# Architectural Styles and Framework for CesarMusicEmporium

The CesarMusicEmporium API is a RESTful MVC service that allows users to query a database that contains music album and artist information. As part of this project, the RESTful architectural style and MVC design pattern has been used in order to implement this system.

The following is a list of styles and patterns that have been used in the development of this project:

- Client-Server Architecture: This architecture has been used in order to separate the backend (servers/databases logic) from the front end by having a separate service that handles API requests, all while having a front-end project that simply makes calls to the service and displays JSON response data to the client.
- Model-View-Controller (MVC): This pattern was used in order to separate data (Model), user interface (View), and handle user interactions such as queries/API requests (Controller).
- Repository: This design pattern was implemented by abstracting data access from other code applications.
- Dependency Injection: This pattern was implemented in order to allow the use of dependencies, in this case connection strings to our local MS Access database.
- Factory: A factory was used in order to create an interface between new objects that are generated.

The use of the previous design patterns and architectural patterns allowed for the quick development of the software product and service, it also ensures that if future additions to the software need to be added, they can be done so quickly with minimal impact on the software, in the case of the MVC pattern, the abstraction and separation of concerns between code allows new endpoints to be implemented easily and quickly.

Technologies used:

- ASP.NET Core 6.0 (The main framework used for the development of this project)
- Swagger API documentation (Allowed the creation of API documentation and allows API endpoints to be tested)
- Microsoft Access Database (Used as our local database to store records from which our API will retrieve data)
- Entity Framework (To facilitate database interactions)
- Newtonsoft.Json (Facilitated the parsing of JSON data)
- Microsoft.Cors (Allows us to make requests by adding correct CORS headers to our endpoint requests)