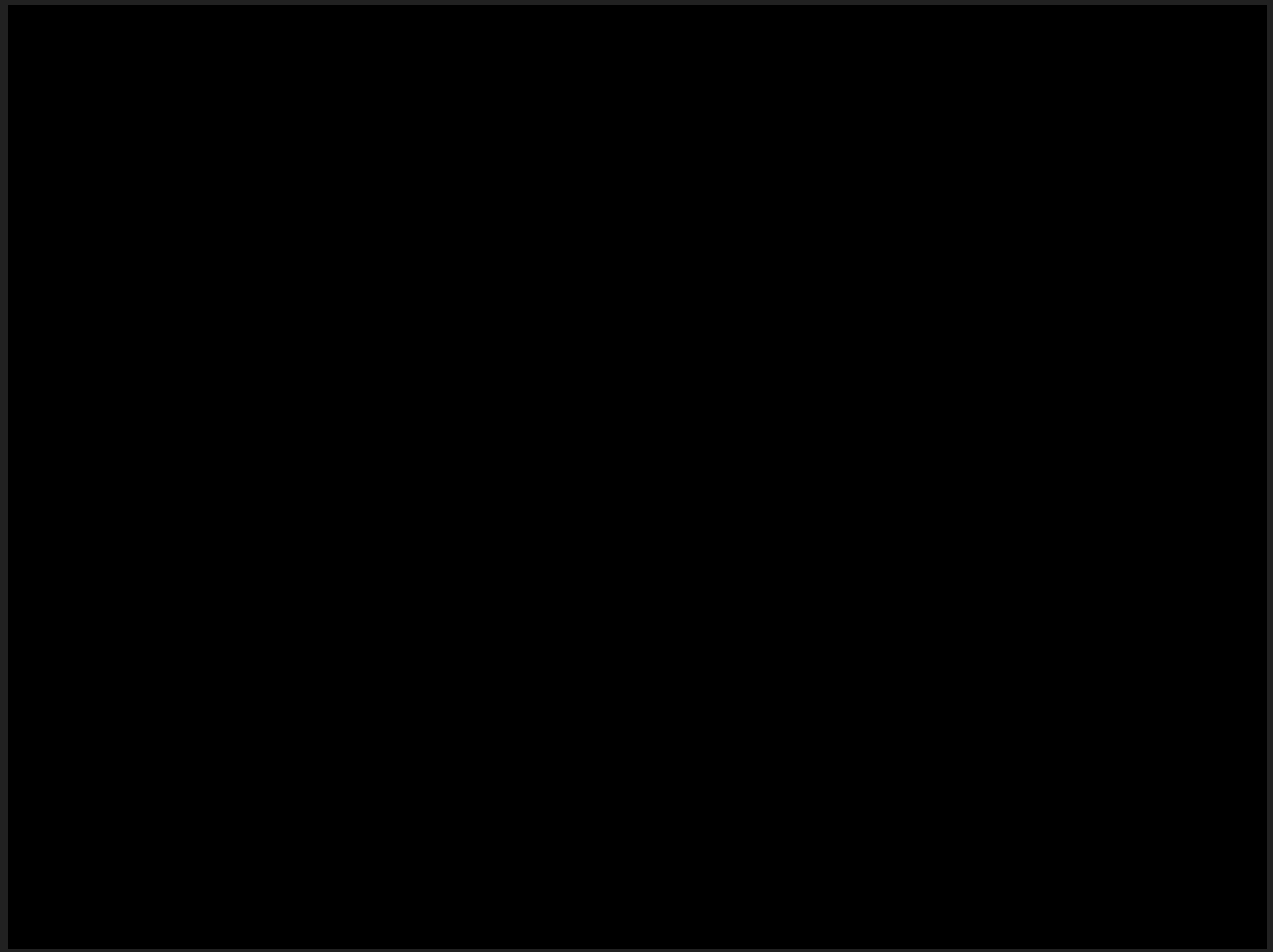
Testing

- Testing was completed in a two-fold approach, with the endpoints being functionally tested manually and the service layer being unit testing individually.
- For functional testing, .NET utilizes Swagger and SwaggerUI which is a tool that allows devs to visualize their API’s.
- We also used insomnia as well just to give variety and exposure to the software.

Tools

- Git/github: Github was used for our versioning system. At all times a MAIN branch was kept as the current version and other branches were created, committed, and merged into the main branch upon approval from the team. Pushes directly to the main branch were blocked and required a merge. This was done to mimic the Agile workflow and concepts of CI/CD as well as Semantic Versioning to keep track of changes.
    - Changelog, tagged releases, and full code available on github
    -
- ASP.NET: ASP.NET is the framework for building API's using the .NET stack of HTML, JS, C#, and SQL. We used C# and MySQL o create OBG API in v0.0.1 and will use JS in future versions.

Final Project Outcome



Video Demo

Final Project Outcome

We were able to accomplish our mVP for v0.0.1

- Database is defined and test tuples are in db for FACT
- All 4 endpoints created
- Endpoint /facts/get returns a single test tuple in Swagger

https://github.com/roderick-bishop11/seniorProject

Future work:

- Expand db & host remotely
- Connect db to API using abstracted configuration
- Implement UI for end users
- Create and host front end using JavaScript
- Expand data models to accept images and larger bodies of text
- Release V1.0.0 as Open Source on github and SCSU website

Learning

- **Roderick Bishop-** This project is very special and dear to my heart. Originally a very rough personal idea, I pitched it and the group agreed to pursue its creation. Together, we decided how this API should work and what we wanted out of this project for CS460 as well as continuing OBGAPI's build out and eventual launch. This project was definitely more than we had asked for at times, and it was a lot of experiential learning along the way. Together we learned so much about C#, SQL, the application & employment of development processes & concepts that we've learned in a previous course at SCSU (CS405), and a lot more about API's as a whole. Overall, I am glad that we were able to build our mVP, and present it to complete the course. Going forward, I'd love to see OBGAPI fully realized and used in many mobile or web apps, websites, etc. It's my hope that this will be an open-source project for young Black Engineers to contribute as well as learn about the wonderful people that helped shape the world that they know and love.

- **Taighlor Moultrie -** This project was something that my partner wanted to create for a long time. Starting this project was very difficult, I didn't know most of the materials my partner wanted me to do. I asked a lot of questions about the project that helped us get it up and running. This project definitely was a challenge but it taught me alot. Together we learned a lot about databases, the application, & api. Overall, I am glad I took CS460. It was definitely a big learning experience. I would recommend this course to anybody because it gives you the opportunity to think outside the box and create something cool.