NET Core 6 application that integrates with Azure Functions and Azure Storage

Task 1

Brief Project Documentation

Piotr Stola

Contents

Introduction

Develop a .NET Core 6 application that integrates with Azure Functions and Azure Storage (both cloud and local emulator options for storage) to periodically fetch data from a specified API and manage logging and data retrieval. The application should:

Utilize Azure Functions to trigger every minute, requesting data from the "https://api.publicapis.org/random?auth=null" endpoint.

Record each attempt (both successful and unsuccessful) in Azure Table Storage, including details for tracking and review.

Save the complete payload received from the API call into Azure Blob Storage for persistence.

Implement a GET API endpoint to enumerate logs within a specified time range (start to end), allowing users to review the history of data fetch attempts.

Provide a GET API endpoint to retrieve the full payload data from a specific log entry stored in the blob, based on log identifiers or criteria.

Ensure the entire codebase is published on GitHub, making it publicly accessible for review, contributions, or deployment

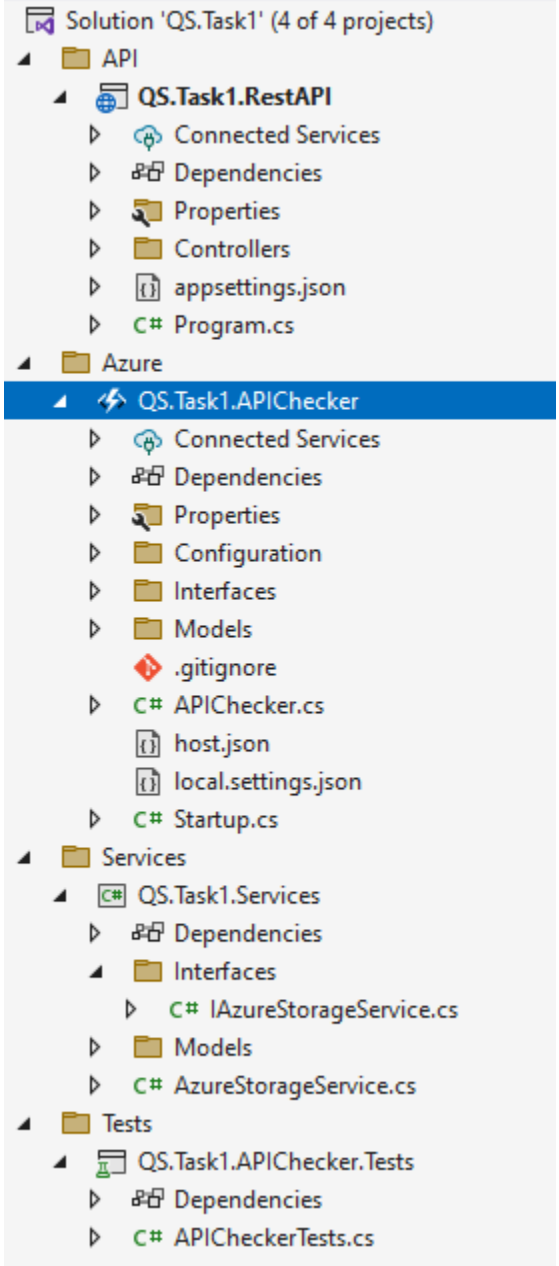
Technology stack

Projects is based on following list of technologies and frameworks.

.NET 6.0
Azure Functions – SDK 4.3.0
Azure Key Vault – Azure Service
Azure Storage Blob - Azure.Storage.Blobs 12.19.1
Azure Storage Tables - Azure.Data.Tables 12.8.3
xUnit Tests – 2.7.0
Moq – 4.20.70

Project structure

The solution exhibits a well-defined project structure, ensuring a clear demarcation between different components. It adheres to key software design principles, specifically embracing the concepts of Separation of Concerns and the Single Responsibility Principle as outlined in the SOLID principle's framework. This approach enhances modularity and maintainability of the codebase.

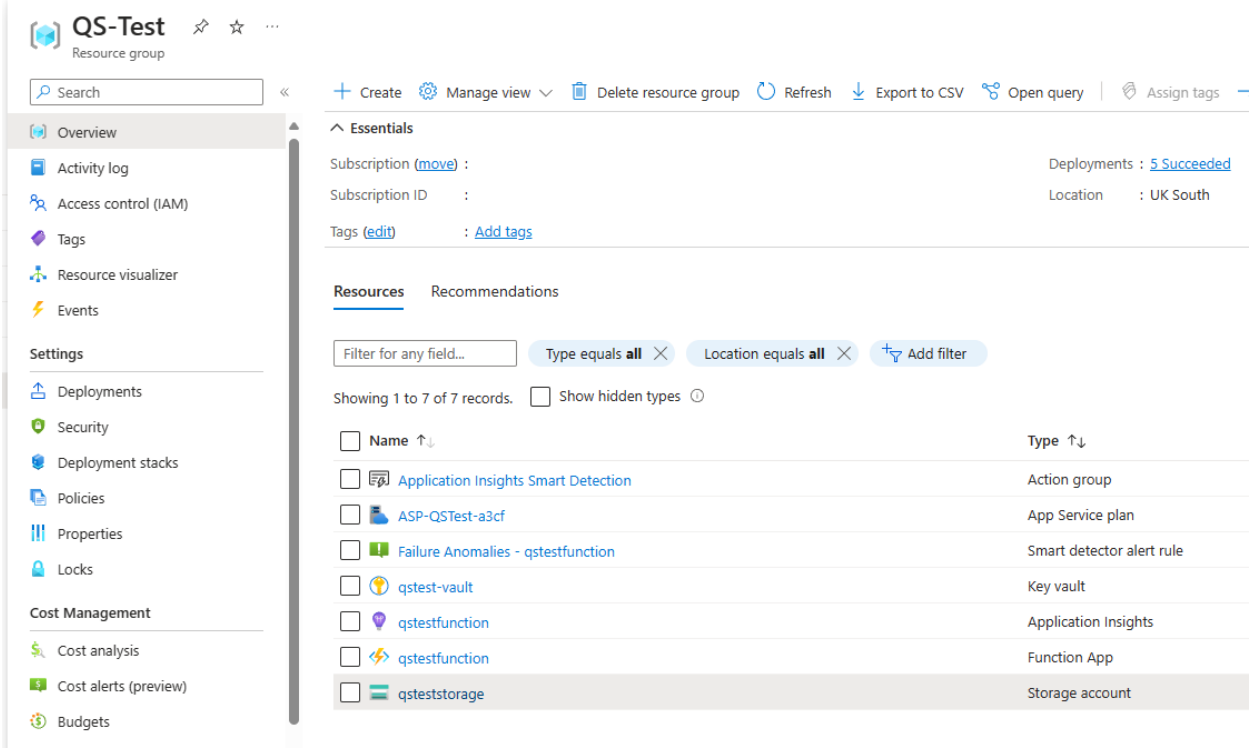


For the core classes within the project, interfaces have been defined to facilitate Dependency Injection, enabling more flexible and decoupled code.

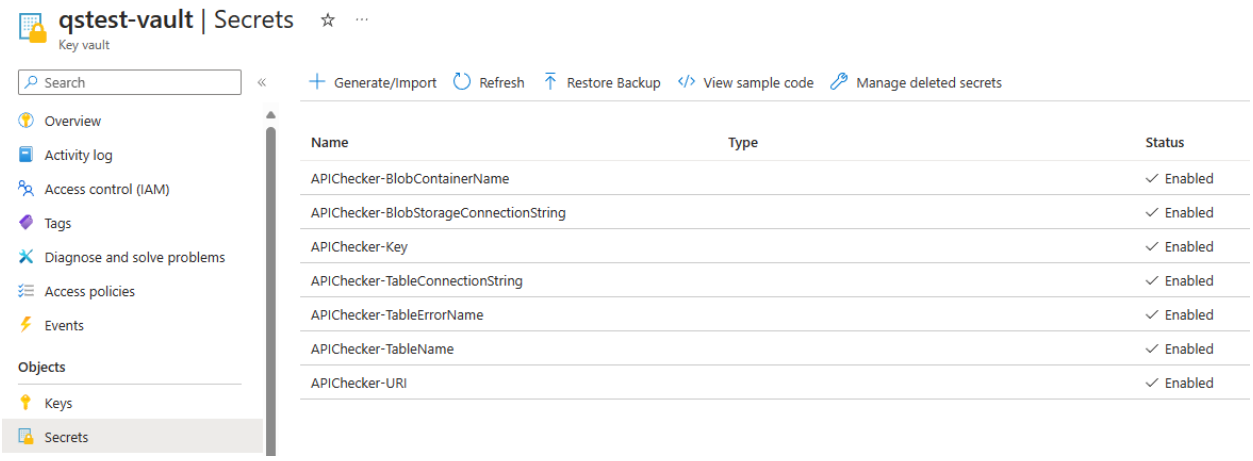
By adhering to these practices, the solution not only promotes code reusability and scalability but also simplifies the process of future enhancements and maintenance.

Azure Project Structure

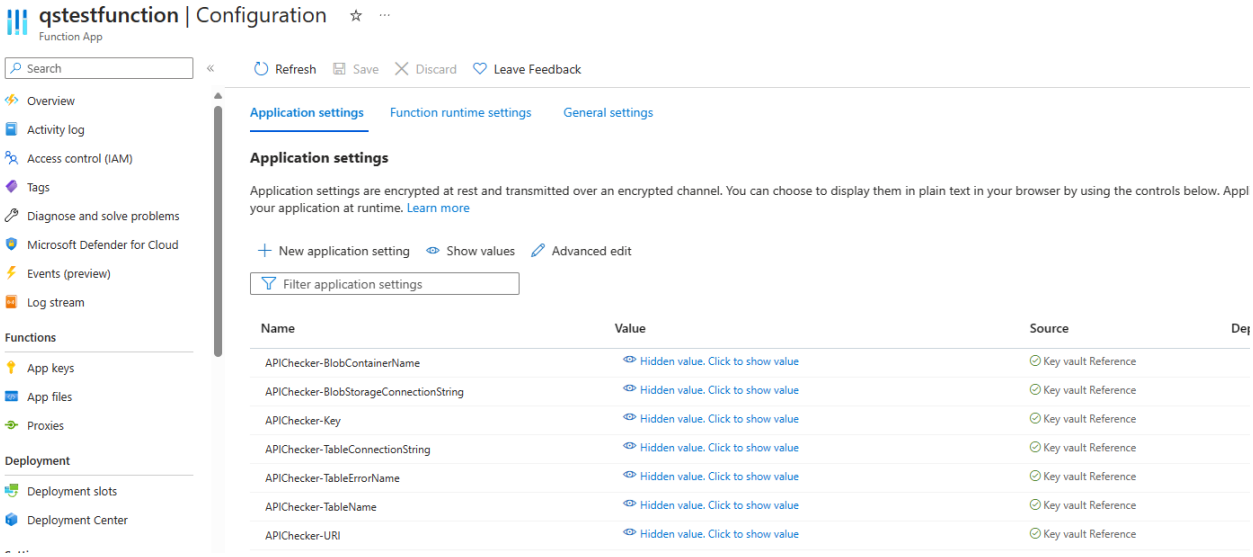
All Azure service in QS-Test resource group



Azure Key Vault Secrets configuration

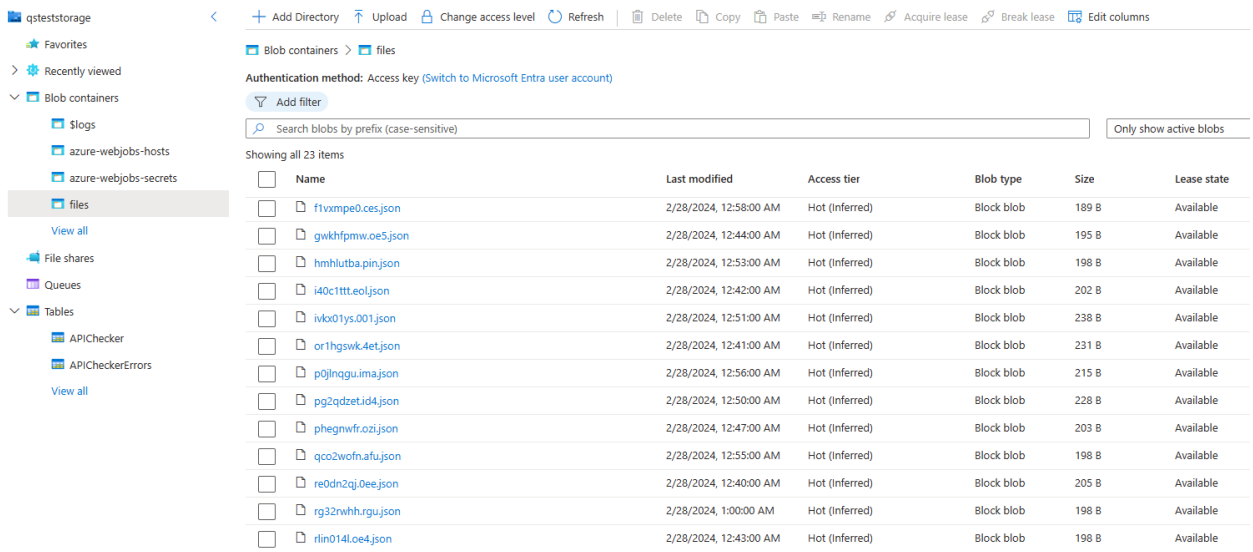


Azure Function bindings App Settings to Azure Key Vault Secrets

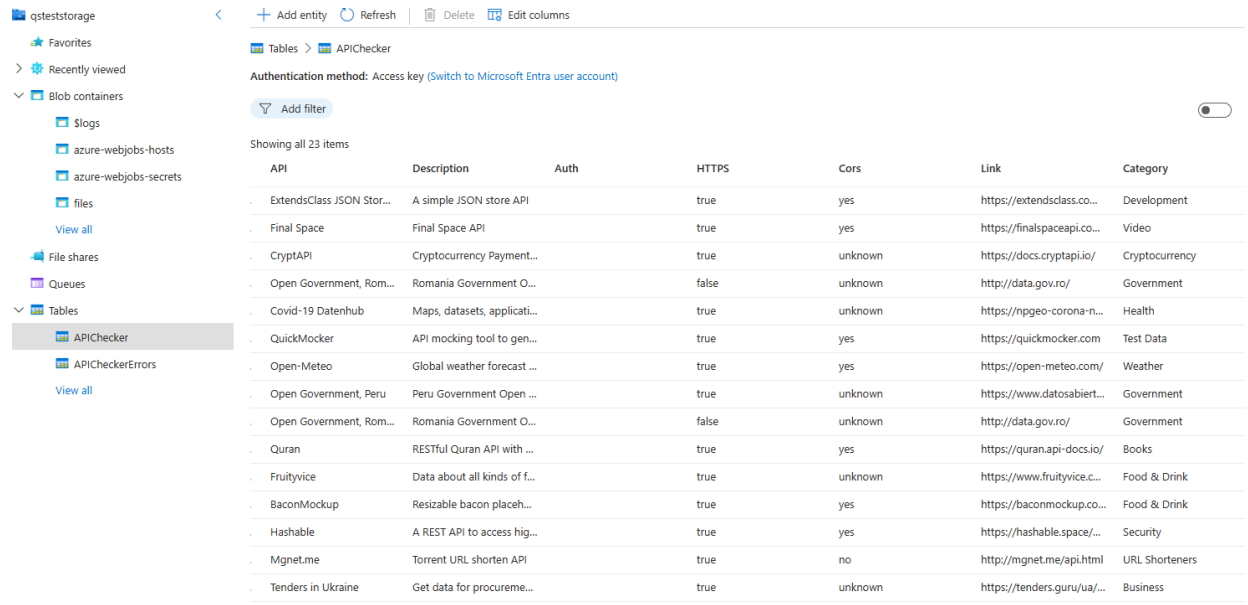


Azure Storage Blog and Tables

Below list of file in Azure Storage blog container and entries in Azure Storage Table



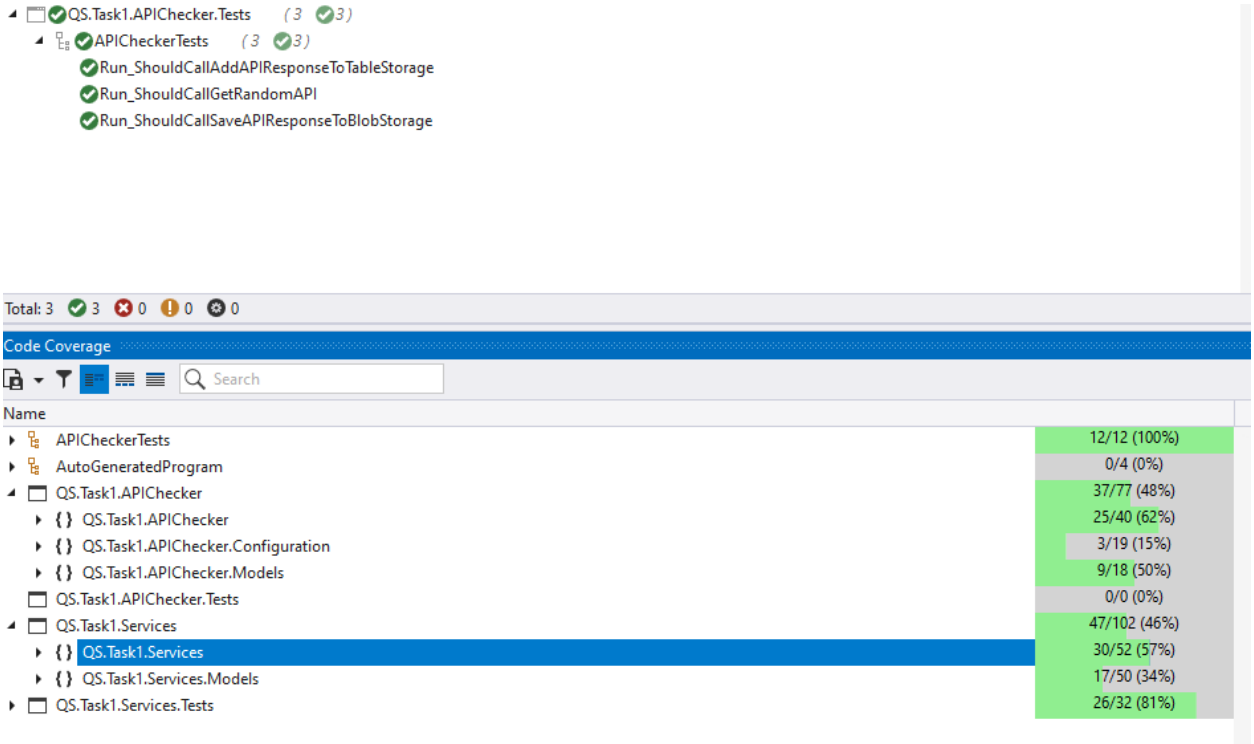
Name	Last modified	Access tier	Blob type	Size	Lease state
f1vxmpe0.ces.json	2/28/2024, 12:58:00 AM	Hot (Inferred)	Block blob	189 B	Available
gwkhfpmw.oe5.json	2/28/2024, 12:44:00 AM	Hot (Inferred)	Block blob	195 B	Available
hmhlutba.pin.json	2/28/2024, 12:53:00 AM	Hot (Inferred)	Block blob	198 B	Available
i40c1ttt.eol.json	2/28/2024, 12:42:00 AM	Hot (Inferred)	Block blob	202 B	Available
ivkx01ys.001.json	2/28/2024, 12:51:00 AM	Hot (Inferred)	Block blob	238 B	Available
or1hgswk.4et.json	2/28/2024, 12:41:00 AM	Hot (Inferred)	Block blob	231 B	Available
p0jlnqgu.ima.json	2/28/2024, 12:56:00 AM	Hot (Inferred)	Block blob	215 B	Available
pg2qdzet.id4.json	2/28/2024, 12:50:00 AM	Hot (Inferred)	Block blob	228 B	Available
phegnwfr.ozi.json	2/28/2024, 12:47:00 AM	Hot (Inferred)	Block blob	203 B	Available
qco2wofn.afu.json	2/28/2024, 12:55:00 AM	Hot (Inferred)	Block blob	198 B	Available
re0dn2qj.0ee.json	2/28/2024, 12:40:00 AM	Hot (Inferred)	Block blob	205 B	Available
rg32rwhh.rgu.json	2/28/2024, 1:00:00 AM	Hot (Inferred)	Block blob	198 B	Available
rlin014l.oe4.json	2/28/2024, 12:43:00 AM	Hot (Inferred)	Block blob	198 B	Available



API	Description	Auth	HTTPS	Cors	Link	Category
ExtendsClass JSON Stor...	A simple JSON store API		true	yes	https://extendsclass.co...	Development
Final Space	Final Space API		true	yes	https://finalspaceapi.co...	Video
CryptAPI	Cryptocurrency Payment...		true	unknown	https://docs.cryptapi.io/	Cryptocurrency
Open Government, Rom...	Romania Government O...		false	unknown	http://data.gov.ro/	Government
Covid-19 Datenhub	Maps, datasets, applicati...		true	unknown	https://npgeo-corona-n...	Health
QuickMocker	API mocking tool to gen...		true	yes	https://quickmocker.com	Test Data
Open-Meteo	Global weather forecast ...		true	yes	https://open-meteo.com/	Weather
Open Government, Peru	Peru Government Open ...		true	unknown	https://www.datosabiert...	Government
Open Government, Rom...	Romania Government O...		false	unknown	http://data.gov.ro/	Government
Quran	RESTful Quran API with ...		true	yes	https://quran.api-docs.io/	Books
Fruityvice	Data about all kinds of f...		true	unknown	https://www.fruityvice.c...	Food & Drink
BaconMockup	Resizable bacon placeh...		true	yes	https://baconmockup.co...	Food & Drink
Hashable	A REST API to access hig...		true	yes	https://hashable.space/...	Security
Mgnet.me	Torrent URL shorten API		true	no	http://mgnet.me/api.html	URL Shorteners
Tenders in Ukraine	Get data for procureme...		true	unknown	https://tenders.guru/ua/...	Business

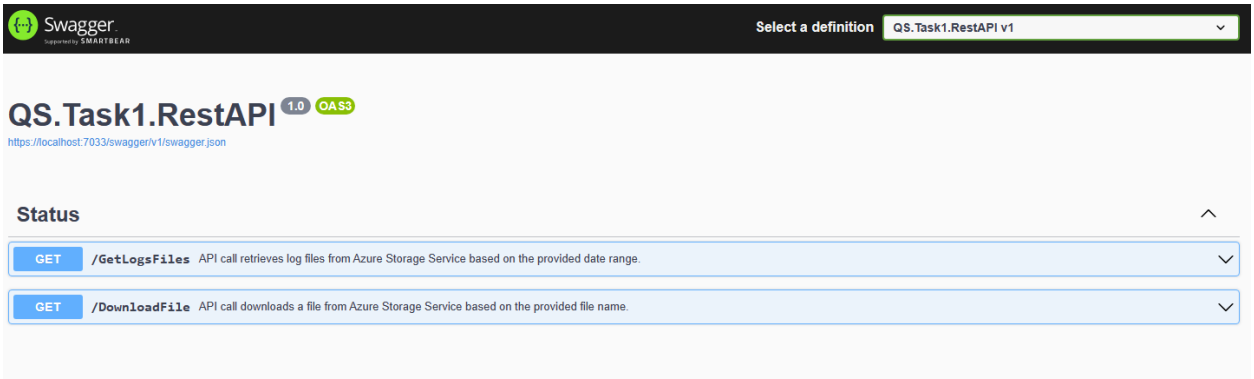
Unit tests and Test Coverage

Please find below unit test for Azure Function code.

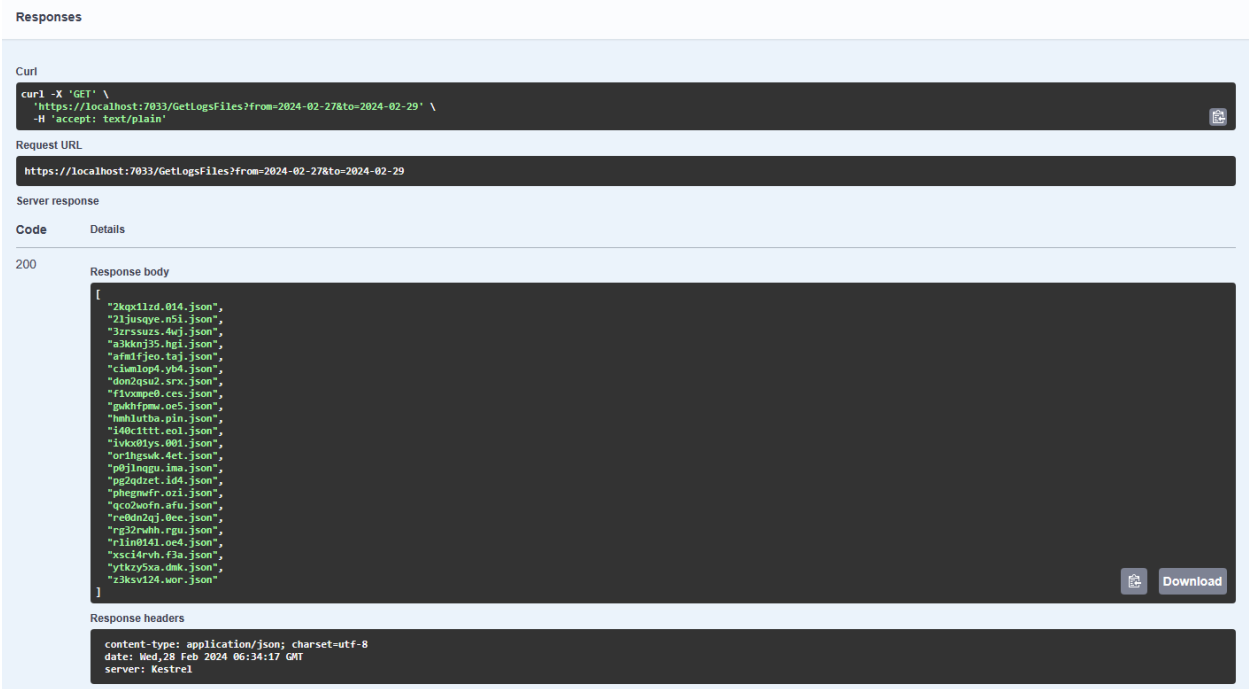


Rest API

The Status API is designed to manage HTTP requests concerning the application's status. It offers endpoints for accessing log files from Azure Storage Service within a specified date range. Additionally, it facilitates the download of a file from Azure Storage Service when a file name is provided. This API plays a crucial role in monitoring and maintaining the application's health by enabling efficient retrieval and download of essential data.



Api executions:



Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7033/DownloadFile?fileName=2kqx1lzd.014.json' \
  -H 'accept: text/plain'
```

Request URL

```
https://localhost:7033/DownloadFile?fileName=2kqx1lzd.014.json
```

Server response

Code	Details
200	Response body Download file Response headers

```
content-disposition: attachment; filename=2kqx1lzd.014.json; filename*=UTF-8''2kqx1lzd.014.json
content-length: 171
content-type: application/octet-stream
date: Wed,28 Feb 2024 06:36:29 GMT
server: Kestrel
```

Downloaded file content

2kqx1lzd.014 (1).json

File Edit View


```
{"count":1,"entries":[{"API":"Kickbox","Description":"Email verification API","Auth":"",
"HTTPS":true,"Cors":"yes","Link":"https://open.kickbox.com/","Category":"Email"}]}
```


Time report

Time used to create, design, and develop specific part of system.

Task Name	Time [h]
Azure Function and Service Project	1,5
Unit test	0,5
API	1
Azure project configuration	1
Documentation	1
Total :	**6h**

Future development and improvements

> "There's always room for improvement, you know-it's the biggest room in the house." —
>
> **Louise Heath Leber**

I would like to introduce few concepts and potential improvements for project future evolution.

Project name:

1. QS.Task1.RestAPI
    a. Implementation of Authentication and Authorization mechanism
    b. Depends on Rest API hosting method, we should consider improving AppSettings key management.
    c. Introduce validation mechanism for API request parameters.
2. QS.Task1.Services
    a. Introduce custom error class
    b. Azure function is already connected to Application insights, I could integrate error logging mechanism with Application Insights.
3. QS.Task1.APIChecker.Tests
    a. Integration tests for QS.Task1.APIChecker
    b. Unit test and integration test for QS.Task1.Services