EuroSkills Test Project

Web Development 17

Module C

Module C – Commercial Open API

Module C will focus on the implementation of a REST API.

Introduction

You are asked to create a REST API that will be used commercially. The API must be built with features which allow it to be commercialized and made available publicly and openly. The functionality created by you in this module, builds on top of the functionality created in module B. However, you are given a working solution of Module B. This solution includes a database dump including the data that is stored in it. You must use this schema and data and are not allowed to use your own Module B solution.

The functionality of the API will be to provide external access to a number of Artificial Intelligence (AI) services, which will be run as separate services. Those AI services expose a REST API themselves, which however should not be publicly accessible, as they do not have any level of security and reliability as the open API that you create. For development purposes you will have access to the AI services, but in a production environment, these services would be sealed away, so that only your server application could access the AI service endpoints. The code of the AI services is not available to you.

A documentation of the AI services APIs in the form of an OpenAPI specification and a generated documentation are provided. You are supposed to build code which wraps these APIs and exposes them through a single API. The AI services expose different ways of accessing their features and how to pass and receive data. Details on how to access the AI services are also provided in a separate document.

There is a specification for the open API that you are tasked to create. This specification is in the form of an OpenAPI specification and a generated documentation based on it. The specification describes the endpoints that you are supposed to implement. You can find the specification of the API in the file `ai-api/api.yml` and the generated documentation in the file `ai-api/api-docs.html`.

The API shall be implemented using one of the provided frameworks.

Assessment

Module C will be assessed using tools which directly access the API created by you. The API will be tested for its functionality and its adherence to the specification. The API will also be tested for its security and reliability.

Any modifications in the provided backend of previous modules, including any changes to the part of the database that was given, will not be taken into account.

Instructions

The API that you are supposed to create builds on top of the functionality created in module B. You may need to extend the schema of the database to store additional data. Please provide a database SQL dump of the schema and data that you are using. Please do not change the given tables and data, but also include it in your final database dump.

Authentication

The API endpoints must require an API token to be passed in the `X-API-TOKEN` header. The token can be created by the user using the solution of module B. The token shall be used to identify the user and workspace. If the token is not valid, the API must return an `401` error.

Quotas

The API endpoints must enforce the user-defined quotas. The quotas are defined by the user in the solution of module B. The quotas define the maximum that the user wants to spend per workspace in total. It is not known at the start of the API call if the quota will be exceeded. The API must

therefore allow the user to make the call if the quota is not exceeded, but must also enforce the quota if it is exceeded. The last API call may lead to exceeding the quota, but that is accepted. The API must return an `403` error if the quota is already exceeded.

Billing

The API endpoints must track the AI service usage and update the user's billing data accordingly. If an operation fails, the billing data must not be updated. The billing data generated by your API service can be viewed using the solution of module B. The price of a request is computed by multiplying the configured price of that AI service with the number of started milliseconds of the request. The price of a request is rounded up to the next 100th of a cent. If the service uses an asynchronous API, the timer starts to count when the job was started until the job is finished.

Error Handling

There are several types of errors that can occur. The API endpoints must handle these errors and return the appropriate HTTP status code. The API endpoints must also return a JSON object with an error message.

Errors:

- `400` Bad Request: The request is malformed. This error shall be returned if the request is malformed or if the request is missing required fields.
- `401` Unauthorized: The user is not authorized to make the request. This error shall be returned if the token is invalid.
- `403` Forbidden: The user is not allowed to make the request. This error shall be returned if the quota is exceeded.
- `404` Not Found: The requested resource was not found. This error shall be returned if the requested conversation, job or image was not found.
- `503` Service Unavailable: The service is not available. This error shall be returned if the AI service is not available or if the AI service returns an unexpected error.

AI Services

The AI services are:

- provided as separate services.
- are not part of the API that you are supposed to create.
- are not publicly accessible.
- are only accessed from the server application.
- are documented in the form of an OpenAPI specification and a generated documentation.

AI Service 1: ChatterBlast - Chat Bot

This service can be used to chat with.

Usage and Differences to the unified API:

The ChatterBlast API requires that you create the conversation first through an endpoint, before you can send a prompt. So, on the unified API, it's one call, behind the scenes you must do 2 calls. The conversation ID is generated by the code you write, is sent to the ChatterBlast API, provided to the user and stored in the database to prohibit concurrent calls.

After the creation of the conversation, the second endpoint can then be used to fetch a partial conversation response in plain text.

The service you build needs to fetch the `GET` endpoint and provide the partial answer to the user until it is final. The frontend will be polling for the answer. After the response is complete, the user can submit another prompt to continue the conversation. It must not be possible to submit a prompt while the service is still processing the previous conversation input. This means you need to store the state of the conversation on your database.

Billing:

An indicator `<EOF>Took {duration in millis}ms` is printed at the end once the answer is final. Example: `<EOF>Took 4254ms`. The duration signifies how long the service took to generate the response. This is the time that must be used to determine how long it took and consequently update the billing. That indicator must not be part of the response forwarded to the frontend.

Specification:

You can find the specification of the provided API in the file `provided-ai-services/chatterblast.yml` and the generated documentation in the file `provided-ai-services/chatterblast.html`.

AI Service 2: DreamWeaver - Image Generation

This service is used to dream up (generate) images based on a text prompt.

Usage and Differences to the unified API:

The DreamWeaver API accepts a JSON object with a text prompt field. The endpoint is asynchronous and returns a JSON object with a job ID. The job ID can be used to query the status, progress and an image URL that contains a preliminary and low resolution image of what has been generated so far. Once the job is finished, the result contains a resource ID and a high resolution image URL that can be used to retrieve the generated image. The user has further actions to choose from: Upscale 2x, zoom in 2x or zoom out 2x. These actions require the generated image resource ID and also return a job ID.

The image URLs will point to an image that is stored on the AI service server. Your server application needs to download the image and store it locally and generate a new URL for the frontend to access the image. This also needs to work for the preliminary images.

Billing:

You must store when the job started and when it ends to update the billing (since it's an async API call).

Specification:

You can find the specification of the provided API in the file `provided-ai-services/dreamweaver.yml` and the generated documentation in the file `provided-ai-services/dreamweaver.html`.

AI Service 3: MindReader - Image Recognition

This service is used to recognize objects in images.

Usage and Differences to the unified API:

The MindReader API accepts a `multipart/form-data` field `image` and returns a JSON object with the recognized objects and their probabilities. The endpoint is synchronous and returns the result within a few seconds.

The API requires a mapping of fields: The unified API you build must return `x`, `y` and `width`, `height` while the provided API returns `top`, `left`, `bottom`, and `right` for the bounding box.

Billing:

The duration of the request must be used to determine how long it took and consequently update the billing.

Specification:

You can find the specification of the provided API in the file `provided-ai-services/mindreader.yml` and the generated documentation in the file `provided-ai-services/mindreader.html`.