

Research

Individual Project Semester 6

Veronika Valeva 4090349

Fontys, Eindhoven

22/3/2023

How to build a Question & Answer service?

Table of Contents

Introduction	3
Problem.....	3
Relational/ non-relational database	3
CSV/ JSON files	4
Cloud storage	4
API	4
Content Management System (CSM)	4
Solution	5
Technology.....	5
Deployment	6
Continuous Integration and Continuous Delivery	6

Introduction

In the case of my individual project (The Quiz Game) There should be a service that provides questions and answers to the game logic service. The Questions & Answers Service Is going to be built to gather questions and answers based on a set of options sent by the game logic service as parameters. This research is going to find a solution to the question what Questions & Answers Service implementation would best fit the current project. The needed Q&A data consists of Questions, Answers and possible Question categories – all of this data needs to be provided by the Questions & Answers Service for a proper process execution.

After logging in the user is going to be able to create a quiz session for himself. The quiz session is created according to a set of options. The set of options consists of number of questions, question category, question difficulty, question difficulty and question type. Number of questions are going to be between ten (minimum) and fifty (maximum). The question categories are going to be, but not limited to Arts & Literature, Movies & TV, Food & Drink, General Knowledge, Geography, History, Music, Science, Society & Culture and Sport & Leisure. The question difficulty being easy, medium, and hard. The question type being either multiple choice (where a question has several answers and one of them is correct) or true/ false (where the question has two answers – “true” or “false” and one of them is correct.)

Problem

To gather Q&A data the service needs to have access to a Q&A data storage. There are several possible solutions for accessing to a Q&A data storage. Relational/ non-relational databases like [MySQL](#), [PostgreSQL](#), and [MongoDB](#) are commonly used for storing structured/ unstructured data. Q&A Data could be stored in files, such as [CSV](#) or [JSON files](#). [Microsoft Azure Storage](#) can be used to store Q&A data. One can also make use of APIs like the [Trivia API](#) or [OpenTrivia API](#) to retrieve Q&A data from external sources. Another solution is using [content management systems \(CMS\)](#) software applications to create, manage, and publish digital Q&A content.

Relational/ non-relational database

A database is an organized collection of data that can be accessed and managed in a structured way. When it comes to storing Q&A data, a database can be a good option if large amounts of structured or unstructured data need to be stored and managed. Using a database, one can easily create tables or collections to store Q&A data, and then use queries to retrieve and analyze that data as needed. One of the key benefits of using a database to store Q&A data is that it provides a reliable and scalable way to store and manage data. Databases can be configured to ensure data integrity and consistency and can be scaled horizontally to handle large volumes of data. As the amount of data stored in the database grows, there may be challenges in maintaining the performance and availability of the database. The database may require more resources, such as storage, memory, or processing power, to maintain optimal performance.

CSV/ JSON files

Storing Q&A data in CSV or JSON files, can be a good option if working with small amount of data and there is no need to store the data in a structured database. Storing Q&A data in files, can be done using a spreadsheet program like [Microsoft Excel](#) or [Google Sheets](#) to create and manage CSV files, or a text editor or specialized software like [Notepad++](#) to create and manage JSON files. One of the main benefits of storing Q&A data in files is that it is a simple and straightforward way to store and manage small amounts of data. Files can be easily saved and shared and managed. However, there are also some potential drawbacks to storing Q&A data in files, such as potential data inconsistencies or errors if the files are manually edited or updated, managing files in this way can be time consuming.

Cloud storage

Cloud storage services like Microsoft Azure Storage provide a scalable and reliable way to store and manage data in a cloud. These services offer a range of features and benefits, such as scalability, availability, and durability, that make them a popular choice for storing large amounts of data. When storing Q&A data in cloud storage services, one can use APIs or web interfaces to upload, manage, and retrieve data from the cloud. These services are designed to handle large volumes of data and can be scaled up or down depending on your specific needs and requirements. However, there are also some potential drawbacks to using cloud storage services, such as potential relying on the provider – if the provider experiences technical issues the data can become temporarily inaccessible. Cloud storage services may charge for data storage, data transfer, and other services, which can result in unexpected costs.

API

APIs (Application Programming Interfaces) allow different software applications to interact with each other by exchanging data and functionality. APIs provide a standardized way for software applications to communicate with each other, which makes it possible to integrate different systems and services seamlessly. In the context of Q&A data, there are various APIs available that allow the retrieve of Q&A data from external sources, such as the Trivia API. These APIs provide access to pre-existing data sources that have already been compiled and organized into question-and-answer formats. To use an API to retrieve Q&A data, one needs to integrate the API into a software application or website. This involves making requests to the API, which will return data in a predefined format, such as JSON or XML. Then the data can be processed and displayed. Using APIs to retrieve Q&A data can be a good option to avoid storing the data and to take advantage of pre-existing data sources. This approach can save time and effort in compiling and organizing Q&A data. However, it is important to carefully choose the API that meets your needs and requirements, and to ensure that the data provided by the API is accurate and up-to-date.

Content Management System (CMS)

A CMS is a software application that enables users to create, manage, and publish digital content, such as text, images, and videos. CMSs provide a range of features that make it easier to manage large amounts of content. How it works? To use a CMS like WordPress to store Q&A data,

one would need to create a new content type specifically for Q&A data. In a CMS, a content type is a pre-defined data structure that specifies the fields and properties of a particular type of content. Once the Q&A content type has been created, Q&A data can be added to the CMS. This can be done manually, by entering the data directly into the CMS, or through automated imports from other sources, such as CSV files. Using a CMS to store Q&A data has several benefits. Firstly, it provides a centralized and organized way to manage large amounts of content. Secondly, CMSs often provide built-in tools for content versioning and workflow management, which can make it easier to collaborate on content creation and updates. Finally, CMSs typically have user-friendly interfaces that make it easy to add, edit, and manage content without requiring specialized technical skills. However, there are also some potential downsides to using a CMS for storing Q&A data. Firstly, CMSs can be resource-intensive and require a significant amount of server space to store large amounts of content. Secondly, using a CMS requires some technical expertise to set up and configure, which can be a barrier to entry for some users. Finally, CMSs may require ongoing maintenance and updates to ensure that they remain secure and functional.

Solution

I have chosen to use the option of calling APIs to retrieve Q&A data from external sources. This solution allows me to outsource the Q&A data storage to pre-existing data sources and take advantage of their already available data. Additionally, it eliminates the need to manage and maintain a database or content management system. Since I have never used APIs to retrieve data before, I am curious about how it works. When choosing the TriviaAPI for retrieving Q&A data, there are several factors that were considered. The TriviaAPI provides accurate and up-to-date Q&A data that meets the requirements of the Quiz Game project. The API provides clear and comprehensive documentation which would ensure a smooth developing process. The API is open-source and free to use for the conditions set by the requirements of the project. In other words, the API offers a range of pricing plans, including a free plan with limited usage, and affordable paid plans for higher usage levels. The TriviaAPI provides a large and diverse collection of high-quality trivia questions and answers. The API offers multiple categories of questions including all of the categories mention in the beginning of the document. The TriviaAPI is a reliable and stable service that has been operating for several years, with minimal downtime or connectivity issues. It is compatible with a wide range of programming languages and frameworks.

Technology

There are several technologies that can be used to build the Q&A service. Some of the commonly used ones include [Python](#), [Java](#), and [Ruby on Rails](#). However, for my case, I have decided to use [FastAPI](#). FastAPI is a modern, fast, web framework for building APIs. As the name suggests, FastAPI is designed to be fast. It uses the high-performance web server, [Uvicorn](#), and is built on top of the [Starlette framework](#), which is itself a lightweight and performant framework. This makes FastAPI an excellent choice for building a service that calls TriviaAPI. FastAPI is designed to be easy to use and has a simple and intuitive API. It also includes features like automatic data validation, which can save you time and effort when building your service. FastAPI has comprehensive and easy-to-understand documentation. It also includes automatic generation of API documentation using the [OpenAPI standard](#), which can be very helpful when building and maintaining code. FastAPI makes

use of modern Python features like [type hints](#), which can help make your code more readable and maintainable. It also supports [asynchronous programming](#) using the `async/await` syntax, which can help you write more efficient and scalable code. Its simplicity, performance, and comprehensive documentation make it an ideal option for building fast and efficient APIs with Python.

Deployment

[Docker](#) is a containerization technology that allows you to package an application and its dependencies into a portable container. This means that the Q&A application can be deployed in a consistent way, regardless of the environment in which it is running. Using Docker to deploy the FastAPI Q&A service, ensures that the application will run the same way in all environments, including local development, staging, and production. This helps to minimize the risk of unexpected issues or bugs when deploying the application to production. In addition, Docker provides benefits for CI/CD workflows, such as easier integration with [GitLab](#) CI, and the ability to automate the deployment of the application to different environments.



Figure 1 - Docker container of the Q&A application

Continuous Integration and Continuous Delivery

Using Continuous Integration and Continuous Delivery (CI/CD) is an essential part of modern software development, and GitLab CI/CD is the tool I have chosen for implementing it. Implementing CI/CD, results in the automation of the build, testing, and deployment processes, which will greatly improve the efficiency of the development workflow and reduce the risk of errors or bugs. The committed code is automatically deployed to Docker using GitLab CI/CD. This simplifies the deployment process and ensure that the code that has been tested in the staging environment is the same as the code that is deployed in production.

```

.gitlab-ci.yml
1  image: docker:latest
2
3  services:
4    - docker:dind
5
6  stages:
7    - build
8    - test
9    - deploy
10
11 build:
12   stage: build
13   image: docker:latest
14   services:
15     - docker:dind
16   script:
17     - docker build -t q-and-a-service .
18
19 test:
20   stage: test
21   before_script:
22     - pip install -r requirements.txt
23   script:
24     - pytest app/test.py
25
26 deploy:
27   stage: deploy
28   image: docker:latest
29   services:
30     - docker:dind
31   script:
32     - docker-compose up -d

```

This is the file configuration (.gitlab-ci.yml) that defines the stages and actions to be performed in the [GitLab](#) CI/CD pipeline for the FastAPI Q&A service. The first part of the file specifies the Docker image to use and the [Docker-in-Docker \(dind\)](#) service to run the pipeline. The second part defines three stages: build, test, and deploy. The build stage creates a Docker image of the Q&A service by running a Docker build command. The test stage installs the service's requirements and runs automated tests using [pytest](#). The deploy stage deploys the service by running a Docker Compose command to start the service in a Docker container. This file automates the building, testing, and deployment of the FastAPI Q&A service using GitLab CI/CD and Docker, making the development process more efficient and reliable.

Figure 2- GitLab CI/CD file




Status	Job	Stage	Name	Duration
 passed	#503285 main → be6039ce	deploy	deploy	00:00:08 1 week ago
 passed	#503284 main → be6039ce	test	test	00:00:25 1 week ago
 passed	#503283 main → be6039ce	build	build	00:00:25 1 week ago

Figure 3 - Pipeline result from the last committed version of the application