Questionnaires API Documentation

This document aims to explain how to use the **questionnaireApi**. This Api is a backend for a question-answering application using C# thatincludes functionality for users to ask questions, provide answers, and optionally vote on both questions and answers. Each question should have a tag so user can filter the questions by the tags.

**Prerequisites:**

* To have a SQL database server.
* Set or Update the connection string for your database server. Open the appsettings.json file on the questionnarieApi project, then add or modify the connection strings section.

"ConnectionStrings": {

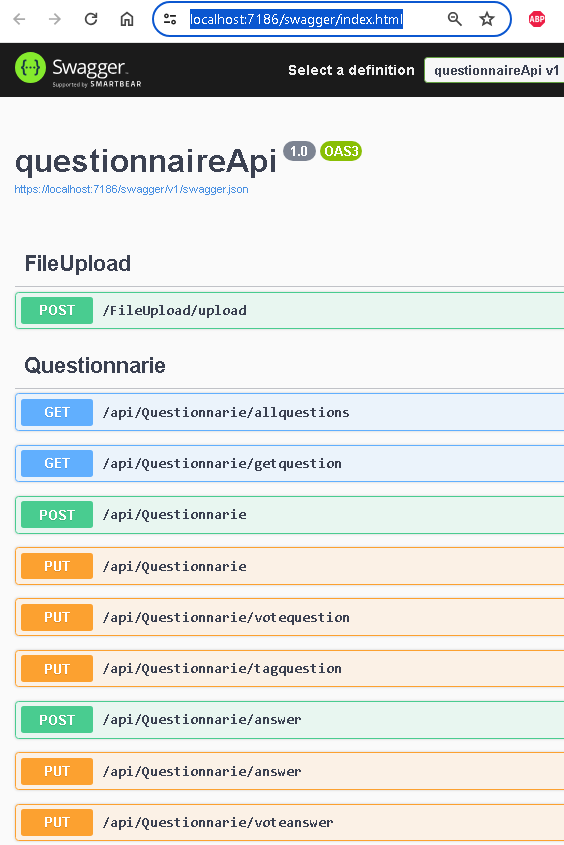
"QConnection": "Server=DESKTOP-QV50632\\SQLEXPRESS; Database=Questionnaries; Trusted\_Connection=True; Encrypt=False; MultipleActiveResultSets=True;"

}

* Make sure the startup project is QuestionarieApi, run it, and verify that the Questionnaries database has been created on the database server.

# Execute

Once the database has been created and the project is running, a browser window will open with the swaager at the following URL <https://localhost:7186/swagger/index.html> as seen below.



Questions

# Get all Questions

It shows all the questions in the database, it also shows associated information such as question votes, the user creator, the answers associated with that question and the tags associated with it.

**Improvements:** Add filters, page number, and page size to avoid bringing in all database records

URL (GET): <https://localhost:7186/api/Questionnarie/allquestions>

# Get a Question

Bring the information to a question specified by your ID. It displays all related information such as votes, the user who asked the question, associated answers, and tags.

**Improvements:** Clean up the response so that it doesn't bring in redundant information.

**URL (GET):**<https://localhost:7186/api/Questionnarie/getquestion?id=3>

# Ask a Question

It allows you to ask a question, you just need to put the question in the value and id of the user asking.

{

"value": "To be the in being?",

"userId": 1

}

URL (POST): <https://localhost:7186/api/Questionnarie>

**Response:**

{

"id": 4,

"value": "To be the in being?",

"votes": 0,

"userId": 1,

"user": null,

"answers": null,

"questionTags": null

}

# Update a question

Allows you to update question, you just need to put the question in the value and id of the user asking.

{

" Id": 1,

"value": "Do I think, therefore do I am?"

"userId": 1

}

URL (PUT): <https://localhost:7186/api/Questionnarie>

**Response:**

{

"id": 1,

"value": "Do I think, therefore do I am?"

"votes": 0,

"userId": 1,

"user": null,

"answers": null,

"questionTags": null

}

# Vote for a question

It is used to increase the vote counter of the question; you just have to invoke it with the id of the question.

**Improvements:** Being able to set votes to zero or any other number, prevent the maximum value of votes from being overflowed. Validate that the user can only vote 1 time.

URL(PUT): <https://localhost:7186/api/Questionnarie/votequestion?id=4>

# Add a tag to a question

It is used to associate a question to a specific topic, you just have to associate the id of the question with the id of the tag, in case that combination already exists, delete it.

URL (POST): <https://localhost:7186/api/Questionnarie/tagquestion>

{

"questionId": 1,

"tagId": 1

}

**Response:**

{

"id": 1,

"questionId": 1,

"tagId": 1,

"tag": {

"id": 1,

"description": "Sports"

}

}

Answers

# Add an answer

It is useful to answer a question, you just have to enter the id of the question and the answer.

URL(POST): <https://localhost:7186/api/Questionnarie/answer>

{

"questionId": 4,

"value": "Answer question 4",

"userId": 1

}

**Response:**

{

"id": 7,

"value": "Answer question 4",

"votes": 0,

"questionId": 4,

"userId": 1,

"user": null

}

# Edit an answer

It is useful to update the answer to a question, you just have to enter the id of the answer and the modified answer.

**Improvements:** Verify that only the user who asked the question can do so.

URL(POST): <https://localhost:7186/api/Questionnarie/answer>

{

"Id": 7,

"value": "Respuis modified to question 4",

"userId": 1

}

**Response:**

{

"id": 7,

"value": "Respuis modified to question 4",

"votes": 0,

"questionId": 4,

"userId": 1,

"user": null

}

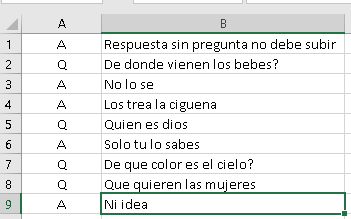
# Vote for an answer

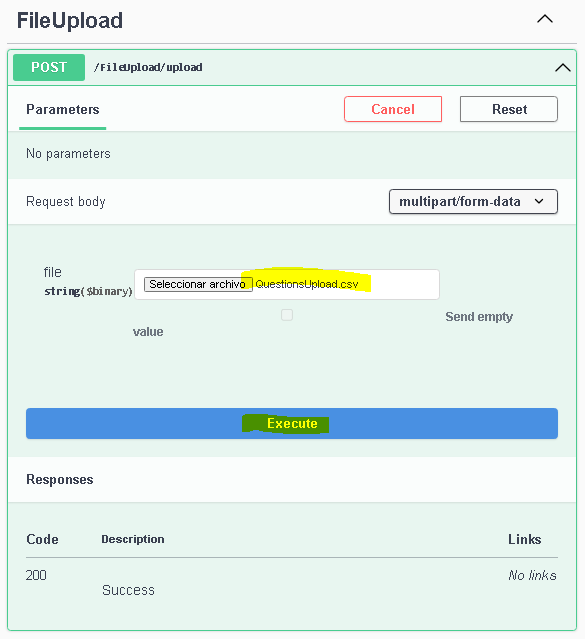
It is used to increase the vote counter of the answer, you just have to invoke it with the id of the answer.

**Improvements:** Being able to set votes to zero or any other number, prevent the maximum value of votes from being overflowed. Validate that the user can only vote 1 time.

URL(PUT): <https://localhost:7186/api/Questionnarie/voteanswer?id=1>

Importing from a file

This option allows you to load a large number of questions and answers at once. A csv file is used, which are comma-separated text records. Only 2 values are needed, the first value is the record type, **Q** for questions and **A** for answers, the second value is the question.

When a question is registered, all subsequent answers will be associated with this question until another question is found, to which the answers that follow will be associated. If the file has answers at the start, these answers are considered not to have a question associated with them, so they will be ignored. There may be questions without answers, but not answers if you ask.

Only CSV files smaller than 10 MB are allowed.

To test it from Swagger you have to select a file that meets the above characteristics and run it.

**Tags**

There is no method in place to add or update tags

**Users**

There is no method in place to add or update users or roles.

**General improvements**

* Security and User Maintenance and Role Management.
* Delete questions and answers only by the user themselves or an administrator.
* When voting for a question or answer, validate that the user can only vote 1 time.
* Add the creation and modification date for the questions and answers.
* Validate that only the user can or an admin user can modify their questions and answers.
* Create new tags.
* When uploading questions from the file, validate if they have already been uploaded to avoid duplicating them.
* Be able to upload questions with tags and votes (optional).