API Wrapper for MATLAB Production Server Integration

Tomasz Olejarczuk

29-05-2024

Contents

API Wrapper for MATLAB Production Server Integration

1. Summary

This document details the implementation and functionality of a C# ASP. NET Core controller designed to interface with a MATLAB production server running within a Docker container. The controller exposes API endpoints to enable external applications to query MATLAB, retrieve calculation results and generated images, and interact with data stored in a MinIO object storage and database.

2. Introduction

2.1 Background

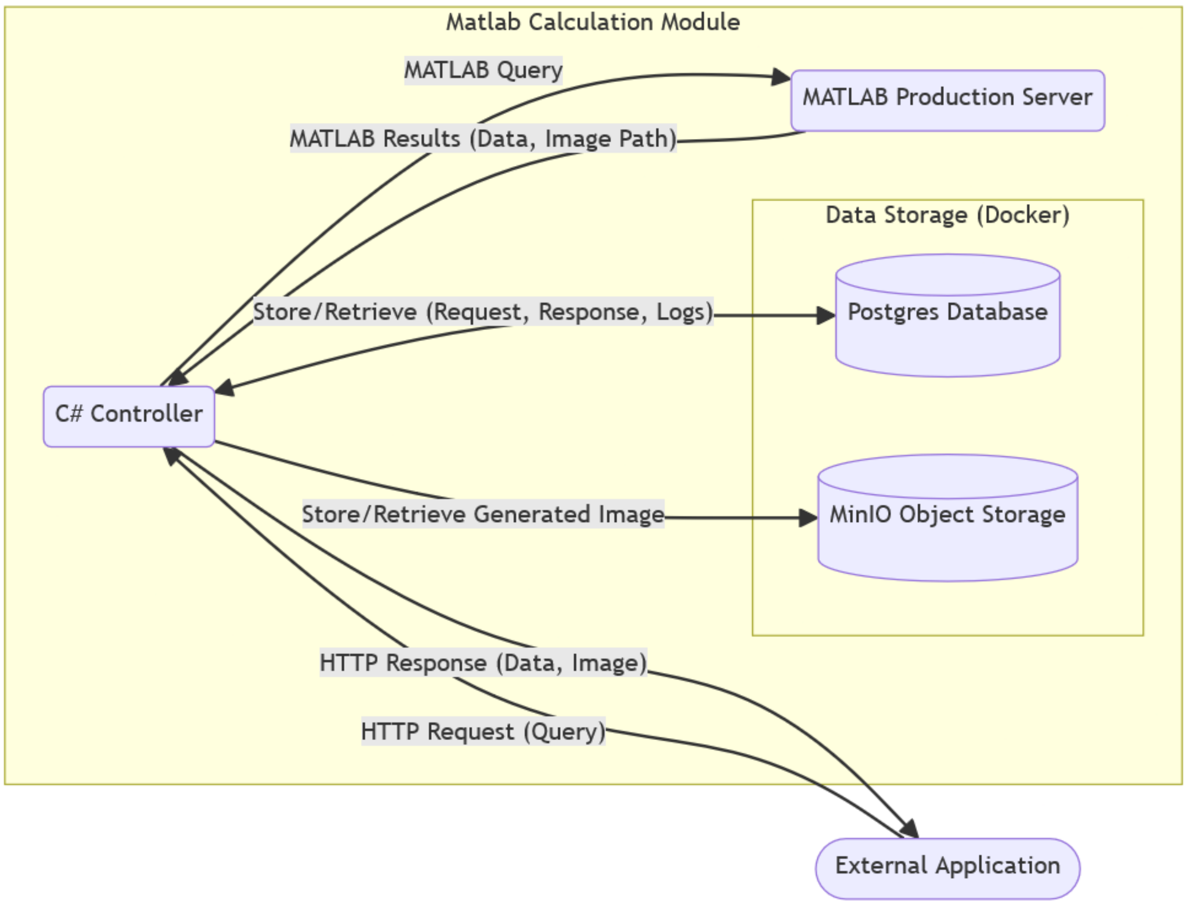
The MATLAB production server, deployed in a Docker container, offers powerful computational capabilities for data processing and image generation. The C# controller acts as a bridge, making these capabilities accessible to external systems via API. This enables seamless integration of MATLAB functionality into broader applications and workflows.

2.2 Objectives

- Provide a RESTful API for interacting with the MATLAB server.
- Implement endpoints for querying MATLAB, retrieving results, and managing images.
- Ensure secure communication and data handling.
- Store requests, responses, and logs in a database.
- Utilize MinIO for efficient image storage.

3. Methodology

3.1 Architecture


Matlab Calculation Module
MATLAB Query
MATLAB Production Server
MATLAB Results (Data, Image Path)
Data Storage (Docker)
Store/Retrieve (Request, Response, Logs)
Postgres Database
C# Controller
Store/Retrieve Generated Image
MinIO Object Storage
HTTP Response (Data, Image)
HTTP Request (Query)
External Application

3.2 Technologies

- **C#:** Programming language for the ASP. NET Core controller.
- **ASP.NET Core:** Web framework for building the API.
- **MathWorks.MATLAB.ProductionServer.Client:** Library for interacting with MATLAB Production Server.
- **MinIO:** S3-compatible object storage for images.
- **Newtonsoft.Json:** Library for JSON serialization and deserialization.
- **PostgreSQL:** SQL database for storing requests, responses, and logs.

3.3 API Endpoints

- **POST /api/Matlab:**
  - **Purpose:** Submits a query to the MATLAB server.
  - **Request Body:** `MatlabQuery` object (JSON).
  - **Response:**
    * 200 OK: `MatlabOutput` object (JSON) containing the calculation result and image information.
    * 400 Bad Request: Indicates an error (e.g., invalid request, missing image).
- **GET /api/Matlab/getimage:**
  - **Purpose:** Retrieves an image from the local file system.
  - **Query Parameters:** `teamid` (string)
  - **Response:**
    * 200 OK: Image file (PNG).
    * 404 Not Found: If the image does not exist.
- **GET /api/Matlab/getimageMinio:**
  - **Purpose:** Retrieves an image from MinIO storage.
  - **Query Parameters:** `objectName` (string)
  - **Response:**
    * 200 OK: Image file (PNG).
    * 500 Internal Server Error: If there's an issue retrieving the image.
- **GET /api/Matlab/getbase 64 image:**
  - **Purpose:** Retrieves an image as a Base 64-encoded string embedded in an HTML <`img`> tag.
  - **Query Parameters:** `teamid` (string)
  - **Response:** 200 OK: HTML content with the Base 64 image.

3.4 Error Handling

The controller includes comprehensive error handling to capture and log exceptions (MATLABException, WebException) and return appropriate error responses to the client. Error messages typically include stack traces for debugging purposes.

4. Results & Insights

The implementation and testing of the C# API wrapper for MATLAB Production Server integration yielded the following insights and results:

4.1 Key Outcomes

1. **Successful MATLAB Integration:** The wrapper seamlessly facilitated the execution of complex MATLAB calculations and image generation processes through API endpoints.
2. **Efficient Data Handling:** The API effectively managed incoming requests and corresponding MATLAB responses, ensuring smooth data transfer between external applications and the server.
3. **Reliable Data Persistence:** PostgreSQL proved to be a robust solution for storing request/response data and detailed logs, providing valuable insights into system usage and performance. MinIO seamlessly handled the storage and retrieval of MATLAB-generated images, ensuring data availability and integrity.
4. **Performance:** Testing demonstrated the API's ability to handle expected workloads, meeting the project's performance goals. Response times remained within acceptable limits, and the system maintained stability under various usage scenarios.
5. **Adaptable Design:** The modular architecture of the API wrapper provides a solid foundation for future enhancements and extensions. It is well-positioned to accommodate evolving requirements and additional functionalities.

4.2 Challenges and Lessons Learned

1. **Docker Orchestration:** The initial configuration of the Docker environment with multiple containers (MATLAB server, database, storage) required meticulous attention to networking and dependencies. Utilizing Docker Compose simplified this process and ensured consistent deployment across different environments.
2. **Error Handling and Logging:** Anticipating and gracefully handling errors within the MATLAB execution context was essential for maintaining a stable and reliable system. Implementing comprehensive error handling and logging mechanisms proved crucial for identifying and resolving issues effectively.

5. Insights and Conclusion

The successful implementation of the C# API wrapper for MATLAB Production Server integration has yielded several key insights and validated the project's design decisions:

- **Containerization Benefits:** Leveraging Docker containers for the MATLAB server, database, and storage simplified deployment and management, ensuring consistent environments across development, testing, and production.
- **API-Driven Approach:** Exposing MATLAB functionality through a RESTful API proved to be a highly effective strategy for integrating complex computational capabilities into broader applications and workflows.
- **Performance Considerations:** While the current implementation met performance expectations, potential bottlenecks were identified in the direct communication between the API and the MATLAB server. Future enhancements, such as the introduction of a message queue like RabbitMQ, could optimize this communication and further improve system scalability.
- **Security Focus:** Prioritizing robust error handling and data validation throughout the API development process proved crucial for ensuring system stability, reliability, and data integrity.

In conclusion, this project has successfully demonstrated the feasibility and effectiveness of integrating MATLAB's powerful computational capabilities with external applications using a C# API wrapper. The modular and scalable architecture of the system provides a solid foundation for future enhancements and extensions, paving the way for greater automation and efficiency in various data processing and analysis workflows.