**Here's my description of the approach and limitations of the implementation, and possible room for improvement in the future:**

I designed my implementation to provide a REST API that returns the most popular movie genres for a given year. I organized the code in a modular way with a separation of concerns between business logic, request logic, and data logic. I also added error handling for various cases and unit and integration tests to ensure the correctness and reliability of the code. I made the code easily run in a production environment using Docker, and I ensured that the service was well-documented.

One limitation of my implementation is that it relies on a specific input file ('movies.tsv') to load the data, which means that the service can only be used with that particular file. While the code is designed to be scalable, it may require additional configuration and infrastructure to handle a very large load.

There are several areas where my implementation could be improved in the future:

* **Flexibility**: To make the service more flexible and adaptable to different use cases, I could consider adding additional endpoints or parameters to the API to allow users to customize the results returned.
* **Performance**: To improve the performance of the service, I could consider adding caching to return the most popular genres more quickly.
* **Security**: To improve the security of the service, I could consider adding authentication and authorization mechanisms to ensure that only authorized users can access the API.

I believe that my implementation provides a solid foundation for a REST API that returns the most popular movie genres for a given year, and there are many opportunities for future improvement and enhancement.